

[Description](#)

[Intended User](#)

[Features](#)

[User Interface Mocks](#)

[LoginActivity](#)

[LoginFragment](#)

[RegisterFragment](#)

[ForgotPasswordFragment](#)

[StoreListActivity](#)

[ItemListActivity](#)

[ProfileActivity](#)

[Notification](#)

[Widget](#)

[Key Considerations](#)

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

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

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

[Describe how you will implement Google Play Services or other external services.](#)

[Next Steps: Required Tasks](#)

[Task 1: Project Setup](#)

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

[Task 3: User account creation and management](#)

[Task 4: Create and save geofence](#)

[Task 5: Item List Manipulation](#)

[Task 6: Send Notification](#)

[Task 7: Add Widget](#)

[Task 8: Data Synchronization](#)

[Implementation Details](#)

[Programming Language](#)

[Tools](#)

[Accessibility](#)

[Resource Management](#)

[Background Services](#)

GitHub Username: nithyakonda

Shopin



Description

You were just at the mall and you forgot to buy milk and bread. Most households have their 'go-to' retail stores for their everyday needs. Wouldn't it be amazing to have an app that reminds you to buy from your shopping list whenever you enter one of these stores?

This is where Shopin comes in!

Shopin is a location-based list app that helps you to stay on top of your everyday shopping needs and also making the most of those unplanned store visits so that you can avoid the "Damn. I was right at the store and I forgot to buy that!" moments.

Shopin also keeps the list synchronized across multiple devices so that your family/roommates don't miss out!

Intended User

This app is intended for families.

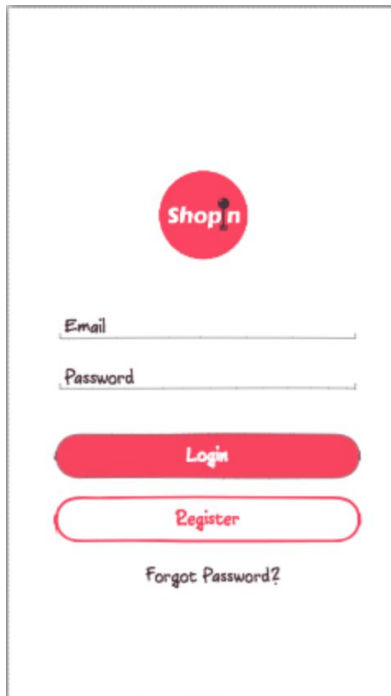
Features

- Creates an account so that the lists can be synced across multiple devices
- Adds list of stores
- Adds list of items
- Sends notification if there are any items to shop when the user enters a store

User Interface Mocks

LoginActivity

LoginFragment



- From the login activity the user can
 - Register if using the app for the first time
 - Login
 - Request for password reset
- The app keeps the user logged in until they explicitly logout
- Account creation and management is handled by Firebase Authentication
- After login is successful, [StoreListActivity](#) is opened

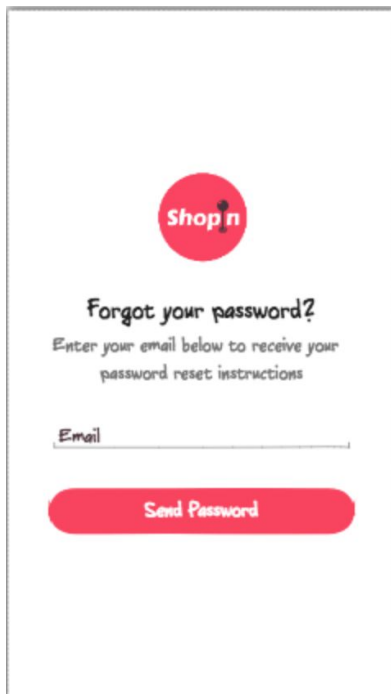
RegisterFragment



The RegisterFragment UI mockup features a white background with a red circular logo at the top center containing the text "shop n" in white. Below the logo, the word "Hello" is displayed in a bold, black, sans-serif font. Underneath "Hello", the text "Create an account for the household" is shown in a smaller, grey, sans-serif font. The form consists of four input fields: "First Name" and "Last Name" are side-by-side, followed by "Email" and "Password" on separate lines. Below the "Password" field is a "Confirm Password" field. At the bottom of the form is a red, rounded rectangular button with the word "Create" in white, italicized text.

- The idea is to have an account per household and the lists and stores can be synchronised across various devices
- After account is created successfully, the [StoreListActivity](#) is opened

ForgotPasswordFragment



The ForgotPasswordFragment UI mockup features a white background with a red circular logo at the top center containing the text "shop n" in white. Below the logo, the text "Forgot your password?" is displayed in a bold, black, sans-serif font. Underneath, the text "Enter your email below to receive your password reset instructions" is shown in a smaller, grey, sans-serif font. The form consists of a single "Email" input field. At the bottom of the form is a red, rounded rectangular button with the text "Send Password" in white, italicized text.

