GitHub Username: j-l5662

AWS Documentation

Description

Improve your AWS knowledge with the AWS Documentation App. View all of the available documentation for various AWS services such as EC2, VPC, and S3. Save documentation pages for offline viewing and share them with coworkers and friends.

The AWS Documentation Android App was developed for both aspiring AWS users and seasoned professionals as a helpful guide and tool. Bringing the documentation to a mobile friendly platform will make it easier for users to access and utilize as a study guide.

Permissions Notice
AWS Documentation may ask for permission to access the following features:
• External storage for downloading offline data

Intended User

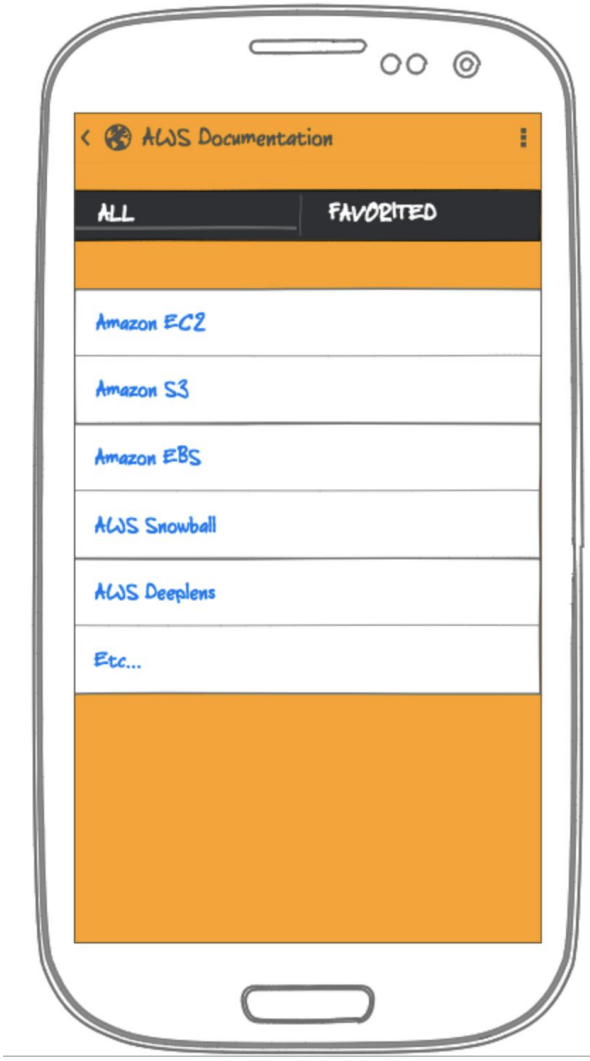
- Individuals who want to get more familiar with AWS services and solutions
- Users who are looking to use the documentation to study for the various AWS certification
- Professionals who are working with AWS that need an accessible reference guide

Features

- View the AWS documentation and navigate through them in an Android application.
- Share a documentation page URL.
- Save a documentation page for a user to view offline.
- Widget to display “Favorites” AWS services in a List View. Users can click on an item to launch the app into the selected service.

