GitHub Username: taurelas

Prudent Cook

Description

Reduces food waste by searching through stored recipes based on the ingredients the user currently have.

The recipes are easily added to the app by copy and share and can be organised using custom labels.

Intended User

The app is in English so a user must be able to understand it in order to use the app.

Any smartphone user who wants to use ingredients currently available in his or her kitchen.

Such user wants to try something different or doesn’t know at all what he or she can cook with a particular set of ingredients. They want to browse from a list of available dishes, pick one and start cooking, using the found recipe.

Another, very similar type of user doesn’t mind buying few things extra or maybe borrowing them from a neighbour. He is related closely to the user above

What further defines such user is who he or she is **not**: a person who wants to simply browse recipes or look for a specific dish.

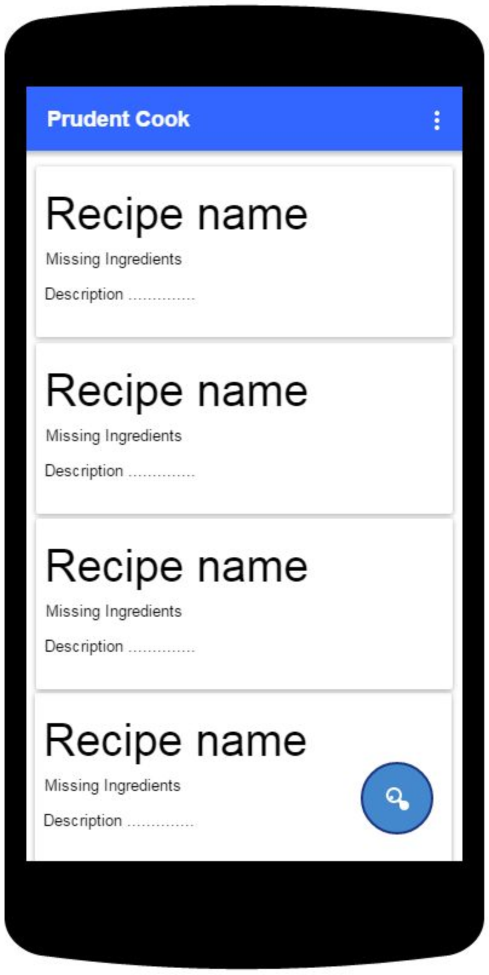
Features

Stores food recipes
Displays recipes matching a list of chosen ingredients
Accepts shared text with recipe and attempts to process it
Saves recipes
Mark recipes as favourite
A widget that displays a recipe of your choice from your favourites

