ONLINE LEARNING APP

A Project Report Submitted
in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF COMPUTER APPLICATION

By

SAURABH VISHWKARMA
University roll no. 1900290140030

Under the Supervision of
Ms. Vidushi
ASSISTANT PROFESSOR
KIET Group of Institutions, Delhi NCR Ghaziabad

Department of Computer Applications
Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY
LUCKNOW
(Formerly Uttar Pradesh Technical University, Lucknow)

(JUNE 2022)

Declaration

I undersigned hereby declare that the project report ("**Online Learning App**") , submitted for partial fulfillment of the requirements for the requirement for the award of the degree of Master of Computer Applications by the 'KIET GROUP OF INSTITUTION'S GHAZIABAD' is a bonafide work done by me under supervision of (Pro. Vidhushi Mishra). This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission.

Name : Saurabh Vishwkarma

Roll No: 1900290140030

Branch: MCA (6^{th} Sem)

(Candidate Signature)

Certificate

Certified that **Saurabh Vishwkarma (enrollment no. 190029014005130)** has carried out the project work having "Order Guide" 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:

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

Date:

Pro. Ms. Vidhushi
Associate Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

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

SUCCESSIVE TECHNOLOGIES

Deloitte. Technology Fast 50

CMMI level 3

Great Place To Work. Certified SEP 2020-AUG 2021 INDIA

Traineeship Letter

Ref: ST/SD/21/883 **Date: 24/05/2022**

TO WHOM IT MAY CONCERN

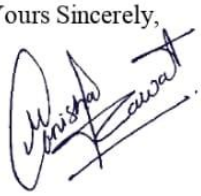
This is to certify that Saurabh Vishwkarma is working as a **Trainee Engineer**, with our organization, i.e. Successive Technologies Private Limited from **06/12/2021** to **04/06/2022**.

During his traineeship period he worked on the POC "Online Learning App" in Mobility department, under guidance of Varun Chauhan.

We wish Saurabh Vishwkarma, all the good luck and success in his future endeavors.

For Successive Technologies

Yours Sincerely,



Manisha Rawat

Senior Manager-Human Resource

Successive Technologies Private Limited

E-29, Sector-11,
Noida-201301, India

support@successive.tech
+91-120-425-9482, +91-120-426-9272

INDIA | USA | UK | SOUTH AFRICA

Abstract

Online learning app is a great learning tool for students who do not have access to a physical classroom. It is useful for students who are working professionals and wish to improve. It allows students living in remote areas to attend classes. This method of learning has many benefits like ease of education. An Online learning app is installed on the smartphone and can be used anytime. It gives access to both live sessions and pre-recorded classes to students. Trainers use audio-visual mode for teaching. Students engage in the online classroom and interact with other students very easily. There are many online learning apps offering millions of courses. Many schools and colleges also give students the benefit of attending classes online in admissions. These allow students to choose subjects of their interest. There are both paid and free courses available to choose from. Students even get a certificate of completion when they finish a course.

ACKNOWLEDGEMENT

I extend my deepest appreciation to Successive Technologies Pvt. Ltd .My esteemed guide, for providing me with the possibility to complete this project with the right guidance and advice.

Special gratitude I give to my respected head of the division **Dr. Ajay Kr Srivastav, Head of Department, and all faculties Master of Computer Applications**, for allowing me to use the facilities available and also help me to coordinate my project
Furthermore, I would also like to acknowledge with much appreciation the crucial role of faculty members on this occasion. Last but not least, I would like to thank friends who help me to assemble the parts and gave a suggestion about the project.

.

TABLEOFCONTENT

CHAPTER 1

INTRODUCTION

1.1 PROJECT DESCRIPION

The online applications for academic purposes are called online learning app. Such applications make use of an internet connection. An Online learning app can be accessed from a smartphone. It is a technology-based study tool that enables information sharing. It is commonly known as mobile apps for learning.

Online learning app is a great learning tool for students who do not have access to a physical classroom. It is useful for students who are working professionals and wish to improve. It allows students living in remote areas to attend classes.

This method of learning has many benefits like ease of education. An Online learning app is installed on the smartphone and can be used anytime. It gives access to both live sessions and pre-recorded classes to students. Trainers use audio-visual mode for teaching. Students engage in the online classroom and interact with other students very easily.

There are many online learning apps offering millions of courses. Many schools and colleges also give students the benefit of attending classes online in admissions. These allow students to choose subjects of their interest. There are both paid and free courses available to choose from. Students even get a certificate of completion when they finish a course.

The method of online learning apps is user-friendly. Students who attend the session have to mark their attendance. Students get assignments to complete within deadlines. Students can track their performances. They can even give feedback for the improvement of learning and teaching methods.

Purpose:-

An Online learning app is installed on the smartphone and can be used anytime. It gives access to both live sessions and pre-recorded classes to students. Trainers use audio-visual mode for teaching. Students engage in the online classroom and interact with other students very easily.

SCOPE:-

The main purpose of education is to achieve upward mobility. Online courses certification programs have been able to provide inexpensive education to the masses and also save time, energy and money. Electronic-learning through certified online courses provides a wide range of courses that caters to the core interests of the student, thus creating a fertile arena for future advancement .There is a misplaced notion, that employers prefer students with traditional brick and mortar college degrees.

On the contrary, corporate organizations in India are recognizing the high skill levels of students who have undergone online courses certification programs from highly acclaimed educational institutions.

CHAPTER 2

Literature Review

Automated SMS plays a great role in simplifying the job of employees at the school and satisfying the need of customers and stakeholders of the school .Even though no documentation is found in Ethiopia to be reviewed, products have been observed at some schools to help understand the problem of managing schools and handling school data. This chapter reviews these products

2.1 Observed Products.

In the year 2003 City Government of Addis Ababa Education Bureau (CGAAEB)was very much interested to have automated school management system to getuniform and quick access to the students' data for administrative purpose onpromoting the students' achievement and related issues. The bureau has selected Wundrad Preparatory School for pilot test. At the time the school principals together with officials from CGAAEB signed a contractual agreement with some software developer company. The developers installed their first version of the product which can register a student offline and generate official transcript with some level ofdifficulty. As the system is not fully automated, it does not support management of attendance, does not support generating report cards and other important functions such as generating school timetable and a web based report for parents. Due to the lack of follow up by the government officials at CGAAEB, the company was unable to complete the project. The school currently is unable to use the partially developed system because of lack of trained person and lack of hardware and software maintenance.

2.2 Manual Timetabling

Manual timetables are prepared by dedicated teachers. In manual timetabling, it is common to proceed in an iterative fashion where each iteration selects and schedules a lesson [3].Scheduling a lesson requires to choose a classroom (fixed for each section of students) and a time slot such that the commitment to the choice will not violate any constraint. In school timetabling, we are required to schedule a given set of meetings such that the resulting timetables are feasible and acceptable to all people involved. Humans are able to prepare the timetable using some hit/miss approach. So it is possible to automate the timetable based on asimulation of the human way of solving the problem. Such techniques, that we call direct heuristics, were based on a successive augmentation. That is, a partial timetable is extended, lecture by lecture, until all lectures have been scheduled. The underlying idea of all approaches is to schedule the most constrained lecture first. Usually some responsible teachers are assigned to schedule subjects and teachers. The number of teachers available per each subject is predefined and the load that each teacher has is calculated. With these data the time table constructor assigns each teacher-subject association to the appropriate classes with the available time slots. The manual solution of the timetabling problem usually requires many

person-days of work. In addition, the solution obtained may be unsatisfactory. The lessons should be fairly distributed satisfying the identified constraints.

2.3 Drawbacks of the Reviewed Systems

The reviews described have the following problems:

- Generate official transcript with some level of difficulty,
- Do not sufficiently produce the required reports to allow parents to view status of their children and reports for officials of kebele and kifle-ketema to help them participate in decision making,
- Do not generate timetable for the schools
- Do not facilitate attendance record keeping by the homeroom teachers

This project work tries to fill the gap by automating the various activities at schools. It tries to satisfy customers need and simplify the works of administrators, record officer and teachers. With an automated school management system parents can easily interact with the school community to follow up their children's achievement and play their role in the school development processes.

CHAPTER 3

PROJECT CATEGORY

3.1 Technology Used

React Native

React Native is an open-source UI software framework created by Meta Platforms, Inc. It is used to develop applications for Android, Android TV, iOS, macOS, tvOS, Web, Windows and UWP by enabling developers to use the React framework along with native platform capabilities. It provides a slick, smooth and responsive user interface, while significantly reducing load time. It's also much faster and cheaper to build apps in React Native as opposed to building native ones, without the need to compromise on quality and functionality.

Software and Applications Used

APPLICATION	:	Visual Studio , Android Studio
OPERATING SYSTEM	:	WINDOWS 10
FRONT END	:	React Native
BACK END	:	SQlite , Firebase

Back-end :

- ✓ **Firebase:-** It is realtime database developed by firebase and then acquired by Google in 2014 used for developing the high quality applications act as a storage of information of various data.

- ✓ **SQlite:-** SQLite is a database engine written in the C language. It is not a standalone app; rather, it is a library that software developers embed in their apps. As such, it belongs to the family of embedded databases

3.2Language Used

This project has been developed in React Native and JavaScript

- ✓ **React Native :** React Native is an open-source UI software framework created by Meta Platforms, Inc. It is used to develop applications for Android, Android TV, iOS, macOS, tvOS, Web, Windows and UWP by enabling developers to use the React framework along with native platform capabilities.

- ✓ **Stylesheet :** StyleSheet is React Native StyleSheet is a way of styling an application using JavaScript code, the main function of react native StyleSheet is concerned with styling and structuring of components in an application, all components of react native make use of a prop known as style, names and properties values work in a similar way as the web StyleSheet . There are three type of styling in react native

- ➢ Inline
- ➢ Internal or Embedded CSS
- ➢ External CSS

- **SQlite:-** SQLite is a database engine written in the C language. It is not a standalone app; rather, it is a library that software developers embed in their apps. As such, it belongs to the family of embedded databases

Chapter 4

Software Requirement Specification

4.1 Hardware Configuration :

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware.

Memory – All software, when run, resides in the random access memory (RAM) of computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

Number	Description
1	PC with 500 GB or more Hard disk
2	PC with 8 GB RAM.
3	2nd generation Intel Core or newer

Table 4.1 Hardware Requirements

4.2 Software Requirements

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view .Every project needs software. We should try to understand what sort of requirements may arise in the requirement elicitation phase and what kinds of requirements are expected from the software system.

Number	Description	Type
1	Operating System	Windows XP / Windows
2	Language	React Native
3	Database	SQlite
4	Android Studio	Emulator
5	Server	NPM

Table 4.2 Software Requirements

4.3 Functional Requirements:

Internet Connectivity:

As discussed that Application will work on Online mode so it need regular Internet Connectivity to signup and login.

Facebook Account:

User can directly login to the facebook account to access this application they don't need to signin in apps environment for which facebook account is mandatory.

Email id and Mobile Number:

To access the application and to signin or login user must have email id and mobile number to fill the mandatory field in the form.

4.4 Non-functional Requirements:

Performance Requirements

- User friendly**:** The system should be user friendly so that it can easily be understand by the user without any difficulty.
- Ease of maintenance :- System should be easy to maintain and use.
- Less time consuming: The system should be less time consuming which could be achieved by good programming.
- Error free: The system should easily handle the user error in any case.
- Static: Application runs on stand alone machine i.e. Android mobile phone of API level 16 and onward. Support only single user.

4.5 Software System Attributes:

1. **Security**: The system should be secure from the unauthorized access and should be password protected so that no other user can access it.

   If the user is new then he needs to Signup with required details and a can also login with the facebook.

2. **Portability**:- The system should be machine independent.

3. **Maintainability**: The system will be designed in a maintainable order. The system can be easily modified and renewed according to the need of the organization.

4.6 Features

- Security of data.
- Ensures data accuracy.
- Minimize manpower.
- Minimum time consumption.
- Greater efficiency.
- Fast
- Better services.
- User friendliness and interactive.
- Minimum time required.
- Easy to update
- User friendly
- Free for the user
- knowing about entry time and exit time

The online applications for academic purposes are called online learning app. Such applications make use of an internet connection. An Online learning app can be accessed from a smartphone. It is a technology-based study tool that enables information sharing. It is commonly known as mobile apps for learning.

Online learning app is a great learning tool for students who do not have access to a physical classroom. It is useful for students who are working professionals and wish to improve. It allows students living in remote areas to attend classes.

This method of learning has many benefits like ease of education. An Online learning app is installed on the smartphone and can be used anytime. It gives access to both live sessions and pre-recorded classes to students. Trainers use audio-visual mode for teaching. Students engage in the online classroom and interact with other students very easily.

There are many online learning apps offering millions of courses. Many schools and colleges also give students the benefit of attending classes online in admissions. These allow students to choose subjects of their interest. There are both paid and free courses available to choose from. Students even get a certificate of completion when they finish a course.

4.7 Preliminary investigation:

Fact Finding:
After obtaining the background knowledge, we began to collect data on the existing system.

The tools that are used in information gathering are as follows:

- On-site observation.
- Questionnaire.
- Review of the peoples.

The model we have used is Waterfall Model. In this model, first of all the existing system is observed, then customer requirements are taken in consideration then planning, modelling, construction and finally deployment.

4.8Approach used: Agile Approach

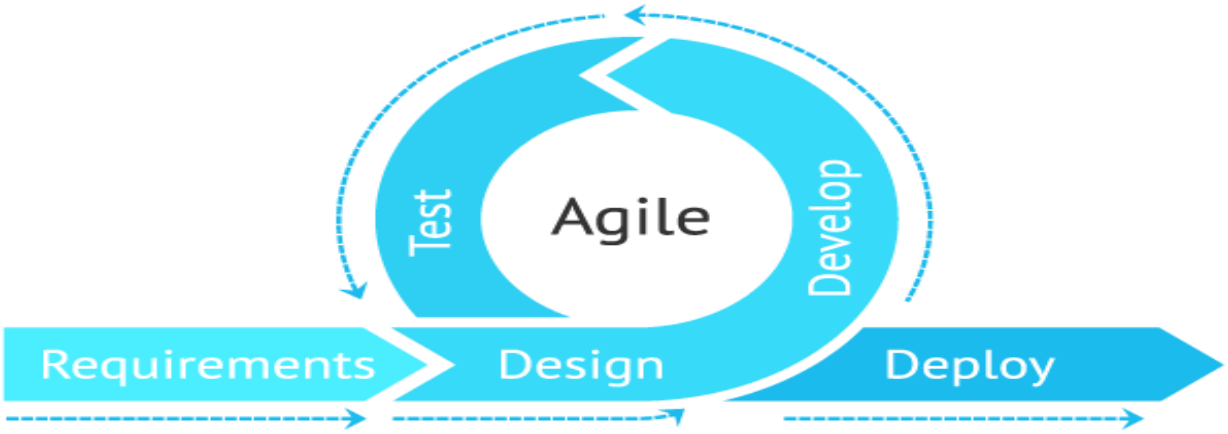


Fig. 3.1

Agile is **an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches**. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments.

4.9 Preliminary Description:

The first step in the system development life cycle is the preliminary investigation to determine the feasibility of the system. The purpose of preliminary investigation is to evaluate project requests. It is not a design study nor does it include the collection of details to describe the system in all respect. Rather, it is the collecting of information that helps committee members to evaluate the merits of project request and make an informed judgment about the feasibility of the proposed project.

Analyst working on the preliminary investigation should accomplish the following objectives:

- Clarify and understand the project request.
- Determine the size of the project.
- Access costs and benefits of alternative approaches.
- Determine the technical and operational feasibility of alternative approaches.
- Report the findings to management with recommendations outlining the acceptance and rejection of the proposal.

