(Team 4)

(Switchify)

Software Design Document SDD (version 2.0)

Switchify, an exchange items service.
Version 2.0.0 approved

Prepared by:

Seif Rady
Mohamed Basuony
Omar Elsayed
Abdallah Abdelnaby

Date: (17 November 2020)

		Software Design Document
TABLE OF CONTENTS		

1. INTRODUCTION

1.1 Purpose

This Software Design Document (SSD) is created to explain the architecture and system design of Switchify, a platform to exchange and trade items. The following document's purpose is to present the general description of the system, the systems architecture, how data is dealt with and stored. It also demonstrates how users interact with the system and use its functionalities, and, finally, how the requirements of the system are met through our design and implementation.

1.2 Scope

Switchify is an application that allows users to trade items that they no longer need/use. It encourages users to circulate products that are already available in the marketplace. Instead of buying, using, and then disposing, users can now buy, use, and exchange. The project is created for academic purposes. Therefore, the scope of this project is defined by the course requirements and its deadlines. The goal of this project is to apply the learned concepts in our course and to get the hang of developing software with a team of developers. The project benefits the developers in their learning process and, hopefully, the users as they recycle their belongings and enjoy their acquisition of new products.

1.3 Overview

This document is created to facilitate analysis, planning, implementation, and decision-making. The main objective of writing this document is to explain how the software product or a feature will be built to meet a set of technical requirements. It also provides the product's data design, architecture design, interface design, components design, and the overall requirements that are met with the implementation.

1.4 Reference Material

The following tests tools and information centers will be used to test the overall functionality of Switchify:
https://firebase.google.com/products/test-lab/?gclid=CjwKCAiAkan9BRAqEiwAP9X6UZLpQ0S1k5-h3OFgpXtoi2S5LioiOX-0HSUlNUOIXJfBWhpLtNnK0xoCsZ8QAvD_BwE

https://infinum.com/the-capsized-eight/10-app-testing-principles

2. SYSTEM OVERVIEW

Switchify is a prototype Computer Software that shall aid a user in supporting the circular economy system through switching items that he/she does not want anymore with other items that he might use from other users.

A circular economy is an alternative to a traditional linear economy (make, use, dispose) in which we keep resources in use for as long as possible, extract the maximum value from them whilst in use, then recover and regenerate products and materials at the end of each service life.

Switchify supports communication between different users through chat features and allows users to put their items in suitable categories and estimate their values through a points system. Users can post pictures,videos and descriptions for their products.

No prior software development exists for this program.

Switchify will be available as a website and android mobile application and shall be used by any person interested in circular economy or wants to replace any of his old items, in an independent manner without operations or maintenance support from the developer.

Switchify version being developed shall be in a Beta state, not suitable for general release, but possessing suitable stability for use by testers under some instructions provided.

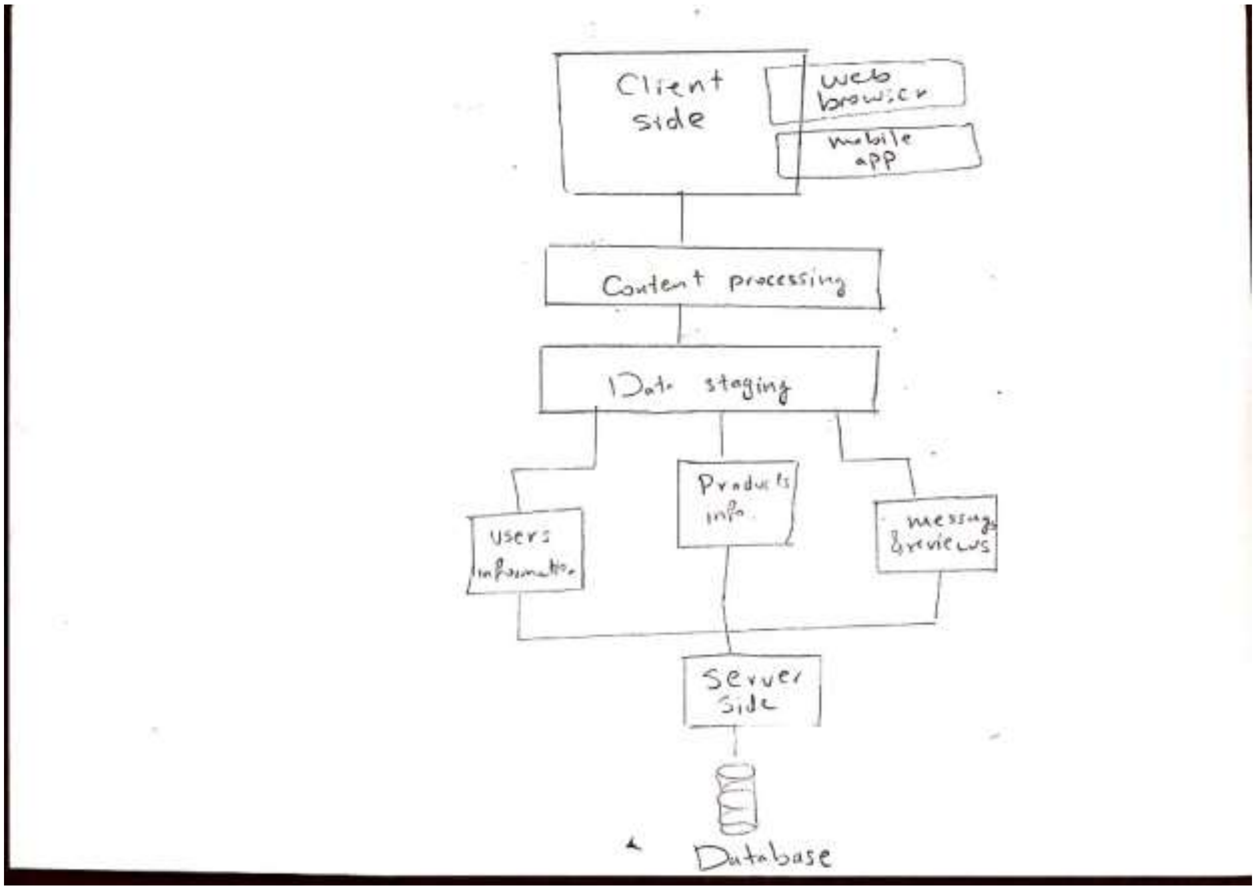
Switchify is being developed by a group of four Computer Science and Engineering students at The American University in Cairo. The development is part of the Software engineering course project for Fall 2020 at The American University in Cairo. The effort is supervised by the course instructor.

3. System Architecture

3.1 Architectural Design

In this section, an explanation of a high level systems and subsystems will take place as well as the relationship between those models, the used pattern, and some other architecture decisions related to the project.

