[20-2-2019]

VERSION: 1.0

<E-MART>

SOFTWARE DESIGN DESCRIPTION DOCUMENT

SARMAD MAQBOOL
BSCS 7TH MORNING
ROLL # IU15M2BA010

Revision History

Date	Description	Author	Comments
<1-1-2019>	<Version 1>	<Sarmad Maqbool>	<First Revision>
<8-1-2019>	<Version 2>	<Sarmad Maqbool>	<2nd Revision>
<18-1-2019>	<Version 3>	<Sarmad Maqbool>	<3rd Revision>
<22-1-2019>	<Version 4>	<Sarmad Maqbool>	<4th Revision>
<03-2-2019>	<Version 5>	<Sarmad Maqbool>	<5th Revision>

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	Dr. Rao Akmal khan	Supervisor, CSIT 21306	<20-2-2019>

1. INTRODUCTION

1.1 Purpose

The purpose of the Software Design Document is to provide a description for the design of a system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to built. This SDD describes the specifications, requirements and architecture of E-MART, an e-commerce solution for home based consumer. It explains the functional features of the application, along with interface details, design constraints and related considerations such as data description, human interface design , screen objects and actions and a comprehensive overview of the system. The previous document (SRS) should be kept in mind while writing the software document design. Such as functional and non-functional requirements, UML diagrams design constraints and many other supporting information.

1.2 Scope

This Software Design Description (SDD) describes the detailed structure of the components of the E-MART application and the precise implementation details required to satisfy the requirements as specified in the Software Requirements Specification (SRS). E-MART will be an android-based software system written in Java and SQLite to aid daily consumer and seller in sales and purchases with regard to simplifying and speeding up the process of selection, ordering and purchasing goods for customers as well as managing a database of users and a database of products for store owners through a conveniently designed Graphical User Interface which will utilize a user-friendly intuitive design approach. The system will be designed to be used by daily consumer with little or no experience in using mobile phone applications and by store owners who will be trained to use E-MART. To aid users, documentation will be provided in the form of a user manual which will contain a detailed description of all system functions.

1.3 Overview

The project on which we are going to work upon is an ecommerce app named as E-Mart. This document is divided into 8 sections and 13 sub-sections.

Each section or chapter consists of some sub-sections which are listed below.

1. INTRODUCTION

2. SYSTEM OVERVIEW

3. SYSTEM ARCHITECTURE

4. DATA DESIGN

5. COMPONENT DESIGN

6. HUMAN INTERFACE DESIGN

7. REQUIREMENTS MATRIX

8. APPENDICES

1.4 Reference Material

List any documents, if any, which were used as sources of information for the test plan.

1.5 Definitions and Acronyms

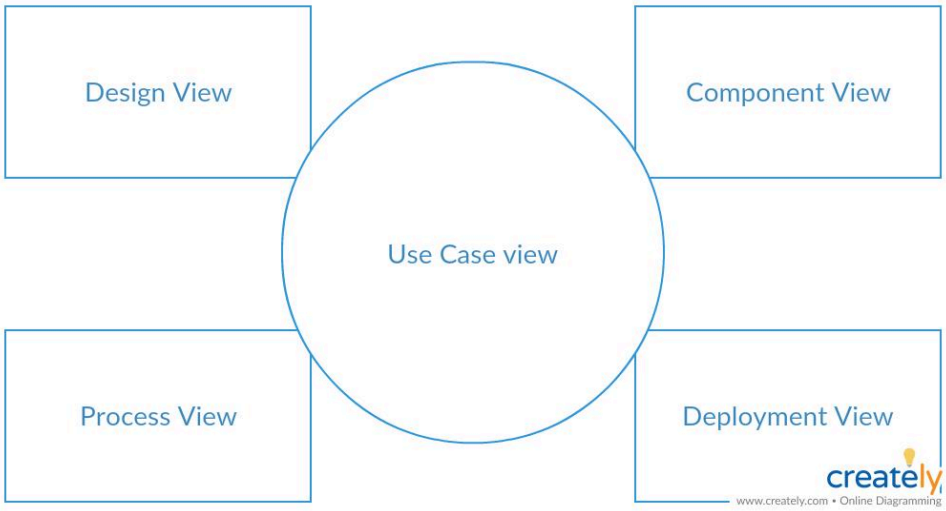
Provide definitions of all terms, acronyms, and abbreviations that might exist to properly interpret the SDD. These definitions should be items used in the SDD that are most likely not known to the audience.

2. SYSTEM OVERVIEW

The system general overview has already been defined in the 1st chapter in subsections overview and definitions and acronyms.

3.Architectural Representation

3.1Architectural Views



Modeling, implementing, and documenting a system requires that the system be viewed from different perspectives. Because of this, the architecture of the EMART will be represented in a five view approach: Use Case View, Design View, Process View, Component View, and Deployment View. The following is a brief description for each of the views:

Use Case View: the main purpose of the use case view is to define main drivers of the system, which are the system requirements.

Design View: this view contains any system definitions as well as class and object diagrams which depict the services that the system will provide to its end-users.

Process View: this view will display the processes that form the systems' mechanism. These will be represented as collaboration, sequence, and activity diagrams.

Component View: this view will include system and user interface specifications, meaning, the different components that make up the system.

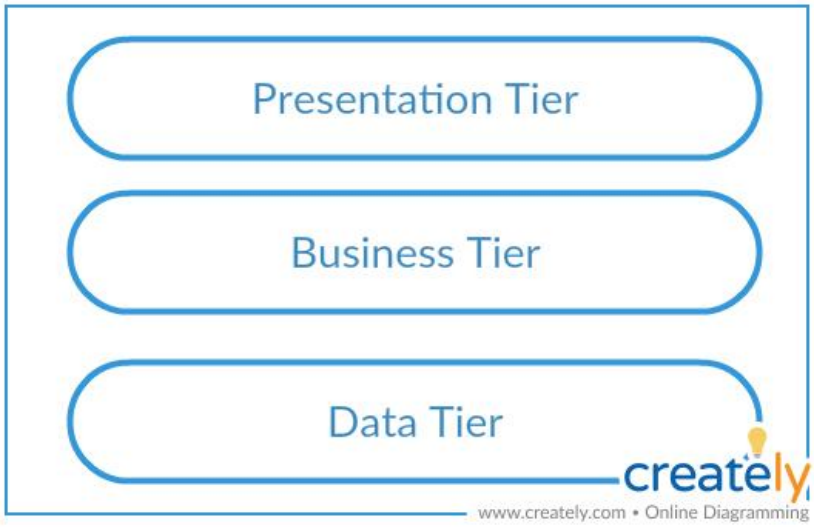
Deployment View: this view will depict how the different systems' hardware nodes will come to life together as well as how each of the hardware nodes will be installed and deployed.

