B3

by vandana asv

Submission date: 23-Jun-2022 10:01PM (UTC+0530)
Submission ID: 1861645691
File name: Batch_B3_-_Placementor_-_Major_Project_Report_2.docx (9.79M)
Word count: 9653
Character count: 48307

ABSTRACT

Every college desire to see their students settled in their career, the first phase of which is them getting a good placement. Placements also reflect the standards of the college. Hence placements are a key element to both college and students. But today, many colleges are using some sort of messaging platform such as WhatsApp, Telegram to establish communication between students and Training & Placement Cell on placement-related activities. This also consists of a lot of manual work and is tedious. Negligence on even a small thing from either party would result in great loss. A better solution is needed to address the issues faced by both college and students in the context of placements. Hence, we are building a platform that enables students to seamlessly access placement-related information while on the other side Training & Placement cell can get an easy way to both monitor students and provide information about the same.

LIST OF FIGURES

CHAPTER - 1

INTRODUCTION

1.1 PROBLEM DEFINITION

As placements play a major role for both college and students, a dedicated platform is required to solve issues like:

1) No proper communication between students and the T&P department.
2) Partial/Incorrect information regarding placement drives.
3) Missing deadlines of opportunities, both on-campus and off-campus.
4) Improper tracking of students' progress from T&P and students themselves.
5) No proper guidance to prepare for placements.
6) Inefficient feedback system.

The problem here lies in the non-availability of a dedicated online single window channel between the Training and Placement team and the students which could solve the above-mentioned issues.

Hence, building a platform that enables students to seamlessly access placement-related information while on the other side Training & Placement cell can be an easy way to both monitor students and provide information about the same.

1.2 EXISTING SOLUTIONS

1.2.1 WhatsApp Groups

In our college, the T&P cell uses WhatsApp as a medium for communication with students about placements. Each branch and batch are asked to join in their respective WhatsApp groups and every information related to any new drive is posted as a message in the group. Sometimes polls like those “who have registered for an exam” or “who got qualified in a placement round” are also posted for which the students reply with a hand raise emoji to confirm their completion or update a list with their roll number and name. Also, when a new company arrives its basic information along with the tech stack required is posted in the group, sometimes previously placed student contact details are provided which would help current students to clear their doubts when needed. Also, if there are any queries related to the drive the students post the doubt in the group and the respective placement coordinator resolves it.

These days many companies started hiring through country-wide hackathons and exams which are considered to be off-campus. Many students do participate in outside college drives and internships which then need to be communicated to the T&P Department and require updation in records regularly.

Advantages:

1) The communication is easily done.
2) There is no need to install any new application (as WhatsApp is highly used by many people).

Disadvantages:

1) A lot of manual work is intended for both faculty and students.
2) There is also a high chance of the messages being spammed which might lead to loss of valuable information.
3) Students might ignore the notifications of WhatsApp thinking it is not from the placement group.
4) The platform cannot be customized as per the required format a college needs.

The existing system is more manually done which intakes many human errors and irregularity in the maintenance of data. Communication is highly based on WhatsApp. It does not provide placement-specific information communication features and issues arise when Meta's server fails. Till now no single proper channel is established between the TnP department and the students.

1.2.2 Google Classrooms

In NIT Hamirpur, the mode of communication being used is Google classroom. It is similar to the usage of WhatsApp; Google Classrooms are made for each branch's batch and respective information is placed in respective classrooms. The Google Meet link for things like pre-placement talk is attached directly in the classroom so that students can easily access it.

Advantages:

1) Communication is relatively easy.
2) No spamming can be done.
3) As classroom notifications differ from WhatsApp notifications, students know that something related to placements has come up when it is from the classroom app.

Disadvantages:

1) There is manual work intended here too to collect information from students.
2) There is no proper feedback system.
3) The platform cannot be customized as per the required format a college needs.

1.2.3 Web Portals (IIT Kanpur)

At the Indian Institute of Technology, Kanpur, a single-window platform is developed by the college where companies register onto this platform and students can directly apply from here for the placements. While this process does not require the direct intervention of the TnP department, they regulate and monitor the platform and the process continuously. Students can directly ask doubts and questions to the firm and get a direct reply. The process has been more automated and organized.

Advantages:

1) Reduce workload on TnP Cell.
2) Provide more transparency in the process.
3) Students would be directly introduced to the real placement world with many more options so that they can be selected for their future.

Disadvantages:

1) Companies would have a lot of workloads and need to be regularly active on the platform.
2) Not all companies are digitally good and active.
3) Due to hectic workloads to companies, placement to the college might get impacted.

The above model cannot be used in our college – VNR VJIET as we have many companies approaching us who are not good with the online world and prefer a direct conversation with the management (or) Placement Cell rather than with students.

CHAPTER - 2

METHODOLOGY

2.1 PROPOSED SYSTEM

We took insights from all the existing solutions and tried to remove some of the disadvantages in them but also retain the advantages. Hence, we are building an app and a website that would enable students and T&P to communicate with each other in a better way.

The Mobile Application

We are building an app that connects students to the T&P Faculty. As apps are handy and we can also use on-device storage to provide details in offline mode too.

The following are the screens and their details along with their design:

1) Login Screen

We have induced a security check to access the app's content. This is a useful step as it protects the college's placement data from the outside world. This data should only be accessible to the related users only.

2) Companies Page

This screen lists out all the available on-campus and off-campus opportunities

Both the lists are separately shown as a horizontal and vertical scroll view and with each company listed as a tile in both the lists. When clicked on any of the tiles the respective company page is opened.

3) On-Campus Company Page

As the college has all the information about the companies visiting, we have a lot of information available on this page which includes

a) **About the firm:** A little information about the firm so that the students get to know the baseline of the organization.
b) **Job Description:** This contains the specific description of the role the company is offering in terms of expectations of the company from the student.
c) **Skillset required:** This section contains the stack required for the role of the company.
d) **Process:** This contains a flow chart that shows the students the steps they need to take in the process of trying to get placed in the company.
e) **Previously placed contacts:** This section provides information about alumni who got placed in the same company or role so that if they have any doubts, they could contact them with just a click.
f) **Experiences:** Some of the experiences from the previous batches are collected and are put here to let the students know a pinch of how the process and company are going to be.
g) **FAQs:** The most frequently asked questions about the firm by the students are answered and are put in this section.

Also, a drive link with all the resources related to the organization in the possession of colleges such as mock tests, etc., and an add-to calendar option would be provided so that this page could be a one-stop-shop for all the requirements of students.

4) Newsfeed Page

This screen is used to provide updates from TnP Cell to students. This can be used to decrease the dependency on online conversation applications like WhatsApp and Telegram. It also provides a feature to conduct polls.

5) Resources Page

This screen provides resources as sample resumes, video resumes, aptitude lectures, etc. This would be helpful for students to refer, watch and develop their knowledge and portfolios anytime anywhere.

6) Profile Page

This page is like a personal screen for the students. Here, they can find options to more efficiently work in the placement season and easily communicate with the TnP Cell.

Some of the basic options available here are:

a) **To-do List:** This provides students to manage placement-related tasks and do them in a planned manner.
b) **TnP Coordinators:** List of TnP cell coordinators department-wise with their contact information.
c) **Off-Campus Form:** A form to communicate with the TnP cell regarding off-campus opportunities.
d) **Raise a Ticket:** To raise an issue to the Placement cell.
e) **Log Out:** To logout the user.

Web Application

The web application is built to assist T&P Faculty to add or update data which is displayed on Mobile application's side. As mostly faculty will be in campus while working, they can use the large screen capabilities to seamlessly do operations.

The following are the screens and their details along with their design

1) Login Screen

To access the website, the faculty has to login to the application. Here normal student logins are separated from faculty logins, allowing only faculty to login to the website while blocking students from modifying data.

2) All Pages Screen

This is the initial Screen that we see after successfully logging in through which all the other screens can be accessed. All the available screens are presented as tiles which when clicked take you to the respective screen.

3) Companies Screen

In this screen we can see both On-Campus and Off-Campus Companies with the help of a toggle button available. In each of the screens the companies are laid out in a grid view as tiles with the most important information visible on the tile itself. When clicked on both On or Off Campus tile, we are navigated to a new screen which shows us all the details related to that company. The details are same as that of company in mobile application.

In each of the detailed screens, we have options to either edit or delete the company. Edit takes you to a new screen with pre-filled form of the company chosen and you can edit the details as per your will and submit the request. Delete takes confirmation if you want to delete the company and deletes the company permanently if chosen to delete.

There is an Add Company Button, which allows us to add both on-campus and off-campus companies via Firebase using Forms which take the respective fields of input

a) **On-Campus Company Form Screen**

This form appears when we choose to add an on-campus company. The form takes the following inputs

i. **Company Name –** String Input
ii. **Company Type –** Dropdown Input
iii. **Role Name –** String Input
iv. **Role Type –** Dropdown Input
v. **About The Firm –** String Input
vi. **Job Description –** String Input
vii. **Eligibility –** String Input
viii. **Package –** Number Input
ix. **Last Date to Apply** – Date Picker Input
x. **Skillset Required –** Complex Input
xi. **Process Timeline –** Complex Input
xii. **Previously Placed Contact Details –** Complex Input
xiii. **Experiences –** Complex Input
xiv. **FAQs –** Complex Input
xv. **Link For Applying –** String Input

xvi. **Google Drive Link –** String Input

b) Off-Campus Company Form Screen

This form appears when we choose to add an off-campus company. The form takes the following inputs

i. **Company Name –** String Input

ii. **Role Name –** String Input

iii. **Description –** String Input

iv. **Link For Applying –** String Input

The input types used above can be summarized as follows :

String Input - Alphabetical Input

Number Input - Numeric Input

Dropdown Input - Input chosen from a given dropdown of options

Date Picker Input - Input chosen from a Date Picker

Complex Input - A collection of inputs taken at once with the help of a pop-up dialog as a single object

4) Tickets Screen

All the tickets raised by students are accumulated here in a list view. The faculty can see who has raised the request and can also mark it resolved or delete it after the ticket is resolved.

5) Off Campus Opportunities Screen

All the off-campus opportunities, the students post are listed here. The information about who put up that update is also available at the same place and the faculty can contact them for any doubts regarding that opportunity. After verification of the

credibility of the company and role, the same can be added to off campus companies and then the specific tile can be deleted.

6) Newsfeed Screen

This screen allows faculty to add new posts and polls which will directly get reflected in the mobile application side.

a) Post

This is generally a piece of information that the faculty wants the students to know. It also has a separate field for any links the faculty wants to provide. The screen also has a feature to view all the posts done and also who all have seen the post can be viewed as a list.

b) Poll

This is generally used to get information from students about something. A question can be given and a number of poll options can be entered, from which the student is supposed to select one. The whole statistics of who selected what can be viewed by the faculty and decisions can be made accordingly.

All the polls and posts can be seen in this screen itself in reverse chronological order. There is a single button available which when clicked gives us the option to add either poll or post through a pop-up window, when all the details are furnished and enter is clicked a new poll or post is created.

7) Resources Screen

This screen directly furnishes the resource screen on mobile application part. Faculty can add new resources via the add button which gives them a pop-up dialog to enter information. They can also edit already added resources and delete resources which are no longer useful to students via the same screen. All the added resources are visible in the same screen in a grid view format with both edit and delete options.

8) T&P Coordinators Screen

In this screen the details of all the T&P Faculty are available in a grid view format, which can also be filtered according to their department and viewed. New T&P faculty can be added via the add button which gives a pop-up window in which the details can be filled. The faculty details can also be deleted using the delete button on specific tile.

9) Students Screen

One of the most important responsibilities of T&P Department is to create student accounts. Although it is done only once a year, it is crucial. To make this process seamless, we have approached this feature in a slightly different way than all others. For creating student account, we take input of a google sheet in a specific format and with the click of a single button all the student accounts are created. All the statuses of creation are shown the google sheet i.e., if any student account is not created successfully, we can review it in a table along with the error that occurred. Similarly, to delete users another google sheet with all the details of students is needed which will then be passed to delete those student accounts.

All the created student details can be viewed in the same screen in a grid view format and can be used to check if all of them are correct.

2.2 REQUIREMENTS

2.2.1 Design Tools:

FIGMA

Figma is a graphics editor and primarily web-based design tool, with desktop applications for macOS and Windows providing additional offline capabilities. Figma models can be observed in real-time on smartphones using the Figma Mirror companion apps for Android and iOS. Figma's feature set is oriented towards user interface and user experience design, with a focus on real-time collaboration.

We selected Figma because of the following reasons:

1) Web-based prototyping and designing tool.
2) Figma Mirror helps to view prototypes on devices and get a complete picture.
3) Easy to learn.
4) Official community of designers at FIGJAM
5) Supports Real-Time Collaboration.
6) Easy and faster to develop applications using design to code plugins.

2.2.2 Frameworks:

FLUTTER

Flutter is an open-source user interface SDK (Software Development Kit). From a single codebase, it is possible to create cross-platform apps for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web.

We selected Flutter because it has the following features

1) Open Source
2) Single Codebase
3) Dart as Programming Language
4) Hot Reload and Development
5) Native App like Performance
6) Huge Tech Community
7) Use of Custom Widgets
8) Attracts More Investors
9) Create Apps for Mobile, Desktop, and Web
10) Requires Less Testing

DART

Dart is a multi-paradigm programming language that is functional, imperative, object-oriented, and reflective. It is developed by Google, designed by Lars Bak and Kasper Lund, and licensed under BSD. The language's extension is ".dart". It has a syntax style of C and can be used to build cross-platform applications. Some of its implementations are Dart VM, dart2native, dart2js and one of its major implementations is flutter.

FIREBASE

Firebase is a computing and development tool that was developed by Firebase.Inc, designed by James Tamplin and Andrew Lee. It was later acquired by Google. It is a

software solution to build mobile (both android and iOS) and web applications which require a backend service. Most of the services provided by Firebase are free of cost for demo purposes and it is a pay-as-you-use platform and only costs for what we use.

It has a large number of features in which some of the important ones are

1) Authentication
2) Firestore Database
3) Realtime Database
4) Cloud Hosting
5) Cloud Functions
6) Crashlytics
7) In-App Messaging
8) Push Notifications

2.2.3 Other Software Requirements:

ANDROID STUDIO

Android Studio is an IDE (Integrated Development Environment) specially built for mobile application development in Android. It is built by Google and JetBrains and is written in Java, Kotlin, and C++. It also supports Flutter to build cross-platform applications. It provides features like AVD (Android Virtual Device) or Emulator which help us run the application on a virtual android device rather than an original one for testing purposes. The Emulator can be used hand in hand with other IDE's like VSCode.

Visual Studio Code IDE

Visual Studio Code is a code editor which supports many languages and frameworks. Its main aim is to provide a single place to work on any language or platform. It was developed by Microsoft. We have chosen it over Android Studio as it is lightweight and won't go hard on the computer's RAM. It has almost all the features of Android Studio and also many more which include

1) Syntax Highlighting
2) Debugging
3) Intellisense
4) Code Refactoring
5) Embedded Git
6) Many Extensions (Flutter, Bracket Colorizer, GitHub Co-pilot)

2.2.4 Hardware Requirements:

1) Android smartphone running Android OS or iOS.
2) Web Browser (Google Chrome, Microsoft Edge, etc.)