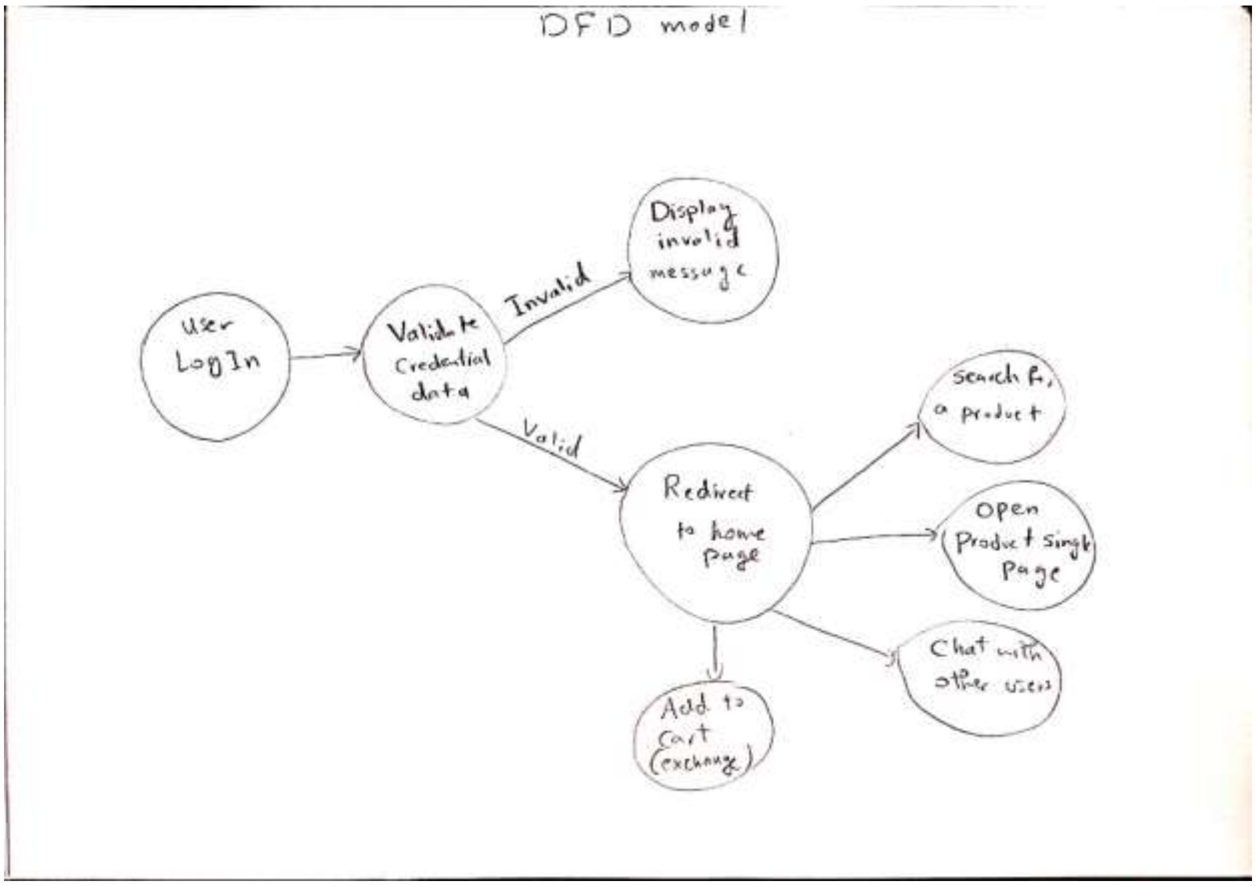
Architectural model:



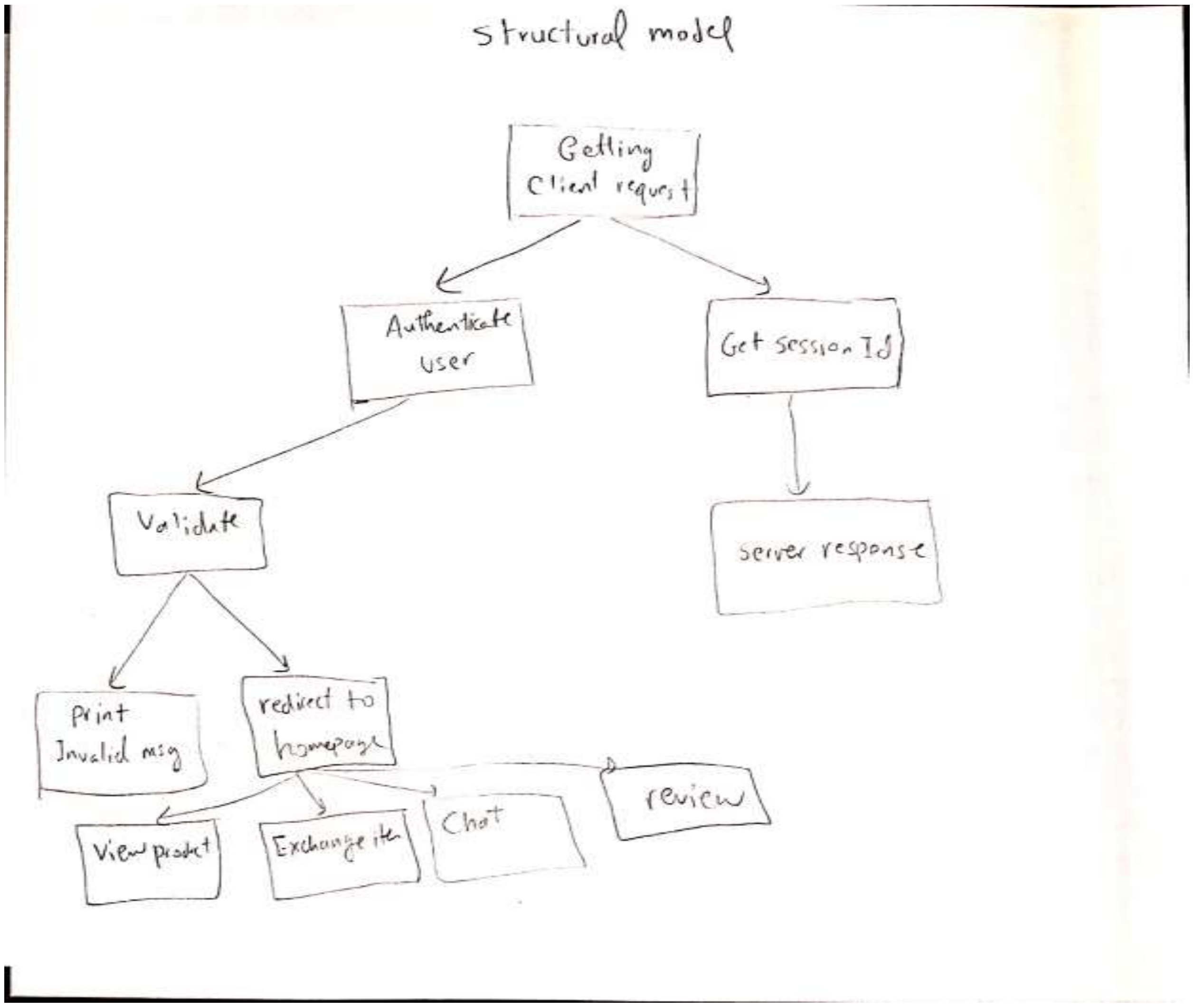
3.2 Decomposition Description

In thi section, a top level data flow diagram and structural diagrams will be preseneted to explain more about the system architecture.