3.2Architectural Design Patterns

The design pattern used to create the EMART is the MVC (Model View Controller) design pattern. The MVC design pattern clearly separated the web application's behavior, presentation and control. The modularity of this design pattern allows for easier code reuse, more centralized control, bugs easier to track down and code easier to modify. The presentation, or view, of the EMART has been implemented keeping in mind the Model 2 usage pattern, which makes use of servlets as front controllers and maps incoming request to specific operation and selects views based on the model and session state. In some steps of the application the Model 1 pattern, in which a servlets is not used as the front controller, is also used.

3.3Architectural Style

As with any other things, a style may be used to satisfy any functional, non-functional or aesthetic needs in a software system. The EMART, in particular, follows the three-tier architectural style: presentation tier, business tier, and data tier. The following is a simple description of what will be included in each of the tiers:



Presentation Tier: used to format and present the information to the user.

Business Tier: used to implement the logic that will drive the system and the reason why the system exists.

Data Tier: in charge of storing the data (databases) and other external services that the system may use.

3.4Architectural Process

EMART follows the Rational Unified Process (RUP), whose goal is to enable the production of the highest quality software that meets end user needs within a predictable schedule and budget. The RUP is an iterative process divided into four phases: inception (the establishment of the projects business case), elaboration (establishment of project plan and system architecture), construction (the system implementation), and transition (system deployment).

During the inception phase, the business case and vision documents will be defined for the EMART. The business case includes the criteria needed for the successful development of the system, a budget estimate of the resources needed to complete the system, and a schedule of the major milestones. The vision document defines the project's problem to be solved, the project's purpose and the project's major stakeholders. During this phase, a PowerPoint presentation for the entire system will be created which will serve as a proof-of-concept for the system. This presentation will be reviewed by the major stakeholders for soundness and feasibility.

During the elaboration phase, the EMART will have an easy to understand software architectural foundation and an implemented project plan. As previously described, the architectural foundation is composed of a set of UML diagrams that entirely describes the system's functionality.

During the construction phase, the EMART will be incrementally and iteratively developed and tested. The development implies the complete coverage of the software requirements by following the different UML diagrams defined in the previous phase. At the end of this phase, the EMART should be a completely designed, implemented, and tested system that is ready to be deployed for use.

Deployment of the EMART will occur during the transition phase where the system will be placed on a server for the use of Consumer, Seller, and database administrators. Once the beta release of the EMART has been released, any system issues that arise will be logged and corrected and a new system version will be released. This process will continue until all new and legacy system requirements have been satisfied.

OBJ OBJ

3.5Design Rationale

We wanted to build a system such that available from everywhere. Therefore, we designed a Client-Server system. Moreover, we don't want to restrict the user to his/her device storage size. So, we designed system such that data is kept in a public database.

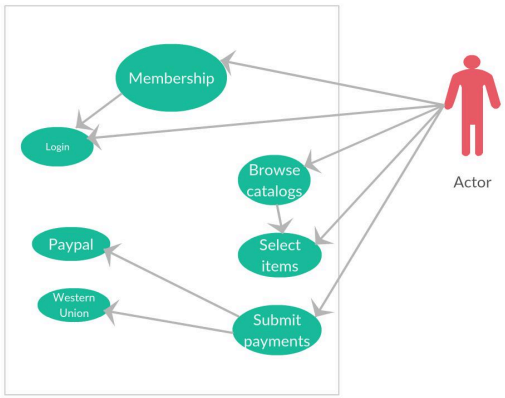
We chose Android for mobile platform because approximately 80% of smartphone in market uses Android for operating system. We wanted to use a RESTful web service because a REST service is: Platform-independent (you don't care if the server is Unix, the client is a Mac, or anything else), Language-independent (C# can talk to Java, etc.), Standards-based (runs on top of HTTP), and Can easily be used in the presence of firewalls.

3.6 Architectural View Decomposition

3.6.1 Use-Case View

The general functionality of the EMART can be seen in the following use case diagram:

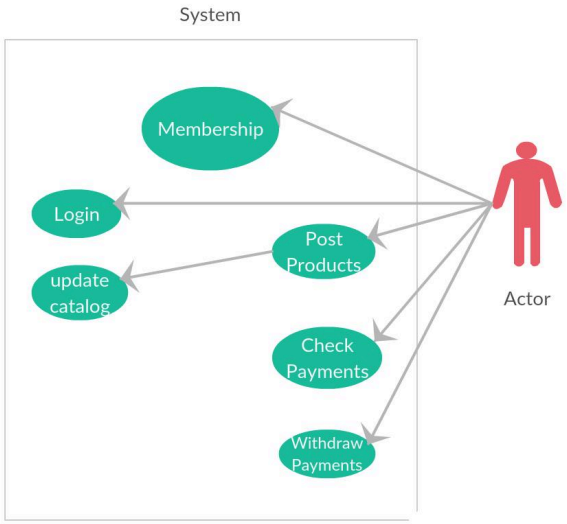
User side



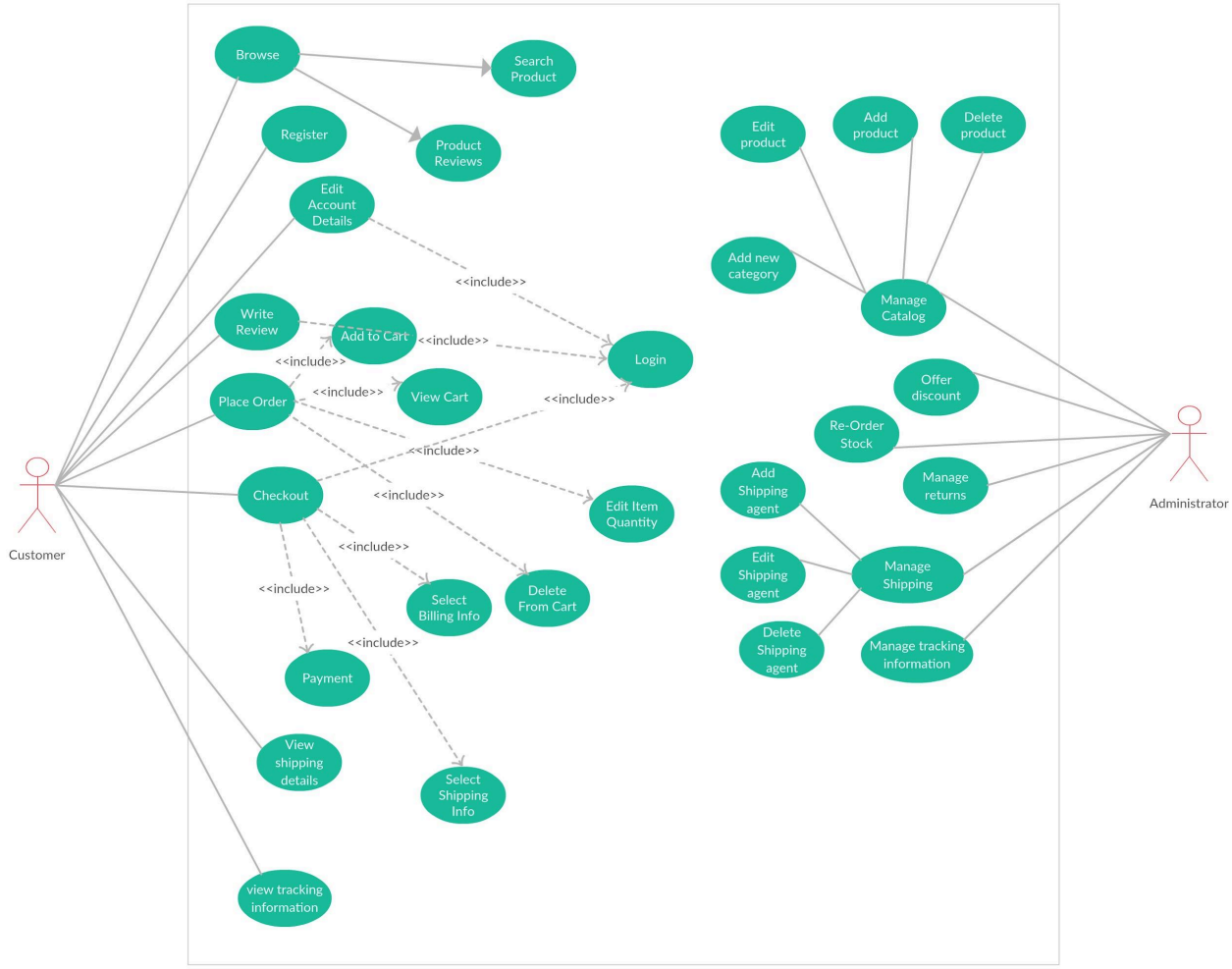
The use case for the user side clearly tells us that the user will firstly register as a member then the user will login to the system. After that the user will take a tour of the app (i.e how it works). The user will browse through catalogs and select his desired category. There the user

will browse items and select a product to ship to his address. Then the user will be asked a payment method to complete his order. After selecting a payment method and completing the formalities of address and payment method, the order will be shipped to user's given address.

Seller Side



The seller will have to register himself as a member the he will be prompted to logged in. after that the seller will post his products on catalog. He will also update the catalogs if there are new arrivals or stock was empty previously. Furthermore the seller can check for payments from the user, if the user have paid for the item then the seller will make item ready to dispatch. The seller can also withdraw his payments by the desired payment method.



General Use Case

1st Actor (User)

This is the comprehensive use case of the entire app and there are two actors. First is user and the second one is administrator or seller. User will firstly register as a member then the user will login to the system. After that the user will take a tour of the app (i.e how it works). The user will browse through catalogs and select his desired category. There the user will browse items and select a product to ship to his address. Then the user will be asked a payment method to complete his order. After selecting a payment method and completing the formalities of address and payment method, the order will be shipped to user's given address. The details enhanced in this

case for user side are that he can update his account and manage account. He can avail discount on his purchases. The user can write a review about the product and he can also also view the tracking information of the item. Moreover a user can also cancel his order if he is not sure about making the order.

2nd Actor (Administrator)

The Administrator will post products on catalog. He will also update the catalogs if there are new arrivals or stock was empty previously. Furthermore the admin can check for payments from the user, if the user have paid for the item then the seller will make item ready to dispatch. The administrator will also add a new category if new items or category is introduced. Moreover the admin will also allocate a shipping agent to the specific order to be delivered on door-step.

4. DATA DESIGN

4.1 Data Description

4.1.1 Data Objects

Item:

id: Identity number given from the database for each item. This attribute is unique for any item.

title: Title of the item.

manufacturer: manufacturer of the item.

description: A small description/summary of the item. The user will be able to edit this text.

cover_image: Cover image of the item.

producer: The company that has produced the item.

manufacture date: The date of manufacturing.

category: The general type of the item.

rating: Average score of the item rated by the users.

User:

id: Identity number given from the database for each user. This attribute is unique for any user.

name: The name and surname of the user.

username: User's nickname for the application.

password: The user's password.

e-mail: E-mail of the user that will be used for the communication information.

phone_number: Phone number of the user that will be used for the communication information.

description: A brief description where the user mentions about his/herself

profil_photo: A picture that may be used to face detection login.

Items-of-User :

user_id: Id number that will be taken from the user table indicating the item to booked.

item_id: Id number taken from the item table indicating which item is to be booked by the user.

is_favourite: The user has selected item as favourite item.

owner_id: Id of the user that sells the item.

History:

id: Primary key.

type_id: Process type id.

date_time: When the process occurred.

description: Detailed explanation about process.

user_id: This is the user id of owner of history entry.

Process_types :

id: Primary key

type: Name of the process. It can be booking, purchasing, adding to cart, deleting, selling, buying, viewing, saving and etc.

