Task 11: Add Tablet layouts

GitHub Username: scarrupt

Backlog Tracker

Description

Backlog Tracker lets you stay on top of issues and focus on priorities. Your app is synced with your online Backlog projects and will be always up to date.

Intended User

This app is intended for people managing their projects on Backlogtool.com.

Features

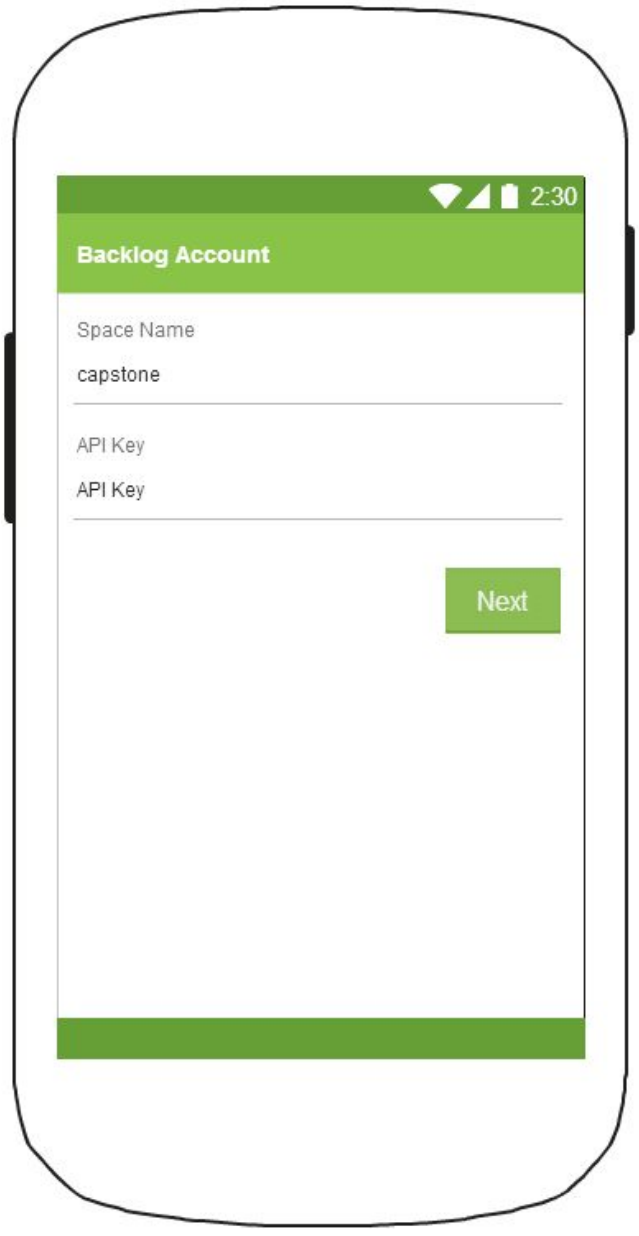
The main features are

- Sync your app with your projects on Backlogtools.com
- Filter issues by open, in progress, and resolved status
- Display the details of an issue
- Browse your projects and issues when offline
- Be informed of the last opened issues