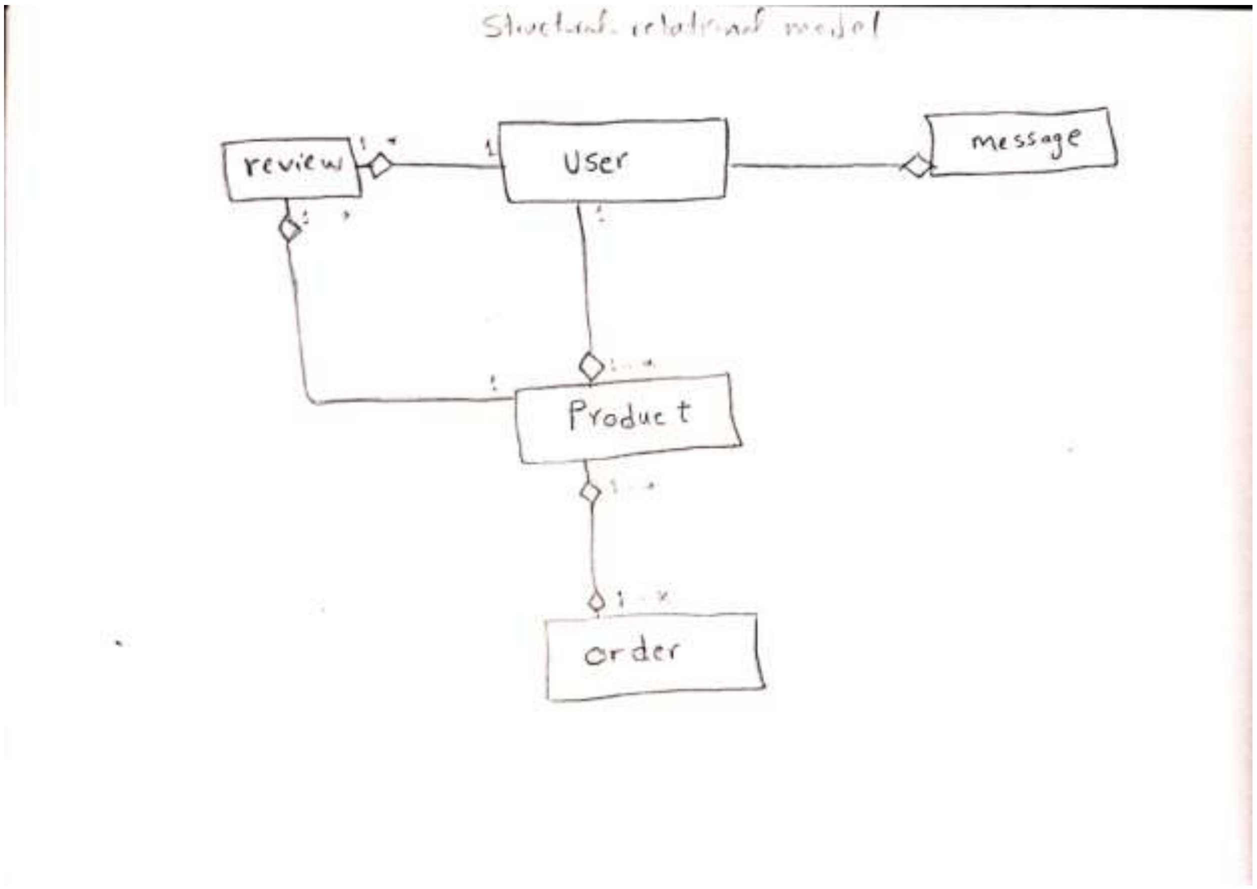
DFD model:



The structural model



Structural diagram that shows the relationship between the classes:



The main and subsystems:

The main systems in the project as presented in the architectural diagram:
1- The supply system
2- The customer system

The subsystems:

The supply system contains:
 1- server side
2- database
3- viewing data

The customer system contains
1- GUI, namely, internet browser to access data by requests
2- interfaces to use different features

