islington college
(इस्लिङ्टन कलेज)

LONDON METROPOLITAN UNIVERSITY

CS6P05NI Final Year Project Computing

Final Year Project

Alumni tracking system

Year and Semester

2019-2020

Student Name: Saugat Poudel

London Met ID: 17031123

College ID: np01cp4a170093

Assignment Due Date: 2020/06/05

Assignment Submission Date: 2020/06/05

Word Count (Where Required): 8893

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

External Supervisor:

Mr. Ishwor Shrestha

Internal Supervisor:

Miss. Subekshya Shrestha

Acknowledgement:

Firstly, I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my sincere gratitude to my supervisor Mr. Ishwor Shrestha and second supervisor Miss. Subekshya Shrestha for their invaluable advice, guidance and enormous patience throughout the completion of this project. Without their guidance and support, I would not have been able to deliver my project on time. Sincere gratitude to them for being a kind supervisor to us.

Secondly, I would also like to thank the all FYP Team for including this final year project in our course, as this project really helps us to apply all the knowledges that we have learned in these past two years and build something creative of our own interest. In addition, I would also like to express my gratitude to my loving parents who had helped and given me encouragement. Also, I would like to thank all my Islington college friends who have been so supportive and helpful towards me.

Abstract:

The structure of this document is categorized into different part. First, it starts with introduction. Introduction part includes overall report scenario discussion. The main introduction to the project topic, the purpose of the topic, problem statement and project as solution.

Similarly, next chapter is background which is totally based on research about the project which contains survey of the project, information gathered from different source about this project. It contains a review of similar system and review of technical aspects. Likewise, selection of software development methodology, their advantages and disadvantages are discussed with diagram. Design section is the next part of the report where all the UML diagrams of the system is shown. After design section the next section is testing section. And finally, the project is concluded.

Table of Contents

Table of Figures:

Table of Tables:

Chapter 1: Introduction

1.1: Topic discussion:

Every organization needs growth, and inorder to grow an organization the system is required which will contains some rules and plans to be followed. System are defined by the organization holders and its entities. System includes data's and rules. As an organization grows, it must adapt to its new size inorder to succeed and continue growing. It must provide new structures to guarantee efficiency in managing the workflow and workload. One of the most important things for an organization is information and human resource management. As more information are generated and more personal employed, the paper record keeping becomes unmanaged and old fashioned which can give bad results to the organization system management. Developing a digital records system for every data of organization becomes an important step in the maturing of an organization in this digital age. (Burke & Noumair, 2015)

The project is about college and university which is also an organization which requires data of students, teachers, alumni, faculty, administration and so on to run effectively. As we discussed above many universities are facing problems due to unmanaged database and to overcome this problem this project is made. This project is based on mobile application. The project also includes a database where all the records of users are stored. Tracking the alumni can be done by both college and students of that community.

1.2. Current scenario:

In today's rapidly competitive era the urge of better and quick technology is the urgent need of any organization. The country is developing, and literacy rate is rapidly increasing. In such context lots of universities are establishing and many students are enrolling. Managing the data of university has become another big challenge due to such high enrollment. In current scenario universities are only focusing on managing currently enrolled student's data to make data fresh and they are deleting data of graduated student. As number of student graduates from university they are not being tracked by the university anymore. Which has become a bigger problem for today's graduates.

Also, In the current system data are stored in flat files. These files can be accessed directly under the supervision of the operating system as separate entities. No structural relationship exists

between any two such files. A user can get complete access over any of the data files in a non-concurrent fashion. If several users wish to get access to a file at the same time, the only way is to have different copies of concerned file. This kind of system result in data duplication. Data are not normalized and full of anomalies. There is no any user control to a file system which results in data insecurity. (Pakhhira, 2013).

1.3: Problem statement:

Most of the university only keeps the data record of the students who are currently enrolling in their college. They simply delete the data once the students are graduated or they left the university to maintain fresh data. They start focusing on only the present students. Due to such bad practice college are not able to identify their own students after some years. If communication between college and alumni stops once alumni leave an institution, their understanding of the university become stale. (Tansey, 2008)

Every student is better on their own faculty and has done many achievements in their field. But when the university need them in future, they already become stranger to them. Not only university, students also need help from their university in future even after their graduation. Then that time due to lack of records of student they will not be able to be in touch with each other. They will not be able to recognize their own students. (Hegde, 2019).

Students who are currently enrolling in the university are also facing many problems due to not having proper link between alumni. Most of the time during study syllabus and guidance students want to communicate with the alumni. Sometimes students don't get the guidance of teacher and may want to get help from alumni because they are already gone through that phage while knowing about college rules and study materials. (Hugley, 2019)

1.4: Project as Solution:

The Purpose of this project is to make a bond between college, their students and alumni's even after they are graduated. the use of this application is to keep records of students and alumni inorder to track them in future. The project will help student and colleges to identify their alumni easily and helps to be in touch.

In this project there will be a database of alumni, students and college with their current address, the jobs they are engaged in and their achievement. These details can be visualized by all student, college and alumni of the same community. These data are normalized and free of anomaly. The system has user control in which an admin will control the system and will get special access to create new college so that the data will be secured. A survey was taken before making this application, to know about what things are hampering to students and alumni during such improper management system. The survey was online question answer form to all the students and alumni about this project. (The survey result is attached in Appendix section B). the system will allow users to update their data so that other user will get updated information. The system will categorize the alumni, student and college into different category by providing a role so that the alumni and student will be easily recognized. University can also create events and call their alumni and students from this application so that the bonding will get more better. Every user will be able to mail another user or call another user from the contact provided in the application.

1.4: Structure of report

Chapter 2: Background: This chapter contains the overall research of the project. This chapter also contains comparison between similar projects, and a brief explanation of how it is different from others.

Chapter 3: Development methodologies: Every project is done according to software development methodology. This chapter of the project gives the brief description of which methodology was used and how it is suitable according to this project scenario. This chapter also contains UML diagrams of system.

Chapter 4: Testing and analysis: The testing section shows all the test cases and defines their applicability in the project.

Chapter 5: Conclusion: The conclusion of this report provides a review of how people will solve their problems with the help of this project in the discussed problem domain.

Chapter 6: References: This chapter contains references of research work.

Chapter 7: Bibliography: This chapter contains bibliography of research work done in this project.

Chapter 8: Appendix: This chapter contains remaining research works, Iterations of development, Survey forms, UML diagrams, additional test cases and sample code of the system.

Chapter 2: Background

2.1: About the end users

A targeted group of users are admin, college, alumni and students. Users experiences are different from usability where the main objectives of it is to fulfill needs among the users. Every individual has their own thought in everything however the result from the survey has helped in understanding every individual opinion.

The questions in the survey is set up according to the need of students, college and alumni's so that this application can be applicable to the wide range of users which will be used to track alumni's records and their interaction. According to the survey most of the students are facing the problems due to not having proper interaction with alumni. the survey is also for alumni so most of the alumni response was there are no such medium so that they can be in touch with their college and students. When they come to know that this kind of system is going to be developed in future, they were found excited for this project.

2.2: Understanding the System:

1. Software Architecture:

The architecture of alumni tracking application is implemented on MVC. Laravel is used for backend api creation in this project. (Shavani, 2019). Laravel is easier for making application programming interface from which we give our data access to not only web application but also to the mobile application. (Note: description of Laravel is shown in tools used section of this project.)

About Model View Controller (MVC) and its implementation in system:

MVC is an architectural pattern that is used in software engineering, whose fundamental principal is based on the idea that the logic of an application should be separated from its presentation. It divides a given software application to three interconnected parts, as Model, View and Controller.

In below paragraphs how MVC architecture is implemented in this project is shown.

MVC Components:

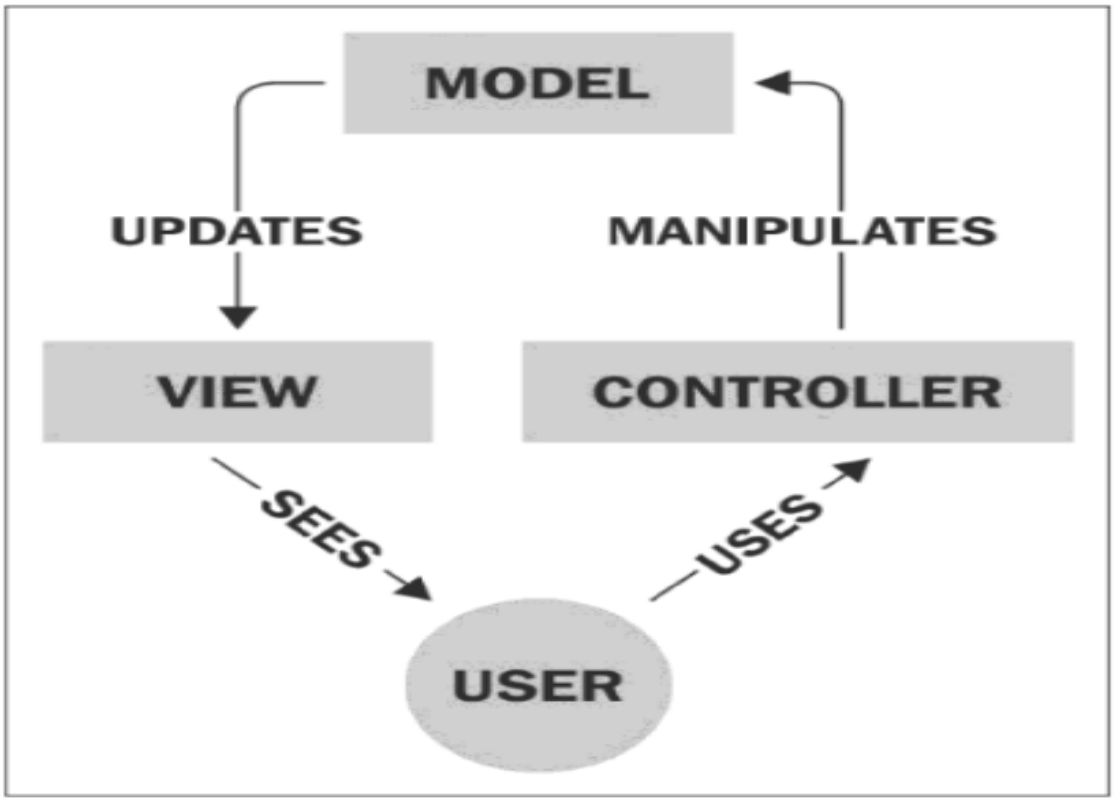


Figure 1 MVC structure.

(Kumar, 2012).

Model: The model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the view and controller components or any other business logic-related data. For example, in the case of this project alumni object will retrieve the alumni data from the database, manipulate it and update it data back to the database.

View: The view component is used for all the UI logic of the application. For example, when a student or alumni or a college want to visit each other profile then they will visit different UI pages which contains UI components like text boxes, images, buttons.

Controller: Controller act as an interface between model and view component to process all the business logic and incoming requests. Manipulate data using the model component and interact with the views to render the output. For example, user's controller will handle all the interacts and

inputs from the users view and update the database using the user model. The same controller will be used to view the user's data. (Kilicdagi & Yilmaz, 2014).