User Interface Mocks

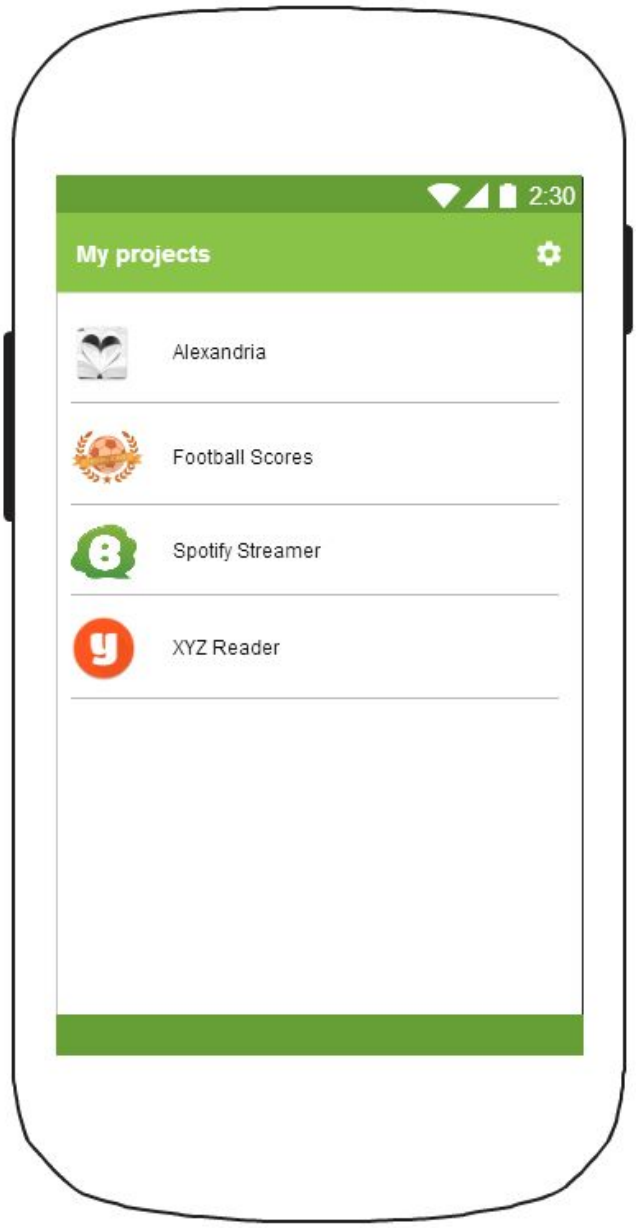
Phone UX

Screen Account



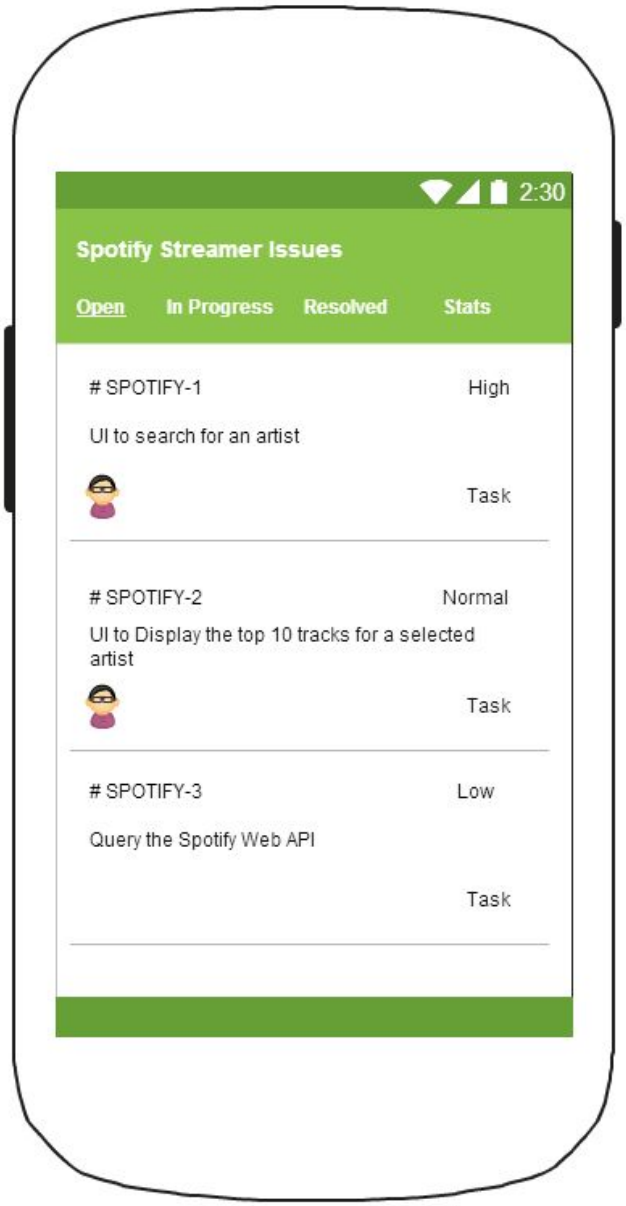
First time users need to enter their Backlog account space name and API key. By clicking on the button the user will navigate to his projects.