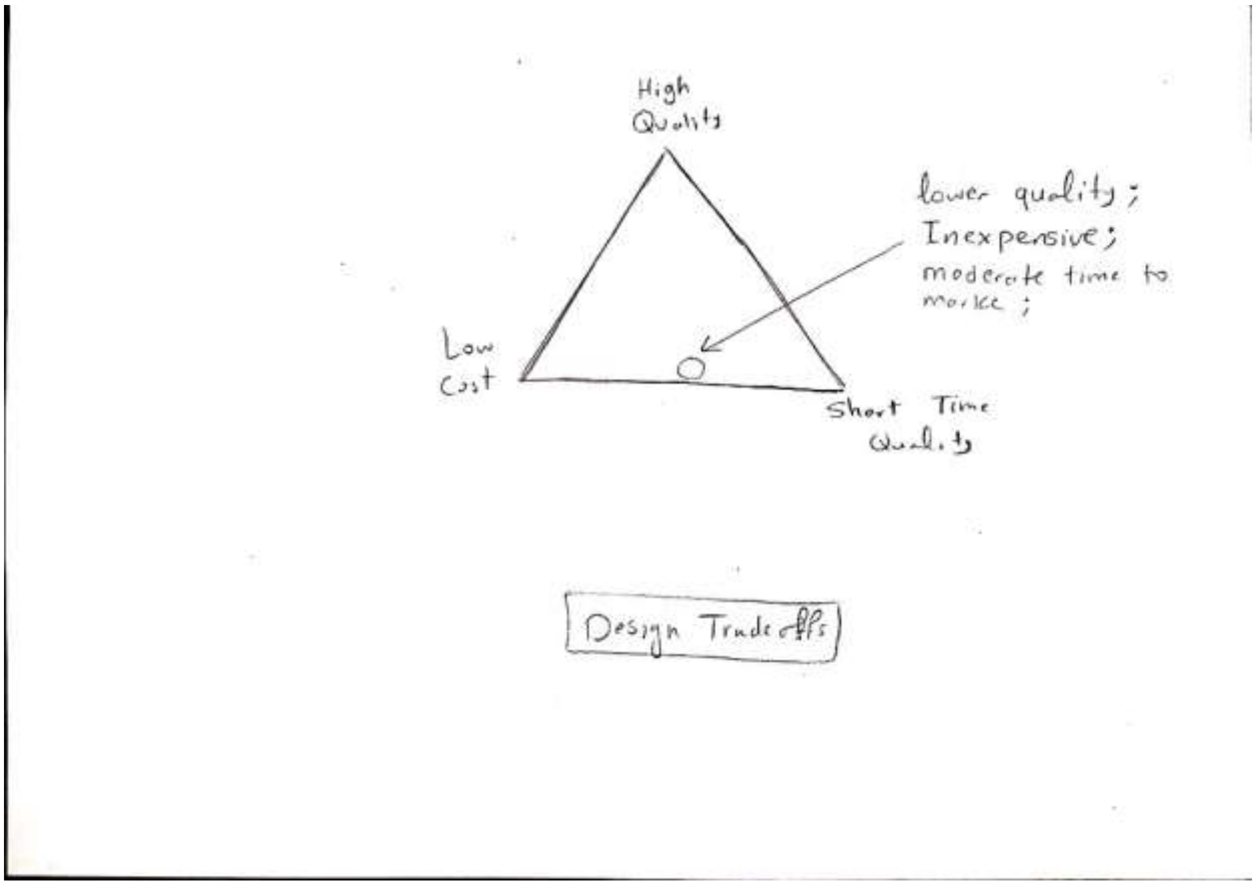
3.3 Design Rationale

This architecture design was chosen as our system follows a client-server model. This diagram is the easiest way to show the flow of the data from the server side to the client Side which are the main systems of the project. Those are some architectural design that were taken into consideration. We have thought of enterprise resource planning and use another generic application that supports the same idea, however, we did not find any open-source version that fits our base requirements till now. This system can use a layered architecture pattern as the one developed by John Donovan which is three layered architecture that is composed of base entities, functions, and interfaces. This part will be elaborated on later in the component design section. The strategy that will be used to control the operation of the model is the Model-View Controller schema that eases the communication between components of web-based application as this. There was a decision at the beginning to use a single-page application (SPA) framework like angular.js, but we did not choose this one as the application does not contain huge-size files that a lot of data; thus, we do not need this kind of fast performance while loading the pages to the users, also, these frameworks has a lot of files that will complicate the concurrency process of the team members working without really getting great difference in the performance. That is why we did not choose this one.

Design Tradeoffs:

We choose to create the application with lower quality, with low cost and moderate time to market.

The cost of the project was the hosting service, the development process cost. The time taken for the application to be launched to the market is 3 months. The low quality is due to low space for user and product items on the current sqlite database as it is a lightweight database that fits more for this kind of light application and fast during the development stage working on localhost.



4. Data Design

4.1 Data Description

For persistent data management, a relational database created by MySQL will be used as the database management system (DBMS) of the Switching Items Management System (Switchify).

This database is used to store all data related to Switchify, including user information, posted items information, chat text data, and Rating data.

Switchify shall implement some authentication mechanism for access to the database. APIs will be written in PHP to return data in JSON format. The web site will use the APIs to access the database.

The Database is organized by storing Data in 7 tables containing data about: Users, products(items), orders, order items, chat sessions, messages and (comments & ratings).

The figure in section 4.1.1 shows High level view of the database of Switchify.

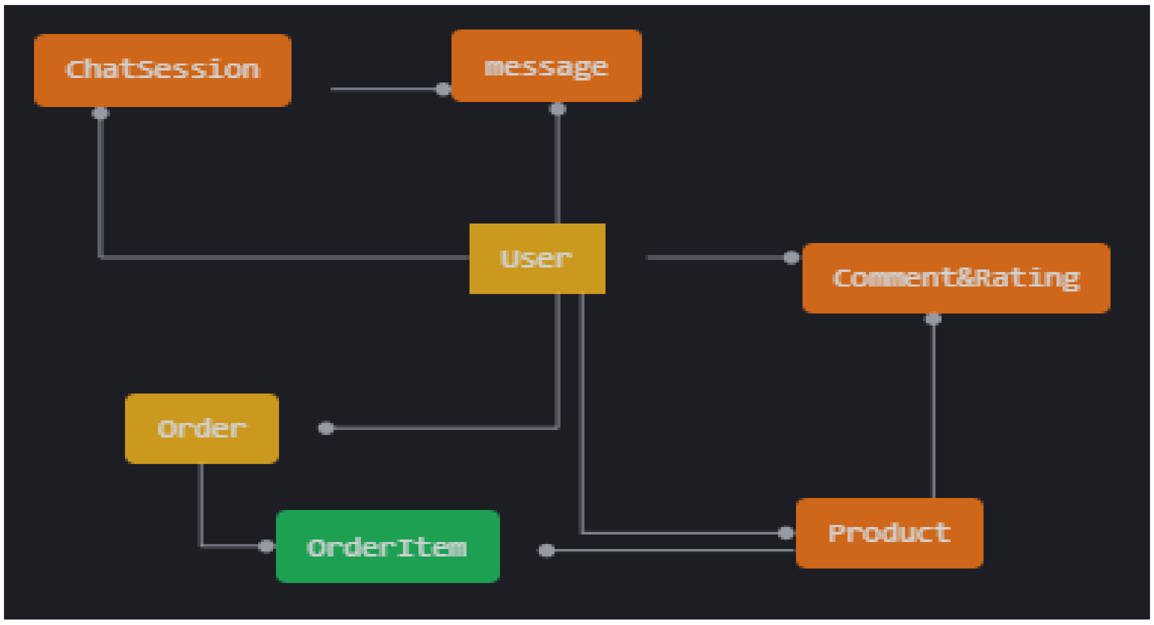


Figure (4.1.1) High level view of database structure organization

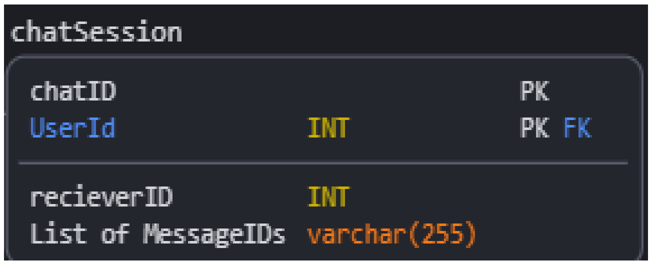
4.2 Data Dictionary

Data is Stored in 7 tables listed alphabetically as follows:

1- Chat Sessions table:

it contains a list of all chat sessions with columns :

- **Chat IDs**
- **User ID** (user who started the chat)
- **Receiver ID** (user receiving first message)
- **List of messages IDs** listed as comma separated string



chatSession

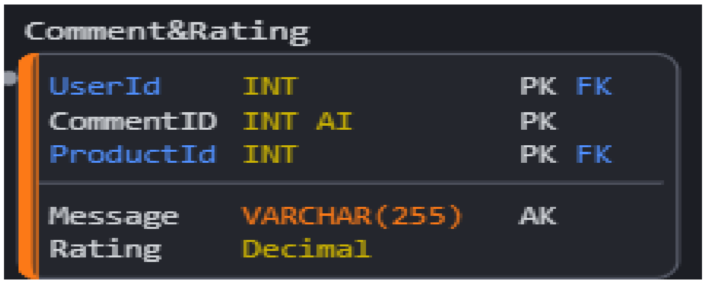
chatID		PK
UserId	INT	PK FK
recieverID	INT	
List of MessageIDs	varchar(255)	

Figure (4.2.1) Chat Sessions table

2- Comments & Ratings table:

it contains a list of all Comments & Ratings for users or items with columns:

- **User IDs** (user who wrote the comment)
- **Comment ID**
- **ProductID** (ID of product being rated)
- **Message** (content of the comment)
- **Rating** (Value of the rating in decimal)



Comment&Rating

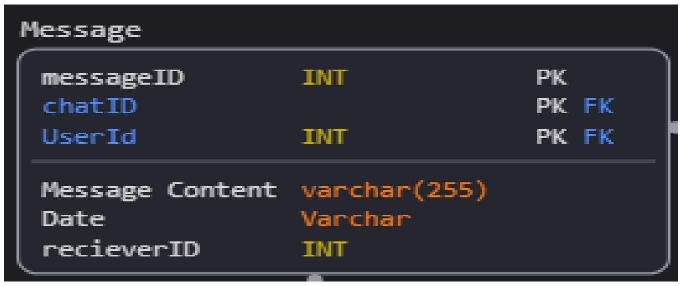
UserId	INT	PK FK
CommentID	INT AI	PK
ProductId	INT	PK FK
Message	VARCHAR(255)	AK
Rating	Decimal	