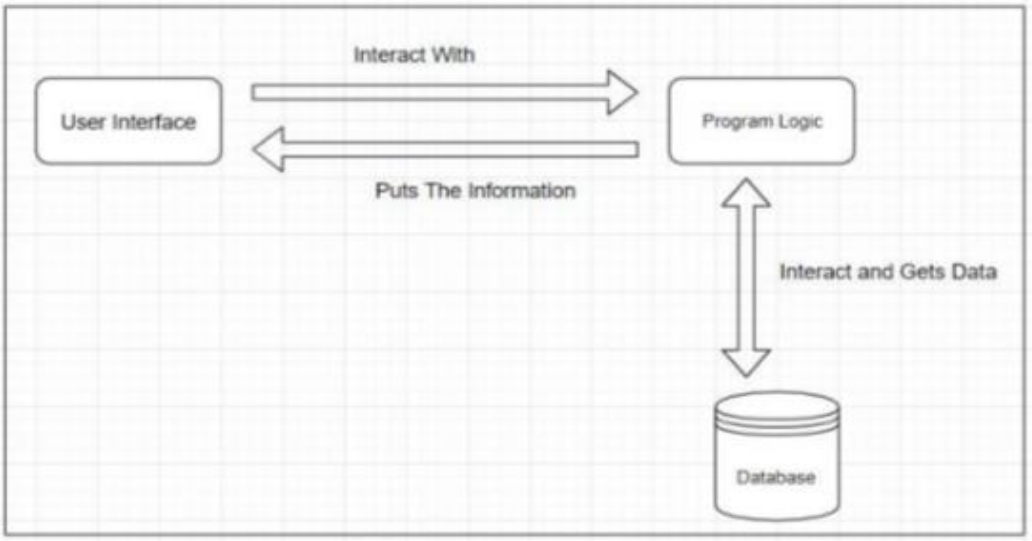


Figure 2 MVC working mechanism

(Guru99, 2020)

Let's take an example of Alumni module which is implemented in MVC architecture. When the Alumni wants to add a new achievement, he first goes to his Profile and sends a request with the required data to the controller. The controller handles the request with the business logic and map the data to the model and save the achievement to the user's database and sends response to the alumni with the status code of 200 i.e. Achievement added successfully. (Sinha, 2019)

Api of users, events have been created and linked with MySQL database. there is also web application for admin from which he can perform his general task.

About front end Implementation:

Taking about the front end and mobile application for this project, flutter is used. (Note: more about flutter tool is discussed in tools used section of this report). Using flutter all the UI of mobile application is created and the api data is fetched in it.

2.3: Review of Similar System

Alumni management application by Sales for Education:

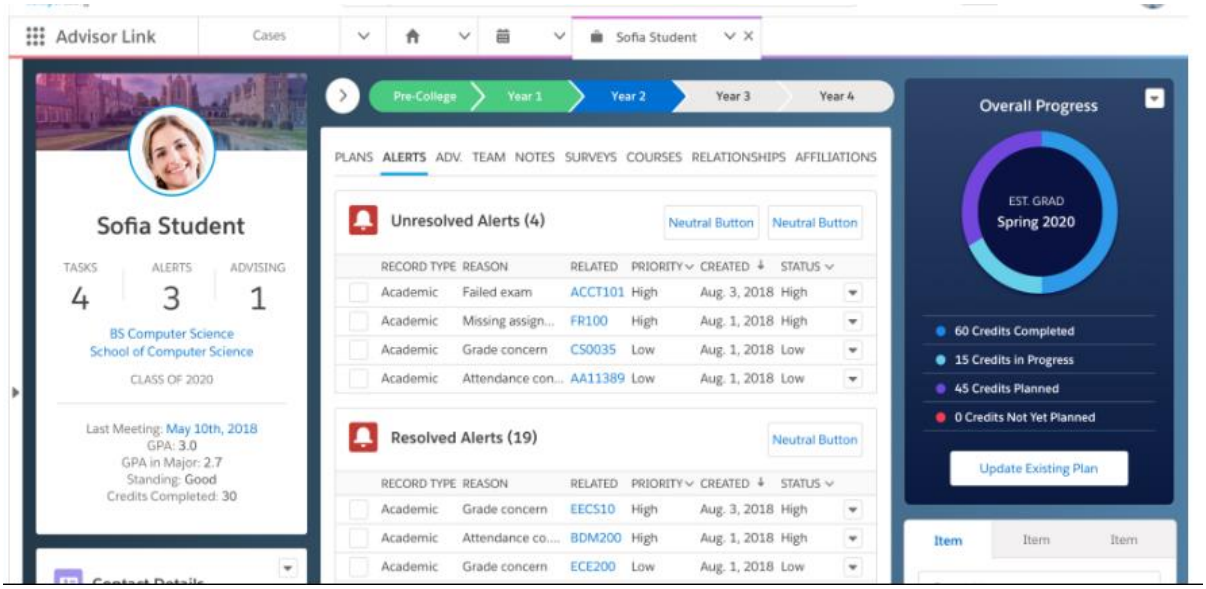


Figure 3 Similar system 1

This is the web-based application about alumni. this application records the data of alumni and students. This application shows the academic progress of student yearly. (salesforceforeducation, 2000) .

Alumni management application by alma student information:

Figure 4 similar system 2

Alma is the web application mainly focused on student and alumni management system. this software register's student gives report cards and view alumni. this application also records attendance of students. (www.getalma.com, 2012)

Comparing the above application with this project:

After researching about 2-3 more similar projects I found that they are mainly focusing on student management rather than alumni management. Table below shows the comparison between above similar project with this project.

Description	Alumni tracking app (My project)	Sales for education (alumni management)	Alma student info
User group	College, student, alumni, admin	Alumni, college, admin.	Student, admin, alumni.
Interaction between each other's.	Available.	Not available.	Not available.
Registration system	Available	Only available by admin.	Only available by admin.
View each other details.	Available	Only by admin.	Only by admin.
Event features	Available	Not available.	Not available.
Mobile application	Available.	Not available.	Not available.

Table 1 comparison of similar application.

Chapter 3: Development

3.1. Considered Methodologies

3.1.1: Waterfall methodology:

The waterfall model originated in 1970 largely through the efforts of dr. Winston Royce. It worked well at the time and has undergone many subsequent changes and revisions. The methodology is currently one of the most widely used software development methodology.

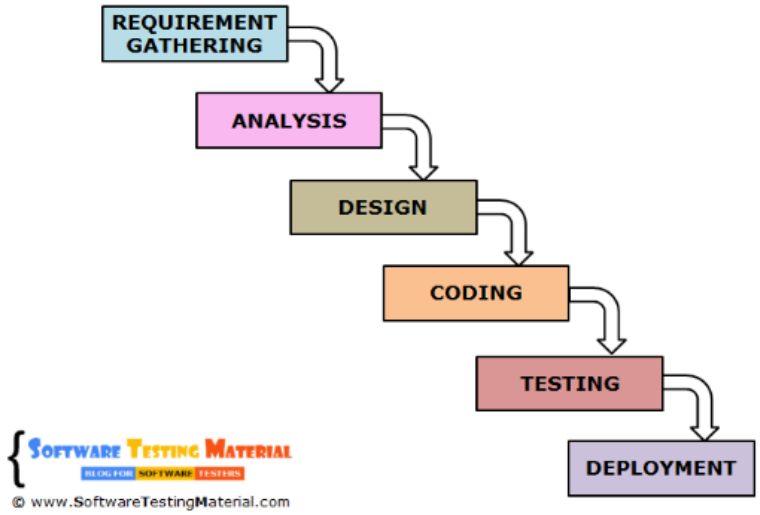


Figure 5 Waterfall model.

(Chi, 2018)

Benefits of a waterfall methodology:

- It provides phase-by-phase checkpoints for the project.
- You may need to proceed to only the following phase after the previous phase has been completed.
- It can also be applied to an iterative approach.

Disadvantages of waterfall methodology:

- There is minimal feedback between project phases.
- You start seeing results only later in the life cycle.
- Each phase is tracked with far too many hard dates and milestones (Charvat, 2003).

3.1.2: Spiral methodology:

Spiral methodology is the combination of waterfall and iterative methodology. Each phage in spiral model begins with a design goal and ends with the client reviewing the progress.

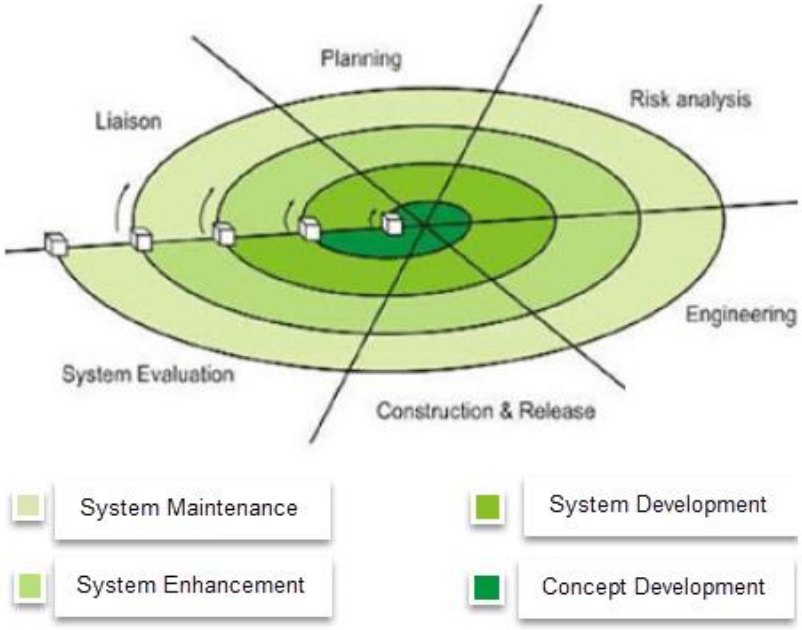


Figure 6 Spiral model.

Advantages of spiral methodology:

- Additional functionality or changes can be done at a later stage.
- Cost estimation becomes easier as the prototype building is done in small fragments.
- Continuous or repeated development helps in risk management
- Development is fast and features are added in systematic way.

Disadvantages of spiral methodology:

- Risk of not meeting the schedule or budget.
- It works best for large projects only also demands risk assessment expertise.
- For its smooth operation spiral model protocol needs to be followed strictly
- Documentation is more as it has intermediate phases.
- It is not advisable for smaller project; it might cost them a lot. (Guru99, 2020)

3.1.3: Rational Unified Process (RUP) Methodology:

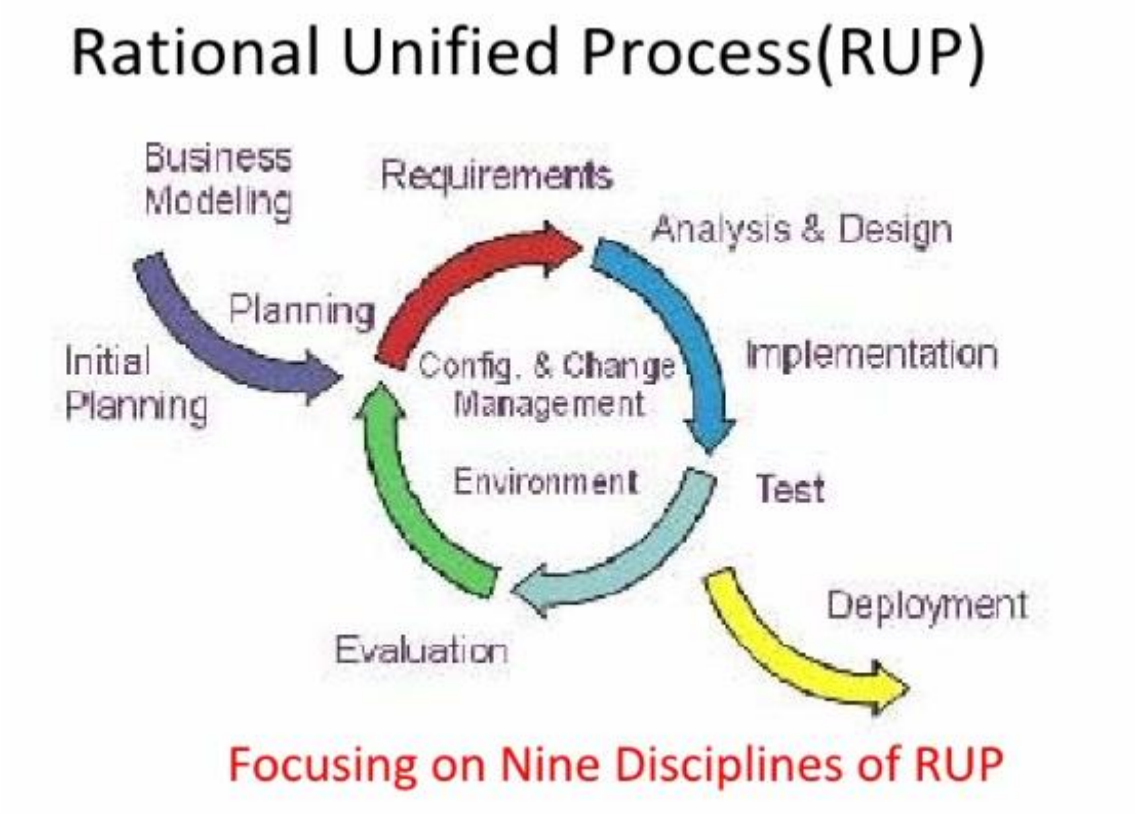


Figure 7 Rational Unified Process.