4.2 Data Dictionary

USER	**Parameters**
Name:	String
ID:	Int
Password:	Char
Address:	String

ADMIN	**Parameters**
Name:	String
ID:	Int
Password:	Char
Address:	String

USER	**Parameters**
Name:	String
ID:	Int
Password:	Char
Address:	String
Phone no:	Char

Product	**Parameters**
Name:	String
ID:	Int
Price	Char

Payment	**Parameters**
Name:	String
Customer ID:	Int
Product ID:	int
Password	Char
Address:	String

5. COMPONENT DESIGN

5.1 User

Class Name	**User**
Brief description	It represents the user components; include the attributes name, id, address,. And methods are login, registration, view product.
Attributes(fields) Name:	Name: Name of the user. Id: User id to identify the user. Address: Address of the user.
Methods (operations)	**Description**
Login()	Method for Login user.
Registration()	Method for Registration the user.
View Product()	Method for view product of the user.

Psuedo Code	Add the User Details(name, id, address,)
	{
	String name;
	Int id;
	String address;
	}
	1. Validation is performed.
	2. Database connection is created.
	3. Insert record in the user table.

5.2 Customer

Class Name	**Customer**
Brief description	It represents the customer; include the attributes name, id, credit card info, shipping, phone no. And methods are login, registration,feedback.
Attributes(fields) Name:	Name: Name of the customer. Id: customer Id. CreditCardInfo: view the credit card of the customer. Shipping Info: Customer shows the shipping info. Phone No: Phone no of customer.
Methods (operations)	**Description**
Login()	Method for Login customer.
Registration()	Method for Registration the customer.
View Product()	Method for view product of the customer.

Psuedo Code	Add the customerDetails(name, id, phone no)
	{
	String name;
	Int id;
	Int phone no ;
	}
	1. Validation is performed.
	2. Database connection is created.
	3. Insert record in the user table.

5.3 Admin

Class Name	**Admin**
Brief description	It represents the Admin;include the attributes name, id, gender, address, phone no. And methods are login, delete product, update product, update profile.
Attributes(fields) Name:	Name: Name of the Admin. Id: Admin Id. Gender: Male\Female Admin. Address: Address of the Admin. Phone No: Phone no of Admin.
Methods (operations)	**Description**
Login()	Method for Login admin.
Delete-product()	It is a method for Delete product.
Update-product()	Method for update product.
Update-profile()	Method for update profiles the admin.

Psuedo Code	Add the Admin Details(name, id, gender ,address, phone
	no)
	{
	String name;
	Int id;
	String gender;
	String address;
	Int phone no ;
	}
	1. Validation is performed.
	2. Database connection is created.
	3. Insert record in the Admin table.

5.4 Registration

Class Name	**Registration**
Brief description	It represents the Registration for u include the attributes name, id, password, address, phone no. And methods are Get-name, Set-name.
Attributes(fields) Name:	Name: Name of the User. Id: User Id. Password: Password of the user. Address: Address of the User. Phone No: Phone no of User.
Methods (operations)	**Description**
Get-name()	It is a method to get name of the user.
Set-name()	It is a method for setting the user name.

Psuedo Code	Add the Registration Details(name, id, password address, phone no)
	{
	String name;
	Int id;
	Char password;
	String address;
	Int phone no ;
	}
	1. Validation is performed.
	2. Database connection is created.
	3. Insert record in the Admin table.

5.5 Product

Class Name	**Product**
Brief description	It represents the Product of user; include attributes name, type, and size. And methods are view, update, delete, add.
Attributes(fields) Name:	Name: Name of the Product. Type: Type of Product. Size: Size of Product.
Methods (operations)	**Description**
View()	Method for viewing products.
Update()	Method for updating the products.
Delete()	Method for deleting the products.
Add()	Method for adding the products.

Psuedo Code	Display Product Details(name, type, size)
	{
	String name;
	Char type;
	Intsize ;
	}
	1.Input product name.
	2. Database connection is created.
	3. Search in product table.
	4. Display all match products.

5.6 Shopping Cart/ Select items

Class Name	**Shopping Cart/ Select items**
Brief description	It represents the shopping cart of user; include attributes Cart id, product id, quantity. And methods view Cart product, update Cart product, update Cart Product,add Cart Product.
Attributes(fields) Name:	Cart ID: Id of the cart. Product Id:Id of the Product. Quantity: Quantity of the Product.
Methods (operations)	**Description**
View cart products()	Method for viewing cart products.
Update cart products()	Method for updating cart products.
Delete cart products()	Method for deleting cart products.
Add to cart()	Method for adding cart products.

Psuedo Code	Display Product Details(Cart id , product id, quantity)
	{
	Int cart id;
	int product id;
	int quantity;
	}
	1.Input product id.
	2. Database connection is created.
	3. Search in product table.
	4. Display all match products.

5.7 Payment

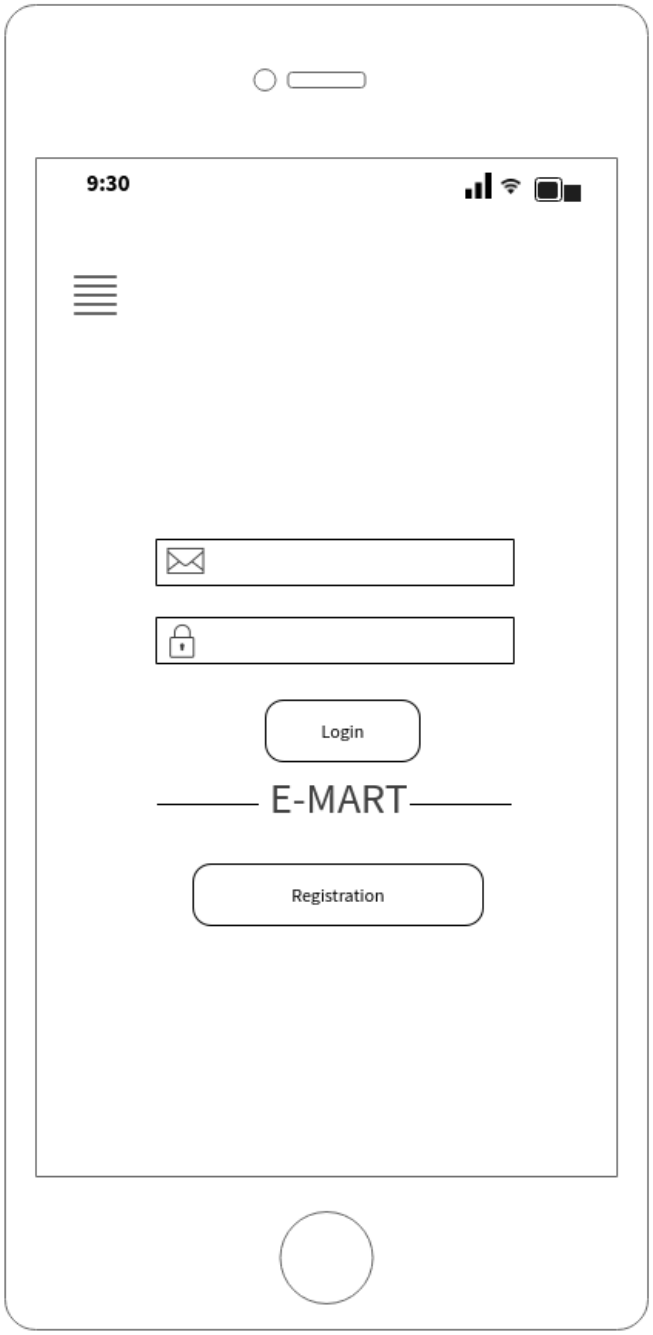
Class Name	**Payment**
Brief description	It represents the payment of user; include attributes name, id, and amount. And method payment details.
Attributes(fields) Name:	Name: Name of the Product Id: Product Id. Amount: Amount of the Product.
Methods (operations)	**Description**
PaymentDetails()	Method for providing payment details