2.3 WORKFLOW DIAGRAMS

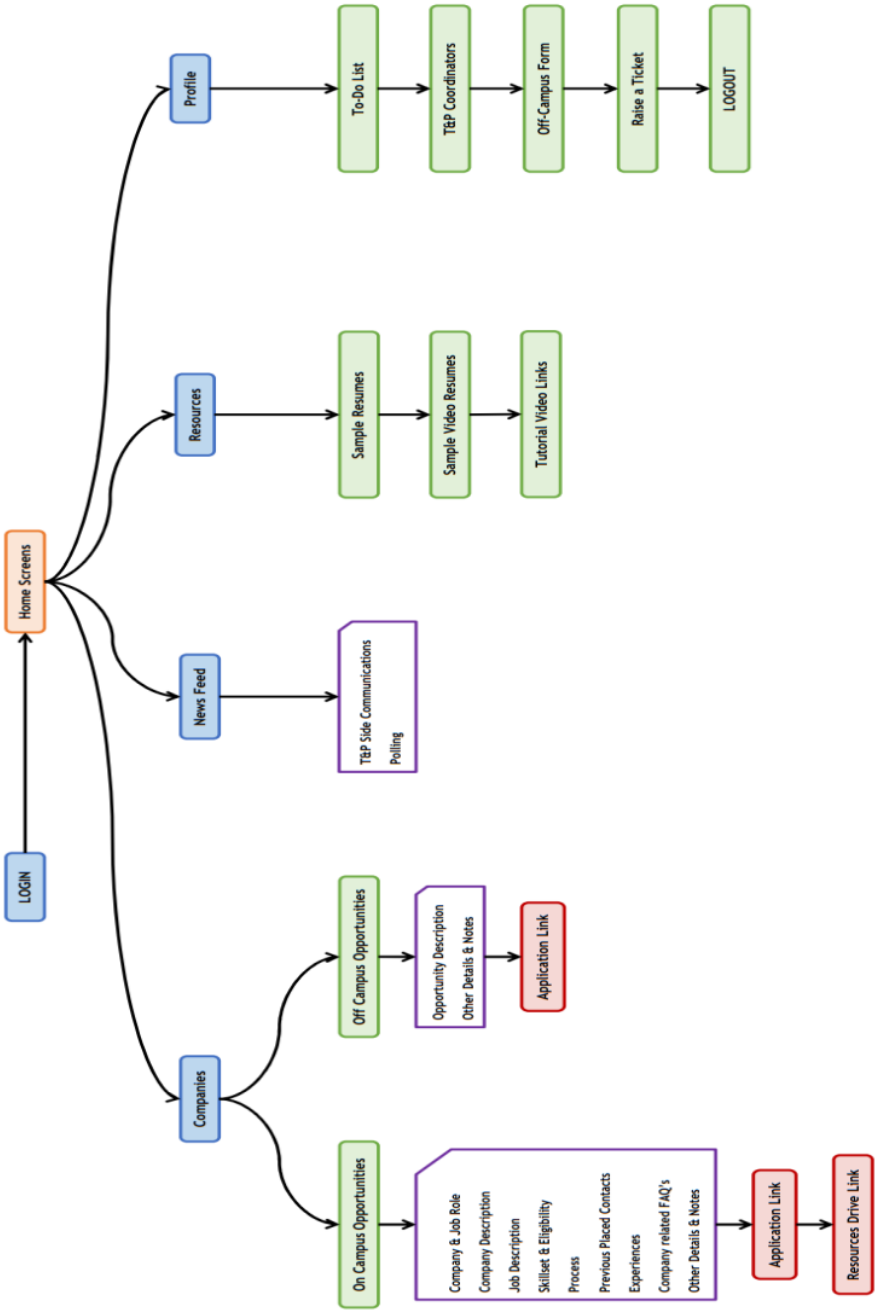


Fig 2.3.1 – Workflow Diagram for Mobile Application

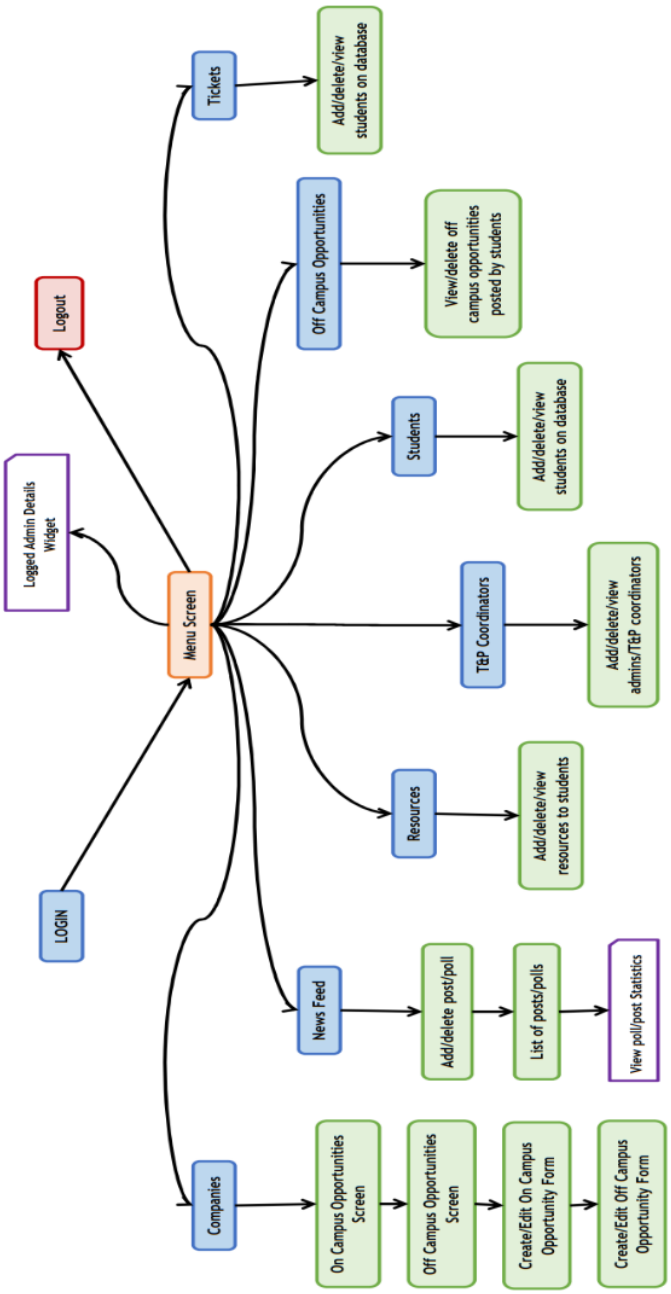


Fig 2.3.2 – Workflow Diagram for Website

2.4 DESIGNS

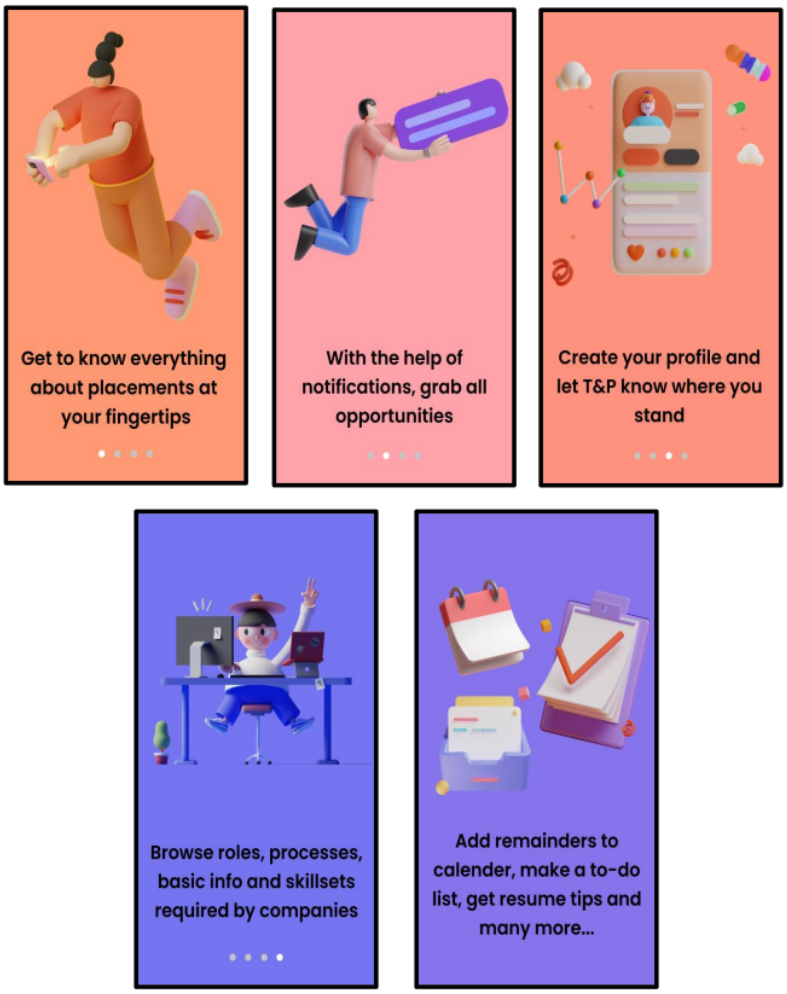


Fig 2.4.1 – On-boarding Screens

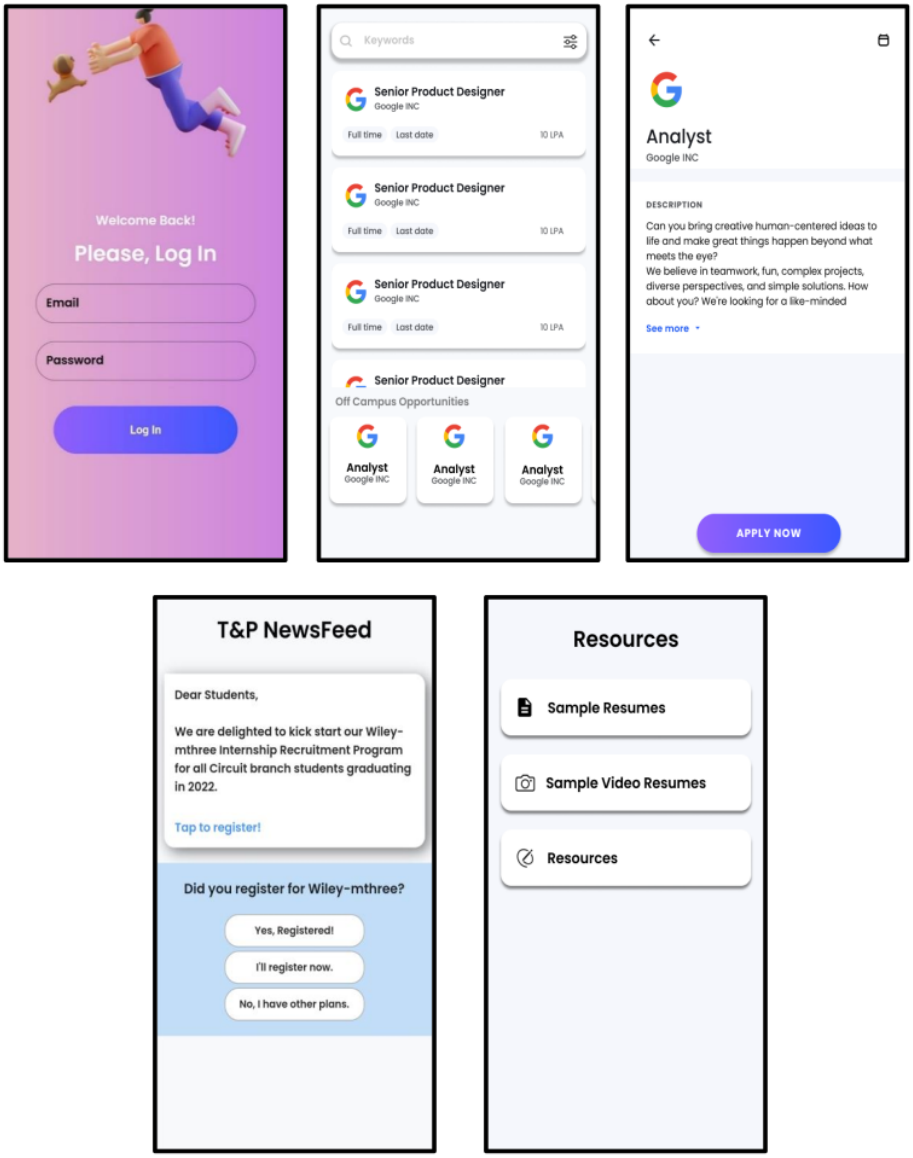


Fig 2.4.2 – Login, Companies Tab, Off-Campus Opportunity Screen, Newsfeed and Resources Tabs

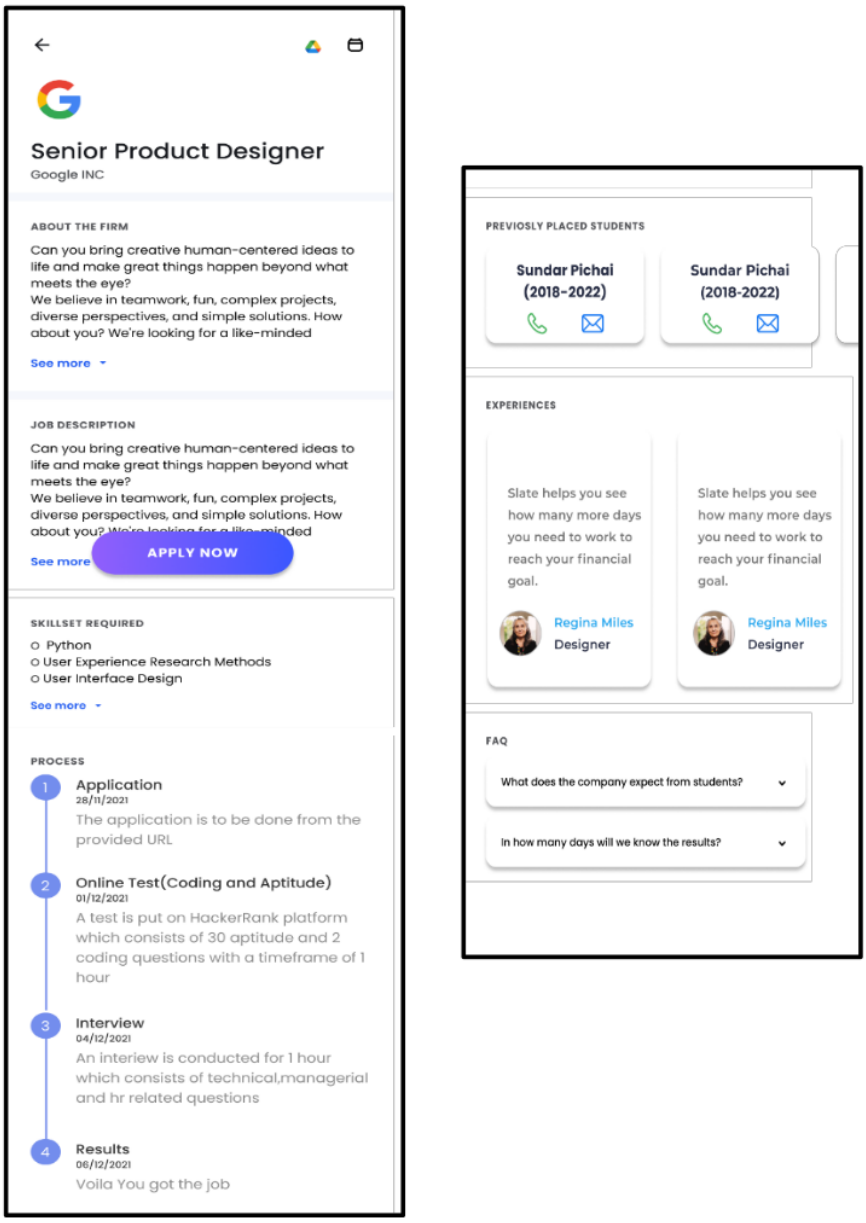


Fig 2.4.3 – On-Campus Opportunities Screen

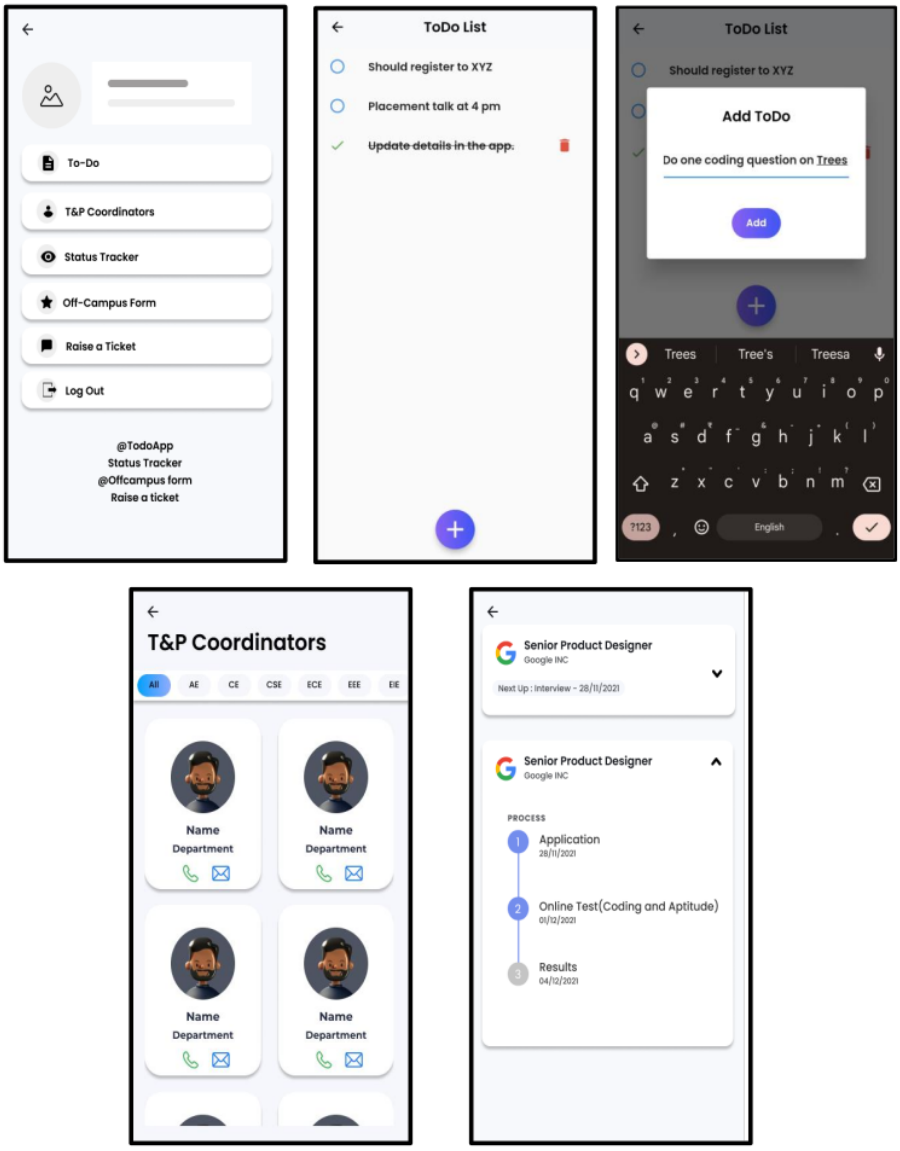


Fig 2.4.4 – Profile and its constituent screens

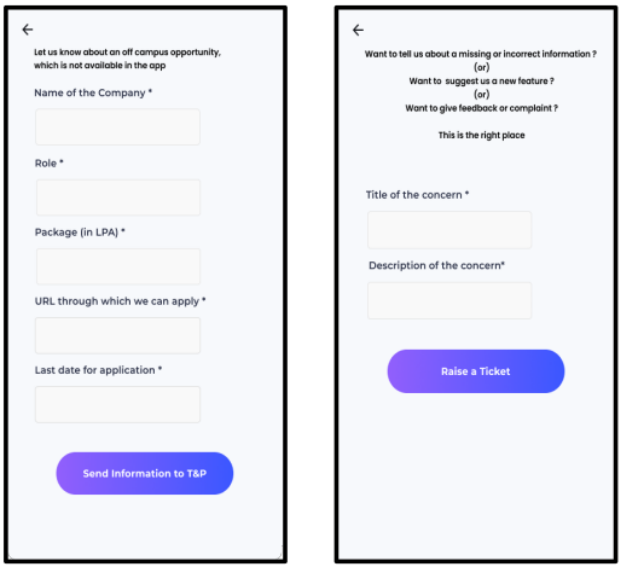


Fig 2.4.5 – Off-Campus Opportunity and Raise Ticket Forms

CHAPTER – 3

IMPLEMENTATION

We used Flutter Framework which is built upon Dart Language by Google as it supports cross platform development i.e., with a single codebase we can deploy the application anywhere which is Play Store and AppStore for mobile and any web browser for websites. For backend we used Firebase which seamlessly connects with Flutter and also has built in database and authentication.