(Kumar, 2009)

The Rational Unified Process framework was initially created by the Rational Software Corporation, which was bought out by IBM in 2003. RUP is based on few fundamental ideas, such as the phases of development and building blocks, which defines who, what, when, and how development will take place (IBM, 1998).

3.2: Selected methodology

3.2.1: Rational Unified Process (RUP):

After learning all the above methodology. It was found that rational unified process is best for this project. It is easy to understand. We can change the requirement throughout the development cycle. It is also called as Iterative development process which is the main reason to choose this software development methodology. In this project there were unnecessary features and data's before, later in development it was found some of the data are not necessary so as the methodology is Rup, the requirement and data were changed. The project requirement is not necessary to know in beginning when using this methodology. By using this methodology, the system can be refined through Iteration, using users' feedback and adaptation. Each iteration will include requirements, analysis, design and implementation of the system. This software methodology is suitable for both small and big projects because it also follows the rules of agile development when it comes to planning. Also, rational unified process phases are not linear which means every phase are not compulsory we can jump from one phase to another phase without ending that phase. The system requirement changes time and again so switching from one phase to another phase frequently will be easier while developing this project. Also, by using this methodology many features were added while developing and RUP suggest doing this for better development so it was a best methodology to choose.

Advantages of Rational Unified Process:

- Allows for the adaptive capability to deal with changing requirements throughout the development life cycle, whether they be from customers or form with the projects itself.
- Emphasizes the need (and proper implementation of) accurate documentation.
- Diffuses potential integration headaches by forcing integration to occur throughout development specifically within the construction phase where all other coding and development is taking place. (Morse, 2017)

(Note: More about rational unified process and its implementation in this system is discussed in appendix section H).

3.3. The Four Life Cycle Phases of Rational Unified Process:

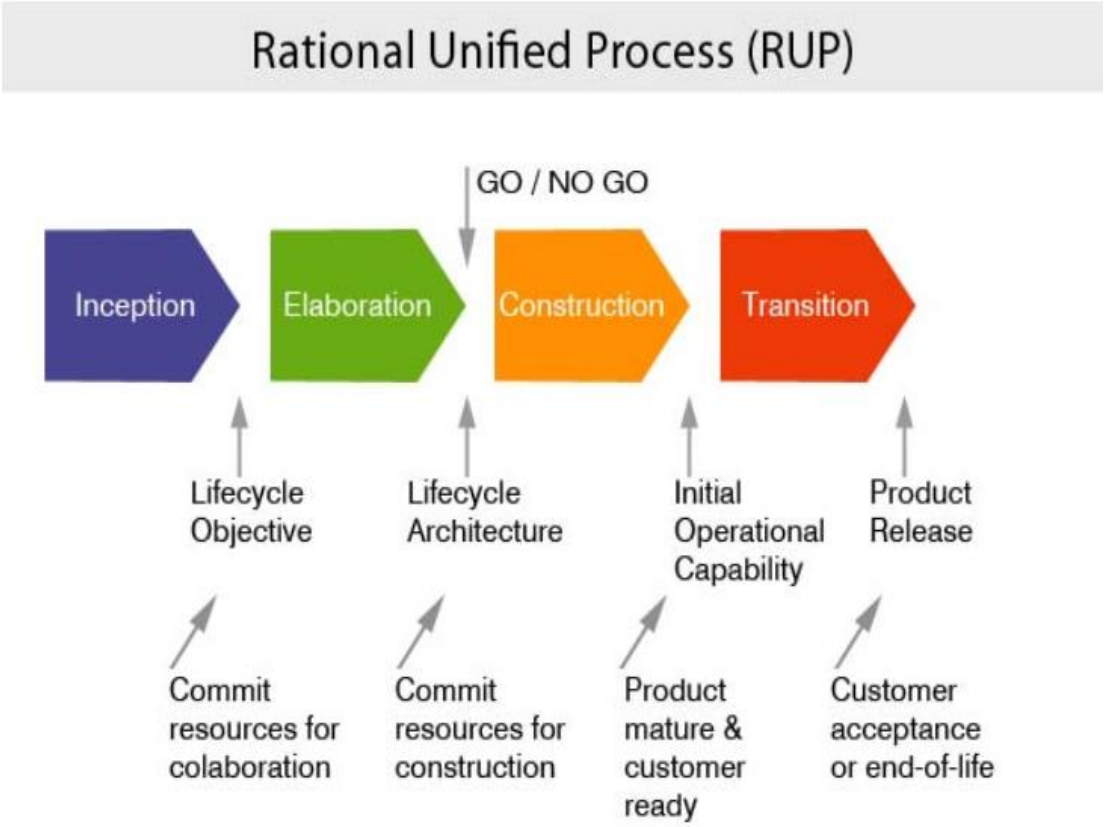


Figure 8 Rup phases.

(infostride.com, 2019)

- **Inception Phase**:

  During this phage the basic idea and structure of the project is determined. The team will sit down and determine if the project is worth pursuing at all based on the proposed purpose of the project, the estimated costs (monetary and time) and what resources will be required to complete the project. In this phase the basic ideas of the projects like where the data of application will be stored, how the user interface should look like and so on was collected.

- **Elaboration Phase:**

  The purpose of the elaboration phase is to analysis the requirements and necessary architecture of the system. The success of this phase is particularly critical, as the final milestone of this phase signifies the transition of the project from low risk to high-risk. In this phase the necessary features, entities and attributes which are required to make this application were collected. For example, the user's data, features like registration, login, view and edit profile, search and view users and events. After collecting all data's, the data was normalized so that they can be plotted in database without any conflicts. Also, in this phase the various use cases and UML diagrams were designed.

- **Construction Phase:**

  Construction phase occurs when the coding and implementation of all application features will take pace. This period is also where integration with other services or existing software should occur. Here in this phase system development was done. Firstly, tasks like Mobile UI designing like login page, registration page were designed. then after completing basic UI designs, API of users and events were created using Laravel. As the backend was in Laravel api of login user, registering user, displaying and editing users was created., After finishing the work of API, connecting them with mobile application was the next task. registering users and authenticating them through mobile application was the next phase of development. Similarly, Admin website was also created from where admin can do basic works without using mobile application. Detail of this phase is included in iteration phase of development in below section.

- **Transition phase:**

  Easier thought of deployment, the transition phase is when the finished product is finally released and delivered to customers. However, the transition phase is more than just the process of deployment; it must also handle all post-release support, bug fixes, patches and so forth. Until this phase all development part was completed and after coming to this phase various testing like unit testing and system testing of the application was done to make the product ready.

  Note: (Each phase has multiple iterations, the detail of each phases of iterations are shown in appendix I section of the report).

Normalization of the system:

The concept of doing normalization is taken from this book. (Gour et al., 2018)

Un- Normalized Form (UNF):

(User_id, Name, email, phone, address, achievement, {role} faculty, job) (event_id, event_name, event_venue, event_time)

here event table don't have any link with other table, so it is already normalized. here a user can have multiple role because a student can be alumni in future.

First Normal Form (1NF):

Users-1: User_id(PK), role_id(FK), Name, email, phone, address, achievement, faculty, job

Role-1: role_id(PK), role

Events-1: event_id(PK), event_name, event_venue, event_time

Second Normal Form (2nf):

Users-2: User_id(PK), role_id(FK), Name, email, phone, address, achievement, faculty, job

Events-2: event_id(PK), event_name, event_venue, event_time

Roles-2: role_id(PK), role

Third Normal Form (3NF):

No anomalies are found in 2nf. hence the table is already in 3nf...

Tools used:

1: Programming:

1.1: Laravel:

Laravel is accessible, yet powerful, providing powerful tools needed for large, robust applications. A superb inversion of control container, expressive migration system, and tightly integrated unit testing support give you the tools you need to build any application with which you are tasked. Laravel attempts to take the pain out of development by easing common task used in most web projects, such as authentication, routing, session and caching. It follows MVC pattern." (laravel, 2011). this programming is used for developing backend creating API's and tables in this project.

1.2: Flutter:

Flutter is Google's UI toolkit for building beautiful, natively compiled applications for mobile, web, and desktop from a single codebase. Its main features are faster development, expressive and flexible UI, Native performance. (flutter, 2020) The programming used in flutter is dart. Dart is object-oriented programming language. Everything in Dart is based upon widgets. Flutter and Dart is used for developing mobile application in this project.

1.3: Firebase:

A comprehensive app development platform which can build apps fast, manage infrastructure, where functionality like analytics, databases, messaging and crash reporting is given. It is made by google. this platform is used in this project for sending notifications. (Firebase, 2020)

3.4. Survey Results:

3.4.1. Pre-Survey Results:

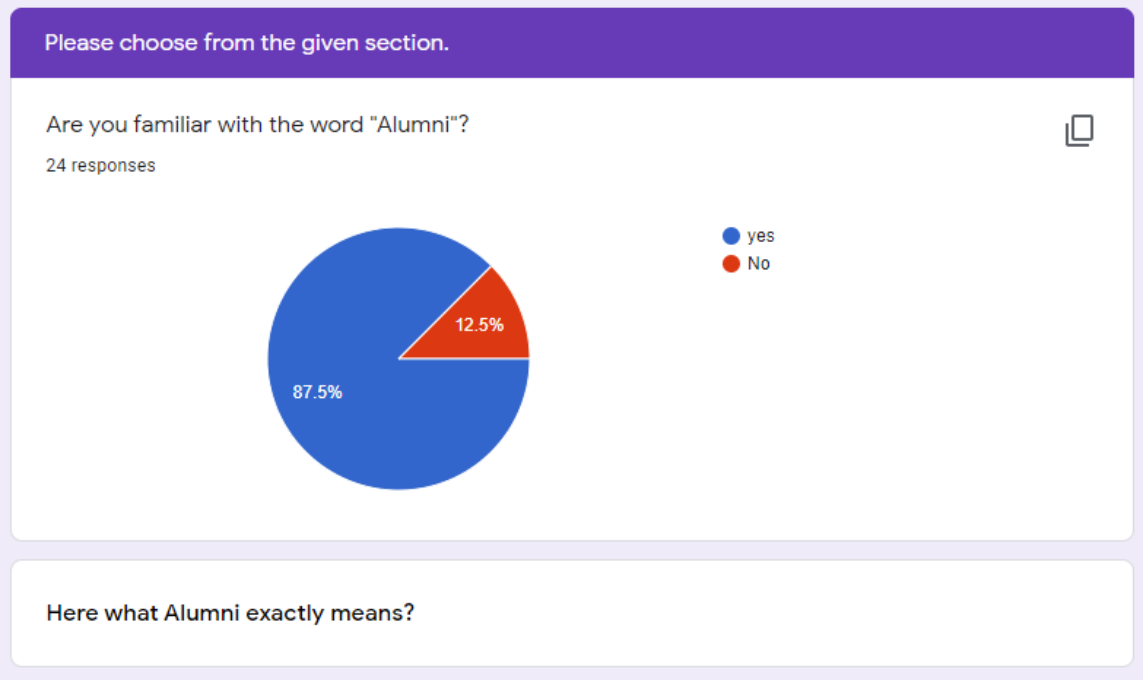


Figure 9 pre survey result 1.