Chapter 5

Feasibility study

After studying and analyzing all the existing and requires functionalities of the system, the next task is to do the feasibility study for the project. Feasibility study includes consideration of all the possible ways to provide a solution to a given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements.

5.1 Economical Feasibility:

It will be freely available on the Google play store without having any cost.

5.2 Technical feasibility:

This included the study of function, performance and constraints that may affect the ability to achieve an acceptable system. For this feasibility study, we studied complete functionalities to be provided in the system, as described in the System Requirement Specification (SRS), and checked if everything was possible using different type of front end and back end platform.

5.3 Operational Feasibility:

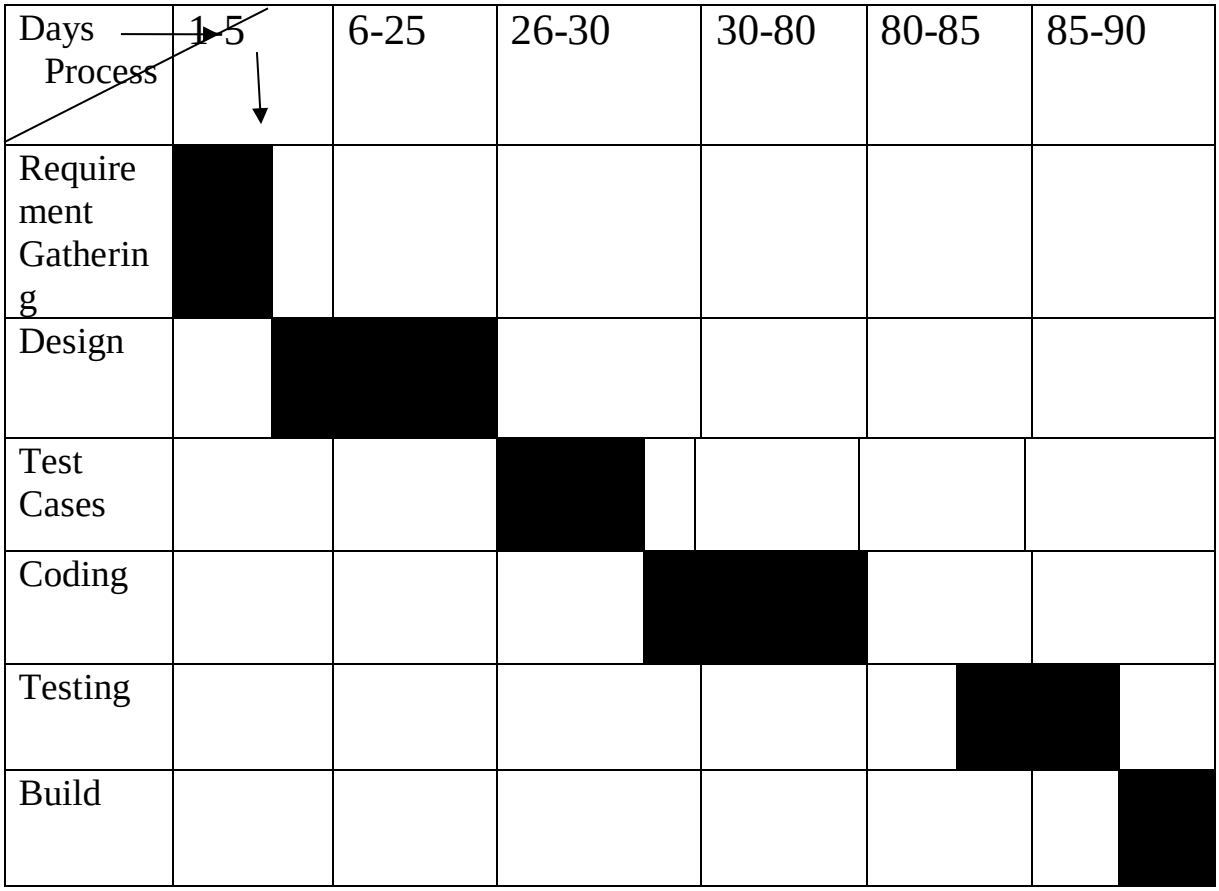
No doubt the technically growing Bihar needs more enhancement in technology, this apps is very user friendly and all inputs to be taken all self-explanatory even to a layman.

Chapter 6

Planning and Scheduling and Flow

6.1 Gantt chart

A Gantt chart can be developed for the entire project or a separate chart can be developed for each function. A tabular form is maintained where rows indicate the task with milestones and columns indicate duration (weeks/months).



Chapter 7

Analysis and Design

7.1 Analysis:

In present all visitor work done on the paper. The whole year visitor is stored in the registers. We can't generate reports as per our requirements because its take more time to calculate the visitors report.

Disadvantage of present system:

- **Not user friendly:** The present system not user friendly because data is not stored in structure and proper format.
- **Manual Control:** All report calculation is done manually so there is a chance of error.
- **Lots of paper work:** Visitors maintain in the register so lots of paper require storing details.
- **Time consuming**

7.2 Design Introduction:

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization.

Once the software requirements have been analyzed and specified the software design involves three technical activities - design, coding, implementation and testing that are required to build and verify the software.

The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer's requirements into finished software or a system.

Design is the place where quality is fostered in development. Software design is a process through which requirements are translated into a representation of software. Software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data.

Chapter 8

Data Flow Diagram

A **data-flow diagram** is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

There are several notations for displaying data-flow diagrams. The notation presented above was described in 1979 by Tom DeMarco as part of structured analysis.

For each data flow, at least one of the endpoints (source and / or destination) must exist in a process. The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes.

The data-flow diagram is a tool that is part of structured analysis and data modeling. When using UML, the activity diagram typically takes over the role of the data-flow diagram. A special form of data-flow plan is a site-oriented data-flow plan.

Data-flow diagrams can be regarded as inverted Petri nets, because places in such networks correspond to the semantics of data memories. Analogously, the semantics of transitions from Petri nets and data flows and functions from data-flow diagrams should be considered equivalent.

Notations in the DFD:

Symbol	Description
	The circle or bubble represents a process. A process is named and each process is represented by a named circle.
	The source or sink is represented as a rectangular box. The source or sink is the net originator or the consumer of the data that flows in the system.
	The arrow represents the flow of data through the system. The labelled arrows enter or leave the bubbles.
	The database is represented with the open box symbol.

8.2 Diagram

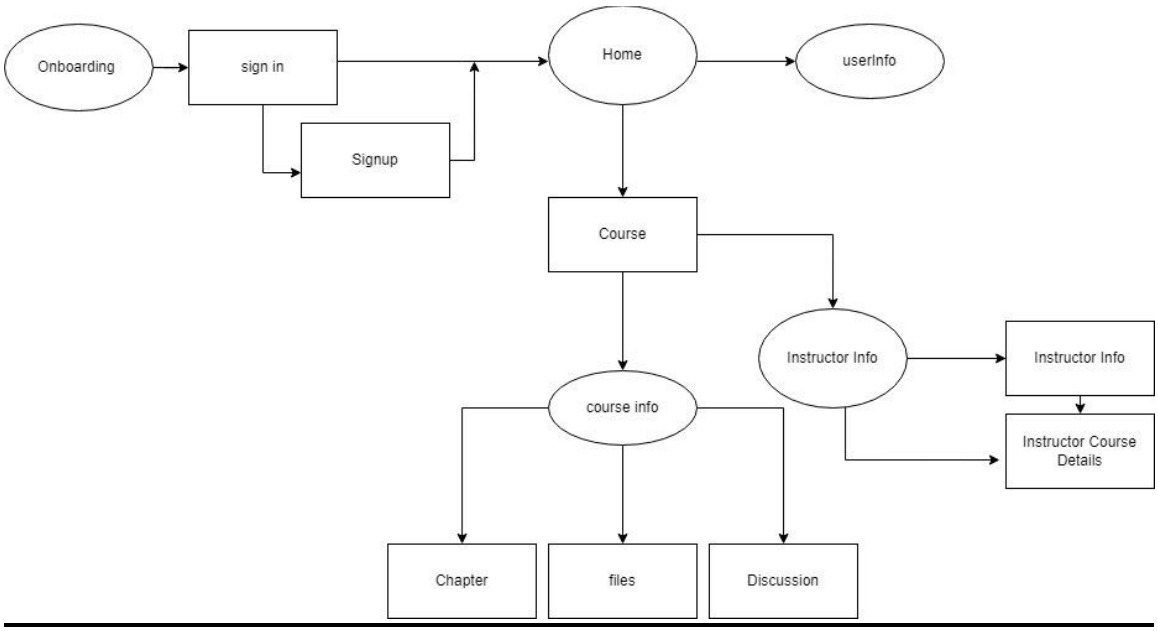


Fig 6.1

Chapter 9

ER Diagram

9.1 Description

The Entity-Relationship (ER) model was originally proposed by Peter in 1976 [Chen76] as a way to unify the network and relational database views. 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 design for the database designer, the utility of the ER model is:

- It maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables.
- It is simple and easy to understand with a minimum of training. Therefore, the model can be used by the database designer to communicate the design to the end user.
- In addition, the model can be used as a design plan by the database developer to implement a data model in specific database management software.

7.1 ER Notation

There is no standard for representing data objects in ER diagrams. Each modeling methodology uses its own notation. The original notation used by Chen is widely used in academics texts and journals but rarely seen in either CASE tools or publications by non-academics. Today, there are a number of notations used; among the more common are Bachman, crow's foot, and IDEFIX.

All notational styles represent entities as rectangular boxes and relationships as lines connecting boxes. Each style uses a special set of symbols to represent the cardinality of a connection. The notation used in this document is from Martin. The symbols used for the basic ER constructs are:

- **Entities** are represented by labeled rectangles. The label is the name of the entity. Entity names should be singular nouns.

- **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.

9.2 Diagram

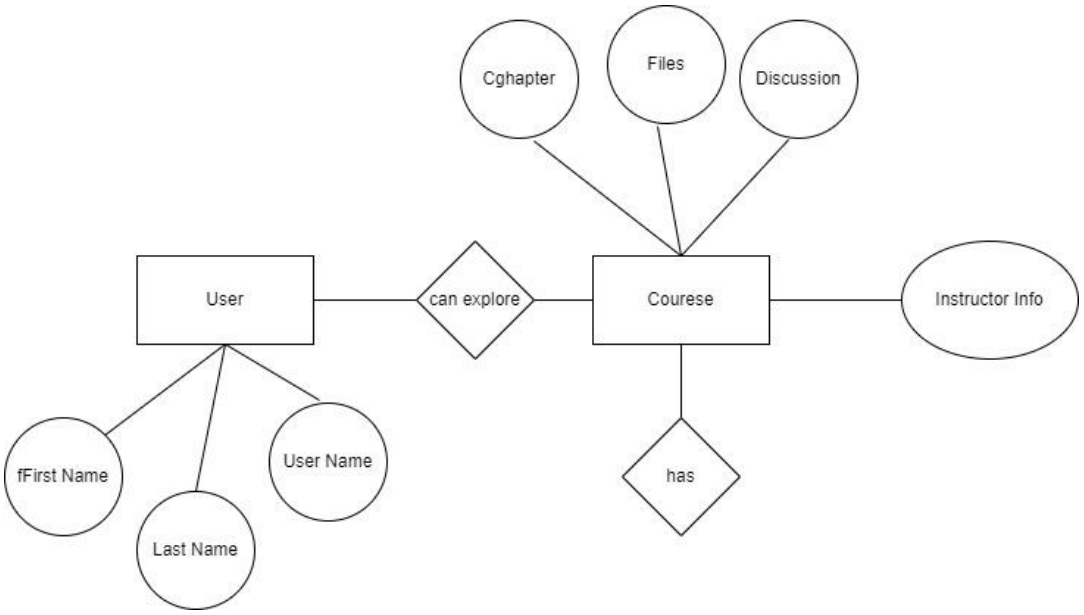


Fig 9.2

Chapter 10

Implementation and System Testing

After all phase have been perfectly done, the system will be implemented to the server and the system can be used.

10.1 System Testing

The goal of the system testing process was to determine all faults in our project.The program was subjected to a set of test inputs and many explanations were made and based on these explanations it will be decided whether the program behaves as expected or not. Our Project went through two levels of testing

1. Unit testing
2. Integration testing

UNIT TESTING

Unit testing is commenced when a unit has been created and effectively reviewed .In order to test a single module we need to provide a complete environment i.e. besides the section we would require

- The procedures belonging to other units that the unit under test calls
- Non local data structures that module accesses
- A procedure to call the functions of the unit under test with appropriate parameters

1. Test for the admin module

- **Testing login form-** This form is used for log in of administrator of the system. In this form we enter the username and password if both are correct administration page will open otherwise if any of data is wrong it will get redirected back to the login page and again ask the details.

INTEGRATION TESTING

In the Integration testing we test various combination of the project module by providing its features.The primary objective is to test the module interfaces in order to confirm that no errors are occurring when one module invokes the other module.

Chapter 11

Valuation

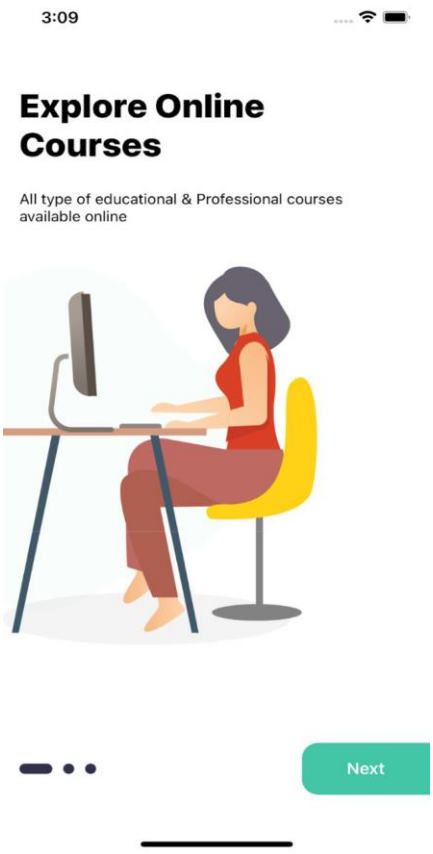
Features:

These are Features of the application that help the user

11.1 Walkthrough Screen

Walkthrough screens give brief introduction of application

- This is App Intro Slider Screen
- That Slide and show Intro of the App



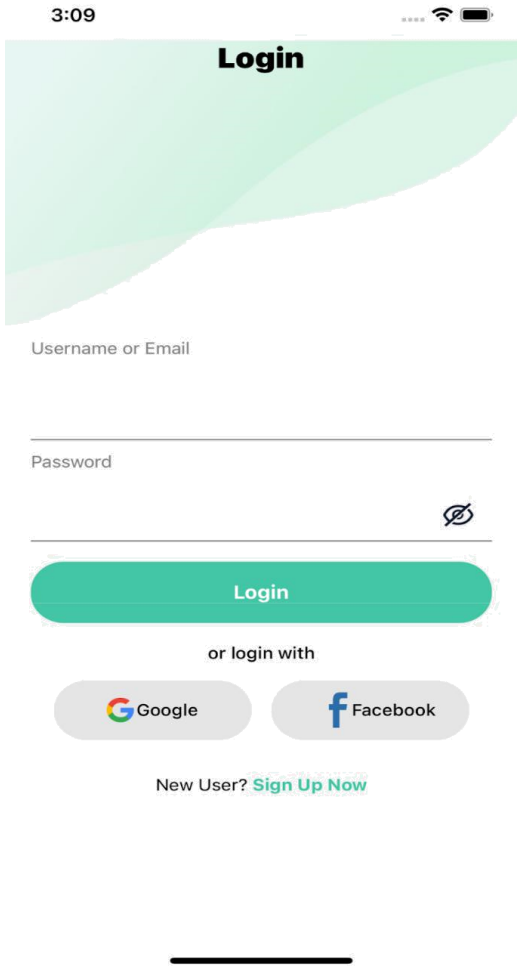
```
import
React,
{memo,
useRef}
from
'react';
           import {Animated, SafeAreaView, StatusBar, View} from 'react-native';
           import {Constant} from '../../config';
           import {COLORS, selectedTheme, SIZES} from '../../config/Themes';
           import {NavigationDataTypes} from '../../models';
           import RenderItems from './Renderitem';
           import styles from './style';


           const onBoardings = Constant.walkthrough;


           const OnboardingScreen = (props: NavigationDataTypes) => {
             const {navigation} = props;
             const [completed, setCompleted] = React.useState(false);


             const flatlistRef = useRef<any | null>(null);


             const scrollToIndex = (data: any) => {
               flatlistRef.current.scrollToIndex({animated: true, index: data + 1});
             };


             const scrollX = new Animated.Value(0);


             React.useEffect(() => {
               scrollX.addListener(({value}) => {
                 if (Math.floor(value / SIZES.width) === onBoardings.length - 1) {
                   setCompleted(true);
                 }
               });


               return () => scrollX.removeAllListeners();
             }, []);


             const renderDots = () => {
```

```
    const dotPosition = Animated.divide(scrollX, SIZES.width);


    return (
      <View style={styles(selectedTheme).dotsContainer}>
        <StatusBar backgroundColor={COLORS.additionalColor13}
barStyle={'dark-content'} />
        {onBoardings.map((item, index) => {
          const dotSize = dotPosition.interpolate({
            inputRange: [index - 1, index, index + 1],
            outputRange: [10, 17, 10],
            extrapolate: 'clamp',
          });


          return (
            <Animated.View
              key={index}
              style={[
                styles(selectedTheme).dot,
                {
                  width: dotSize,
                  height: dotSize,
                },
              ]}
            />
          );
        })}
      </View>
    );
  };


  return (
    <SafeAreaView style={styles(selectedTheme).container}>
      <View>
        <Animated.FlatList
          ref={flatlistRef}
          horizontal
          pagingEnabled={true}
          data={onBoardings}
          showsHorizontalScrollIndicator={false}
          keyExtractor={(item, index) => item + index.toString()}
          renderItem={({item}) => (
            <RenderItems
              item={item}
```

```
              navigation={navigation}
              nextdata={() => {
                scrollToIndex(item.id);
              }}
            />
          )}
          onScroll={e => scrollX.setValue(e.nativeEvent.contentOffset.x)}
        />
        <View style={styles(selectedTheme).dotsRootContainer}>
          {renderDots()}
        </View>
      </View>
    </SafeAreaView>
  );
};


export default memo(OnboardingScreen);
```

11.2. Login Screen :

- Login with credentials (username/email,password)
- Google Login
- Facebook Login



```
import
React,
{memo}
from
'react';
           import {
             View,
             Text,
             TextInput,
             TouchableOpacity,
             Image,
             SafeAreaView,
             ImageBackground,
             ScrollView,
           } from 'react-native';
           import {selectedTheme} from '../../config/Themes';
           import {
             buttons,
             form,
             Icon,
             Images,
             RouteScreens,
             screensData,
           } from '../../config';
           import styles from './style';


           interface loginpageprops {
             navigation: any;
             visible: boolean;
             setvisibe: React.Dispatch<React.SetStateAction<boolean>>;
             isvalidEmail: boolean;
             isvalidPassword: boolean;
             Emailinpute: (text: string) => void;
             EnterPassword: (text: string) => void;
             isEnabled: boolean;
             submit: any;
             signIn: any;
             Facebooklogin: any;
           }


           const LoginScreen = (props: loginpageprops) => {
             const {
               navigation,
               visible,
```

```
    setvisibe,
    isvalidEmail,
    isvalidPassword,
    Emailinpute,
    EnterPassword,
    isEnabled,
    signIn,
    Facebooklogin,
    submit,
  } = props;


  return (
    <SafeAreaView style={styles(selectedTheme).mainContainer}>
      <ImageBackground
        imageStyle={styles(selectedTheme).imageBackgroundStyle}
        source={selectedTheme.name == 'light' ? Images.BG : Images.BG_DARK}
        style={styles(selectedTheme).bgImage}>
        <Text
style={styles(selectedTheme).heading}>{screensData.LOGIN}</Text>
      </ImageBackground>
      <ScrollView
        style={styles(selectedTheme).container}
        showsVerticalScrollIndicator={false}>
        <View style={styles(selectedTheme).input}>
          <Text style={styles(selectedTheme).formText}>
            {form.USERNAMEOREMAIL}
          </Text>
          <View style={styles(selectedTheme).inputContainer}>
            <TextInput
              keyboardType={'email-address'}
              onChangeText={Emailinpute}
              style={styles(selectedTheme).inputText}
            />
          </View>
          {!isvalidEmail ? (
            <Text style={styles(selectedTheme).invalidText}>
              {form.validation.EMAILVALIDATION}
            </Text>
          ) : null}
        </View>
        <View style={styles(selectedTheme).input}>
          <Text
style={styles(selectedTheme).formText}>{form.PASSSWORD}</Text>
          <View style={styles(selectedTheme).inputContainer}>
            <TextInput
```

```
              style={styles(selectedTheme).inputText}
              secureTextEntry={!visible}
              onChangeText={EnterPassword}
            />
            <TouchableOpacity
              style={styles(selectedTheme).eyeContainer}
              onPress={() => (visible ? setvisibe(false) :
setvisibe(true))}>
              <Image
                source={visible ? Icon.EYE : Icon.EYE_CLOSE}
                style={styles(selectedTheme).eyeIcon}
              />
            </TouchableOpacity>
          </View>
          {!isvalidPassword ? (
            <Text style={styles(selectedTheme).invalidText}>
              {form.validation.PASSWORDVALIDATION}
            </Text>
          ) : null}
        </View>


        <TouchableOpacity style={styles(selectedTheme).button}
onPress={submit}>
          <Text
style={styles(selectedTheme).buttonText}>{buttons.LOGIN}</Text>
        </TouchableOpacity>


        <Text style={styles(selectedTheme).text}>
          {screensData.loginPage.OR_LOGIN_WITH}
        </Text>


        <View style={styles(selectedTheme).socialButtonContainer}>
          <TouchableOpacity
            style={styles(selectedTheme).socialButton}
            onPress={() => signIn()}>
            <Image source={Icon.GOOGLE} style={styles(selectedTheme).icon}
/>
            <Text style={styles(selectedTheme).socialButtonText}>
              {buttons.GOOGLE}
            </Text>
          </TouchableOpacity>
```

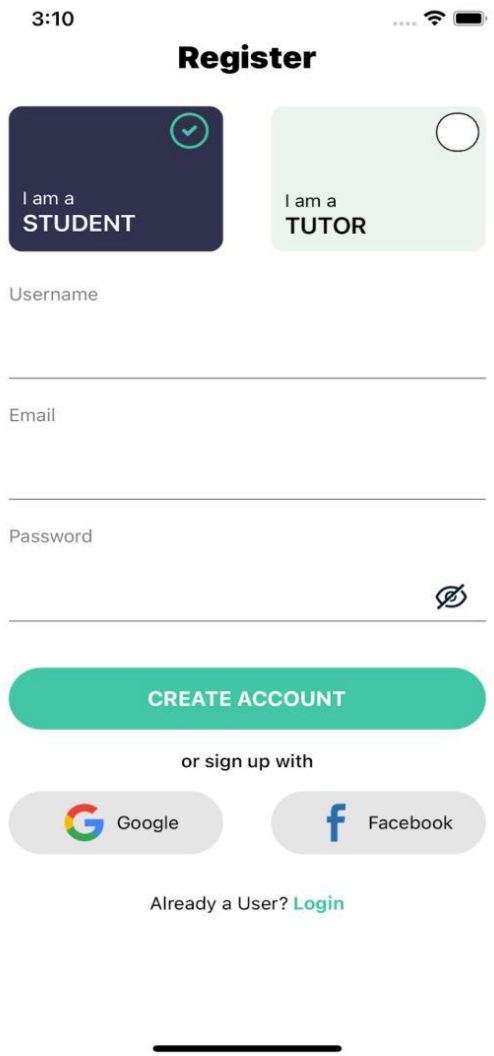
```
          <TouchableOpacity
            onPress={() => Facebooklogin()}
            style={styles(selectedTheme).socialButton}>
            <Image source={Icon.FACEBOOK} style={styles(selectedTheme).icon}
/>
            <Text style={styles(selectedTheme).socialButtonText}>
              {buttons.FACEBOOK}
            </Text>
          </TouchableOpacity>
        </View>


        <View style={styles(selectedTheme).buttonsContainer}>
          <Text style={styles(selectedTheme).newUserText}>
            {screensData.loginPage.NEW_USER_}
          </Text>


          <TouchableOpacity
            onPress={() => navigation.navigate(RouteScreens.REGISTERMODEL)}>
            <Text style={styles(selectedTheme).signupText}>
              {buttons.SIGNUPNOW}
            </Text>
          </TouchableOpacity>
        </View>
      </ScrollView>
    </SafeAreaView>
  );
};
export default memo(LoginScreen);
```

11.3. Register Screen :

- Users can register as a Student or Tutor
- Users can register by filling all the input fields

Here is some other and quick way to register in this Application :

- Register via Google
- Register via Facebook



```
import
React,
{useRef}
from
'react';
           import {
             View,
             Text,
             Image,
             TextInput,
             TouchableOpacity,
             ScrollView,
             StatusBar,
             Animated,
             SafeAreaView,
           } from 'react-native';
           import {buttons, form, Icon, RouteScreens, screensData} from '../../config';
           import {COLORS, selectedTheme} from '../../config/Themes';
           import styles from './style';


           interface Registerprops {
             navigation: any;
             visible: boolean;
             setvisibe: React.Dispatch<React.SetStateAction<boolean>>;
             selected: any;
             setselected: any;
             isvalidEmail: boolean;
             isvalidPassword: boolean;
             isvalidusername: boolean;
             Emailinpute: (text: string) => void;
             EnterPassword: (text: string) => void;
             EnterUserName: (text: string) => void;
             submit: any;
             register: {
               id: number;
               label: string;
             }[];
             defaultitem: any;
             setdefaultitem: any;
             GoogleSignin: () => Promise<void>;
             Facebooklogin: () => void;
           }


           const RegisterScreen = (props: Registerprops) => {
```

```
const {
  navigation,
  visible,
  setvisibe,
  selected,
  setselected,
  isvalidEmail,
  isvalidPassword,
  isvalidusername,
  Emailinpute,
  EnterPassword,
  EnterUserName,
  register,
  defaultitem,
  setdefaultitem,
  GoogleSignin,
  Facebooklogin,
  submit,
} = props;


const animationValue = useRef(new Animated.Value(0)).current;
const scaleValue = useRef(0);


const NEWButton = Animated.createAnimatedComponent(TouchableOpacity);


const runAnimationOnClick = () => {
  scaleValue.current = scaleValue.current === 0 ? 1 : 0;
  Animated.spring(animationValue, {
    toValue: scaleValue.current,
    delay: 1,
    friction: 0.5,
    useNativeDriver: true,
  }).start();
};


return (
  <SafeAreaView style={styles(selectedTheme).mainConatainer}>
    <StatusBar backgroundColor={'rgba(0,0,0,0)'} barStyle={'dark-content'} />


    <Text style={styles(selectedTheme).heading}>{screensData.REGISTER}</Text>
```

```
<ScrollView style={styles(selectedTheme).flex}>
  <View style={styles(selectedTheme).boxContainer}>
    {register.map((item, index) => {
      return (
        <NEWButton
          key={index}
          onPress={() => {
            setdefaultitem(item.id), runAnimationOnClick();
          }}
          style={[
            styles(selectedTheme).box,
            {
              backgroundColor:
                defaultitem == item.id
                  ? selectedTheme.backgroundblueNblack
                  : selectedTheme.backgroundgray10Ngray70,
              borderColor:
                defaultitem == item.id
                  ? selectedTheme.borderColor1
                  : selectedTheme.borderColor1,
            },
          ]}>
          <Animated.View
            style={[
              styles(selectedTheme).checkedContainer,
              {
                transform:
                  defaultitem == item.id
                    ? [
                        {
                          translateX: animationValue.interpolate({
                            inputRange: [0, 1],
                            outputRange: [10, 15],
                          }),
                        },
                      ]
                    : [],
              },
            ]}>
            {defaultitem == item.id ? (
              <Image
                source={Icon.CHECKBOX_ON_DARK}
                style={styles(selectedTheme).icons}
              />
            ) : (
```

```
                    <View style={styles(selectedTheme).icons}></View>
                  )}
                </Animated.View>
                <Text
                  style={{
                    color: defaultitem == item.id ? COLORS.white :
COLORS.black,
                  }}>
                  {screensData.register.I_AM_A}{' '}
                </Text>
                <Text
                  style={[
                    styles(selectedTheme).labelText,
                    {
                      color:
                        defaultitem == item.id ? COLORS.white : COLORS.black,
                    },
                  ]}>
                  {item.label}
                </Text>
              </NEWButton>
            );
          })}
        </View>
        <View style={styles(selectedTheme).container}>
          <View style={styles(selectedTheme).input}>
            <Text style={styles(selectedTheme).formText}>{form.USER}</Text>
            <View style={styles(selectedTheme).inputContainer}>
              <TextInput
                style={styles(selectedTheme).inputText}
                onChangeText={EnterUserName}
              />
            </View>
            {!isvalidusername ? (
              <Text style={styles(selectedTheme).invalidText}>
                {form.validation.USERNAME}
              </Text>
            ) : null}
          </View>


          <View style={styles(selectedTheme).input}>
            <Text style={styles(selectedTheme).formText}>{form.EMAIL}</Text>
            <View style={styles(selectedTheme).inputContainer}>
              <TextInput
                keyboardType={'email-address'}
```

