GitHub Username: anusri304

Essentials

Overview Description

- **Essentials** is an ecommerce App used for ordering products which will be written in Java programming language. The app will be utilizing stable release versions of all libraries, Gradle, and Android Studio.

  Description:
  - Various categories like Appliances, Toys, Toiletries etc can be purchased through the App.
  - Options for adding to the cart and wish list are available in the App.
  - Products in the wish list and cart are easily visible in the App.
  - Products are distinguished into separate categories and these categories are easily seen through the App.
  - The products in promotion are easily visible and can be purchased through the App.
  - Shopping is secure as the user is validated.
  - Ability to store data offline and sync with the server when there is internet connection.
  - The App offers a better shopping experience as it has the above features.

  What problem does your app solve?

  - Within the next two years, mobile commerce (mCommerce) will control 73% of all eCommerce sales across the globe.
  - Products can be easily purchased through the App anytime and anywhere without going to the shops.

  - As people are familiar using mobile devices the App can be used conveniently.
  - The mobile App will be in Sync with the Open Cart ecommerce store which will be deployed in the server.
  - 78% of consumers would rather use an app to buy from an eCommerce shop than a mobile website as it is convenient.
  - As Mobile sales are dominating the eCommerce world the App will contribute to mobile commerce sales.

Intended User

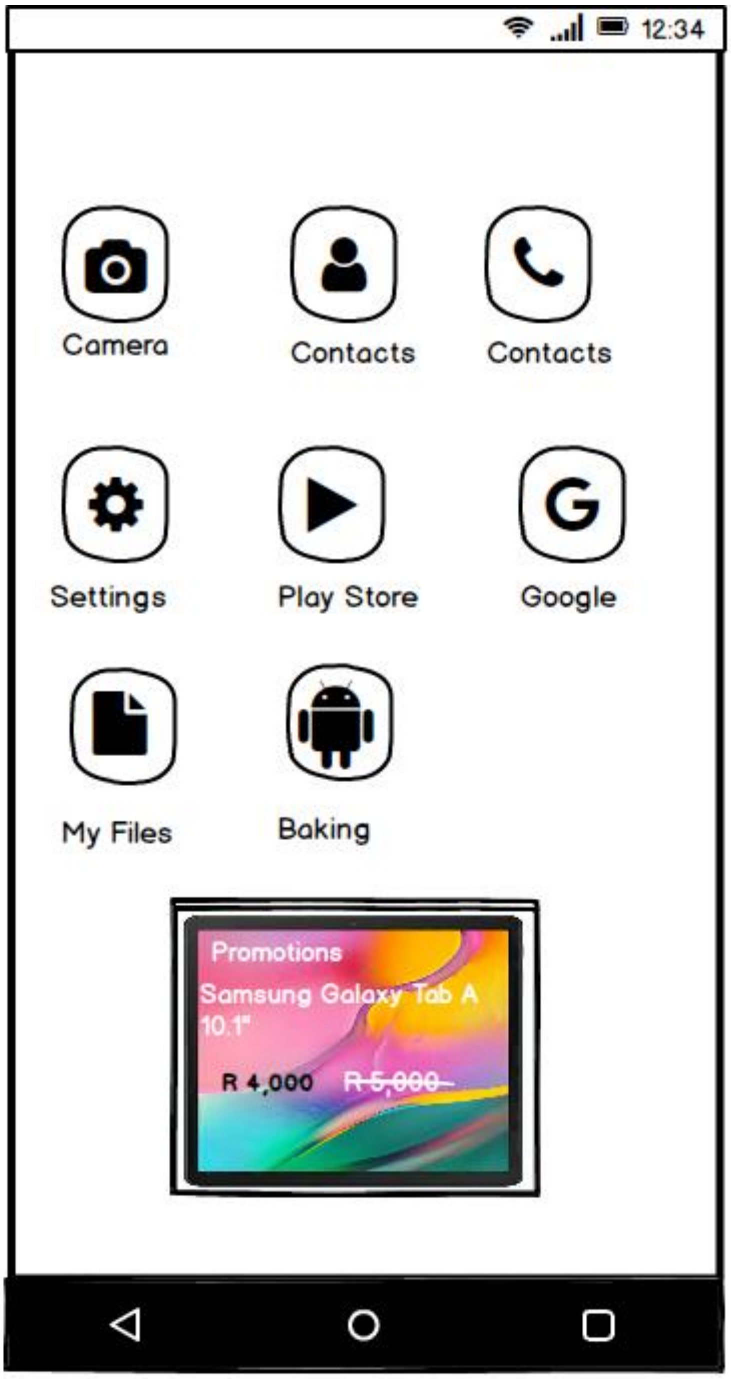
- The App is intended for people who have smartphone and wanted to purchase products through mobile App.
- All the essential products from Toiletries to Appliances can be purchased through the App thus attracting a wide range of audience.
- As the products can be ordered anytime and anywhere offering a contactless delivery the App should attract more people.

Features

- Display and search essential products
- View products in Promotion.
- App is always in Sync with Opencart website (If a product is added to the open cart website it will appear in the mobile App)
- Order products.
- Save delivery address, customer details and card details.
- View ordered products.
- Add products to a wishlist/cart.
- Ability to view categorized products (categories like Toys, Appliances)
- Orders can be placed offline and will be synced to the server when there is internet connection.

User Interface Mocks

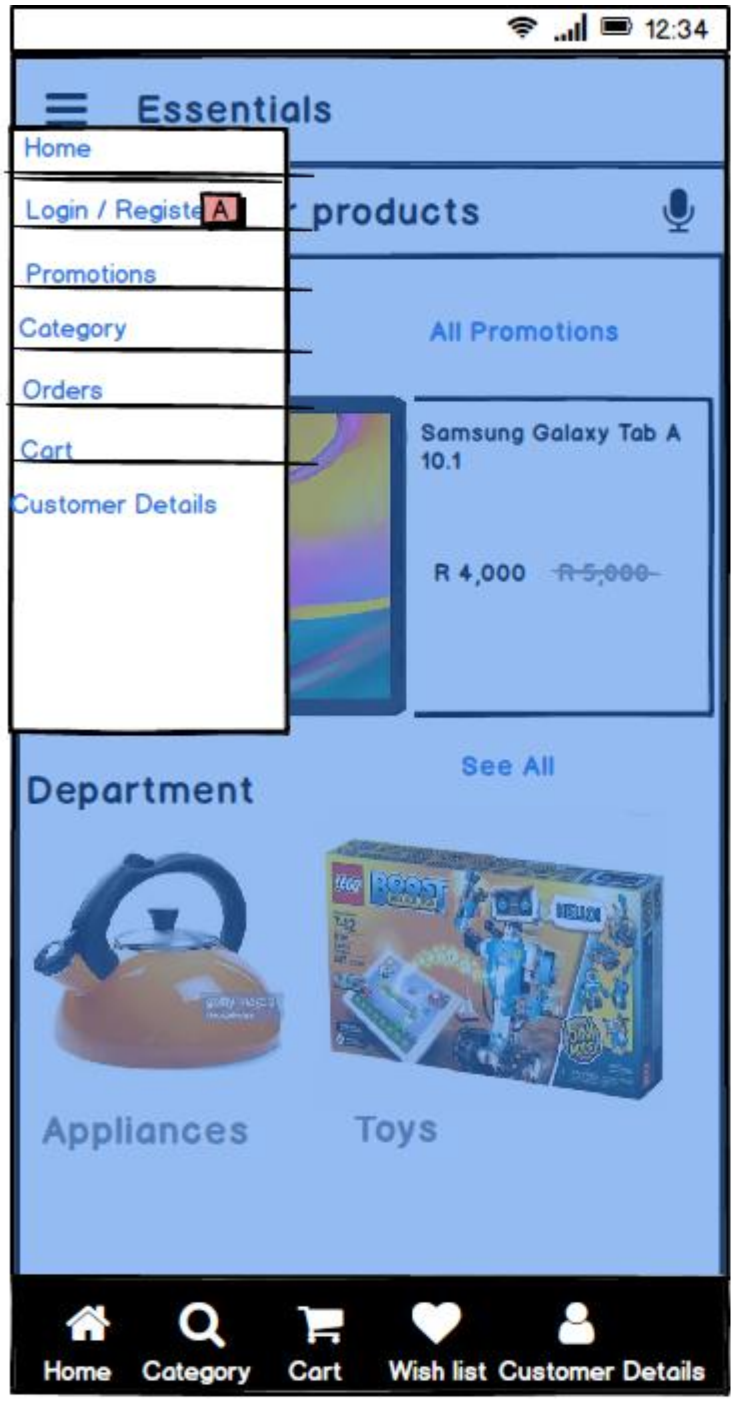
1) Widget Screen



This is the home page in the android mobile App having the Widget for the App.
The product in promotion will be displayed.
The product title along with the original and the reduced price will be displayed in the widget.