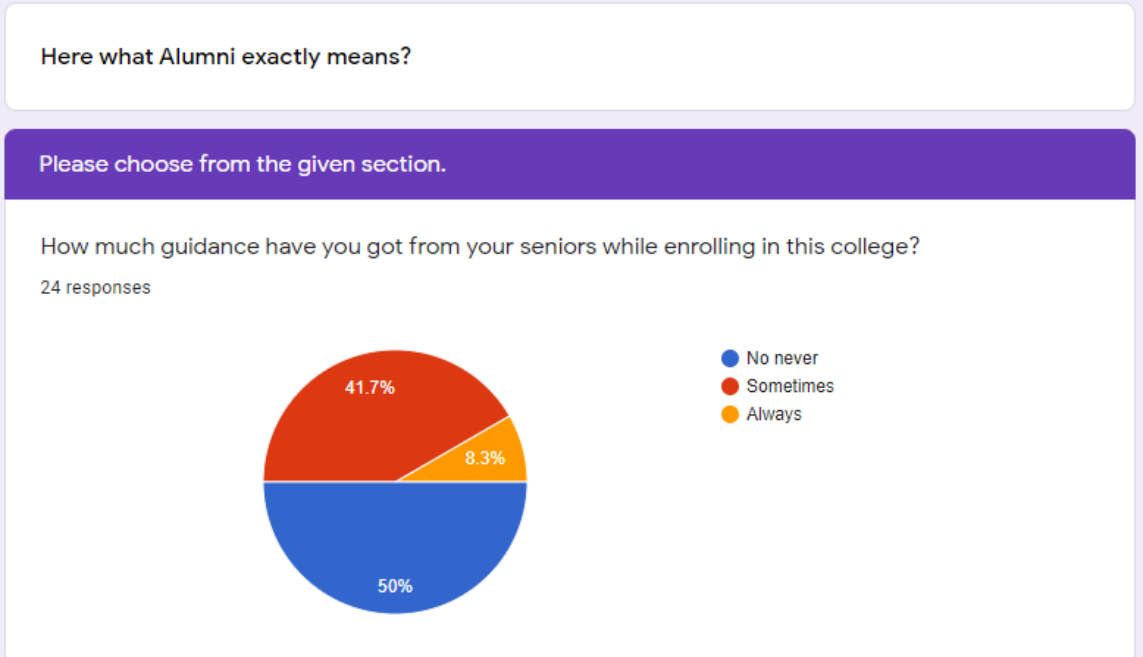


Figure 10 pre survey result 2.

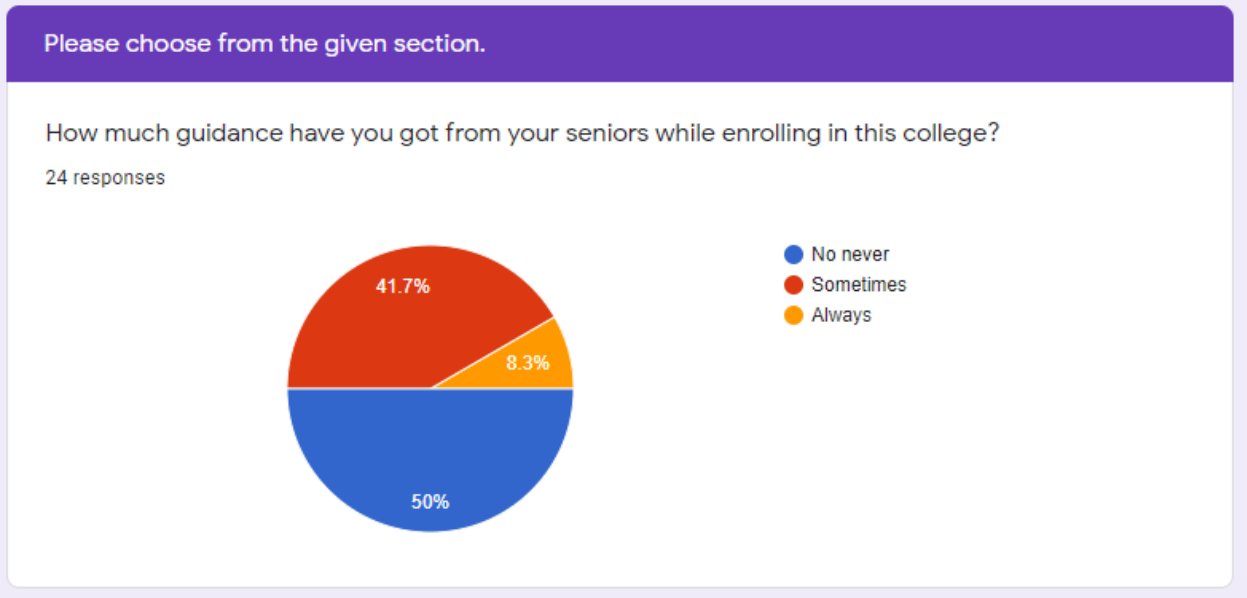


Figure 11 pre survey result 3.

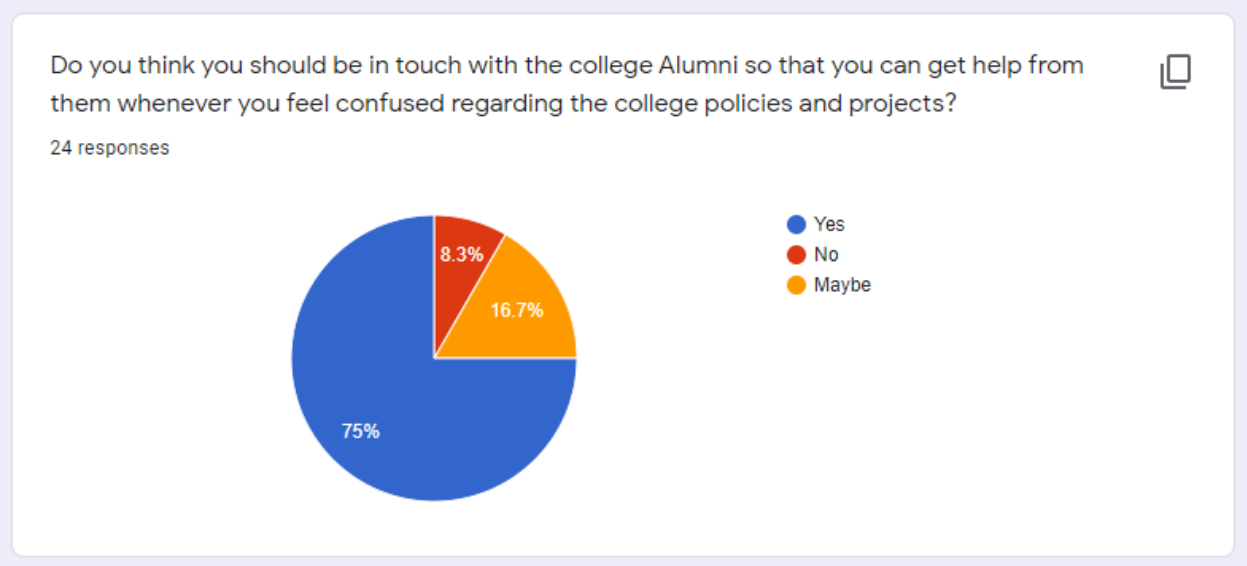


Figure 12 pre survey result 4.

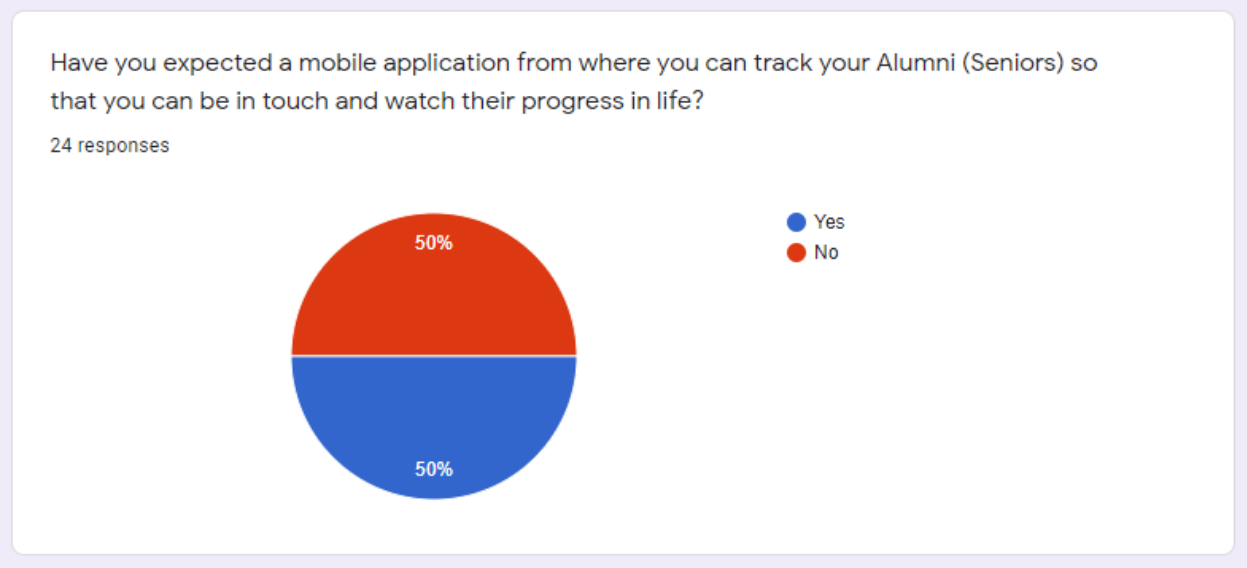


Figure 13 pre survey result 5.