Figure (4.2.2) Comments & Ratings table

3-Messages table:

it contains a list of all Messages between users with columns:

- **Message ID**
- **Chat ID** (ID of the chat where the message belong)
- **User ID** (user who wrote the message)
- **Message content** (text of the message)
- **Date and time** (of writing the message)
- **Receiver ID** (ID of user who received the message)



Message

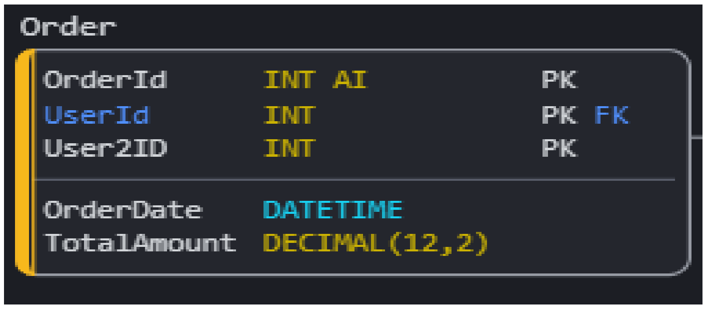
messageID	INT	PK
chatID		PK FK
UserId	INT	PK FK
Message Content	varchar(255)	
Date	Varchar	
recieverID	INT	

Figure (4.2.3) Messages table

4- Orders Table:

it contains a list of Orders with columns:

- **Order Id**
- **User ID** (ID of first user switching)
- **User2 ID** (ID of second user switching)
- **Order Date**
- **Total Amount** (Total amounts of items switched)



Order

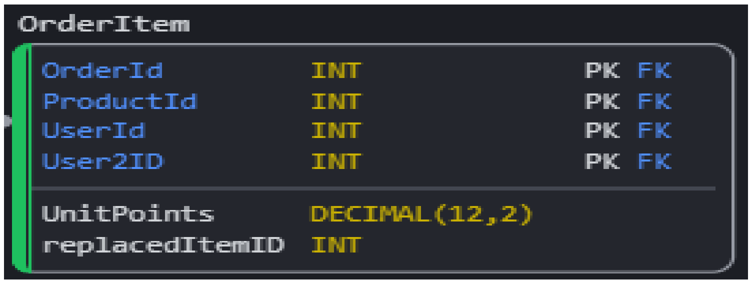
OrderId	INT AI	PK
UserId	INT	PK FK
User2ID	INT	PK
OrderDate	DATETIME	
TotalAmount	DECIMAL(12,2)	

Figure (4.2.4) Orders Table

5- Order Items Table:

it contains a list of all Order items with columns:

- **Order Id**
- **Product ID** (ID of first product being switched)
- **User ID** (ID of first user switching)
- **User2 ID** (ID of second user switching)
- **Unit points** (value of points of products switched)
- **replacedItemID** (ID of Second product being switched)



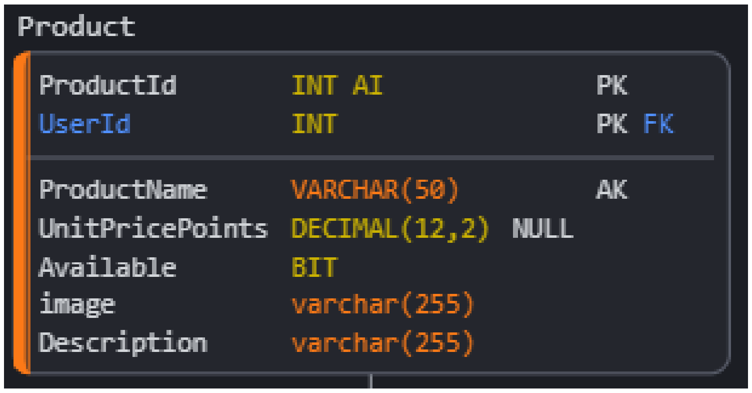
OrderItem		
OrderId	INT	PK FK
ProductId	INT	PK FK
UserId	INT	PK FK
User2ID	INT	PK FK
UnitPoints	DECIMAL(12,2)	
replacedItemID	INT	

Figure (4.2.5) Order Items Table

6- Products Table:

it contains a list of all items with columns:

- **Product ID** (ID of first product being switched)
- **User ID** (ID of item current owner)
- **Product name**
- **UnitPricepoints** (estimated value of the product in points)
- **Available** (binary flag)
- **Image** (link to source image)
- **Description** (text description of the product)



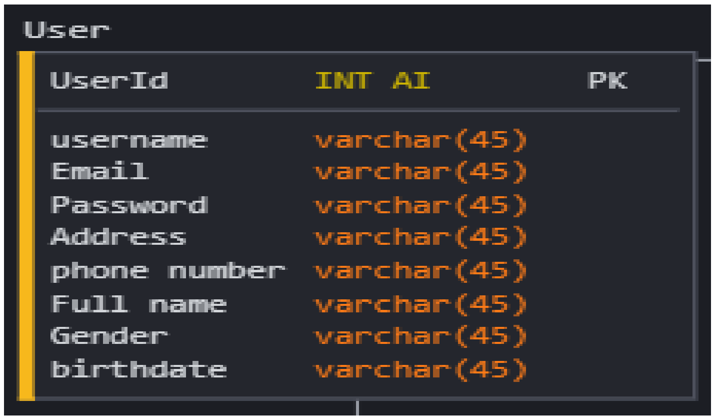
Product			
ProductId	INT AI		PK
UserId	INT		PK FK
ProductName	VARCHAR(50)		AK
UnitPricePoints	DECIMAL(12,2)	NULL	
Available	BIT		
image	varchar(255)		
Description	varchar(255)		

Figure (4.2.6) Products Table

7- Users Table:

it contains a list of all Users with columns:

- **User ID**
- **Username**
- **Email**
- **Password**
- **Address**
- **Phone number**
- **Full name**
- **Gender**
- **birthdate**