User Interface Mocks

Screen 1

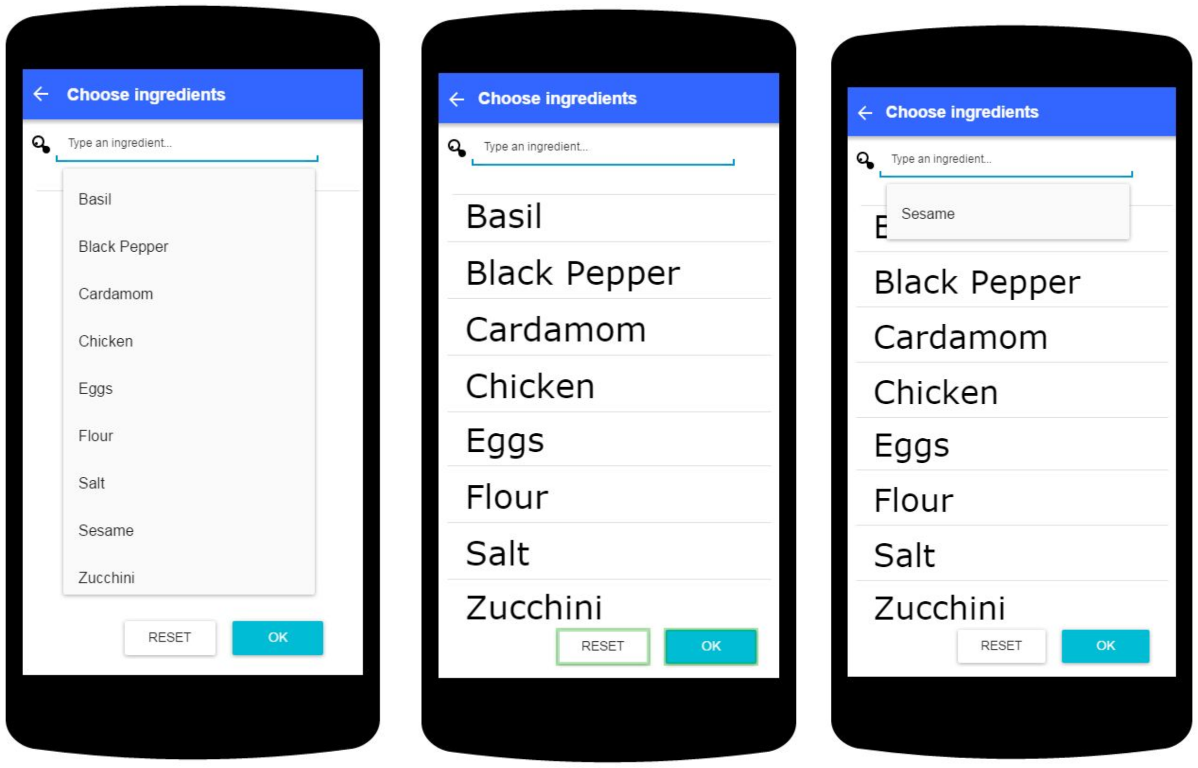
Here we need to display a list of chosen ingredients on top and recipe names, ingredients and beginning of the description, grouped as an element of a list. (see Screen 3 for another state)



MainActivity - FAB button launches IngredientsActivity. Clicking on the Recipe launches RecipeActivity

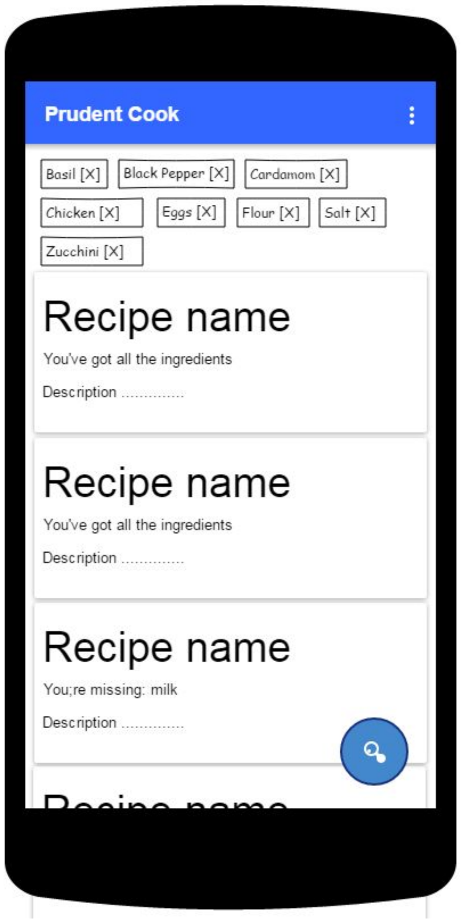
Screen 2

After clicking the FAB button on Screen 1, a list of chosen ingredients is displayed and an EditText above. When the user starts typing, remaining available ingredients appear



IngredientsActivity - adding an item to RecyclerView removes it from the list of suggestions. Pressing OK, Up or Back accepts changes. Pressing Reset clears the list