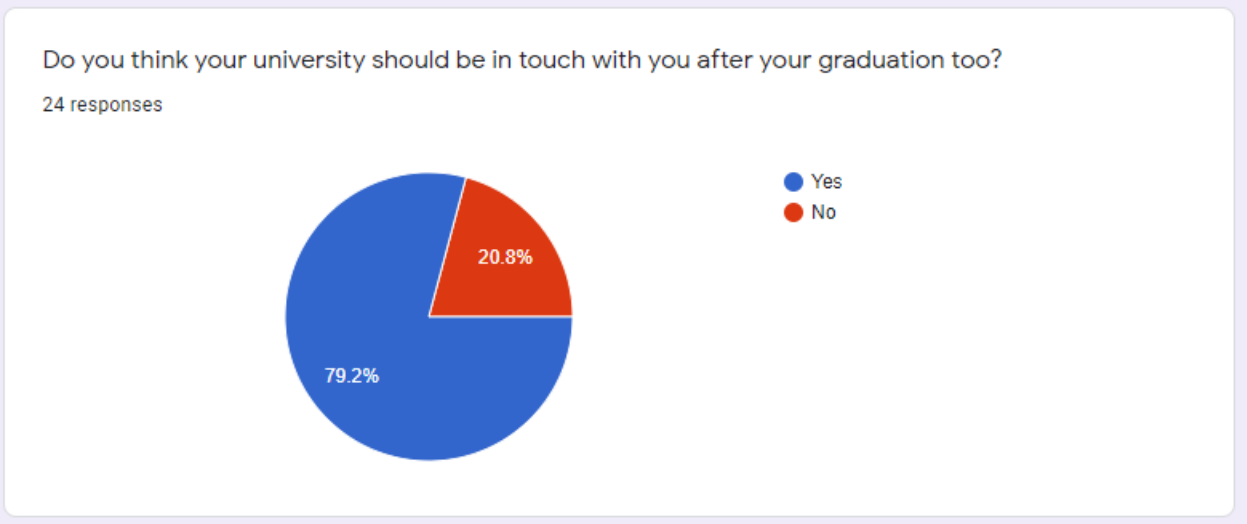


Figure 14 pre survey result 6.

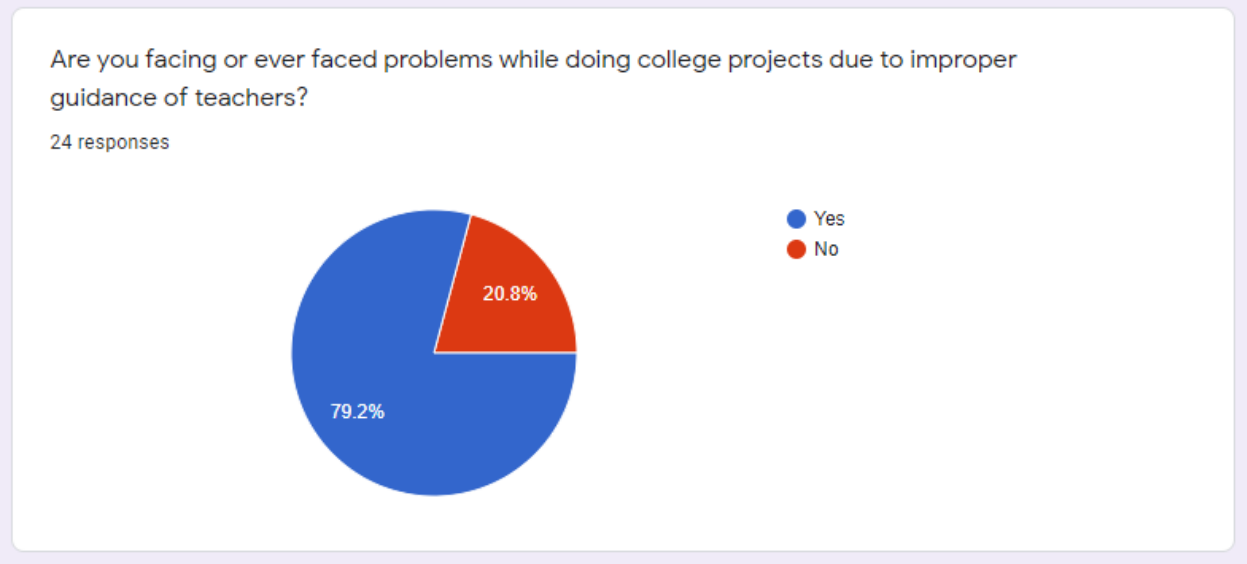


Figure 15 pre survey result 7.

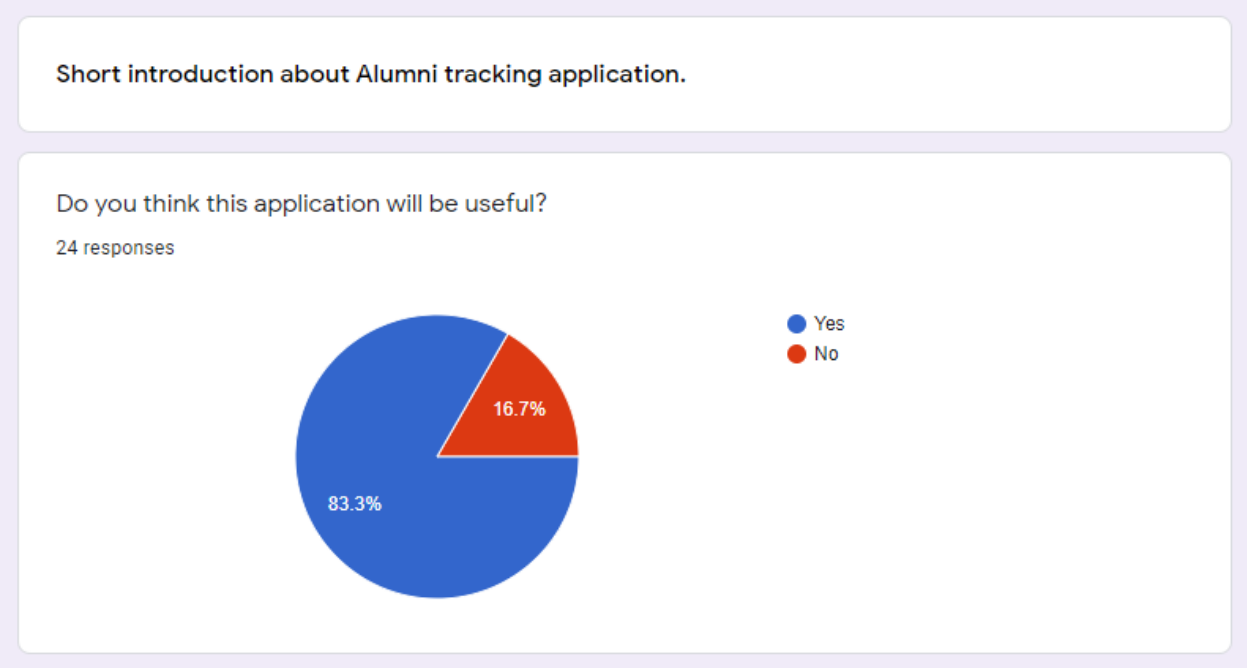


Figure 16 pre survey result 8.

3.4.2. Post Survey Results:

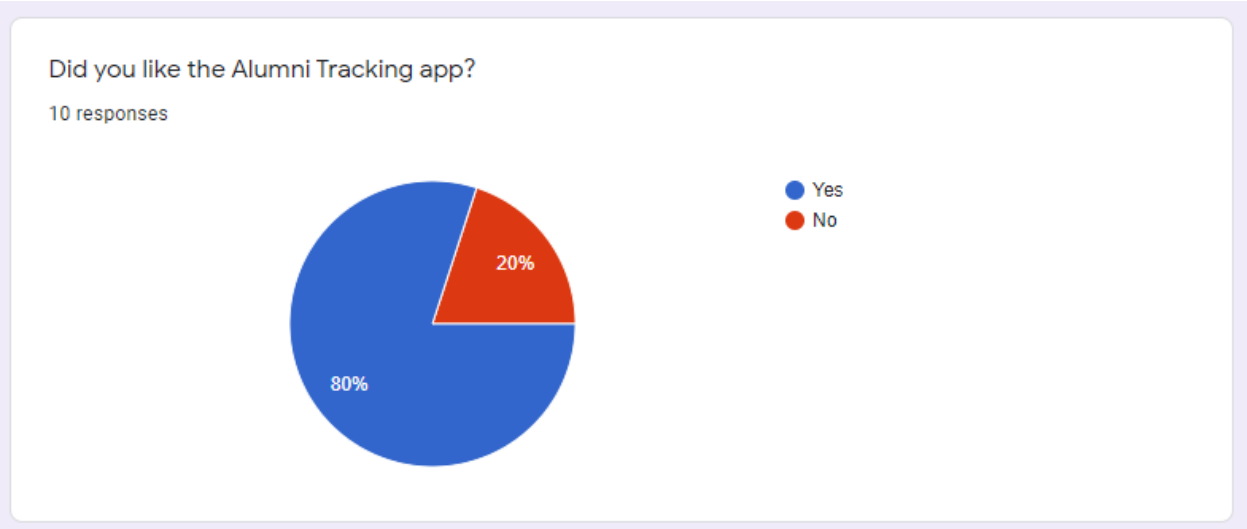


Figure 17 post survey results 1.

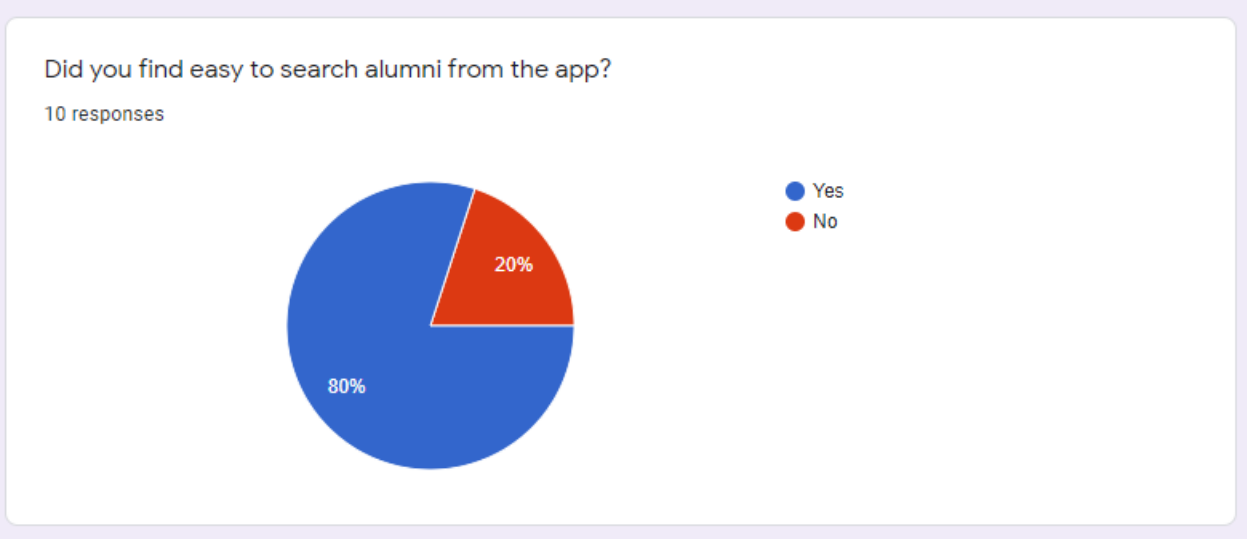


Figure 18 post survey results 2.