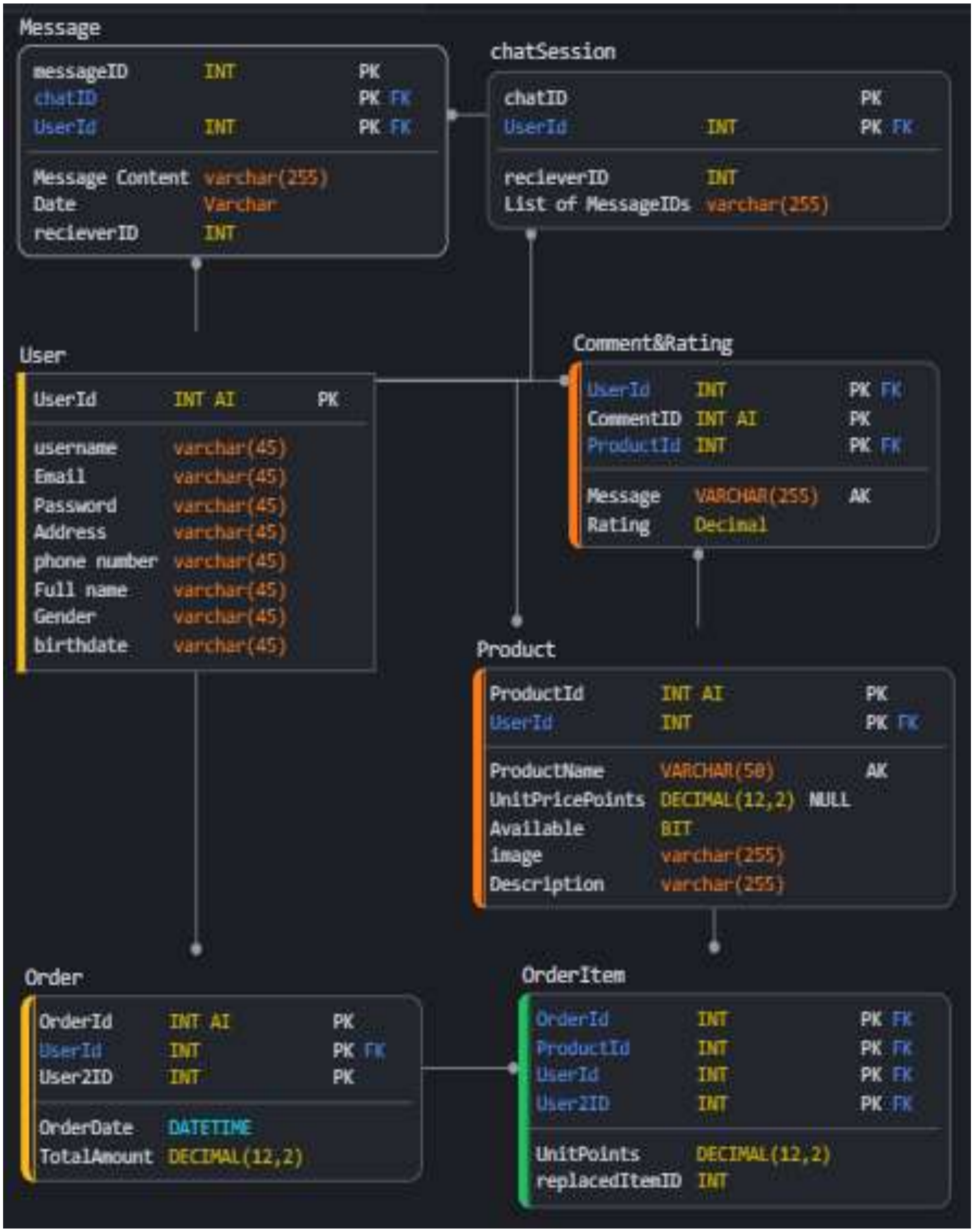


User

UserId	INT AI	PK
username	varchar(45)	
Email	varchar(45)	
Password	varchar(45)	
Address	varchar(45)	
phone number	varchar(45)	
Full name	varchar(45)	
Gender	varchar(45)	
birthdate	varchar(45)	

Figure (4.2.7) Users Table

Finally, Figure (4.2.8) shows the database sample diagram



5. Component Design

In this section, the component-level design will define the data structures, algorithms, base entities, functions, interface characteristics, and communication mechanisms allocated to each component for the system development. The components in this system will be represented as a bunch of classes using a layered architecture pattern especially following the J.D pattern. This pattern divides the classes into three layers, namely, base entities, functions, and interfaces.

The first component:

The base entities:

1- User

2- Product

3- Profile

4- Chat

The second component:

The behavior:

1- login

2- signup

3- password change

3- chatting

4- reviewing products

5- reviewing other customers

6- add product

7- edit product

8- search for a product

<u>The third component:</u>

<u>The interface:</u>

Note that every behavior has an interface for the user to enable him from using those features.

1- login form

2- signup form

3- password change form

4- chatting interface

5- form to review other customers and their products

6- add product form

7- edit product form

8- search form to search for a product

Those are the attributes associated for each entity.

<u>The user class:</u>

The user class contains those attributes.

1- username

2- email

3- password

4- address

5- phone number

6- full name

7- customer name

8- gender

9- full date

<u>The product class:</u>

The user class contains those attributes.

1- product name

2- unit price points

3- available

4- image

5- description

<u>The chat class:</u>

The chat class contains those attributes.

1- message Id

2- sender Id

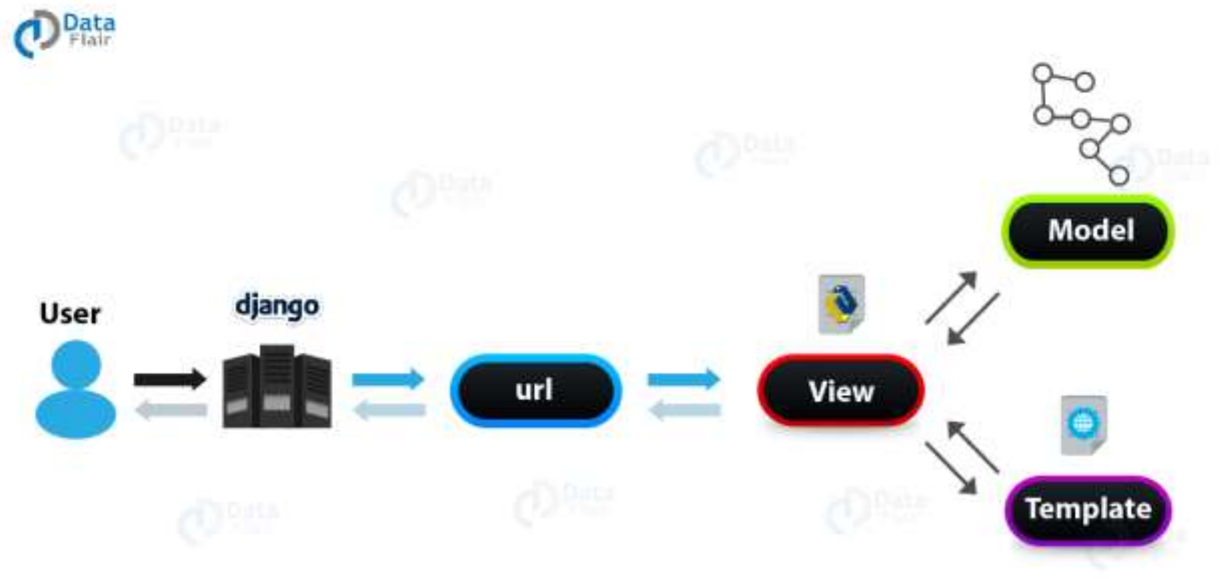
3- receiver Id

4- Message content

5- date

Communication mechanism:

Design pattern: The system will follow a model-view-controller framework. The models are the database entity classes aforementioned. The controller will do the logical functionalities to extract the needed data from the database. The view will be responsible for displaying those data to the user. The communication between those components can be visualized as follows:



This diagram is adapted from data flair training.

As shown in the figure above, the view - in this framework - will take the responsibility of the controller, the view communicates with the model from one side and from the template from another side. It communicates with the model to get the data from the database, then processes and filters as needed then sends the last version of the data to the template to display it.

Relation between models:

The classes that have one to many relationships:

1- The user and the products.