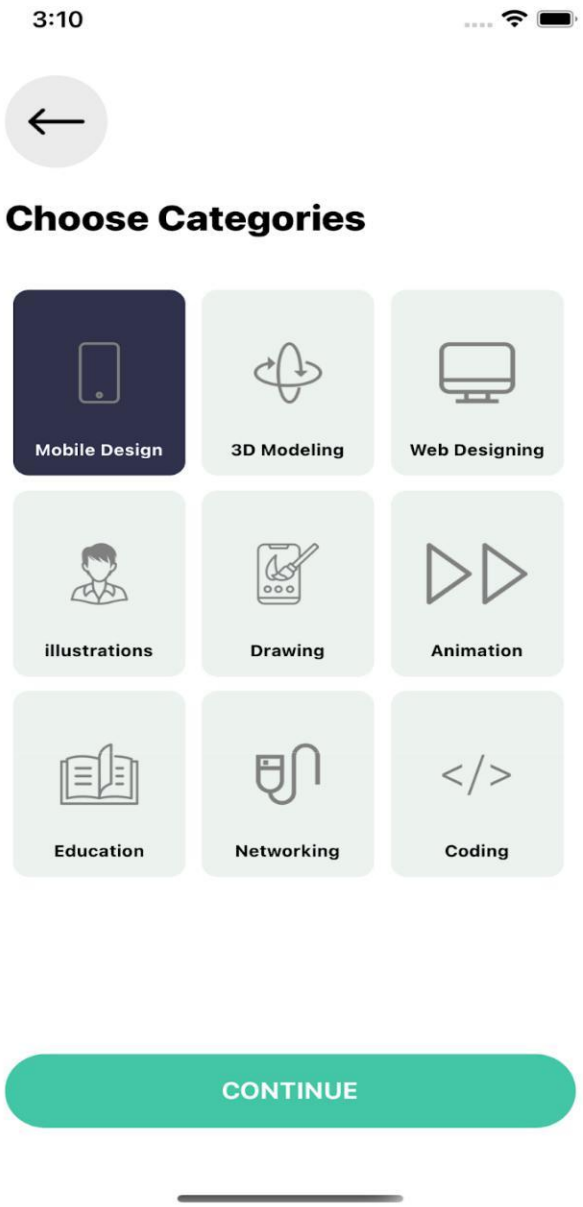
```
                onChangeText={Emailinpute}
                style={styles(selectedTheme).inputText}
              />
            </View>
            {!isvalidEmail ? (
              <Text style={styles(selectedTheme).invalidText}>
                {form.validation.EMAILVALIDATION}
              </Text>
            ) : null}
          </View>


          <View style={styles(selectedTheme).input}>
            <Text
style={styles(selectedTheme).formText}>{form.PASSSWORD}</Text>
            <View style={styles(selectedTheme).inputContainer}>
              <TextInput
                style={styles(selectedTheme).inputText}
                secureTextEntry={!visible}
                onChangeText={EnterPassword}
              />
              <TouchableOpacity
                style={styles(selectedTheme).eyeContainer}
                onPress={() => (visible ? setvisibe(false) : setvisibe(true))}>
                <Image
                  source={visible ? Icon.EYE : Icon.EYE_CLOSE}
                  style={styles(selectedTheme).eyeIcon}
                />
              </TouchableOpacity>
            </View>
            {!isvalidPassword ? (
              <Text style={styles(selectedTheme).invalidText}>
                {form.validation.PASSWORDVALIDATION}
              </Text>
            ) : null}
          </View>


          <TouchableOpacity
            style={styles(selectedTheme).button}
            onPress={submit}>
            <Text style={styles(selectedTheme).buttonText}>
              {buttons.CREATE_ACCOUNT}
            </Text>
          </TouchableOpacity>
```

```
<Text style={styles(selectedTheme).text}>
  {screensData.register.OR_SIGNUP_WITH}
</Text>


<View style={styles(selectedTheme).socialButtonContainer}>
  <TouchableOpacity
    style={styles(selectedTheme).socialButton}
    onPress={() => GoogleSignin()}>
    <Image source={Icon.GOOGLE} style={styles(selectedTheme).icon} />
    <Text style={styles(selectedTheme).socialButtonText}>
      {buttons.GOOGLE}
    </Text>
  </TouchableOpacity>
```

```
            <TouchableOpacity
              style={styles(selectedTheme).socialButton}
              onPress={() => Facebooklogin()}>
              <Image
                source={Icon.FACEBOOK}
                style={styles(selectedTheme).icon}
              />
              <Text style={styles(selectedTheme).socialButtonText}>
                {buttons.FACEBOOK}
              </Text>
            </TouchableOpacity>
          </View>


          <View style={styles(selectedTheme).buttonContainer}>
            <Text style={styles(selectedTheme).newUserText}>
              {screensData.register.ALREDY_A_USER}
            </Text>


            <TouchableOpacity
              onPress={() => navigation.navigate(RouteScreens.LOGINMODEL)}>
              <Text style={styles(selectedTheme).loginText}>
                {screensData.register.LOGIN}
              </Text>
            </TouchableOpacity>
          </View>
        </View>
      </ScrollView>
    </SafeAreaView>
  );
};
export default RegisterScreen;
```

11.4 <u>Category Screen</u> :

- Users can choose categories and navigate according to selected category

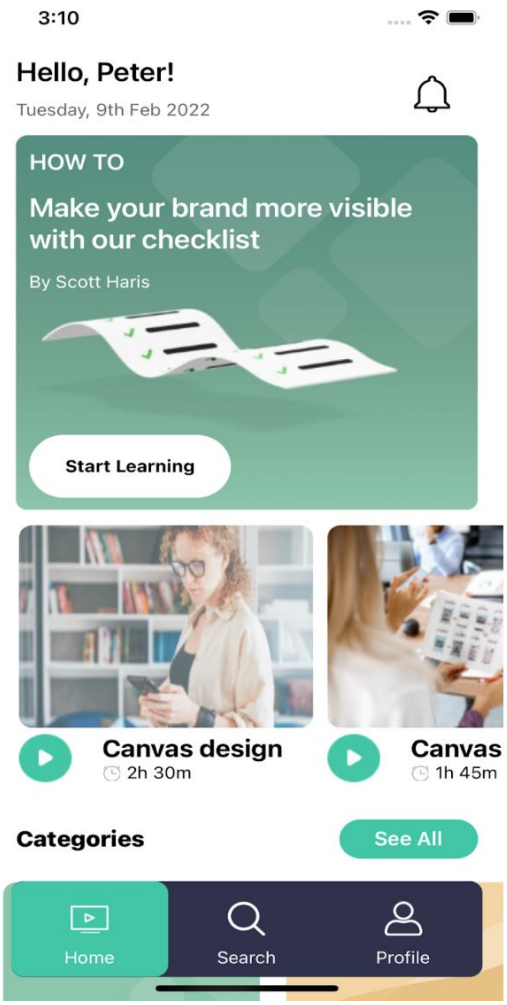


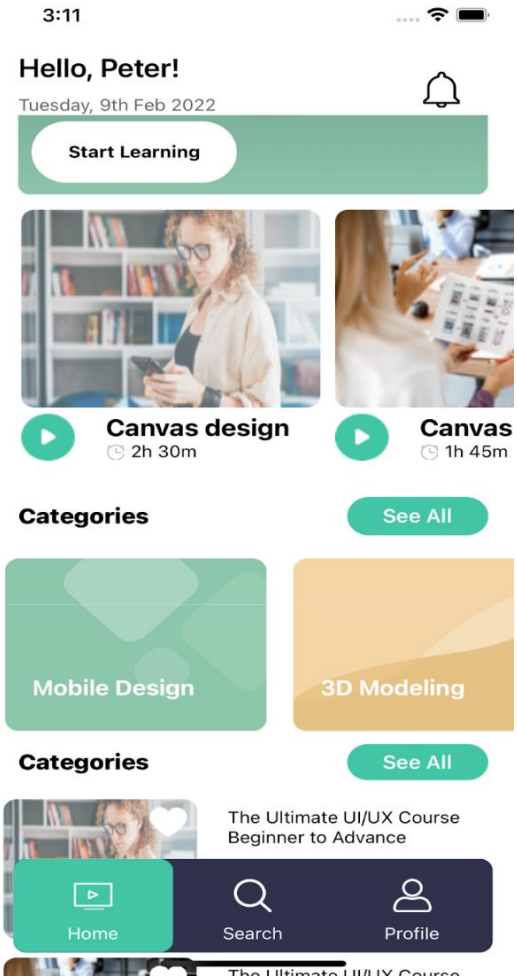
```
import
React,
{memo}
from
'react';
           import {
             FlatList,
             Image,
             SafeAreaView,
             Text,
             TouchableOpacity,
             View,
           } from 'react-native';
           import {selectedTheme} from '../../config/Themes';
           import {Icon, Constant, screensData, buttons, RouteScreens} from '../../config';
           import Renderitem from './Renderitem';
           import styles from './style';


           interface CategoryProps {
             navigation: any;
             defaultitem: string;
             setdefaultitem: React.Dispatch<React.SetStateAction<string>>;
             defaulticon: undefined;
             setdefaulticon: React.Dispatch<React.SetStateAction<undefined>>;
           }


           const CategoryScreen = (props: CategoryProps) => {
             const {navigation, defaultitem, setdefaultitem, setdefaulticon, defaulticon} =
               props;


             return (
               <SafeAreaView style={styles(selectedTheme).mainConatiner}>
                 <View style={styles(selectedTheme).container}>
                   <TouchableOpacity
                     style={styles(selectedTheme).leftButton}
                     onPress={() => navigation.goBack()}>
                     <Image
                       source={Icon.BACK}
                       style={styles(selectedTheme).headerLeftIcon}
                     />
                   </TouchableOpacity>
                   <Text style={styles(selectedTheme).text}>
                     {screensData.home.CATEGORIES}
```

```
        </Text>


        <View style={styles(selectedTheme).flatlistContainer}>
          <FlatList
            data={Constant.categories}
            extraData={Constant.categories}
            renderItem={({item, index}) => (
              <Renderitem
                item={item}
                index={index}
                defaultitem={defaultitem}
                setdefaultitem={setdefaultitem}
                setdefaulticon={setdefaulticon}
              />
            )}
            numColumns={3}
            keyExtractor={(_, index) => index.toString()}
          />


          <TouchableOpacity
            style={styles(selectedTheme).button}
            onPress={() =>
              navigation.navigate(RouteScreens.COURSELISTINGMODEL, {
                Header: defaultitem,
                Icon: defaulticon,
              })
            }>
            <Text style={styles(selectedTheme).buttonText}>
              {buttons.CONTINUE}{' '}
            </Text>
          </TouchableOpacity>
        </View>
      </View>
    </SafeAreaView>
  );
};


export default memo(CategoryScreen);
```

11.5 Home Screen :

- This is the Home screen and there are several categories like(Popular courses and categories)
- User can navigate to any screen according to selected category
- This also contains a bottom tab navigation which opens Home , Search , Profile Screen
- At the right top right corner there is notification icon which opens Notification Screen





```
import
React,
{memo}
from
'react';
           import {
             View,
             Text,
             TouchableOpacity,
             Image,
             ScrollView,
             ImageBackground,
             Animated,
             SafeAreaView,
           } from 'react-native';
           import {selectedTheme} from '../../config/Themes';
           import {buttons, Icon, Images, RouteScreens, screensData} from '../../config';
           import RenderItem from './RenderItem';
           import styles from './style';


           interface Homepageprops {
             navigation: any;
             courselist1: {
               id: number;
               title: string;
               duration: string;
               thumbnail: any;
             }[];
             category: {
               id: number;
               title: string;
               thumbnail: any;
               icon: any;
             }[];
             courselist2: {
               id: number;
               title: string;
               clsss_level: string;
               creted_on: string;
               duration: number;
               instructor: string;
               ratings: number;
               price: number;
```

```
    is_favourite: boolean;
    thumbnail: any;
  }[];
  setseeall: any;
  isfavourite: any;
  setisfavourite: any;
}


const HomeScreen = (props: Homepageprops) => {
  const {
    navigation,
    courselist1,
    courselist2,
    category,
    setseeall,
    isfavourite,
    setisfavourite,
  } = props;


  const animation = new Animated.Value(0);


  const runAnimationOnClick = () => {
    Animated.loop(
      Animated.sequence([
        Animated.timing(animation, {
          toValue: -1,
          duration: 100,
          useNativeDriver: true,
        }),
        Animated.timing(animation, {
          toValue: 1,
          duration: 100,
          useNativeDriver: true,
        }),
        Animated.timing(animation, {
          toValue: 0,
          duration: 100,
          useNativeDriver: true,
        }),
      ]),
      {
        iterations: 4,
      },
```

```
  ).start();
};


const rotateanimation = animation.interpolate({
  inputRange: [-1, 1, 2],
  outputRange: ['-20deg', '20deg', '0deg'],
});
return (
  <SafeAreaView style={styles(selectedTheme).container}>
    <View style={styles(selectedTheme).mainSubContainer}>
      <View style={styles(selectedTheme).headercontainer}>
        <View style={styles(selectedTheme).nameAndDate}>
          <Text style={styles(selectedTheme).text}>
            {screensData.home.NAME}
          </Text>
          <Text style={styles(selectedTheme).subText}>
            {screensData.home.DATE}
          </Text>
        </View>
        <Animated.View
          onTouchStart={runAnimationOnClick}
          style={{
            transform: [{rotate: rotateanimation}],
          }}>
          <TouchableOpacity
            onPress={() => navigation.navigate(RouteScreens.NOTIFICATIONTAB)}>
            <Image
              source={Icon.NOTIFICATION}
              style={styles(selectedTheme).notifiactionicon}
            />
          </TouchableOpacity>
        </Animated.View>
      </View>
      <ScrollView
        nestedScrollEnabled={true}
        showsVerticalScrollIndicator={false}
        style={styles(selectedTheme).mainScrollView}>
        <ImageBackground
          source={Images.FEATURED_BG_IMAGE}
          imageStyle={styles(selectedTheme).backgroundImage}
          style={styles(selectedTheme).subContainer}>
          <Text style={styles(selectedTheme).howToText}>
            {screensData.home.HOWTO}
          </Text>
          <Text style={styles(selectedTheme).subtextdata}>
```

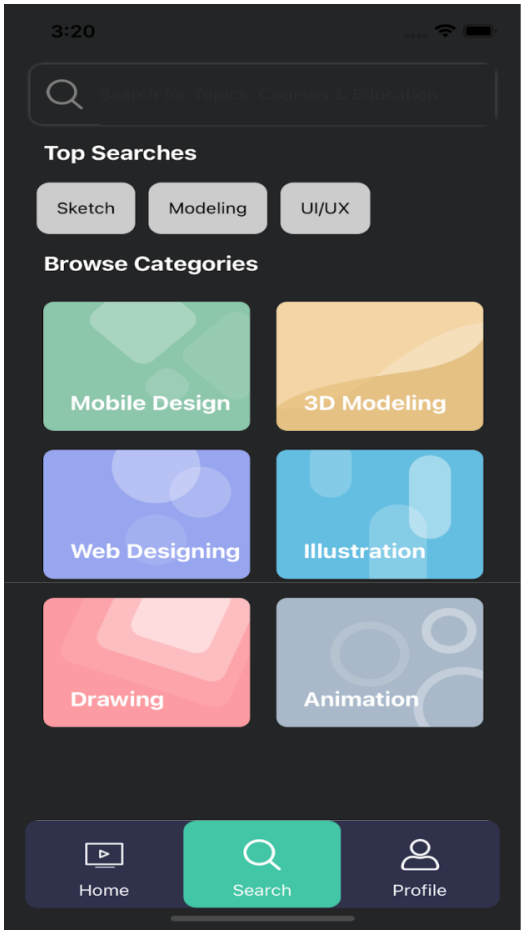
```
    {screensData.home.SUBPARAGRAPH}
  </Text>
  <Text style={styles(selectedTheme).author}>
    {screensData.home.AUTHORNAME}
  </Text>
  <Image
    source={Images.START_LEARNING}
    style={styles(selectedTheme).bannerImage}
  />
  <TouchableOpacity
    style={styles(selectedTheme).learnButton}
    onPress={() => navigation.navigate(RouteScreens.CATEGORYMODEL)}>
    <Text style={styles(selectedTheme).startLearningText}>
      {screensData.home.STARTLERNING}
    </Text>
  </TouchableOpacity>
</ImageBackground>
<ScrollView
  horizontal
  showsHorizontalScrollIndicator={false}
  style={styles(selectedTheme).thirdContainer}>
  {courselist1.map((item, index) => {
    return (
      <TouchableOpacity
        key={index}
        onPress={() =>
          navigation.navigate(RouteScreens.COURSEMODEL, {
            Title: item.title,
          })
        }>
        <Image
          source={item.thumbnail}
          style={styles(selectedTheme).thumbnail}
        />
        <View style={styles(selectedTheme).row}>
          <Image
            source={Icon.PLAY_1}
            style={styles(selectedTheme).playicon}
          />
          <View style={styles(selectedTheme).textAndTimeContainer}>
            <Text
              style={styles(selectedTheme).thirdContainerTextTitle}>
              {item.title}
            </Text>
            <View style={styles(selectedTheme).row}>
              <Image
```

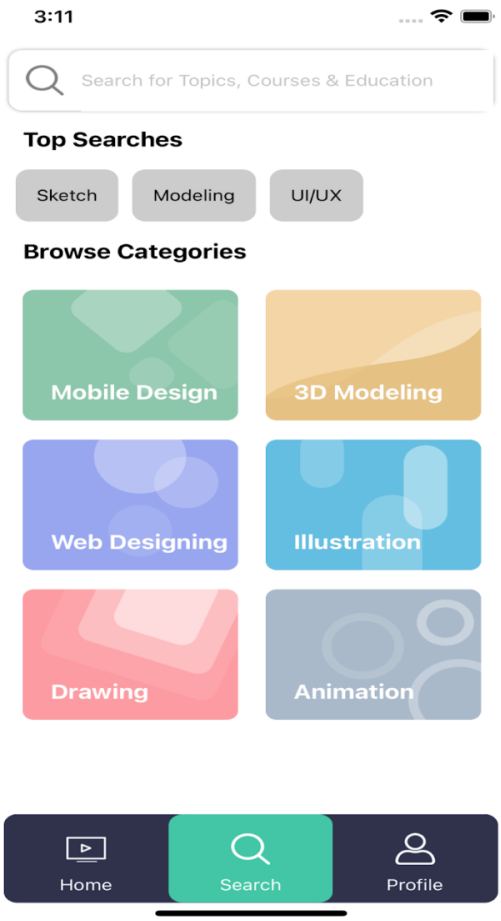
```
                source={Icon.TIME}
                style={styles(selectedTheme).timeIcon}
              />
              <Text style={styles(selectedTheme).time}>
                {' '}
                {item.duration}
              </Text>
            </View>
          </View>
        </View>
      </TouchableOpacity>
    );
  })}
</ScrollView>


<View style={styles(selectedTheme).itemContainer}>
  <Text style={styles(selectedTheme).thirdContainerTextTitle}>
    {screensData.home.CATEGORIES}
  </Text>
  <TouchableOpacity
    style={styles(selectedTheme).seeAllButton}
    onPress={() => navigation.navigate(RouteScreens.CATEGORYMODEL)}>
    <Text style={styles(selectedTheme).seeAllText}>
      {buttons.SEEALL}
    </Text>
  </TouchableOpacity>
</View>


<ScrollView horizontal showsHorizontalScrollIndicator={false}>
  {category.map((item, index) => {
    return (
      <TouchableOpacity
        key={index}
        onPress={() =>
          navigation.navigate(RouteScreens.COURSELISTINGMODEL, {
            Header: item.title,
            Icon: item.icon,
          })
        }
        style={styles(selectedTheme).item}>
        <ImageBackground
          source={item.thumbnail}
          imageStyle={styles(selectedTheme).itemIcons}
          style={styles(selectedTheme).icons}>
```

```
                    <Text style={[styles(selectedTheme).itemText]}>
                      {item.title}
                    </Text>
                  </ImageBackground>
                </TouchableOpacity>
              );
            })}
          </ScrollView>


          <View style={styles(selectedTheme).itemContainer}>
            <Text style={styles(selectedTheme).thirdContainerTextTitle}>
              {screensData.home.POPULARCOURSES}
            </Text>
            <TouchableOpacity
              style={styles(selectedTheme).seeAllButton}
              onPress={() => setseeall(true)}>
              <Text style={styles(selectedTheme).seeAllText}>
                {buttons.SEEALL}
              </Text>
            </TouchableOpacity>
          </View>
          <ScrollView>
            {courselist2.map((item, index) => {
              return (
                <RenderItem
                  navigation={navigation}
                  key={index}
                  item={item}
                  index={index}
                  isfavourite={isfavourite}
                  setisfavourite={setisfavourite}
                />
              );
            })}
          </ScrollView>
        </ScrollView>
      </View>
    </SafeAreaView>
  );
};
export default memo(HomeScreen);
```

11.6 Search Screen:

- This is the Search Screen which contains a search input field and Top searches suggestions and also categories
- User can navigate to a new screen according to the selected category





(Dark Theme) (Light Theme)

```
import
React
from
'react';
           import {
             FlatList,
             Image,
             SafeAreaView,
             ScrollView,
             Text,
             TextInput,
             TouchableOpacity,
             View,
           } from 'react-native';
           import {selectedTheme} from '../../config/Themes';
           import {Icon, screensData, DummyData} from '../../config';
           import RenderItem from '../Home/RenderItem';
           import Renderitem from './Renderitem';
           import styles from './style';


           interface SearchScreenProps {
             navigation: any;
             visible: any;
             setvisibe: any;
             defaultitem: number;
             setdefaultitem: React.Dispatch<React.SetStateAction<number>>;
             SearchTexthandler: any;
             isSearch: boolean;
             searchText: string;
             setisSearch: any;
             resetsearch: any;
             flatlistdata: {
               id: number;
               title: string;
               thumbnail: any;
               icon: any;
             }[];
             DATA: any;
             isfavourite: any;
             setisfavourite: any;
           }


           const SearchScreen = (props: SearchScreenProps) => {
             const {
```

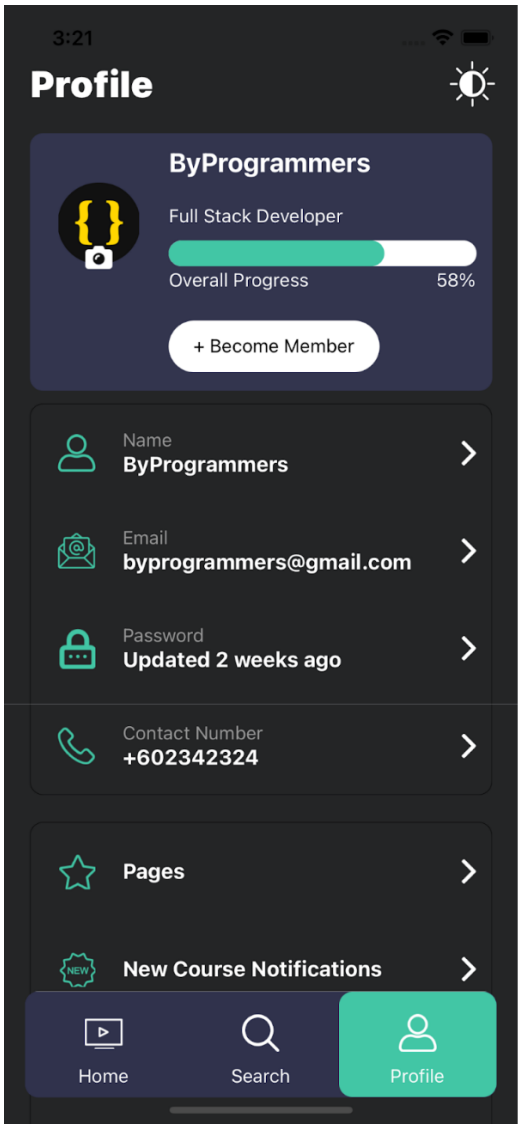
```
  navigation,
  visible,
  setvisibe,
  defaultitem,
  setdefaultitem,
  SearchTexthandler,
  isSearch,
  searchText,
  setisSearch,
  resetsearch,
  flatlistdata,
  DATA,
  isfavourite,
  setisfavourite,
} = props;


const handleEmpty = () => {
  return <Text> {screensData.search.NO_DATA_PRESENT}</Text>;
};


return (
  <SafeAreaView style={styles(selectedTheme).mainContainer}>
    <View style={styles(selectedTheme).container}>
      <View style={styles(selectedTheme).searchBar}>
        <TouchableOpacity>
          <Image
            source={Icon.SEARCH}
            style={styles(selectedTheme).headerLeftIcon}
          />
        </TouchableOpacity>


        <TextInput
          style={[{width: !isSearch ? '100%' : '85%'}]}
          onChangeText={SearchTexthandler}
          placeholder={screensData.search.PLACEHOLDER}
          value={searchText}
          placeholderTextColor={selectedTheme.textgray8Ngray4}
        />
        {isSearch ? (
          <TouchableOpacity
            onPress={() => {
              setisSearch(false), resetsearch(Text);
            }}>
```

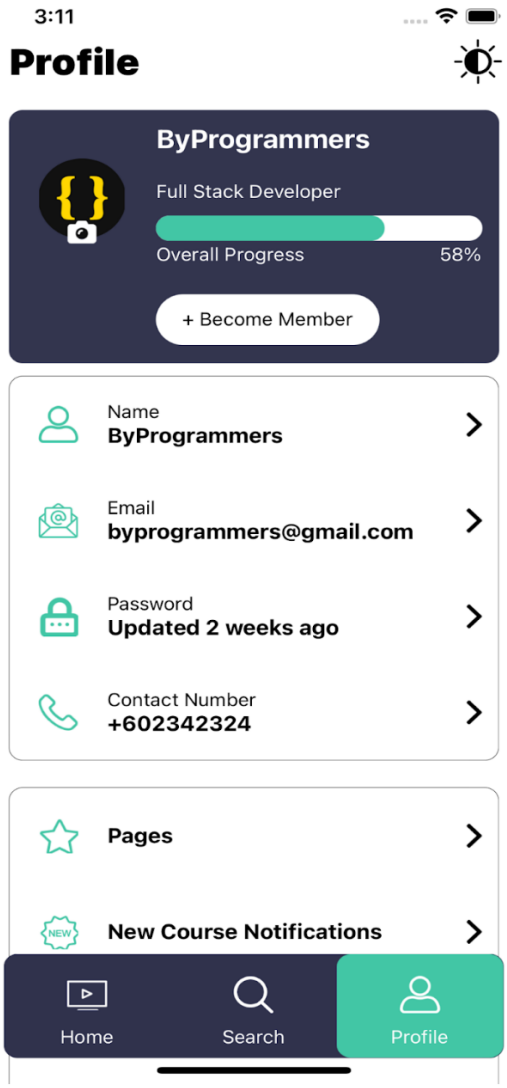
```
      <Image
        source={Icon.CROSS}
        style={[styles(selectedTheme).searchBarImage]}
      />
    </TouchableOpacity>
  ) : null}
</View>


{!isSearch ? (
  <View>
    <Text style={styles(selectedTheme).text}>
      {screensData.search.TOPSEARCHES}
    </Text>
    <ScrollView horizontal showsHorizontalScrollIndicator={false}>
      <View style={styles(selectedTheme).topSearch}>
        {DummyData.topSearches.map((item, index) => {
          return (
            <View key={index} style={{flex: 1}}>
              <TouchableOpacity
                style={styles(selectedTheme).topSearchContainer}
                onPress={() => SearchTexthandler(item.label)}>
                <Text
                  key={index}
                  style={styles(selectedTheme).searchList}>
                  {item.label}
                </Text>
              </TouchableOpacity>
            </View>
          );
        })}
      </View>
    </ScrollView>


    <FlatList
      key={1}
      data={flatlistdata}
      extraData={flatlistdata}
      showsVerticalScrollIndicator={false}
      renderItem={({item, index}) => (
        <Renderitem item={item} index={index} navigation={navigation} />
      )}
      numColumns={2}
      keyExtractor={(_, index) => index.toString()}
      ListHeaderComponent={
```

```
                <Text style={styles(selectedTheme).text}>
                  {screensData.search.BROWSECATEGORIES}
                </Text>
              }
            />
          </View>
        ) : (
          <FlatList
            data={DATA}
            showsVerticalScrollIndicator={false}
            ListEmptyComponent={handleEmpty}
            keyExtractor={(item, index) => item + index.toString()}
            renderItem={({item}) => (
              <RenderItem
                item={item}
                isfavourite={isfavourite}
                setisfavourite={setisfavourite}
                navigation={navigation}
              />
            )}
          />
        )}
      </View>
    </SafeAreaView>
  );
};


export default SearchScreen;
```

11.7 Profile Screen :

- This is the Profile screen which consists of a theme change Icon which change theme into light to dark and vice-versa
- This Screen shows the details of user
- It also consists of a change profile functionality
- After Pressing the Become member button it opens a Become membership Screen





```
import
React,
{memo,
useState}
from
'react';
            import {
              Image,
              Modal,
              SafeAreaView,
              ScrollView,
              Switch,
              Text,
              TouchableOpacity,
              TouchableWithoutFeedback,
              View,
            } from 'react-native';
            import {ProgressBar} from '../../components';
            import {COLORS, selectedTheme, changeTheme} from '../../config/Themes';
            import {buttons, Icon, screensData, DummyData} from '../../config';
            import styles from './style';


            interface profileprops {
              navigation: any;
              visible: any;
              setvisibe: any;
              setrender: any;
              render: any;
              modalVisible: any;
              setModalVisible: any;
              launchLibrary: any;
              openCamara: any;
              imageUriGallary: any;
              dispachcall: any;
            }
            const ProfileScreen = (props: profileprops) => {
              const {
                navigation,
                visible,
                setvisibe,
                setrender,
                render,
                modalVisible,
                setModalVisible,
                openCamara,
```

```
  launchLibrary,
  imageUriGallary,
  dispachcall,
} = props;


return (
  <SafeAreaView style={styles(selectedTheme).mainContainer}>
    <View style={styles(selectedTheme).mainSubContainer}>
      <View style={styles(selectedTheme).headerContainer}>
        <Text style={styles(selectedTheme).headerText}>
          {screensData.Profile.NAME}
        </Text>
        <TouchableOpacity
          onPress={() => {
            setrender(!render), changeTheme(), dispachcall();
          }}>
          <Image
            source={Icon.SUN}
            style={styles(selectedTheme).headerIcons}
          />
        </TouchableOpacity>
      </View>
      <ScrollView
        showsVerticalScrollIndicator={false}
        style={{marginTop: 20}}>
        <View style={styles(selectedTheme).subContainer}>
          <View style={{flexDirection: 'row'}}>
            <TouchableOpacity
              style={styles(selectedTheme).imageContainer}
              onPress={() => setModalVisible(true)}>
              <Image
                source={{uri: imageUriGallary}}
                style={styles(selectedTheme).profileIcon}
              />
              <View style={styles(selectedTheme).imageContainerView}>
                <View style={styles(selectedTheme).imageViewContainer}>
                  <Image
                    source={Icon.CAMERA}
                    style={styles(selectedTheme).cameraIcon}
                  />
                </View>
              </View>
            </TouchableOpacity>
            <View style={styles(selectedTheme).userHeaderDataContainer}>
              <Text style={styles(selectedTheme).username}>
```

```
            {screensData.instructor.NAME}{' '}
          </Text>
          <Text style={styles(selectedTheme).body}>
            {screensData.instructor.DES}
          </Text>
          <ProgressBar
            containerstyle={styles(selectedTheme).progressBar}
            progress="90%"
          />
          <View style={styles(selectedTheme).progressTextContainer}>
            <Text style={styles(selectedTheme).progressText}>
              {screensData.Profile.OVERALLPROGRESS}
            </Text>
            <Text style={styles(selectedTheme).progressText}>
              {screensData.Profile.PERCENTAGE}
            </Text>
          </View>
        </View>
      </View>
      <View>
        <TouchableOpacity
          style={styles(selectedTheme).learnButton}
          onPress={() => navigation.navigate('MembershipModel')}>
          <Text style={styles(selectedTheme).learnButtonText}>
            {buttons.BECOMEMEMBER}
          </Text>
        </TouchableOpacity>
      </View>
    </View>
    <View style={styles(selectedTheme).userDataContainer}>
      {DummyData.UserData.map((item: any, index: any) => {
        return (
          <View key={index}>
            <View style={styles(selectedTheme).userContainer}>
              <View style={styles(selectedTheme).subUserContainer}>
                <Image
                  source={item.icon}
                  style={styles(selectedTheme).icon}
                />
                <View style={styles(selectedTheme).textContainer}>
                  <Text style={styles(selectedTheme).label}>
                    {item.label}
                  </Text>
                  <Text style={styles(selectedTheme).value}>
                    {item.Value}
                  </Text>
```

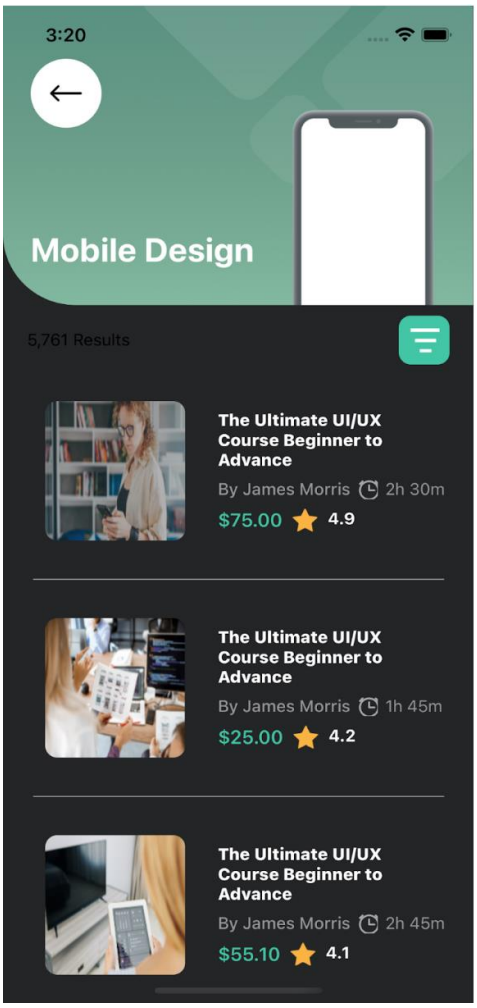
```
            </View>
          </View>
          <TouchableOpacity
            style={styles(selectedTheme).rightImageContainer}>
            <Image
              source={Icon.RIGHT_ARROW}
              style={styles(selectedTheme).rightArrowIcon}
            />
          </TouchableOpacity>
        </View>
        {index != 3 ? (
          <View style={styles(selectedTheme).itemSeperator} />
        ) : null}
      </View>
    );
  })}
</View>
<View style={styles(selectedTheme).userDataContainer}>
  {DummyData.UserData2.map((item: any, index: any) => {
    const [isEnabled, setIsEnabled] = useState(false);
    const toggleSwitch = () =>
      setIsEnabled(previousState => !previousState);
    return (
      <View key={index}>
        <View style={styles(selectedTheme).userContainer}>
          <View style={styles(selectedTheme).subUserContainer}>
            <Image
              source={item.icon}
              style={styles(selectedTheme).icon}
            />
            <View style={styles(selectedTheme).textContainer}>
              <Text style={styles(selectedTheme).lableData}>
                {item.label}
              </Text>
            </View>
          </View>
          <TouchableOpacity
            style={styles(selectedTheme).rightImageContainer}>
            {index == 0 ? (
              <Image
                source={Icon.RIGHT_ARROW}
                style={styles(selectedTheme).rightArrowIcon}
              />
            ) : (
              <Switch
```

```
                        trackColor={{false: COLORS.gray20, true:
COLORS.additionalColor13}}
                        thumbColor={isEnabled ? COLORS.primary :
COLORS.gray40}
                        ios_backgroundColor="#3e3e3e"
                        onValueChange={toggleSwitch}
                        value={isEnabled}
                      />
                    )}
                  </TouchableOpacity>
                </View>
                {index != 2 ? (
                  <View style={styles(selectedTheme).itemSeperator} />
                ) : null}
              </View>
            );
          })}
        </View>
      </ScrollView>
      <Modal
        animationType="slide"
        transparent={true}
        visible={modalVisible}
        onRequestClose={() => {
          setModalVisible(!modalVisible);
        }}>
        <TouchableWithoutFeedback
          onPress={() => setModalVisible(!modalVisible)}>
          <View style={styles(selectedTheme).drawerContainer} />
        </TouchableWithoutFeedback>


        <View style={styles(selectedTheme).modalView}>
          <TouchableOpacity onPress={() => setModalVisible(!modalVisible)}>
            <Image
              source={Icon.CROSS}
              style={styles(selectedTheme).modalIcon}
            />
            <Text style={styles(selectedTheme).modalText}>
              {buttons.CANCEL}
            </Text>
          </TouchableOpacity>


          <TouchableOpacity onPress={() => openCamara()}>
            <Image
```

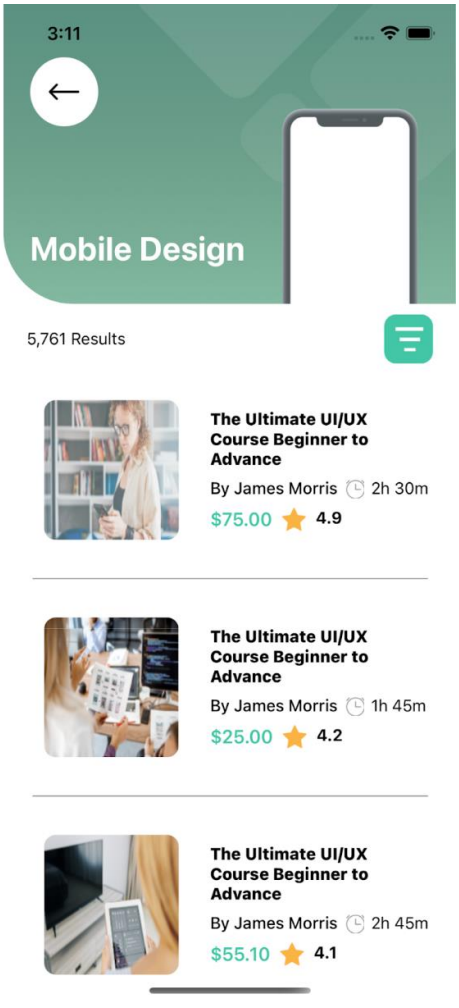
```
                source={Icon.CAMERA}
                style={styles(selectedTheme).modalIcon}
              />
              <Text style={styles(selectedTheme).modalText}>
                {buttons.CAMERA}
              </Text>
            </TouchableOpacity>


            <TouchableOpacity onPress={() => launchLibrary()}>
              <Image
                source={Icon.GALLERY}
                style={styles(selectedTheme).modalIcon}
              />
              <Text style={styles(selectedTheme).modalText}>
                {buttons.GALLERY}
              </Text>
            </TouchableOpacity>
          </View>
        </Modal>
      </View>
    </SafeAreaView>
  );
};


export default memo(ProfileScreen);
```

11.8 Course Listing Screen :

- This Screen contains various courses
- It contains a filter button that opens the filter modal that's help the user to filter his choice of courses
- After Clicking the particular course it opens a course chapter screen



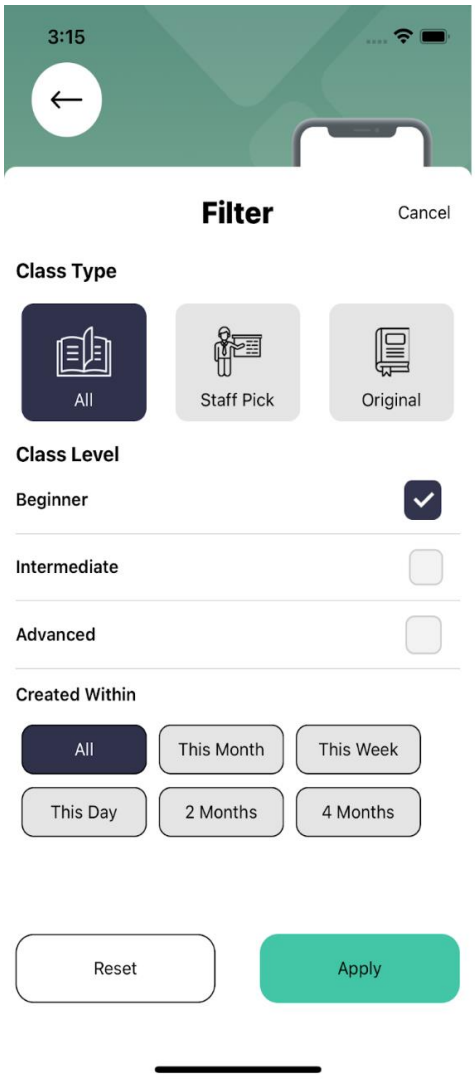
(Dark Theme)

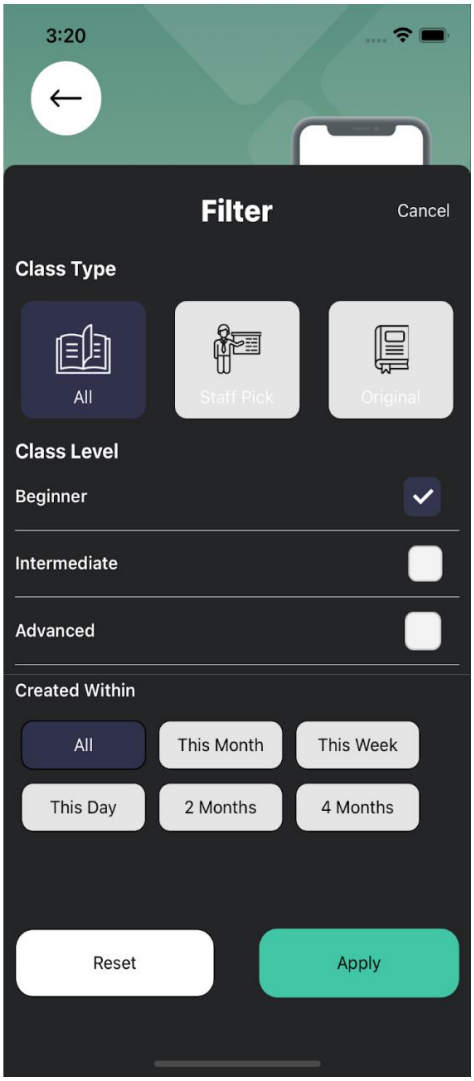


(Light Theme)

11.9 Filter Modal:

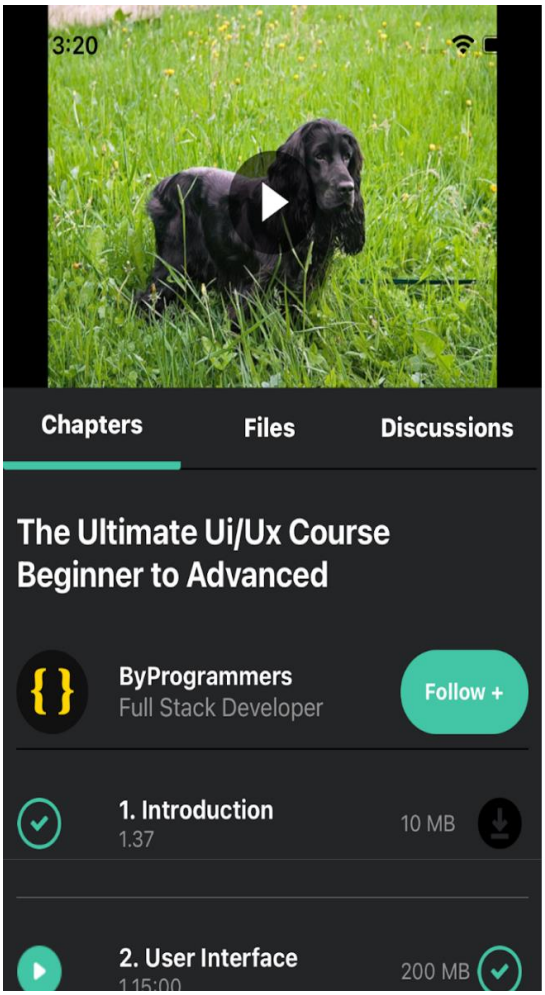
- This is a Filter Modal Screen that filters data according to user selected type
- In this screen there are some option like class type, class level witch help the user to filter the courses as according his choice

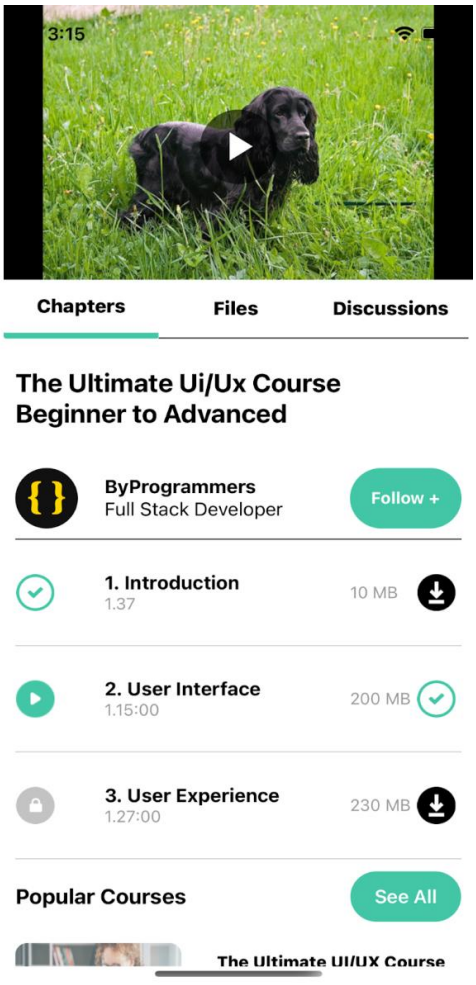




11.10 Course Chapter Screen :

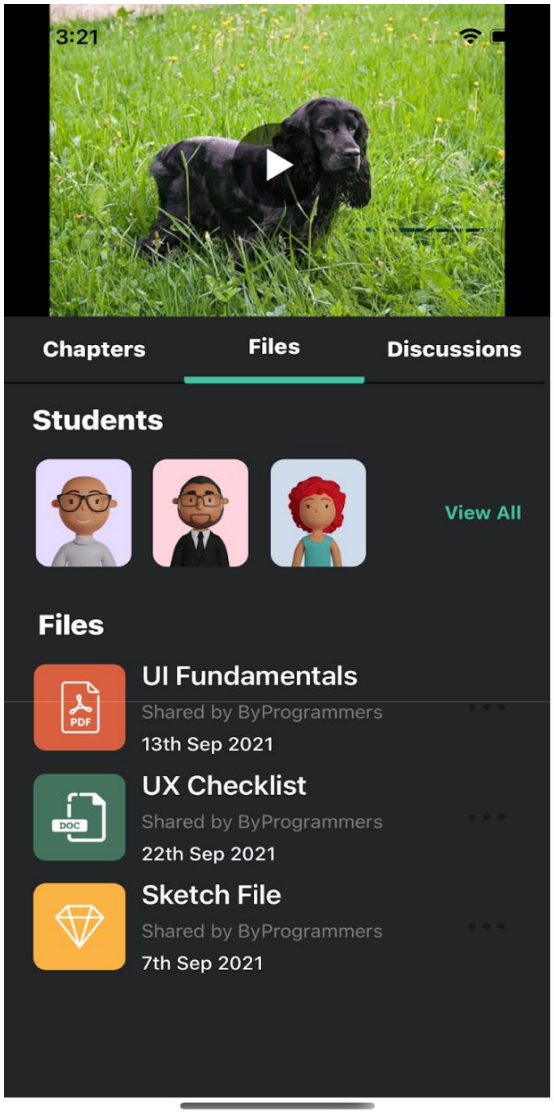
- This Screen shows the chapters of a particular course
- User can watch a Videos of Courses
- Users can also go to Instructor profile by clicking on follow button
- It also suggests the popular courses related to course

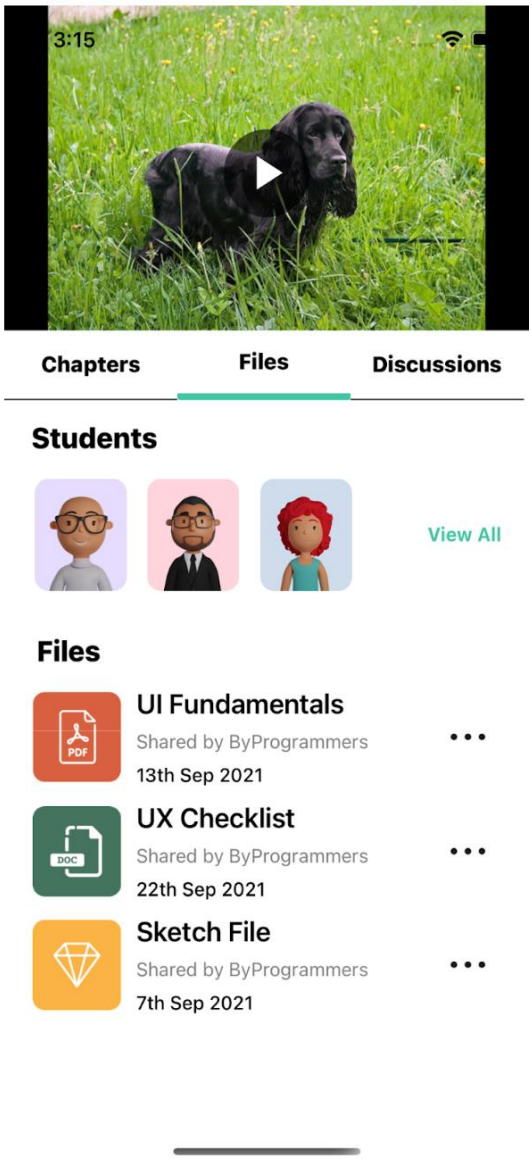




11.11 Chapter Files Discussion :

- In this screen shows No. of Students
- It also contains a course learning kit like(UI Fundamentals ,UX Checklist)





```
import
React,
{memo}
from
'react';
           import {
             Text,
             View,
             Image,
             TouchableOpacity,
             ScrollView,
             SafeAreaView,
           } from 'react-native';
           import {selectedTheme} from '../../config/Themes';
           import {buttons, Icon, Images, RouteScreens, screensData} from '../../config';
           import {courseDetailsModel} from '../../models';
           import RenderItem from '../Home/RenderItem';
           import styles from './style';


           interface Corselistingprops {
             navigation: any;
             DATA: courseDetailsModel;
             flatlistdata: {
               id: number;
               title: string;
               clsss_level: string;
               creted_on: string;
               duration: number;
               instructor: string;
               ratings: number;
               price: number;
               is_favourite: boolean;
               thumbnail: any;
             }[];
             Title: string;
             isfavourite: any;
             setisfavourite: any;
           }


           const CourseChapterScreen = (props: Corselistingprops) => {
             const {DATA, navigation, flatlistdata, Title, isfavourite, setisfavourite} =
               props;
             return (
               <SafeAreaView style={styles(selectedTheme).mainContainer}>
```

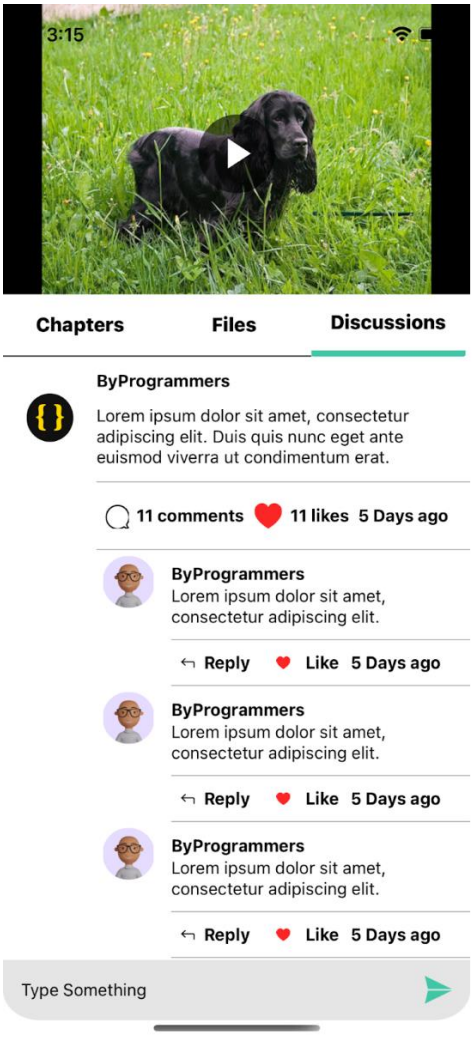
```
<View style={styles(selectedTheme).conatiner}>
  <View style={styles(selectedTheme).headerContainer}>
    <Text style={styles(selectedTheme).userTitle}>{Title}</Text>
    <View style={styles(selectedTheme).studentData}>
      <Text style={styles(selectedTheme).smallText}>
        {DATA.number_of_students}
      </Text>
      <Image source={Icon.TIME} style={styles(selectedTheme).timeIcon} />
      <Text style={styles(selectedTheme).smallText}>{DATA.duration}</Text>
    </View>
  </View>
  <View style={styles(selectedTheme).profileContainer}>
    <View style={styles(selectedTheme).subProfileContainer}>
      <Image
        source={Images.PROFILE}
        style={styles(selectedTheme).profileIcon}
      />
      <View>
        <Text style={styles(selectedTheme).userNameText}>
          {DATA.instructor.name}
        </Text>
        <Text style={styles(selectedTheme).programmerText}>
          {DATA.instructor.title}
        </Text>
      </View>
    </View>
    <TouchableOpacity
      style={styles(selectedTheme).followButton}
      onPress={() =>
        navigation.navigate(RouteScreens.INSTRUCTORPROFILEMODEL)
      }>
      <Text style={styles(selectedTheme).followText}>
        {buttons.FOLLOW} +
      </Text>
    </TouchableOpacity>
  </View>
</View>
<View style={styles(selectedTheme).seperator} />
<View style={styles(selectedTheme).conatiner}>
  <View>
    {DATA.videos.map((item, index) => {
      return (
        <View key={index} style={styles(selectedTheme).videoContainer}>
          <View style={styles(selectedTheme).subVideoContainer}>
            <View style={styles(selectedTheme).playButton}>
              <Image
```

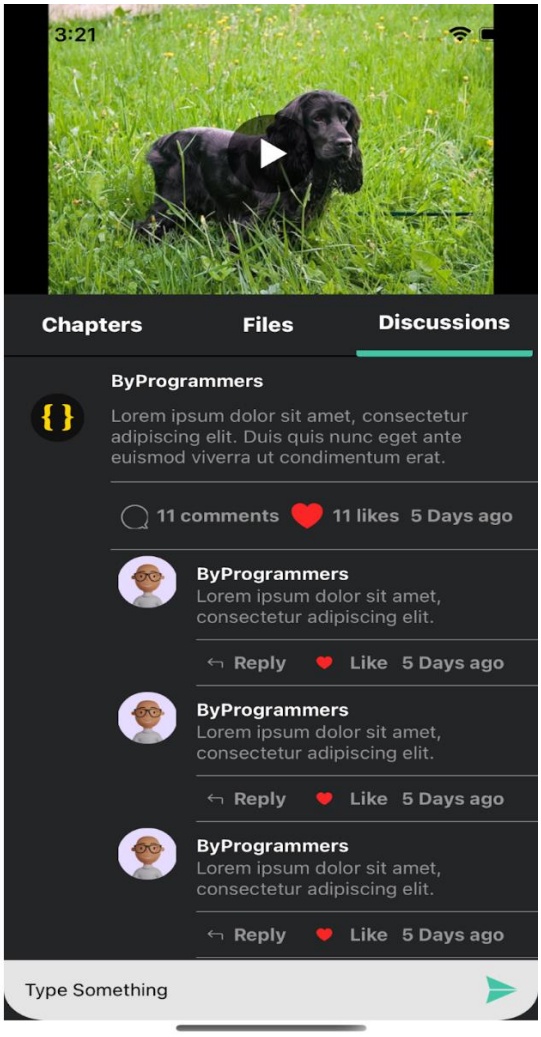
```
              source={Icon.PLAY}
              style={styles(selectedTheme).videoIcon}
            />
          </View>
          <View>
            <Text style={styles(selectedTheme).title}>
              {item.title}
            </Text>
            <Text style={styles(selectedTheme).timeText}>
              {item.duration}
            </Text>
          </View>
        </View>
        <View style={styles(selectedTheme).buttonContainer}>
          <Text style={styles(selectedTheme).timeText}>
            {item.size}
          </Text>
          <TouchableOpacity>
            <Image
              source={Icon.DOWNLOAD}
              style={styles(selectedTheme).downloadIcon}
            />
          </TouchableOpacity>
        </View>
      </View>
    );
  })}
</View>


<View style={styles(selectedTheme).itemContainer}>
  <Text style={styles(selectedTheme).popularCoursesText}>
    {screensData.home.POPULARCOURSES}
  </Text>
  <TouchableOpacity style={styles(selectedTheme).seeAllButton}>
    <Text style={styles(selectedTheme).seeAllText}>
      {buttons.SEEALL}
    </Text>
  </TouchableOpacity>
</View>


<ScrollView>
  {flatlistdata.map((item, index) => {
    return (
      <RenderItem
```

```
                key={index}
                navigation={navigation}
                item={item}
                index={index}
                isfavourite={isfavourite}
                setisfavourite={setisfavourite}
              />
            );
          })}
        </ScrollView>
      </View>
    </SafeAreaView>
  );
};
export default memo(CourseChapterScreen);
```

11.12 Course Discussion Screen :

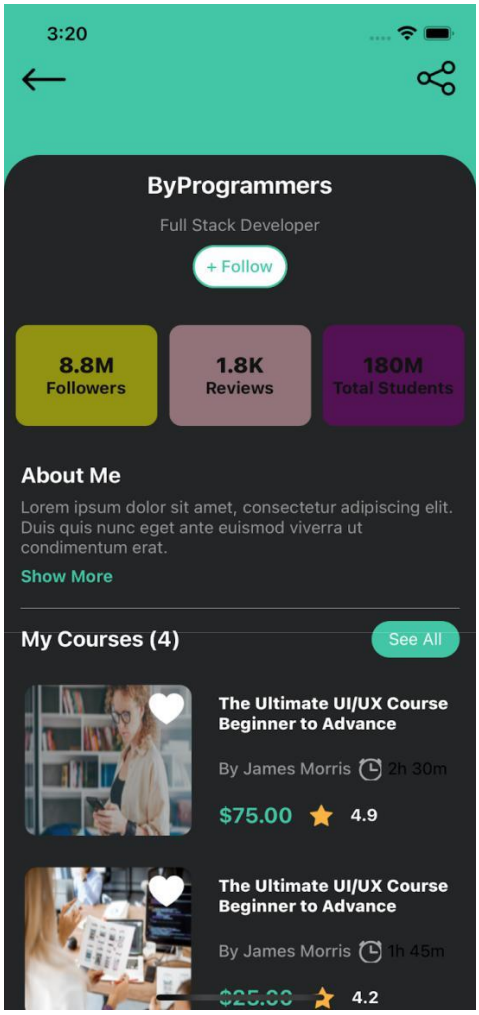
- This Screen contains the discussion section like comments and doubt related to courses
- Users can read and write new comments

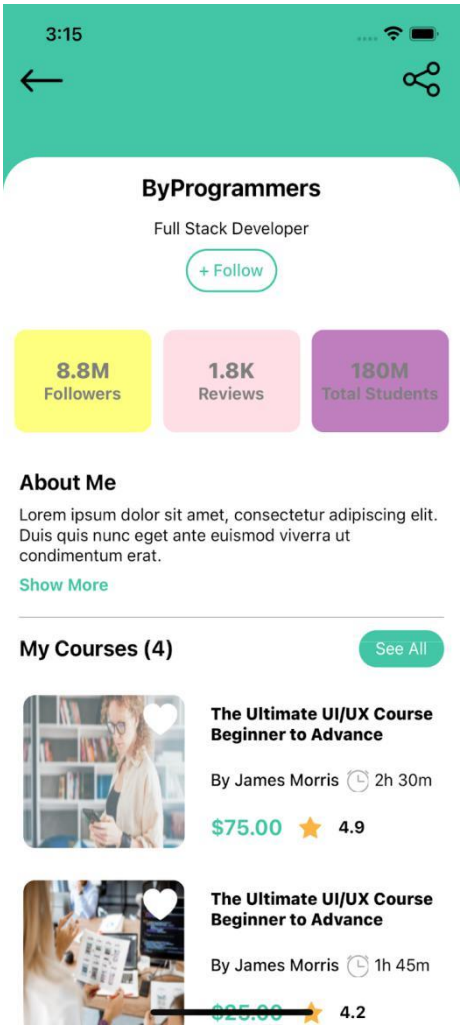


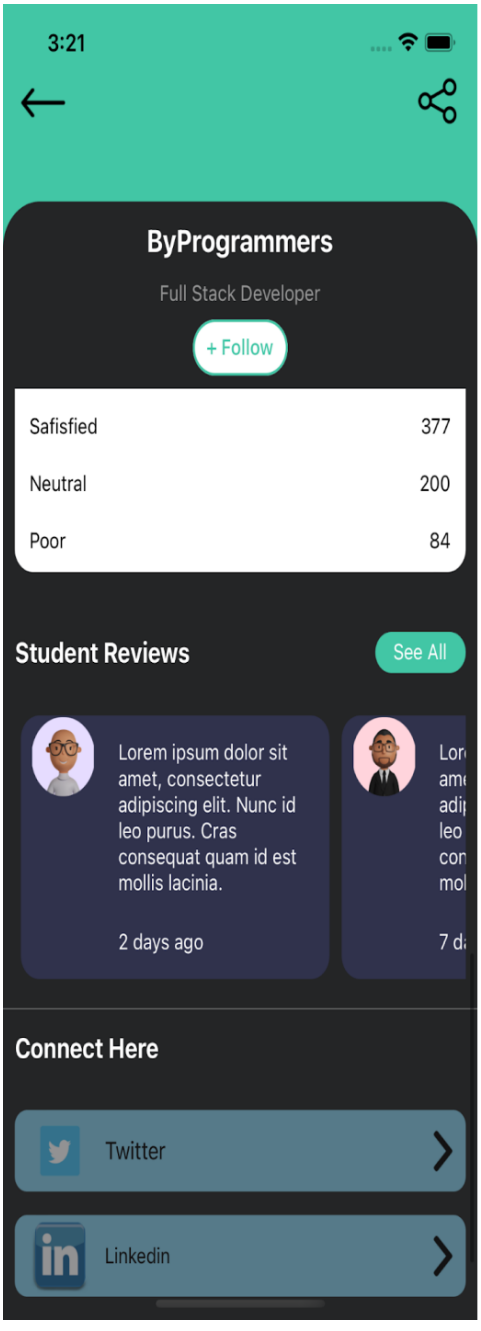


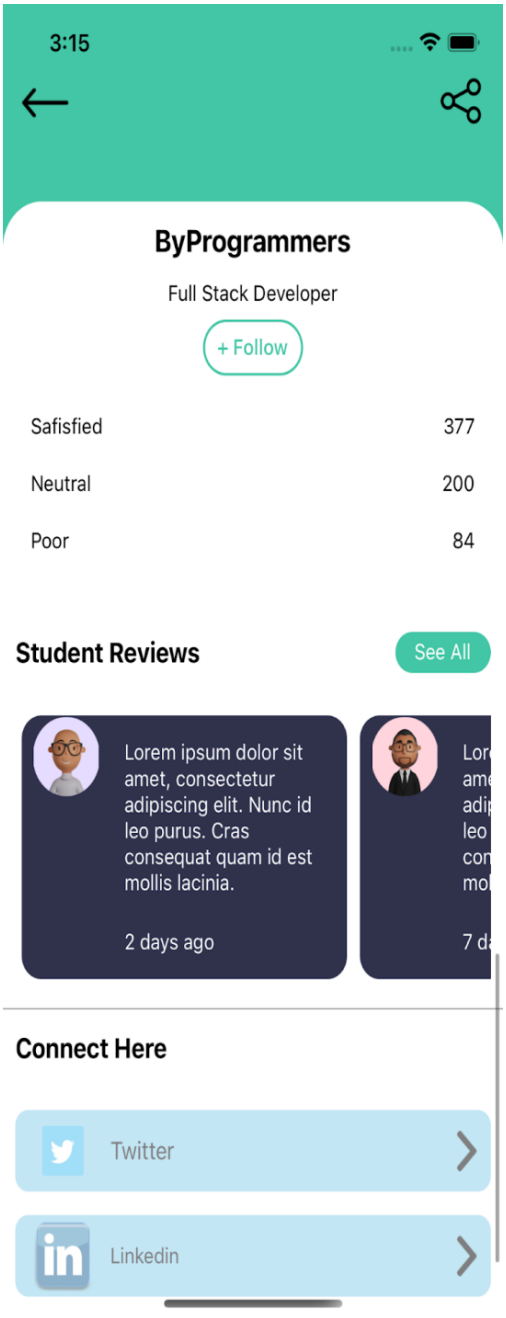
11.13 Instructor Screen :

- This is the Instructor profile screen which shows some basic information of Instructor
- Users can also follow the Instructor
- It also shows how many courses a users have
- Reviews submitted by students for the Instructor
- Users can also connect Instructor to social platform (Twitter, linkedin) by clicking the Twitter, linkedin Buttons



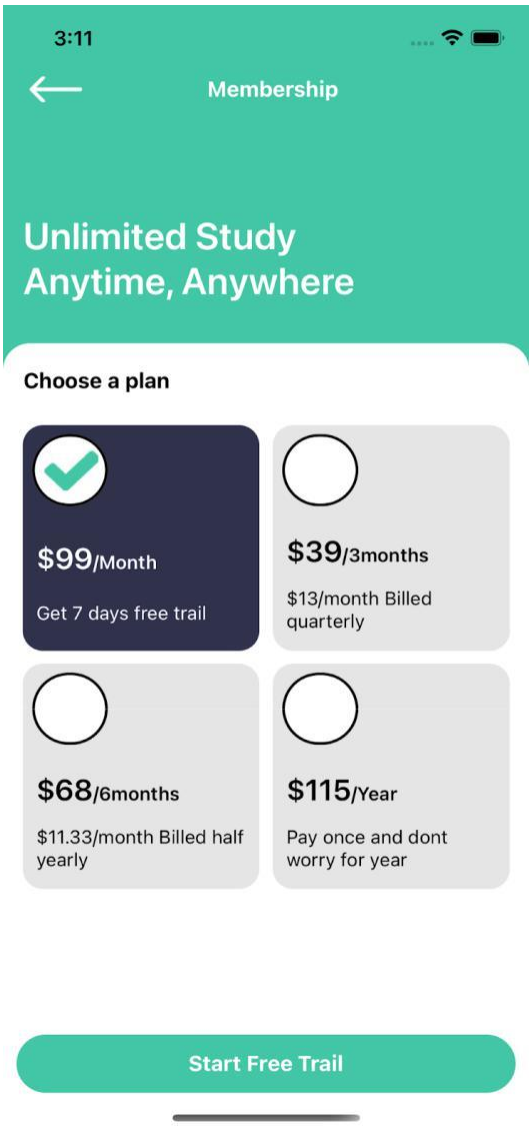



3:21
ByProgrammers
Full Stack Developer
+ Follow
Safisfied 377
Neutral 200
Poor 84
Student Reviews
See All
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc id leo purus. Cras consequat quam id est mollis lacinia.
2 days ago
Connect Here
Twitter
Linkedin


3:15
ByProgrammers
Full Stack Developer
+ Follow
Safisfied 377
Neutral 200
Poor 84
Student Reviews
See All
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc id leo purus. Cras consequat quam id est mollis lacinia.
2 days ago
Connect Here
Twitter
Linkedin

2.14 **Membership Screen** :

- This screen contains information of membership of courses



```
import
React,
{memo}
from
'react';
           import {
             FlatList,
             Image,
             Pressable,
             SafeAreaView,
             Text,
             TouchableOpacity,
             View,
           } from 'react-native';
           import {selectedTheme} from '../../config/Themes';
           import {buttons, Icon, screensData, Constant, RouteScreens} from '../../config';
           import Renderitem from './Renderitem';
           import styles from './style';


           interface CategoryProps {
             navigation: any;
             visible: any;
             setvisibe: any;
             defaultitem: number;
             setdefaultitem: React.Dispatch<React.SetStateAction<number>>;
           }


           const MembershipScreen = (props: CategoryProps) => {
             const {navigation, visible, setvisibe, defaultitem, setdefaultitem} = props;


             return (
               <SafeAreaView style={styles(selectedTheme).mainContainer}>
                 <View style={styles(selectedTheme).headContentContainer}>
                   <View style={styles(selectedTheme).header}>
                     <TouchableOpacity
                       style={styles(selectedTheme).leftButton}
                       onPress={() => navigation.goBack()}>
                       <Image
                         source={Icon.LEFT_ARROW}
                         style={styles(selectedTheme).headerLeftIcon}
                       />
                     </TouchableOpacity>
                     <Text style={styles(selectedTheme).headerText}>
```

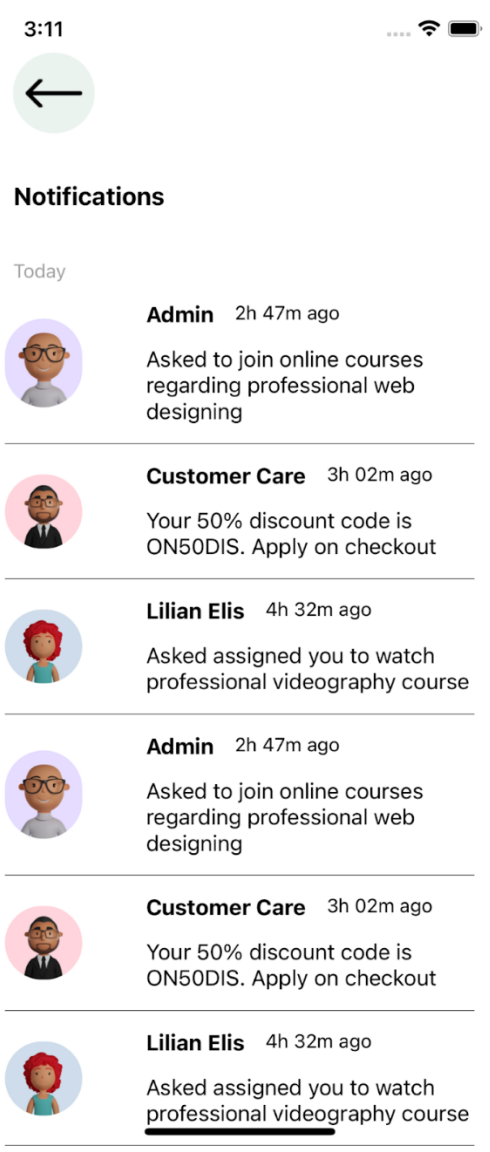
```
      {screensData.membership.NAME}
    </Text>
  </View>
  <Text style={styles(selectedTheme).text}>
    {screensData.membership.HEADER}
  </Text>
</View>
```

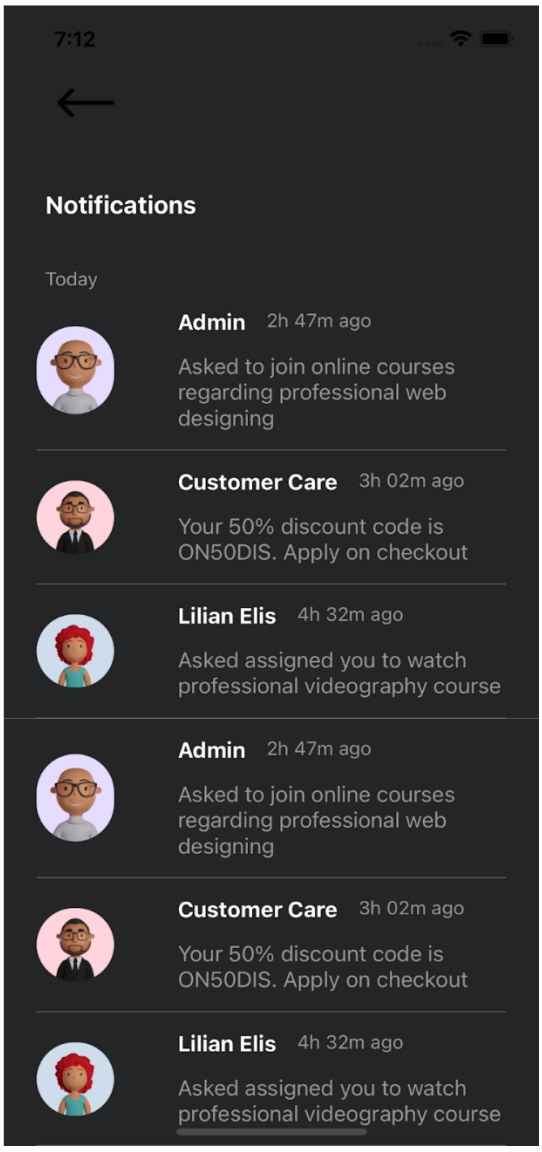
```
<View style={styles(selectedTheme).container}>
  <Text style={styles(selectedTheme).itemBigText}>
    {screensData.membership.DATA}
  </Text>


  <View style={styles(selectedTheme).flatlistView}>
    <FlatList
      data={Constant.membership}
      showsVerticalScrollIndicator={false}
      extraData={Constant.membership}
```

```
          renderItem={({item, index}) => (
            <Renderitem
              item={item}
              index={index}
              defaultitem={defaultitem}
              setdefaultitem={setdefaultitem}
            />
          )}
          numColumns={2}
          keyExtractor={(_, index) => index.toString()}
        />


        <TouchableOpacity
          style={styles(selectedTheme).button}
          onPress={() => navigation.navigate(RouteScreens.ROOTTAB)}>
          <Text style={styles(selectedTheme).buttonText}>
            {buttons.STARTFREETRAIL}
          </Text>
        </TouchableOpacity>
      </View>
    </View>
  </SafeAreaView>
);
       export default memo(MembershipScreen);
```

2.15 Notification Screen :

- This screen shows notifications like offers, mentor message new courses etc





```
import
React,
{memo}
from
'react';
           import {
             Text,
             View,
             SectionList,
             Image,
             TouchableOpacity,
             SafeAreaView,
           } from 'react-native';
           import {selectedTheme} from '../../config/Themes';
           import {Icon, Images, screensData} from '../../config';
           import RenderItems from './Renderitem';
           import styles from './style';


           interface NotificationTabprops {
             navigation: any;
             DATA: {
               title: string;
               data: {
                 id: number;
                 avatar: any;
                 name: string;
                 created_at: string;
                 message: string;
               }[];
             }[];
           }


           const NotificationTab = (props: NotificationTabprops) => {
             const {DATA, navigation} = props;
             return (
               <SafeAreaView style={styles(selectedTheme).mainContainer}>
                 <Image
                   source={selectedTheme.name == 'light' ? Images.BG : Images.BG_DARK}
                   style={styles(selectedTheme).bgImage}
                 />
                 <View style={styles(selectedTheme).container}>
                   <TouchableOpacity
                     onPress={() => navigation.goBack()}
                     style={styles(selectedTheme).leftButton}>
```

```
          <Image
            source={Icon.BACK}
            style={styles(selectedTheme).headerLeftIcon}
          />
        </TouchableOpacity>
        <Text style={styles(selectedTheme).headerText}>
          {screensData.notification.NOTIFICATIONS}
        </Text>


        <SectionList
          sections={DATA}
          showsVerticalScrollIndicator={false}
          keyExtractor={(item, index) => item + index.toString()}
          renderItem={({item}) => <RenderItems item={item} />}
          renderSectionHeader={({section: {title}}) => (
            <Text style={styles(selectedTheme).header}>{title}</Text>
          )}
        />
      </View>
    </SafeAreaView>
  );
};
export default memo(NotificationTab);
```

Conclusion

Online learning is beneficial to the students, tutors and the institution offering these courses. I would therefore recommend that online learning be implemented on all learning institutions and research on how to improve this learning process should be carried out. Online learning is not just a change of technology. It is part of a redefinition of how we as a species transmit knowledge, skills, and values to younger generations of workers and students. I will end this book by daring to make a few predictions of how e-learning and the functions it serves will continue to develop.

References:

1. M. Backes, S. Bugiel, E. Derr, P. McDaniel, D. Octeau and S. Weisgerber, "On demystifying the android application framework: Re-visiting android permission specification analysis", *25th {USENIX} Security Symposium ({USENIX) Security 16)*.
2. O. Cinar, "Android Platform" in Android Quick APIs Reference, Berkeley, CA:Apress, pp. 1-14, 2015.
3. S.Holla and M. M. Katti, "Android based mobile application development and its security", *International Journal of Computer Trends and Technology*, vol. 3, no. 3, pp. 486-490, 2012.
4. M. Latif, Y. Lakhrissi, E. H. Nfaoui and N. Es-Sbai, "Cross platform approach for mobile application development: A survey", *2016 International Conference on Information Technology for Organizations Development (IT40D)*, pp. 1-5, March 2016.
5. P. R. De Andrade, A. B. Albuquerque, O. F. Frota, R. V. Silveira and F. A. da Silva, *Cross platform app: a comparative study*, 2015
6. M. Latif, Y. Lakhrissi, E. H. Nfaoui and N. Es-Sbai, "Review of mobile cross platform and research orientations", *2017 International Conference on Wireless Technologies Embedded and Intelligent Systems (WITS)*, pp. 1-4, April 2017.
7. T. Bläsing, L. Batyuk, A. D. Schmidt, S. A. Camtepe and S. Albayrak, "An android application sandbox system for suspicious software detection", *2010 5th International Conference on Malicious and Unwanted Software*, pp. 55-62, October 2010.
8. J. Philip and M. Raju, "A Formal Overview of Application Sandbox in Android and iOS with the Need to Secure Sandbox Against Increasing Number of Malware Attack", *Indian Journal of Computer Science*, vol. 4, no. 3, pp. 32-40, 2019.
9. M. Spreitzenbarth, T. Schreck, F. Echtler, D. Arp and J. Hoffmann, "Mobile-Sandbox: combining static and dynamic analysis with machine-learning techniques", *International Journal of Information Security 14*, no. 2, pp. 141-153, 2015.
10. X. Wang, K. Sun, Y. Wang and J. Jing, "DeepDroid: Dynamically Enforcing Enterprise Policy on Android Devices", *Ndss*, February 2015.
11. P29119-4-DISMay2013 - IEEE Draft International Standard for Software and Systems Engineering--Software Testing--Part 4: Test Techniques.
12. 15026-2-2011 - IEEE Standard--Adoption of ISO/IEC 15026-2:2011 Systems and Software Engineering--Systems and Software Assurance--Part 2: Assurance Case.
13. 730-2014 - IEEE Standard for Software Quality Assurance Processes.
14. 24765-2017 - ISO/IEC/IEEE International Standard - Systems and software engineering—Vocabulary.

828-2012 - IEEE Standard for Configuration Management in Systems and Software Engineering