The product in the widget will be updated everyday.
Clicking the widget will launch the App.

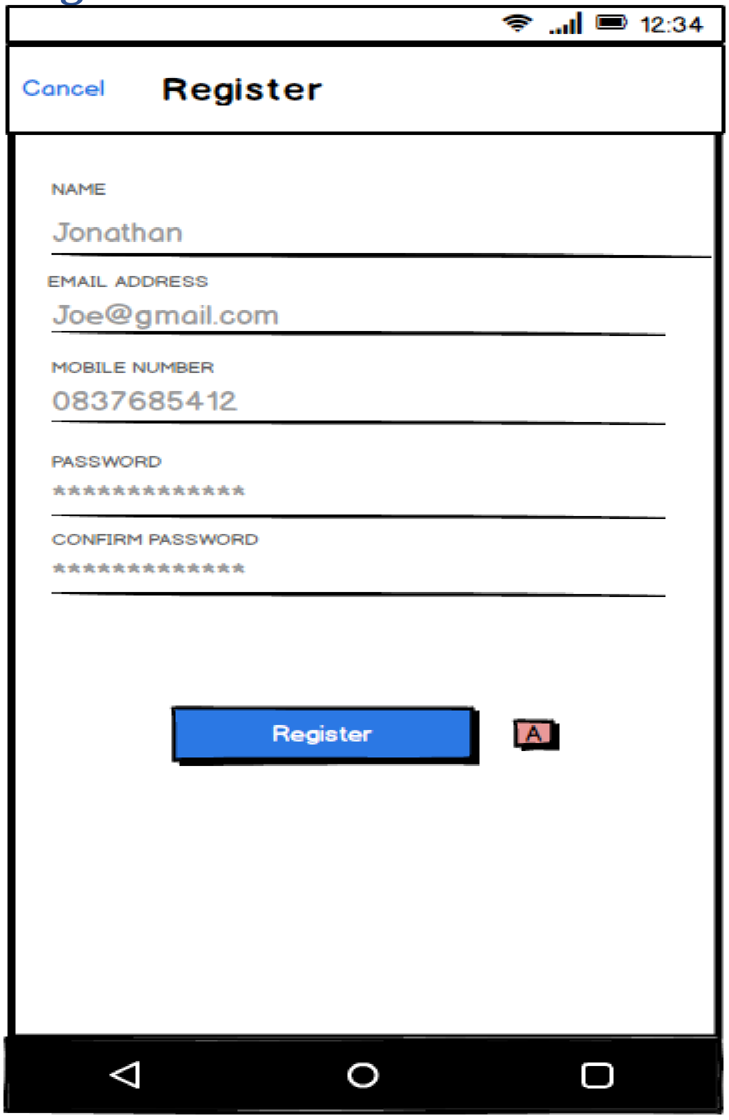
2) Menu



This is the Menu Screen providing Menu for the following:

- Register - Customer Creation screen
- Login- Screen for Customer login
- Promotions – Screen for seeing the products in Promotion
- Category – Screen for seeing various categories in the App like Appliances, Toys etc.
- Orders – Screen for seeing order details
- Cart – Screen for seeing products in the Cart
- Customer Details – Screen for viewing personal details.

3) Register Screen



12:34

Cancel **Register**

NAME
Jonathan

EMAIL ADDRESS
Joe@gmail.com

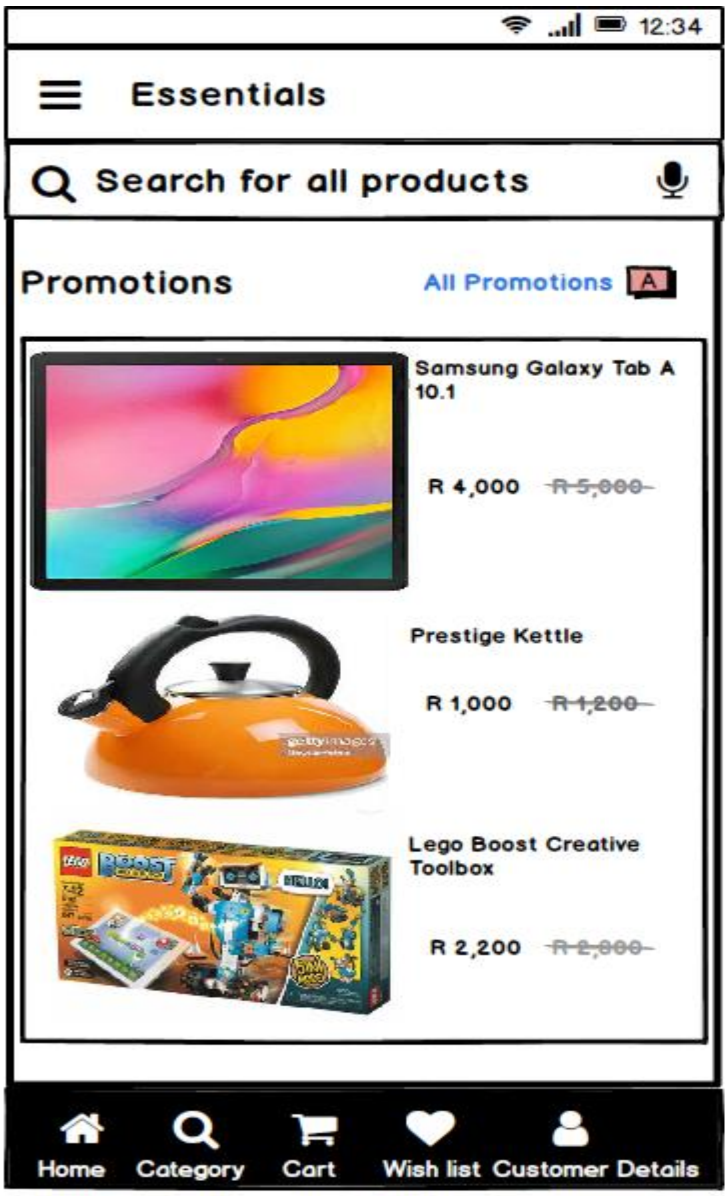
MOBILE NUMBER
0837685412

PASSWORD
****

CONFIRM PASSWORD
****

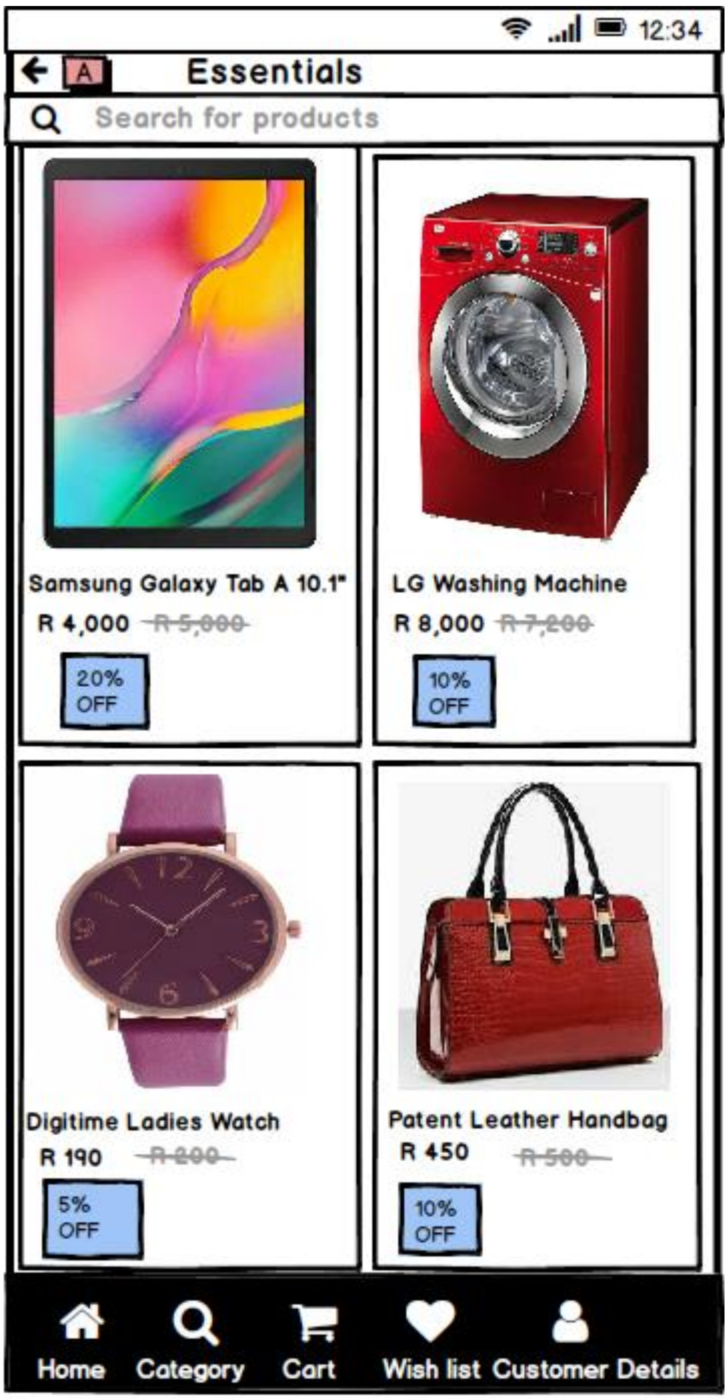
This is the screen used for creating the customer. The customer details are stored in the Firestore and is synced with the open cart server.

