Step Up Code (Learning App)
BY Rohit Gupta

Contents

1.Abstract of The Project

Step up code (Name of the app under online Learning system project) is very useful app for the Students and teacher where they can learn or teach various type of technology's and subject. In current Covid-19 pandemic situation student and teachers are facing problems. By using this platform teachers can teach there student in online by using video class live classes.

Student can view the courses and purchase them using different payment method and learn from home. Student can watch video, download content, give exam .

2. Introduction

Objective

Step up Code (Name of the app on my project online Learning system) is a home based tutor system where student can learn from home from tutors through video classes.

Objective of the learning system is:-

It is a mobile-based app that manages the student learning.

Objectives of this app:-

1. To authenticate student with the help of registration process with basic information.
2. To build a monitoring system that is able to monitor and manage all service operations efficiently.
3. To give an opportunity to learn from home in this covid-19 pandemic situation.
4. This system basically has five types of modules :

    a) Manage Account module
    b) Search course module
    c) Purchase course
    d) Payment Generator Module
    e) Learning through video

5. By using Step up code the process of online learning become easy for student. This App provides a user-friendly application where Student first login then can view various Courses, they can choose the Course then purchase the course by using online payment system. After that the can learn from the video lectures.

3. System Analysis

3.1: IDENTIFICATION OF NEED

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studies to the minutest detail and analysed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The System is viewed as a whole and the input to the system are identified. The outputs from the organization are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and Decisional variables, analysis and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action. A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be 9scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem area are identified. The designer now function as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user .The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

3.2: FEASIBILITY STUDY

The objective behind the feasibility study is to create the reasons for developing the software that is acceptable to users, flexible to change and conformable to established standards.

Types of Feasibility:

Technical Feasibility - Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.

Operational Feasibility - Operational feasibility assesses the range in which the required software performs a series of levels to solve business problems and customer requirements.

Economic Feasibility - Economic feasibility decides whether the necessary software can generate financial profits for an organization.

Feasibility study is made to see if the project on completion will serve the purpose the organization for the amount of work.

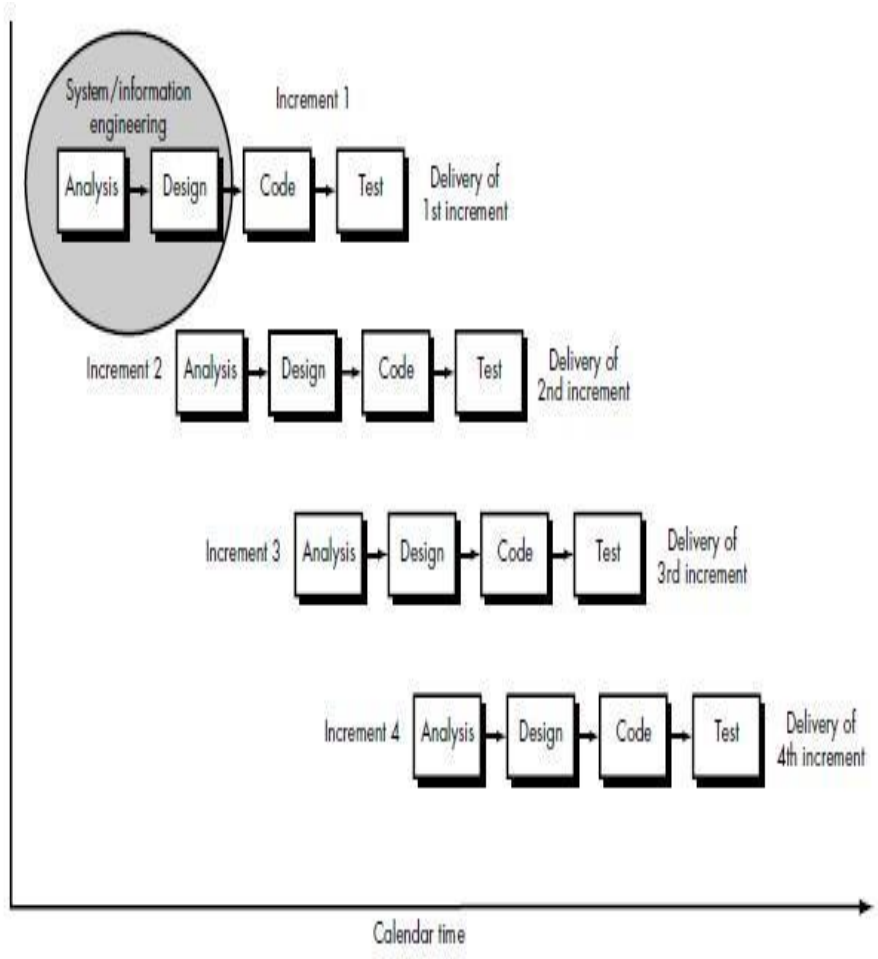
Effort and the time that spend on it.

The document provide the feasibility of the project that is being designed and lists various area that were considered very carefully during the feasibility study of this project such as Technical, Economical and operational feasibilities.

3.3: WORK FLOW

This Document plays a vital role in the development life cycle (SDLC) as it describes the complete requirement of the system. It is meant for use by the developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

INCREMENTAL MODEL was being chosen •



The developer is responsible for:-

• Developing the system, which meets the SRS and solving all the requirements of the system.

• Demonstrating the system and installing the system at client's location after the acceptance testing is successful.

• Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.

• Conducting any user training that might be needed for using the system.

• Maintaining the system for a period of one

3.4: FUNCTIONAL REQUIREMENTS

Modules:

The modules used in this software are as follows:

- ✓ **LOG IN:** Student can login into their account with unique password for viewing all available Courses, syllabus and also can manage their account.
- ✓ **SIGN UP:** New Students can sign up into the application by creating a new account with a new unique password.
- ✓ **View Course:** Student can view all the available course and watch free videos to learn new things.
- ✓ **Payment:** By using Payment module student can buy course.
- ✓ **Learning:** In this module after enrolling the course student can access the course content and watch videos.

3.5: NON-FUNCTIONAL REQUIREMENTS:-

- ✓ **Usability Requirement**: The system shall allow the users to access the system from any android or ios smartphone, no special training is required. The system user friendly and the system is written in simple English.
- ✓ **Availability Requirement**: The system is available 100% for the user and is used by 24 hours a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.
- ✓ **Accuracy**: The system should accurately provide real time information taking into consideration various issues. The system shall provide 100% access reliability.
- ✓ **Performance Requirement**: The information is refreshed at regular intervals depending upon whether some updates have occurred or not. The system shall respond the member in less than 2 seconds.
- ✓ **Security Requirement**: System will use a secured database and the system will have different users and each user has different types of constraints. Only admins have the rights to update database information of other users.
- ✓ **Reliability Requirement**: The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect data. The system will run 7 days a week and 24 hours a day

3.6: Software & Hardware Requirements

CLIENT REQUIREMENTS:-

- Mobile device:- Android(minimum version 4.0) or ios(minimum version 6)
- 2 GB min. RAM.

DEVELOPER REQUIREMENTS:-

Hardware:

- Laptop / Desktop(with 8gb ram and i3 processor)
- Mobile device:- Android(minimum version 4.0) or ios(minimum version 6)

Software:

- Android studio/Xcode
- Visual Studio code 2019 Edition.
- React Native 0.62
- My SQL/ Firebase

4.System Design

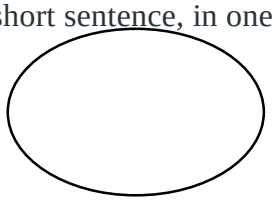
4.1: Dataflow Diagram:-

DFD is the abbreviation for **Data Flow Diagram**. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

Symbols of DFD:-

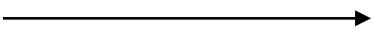
Process

Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence



Data Flow

Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved.



Warehouse

The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet.

Terminator

The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations like banks, groups of people like customers or different departments of the same organization, which is not a part of the model system and is an external entity. Modeled systems also communicate with terminator.

Steps to Construct Data Flow Diagram

Four Steps are generally used to construct a DFD:-

 Process should be named and referred for easy reference. Each name should be representative of the reference.

 The destination of flow is from top to bottom and from left to right.

 When a process is distributed into lower level details they are numbered.

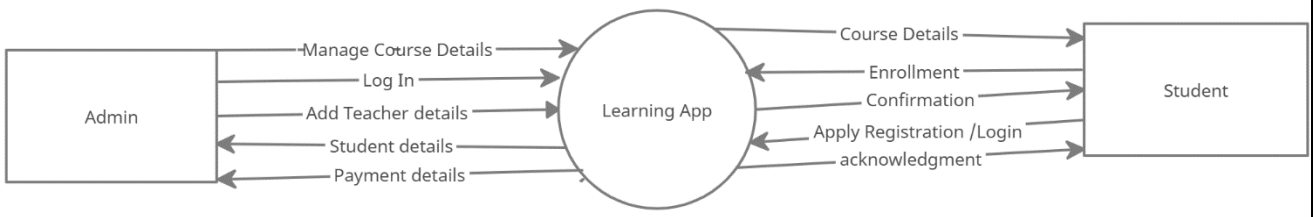
 The names of data stores, sources and destinations are written in capital letters.

Rules for constructing a Data Flow Diagram:-

 Arrows should not cross each other.

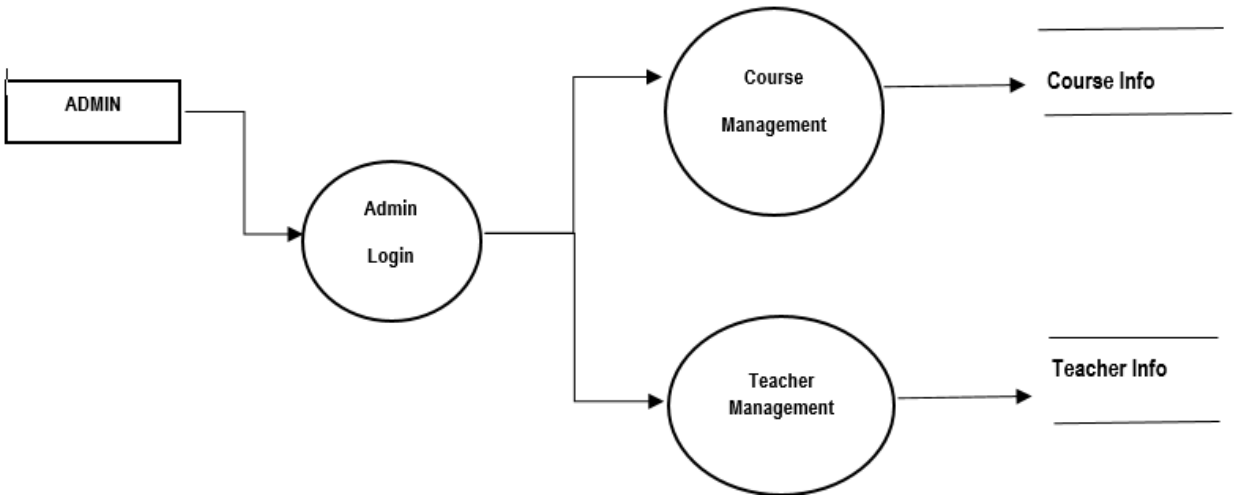
- Squares, Circles, Files must bear a name.
- Decomposed data flow squares and circles can have same names.
- Draw all data flow around the outside of the diagram

DFD Zero Level For Learning APP

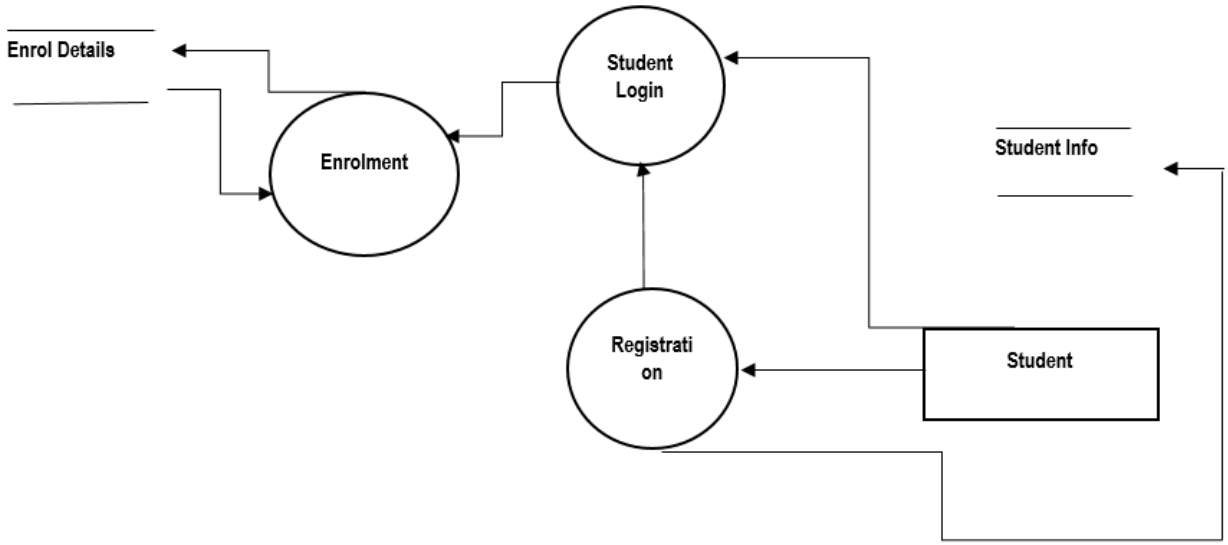


DFD Level 1 for Learning App

For Admin side:



For Student side:



4.2: ENTITY RELATIONSHIP DIAGRAM:-

In software engineering, an **entity–relationship model** (**ER model**) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational database. The main components of ER models are entities (things) and the relationships that can exist among them. However, variants of the idea existed previously, and have been devised subsequently such as super type and subtype data entities and commonality relationships.

An entity–relationship model is a systematic way of describing and defining a business process. The process is modelled as components (*entities*) that are linked with each other by *relationships* that express the dependencies and requirements between them, such as: *one building may be divided into zero or more apartments, but one apartment can only be located in one building.* Entities may have various properties (*attributes*) that characterize them.

Diagrams created to represent these entities, attributes, and relationships graphically are called entity–relationship diagrams.

An ER model is typically implemented as a database. In the case of a relational database, which stores data in tables, every row of each table represents one instance of an entity. Some data fields in these tables point to indexes in other tables; such pointers represent the relationships.

The three schema approach to software engineering uses three levels of ER models that may be developed.

An entity may be defined as a thing capable of an independent existence that can be uniquely identified. An entity is an abstraction from the complexities of a domain. When we speak of an entity, we normally speak of some aspect of the real world that can be distinguished from other aspects of the real world.

A relationship captures how entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns.

Cardinality constraints are expressed as follows:

- A double line indicates a *participation constraint*, totality or subjectivity : all entities in the entity set must participate in *at least one* relationship in the relationship set;

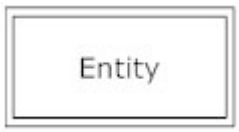
• An arrow from entity set to relationship set indicates a key constraint, i.e. injectivity: each entity of the entity set can participate in *at most one* relationship in the relationship set;

• A thick line indicates both, i.e. bijectivity: each entity in the entity set is involved in *exactly one* relationship.

• An underlined name of an attribute indicates that it is a key: two different entities or relationships with this attribute always have different values for this attribute.

Symbols used in ERD:-

Entities, which are represented by rectangles. An entity is an object or concept about which you want to store information.



A weak entity is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.

Actions, which are represented by diamond shapes, show how two entities share information in the database.

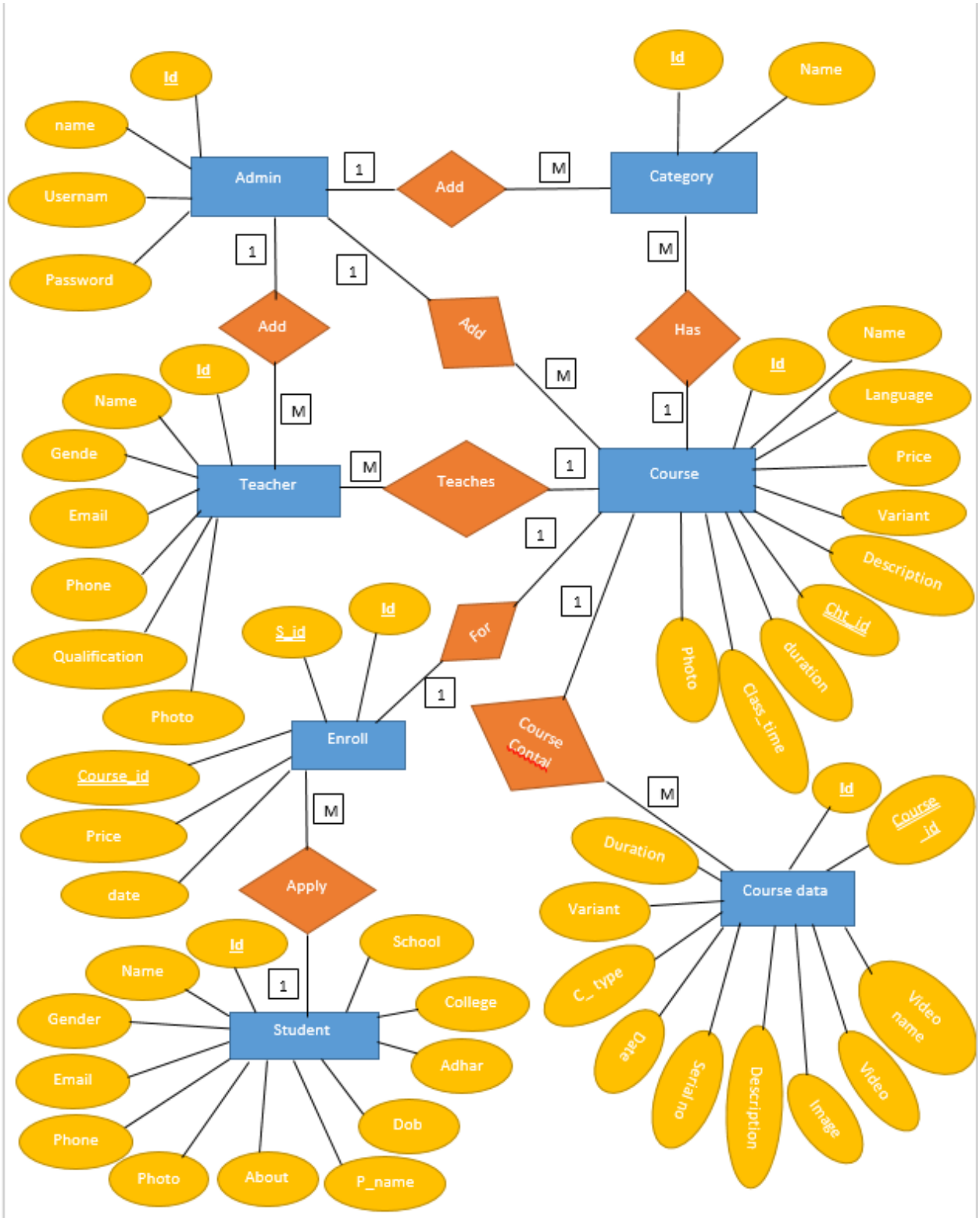
Attributes, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.

A multivalued attribute can have more than one value

A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.

Connecting lines, solid lines that connect attributes to show the relationships of entities in the diagram.

Cardinality specifies how many instances of an entity relate to one instance of another entity. Ordinality is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinality describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinality specifies the absolute minimum number of relationships.


Id
name
Usernam
Password
Admin
1
Add
M
Id
Name
Category
1
1
M
Add
Add
Has
M
1
Id
Name
Gende
Email
Phone
Qualification
Photo
Teacher
M
Teaches
1
Course
Name
Id
Language
Price
Variant
Description
Cht_id
duration
Class_time
Photo
1
1
For
S_id
Id
1
Enroll
Course_id
Price
date
M
Apply
Course Contai
M
Id
Course _id
Duration
Variant
C_type
Date
Serial no
Description
Image
Video
Video name
Course data
Id
Name
Gender
Email
Phone
Photo
About
P_name
1
Student
School
College
Adhar
Dob

4.3: USE 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.

So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So use case diagrams are consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

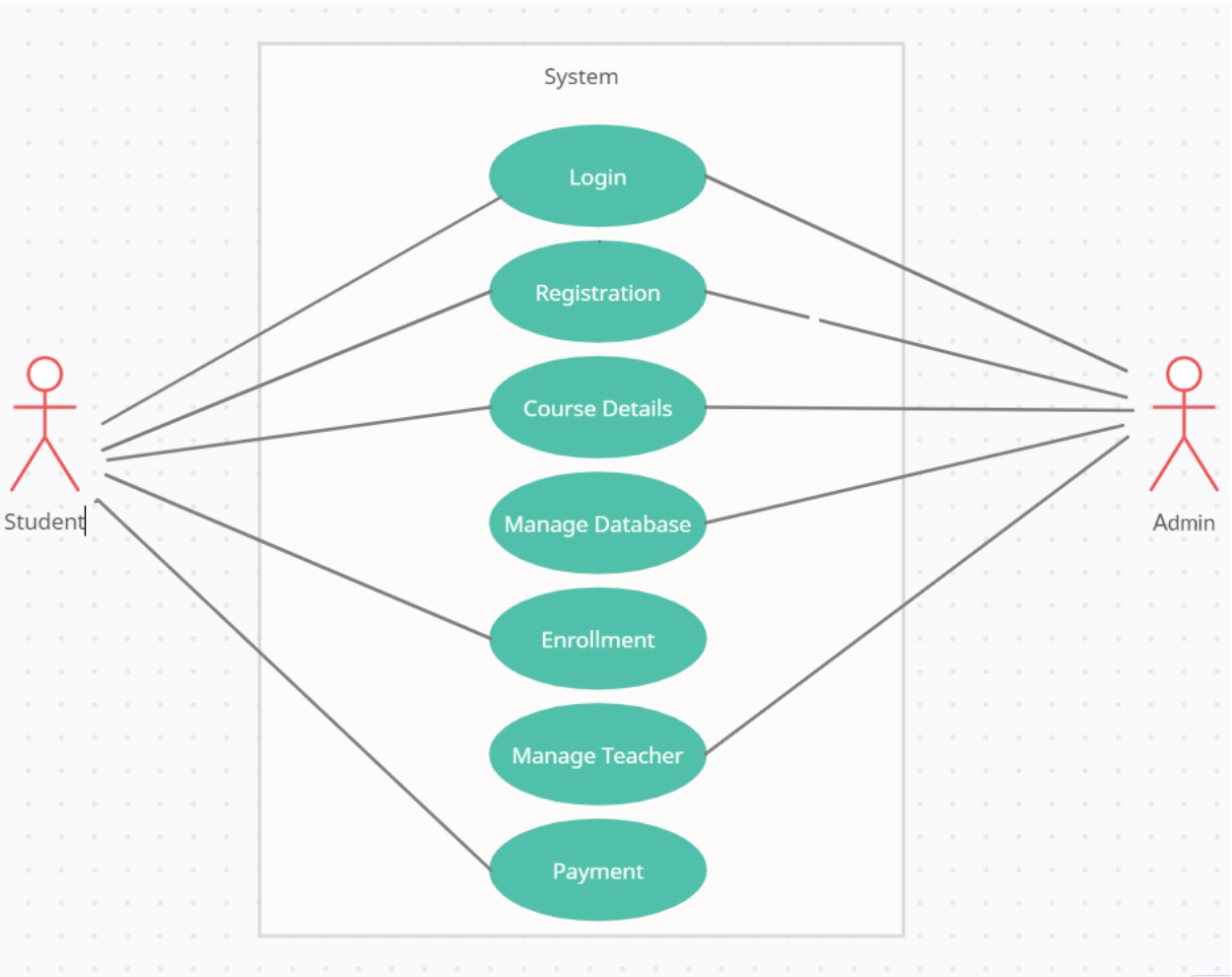
So to model the entire system numbers of use case diagrams are used. The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose. Because other four diagrams (activity, sequence, collaboration and State chart) are also having the same purpose. So we will look into some specific purpose which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analysed to gather its functionalities use cases are prepared and actors are identified.

Now when the initial task is complete use case diagrams are modelled to present the outside view. So in brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors.

Use case diagram for Learning App



4.4: DATABASE DESIGN:-

A database is an organized mechanism that has capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected.

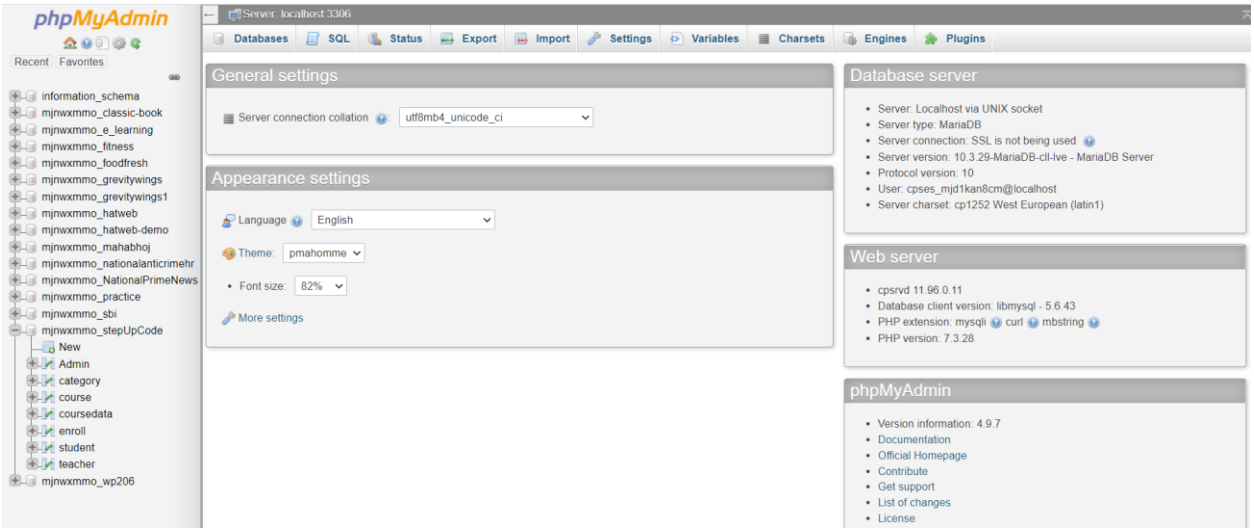
The database design is two level processes. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called information Level design and it is taken independent of any individual DBMS.

Firebase:

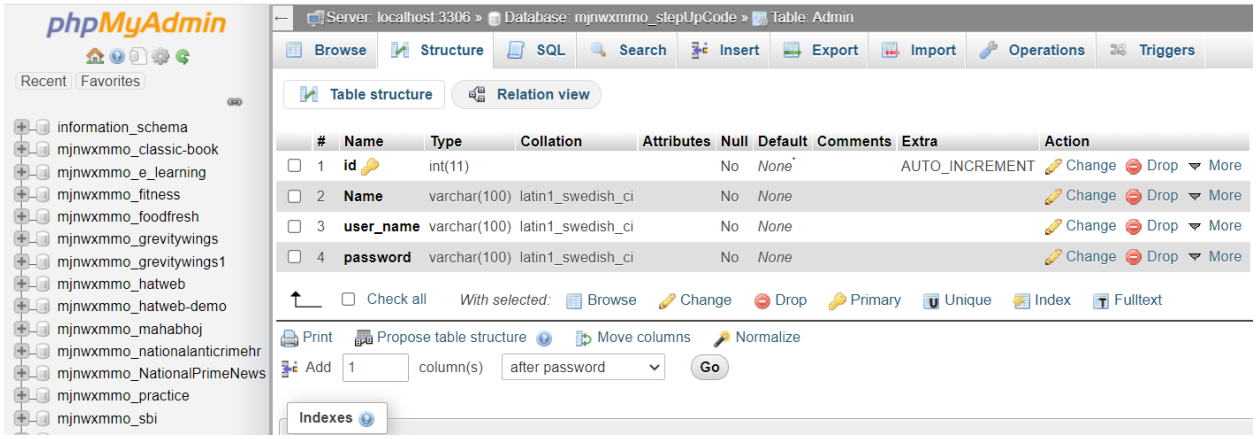
Firebase is a platform developed by Google for creating mobile and web applications. It was originally an independent company founded in 2011. In 2014, Google acquired the platform and it is now their flagship offering for app development.

In the following snapshots we display the way we have used MySQL as the backend RDBMS for our project and the various entities that have been used along with their table definition and table data

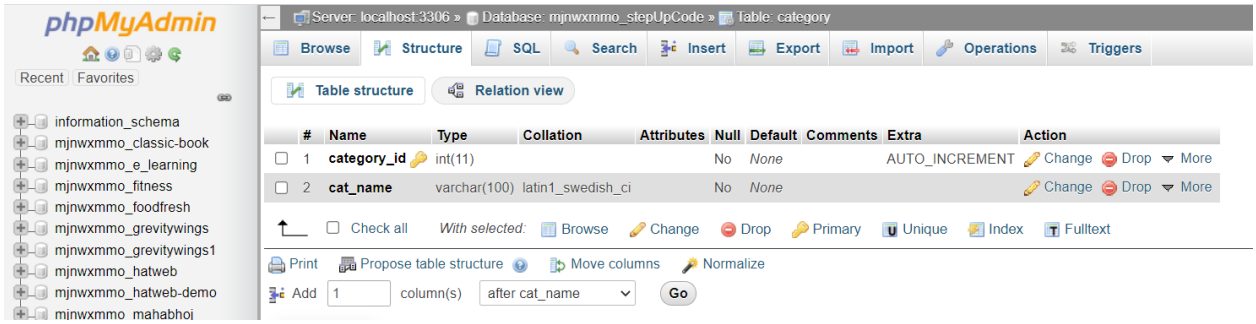
13. Database:



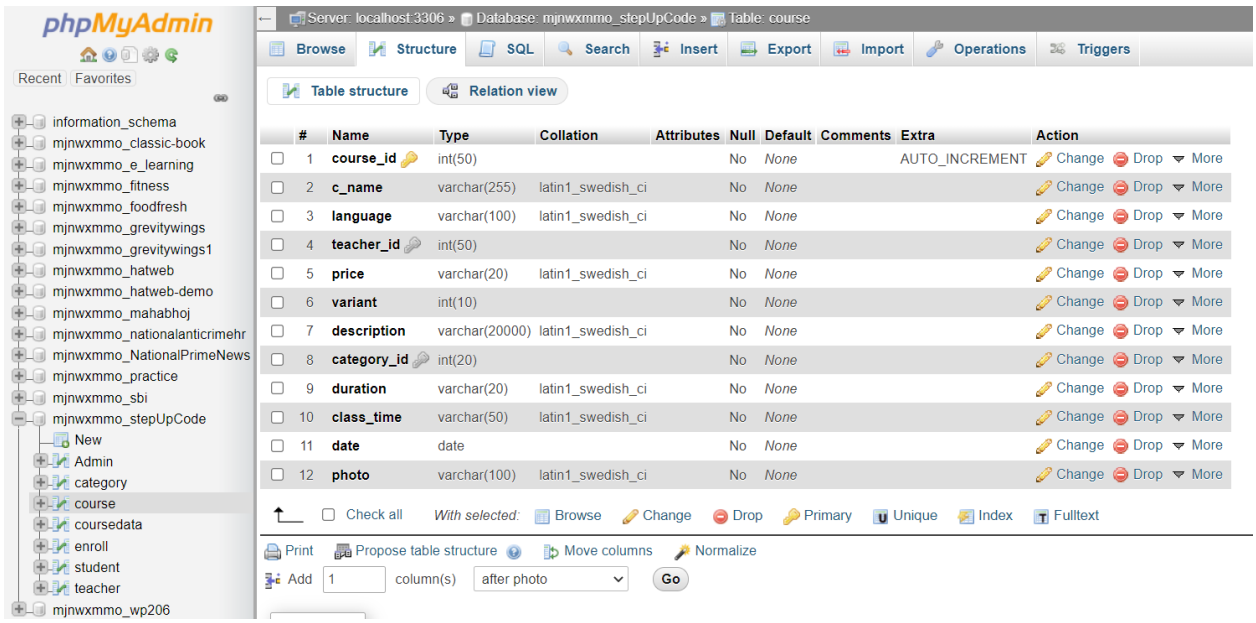
admin Table:



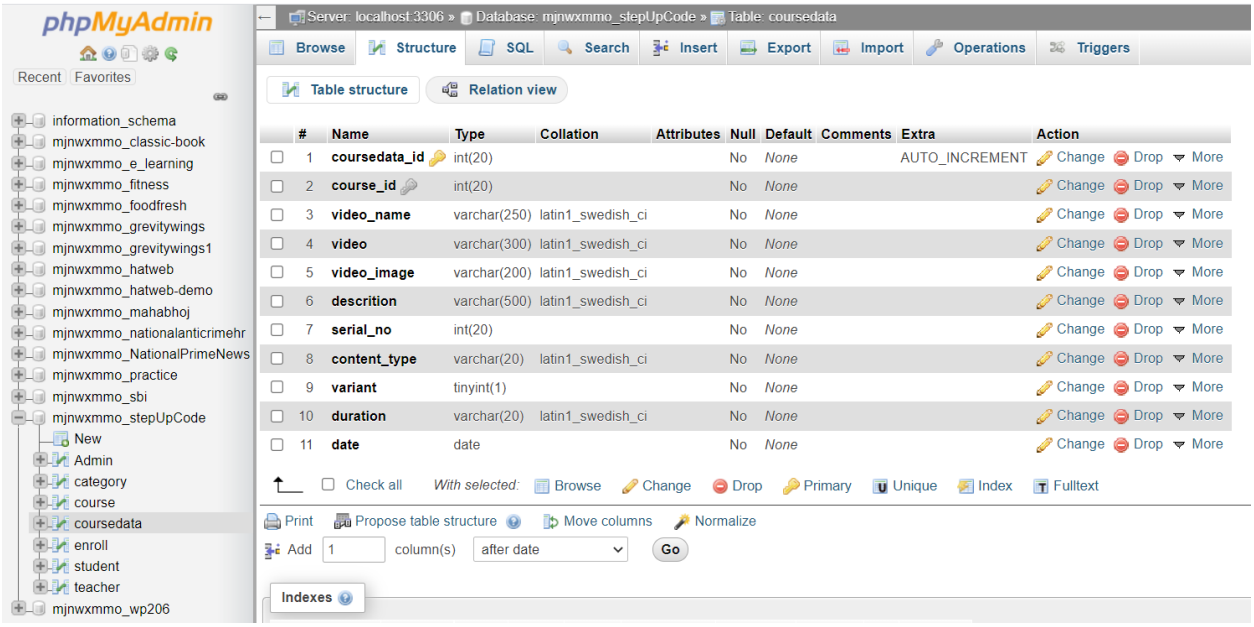
category Table:



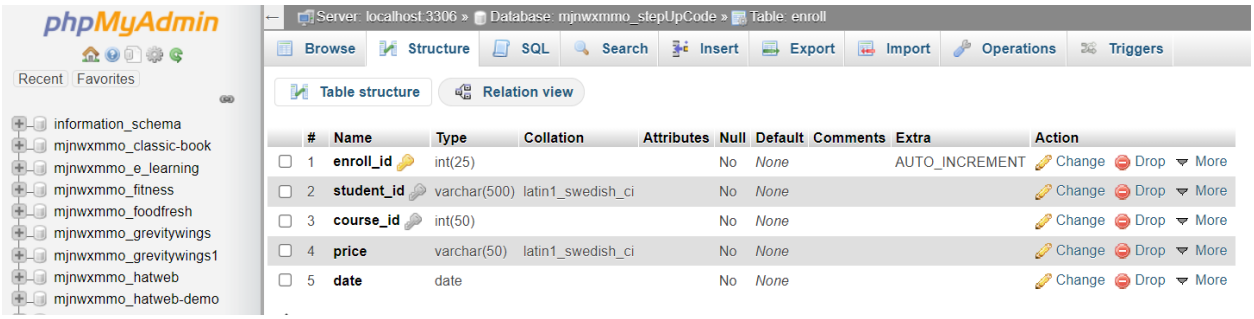
course Table:



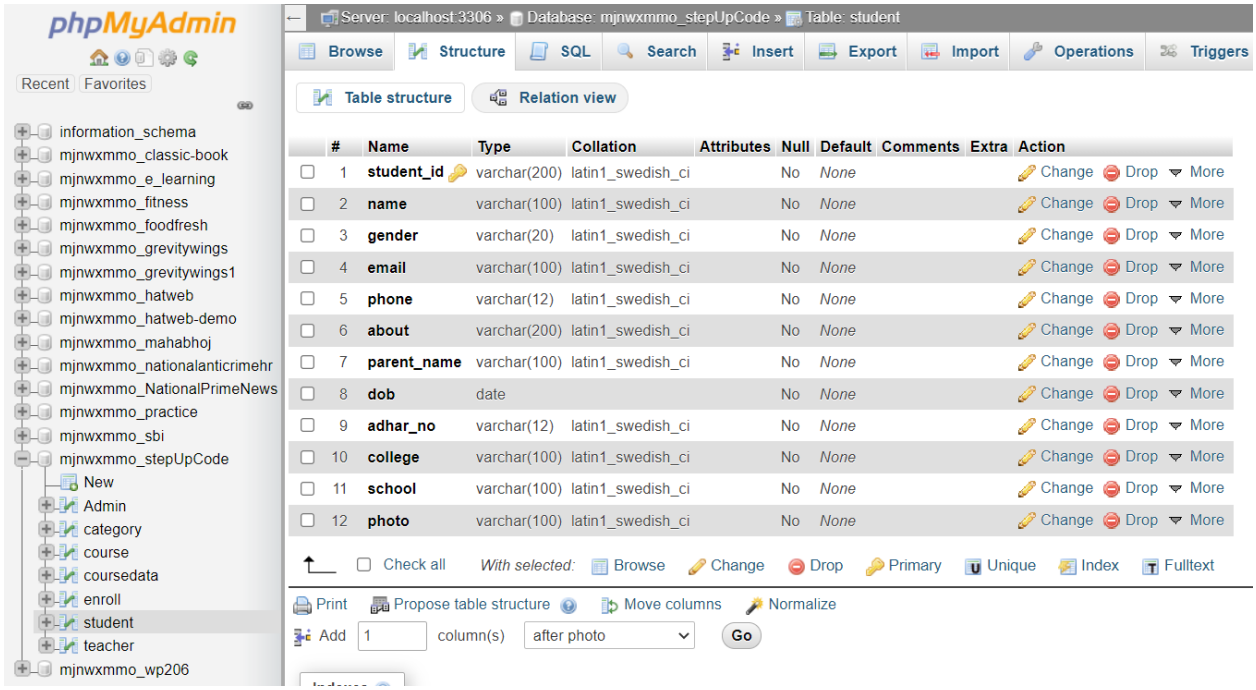
coursedata Table:



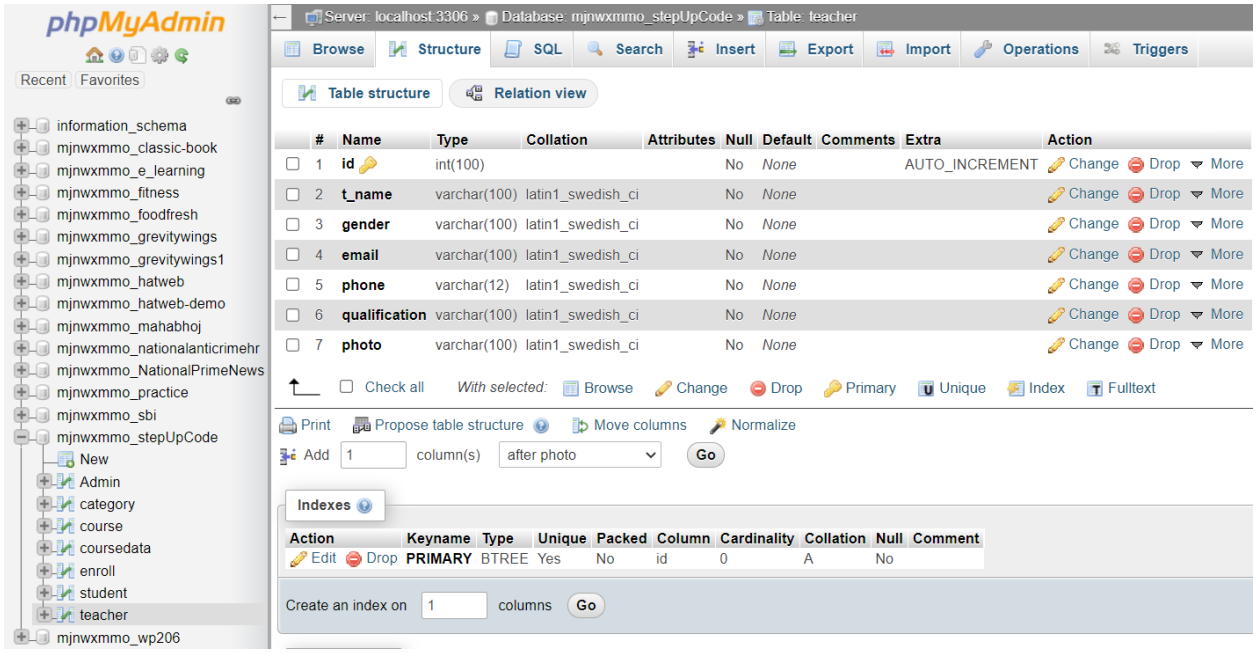
enroll Table:



student Table:



teacher Table:



5.USER INTERFACE DESIGN & IMPLEMENTATION

User interface design (UID) or **user interface engineering** is the design of user interfaces for machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centred design).

Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to it. Graphic design and typography are utilized to support its usability, influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the interface. The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

Interface design is involved in a wide range of projects from computer systems, to cars, to commercial planes; all of these projects involve much of the same basic human interactions yet also require some unique skills and knowledge. As a result, designers tend to specialize in certain types of projects and have skills centered on their expertise, whether that be software design, user research, web design, or industrial design.

5.1: Coding

index.js

```
JS index.js
5   import { AppRegistry } from 'react-native';
6   import App from './App';
7   import { name as appName } from './app.json';
8   import * as React from 'react';
9   import { DefaultTheme, Provider as PaperProvider, Colors } from 'react-native-paper';
10
11  AppRegistry.registerComponent(appName, () => App);
12
```

App.js

```
JS App.js > App
1   import React from 'react';
2   import { StyleSheet, Text, View, SafeAreaView, StatusBar } from 'react-native'
3   import { Colors } from 'react-native-paper'
4   import HomeScreen from './src/screen/HomeScreen'
5   import Demo from './src/screen/Demo';
6   import Splash from './src/screen/Splash';
7   import LogIn from './src/screen/LogIn';
8   import Front from './src/screen/Front'
9   import SignUp from './src/screen/SignUp';
10
11  import Loading from './src/Components/Loading';
12  import Videocard from './src/Components/Videocard';
13  import Shop from './src/screen/Shop';
14  import NavigationMain from './NavigationMain';
15  import { NavigationContainer } from '@react-navigation/native';
16  import { DefaultTheme, Provider as PaperProvider } from 'react-native-paper';
17
18  const theme = {
19    ...DefaultTheme,
20    roundness: 2,
21    colors: {
22      ...DefaultTheme.colors,
23
24    },
```

```
25  };
26  const App = () => {
27
28    return (
29
30      <NavigationContainer>
31
32        <NavigationMain />
33
34      </NavigationContainer>
35    )
36  }
37
38  const styles = StyleSheet.create({})
39
40  export default App;
```

NavigationMain.js

NavigationMain.js > NavigationMain

```
1   import React from 'react'
2   import { StyleSheet, Text, View, SafeAreaView, StatusBar } from 'react-native'
3   import { Colors } from 'react-native-paper'
4   import { NavigationContainer, DrawerActions } from '@react-navigation/native';
5   import { createStackNavigator } from '@react-navigation/stack';
6   import { createDrawerNavigator } from '@react-navigation/drawer';
7   import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
8   import Splash from './src/screen/Splash';
9   import SignUp from './src/screen/SignUp';
10  import LogIn from './src/screen/LogIn';
11  import HomeScreen from './src/screen/HomeScreen';
12  import Shop from './src/screen/Shop'
13  import CourseDetailse from './src/screen/CourseDetailse';
14  import TabNavigator from './TabNavigator'
15  import Notification from './src/screen/Notification';
16  import DrawerContent from './DrawerContent';
17  import MyVideo from './src/screen/MyVideo';
18  import Videoplayer from './src/screen/Videoplayer'
19  import Demo from './src/screen/Demo'
20  import EditProfile from './src/screen/EditProfile';
21  import VideoDemo from './src/screen/VideoDemo'
22  import Front from './src/screen/Front'
23  const Stack = createStackNavigator();
24  const Drawer = createDrawerNavigator();
```

NavigationMain.js > NavigationMain

```
27  function DrawerScreen({ navigtion }) {
28      return (
29          <Drawer.Navigator initialRouteName="Home" drawerContent={props => <DrawerContent {...props} />}>
30              <Drawer.Screen name="Tab" component={TabNavigator} />
31
32              {/*   <Drawer.Screen name="LogIN" component={LogIn} />
33              <Drawer.Screen name="SignUp" component={SignUp} />
34      */}
35          </Drawer.Navigator>
36      )
37  }
38  export default function NavigationMain({ navigtion }) {
39      return (
40
41          <Stack.Navigator initialRouteName='Splash' headerMode={'none'}>
42
43              <Stack.Screen name="SignUp" component={SignUp} />
44              <Stack.Screen name="Splash" component={Splash} options={{ headerShown: false }} />
45              <Stack.Screen name="CourseDetailse" component={CourseDetailse} options={{ headerShown: false }} />
46              <Stack.Screen name="Home" component={DrawerScreen} />
47              <Stack.Screen name="Shop" component={Shop} />
48              <Stack.Screen name="Notification" component={Notification} />
49              <Stack.Screen name="Videoplayer" component={Videoplayer} />
50              <Stack.Screen name="Demo" component={Demo} />
```

```
51              <Stack.Screen name="EditProfile" component={EditProfile} />
52              <Stack.Screen name="VideoDemo" component={VideoDemo} />
53              <Stack.Screen name="Front" component={Front} />
54              <Stack.Screen name="LogIn" component={LogIn} />
55              {/*   <Stack.Screen name="HomeScreen" component={DrawerScreen} />*/}
56
57          </Stack.Navigator>
58      )
59  }
60
61
62  const styles = StyleSheet.create({})
63
```

TabNavigaror.js

```
JS TabNavigator.js > TabNavigator
1   import React from 'react'
2   import { createMaterialBottomTabNavigator } from '@react-navigation/material-bottom-tabs';
3   import MaterialCommunityIcons from 'react-native-vector-icons/MaterialCommunityIcons';
4   import HomeScreen from './src/screen/HomeScreen';
5   import Shop from './src/screen/Shop';
6   import MyVideo from './src/screen/MyVideo';
7   import Profile from './src/screen/Profile';
8   import { Colors } from 'react-native-paper';
9   import { colors } from './src/Components/Theme'
10  const Tab = createMaterialBottomTabNavigator();
11
12  function TabNavigator() {
13      return (
14          <Tab.Navigator
15              initialRouteName="HomeScreen"
16              activeColor={Colors.white}
17              barStyle={{ backgroundColor: colors.primary }}
18          >
19              <Tab.Screen
20                  name="HomeScreen"
21                  component={HomeScreen}
22                  options={{
23                      tabBarLabel: 'Home',
24                      tabBarIcon: ({ color }) => (
```

```
                        <MaterialCommunityIcons name="home" color={color} size={26} />
                    ),
                }}
            />
            <Tab.Screen
                name="Shop"
                component={Shop}
                options={{
                    tabBarLabel: 'Store',
                    tabBarIcon: ({ color }) => (
                        <MaterialCommunityIcons name="shopping" color={color} size={26} />
                    ),
                }}
            />
            <Tab.Screen
                name="Video"
                component={MyVideo}
                options={{
                    tabBarLabel: 'My Video',
                    tabBarIcon: ({ color }) => (
                        <MaterialCommunityIcons name="video-plus" color={color} size={26} />
                    ),
                }}
            />
```

```
            <Tab.Screen
                name="Profile"
                component={Profile}
                options={{
                    tabBarLabel: 'Profile',
                    tabBarIcon: ({ color }) => (
                        <MaterialCommunityIcons name="account" color={color} size={26} />
                    ),
                }}
            />
        </Tab.Navigator>
    );
}
export default TabNavigator
```

DrawerContent.js

JS DrawerContent.js > DrawerContent > logout > then() callback

```
import React, { useState, useEffect } from 'react'
import { StyleSheet, View } from 'react-native'
import { DrawerContentScrollView, DrawerItem } from '@react-navigation/drawer'
import {
    Avatar, Title, Caption, Paragraph, Drawer, Text, TouchableRipple, Switch
} from 'react-native-paper'
import Icon from 'react-native-vector-icons/MaterialCommunityIcons'
import { SocialIcon } from 'react-native-elements'
import { colors } from './src/Components/Theme'
import * as OpenAnything from 'react-native-openanything'

import Share from 'react-native-share';
import auth from '@react-native-firebase/auth';
import { clearAll } from './src/localSession'
const DrawerContent = (props) => {
    const [data, setData] = useState([])
    const showAll = () => {
        var SearchAPIURL = "https://hatwebsolution.com/stepUpCode/Profile_api.php"

        var header = {
            'Accept': 'Application/json',
            'Content-Type': 'Application.json'
        };
```

```
        var Data = {
            StudentId: 'ePumjDN3R7fqVMVnIS2LsR0U3pE3',
        };
        fetch(
            SearchAPIURL,
            {
                method: 'POST',
                headers: header,
                body: JSON.stringify(Data)
            }
        )
            .then((response) => response.json())
            .then((response) => {


                setData(response[0])
            })
            .catch((error) => {
                alert("Api Error", +error)
                console.log(error)
            })
    }
    useEffect(() => {
        showAll();
```

```
50      }, []);
51      const myCustomShare = async () => {
52          const ShareOption = {
53              method: Share.FacebookStories.SHARE_BACKGROUND_AND_STICKER_IMAGE, // iOS only
54              backgroundVideo: 'URI_TO_MP4', // Android only (uri to a local file)
55              backgroundImage: 'http://urlto.png', // url or an base64 string
56              //or you can use "data:" url
57              appId: '219376304', //facebook appId
58              social: Share.Social.FACEBOOK_STORIES
59          };
60          try {
61              const ShareResponse = await Share.shareSingle(ShareOption)
62          } catch (error) {
63              console.log('Error=>', error)
64          }
65      }
66      const logout = () => {
67
68          auth()
69              .signOut()
70              .then(() => clearAll().then((res) => {
71                  if (res) {
72                      alert('logout succesful')
73                      props.navigation.pop()
```

```
74                      props.navigation.navigate('Front')
75                  }
76
77              }).catch((e) => console.log(e))
78
79
80              )
81      }
82      return (
83          <View style={{ flex: 1 }}>
84
85              <DrawerContentScrollView {...props}>
86                  <View style={styles.drawerContent}>
87                      <View style={styles.userInfoSection}>
88                          <View style={{ flexDirection: 'row', marginTop: 15 }}>
89                              <Avatar.Image source={{ uri: 'https://i.pinimg.com/564x/6f/de/
                                85/6fde85b86c86526af5e99ce85f57432e.jpg' }}
90                                  size={50} />
91                              <View style={{ marginLeft: 15, flexDirection: 'column' }}>
92                                  <Title style={styles.title}>{data.Name}</Title>
93                                  <Caption style={styles.Caption}>{data.email}</Caption>
94                              </View>
95                          </View>
96                          <View style={styles.row}>
```

```
96                  <View style={styles.row}>
97                      <View style={styles.section}>
98                          <Paragraph style={[styles.paragraph, styles.caption]}>80</
                            Paragraph>
99                          <Caption style={styles.caption}>Following</Caption>
100                     </View>
101                     <View style={styles.section}>
102                         <Paragraph style={[styles.paragraph, styles.caption]}>80</
                            Paragraph>
103                         <Caption style={styles.caption}>Follower</Caption>
104                     </View>
105                 </View>
106             </View>
107             <Drawer.Section style={styles.drawerSection}>
108                 <DrawerItem
109                     icon={({ color, size }) => (
110                         <Icon name="home-outline"
111                             color={color}
112                             size={size} />
113                     )}
114                     label="Home"
115                     onPress={() => { props.navigation.navigate('Home') }}
116                 />
```

```
117                 <DrawerItem
118                     icon={({ color, size }) => (
119                         <Icon name="account-edit-outline"
120                             color={color}
121                             size={size} />
122                     )}
123                     label="Edit Profile"
124                     onPress={() => { props.navigation.navigate('Profile') }}
125                 />
126                 <DrawerItem
127                     icon={({ color, size }) => (
128                         <Icon name="credit-card-outline"
129                             color={color}
130                             size={size} />
131                     )}
132                     label="Payments"
133                     onPress={() => { console.log("Payments") }}
134                 />
```

```
                        <DrawerItem
                            icon={({ color, size }) => (
                                <Icon name="cog-outline"
                                    color={color}
                                    size={size} />
                            )}
                            label="Settings"
                            onPress={() => { console.log("Settings") }}
                        />
                        <DrawerItem
                            icon={({ color, size }) => (
                                <Icon name="file-document-edit-outline"
                                    color={color}
                                    size={size} />
                            )}
                            label="Terms & Condtion "
                            onPress={() => { OpenAnything.pdf('./Bubble.pdf') }}
                        />
                    </Drawer.Section>
```

```
                    <Drawer.Section style={styles.drawerSection}>
                        <SocialIcon
                            title='Share on Facebook'
                            button
                            type='facebook'
                            style={{ backgroundColor: colors.primary }}
                            onPress={myCustomShare}
                        />

                    </Drawer.Section>
                </View>
            </DrawerContentScrollView>
            <Drawer.Section style={styles.bottomDrawerSection}>
                <DrawerItem
                    icon={({ color, size }) => (
                        <Icon name="exit-to-app"
                            color={color}
                            size={size} />
                    )}
                    label="Sign Out"
                    onPress={logout}
                />
            </Drawer.Section>
        </View>
```

```
178         )
179     }
180
181     const styles = StyleSheet.create({
182         drawerContent: {
183             flex: 1,
184         },
185         userInfoSection: {
186             paddingLeft: 20,
187         },
188         title: {
189             fontSize: 16,
190             marginTop: 3,
191             fontWeight: 'bold',
192         },
193         caption: {
194             fontSize: 14,
195             lineHeight: 14,
196         },
197         row: {
198             marginTop: 20,
199             flexDirection: 'row',
200             alignItems: 'center',
201         },
```

```
    section: {
        flexDirection: 'row',
        alignItems: 'center',
        marginRight: 15,
    },
    paragraph: {
        fontWeight: 'bold',
        marginRight: 3,
    },
    drawerSection: {
        marginTop: 15,
    },
    bottomDrawerSection: {
        marginBottom: 15,
        borderTopColor: '#f4f4f4',
        borderTopWidth: 1
    },
    preference: {
        flexDirection: 'row',
        justifyContent: 'space-between',
        paddingVertical: 12,
        paddingHorizontal: 16,
    },
})
```

```
export default DrawerContent
```

HomeScreen.js

src > screen > JS HomeScreen.js > HomeScreen

```
import React, { useState } from 'react';
import { StyleSheet, View, ScrollView, StatusBar } from 'react-native'
import { IconButton, Appbar, Colors, BottomNavigation, Text, ActivityIndicator, Surface,
Button } from 'react-native-paper';
import { NavigationContainer, DrawerActions } from '@react-navigation/native';
import { Image, } from 'react-native-elements';
import FreeVideo from '../Components/FreeVideo';
import { colors } from '../Components/Theme'
import Loading from '../Components/Loading';

const HomeScreen = ({ navigation }) => {
    const [loading, setLoading] = useState(true);
    const pressbutton = (Id) => {
        navigation.navigate('CourseDetailse', { id: Id })
    }
    const loadCheck = () => {
        setLoading(false)
    }
    const viewAll = () => {
        navigation.navigate('Video')
    }
    const playVideo = (name) => {
        navigation.navigate('Videoplayer', { name: name })
    }
```

```
    return (
        <View style={styles.container}>
            <View>
                <StatusBar backgroundColor={colors.primary} />
                <Appbar.Header>
                    <Appbar.Action icon="menu" onPress={() => navigation.dispatch
                    (DrawerActions.toggleDrawer())} />
                    <Appbar.Content title="Step Up Code" />

                    <Appbar.Action icon="bell" onPress={() => navigation.push
                    ('Notification')} />
                </Appbar.Header>
            </View>
            <ScrollView>
                <View>


                    <Image
                        source={require('../images/bannar.png')}
                        style={{ width: "100%", height: 200, padding: 20 }}
                        PlaceholderContent={<ActivityIndicator />}
                    />
                </View>
                <View>
```

```
                    <Text>What do You whant to do ?</Text>
                </View>
                <View>
                    <Surface style={styles.surface}>
                        <View style={styles.surfaceElement}>
                            <IconButton
                                icon="video-vintage"
                                color={colors.primary}
                                size={40}
                                onPress={() => navigation.push('Videoplayer')}
                            />
                            <Text style={{ top: -10 }}>Watch Video</Text>
                        </View>
                        <View style={styles.surfaceElement}>
                            <IconButton
                                icon="clipboard"
                                color={colors.primary}
                                size={40}
                                onPress={() => navigation.push('Demo')}
                            />
                            <Text style={{ top: -10 }}>Practice Now</Text>
                        </View>

                        <View style={styles.surfaceElement}>
                            <IconButton
                                icon="chart-line"
                                color={colors.primary}

                                size={40}
                                onPress={() => navigation.push('VideoDemo')}
                            />
                            <Text style={{ top: -10 }}>Performance</Text>
                        </View>
                    </Surface>
                </View>
                <View>
                    <FreeVideo press={pressbutton} lode={loadCheck} viewVideo={viewAll}
                    play={playVideo} />
                </View>
            </ScrollView>
        </View>
    )
}
```

```
89  const styles = StyleSheet.create({
90      container: {
91          justifyContent: 'center',
92          height: "100%",
93          backgroundColor: colors.background
94
95      },
96      surface: {
97          flex: 1,
98          flexDirection: "row",
99          padding: 10,
100         marginTop: 10,
101         height: 100,
102         width: "100%",
103         alignItems: 'center',
104         justifyContent: 'center',
105         elevation: 8,
106         color: colors.surface
107
108     },
```

```
109     surfaceElement: {
110         justifyContent: 'space-between',
111         paddingHorizontal: 20,
112         marginBottom: 5
113     }
114 })
115 export default HomeScreen;
```

Shop.js

src > screen > JS Shop.js > ...

```
1   import React, { useEffect, useState } from 'react'
2   import { StyleSheet, Text, View, ScrollView } from 'react-native'
3   import { Colors } from 'react-native-paper'
4   import { Searchbar, Appbar } from 'react-native-paper';
5   import { ListItem } from 'react-native-elements'
6   import CourseCard from '../Components/CourseCard'
7   import Loading from '../Components/Loading';
8   import { DrawerActions } from '@react-navigation/native';
9
10
11  export default function Shop({ navigation }) {
12      const [searchQuery, setSearchQuery] = React.useState('');
13      const [result, setResult] = useState([]);
14      const [loading, setLoading] = useState(true);
15      const showAll = () => {
16          var ShowApIURL = "https://hatwebsolution.com/stepUpCode/showCourse_api.php";
17          var headers = {
18              'Accept': 'Application/json',
19              'Content-Type': 'Application.json'
20          };
21          fetch(ShowApIURL, {
22              method: 'GET',
23              headers: headers
24          })
```

```
            .then((response) => response.json())
            .then((response) => {
                setResult(response)
            })
            .catch((error) => {
                alert("API Error" + error);
                console.error(error)
            });
    }

    useEffect(() => {
        showAll();
        setLoading(false);
    }, []);
    const ShowCourse = (Id) => {
        navigation.navigate('CourseDetailse', { id: Id })
    }
    return (
        loading ? <>
            <Loading />
        </> :
            <View style={styles.container}>
                <View>
                    <Appbar.Header>
```

```
                        <Appbar.Action icon="menu" onPress={() => navigation.dispatch
                        (DrawerActions.toggleDrawer())} />
                        <Appbar.Content title="Shop" />
                        <Appbar.Action icon="bell" onPress={() => navigation.navigate
                        ('Notification')} />
                    </Appbar.Header>
                </View>
                <View style={{ padding: 5 }}>
                    <Searchbar
                        placeholder="Search"
                        onChangeText={(input) => setSearchQuery(input)}
                        value={searchQuery}

                    />
                </View>
                <ScrollView>

                    {result.map((element, i) =>

                        <View key={i}>

                            <CourseCard data={element} press={ShowCourse} />
                        </View>

```

```
71                          )
72
73
74                          }
75                      </ScrollView>
76                  </View >
77          )
78      }
79
80      const styles = StyleSheet.create({
81          container: {
82              justifyContent: 'center',
83              height: "100%",
84              backgroundColor: Colors.grey200
85
86          },
87      })
88
```

MyVideo.js

src > screen > JS MyVideo.js

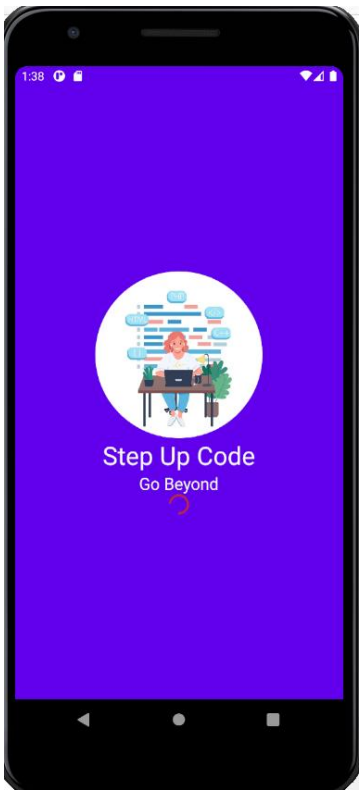
```
import React, { useEffect, useState } from 'react'
import { StyleSheet, Text, View, SafeAreaView, ScrollView, TouchableOpacity } from 'react-native'
import { Appbar, Surface } from 'react-native-paper'
import { Image, Divider, ListItem, Avatar, FlatList } from 'react-native-elements';
import { colors, fontSizes } from '../Components/Theme'
import { DrawerActions } from '@react-navigation/native';
import Loading from '../Components/Loading';
import { set } from 'react-native-reanimated';
const MyVideo = ({ navigation }) => {
    const [loading, setLoading] = useState(true);
    const [result, setResult] = useState([]);
    const showFree = () => {
        var ShowApIURL = "https://hatwebsolution.com/stepUpCode/show_free_api.php";
        var headers = {
            'Accept': 'Application/json',
            'Content-Type': 'Application.json'
        };
        fetch(ShowApIURL, {
            method: 'GET',
            headers: headers
        })
            .then((response) => response.json())
            .then((response) => {
                console.log("response = ", response);
                setResult(response)
                setLoading(false);

            })
            .catch((error) => {
                alert("API Error" + error);
                console.error(error)
            });
    }

    useEffect(() => {

        showFree();

        console.log(loading, result)


    }, [loading]);


    return (
        loading ?
            <>
```

```
48                <Loading />
49            </> :
50            <SafeAreaView>
51                <View style={styles.container}>
52                    <View>
53                        <Appbar.Header>
54                            <Appbar.Action icon="menu" onPress={() => navigation.dispatch
                              (DrawerActions.toggleDrawer())} />
55                            <Appbar.Content title="My Video" />
56
57                            <Appbar.Action icon="bell" onPress={() => navigation.navigate
                              ('Notification')} />
58                        </Appbar.Header>
59                    </View>
60                    <ScrollView>
61                        <View style={{ padding: 5 }}>
62                            <Text style={{ fontSize: fontSizes.title, fontWeight: 'bold' }}
                              >Free Courses</Text>
63                        </View>
64
65                        <View>
66                            {
67                                result.map((l, i) => (
68                                    <TouchableOpacity key={i}>
```

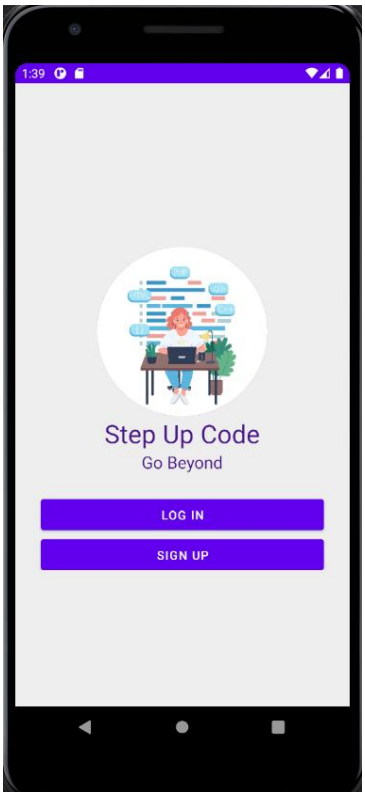
```
69                                      <ListItem bottomDivider containerStyle={{ height:
                                        100, margin: 5 }}>
70                                          <Avatar size='large' source={require('../
                                            images/bannar.png')} />
71                                          <ListItem.Content>
72                                              <ListItem.Title>{l.CourseName}</ListItem.
                                                Title>
73                                              <ListItem.Subtitle>sub</ListItem.Subtitle>
74                                          </ListItem.Content>
75                                          <ListItem.Chevron style={{ color: colors.
                                            primary }} />
76                                      </ListItem>
77                                  </TouchableOpacity>
78                              ))
79                          }
80                      </View>
81                      <View style={{ padding: 5 }}>
82                          <Text style={{ fontSize: fontSizes.title, fontWeight: 'bold' }}
                            >Your Courses</Text>
83                      </View>
84                  </ScrollView>
85
86              </View>
87          </SafeAreaView>
```

```
88      )
89  }
90
91  const styles = StyleSheet.create({
92      container: {
93          backgroundColor: colors.background,
94          height: '100%'
95      },
96      surface: {
97          padding: 8,
98          height: 150,
99          width: '100%',
100
101         elevation: 20,
102     },
103 })
104 export default MyVideo;
```

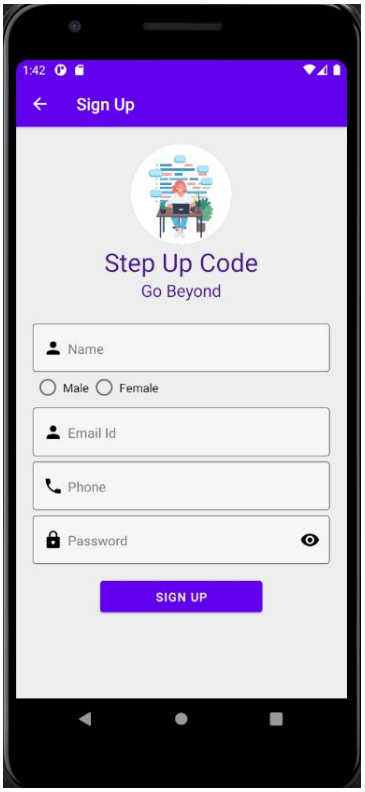
5.2: Snapshot



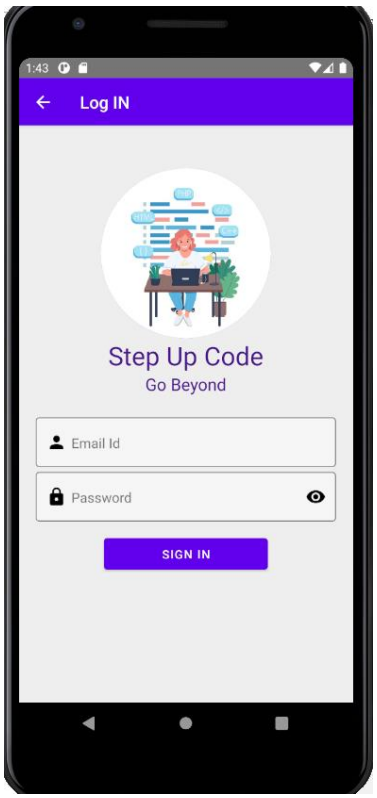
Splash Screen

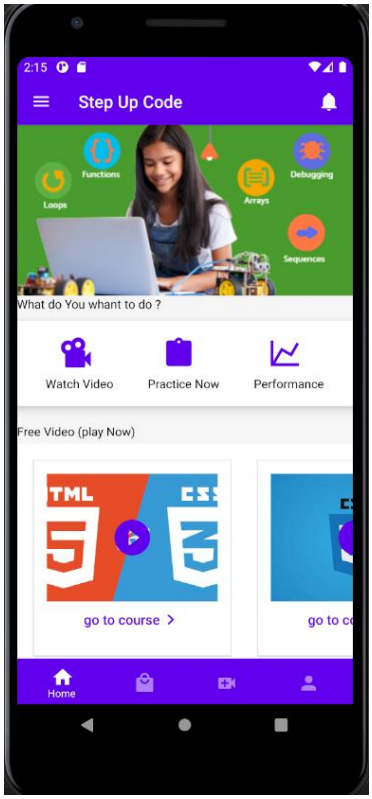


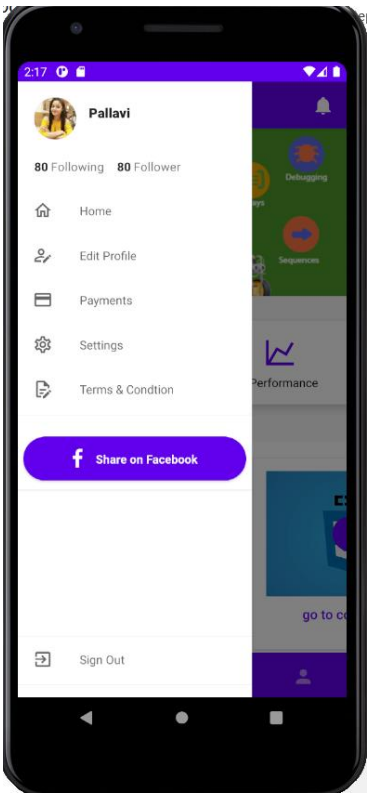
Front screen



Sign Up Screen



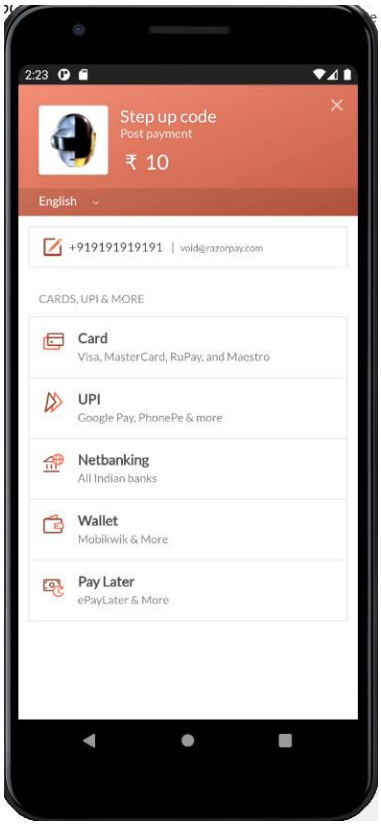




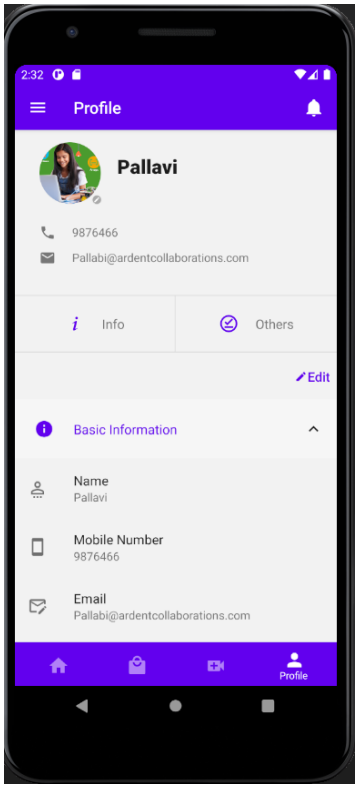
Login Screen **Home screen** **Drawer navigation**

Shop Screen

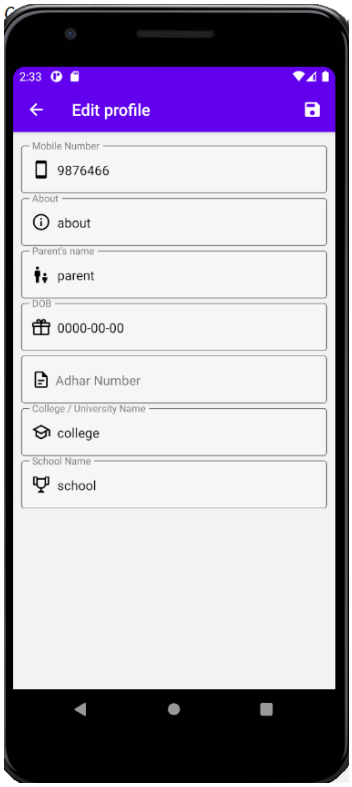
Course details



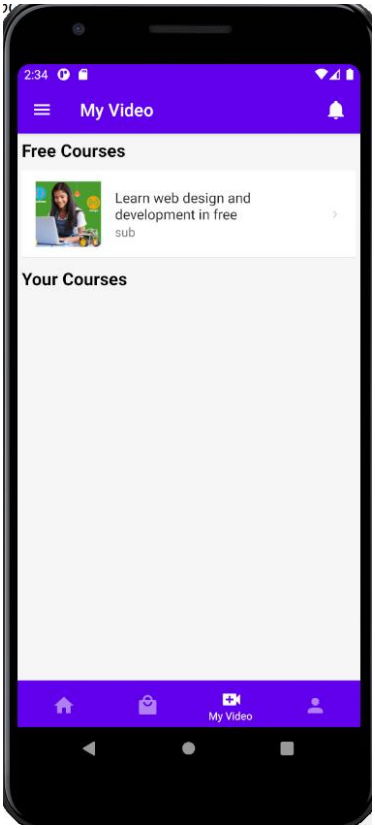
Payment screen

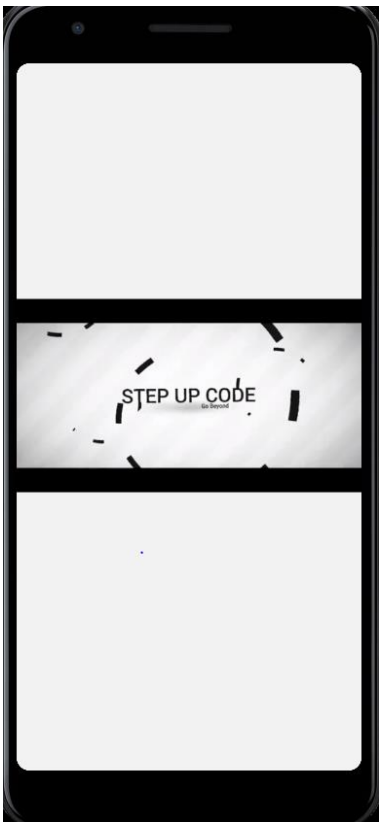


Profile screen



Edit Profile





My video **Video player**

6. Why responsive in mobile??

We have designed a fully responsive APP with the help of frontend framework and design libraries. Now a days smartphones are different size and different resolution most widely used and every user would fancy all web activity to be done through those gadgets. Hence we have come forward with this idea of a fully responsive app with a responsive menu which automatically fits the app in any screen size and readjusts its components as and when required, be it smartphones.

7. Testing and Maintenance :-

A software system test plan is a document that describes the objectives, scope, approach and focus of software testing effort. The process of preparing a test plan is a usual way to think the efforts needed to validate the acceptability of a software product. The complete document will help people outside the test group understand the "WHY" and "HOW" product validation. It should be through enough to be useful but not so through that no one outside the test group will read it.

Introduction of Testing

Testing is the process of running a system with the intention of finding errors. Testing enhances the integrity of a system by detecting deviations in design and errors in the system. Testing aims at detecting error-prone areas. This helps in the prevention of errors in a system. Testing also adds value to the product by conforming to the user requirements.

The main purpose of testing is to detect errors and error-prone areas in a system. Testing must be thorough and well-planned. A partially tested system is as bad as an untested system. And the price of an untested and under-tested system is high.

The implementation is the final and important phase. It involves user-training, system testing in order to ensure successful running of the proposed system. The user tests the system and changes are made according to their needs. The testing involves the testing of the developed system using various kinds of data. While testing, errors are noted and correctness is the mode.

7.1: OBJECTIVES OF TESTING

The objective our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. Our user interface to utilize these functions is designed to be user-friendly and provide easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

Process Overview

The following represents the overall flow of the testing process:

1. Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.
2. Identify which particular test(s) will be used to test each module.
3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
4. Identify the expected results for each test.
5. Document the test case configuration, test data, and expected results.
6. Perform the test(s).
7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report (STR).
8. Successful unit testing is required before the unit is eligible for component integration/system testing.
9. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.
10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

7.2: TEST CASES

A test case is a document that describe an input, action, or event and expected response, to determine if a feature of an application is working correctly. A test case should contain

particular such as test case identifier, test condition, input data. The process of developing test cases can help find problems in the requirement or design of an application, since it requires completely thinking through the operation of the application.

WHITE BOX TESTING

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test case designers shall generate cases that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once. To ensure this happens, we will be applying Branch Testing. Because the functionality of the program is relatively simple, this method will be feasible to apply.

Each function of the binary tree repository is executed independently; therefore, a program flow for each function has been derived from the code.

BLACK BOX TESTING

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Equivalence Partitioning and Boundary Value Analysis testing on our application.

System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case well focus only on function validation and performance. And in both cases, we will use the black-box method of testing

Test Cases

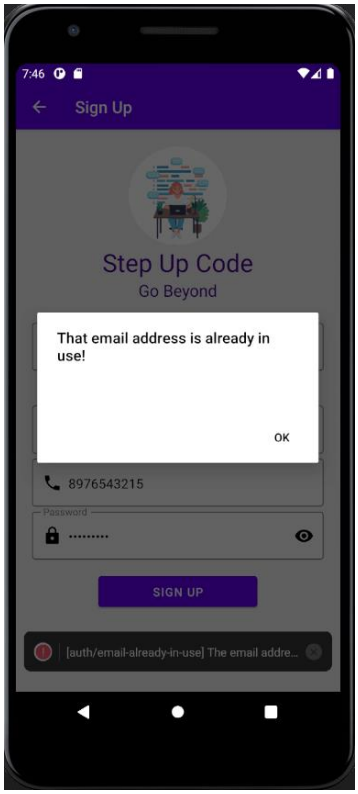
Test case Number	1
Test case Name	Registration page

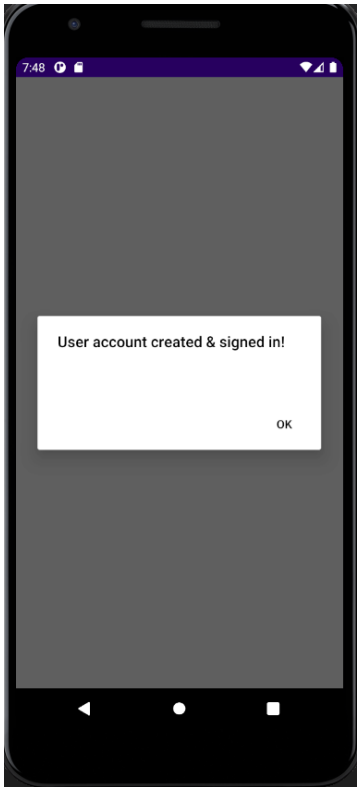
Test case Description	Should update student details into server, verify email
Test Performed By	Rohit

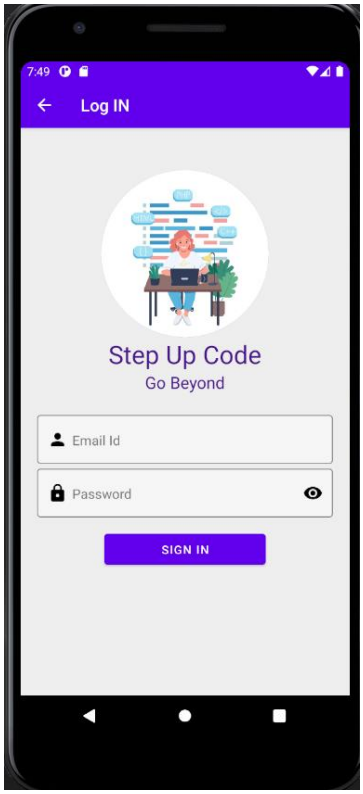
Items to be tested:

- Successfully sign up
- Redirect to login page.

Input	Expected output
1. Fill the form an tab the button	1. Successfully register 2. Redirect to the login page. 3. Sign up failure message.





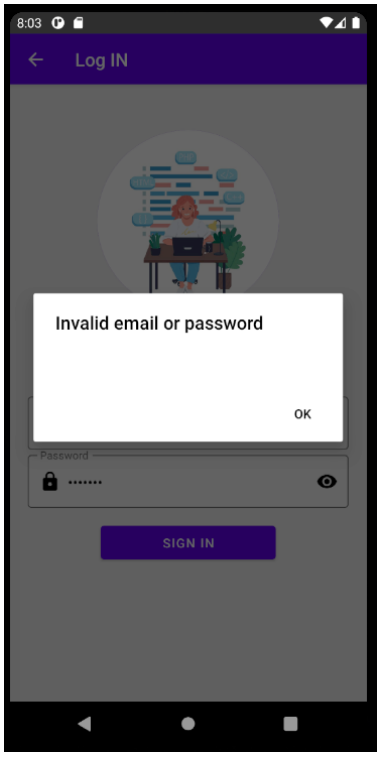


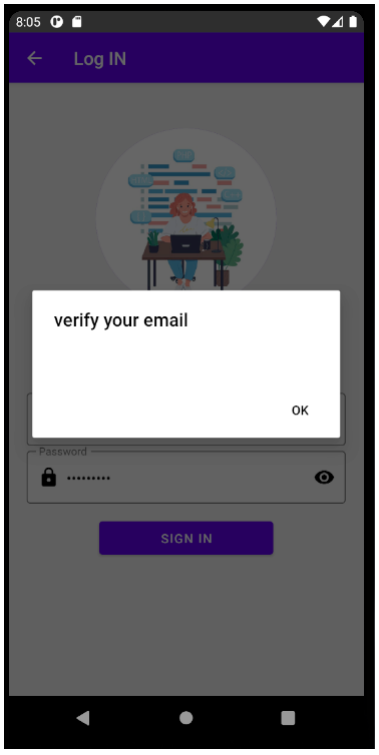
Test case Number	2
Test case Name	Login page
Test case Description	Login using email and password
Test Performed By	Rohit

Items to be tested:

- Successfully login
- Redirect to Home page.

Input	Expected output
1. Fill the form an tab the button	2. Successfully Login 3. Redirect to the Home page 4. Login failure message.



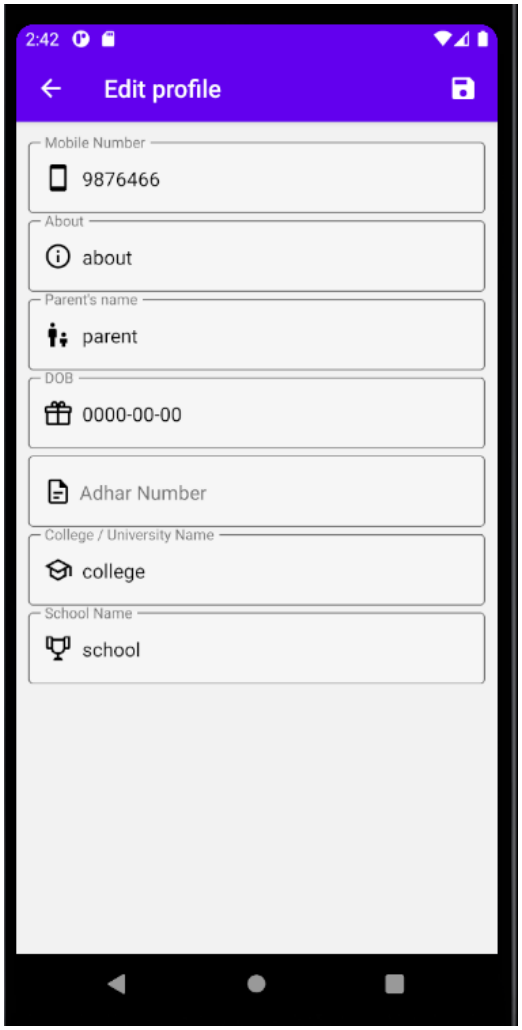


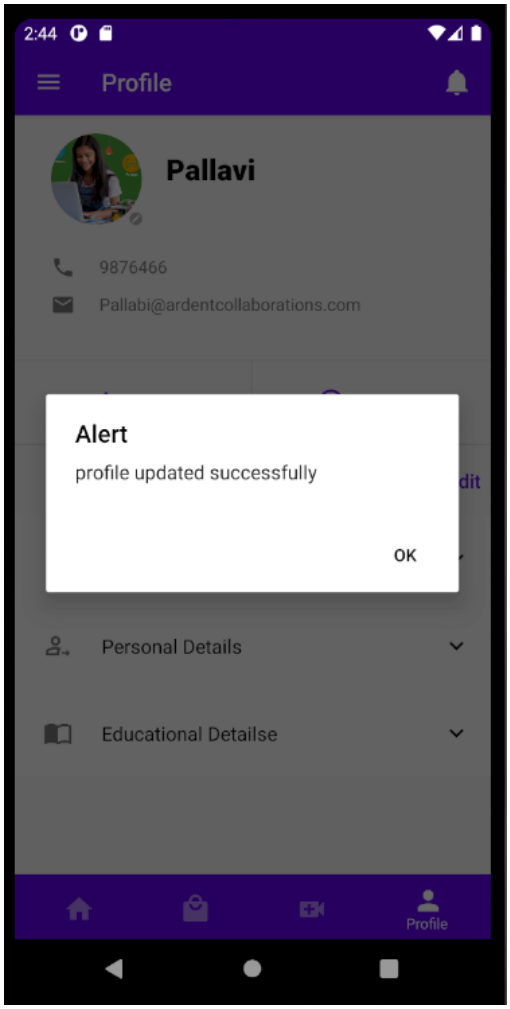
Test case Number	3
Test case Name	Profile Update
Test case Description	Give the details of student
Test Performed By	Rohit

Items to be tested:

- Successfully update

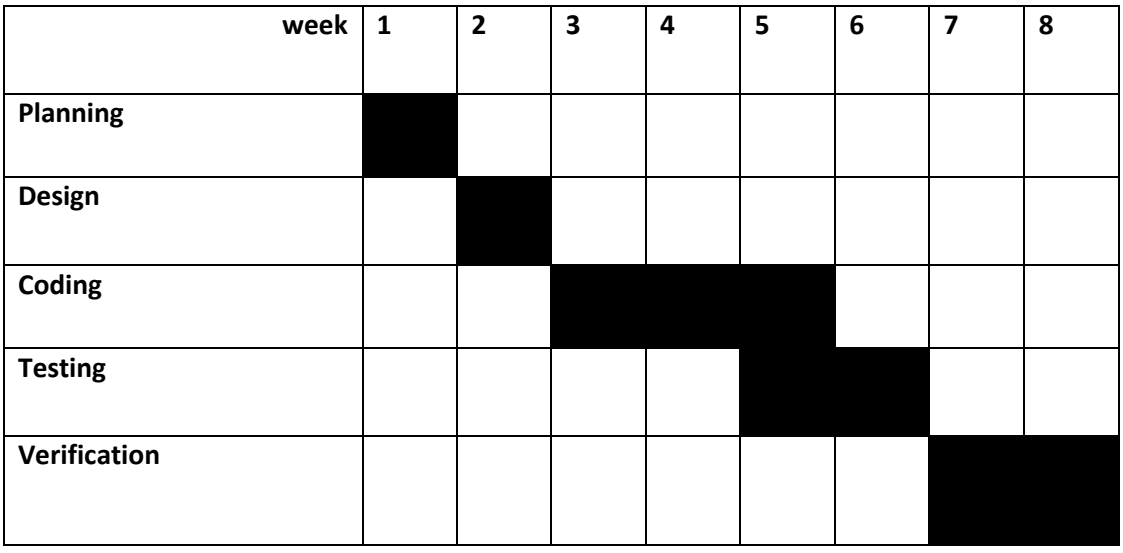
Input	Expected output
5. Fill the form an tab the button	1. Successfully update. 2. Update failure message.





7.3:GNATT CHART

SLno.	Task	Start date	End date
1	Planning	7th May	12th May
2	Design	13th May	26th May
3	Coding	27th May	20th June
4	Testing	21th June	30thJune
5	Verification	1th July	8th July



8. SYSTEM SECURITY MEASURES

DATABASE SECURITY

System security measure is meant to be provided to make your system reliable and secured from unauthorized user may create threats to the system. So you should follow some security measures. We have used security levels in database level at system level

SYSTEM SECURITY

If we talk about the system security in our proposed system we have implemented with the help of maintain the session throughout the system's use. Once a user has logged out than he/she will not be able to perform any task before signing back again.
A high level of authentic login is given to the system so this is a very tedious task to enter without authorization and authentication.

LIMITATION

There are some limitations in the App:

1. The app only supports vertical mode not Horizontal mode.
2. We can't fast forward the video.
3. user must have account to access the app.
4. Brightness control is not available while playing the video.

9. FUTURE SCOPE AND FURTHER ENHANCEMENTS

In future, we would like to keep working on this project and make new additions to provide users with more advanced features and more detailed information. We have set our sights on the following additions in future-

1. Forgot Password
2. Push notification
3. Use live classes feature.
4. Add examination feature.
5. Live chat

10. CONCLUSION

This project has been appreciated by all the users in the organization. It is easy to use, since it uses the GUI provided in the user dialog. User friendly screens are provided. The usage of software increases the efficiency, decreases the effort. It has been efficiently employed as a Site management mechanism. It has been thoroughly tested and implemented.

The project **"Step up code"** is the ideal place for every Student who want to learn from home in this covid-19 pandemic situation. The software provides a reliable platform for keeping all sensitive information. For this kind of online business, the special software must be installed on the server which host the site, or on a secure server which receives all sensitive data.

11. REFERENCES

1. Head First PHP & MySQL – by Lynn Beighley & Michael Morrison

2. A Smarter Way to Learn JavaScript by Mark Myers.

3. SQL, PL/SQL: The Programming Language of Oracle
by Ivan Bayross (shelved 5 times as dbms)

The links of the website Listed below:

1. Viewed July 05.2021:- https://reactnative.dev/
2. Viewed July 03.2021 :- https://en.wikipedia.org/wiki/React_Native
3. Viewed July 04.2021:- https://www.npmjs.com/package/react-native
4. Viewed July 05.2021:-https://www.javatpoint.com/react-native-introduction
5. Viewed June 30.2021 :-https://www.tutorialspoint.com/react_native/index.htm
6. Viewed June 30.2021:-https://github.com/react-native-community