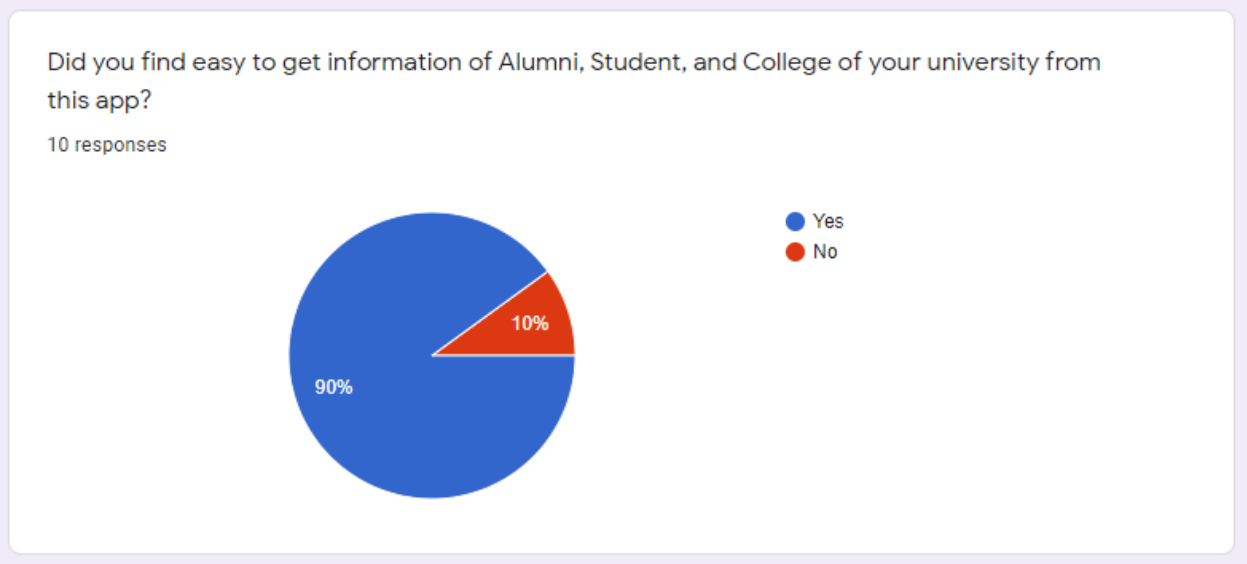


Figure 19 post survey results 3.

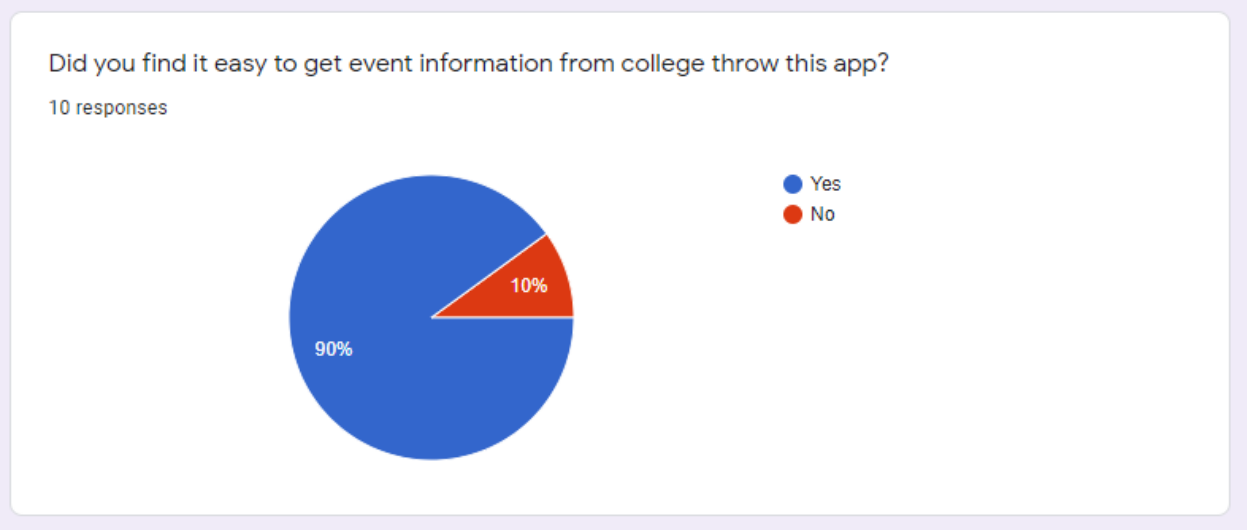


Figure 20 post survey results 4.

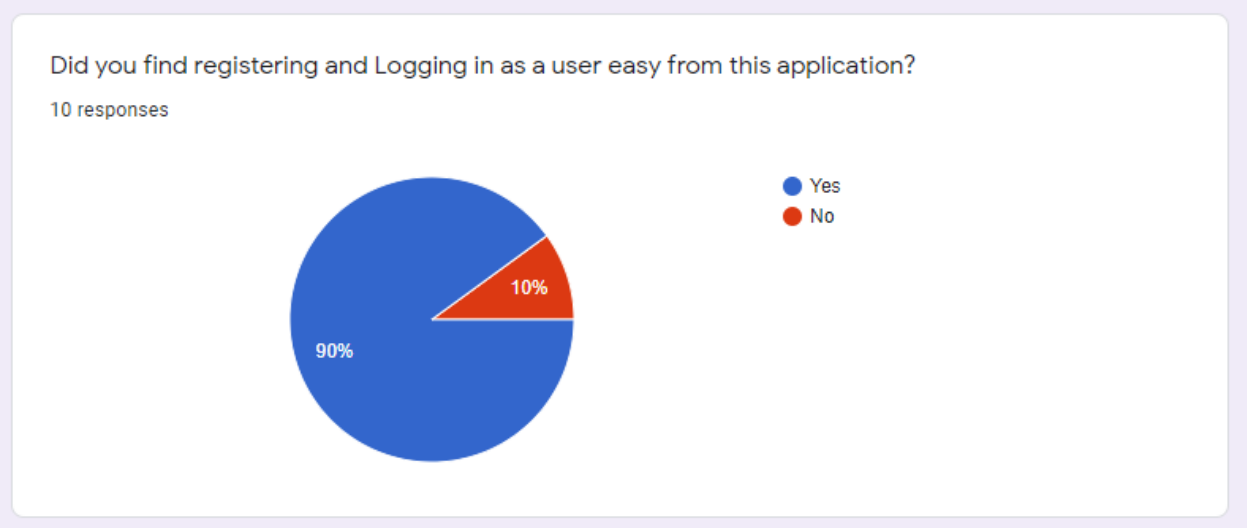


Figure 21 post survey results 5.

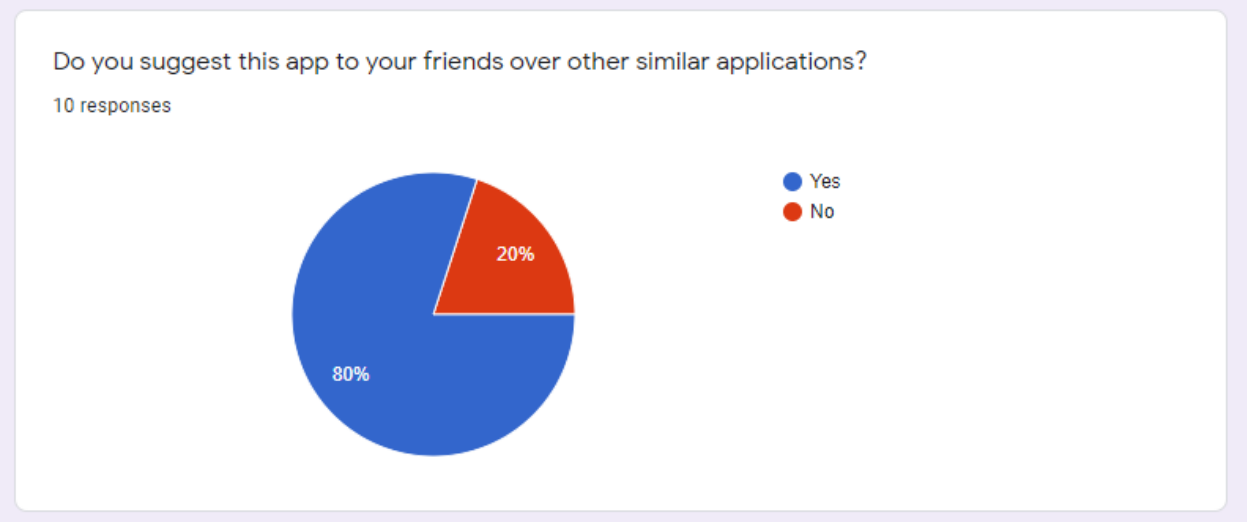


Figure 22 post survey results 6.

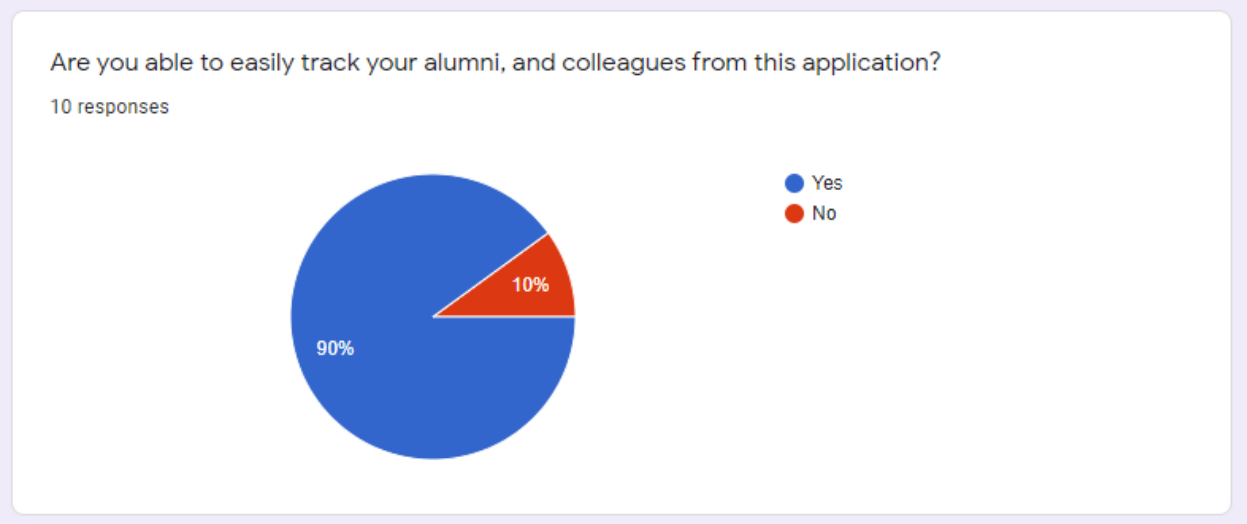


Figure 23 post survey results 7.

Amazing features and user friendly

Nice application

I found the application is very useful, but I am expecting a chat-bot in the near future.
#superapp #alumnitrackingapp

application is overall very useful but i expect an news feed page where users can view other users daily achievements instead of visiting their individual profile.

Figure 24 post survey results feedback.

3.5. Requirement Analysis:

This section contains all the system requirements which is written in MOSCOW prioritization.

M

Must have
1. All users must register themselves to the application except college. 2. All users must provide all information asked in registration page. 3. All users must login to the application. 4. College should add events. 5. All users must login to the application to view the data.

S

Should have
1. All users should login to the application. 2. College should add detail information of the events. 3. Admin should track users from both app and website. 4. Admin should create or register new college. 5. Admin should hand the account of college to respective college. 6. All users should search users to get user information from application. 7. All users should add achievements and details. 8. All users should provide correct information to the application.

C

Could have
1. All users could have achievements. 2. All users could update profiles. 3. All users could view event details. 4. All users could phone one another from application. 5. All users could search each other and view profile. 6. Every user could see their own profile. 7. All users could search events and view details. 8. All users could get event notification. 9. Student could be alumni.

W

Won't have
1. All users won't have event creating authority. 2. College won't register themselves to the application. 3. Admin won't have access to delete user's data and information. 4. Alumni cannot be students, college, and admin.

