

Project Proposal | Daily 49er Android Application

CECS 343, Spring 2011

Team Name: Beachin' Software

Individual Names: Alex Chavez, Lynn (Punravee) Cherngchaosil, Amanda Nguyen

Proposal Name: Daily 49er Android Application

Project Summary:

The purpose of the application is to create a fast and functional companion application for the CSULB newspaper the Daily 49er on mobile devices. Given the time constraints of this project and the popularity of Android platform, this mobile application will be created for use on those devices. The intent of having a mobile application is to allow for news articles to be easily accessible and readable in a user-friendly way. In addition to news articles, subcategories will also be available that consist of: videos, podcasts, local events, social networking features, contact information and advertisements (for revenue).

News articles will be the main and most sought-out feature for the users who download this application. Our intent is to present these articles to the user in a way that is clear and readable on a mobile device while being conveniently accessible. This will be more convenient to a user who uses an Android device as these devices have proportionally smaller screens and are mostly navigated through touch. We want to make it easy for the user to access content more efficiently and naturally from their respective phone environments as opposed to visiting a clunky website (<http://www.daily49er.com/>) that lacks such functionality and user-friendliness. Similarly, the user will also be able to view alternative media in the form of videos, podcasts and accompanying photo galleries related to current news and issues on campus. Such multimedia content can be optimized and easier to view on Android phones.

In addition to obtaining news for personal viewing, the user of this application will be able to share stories and media with others through social networks such as Facebook and Twitter. Because many students are no longer interested in the traditionally printed newspaper, this application will entice CSULB students to not only follow, but also share, the school news through the social networking share features. To further expand the functionality of this application, we will also integrate Google Maps to retrieve directions to social, academic, and athletic events that can be found on the calendar on the Daily49er website.

Checklist of Deliverables | Daily 49er Android Application

1) Communication:

- a. Marketing Request (Summary written by your customer)
- b. Project Scope Document (PSD_Template)

Due: 2/16/2011

2) Planning:

- a. Risk Plan Due: 2/21/2011
 - i. Identify the possible risks associated with this project (5 to 10)
 - ii. For each risk:

- 1. Identify the probability (High, Medium, Low)
- 2. Identify the delay if the risk materializes
- 3. Identify a mitigation plan to reduce the risk.

- b. Schedule Due: 2/23/2011

- i. Lay out the timeline for the project. Include high-level tasks and milestones. You can use any application software you want. Just make it look nice and presentable. You must present the Planning Exit Schedule BEFORE you move to the next phase (Modeling). This will serve as a baseline.

- c. Test Plan Due: 4/14/2011

3) Modeling

- a. Prototype or Story Board Due: 3/2/2011
- b. Detailed Use Cases Due: 3/23/2011
- c. Detailed Test Cases Due: 4/11/2011

4) Construction

- a. Writing the code
- b. Testing the product
- c. Correcting the errors

- 5) Deployment Due: 5/9/2011 & 5/11/2011



Project Scope Document (PSD)

Project Title: Daily49er Android Application

Author: Beachin' Software (Alex Chavez, Punravee (Lynn) Cherngchoasil, Amanda Nguyen)

Revision #: 2.0

Revision Date: May 9, 2011

Revision History			
Rev	Description	Date	Author
1.0	Initial completed draft of the PSD has been approved by Professor Steve Gold. Case diagrams will be added at a later revision.	2/16/2011	Alex, Punravee, Amanda
2.0	Updated minor info throughout the PSD. Added references to the Apache license under Section 1.3. Added activity diagram.	4/9/2011	Alex

Table of Contents

1 Project Objectives	3
1.1. Key Areas.....	3
1.2. Performance.....	3
1.3. Licensing	3
1.4. Security	3
1.5. Documentation	3
1.6. Infrastructure	4
1.7. Hardware.....	4
1.8. Licensed Software.....	4
2 MRQ to Use Case Mapping	4
3 Controlling Artifacts	5
3.1. Artifact Locations.....	5
3.1.1. Use Cases	5-6
3.1.2. Activity Diagrams.....	6
4 Operating Systems and Hardware Supported	7
5 Release Criteria.....	7
6 Release Mechanism.....	7
7 Special Considerations	7
8 Patent Applications	7

1 Project Objectives

The purpose of the Daily 49er Android Application is to create a fast and functional companion application for the California State University Long Beach's newspaper (The Daily 49er) on Android mobile devices. The intent of having this application is to allow for news articles to be easily accessible and user-friendly. In addition to news articles, there will also be features allowing users to access videos, podcasts, local events, and contact information related to the campus. Users will also be able to share this media with friends through social networking features such as Facebook and Twitter. The overall purpose of this application is to allow for efficient and convenient access to such information through Android devices.

1.1. Key Areas

The Daily 49er Application will have the expected performance of any typical Android application. There will be no special licensing requirements, nor will there be any login information required to gain full access of all features. No user guides will be available, as the simplicity of the application will allow for easy navigation. Minimum requirements consist of a smart phone running Android OS v.2.1 (nicknamed Froyo) or higher, with Internet access.

1.2. Performance

Performance will be dependent on the user's type of Internet connection of their Android devices. Users will generally have a more stable and faster Internet connection through wireless Internet in comparison to a data connection through a cellular phone service provider. A user who is using data connections such as 3G or EDGE may expect to wait up to three times longer than that of one who is accessing the application through Wi-Fi.

1.3. Licensing

There will be no limitations regarding licensing for the end user. Anybody who downloads this application from the Android Market will have full access to its features and will have to abide to the terms and conditions set by Google for the Android Market. Furthermore, anyone may get our project code that is hosted on GitHub and may modify and redistribute the application as pleased as long as it's in accordance to the Apache license (<http://www.apache.org/licenses/LICENSE-2.0>) since The Daily49er Android App is under the Apache license.

1.4. Security

Anyone who downloads this application from the Android Market can run the program on their devices. No username and password are required for use of this application, although Facebook and Twitter social networking features will require username and password, but is handled through their respective API's.

1.5. Documentation

No documentation will be provided, as this mobile application will be simple and easy to navigate.

1.6. Infrastructure

Android Operating System version 2.1 and above is required.

1.7. Hardware

Application requires a touch-screen smart phone with Android Operating System, have minimum 320x240 pixel screen resolution, data plan provided by cell phone carrier and/or wireless Internet access.

1.8. Licensed Software

- Eclipse
- Android Development Tools (ADT) plugin for Eclipse
- Google API and software services
- Android SDK
- Facebook Connect API
- Twitter API
- Java Development Kit (JDK)
- OmniGraffle
- Dropbox
- Git/Gitub

2 MRQ to Use Case Mapping

Requirement	Supporting Use Case
Downloading the application	Searching and download the application from the Android Market.
Access articles	Selecting an article from a list of news categories.
Access videos	View videos that have been uploaded to the Daily49er YouTube channel.
Access podcasts	Listen to podcasts that have been uploaded to the Daily49er YouTube channel.
Social networking integration	Share links to articles through Facebook and Twitter.
Event calendar set of tabs that include "News", "Media", "Events" and "Settings".	List any events that have been entered by the Daily49er through Google Calendar and map address locations through Google Maps.
Contact information	Submit any tips or comments to the Daily 49er team via a feedback form.
Advertisements	Display advertisements in a non-obtrusive manner so hits and revenue can be made.

3 Controlling Artifacts

3.1. Artifact Locations

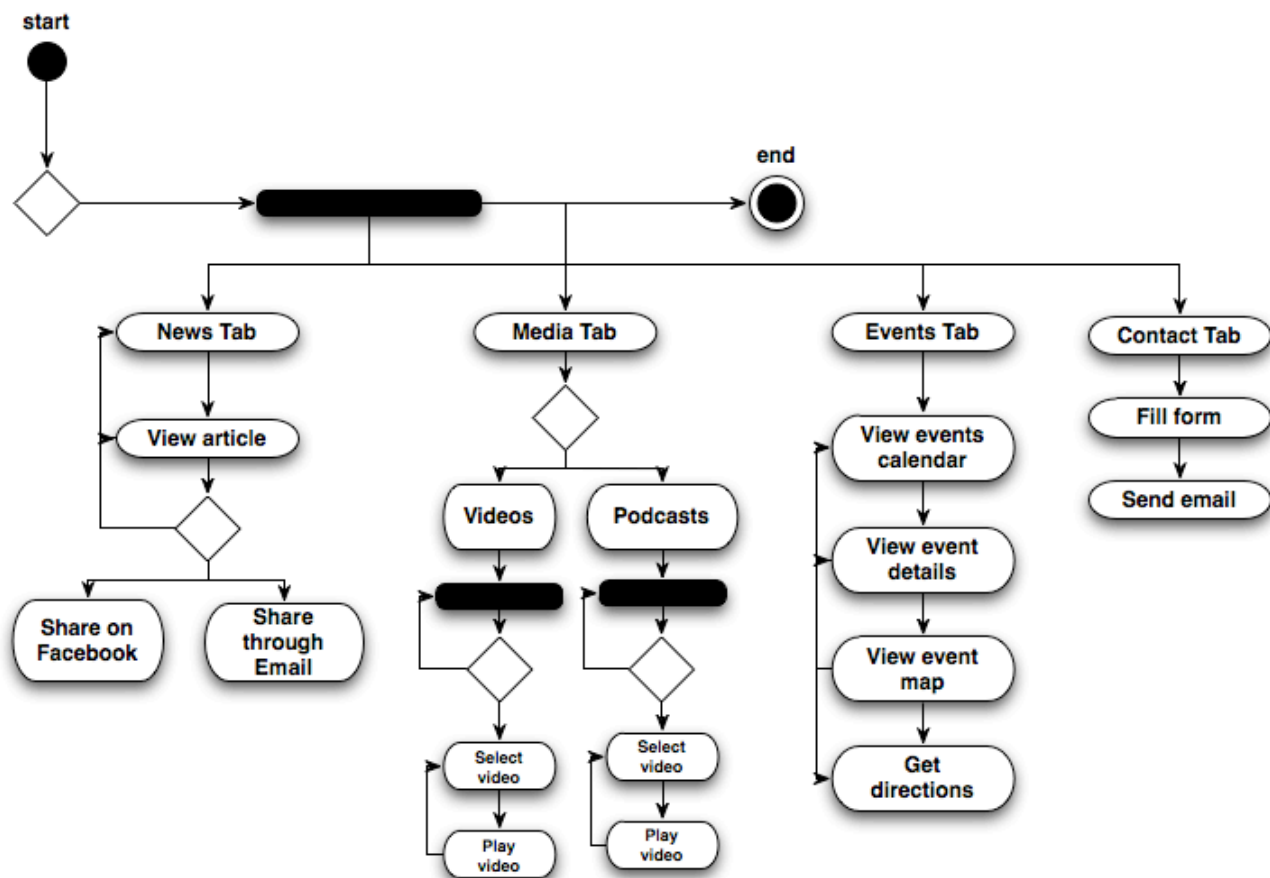
Artifact	Location
Project proposal, project schedule	Google Documents & Dropbox
Risk plan, use cases, diagrams, screenshot and miscellaneous documents	Dropbox
Source code	Github and Dropbox

3.1.1. Use Cases

UC#	UC Title	Objective
01	Downloading the application	Download the application from the Android Marketplace.
02	Opening the application	Touch the application to launch the application and present the user with different categories of news
03	Navigating categories	Navigate the set of category tabs include “News”, “Media”, “Events” and “Settings”.
04	Navigating subcategories	Navigate a subcategory of the parent category to view articles of the designated category
05	Viewing an article	Touch the name of the article from the list and scroll down to view the rest of the articles as well as have options to change viewing options such as font size
06	Listening to podcasts or watching video	Plays a selected video or podcast from the list selection through Android’s integrated media player
07	Event calendar	Displays events through a mobile version of Google Calendar inside a web frame
08	Event directions	User selects the event address (which will be displayed as a URL) and displays an embedded MapView of Google Maps
09	Share features	Through the press of the “Share” button, the user can share the currently displayed article through e-mail, Facebook or Twitter.
10	Advertisements	The user has the option of clicking an embedded advertisement within an article

UC#	UC Title	Objective
11	Close application	Terminate the application when the user presses the phones home button or closes it through Android's task manager

3.1.2. Activity Diagrams



4 Operating Systems and Hardware Supported

Touch-screen smartphone with Android OS v2.1 or higher.

5 Release Criteria

The application will be deemed fit for release once volunteer field testers (any CSULB student with an Android phone and access to a data network) have used the application and have been able to access content without any major errors in the retrieval and presentation of Daily49er content. The application will be released if there are no major software bugs (i.e. application crashes, memory leaks, excessive loading times and misc. performance issues, installation issues, application termination issues, access of articles, videos and podcasts) that might be reported by field testers that obstruct such functionality that is expected of a stable application.

6 Release Mechanism

Potential users of this application can download the Daily 49er Application through the Android Market.

7 Special Considerations

Must meet any specifications requested by customer (The Daily 49er).

8 Patent Applications

N/A

Feb 21:

- Get all the basic user interfaces done
- Finish risk plan
- Finish Schedule
- Get more info on RSS, UI, Google map and calendar integration.
- Print out the new PSD.

Feb 28:

- Get prototype and storyboard done
- Figure out how to parse the RSS feed by now.
- Do some Google map and calendar integration and actually have it working on the application by now.
- Youtube video.

March 7

- Make an appointment with the online editor and show the product. Get feedback and see what else do they want to change, ex color, layout, functionality, etc. (ask them when do we meet them???)
- Have a working application with all the basic feature
- Create Google account and group e-mail for contact purpose and everybody will have access to it.
- Get volunteer's name who's going to test the application. Get their name, phone number, e-mail, etc...

March 14

- Test Plan due
- Integrate youtube, google maps and media
- Facebook and twitter integration (Research & implementation).

March 21

- Layout revision
- Use cases done (user experience with software).
- Don't forget the UML

March 28

- Test cases done (ways to test failure of software).

April 4

- Project should be almost done.

April 11

- Have all the documentation done.
- Have it done and be fully tested by the developers (will be done through emulator since no one owns an actual device).
- Fix all bugs that we encounter.

- Write the report on the bugs.
- Put together the documentation and print it out.

April 18

- Load the application on the actual device and have them tested.
- Show it to the daily 49ers and have some of their staff load the application on the device and tested
- Get report or feedback from the user.
- Have this week and next week to test and fix all bugs and make sure that everything is fine.
- Print out the documentation.

April 25

- Get feedback and do some final revision
- Ready to put on the app store.

May 2

- Project completed.

Event	Problem	Solution
Finals week (May1st – May13th)	Extra time spent studying for exams, fall behind project schedule	Prepare for finals by setting expected completion to earlier date (April 11 th).
Sickness and/or injury of one or more members	Responsibilities of injured/sick member will not be fulfilled on time	Remaining healthy members will compensate by dividing these responsibilities
Lack of volunteers to test-pilot the application	Must find other ways to determine release readiness	Purchase Android phone for testing and bug detection
Difficulties in contacting customer through e-mail for meeting arrangements	Need frequent customer communication and feedback	Go find customer (Daily 49er) in SSPA Building (basement).
Customer requests change in requirements and/or specifications near due date	Must add, delete, or modify functions and/or features that have already been created.	Follow an iterative development plan to avoid backsets due to requested changes in software, contact customer on regular basis.
Encounter difficulties with fixing bugs	Excessive amount of time and effort spent on attempting to resolve a complex issue.	Get help from more experienced Android programmers (through forums, blogs, websites, etc).
Loss of documentation and/or source code	Unable to further project progress due to loss of files.	Prevent loss of data by using multiple storages including DropBox, GitHub, GoogleDocs.
Technical difficulties with one or more devices used to develop software (e.g. broken or lost computer)	Developer is unable to continue contributing to the project.	Access files from other group members and find a different device to continue work.
Unable to complete program before class due date	Software is not fully functional and not ready for release.	Present properly functioning features of application and continue to work on project until complete.

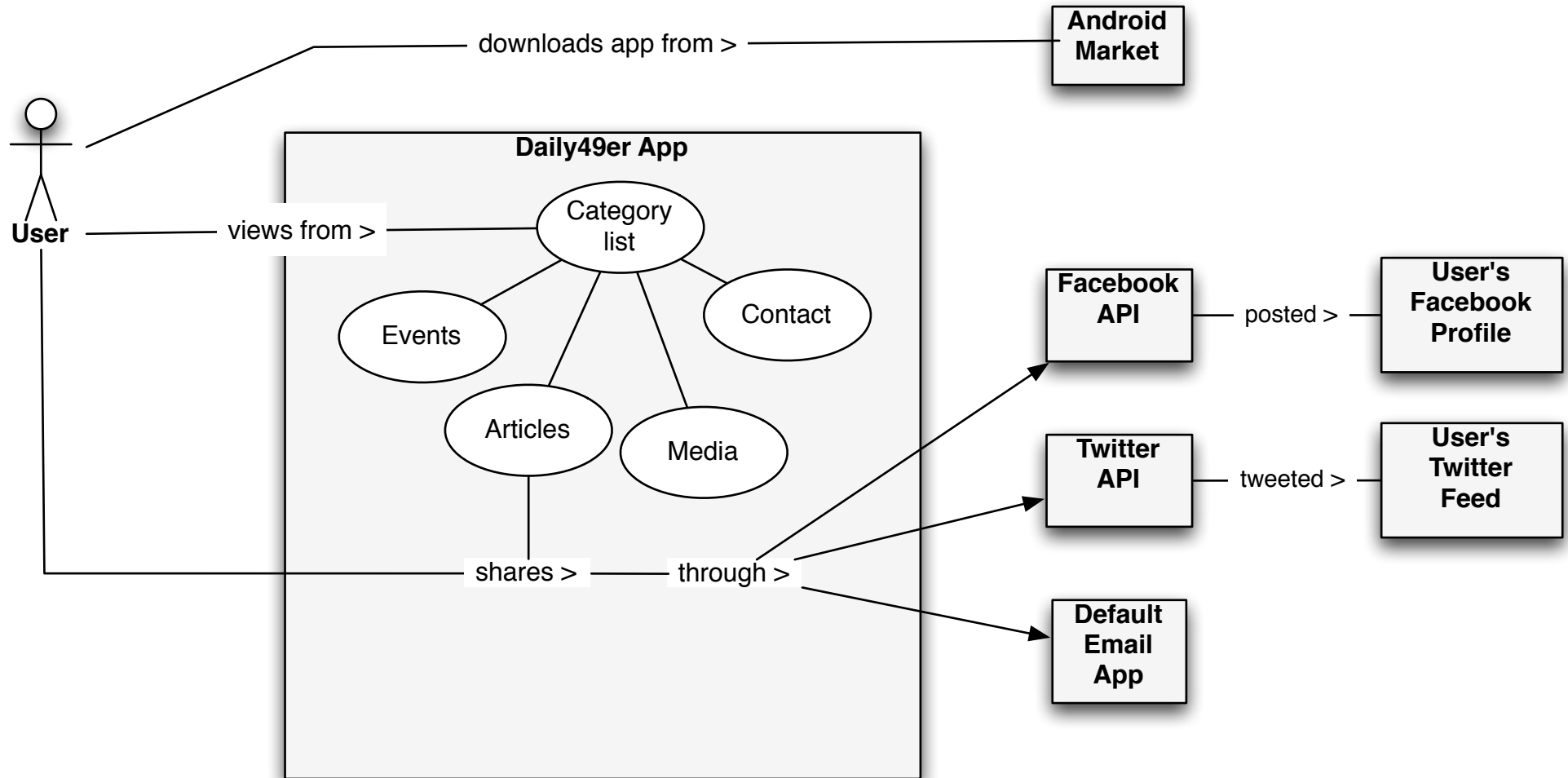
Use Cases (Revision 1) | Daily 49er Android Application

Use Case	Actor(s)	Preconditions	Trigger	Scenario
Download the Daily 49er application to user's Android device	User	Must have Android v.2.1 or higher and an account with Android Market	Would like to view Daily49er news on mobile device	<ol style="list-style-type: none"> 1. User enters Android Market 2. User searches for Daily 49er Application or browses for it under News category 3. User selects to download 4. Program downloads and install on mobile device
Launch the application on the user's device	User	Must have downloaded and installed the application from Android Market	Would like to access news, events, and/or other media from the Daily 49er.	<ol style="list-style-type: none"> 1. Locate Daily 49er application icon on mobile device home 2. Select icon to open
Select desired category from three (news, media, events)	User	Must have application open	Would like to access news, events, and/or other media from the Daily 49er.	<ol style="list-style-type: none"> 1. User decides what type of data they would like to view 2. Select news tab if user would like to view articles, media tab if user would like to view videos/listen to podcasts, events tab if user would like to see campus events calendar
View a specific article	User, mobile device	Must have application open	Would like to access a single complete article	<ol style="list-style-type: none"> 1. User selects news tab 2. User selects a category (News, Sports, Opinion, or Diversions) 3. Mobile device displays a list of articles with brief description 4. User browses list and selects desired article 5. Mobile device displays selected article
Listen to podcast		Must have application open	Would like to listen to a podcast from the Daily 49er	
View a video		Must have application open	Would like to watch a Daily 49er video	
View calendar	User, mobile device	Must have application open	Would like to view event dates, addresses, and directions	
Facebook share	User, mobile device, Facebook API	Must have application open, must be viewing a specific article	Would like to post an article to facebook wall	<ol style="list-style-type: none"> 1. User presses menu button 2. Mobile device displays menu at bottom of the screen 3. User selects Facebook from menu 4. Facebook API prompts user for login information 5. User inputs login information 6. Facebook API requests permission to grant access to Daily 49er application 7. User selects Allow 8. Facebook API displays article in new wall post 9. User selects Publish 10. Facebook application publishes article to user's facebook wall

Use Cases (Revision 1) | Daily 49er Android Application

Twitter share	User, mobile device, Twitter API	Must have application open, must be viewing a specific article	Would like to post an article to twitter feed	<ol style="list-style-type: none"> 1. User presses menu button 2. Mobile device displays menu at bottom of the screen 3. User selects Twitter from menu 4. Twitter API prompts user for login information 5. User inputs login information 6. Twitter API requests permission to grant access to Daily 49er application 7. User selects Allow 8. Twitter API displays article in a new tweet 9. User selects Send 10. Twitter application publishes article to user's twitter feed
E-mail share	User, mobile device, default email application on device	Must have application open, must be viewing a specific article	Would like to email an article to specified email address(es)	<ol style="list-style-type: none"> 1. User presses menu button 2. Mobile device displays menu at bottom of the screen 3. User selects Email from menu 4. Default e-mail application is launched and the title of the articles is passed in to the "Subject" field of the e-mail. A URL is generated and embedded to the body of the e-mail. 5. The default e-mail application takes control at this point.

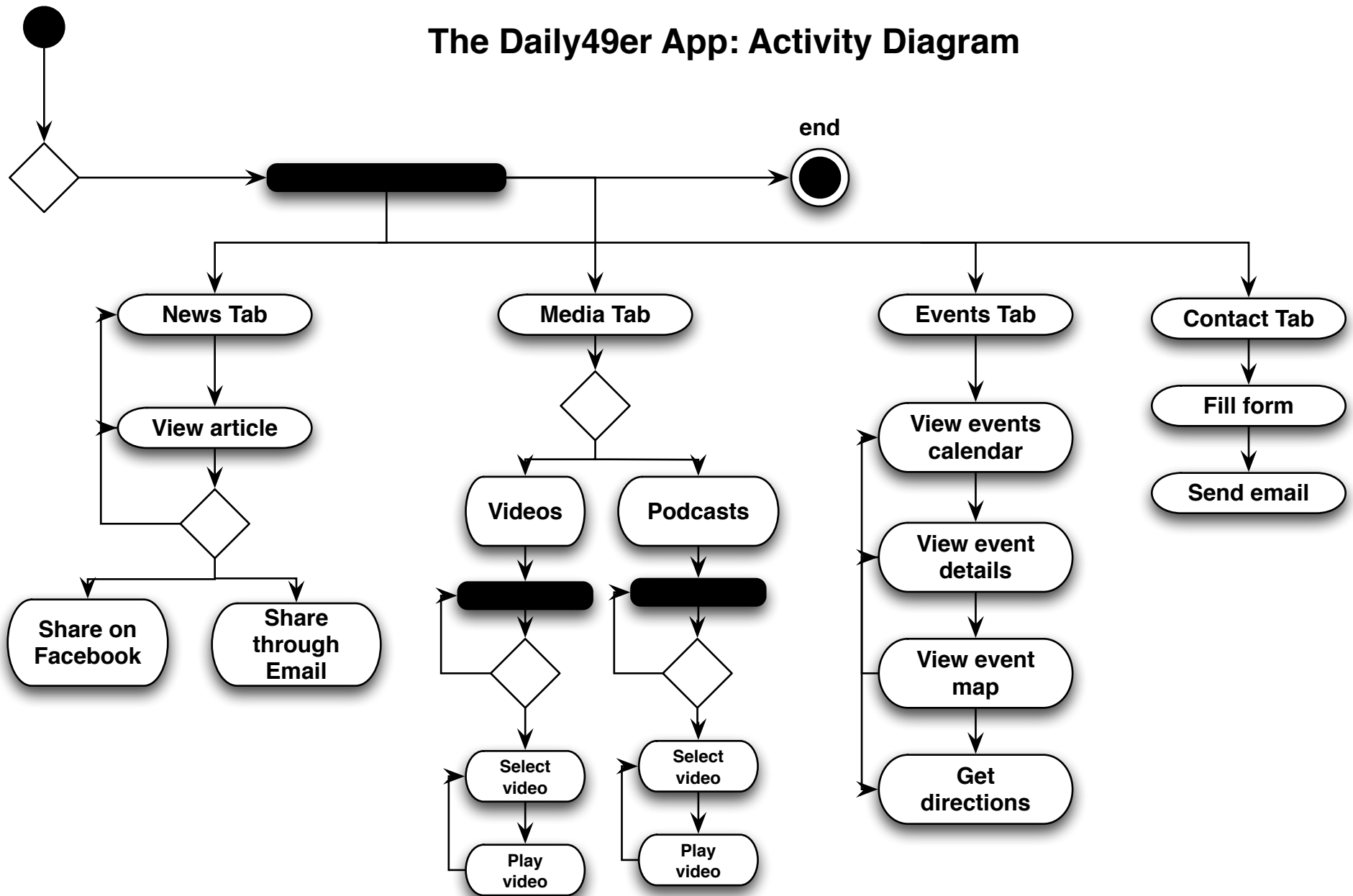
The Daily49er App: UML Use Case Diagram



start

The Daily49er App: Activity Diagram

end



Test Cases | Daily 49er Android Application

Test Case Scenario	Type of Test Case	Pre-conditions	Test Steps	Expected Result	Actual Result	Pass/Fail
Launch the application on the user's device	GUI	Application must be installed	1. Locate Daily 49er application icon on the application launcher 2. Tap the application icon to launch the application	The application will launch and the list of news articles will be loaded and populated in less than 1 minute. The application should be at the "News" tab by default.		
Verify that the most current articles are loaded	Functional	1. The RSS feed must be online and available at the Daily49er's website. 2. Must have an active Wi-Fi or wireless modem connection	1. Launch the application or select the "News" tab.	The application will load the most current 30 articles from the RSS feed.		
Switch between the four different tabs (News, Media, Events, Settings)	GUI	None	1. Tap each tab in this order: Media, Events, Settings and News. 2. Make sure each tab is highlighted and it's respective content is shown.	The selected tab will be highlighted and the content for that tab will load.		
Share an article through Facebook	Functional	1. Must be logged in through Android Facebook Application or login through a login dialogue. 2. Must be currently viewing an article and have pressed the "Facebook" button under the context menu.	1. Select an article. 2. Select a menu button and click on the Facebook button.	The article will be posted on the user's Facebook "wall"		
Share an article through Twitter	Functional	1. Must be logged in through Android Twitter Application or login through a login dialogue. 2. Must be currently viewing an article and have pressed the "Twitter" button under the context menu.	1. Select an article. 2. Select a menu button and click on the Twitter button.	The URL and the article's title will be "tweeted" on the user's Twitter.		