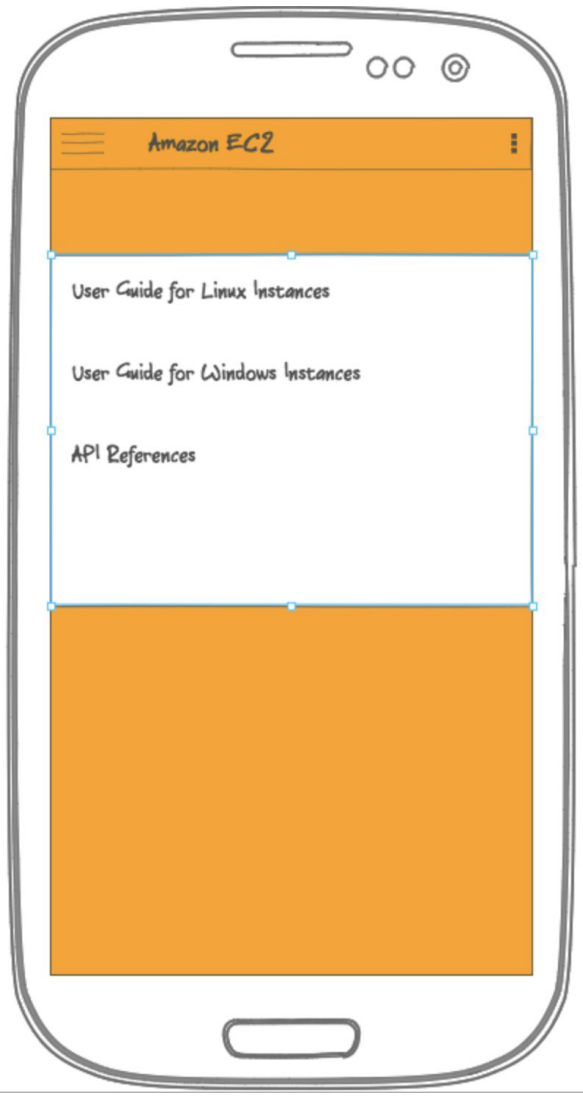
User Interface Mocks

Main Activity Screen



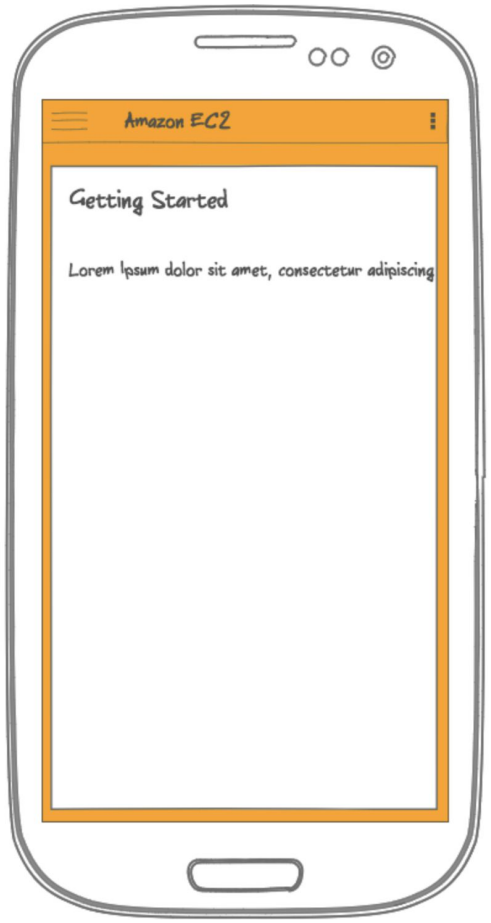
Main Activity screen with all of the services listed.

Detail Fragment Screen



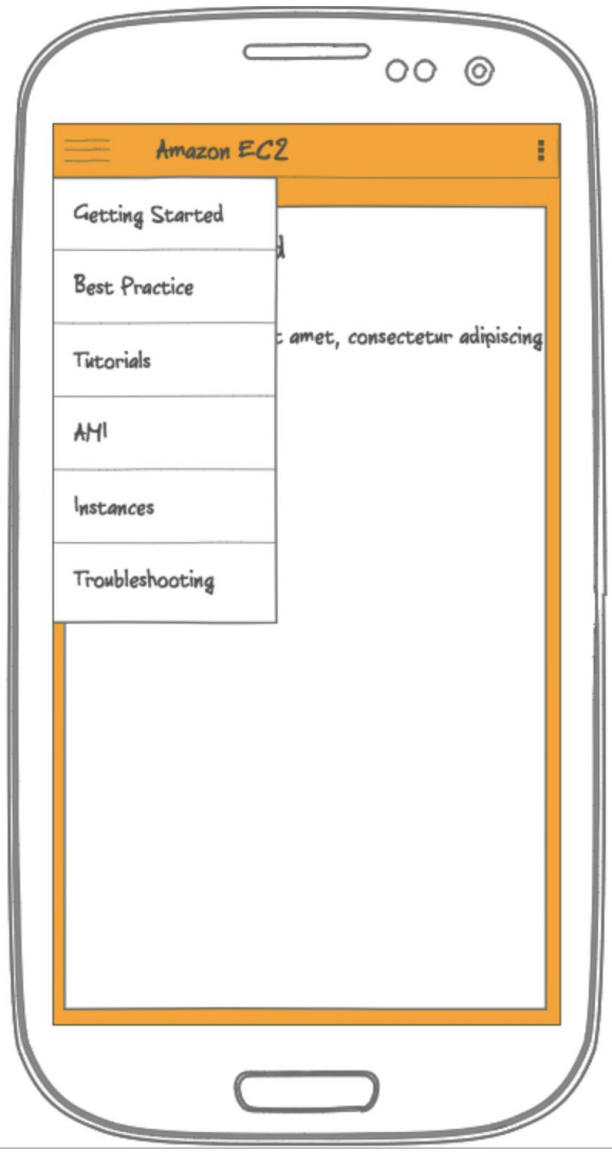
Detail fragment displaying the a page with the list of options that a user can choose.