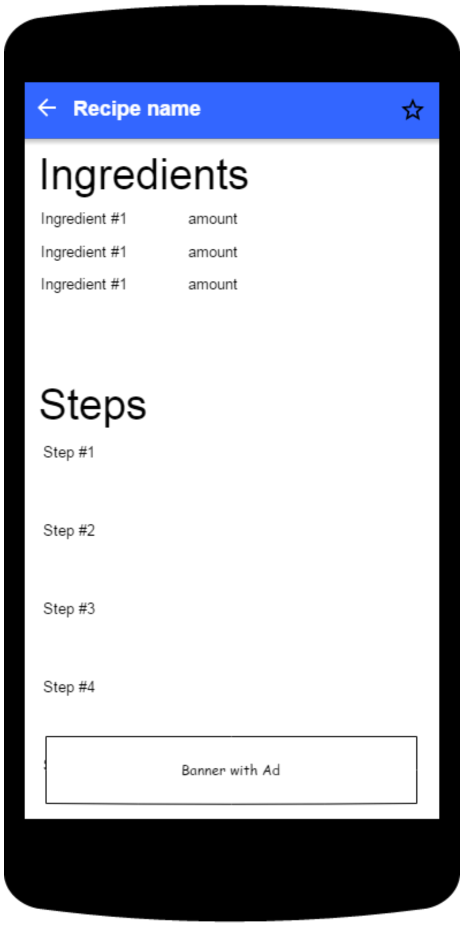
Screen 3



MainActivity with a list of chosen ingredients. The ingredients visible for each recipe are those missing from the list above. Clicking on a recipe will open a RecipeActivity

Screen 4

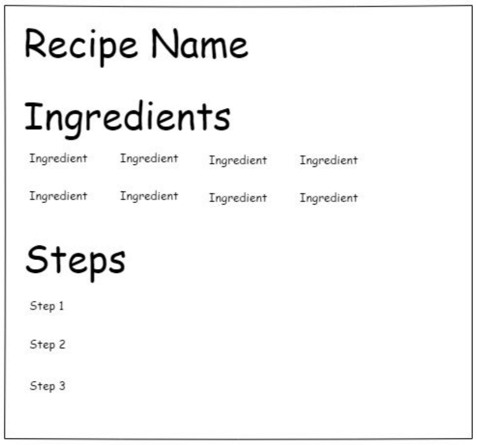
Here the full recipe is displayed



RecipeActivity - displays the recipe, back and up button go back to the list

Screen 5

This is a widget, it has flexible width and height



Key Considerations

General considerations

- App is written solely in the Java Programming Language, save for UI elements which rely on XML.
- App does not crash, force close, freeze, or otherwise function abnormally on any targeted device.
- App adheres to the Google Play Store App policies.
- All app dependencies are managed by Gradle.

Data persistence

The data will be saved in MySQL db using Room library so it doesn't have to be fetched
Simple settings will be stored in SharedPreferences

Describe any edge or corner cases in the UX.

App allows RTL layout switching on all layouts.

MainActivity:

- Pressing the FAB search button opens IngredientsActivity
- Clicking any part of the item (recipe) opens RecipeActivity
- When the phone rotates, the RecyclerView position is maintained, data is preserved in ViewModel so no additional calls to API & db are made
- When the user presses the back button, the Activity is destroyed
- When the user presses the home button RV position and the displayed RV data is restored to the same state when the user returns to the app
- When any ingredients are chosen, they appear on the top of the screen, with an X button next to each. Clicking such button removes the ingredient from the screen and updates the list of recipes. **(A custom view)**

IngredientsActivity

- Reset button resets the state of the activity
- Back button moves to MainActivity, keeping ingredients choices
- OK button works just like Back button
- User by typing displays a list of suggestions, it does not feature elements that are already added to the list of ingredients
- Taping a suggestion adds it to the activity's recyclerview
- When the user presses the back button, they are sent back to MainActivity

- When the user pressed the home button
- User text input is preserved on rotation.

RecipeActivity
- Pressing back simply closes the activity
- Tapping the star adds the recipe to favorites, tapping it again removes it.