Screen My Projects



This activity lists the user projects. For each project, its thumbnail and name is displayed. By clicking on a project, the user navigates to the issues screens for the selected project.

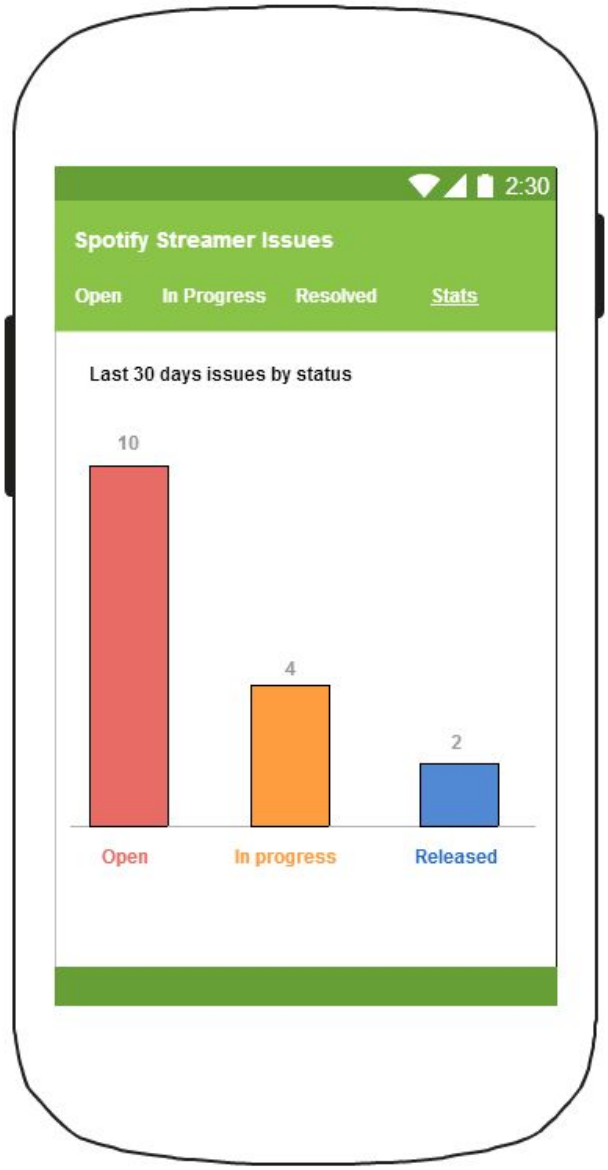
Screen Issues



This screen displays a summary of the issues for the selected project. Issues are splitted between tabs depending on their status (Open, In Progress, Resolved) and sorted by their priority. Each issue summary is composed of:

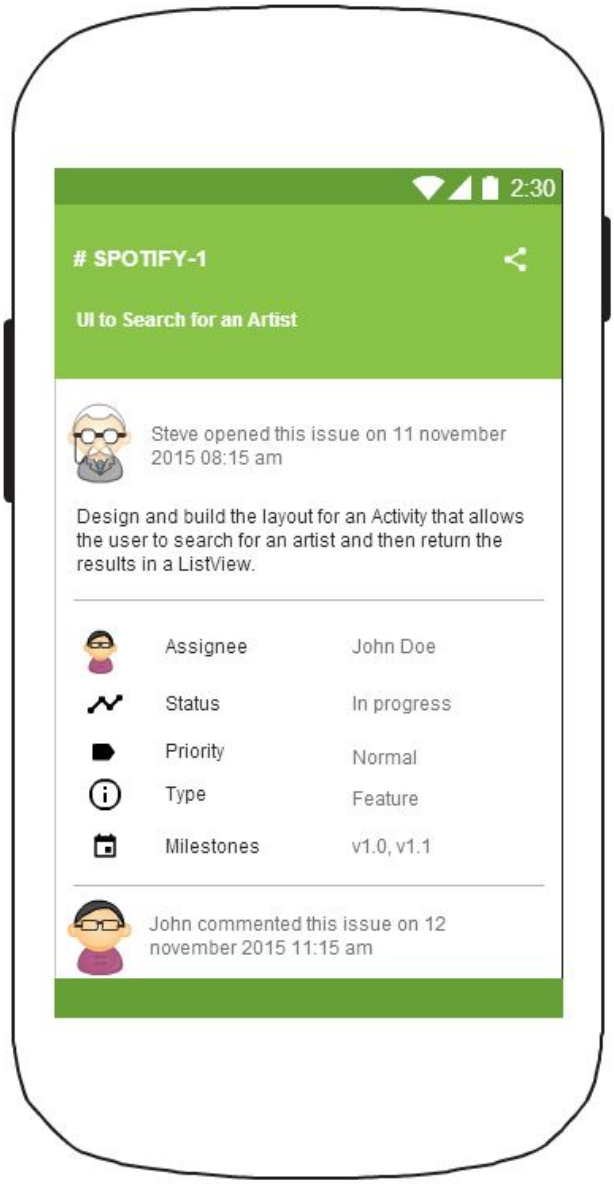
- Issue key
- Issue summary
- Issue priority
- Issue type
- Assignee thumbnail