Test Cases | Daily 49er Android Application

Verify that the video and podcast buttons work	GUI	1. Must have the official Android Youtube application installed or have a web browser with flash installed.	1. Select the media tab. 2. Click on the “Video” button. 3. If only official Youtube application is installed, then that application will be launched. Else, if there is more than one application that can view Youtube video installed then the user will presented with an application launcher for the user to select which application to use and also have the option to select that application to use as the default application for future viewing of the Youtube video. 4. Browse the video listing. 5. Press the “back” button on the device to return to the media tab. 6. Repeat step 3,4, and 5 for the “podcast” button.	The application that is selected to handle Youtube video should be launched.		
Verify that the event calendar works	Functional and GUI	1. Must have an active Wi-Fi or wireless modem connection	1. Select the event tab.	The Daily 49er’s diversion Google calendar launches.		

Comments:

Revised Test Cases | Daily49er Android Application

Test Case Scenario	Type of Test Case	Pre-conditions	Test Steps	Expected Result	Actual Result	Pass/Fail
Launch the application on the user's device	GUI	Application must be installed	1. Locate Daily 49er application icon on the application launcher 2. Tap the application icon to launch the application	The application will launch and the list of news articles will be loaded and populated in less than 1 minute. The application should be at the "News" tab by default.		
Verify that the most current articles are loaded	Functional	1. The RSS feed must be online and available at the Daily49er's website. 2. Must have an active Wi-Fi or wireless modem connection	1. Launch the application or select the "News" tab.	The application will load the most current 30 articles from the RSS feed.		
Switch between the four different tabs (News, Media, Events, Contact)	GUI	None	1. Tap each tab in this order: News, Media, Events and Contact. 2. Make sure each tab is highlighted and its respective content is shown.	The selected tab will be highlighted and the content for that tab will load.		
Share an article through Facebook	GUI/Functional	1. Must be logged in through Android Facebook Application or login through a login dialogue. 2. Must be currently viewing an article and have pressed the	1. Press the "Menu" button on the Android device while viewing an article and select the "Facebook" button from the context menu. 2. Login to Facebook, if prompted. 3. Write comment in the comment text field. 4. Click on "Publish" to post	The article will be posted on the user's Facebook "wall"		

Revised Test Cases | Daily49er Android Application

		"Facebook" button under the context menu.	article story's URL to the wall,			
Share an article through E-mail.	Functional	<ol style="list-style-type: none"> 1. Must have an e-mail application installed on the device. 2. Must have at least one e-mail application configured. 	<ol style="list-style-type: none"> 1. Select an article. 2. Select a menu button and click on the E-mail button. 3. If only one e-mail application is installed, then that e-mail application will be launched. Else, if there is more than one e-mail application installed then the user will presented with an application launcher for the user to select which e-mail application to use and also have the option to select that e-mail application to use as the default e-mail application for future emailing of the article. Then, the e-mail's subject line will be filled with the title of the article and the body of the e-mail filled with the article's title, authors, and URL. 	The article will be sent via the user's e-mail to any entered recipients.		
Verify that the video and podcast buttons work	GUI	<ol style="list-style-type: none"> 1. Must have the official Android Youtube application installed or have a web browser with flash installed. 	<ol style="list-style-type: none"> 1. Select the media tab. 2. Click on the "Video" button. 3. If only official Youtube application is installed, then that application will be launched. Else, if there is more than one application that can view Youtube video installed then the user will presented with an application launcher for the user to select which 	The application that is selected to handle Youtube video should be launched.		

Revised Test Cases

Daily49er Android Application

			application to use and also have the option to select that application to use as the default application for future viewing of the Youtube video. 4. Browse the video listing. 5. Press the “back” button on the device to return to the media tab. 6. Repeat step 3,4, and 5 for the “podcast” button.			
Verify that the event calendar works	Functional and GUI	1. Must have an active Wi-Fi or wireless modem connection	1. Select the event tab.	The Daily 49er’s diversion Google calendar launches.		

Comments:

INTRODUCTION

This is the Master Test Plan for the Daily 49er mobile application on Android. This plan will address the functionality of the Daily 49er application for Android devices. The primary focus of this plan is to test the features of the application and ensure that these features meet the project requirements.

GOAL

The project will undergo several categories of testing, regarding ability to retrieve and display RSS news feed, access Google Calendar, launch YouTube (or other appropriate application used to play media), and modify application settings. The details for each category of testing are addressed in the features to be tested section and will be further defined in the level specific plans.

FEATURES TO BE TESTED

The following is a list of the items and features that are to be tested:

- The download and installation of the application from the Android Market.
- Application launch and successful retrieval of news articles from the RSS feed.
- Selecting an article from a scrolling list to read.
- Sharing an article that the user is viewing through Facebook, Twitter and the device's default e-mail application.
- The successful display and viewing of video and podcasts of The Daily49er YouTube channels on the official YouTube App that is commonly pre-loaded on many devices.
- The viewing and navigation of local events through the Daily 49er's diversions calendar hosted and managed by Google Calendar.
- Switching between the four main category tabs: News, Media, Events and Settings.
- Changing the font size of an article for easier accessibility.

*Denotes that the feature will be tested at deployment.

SOFTWARE RISK ISSUES

There are several parts of the project that are not within the control of the Daily49er Mobile App but have direct impacts on the process and must be checked as well. Such risks include:

- Having an unstable, corrupt or unavailable XML file that contains the news articles for the application, which is hosted on the Daily49er's website (<http://www.daily49er.com/>).
- The possible unavailability and downtime of Google's services that include Google Calendar and YouTube where the events, videos and podcasts are hosted.
- "Slow" and/or throttled cellular network speeds (e.g. GPRS, EDGE, 3G/4G, etc.) that can directly affect the performance of retrieving content. A user's perspective of the applications performance and stability is dependent on having a solid network connection with good upload/download bandwidth.
- The look-and-feel of the application on an Android devices, since there are many Android devices that have varying screen sizes and custom implementations of the Android OS GUI.

TEST TOOLS AND TESTING ENVIRONMENT

A. The Eclipse IDE with the Android SDK plug-in will be used for the editing, compiling, debugging and deployment of the app.

B. Android Software Development Kit will be used for development and simulation of the application before targeting and deploying the application to a mobile device.

C. Mobile device running Android OS, v.2.1 (Eclair) or higher will be used by testers to pilot the application and evaluate its features and functionality.

The test team will meet at least once every week (or as necessary) to evaluate progress to date and to identify error trends and problems as early as possible. The test team leader will meet with development and the project manager during this timeframe as well. These meetings will be scheduled on different weeks. Additional meetings can be called as required for emergency situations or new additions to the application.

MEASURES AND METRICS

The following information will be collected by the Development team throughout the testing process. This information will be recorded in the form of incident/defect reports.

1. Date and time of incident/defect.
2. Module from which incident/defect was caused.
3. Severity of incident/defect (Low, Moderate, High).
4. Number of times incident/defect has been reported.
5. Status of incident/defect (Unaddressed, Addressed).
6. Type of incident/defect (RSS feed, User Interface, Settings, Other).

ITEM PASS/FAIL CRITERIA

The test process will be initiated through the Android simulator. This process will be followed by the download and use of the Daily 49er Application on several Android mobile devices. Each tester will be provided with test cases and must rate each case pass or fail after one week of continuous use of the application. Testers are expected to provide feedback and comments on the application. Any functions that have failed the test will be debugged and modified accordingly. The test process will be completed once all features listed in the test cases documents have received a "pass" from each of the testers. The Daily 49er Application will then be released for public use.

SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

A. No users are ready for testing at pilot initiation.

The pilot project will be delayed until at least five volunteer users are ready to initiate the pilot process.

B. Developmental delays

In the event of delay in the delivery or availability of completed functioning software, the pilot testing process will be delayed until software is ready.

TEST DELIVERABLES

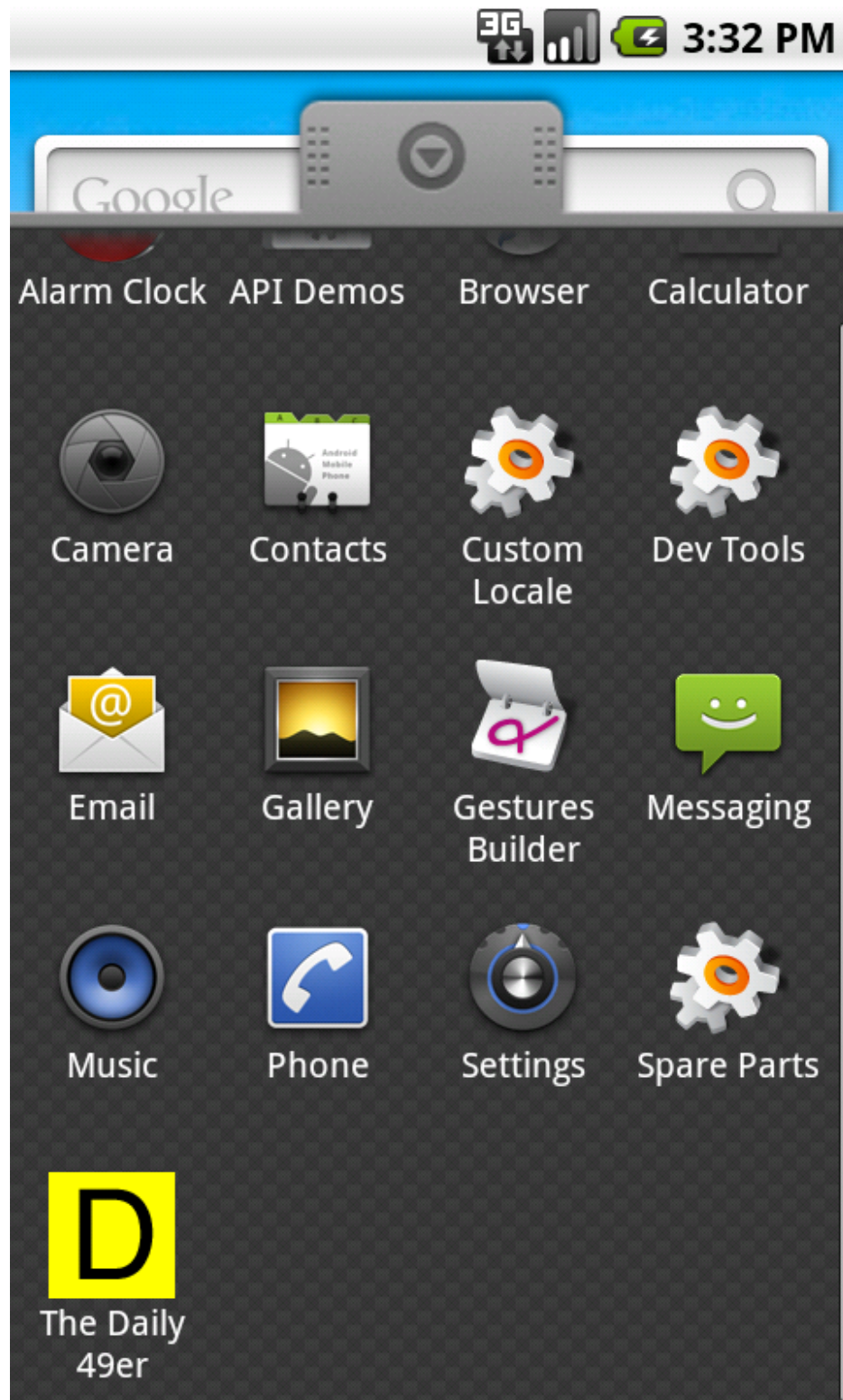
- Official test case document
- Pass/fail and feedback documentation submitted by testers
- Defect/incident reports and summaries
- Test logs

ENVIRONMENTAL NEEDS

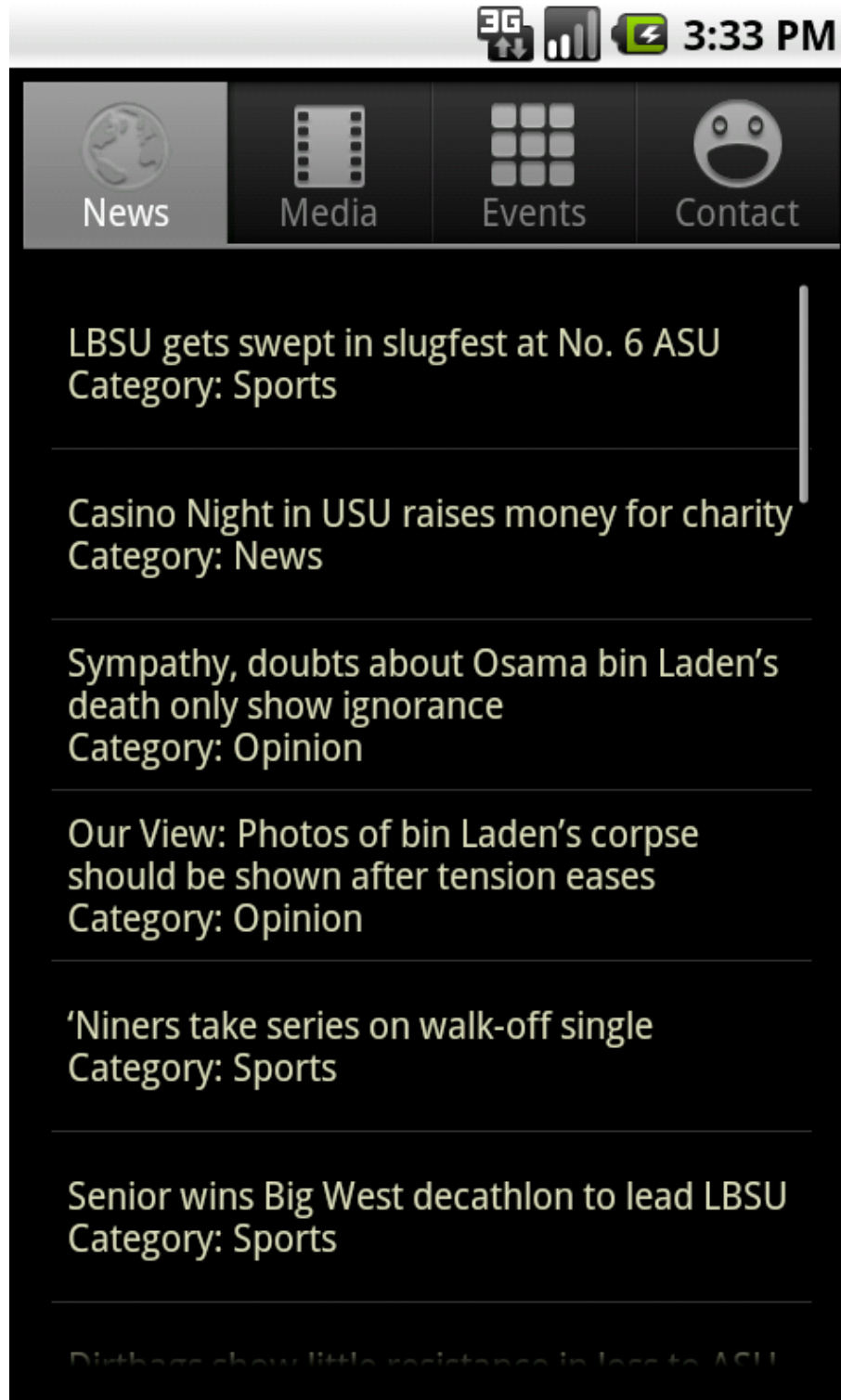
The following elements are required of the mobile device to support the overall testing effort:

- A. Android platform, version 2.1 or higher.
- B. Must be able to access to the Android Market.
- C. Must have installed an e-mail application, an application that can play YouTube videos, such as the YouTube application, a web browser with Adobe Flash 10.1 or higher, or any other application that is able to handle and play Youtube videos on the mobile device.
- D. Must be able to access the internet through a cellular network service or through Wi-Fi.

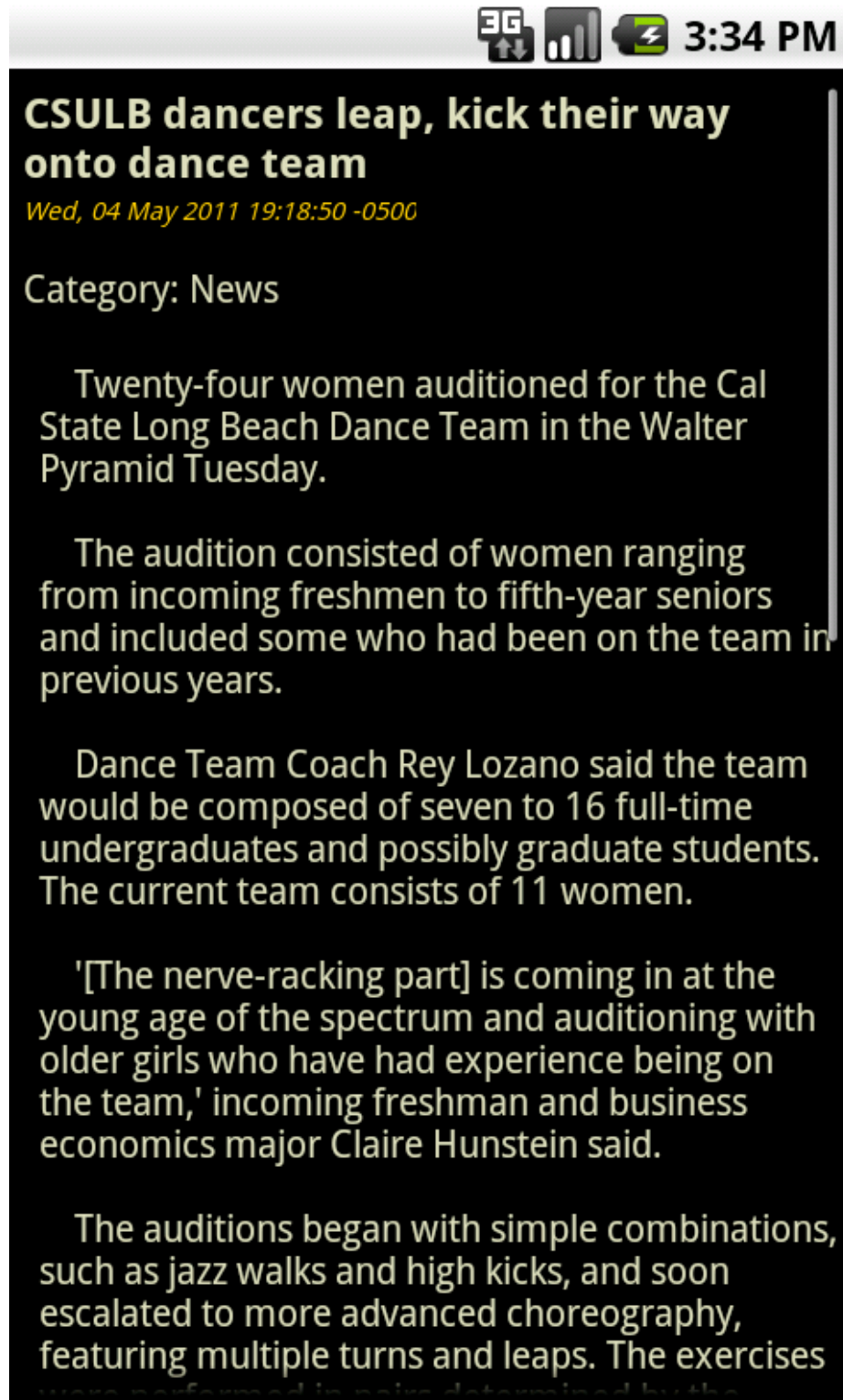
Application in the app tray of an Android v2.1 operating system




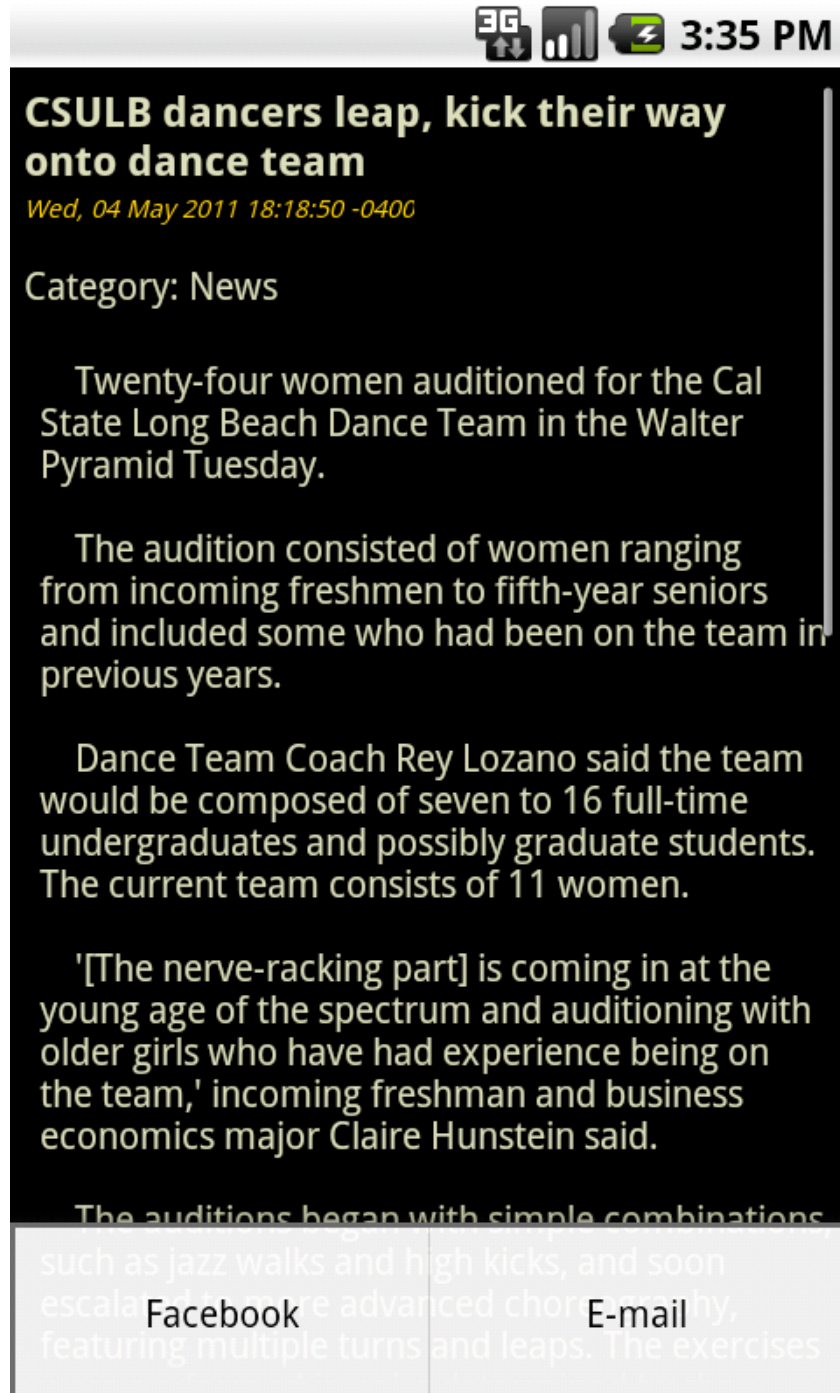
The News Tab is selected by default when the application launches. A list of articles is scrollable and any article listed is available for a user to select and view.