Psuedo Code	Add the PaymentDetails(name, id, Amount)
	{
	String name;
	Int id;
	Int amount;
	}
	1. Validation is performed.
	2. Database connection is created.
	3. Insert record in the Admin table.

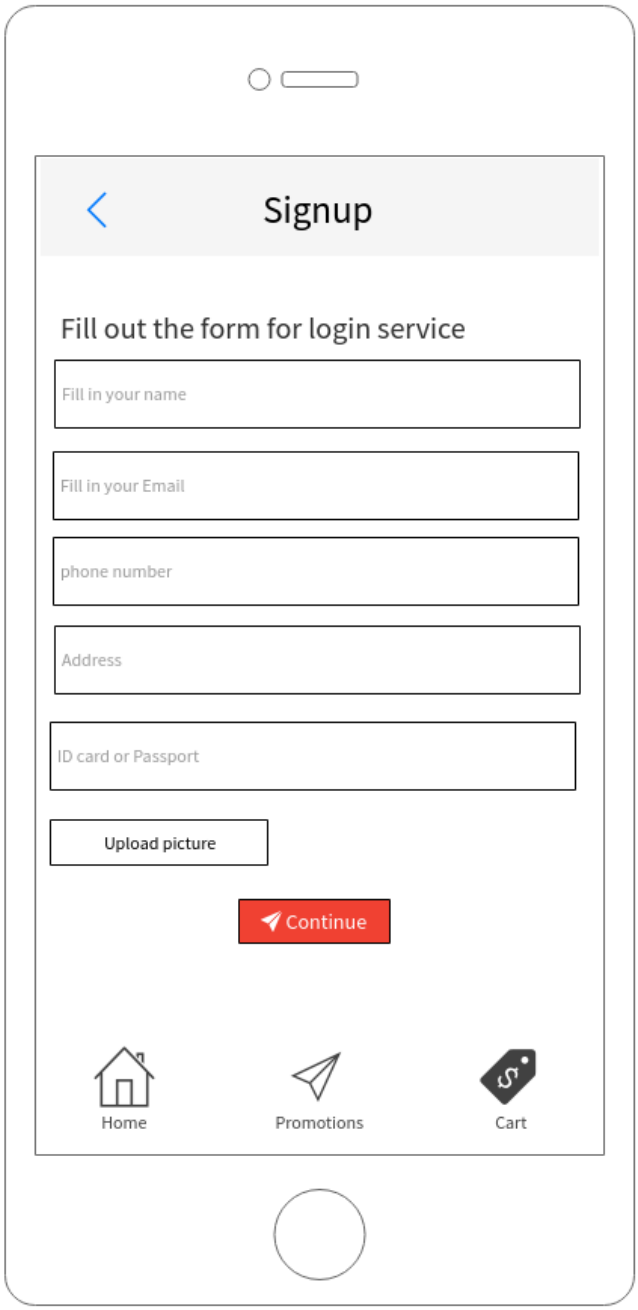
6. HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