Screen Stats



This tab shows a bar chart. Each bar represents the number of issues of a certain status (Open, In Progress, Resolved) for the last 30 days.

Screen Issue

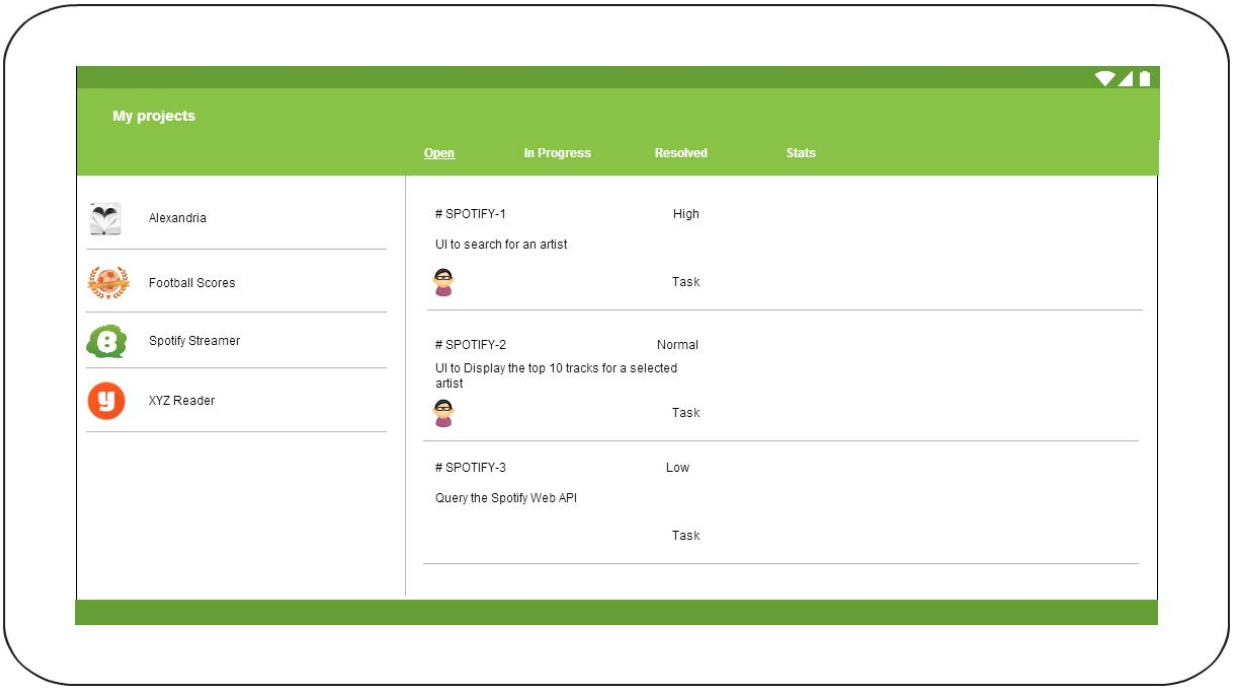


This fragment details the selected issue. The layout is splitted in 3 main parts:

- Author thumbnail and name, issue creation date and description
- Issue properties:
    - assignee name and thumbnail
    - status
    - priority
    - type
    - milestones
- The list of comments. Each comment is composed of:
    - author thumbnail and name, and comment creation date