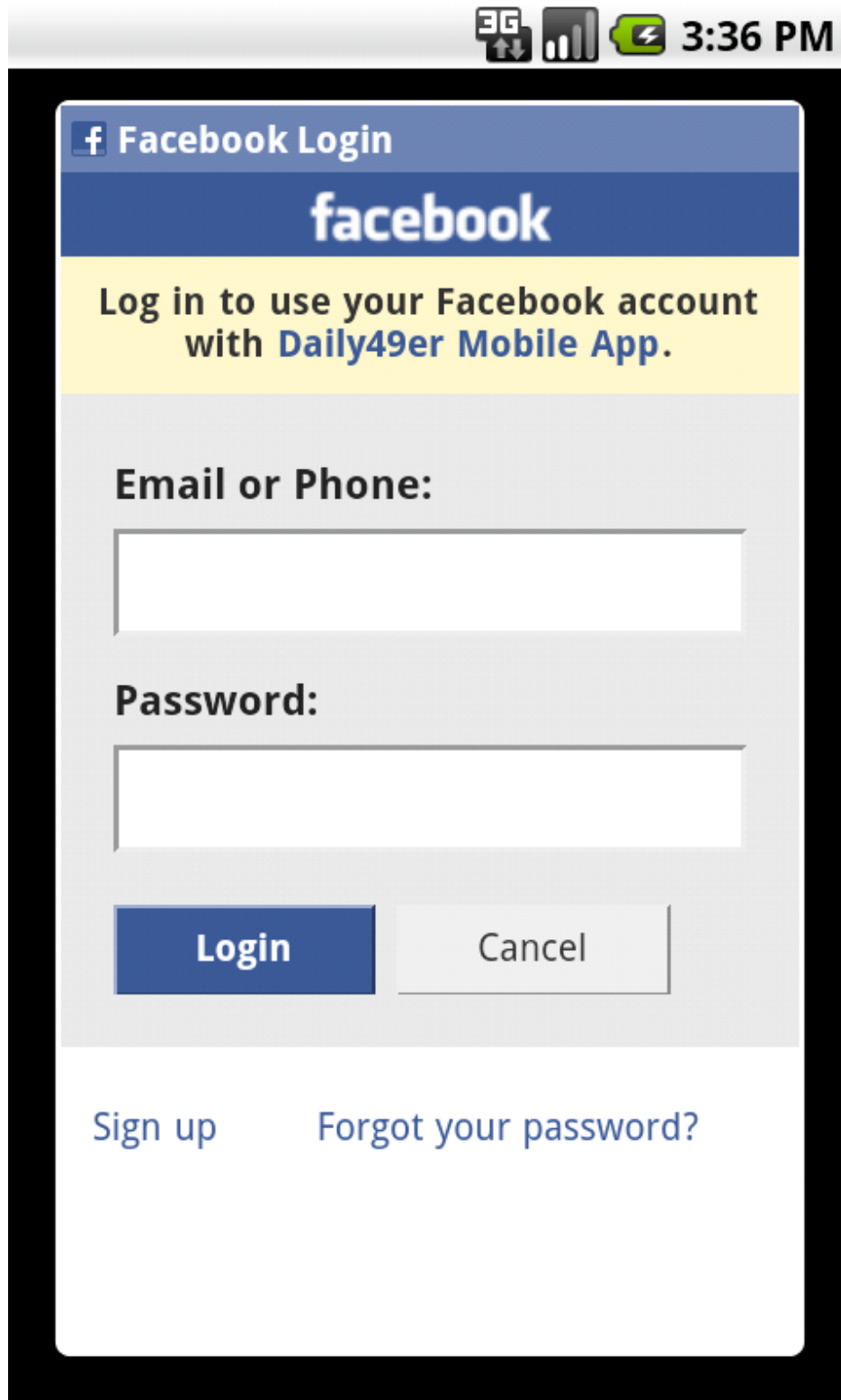
An article's contents are displayed when the user selects one from the list.



An article can be shared with others by pressing the Android device's hardware "menu" button . Pressing the menu button will bring up a context menu allowing a user the ability to publish an article to the user's Facebook wall or send the article through the device's default e-mail application.



When the user first shares an article through Facebook, he/she will be prompted to login.



3G [signal strength] [battery icon] 3:36 PM

f Facebook Login

facebook

Log in to use your Facebook account
with **Daily49er Mobile App.**

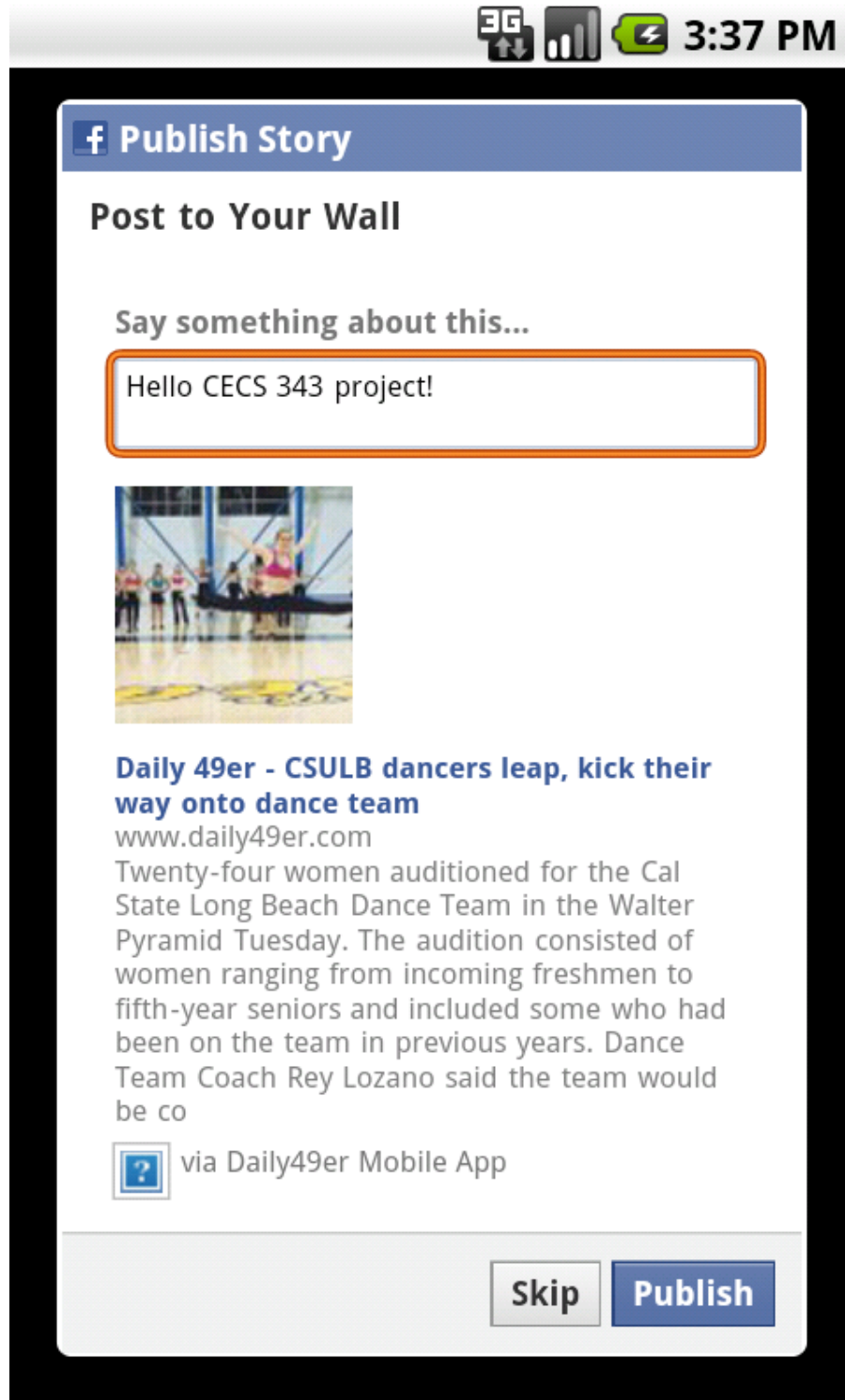
Email or Phone:

Password:

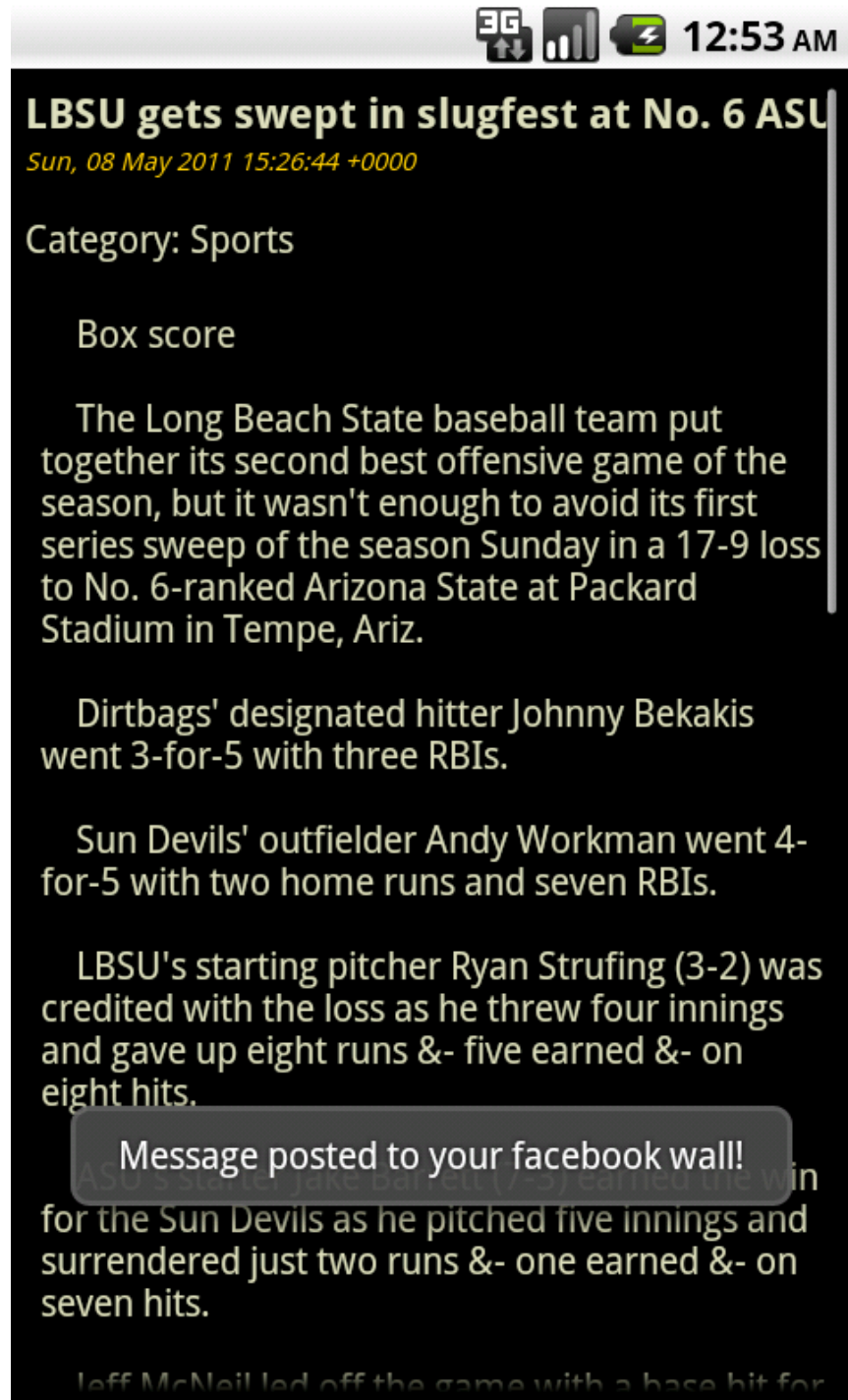
Login Cancel

[Sign up](#) [Forgot your password?](#)

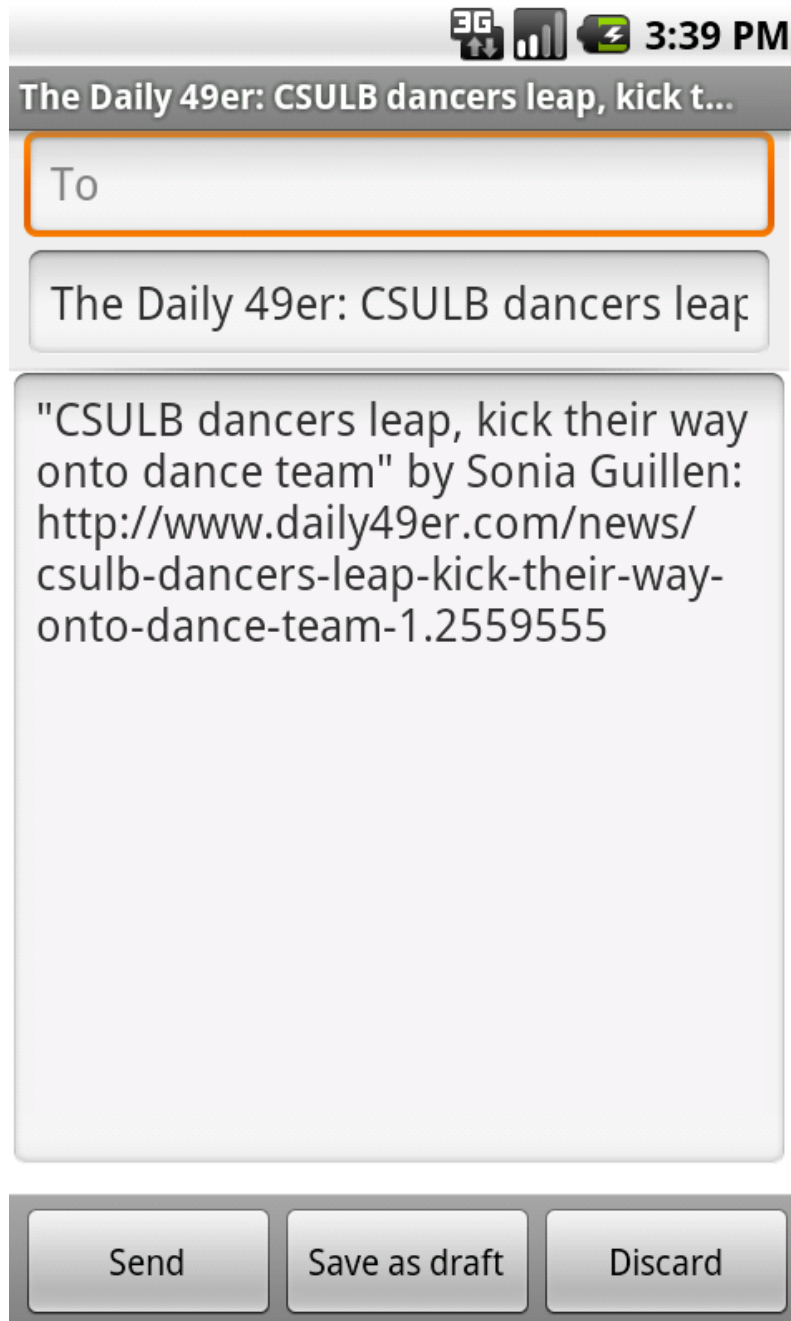
After successful login the user will have a preview of the article's title, description and picture containing a link to the news article in the Daily49ers website. The user has the option to enter a comment into the text field.



Upon successful completion of posting a message, the toast message will appear informing the user that the post has successfully been published. Similarly, a toast message will appear with an error message if the post does not get published or some other exception is caught.



A user may also share an article that is hosted on the Daily49er's website to others via email. When the "E-Mail" button is selected from the context menu, the default e-mail application will launch with the subject pre-filled with "The Daily49er:" followed by the title of the article. The body will also be pre-filled with the article's title encapsulated in quotation marks, the author's name and a URL to the article.



3G 3:39 PM

The Daily 49er: CSULB dancers leap, kick t...

To

The Daily 49er: CSULB dancers leap

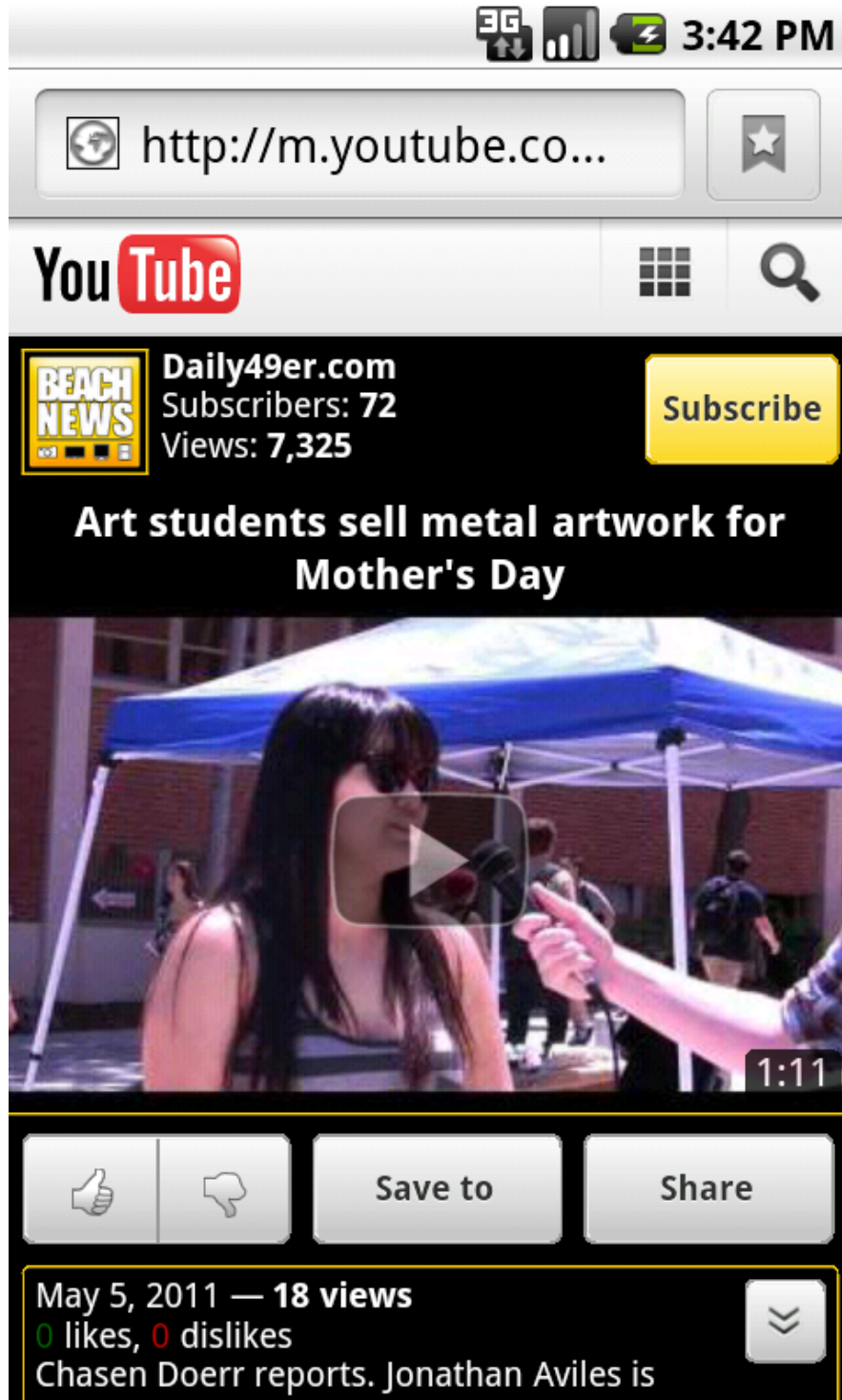
"CSULB dancers leap, kick their way onto dance team" by Sonia Guillen:
<http://www.daily49er.com/news/csulb-dancers-leap-kick-their-way-onto-dance-team-1.2559555>

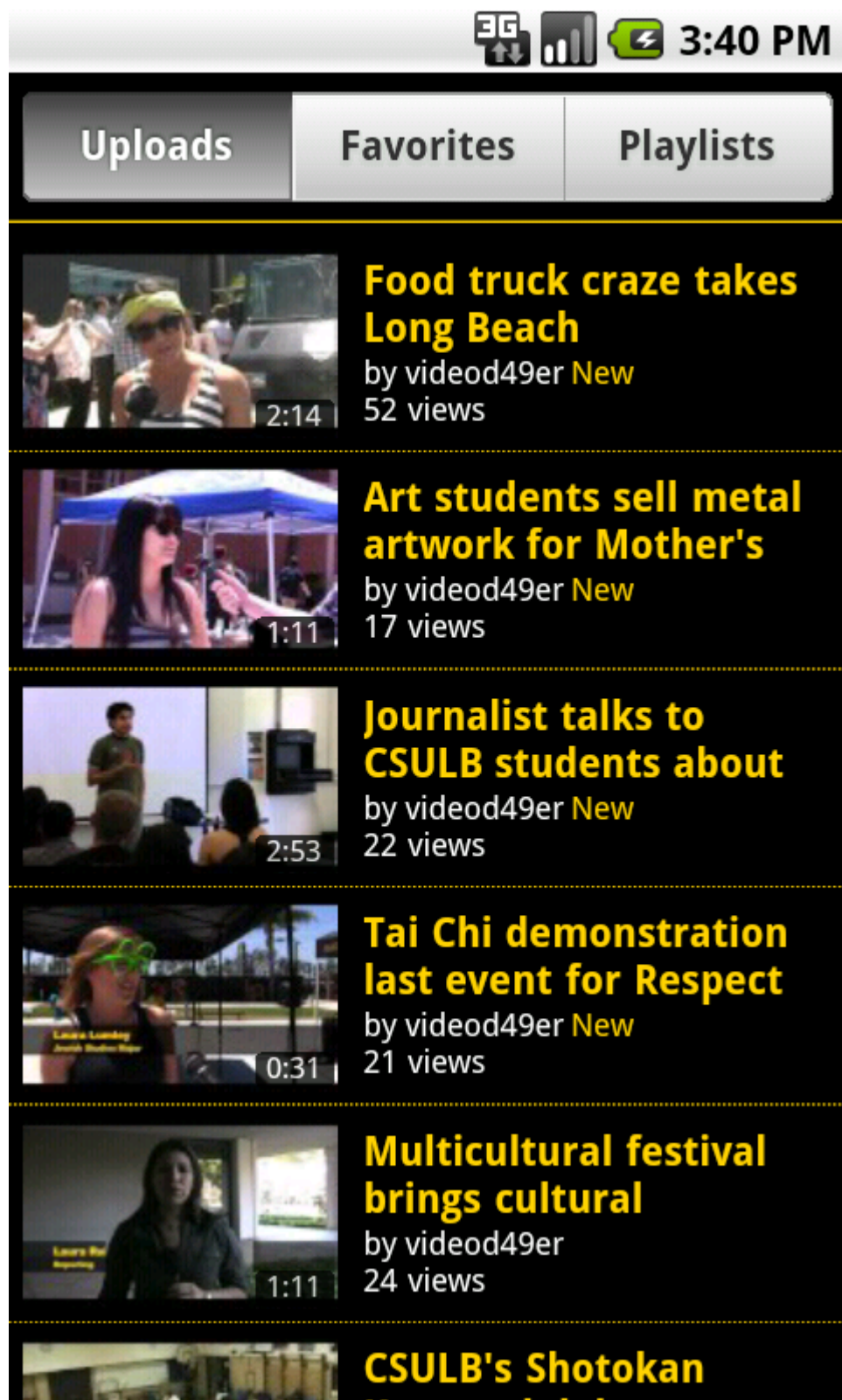
Send Save as draft Discard

When the Media Tab is selected, two buttons appear with a background. The user can select to watch the Daily49er's videos through YouTube or listen to their podcasts through YouTube as well.






When the Video button is selected, the last uploaded video is available to play and is followed by a list of other videos below it. The user may scroll through this page.







A similar layout is implemented for the Podcasts button.

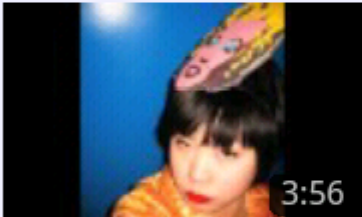


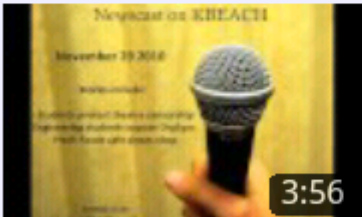
 3:45 PM

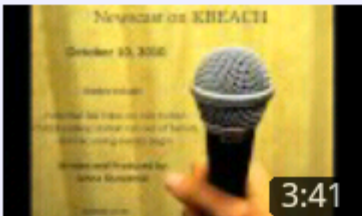
Uploads **Favorites** **Playlists**


**D49er Podcast: CSULB President explains \$94**
by audiod49er
66 views

**D49er Audio Podcast: Cornel West speaks at**
by audiod49er
189 views

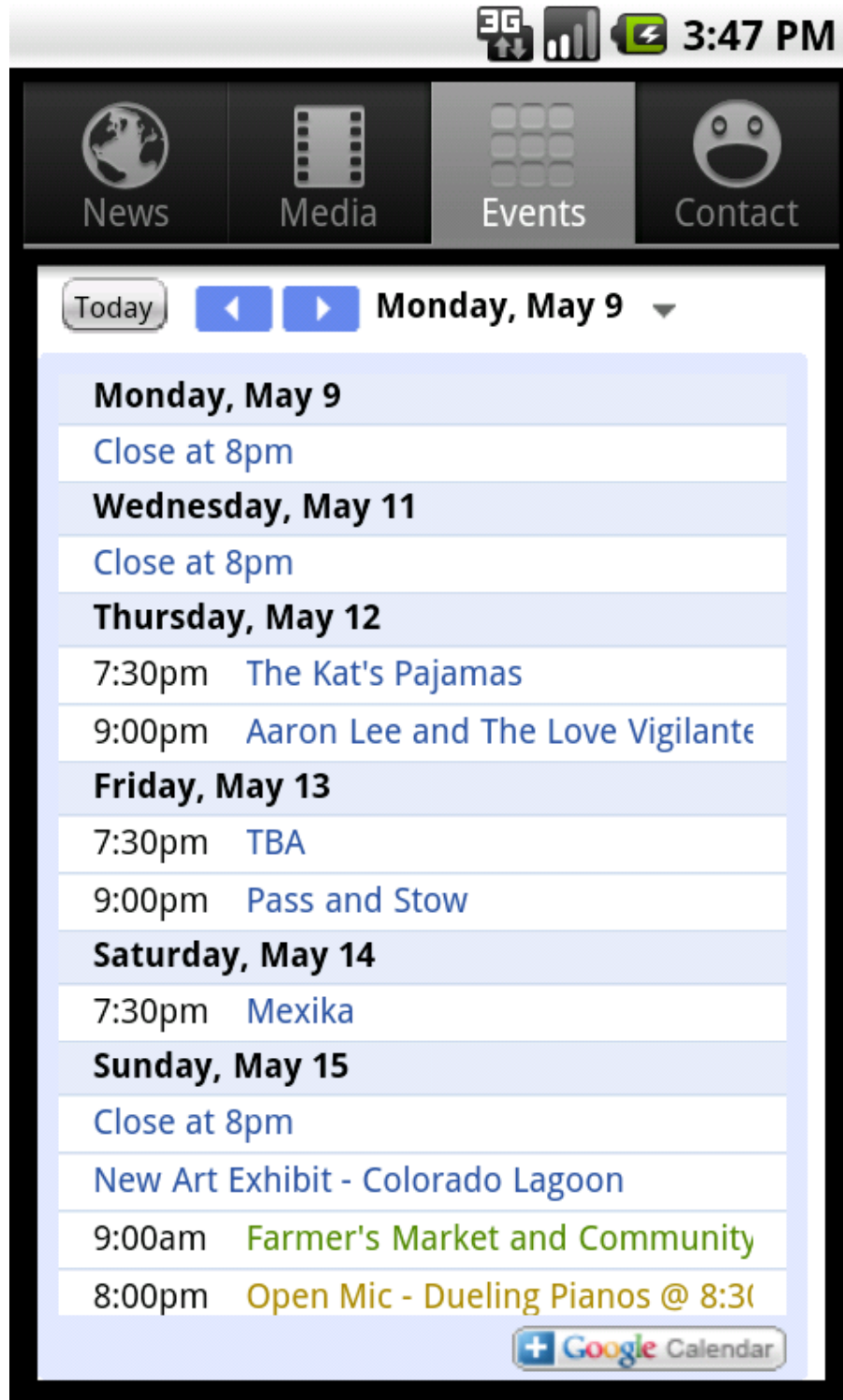
**D49er Podcast: CSUs face more cuts**
by audiod49er
27 views

