
[Description](#)

[Intended User](#)

[Features](#)

[User Interface Mocks](#)

[Screen 1](#)

[Screen 2](#)

[Screen 3](#)

[Screen 4](#)

[Screen 5](#)

[Screen 6](#)

[Key Considerations](#)

[How will your app handle data persistence?](#)

[Describe any corner cases in the UX.](#)

[Describe any libraries you'll be using and share your reasoning for including them.](#)

[Next Steps: Required Tasks](#)

[Task 1: Project Setup](#)

[Task 2: Implement UI for Each Activity and Fragment](#)

[Task 3: Implement Authentication](#)

[Task 4: Implement order communication](#)

[Task 5: Implement handling orders](#)

[Task 6: Implement notifications](#)

GitHub Username: [gabyrh77](#)

Bakery Lovers

Description

Bakey lovers allows you to order delicious desserts, pastry, breads, and a variety of bakery food, delivered directly to your location in no time.

Download the app now, sign in with your google account, search through the menu, order whatever you like, place an order, and you will receive a notification with the estimated delivery time.

Intended User

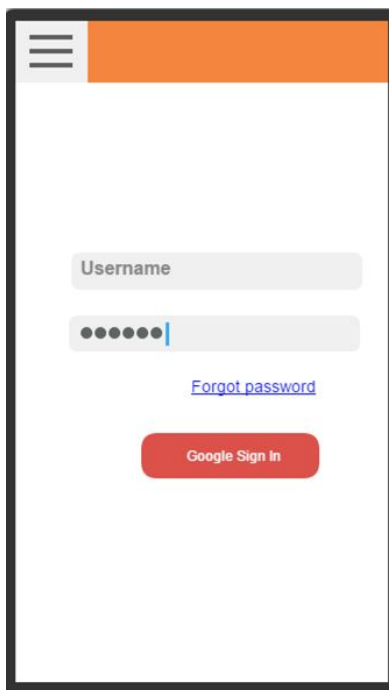
The app is intended for adults users (18+ years old) in the San Jose area, that would enjoy bakery food delivery.

Features

- Google sign-in
- Online Shopping
- Review Order History
- Real time geolocation with Google maps
- Notifications

User Interface Mocks

Screen 1



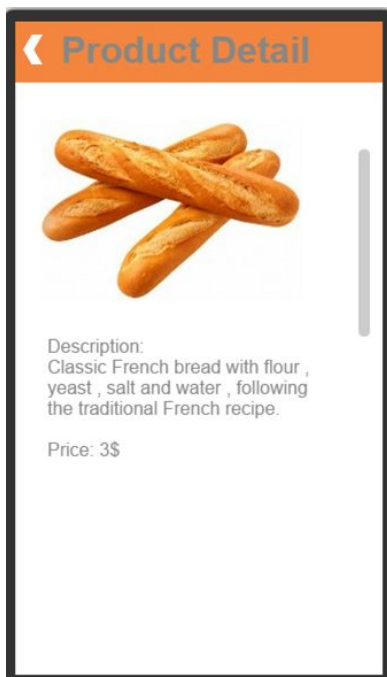
Login screen will allow the user to sign in using google account.

Screen 2



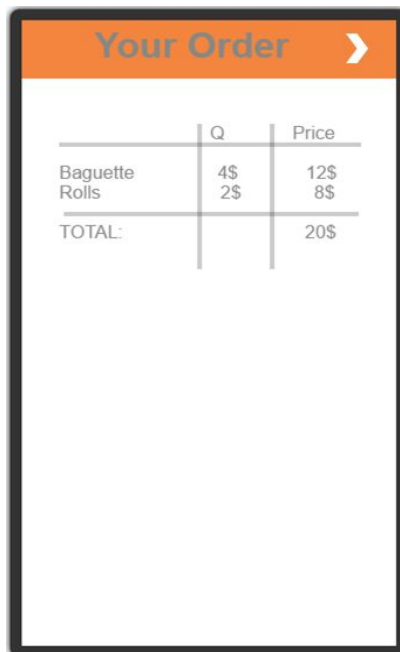
Menu items and handling orders UI. This screen will allow to search, add and remove items from a current order.