2- The product and the image.

3- The user and the chat

The classes that have one to one relationships:

1- The user and the profile

2- The profile and the Image

This is a quick discussion on the attributes and methods of each class.

<u>The user attributes:</u>

1- id (auto-generated primary key)

2- name (text field for the user’s name)

3- password (password encrypted field for the user’s password)

<u>The product attributes:</u>

1- id (This is auto-generated column will serve as primary key)

2- name (This is text field for the name of the product, this can take null value)

3- description (This is detailed description about the product)

4- points (This is text field that contains the points (value) that this product worth)

5- color (text field contains the color of the product)

6- image (image field for the product)

<u>The profile attributes:</u>

1- user (This is foreign key for the user)

2- phone number (text field optional to hold phone number)

3- image (image field for the profile picture for the user)

<u>The chat attributes:</u>

1- sender (text field to hold the sender’s name)

2- receiver (text field for the receiver user)

3- message (the body of the message)

This was an Object Oriented view for the set of collaborating classes that contribute to the system.

6. Human Interface Design

6.1 Overview of User Interface

The user interface has many activities such as Main Activity, Home Activity, Chat Activity, and Edit Profile Activity.

Main Activity:
The main user interface starts with a logo image, three edit text boxes for the credentials of the user, login button, signup button, and two clickable text views: one for resetting the password and another for creating a new account. It also has an image button to add a user profile image.

Home Activity:
The home activity has a bottom navigation viewer that navigates through four different fragments to be displayed on the home page. The default button is the home button which displays the home fragment that contains all the user posts. The user gets to see a chat button under each post in order to be able to communicate with other users. Each post has a username, a user profile image, a product image, and a product description. The second navigation button is the add post button which displays the add post fragment in the home page. The post fragment contains an image button to add an image for the product, three edit text boxes to add the product name, product description, and product type, and a post button to add the post to the home fragment. The third navigation button is the profile button which displays the profile fragment on the home page. The profile image has a section for

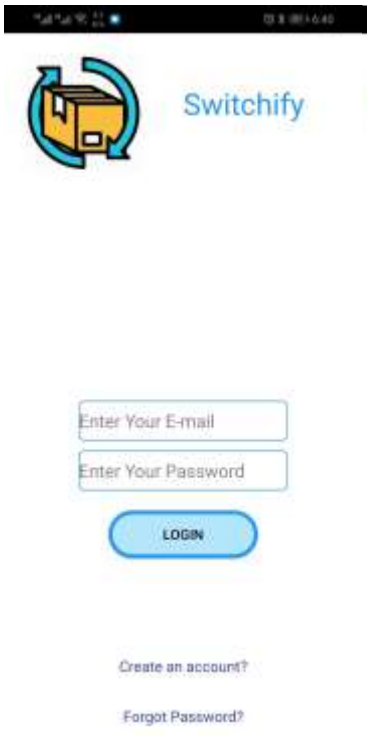
displaying the user products that he previously posted on the app. Also it has a button for editing the profile, and a the user profile images. Moreover, it counts the user posts in a text view box. The fourth navigation button is the chat button which displays the chat fragment on the home page. The chat fragment displays a clickable list of the chats that the user had with other users about their products.

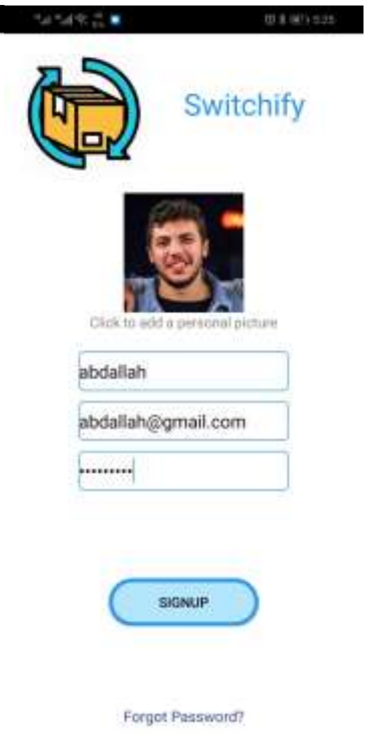
Chat Activity:
The chat activity is the activity where the user can discuss the details of a product with another user. It consists of a toolbar that contains the image and the username of the person whom the user chats with. It also has a list of text boxes that contains the messages of the users. Also, it has an edit text box and a send button at the bottom of the page to type the messages and send them.

Edit Profile Activity:
The Edit profile activity is the activity where the user can edit his data such as password, profile image, and username. It has three edit text boxes for the username, the current password, and the new password. It also has an image button to add a new image and a save button to save the new data.

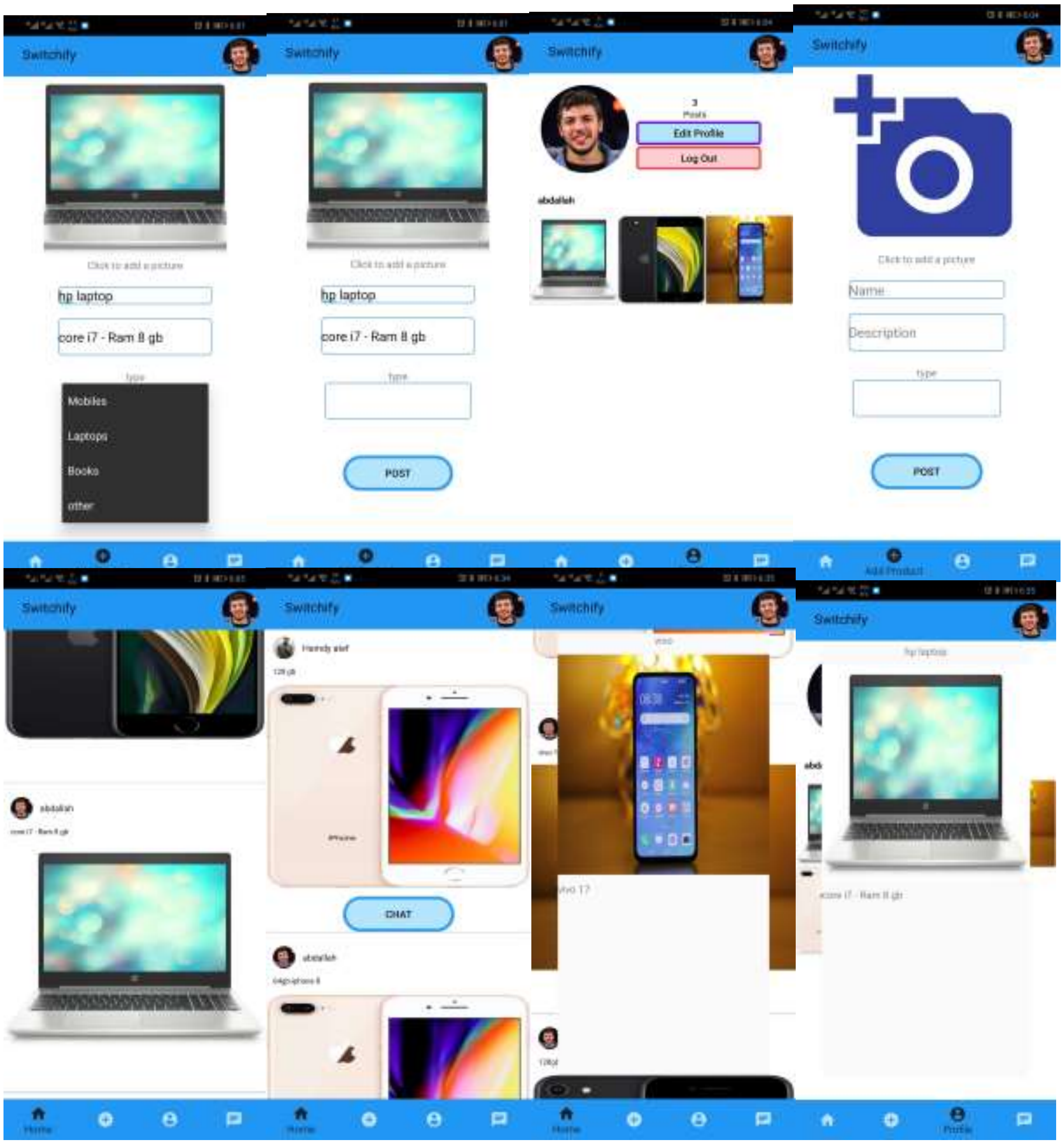
6.2 Screen Images

Main Activity:





Home Activity:



hp laptop

core i7 - Ram 8 gb

abdallah

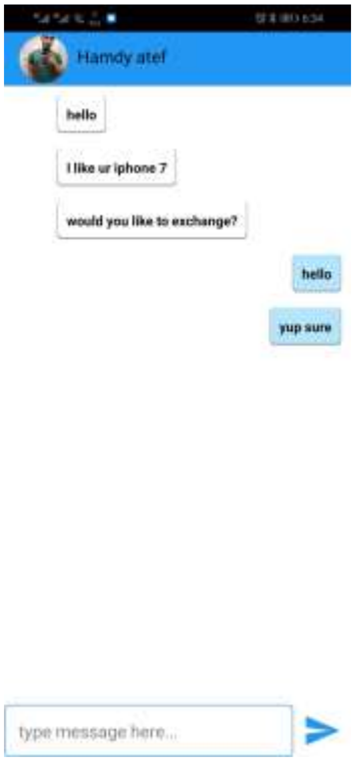
Hamdy atef

128 gb

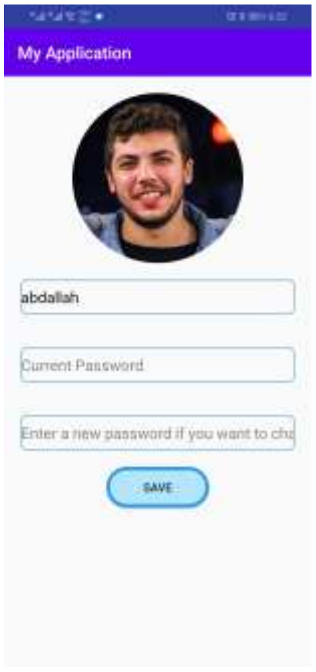
64gb iphone 8

vivo 17

Chat Activity:



Edit Profile Activity:



6.3 Screen Objects and Actions

Main Activity:
Main activity starts by disappearing the signup button and the username edit text box. Once the user clicks on creating a new account, the signup button appears and another edit text box appears to collect the user name. Moreover, the login button turns invisible. By pressing on the backward button, the program terminates if the user is in the login phase, or makes the user name edit text box and the signup button invisible if the user is in the signup phase. By pressing on the login button, the system checks the database to confirm the user credentials typed in the edit text boxes: email and password then starts the Home Activity. Similarly with the signup button, once the user enters his details in the edit text boxes, the system saves the user data in the database and authenticates the user, then starts the home activity.

Home Activity:
Once the user clicks on the navigation home button, the system terminates the displayed fragment and starts the home fragment. The home fragment contains posts that have a chat button under each of them. Once the user clicks on the chat button, the system starts the chat activity between the two users.

Once the user clicks on the navigation add post button, the system terminates the displayed fragment and starts the add post fragment. The add post fragment contains a post button and an image button. Once the image button is pressed, the system opens the phone gallery for the user to choose an image of the product. Once the image is chosen the system uploads it to the database and saves a string URL for it. Once this post button is pressed, the system collects the URL, name, description and type of the product and posts it to the home page.

Once the user clicks on the navigation profile button, the system terminates the displayed fragment and starts the profile fragment. The profile has a clickable list of images which the user can click on to display the product details on a separate activity. It also has two buttons: one to initiate the edit profile activity and the other to logout and end the home activity displaying only the main activity.

Once the user clicks on the navigation chat button, the system terminates the displayed fragment and starts the chat fragment. The chat fragment has a clickable list of users that when clicked it sends the user to the chat activity.

Chat Activity:

It has a send button that when clicked, it collects the message typed in the message edit text box and sends it to the database. Then, all the messages in the database are displayed on a text view list.

Edit Profile Activity:
Once the image button is clicked, the system opens the phone gallery to choose the new image. Then the system uploads the new image to the database and saves a string URL of it. Once the edit button is clicked the system collects the data in the text boxes and checks the validation of the user by his old password, then save the new data in the database.

7. Requirements Matrix

Req ID	Requirement description	TestCase ID	TC description	Test designer	Test results	Req coverage status
REQ01	Login to the application	TC01	Login with invalid username or invalid password	Abdallah	Passed	Completely covered
		TC02	Login with valid username and valid password	Abdallah	Passed	Completely covered
REQ02	Signup	TC03	Use correct information but invalid email	Seif	Passed	Completely covered

		TC04	Use valid email and correct information	Seif	Passed	Completely covered	
REQ03	Browse feed	TC05	Check if the feed should include all the posted items in chronological order	Basuony	Passed	Partially covered: All the items on the feed are posted by us and are browsed offline. Still needs testing online.	
REQ04	View item	TC06	View item’s full description by clicking on it.	Omar	Passed	Completely covered	
REQ06	Post item	TC07	Post an item with its pictures and description.	Omar	Passed	Completely covered	
		TC08	Get help to determine what your item is worth	Omar	Needs testing	Not covered yet	
REQ07	Edit item	TC09	Edit pictures or description of an item	Abdallah	Passed	Completely covered	

REQ08	Delete item	TC10	Delete your item with its pictures and description	Abdallah	Passed	Completely covered
REQ09	Search about items	TC11	Type the name of the item you want to search for	Basuony	Needs testing	Partially covered: The app doesn't have enough items to be tested.
REQ10	Rate other users	TC12	When a trade is done, rate the other user.	Basuony	Needs testing	Partially covered: The app needs to be on the production server to test this feature.
REQ11	Filter posted items	TC13	When viewing items on the feed, filter items by value.	Seif	Needs testing	Partially covered: The app doesn't have enough items to be tested.

REQ12	Verify that a trade is done successfully	TC15	When meeting other users to complete the trade, scan that user’s QR code	Omar	Needs testing	Partially covered: The app needs to be live in order to test this feature.
		TC16	After chatting with other user about a certain item, prompt a questionnaire when the use opens the app again.	Abdallah	Needs testing	Partially covered: The app needs to be live in order to test this feature.