**D49er Podcast: Students protest**
by audiod49er
130 views

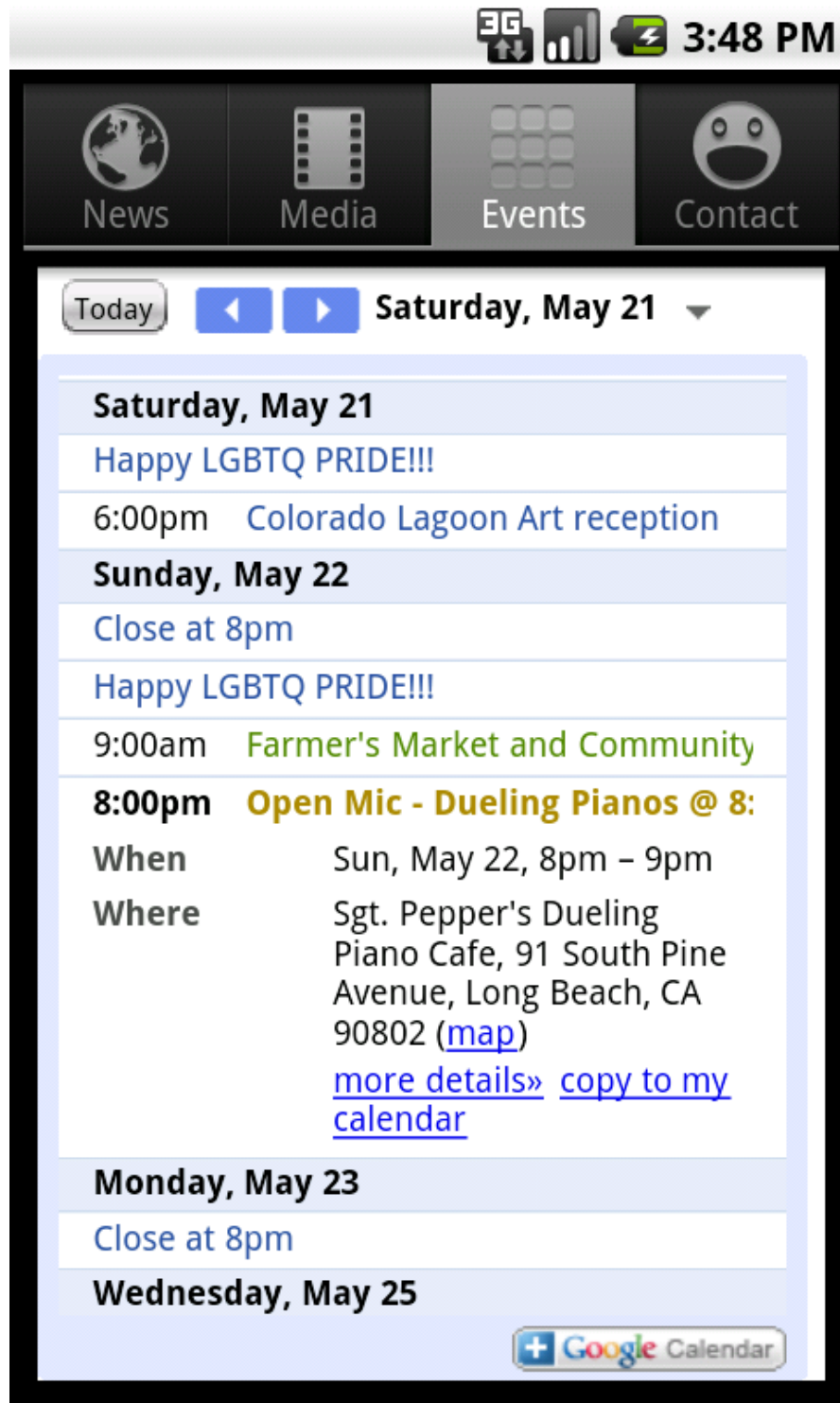
**D49er Podcast: CSUs set to approve fee**
by audiod49er
37 views

**D49er Podcast: CSULB**

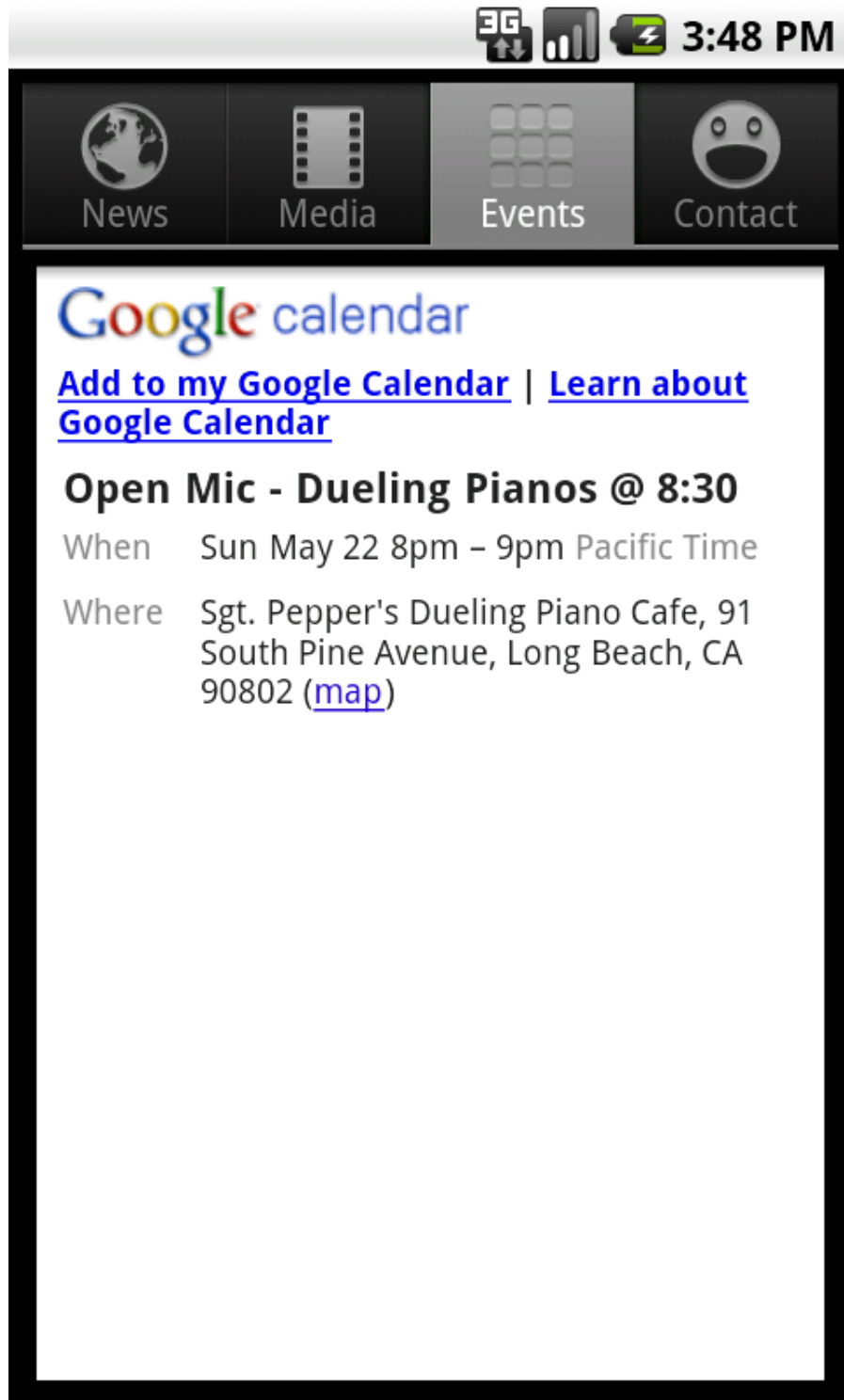
In the Events Tab, the Daily49er's diversions calendar is displayed in via a list view through an embedded WebView. The WebView contains HTML code to a shared Google Calendar under the Daily49er's Google account.



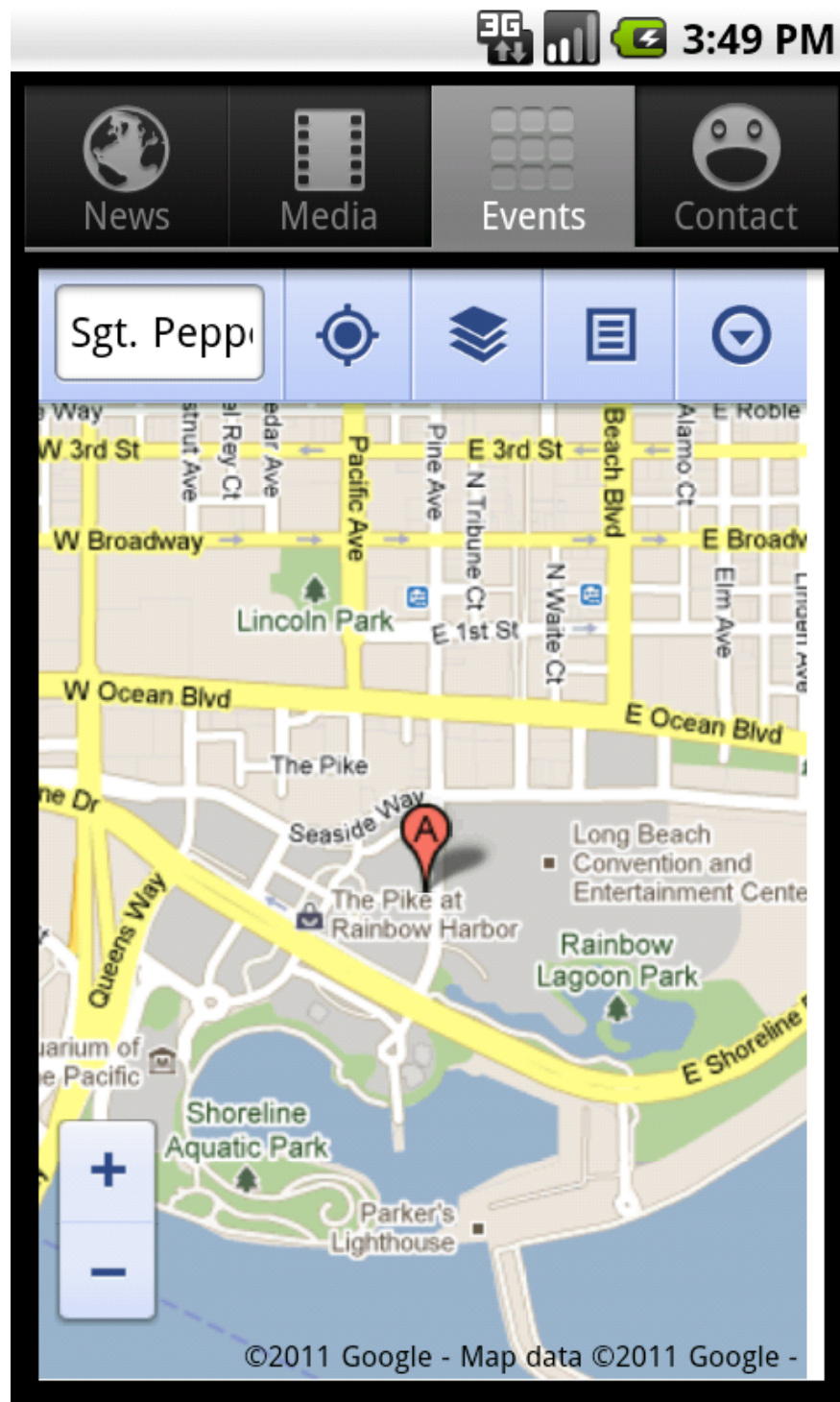
A user may scroll through the events through the left and right blue arrows. A user may select on an event to get a quick overview.



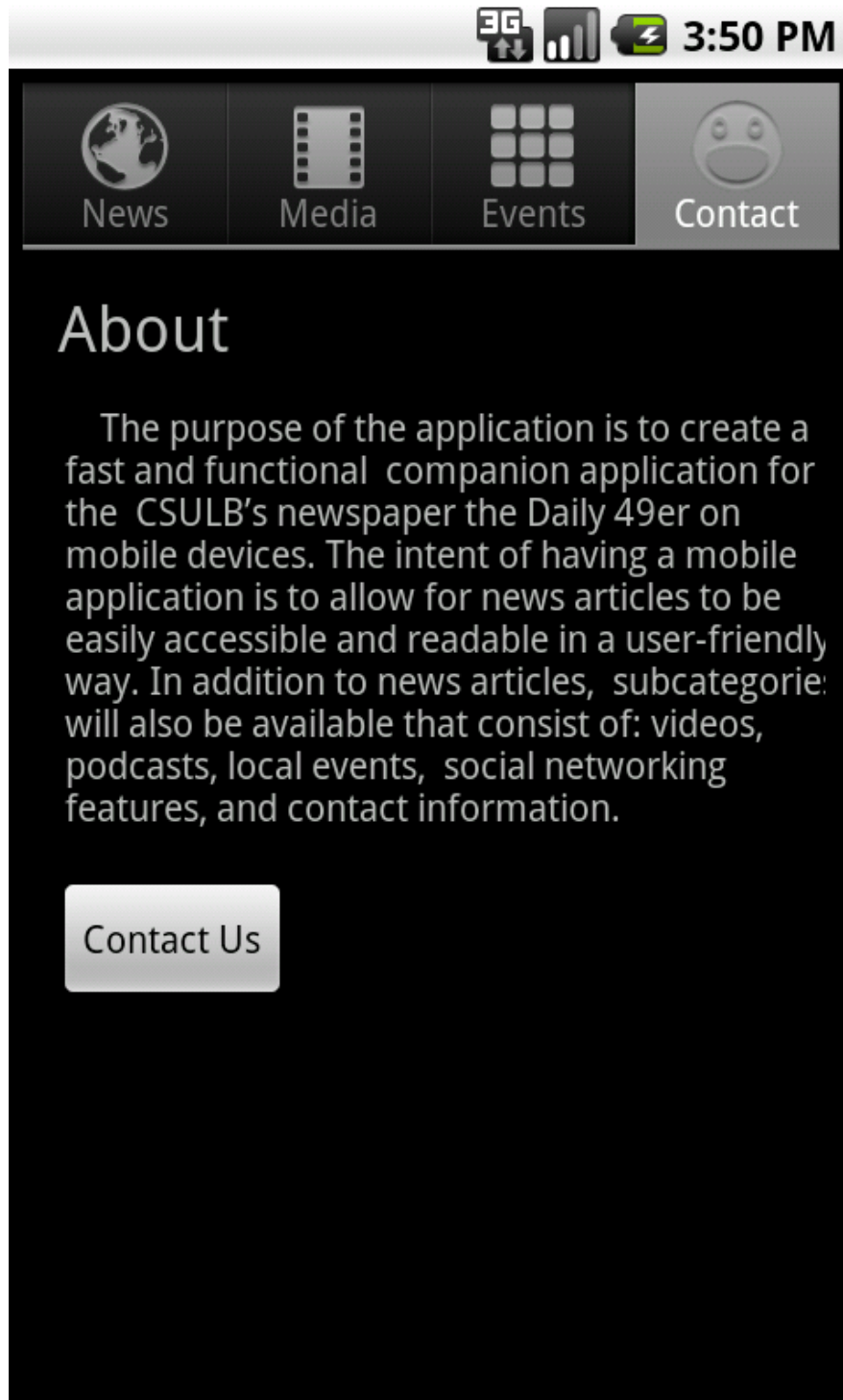
“More details>>” may be pressed on to bring up more details about the event.



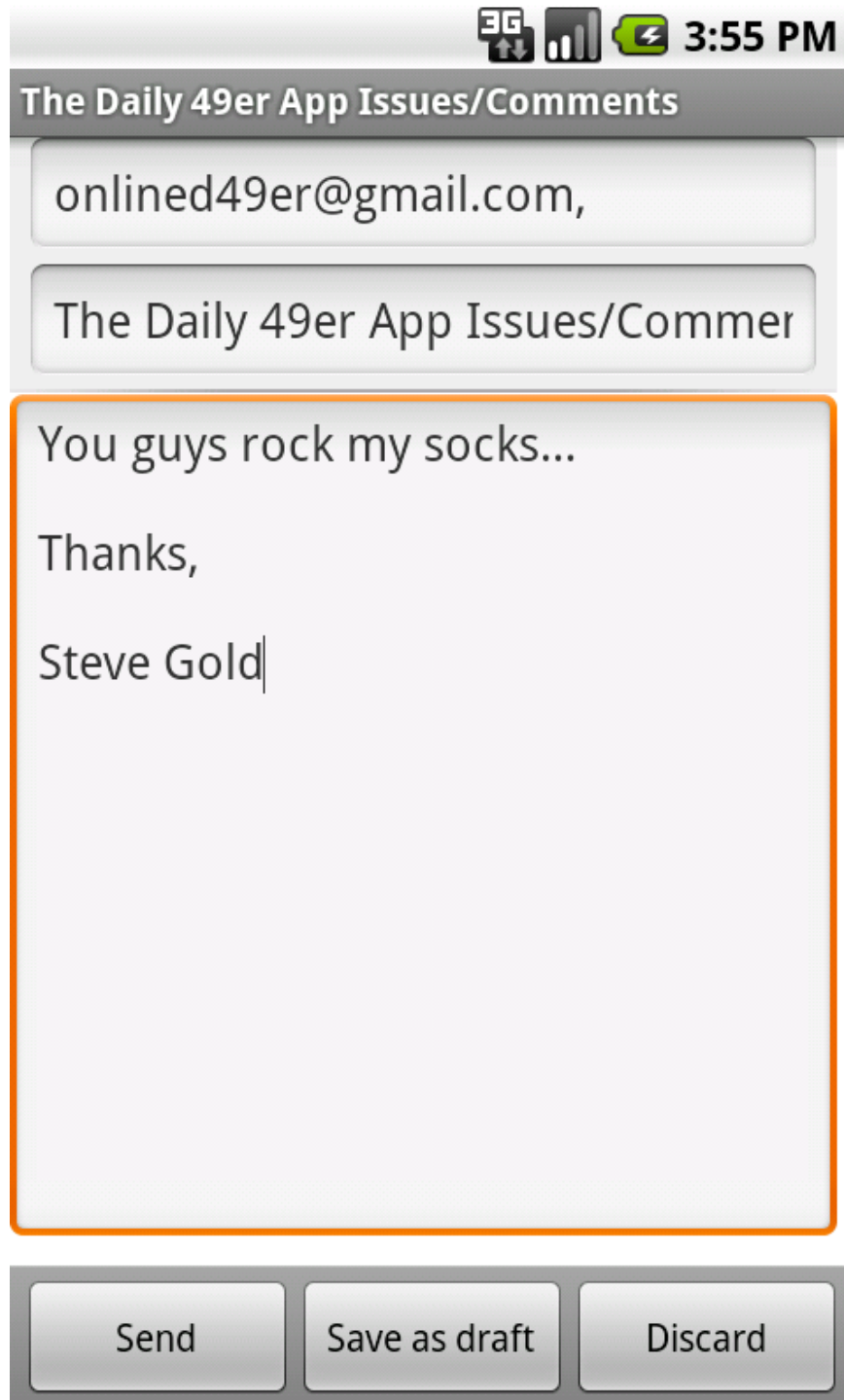
“Map” may be pressed to take the user to a mobile version of Google Maps with the event location thumbtacked. From this thumbtack, the user may explore the map and may get directions.



When the Contact Tab is selected, the user may read a brief overview of the app. The “About Us” text currently serves as a placeholder. We plan to later expand this area to include help instructions and customizable settings. The user may send an e-mail to the Daily49er online editor by pressing the “Contact Us” button.



The “Contact Us” button will launch the default e-mail application with the “To:” field pre-filled with the e-mail address of the online editor and the subject field pre-filled with “The Daily49er App Issues/Comments”. The user may then write comments and send the e-mail to the Dially49er online editor staff.



Apache License
Version 2.0, January 2004
<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications

represent, as a whole, an original work of authorship. For the purposes

of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces
of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright
owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control
systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct
or contributory patent infringement, then any patent licenses
granted to You under this License for that Work shall terminate

as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
 - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
 - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
 - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
 - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of

this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

The Daily49er Android App Javadocs

Packages

com.facebook.android	
daily49er.android.app	

Package daily49er.android.app

Interface Summary

FeedParser	
----------------------------	--

Class Summary

AndroidSaxFeedParser	
--------------------------------------	--

Article	
-------------------------	--

BaseFeedParser	
--------------------------------	--

ContactTab	
----------------------------	--

CustomAdapter	
-------------------------------	--

Daily49er	
---------------------------	--

EventsTab	The Events Tab is composed of the Daily49er's "Diversions" calendar which is implemented via an embedded WebView.
---------------------------	---

FacebookShare	Implementation for posting an article to the user's Facebook "wall".
-------------------------------	--

FeedParserFactory	
-----------------------------------	--

LoadPage	The URL for a given WebView is loaded here.
--------------------------	---

MediaTab	The media tab contains two buttons selection, "Video" and "Podcast".
--------------------------	--

Message	
-------------------------	--

NewsTab	
-------------------------	--

R	
-------------------	--

R.array	
-------------------------	--

R.attr	
------------------------	--

R.color	
-------------------------	--

R.drawable	
----------------------------	--

R.id	
----------------------	--

R.layout	
--------------------------	--

R.string	
--------------------------	--

RssHandler	
----------------------------	--

SaxFeedParser	
-------------------------------	--

SettingsTab	The 'Settings' tab for the application.
-----------------------------	---

SplashScreen	This class creates a new thread for running the splash screen on the start of the application.
TwitterShare	

[Overview](#) **[Package](#)** **[Class](#)** **[Use Tree](#)** **[Deprecated](#)** **[Index](#)** **[Help](#)**

[PREV PACKAGE](#)

[NEXT PACKAGE](#)

[FRAMES](#)

[NO FRAMES](#)

[All Classes](#)

daily49er.android.app

Interface FeedParser

All Known Implementing Classes:
[AndroidSaxFeedParser](#), [BaseFeedParser](#), [SaxFeedParser](#)

public interface **FeedParser**

Method Summary	
java.util.List< Message >	parse ()

Method Detail

parse

java.util.List<[Message](#)> **parse**()

daily49er.android.app

Class AndroidSaxFeedParser

java.lang.Object

└─ [daily49er.android.app.BaseFeedParser](#)

└─ **daily49er.android.app.AndroidSaxFeedParser**

All Implemented Interfaces:

[FeedParser](#)

```
public class AndroidSaxFeedParser
extends BaseFeedParser
```

Constructor Summary

[AndroidSaxFeedParser](#)(java.lang.String feedUrl)

Method Summary

java.util.List<[Message](#)> [parse](#)()

Methods inherited from class java.lang.Object

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

Constructor Detail

AndroidSaxFeedParser

```
public AndroidSaxFeedParser(java.lang.String feedUrl)
```

Method Detail

parse

```
public java.util.List<Message> parse()
```

daily49er.android.app

Class Article

```
java.lang.Object
└─ ListActivity
    └─ daily49er.android.app.Article
```

```
public class Article
    extends ListActivity
```

Constructor Summary

[Article](#)()

Method Summary

boolean	onCreateOptionsMenu (Menu menu) Creates a context menu from /res/layout/menu_button.xml; the context menu will present the user options to share an article that is being currently viewed through Facebook, Twitter and e-mail.
boolean	onOptionsItemSelected (MenuItem item) Perform the respective sharing actions for each type of service selected from the sharing context menu.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

Article

```
public Article()
```

Method Detail

onCreateOptionsMenu

```
public boolean onCreateOptionsMenu(Menu menu)
```

Creates a context menu from /res/layout/menu_button.xml; the context menu will present the user options to share an article that is being currently viewed through Facebook, Twitter and e-mail.

Parameters:

menu - - the menu that will be created through the menu inflater.

Returns:

true - if menu has been inflated and shown.

onOptionsItemSelected

```
public boolean onOptionsItemSelected(MenuItem item)
```

Perform the respective sharing actions for each type of service selected from the sharing context menu.

Parameters:

item - - the button selected from the menu button.

Returns:

true - once the case's action has been completed (if applicable).

[Overview](#) **[Package](#)** **[Class](#)** **[Use Tree](#)** **[Deprecated](#)** **[Index](#)** **[Help](#)**

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

SUMMARY: NESTED | FIELD | [CONSTR](#) | [METHOD](#)

DETAIL: FIELD | [CONSTR](#) | [METHOD](#)

daily49er.android.app

Class BaseFeedParser

java.lang.Object
└─daily49er.android.app.BaseFeedParser

All Implemented Interfaces:
[FeedParser](#)

Direct Known Subclasses:
[AndroidSaxFeedParser](#), [SaxFeedParser](#)

public abstract class **BaseFeedParser**
extends java.lang.Object
implements [FeedParser](#)

Method Summary

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface daily49er.android.app.[FeedParser](#)

[parse](#)

daily49er.android.app

Class ContactTab

java.lang.Object

└ Activity

└ daily49er.android.app.ContactTab

```
public class ContactTab
extends Activity
```

Constructor Summary

[ContactTab\(\)](#)

Method Summary

void	onClick (View view)
void	onCreate (Bundle savedInstanceState)

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

ContactTab

```
public ContactTab()
```

Method Detail

onCreate

```
public void onCreate(Bundle savedInstanceState)
```

onClick

```
public void onClick(View view)
```

Overview **Package** **Class** **Use Tree** **Deprecated** **Index** **Help**

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: NESTED | FIELD | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: FIELD | [CONSTR](#) | [METHOD](#)

daily49er.android.app

Class CustomAdapter

java.lang.Object



└ daily49er.android.app.CustomAdapter

```
public class CustomAdapter
extends
```

Constructor Summary

[CustomAdapter](#)(Context context, int textViewResourceId,
java.util.List<[Message](#)> titles)

Method Summary

View	getView (int position, View convertView, ViewGroup parent)
------	--

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CustomAdapter

```
public CustomAdapter(Context context,
                     int textViewResourceId,
                     java.util.List<Message> titles)
```

Method Detail

getView

```
public View getView(int position,
                   View convertView,
                   ViewGroup parent)
```


daily49er.android.app

Class Daily49er

java.lang.Object

└─ TabActivity

└─ **daily49er.android.app.Daily49er**

```
public class Daily49er
    extends TabActivity
```

Constructor Summary

[Daily49er\(\)](#)

Method Summary

void	onCreate (Bundle savedInstanceState) Called when the activity is first created.
------	--

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

Daily49er

```
public Daily49er()
```

Method Detail

onCreate

```
public void onCreate(Bundle savedInstanceState)
```

Called when the activity is first created.