Screen 3



The product detail page will appear when a user clicks in a specific item.

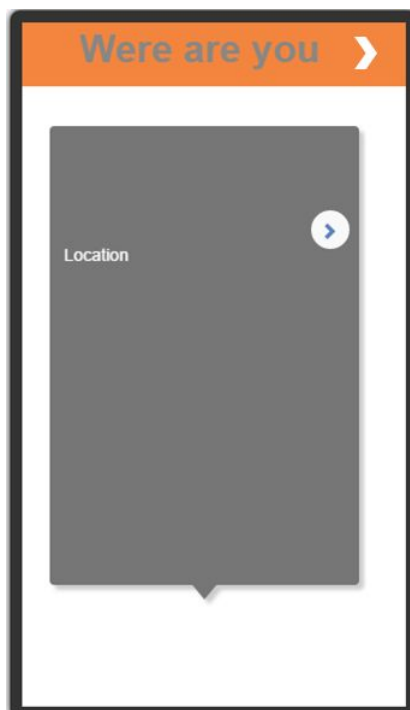
Screen 4



	Q	Price
Baguette	4\$	12\$
Rolls	2\$	8\$
TOTAL:		20\$

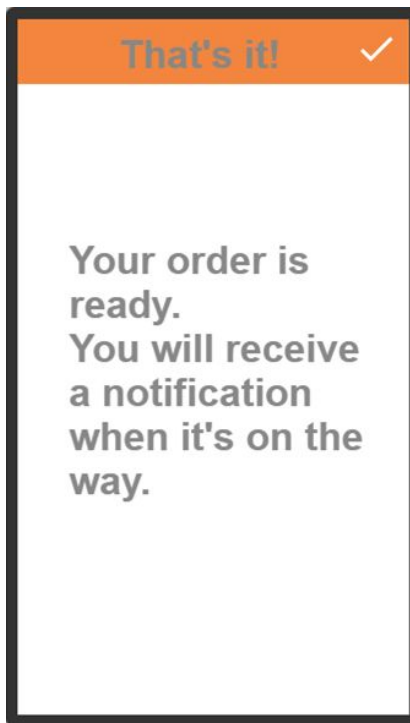
Review your order page contains the list of items, quantity and price for the order. It will allow to make modifications to the order.

Screen 5



The map view will allow the user to select the location for the delivery.

Screen 6



After the order is placed the user will see a success message, if there was a problem sending the order they will see an error message.

Key Considerations

How will your app handle data persistence?

I will build a Content Provider to store orders information of the orders, users and products.

Describe any corner cases in the UX.

The main screen will be the menu screen where there list of items will be presented with a search box. Every item will have a link to go to the product detail for that item, and action buttons that will allow to add or remove the item from the order. It's also important to handle the state of the order correctly during the entire UI flow.

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

- Glide to handle the loading and caching of images.
- Google maps API to determine the user's location.
- Retrofit to handle API order requests
- Google Sign-In API to handle authentication

Next Steps: Required Tasks

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

Task 1: Project Setup

You may want to list the subtasks. For example:

- Configure a google developer project with the required API
- Define the data objects and their attributes.
- Design the endpoints, responses and parameters to use.
- Configure android app project structure

Task 2: Implement UI for Each Activity and Fragment

- Build UI for Main Order Activity
- Build UI for the App Menu
- Build UI for the Login screens
- Build UI for Product Detail
- Build UI for Review Order
- Build UI for Map Location
- Build UI for Order Placed

Task 3: Implement Authentication

- Implement Google Sign In API
- Implement users endpoint.
- Implement sign in and sign out logic.
- Implement request and store user information
- Integrate authentication logic with the UI

Task 4: Implement order communication

- Implement get products endpoint
- Implement get orders endpoint
- Implement request and store products and orders
- Integrate orders and products with the UI

Task 5: Implement handling orders

- Implement handle order state class
- Implement add and remove items from orders
- Implement order endpoint
- Implement place order request
- Handle error cases

Task 6: Implement notifications

- Implement sending notifications
- Implement handle notifications on the phone