Documentation Fragment Screen



Detail fragment displaying the documentation.

Navigation Drawer Screen



Displaying the navigation drawer to display other options within the documentation page.

Widget Mockup

Displaying the widget listview for users to view their current favorite documentation page and to click on them to launch the Detail Activity/Fragment.

Key Considerations

How will your app handle data persistence?

Room - Used to store the services documentation when a user decides to download it for offline view.

Sharedpreferences - Used to keep track of a users favorited service. The widget will utilize the shared preference to list out the services.

Describe any edge or corner cases in the UX.

Access resources when offline: If a user has clicked on a link that is referring to a resource that has not been downloaded, the app will display a toast message that tells the user that it is unavailable.

Rotating the screen during a network call: Using ViewModels and Live data will prevent lifecycle changes from affecting the UI.

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

For example, Picasso or Glide to handle the loading and caching of images.
Picasso to load images from URLs.
Room to store information about the documentation for offline viewing.
Volley to perform network tasks such as retrieving the documentation.
Google Play Services: Analytics and Crashlytics

Library	Gradle Implementation
Picasso	implementation 'com.squareup.picasso:picasso:2.71828'
Room	implementation "android.arch.persistence.room:runtime:1.1.1"
Volley	implementation 'com.android.volley:volley:1.1.1'
Firebase Analytics	implementation 'com.google.firebase:firebase-core:16.0.1'
Firebase Crashlytics	implementation 'com.crashlytics.sdk.android:crashlytics:2.9.3'

Describe how you will implement Google Play Services or other external services.

Google Analytics for Firebase - Implement Firebase to track how often users are downloading the documentation page for offline viewing.

Firebase Crashlytics - Implement Crashlytics to document and track crash reports that occur on a user application. Utilize the reports to perform QA and improvements on the app.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

- Write application only in Java
- **Utilize Android Studio (v.3.1) and Gradle (v4.4)**
- Configure libraries (Room, Volley, and Picasso)
- Document hardcoded strings (Documentation URL, Application Title, and Button Descriptions) within the strings.xml file.
- Update Build.Gradle file to a supported API level
- Enable RTL layout switching

Task 2: Implement UI for the Main Activity/Fragment and App theme

Build UI

- Build UI for MainActivity within a Fragment
    - Set up Recyclerview for the list of services.
- Set up Volley task to retrieve list of service from AWS documentation site.
    - Perform short duration, on demand network requests using Volley to download images and titles for each AWS service.
    - Create Java class to store data
- Configure App themes
- **Implement a Shared Element Transition between the Main Activity and the Detail Activity**
- **Incorporate Content Description within the ImageViews**

Task 3: Create Detail Activity/Fragment

Create Detail Layout to

- Create Detail Activity/Fragment to display the data
- Configure Volley task to retrieve the data from the AWS service documentation page.
    - Perform short duration, on demand network requests using Volley to the documentation information for each AWS service.
    - **Implement AsyncTasks to display a progress bar while the application is saving the service documentation into the database.**
    - Store the data and the links in a Java class

Task 4: Develop Navigation Drawer

Add the navigation drawer to the detail fragment

- Utilize links to store in the navigation drawer
- Pass the data into the fragment
- Test navigation drawer functionality and detail fragment updates

Task 5: Implement Favorite Functionality

Implement SharedPreferences functionality

- Include shared preference functionality for users to favorite a page
- Add additional tabs in the main activity to display saved pages

Task 6: Implement Room Database

Add database functionality

- Implement the room database to store favorite pages.
- Add offline capabilities for the Main Activity and Detail Fragments

Task 7: Implement Widget

Create Widget

- Develop a widget to display the favorite pages
- Implement onclick functionality to open up the specified page

Task 8: Testing / QA

Test to ensure smooth experience for the user

- Display toast messages for unavailable pages
- Incorporate espresso testing to test UI
- Test database queries and data retrievals
- Test Async Tasks