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

SUMMARY: NESTED | FIELD | [CONSTR](#) | [METHOD](#)

DETAIL: FIELD | [CONSTR](#) | [METHOD](#)

daily49er.android.app

Class EventsTab

java.lang.Object
└ Activity
└ daily49er.android.app.EventsTab

```
public class EventsTab
extends Activity
```

The Events Tab is composed of the Daily49er's "Diversions" calendar which is implemented via an embedded WebView.

Author:
Alex Chavez

Constructor Summary

[EventsTab\(\)](#)

Method Summary

void	onCreate (Bundle savedInstanceState) Called when the activity is first created.
boolean	onKeyDown (int keyCode, KeyEvent event) This callback method will be called anytime a button is pressed while in the Events tab WebView; it will go back to a previous page in the webview.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

EventsTab

```
public EventsTab()
```

Method Detail

onCreate

```
public void onCreate(Bundle savedInstanceState)
```

Called when the activity is first created.

onKeyDown

```
public boolean onKeyDown(int keyCode,  
                           KeyEvent event)
```

This callback method will be called anytime a button is pressed while in the Events tab WebView; it will go back to a previous page in the webview.

Parameters:

keyCode - - the key code that is passed whenever any of the standard device buttons are pressed.

keyEvent - - the event that is triggered by the button press.

Returns:

true - if the "back" button has been pressed; then return the event and keycode for the button that was pressed.

[Overview](#) **[Package](#)** **[Class](#)** **[Use Tree](#)** **[Deprecated](#)** **[Index](#)** **[Help](#)**

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

daily49er.android.app

Class FacebookShare

```
java.lang.Object
└─Activity
    └─daily49er.android.app.FacebookShare
```

```
public class FacebookShare
extends Activity
```

Implementation for posting an article to the user's Facebook "wall". In order to make this class work, you'll need to create a new project that contains facebook SDK files and reference it to this project.

Author:

Alex Chavez

Constructor Summary

[FacebookShare](#)()

Method Summary

void	onActivityResult (int requestCode, int resultCode, Intent data) No clue what this does, it's used by the Facebook API.
void	onCreate (Bundle savedInstanceState) Called when the activity is first created.
boolean	restoreCredentials (Facebook facebook) Retrieve and restore the user's Facebook login credentials from the app's shared preferences.
boolean	saveCredentials (Facebook facebook) Saves the user's authenticated Facebook login credentials.
static void	setUrl (java.lang.String url) The URL is set in Article.java as it is the ciurrent article being viewed.
void	updateStatus () Creates and displays a Facebook dialog with the URL set; the dialog gives the user the option to enter a message into the text field.

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

FacebookShare

```
public FacebookShare()
```

Method Detail

setUrl

```
public static void setUrl(java.lang.String url)
```

The URL is set in Article.java as it is the current article being viewed.

Parameters:

`url` -- the URL that will be posted in the user's Facebook wall.

onCreate

```
public void onCreate(Bundle savedInstanceState)
```

Called when the activity is first created.

updateStatus

```
public void updateStatus()
```

Creates and displays a Facebook dialog with the URL set; the dialog gives the user the option to enter a message into the text field.

onActivityResult

```
public void onActivityResult(int requestCode,  
                             int resultCode,  
                             Intent data)
```

No clue what this does, it's used by the Facebook API.

saveCredentials

```
public boolean saveCredentials(Facebook facebook)
```

Saves the user's authenticated Facebook login credentials.

Parameters:

facebook - - an instantiated facebook object

Returns:

the saved credentials

restoreCredentials

```
public boolean restoreCredentials(Facebook facebook)
```

Retrieve and restore the user's Facebook login credentials from the app's shared preferences.

Parameters:

facebook - - an instantiated facebook object

Returns:

a boolean indicating if the stored login credentials are valid

[Overview](#) **[Package](#)** **[Class](#)** **[Use](#)** **[Tree](#)** **[Deprecated](#)** **[Index](#)** **[Help](#)**

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

daily49er.android.app

Class FeedParserFactory

java.lang.Object
└─daily49er.android.app.FeedParserFactory

```
public class FeedParserFactory
extends java.lang.Object
```

Constructor Summary

[FeedParserFactory\(\)](#)

Method Summary

static FeedParser	getParser()
-----------------------------------	-----------------------------

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

FeedParserFactory

```
public FeedParserFactory()
```

Method Detail

getParser

```
public static FeedParser getParser()
```


daily49er.android.app

Class LoadPage

```
java.lang.Object
├── WebViewClient
│   └── daily49er.android.app.LoadPage
```

```
public class LoadPage
    extends WebViewClient
```

The URL for a given WebView is loaded here.

Author:

Alex Chavez

Method Summary

boolean	shouldOverrideUrlLoading (WebView pageView, java.lang.String theUrl) Gives the application control of the WebView intent to stay in the application and not open a separate page.
---------	--

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

shouldOverrideUrlLoading

```
public boolean shouldOverrideUrlLoading(WebView pageView,
                                         java.lang.String theUrl)
```

Gives the application control of the WebView intent to stay in the application and not open a separate page.

Parameters:

view - - the WebView object that will load a web page.
theUrl - - the URL of the web page to be loaded.

daily49er.android.app

Class MediaTab

```
java.lang.Object
└─Activity
   └─daily49er.android.app.MediaTab
```

```
public class MediaTab
extends Activity
```

The media tab contains two buttons selection, "Video" and "Podcast". When a button is clicked, it redirects the user to the phone's default media viewer application for the user to view the respective content.

Constructor Summary

[MediaTab](#)()

Method Summary

void	onClick (View view)
void	onCreate (Bundle savedInstanceState)

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

MediaTab

```
public MediaTab()
```

Method Detail

onCreate

```
public void onCreate(Bundle savedInstanceState)
```

onClick

```
public void onClick(View view)
```

Overview **Package** **Class** **Use Tree** **Deprecated** **Index** **Help**

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [All Classes](#)
SUMMARY: NESTED | FIELD | [CONSTR](#) | [METHOD](#) DETAIL: FIELD | [CONSTR](#) | [METHOD](#)

daily49er.android.app

Class Message

java.lang.Object
└─daily49er.android.app.Message

All Implemented Interfaces:

java.lang.Comparable<[Message](#)>

public class **Message**
extends java.lang.Object
implements java.lang.Comparable<[Message](#)>

Constructor Summary	
Message	()

Method Summary	
int	compareTo (Message another)
Message	copy ()
boolean	equals (java.lang.Object obj)
java.lang.String	getAuthor ()
java.lang.String	getCategory ()
java.lang.String	getDate ()
java.lang.String	getDescription ()
java.net.URL	getLink ()
java.lang.String	getTitle ()
int	hashCode ()

void	<code>setAuthor</code> (java.lang.String author)
void	<code>setCategory</code> (java.lang.String category)
void	<code>setDate</code> (java.lang.String date)
void	<code>setDescription</code> (java.lang.String description)
void	<code>setLink</code> (java.lang.String link)
void	<code>setTitle</code> (java.lang.String title)
java.lang.String	<code>toString</code> ()

Methods inherited from class java.lang.Object

`getClass`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Constructor Detail

Message

```
public Message()
```

Method Detail

`getAuthor`

```
public java.lang.String getAuthor()
```

`setAuthor`

```
public void setAuthor(java.lang.String author)
```

`getCategory`

```
public java.lang.String getCategory()
```

`setCategory`

```
public void setCategory(java.lang.String category)
```

getTitle

```
public java.lang.String getTitle()
```

setTitle

```
public void setTitle(java.lang.String title)
```

getLink

```
public java.net.URL getLink()
```

setLink

```
public void setLink(java.lang.String link)
```

getDescription

```
public java.lang.String getDescription()
```

setDescription

```
public void setDescription(java.lang.String description)
```

getDate

```
public java.lang.String getDate()
```

setDate

```
public void setDate(java.lang.String date)
```

toString

```
public java.lang.String toString()
```

Overrides:

```
toString in class java.lang.Object
```

copy

```
public Message copy()
```

hashCode

```
public int hashCode()
```

Overrides:
hashCode in class java.lang.Object

equals

```
public boolean equals(java.lang.Object obj)
```

Overrides:
equals in class java.lang.Object

compareTo

```
public int compareTo(Message another)
```

Specified by:
compareTo in interface java.lang.Comparable<[Message](#)>

daily49er.android.app

Class NewsTab

```
java.lang.Object
└─ ListActivity
    └─ daily49er.android.app.NewsTab
```

```
public class NewsTab
    extends ListActivity
```

Field Summary

static java.util.List< Message >	messageList
--	-----------------------------

Constructor Summary

NewsTab ()

Method Summary

void	loadFeed () Parse, format, and print article's title in list view order.
void	onCreate (Bundle savedInstanceState) Called when the activity is first created.

Methods inherited from class java.lang.Object

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

Field Detail

messageList

```
public static java.util.List<Message> messageList
```

Constructor Detail

NewsTab

```
public NewsTab()
```

Method Detail

onCreate

```
public void onCreate(Bundle savedInstanceState)
```

Called when the activity is first created.

loadFeed

```
public void loadFeed()
```

Parse, format, and print article's title in list view order.

[Overview](#) [Package](#) **Class** [Use](#) [Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

daily49er.android.app

Class R

java.lang.Object
 └─ **daily49er.android.app.R**

```
public final class R
extends java.lang.Object
```

Nested Class Summary	
static class	R.array
static class	R.attr
static class	R.color
static class	R.drawable
static class	R.id
static class	R.layout
static class	R.string

Constructor Summary	
	R()

Method Summary	
----------------	--

Methods inherited from class java.lang.Object	
	equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

--	--

Constructor Detail

R

public **R**()

Overview **Package** **Class** **Use Tree** **Deprecated** **Index** **Help**

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

SUMMARY: [NESTED](#) | FIELD | [CONSTR](#) | [METHOD](#)

DETAIL: FIELD | [CONSTR](#) | METHOD

daily49er.android.app

Class R.array

java.lang.Object

└─ **daily49er.android.app.R.array**

Enclosing class:

[R](#)

```
public static final class R.array
extends java.lang.Object
```

Field Summary

static int	list_array
static int	list_values

Constructor Summary

[R.array\(\)](#)

Method Summary

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

list_array

```
public static final int list_array
```

See Also:

[Constant Field Values](#)

list_values

```
public static final int list_values
```

See Also:
[Constant Field Values](#)

Constructor Detail

R.array

```
public R.array()
```

daily49er.android.app

Class R.attr

java.lang.Object

└─daily49er.android.app.R.attr

Enclosing class:

[R](#)

```
public static final class R.attr
extends java.lang.Object
```

Constructor Summary

[R.attr\(\)](#)

Method Summary

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

R.attr

```
public R.attr()
```

daily49er.android.app

Class R.color

java.lang.Object

└─daily49er.android.app.R.color

Enclosing class:

[R](#)

```
public static final class R.color
extends java.lang.Object
```

Field Summary

static int	yellow
------------	------------------------

Constructor Summary

[R.color\(\)](#)

Method Summary

Methods inherited from class java.lang.Object

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

Field Detail

yellow

```
public static final int yellow
```

See Also:

[Constant Field Values](#)

Constructor Detail

R.color

```
public R.color()
```

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [All Classes](#)
SUMMARY: NESTED | [FIELD](#) | [CONSTR](#) | [METHOD](#) DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

daily49er.android.app

Class R.drawable

java.lang.Object

└─ **daily49er.android.app.R.drawable**

Enclosing class:

[R](#)

```
public static final class R.drawable
extends java.lang.Object
```

Field Summary

static int	<u>facebook_icon</u>
static int	<u>globe</u>
static int	<u>happy</u>
static int	<u>ic_tab_contact</u>
static int	<u>ic_tab_events</u>
static int	<u>ic_tab_home</u>
static int	<u>ic_tab_media</u>
static int	<u>icon</u>
static int	<u>media_bg</u>
static int	<u>movie</u>
static int	<u>splash_screen</u>
static int	<u>tiles</u>

Constructor Summary

[R.drawable\(\)](#)

Method Summary

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

facebook_icon

```
public static final int facebook_icon
```

See Also:

[Constant Field Values](#)

globe

```
public static final int globe
```

See Also:

[Constant Field Values](#)

happy

```
public static final int happy
```

See Also:

[Constant Field Values](#)

ic_tab_contact

```
public static final int ic_tab_contact
```

See Also:

[Constant Field Values](#)

ic_tab_events

public static final int **ic_tab_events**

See Also:

[Constant Field Values](#)

ic_tab_home

public static final int **ic_tab_home**

See Also:

[Constant Field Values](#)

ic_tab_media

public static final int **ic_tab_media**

See Also:

[Constant Field Values](#)

icon

public static final int **icon**

See Also:

[Constant Field Values](#)

media_bg

public static final int **media_bg**

See Also:

[Constant Field Values](#)

movie

public static final int **movie**

See Also:

[Constant Field Values](#)

splash_screen

public static final int **splash_screen**

See Also:

tiles

```
public static final int tiles
```

See Also:

[Constant Field Values](#)

Constructor Detail

R.drawable

```
public R.drawable()
```

[Overview](#) **[Package](#)** **[Class](#)** **[Use Tree](#)** **[Deprecated](#)** **[Index](#)** **[Help](#)**

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: NESTED | [FIELD](#) | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

daily49er.android.app

Class R.id

java.lang.Object

└─ **daily49er.android.app.R.id**

Enclosing class:

[R](#)

```
public static final class R.id
extends java.lang.Object
```

Field Summary

static int	articleview
static int	authorview
static int	categoryview
static int	contactButton
static int	email_share
static int	facebook_share
static int	ImageView01
static int	podcastButton
static int	ScrollView01
static int	TextView01
static int	titleview
static int	TwitterLoginButton

static int	TwitterLoginID
static int	TwitterPassword
static int	TwitterTitle
static int	videoButton
static int	widget30
static int	widget34
static int	widget35

Constructor Summary

[R.id\(\)](#)

Method Summary

Methods inherited from class `java.lang.Object`

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

ImageView01

public static final int **ImageView01**

See Also:

[Constant Field Values](#)

ScrollView01

public static final int **ScrollView01**

See Also:

[Constant Field Values](#)

TextView01

```
public static final int TextView01
```

See Also:

[Constant Field Values](#)

TwitterLoginButton

```
public static final int TwitterLoginButton
```

See Also:

[Constant Field Values](#)

TwitterLoginID

```
public static final int TwitterLoginID
```

See Also:

[Constant Field Values](#)

TwitterPassword

```
public static final int TwitterPassword
```

See Also:

[Constant Field Values](#)

TwitterTitle

```
public static final int TwitterTitle
```

See Also:

[Constant Field Values](#)

articleview

```
public static final int articleview
```

See Also:

[Constant Field Values](#)

authorview

```
public static final int authorview
```


See Also:

[Constant Field Values](#)

categoryview

```
public static final int categoryview
```

See Also:

[Constant Field Values](#)

contactButton

```
public static final int contactButton
```

See Also:

[Constant Field Values](#)

email_share

```
public static final int email_share
```

See Also:

[Constant Field Values](#)

facebook_share

```
public static final int facebook_share
```

See Also:

[Constant Field Values](#)

podcastButton

```
public static final int podcastButton
```

See Also:

[Constant Field Values](#)

titleview

```
public static final int titleview
```

See Also:

[Constant Field Values](#)

videoButton

```
public static final int videoButton
```

See Also:
[Constant Field Values](#)

widget30

```
public static final int widget30
```

See Also:
[Constant Field Values](#)

widget34

```
public static final int widget34
```

See Also:
[Constant Field Values](#)

widget35

```
public static final int widget35
```

See Also:
[Constant Field Values](#)

Constructor Detail

R.id

```
public R.id()
```

daily49er.android.app

Class R.layout

java.lang.Object

└─ **daily49er.android.app.R.layout**

Enclosing class:

[R](#)

```
public static final class R.layout
extends java.lang.Object
```

Field Summary

static int	article_page
static int	contact_tab
static int	list
static int	main
static int	media
static int	menu_button
static int	row
static int	splash
static int	twitter

Constructor Summary

[R.layout](#)()

Method Summary

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

article_page

```
public static final int article_page
```

See Also:

[Constant Field Values](#)

contact_tab

```
public static final int contact_tab
```

See Also:

[Constant Field Values](#)

list

```
public static final int list
```

See Also:

[Constant Field Values](#)

main

```
public static final int main
```

See Also:

[Constant Field Values](#)

media

```
public static final int media
```

See Also:

[Constant Field Values](#)

menu_button

```
public static final int menu_button
```

See Also:
[Constant Field Values](#)

row

```
public static final int row
```

See Also:
[Constant Field Values](#)

splash

```
public static final int splash
```

See Also:
[Constant Field Values](#)

twitter

```
public static final int twitter
```

See Also:
[Constant Field Values](#)

Constructor Detail

R.layout

```
public R.layout()
```

daily49er.android.app

Class R.string

java.lang.Object
└─ **daily49er.android.app.R.string**

Enclosing class:

[R](#)

```
public static final class R.string  
extends java.lang.Object
```

Field Summary

static int	app_name
static int	email_label Twitter Twitter sharing twt
static int	email_shortcut
static int	email_title
static int	facebook_label BEGIN MENU BUTTON STRINGS
static int	facebook_shortcut
static int	facebook_title
static int	hello
static int	twitterID
static int	twitterlogin
static int	twitterPW
static int	twitterertitle END MENU BUTTON STRINGS

Constructor Summary

[R.string\(\)](#)

Method Summary

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

app_name

```
public static final int app_name
```

See Also:

[Constant Field Values](#)

email_label

```
public static final int email_label
```

Twitter Twitter sharing twt

See Also:

[Constant Field Values](#)

email_shortcut

```
public static final int email_shortcut
```

See Also:

[Constant Field Values](#)

email_title

```
public static final int email_title
```

See Also:

[Constant Field Values](#)

facebook_label

```
public static final int facebook_label
```

BEGIN MENU BUTTON STRINGS

See Also:

[Constant Field Values](#)

facebook_shortcut

```
public static final int facebook_shortcut
```

See Also:

[Constant Field Values](#)

facebook_title

```
public static final int facebook_title
```

See Also:

[Constant Field Values](#)

hello

```
public static final int hello
```

See Also:

[Constant Field Values](#)

twitterID

```
public static final int twitterID
```

See Also:

[Constant Field Values](#)

twitterPW

```
public static final int twitterPW
```

See Also:

[Constant Field Values](#)

twitterlogin


```
public static final int twitterlogin
```

See Also:

[Constant Field Values](#)

twittertitle

```
public static final int twittertitle
```

END MENU BUTTON STRINGS

See Also:

[Constant Field Values](#)

Constructor Detail

R.string

```
public R.string()
```

Overview **Package** **Class** **Use Tree** **Deprecated** **Index** **Help**

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [All Classes](#)
SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

daily49er.android.app

Class RssHandler

```
java.lang.Object
├─ org.xml.sax.helpers.DefaultHandler
│   └─ daily49er.android.app.RssHandler
```

All Implemented Interfaces:

org.xml.sax.ContentHandler, org.xml.sax.DTDHandler, org.xml.sax.EntityResolver, org.xml.sax.ErrorHandler

```
public class RssHandler
extends org.xml.sax.helpers.DefaultHandler
```

Constructor Summary	
RssHandler	()

Method Summary	
void	characters (char[] ch, int start, int length)
void	endElement (java.lang.String uri, java.lang.String localName, java.lang.String name)
java.util.List< Message >	getMessages ()
void	startDocument ()
void	startElement (java.lang.String uri, java.lang.String localName, java.lang.String name, org.xml.sax.Attributes attributes)

Methods inherited from class org.xml.sax.helpers.DefaultHandler
endDocument, endPrefixMapping, error, fatalError, ignorableWhitespace, notationDecl, processingInstruction, resolveEntity, setDocumentLocator, skippedEntity, startPrefixMapping, unparsedEntityDecl, warning

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

RssHandler

```
public RssHandler()
```

Method Detail

getMessages

```
public java.util.List<Message> getMessages()
```

characters

```
public void characters(char[] ch,  
                       int start,  
                       int length)  
    throws org.xml.sax.SAXException
```

Specified by:

`characters` in interface `org.xml.sax.ContentHandler`

Overrides:

`characters` in class `org.xml.sax.helpers.DefaultHandler`

Throws:

`org.xml.sax.SAXException`

endElement

```
public void endElement(java.lang.String uri,  
                       java.lang.String localName,  
                       java.lang.String name)  
    throws org.xml.sax.SAXException
```

Specified by:

`endElement` in interface `org.xml.sax.ContentHandler`

Overrides:

`endElement` in class `org.xml.sax.helpers.DefaultHandler`

Throws:

`org.xml.sax.SAXException`

startDocument

```
public void startDocument()  
    throws org.xml.sax.SAXException
```

Specified by:

startDocument in interface org.xml.sax.ContentHandler

Overrides:

startDocument in class org.xml.sax.helpers.DefaultHandler

Throws:

org.xml.sax.SAXException

startElement

```
public void startElement(java.lang.String uri,  
    java.lang.String localName,  
    java.lang.String name,  
    org.xml.sax.Attributes attributes)  
    throws org.xml.sax.SAXException
```

Specified by:

startElement in interface org.xml.sax.ContentHandler

Overrides:

startElement in class org.xml.sax.helpers.DefaultHandler

Throws:

org.xml.sax.SAXException

[Overview](#) [Package](#) [Class](#) [Use Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

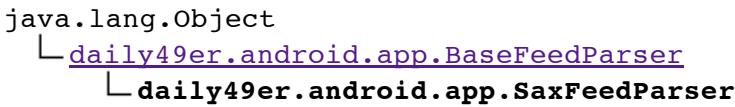
[FRAMES](#) [NO FRAMES](#) [All Classes](#)

SUMMARY: NESTED | FIELD | [CONSTR](#) | [METHOD](#)

DETAIL: FIELD | [CONSTR](#) | [METHOD](#)

daily49er.android.app

Class SaxFeedParser



All Implemented Interfaces:
[FeedParser](#)

```
public class SaxFeedParser
extends BaseFeedParser
```

Method Summary	
java.util.List< Message >	parse()

Methods inherited from class java.lang.Object
equals , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

Method Detail

parse

```
public java.util.List<Message> parse()
```

daily49er.android.app

Class SettingsTab

java.lang.Object
└─ PreferenceActivity
 └─ **daily49er.android.app.SettingsTab**

```
public class SettingsTab  
extends PreferenceActivity
```

The 'Settings' tab for the application.

Author:
Alex Chavez

Constructor Summary

[SettingsTab\(\)](#)

Method Summary

void	onCreate (Bundle savedInstanceState) Called when the activity is first created.
------	--

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

SettingsTab

```
public SettingsTab()
```

Method Detail

onCreate

```
public void onCreate(Bundle savedInstanceState)
```

Called when the activity is first created.

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: NESTED | FIELD | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: FIELD | [CONSTR](#) | [METHOD](#)

daily49er.android.app

Class SplashScreen

java.lang.Object

└ Activity

└ daily49er.android.app.SplashScreen

```
public class SplashScreen
extends Activity
```

This class creates a new thread for running the splash screen on the start of the application.

Author:

Alex Chavez

Constructor Summary

[SplashScreen](#)()

Method Summary

void	onCreate (Bundle savedInstanceState) Called when the activity is first created.
------	--

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

SplashScreen

```
public SplashScreen()
```

Method Detail

onCreate

```
public void onCreate(Bundle savedInstanceState)
```


Called when the activity is first created.

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: NESTED | FIELD | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: FIELD | [CONSTR](#) | [METHOD](#)

Package com.facebook.android

Interface Summary

AsyncFacebookRunner.RequestListener	Callback interface for API requests.
Facebook.DialogListener	Callback interface for dialog requests.

Class Summary

AsyncFacebookRunner	A sample implementation of asynchronous API requests.
DialogError	Encapsulation of Dialog Error.
Facebook	Main Facebook object for interacting with the Facebook developer API.
FacebookError	Encapsulation of a Facebook Error: a Facebook request that could not be fulfilled.
FbDialog	
R	
R.array	
R.attr	
R.color	
R.drawable	
R.id	
R.layout	
R.string	
Util	Utility class supporting the Facebook Object.

com.facebook.android

Interface AsyncFacebookRunner.RequestListener

Enclosing class:

[AsyncFacebookRunner](#)

```
public static interface AsyncFacebookRunner.RequestListener
```

Callback interface for API requests. Each method includes a 'state' parameter that identifies the calling request. It will be set to the value passed when originally calling the request method, or null if none was passed.

Method Summary

void	onComplete (java.lang.String response, java.lang.Object state) Called when a request completes with the given response.
void	onFacebookError (FacebookError e, java.lang.Object state) Called when the server-side Facebook method fails.
void	onFileNotFoundException (java.io.FileNotFoundException e, java.lang.Object state) Called when a request fails because the requested resource is invalid or does not exist.
void	onIOException (java.io.IOException e, java.lang.Object state) Called when a request has a network or request error.
void	onMalformedURLException (java.net.MalformedURLException e, java.lang.Object state) Called if an invalid graph path is provided (which may result in a malformed URL).

Method Detail

onComplete

```
void onComplete(java.lang.String response,  
                java.lang.Object state)
```

Called when a request completes with the given response. Executed by a background thread: do not update the UI in this method.

onIOException

```
void onIOException(java.io.IOException e,  
                  java.lang.Object state)
```

Called when a request has a network or request error. Executed by a background thread: do not update the UI in this method.

onFileNotFoundException

```
void onFileNotFoundException(java.io.FileNotFoundException e,  
                              java.lang.Object state)
```

Called when a request fails because the requested resource is invalid or does not exist. Executed by a background thread: do not update the UI in this method.

onMalformedURLException

```
void onMalformedURLException(java.net.MalformedURLException e,  
                             java.lang.Object state)
```

Called if an invalid graph path is provided (which may result in a malformed URL). Executed by a background thread: do not update the UI in this method.

onFacebookError

```
void onFacebookError(FacebookError e,  
                     java.lang.Object state)
```

Called when the server-side Facebook method fails. Executed by a background thread: do not update the UI in this method.

[Overview](#) **[Package](#)** **[Class](#)** **[Use](#)** **[Tree](#)** **[Deprecated](#)** **[Index](#)** **[Help](#)**

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#)

DETAIL: FIELD | CONSTR | [METHOD](#)

com.facebook.android

Interface Facebook.DialogListener

Enclosing class:

[Facebook](#)

```
public static interface Facebook.DialogListener
```

Callback interface for dialog requests.