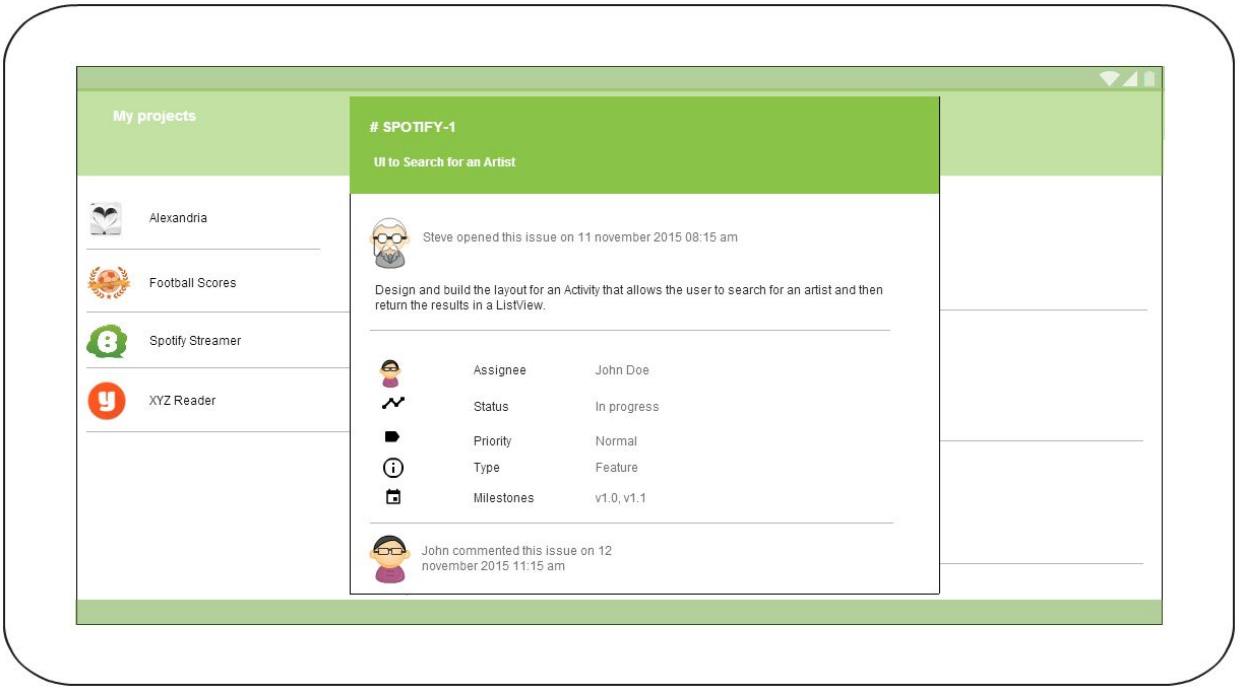
Tablet UX

Screen Main



The main screen is divided in two parts: the project list and the issue list (Master/Detail flow).

Screen Issue



The details of an issue is listed in a Dialog fragment.

Key Considerations

How will your app handle data persistence?

Project data are stored online at backlogtool.com. After user sign in, the data will be synced to the device via a SyncAdapter and by using the Backlog API. When online data are modified or added a notification will be sent to the device via Google Cloud messaging. A webhook is configured on Backlogtool.com to notify Google Cloud Endpoint. Once received by the device, the SyncAdapter will update the local data.

On the device the data are stored at multiple locations:

- Account information will be handle by the AccountManager
- Backlog data will be stored in a SQLite content provider so they can be browsed offline.
- App preferences and configuration properties will be saved in Sharedpreferences.
- Project and user thumbnaisl will be downloaded to the internal storage

Describe any corner cases in the UX.

The main corner cases are:

- The space name should be between 3 and 10 characters long, else an error message is displayed.
- If the space name or/and the API Key are wrong then an error message should be displayed.
- If the user had already entered his space name and API key and has not logged out, the My Project screen should be displayed instead of the Account screen.
- If the device has no network and the user is adding his account, an error message should be displayed

Describe any libraries you'll be using and share your reasoning for including them.