If there is no internet connection, the user should have access to at least recipes marked as favorites

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

AppCompat for handling older versions of Android
Timber 4.7.1 for displaying logs
Room for CRUD operations on MySQL database
~~Retrofit for accessing API with recipes~~ (no point I guess since it's required to use IntentService)
GSON for deserializing JSON into Java Objects
Google Play Services Ads for displaying ad banner
Firebase core for Firebase Analytics

App utilizes only stable release versions of all libraries and Gradle

IDE
The app will be developed in Android Studio 3.1.3

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

GPS Ads will display an AdMob test ad in RecipeActivity
Firebase Analytics will gather anonymous data about ingredients chosen
The app will use Edamam API for source of recipes and ingredients

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

- Create a PrudentCook project in Android Studio with a Simple Activity
- Add the dependencies for Timber, Firebase, Firebase-Auth and Google Play Services Ads
- Create a github repository

If it helps, imagine you are describing these tasks to a friend who wants to follow along and build this app with you.

Task 2: Implement UI for Each Activity

- Ensure the theme extends AppCompat
- Create a set of Material colors and add to the Theme
- Create a custom view for ingredient for MainActivity as shown on Screen 3
- Build UI for MainActivity, with an app bar and associated toolbars.
- Build UI for IngredientsActivity - the Activity only allows the ingredients provided by API
- Build UI for RecipeActivity
- Optional for RecipeActivity: Implement sharing functionality in your app, making use of intent extras to share rich content (i.e. a paragraph of content-specific text, a link and description, an image, etc).
- Use standard and simple transitions between activities OR shared element transitions across activities and parallax scrolling where two or more items must scroll in the same activity

Task 3: Implement AdBanner in RecipeActivity

- Implement test ad in RecipeActivity
- Set up AdMob project
- Add correct ids to release build

Task 4: Link up the activities

- Create an Intent and Callback for MainActivity-IngredientsActivity
- Something else

Task 5: Create Room Database

- Define interface required
- Implement the database

Task 6: Build Repository

- Create repository class & define interface
- Implement ~~Retrofit~~ API calls using IntentService
- Implement validation for the data received via Retrofit so the app does not crash and llog any issues
- Implement cache so no unnecessary calls to API are made
- Store the data in the Room db

Task 7: Create ViewModels

- Create ViewModel for MainActivity
- Wire MainActivity with the ViewModel
- Create ViewModel for IngredientsActivity
- Wire IngredientsActivity with ViewModel

Task 8: Create a widget

- Design widget UI
- Implement a widget
- Access app data via the widget

Task 9: Attach Firebase Analytics

The app creates only one analytics instance.

- Implement Analytics in MainActivity
- Configure Firebase Analytics in Firebase

Final Task

Ensure that:

- App keeps all strings in a strings.xml file and enables RTL layout switching on all layouts.
- All colors are defined in colors.xml and resul
- Each service imported in the build.gradle is used in the app.
- All content is safe for work content.
- App's code follows standard Java/Android Style Guidelines.
- App includes support for accessibility. That includes content descriptions & navigation using a D-pad
- App does not redefine or misuse Android UI patterns, such that icons or behaviors could be misleading or confusing to users.
- App does not request permissions to access sensitive data or services that can cost the user money, unless related to a core capability of the app.
- App builds from a clean repository checkout with no additional configuration.
- App builds and deploys using the installRelease Gradle task.
- App correctly preserves and restores user or app state, that is , student uses a bundle to save app state and restores it via onSaveInstanceState/onRestoreInstanceState.
- When an activity is displayed, the same activity appears on rotation.
- When the app is resumed after the device wakes from sleep (locked) state, the app returns the user to the exact state in which it was last used.
- When the app is relaunched from Home or All Apps, the app restores the app state as closely as possible to the previous state.
- 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.

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
  - Make sure the PDF is named “**Capstone_Stage1.pdf**”
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
- Add this document to your repo. Make sure it's named “**Capstone_Stage1.pdf**”