Method Summary

void	onCancel () Called when a dialog is canceled by the user.
void	onComplete (Bundle values) Called when a dialog completes.
void	onError (DialogError e) Called when a dialog has an error.
void	onFacebookError (FacebookError e) Called when a Facebook responds to a dialog with an error.

Method Detail

onComplete

```
void onComplete(Bundle values)
```

Called when a dialog completes. Executed by the thread that initiated the dialog.

Parameters:

values - Key-value string pairs extracted from the response.

onFacebookError

```
void onFacebookError(FacebookError e)
```

Called when a Facebook responds to a dialog with an error. Executed by the thread that initiated the dialog.

onError

```
void onError(DialogError e)
```

Called when a dialog has an error. Executed by the thread that initiated the dialog.

onCancel

```
void onCancel()
```

Called when a dialog is canceled by the user. Executed by the thread that initiated the dialog.

`com.facebook.android`

Class AsyncFacebookRunner

`java.lang.Object``└ com.facebook.android.AsyncFacebookRunner`

```
public class AsyncFacebookRunner
extends java.lang.Object
```

A sample implementation of asynchronous API requests. This class provides the ability to execute API methods and have the call return immediately, without blocking the calling thread. This is necessary when accessing the API in the UI thread, for instance. The request response is returned to the caller via a callback interface, which the developer must implement. This sample implementation simply spawns a new thread for each request, and makes the API call immediately. This may work in many applications, but more sophisticated users may re-implement this behavior using a thread pool, a network thread, a request queue, or other mechanism. Advanced functionality could be built, such as rate-limiting of requests, as per a specific application's needs.

Author:

Jim Brusstar (jimbru@fb.com), Yariv Sadan (yariv@fb.com), Luke Shepard (lshepard@fb.com)

See Also:

[The callback interface.](#)

Nested Class Summary

static interface	AsyncFacebookRunner.RequestListener
	Callback interface for API requests.

Constructor Summary

AsyncFacebookRunner	(Facebook fb)
--	--------------------------------

Method Summary

void	logout (Context context, AsyncFacebookRunner.RequestListener listener)
void	logout (Context context, AsyncFacebookRunner.RequestListener listener, java.lang.Object state) Invalidate the current user session by removing the access token in memory, clearing the browser cookies, and calling auth.expireSession through the API.
void	request (Bundle parameters, AsyncFacebookRunner.RequestListener listener)

void	<code>request</code> (Bundle parameters, <code>AsyncFacebookRunner.RequestListener</code> listener, java.lang.Object state) Make a request to Facebook's old (pre-graph) API with the given parameters.
void	<code>request</code> (java.lang.String graphPath, Bundle parameters, <code>AsyncFacebookRunner.RequestListener</code> listener)
void	<code>request</code> (java.lang.String graphPath, Bundle parameters, <code>AsyncFacebookRunner.RequestListener</code> listener, java.lang.Object state) Make a request to the Facebook Graph API with the given string parameters using an HTTP GET (default method).
void	<code>request</code> (java.lang.String graphPath, Bundle parameters, java.lang.String httpMethod, <code>AsyncFacebookRunner.RequestListener</code> listener, java.lang.Object state) Make a request to the Facebook Graph API with the given HTTP method and string parameters.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

AsyncFacebookRunner

```
public AsyncFacebookRunner(Facebook fb)
```

Method Detail

logout

```
public void logout(Context context,
    AsyncFacebookRunner.RequestListener listener,
    java.lang.Object state)
```

Invalidate the current user session by removing the access token in memory, clearing the browser cookies, and calling `auth.expireSession` through the API. The application will be notified when logout is complete via the callback interface. Note that this method is asynchronous and the callback will be invoked in a background thread; operations that affect the UI will need to be posted to the UI thread or an appropriate handler.

Parameters:

`context` - The Android context in which the logout should be called: it should be the same context in which the login occurred in order to clear any stored cookies

`listener` - Callback interface to notify the application when the request has completed.

`state` - An arbitrary object used to identify the request when it returns to the callback. This has no effect on the request itself.

logout

```
public void logout(Context context,  
                   AsyncFacebookRunner.RequestListener listener)
```

request

```
public void request(Bundle parameters,  
                   AsyncFacebookRunner.RequestListener listener,  
                   java.lang.Object state)
```

Make a request to Facebook's old (pre-graph) API with the given parameters. One of the parameter keys must be "method" and its value should be a valid REST server API method. See <http://developers.facebook.com/docs/reference/rest/> Note that this method is asynchronous and the callback will be invoked in a background thread; operations that affect the UI will need to be posted to the UI thread or an appropriate handler. Example: `Bundle parameters = new Bundle(); parameters.putString("method", "auth.expireSession", new Listener()); String response = request(parameters);`

Parameters:

- parameters - Key-value pairs of parameters to the request. Refer to the documentation: one of the parameters must be "method".
 - listener - Callback interface to notify the application when the request has completed.
 - state - An arbitrary object used to identify the request when it returns to the callback. This has no effect on the request itself.
-

request

```
public void request(Bundle parameters,  
                   AsyncFacebookRunner.RequestListener listener)
```

request

```
public void request(java.lang.String graphPath,  
                   Bundle parameters,  
                   AsyncFacebookRunner.RequestListener listener,  
                   java.lang.Object state)
```

Make a request to the Facebook Graph API with the given string parameters using an HTTP GET (default method). See <http://developers.facebook.com/docs/api> Note that this method is asynchronous and the callback will be invoked in a background thread; operations that affect the UI will need to be posted to the UI thread or an appropriate handler.

Parameters:

- graphPath - Path to resource in the Facebook graph, e.g., to fetch data about the currently logged authenticated user, provide "me", which will fetch <http://graph.facebook.com/me>
- parameters - key-value string parameters, e.g. the path "search" with parameters "q" : "facebook" would produce a query for the following graph resource:

<https://graph.facebook.com/search?q=facebook>

listener - Callback interface to notify the application when the request has completed.

state - An arbitrary object used to identify the request when it returns to the callback. This has no effect on the request itself.

request

```
public void request(java.lang.String graphPath,  
                    Bundle parameters,  
                    AsyncFacebookRunner.RequestListener listener)
```

request

```
public void request(java.lang.String graphPath,  
                    Bundle parameters,  
                    java.lang.String httpMethod,  
                    AsyncFacebookRunner.RequestListener listener,  
                    java.lang.Object state)
```

Make a request to the Facebook Graph API with the given HTTP method and string parameters. Note that binary data parameters (e.g. pictures) are not yet supported by this helper function. See <http://developers.facebook.com/docs/api> Note that this method is asynchronous and the callback will be invoked in a background thread; operations that affect the UI will need to be posted to the UI thread or an appropriate handler.

Parameters:

graphPath - Path to resource in the Facebook graph, e.g., to fetch data about the currently logged authenticated user, provide "me", which will fetch <http://graph.facebook.com/me>

parameters - key-value string parameters, e.g. the path "search" with parameters {"q" : "facebook"} would produce a query for the following graph resource:

<https://graph.facebook.com/search?q=facebook>

httpMethod - http verb, e.g. "POST", "DELETE"

listener - Callback interface to notify the application when the request has completed.

state - An arbitrary object used to identify the request when it returns to the callback. This has no effect on the request itself.

[Overview](#) [Package](#) [Class](#) [Use Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

com.facebook.android

Class DialogError

```
java.lang.Object
└─ java.lang.Throwable
    └─ com.facebook.android.DialogError
```

All Implemented Interfaces:

java.io.Serializable

```
public class DialogError
extends java.lang.Throwable
```

Encapsulation of Dialog Error.

Author:

ssoneff@facebook.com

See Also:

[Serialized Form](#)

Constructor Summary

[DialogError](#)(java.lang.String message, int errorCode, java.lang.String failingUrl)

Method Summary

Methods inherited from class java.lang.Throwable

fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

DialogError

```
public DialogError(java.lang.String message,  
                  int errorCode,  
                  java.lang.String failingUrl)
```

[Overview](#) **[Package](#)** **[Class](#)** **[Use Tree](#)** **[Deprecated](#)** **[Index](#)** **[Help](#)**

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

SUMMARY: NESTED | FIELD | [CONSTR](#) | [METHOD](#)

DETAIL: FIELD | [CONSTR](#) | METHOD

com.facebook.android

Class Facebook

java.lang.Object

└ **com.facebook.android.Facebook**

```
public class Facebook
extends java.lang.Object
```

Main Facebook object for interacting with the Facebook developer API. Provides methods to log in and log out a user, make requests using the REST and Graph APIs, and start user interface interactions with the API (such as pop-ups promoting for credentials, permissions, stream posts, etc.)

Author:

Jim Brusstar (jimbru@facebook.com), Yariv Sadan (yariv@facebook.com), Luke Shepard (lshepard@facebook.com)

Nested Class Summary

static interface	Facebook.DialogListener Callback interface for dialog requests.
------------------	--

Field Summary

static java.lang.String	CANCEL_URI
static java.lang.String	EXPIRES
static java.lang.String	FB_APP_SIGNATURE
static int	FORCE_DIALOG_AUTH
static java.lang.String	REDIRECT_URI
static java.lang.String	SINGLE_SIGN_ON_DISABLED
static java.lang.String	TOKEN

Constructor Summary

Facebook(java.lang.String appId)
Constructor for Facebook object.

Method Summary

void	authorize (Activity activity, Facebook.DialogListener listener) Default authorize method.
void	authorize (Activity activity, java.lang.String[] permissions, Facebook.DialogListener listener) Authorize method that grants custom permissions.
void	authorize (Activity activity, java.lang.String[] permissions, int activityCode, Facebook.DialogListener listener) Full authorize method.
void	authorizeCallback (int requestCode, int resultCode, Intent data) IMPORTANT: This method must be invoked at the top of the calling activity's onActivityResult() function or Facebook authentication will not function properly! If your calling activity does not currently implement onActivityResult(), you must implement it and include a call to this method if you intend to use the authorize() method in this SDK.
void	dialog (Context context, java.lang.String action, Bundle parameters, Facebook.DialogListener listener) Generate a UI dialog for the request action in the given Android context with the provided parameters.
void	dialog (Context context, java.lang.String action, Facebook.DialogListener listener) Generate a UI dialog for the request action in the given Android context.
long	getAccessExpires () Retrieve the current session's expiration time (in milliseconds since Unix epoch), or 0 if the session doesn't expire or doesn't exist.
java.lang.String	getAccessToken () Retrieve the OAuth 2.0 access token for API access: treat with care.
java.lang.String	getAppId ()
boolean	isSessionValid ()
java.lang.String	logout (Context context) Invalidate the current user session by removing the access token in memory, clearing the browser cookie, and calling auth.expireSession through the API.
java.lang.String	request (Bundle parameters) Make a request to Facebook's old (pre-graph) API with the given parameters.
java.lang.String	request (java.lang.String graphPath, Bundle parameters) Make a request to the Facebook Graph API with the given string parameters using an HTTP GET (default method).

java.lang.String	<code>request</code> (java.lang.String graphPath, Bundle params, java.lang.String httpMethod) Synchronously make a request to the Facebook Graph API with the given HTTP method and string parameters.
void	<code>setAccessExpires</code> (long time) Set the current session's expiration time (in milliseconds since Unix epoch), or 0 if the session doesn't expire.
void	<code>setAccessExpiresIn</code> (java.lang.String expiresIn) Set the current session's duration (in seconds since Unix epoch).
void	<code>setAccessToken</code> (java.lang.String token) Set the OAuth 2.0 access token for API access.
void	<code>setAppId</code> (java.lang.String appId)

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

REDIRECT_URI

```
public static final java.lang.String REDIRECT_URI
```

See Also:

[Constant Field Values](#)

CANCEL_URI

```
public static final java.lang.String CANCEL_URI
```

See Also:

[Constant Field Values](#)

TOKEN

```
public static final java.lang.String TOKEN
```

See Also:

[Constant Field Values](#)

EXPIRES

```
public static final java.lang.String EXPIRES
```

See Also:

[Constant Field Values](#)

SINGLE_SIGN_ON_DISABLED

```
public static final java.lang.String SINGLE_SIGN_ON_DISABLED
```

See Also:

[Constant Field Values](#)

FORCE_DIALOG_AUTH

```
public static final int FORCE_DIALOG_AUTH
```

See Also:

[Constant Field Values](#)

FB_APP_SIGNATURE

```
public static final java.lang.String FB_APP_SIGNATURE
```

See Also:

[Constant Field Values](#)

Constructor Detail

Facebook

```
public Facebook(java.lang.String appId)
```

Constructor for Facebook object.

Parameters:

appId - Your Facebook application ID. Found at www.facebook.com/developers/apps.php.

Method Detail

authorize

```
public void authorize(Activity activity,  
    Facebook.DialogListener listener)
```

Default authorize method. Grants only basic permissions. See [authorize\(\)](#) below for @params.

authorize


```
public void authorize(Activity activity,  
    java.lang.String[] permissions,  
    Facebook.DialogListener listener)
```

Authorize method that grants custom permissions. See `authorize()` below for @params.

authorize

```
public void authorize(Activity activity,  
    java.lang.String[] permissions,  
    int activityCode,  
    Facebook.DialogListener listener)
```

Full authorize method. Starts either an Activity or a dialog which prompts the user to log in to Facebook and grant the requested permissions to the given application. This method will, when possible, use Facebook's single sign-on for Android to obtain an access token. This involves proxying a call through the Facebook for Android stand-alone application, which will handle the authentication flow, and return an OAuth access token for making API calls. Because this process will not be available for all users, if single sign-on is not possible, this method will automatically fall back to the OAuth 2.0 User-Agent flow. In this flow, the user credentials are handled by Facebook in an embedded WebView, not by the client application. As such, the dialog makes a network request and renders HTML content rather than a native UI. The access token is retrieved from a redirect to a special URL that the WebView handles. Note that User credentials could be handled natively using the OAuth 2.0 Username and Password Flow, but this is not supported by this SDK. See <http://developers.facebook.com/docs/authentication/> and <http://wiki.oauth.net/OAuth-2> for more details. Note that this method is asynchronous and the callback will be invoked in the original calling thread (not in a background thread). Also note that requests may be made to the API without calling `authorize` first, in which case only public information is returned. **IMPORTANT:** Note that single sign-on authentication will not function correctly if you do not include a call to the `authorizeCallback()` method in your `onActivityResult()` function! Please see below for more information. single sign-on may be disabled by passing `FORCE_DIALOG_AUTH` as the `activityCode` parameter in your call to `authorize()`.

Parameters:

`activity` - The Android activity in which we want to display the authorization dialog.
`applicationId` - The Facebook application identifier e.g. "350685531728"
`permissions` - A list of permissions required for this application: e.g. "read_stream", "publish_stream", "offline_access", etc. see <http://developers.facebook.com/docs/authentication/permissions> This parameter should not be null -- if you do not require any permissions, then pass in an empty String array.
`activityCode` - Single sign-on requires an activity result to be called back to the client application -- if you are waiting on other activities to return data, pass a custom activity code here to avoid collisions. If you would like to force the use of legacy dialog-based authorization, pass `FORCE_DIALOG_AUTH` for this parameter. Otherwise just omit this parameter and Facebook will use a suitable default. See <http://developer.android.com/reference/android/app/Activity.html> for more information.
`listener` - Callback interface for notifying the calling application when the authentication dialog has completed, failed, or been canceled.

authorizeCallback

```
public void authorizeCallback(int requestCode,  
                             int resultCode,  
                             Intent data)
```

IMPORTANT: This method must be invoked at the top of the calling activity's `onActivityResult()` function or Facebook authentication will not function properly! If your calling activity does not currently implement `onActivityResult()`, you must implement it and include a call to this method if you intend to use the `authorize()` method in this SDK. For more information, see [http://developer.android.com/reference/android/app/Activity.html#onActivityResult\(int, int, android.content.Intent\)](http://developer.android.com/reference/android/app/Activity.html#onActivityResult(int, int, android.content.Intent))

logout

```
public java.lang.String logout(Context context)  
    throws java.net.MalformedURLException,  
           java.io.IOException
```

Invalidate the current user session by removing the access token in memory, clearing the browser cookie, and calling `auth.expireSession` through the API. Note that this method blocks waiting for a network response, so do not call it in a UI thread.

Parameters:

`context` - The Android context in which the logout should be called: it should be the same context in which the login occurred in order to clear any stored cookies

Returns:

JSON string representation of the `auth.expireSession` response ("true" if successful)

Throws:

`java.io.IOException`
`java.net.MalformedURLException`

request

```
public java.lang.String request(Bundle parameters)  
    throws java.net.MalformedURLException,  
           java.io.IOException
```

Make a request to Facebook's old (pre-graph) API with the given parameters. One of the parameter keys must be "method" and its value should be a valid REST server API method. See <http://developers.facebook.com/docs/reference/rest/> Note that this method blocks waiting for a network response, so do not call it in a UI thread. Example: `Bundle parameters = new Bundle(); parameters.putString("method", "auth.expireSession"); String response = request(parameters);`

Parameters:

`parameters` - Key-value pairs of parameters to the request. Refer to the documentation: one of the parameters must be "method".

Returns:

JSON string representation of the response

Throws:

`java.io.IOException` - if a network error occurs
`java.net.MalformedURLException` - if accessing an invalid endpoint

request

```
public java.lang.String request(java.lang.String graphPath,  
                                Bundle parameters)  
    throws java.net.MalformedURLException,  
           java.io.IOException
```

Make a request to the Facebook Graph API with the given string parameters using an HTTP GET (default method). See <http://developers.facebook.com/docs/api> Note that this method blocks waiting for a network response, so do not call it in a UI thread.

Parameters:

graphPath - Path to resource in the Facebook graph, e.g., to fetch data about the currently logged authenticated user, provide "me", which will fetch <http://graph.facebook.com/me>
parameters - key-value string parameters, e.g. the path "search" with parameters "q" : "facebook" would produce a query for the following graph resource:
<https://graph.facebook.com/search?q=facebook>

Returns:

JSON string representation of the response

Throws:

java.io.IOException
java.net.MalformedURLException

request

```
public java.lang.String request(java.lang.String graphPath,  
                                Bundle params,  
                                java.lang.String httpMethod)  
    throws java.io.FileNotFoundException,  
           java.net.MalformedURLException,  
           java.io.IOException
```

Synchronously make a request to the Facebook Graph API with the given HTTP method and string parameters. Note that binary data parameters (e.g. pictures) are not yet supported by this helper function. See <http://developers.facebook.com/docs/api> Note that this method blocks waiting for a network response, so do not call it in a UI thread.

Parameters:

graphPath - Path to resource in the Facebook graph, e.g., to fetch data about the currently logged authenticated user, provide "me", which will fetch <http://graph.facebook.com/me>
params - Key-value string parameters, e.g. the path "search" with parameters {"q" : "facebook"} would produce a query for the following graph resource:
<https://graph.facebook.com/search?q=facebook>
httpMethod - http verb, e.g. "GET", "POST", "DELETE"

Returns:

JSON string representation of the response

Throws:

java.io.IOException
java.net.MalformedURLException

dialog

```
public void dialog(Context context,  
                   java.lang.String action,  
                   Facebook.DialogListener listener)
```

Generate a UI dialog for the request action in the given Android context. Note that this method is asynchronous and the callback will be invoked in the original calling thread (not in a background thread).

Parameters:

context - The Android context in which we will generate this dialog.
action - String representation of the desired method: e.g. "login", "stream.publish", ...
listener - Callback interface to notify the application when the dialog has completed.

dialog

```
public void dialog(Context context,  
                   java.lang.String action,  
                   Bundle parameters,  
                   Facebook.DialogListener listener)
```

Generate a UI dialog for the request action in the given Android context with the provided parameters. Note that this method is asynchronous and the callback will be invoked in the original calling thread (not in a background thread).

Parameters:

context - The Android context in which we will generate this dialog.
action - String representation of the desired method: e.g. "feed" ...
parameters - String key-value pairs to be passed as URL parameters.
listener - Callback interface to notify the application when the dialog has completed.

isSessionValid

```
public boolean isSessionValid()
```

Returns:

boolean - whether this object has an non-expired session token

getAccessToken

```
public java.lang.String getAccessToken()
```

Retrieve the OAuth 2.0 access token for API access: treat with care. Returns null if no session exists.

Returns:

String - access token

getAccessExpires

```
public long getAccessExpires()
```

Retrieve the current session's expiration time (in milliseconds since Unix epoch), or 0 if the session doesn't expire or doesn't exist.

Returns:

long - session expiration time

setAccessToken

```
public void setAccessToken(java.lang.String token)
```

Set the OAuth 2.0 access token for API access.

Parameters:

token -- access token

setAccessExpires

```
public void setAccessExpires(long time)
```

Set the current session's expiration time (in milliseconds since Unix epoch), or 0 if the session doesn't expire.

Parameters:

time -- timestamp in milliseconds

setAccessExpiresIn

```
public void setAccessExpiresIn(java.lang.String expiresIn)
```

Set the current session's duration (in seconds since Unix epoch).

Parameters:

expiresIn -- duration in seconds

getAppId

```
public java.lang.String getAppId()
```

setAppId

```
public void setAppId(java.lang.String appId)
```

Overview **Package** **Class** **Use Tree** **Deprecated** **Index** **Help**

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#)

DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

com.facebook.android

Class FacebookError

```
java.lang.Object
├─ java.lang.Throwable
│   └─ com.facebook.android.FacebookError
```

All Implemented Interfaces:

java.io.Serializable

```
public class FacebookError
extends java.lang.Throwable
```

Encapsulation of a Facebook Error: a Facebook request that could not be fulfilled.

Author:

ssoneff@facebook.com

See Also:

[Serialized Form](#)

Constructor Summary

[FacebookError](#)(java.lang.String message)

[FacebookError](#)(java.lang.String message, java.lang.String type, int code)

Method Summary

int	getErrorCode ()
-----	---------------------------------

java.lang.String	getErrorType ()
------------------	---------------------------------

Methods inherited from class java.lang.Throwable

fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

FacebookError

```
public FacebookError(java.lang.String message)
```

FacebookError

```
public FacebookError(java.lang.String message,  
                      java.lang.String type,  
                      int code)
```

Method Detail

getErrorCode

```
public int getErrorCode()
```

getErrorType

```
public java.lang.String getErrorType()
```

[Overview](#) [Package](#) [Class](#) [Use Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: NESTED | FIELD | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: FIELD | [CONSTR](#) | [METHOD](#)

com.facebook.android

Class FbDialog

java.lang.Object

└ Dialog

└ com.facebook.android.FbDialog

```
public class FbDialog
    extends Dialog
```

Constructor Summary

[FbDialog](#)(Context context, java.lang.String url, [Facebook.DialogListener](#) listener)

Method Summary

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

FbDialog

```
public FbDialog(Context context,
                java.lang.String url,
                Facebook.DialogListener listener)
```

com.facebook.android

Class R

java.lang.Object
└─ com.facebook.android.R

```
public final class R
extends java.lang.Object
```

Nested Class Summary	
static class	R.array
static class	R.attr
static class	R.color
static class	R.drawable
static class	R.id
static class	R.layout
static class	R.string

Constructor Summary	
	R()

Method Summary	
----------------	--

Methods inherited from class java.lang.Object	
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait	

--

Constructor Detail

R

public **R**()

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

SUMMARY: [NESTED](#) | FIELD | [CONSTR](#) | [METHOD](#)

DETAIL: FIELD | [CONSTR](#) | METHOD

com.facebook.android

Class R.array

java.lang.Object

└─ **com.facebook.android.R.array**

Enclosing class:

[R](#)

```
public static final class R.array
extends java.lang.Object
```

Field Summary

static int	list_array
static int	list_values

Constructor Summary

[R.array\(\)](#)

Method Summary

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

list_array

```
public static final int list_array
```

See Also:

[Constant Field Values](#)

list_values

```
public static final int list_values
```

See Also:

[Constant Field Values](#)

Constructor Detail

R.array

```
public R.array()
```

com.facebook.android

Class R.attr

java.lang.Object

└ **com.facebook.android.R.attr**

Enclosing class:

[R](#)

```
public static final class R.attr
extends java.lang.Object
```

Constructor Summary

[R.attr\(\)](#)

Method Summary

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

R.attr

```
public R.attr()
```

com.facebook.android

Class R.color

java.lang.Object

└─ com.facebook.android.R.color

Enclosing class:

[R](#)

```
public static final class R.color
extends java.lang.Object
```

Field Summary

static int	yellow
------------	------------------------

Constructor Summary

R.color ()

Method Summary

Methods inherited from class java.lang.Object

<code>equals</code> , <code>getClass</code> , <code>hashCode</code> , <code>notify</code> , <code>notifyAll</code> , <code>toString</code> , <code>wait</code> , <code>wait</code> , <code>wait</code>
--

Field Detail

yellow

```
public static final int yellow
```

See Also:

[Constant Field Values](#)

Constructor Detail

R.color

```
public R.color()
```

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: NESTED | [FIELD](#) | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: [FIELD](#) | [CONSTR](#) | METHOD

com.facebook.android

Class R.drawable

java.lang.Object

└ **com.facebook.android.R.drawable**

Enclosing class:

[R](#)

```
public static final class R.drawable
extends java.lang.Object
```

Field Summary

static int	facebook_icon
static int	globe
static int	happy
static int	ic_tab_contact
static int	ic_tab_events
static int	ic_tab_home
static int	ic_tab_media
static int	icon
static int	media_bg
static int	movie
static int	splash_screen
static int	tiles

Constructor Summary

[R.drawable\(\)](#)

Method Summary

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

facebook_icon

```
public static final int facebook_icon
```

See Also:

[Constant Field Values](#)

globe

```
public static final int globe
```

See Also:

[Constant Field Values](#)

happy

```
public static final int happy
```

See Also:

[Constant Field Values](#)

ic_tab_contact

```
public static final int ic_tab_contact
```

See Also:

[Constant Field Values](#)

ic_tab_events

public static final int **ic_tab_events**

See Also:

[Constant Field Values](#)

ic_tab_home

public static final int **ic_tab_home**

See Also:

[Constant Field Values](#)

ic_tab_media

public static final int **ic_tab_media**

See Also:

[Constant Field Values](#)

icon

public static final int **icon**

See Also:

[Constant Field Values](#)

media_bg

public static final int **media_bg**

See Also:

[Constant Field Values](#)

movie

public static final int **movie**

See Also:

[Constant Field Values](#)

splash_screen

public static final int **splash_screen**

See Also:

tiles

```
public static final int tiles
```

See Also:

[Constant Field Values](#)