Development

- Android Design Support Library for material design
- Butterknife to bind fields and reduce boilerplate code
- Dagger 2 for dependency injection
- Google Analytics to collect app stats
- Google Cloud Messaging for receiving notifications from App Server
- Glide to handle the loading and caching of images
- Gson for parsing messages in JSON
- Guava to reduce Java boilerplate code.
- Retrofit 2 to communicate with Backlog API
- RxAndroid to asynchronously call Backlog API

Testing

- Espresso for UI testing
- JUnit with Hamcrest for unit testing
- Mockito for mocking method calls
- Robolectric for integration testing
- WireMock for mocking API server

Next Steps: Required Tasks

Task 1: Setup

First task is to create an android project and then create a backlog project that can be used for tests.

Android Project

- Create an android project for phone and tablet with the following package structure
    - gcm
    - injector
    - provider
    - sync
    - widget
    - view
- Add the required libraries
- Add a java library module (backlog-api) for calling Backlog Api endpoints. The package structure is
    - models
    - operations
- Add backlog-api module to the App build.gradle file

Backlog Project

- Create a project on backlogtool.com
    - Go to http://backlogtool.com/
    - Sign up for a new space
    - Choose the Free plan
    - Enter the space and owner information
    - Add a project
    - Add some issues
    - Go to Personal settings
    - Generate a new API Key

For testing purpose, a space will be created on Backlogtool.com and account information will be forwarded to Udacity code reviewer.

Task 2: Create Backlog Api client

The Backlog Api client will be implemented in the backlog-api java library module:

- Add Retrofit and RxJava libraries to the build.gradle
- Create a client that uses Retrofit 2 and calls the Backlog API endpoints
    - Add a constructor that accepts Backlog space name and API key as parameters and builds the base URL.
- Define a request interceptor that sets the API Key to every url.
    - Refer to http://developer.nulab-inc.com/docs/backlog/auth#apikey for more information
- Create a response interceptor that translates a Http 404 exception to a Backlog exception
    - Refer to http://developer.nulab-inc.com/docs/backlog/error-response

Task 3: Implement Account screen

Backlog-api

- Create a Retrofit interface that defines user operations in the package operations
- Add Get Own User endpoint to the User Operations interface
    - Refer to http://developer.nulab-inc.com/docs/backlog/api/2/get-myself-user
- Create the User model in the package models
    - http://www.jsonschema2pojo.org/ could be used to generate the class.

App

- Create an activity that extends AccountAuthenticatorActivity in the view.account package
    - The activity should call GetOwnUser endpoint to verify the account information
    - An error message should be displayed if the account information are wrong
- Implement layout and resources files
- Save the account information into the AccountManager
- Create an authenticator that extends AbstractAccountAuthenticator
- Create an authentication service
- Update Android manifest file by adding required permissions and authentication service

Task 4: Implement My projects screen

Backlog-api

- Create a Retrofit interface that defines project operations in the package operations
- Add Get Project List endpoint to the Project Operations interface
    - Refer to http://developer.nulab-inc.com/docs/backlog/api/2/get-projects
- Create the Project model in the package models
    - http://www.jsonschema2pojo.org/ could be used to generate the class.
- Add Get Project Icon endpoint to the Project Operations interface
    - Refer to http://developer.nulab-inc.com/docs/backlog/api/2/get-project-icon

App

- Create the content provider for Project data in the package provider
    - Create the contract for Project data

    - Create a provider with methods for inserting, updating, deleting, and querying Project data
    - Implement a DbHelper that creates the Project table
- Create a Sync adapter that extends AbstractThreadedSyncAdapter
- Call Get Project List and save the data in the project table
    - For each project, call Get Project Icon and save each image to the internal storage
- Create an activity in the package view.project
- Create a fragment that implements LoaderManager.LoaderCallbacks
- Implement layout and resources files
- Create a Project list adapter that extends CursorAdapter
- Display project data from the cursor
- Update Android manifest file by adding required permissions and content provider