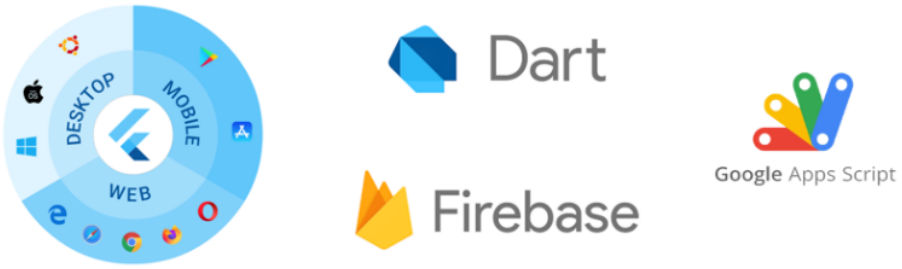
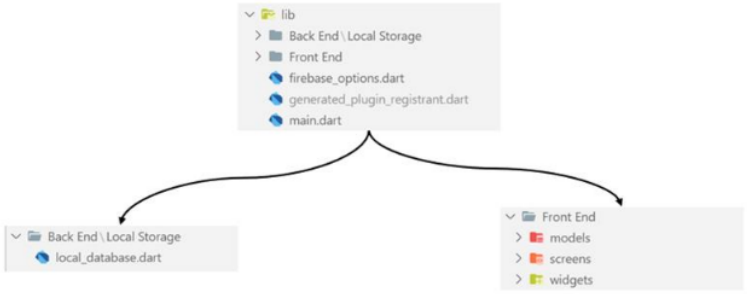
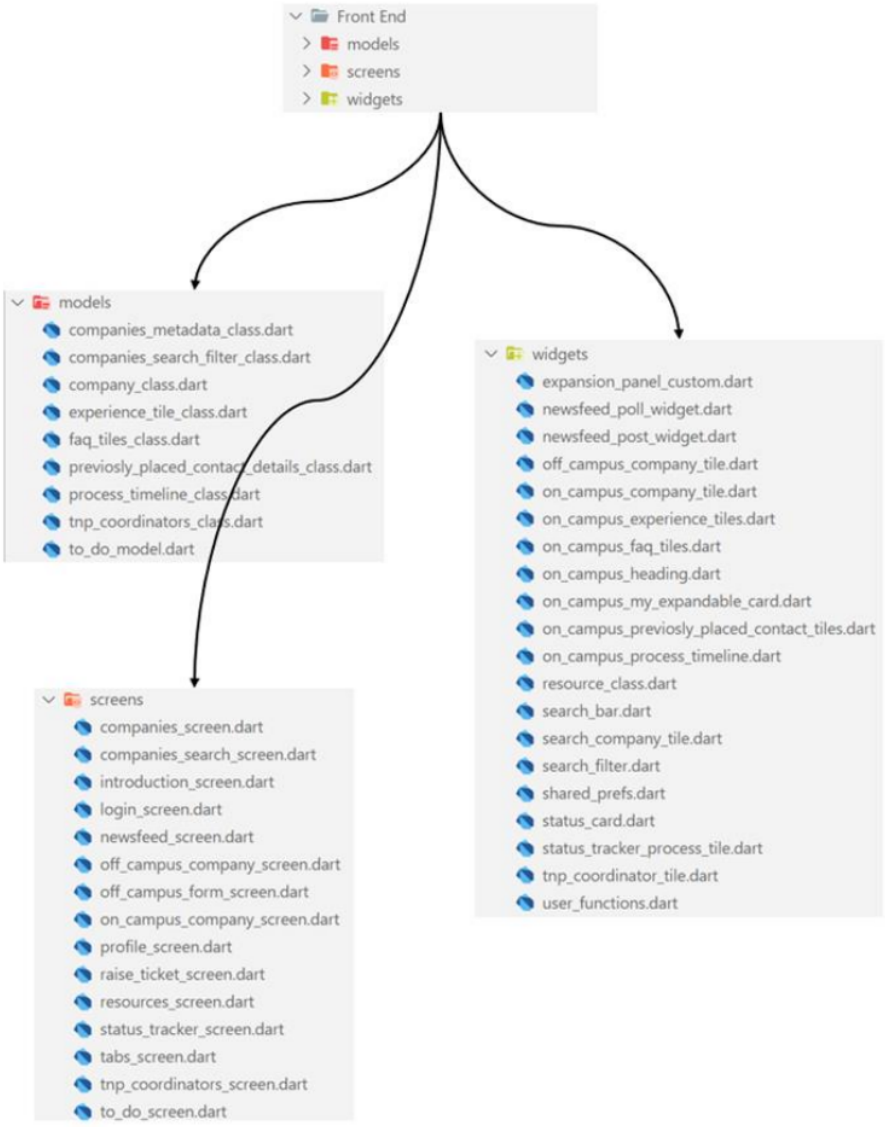


Fig 3.1 – Technologies Used

3.1 FILE STRUCTURE

Mobile Application




Front End
models
screens
widgets
models
companies_metadata_class.dart
companies_search_filter_class.dart
company_class.dart
experience_tile_class.dart
faq_tiles_class.dart
previosly_placed_contact_details_class.dart
process_timeline_class.dart
tnp_coordinators_class.dart
to_do_model.dart
widgets
expansion_panel_custom.dart
newsfeed_poll_widget.dart
newsfeed_post_widget.dart
off_campus_company_tile.dart
on_campus_company_tile.dart
on_campus_experience_tiles.dart
on_campus_faq_tiles.dart
on_campus_heading.dart
on_campus_my_expandable_card.dart
on_campus_previosly_placed_contact_tiles.dart
on_campus_process_timeline.dart
resource_class.dart
search_bar.dart
search_company_tile.dart
search_filter.dart
shared_prefs.dart
status_card.dart
status_tracker_process_tile.dart
tnp_coordinator_tile.dart
user_functions.dart
screens
companies_screen.dart
companies_search_screen.dart
introduction_screen.dart
login_screen.dart
newsfeed_screen.dart
off_campus_company_screen.dart
off_campus_form_screen.dart
on_campus_company_screen.dart
profile_screen.dart
raise_ticket_screen.dart
resources_screen.dart
status_tracker_screen.dart
tabs_screen.dart
tnp_coordinators_screen.dart
to_do_screen.dart

Website

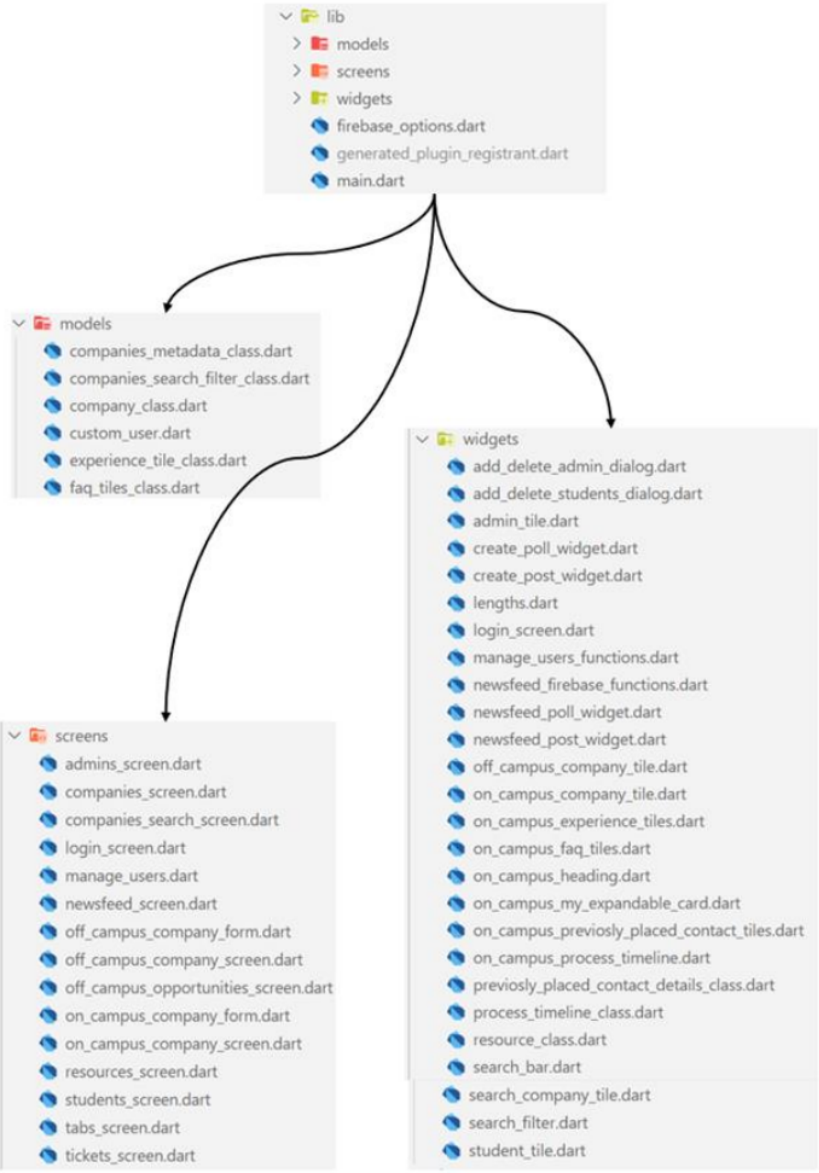


Fig 3.1.1 – File Structures

The above file structure is used to build the application

- Backend
  - Has the logic and classes related to backend
- Frontend
  - Models
    - Contains Blueprints of data
  - Screens
    - Contains Main Screens Classes
  - Widgets
    - Contains classes which aids the Screens

3.2 MODELS

Models are the skeleton of the data in the project which defines the structure of each and every data element in the application. They standardize the format of each element using classes. It contains class variables, constructors and other related and useful functions. Most of the models are common on both App and Website side.

Many models are fetched from and pushed to Firebase, hence, they contain 'fromJSON' and 'toJSON' functions. To print the models, we use the 'toString' function. Many classes, class variables, class functions are self-explanatory.

1) Companies Metadata Model

Companies Metadata contains metadata or cumulative data aggregated of the companies. These data can be used to provide constrained options or easy form fill data to the users.

Class variables

```
final List companyNames;
final List companyTypes;
final List roleNames;
final List roleTypes;
final List offerTypes;
final List skillset;
final double packageMax;
```

2) Companies Search Filter Model

Companies Search Filter model is used to manage the filters in companies search. It contains function 'filterCompanies' which filter the list of companies based on the applied filters and returns a string for each filtered company.

Class variables

```
late List<Company> companies;
List companyNameFilter = [];
List companyTypeFilter = [];
List roleNameFilter = [];
List roleTypeFilter = [];
List offerTypeFilter = [];
List skillsetFilter = [];
late List<int> packageFilter;
bool lateVariablesInitialized = false;
late double packageMax;
```

3) Company Model

Company model is the model class for every company. It contains function 'toSearchTypeString' which converts the company object to searchable string.

Class variables

```
final String id;
final String companyName;
final String? companyType;
final String roleName;
final String? roleType;
final String? aboutTheFirm;
final String jobDescription;
final List? skillset;
final ProcessTimelineClass? processTimeline;
final List<PreviouslyPlacedContactDetailsClass>? previouslyPlacedContacts;
final List<ExperienceTileClass>? experiences;
final List<FAQTilesClass>? faqs;
final String? lastDate;
final double? package;
final String linkToApply;
final String? driveLink;
final String? eligibility;
final String offerType;
```

4) Experience Tile Class Model

Experience Tile Class model is the class for every experience registered in all companies.

Class variables

```
final String experience;
final String name;
final String batch;
final String rollNo;
```

5) FAQ Tiles Class Model

FAQ Tiles Class model the for every FAQ registered in all companies.

Class variables

```
final String question;
final String answer;
final String timestamp;
```

6) Previously Placed Contact Details Class Model

Previously Placed Contact Details Class model for every previously placed contacts registered in all companies.

Class variables

```
final String name;
final String rollNo;
final String email;
final String phone;
final String linkedin;
```

7) Process Timeline Class Model

Process Timeline Class model for timeline mentioned in all companies

Class variables

```
final List steps;
final List names;
final List dates;
final List descriptions;
```

8) T&P Coordinator Model

T&P Coordinator model a model for all admins/T&P coordinators in the Profile->T&P Coordinators Page.

Class variables

```
String? name;
String? email;
String? password;
String? uid;
String? department;
String? phone;
String? employeeId;
```

3.2 MOBILE APPLICATION IMPLEMENTATION

1) Introduction Screen

This screen appears only for the first time when the application is opened in a specific device, it basically takes the students through the capabilities of the application. It acts as an onboarding screen to the students. All the code of this screen is available in introduction_screen.dart file. Each page in the carousel is created using a custom widget "CarouselPage" which takes a picture, background color and a text to be displayed on the screen.

A package "introduction_screen" is used to set up this with options of going to next page via swiping or using the next button, skipping to the last page using the skip button and in the last page when done is clicked the user is taken to the login screen with which when logged in, the student can use the application.

```
onDone: () {
  Navigator.of(context).pushReplacementNamed('/loginPage');
},
```

2) Login Screen

This screen is used to authenticate the user into the application. We are using Firebase as a service for authentication. The code of this screen is present in login_screen.dart file.

A two-field form which takes name and password as input is shown on the screen and when correct details are entered the user/student is authenticated and is logged into the application, else a pop-up dialog will be shown indicating that some error occurred like password mismatch, invalid email id, etc.

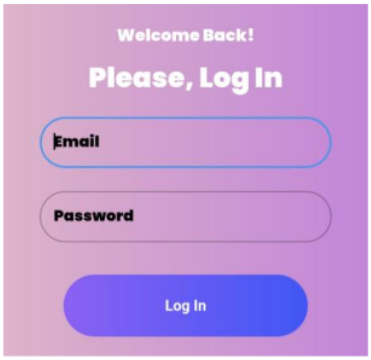


Fig 3.2.1 – Login Widget

Whenever the submit button is clicked the _submit function is called which sends a login request to the firebase which in turn returns the status of the user, real or not.

If successfully logged in the user sees the Tabs Screen.

```
Future<void> _submit() async {
  if (!_formKey.currentState!.validate()) {
    // Invalid!
    return;
  }
  _formKey.currentState!.save();
  try {
    setState(() {});
    await FirebaseAuth.instance.signInWithEmailAndPassword(
      email: _email,
      password: _password,
    );
  } on PlatformException catch (err) {
    var message = 'An error occurred, pelase check your credentials!';

    if (err.message != null) {
      message = err.message!;
    }
    showErrorDialog(message);
  } catch (error) {
    var errorMessage = 'Could not authenticate you. Please try again later.';
    showErrorDialog(errorMessage);
  }
}
```

```
void showErrorDialog(String errorMessage) {
  showDialog(
    context: context,
    builder: (ctx) => AlertDialog(
      title: const Text('An Error Occured'),
      content: Text(errorMessage),
      actions: <Widget>[
        TextButton(
          child: const Text('OKAY'),
          onPressed: () {
            Navigator.of(ctx).pop();
          },
        ), // TextButton
      ], // <Widget>[]
    ), // AlertDialog
  );
}
```

3) Tabs Screen

This is the screen that contains all the other main screens of the App. It provides us with a bottom navigation bar which allows us to navigate to different screens of the app. By default, we will be on Companies Page. The code for this is available in tabs_screen.dart file.

The available screens are Companies Screen, Newsfeed Screen, Resources Screen and Profile Screen.

Fig 3.2.2 – Bottom Navigation Bar

Whenever one of the icons is clicked the _handleNavigationChange function is triggered and the respective screen is displayed.

```
void _handleNavigationChange(int index) {
  setState(
    () {
      switch (index) {
        case 0:
          _child = const CompaniesScreen();
          break;
        case 1:
          _child = const NewsfeedScreen();
          break;
        case 2:
          _child = const ResourcesScreen();
          break;
        case 3:
          _child = const ProfileScreen();
          break;
      }
      _child = AnimatedSwitcher(
        switchInCurve: Curves.easeOut,
        switchOutCurve: Curves.easeIn,
        duration: const Duration(milliseconds: 500),
        child: _child,
      ); // AnimatedSwitcher
    },
  );
}
```

4) Companies Screen

This is one of the most important screens in the whole application. Here students can see all the available opportunities to them. It has three sections in itself which are Search Bar, On Campus Opportunities and Off Campus Opportunities. The code is present in companies_screen.dart file.

All the data is fetched from Firebase, there are two main collections we fetch here, the companies details and their meta data. The code is as follows

```
Future<QuerySnapshot<Company>> _getCompaniesSetMetadata() async {
  companiesMetadata = (await _getCompaniesMetadata()).data();
  return FirebaseFirestore.instance
      .collection('Companies')
      .withConverter<Company>(
        fromFirestore: (json, _) => Company.fromJson(json.data()!),
        toFirestore: (company, _) => company.toJson(),
      )
      .get();
}

Future<DocumentSnapshot<CompaniesMetadataClass>> _getCompaniesMetadata() {
  return FirebaseFirestore.instance
      .collection('Metadata')
      .doc('CompaniesMetadata')
      // .get();
      .withConverter<CompaniesMetadataClass>(
          fromFirestore: (json, _) =>
              CompaniesMetadataClass.fromJson(json.data()!),
          toFirestore: (companyMetadata, _) => companyMetadata.toJson())
      .get();
}
```

The whole screen is wrapped with a RefreshIndicator so as to provide Pull to Refresh Functionality so as to update the data whenever required with just a pull. The code is

```
return RefreshIndicator(
  onRefresh: () {
    return Future(() {
      setState(() {});
    }); // Future
  },
```

The on campus and off campus companies are fetched together from the Firebase as they are stored in a single collection with a differentiator “Offer Type” which is “On Campus” for on campus companies and vice versa. The separation of companies’ code is

```
List<Company> companies =
    snapshot.data!.docs.map((doc) => doc.data()).toList();
List<Company> onCampusCompanies = companies
    .where((element) => element.offerType == "On Campus")
    .toList();
List<Company> offCampusCompanies = companies
    .where((element) => element.offerType == "Off Campus")
    .toList();
```

5) Search Bar

This allows us to search in all the available information along with filters, so that it is easy to find the opportunity we are looking for in a seamless way.

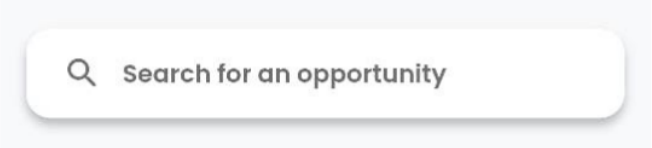


Fig 3.2.3 – Search Bar

When clicked we are navigated to the search screen with a filter option. The fetched firebase data is passed through to the page. The code is as follows

6) On Campus Opportunities

We can see all the companies that come to college along with all the information available about them in the form of tiles in a list view. Each tile contains the role name, company name, the type of role, last date of application for the role and the package.