Constructor Detail

R.drawable

```
public R.drawable()
```

[Overview](#) **[Package](#)** **[Class](#)** **[Use Tree](#)** **[Deprecated](#)** **[Index](#)** **[Help](#)**

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: NESTED | [FIELD](#) | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

com.facebook.android

Class R.id

java.lang.Object

└─ com.facebook.android.R.id

Enclosing class:

[R](#)

```
public static final class R.id
extends java.lang.Object
```

Field Summary	
static int	articleview
static int	authorview
static int	categoryview
static int	contactButton
static int	email_share
static int	facebook_share
static int	ImageView01
static int	podcastButton
static int	ScrollView01
static int	TextView01
static int	titleview
static int	TwitterLoginButton

static int	TwitterLoginID
static int	TwitterPassword
static int	TwitterTitle
static int	videoButton
static int	widget30
static int	widget34
static int	widget35

Constructor Summary

[R.id\(\)](#)

Method Summary

Methods inherited from class `java.lang.Object`

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

ImageView01

public static final int **ImageView01**

See Also:

[Constant Field Values](#)

ScrollView01

public static final int **ScrollView01**

See Also:

[Constant Field Values](#)

TextView01

```
public static final int TextView01
```

See Also:

[Constant Field Values](#)

TwitterLoginButton

```
public static final int TwitterLoginButton
```

See Also:

[Constant Field Values](#)

TwitterLoginID

```
public static final int TwitterLoginID
```

See Also:

[Constant Field Values](#)

TwitterPassword

```
public static final int TwitterPassword
```

See Also:

[Constant Field Values](#)

TwitterTitle

```
public static final int TwitterTitle
```

See Also:

[Constant Field Values](#)

articleview

```
public static final int articleview
```

See Also:

[Constant Field Values](#)

authorview

```
public static final int authorview
```

See Also:

[Constant Field Values](#)

categoryview

```
public static final int categoryview
```

See Also:

[Constant Field Values](#)

contactButton

```
public static final int contactButton
```

See Also:

[Constant Field Values](#)

email_share

```
public static final int email_share
```

See Also:

[Constant Field Values](#)

facebook_share

```
public static final int facebook_share
```

See Also:

[Constant Field Values](#)

podcastButton

```
public static final int podcastButton
```

See Also:

[Constant Field Values](#)

titleview

```
public static final int titleview
```

See Also:

[Constant Field Values](#)

videoButton

```
public static final int videoButton
```

See Also:
[Constant Field Values](#)

widget30

```
public static final int widget30
```

See Also:
[Constant Field Values](#)

widget34

```
public static final int widget34
```

See Also:
[Constant Field Values](#)

widget35

```
public static final int widget35
```

See Also:
[Constant Field Values](#)

Constructor Detail

R.id

```
public R.id()
```

com.facebook.android

Class R.layout

java.lang.Object

└ **com.facebook.android.R.layout**

Enclosing class:

[R](#)

```
public static final class R.layout
extends java.lang.Object
```

Field Summary

static int	article_page
static int	contact_tab
static int	list
static int	main
static int	media
static int	menu_button
static int	row
static int	splash
static int	twitter

Constructor Summary

[R.layout](#)()

Method Summary

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

article_page

```
public static final int article_page
```

See Also:

[Constant Field Values](#)

contact_tab

```
public static final int contact_tab
```

See Also:

[Constant Field Values](#)

list

```
public static final int list
```

See Also:

[Constant Field Values](#)

main

```
public static final int main
```

See Also:

[Constant Field Values](#)

media

```
public static final int media
```

See Also:

[Constant Field Values](#)

menu_button

```
public static final int menu_button
```

See Also:
[Constant Field Values](#)

row

```
public static final int row
```

See Also:
[Constant Field Values](#)

splash

```
public static final int splash
```

See Also:
[Constant Field Values](#)

twitter

```
public static final int twitter
```

See Also:
[Constant Field Values](#)

Constructor Detail

R.layout

```
public R.layout()
```

com.facebook.android

Class R.string

java.lang.Object

└─ com.facebook.android.R.string

Enclosing class:

[R](#)

```
public static final class R.string
extends java.lang.Object
```

Field Summary	
static int	app_name
static int	email_label Twitter Twitter sharing twt
static int	email_shortcut
static int	email_title
static int	facebook_label BEGIN MENU BUTTON STRINGS
static int	facebook_shortcut
static int	facebook_title
static int	hello
static int	twitterID
static int	twitterlogin
static int	twitterPW
static int	twitterertitle END MENU BUTTON STRINGS

Constructor Summary

[R.string\(\)](#)

Method Summary

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

app_name

```
public static final int app_name
```

See Also:

[Constant Field Values](#)

email_label

```
public static final int email_label
```

Twitter Twitter sharing twt

See Also:

[Constant Field Values](#)

email_shortcut

```
public static final int email_shortcut
```

See Also:

[Constant Field Values](#)

email_title

```
public static final int email_title
```

See Also:

[Constant Field Values](#)

facebook_label

```
public static final int facebook_label
```

BEGIN MENU BUTTON STRINGS

See Also:

[Constant Field Values](#)

facebook_shortcut

```
public static final int facebook_shortcut
```

See Also:

[Constant Field Values](#)

facebook_title

```
public static final int facebook_title
```

See Also:

[Constant Field Values](#)

hello

```
public static final int hello
```

See Also:

[Constant Field Values](#)

twitterID

```
public static final int twitterID
```

See Also:

[Constant Field Values](#)

twitterPW

```
public static final int twitterPW
```

See Also:

[Constant Field Values](#)

twitterlogin

```
public static final int twitterlogin
```

See Also:

[Constant Field Values](#)

twittertitle

```
public static final int twittertitle
```

END MENU BUTTON STRINGS

See Also:

[Constant Field Values](#)

Constructor Detail

R.string

```
public R.string()
```

Overview **Package** **Class** **Use Tree** **Deprecated** **Index** **Help**

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [All Classes](#)
SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

com.facebook.android

Class Util

java.lang.Object

└ **com.facebook.android.Util**

```
public final class Util
extends java.lang.Object
```

Utility class supporting the Facebook Object.

Author:

ssoneff@facebook.com

Constructor Summary

[Util](#) ()

Method Summary

static void	clearCookies (Context context)
static Bundle	decodeUrl (java.lang.String s)
static java.lang.String	encodePostBody (Bundle parameters, java.lang.String boundary) Generate the multi-part post body providing the parameters and boundary string
static java.lang.String	encodeUrl (Bundle parameters)
static java.lang.String	openUrl (java.lang.String url, java.lang.String method, Bundle params) Connect to an HTTP URL and return the response as a string.
static org.json.JSONObject	parseJson (java.lang.String response) Parse a server response into a JSON Object.
static Bundle	parseUrl (java.lang.String url) Parse a URL query and fragment parameters into a key-value bundle.
static void	showAlert (Context context, java.lang.String title, java.lang.String text) Display a simple alert dialog with the given text and title.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

Util

```
public Util()
```

Method Detail

encodePostBody

```
public static java.lang.String encodePostBody(Bundle parameters,  
                                                java.lang.String boundary)
```

Generate the multi-part post body providing the parameters and boundary string

Parameters:

`parameters` - the parameters need to be posted

`boundary` - the random string as boundary

Returns:

a string of the post body

encodeUrl

```
public static java.lang.String encodeUrl(Bundle parameters)
```

decodeUrl

```
public static Bundle decodeUrl(java.lang.String s)
```

parseUrl

```
public static Bundle parseUrl(java.lang.String url)
```

Parse a URL query and fragment parameters into a key-value bundle.

Parameters:

`url` - the URL to parse

Returns:

a dictionary bundle of keys and values

openUrl

```
public static java.lang.String openUrl(java.lang.String url,  
                                         java.lang.String method,  
                                         Bundle params)  
    throws java.net.MalformedURLException,  
           java.io.IOException
```

Connect to an HTTP URL and return the response as a string. Note that the HTTP method override is used on non-GET requests. (i.e. requests are made as "POST" with method specified in the body).

Parameters:

url -- the resource to open: must be a wellformed URL
method -- the HTTP method to use ("GET", "POST", etc.)
params -- the query parameter for the URL (e.g. access_token=foo)

Returns:

the URL contents as a String

Throws:

java.net.MalformedURLException -- if the URL format is invalid
java.io.IOException -- if a network problem occurs

clearCookies

```
public static void clearCookies(Context context)
```

parseJson

```
public static org.json.JSONObject parseJson(java.lang.String response)  
    throws org.json.JSONException,  
           FacebookError
```

Parse a server response into a JSON Object. This is a basic implementation using org.json.JSONObject representation. More sophisticated applications may wish to do their own parsing. The parsed JSON is checked for a variety of error fields and a FacebookException is thrown if an error condition is set, populated with the error message and error type or code if available.

Parameters:

response -- string representation of the response

Returns:

the response as a JSON Object

Throws:

org.json.JSONException -- if the response is not valid JSON
[FacebookError](#) -- if an error condition is set

showAlert

```
public static void showAlert(Context context,  
                             java.lang.String title,  
                             java.lang.String text)
```

Display a simple alert dialog with the given text and title.

Parameters:

`context` - Android context in which the dialog should be displayed

`title` - Alert dialog title

`text` - Alert dialog message

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: NESTED | FIELD | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: FIELD | [CONSTR](#) | [METHOD](#)

```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: AndroidSaxFeedParser.java */

package daily49er.android.app;

import java.util.ArrayList;
import java.util.List;

import android.sax.Element;
import android.sax.EndElementListener;
import android.sax.EndTextElementListener;
import android.sax.RootElement;
import android.util.Xml;

public class AndroidSaxFeedParser extends BaseFeedParser {
    static final String RSS = "rss";
    public AndroidSaxFeedParser(String feedUrl) {
        super(feedUrl);
    }

    public List<Message> parse() {
        final Message currentMessage = new Message();
        RootElement root = new RootElement(RSS);
        final List<Message> messages = new ArrayList<Message>();
        Element channel = root.getChild(CHANNEL);
        Element item = channel.getChild(ITEM);
        item.setEndElementListener(new EndElementListener(){
            public void end() {
                messages.add(currentMessage.copy());
            }
        });

        item.getChild(TITLE).setEndTextElementListener(new
            EndTextElementListener(){
                public void end(String body) {
                    currentMessage.setTitle(body);
                }
            });
        item.getChild(LINK).setEndTextElementListener(new
            EndTextElementListener(){
                public void end(String body) {
                    currentMessage.setLink(body);
                }
            });
    }
}
```

```
        item.getChild(DESCRIPTION).setEndElementListener(new
            EndTextElementListener(){
                public void end(String body) {
                    currentMessage.setDescription(body);
                }
            });
        item.getChild(PUB_DATE).setEndElementListener(new
            EndTextElementListener(){
                public void end(String body) {
                    currentMessage.setDate(body);
                }
            });
        item.getChild(AUTHOR).setEndElementListener(new
            EndTextElementListener(){
                public void end(String body){
                    currentMessage.setAuthor(body);
                }
            });
        item.getChild(CATEGORY).setEndElementListener(new
            EndTextElementListener(){
                public void end(String body){
                    currentMessage.setCategory(body);
                }
            });
        try {
            Xml.parse(this.getInputStream(), Xml.Encoding.UTF_8, root.
                getContentHandler());
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
        return messages;
    }
}
```

```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: Article.java */

package daily49er.android.app;

import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.util.Log;
import android.widget.AdapterView;
import daily49er.android.app.NewsTab;
import java.util.ArrayList;
import java.util.List;

public class Article extends ListActivity
{
    NewsTab newsTab = new NewsTab();
    private Long articlePosition;
    String articleBody;
    String articleTitle;
    String articleUrl;
    String articleAuthor;
    String articleDate;
    String articleCategory;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.list);
        articlePosition = this getIntent().getExtras().getLong("position");
        loadFeed();
    } //end onCreate()

    /**
     * Creates a context menu from /res/layout/menu_button.xml; the
     * context menu will present the user options to share an article
     * that is being currently viewed through Facebook, Twitter
     * and e-mail.
     * @param menu - the menu that will be created through the

```

```

    * menu inflater.
    * @return true - if menu has been inflated and shown.
    */
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
    super.onCreateOptionsMenu(menu);
    /*
     * Returns an instance of MenuInflater that we use to read
     * the menu definition from menu_button.xml and turns it
     * into a "real view".
     */
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.layout.menu_button, menu);
    return true;
} //end onCreateOptionsMenu()

/**
 * Perform the respective sharing actions for each type of service
 * selected from the sharing context menu.
 * @param item - the button selected from the menu button.
 * @return true - once the case's action has been completed (if applicable)
 */
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
    switch(item.getItemId())
    {
        case R.id.facebook_share:
            FacebookShare.setUrl(articleUrl);
            startActivity(new Intent(this, FacebookShare.class));
            return true;
        /* Twitter Integration Goes Here
        case R.id.twitter_share:
            startActivity(new Intent(this, TwitterShare.class));
            return true;
        END TWITTER INTEGRATION*/
        case R.id.email_share:
            Intent emailIntent = new Intent(android.content.Intent.
                ACTION_SEND);
            emailIntent.setType("plain/text");
            emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, "The
                Daily 49er: " + articleTitle);
            emailIntent.putExtra(android.content.Intent.EXTRA_TEXT, "\"" +
                articleTitle + "\"" +
                " by " + articleAuthor + ": " + articleUrl);
            //Create a launcher to select which e-mail app to use in the
            case that there is more than one installed
            startActivity(Intent.createChooser(emailIntent, "Share your
                article through:"));
            return true;
        }
    }
    return false;
} //end onOptionsItemSelected()

```



```
/**
 * Parse, format, and display title, category, publication date, and
 * content of the article.
 */
private void loadFeed()
{
    try
    {
        String castLong = Long.toString(articlePosition);
        int index = Integer.parseInt(castLong);

        articleBody = newsTab.messageList.get(index).getDescription();
        articleTitle = newsTab.messageList.get(index).getTitle();
        articleUrl = newsTab.messageList.get(index).getLink().toString();
        articleAuthor = newsTab.messageList.get(index).getAuthor();
        articleDate = newsTab.messageList.get(index).getDate();
        articleCategory = newsTab.messageList.get(index).getCategory();

        int dotDotDot = articleBody.indexOf("...");
        articleBody = articleBody.replaceAll("&nbsp;", "");
        int disclaimer = articleBody.indexOf("Disclaimer:");
        articleBody = "\n\n\t" + articleBody.substring(dotDotDot + 3,
            disclaimer).trim();

        Message newMessage = new Message();
        List<Message> titles = new ArrayList<Message>(newsTab.messageList.
            size());
        newMessage.setTitle(articleTitle);
        newMessage.setAuthor(articleAuthor);
        newMessage.setCategory(articleCategory);
        newMessage.setDescription(articleBody);
        newMessage.setDate(articleDate);
        titles.add(newMessage);
        ArrayAdapter<Message> adapter = new CustomAdapter(this, R.layout.
            article_page,titles);
        this.setListAdapter(adapter);

    }catch(Throwable t)
    {
        Log.e("AndroidNews",t.getMessage(),t);
    } //end try-catch
} //end loadFeed()
} //end Article
```

```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: BaseFeedParser.java */

package daily49er.android.app;

import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;

public abstract class BaseFeedParser implements FeedParser {
    // names of the XML tags
    static final String CHANNEL = "channel";
    static final String PUB_DATE = "pubDate";
    static final String DESCRIPTION = "description";
    static final String LINK = "link";
    static final String TITLE = "title";
    static final String ITEM = "item";
    static final String AUTHOR = "author";
    static final String CATEGORY = "category";

    private final URL feedUrl;

    protected BaseFeedParser(String feedUrl){
        try {
            //set the URL link by the parameter string.
            this.feedUrl = new URL(feedUrl);
        } catch (MalformedURLException e) {
            throw new RuntimeException(e);
        }
    }

    protected InputStream getInputStream() {
        try {
            return feedUrl.openConnection().getInputStream();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```



```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: ContactTab.java */

package daily49er.android.app;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Toast;

public class ContactTab extends Activity implements OnClickListener{
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        this setContentView(R.layout.contact_tab);
        View videoButton = findViewById(R.id.contactButton);
        videoButton.setOnClickListener(this);
    }

    public void onClick(View view)
    {
        String aEmailList[] = {"onlined49er@gmail.com"};
        Intent emailIntent = new Intent(android.content.Intent.ACTION_SEND);
        emailIntent.setType("plain/text");
        emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL, aEmailList);
        emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, "The Daily
            49er App Issues/Comments");
        //Create a launcher to select which e-mail app to use in the case that
        there is more than one installed
        startActivity(Intent.createChooser(emailIntent, "Share your article
            through:"));
    }
}
```



```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: CustomAdapter.java */

package daily49er.android.app;

import java.util.List;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.TextView;

public class CustomAdapter extends ArrayAdapter<Message>{
    private List<Message> items;

    public CustomAdapter(Context context, int textViewResourceId, List<Message>
        titles)
    {
        super(context, textViewResourceId, titles);
        this.items = titles;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent)
    {
        View v = convertView;
        if (v == null)
        {
            LayoutInflater vi = (LayoutInflater)getContext().getSystemService
                (Context.LAYOUT_INFLATER_SERVICE);
            v = vi.inflate(R.layout.article_page, null);
        }

        Message o = items.get(position);

        if (o != null)
        {
            TextView tt = (TextView) v.findViewById(R.id.titleview);
            TextView mt = (TextView) v.findViewById(R.id.authorview);
            TextView bt = (TextView) v.findViewById(R.id.categoryview);
            TextView at = (TextView) v.findViewById(R.id.articleview);
            TextView dt = (TextView) v.findViewById(R.id.authorview);
        }
    }
}
```

```
        if (tt != null)
        {
            tt.setText(o.getTitle());
        }
        if(mt != null)
        {
            mt.setText(o.getAuthor());
        }
        if(dt != null)
        {
            dt.setText(o.getDate() + "\n");
        }
        if(bt != null)
        {
            bt.setText("Category: " + o.getCategory() + "\n");
        }
        if(at != null){
            at.setText("\t" + o.getDescription());
        }
    }

    return v;
}

}
```

```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: Daily49er.java */

package daily49er.android.app;

import android.os.Bundle;
import android.app.TabActivity;
import android.content.Intent;
import android.content.res.Resources;
import android.widget.TabHost;

public class Daily49er extends TabActivity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        //Create a new intent to launch the splash screen
        /*Intent splashIntent = new Intent();
        splashIntent.setClassName("daily49er.android.app",
            "daily49er.android.app.SplashScreen");
        startActivity(splashIntent);*/

        //Continue on to create the main instance
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Resources res = getResources(); // Resource object to get Drawables
        TabHost tabHost = getTabHost(); // The activity TabHost
        TabHost.TabSpec spec; // Reusable TabSpec for each tab
        Intent intent; // Reusable Intent for each tab

        // Create an Intent to launch an Activity for the tab (to be reused)
        intent = new Intent().setClass(this, NewsTab.class);

        // Initialize a TabSpec for each tab and add it to the TabHost
        spec = tabHost.newTabSpec("news").setIndicator("News",
            res.getDrawable(R.drawable.ic_tab_home)).setContent(intent);
        tabHost.addTab(spec);

        // Create an Intent to launch an Activity for the tab (to be reused)
        intent = new Intent().setClass(this, MediaTab.class);
```



```
// Initialize a TabSpec for each tab and add it to the TabHost
spec = tabHost.newTabSpec("media").setIndicator("Media",
    res.getDrawable(R.drawable.ic_tab_media)).setContent(intent);
tabHost.addTab(spec);

// Create an Intent to launch an Activity for the tab (to be reused)
intent = new Intent().setClass(this, EventsTab.class);

// Initialize a TabSpec for each tab and add it to the TabHost
spec = tabHost.newTabSpec("events").setIndicator("Events",
    res.getDrawable(R.drawable.ic_tab_events))
    .setContent(intent);
tabHost.addTab(spec);

// Create an Intent to launch an Activity for the tab (to be reused)
intent = new Intent().setClass(this, ContactTab.class);

// Initialize a TabSpec for each tab and add it to the TabHost
spec = tabHost.newTabSpec("Contact").setIndicator("Contact",
    res.getDrawable(R.drawable.ic_tab_contact)).setContent(intent);
tabHost.addTab(spec);
```

```
}
}
```

```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: EventsTab.java */

package daily49er.android.app;

import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.webkit.WebView;

/**
 * The Events Tab is composed of the Daily49er's "Diversions" calendar which is
 * implemented via an embedded WebView.
 * @author Alex Chavez
 */
public class EventsTab extends Activity
{
    //Create a new WebView object
    WebView calendarView;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        calendarView = new WebView(this);
        calendarView.getSettings().setJavaScriptEnabled(true);
        String url = "https://www.google.com/calendar/embed?showTitle=0&
            showPrint=0&showTabs=0&showCalendars=0" +
            "&showTz=0&mode=AGENDA&height=600&wkst=2&bgcolor=%23FFFFFF" +
            //Error on this calendar:
            //"&src=diversionsd49er%40gmail.com&color=%235229A3" +
            "&src=calendard49er%40gmail.com&color=%23528800" +
            "&src=ta3lad7oton86pe1a5u9l2eq8k%40group.calendar.google.com&
            color=%230D7813" +
            "&src=d6i4qlrmtkgdpffmi1370d0to8%40group.calendar.google.com&
            color=%2328754E" +
            "&src=236k2n013kavc6bh61nalpsp5g%40group.calendar.google.com&
            color=%23B1440E" +
            "&src=luoc3u2led2otfvp65sb02rktk%40group.calendar.google.com&
            color=%234A716C" +
            "&src=q80db59fuub8tkdd330ls7rvpo%40group.calendar.google.com&
            color=%23A32929" +
    }
```

```

        "&src=ol9i0lbm6iolpki1mrar2tpt08%40group.calendar.google.com&
        color=%231B887A" +
        "&src=uaf7e8mhimsqchp6toij9g5s%40group.calendar.google.com&
        color=%23AB8B00" +
        "&src=afuqbd3vbkbj86nov7k0e6t3p4%40group.calendar.google.com&
        color=%23705770" +
        "&src=1b0grqpd47ij9r0fkd90k9ktno%40group.calendar.google.com&
        color=%232952A3" +
        "&src=g9qta3g7n4imnf05dmv6eqenm0%40group.calendar.google.com&
        color=%23B1365F" +
        "&src=l6ps3eqn0620lerpemcvtr7ii4%40group.calendar.google.com&
        color=%238D6F47" +
        //Error on this calendar:
        //"&src=tslm8ob4srqmcepnvukmjlkeo%40group.calendar.google.com&
        color=%235A6986" +
        "&src=vyabooking%40gmail.com&color=%232952A3&ctz=America%2FLos_
        Angeles";

        calendarView.setVerticalScrollBarEnabled(true);
        calendarView.setWebViewClient(new LoadPage(calendarView, url));
        setContentView(calendarView);
    }

    /**
     * This callback method will be called anytime a button is pressed while in
     * the Events tab
     * WebView; it will go back to a previous page in the webview.
     * @param keyCode - the key code that is passed whenever any of the
     * standard device buttons are
     * pressed.
     * @param keyEvent - the event that is triggered by the button press.
     * @return true - if the "back" button has been pressed; then return the
     * event and keycode for
     * the button that was pressed.
     */
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        if((keyCode == KeyEvent.KEYCODE_BACK) && calendarView.canGoBack())
        {
            calendarView.goBack();
            return true;
        }
        return super.onKeyDown(keyCode, event);
    }
}

```

```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: FacebookShare.java */

package daily49er.android.app;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.os.Bundle;
import android.view.KeyEvent;
import android.widget.Toast;

import com.facebook.android.*;
import com.facebook.android.Facebook.*;

/**
 * Implementation for posting an article to the user's Facebook "wall".
 * In order to make this class work, you'll need to create a new project that
 * contains facebook SDK files
 * and reference it to this project.
 * @author Alex Chavez
 */
public class FacebookShare extends Activity
{
    private static final String APP_ID = "152474111482866";
    private static final String TOKEN = "access_token";
    private static final String EXPIRES = "expires_in";
    private static final String KEY = "facebook-credentials";
    private static final String[] PERMISSIONS = new String[] {"publish_stream"}
    ;
    private static String articleUrl;

    //Declare a new facebook object
    Facebook facebook;

    /**
     * The URL is set in Article.java as it is the current article being
     * viewed.
     * @param url - the URL that will be posted in the user's Facebook wall.
     */
    public static void setUrl(String url)
```

```
{
    articleUrl = url;
}

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    //setContentView(R.layout.main);
    facebook = new Facebook(APP_ID);
    //Restore the previously stored credentials, if any, from the
        SharedPreferences of the app
    restoreCredentials(facebook);
    updateStatus();
}

/**
 * Creates and displays a Facebook dialog with the URL set; the dialog
 * gives
 * the user the option to enter a message into the text field.
 */
public void updateStatus()
{
    saveCredentials(facebook);
    Bundle bundle = new Bundle();
    bundle.putString("link", articleUrl);
    facebook.dialog(this, "stream.publish", bundle, new
        WallPostDialogListener());
}

/**
 * No clue what this does, it's used by the Facebook API.
 */
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data)
{
    super.onActivityResult(requestCode, resultCode, data);
    facebook.authorizeCallback(requestCode, resultCode, data);
}

/**
 * A custom listener for all wall post dialogues created and displayed to
 * the user.
 * @author Steffen Luypaert (http://integratingstuff.com/category/web-services/facebook/)
 */
class WallPostDialogListener implements DialogListener
{
    /**
     * If there is a post ID (meaning a post was successfully posted to the
     * user's facebook wall), then display
     * a toast message with confirmation; else state otherwise.
     * @param values - any values set for the dialogue; consists of the
     * user's entered message (may be null)
     * and the article URL.
     */
}
```

```
public void onComplete(Bundle values)
{
    final String postId = values.getString("post_id");
    if (postId != null) showToast("Message posted to your facebook wall!");
    else showToast("Wall post cancelled.");
    finish();
}

/**
 * Display the user a toast message that that something went wrong with
 * the
 * Facebook API and end the FacebookShare activity.
 * @param e - Error given by the Facebook API
 */
public void onFacebookError(FacebookError e)
{
    showToast("Failed to post to wall!");
    finish();
}

/**
 * Display the user a toast message that the posting of a dialogue has
 * failed
 * and end the FacebookShare activity.
 * @param e - Error given by the dialogue
 */
public void onError(DialogError e)
{
    showToast("Failed to post to wall!");
    finish();
}

/**
 * Display the user a toast message that the posting of a dialogue has
 * registered
 * the dialogue cancellation and end the FacebookShare activity.
 */
public void onCancel()
{
    showToast("Wall post cancelled!");
    finish();
}
} //end class WallPostDialogListener()