4) Landing Screen



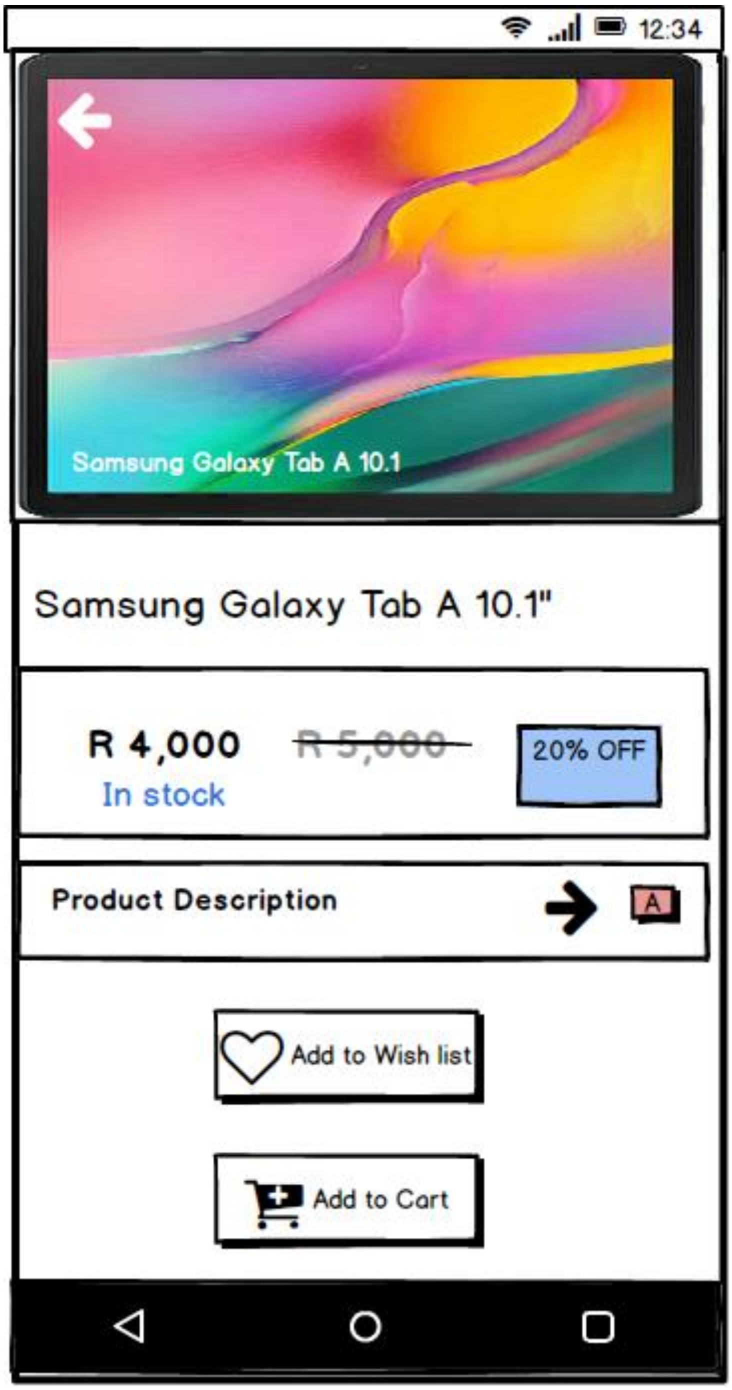
After the user has logged in successfully, the product details like the title, discounted price, original price, availability, if the product is in wishlist/cart is retrieved from the Opencart server using the Retrofit API and displayed in the Landing Screen..
These product details are saved in the Firestore.
The product information will be updated periodically.
The landing page of the App which shows some of the products in Promotion.
There is also option to search for the desired product.
This screen launches the screen for viewing all the products in promotion.

5) All Promotions Screen



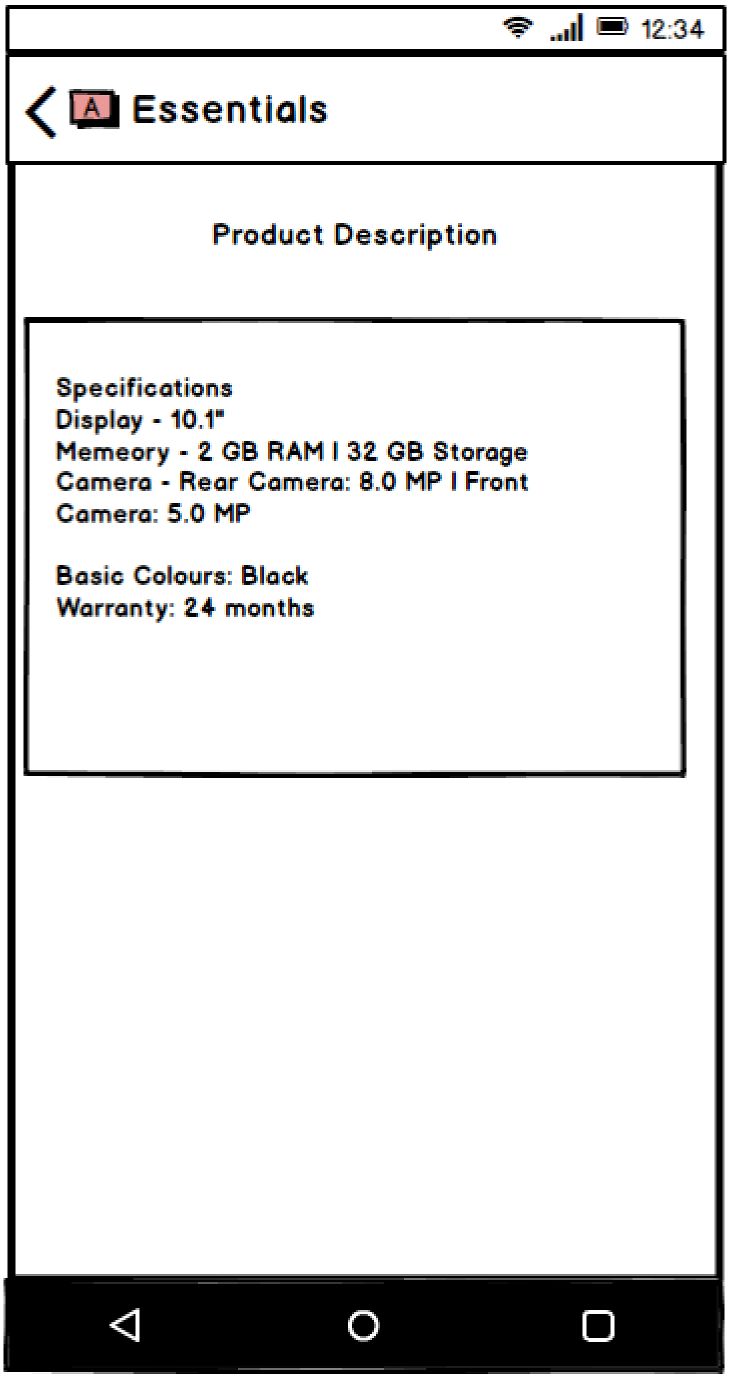
This screen is used to view all the products in Promotion. There will be a scroll bar to show all the products with the original price, discounted price and the percentage reduced.

6) Product Detail Screen



This is the detail screen for the product which displays the image of the product, price (original and reduced), percentage reduced. This screen also informs if the product is in stock or not. This screen provides the option to add the product to the cart/wish list. This screen lunches the screen for displaying the product description.

7) Product Description Screen



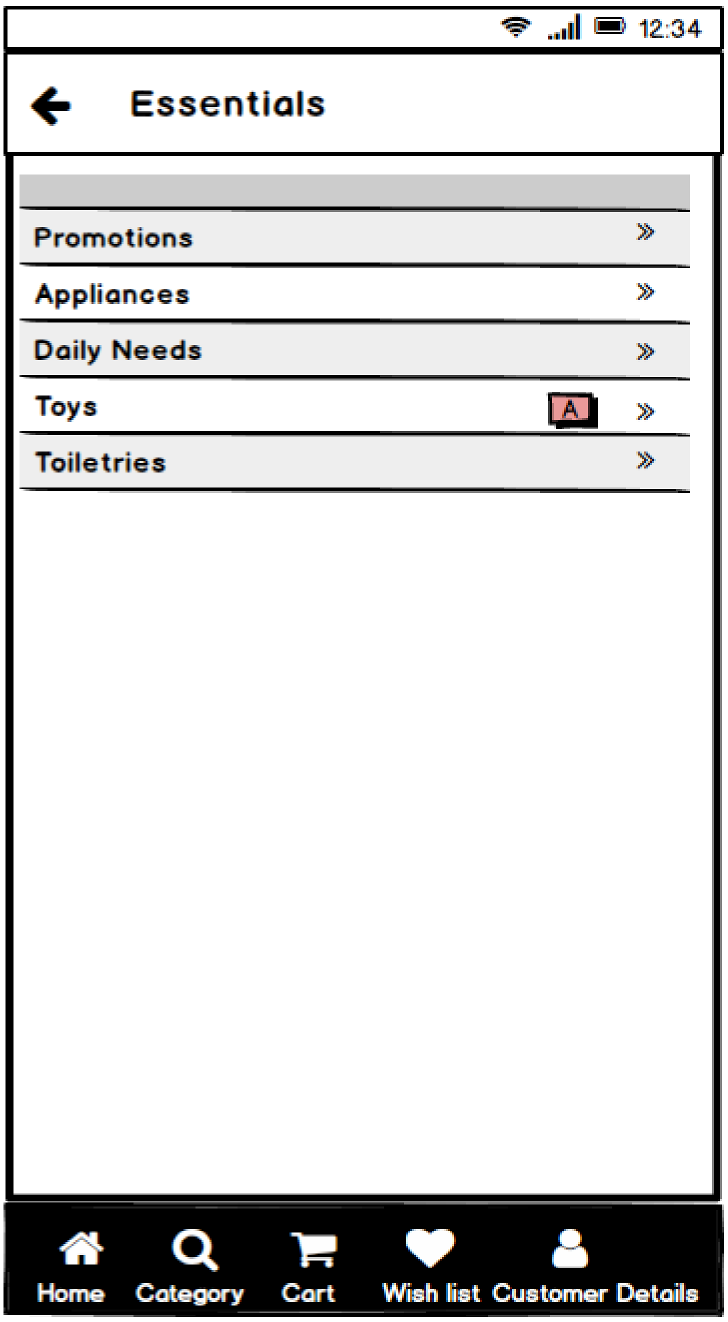
Product Description

Specifications
Display - 10.1"
Memeory - 2 GB RAM I 32 GB Storage
Camera - Rear Camera: 8.0 MP I Front
Camera: 5.0 MP

Basic Colours: Black
Warranty: 24 months

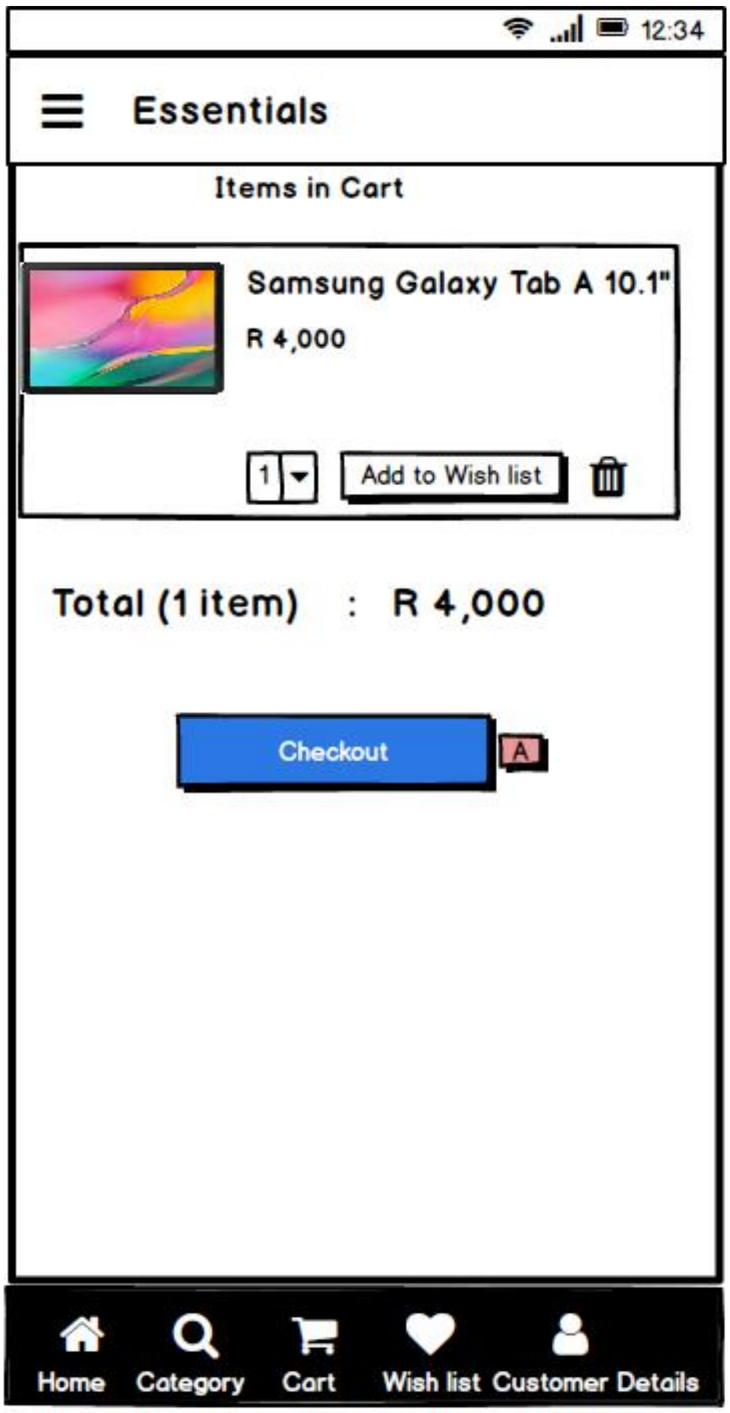
This screen provides the ability to view the product description.

8) Category Screen



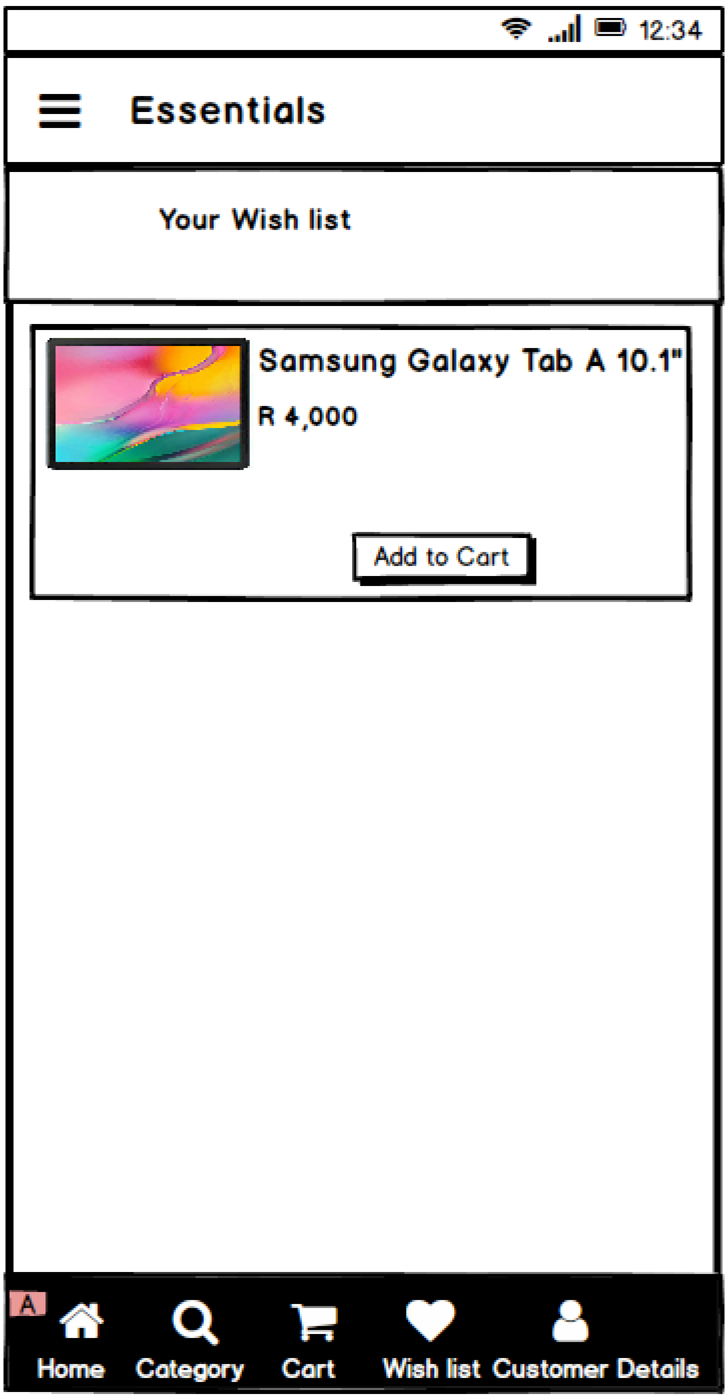
This screen provides the ability to display various categories of products like Appliances, Toys, Toiletries etc.

9) Cart screen



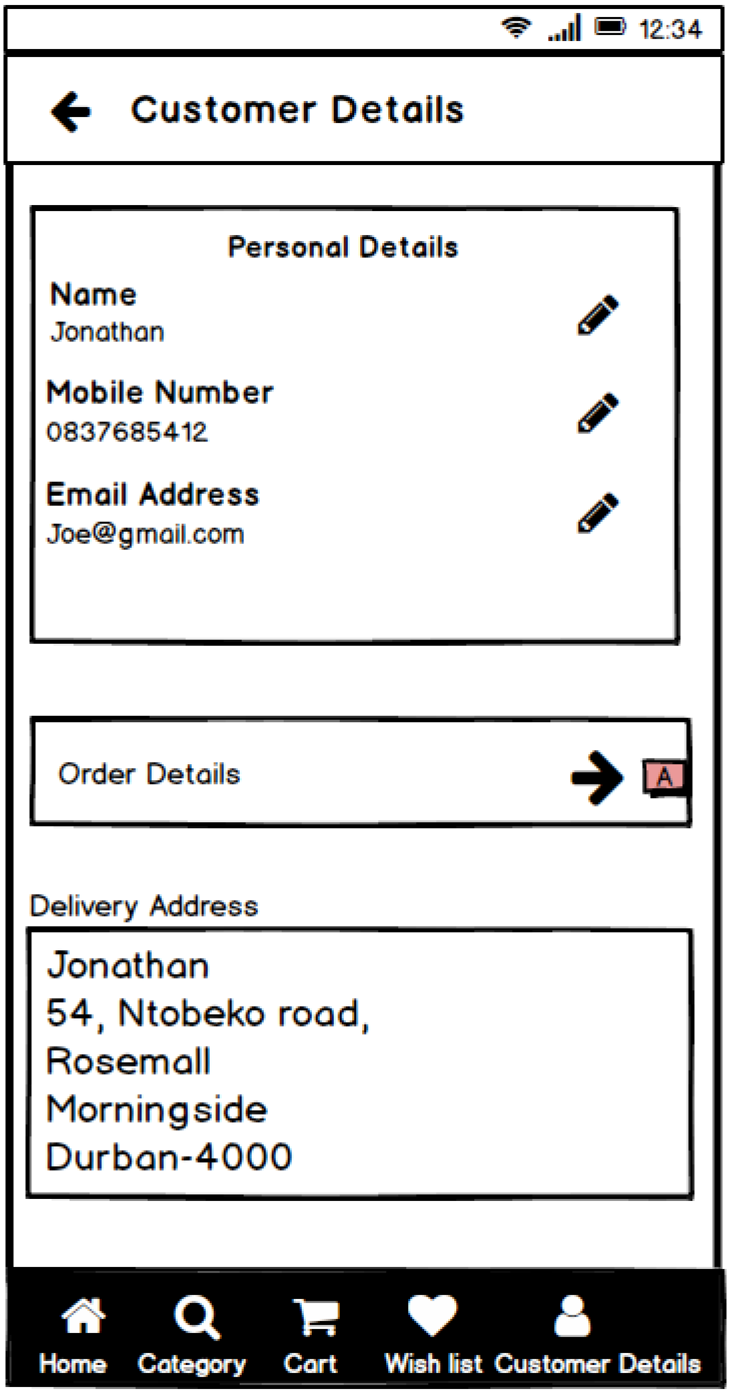
This screen provides the ability to view the items in the Cart. There is also option to delete items and to increment/decrement items. The total is displayed at the bottom allowing the user to checkout.

10) Wish list Screen



This screen provides the ability to view the items in the Wish list. There is also an option to add the products to the cart.

11) Customer Details Screen


12:34

Customer Details

Personal Details

Name
Jonathan

Mobile Number
0837685412

Email Address
Joe@gmail.com

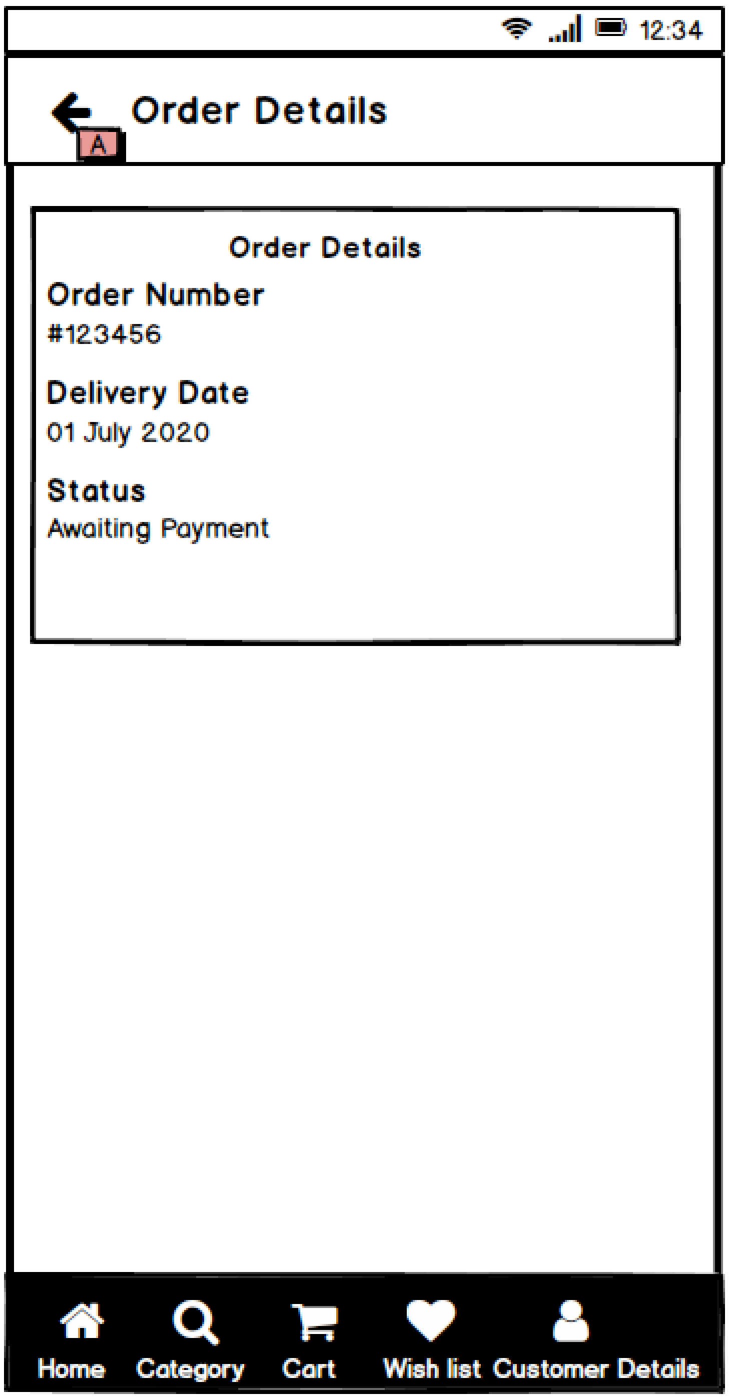
Order Details

Delivery Address

Jonathan
54, Ntobeko road,
Rosemall
Morningside
Durban-4000

This screen provides the ability to view personal details of the customer(name, mobile number, email address) along with the preferred delivery address. This screen launches the screen for displaying the order details.

12) Order Details



Order Details

Order Details

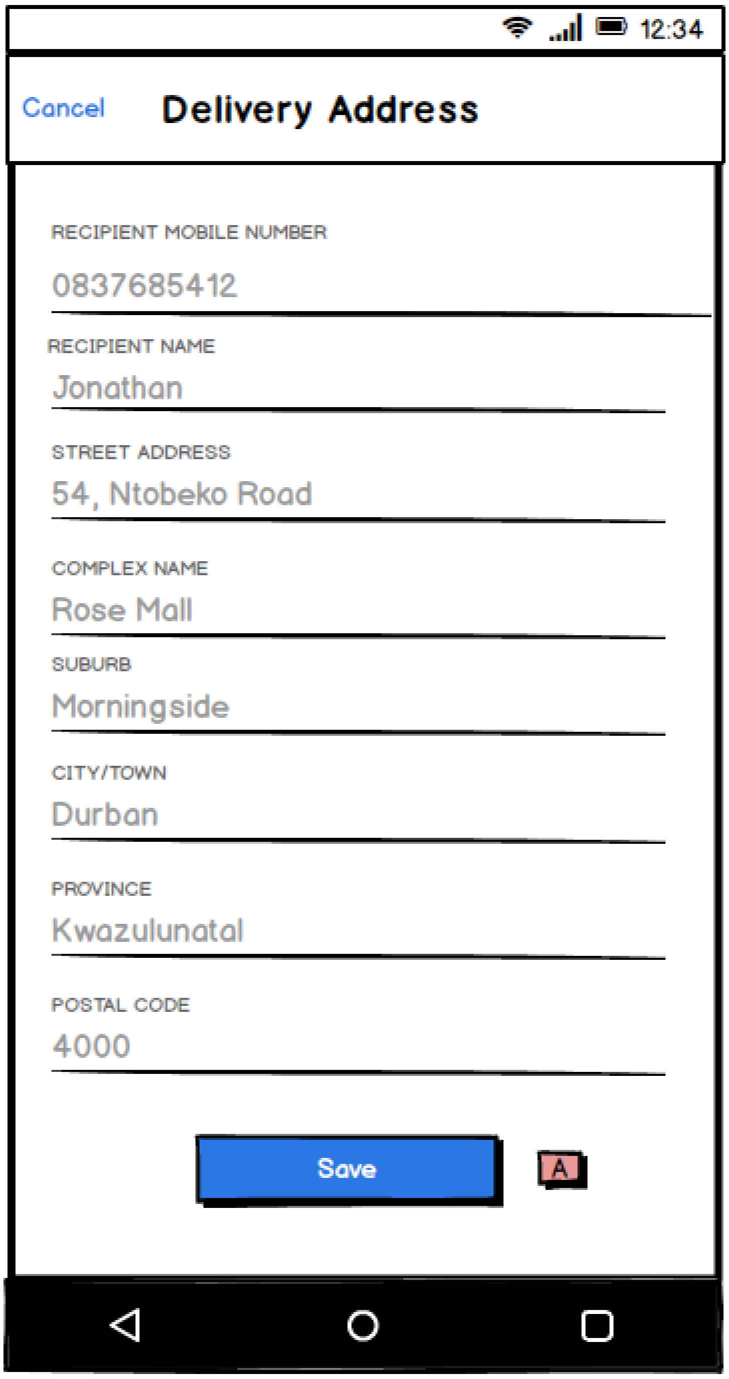
Order Number
#123456

Delivery Date
01 July 2020

Status
Awaiting Payment

This screen provides the ability to view the order details like order number, delivery date and status of the order. The order number will be generated if the payment is made successfully.

13) Add delivery address screen



12:34

Cancel Delivery Address

RECIPIENT MOBILE NUMBER
0837685412

RECIPIENT NAME
Jonathan

STREET ADDRESS
54, Ntobeko Road

COMPLEX NAME
Rose Mall

SUBURB
Morningside

CITY/TOWN
Durban

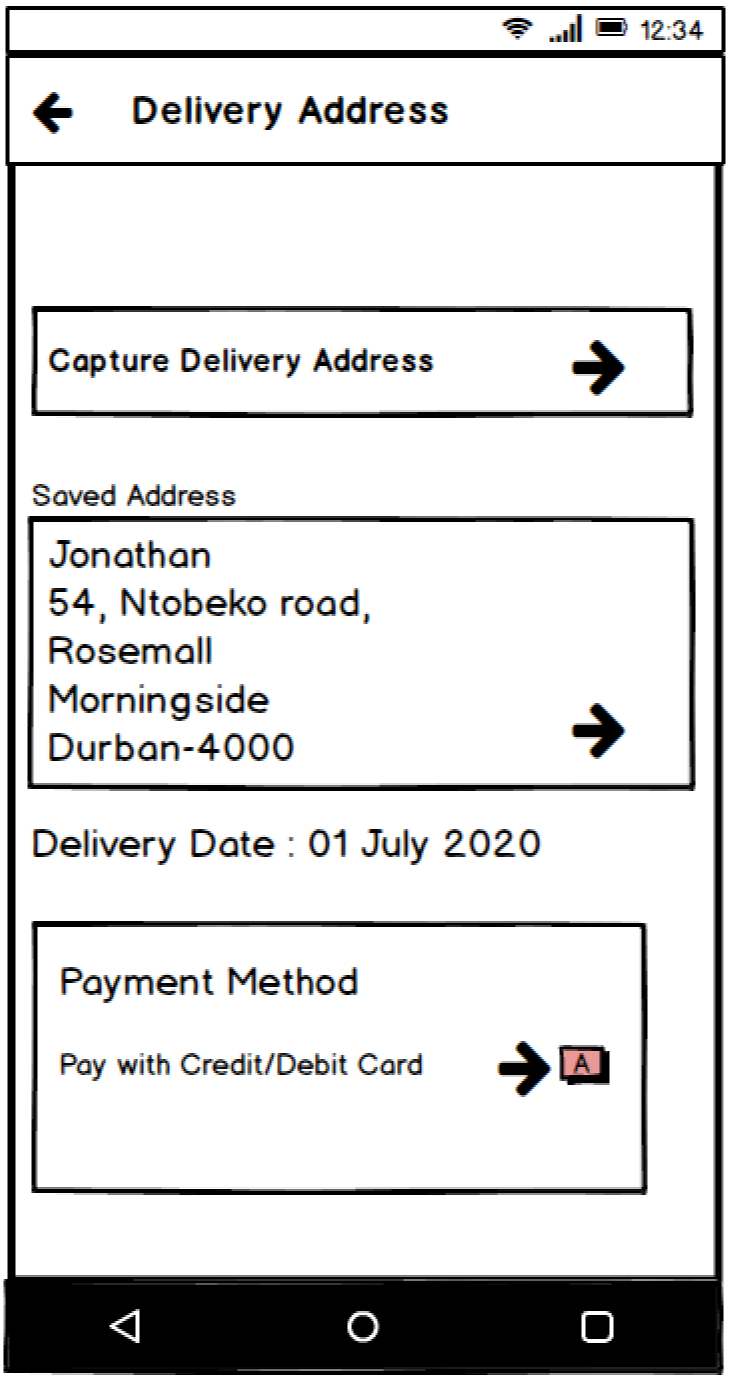
PROVINCE
Kwazulunatal

POSTAL CODE
4000

Save

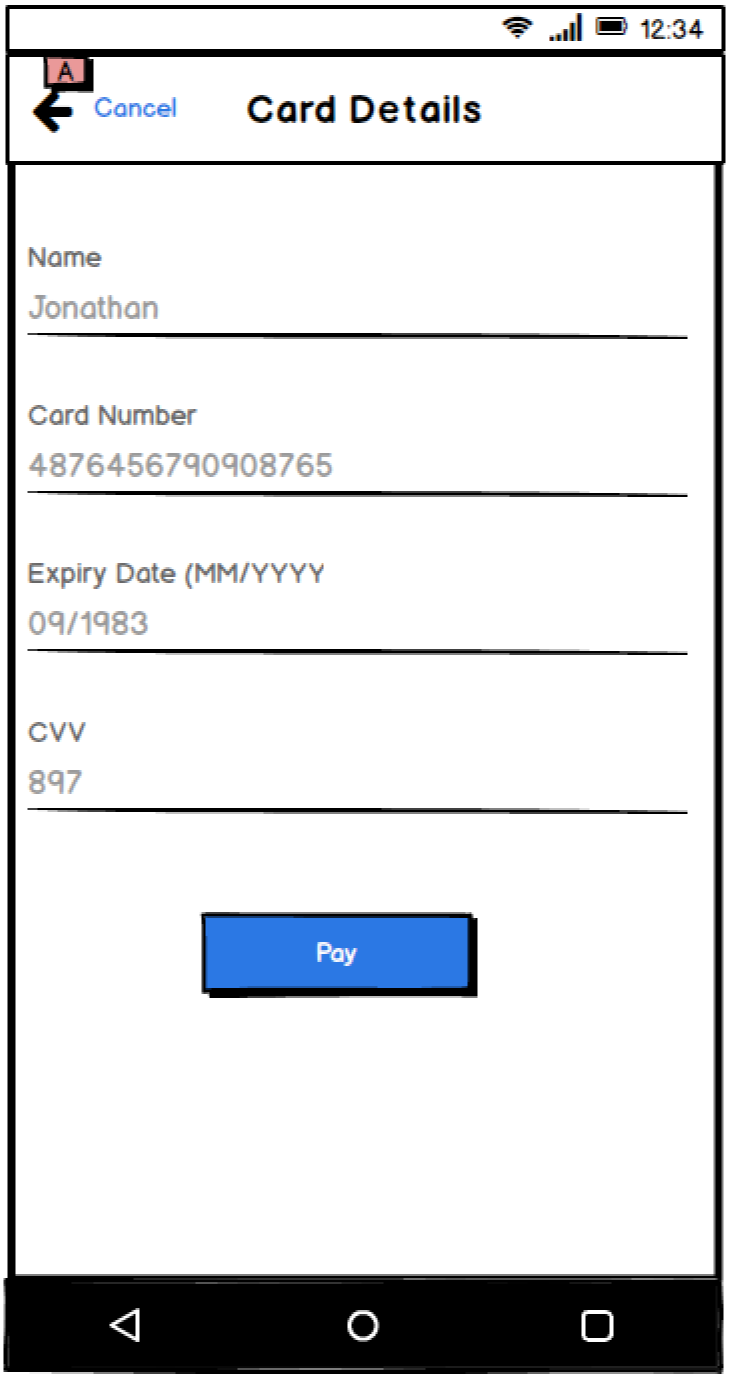
This screen provides the ability to add the delivery address of the customer. This includes the recipient mobile number, name, street address, complex name, suburb, city/town, province and the postal code. The details are saved in the Firestore and synced to the open cart server.

14) Saved Delivery address screen



This screen provides the ability to view the saved delivery address. This also launches the screen for making payment and capturing delivery address.

15) Card details screen



Name
Jonathan

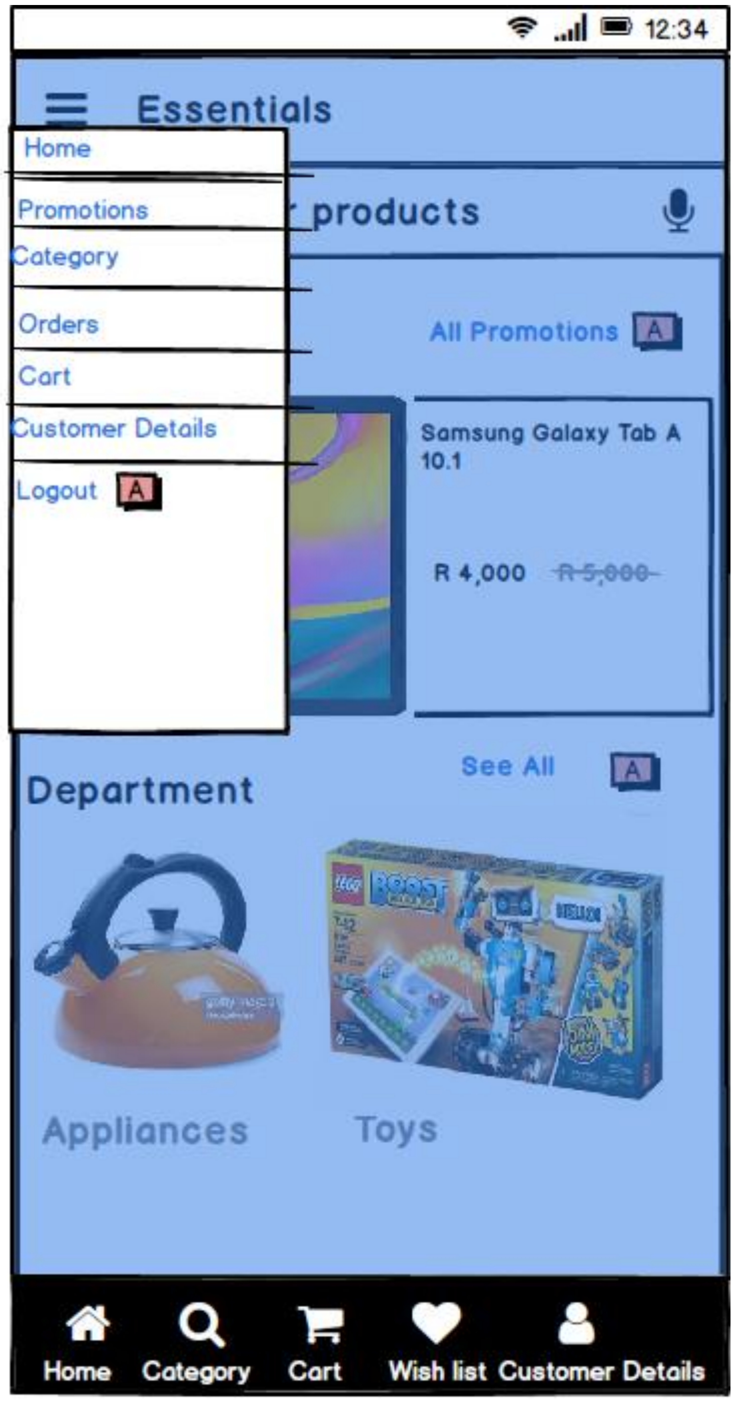
Card Number
4876456790908765

Expiry Date (MM/YYYY
09/1983

CVV
897

This screen provides the ability to add the card details of the customer. **The Payment gateway will not be implemented.**

16) Logout Toast



This menu allows the customer to logout.

I have uploaded a PDF of the mockups to the github project. Please go through them for the correct flow.

Key Considerations

How will your app handle data persistence?

The App will use Cloud Firestore to store data received from the open cart website.

When a user is created in the App the details of the user like the firstname, lastname, mobile number and email address are stored in the Firestore and synced with the Opencart server.
After the user has logged in successfully, the product details like the title, discounted price, original price, availability, if the product is in wishlist/cart is retrieved from the Opencart server using the Retrofit API and displayed in the App.
The product categories are also retrieved and displayed.
These product details, categories are saved in the Firestore.
The product and the category information will be updated periodically.
When the customer captures the delivery address and the card details they are stored in the Firestore and synced with the opencart server.

Implementation Details:
After creating the project in Android Studio use Firebase assistant to configure the Firestore. Follow the steps below:

- Connect the app to the Firebase (Create new Firebase project)
- Add Cloud Firestore dependencies to the App as below in the build.gradle
    - apply plugin: `'com.google.gms.google-services'`
    - implementation 'com. google.firebase:firebase-firestore:17.1.2'
- Declare an instance of FireBaseFireStore class
- Save and retrieve data from the Firestore.

Describe any edge or corner cases in the UX.

There is a provision to add the products to the cart. The App defaults to the Cart Screen to show the items in the cart when the user is offline.

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

- Work Manager– v 2.3.4- For scheduling background tasks to be in sync with the open cart mobile App.
- Glide- v 4.11.0 - Loading/Caching images
- Retrofit – v 2.9.0 - To retrieve / upload JSON via REST based webservice.
- Lombok- v 1.18.12- To generate constructors, getters and setters
- Espresso -v 3.1.0- Unit testing

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

- **Cloud FireStore** for storing and syncing data. Add Cloud Firestore dependencies to the App as below in the build.gradle:
    - apply plugin: `'com.google.gms.google-services'`
    - implementation 'com.google.firebase:firebase-firestore:17.1.2'
- Declare an instance of FireBaseFireStore class
- Save and retrieve data from the Firestore.

- **Google Analytics for Firebase –** Add analytics dependencies in the build.gradle as below
    - implementation 'com.google.firebase:firebase-core:16.0.4'

- Declare the com.google.firebase.analytics.FirebaseAnalytics object at the top of your activity.
- Make sure only one analytics instance is created.

- **Crashlytics –** Add the dependencies in the build.gradle as below

    - apply plugin: 'com.google.firebase.crashlytics'
    - implementation 'com.google.firebase:firebase-analytics:17.4.3'
    - implementation 'com.google.firebase:firebase-crashlytics:17.1.0'

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 new Essentials Android project using Android studio v 4.0. (use the latest compile and target SDK)
- Configure the latest gradle v 6.3
- Configure the latest android gradle plugin (v 3.6.3)
- Configure the libraries needed for the project
    - WorkManager
    - Glide
    - Lombok
    - Retrofit
    - Expresso
- Configure the Firestore, Google Analytics and Crashlytics dependencies.
- Use the Firebase assistant to configure the Firestore database so that it can be seen in the Firebase console.
- Make sure that the stable versions of the libraries are used and not alpha, beta, RC versions.

Task 2: Implement Customer Registration Screen.

- Build layout for Registration Activity
- Write code to load the layout and save the customer details to the Cloud Firestore and sync them with the opencart server.

Task 3: Implement Customer Login Screen.

- Build layout for Login Activity
- Write code to load the layout and save the login details to the Cloud Firestore and sync them with open cart server.

- Write validation to check if all the fields are entered and if the email address is in a proper format.

Task 4: Implement the Landing screen for the App.

- Code tasks with workmanager to fetch product details from Cloud Firestore. The product details are updated periodically and kept in sync with the open cart server.
- Implement the toolbar for displaying the name of the App.
- Validate the Json received through the Retrofit API.(the JSON will be received from the open cart store) and store the product details in the Cloud Firestore.
- If there are errors in JSON log the errors so that the app does not crash.
- Build the layout for the product fragment and attach to the Landing Page Activity.
- Load the products (that are in Promotion) that fit in the screen using Glide into a cardview from the Cloud Firestore.
- Display the product along with its original price, reduced price, and the product title.
- Implement code to search for desired product.
- Write code to view all the products in Promotion through an Hypherlink.
- Write code to click a particular product to view the product detail screen.

Task 5: Implement the Promotion Screen for the App.

- Build the layout for the promotion fragment and attach to the Promotion Activity.
- Load the products (that are in Promotion) that fit in the screen using Glide into a cardview.
- Use Recycler view to manage the vertical scrolling.
- Display the product along with its original price, reduced price, product title and the percentage reduced.
- Write code to search for any product that is in promotion.
- Write code to click a product to view the product detail screen.

Task 6: Implement the Product Detail Screen for the App.

- Build the layout for the Product detail activity.
- Load the product image using Glide.
- Implement a **collapsing tool bar layout** to display the name of the product when scrolling at the top
- Display the product along with its original price, reduced price, product title and the percentage reduced.
- Display if the product is in stock or not.
- Write code to view product description.
- Write code to add the products to a cart and display a toast message when product is added successfully.
- Write code to add the products to a wishlist and display a toast message when product is added successfully.

Task 7: Implement the Category Screen

- Build the layout to load categories like Appliances, Toys, Daily Needs, Toiletries etc for the Category Activity
- The category list will be obtained from the open cart store as JSON via Retrofit API and will be stored in the Cloud Firestore.
- .Build the Category activity to load the categories from the Cloud Firestore.
- Implement code to launch the products for the specific category when it is clicked.

Task 8: Implement Cart Screen

- Build the layout to load the items that are in Cart for the Cart activity
- Display the product image, title, price.
- Write code to increment/decrement/delete items in the cart.
- Write code to move the product from the cart to Wish list.
- Write code to display the total at the bottom along with the number of items.
- Write code to allow the user to do a checkout and sync order details to opencart.

Task 9: Implement Wish list Screen

- Build the layout to load the items that are in Wish list for the WishList activity
- Display the product image, title, price.
- Write code to move the product from the Wish list to cart

Task 10: Implement Delivery Address Screen

- Build the layout to load the delivery address screen for the DeliveryAddress activity
- Display the saved delivery address along with the delivery date.
- Write code to launch AddDeliveryAddress activity to capture new delivery address.
- Write code to launch CardDetailsActivity to capture new card details.

Task 11: Implement Add Delivery Address Screen

- Build the layout to load the add delivery address screen for the AddDeliveryAddress activity
- Write code to capture the delivery address and save to Cloud FireStore.
- Add validation to capture the necessary fields.
- Sync delivery address to Opencart store.

Task 12: Implement Card Details Screen

- Build the layout to load the card details screen for the CardDetails activity
- Write code to capture the card details address and save to Cloud FireStore.
- Add validation to capture the necessary fields.
- Sync card details to Opencart store.

Task 13: Implement Bottom Navigation bar

- Add the BottomNavigationView to the layout file for the NavigationBarActivity.
- Implement the home,cart,category,wishlist and customer details as menu items.
- Launch the LandingPageActivity when the home icon is clicked.
- Launch the CartActivity when the cart icon is clicked.
- Launch the WishlistActivity when the wishlist icon is clicked.
- Launch the CustomerDetailsActivity when the customer details icon is clicked.
- Launch the CategoryActivity when the category icon is clicked.

Task 14: Implement Customer Details screen

- Design the layout file for the CustomerDetailsActivity.
- Display the personal details of the customer along with the ability to edit them.
- Display the delivery address of the customer.
- Write code to launch the Order Details activity when Order Details is clicked.

Task 15: Implement Order Details screen

- Design the layout file for the OrderDetailsActivity.
- Display the Order number, delivery date and status.

Task 16: Implement Menu bar

- Design the layout file for the MenuActivity.
- Implement the home,cart,category,wishlist,customer details,orders,login,register,logout as menu items.
- Launch the LandingPageActivity when the home menu is clicked.
- Launch the CartActivity when the cart menu is clicked.
- Launch the WishlistActivity when the wishlist menu is clicked.
- Launch the CustomerDetailsActivity when the customer details menu is clicked.
- Launch the CategoryActivity when the category menu is clicked.
- Launch the OrderActivity when the order menu is clicked.
- Launch the LoginActivity when the login menu is clicked.
- Display a toast that the user has logged out successfully when the logout menu is clicked.
- Launch the RegisterActivity when the register menu is clicked.

Task 17: Home widget for the App

- Design the Widget layout for the App
- Code the class for the Widget Provider
- Code tasks with workmanager to display/update the promoted product in the home widget.
- Load the product which is in promotion to the widget.
- Clickig the widget should launch the App.

Task 18: Configure signing configuration for the App

- Generate the keystore and protect it with a password.
- Refer the keystore using the relative path in the project in the module build.gradle

- Configure the build type as Release for the keystore in build.gradle.

Task 19: Create tests

- Use espresso to code UI tests for the relevant screens.
- Make sure that the unit test passes.

Task 19: Project cleanup

- Make sure all the strings in the project comes from the strings.xml file.
- Make sure each service imported in the build.gradle is used in the app.
- Make sure App theme extends AppCompat.
- Make sure App enables RTL layout switching on all layouts.
- App includes support for accessibility. That includes content descriptions, navigation using a D-pad, and, if applicable, non-audio versions of audio cues.

Task 19: Implement Firebase services

Implement code for Google Analytics for Firebase so that the app analytics can be seen.

Implement code for Crashlytics so that the app can provide real time crash reporting.

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**"