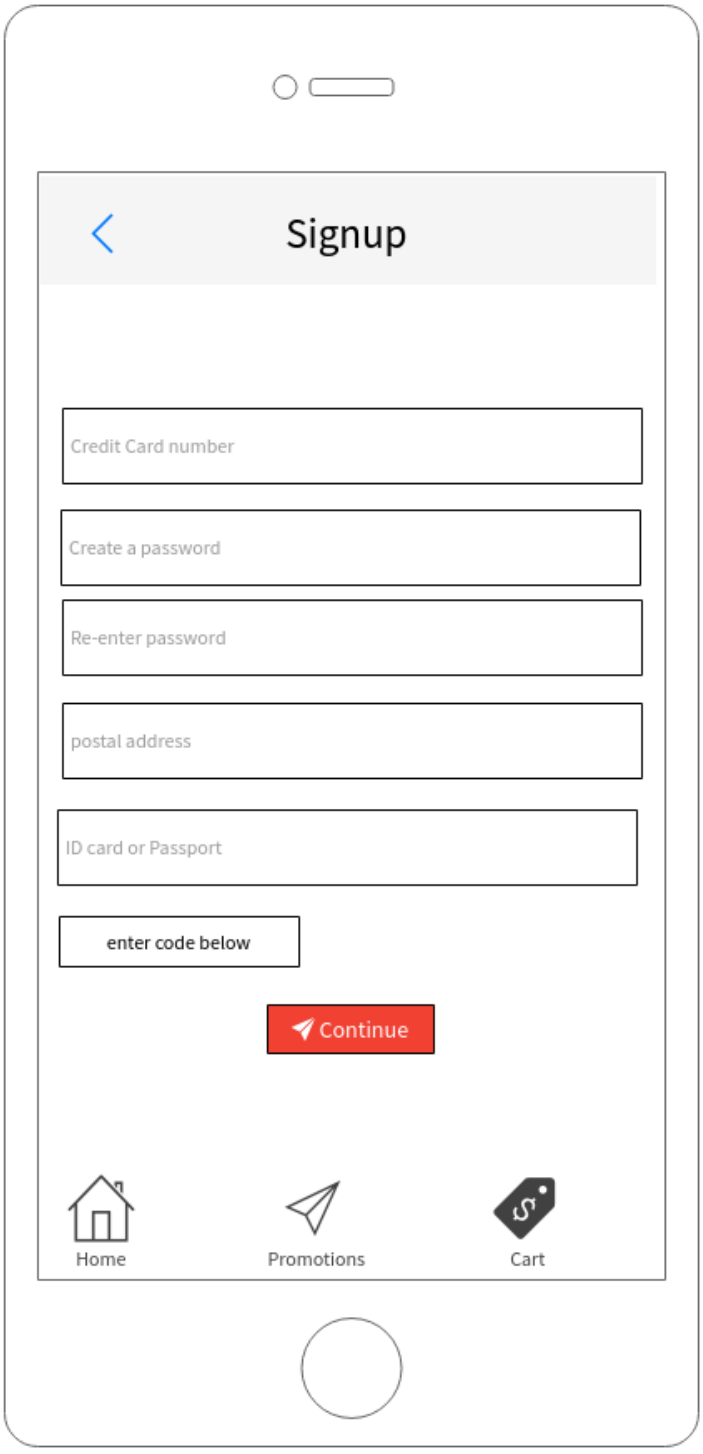
Login



Signup

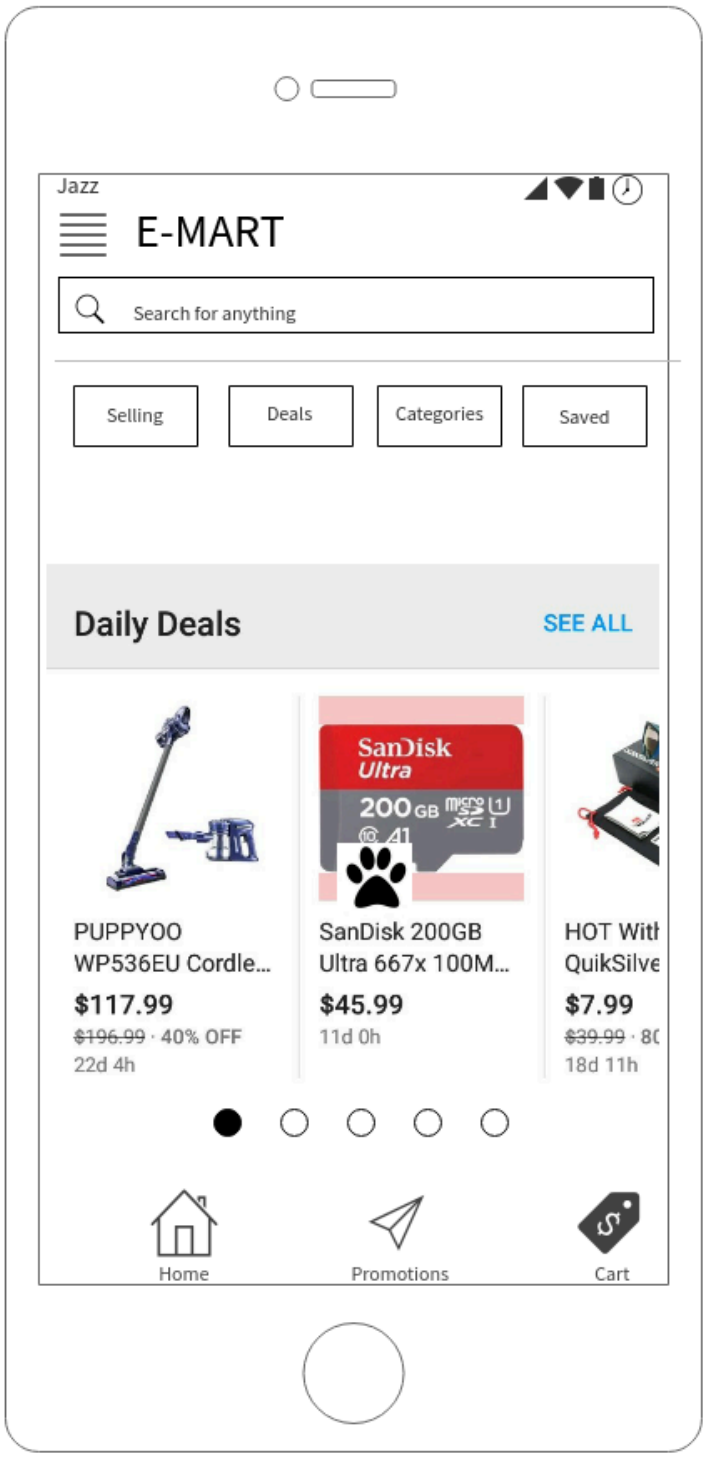

Signup
Fill out the form for login service
Fill in your name
Fill in your Email
phone number
Address
ID card or Passport
Upload picture
Continue
Home
Promotions
Cart