/**
 * A custom listener for all events related to facebook authentication.
 * @author Steffen Luypaert (http://integratingstuff.com/category/web-services/facebook/)
 */
class LoginDialogListener implements DialogListener
{
    /**
     * Saves the user's authenticated Facebook credentials for later use
     * and posts the
     * article URL to the user's Facebook "wall".
     * @param values - any values that have been set (at this point, there
```

```
        aren't any set)
    */
    public void onComplete(Bundle values)
    {
        saveCredentials(facebook);
        updateStatus();
    }

    /**
     * Display the user a toast message that authentication has failed and
     * end the
     * FacebookShare activity.
     * @param error - Error given by the Facebook API
     */
    public void onFacebookError(FacebookError error)
    {
        showToast("Authentication with Facebook failed!");
        finish();
    }

    /**
     * Display the user a toast message that authentication has failed and
     * end the
     * FacebookShare activity.
     * @param error - Error given by the facebook dialogue.
     */
    public void onError(DialogError error)
    {
        showToast("Authentication with Facebook failed!");
        finish();
    }

    /**
     * Display the user a toast message that authentication has been
     * cancelled by
     * the user of Facebook API and end the FacebookShare activity.
     * @param error - Error given by the cancellation of an authentication
     * session
     */
    public void onCancel()
    {
        showToast("Authentication with Facebook cancelled!");
        finish();
    }
} //end class LoginDialogListener()

/**
 * Display a custom toast message (a popup message) to the user.
 * @param message - a message that the toast message will consist of.
 */
private void showToast(String message)
{
    Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT).
        show();
}

/**
```

```
* Saves the user's authenticated Facebook login credentials.
* @param facebook - an instantiated facebook object
* @return the saved credentials
*/
public boolean saveCredentials(Facebook facebook)
{
    Editor editor = getApplicationContext().getSharedPreferences(KEY,
        Context.MODE_PRIVATE).edit();
    editor.putString(TOKEN, facebook.getAccessToken());
    editor.putLong(EXPIRES, facebook.getAccessExpires());
    return editor.commit();
}

/**
 * Retrieve and restore the user's Facebook login credentials from the app
 * 's shared preferences.
 * @param facebook - an instantiated facebook object
 * @return a boolean indicating if the stored login credentials are valid
 */
public boolean restoreCredentials(Facebook facebook)
{
    SharedPreferences sharedPreferences = getApplicationContext().
        getSharedPreferences(KEY, Context.MODE_PRIVATE);
    facebook.setAccessToken(sharedPreferences.getString(TOKEN, null));
    facebook.setAccessExpires(sharedPreferences.getLong(EXPIRES, 0));
    return facebook.isSessionValid();
}
} //end FacebookShare class
```



```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: FeedParser.java */

package daily49er.android.app;

import java.util.List;

public interface FeedParser
{
    List<Message> parse();
}
```



```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: FeedParserFactory.java */

package daily49er.android.app;

public class FeedParserFactory
{
    static String feedUrl = "http://www.daily49er.com/se/daily-49er-rss-1.111?
        detailRSS=true";

    public static FeedParser getParser()
    {
        return new AndroidSaxFeedParser(feedUrl);
    }
}
```



```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: LoadPage.java */

package daily49er.android.app;

import android.webkit.WebViewClient;
import android.webkit.WebView;

/**
 * The URL for a given WebView is loaded here.
 * @author Alex Chavez
 */
public class LoadPage extends WebViewClient
{
    /**
     * Loads a web page by it's URL by overriding a URL loading instance in
     * order to keep
     * the WebView within the constraints of our project and prevent it from
     * opening a
     * new web browser window.
     * @param pageView - the WebView we're using to launch a new WebView intent
     * with a
     * given URL.
     * @param url - the URL for a web page we want to open.
     */
    LoadPage(WebView pageView, String url)
    {
        shouldOverrideUrlLoading(pageView, url);
    }

    /**
     * Gives the application control of the WebView intent to stay in the
     * application and
     * not open a seperate page.
     * @param view - the WebView object that will load a web page.
     * @param theUrl - the URL of the web page to be loaded.
     */
    @Override
    public boolean shouldOverrideUrlLoading(WebView pageView, String theUrl)
    {
        pageView.loadUrl(theUrl);
        return true;
    }
}
```

}

```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: MediaTab.java */

package daily49er.android.app;

import java.io.IOException;
import android.app.Activity;
import android.app.ListActivity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.webkit.WebView;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;

/**
 * The media tab contains two buttons selection, "Video" and "Podcast".
 * When a button is clicked, it redirects the user to the phone's default media
 * viewer application
 * for the user to view the respective content.
 */
public class MediaTab extends Activity implements OnClickListener
{
    //almost done, just have to change UI to make it more interesting.
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        this setContentView(R.layout.media);

        //set up click listeners for all the buttons
        View videoButton = findViewById(R.id.videoButton);
        videoButton.setOnClickListener(this);
        View podcastButton = findViewById(R.id.podcastButton);
        podcastButton.setOnClickListener(this);
    }
}
```

```
public void onClick(View view){
    if(view.getId() == R.id.videoButton){
        startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("http://
            www.youtube.com/user/videod49er")));
    }
    else{
        startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.
            youtube.com/user/audiod49er")));
    }
}
}
```



```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: Message.java */

package daily49er.android.app;

import java.net.MalformedURLException;
import java.net.URL;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Message implements Comparable<Message>{
    static SimpleDateFormat FORMATTER =
        new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss Z");
    private String title;
    private URL link;
    private String description;
    private String author;
    private String category;
    private Date date;

    public String getAuthor(){
        return author;
    }

    public void setAuthor(String author){
        this.author = author.trim();
    }

    public String getCategory(){
        return category;
    }

    public void setCategory(String category){
        this.category = category.trim();
    }

    public String getTitle() {
        return title;
    }
}
```

```
public void setTitle(String title) {
    this.title = title.trim();
}
// getters and setters omitted for brevity
public URL getLink() {
    return link;
}

public void setLink(String link) {
    try {
        this.link = new URL(link);
    } catch (MalformedURLException e) {
        throw new RuntimeException(e);
    }
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description.trim().replaceAll("\\<.*?>", "");
}

public String getDate() {
    return FORMATTER.format(this.date);
}

public void setDate(String date) {
    // pad the date if necessary
    while (!date.endsWith("00")){
        date += "0";
    }
    try {
        this.date = FORMATTER.parse(date.trim());
    } catch (ParseException e) {
        throw new RuntimeException(e);
    }
}

@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("Title: ");
    sb.append(title);
    sb.append('\n');
    sb.append("Date: ");
    sb.append(this.getDate());
    sb.append('\n');
    sb.append("Link: ");
    sb.append(link);
    sb.append('\n');
    sb.append("Description: ");
    sb.append(description);
    sb.append('\n');
    sb.append("Author :");
    sb.append(author);
}
```

```
        sb.append('\n');
        sb.append("Paragraph: ");
        sb.append("Category: ");
        sb.append('\n');
        return sb.toString();
    }

    public Message copy(){
        Message copy = new Message();
        copy.title = title;
        copy.link = link;
        copy.description = description;
        copy.date = date;
        copy.author = author;
        copy.category = category;
        return copy;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((date == null) ? 0 : date.hashCode());
        result = prime * result + ((description == null) ? 0 : description.
            hashCode());
        result = prime * result + ((link == null) ? 0 : link.hashCode());
        result = prime * result + ((title == null) ? 0 : title.hashCode());
        result = prime * result + ((author == null) ? 0 : author.hashCode());
        result = prime * result + ((category == null) ? 0 : category.hashCode())
            ;
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Message other = (Message) obj;
        if (date == null) {
            if (other.date != null)
                return false;
        } else if (!date.equals(other.date))
            return false;
        if (description == null) {
            if (other.description != null)
                return false;
        } else if (!description.equals(other.description))
            return false;
        if (link == null) {
            if (other.link != null)
                return false;
        } else if (!link.equals(other.link))
            return false;
    }
```

```
    if (title == null) {
        if (other.title != null)
            return false;
    } else if (!title.equals(other.title))
        return false;
    if(author == null){
        if(other.author != null)
            return false;
    }else if(!author.equals(other.author))
        return false;
    if(category == null){
        if(other.category != null)
            return false;
    }else if(!category.equals(other.category))
        return false;
    return true;
}

public int compareTo(Message another){
    if(another == null) return 1;
    return another.date.compareTo(date);
}
}
```

```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: NewsTab.java */

package daily49er.android.app;

import java.io.StringWriter;
import java.util.ArrayList;
import java.util.List;
import org.xmlpull.v1.XmlSerializer;
import android.app.ListActivity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.util.Xml;
import android.view.View;
import android.webkit.WebView;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

public class NewsTab extends ListActivity
{
    public static List<Message> messageList;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.list);
        getListView().setTextFilterEnabled(true);
        loadFeed();
    }
    /**
     * Parse, format, and print article's title in list view order.
     */
    public void loadFeed()
    {
        try
        {
            String articleTitle;
            FeedParser parser = FeedParserFactory.getParser();
```

```
messageList = parser.parse();
List<String> titles = new ArrayList<String>(messageList.size());
for(Message article : messageList)
{
    articleTitle = article.getTitle() + "\nCategory: " + article.
        getCategory();
    titles.add(articleTitle);
}

ArrayAdapter<String> adapter =
    new ArrayAdapter<String>(this, R.layout.row, titles);

    this.setListAdapter(adapter);
}
catch (Throwable t)
{
    Log.e("Daily49er",t.getMessage(),t);
}
}

/*
 * When an item on the list is clicked, it saved the position ID of the
 * item to be used later for
 * viewing the article.
 * @see android.app.ListActivity#onListItemClick(android.widget.ListView,
 * android.view.View, int, long)
 */
protected void onListItemClick(ListView l, View v, int position, long id)
{
    super.onListItemClick(l, v, position, id);
    Intent viewMessage = new Intent(this, Article.class);
    viewMessage.putExtra("position", id);
    this.startActivity(viewMessage);
}
}
```

```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: RssHandler.java */

package daily49er.android.app;

import java.util.ArrayList;

import java.util.List;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

import static daily49er.android.app.BaseFeedParser.*;
public class RssHandler extends DefaultHandler{
    private List<Message> messages;
    private Message currentMessage;
    private StringBuilder builder;

    public List<Message> getMessages(){
        return this.messages;
    }

    @Override
    public void characters(char[] ch, int start, int length)
        throws SAXException{
        super.characters(ch, start, length);
        builder.append(ch,start,length);
    }

    public void endElement(String uri, String localName, String name)
        throws SAXException{
        super.endElement(uri, localName, name);
        if(this.currentMessage != null){
            if (localName.equalsIgnoreCase(TITLE)){
                currentMessage.setTitle(builder.toString());
            } else if (localName.equalsIgnoreCase(LINK)){
                currentMessage.setLink(builder.toString());
            } else if (localName.equalsIgnoreCase(DESCRIPTION)){
                currentMessage.setDescription(builder.toString());
            } else if (localName.equalsIgnoreCase(PUB_DATE)){
                currentMessage.setDate(builder.toString());
            } else if (localName.equalsIgnoreCase(AUTHOR)){

```

```
        currentMessage.setAuthor(builder.toString());
    }else if (localName.equalsIgnoreCase(CATEGORY)){
        currentMessage.setCategory(builder.toString());
    } else if (localName.equalsIgnoreCase(ITEM)){
        messages.add(currentMessage);
    }
    builder.setLength(0);
}
}

@Override
public void startDocument() throws SAXException {
    super.startDocument();
    messages = new ArrayList<Message>();
    builder = new StringBuilder();
}

@Override
public void startElement(String uri, String localName, String name,
    Attributes attributes) throws SAXException {
    super.startElement(uri, localName, name, attributes);
    if (localName.equalsIgnoreCase(ITEM)){
        this.currentMessage = new Message();
    }
}
}
```



```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: SaxFeedParser.java */

package daily49er.android.app;

import java.util.List;

import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

public class SaxFeedParser extends BaseFeedParser {
    protected SaxFeedParser(String feedUrl){
        super(feedUrl);
    }

    public List<Message> parse() {
        SAXParserFactory factory = SAXParserFactory.newInstance();
        try {
            SAXParser parser = factory.newSAXParser();
            RssHandler handler = new RssHandler();
            parser.parse(this.getInputStream(), handler);
            return handler.getMessages();
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```



```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: SettingsTab.java */

package daily49er.android.app;

import android.content.Context;
import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.preference.PreferenceManager;

/**
 * The 'Settings' tab for the application.
 * @author Alex Chavez
 */
public class SettingsTab extends PreferenceActivity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        //Reads the settings definition from XML and inflates it
        //into views in the current activity.
        //addPreferencesFromResource(R.xml.preferences);
    } //end onCreate()
} //end SettingsTab class
```



```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: SplashScreen.java */

package daily49er.android.app;

import android.app.Activity;
import android.os.Bundle;

/**
 * This class creates a new thread for running the splash screen on the start
 * of the application.
 * @author Alex Chavez
 */
public class SplashScreen extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.splash);
        Thread splashThread = new Thread()
        {

            /**
             * Run the splash screen for a total of 3 seconds. We should improve
             * this
             * to stay in the splash screen while loading/parsing the RSS feed.
             */
            @Override
            public void run()
            {
                try
                {
                    int waited = 0;
                    while (waited < 3000)
                    {
                        sleep(100);
                        waited += 100;
                    }
                }
                catch (InterruptedException e)
                {
                }
            }
        };
    }
}
```

```
        //Do nothing
    }
    finally
    {
        finish();
    }
}
};
    splashThread.start();
} //end onCreate()
} //end SplashScreen class
```

```
/*
 * Copyright 2011 Alex Chavez, Punravee Cherngchaosil, Amanda Nguyen
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/** Filename: TwitterShare.java */

package daily49er.android.app;

/*import android.app.Activity;
import android.os.Bundle;
import android.widget.Button;
import winterwell.jtwitter.*;
import oauth.signpost.*;*/

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;

public class TwitterShare extends Activity
{
    private final String JTWITTER_OAUTH_KEY = "";
    private final String JTWITTER_OAUTH_SECRET = "";

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        /* Create a new HTTP-RequestQueue. */
        // android.net.http.RequestQueue rQueue = new RequestQueue(this);

        /* Prepare the Post-Text we are going to send. */
    } //end onCreate()
}
```



```
<!-- ic_tab_contact.xml -->
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
  <!-- When tab is selected-->
  <item android:drawable="@drawable/happy"
        android:state_selected="true" />
  <!-- When tab is not selected-->
  <item android:drawable="@drawable/happy"
        android:state_selected="false" />
</selector>
```



```
<!-- ic_tab_events.xml -->
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
  <!-- When tab is selected-->
  <item android:drawable="@drawable/tiles"
        android:state_selected="true" />
  <!-- When tab is not selected-->
  <item android:drawable="@drawable/tiles"
        android:state_selected="false" />
</selector>
```



```
<!-- ic_tab_home.xml -->
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
  <!-- When tab is selected-->
  <item android:drawable="@drawable/globe"
        android:state_selected="true" />
  <!-- When tab is not selected-->
  <item android:drawable="@drawable/globe"
        android:state_selected="false" />
</selector>
```



```
<!-- ic_tab_media.xml -->
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
  <!-- When tab is selected-->
  <item android:drawable="@drawable/movie"
        android:state_selected="true" />
  <!-- When tab is not selected-->
  <item android:drawable="@drawable/movie"
        android:state_selected="false" />
</selector>
```



```
<!-- article_page.xml -->

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

<TextView android:id="@+id/titleview"
    android:textSize="16sp"
    android:textStyle="bold"
    android:textColor="@color/yellow"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

<TextView android:id="@+id/authorview"
    android:textSize="10sp"
    android:textStyle="italic"
    android:textColor="#ffcc00"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

<TextView android:id="@+id/categoryview"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="@color/yellow"/>

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/articleview"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:gravity="center_vertical"
    android:paddingLeft="6dip"
    android:minHeight="?android:attr/listPreferredItemHeight"
    android:textColor="@color/yellow"/>

</LinearLayout>
```



```
<!-- contact_tab.xml -->

<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget30"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <TextView
        android:id="@+id/widget34"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="24sp"
        android:text="About"
        android:layout_x="12px"
        android:layout_y="11px"
    >
    </TextView>
    <TextView
        android:id="@+id/widget35"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="\n\tThe purpose of the application is to create a fast and
            functional
            companion application for the CSULB's newspaper the Daily 49er on mobile
            devices.
            The intent of having a mobile application is to allow for news articles to be
            easily
            accessible and readable in a user-friendly way. In addition to news articles,
            subcategories will also be available that consist of: videos, podcasts, local
            events,
            social networking features, and contact information. "
        android:layout_x="16px"
        android:layout_y="53px"
    >
    </TextView>

    <Button
        android:id="@+id/contactButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:text="Contact Us"
        android:layout_x="10px"
        android:layout_y="350px"
    >
    </Button>
</AbsoluteLayout>
```



```
<!-- list.xml -->

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding = "5dp" >

    <ListView android:id="@+id/android:list"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        ></ListView>

</LinearLayout>
```



```
<!-- main.xml -->

<?xml version="1.0" encoding="utf-8"?>
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/tabhost"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:padding="5dp">
        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="5dp" />
    </LinearLayout>
</TabHost>
```



```
<!-- media.xml -->
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:background="@drawable/media_bg">

    <Button android:id="@+id/videoButton" android:layout_height="wrap_content"
        android:layout_width="144px" android:text="Videos" android:layout_x="75dip"
        android:layout_y="90dip"></Button>

    <Button android:id="@+id/podcastButton"
        android:layout_height="wrap_content"
        android:layout_width="144px" android:text="Podcasts"
        android:layout_x="180dip"
        android:layout_y="90dip"></Button>

</AbsoluteLayout>
```



```
<!-- menu_button.xml -->
<?xml version="1.0" encoding="utf-8"?>
  <menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/facebook_share" android:title="@string/
      facebook_label" android:alphabeticShortcut="@string/
      facebook_shortcut" />
    <!-- <item android:id="@+id/twitter_share" android:title="@string/
      twitter_label" android:alphabeticShortcut="@string/
      twitter_shortcut" /> -->
    <item android:id="@+id/email_share" android:title="@string/email_label"
      android:alphabeticShortcut="@string/email_shortcut" />
  </menu>
```



```
<!-- row.xml -->

<?xml version="1.0" encoding="utf-8"?>

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/TextView01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:gravity="center_vertical"
    android:paddingLeft="6dip"
    android:minHeight="?android:attr/listPreferredItemHeight"
    android:textColor = "@color/yellow"/>
```



```
<!-- splash.xml -->
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <ImageView android:src="@drawable/splash_screen"
        android:id="@+id/ImageView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </ImageView>
</LinearLayout>
```



```
<!-- twitter.xml -->

<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/ScrollView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:scrollbars="vertical">
    <LinearLayout
        android:layout_width="fill_parent"
        android:orientation="vertical"
        android:layout_height="fill_parent">

        <TextView
            android:id="@+id/TwitterTitle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/twittertitle"
            android:textSize="10pt">
        </TextView>

        <EditText
            android:id="@+id/TwitterLoginID"
            android:layout_height="wrap_content"
            android:hint="@string/twitterID"
            android:layout_width="fill_parent">
        </EditText>

        <EditText
            android:id="@+id/TwitterPassword"
            android:layout_height="wrap_content"
            android:hint="@string/twitterPW"
            android:layout_width="fill_parent">
        </EditText>

        <Button
            android:id="@+id/TwitterLoginButton"
            android:layout_height="wrap_content"
            android:text="@string/twitterlogin"
            android:onClick="sendFeedback"
            android:layout_width="fill_parent">
        </Button>

    </LinearLayout>
</ScrollView>
```



```
<!-- strings.xml -->

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">The Daily 49er</string>
    <color name = "yellow">#dddbb</color>

    <!--BEGIN MENU BUTTON STRINGS -->
    <string name="facebook_label">Facebook</string>
    <string name="facebook_title">Facebook sharing</string>
    <string name="facebook_shortcut">fb</string>

    <!--<string name="twitter_label">Twitter</string>
    <string name="twitter_title">Twitter sharing</string>
    <string name="twitter_shortcut">tw</string> -->

    <string name="email_label">E-mail</string>
    <string name="email_title">Email sharing</string>
    <string name="email_shortcut">em</string>
    <!--END MENU BUTTON STRINGS -->

    <string name ="twittertitle">Please log in to Twitter:</string>
    <string name ="twitterID">Username</string>
    <string name ="twitterPW">Password</string>
    <string name ="twitterlogin">Log In</string>
</resources>
```



```
<!-- arrays.xml -->
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="list_array">
        <item>Small</item>
        <item>Normal</item>
        <item>Large</item>
    </string-array>

    <string-array name="list_values">
        <item>smallText</item>
        <item>normalText</item>
        <item>largeText</item>
    </string-array>
</resources>
```


Test Case Scenario	Type of Test Case	Pre-conditions	Test Steps	Expected Result	Actual Result	Pass/Fail
Launch the application on the user's device	GUI	Application must be installed	1. Locate Daily 49er application icon on the application launcher 2. Tap the application icon to launch the application	The application will launch and the list of news articles will be loaded and populated in less than 1 minute. The application should be at the "News" tab by default.	Launched in 12 seconds. News tab already selected.	Pass
Verify that the most current articles are loaded	Functional	1. The RSS feed must be online and available at the Daily49er's website. 2. Must have an active Wi-Fi or wireless modem connection	1. Launch the application or select the "News" tab.	The application will load the most current 30 articles from the RSS feed.	Most recent article was dated on May 18, 2011. 17:26:44	Pass
Switch between the four different tabs (News, Media, Events, Settings)	GUI	None	1. Tap each tab in this order: Media, Events, Settings and News. 2. Make sure each tab is highlighted and it's respective content is shown.	The selected tab will be highlighted and the content for that tab will load.	All tabs were selected and shown appropriately	Pass
Share an article through Facebook	Functional	1. Must be logged in through Android Facebook Application or login through a login dialogue. 2. Must be currently viewing an article and have pressed the "Facebook" button under the context menu.	1. Select an article. 2. Select a menu button and click on the Facebook button.	The article will be posted on the user's Facebook "wall"	Sign in was required to post wall post. Successfully posted.	Pass
Share an article through	Functional	1. Must be logged in through Android	1. Select an article. 2. Select a menu button and click on	The URL and the article's title will be	Did not seen an option to post thru twitter	Fail

Twitter		Twitter Application or login through a login dialogue. 2. Must be currently viewing an article and have pressed the "Twitter" button under the context menu.	the Twitter button.	"tweeted" on the user's Twitter.		
Share an article through E-mail.	Functional	1. Must have an e-mail application installed on the device. 2. Must have at least one e-mail application configured.	1. Select an article. 2. Select a menu button and click on the E-mail button. 3. If only one e-mail application is installed, then that e-mail application will be launched. Else, if there is more than one e-mail application installed then the user will presented with an application launcher for the user to select which e-mail application to use and also have the option to select that e-mail application to use as the default e-mail application for future emailing of the article. Then, the e-mail's subject line will be filled with the title of the article and the body of the e-mail filled with the article's title, authors, and URL.	The article will be sent via the user's e-mail to any entered recipients.	Emailed myself.	Pass
Verify that the video and podcast	GUI	1. Must have the official Android Youtube application	1. Select the media tab. 2. Click on the "Video" button. 3. If only official Youtube application is	The application that is selected to handle Youtube video should	Video and podcast buttons refer me to youtube pages.	Pass

buttons work		installed or have a web browser with flash installed.	installed, then that application will be launched. Else, if there is more than one application that can view Youtube video installed then the user will presented with an application launcher for the user to select which application to use and also have the option to select that application to use as the default application for future viewing of the Youtube video. 4. Browse the video listing. 5. Press the “back” button on the device to return to the media tab. 6. Repeat step 3, 4, and 5 for the “podcast” button.	be launched.		
Verify that the event calendar works	Functional and GUI	1. Must have an active Wi-Fi or wireless modem connection	1. Select the event tab.	The Daily 49er’s diversion Google calendar launches.	Event Calender is operational. Buttons work as expected.	

Comments:

1. Holding the finger over an article under the “news” tab highlights the article but nothing happens. Usually, a popup screen with sharing options appears.

-Joshua Liong

Test Case Scenario	Type of Test Case	Test Steps	Expected Result	Actual Result	Pass/Fail
Launch the application on the user's device	GUI	1. Locate Daily 49er application icon on the application launcher 2. Tap the application icon to launch the application	The application will launch and the list of news articles will be loaded and populated in less than 1 minute. The application should be at the "News" tab by default.		Pass
Verify that the most current articles are loaded	Functional	1. Launch the application or select the "News" tab.	The application will load the most current 30 articles from the RSS feed.		Pass
Switch between the four different tabs (News, Media, Events, Settings)	GUI	1. Tap each tab in this order: Media, Events, Settings and News. 2. Make sure each tab is highlighted and it's respective content is shown.	The selected tab will be highlighted and the content for that tab will load.	No settings	fail
Share an article through Facebook	GUI	1. Select the desire font size from the setting tab. 2. Switch back to the news tab.	Changes in font size according to the selected setting.	No settings	fail
Share an article through Twitter	Functional	1. Select an article. 2. Select a menu button and click on the Facebook button.	The article will be posted on the user's Facebook "wall"		Pass

Share an article through E-mail.	Functional	1. Select an article. 2. Select a menu button and click on the Twitter button.	The URL and the article's title will be "tweeted" on the user's Twitter.	Pass
Verify that the video and podcast buttons work	Functional	1. Select an article. 2. Select a menu button and click on the E-mail button. 3. If only one e-mail application is installed, then that e-mail application will be launched. Else, if there is more than one e-mail application installed then the user will presented with an application launcher for the user to select which e-mail application to use and also have the option to select that e-mail application to use as the default e-mail application for future emailing of the article. Then, the e-mail's subject line will be filled with the title of the article and the body of the e-mail filled with the article's title, authors, and URL.	The article will be sent via the user's e-mail to any entered recipients.	Pass
Verify that the event calendar works	GUI	1. Select the media tab. 2. Click on the "Video" button. 3. If only official Youtube application is installed, then that application will be launched. Else, if there is more than one application that can view Youtube video installed then the user will	The application that is selected to handle Youtube video should be launched.	pass

presented with an application launcher for the user to select which application to use and also have the option to select that application to use as the default application for future viewing of the Youtube video.

4. Browse the video listing.

5. Press the “back” button on the device to return to the media tab.

6. Repeat step 3,4, and 5 for the “podcast” button.

1. Select the event tab.

Functional and GUI

The Daily 49er’s
diversion Google
calendar launches.

pass

Comments: