BOOKHUB APP

A PROJECT REPORT

Submitted By

MADHUR MAHESHWARI (2000290140063)
NISTHA GOYAL (2000290140077)
SAKSHI PANWAR (2000290140109)

Session:2021-2022 (4th Semester)

Submitted in partial fulfillment of the Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the supervision of
Dr. Vipin Kumar
Associate Professor

KIET Group of Institutions, Delhi-NCR, Ghaziabad

DEPARTMENT OF COMPUTER APPLICATIONS
KIET GROUP OF INSTITUTIONS, DELHI-NCR, GHAZIABAD-201206
(JUNE- 2022)

Declaration

We hereby declare that the work presented in this report entitled “BookHub", was carried out by us. We have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.
We have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not our original contribution. We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.
We affirm that no portion of our work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, we shall be fully responsible and answerable.

Name: - Madhur Maheshwari
Roll no: - 2000290140063

Name: - Nistha Goyal
Roll no: - 2000290140077

Name: - Sakshi Panwar
Roll no: - 2000290140109

Signature of Candidate

CERTIFICATE

Certified that **Madhur Maheshwari (2000290140063), Nistha Goyal (2000290140077), Sakshi Panwar (2000290140109)** have carried out the project work having "**BookHub**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Madhur Maheshwari (2000290140063)
Nistha Goyal (2000290140077)
Sakshi Panwar (2000290140109)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Vipin Kumar
Associate Professor
Department of Computer Application
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

Dr. Ajay Kumar Shrivastava

Head, Department of computer application

KIET group of institutions, Ghaziabad

ABSTRACT

The purpose of BookHub project is to automate the existing system using computerized equipment's and full-fledged software, for fulfil the requirements, so that valuable information can be kept for a period of time with easy accessing.
The required hardware and software are easily available and it's also to work with.

BookHub as mentioned above, can lead to secure, bug free, reliable, and fast system. It can assist the concentrate on their different tasks instead to concentrate on the keeping of the record. Thus, it will help organization in better utilization of resources. The system can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

The aim of the project "BookHub" is to manage the details of book, stock, customer, order, payment, donation of books. It manages all the information about Books, Bill, Payment.
The project is totally built at administrative end thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work of managing the Books, Stock, Bill, Customer.
It tracks all the details about the customer, Order, Payment.

Acknowledgement

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Dr. Vipin Kumar** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Ajay Kumar Shrivastava**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Madhur Maheshwari

Nistha Goyal

Sakshi Panwar

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Introduction of the project

The Project “BookHub” has been developed to override the problems which was created by the practicing manual system. It supports to eliminate as well as reduce the hardships faced entering the by company to carry out operations in a smooth and effective manner.

The application avoid errors and reduced as much as possible to while entering the data. It provides error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus, by this all it proves it is user- friendly.

“BookHub” as described above can lead to secure, error free, fast management system and reliable. It can assist to concentrate on other activities rather to concentrate on the record keeping. Thus, it will help organization in better utilization of resources.

Every organization, whether small or big has challenges to overcome and managing the information of stock, Bill, Books, Order, payment. Every online bookstore has different book’s needs; therefore, we design management systems that are adapted to your managerial requirements. This is designed to assist in strategic planning and will help you ensure that your organization is equipped with the right level of information and details for your future goals.

Also for those busy executive who are always on the go, our systems come with remote access features, which will allow you to manage your workforce anytime , at all times.

These systems will ultimately allow you to better manage resources.

1.2 Problem Statement

- Online Book Store is a specific requirement of the client that integrates the buying and selling services specifically to their customers.
- Reports can be generated at any time within few seconds, so that manual labor is not required, and also analysis can be performed much more frequently which helps in taking decision.
- The details regarding all users, books can also be maintained as their information is very helpful and sometimes becomes a critical requirement.
- Allows user to get registered from their places and transact for the required product.

1.3 Objective of the project

The objective of the project “BookHub” is to manage the details of book, stock, customer, order, payment, donation of books. It manages all the information about Books, Bill, Payment .
The project is totally built at administrative end thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work of managing the Books, Stock, Bill, Customer.
It tracks all the details about the customer, Order, Payment.

Functionalities provided by BookHub are as follows:

- Provides the searching facilities based on various factors. Such as Books, Customer, Order, Payment
- It tracks all the information of Stock, Bill, Order etc.
- Shows the information and description of the books, customer
- To increase efficiency of managing the Books, stock
- It deals with monitoring the information and transactions of order.
- Store records of old books.

1.4 Project Scope

The project successfully delivered on all requirement specification specified by the user. Care was ensured during the design to make sure data integrity is maintained and to avoid all forms of redundancies associated with data. The user is assured a very friendly interface, behind which there are wide ranging technical details that went in. The user guide is a mere formality because, the project was specially created bearing in mind interaction and designs that would make users feel as though they have used a system such as this.

This project has also been built in such a manner that future changes or modifications that are required can easily be implemented without affecting the functionality of the system. This project is used on android environment and can be used on any version so it can be used by individuals with different levels of android devices.

The technical document that is provided in the synopsis of this project will help developers understand the internal workings of the system.

1.5 Hardware / Software requirements

- Hardware –
  - On Smart Phone

    1) Processor should be dual core.
    2) Processor speed be 1.3 GHz.
    3) ROM 1 GB.
    4) RAM 512 MB.
    5) Android Version 5.0 Lollipop and above.

  - On Personal Computer (PC)

    1) Processor should be dual core.
    2) Processor speed be 1.5 GHz.
    3) RAM more than 4 GB.
    4) ROM more than 30 GB.

- Software –

    1) Sublime text.
    2) React native.
    3) HTML.
    4) MongoDB
    5) Firebase

1.7 Feasibility Study

Feasibility study is used to check the visibility of the project under the consideration of the theoretical project and its various types of feasibility are conducted below are important which are represented are as follows:-

1.6.1 Technical Feasibility

Technical Feasibility determines whether the work for the project can be done with the existing equipment's software technology and available personals technical feasibility is concerned with the specified equipment and software that will satisfy the user requirement this project is feasible on technical remark also as the proposed system is beneficiary in term of having a soundproof system with new technical equipment's install on the system that proposed system can run on any machine supporting window and work on the best software and hardware that had been used while designing the system so it would be visible in all technical term of feasibility3.

1.6.1 Economic Feasibility

With manual system the operation cost of this system is about 60 lacs for annual this cost comprises salary of Swiss 25 people's stationary, building rent, electricity water telephone etc. but with a new system this offering cost comes out to be about 20 lacs for renewal and the new system is economically feasible.

1.6.2 Operational Feasibility

The new solution is possible in all sense but operationally is not the new system demands the explosion of at least 15 people from the company it creates an environment of joblessness and fear among the employees it can lead to indefinite Strike in the company also, so the management must take corrective action prior in advance to start further proceeding.

1.6.3 Behavioral Feasibility

Behavioral Feasibility is the measure of how the society is looking towards our project, what is the reaction of people who are going to use this in upcoming future.

1. It includes how strong the reaction of user will be towards the development of new system that involves device use in their daily life for connecting with faculties.

2.LITERATURE REVIEW

2.1 Literature Review

A lot of websites and applications can be found when we search the google which are developed for learning purposes. But there is ambiguity in choosing the appropriate content in appropriate time. Some websites have been developed which consists of stories, novels, essays etc. Similarly, some personal blogs and websites are developed for studying purpose. Electronic Commerce (e-commerce) applications support the interaction between different parties participating in a commerce transaction via the 24 Sep 2017 Shop new, used, rare, and out-of-print books. literature review for online bookstore. prototype provides a guideline in developing a real system of Online Ordering Book based on user's perspective, particularly in the perspective of academic will be handled by the university's bookstore.

However, the ordering is still done manually. Based on literature reviews and website reviews of other universities. This research was undertaken to provide background for the SCONUL and Jisc eBook Co-Design Project. Since the 1980's, when e-books first the idea of this review paper, begins from writer's disappointment about online bookstore service quality in. Indonesia. Which is, as a consumer in several transactions with the online bookstore.

In overall, the writers did not satisfy towards their services. Lions Villa Probation Hostel Gate Pass Management System at Coimbatore Campus Amrita the Computerized Gate Pass Management System 27 Sep 2017. Also includes introduction to meta-analysis. Relevant to students from any discipline. Includes contributions from both a professor and a librarian.

2.2 Existing System

In the present scenario, people have to physically visit the bookshops or vendors for purchasing books of their need and have to make payment through cash mode most of the times due to unawareness of advanced technologies at certain places. In this method time as well as physical work is required, among which time is something that no one has an ample amount. The traditional book purchasing procedure is not efficient enough for shopkeepers as well as customers, as they have to deal with the crowd, in their shops. The old methods are classified into two ways where the most popular one was you need to go to the shop and ask the shopkeeper for the book.

If he has that book then he will give you and demand money, which could be more than you thought for. The other one is if you know any retailer or shopkeepers you can directly contact him on phone and ask him to give you that books at your place and take the money and extra convenience charges of transportation.

These both the methods are slow and cost you more sometimes if you are a novice in marketing or purchasing any new thing.

3.SYSTEM DESIGN

3.1 Introduction

System design is the solution of a “how to approach to the creation of the new system. It is composed of several steps. It facilitates the understanding and provides the procedural details necessary for implementation of the system recommended in the feasibility study. Emphasis is given on translating the performance requirements into design specification. Design goes through logical and physical stages of development.

Logical design reviews the present physical system; prepares input and output specification; make editing; security and control specification; details the implementation plan, and prepare logical design walk through. The physical design maps out the details of the physical system; plans the system implementation plan and specifies hardware and software. System design translates the system requirement into the ways of the system as recommended in the feasibility study. Thus, the system design is the translation from user-oriented document to a programmer or a database personal oriented document. System design is a highly creative process that can be greatly facilitated by the following: -

- Strong Problem Definition
- Pictorial description of the Existing System
- Set of Requirements of the new system

3.2 Module Description

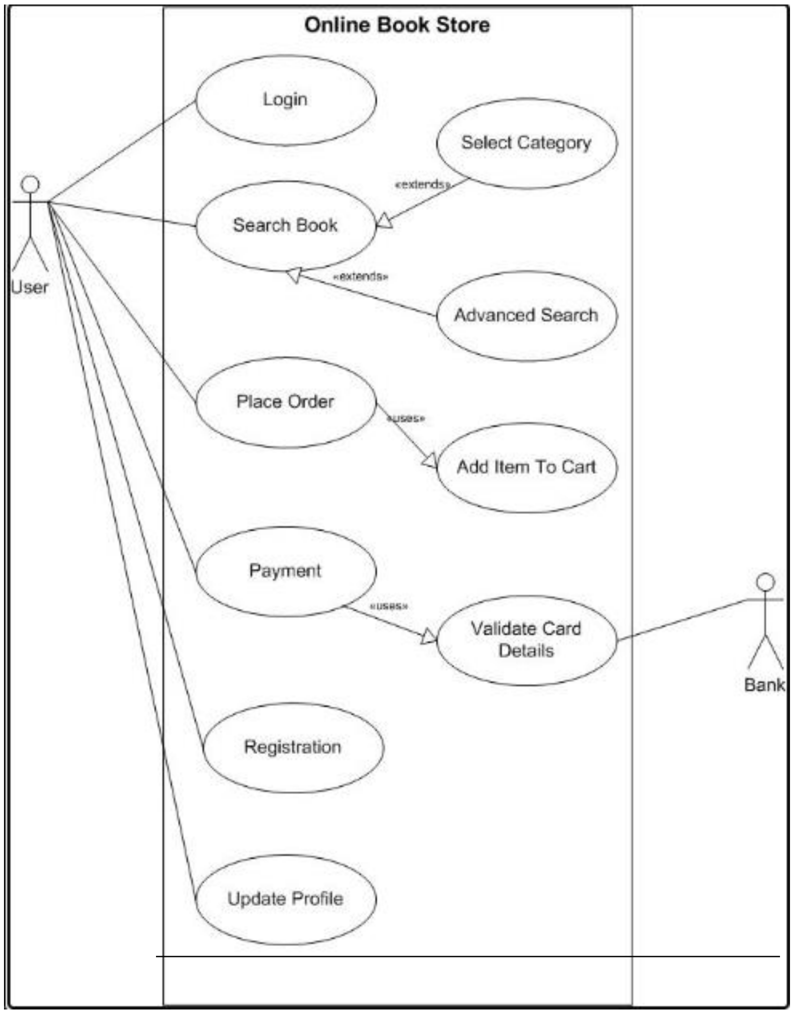
1. Registration: Customer can register their account here to continue shopping.
2. Admin: Admin can add books, check orders and make sure the orders are delivered on time and can confirm payments by the customers.
3. Shopping Cart: Customers after login can browse through the different books and choose one or more products and can add them to cart.
4. Payment: Cash on Delivery facility is available.

3.3 CONCEPTUAL MODELS

3.3.1USE CASE DIAGRAM

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

While a use case itself might drill into a lot of detail about every possibility, a use-case diagram can help provide a higher-level view of the system. It has been said before that "Use case diagrams are the blueprints for your system”. They provide the simplified and graphical representation of what the system must do.



3.3.2 DATA FLOW DIAGRAM

What it is?

1.7 The Data Flow Diagram shows the flow of data or information. It can be partitioned into single processes or functions. Data Flow Diagrams can be grouped together or decomposed into multiple processes. There can be physical DFD's that represent the physical files and transactions, or they can be business DFD's (logical, or conceptual).

When it's used?

The DFD is an excellent communication tool for analysts to model processes and functional requirements. One of the primary tools of the structured analysis efforts of the 1970's it was developed and enhanced by the likes of Yourdon, McMenamin, Palmer, Gane and Sarson. It is still considered one of the best modelling techniques for eliciting and representing the processing requirements of a system.

Used effectively, it is a useful and easy to understand modeling tool. It has broad application and usability across most software development projects. It is easily integrated with data modeling, workflow modeling tools, and textual specs. Together with these, it provides analysts and developers with solid models and specs. Alone, however, it has limited usability. It is simple and easy to understand by users and can be easily extended and refined with further specification into a physical version for the design and development teams.

The different versions are Context Diagrams (Level 0), Partitioned Diagrams (single process only -- one level), functionally decomposed, leveled sets of Data Flow Diagrams.

Data Store

It is a repository of information. In the physical model, this represents a file, table, etc. In the logical model, a data store is an object or entity.

DataFlows

DFDs show the flow of data from external entities into the system, showed how the data moved from one process to another, as well as its logical storage. There are only four symbols:

- Squares representing **external entities**, which are sources or destinations of data.
- Rounded rectangles representing **processes**, which take data as input, do something to it, and output it.
- Arrows representing the **data flows**, which can either, be electronic data or physical items.
- Open-ended rectangles representing **data stores**, including electronic stores such as databases or XML files and physical stores such as or filing cabinets or stacks of paper.

There are several common modelling rules for creating DFDs:

- All processes must have at least one data flow in, and one data flow out.
- All processes should modify the incoming data, producing new forms of outgoing data.
- Each data store must be involved with at least one data flow.
- Each external entity must be involved with at least one data flow.
- A data flow must be attached to at least one process.

DFDs are nothing more than a network of related system functions and indicate from where information is received and to where it is sent. It is the starting point in the system that decomposes the requirement specifications down to the lowest level detail.

The four symbols in DFD, each of which has its meaning. They are given below:

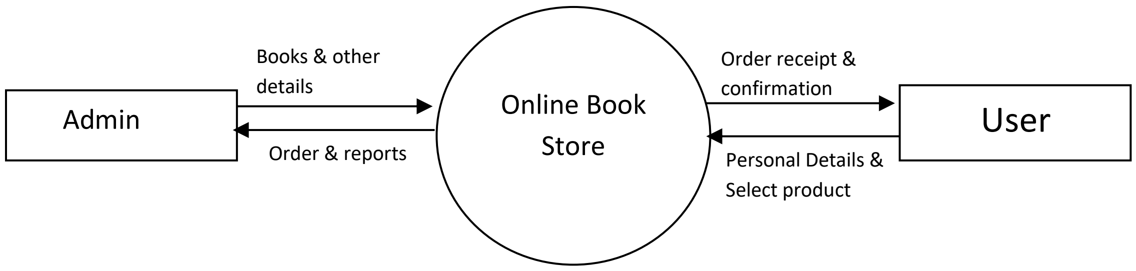
- External entities are outside to system but they either supply input data in the system or use the system output. These are represented by square of rectangle. External entities that supply data into a system are sometimes called Sources. External entities that use system data are sometimes called sinks.
- Dataflow models that passage of data in the system and are represented by line by joining system components. An arrow indicates the direction of the flow, and the line is labeled by the name of the dataflow.
- Process show that the systems do. Each process has one or more data inputs and one or data outputs. Circles in DFD represent them. Each high-level process may be consisting of more than one lower-level processes. Process will be expanded in sequent level DFD. A circle or a bubble represents a process that transforms incoming data flow into outgoing dataflow.

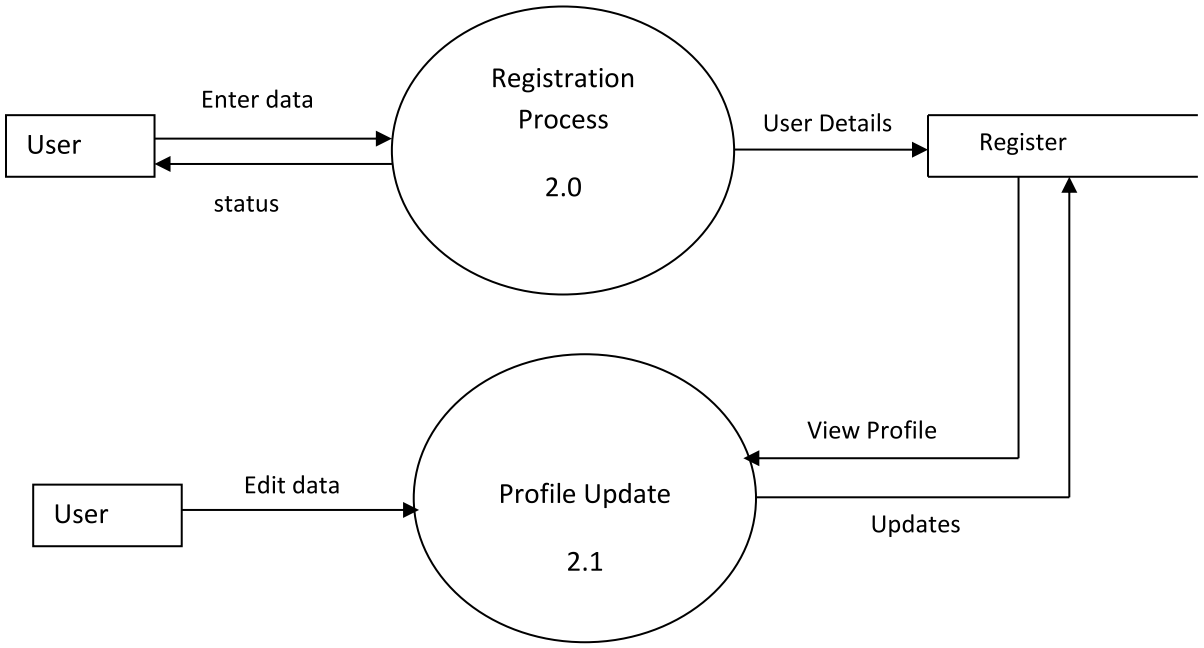
The high-level processes in a system are:

- Receivable process.
- Verifiable process.
- Disposal process.

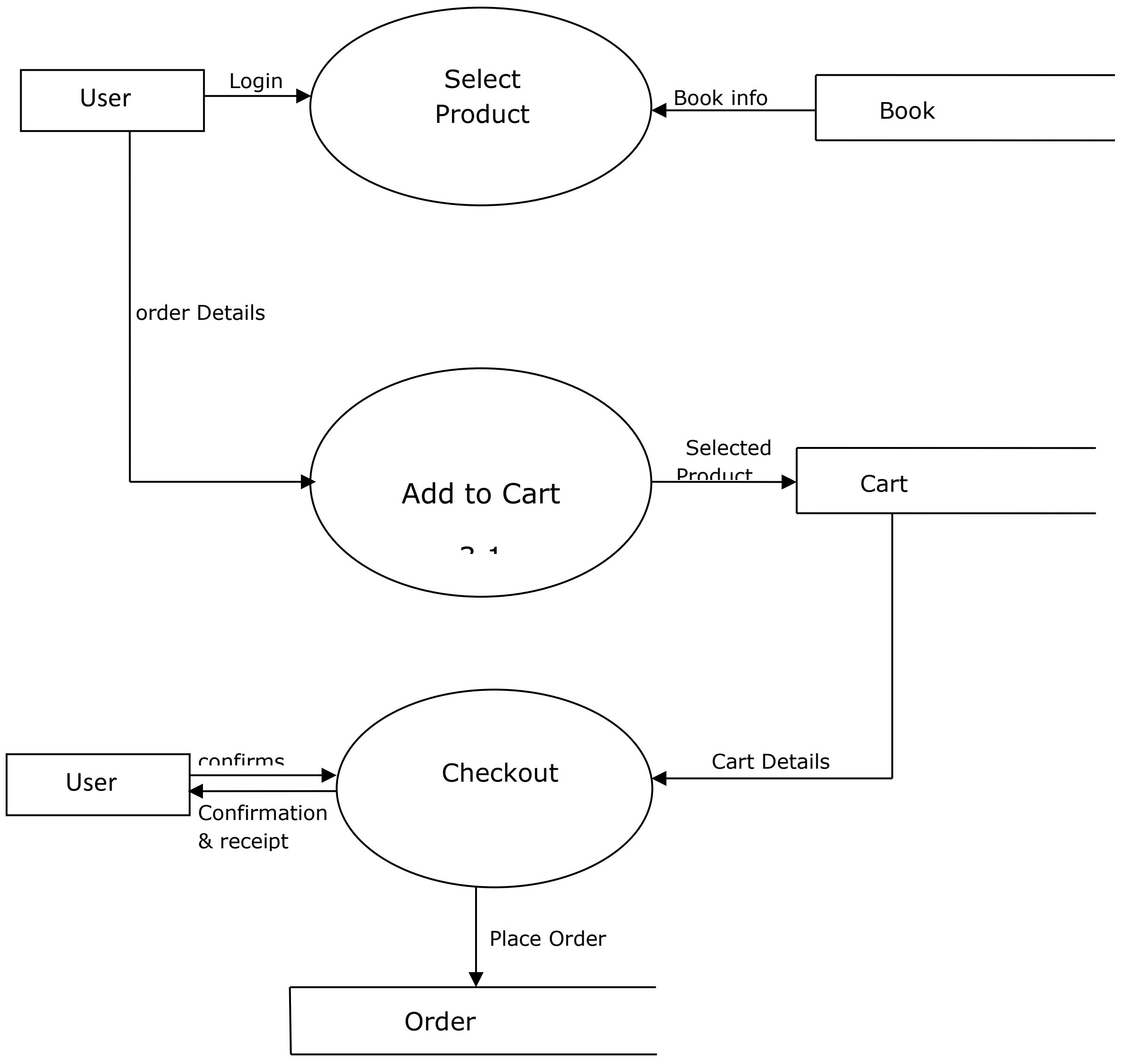
- File or data store is a repository of data. They contain data that is retained in the system. Process can enter data into data store or retrieved data from the data store. An open rectangle is a data store, data at rest.

0-Level DFD:





<u>DFD For User Registration and Profile Update</u>



DFD for shopping and checkout process

3.3.3 Entity-Relationship Model

Simply stated the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represent data objects. Since Chen wrote his paper the model has been extended and today it is commonly used for database.

Basic Constructs of E-R Modelling

The ER model views the real world as a construct of entities and association between entities.

Entities

Entities are the principal data object about which information is to be collected. Entities are classified as independent or dependent (in some methodologies, the terms used are strong and weak, respectively). An independent entity is one that does not rely on another for identification. A dependent entity is one that relies on another for identification. .

Relationships

A Relationship represents an association between two or more entities. Relationships are classified in terms of degree, connectivity, cardinality, and existence.

Attributes

Attributes describe the entity of which they are associated. A particular instance of an attribute is a value. The domain of an attribute is the collection of all possible values an attribute can have. The domain of Name is a character string.

Classifying Relationships

Relationships are classified by their degree, connectivity, cardinality, direction, type, and existence. Not all modeling methodologies use all these classifications.

Degree of a Relationship

The degree of a relationship is the number of entities associated with the relationship. The n-ary relationship is the general form for degree n. Special cases are the binary, and ternary, where the degree is 2 and 3 respectively.

Connectivity and Cardinality

The connectivity of a relationship describes the mapping of associated entity instances in the relationship. The values of connectivity are "one" or "many". The cardinality of a relationship is the actual number of related occurrences for each of the two entities. The basic types of connectivity for relations are: one-to-one, one-to-many, and many-to-many.

Direction

The direction of a relationship indicates the originating entity of a binary relationship. The entity from which a relationship originates is the parent entity; the entity where the relationship terminates is the child entity.

The direction of a relationship is determined by its connectivity type .An identifying relationship is one in which one of the child entities is also a dependent entity. A non-identifying relationship is one in which both entities are independent.

Existence

Existence denotes whether the existence of an entity instance is dependent upon the existence of another, related, entity instance. The existence of an entity in a relationship is defined as either mandatory or optional.

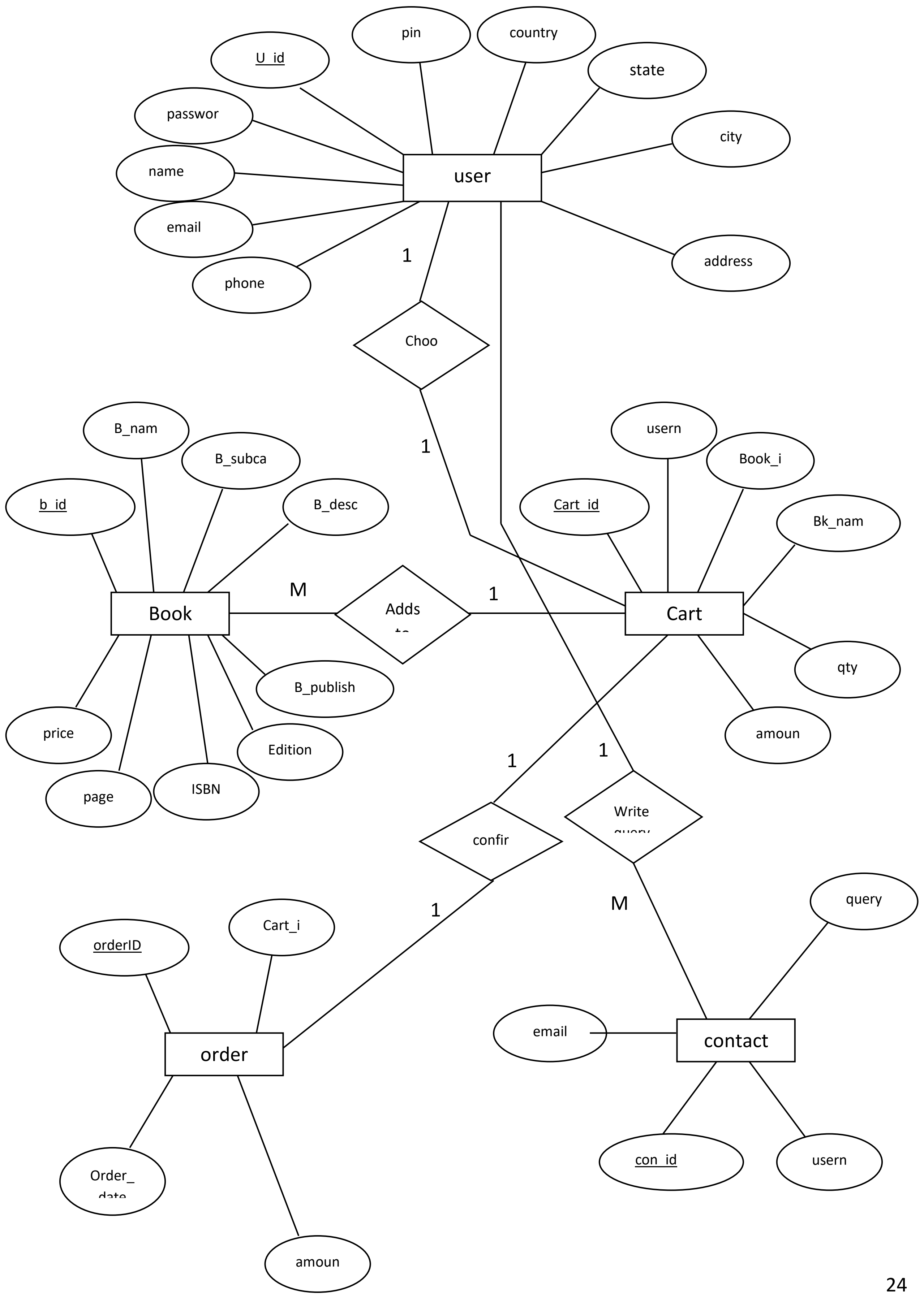
Generalization Hierarchies

A generalization hierarchy is a form of abstraction that specifies that two or more entities that share common attributes can be generalized into a higher-level entity type called a supertype or generic entity. The lower-level of entities become the subtype, or categories, to the supertype. Subtypes are dependent entities.

ER Notation

The symbols used for the basic ER constructs are:

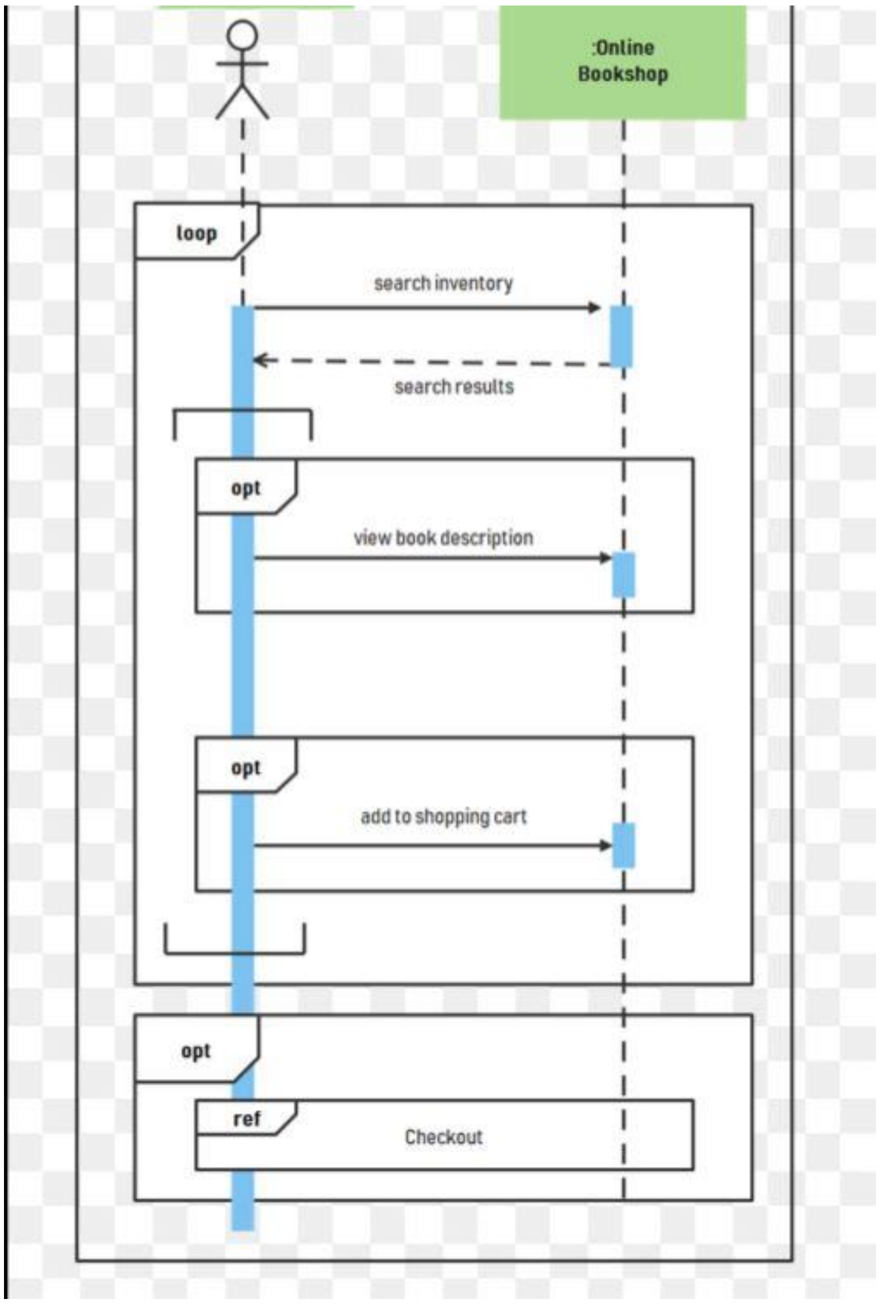
- Entities are represented by labelled rectangles. The label is the name of the entity.
- Relationships are represented by a solid line connecting two entities. The name of the relationship is written above the line. Relationship names should be verbs.
- Attributes, when included, are listed inside the entity rectangle. Attributes which are identifiers are underlined. Attribute names should be singular nouns.
- Cardinality of many is represented by a line ending in a crow's foot. If the crow's foot is omitted, the cardinality is one.
- Existence is represented by placing a circle or a perpendicular bar on the line. Mandatory existence is shown by the bar (looks like a 1) next to the entity for an instance is required. Optional existence is shown by placing a circle next to the entity that is optional.
- Existence is represented by placing a circle or a perpendicular bar on the line. Mandatory existence is shown by the bar (looks like a 1) next to the entity for an instance is required. Optional existence is shown by placing a circle next to the entity that is optional.



U_id
pin
country
state
passwor
city
name
user
email
address
phone
1
Choo
1
B_nam
usern
B_subca
Book_i
b_id
B_desc
Cart_id
Bk_nam
M
1
Book
Adds to
Cart
B_publish
qty
price
Edition
amoun
1
1
page
ISBN
Write query
confir
query
1
M
orderID
Cart_i
email
contact
order
Order_date
con_id
usern
amoun

3.4SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.



3.5 DATABASE DESIGN

The general theme behind a database is to handle information as an integrated whole. A database is a collection of inter-related data stored with minimum redundancy to serve single users quickly and efficiently. The general objective is to make information necessary, quick, inexpensive, and flexible for the user.

Database Tables

user Table

Field Name	**Data Type**	**Size**	**Allow Null**	**Constrain**
u_id	int	4	No	PK
u_fnm	varchar	35	No	
u_unm	varchar	25	No	
u_pwd	varchar	20	No	
u_gender	varchar	7	No	
u_email	varchar	35	No	
u_contact	varchar	12	No	
u_city	varchar	20	No	

Category Table

Field Name	**Data Type**	**Size**	**Allow Null**	**Constrain**
cat_id	int	4	No	PK
cat_nm	varchar	30	No	

Subcat Table

Field Name	Data Type	Size	Allow Null	Constrain
subcat_id	int	4	No	PK
Parent_id	Int	4	No	
Subcat_nm	varchar	35	No	

Book Table

Field Name	Data Type	Size	Allow Null	Constrain
b_id	int	4	No	PK
b_nm	varchar	60	No	
b_subcat	varchar	25	No	
b_desc	longtext	0	No	
b_publisher	varchar	40	No	
b_edition	varchar	20	No	
b_isbn	varchar	10	No	
b_page	int	5	No	
b_price	int	5	No	
b_img	longtext	0	No	
b_pdf	longtext	0	No	

Cart Table

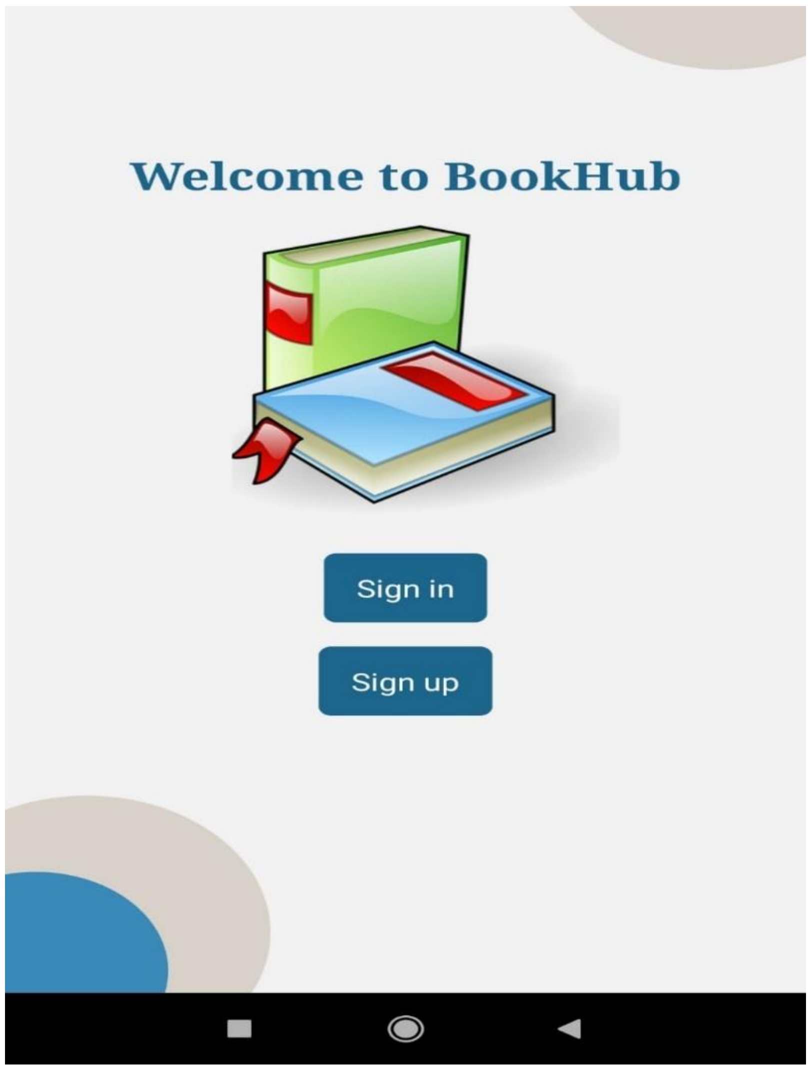
Field Name	**Data Type**	**Size**	**Allow Null**	**Constrain**
cart_id	int	4	No	PK
user_nm	varchar	20	No	
Book_id	varchar	10	No	PK
Book_name	varchar	25	No	
qty	int	4	No	
Amount	Float		No	

Checkout Table

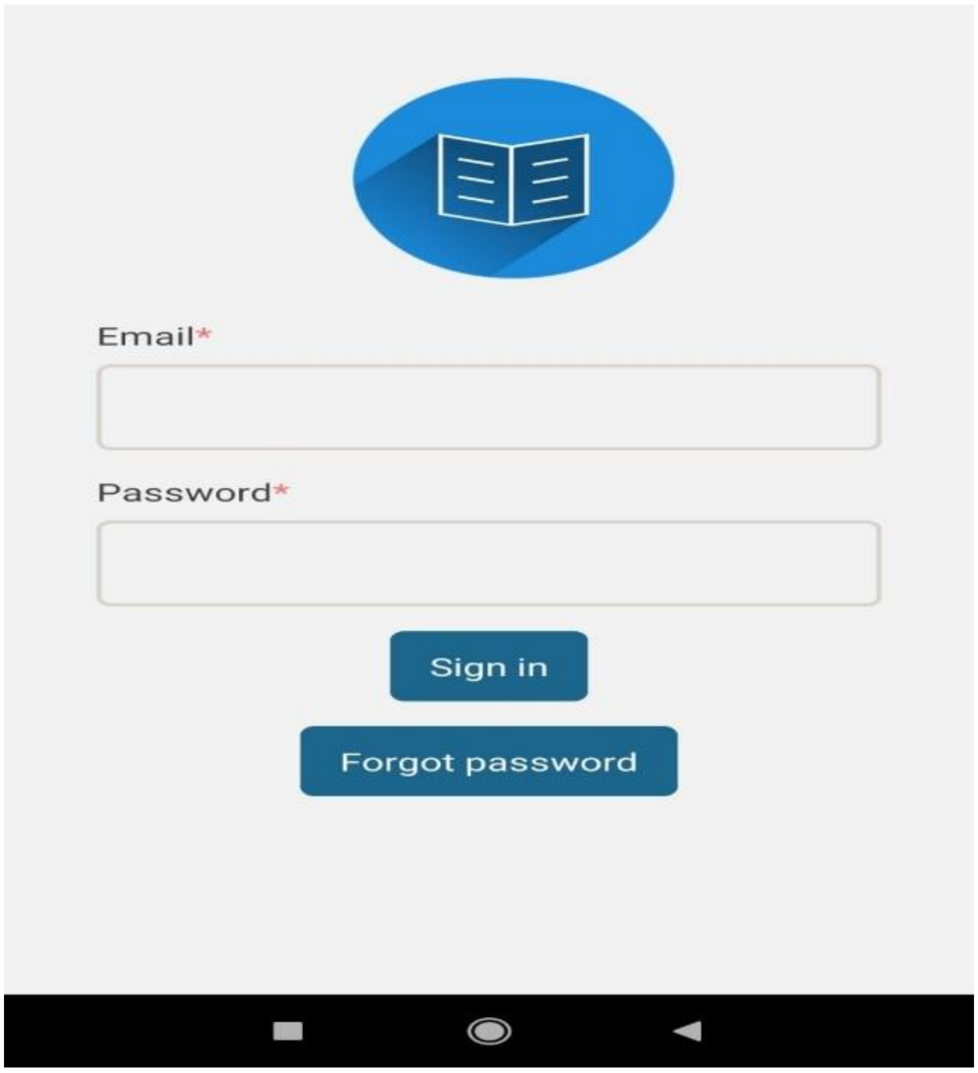
Field Name	**Data Type**	**Size**	**Allow Null**	**Constrain**
order_id	int	4	No	PK
Cart_id	varchar	60	No	
Order_date	datetime		No	
Total_Amount	Float	0	No	

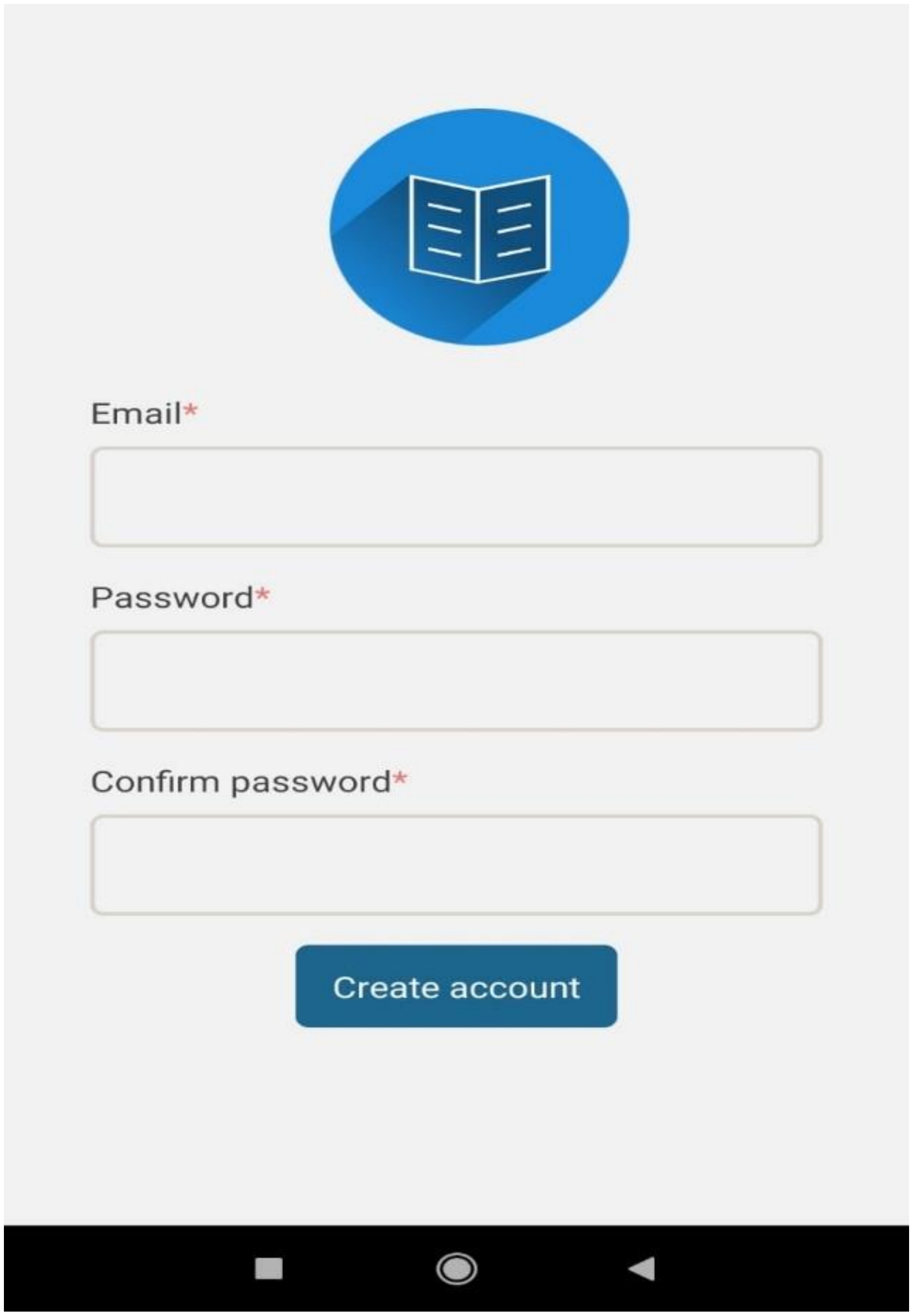
4.IMPLEMENTATION

4.1Home Page

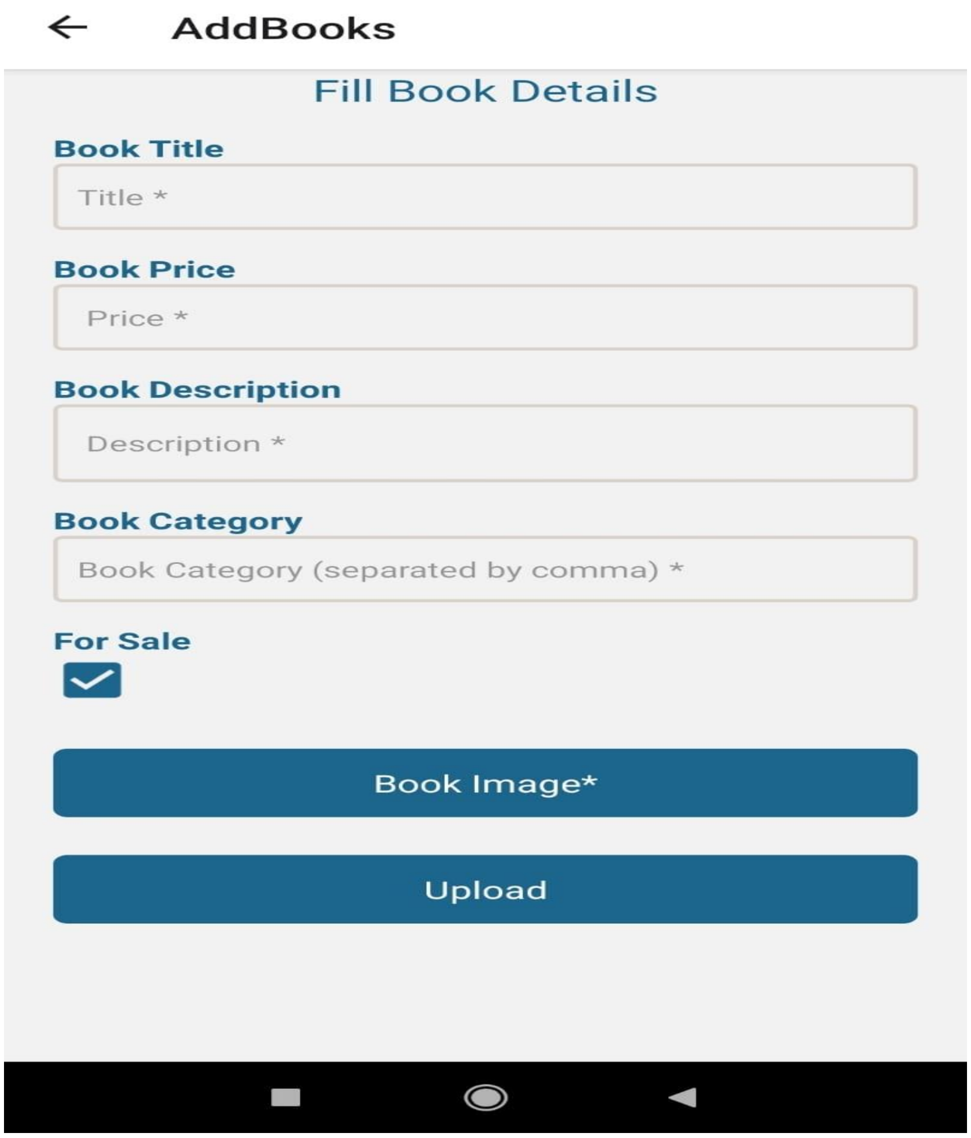


4.2 Sign in/ Sign up

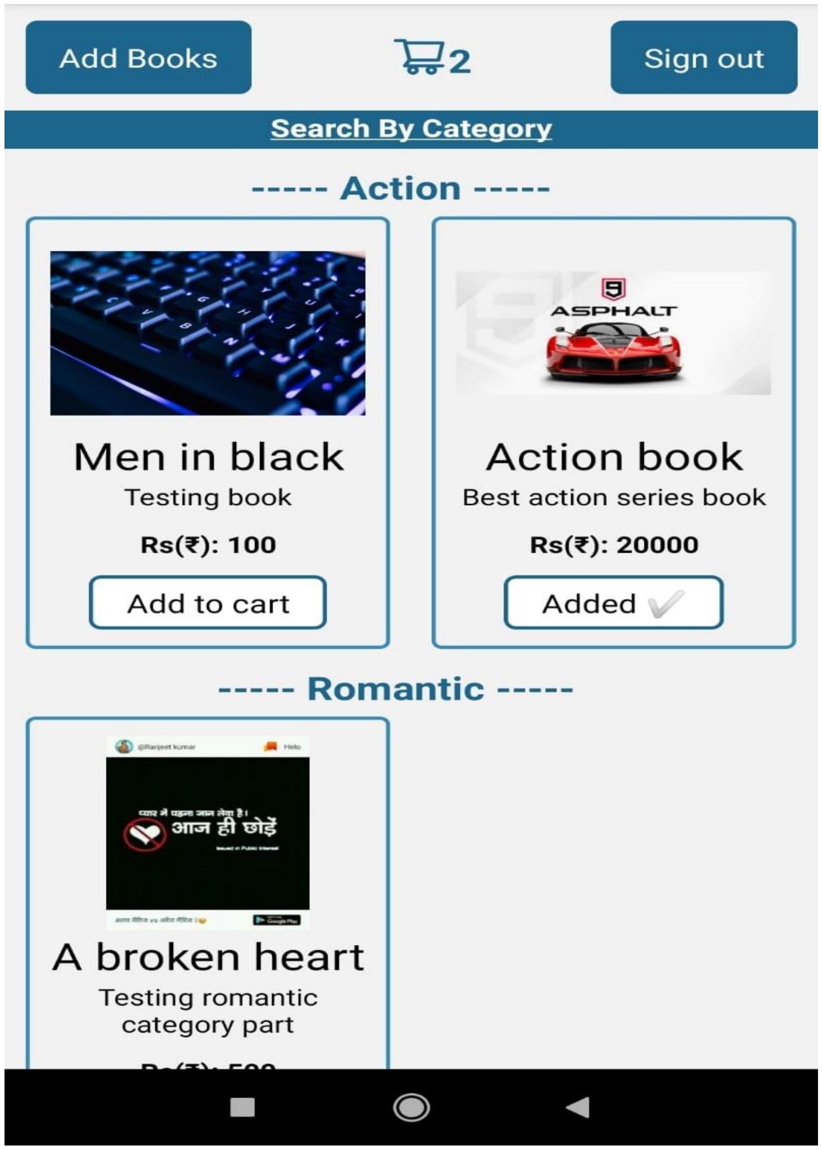



Email*
Password*
Confirm password*
Create account

4.3 Add book


←
AddBooks
Fill Book Details
Book Title
Title *
Book Price
Price *
Book Description
Description *
Book Category
Book Category (separated by comma) *
For Sale
Book Image*
Upload

4.4Search Book



4.5 Add to cart

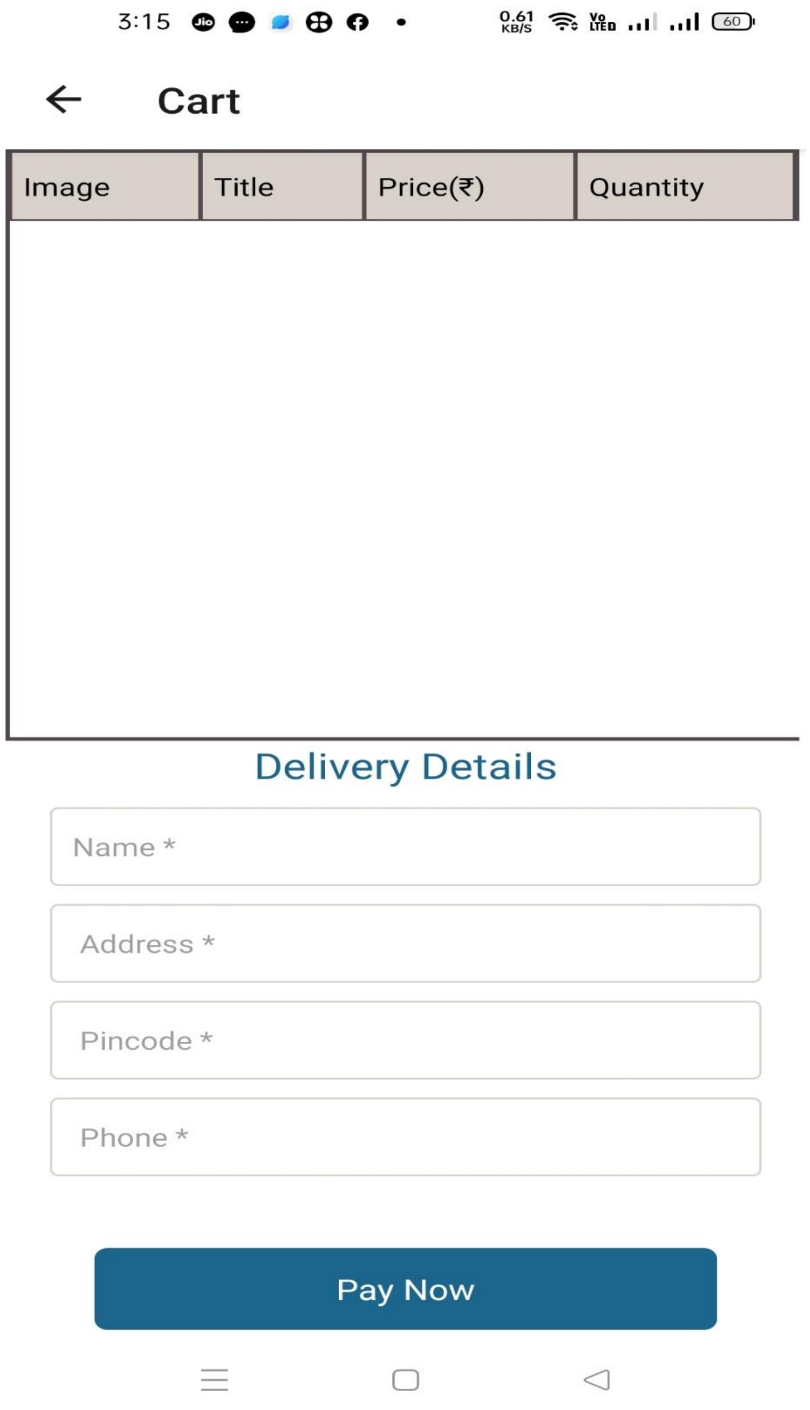


Image	Title	Price(₹)	Quantity

Delivery Details

Name *

Address *

Pincode *

Phone *

5.CODING

5.1 GUEST WELCOME

```
import React from 'react'

import PropTypes from 'prop-types'

import { VStack, Center, Button } from 'native-base'

import { Image, View, Text } from 'react-native';

export const GuestWelcomeScreen = ({ navigation }) => {

  const handlePressOnSignIn = () => {

    navigation.navigate('SignIn')

  }


  const handlePressOnSignUp = () => {

    navigation.navigate('SignUp')

  }


  return (

    <Center flex={1}>

      <View
style={{width:180,height:180,backgroundColor:"#D8D2CB",borderRadius:90,position:'absolute'
,top:-50,right:-50}}>

       <View
style={{width:130,height:130,backgroundColor:"#398AB9",borderRadius:65,position:"absolute
",top:0,right:0}}></View>

      </View>

         <View
style={{width:180,height:180,backgroundColor:"#D8D2CB",borderRadius:90,position:'absolute'
,bottom:-50,left:-50}}>
```

```
       <View
style={{width:130,height:130,backgroundColor:"#398AB9",borderRadius:65,position:"absolute
",bottom:0,left:0}}></View>



      </View>

      <VStack space={4} alignItems="center" w="90%">

      <Text
style={{color:"#1C658C",fontSize:25,fontWeight:"bold",fontFamily:"serif"}}>Welcome to
BookHub</Text>

     <Image source={require('../../../../assets/book.png')}
style={{width:190,height:190,marginBottom:10,marginLeft:20}} />

        <Center>

          <Button bg="#1c658c" onPress={handlePressOnSignIn}>Sign in</Button>

        </Center>

        <Center>

          <Button bg="#1c658c" onPress={handlePressOnSignUp}>Sign up</Button>

        </Center>

      </VStack>

    </Center>

  )

}



GuestWelcomeScreen.propTypes = {

  navigation: PropTypes.object.isRequired,

}
```

5.2 SIGN UP

- **Hooks(signup.js)**

```
import { useState } from 'react'
import { signUp } from '../../../api/user'

export const useSignUp = () => {
  const [state, setState] = useState({
    isLoading: false,
    error: null,
  })

  const handleSignUp = async (values) => {
    console.log("inside handleSignUp",values)
    setState({ isLoading: true, error: null })

    try {
      await signUp(values)
      setState({ isLoading: false, error: null })
    } catch (error) {
      setState({ isLoading: false, error })
    }
  }

  return [handleSignUp, { ...state }]
}
```

- **Screens**

```
import React from 'react'

import { Center, VStack } from 'native-base'
import { useSignUp } from '../hooks/use-sign-up'
import { ErrorMessage } from '../../../components/ErrorMessage'
import { EmailAndPasswordForm } from '../../../components/EmailAndPasswordForm'

export const SignUpScreen = () => {
  const [signUp, { isLoading, error }] = useSignUp()

  return (
    <Center flex={1}>

      <VStack space={4} alignItems="center" w="90%">
        <ErrorMessage error={error} />
        <EmailAndPasswordForm
          onSubmit={signUp}
          isLoading={isLoading}
          withPasswordConfirmation={true}
        />
      </VStack>
    </Center>
  )
}
```

5.3 Sign In

- **Hooks**

```
import { useState } from 'react'
import { signIn } from '../../../api/user'

export const useSignIn = () => {
  const [state, setState] = useState({
    isLoading: false,
    error: null,
  })

  const handleSignIn = async (values) => {
    setState({ isLoading: true, error: null })

    try {
      await signIn(values)
      setState({ isLoading: false, error: null })
    } catch (error) {
      setState({ isLoading: false, error })
    }
  }

  return [handleSignIn, { ...state }]
}
```

- **Screen**

```
import React from 'react'
import PropTypes from 'prop-types'
import { Center, VStack, Button } from 'native-base'
import { useSignIn } from '../hooks/use-sign-in'
import { ErrorMessage } from '../../../components/ErrorMessage'
import { EmailAndPasswordForm } from '../../../components/EmailAndPasswordForm'

export const SignInScreen = ({ navigation }) => {
  const [signIn, { isLoading, error }] = useSignIn()

  const handlePressOnForgotPassword = () => {
    navigation.navigate('ForgotPassword')
  }

  return (
    <Center flex={1}>
      <VStack space={4} alignItems="center" w="90%">
        <ErrorMessage error={error} />
        <EmailAndPasswordForm
          onSubmit={signIn}
          isLoading={isLoading}
          buttonText="Sign in"
        />
        <Button        bg="#1c658c"        onPress={handlePressOnForgotPassword}>Forgot
password</Button>
      </VStack>
    </Center>
  )
}

SignInScreen.propTypes = {
  navigation: PropTypes.object.isRequired,
}
```

5.4 ADD BOOK

```
 import React from 'react'
import PropTypes from 'prop-types'
import { VStack, Center, Button } from 'native-base'
import { Alert, TextInput, View, StyleSheet,Text,KeyboardAvoidingView ,
  Image,ScrollView, Dimensions,TouchableOpacity,ActivityIndicator} from 'react-native';
  import Checkbox from 'expo-checkbox';
import * as DocumentPicker from 'expo-document-picker';
import {Picker} from '@react-native-picker/picker';
import axios from 'axios';
import AsyncStorage from '@react-native-async-storage/async-storage';
import * as FileSystem from 'expo-file-system';
export default AddBook = ({ navigation }) => {
  const [state,setstate] = React.useState({'for_sale':true})
   const [ btnloader, setbtnloader ] = React.useState(false);
   const [image, setImage] = React.useState(null);


  const handlePressOnSignIn = () => {
    navigation.navigate('SignIn')
  }
  const handlePressOnSignUp = () => {
    navigation.navigate('SignUp')
  }
  const handlechange=(val,key)=>{
    let temp=state
    if(key == "name" || key == "email" || key=="category")
    temp[key]= val.toLowerCase()
    else
    temp[key]= val

    setstate({...temp})
    console.log(state,"state is ======")

  }


const encode = (uri) => {
  if (Platform.OS === 'android') return encodeURI(`${uri}`)
  else return uri
}
  const handleUpload=  async (e)=>{

    console.log("inside handl upload============")

      console.log("before doc picker")

     let result = await DocumentPicker.getDocumentAsync({ type:'*/*'})
     // let uri = encode(result.uri)
     // const uri = FileSystem.documentDirectory+result.name;
     let uri = result.uri
console.log("uri is ",uri)



 //     .then(async (result)=>{
 //   console.log(result)
 //    console.log("+++++++++++")
 //    console.log("+++++++++++")
 //    console.log("+++++++++++")
 //    console.log("+++++++++++")

 const fileInfo = await FileSystem.getInfoAsync(uri);
 console.log("fileinfo is ",fileInfo)
 // setImage(result.uri)
    let fileBase64 = await FileSystem.readAsStringAsync(uri, { encoding: 'base64'  });
    // console.log(fileBase64)
    result.uri=`data:image/${result.uri.split(".").pop().toLowerCase()};base64,`+fileBase64
    // console.log("+++++++++++base 64 is ",result)
      // setImage(result.uri)
    setstate({...state,resume:result})
//     }).catch(err=>{
//       console.log(err)
//     })

    // console.log("result i s",result)


  }




   const onSubmit=()=> {
    console.log("state is",state)
    //     && state.title && state.price && state.category && state.description
   if(Object.keys(state).length >0 && state.title && state.price && state.category &&
state.description && state.resume && state.resume.uri)
    {
     setbtnloader(true)


         var formData = new FormData();
       var list="123456789abcdefghijklmnopqrstuvwxyz";
   var name='';
   for(let i=0;i<15;i++)
   {
     name+=list.charAt(Math.floor(Math.random() * list.length))

   }
    //JSON.stringify({"uri":state.resume.uri,"type":`application/pdf`,"name":"mypdf.pdf"})
    // if(!state.resume.name.split(".").pop().endsWith("jpg"))
    formData.append("file",state.resume.uri)
  // else
  //     formData.append("file",state.resume)
     formData.append("upload_preset","bookhub")
    formData.append("cloud_name","dfk9dkjuf")
    // console.log("state resume file=================", state.resume)


    formData.append("public_id",`${name}.${state.resume.name.split(".").pop()}`)

       fetch("https://api.cloudinary.com/v1_1/dfk9dkjuf/auto/upload", {
      method: "post",
      body: formData
    }).then(res => res.json()).
      then(data => {
        // auth_user=props.route.params.auth_user
        // console.log("data uploadewd",JSON.stringify(data))
        console.log("++++++++++++++++++++++++++++++++++++++++++++++")
        console.log("++++++++++++++++++++++++++++++++++++++++++++++")

        console.log("==================data
uploadewd====================",JSON.stringify(state))

        // console.log("data uploadewd",JSON.stringify(auth_user))


axios.post('https://madbookhub.herokuapp.com/api/save/book',{...state,secure_url:data.secure_ur
l,keywords:state.category.split(",")}).then(resp=>{
    console.log(resp.data)
    if(resp.data.success == 1)
    {
     setstate({'for_sale':true})
    console.log("++++++++++++++++++++++++++++++++++++++inside                         then
if",resp.data.message)
     Alert.alert("Details saved successfully")

    }
    else
    {
        console.log("++++++++++++++++++++++++++++++++++++++inside                     then
else",resp.data.message)
     Alert.alert("Erro occured during detail saving")
    }
     setbtnloader(false)
  }).catch(err=>{
    console.log(err)
     setbtnloader(false)
     Alert.alert("Some Technical Error occurred please try again")

  })
        // setPhoto(data.secure_url)
      }).catch(err => {
         setbtnloader(false)
        console.log("eror is ",err)
        Alert.alert("An Error Occured While Uploading")
      })
```

```
}

    else
    {
      Alert.alert("Please fill all the mandatory fields..")
    }

    // Alert.alert('Credentials', `${username} + ${password}`);
  }

  return (
    <>
    <View style={{paddingLeft:20,paddingRight:20}}>
          <Text            style={{textAlign:"center",fontSize:20,color:"#1C658C"}}>Fill            Book
Details</Text>

            <Text
style={{textAlign:"left",fontSize:15,fontWeight:"bold",marginTop:15,color:"#1C658C"}}>Boo
k Title</Text>
        <TextInput
          value={state.title?state.title:''}
          onChangeText={(username) => handlechange(username,"title")}
          placeholder={'Title *'}
          style={styles.input}
        />
            <Text
style={{textAlign:"left",fontSize:15,fontWeight:"bold",marginTop:5,color:"#1C658C"}}>Book
Price</Text>

            <TextInput
          value={state.price?state.price:''}
          onChangeText={(salary) => handlechange(salary,"price")}
          placeholder={' Price *'}
          keyboardType='numeric'
          style={styles.input}
        />
            <Text
style={{textAlign:"left",fontSize:15,fontWeight:"bold",marginTop:5,color:"#1C658C"}}>Book
Description</Text>

        <TextInput
          value={state.description?state.description:''}
          multiline={true}
          onChangeText={(email) => handlechange(email,"description")}
          placeholder={' Description *'}
          style={{...styles.input,minHeight:44,height:'auto'}}
        />
```

```
            <Text
style={{textAlign:"left",fontSize:15,fontWeight:"bold",marginTop:5,color:"#1C658C"}}>Book
Category</Text>

{/*          <TextInput
          value={state.category?state.category:''}
          onChangeText={(phone) => handlechange(phone,"category")}
          placeholder={'Book Category (separated by comma) *'}
          style={styles.input}
        />*/}
        <View
  style={{
  borderWidth: 2,
  borderRadius: 4,
  borderColor:'#D8D2CB',
}}>
                <Picker style={styles.pickerStyle}
                       selectedValue={state.category?state.category:''}
                       onValueChange={(itemValue, itemPosition) =>
                           handlechange(itemValue, "category")}
                   >
                   <Picker.Item label="Action" value="action" />
                   <Picker.Item label="Romance" value="romance" />
                   <Picker.Item label="Adventure" value="adventure" />
                </Picker>
                </View>

            <Text
style={{textAlign:"left",fontSize:15,fontWeight:"bold",marginTop:5,color:"#1C658C"}}>For
Sale</Text>

         <Checkbox
          value={state['for_sale']}
          onValueChange={(val)=>handlechange(val,'for_sale')}
          style={{transform: [{ scaleX: 1.3 }, { scaleY: 1.3 }]}}
           tintColors={{ true: '#1C658C', false: 'black' }}
        />

        <View>
<Text style={{marginTop:5}}> {state.resume && state.resume.uri ? <Text
style={{color:"green"}}> (File Uploaded)</Text>:null}</Text>
         <Button bg="#1c658c" style={{marginTop:5}} onPress={handleUpload}>
    Book Image*
      </Button>
        </View>
```

```
            <Button          bg="#1c658c"          style={{marginTop:25}}          onPress={onSubmit}
isLoading={btnloader}>
       Upload
     </Button>
       </View>
       </>

  )
}


const styles = StyleSheet.create({

    container: {
    // flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
    backgroundColor:"red",
    backgroundColor: '#fff',
    minHeight:Dimensions.get('window').height-135
  },
  input: {
    minWidth: 240,
    height: 44,
    padding: 10,
    borderWidth: 2,
    marginBottom:10,
    borderColor: '#D8D2CB',
    borderRadius:4
  },
      pickerStyle:{
        width: "100%",
        color: '#1C658C',

    }

});
```

5.5 CART

```
import { Table, TableWrapper, Cell, Row, Rows, Col, Cols } from 'react-native-table-component';
import { StatusBar } from 'expo-status-bar';
import React, { Component } from 'react';
import { Alert, TextInput, View,FlatList, StyleSheet,Text ,Image,ScrollView,
Dimensions,TouchableOpacity,ActivityIndicator} from 'react-native';
import axios from 'axios';
import {  Input, Button } from 'native-base'
import AsyncStorage from '@react-native-async-storage/async-storage';
import { Entypo,AntDesign  } from '@expo/vector-icons';
export default function Cart(props)
{
  const [ cart, setcart ] = React.useState({})
  const [ payment, setpayment ] = React.useState({})
  const [ total, settotal ] = React.useState(0)


       const [ state, setstate ] =
React.useState({tableHead:['Image','Title','Price(₹)','Quantity'],tableData: [
      ] });

   React.useEffect(()=>{
    if(props.route.params && props.route.params.cart)
      {
    let bookid = Object.keys(props.route.params.cart)
    console.log("book id is ",props)
    setcart(props.route.params.cart)
    // https://madbookhub.herokuapp.com/
    axios.post('https://madbookhub.herokuapp.com/api/find/book/',{bookid }).then(resp=>{
      if(resp.data.success == 1)
      {
        console.log(resp.data.books)

        setstate({...state,tableData:resp.data.books})

        let books = resp.data.books
        let count=0;
        for(let x=0;x<books.length;x++)
        {
          count+= + books[x].price
        }
        settotal(count)
        console.log("total count is =================",count)
        console.log("total count is =================",count)
        console.log("total count is =================",count)
        console.log("total count is =================",count)
        console.log("total count is =================",count)
        console.log("total count is =================",count)
        console.log("total count is =================",count)
        console.log("total count is =================",count)
        console.log("total count is =================",count)
        console.log("total count is =================",count)
        console.log("total count is =================",count)
        console.log("total count is =================",count)

      }
      console.log(resp.data)
    }).catch(err=>{
      console.log("Error is ",err)
    })

      }

   },[])

            const element = (data, index) => (
      <TouchableOpacity onPress={() => download(data)}>
        <View style={styles.btn}>
          <Text  style={styles.btnText}><Entypo  name="download"  size={20}  color="#004080"
/></Text>
        </View>
      </TouchableOpacity>
    );

        const handleIncDec=(bookid,type,price)=>{
          console.log("bookid is ",price)
          let temp=cart
          if(type=="inc")
          {
          temp[bookid]=cart[bookid]+1
          settotal(total+ +price)
          }
          else
          {
            if(cart[bookid]-1 >=0 )
            {
              temp[bookid]=cart[bookid]-1
              settotal(total- +price)
            }
          }
          setcart({...temp})
          console.log("cart is ",cart)

        }

        const handlechange=(val,key)=>{
    let temp=payment
    if(key == "name" || key == "address")
    temp[key]= val.toLowerCase()
    else
    temp[key]= val

    setpayment({...temp})
    console.log(payment)

        }
        const checkOut=async (type)=>{
          console.log("payment are ",cart)
          function savePayment()
          {
   axios.post('https://madbookhub.herokuapp.com/api/save/payment',{...payment,     method:type,
cart:JSON.stringify(cart) }).then(resp=>{
      if(resp.data.success == 1)
      {
       console.log("Order booked successfully")


      }
    }).catch(err=>{
      console.log("Error is ",err)
    })

          }
    if(type=="cod")
          {

           savePayment()
           Alert.alert("Order booked successfully")
             props.navigation.navigate("Home")
          }
          else
          {
        console.log("redirecting to payment page ...")
        savePayment()
        props.navigation.navigate('Payment')
          }

            try {
               // await AsyncStorage.removeItem("mycart")
               let mycart = await AsyncStorage.getItem("mycart")
              }
                  catch(err)
            {
              console.log(err)
            }
```

```
        }

       return(
    <>

          <View style={{backgroundColor:"#fff"}}>
     <ScrollView>
     <ScrollView horizontal={true} style={{width:Dimensions.get("window").width }} >
          <Table borderStyle={{borderWidth: 2, borderColor: '#524A4E'}}
style={{backgroundColor:"white",minHeight:330}}>
          <Row data={state.tableHead} style={styles.head} textStyle={styles.text}/>
          {

             state.tableData.map((cellData, cellIndex) => (
             <TableWrapper key={Math.random()} style={styles.row}>
               {

                 <React.Fragment key={Math.random()} >
                   <Cell key={Math.random()} data={(<Image
        style={{width:50,height:50}}
        source={{
          uri: cellData.image,
        }}
      />)} textStyle={{margin: 7}} />
                   <Cell key={Math.random()} data={[cellData.title]} textStyle={{margin:
7,fontWeight:"bold"}}/>
                   <Cell key={Math.random()} data={[cellData.price]} textStyle={{margin:
7,fontWeight:"bold"}}/>
                   <Cell key={Math.random()} data={(<View
style={{flexDirection:'row',alignItems:"center",overflow: "visible"}}><AntDesign
name="minuscircleo" size={30} color="#1C658C"
onPress={()=>{handleIncDec(cellData._id,"dec",cellData.price)}} /><Text
style={{fontSize:20}}> {cart[cellData._id]} </Text><AntDesign name="pluscircleo" size={30}
color="#1C658C" onPress={()=>{handleIncDec(cellData._id,"inc",cellData.price)}}
/></View>)} textStyle={{margin: 0,fontWeight:"bold"}}/>

                 </React.Fragment>

               }
             </TableWrapper>
           ))
          }



        </Table>
          </ScrollView>
          </ScrollView >
        <ScrollView>

          <Text style={{textAlign:"center",fontSize:20,color:'#1c658c'}}>Delivery Details</Text>
        <TextInput
          value={payment.name?payment.name:''}
          onChangeText={(name) => handlechange(name,"name")}
          placeholder={'Name *'}
          style={styles.input}
        />
            <TextInput
          value={payment.address?payment.address:''}
          onChangeText={(address) => handlechange(address,"address")}
          placeholder={' Address *'}
          style={styles.input}
        />
        <TextInput
          value={payment.pincode?payment.pincode:''}
          onChangeText={(pincode) => handlechange(pincode,"pincode")}
          placeholder={' Pincode *'}
            keyboardType='numeric'
           maxLength={6}
          style={styles.input}
        />

             <TextInput
          value={payment.phone?payment.phone:''}
          onChangeText={(phone) => handlechange(phone,"phone")}
          placeholder={' Phone *'}
            keyboardType='numeric'
           maxLength={10}
          style={styles.input}
        />

          <Button bg="#1c658c" mt="10" mx="10" isDisabled={total==0? true :false}
onPress={()=>checkOut("pay")}>{total ? `Pay Now ₹ ${total}` : "Pay Now"}</Button>


  <Button bg="#1c658c" mt="5" mx="10" mb="30" isDisabled={total==0? true :false}
onPress={()=>checkOut("cod")}>{total ? `Cash on delivery ₹ ${total}` : "Cash on
delivery"}</Button>




   </ScrollView>
         </View>

        </>
        )
}

const styles = StyleSheet.create({

 container: { padding: 5, paddingTop: 30, backgroundColor: '#fff',minHeight:Dimensions.get("window").height },
  // input: {
  //   width: 200,
  //   height: 44,
  //   padding: 10,
  //   borderWidth: 1,
  //   marginTop:10,
  //   borderColor: 'black',
  // },

    input: {
    minWidth: 240,
    height: 44,
    padding: 10,
    borderWidth: 1,
    marginTop:10,
    borderColor: '#D8D2CB',
    borderRadius:4,
    marginLeft:20,
    marginRight:20
  },
    head: { height: 40, backgroundColor: '#D8D2CB',borderColor:"#524A4E",borderWidth:1,borderTopWidth:0
,minWidth:Dimensions.get("window").width-5},
  text: { margin: 5 },
    row: { flexDirection: 'row', backgroundColor: '#EEEEEE',marginTop:5,padding:5},
    btn: { height: 20,  borderRadius: 2,maxWidth:80 },
  btnText: { textAlign: 'center', color: '#004080' }
});
```

5.6 EMAIL VERIFICATION

- **Hooks**

```
    import { useState } from 'react'
import { sendVerification } from '../../../api/user'

export const useSendVerification = () => {
  const [state, setState] = useState({
    isLoading: false,
    error: null,
  })

  const handleSendVerification = async () => {
    setState({ isLoading: true, error: null })

    try {
      await sendVerification()
      setState({ isLoading: false, error: null })
    } catch (error) {
      setState({ isLoading: false, error })
    }
  }

  return [handleSendVerification, { ...state }]
}
```

- **Screen**

```
 import React from 'react'
import { VStack, Center, Heading, Text, Button } from 'native-base'
import { useSendVerification } from '../hooks/use-send-verification'
import { useUserContext } from '../../../context/UserContext'
import { useSignOut } from '../../../hooks/use-sign-out'

export const VerifyEmailScreen = () => {
  const { reload: reloadUser, isLoading: isReloadingUser } = useUserContext()
  const [sendVerification, { isLoading: isResendingVerification }] =
    useSendVerification()
  const [signOut, { isLoading: isSigninOut }] = useSignOut()

  return (
    <Center flex={1}>
      <VStack space={4} alignItems="center" w="90%">
        <Heading>Check your email</Heading>
        <Text>
          We sent you an email with instructions on how to verify your email
          address. Click on the link in the email to get started.
        </Text>
        <Button onPress={reloadUser} isLoading={isReloadingUser}>
          Done
        </Button>
        <Button onPress={sendVerification} isLoading={isResendingVerification}>
          Resend
        </Button>
        <Button onPress={signOut} isLoading={isSigninOut}>
          Cancel
        </Button>
      </VStack>
```

```
        </Center>
      )
    }
```

5.7 PaymentUI

```
import React, { useEffect, useState } from 'react';
import { Alert,Text } from 'react-native';
import { useStripe } from '@stripe/stripe-react-native';
import * as stripe from '@stripe/stripe-react-native';
import Button from './Button';
import PaymentScreen from './PaymentScreen';
import { API_URL } from './Config';

export default function PaymentsUICompleteScreen({ navigation }) {
  const { initPaymentSheet, presentPaymentSheet } = useStripe();
  const [paymentSheetEnabled, setPaymentSheetEnabled] = useState(false);
  const [loading, setLoadng] = useState(false);
  const [clientSecret, setClientSecret] = useState();

  const fetchPaymentSheetParams = async () => {
    const response = await fetch(`${API_URL}/payment-sheet`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
    });
    const { paymentIntent, ephemeralKey, customer } = await response.json();
    console.log(customer)
    console.log(paymentIntent)
//     const pi = await stripe.paymentIntents.create({
//   amount: 1099,
//   currency: 'usd',
//   payment_method_types: ['card'],
// });

//     console.log("pi is ==================",pi)


    setClientSecret(paymentIntent);
    return {
      paymentIntent,
      ephemeralKey,
      customer
    };
  };
```

```
const openPaymentSheet = async () => {
  if (!clientSecret) {
    return;
  }
  setLoadng(true);

  const data = await presentPaymentSheet({
    clientSecret,
  });
  console.log("client secret is ",clientSecret,data)

  if (data.error) {
    console.log(`Error code: ${data.error.code}`, data.error.message)
    Alert.alert(`Error code: ${data.error.code}`, data.error.message);
  } else {
    // Alert.alert('Success', 'The payment was confirmed successfully');
    Alert.alert("payment confirmed ")
    // navigation.navigate("Home")

    // Alert.alert(JSON.stringify(data))
  }

  setPaymentSheetEnabled(false);
  setLoadng(false);
  console.log("set btn loading false",loading)
};

const initialisePaymentSheet = async () => {
  const {
    paymentIntent,
    ephemeralKey,
    customer,
  } = await fetchPaymentSheetParams();
  console.log("intent",paymentIntent,ephemeralKey,customer)

  const { error } = await initPaymentSheet({
    customerId: customer,
    customerEphemeralKeySecret: ephemeralKey,
    paymentIntentClientSecret: paymentIntent,
    customFlow: false,
    merchantDisplayName: 'BookHub Team MNS',
    style: 'alwaysDark',
  });
  if (!error) {
      console.log("no error occured")
      setPaymentSheetEnabled(true);
    }
    else{
      console.log("Error occured ",err)
    }
  };

  useEffect(() => {
    // In your app’s checkout, make a network request to the backend and initialize PaymentSheet.
    // To reduce loading time, make this request before the Checkout button is tapped, e.g. when the
screen is loaded.
    initialisePaymentSheet();

    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, []);

  return (
    <>
        <PaymentScreen>
      <Button
        variant="primary"
        loading={loading}
        disabled={!paymentSheetEnabled}
        title="Checkout"
        onPress={openPaymentSheet}
      />
    </PaymentScreen>
    </>
  );
}
```

5.8 Button.js

```
import React from 'react';
import {
  AccessibilityProps,
  ActivityIndicator,
  StyleSheet,
  Text,
  TouchableOpacity,
  View,
} from 'react-native';
import { colors } from './colors';

type Props = AccessibilityProps & {
  title?: string | React.ReactElement;
  variant?: 'default' | 'primary';
  disabled?: boolean;
  loading?: boolean;
  onPress(): void;
};

export default function Button({
  title,
  variant = 'default',
  disabled,
  loading,
  onPress,
  ...props
}) {
  const titleElement = React.isValidElement(title) ? (
    title
  ) : (
    <Text style={[styles.text, variant === 'primary' && styles.textPrimary]}>
      {title}
    </Text>
  );
  return (
    <View style={disabled && styles.disabled}>
      <TouchableOpacity
        disabled={disabled}
        style={[
          styles.container,
          variant === 'primary' && styles.primaryContainer,
        ]}
        onPress={onPress}
        {...props}
      >
        {loading ? (
          <ActivityIndicator color={colors.white} size="small" />
        ) : (
          titleElement
        )}
      </TouchableOpacity>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    paddingVertical: 12,
    borderRadius: 12,
  },
  primaryContainer: {
    backgroundColor: colors.slate,
    alignItems: 'center',
  },
  text: {
    color: colors.slate,
    fontWeight: '600',
    fontSize: 16,
  },
  textPrimary: {
    color: colors.white,
  },
  disabled: {
    opacity: 0.3,
  },
});
```

5.9 EmailandPasswordForm

```
import React from 'react'
import PropTypes from 'prop-types'
import {Text, Image} from 'react-native'
import { VStack, FormControl, Input, Button } from 'native-base'
import { useFormik, getIn } from 'formik'
import * as Yup from 'yup'

const buildValidationSchema = (withPasswordConfirmation) =>
  Yup.object({
    email: Yup.string().email().required("Please enter email"),
    password: Yup.string().required().min(6,"Your password must be longer than 5 characters."),
      //     email: Yup.string()
      //   .email("Invalid email address")
      //   .required("Please enter email"),
      // password: Yup.string().required("Please enter password"),
    // Optionally require password confirmation
    ...(withPasswordConfirmation && {
      passwordConfirmation: Yup.string()
        .oneOf([Yup.ref('password'), null])
        .required(),
    }),
  })

export const EmailAndPasswordForm = ({
  buttonText = 'Create account',
  isLoading,
  onSubmit,
  withPasswordConfirmation = false,
}) => {
  const { handleChange, handleBlur, handleSubmit, values, touched, errors } =
    useFormik({

      initialValues: {
        email: '',
        password: '',
        ...(withPasswordConfirmation && { passwordConfirmation: '' }),
      },
      validationSchema: buildValidationSchema(withPasswordConfirmation),
      onSubmit,

         enableReinitialize:false
    })

  return (
    <VStack space={4} alignItems="center" w="90%">
     <Image                                                    source={require('../../assets/booklogo.png')}
style={{width:130,height:130,marginBottom:10,marginLeft:20}} />

      <FormControl
        isRequired
        isInvalid={getIn(errors, 'email') && getIn(touched, 'email')}
      >
        <FormControl.Label >Email</FormControl.Label>
        <Input
          autoCapitalize="none"
          keyboardType="email-address"
          onBlur={handleBlur('email')}
          onChangeText={handleChange('email')}
          value={values.email}
           style={{borderWidth:2,borderColor:"#D8D2CB"}}
        />
        {errors && errors.email ? <Text style={{color:"red"}}>{errors.email}</Text>: null}
      </FormControl>

      <FormControl
        isRequired
        isInvalid={getIn(errors, 'password') && getIn(touched, 'password')}
      >
        <FormControl.Label>Password</FormControl.Label>
        <Input
          autoCapitalize="none"
          secureTextEntry
          autoCorrect={false}
          autoCompleteType="password"
          onBlur={handleBlur('password')}
          onChangeText={handleChange('password')}
          value={values.password}
          style={{borderWidth:2,borderColor:"#D8D2CB"}}
        />
         {errors && errors.password &&errors.password.toLowerCase() != 'password is a required
field' ? <Text style={{color:"red"}}>{errors.password}</Text>: null}
      </FormControl>

      {withPasswordConfirmation && (
        <FormControl
          isRequired
          isInvalid={
            getIn(errors, 'passwordConfirmation') &&
            getIn(touched, 'passwordConfirmation')
          }
        >
          <FormControl.Label>Confirm password</FormControl.Label>
          <Input
            autoCapitalize="none"
            secureTextEntry
            autoCorrect={false}
            autoCompleteType="password"
            onBlur={handleBlur('passwordConfirmation')}
            onChangeText={handleChange('passwordConfirmation')}
            value={values.passwordConfirmation}
 style={{borderWidth:2,borderColor:"#D8D2CB"}}
          />
        </FormControl>
      )}

      <Button bg="#1c658c" onPress={handleSubmit} isLoading={isLoading}>
        {buttonText}
      </Button>
    </VStack>
  )
}

EmailAndPasswordForm.propTypes = {
  buttonText: PropTypes.string,
  isLoading: PropTypes.bool.isRequired,
  onSubmit: PropTypes.func.isRequired,
  withPasswordConfirmation: PropTypes.bool,
}
```

5.10 ErrorMessage

```
import React from 'react'
import PropTypes from 'prop-types'
import { Alert } from 'native-base'

export const ErrorMessage = ({ error }) => {
  if (!error) return null

  return (
    <Alert status="error">
      <Alert.Icon />
      <Alert.Title>{error.message}</Alert.Title>
    </Alert>
  )
}

ErrorMessage.propTypes = {
  error: PropTypes.object,
}
```

5.10 PaymentScreen

```
import { initStripe } from '@stripe/stripe-react-native';
import React, { useEffect, useState } from 'react';
import { ActivityIndicator, ScrollView, StyleSheet, Text } from 'react-native';
import { colors } from './colors';
import { fetchPublishableKey } from './helpers';


const PaymentScreen = ({ paymentMethod, children }) => {
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    async function initialize() {
      const publishableKey = await fetchPublishableKey(paymentMethod);
          console.log("children are ",children)
      if (publishableKey) {
        await initStripe({
          publishableKey,
          merchantIdentifier: 'merchant.com.stripe.react.native',
          urlScheme: 'stripe-example',
          setUrlSchemeOnAndroid: true,
        });
        setLoading(false);
      }
    }
    initialize();
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, []);

  return loading ? (
    <ActivityIndicator size="large" style={StyleSheet.absoluteFill} />
  ) : (
    <ScrollView
      accessibilityLabel="payment-screen"
      style={styles.container}
      keyboardShouldPersistTaps="handled">
      {children}
      {/* eslint-disable-next-line react-native/no-inline-styles */}
      <Text style={{ opacity: 0 }}>appium fix</Text>
    </ScrollView>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: colors.white,
    paddingTop: 20,
    paddingHorizontal: 16,
  },
});

export default PaymentScreen;
```

5.11 Helpers

```
import { Alert } from 'react-native';
import { API_URL } from './Config';

export async function fetchPublishableKey(
  paymentMethod
){
  try {
    const response = await fetch(
      `${API_URL}/stripe-key?paymentMethod=${paymentMethod}`
    );

    const { publishableKey } = await response.json();

    return publishableKey;
  } catch (e) {
    console.warn('Unable to fetch publishable key. Is your server running?');
    Alert.alert(
      'Error',
      'Unable to fetch publishable key. Is your server running?'
    );
    return null;
  }
}
```

5.12 UserContext

```
import React, { createContext, useContext, useEffect, useState } from 'react'
import PropTypes from 'prop-types'
import { onAuthStateChanged, reload, getUser } from '../api/user'

const UserContext = createContext()

export const UserContextProvider = ({
  children,
  initialState = { user: null, isLoading: true, error: null },
}) => {
  const [state, setState] = useState(initialState)

  // Listen to Firebase authentication state changes
  useEffect(() => {
    const unsubscribe = onAuthStateChanged((user) => {
      setState({ user, isLoading: false, error: null })
    })

    return () => {
      unsubscribe()
    }
  }, [])

  const handleReload = async () => {
    try {
      await reload()
      const user = getUser()
      setState({ user, isLoading: false, error: null })
    } catch (error) {
      setState({ user: null, isLoading: false, error })
    }
  }

  const value = {
    ...state,
    reload: handleReload,
  }

  return <UserContext.Provider value={value}>{children}</UserContext.Provider>
}

UserContextProvider.propTypes = {
  children: PropTypes.node,
  initialState: PropTypes.object,
}

export const useUserContext = () => {
  const userContext = useContext(UserContext)

  if (typeof userContext === 'undefined') {
    throw new Error('useUserContext must be used within a UserContextProvider')
  }

  return userContext
}
```

5.13 ForgetPassword

- **Components**

```
import React from 'react'
import PropTypes from 'prop-types'
import { VStack, FormControl, Input, Button } from 'native-base'
import { useFormik, getIn } from 'formik'
import * as Yup from 'yup'

const validationSchema = Yup.object({
  email: Yup.string().email().required(),
})

export const EmailForm = ({ isLoading, onSubmit }) => {
  const { handleChange, handleBlur, handleSubmit, values, touched, errors } =
    useFormik({
      initialValues: { email: '' },
      validationSchema,
      onSubmit,
    })

  return (
    <VStack space={4} alignItems="center" w="100%">
      <FormControl
```

```
          isRequired
          isInvalid={getIn(errors, 'email') && getIn(touched, 'email')}
        >
          <FormControl.Label>Email</FormControl.Label>
          <Input
            autoCapitalize="none"
            keyboardType="email-address"
            onBlur={handleBlur('email')}
            onChange={handleChange('email')}
            value={values.email}
          />
        </FormControl>

        <Button onPress={handleSubmit} isLoading={isLoading}>
          Send email
        </Button>
      </VStack>
    )
  }

  EmailForm.propTypes = {
    isLoading: PropTypes.bool.isRequired,
    onSubmit: PropTypes.func.isRequired,
  }
```

- **Hooks**

```
import { useState } from 'react'
import { sendPasswordReset } from '../../../api/user'

export const usePasswordReset = () => {
  const [state, setState] = useState({
    isLoading: false,
    error: null,
  })

  const handlePasswordReset = async (values) => {
    setState({ isLoading: true, error: null })

    try {
      await sendPasswordReset(values)
      setState({ isLoading: false, error: null })
    } catch (error) {
      setState({ isLoading: false, error })
      throw error
    }
  }

  return [handlePasswordReset, { ...state }]
}
```

- **Screens**

```
import React from 'react'
import PropTypes from 'prop-types'
import { Center, VStack, Text } from 'native-base'
import { usePasswordReset } from '../hooks/use-password-reset'
import { ErrorMessage } from '../../../components/ErrorMessage'
import { EmailForm } from '../components/EmailForm'

export const ForgotPasswordScreen = ({ navigation }) => {
  const [sendPasswordReset, { isLoading, error }] = usePasswordReset()

  const handlePasswordReset = async (values) => {
    try {
      await sendPasswordReset(values)
      navigation.navigate('SignIn')
    } catch (err) {
      console.error(err.message)
    }
  }

  return (
    <Center flex={1}>
      <VStack space={4} alignItems="center" w="90%">
        <Text>
          Please, enter your email address. You will receive link to create a
          new password via email.
        </Text>
        <ErrorMessage error={error} />
        <EmailForm onSubmit={handlePasswordReset} isLoading={isLoading} />
      </VStack>
    </Center>
  )
}

ForgotPasswordScreen.propTypes = {
  navigation: PropTypes.object.isRequired,
}
```

5.14 homescreen

```
import React from 'react'
import PropTypes from 'prop-types'
import axios from 'axios';
import { VStack, Center, Heading, Button, Divider , Thumbnail, List, ListItem } from 'native-base'
import {
  Text,
  View,
  SafeAreaView,Dimensions,ScrollView,Image,TouchableOpacity } from 'react-native';
import Checkbox from 'expo-checkbox';
import { useUserContext } from '../../../context/UserContext'
import { useSignOut } from '../../../hooks/use-sign-out'
import Carousel from 'react-native-snap-carousel';
import AsyncStorage from '@react-native-async-storage/async-storage';
import { AntDesign } from '@expo/vector-icons';
import { useIsFocused } from '@react-navigation/native';

// ...
import {
  Collapse,
  CollapseHeader,
  CollapseBody
} from 'accordion-collapse-react-native';
export const HomeScreen = ({ navigation }) => {
  const isFocused = useIsFocused();
  const { width: viewportWidth, height: viewportHeight } = Dimensions.get('window');
const SLIDE_WIDTH = Math.round(viewportWidth / 2.6);
const ITEM_HORIZONTAL_MARGIN = 15;
const ITEM_WIDTH = SLIDE_WIDTH + ITEM_HORIZONTAL_MARGIN * 2;
const SLIDER_WIDTH = viewportWidth;
  const { user } = useUserContext()
  const [signOut, { isLoading }] = useSignOut()
  const [cartcount,setcartcount] = React.useState(0)
  const [cart,setcart] = React.useState({})
  const [itembycategory,setitembycategory] = React.useState({})
  const [carouseload , setcarouseload] = React.useState(true)
  const [searchbar , setsearchbar] = React.useState(false)

  const [searchfilter , setsearchfilter] = React.useState([])


  const [state,setstate] = React.useState({
          activeIndex:0,
          carouselItems: [
          {
              title:"Item 1",
              text: "Text 1",
          },
          {
              title:"Item 2",
              text: "Text 2",
          },
          {
              title:"Item 3",
              text: "Text 3",
          },
          {
              title:"Item 4",
              text: "Text 4",
          },
          {
              title:"Item 5",
              text: "Text 5",
          },
        ]
      })

  let carousel = React.useRef();

  React.useEffect(()=>{
    populateCart()
    async function populateCart()
    {
      let cart={}
       try {
              // await AsyncStorage.removeItem("mycart")
              let mycart = await AsyncStorage.getItem("mycart")
              console.log("mycart",mycart)
              if(mycart)
              {
                cart=JSON.parse(mycart)
              }
              setcart(cart)
              setcartcount(Object.keys(cart).length)
            }
            catch(err)
            {
              console.log(err)
            }
      }


    axios.get("https://madbookhub.herokuapp.com/api/fetch/book/").then(resp=>{
      if(resp.data.success == 1)
      {
        setstate({...state,carouselItems:resp.data.books})

        let tempst = itembycategory
        tempst['action']=resp.data.books.filter((item)=>(item.keywords.includes("action")              ||
item.keywords.includes("fight")))
        tempst['romance']=resp.data.books.filter((item)=>(item.keywords.includes("romance")       ||
item.keywords.includes("romantic") || item.keywords.includes("love")))

        tempst['adventure']=resp.data.books.filter((item)=>item.keywords.includes("adventure"))


        //                                                        setitembycategory({...itembycategory
,action:resp.data.books.filter((item)=>item.keywords.includes("action"))})
        //                                                        setitembycategory({...itembycategory
,romance:resp.data.books.filter((item)=>item.keywords.includes("romance"))})
        //                                                        setitembycategory({...itembycategory
,adventure:resp.data.books.filter((item)=>item.keywords.includes("adventure"))})
       setitembycategory({...tempst})
        console.log("===========carousel loading is ",carouseload)

        setcarouseload(false)

        console.log("carousel loading is ",carouseload)
        //
console.log("++++++++++++",resp.data.books.filter((item)=>item.keywords.includes("romance
")))

      }

    }).catch(err=>{
      console.log("error occurred",err)
    })

  },[carouseload,isFocused])

  const handlePressOnUpdatePassword = () => {
    navigation.navigate('Addbook')      //Addbook
  }

  const goToCart=()=>{
 navigation.navigate('AddToCart',{cart})
  }
  const addToCart=async (id)=>{
    console.log("id is ",id)
    let cart = {}

            try {
```

```
            // await AsyncStorage.removeItem("mycart")
            let mycart = await AsyncStorage.getItem("mycart")
            console.log("mycart",mycart)
            if(mycart)
            {
              cart=JSON.parse(mycart)
            }

            cart[id]=1

            console.log("cart is ",cart)
            setcart(cart)

            setcartcount(Object.keys(cart).length)
          await AsyncStorage.setItem("mycart",JSON.stringify(cart))
  } catch (error) {
    // Error saving data
    console.log("Error occured during remoing data")
  }

  }
 const renderItem = ({item, index}) => {
       return (
          <View style={{
            // backgroundColor:'green',
            borderColor:"#398AB9",
            borderWidth:2,
            borderRadius: 5,
            // height: 250,
            padding: 10,
            marginLeft: 10,
            marginRight:
10,textAlign:"center",display:"flex",justifyContent:"center",alignItems:"center" }}>
          <Image source={{
         uri: item.image,
       }}
       resizeMode={'contain'}
       style={{width:'100%',height:120}}
     />
          <Text style={{fontSize: 23}} numberOfLines={1}>{item.title}</Text>
          <Text numberOfLines={2} style={{textAlign:'center'}}>{item.description}</Text>
          <Text style={{fontWeight:'bold',marginTop:10}}>Rs(₹): {item.price}</Text>
             <Button bg="#fff" onPress={()=>addToCart(item._id)}
style={{marginTop:10,borderColor:"#1c658c",borderWidth:2,padding:0,backgroundColor:"#1c
658c"}} px="4" py="1" >{cart && cart[item._id] ? "Added ✅" : "Add to cart" }</Button>

        </View>
```

```
      );
    }


  return (
    <>

    <ScrollView>
    <View style={{flexDirection:"row",justifyContent:"space-between",paddingLeft:10,paddingRight:10,marginTop:10,alignItems:"center"}}>
          <Button bg="#1c658c" onPress={handlePressOnUpdatePassword}>Add Books</Button>
          <Text style={{color:"#1c658c",fontWeight:"bold",fontSize:20}} onPress={goToCart}><AntDesign name="shoppingcart" size={28} color="#1c658c" />{cartcount}</Text>
        <Button bg="#1c658c" onPress={signOut} isLoading={isLoading}>
          Sign out
        </Button>
          </View>


  <View style={{paddingTop:10}}>
 <Collapse
         isExpanded={searchbar}
         onToggle={(isExpanded)=>setsearchbar(isExpanded)}>
         <CollapseHeader>
           <Text style={{backgroundColor:"#1c658c",fontSize:15,textAlign:"center",fontWeight:"bold",color:"#fff",textDecorationLine: "underline",paddingBottom:3}}>Search By Category</Text>
         </CollapseHeader>
         <CollapseBody>
         <View style={{display:"flex",justifyContent:"space-around",flexDirection:"row",paddingTop:20}}>
             <View style={{display:"flex",alignItems:"center",justifyContent:"center"}}>
               <Text >Action</Text>
              <Checkbox
             value={searchfilter.includes("action")}
             onValueChange={()=>{searchfilter.includes("action") ? setsearchfilter(searchfilter.filter(e=>e!="action")) : setsearchfilter([...searchfilter, "action"])}}
             tintColors={{ true: '#1c658c'}}

            />
            </View>
                  <View style={{display:"flex",alignItems:"center",justifyContent:"center"}}>
               <Text>Romantic</Text>
              <Checkbox
             value={searchfilter.includes("romance")}
```

```
                onValueChange={()=>{searchfilter.includes("romance")                                    ?
setsearchfilter(searchfilter.filter(e=>e!="romance"))          :          setsearchfilter([...searchfilter,
"romance"])}}
                tintColors={{ true: '#1c658c'}}

              />
              </View>
                      <View style={{display:"flex",alignItems:"center",justifyContent:"center"}}>
                  <Text>Adventure</Text>
                 <Checkbox
                value={searchfilter.includes("adventure")}
                onValueChange={()=>{searchfilter.includes("adventure")                                  ?
setsearchfilter(searchfilter.filter(e=>e!="adventure"))          :          setsearchfilter([...searchfilter,
"adventure"])}}
                tintColors={{ true: '#1c658c'}}

              />
              </View>

          </View>
          </CollapseBody>
        </Collapse>


          {searchfilter.length > 0 ?  <View style={{marginTop:5}}>
       {searchfilter.map(book=>(
        <>

         <Text
style={{fontSize:14,fontWeight:"bold",textAlign:"center",paddingBottom:10}}>Search
Category : {book}</Text>

       <Carousel
              ref={(c) => { carousel = c; }}
              data={itembycategory[book] }
              renderItem={renderItem}
              sliderWidth={SLIDER_WIDTH}
              itemWidth={ITEM_WIDTH+15}
              activeSlideAlignment={'start'}
              inactiveSlideScale={1}
              inactiveSlideOpacity={1}
              loop={true}

            />
            </>

        ))}
```

```
      <View
  style={{
    borderBottomColor: 'black',
    borderBottomWidth: 1,
    marginTop:10
  }}
></View>


    </View>:     (!carouseload  ? <>
          <Text
style={{fontSize:20,paddingRight:10,fontWeight:"bold",color:"#1C658C",textAlign:"center",ma
rginTop:10}}>----- Action -----</Text>

    <View style={{marginTop:5}}>

      <Carousel
            ref={(c) => { carousel = c; }}
            data={itembycategory['action'] }
            renderItem={renderItem}
            sliderWidth={SLIDER_WIDTH}
            itemWidth={ITEM_WIDTH+15}
            activeSlideAlignment={'start'}
            inactiveSlideScale={1}
            inactiveSlideOpacity={1}
            loop={true}

          />


    </View>
<Text
style={{fontSize:20,paddingRight:10,fontWeight:"bold",color:"#1C658C",textAlign:"center",ma
rginTop:10}}>----- Romantic ----- </Text>
       <View style={{marginTop:5}}>
      <Carousel
            ref={(c) => { carousel = c; }}
            data={itembycategory['romance'] }
            renderItem={renderItem}
            sliderWidth={SLIDER_WIDTH}
          itemWidth={ITEM_WIDTH+15}
          activeSlideAlignment={'start'}
          inactiveSlideScale={1}
          inactiveSlideOpacity={1}
          loop={true}
          />
    </View>
<Text
style={{fontSize:20,paddingRight:10,fontWeight:"bold",color:"#1C658C",textAlign:"center",ma
rginTop:10}}>----- Adventure ----- </Text>
        <View style={{marginTop:5}}>
       <Carousel
              ref={(c) => { carousel = c; }}
              data={itembycategory['adventure']}
              renderItem={renderItem}
        sliderWidth={SLIDER_WIDTH}
  itemWidth={ITEM_WIDTH+15}
  activeSlideAlignment={'start'}
  inactiveSlideScale={1}
  inactiveSlideOpacity={1}
  loop={true}
            />


    </View>
</>:<Text>"loading..."</Text>) }
  </View>

    </ScrollView>

    </>

  )
}

HomeScreen.propTypes = {
  navigation: PropTypes.object.isRequired,
}
```

5.15 update-password

- **Components**

```
import React from 'react'
import PropTypes from 'prop-types'
import { VStack, FormControl, Input, Button } from 'native-base'
import { useFormik, getIn } from 'formik'
import * as Yup from 'yup'

const validationSchema = Yup.object({
  password: Yup.string().required().min(6),
  passwordConfirmation: Yup.string()
    .oneOf([Yup.ref('password'), null])
    .required(),
})

export const UpdatePasswordForm = ({ isLoading, onSubmit }) => {
  const { handleChange, handleBlur, handleSubmit, values, touched, errors } =
    useFormik({
      initialValues: {
        password: '',
        passwordConfirmation: '',
      },
      validationSchema,
      onSubmit,
    })

  return (
    <VStack space={4} alignItems="center" w="100%">
      <FormControl
        isRequired
        isInvalid={getIn(errors, 'password') && getIn(touched, 'password')}
      >
        <FormControl.Label>New password</FormControl.Label>
        <Input
          autoCapitalize="none"
          secureTextEntry
          autoCorrect={false}
          autoCompleteType="password"
          onBlur={handleBlur('password')}
          onChange={handleChange('password')}
          value={values.password}
        />
      </FormControl>

      <FormControl
        isRequired
        isInvalid={
          getIn(errors, 'passwordConfirmation') &&
          getIn(touched, 'passwordConfirmation')
        }
      >
        <FormControl.Label>Confirm new password</FormControl.Label>
        <Input
          autoCapitalize="none"
          secureTextEntry
          autoCorrect={false}
          autoCompleteType="password"
          onBlur={handleBlur('passwordConfirmation')}
          onChange={handleChange('passwordConfirmation')}
    value={values.passwordConfirmation}
   />
  </FormControl>

  <Button onPress={handleSubmit} isLoading={isLoading}>
   Update password
  </Button>
 </VStack>
 )
}

UpdatePasswordForm.propTypes = {
 isLoading: PropTypes.bool.isRequired,
 onSubmit: PropTypes.func.isRequired,
}
```

- **Hooks**
    1. **User-reauthentication**

```
import { useState } from 'react'
import { reauthenticate } from '../../../api/user'

export const useReauthenticate = () => {
 const [state, setState] = useState({
  isLoading: false,
  error: null,
 })

 const handleReauthenticate = async (values) => {
  setState({ isLoading: true, error: null })

  try {
   await reauthenticate(values)
   setState({ isLoading: false, error: null })
  } catch (error) {
   setState({ isLoading: false, error })
   throw error
  }
 }

 return [handleReauthenticate, { ...state }]
}
```

    2. **User-update-password**

```
import { useState } from 'react'
import { updatePassword } from '../../../api/user'

export const useUpdatePassword = () => {
  const [state, setState] = useState({
    isLoading: false,
    error: null,
  })

  const handleUpdatePassword = async (values) => {
    setState({ isLoading: true, error: null })

    try {
      await updatePassword(values)
      setState({ isLoading: false, error: null })
    } catch (error) {
      setState({ isLoading: false, error })
      throw error
    }
  }

  return [handleUpdatePassword, { ...state }]
}
```

- **Screens**

1. **Reauthentication**

```
import React from 'react'
import PropTypes from 'prop-types'
import { VStack, Center } from 'native-base'
import { useReauthenticate } from '../hooks/use-reauthenticate'
import { ErrorMessage } from '../../../components/ErrorMessage'
import { EmailAndPasswordForm } from
'../../../components/EmailAndPasswordForm'

export const ReauthenticateScreen = ({ navigation }) => {
  const [reauthenticate, { isLoading, error }] = useReauthenticate()

  const handleReauthenticate = async (values) => {
    try {
      await reauthenticate(values)
      navigation.navigate('UpdatePassword')
    } catch (err) {
      console.error(err.message)
    }
  }

  return (
    <Center flex={1}>
      <VStack space={4} alignItems="center" w="90%">
        <ErrorMessage error={error} />
        <EmailAndPasswordForm
          onSubmit={handleReauthenticate}
          isLoading={isLoading}
          buttonText="Re-authenticate"
        />
      </VStack>
    </Center>
  )
}

ReauthenticateScreen.propTypes = {
  navigation: PropTypes.object.isRequired,
}
```

2. **Update-password**

```
import React from 'react'
import PropTypes from 'prop-types'
import { VStack, Center } from 'native-base'
import { useUpdatePassword } from '../hooks/use-update-password'
import { ErrorMessage } from '../../../components/ErrorMessage'
import { UpdatePasswordForm } from '../components/UpdatePasswordForm'

export const UpdatePasswordScreen = ({ navigation }) => {
  const [updatePassword, { isLoading, error }] = useUpdatePassword()

  const handleUpdatePassword = async (values) => {
    try {
      await updatePassword(values)
      navigation.popToTop()
    } catch (err) {
      console.error(err.message)
    }
  }

  return (
    <Center flex={1}>
      <VStack space={4} alignItems="center" w="90%">
        <ErrorMessage error={error} />
        <UpdatePasswordForm
```

```
          onSubmit={handleUpdatePassword}
          isLoading={isLoading}
        />
      </VStack>
    </Center>
  )
}

UpdatePasswordScreen.propTypes = {
  navigation: PropTypes.object.isRequired,
}
```

5.16 App.js

```
import React from 'react'

import { Center, Spinner } from 'native-base'
import { ErrorMessage } from './components/ErrorMessage'
import { useUserContext } from './context/UserContext'
import { GuestAppNavigator } from './navigation/GuestAppNavigator'
import { UnverifiedAppNavigator } from './navigation/UnverifiedAppNavigator'
import { VerifiedAppNavigator } from './navigation/VerifiedAppNavigator'

export const App = () => {
  const { user, isLoading, error } = useUserContext()

  if (error)
    return (
      <Center flex={1}>
        <ErrorMessage error={error} />
      </Center>
    )

  if (isLoading)
    return (
      <Center flex={1} accessibilityLabel="Loading user profile...">
        <Spinner />
      </Center>
    )

  if (user && user.emailVerified) return <VerifiedAppNavigator />

  if (user) return <UnverifiedAppNavigator />

  return <GuestAppNavigator />
}
```

5.17 Root.js

```
import './api'
import { registerRootComponent } from 'expo'
import React from 'react'
import { StatusBar } from 'expo-status-bar'
import { NavigationContainer } from '@react-navigation/native'
import { NativeBaseProvider } from 'native-base'
import { App } from './App'
import { UserContextProvider } from './context/UserContext'

const Root = () => (
  <NavigationContainer>
    <NativeBaseProvider>
      <UserContextProvider>
        <App />
        <StatusBar style="auto" />
      </UserContextProvider>
    </NativeBaseProvider>
  </NavigationContainer>
)

registerRootComponent(Root)
```

5.18 User.js

```
import firebase from 'firebase'

export const getUser = () => firebase.auth().currentUser

export const onAuthStateChanged = (args) =>
  firebase.auth().onAuthStateChanged(args)

// Notice Firebase automatically signs user in when their account is created
export const signUp = async ({ email = '', password = '' }) => {
  await firebase.auth().createUserWithEmailAndPassword(email, password)
  sendVerification()
}

export const signIn = ({ email = '', password = '' }) =>
  firebase.auth().signInWithEmailAndPassword(email, password)

export const sendVerification = () => getUser().sendEmailVerification()
export const signOut = () => firebase.auth().signOut()

export const reload = () => getUser().reload()

export const reauthenticate = ({ email = '', password = '' }) =>
  getUser().reauthenticateWithCredential(
    firebase.auth.EmailAuthProvider.credential(email, password)
  )

export const updatePassword = ({ password = '' }) =>
  getUser().updatePassword(password)

export const sendPasswordReset = ({ email = '' }) =>
  firebase.auth().sendPasswordResetEmail(email)
```

6.TESTING

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of Validation and Verification.

- **Unit Testing**

While coding, the programmer performs some tests on that unit of program to know if it is error free. Testing is performed under white-box testing approach. Unit testing helps developers decide that individual units of the program are working as per requirement and are error free.

- **Integration Testing**

Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passes and data updating etc.

- **System Testing**

The software is compiled as product and then it is tested. This can be accomplished using one or more of the following tests:

- **Functionality testing**

Tests all functionalities of the software against the requirement.

- **Performance testing**

This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environment conditions.

- **Acceptance Testing**

When the software is ready to hand over to the customer it must go through last phase of testing where it is tested for user-interaction and response. This is important because even if the software matches all user requirements and if user does not like the way it appears or works, it may be rejected.

- **Alpha testing**

The team of developer themselves perform alpha testing by using the system as if it is being used in work environment. They try to find out how user would react to some action in software and how the system should respond to inputs.

- **Beta testing**

After the software is tested internally, it is handed over to the users to use it under their production environment only for testing purpose. This is not yet the delivered product. Developers expect that users at this stage will bring minute problems, which were skipped to attend.

Test case: -

Feature Name	Story ID	Test Case ID	Summary	Precondition	Execution Steps	Expected Result	Actual Result
Uninstall	Mob-1214	Mob-1214_2	Verify that application should be Uninstalled successfully.	1. Execute test case ID: Mob-1214_1	1. Click on settings. 2. Select the newly added application on Mob-1214_1. 3. Click on Uninstall button. 4. Verify.	The application should be Uninstalled successfully.	uninstalled successfully
Interruption by Calls	Mob-1214	Mob-1214_3	Verify that user should able to accept Phone calls when application is running and should continue from the same point.	1. Execute test case ID: Mob-1214_1	1. Open the application. 2. Navigate here an there for a moment. 3. Make a call from another device to the device where you have opened the application. 4. Pick up the call. 5. Now disconnect it and verify.	User should able to accept Phone calls when application is running and should continue from the same point.	user able to accept phone call
Interruption by Messages	Mob-1214	Mob-1214_4	Verify that user should able to accept messages when application is running and should continue from the same point after reading the message.	1. Execute test case ID: Mob-1214_1	1. Open the application. 2. Navigate here an there for a moment. 3. Send a message from another device to the device where you have opened the application. 4. Read the message. 5. Close the message app and verify.	User should able to accept messages when application is running and should continue from the same point after reading the message.	user able to accept messages
Exit application	Mob-1214	Mob-1214_6	Verify that user should able to exit from application if we click on end key.	1. Execute test case ID: Mob-1214_1	1. Click on app and open it. 2. Now press the end key and verify.	User should able to exit from application if we click on end key.	user able to exit frim application
Battery	Mob-1214	Mob-1214_7	Verify that user should able to see the alert when battery is low.	1. Execute test case ID: Mob-1214_1	1. Click on app and open it. 2. Use the application till you get the low battery indication.	When battery is low the alert should display.	battery low alert
Battery Consumption	Mob-1214	Mob-1214_8	Verify that application should not consume more battery	1. Execute test case ID: Mob-1214_1 2. Full charge your device.	1. Click on app and open it. 2. Use the application and verify the status of the battery in 15 mins interval of time.	Application should not consume more battery	application not consuming more battery
Charge	Mob-1214	Mob-1214_8	Verify that application should run when inserting the charger. It will not affect the application	1. Execute test case ID: Mob-1214_1	1. Click on app and open it. 2. Insert the charging pin in between running of the application and verify.	Application should run when inserting the charger. It will not affect the application	application is running properly

LIST OF ABBREVIATIONS

Abbreviations	**Name of Figure**
GUI	Graphical User Interface
UC	Use Case
Approx	Approximately
iOS	iPhone Operating System