More detailed signup

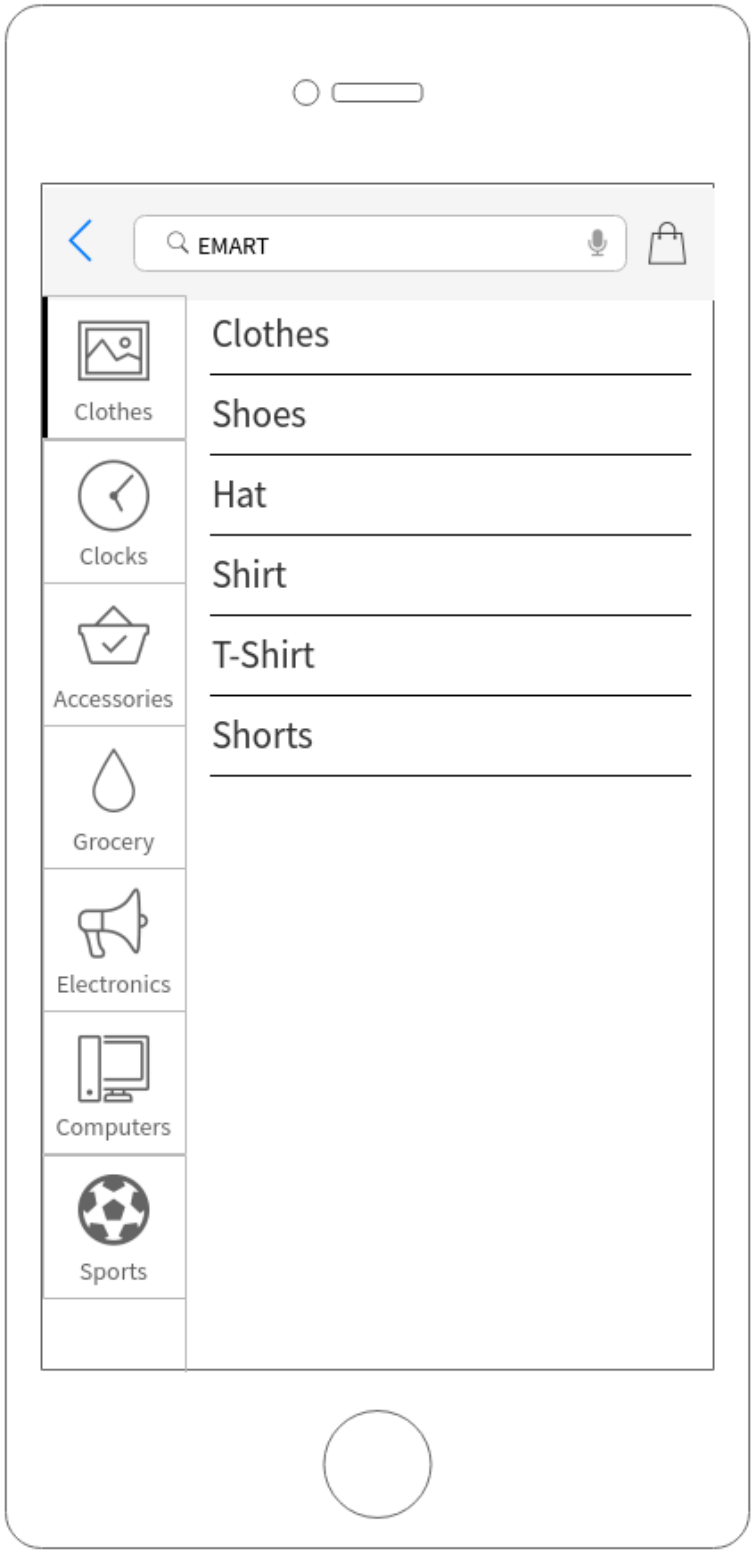


6.2 Screen Images

Home

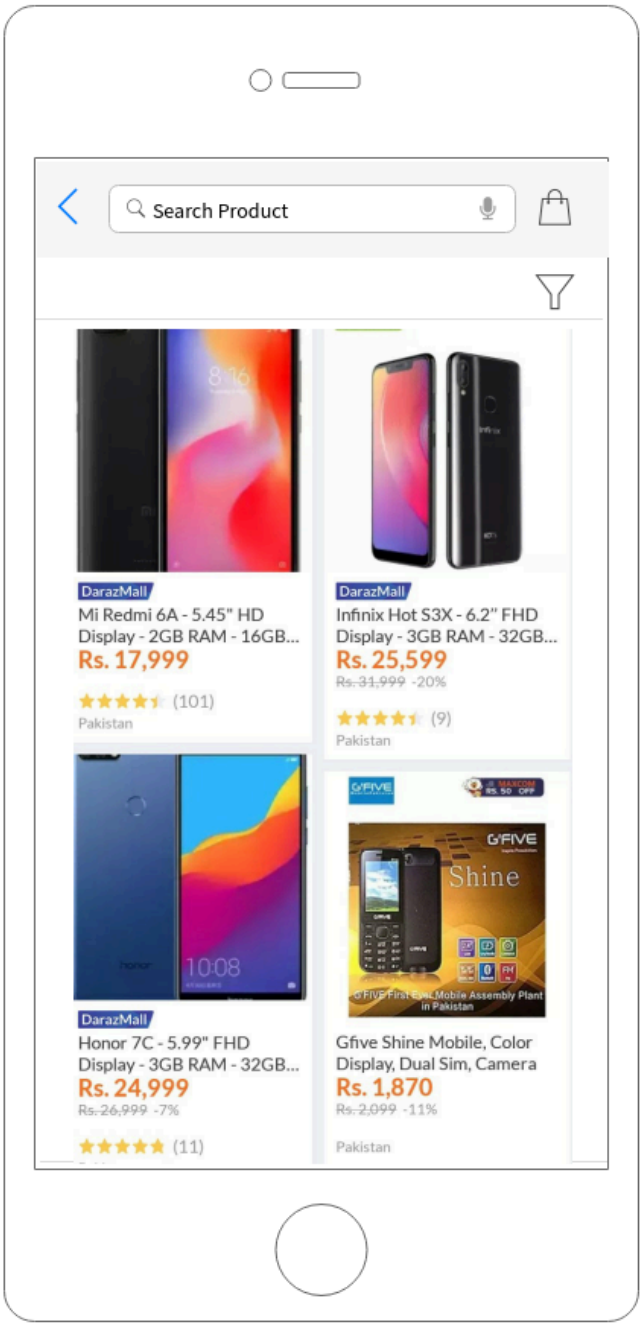


Categories

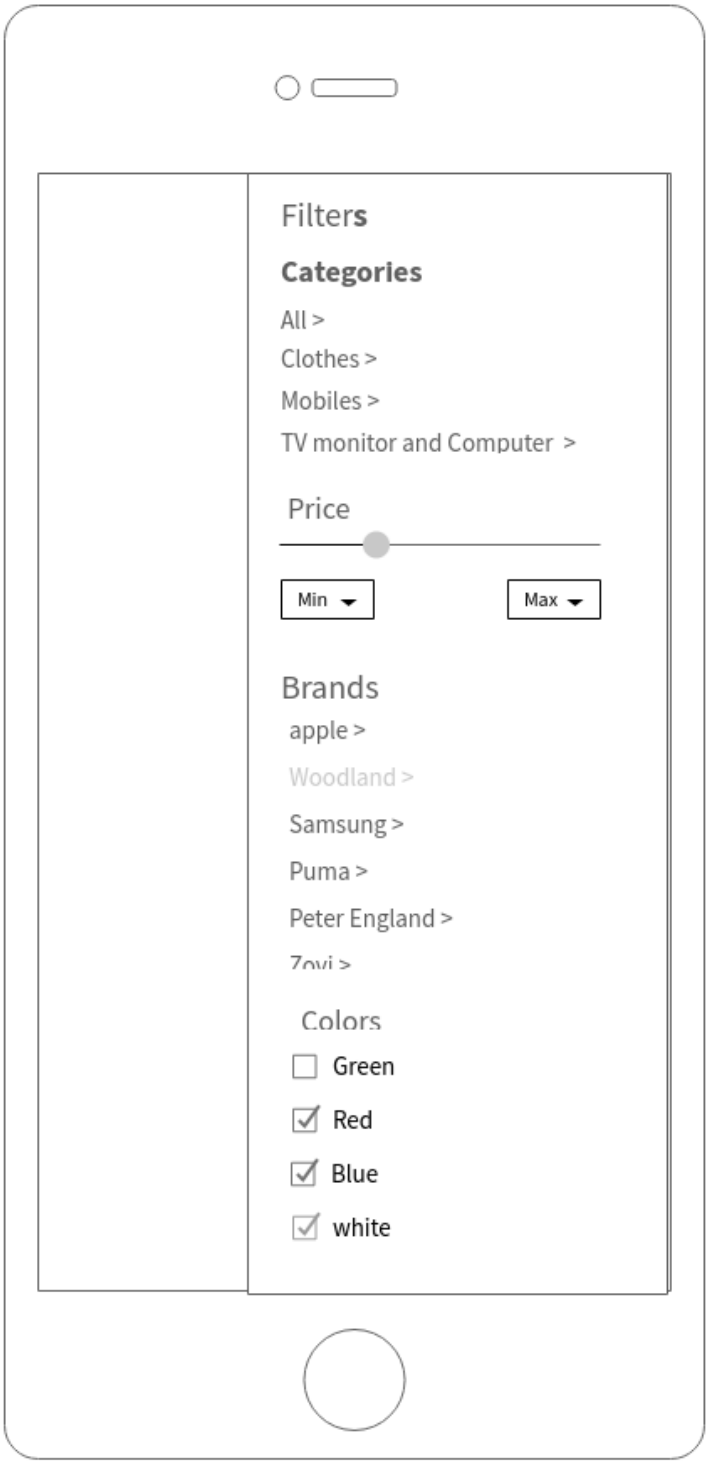

EMART
Clothes
Clocks
Accessories
Grocery
Electronics
Computers
Sports
Clothes
Shoes
Hat
Shirt
T-Shirt
Shorts

6.3 Screen Objects and Actions

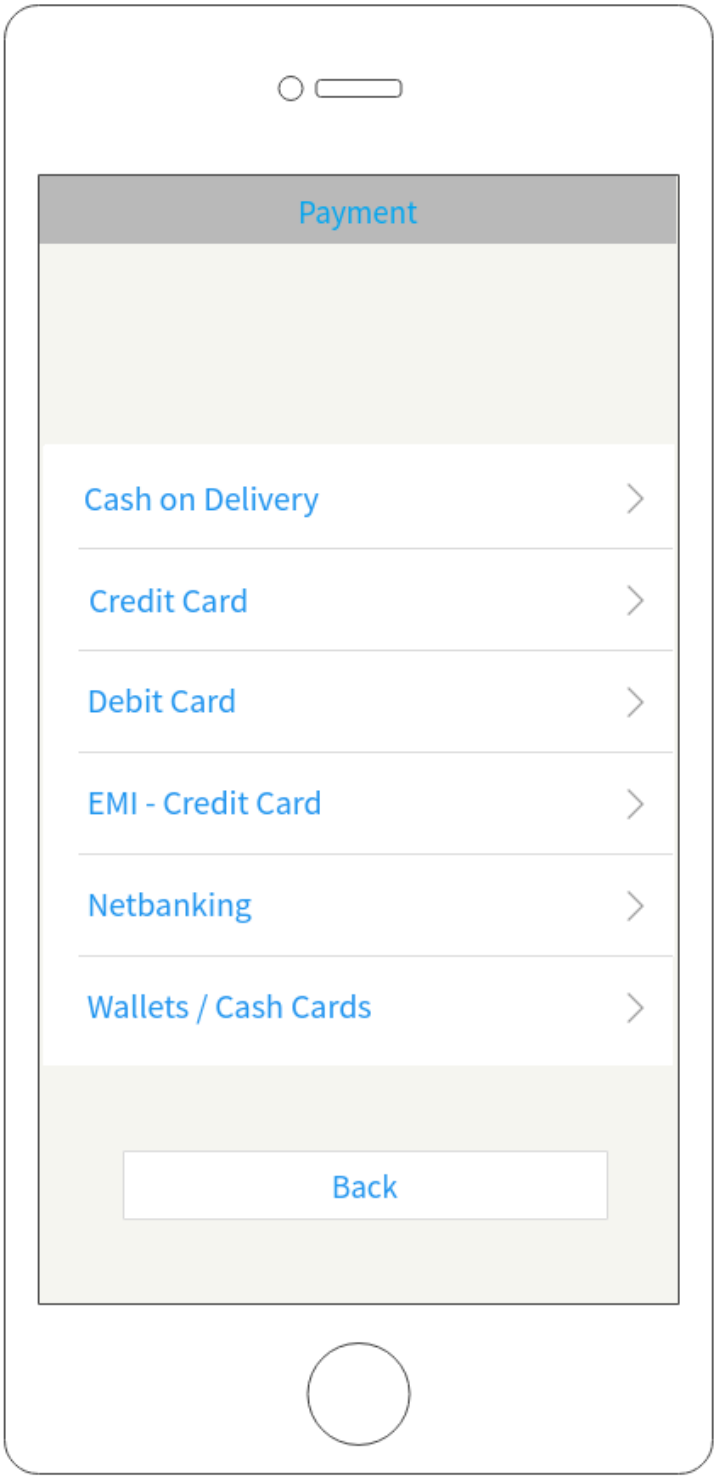
Search



Filters


Filters
Categories
All >
Clothes >
Mobiles >
TV monitor and Computer >
Price
Min
Max
Brands
apple >
Woodland >
Samsung >
Puma >
Peter England >
Zovi >
Colors
Green
Red
Blue
white

Payment


Payment
Cash on Delivery
Credit Card
Debit Card
EMI - Credit Card
Netbanking
Wallets / Cash Cards
Back

7. APPENDICES

This section is optional.

Appendices may be included, either directly or by reference, to provide supporting details that could aid in the understanding of the Software Design Document.