A password reset email is sent to the provided email address.

StoreListActivity



This landing page contains cards of various stores and the items to be purchased there.

- Store cards can be deleted by swiping right/left
- New store can be added by clicking the “+” fab. This opens a Place Picker using the Places SDK.
- On click of the settings button on the top right corner, the user profile page is opened
- When a place is picked or an existing card is clicked, [ItemListActivity](#) is displayed.

ItemListActivity



In this screen, the user can

- Check purchased items
- Add new items
- Delete items

Up navigation is present to go back to the [StoreListActivity](#)

ProfileActivity



The ProfileActivity UI mockup is a white rectangular box containing a red circular profile picture with the letters 'NK' and a small black plus icon. Below the picture is the text 'Hi Konda !'. Underneath are three text input fields: the first contains 'Nithya' and 'Konda', the second contains 'nithyakonda@gmail.com', and the third is labeled 'Display Name'. At the bottom is a red rounded rectangle button with the text 'Logout'.

Allows user to

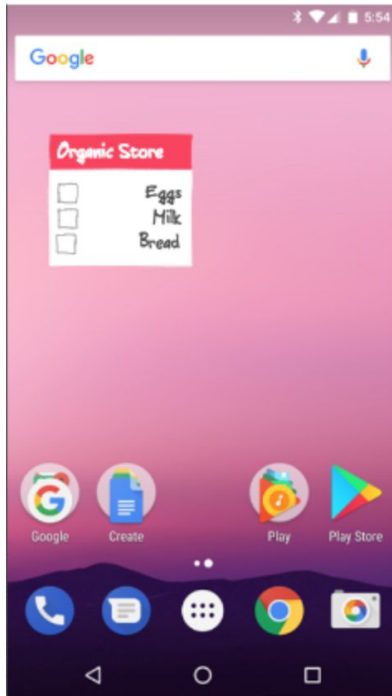
- Change only
 - Display picture
 - Display name
- Logout
 - Navigates to [LoginActivity](#)

Notification



- Whenever the user enters the vicinity of a store that has items to be purchased, a notification like above is sent.
- On clicking the notification, [ItemListActivity](#) is opened.

Widget



- Clicking anywhere on the widget takes the user to [ItemListActivity](#)
- User can only check items as done from the widget

Key Considerations

How will your app handle data persistence?

App uses Firebase Realtime Database to handle data persistence.

Describe any edge or corner cases in the UX.

- Since there is a limit of having 100 geo fences per device, the app can only allow the user to add up to 100 stores.
- If a selected place already exists, the app should open the existing list instead of creating a duplicate list.

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

Library	Version	Purpose
Firebase Authentication	16.0.1	For account management
Firebase Realtime Database	16.0.1	For persistence

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

Places SDK	To select a store location
Location APIs	Geofence creation and management

Next Steps: Required Tasks

Task 1: Project Setup

- Signup and get Google API Key to use Places SDK and Locations APIs
- Setup Authentication
- Setup Firebase Realtime Database
- Setup Database Rules

Task 2: Implement UI for Each Activity and Fragment

- Build UI for all activities and fragments
- Handle orientation changes
- Add transitions between screens

Task 3: User account creation and management

- Create new account in Firebase when new user is registered
- Login when valid credentials are entered
- Send password reset email
- Change display name and display picture
- Logout from session
- Ensure that the app keeps the user logged in until they choose to logout

Task 4: Create and save geofence

- Launch PlacePicker on click of Add button
- After picking a place
 - Create a geofence for the selected place
 - Save it in the database
 - Start ItemListActivity

Task 5: Item List Manipulation

- Update UI and database for the following actions
 - Add new item
 - Delete selected item
 - Mark item as checked - move it to the bottom of the list
- Navigate to StoreListActivity onBackPressed or Up navigation

Task 6: Send Notification

- Send notification when user enters any registered geofence
- Open ItemListActivity of that store, when clicked

Task 7: Add Widget

- Create a widget that shows items from the last modified store
- Mark items as checked and update the database
- Open ItemListActivity of that store, on clicking anywhere else except the checkbox

Task 8: Data Synchronization

- Make sure the data is in sync across different mobile devices for the same account
- Make sure the geofences are updated whenever data is synced on other devices
- Logging off on one device will automatically log off open sessions on other devices

Implementation Details

Programming Language

The app will be programmed using Java

Tools

Android Studio	3.1.2
Gradle	4.4.1

Accessibility

The app supports accessibility by providing content description for UI views.

Resource Management

- All strings and colors are saved in their corresponding xml files
- App icons and other images are saved as drawables
- Layouts support RTL switching

Background Services

The app uses IntentService to post a notification when a geofence transition occurs