
GitHub Username: schrickl

Sky Meal

Description

Sky Meal is an app for airlines to allow their customers to purchase food and drink either before flight or during flight and have it delivered to their assigned seat during flight. The user of the app (traveler) would pay for their items via the app eliminating the need for the flight attendants to deal with credit cards.

Intended User

Airline travelers who want to purchase food and drink for their flight directly from their phone.

Features

- App is written in the Java programming language using stable release versions of Android Studio, Gradle, and all libraries.
- Provides the user with an account login screen or a screen to create an account if one doesn't exist. The user can also choose to login as a guest.
- Provides the user with a menu of food and drink options that the airline offers.
- Provides the user with the ability to order items from the menu.
- Displays a summary screen of the user's order before submitting.
- Emails a receipt to the user's email address.
- (Maybe) Gives the user the option to pay via Google Pay. **Question to reviewer: Is this easy to implement while still securing the user's credit card information or should this part be left out of the app?**

User Interface Mocks

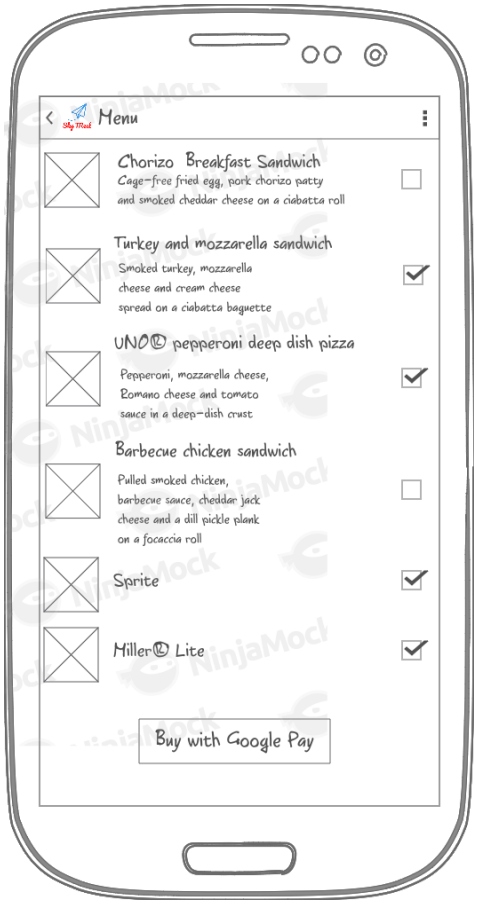
Splash Screen

This Splash Screen will appear to transition the user into the app. It will be implemented as an AsyncTask.

Login Screen

This screen will offer the user the ability to log into the app, create an account if they don't have one, or continue as a guest. It will utilize FIrebase Authentication for authorizing users.

Menu / Order Screen



This screen will present the user with the food and drink options they can purchase. Hitting the back button from here will prompt the user if they would like to discard their order.

Summary Screen

This screen will provide the user with a summary of their order before submitting. Hitting the back button from here will take the user back to the order screen.

Pay Screen

This screen is for entering payment information. It will utilize the Google Pay API to process the request.

Confirmation Screen

This screen will provide a confirmation/receipt that the users order was processed successfully. From here, the user will have the ability to share the receipt with an email address.

Network Error Screen

This screen will appear if there is no network connection available.

App Widget Screen

This is the app widget that the user can place on their phone's home screen. It will display the total cost of the most recently placed order or a quick link to the airline menu if no recent order exists.

Key Considerations

How will your app handle data persistence?

- User account information and airline menu information will be stored using Firebase Realtime Database.
- The cost of the user's most recent order will be stored in SharedPreferences and displayed in the app widget. If no recent orders are available, a quick link to the airline menu will be displayed in the app widget.

Describe any edge or corner cases in the UX.

- If no network connection is available, a suitable message will be shown to the user.
- If the user logs out of the app, it returns to the login screen.
- If the user clicks on the back button while in the process of ordering a dialog will be shown asking if they would like to discard the order.
- If the user loses connectivity while creating an order, the app will utilize the Firebase Realtime Database caching feature to preserve the order until connectivity is restored.

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

- Android Studio 3.1.4
- Gradle 3.1.4
- AppCompat 27.1.1
- Support 27.1.1
- RecyclerView 27.1.1
- CardView 27.1.1
- Firebase Realtime Database 16.0.1 for storing and syncing data
- Firebase Authentication 16.0.3 for secure user login
- Google Pay 16.0.0 for user payments
- Picasso 2.7 for picture loading
- ButterKnife 8.8.1 for Android view field and method binding

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

- The app will use Firebase Realtime Database for storing menu information and user account information.
- The app will use Firebase Authentication to validate user accounts.
- The app will use the Google Pay API to allow users to pay for their order in the app.

Next Steps: Required Tasks

Task 1: Prepare UI Mockups and App Design

- This document

Task 2: Project Setup

- Create a blank project in Android Studio

- Setup project build.gradle with stable release of Gradle
- Setup app build.gradle with stable releases of all libraries
- Add required permissions to AndroidManifest.xml
- Add support for RTL to Android Manifest.xml
- Create a github repository for the project

Task 3: Implement UI for Each Activity and Fragment

- Create XML layout for Splash Screen
- Create XML layout for MainActivity
- Create XML layout for Menu Screen
- Create XML layout for Summary Screen
- Create XML layout for Confirmation Screen
- Create XML layout for App Widget

Task 4: Core Platform Development

- Integrate ButterKnife into all activities or fragments
- Perform validation on all input fields
- Include content descriptors where necessary and implement D-pad navigation to support accessibility. Note: non-audio versions of audio cues are not applicable to this app.
- Place all app strings in a strings.xml file and not hard-coded in the java source code
- Support both English and Spanish languages by using string localization
- Show the cost of most recently placed order in the app widget. Or, a link to the airline menu if no recent order exists.

Task 5: Implement all Java classes

- Implement Splash Screen utilizing an Android AsyncTask
- This could include Activities, Fragments, Adapters, or any other supporting classes.

Task 6: Implement Google Play Services

- Add Firebase Realtime Database implementation
- Add Firebase Authentication implementation
- Add Google Pay implementation

Task 7: Incorporate Material Design

- Include app theme that extends from AppCompat

- Include an app bar and associated toolbars
- Include standard and simple transitions between activities.

Task 8: Build and Release

- App builds and deploys from a clean repository checkout using the installRelease Gradle task.
- App is equipped with a signing configuration, and the keystore and passwords are included in the repository. Keystore is referred to by a relative path.