Task 5: Implement Issues screen

Backlog-api

- Add Get User List endpoint to the User Operations interface
    - Refer to http://developer.nulab-inc.com/docs/backlog/api/2/get-users
- Add Get User Icon endpoint to the User Operations interface
    - Refer to http://developer.nulab-inc.com/docs/backlog/api/2/get-user-icon
- Create a Retrofit interface that defines issue operations in the package operations
- Add Get Issue List endpoint to the Issue operations interface
    - Refer to http://developer.nulab-inc.com/docs/backlog/api/2/get-issues
- Create the Issue, Priority, and Status models in the package models
    - http://www.jsonschema2pojo.org/ could be used to generate the class

App

- Add Issue and User data to the content provider
    - Add User and Issue columns to the contract
    - Add methods to the provider for inserting, updating, deleting, and querying User and Issue data
    - Add to SQLiteOpenHelper User and Issue tables
    - Create a SQL view for retrieving issue and assignee data
- Add methods to retrieve User and Issue data to the Sync adapter
    - Call Get User List and save the data to the User table
        - For each user, call Get User Icon and save each image to the internal storage
    - Call Get Issue List and save the data to the Issue table
- Create an activity in the package view.issue
- For each issue status, create a fragment that implements LoaderManager.LoaderCallbacks
- Implement layout and resources files
- Create an issue list adapter that extends CursorAdapter

- Display Issue and Assignee data from the cursor
- Create a fragment for stats
    - Implement a custom view that draws the chart bar by using the Android canvas

Task 6: Implement Issue screen

Backlog-api

- Add Get Comment List endpoint to the UserOperations interface
    - Refer to http://developer.nulab-inc.com/docs/backlog/api/2/get-comments
- Create the Issue Type, Comment, Milestone models in the package models
    - http://www.jsonschema2pojo.org/ could be used to generate the class.

App

- Add Issue Type and Comment data to the content provider
    - Add Issue Type and Comment columns and URIs to the contract
    - Add methods to the provider for inserting, updating, deleting, and querying Issue Type and Comment data
    - Add to DbHelper the Issue type and Comment tables
    - Update the Issue table by adding a milestones column
    - Create a SQL view for joining Issue, Assignee, Created user, and Issue Type data
- Add methods to retrieve Comment data to the sync adapter
    - For each issue, save the Issue Type data to the Issue Type table if it doesn't already exist
    - Call Get Comment List and save the data to the Comment table
    - Concat the milestones of an issue and save it in the Milestones column of the Issue Table
- Create an activity in the package view.issue
- Create a fragment that implements LoaderManager.LoaderCallbacks
- Implement layout and resources files
- Create an adapter for comment that extends CursorAdapter
- Display Issue and Comment data from the cursor
- Add a ShareActionProvider to share issue

Task 7: Implement Settings activity

- Create a Settings activity in the view.settings package
- Add Issue Notification option
- Update Sharedpreferences with the selected option value

Task 8: Implement Last 10 opened issues widget

- Create a widget in the widget package that retrieves the last 10 opened issues
- Create a provider that extends AppWidgetProvider
- Create a service that extends RemoteViewsService
    - Click on an issue should open the corresponding Issue screen

- Create the layout and resources
- Create the appwidget-provider configuration file
- Add required permissions and service to the Android manifest file

Task 9: Add Google Analytics

- Add Google Analytics tracker to every screen
    - Refer to https://developers.google.com/analytics/devguides/collection/android/v4/

Task 10: Add Google Cloud Messaging

Backend

- Add a Google Cloud Module of type App Engine Backend with Google Cloud Messaging
- Implement device token registration and message delivery Api methods
- Build a web page that can send simulated backlog messages to device

For testing purpose, update messages from backlogtool.com will be simulated

App

- Setup Google Cloud Messaging client in the app
    - Refer to https://developers.google.com/cloud-messaging/android/client
- Register device by calling Backend registration API
- Create a BroadCastReceiver that asks the Sync adapter to update the backlog data in the content provider
- Create a Gcm listener that will send a notification to the BroadCastReceiver when a message from Backend is received.
    - If the message is of type Issue Created a notification should be displayed
    - Notifications should be grouped by project.
    - A click on a notification should open the corresponding issue.
    - A click on a group of notifications should open the corresponding project issue list

Task 11: Add Tablet layouts

- Create tablet layout and resources files
- Update My projects and Issues fragments so they can be displayed in a Master/Detail flow
- Update the Issue fragment so it can be displayed as a DialogFragment when an issue is selected