3.6. Designs

3.6.1: Use Case Diagram:

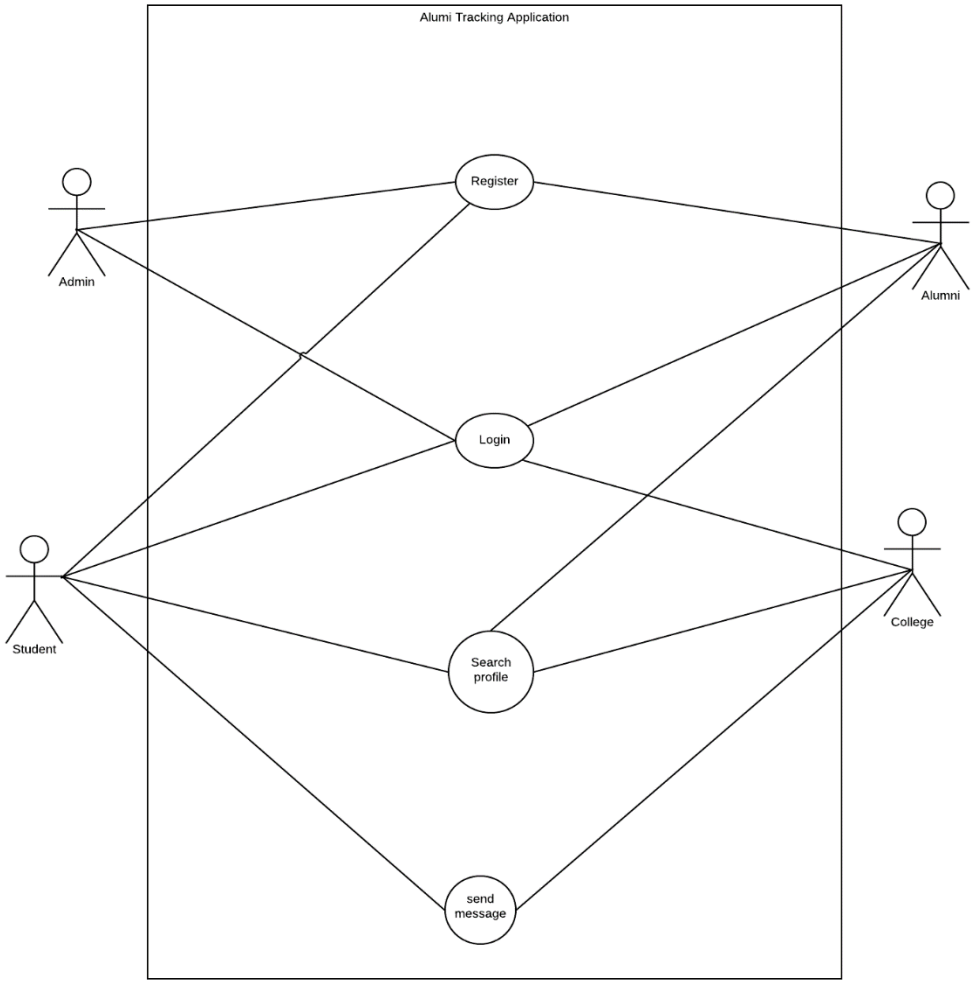


Figure 25 Use case diagram

(Lucid Chart, 2017)

Use case description:

The above use case shows the overall workflow of alumni tracking application. Users will use application to interact with the program. Here users are divided into 4 categories as actors. Alumni, college, students, admin. The above use case is general view of the application. The more detailed use case is described in the high-level use case diagram. All the users can register each other

through the application. Every user except admin can interact in application whereas admin will control other users from the web application. Student, college and alumni can view each other's profile and send message.

3.6.2: High level use case diagram

High level use case diagram for Register:

Use case: Register

Actor involved: Admin, Student, Alumni

Description: Admin can register a college. Whereas student and alumni can register themselves with the application.

High level use case diagram for Login:

Use case: Login

Actor involved: Admin, College, Student, Alumni

Description: Every user can login themselves with application, but admin will only login with web.

High level use case diagram for Update profile:

Use case: Update Profile

Actor involved: Alumni, college, student, admin

Description: Every user will be able to update their own profile

High level use case diagram for Add achievement:

Use case: Add achievement

Actor involved: Alumni

Description: Alumni can only add the achievements they have done but it can be viewed by all the users.

High level use case diagram for Events:

Use case: Events

Actor involved: College, Alumni

Description: College can add new events and alumni can view the events.

High level use case diagram for Search profile:

Use case: Search Profile

Actor involved: College, students, Alumni

Description: All the users can view, and search profiles accept of admin. Admin can only view through web.

High level use case diagram for send message:

Use case: Send message

Actor involved: Alumni, college, student

Description: All users expect admin can send message to each other.

3.6.4: Extended Use Case Diagram

Use case: Register

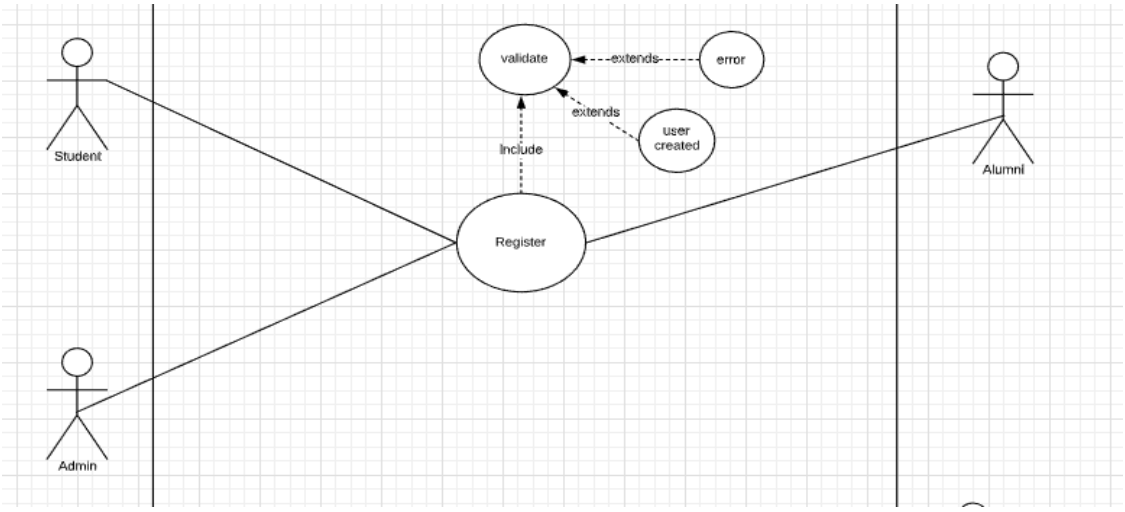


Figure 26 extended use case register

Actor involved: Admin, Student, Alumni

Description: Admin can register a college. Whereas student and alumni can register themselves with the application.

User interaction	System response
1. User (Admin, Student, Alumni) click on register button from application dashboard.	2. System shows a option to register either as alumni or student.
3. User selects any one option.	4. System lands the user to the appropriate registration page.
5. User enters all the details and clicks on register button.	6. System checks the user's validation and field validation and give appropriate response.

Table 2 Extended use case register.

Extended use case diagram for Login:

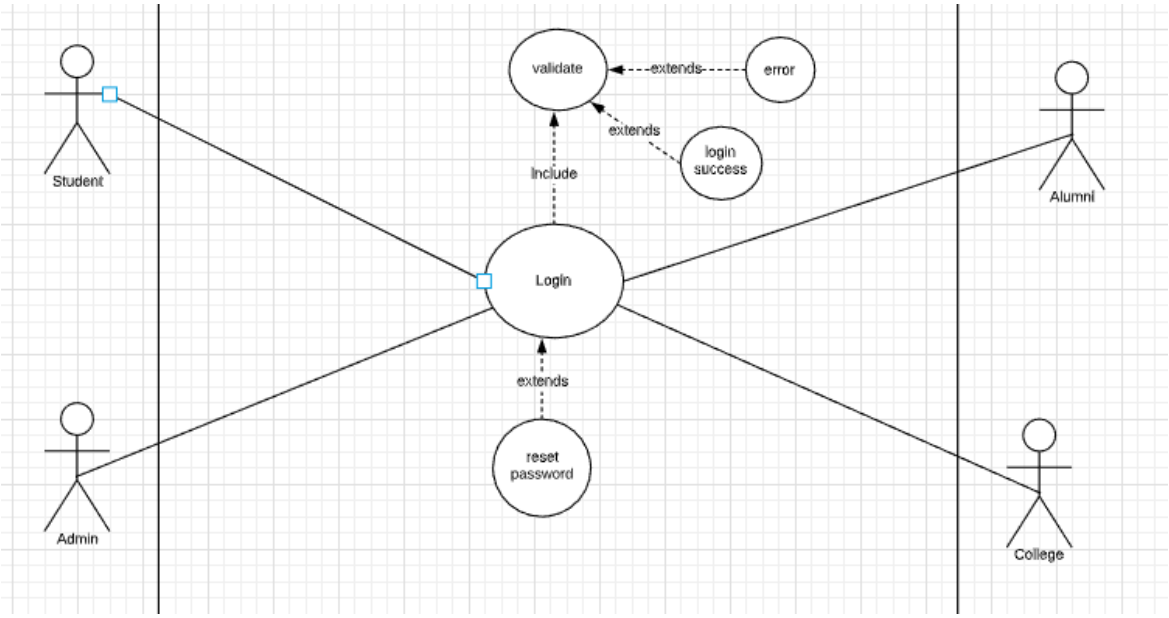


Figure 27 extended use case login

Use case: Login

Actor involved: Admin, College, Student, Alumni

Description: Every user can login themselves with application, but admin will only login with web.

User interaction	System Response
1. User (all) clicks on login page.	2. System Shows login page to users.
3. User enters the valid login credentials.	4. System checks the user's identity and verify user if user verified lands the user to appropriate page. If not shows error response.

Table 3 Extended use case login.

Extended use case diagram for Update profile:

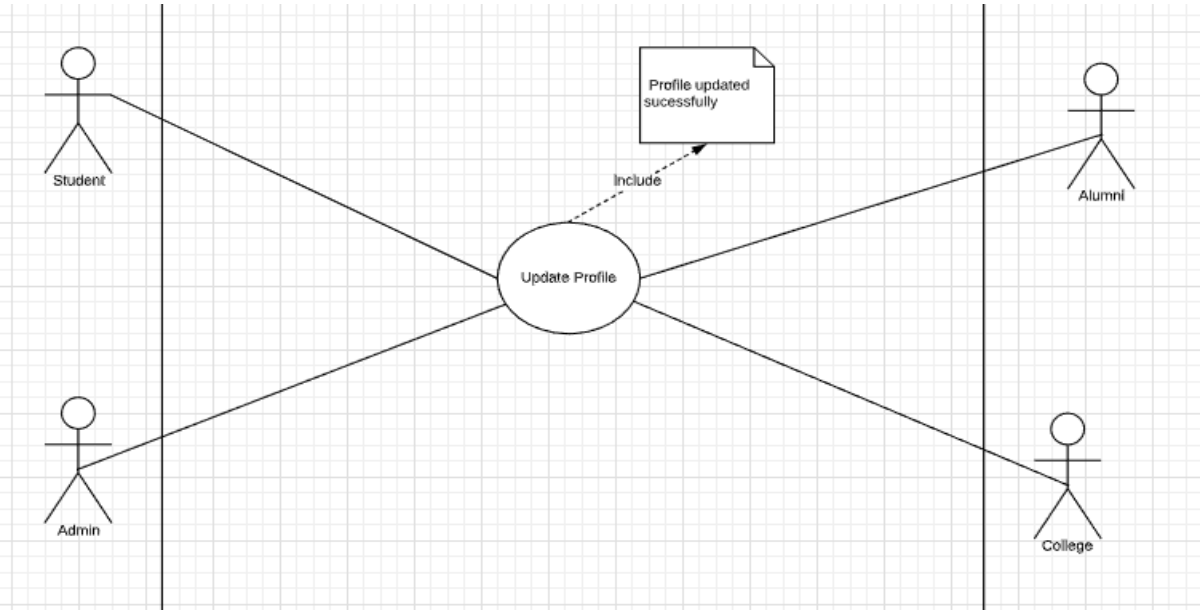


Figure 28 extended use case update profile.