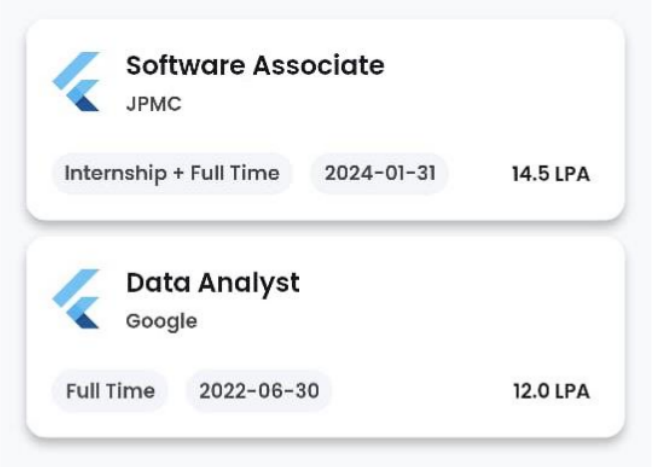


Fig 3.2.4 – On-Campus Opportunity Tiles

Each company tile when clicked takes us to On Campus Company Screen in which we can see a more detailed view of the company. The code of each tile is as follows

```
itemBuilder: (context, index) => OnCampusCompanyTile(
  id: onCampusCompanies[index].id,
  companyName: onCampusCompanies[index].companyName,
  companyType: onCampusCompanies[index].companyType,
  roleName: onCampusCompanies[index].roleName,
  roleType: onCampusCompanies[index].roleType,
  aboutTheFirm: onCampusCompanies[index].aboutTheFirm,
  jobDescription: onCampusCompanies[index].jobDescription,
  skillset:
      onCampusCompanies[index].skillset as List<String>,
  processTimeline:
      onCampusCompanies[index].processTimeline,
  previouslyPlacedContacts:
      onCampusCompanies[index].previouslyPlacedContacts,
  experienceTilesInfo:
      onCampusCompanies[index].experiences,
  faqs: onCampusCompanies[index].faqs,
  lastDate: onCampusCompanies[index].lastDate,
  package: onCampusCompanies[index].package,
  linkToApply: onCampusCompanies[index].linkToApply,
  driveLink: onCampusCompanies[index].driveLink,
  eligibility: onCampusCompanies[index].eligibility,
  offerType: onCampusCompanies[index].offerType,
), // OnCampusCompanyTile
itemCount: onCampusCompanies.length,
```

7) Off Campus Opportunities

In this section we can see the companies that do not come to the college but are available for everyone to apply and are verified by the college faculty that they are legit. The off-campus companies are shown as tiles in a horizontal list of tiles. Each tile contains a bit of less information than on campus as the college is not associated with the company in providing the opportunity. Each tile contains the role name and company name.

Fig 3.2.5 – Off-Campus Opportunity Tiles

Each tile is built using the following code and when clicked on it takes us to Off Campus Company Screen where there is a detailed view of the company and role.

8) Search Screen

This screen facilitates students to search all the available on campus and off campus companies. We have text input being taken in the search bar which allows students to type their query like role name, skill, company name, etc. It also has a filter option in the end which when clicked gives us a pop up with all the available options of filtering the companies. The code of this screen is present in companies_searcg_screen.dart file.

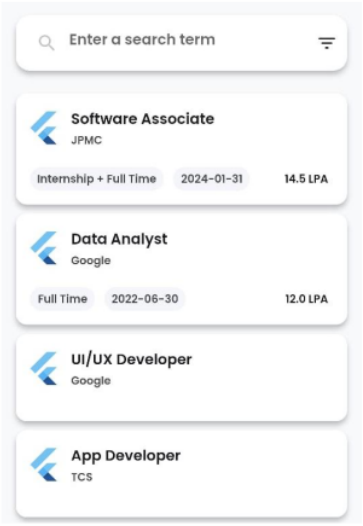


Fig 3.2.6 – Search Screen

Whenever the student types something in the search bar the following function is called and the list of companies being shown is refreshed according to the input. The search is done using fuzzy logic and probability using a third-party package “fuzzywuzzy”. Also, the tiles here are common for both on campus and off campus and are implemented using SearchCompanyTile which is present in search_company_tile.dart file.

The function called when text is types in the search bar is as follows

```
onSearchTextChanged: (searchText) {
  setState(() {
    if (searchText.length >= 3) {
      Map<String, int> _weightedSearchMap = {};
      companiesSearchFilter
          .filterCompanies()
          .forEach((companyId, companySearchText) {
        int _weightedRatio =
            weightedRatio(searchText, companySearchText);
        if (_weightedRatio >= 50) {
          _weightedSearchMap[companyId] = _weightedRatio;
        }
      });
      var _weightedSearchList =
          _weightedSearchMap.entries.toList();
      _weightedSearchList
          .sort((a, b) => b.value.compareTo(a.value));
      searchedCompaniesIndex =
          _weightedSearchList.map((e) => e.key).toList();
    } else {
      searchedCompaniesIndex =
          companiesSearchFilter.filterCompanies().keys.toList();
    }
  });
},
```

Whenever the filter is changed the following function is called

```
onFilterChanged: () {
  setState(() {
    searchedCompaniesIndex =
        companiesSearchFilter.filterCompanies().keys.toList();
  });
```

The search bar is implemented in search_bar.dart file. The functions that are passed from the previous file are called here to get the functionality. The filter button is implemented in a separate file search_filter.dart.

The filter button when clicked gives us a pop up with all the options available. With respect to the type of the input there are three types of components in the filter dialog. The following are the different kinds of filters available with both initial and final states.

Fig 3.2.7 – Filter Options

All the chips being shown are implemented using a custom class “Chip” and “Filter”. Company Type, Role Type and Offer Type are created using this and the function being called when the chip is tapped while enabled or disabled is

```
selected: widget.selectedChips.contains(widget.chipsList[i].label)
    ? true
    : false,
onSelected: (bool value) {
  setState(() {
    widget.chipsList[i].isSelected = value;
    if (value) {
      widget.selectedChips.add(widget.chipsList[i].label);
    } else {
      widget.selectedChips.remove(widget.chipsList[i].label);
    }
  });
```

The package filter, the slider is created using the widget PackageSlider. It looks at the highest package available in the available companies and automatically sets its highest to the same. The students can slide both the sides to create a range of packages available for them. The code is as follows

```
return RangeSlider(
  values: widget.currentRangeValues,
  max: widget.packageMax.toInt() + 1,
  divisions: widget.packageMax.toInt() + 1,
  labels: RangeLabels(
    widget.currentRangeValues.start.round().toString(),
    widget.currentRangeValues.end.round().toString(),
  ), // RangeLabels
  onChanged: (RangeValues values) {
    setState(() {
      widget.currentRangeValues = values;
      widget.onRangeChanged(values.start.toInt(), values.end.toInt());
    });
  },
); // RangeSlider
```

The search filters are created using the widget LittleSearch. Initially we won't be seeing any values, but when we start typing, the respective field's matching elements will be shown as chips and then can be interacted with. Company Name, Role Name and Skillset are implemented using this. The filter widget is itself modified to fit this use case as follows

```
Filter(
  chipsList: widget._chipsList
      .where((element) => element
          .toLowerCase()
          .contains(widget._controller.text.toLowerCase()))
      .map((e) => Chip(label: e, isSelected: false))
      .toList(),
  selectedChips: widget._selectedChipsList,
), // Filter
```

We have two buttons along with these, the clear and apply whose functionalities are as follows

```
TextButton(
  child: const Text("Clear"),
  onPressed: () {
    setState(() {
      _companyNameController.clear();
      _roleNameController.clear();
      _skillsetController.clear();
      widget.companiesSearchFilter.clearFilter();
    });
  },
), // TextButton
TextButton(
  child: const Text("Apply"),
  onPressed: () => Navigator.pop(context),
) // TextButton
```

9) On Campus Company Screen

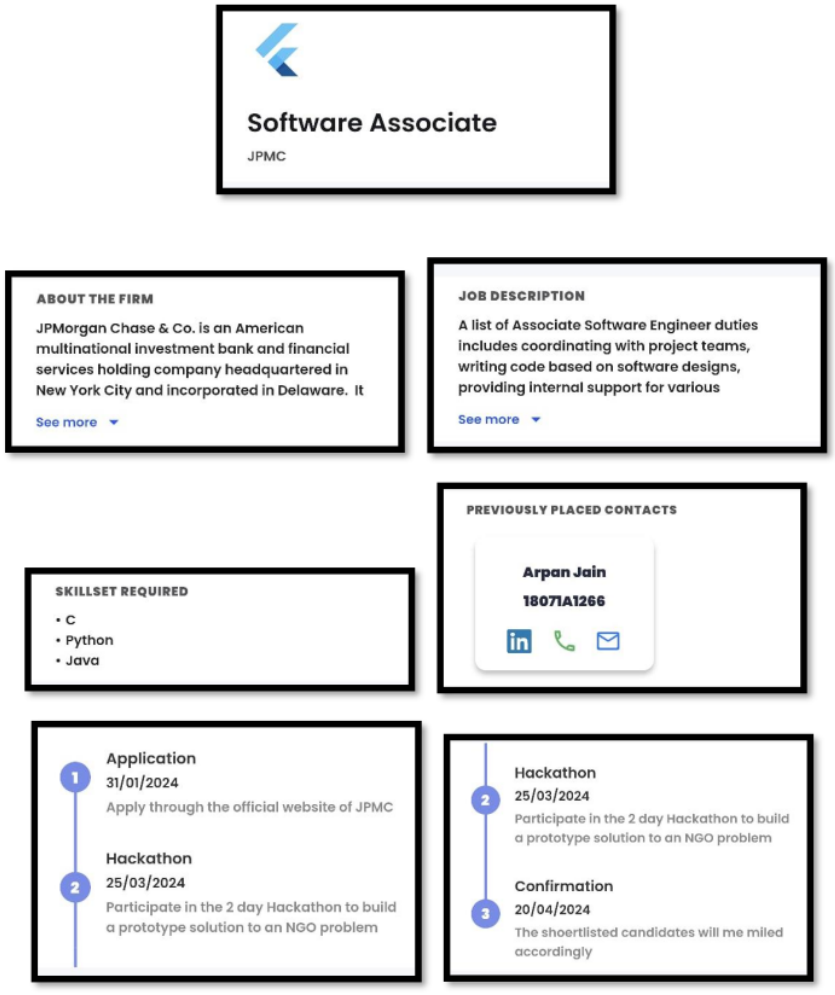
This is an in-detail screen of On Campus Company, which appears when one of the On Campus Company Tile is tapped on the Companies Screen. It has a lot of sections which are implemented separately in many files so as to maintain isolation. The main code is present in on_campus_company_screen.dart file. Let us go through each of the components.

On the top we have two buttons, one which takes us back to the companies' screen and the other one is a drive link button which opens an in-app browser and takes us to the location of resources available about the role of that company. The functionality is as follows

```
onPressed: () {
  Navigator.pop(context);
},
```

```
onTap: () async {
  if (company.driveLink != null) {
    final url = company.driveLink!;
    if (await canLaunchUrlString(url)) {
      await launchUrlString(url);
    }
  }
```

Then we have the heading part of the screen which contains the company name and role name. It is implemented using OnCampusHeading widget and it is present in on_campus_heading.dart file.



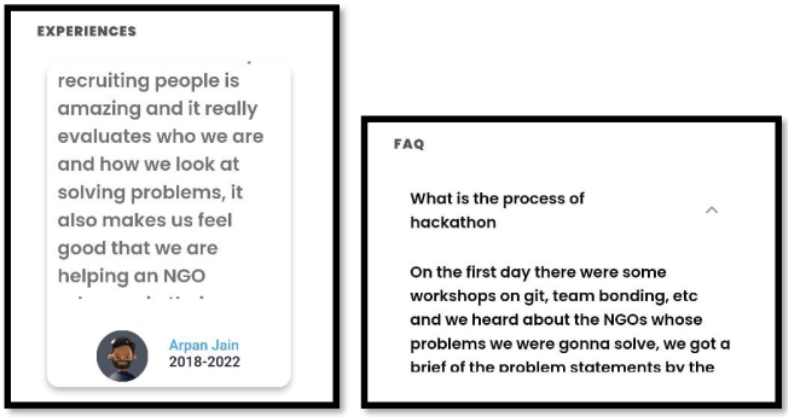


Fig 3.2.8 – On-Campus Company Screen Widgets

Then we have “About The Firm” and “Job Description” which basically give information about the company and the role. As these texts can be a lot bigger, they are constrained to 4 lines and a “See more” button is provided which when clicked will enlarge the tile and show the whole text. Both these are implemented using a custom widget MyExpandableCard which is present in on_campus_my_expandable_card.dart file.

The functionality of “See more” is as follows, the expanded property is set to true or false accordingly and the UI updates itself.

```
return AnimatedContainer(
  duration: const Duration(seconds: 1),
  curve: Curves.fastOutSlowIn,
  child: widget.isExpanded ? widget.expandedChild : widget.collapsedChild,
); // AnimatedContainer
```

Then we have the “Skillset Required” space, which gives a bulleted list of the skills required for the job opportunity. We take the data from a list and put it on UI in such a manner that it appears as bullet points.

The Process timeline is a way of showing the whole process of the company's hiring process in steps in which each contains a heading, date and description of the step. It is implemented using ProcessTimeline widget which is present in on_campus_process_timeline.dart file which uses a third part package "timeline_tile". It uses its own model to store data.

Then we have a "Previously Placed Contacts" which contains details about the alumni who got placed in the same company and role. They are shown as a horizontal list view in tiles and each tile contains the alumni's name, roll number and three icon buttons which correspond to their LinkedIn, phone number and email which when clicked take the student directly to the respective platform. The code for this is present in on_campus_previously_placed_contact_tiles.dart file and it is implemented using the PreiouslyPlacedContactTiles widget.

Similar to "Previously Placed Contacts" the "Experiences" are feedback given by alumni of the college about the role or company. It is also a horizontal List view which has tiles which contain the experience, name and batch of the alumni. This can help students understand what to expect from the organization. It is implemented using ExperienceTiles widget which is present in on_campus_experience_tiles.dart file.

At the end, we have the "FAQ" section which contains frequently asked questions as a list of expandable cards. These are collected from people who have attended for the company or role before and are cumulated to assist the students with general questions. It is implemented using FAQTiles widget which is present in on_campus_faq_tiles.dart file.

The "APPLY NOW" button hovers over all the items and is always visible to the student. When clicked it basically takes the student to the registration page of the role

of the company which will allow them to apply to it instantly if they want to seize the opportunity.

The function called when it is clicked is

```
onTap: () async {
  final url = company.linkToApply;
  if (await canLaunchUrlString(url)) {
    await launchUrlString(url);
  }
```

10) Off Campus Company Screen

This is a similar screen to that of On Campus Company Screen, but is for the Off Campus Companies. As the college doesn't have much information on these companies and roles it has rather limited sections when compared to On Campus Screen. The code is present in off_campus_company_screen.dart file.

The contents are as follows

Fig 3.2.9 – Off-Campus Company Screen Widgets

We have the back button which takes us to the companies' screen, its code is as follows

```
onPressed: () {
  Navigator.pop(context);
},
```

Then we have the same heading as that of On Campus one which uses the OnCampusHeading widget. It shows the company name and role name. The code is present in on_campus_heading.dart file.

Then we have a “Description” section in which all the information known about the opportunity can be shown or seen.

The “APPLY NOW” button’s functionality is same as that of On Campus one, when clicked it will redirect the student to the website in which the opportunity is showcased. The function which is called is

11) News Feed Screen

This screen is used as a direct connect between T&P Faculty and students. All the polls and posts furnished by the faculty can be seen by the students and they can react to them.

The code of this screen is in newsfeed_screen.dart file. It does import newsfeed_poll_widget.dart and newsfeed_post_widget.dart to use the poll and post widgets in this screen.

The post widget has a title, link, posted by information and a button which allows students to acknowledge T&P that they have read the post.

The poll widget has a query and options related to it and the students can choose any one of those options as their answer and T&P can directly see their answers in the website.

The whole data is first fetched from Firebase and then the screen is populated with the polls and posts chronologically. The function used to fetch the data is as follows:

```
_controller = StreamController<QuerySnapshot>();

_subscribedData = FirebaseFirestore.instance
    .collection("Newsfeed")
    .orderBy("timestamp", descending: true)
    .snapshots()
    .listen((data) {
  _controller?.add(data);
});
```

Post

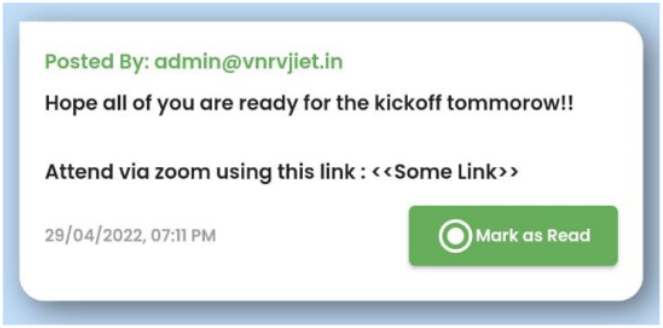


Fig 3.2.10 – Newsfeed Post Widget

The "Mark as Read" button's functionality is as follows, it basically sends an update request to Firebase which fills in the data of "readBy" for that particular post with the roll No of the student who marked it as read

```
onPressed: () {
  if (isRead!) {
    return;
  }
  FirebaseFirestore.instance
      .collection('Newsfeed')
      .doc(widget.post.id)
      .update({
    'readBy': FieldValue.arrayUnion([userRollNo])
  }).then((_) {
    setState(() {});
  });
},
```

Poll

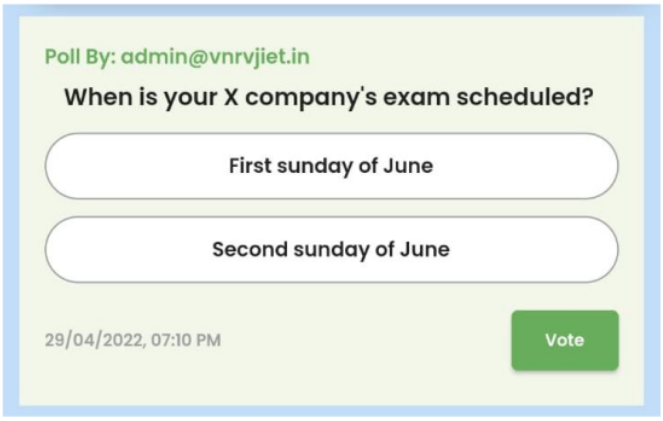


Fig 3.2.11 – Newsfeed Poll Widget

Whenever the student selects an option and clicks vote, the information is populated and sent to firebase also updating the UI to reflect how much percentage of people selected which option. The functionality is as follows

```
onPressed: () async {
  FirebaseFirestore.instance
      .collection('Newsfeed')
      .doc(widget.poll.id)
      .update({
    'options.$optionSelectedKey.selectedBy':
        FieldValue.arrayUnion(
            [userRollNo])
  }).then((_) {
    Navigator.of(context).pop();
  });
},
```

Once voted the student cannot interact with the poll but just look at the results, the status is checked using

```
bool checkIfAlreadyVoted(Map<String, dynamic> pollOptions) {
  String result = '-1';
  pollOptions.entries.toList().forEach((e) {
    if (e.value["selectedBy"].contains(userRollNo)) {
      result = e.key;
    }
  });

  if (result != '-1') {
    optionSelectedKey = result;
  }
  return result != '-1';
}
```

12) Resources Screen

In this screen all the resources that are made available by the T&P Cell are shown as tiles in a list view. Each tile shows the title and a small description about the resource and when clicked on the tile we are navigated to the Drive location where the resource is present. The code of this screen is present in resources_screen.dart and each tile is build using a custom class “ResourceTile” which is present in the same file.

Fig 3.2.12 – Resource Tile

The data is fetched via Firebase using the following code

```
final firebaseInstance = FirebaseFirestore.instance
    .collection('Resources')
    .withConverter<ResourceClass>(
      fromFirestore: (snapshot, _) => ResourceClass.fromJson(snapshot.data()!),
      toFirestore: (resource, _) => resource.toJson(),
    );
```

The function that helps us open the Drive location is as follows, it uses the url_launcher package

```
onTap: () async {
  final url = resource.link;
  if (await canLaunch(url)) {
    await launch(url);
  }
},
```

13) Profile Screen

This screen contains utilities for the student, it gives them access to some of the features with which they can make their experience better in the app. The code is present in profile_screen.dart file and it contains an import for each of the feature. Each feature tile is implemented using a custom class ProfileItem which is present in the same file.

The initial thing is that they can see their Username and Roll No on the top of the Screen which is fetched from Firebase.

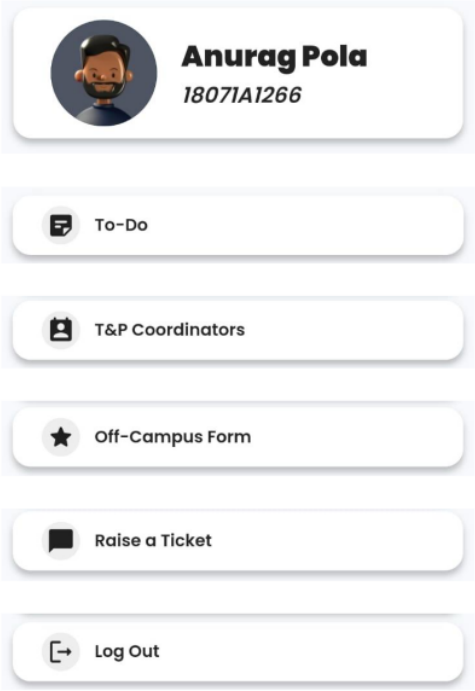


Fig 3.2.13 – Profile Screen Widgets

Then we have a To-Do feature, which is an in-app list of things students can add and track to get ready for placements. The code is in to_do_screen.dart file.

We can add items using the "+" button in the screen, it gives us a pop-up which takes the text of the to-do to be added, and when submitted it is added to the list. It is also added to the local storage of the mobile so that the details persist even when the application is closed and reopened.

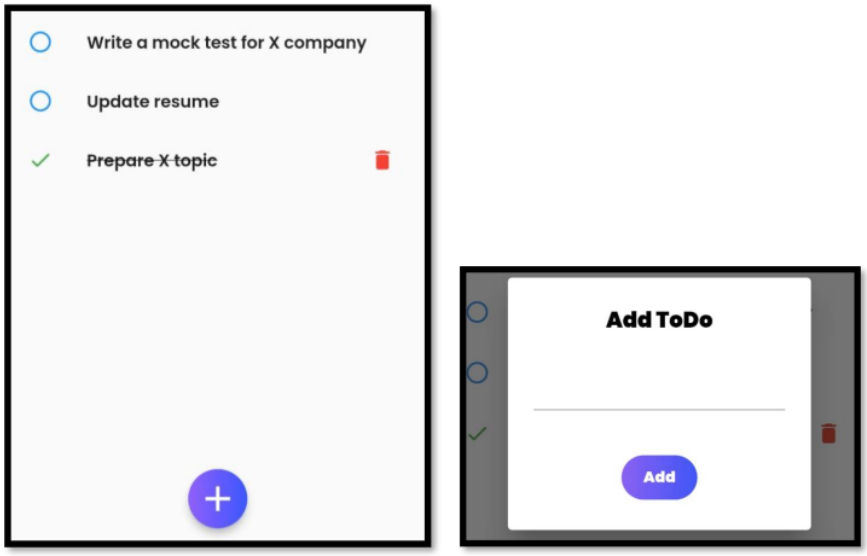


Fig 3.2.14 – To Do Screen Widgets

The list item can be clicked upon to mark it done, which then appears as a crossed off list item, and also gives us an option to delete that item.

We can see that all functions are being done by “databaseHelper” which is imported from local_database.dart file which has all the functions related to adding, crossing off and deleting the list item from the local storage of the mobile.

```
// Fetch Operation: Get all todo objects from database
Future<List<Map<String, dynamic>>> getTodoMapList() async {
  Database db = await tododatabase;
  var result = await db.query(todoTable);
  return result;
}

Future<List<Map<String, dynamic>>> getDoneMapList() async {
  Database db = await donedatabase;
  var result = await db.query(doneTable);
  return result;
}
```

Then we have the T&P Coordinators, which basically gives the information about all the T&P Coordinators of every department with the following details – Name, Department, Email ID and Phone Number. The code is present in tnp_coordinators_screen.dart file.

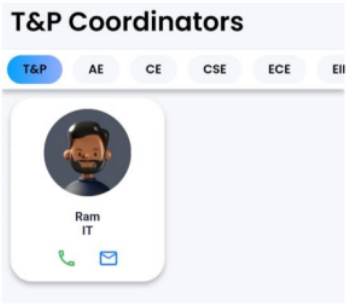


Fig 3.2.15 – T&P Coordinators Screen Widgets

We have a strip which contains all the available branches present in the college as chips. When clicked on each of the chip the respective T&P Faculty will be visible. Each T&P faculty is shown in a card like item which has an image, name of the faculty, their department and two icons, one being phone which directly allows them to call the faculty hen clicked and the other one is email which also directly allows them to email to the faculty. Both these functions are done using any suitable app available on the device being used.

All the data is fetched from Firebase via

```
Expanded(
  child: GridView.builder(
    gridDelegate:
        const SliverGridDelegateWithFixedCrossAxisCount(
      crossAxisCount: 2,
      mainAxisExtent: 230,
    ), // SliverGridDelegateWithFixedCrossAxisCount
    itemCount: nameSwitchedOn == "T&P"
        ? tnpcoordinators.length
        : tnpcoordinators
            .where((element) =>
                element.department == nameSwitchedOn)
            .length,
    itemBuilder: (context, index) => TnPCoordinatorTile(
      tnpcoordinator: nameSwitchedOn == "T&P"
          ? tnpcoordinators[index]
          : tnpcoordinators
              .where((element) =>
                  element.department == nameSwitchedOn)
              .toList()[index],
    ), // TnPCoordinatorTile
  ), // GridView.builder
), // Expanded
```

The filtering of the tiles of faculty based on chips(which are implemented using TnPBranchChip class present in the same file) is done using this

Then we have the Off Campus Form which allows the students to suggest any missing company in the database to the faculty. The code for this page is present in off_campus_form_screen.dart file. It basically takes input using a form of all the necessary details that would allow the faculty to enquire about the company and if legit add it to the database.

The details taken are Name of the company, Role being offered, the package provided in LPA and the URL through which we can apply to the company or reach it. By filling all the details and hitting the “Send Information to T&P” button, the details will be furnished on T&P website and a snack bar of success is shown. All the form elements are built using a custom class EachQuestion which is present in the same file.

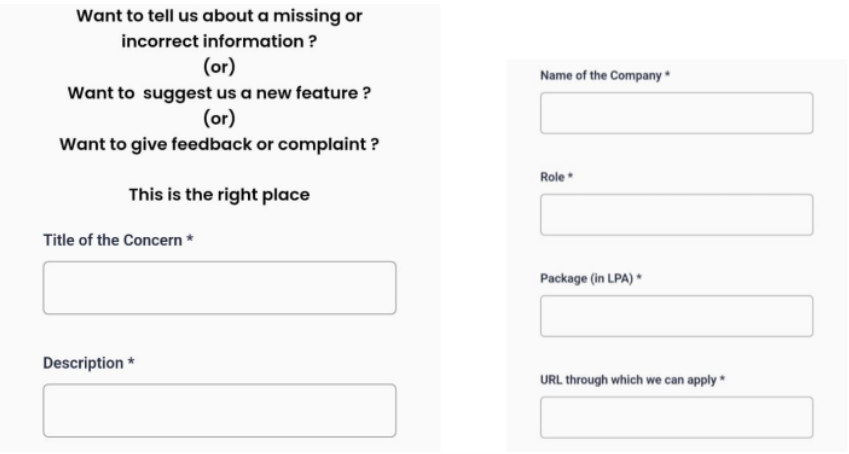


Fig 3.2.16 – Ticket and Off Campus Opportunity Screens

Then we have the "Raise a Ticket" screen which allows students to raise any concern regarding placements to T&P Cell. The request along with the Roll NO of the student who sent it is visible to the faculty which can be used to contact him/her/them if they need more clarification on the issue. The code is present in raise_ticket_screen.dart file.

It basically takes only two inputs, the title and the description of the concern and when "Raise a Ticket" button is clicked the concern is raised on the website.

Last but not the least, we have the Logout button which enable the students to logout from their account in the application. It's code is presnt in Profile Screen itself as it is a single liner that send a logout request to firebase and the token that is used to authenticate him/her/them is destroyed, thus logging them out.

```
await FirebaseAuth.instance.signOut();
```

3.3 WEBSITE IMPLEMENTATION

1) Login Screen

The login screen is almost similar to that of in mobile application. It takes two inputs mail, password and tries to authenticate the user. But here we have an extra check of if the user is an admin or not as only admins can access the website as it has all the controls that could be used to furnish information in the mobile application.

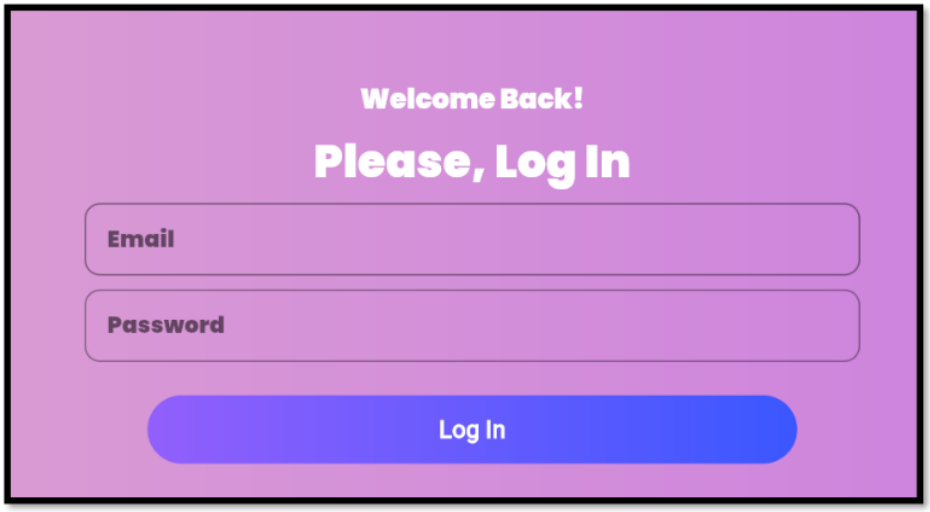


Fig 3.3.1 – Login Screen

The “Log In” button’s functionality is as follows

```
UserCredential user = await auth.signInWithEmailAndPassword(
  email: _email,
  password: _password,
);
```

2) Main Screen

This is the first screen we see when logged in. In this screen we first see the details of who logged in and then all the available options. The code is present in tabs_screen.dart file.

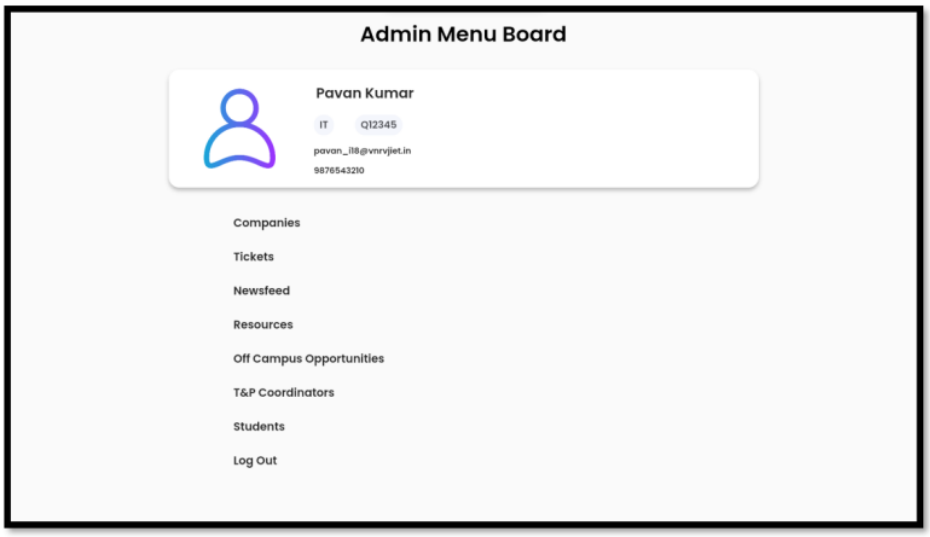


Fig 3.3.2 – Main Screen

Each option navigates us to its respective screen with the help of map

```
final Map<dynamic, dynamic> _tabsRouteMap = {
  'Companies': '/companies',
  'Tickets': '/tickets',
  'Newsfeed': '/newsfeed',
  'Resources': '/resources',
  'Off Campus Opportunities': '/off-campus-opportunities',
  'T&P Coordinators': '/admins-page',
  'Students': '/students-page'
};
```

3) Companies Screen

This is the screen which is crucial as it manages the companies which appear on the students' side, the faculty can also view both the on campus and off campus with the help of a toggle button in the screen. It also has the search bar with which we can search the companies regardless of type. It is implemented in companies_screen.dart.

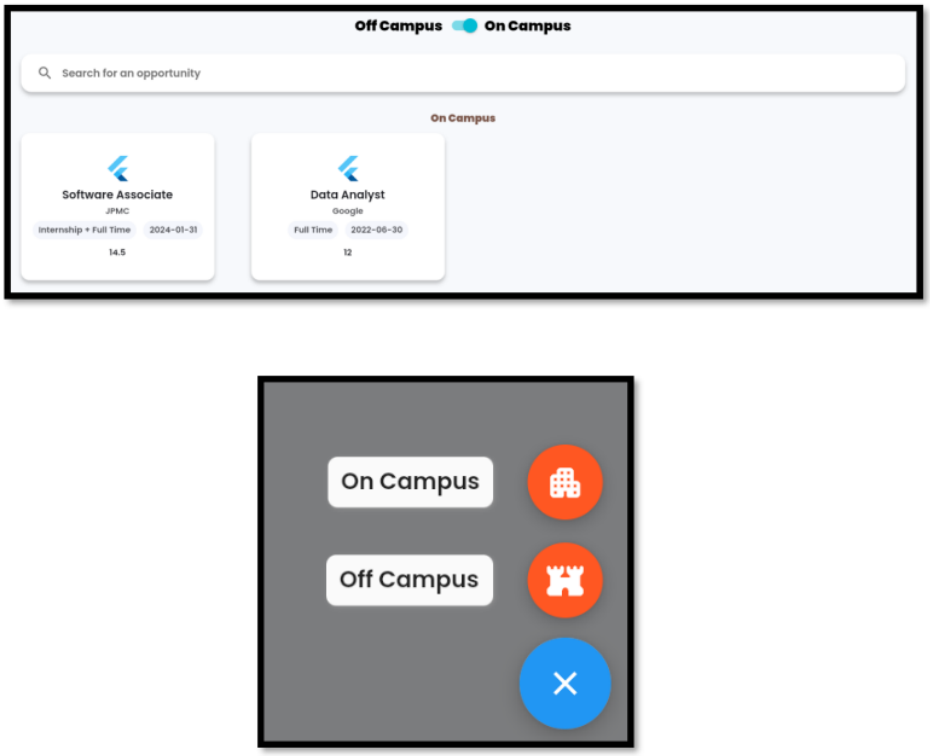


Fig 3.3.3 – Companies Screen

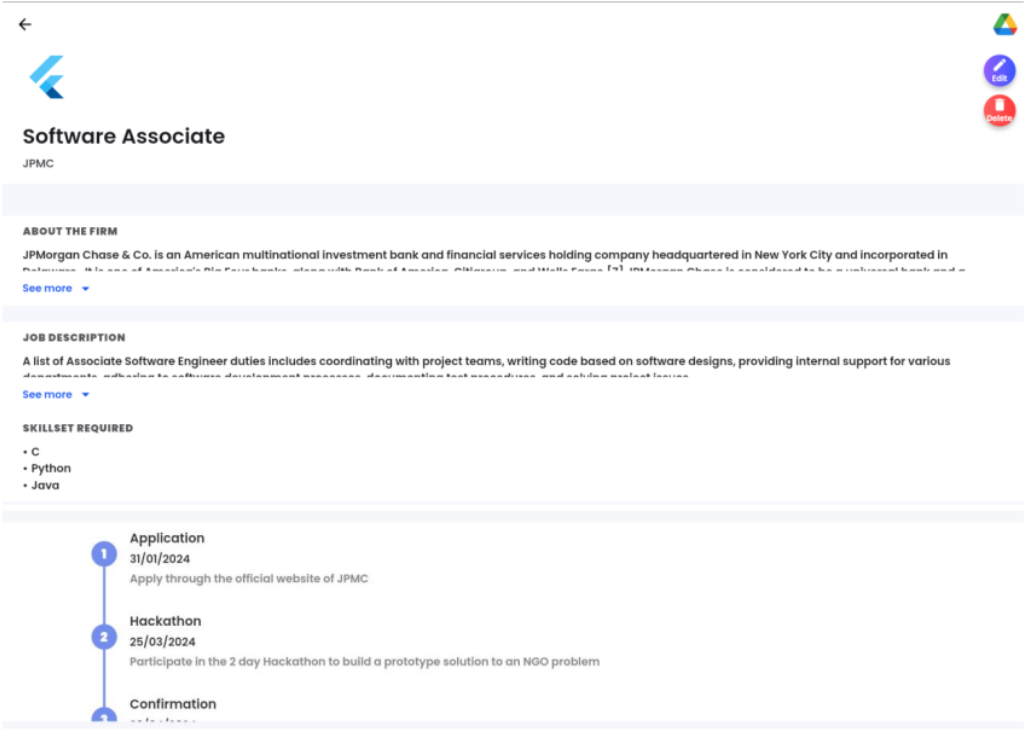
All the other code is similar to that of in mobile application and is implemented in the following files using the respective widgets. The on-campus company tiles are implemented using OnCampusCompanyTile written on on_campus_company_tile.dart

file. The off-campus tiles are implemented using OffCampusCompayTile written in off_campus_company_tile.dart file.

We have two buttons in the bottom of Company Screen with which we can add a new On Campus and Off Campus company. When clicked both of them take us to the respective form with empty fields and when filled and submitted it populates the companies.

4) On Campus Company Screen

The On Campus tile when clicked on will take us to On Campus Company Screen which is implemented using OnCampusCompanyScreen which is present in on_campus_company_screen.dart file. The code is same as that of mobile application with a bit of modifications. The screen is as follows



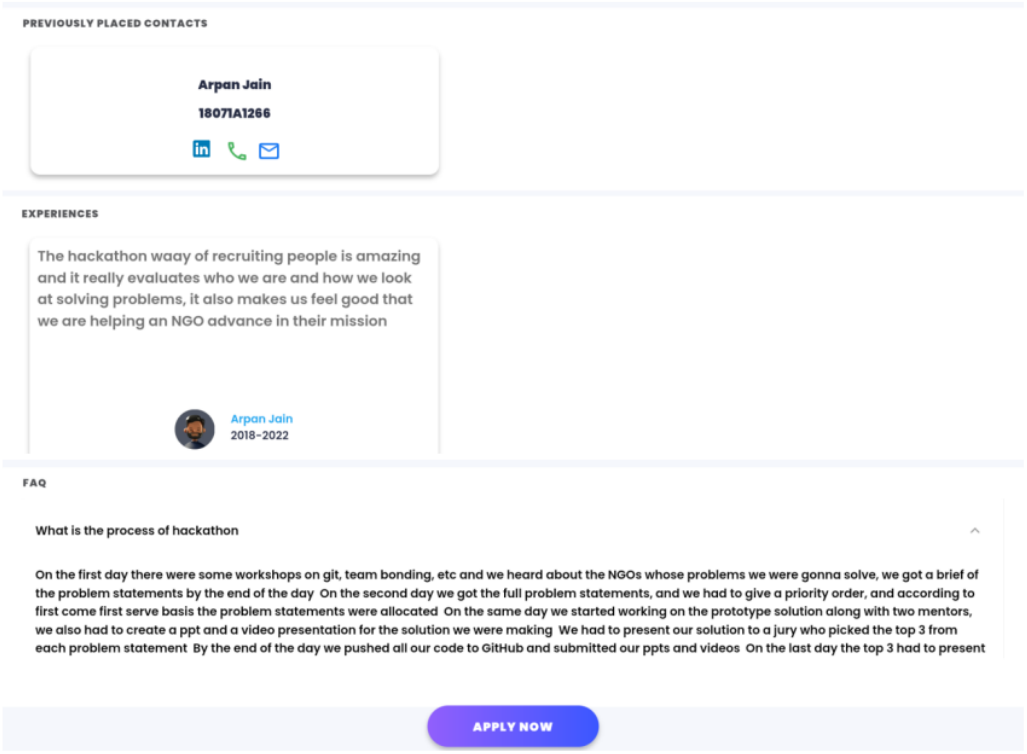


Fig 3.3.4 – On Campus Company Screen

Along with all the sections which are same as that of mobile application we have two extra buttons which one allows us to edit the company details and the other deletes the company after taking a confirmation.

The edit button when clicked takes us to the fully filled On-campus Company Form with the details of the current company. We can edit the details as we need and when submitted the details are updated in Firebase which in turn updates it in the mobile application.

5) Off Campus Company Screen

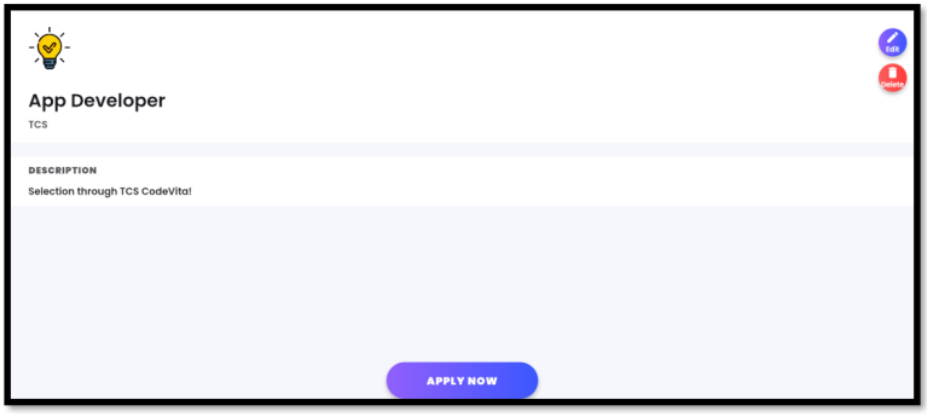


Fig 3.3.5 – Off Campus Company Screen

It is similar to that of in the mobile application with the same edit and delete options available. The code implementation is same as that of app and on campus company screen. It is implemented using OffCampusCompanySCreen which is present in off_campus_company_screen.dart file.

6) Search Screen

It is same as Search screen in mobile application. It is implemented using CompaniesSearchScreen and is present in companies_search_screen.dart. A third-party package fuzzywuzzy is used for searching purposes. The tiles are common for both on campus and off campus which are implemented using SearchCompanyTile which is present in search_company_tile.dart file.

Fig 3.3.6 – Companies Search Screen

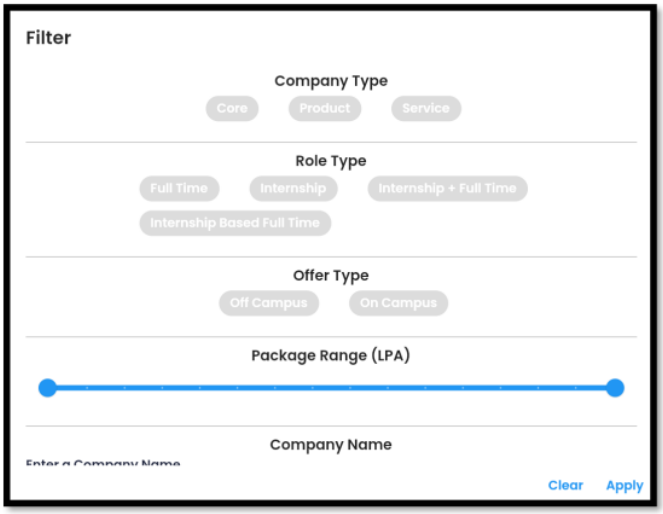


Fig 3.3.7 – Search Filter Dialog

7) On Campus Company Form

This is the form which comes when we either want to create a new On Campus Company or edit an existing one. It has all the fields that are required to create an On Campus Company. It is implemented in on_campus_company_form.dart file. The form takes input of Company Name, Company Type (Dropdown), Role Name, Role Type (Dropdown), About the Firm, Job Description, Eligibility, Package (numeric), Last Date To Apply (Date Picker), Skillset Required (Multi input), Process Timeline (Multi

input), Previously Placed Contact Details (Multi input), Experiences (Multi input), FAQs (Multi input), Link For Applying and Google Drive Link. The form has its own validation and only allows users to submit if nothing is empty. The UI is as follows

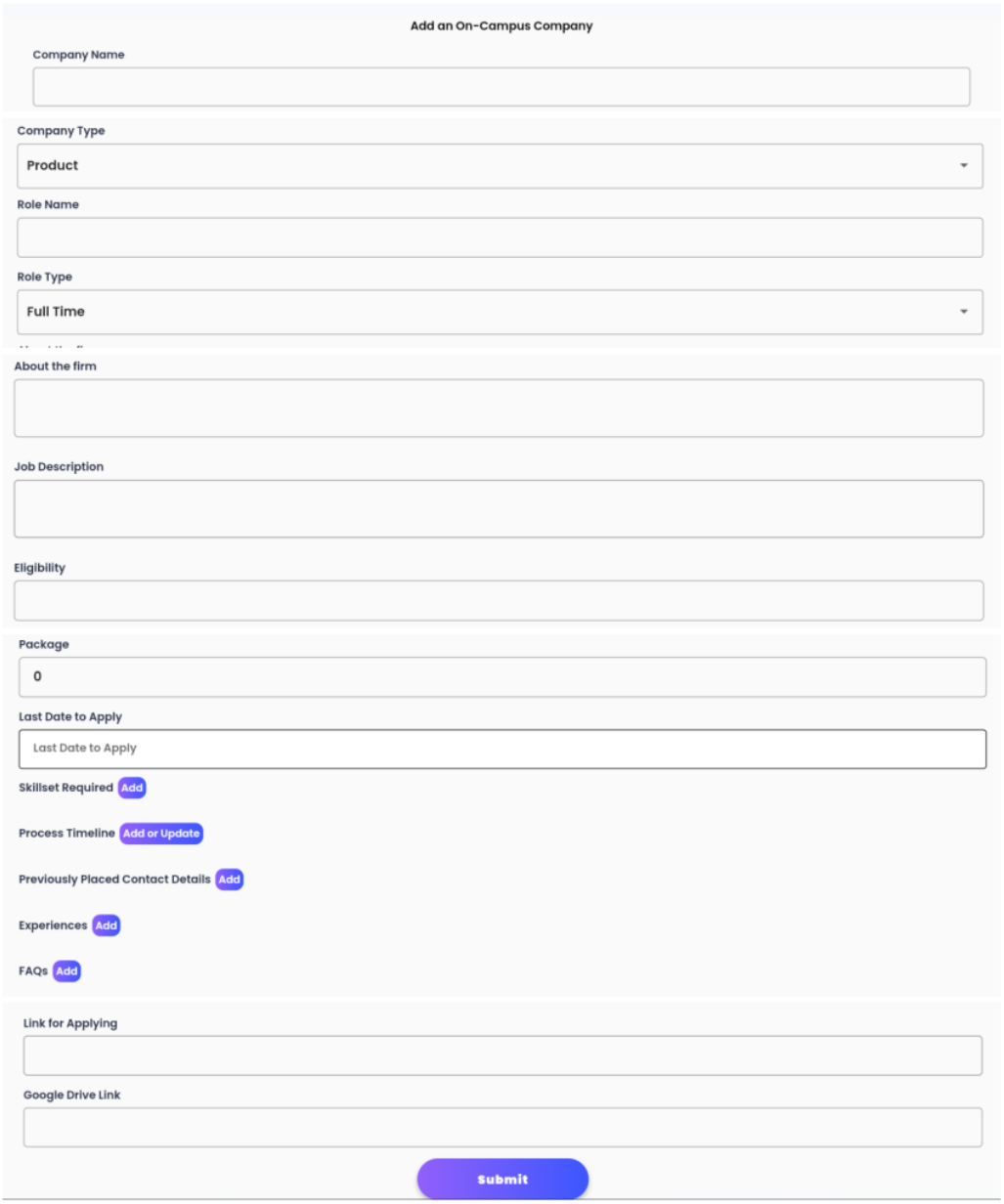


Fig 3.3.8 – On Campus Company Form

The normal string inputs are made using the widget “Field” which is present in the same file.

The dropdowns are made using AppDropDownInput widget which is present in the same file. It takes the options to be shown from Firebase and the faculty can select one of the available options. An example of available options is as follows

For all the multi-input fields, “Complex Field” widget was used which is present in the same file. Here each input format was different, so a separate dialog was made for each kind of input, and based on the input it is processed accordingly and added to the Firebase. Let us look at different dialogs used

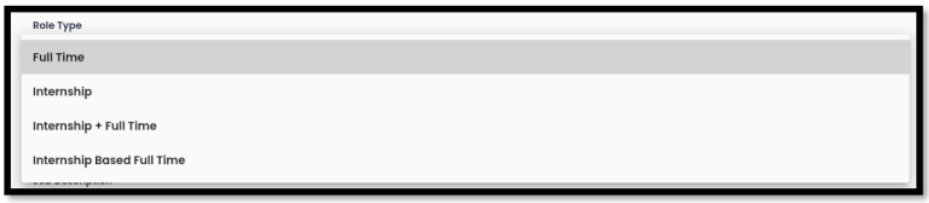


Fig 3.3.9 – Dropdown Input Field

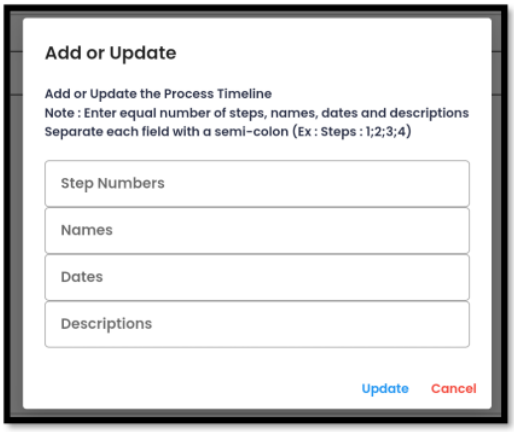


Fig 3.3.10 – Input Dialogs

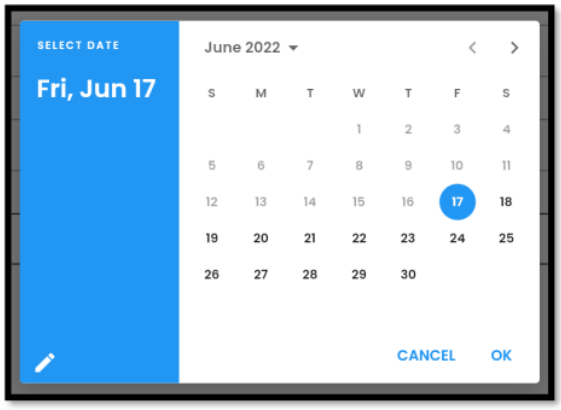


Fig 3.3.11 – Date Picker

For adding a new skill, we have the skillsetDialog widget. It takes only one input – skill. When entered and submitted the skill gets added to the skill list. The skills look as such with a delete button in them.

The process timeline is populated using "processTimelineDialog". It takes four inputs, namely step numbers, names, dates and descriptions. Here each input basically has a

number of inputs separated by semi colons and it has a constraint that all the fields must contain same number of inputs.

For adding a new alumni contact, we have the "previouslyPlacedContactDetailsDialog". It takes input as name, roll number, email, phone number and LinkedIn. Each contact added looks as above after getting added to the list with a delete button.

For experiences, we have the "experiencesDialog". It takes input of experience, name, batch and roll number. Roll number is used as a key and is not shown in the UI. The added experiences look as such along with a delete button.

For adding FAQ's, we have "faqsDialog". It takes input of question and answer and adds it to the FAQ list. The added FAQs look as such with a delete button.

The Last Date to Apply field is also implemented separately using "LastDateToApplyField". It gives a date picker when tapped on and when selected will update to the selected date.

```
final company = Company.fromJson(
  {
    "id": widget.id == ""
        ? DateTime.now()
            .millisecondsSinceEpoch
            .toString()
        : widget.id,
    "companyName": _companyNameController.text,
    "companyType": companyType,
    "roleName": _roleNameController.text,
    "roleType": roleType,
    "aboutTheFirm": _aboutTheFirmController.text,
    "jobDescription": _jobDescriptionController.text,
    "eligibility": _eligibilityController.text,
    "package": double.parse(_packageController.text),
    "lastDate": lastDateToApplyController.text,
    "linkToApply": _applyLinkController.text,
    "driveLink": _googleDriveLinkController.text,
    "skillset": globalSkillset,
    "processTimeline": globalProcessTimeline.toJson(),
    "previouslyPlacedContacts":
        previoslyPlacedContactDetails,
    "experiences": experiences,
    "faqs": faqs,
    "offerType": "On Campus",
  },
); // Company.fromJson
```

```
firebaseInstance.doc(company.id).set(company);

ScaffoldMessenger.of(context).showSnackBar(
  const SnackBar(
      content: Text('Opportunity published successfully.')), // SnackBar
);
await Future.delayed(const Duration(seconds: 2));
Navigator.pop(context);
```

8) Off Campus Company Form

It is similar to that of On Campus Form, but has a smaller number of fields corresponding to Off Campus Screen. It is implemented using OffCampusCompanyForm widget which is present in the file

off_campus_company_form.dart file. It takes input of Company Name, Role Name, Description and Link for Applying.

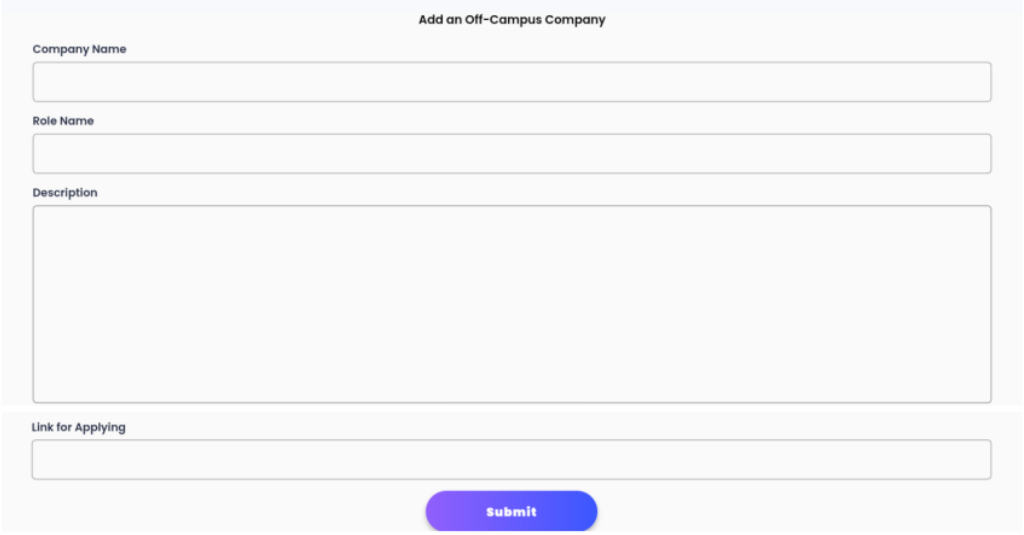


Fig 3.3.12 – Off Campus Company Form

9) Tickets Screen

This screen shows all the tickets raised by the students in a single place. The tickets are shown in a list view with the title, description, information about who posted it and a "Resolved" button which can be clicked when the ticket is resolved and when clicked, it deletes the ticket. The code is present in tickets_screen.dart file.

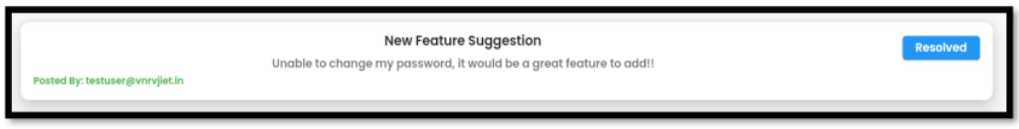


Fig 3.3.13 – Ticket widget

The tickets data is fetched from Firebase using

```
final Stream<QuerySnapshot> _ticketsStream = FirebaseFirestore.instance
    .collection('Tickets')
    .snapshots(includeMetadataChanges: true);
```

The "Resolved" button when clicked will initially give a pop-up dialog which takes confirmation that they really want to delete it, when clicked "Delete" the ticket will be deleted.

10) Newsfeed Screen

This is an information provider screen for its counterpart on the mobile application side. In this screen we can see all the posts and polls which were already posted and also add new polls and posts. The code for this screen is present in newsfeed_screen.dart file. The poll and post are created using custom widgets taken from newsfeed_poll_widget.dart and newsfeed_post_widget.dart.

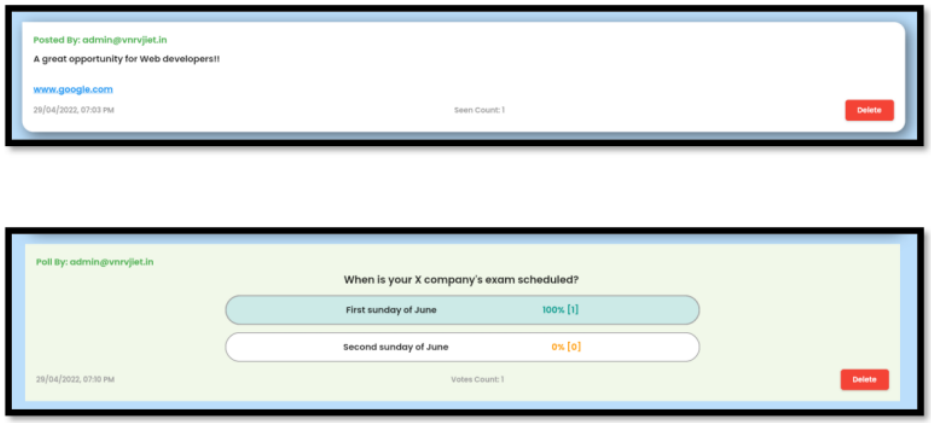


Fig 3.3.14 – Post and Poll Widget

A post has information about who posted it, the post content, a supposed link, the date and time of the post, two buttons, namely seen count and delete. When seen count is clicked, all the students roll numbers who have marked the post as read can be seen along with the count.

A poll has information about who polled it, the question, the options along with the percentage and count of people who voted for that option, time and date, and two buttons, namely vote count and delete. When voted count is clicked, we get the list of roll numbers of student swho polled along with the option they chose

New polls and posts can be created using the speed dial available at the right bottom corner of the screen, it gives us two options one to create a post and the other to create a poll.

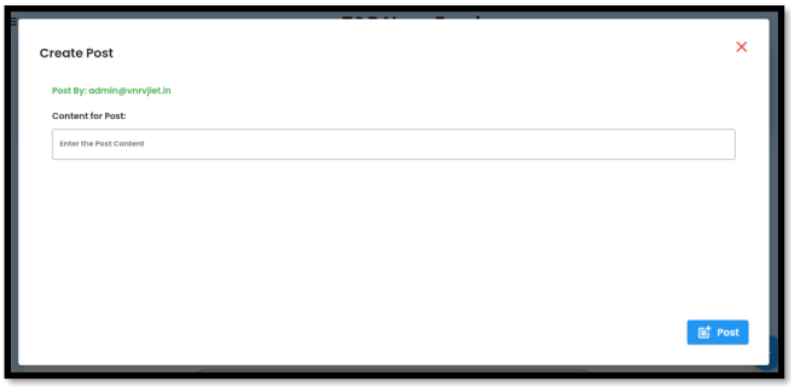


Fig 3.3.15 – Create Post Dialog

Fig 3.3.16 – Crate Poll Dialog

Post is created using the following code, the createPostWidget function is available in create_post_widget.dart file. It gives a pop-up in which we can furnish the details and create a post. Poll is created using the following code, the createPollWidget function is available in create_poll_widget.dart file.

11) Resources Screen

This screen helps the T&P faculty to populate the resources screen counterpart in mobile application. In this all the resources can be seen in a grid view. The code is present in resources_screen.dart file. Each resource is made up of a custom class Resource in the same file.

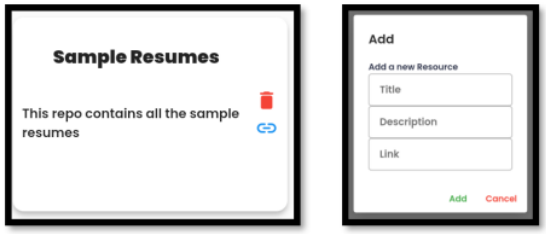


Fig 3.3.17 – Resource Tile and Add Resource Dialog

Each resource contains the title , description, two icon buttons namely link which when clicked takes us to the resource and delete button with which we can delete the resource. We can add new resources using the add button and when clicked a pop-up dialog is shown which takes all the inputs required and when submitted adds a new resource in firebase and reflects the same in both app and website.

The resources are fetched from Firebase using the following code

```
final firebaseInstance = FirebaseFirestore.instance
    .collection('Resources')
    .withConverter<ResourceClass>(
      fromFirestore: (snapshot, _) => ResourceClass.fromJson(snapshot.data()!),
      toFirestore: (resource, _) => resource.toJson(),
    );
```

12) Off Campus Opportunities Screen

Off-campus opportunities the students send to T&P appear here. The information students give is initially populated in Firebase and then is collected and shown here as a list of tiles. The code is present in off_campus_opportunities_screen.dart file.

Fig 3.3.18 – Off Campus Opportunity Tile

All the tiles contain the company name, role name, package, information about who gave the information, two buttons in which one is the link to the companies' application page and the other button is to delete the tile if it is taken care of. The delete button when clicked gives a pop-up and takes confirmation to delete the information

The information is fetched from Firebase as a stream of data via

```
final Stream<QuerySnapshot> _offCampusOpportunitiesStream = FirebaseFirestore
    .instance
    .collection('OffCampusOpportunities')
    .snapshots(includeMetadataChanges: true);
```

13) Admin Screen

It is almost similar to T&P Coordinators Screen in the mobile application. We have a strip of all the available departments on the top and when clicked on a specific department faculty of only that department are shown. The faculty details can be seen in a grid view which contains name, department, email, phone number and a delete button to delete the tile. The code is present in admin_screen.dart file and the department chips, strip, grid view is created using AdminChip, AdminChipBar and AdminGrid custom widgets respectively.

New faculty details can be added using the add button In the bottom right corner which gives a pop-up and takes all the required information and when submitted creates a new admin

Fig 3.3.19 – Add Admin/Faculty Dialog

The Add Admin button when clicked will execute the following function which is implemented in add_delete_admin_dialog.dart file and manage_users_functions.dart file.

```
Future<Map<String, dynamic>> registerAdminWithEmailPassword(
    CollectionReference adminDataCollectionRef,
    CustomAdminUser adminUser) async {
  Map<String, dynamic> _response = {"status": "SUCCESS", "body": []};
  try {
    Map<String, dynamic> _secondaryAppResponse = await secondaryAppGetter();
    if (_secondaryAppResponse["status"] == "ERROR") {
      throw (_secondaryAppResponse["body"]);
    }
    FirebaseApp secondaryApp = _secondaryAppResponse["body"];

    Map<String, dynamic> _resgisterUser =
        await registerAdmin(adminUser, secondaryApp);
    if (_resgisterUser["status"] == "ERROR") {
      throw (_resgisterUser["body"]);
    }
    adminUser = _resgisterUser["body"];
    Map<String, dynamic> _adminCreateDB =
        await addAdminToDB(adminDataCollectionRef, admin: adminUser);
    if (_adminCreateDB["status"] == "ERROR") {
      throw (_adminCreateDB["body"]);
    }
    _response["body"] = adminUser;
  } catch (e) {
    _response["status"] = "ERROR";
    _response["body"] = e;
  }
  return _response;
}
```

14) Students Screen

The students' screen is similar to that of Admins Screen but for students instead of faculty. Here the details of all the students who were enrolled by the T&P cell can be seen. The students are seen as tiles which are implemented using StudentTile widget which is present in student_tile.dart file. Every student tile has their name, mail id and roll number.

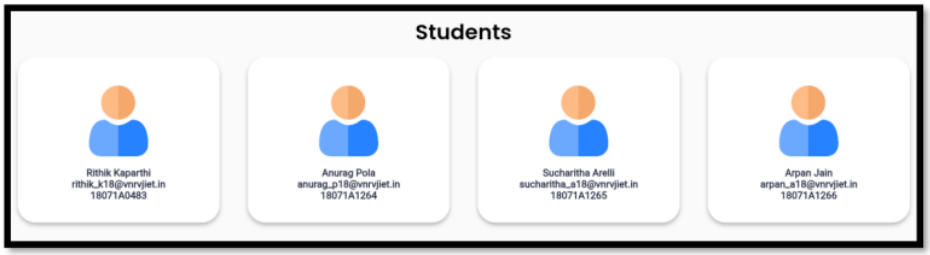


Fig 3.3.20 – Students Screen

Fig 3.3.21 – Action Summary Table

We can both add and delete new students using the Speed Dial in the screen. When either of them is pressed we get to see a google sheet's address, which when populated based on what is selected adds or deletes them. The status of added or deleted is shown in a table dorm as shown below for both the processes.

The functions used to manage students are as follows

```
Future<Map<String, dynamic>> registerStudentWithEmailPassword(
    QuerySnapshot<Object?> metadataQuerySnapshot,
    CollectionReference studentDataCollectionRef) async {
  Map<String, dynamic> _response = {"status": "SUCCESS", "body": []};
  try {
    Map<String, dynamic> _secondaryAppResponse = await secondaryAppGetter();
    if (_secondaryAppResponse["status"] == "ERROR") {
      throw (_secondaryAppResponse["body"]);
    }
    FirebaseApp secondaryApp = _secondaryAppResponse["body"];
    String createUsersAppScriptUrl = metadataQuerySnapshot.docs
        .firstWhere((element) => element.id == "AppScriptUrl")
        .get("createUsers")
        .toString();
    String createUsersSheetsUrl = metadataQuerySnapshot.docs
        .firstWhere((element) => element.id == "SheetsUrl")
        .get("createUsers")
        .toString();
    Uri createUsersWebAPI = Uri.parse(
        "$createUsersAppScriptUrl?CreateUsersSheetUrl=$createUsersSheetsUrl");
    final response = await http.get(
      createUsersWebAPI,
    );
    if (response.statusCode != 200) {
      throw ("${response.statusCode} - ${response.body}");
    }
final body = json.decode(response.body);
if (body["status"] != "SUCCESS") {
  throw ("${body["status"]} - $body");
}
for (var eachUser in body["data"]) {
  CustomStudentUser user = CustomStudentUser(
      name: eachUser["name"],
      email: eachUser["email"],
      password: eachUser["password"],
      rollNo: eachUser["id"],
      uid: null);
  Map<String, dynamic> _responsePerUser = {
    "status": "SUCCESS",
    "body": "",
    "user": user
  };
  try {
    Map<String, dynamic> _resgisterUser =
        await registerStudent(user, secondaryApp);
    if (_resgisterUser["status"] == "ERROR") {
      throw (_resgisterUser["body"]);
    }
    user = _resgisterUser["body"];
    Map<String, dynamic> _studentCreateDB =
        await addStudentToDB(studentDataCollectionRef, student: user);
    if (_studentCreateDB["status"] == "ERROR") {
      throw (_studentCreateDB["body"]);
    }

    } catch (e) {
      _responsePerUser["status"] = "ERROR";
      _responsePerUser["body"] = e;
    }
    _response["body"].add(_responsePerUser);
  }
  } catch (e) {
    _response["status"] = "ERROR";
    _response["body"] = e;
  }
  return _response;
}
```

3.4 OTHER IMPLEMENTATIONS

1) Firebase

Firebase is used for authenticating users using FirebaseAuth and used to store data using FireStore services.

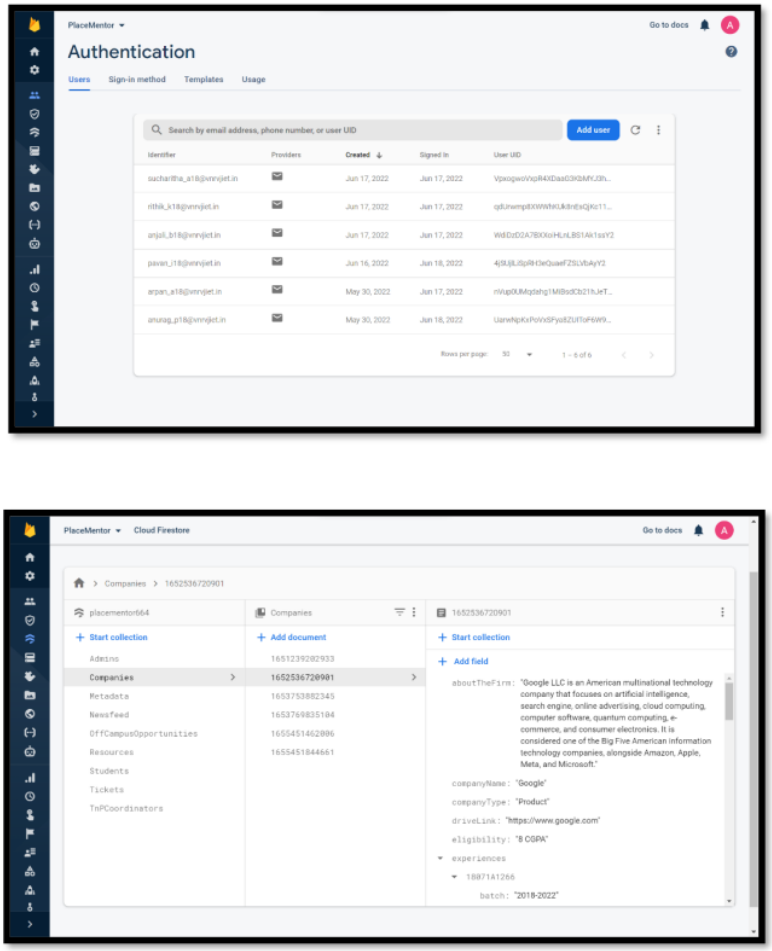
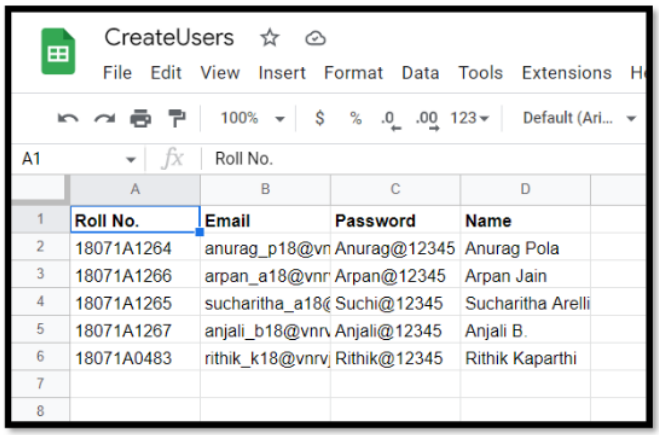


Fig 3.4.1 – FirebaseAuth and FireStore Services

2) Google Apps Script

Google Apps Script is used to run batch processes. We use it to add or delete students from the application. Student details are stored in Google Sheet and then the batch process is invoked.



CreateUsers

File Edit View Insert Format Data Tools Extensions H

100% $ % .0 .00 123 Default (Ari...

A1 Roll No.

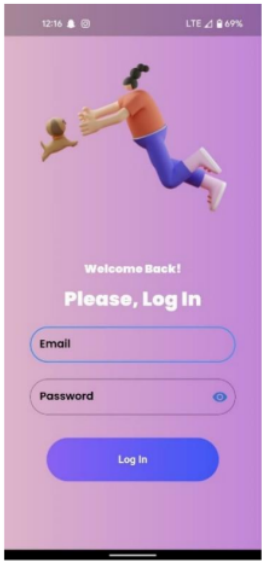
	A	B	C	D
1	**Roll No.**	**Email**	**Password**	**Name**
2	18071A1264	anurag_p18@vn	Anurag@12345	Anurag Pola
3	18071A1266	arpan_a18@vnr	Arpan@12345	Arpan Jain
4	18071A1265	sucharitha_a18(	Suchi@12345	Sucharitha Arelli
5	18071A1267	anjali_b18@vnrv	Anjali@12345	Anjali B.
6	18071A0483	rithik_k18@vnrv	Rithik@12345	Rithik Kaparthi
7				
8				

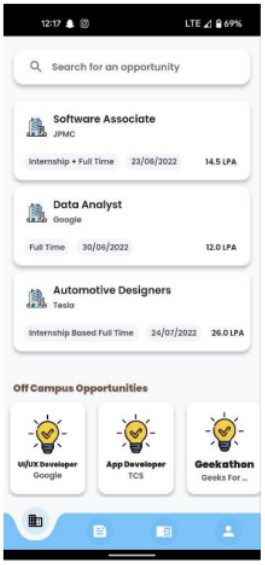
```
function doGet(request) {
  var result = { "status": "SUCCESS" };
  try {

    var createUsersSheetUrl = request.parameter.CreateUsersSheetUrl;

    var sheet = SpreadsheetApp.openByUrl(createUsersSheetUrl);
    var data = sheet.getDataRange().getValues();
    var dataSize = data.length;
    var returnData = new Array();
    if (dataSize >= 2) {
      for (i = 1; i < dataSize; i++) {
        var obj = {
          "id": data[i][0],
          "email": data[i][1],
          "password": data[i][2],
          "name": data[i][3],
        };
        returnData.push(obj);
      }
      result = {
        "status": "SUCCESS",
        "data": returnData
      };
    } else {
      throw new Error("No Data Available in Index Sheet");
    }
```

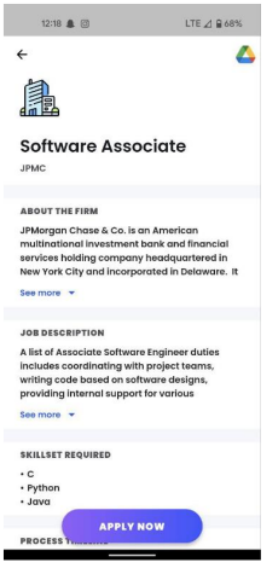
Fig 3.4.2 – Sample Google Sheet and Apps Script Code to Create Students

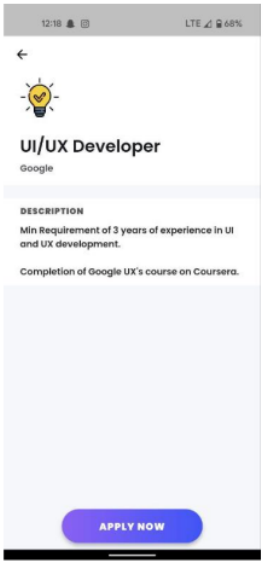
CHAPTER – 4

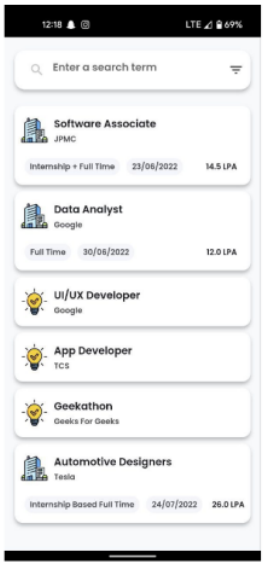
OUTPUT SCREENSHOTS

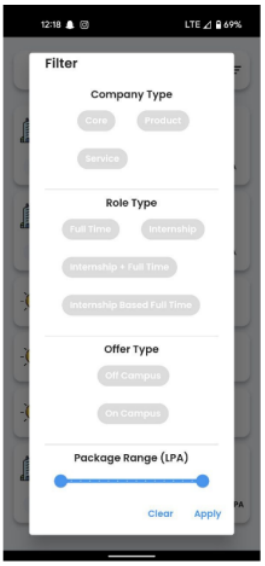












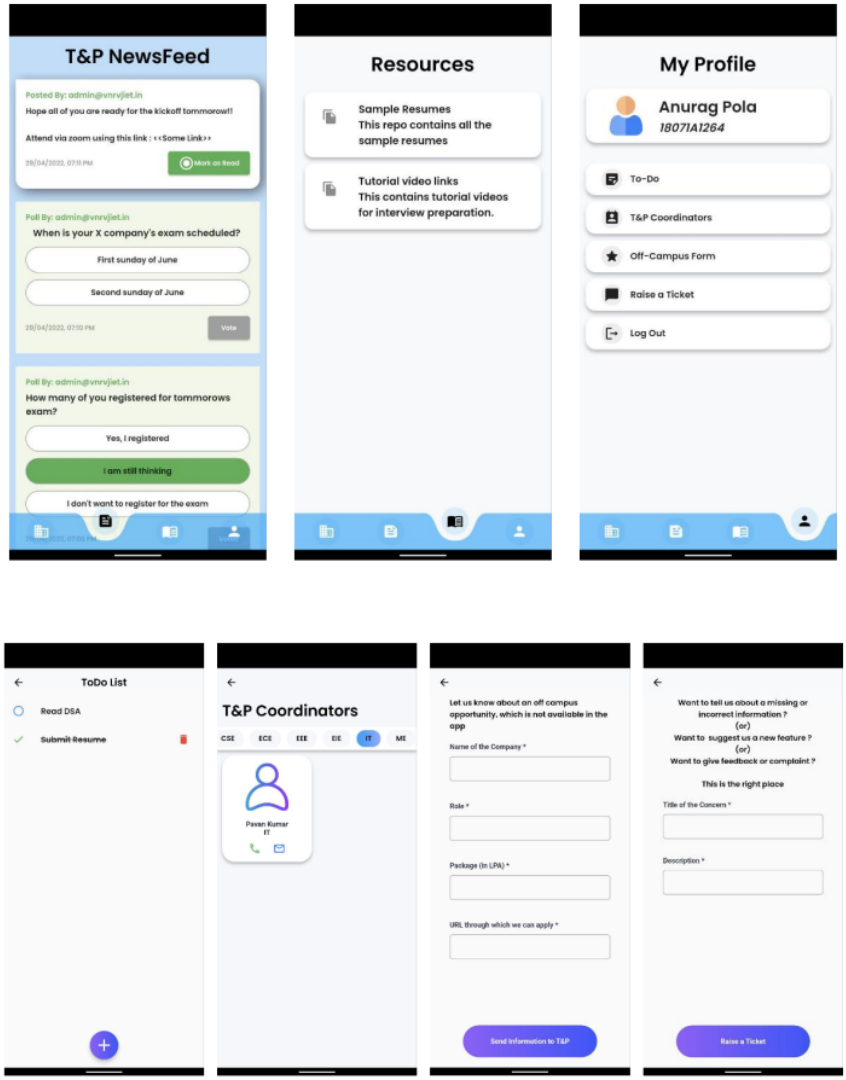
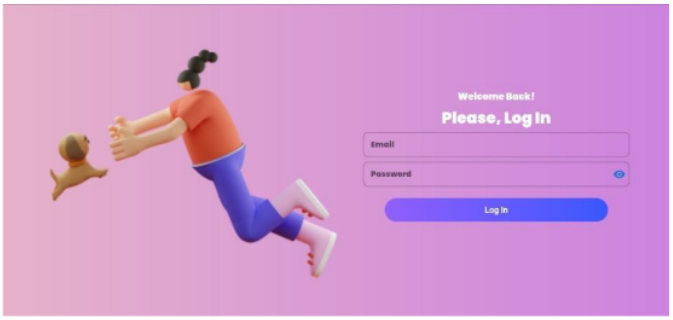
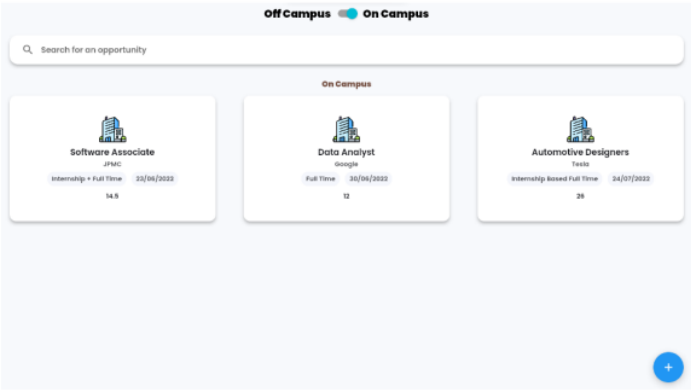
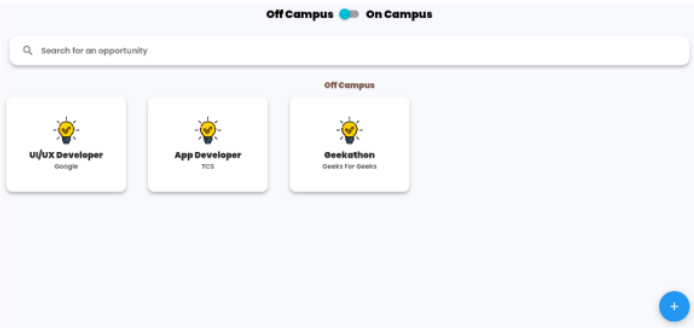


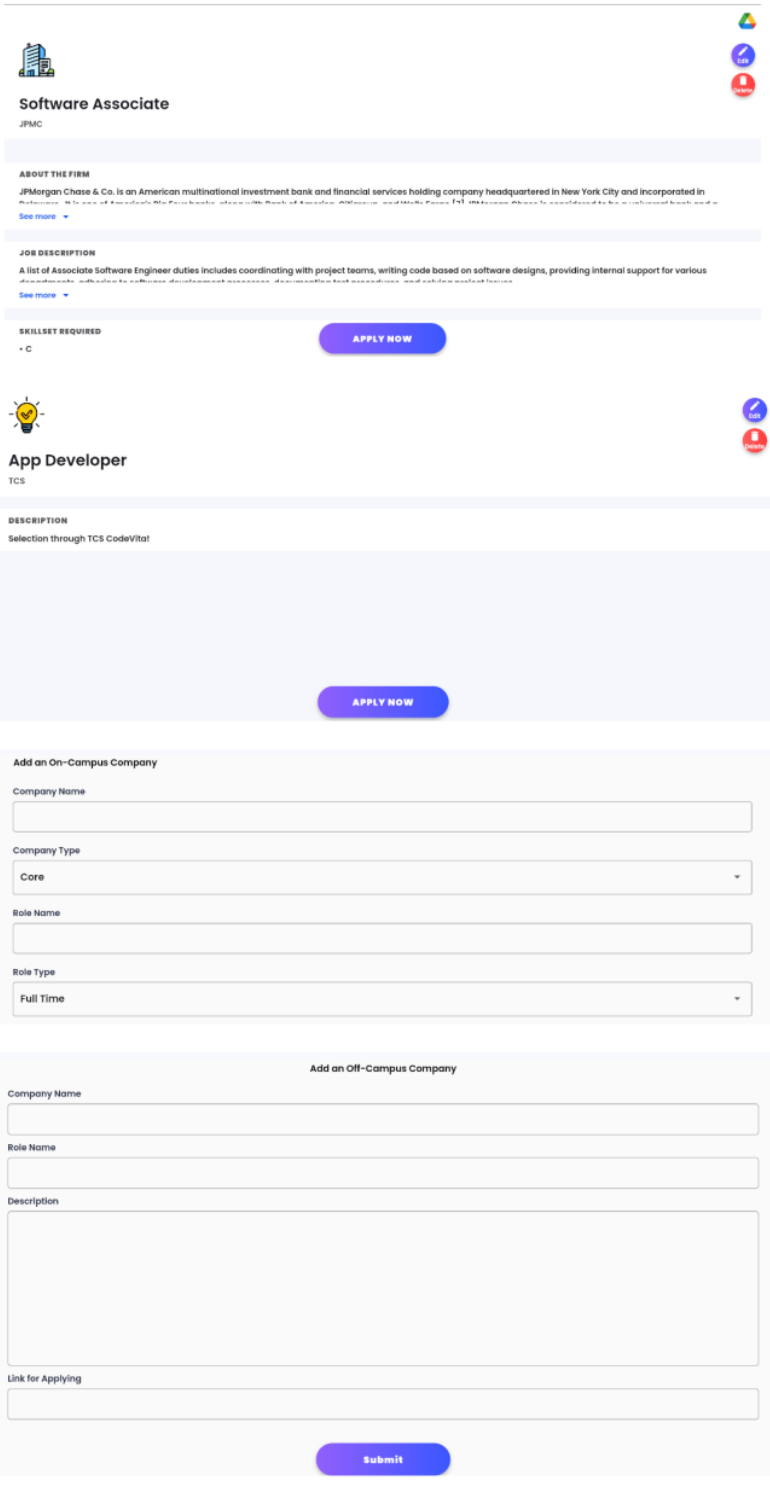
Fig 4.1– Mobile Application Output Screenshots

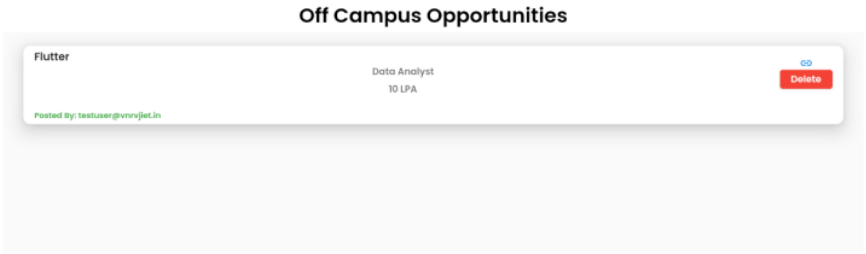

Welcome Back!
Please, Log in
Email
Password
Log in

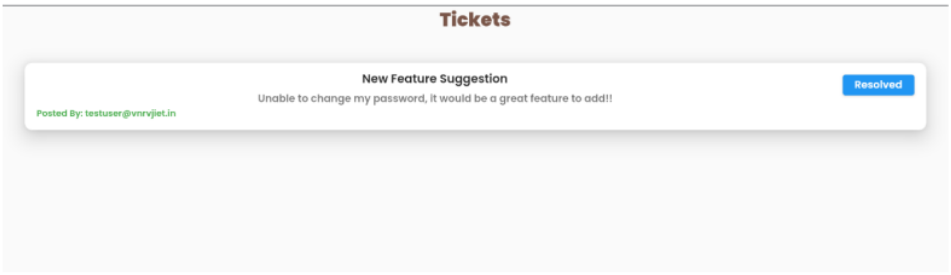

Admin Menu Board
Pavan Kumar
IT
CI12345
9879543210
Companies
Tickets
Newsfeed
Resources
Off Campus Opportunities
T&P Coordinators
Students
Log Out


Off Campus
On Campus
Search for an opportunity
On Campus
Software Associate
JPMC
Internship + Full Time
22/06/2022
14.5
Data Analyst
Google
Full Time
30/06/2022
12
Automotive Designers
Tesla
Internship Based Full Time
24/07/2022
26

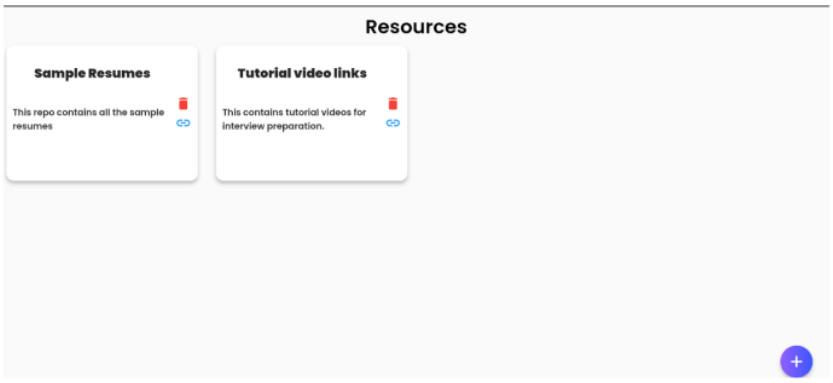

Off Campus
On Campus
Search for an opportunity
Off Campus
UI/UX Developer
Google
App Developer
TCS
Geekathon
Geeks For Geeks


Software Associate
JPMC
ABOUT THE FIRM
JPMorgan Chase & Co. is an American multinational investment bank and financial services holding company headquartered in New York City and incorporated in
See more
JOB DESCRIPTION
A list of Associate Software Engineer duties includes coordinating with project teams, writing code based on software designs, providing internal support for various
See more
SKILLSET REQUIRED
• C
APPLY NOW
App Developer
TCS
DESCRIPTION
Selection through TCS CodeVita!
APPLY NOW
Add an On-Campus Company
Company Name
Company Type
Core
Role Name
Role Type
Full Time
Add an Off-Campus Company
Company Name
Role Name
Description
Link for Applying
Submit


Off Campus Opportunities
Flutter
Data Analyst
10 LPA
Delete
Posted By: testuser@vnrvjiet.in


Tickets
New Feature Suggestion
Unable to change my password, it would be a great feature to add!!
Resolved
Posted By: testuser@vnrvjiet.in


T&P NewsFeed
Posted By: admin@vnrvjiet.in
Hope all of you are ready for the kickoff tommorow!!
Attend via zoom using this link : <<Some Link>>
Delete
Poll By: admin@vnrvjiet.in
When is your X company's exam scheduled?
First sunday of June
100% [1]
Second sunday of June
0% [0]
Delete
Poll By: admin@vnrvjiet.in
How many of you registered for tommorows exam?
Yes, I registered
0% [0]
I am still thinking
100% [2]
I don't want to register for the exam
0% [0]
Delete


Resources
Sample Resumes
This repo contains all the sample resumes
Tutorial video links
This contains tutorial videos for interview preparation.

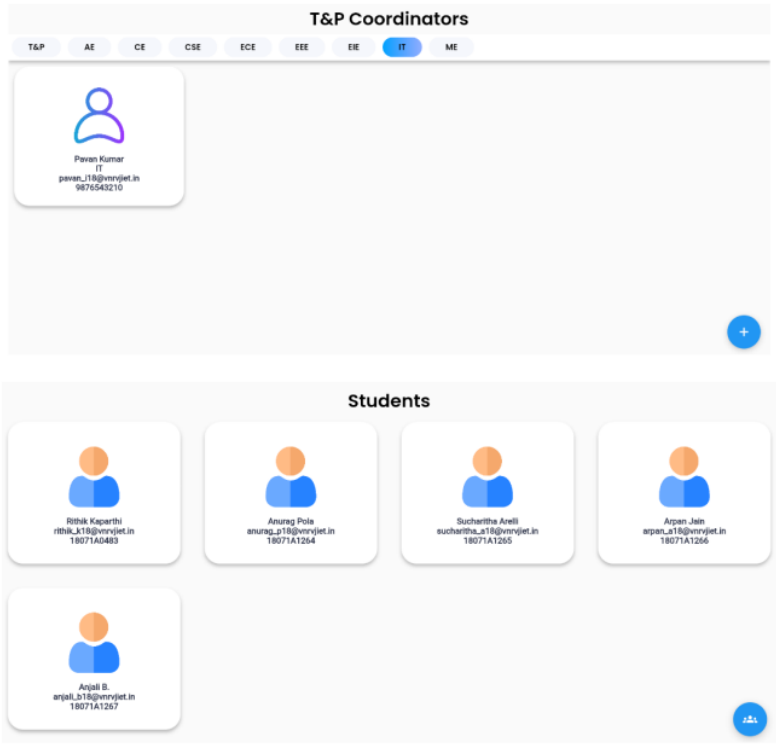


Fig 4.2 – Website Output Screenshots

CHAPTER – 5

CONCLUSION

Placementor, both App and Website combinedly, would constructively help the college and students to have the process of placements being seamless and effective.

Based on the existing solutions, we considered their advantages and disadvantages and tried to integrate all the positive points while also trying to reduce the negative points in them to create a new solution for our college. The UI/UX has been built in such a manner that the user would get most of his/her placement related needs at his/her fingertips without confusion. All the information and functionalities have been embedded into the application (Both App and Website) in an easily understandable and usable manner.

It would reduce the work needed by placements team at the same time giving the maximum information to the student so that they can reach their goals. A complete two-way bridge of communication will be built between the college and students to ensure maximum efficiency from both of them.

CHAPTER – 6

FUTURE SCOPE

The solution can be improved and extended to make it more feasible. Some of the extensions are mentioned below.

1) Integration with Eduprime

The application can be directly integrated with Eduprime, our college's one stop for all the information of a student and share data in both directions which would benefit the application as well as college.

2) Providing the same services in other colleges

The same application can be customized according to needs of other colleges and can be supplied to them.

3) Direct Integration with companies

Companies can be directly contacted and can be asked to give their placements via the mobile app removing the burden of placements team of colleges.

4) Providing placement status tracker and personalized trainings to students

Students can track their placement timeline for all opportunities they have applied for. For placements, they can be provided with personalized training so as to boost performance.

5) Overview of placements to college and public

Providing a complete statistical view of placements in the form of numbers and visualizations to T&P department, Management and Public.

CHAPTER - 7

REFERENCES

1) Flutter Documentation

   https://www.flutter.dev/docs

2) Dart Documentation

   https://dart.dev/guides

3) Package Repository for Dart and Flutter

   https://www.pub.dev/

4) Google Firebase Documentation

   https://www.firebase.google.com/docs

5) Android & Android Studio Documentation

   https://www.developer.android.com/docs

B3

ORIGINALITY REPORT

1%	1%	0%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1%
2	Submitted to University of North Texas Student Paper	<1%
3	Submitted to Babes-Bolyai University Student Paper	<1%
4	dspace.jdvu.ac.in Internet Source	<1%
5	Submitted to King's Own Institute Student Paper	<1%
6	Submitted to University of Westminster Student Paper	<1%
7	www.technewstoday.com Internet Source	<1%
8	Submitted to University of Huddersfield Student Paper	<1%
9	www.ce.utexas.edu Internet Source	<1%

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		