Use case: Update Profile

Actor involved: Alumni, college, student, admin

Description: Every user will be able to update their own profile

Users Interaction	System response
1. User click on view profile option.	2. System shows profile of that particular user.
3. User click on update profile option.	4. System shows update profile page.
5. User edit the data and press update.	6. System updates the user data.

Table 4 Extended use case update profile.

Extended use case diagram for Add achievement:

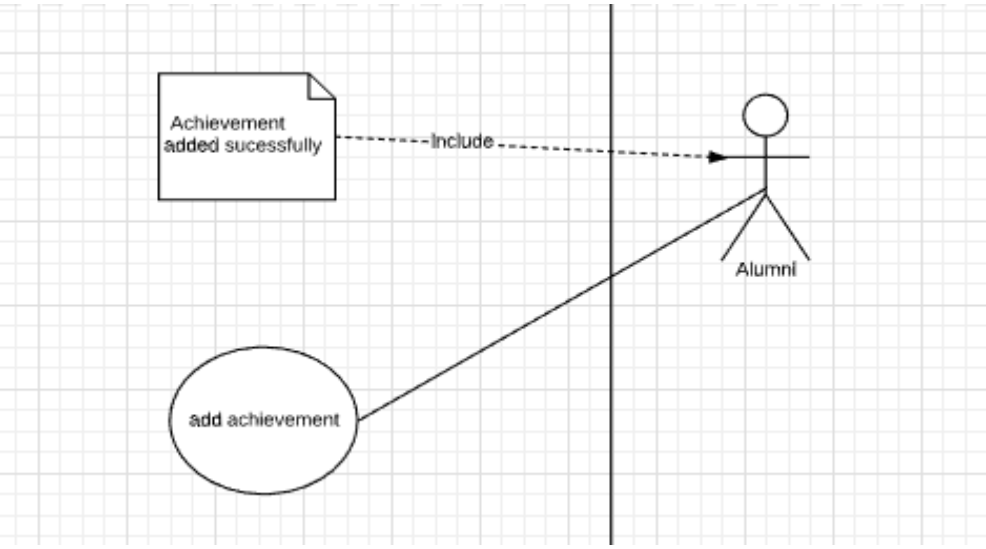


Figure 29 extended use case add achievement

Use case: Add achievement

Actor involved: Alumni

Description: Alumni can only add the achievements they have done but it can be viewed by all the users.

Users interaction	System response
1. User (Alumni) click on view profile option.	2. System shows profile of that alumni.
3. User click on add achievements.	4. System shows achievement page.
5. User add achievements and press submit.	6. System add the achievement to the profile.

Table 5 Extended use case add achievement.

Extended use case diagram for Events:

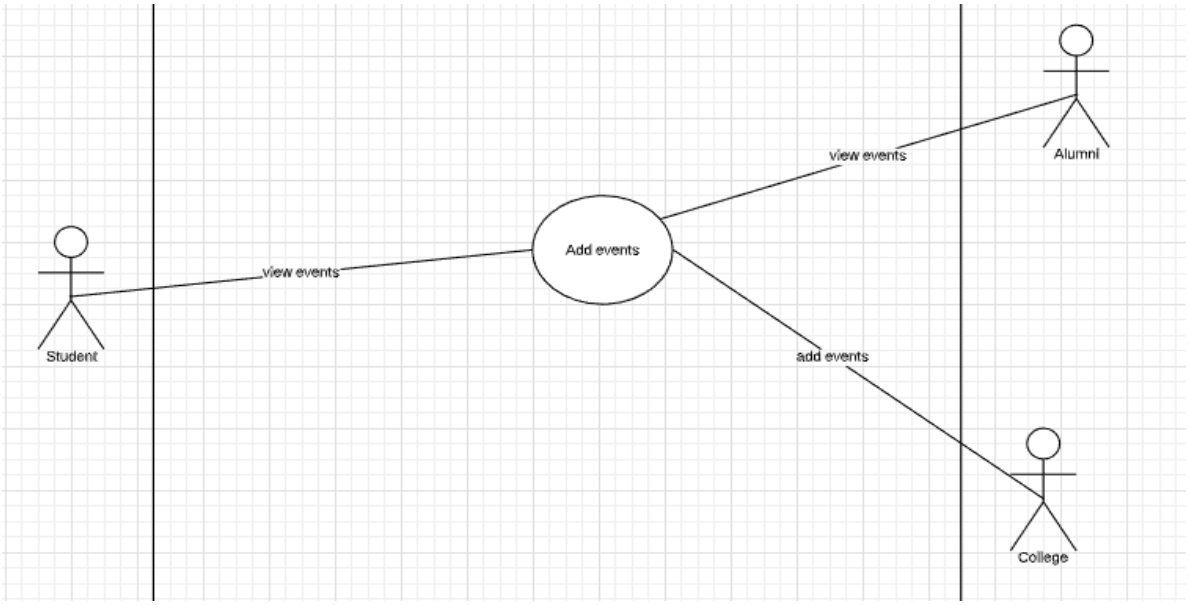


Figure 30 extended use case events

Use case: Events

Actor involved: College, Alumni, Students

Description: College can add new events and alumni can view the events.

Users Interaction	System response
1. User (College) open event tab from app.	2. System shows event page.
3. College adds events and press submit.	4. System gets the event and notify to alumni and students.
5. Alumni and student open event page.	6. System shows new events.

Table 6 Extended use case events.

Extended use case diagram for Search profile:

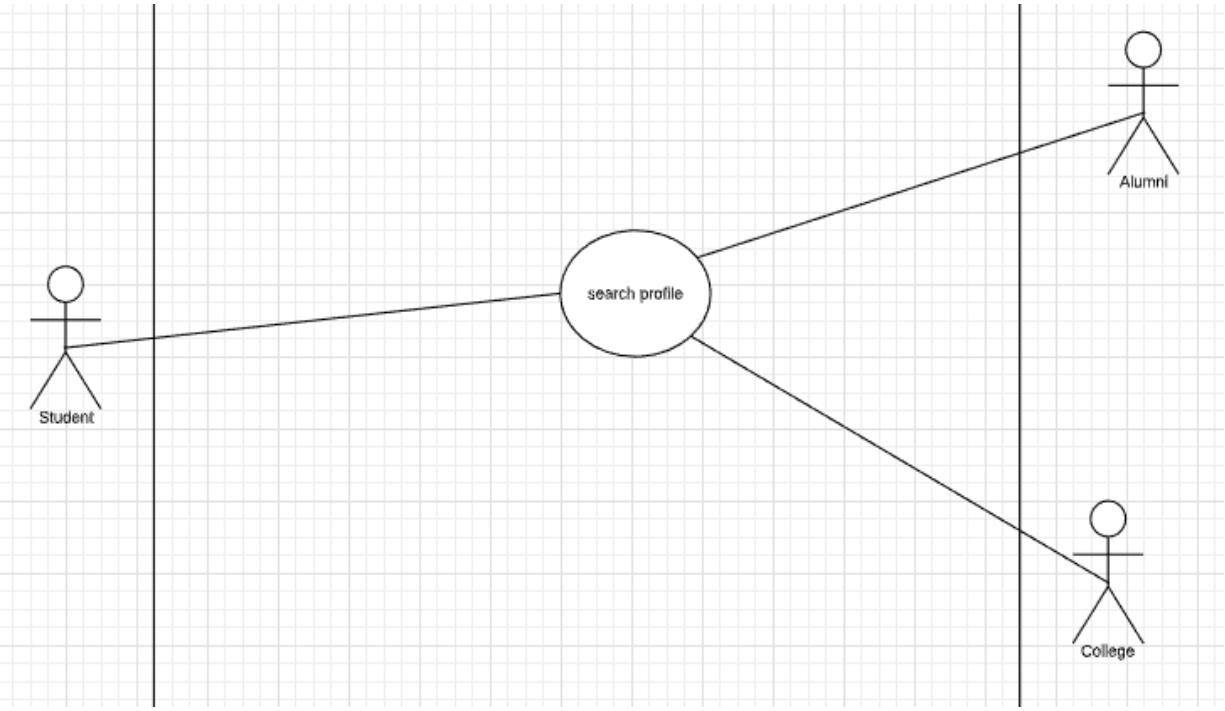


Figure 31 extended use case search profile.

Use case: Search Profile

Actor involved: College, students, Alumni

Description: All the users can view, and search profiles accept of admin. Admin can only view through web.

Users interaction	System response
1. User (College, student, alumni) clicks on search alumni.	2. System shows all the list of alumni.
3. User type alumni name or search and clicks on it.	4. System opens the respective alumni's profile page.

Table 7 Extended use case search profile.

Extended use case diagram for send message:

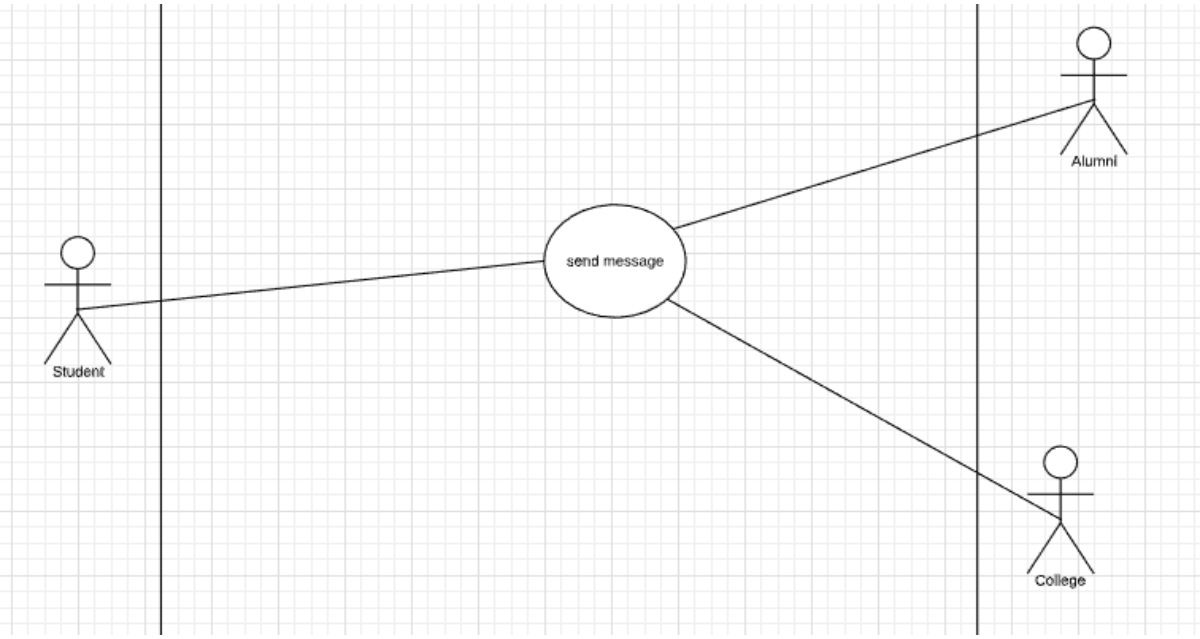


Figure 32 extended use case send message.

Use case: Send message

Actor involved: Alumni, college, student

Description: All users expect admin can send message to each other.

User's interaction	System response
1. User clicks on message tab.	2. System shows message page.
3. User select one person and send message.	4. System forward that message to appropriate user.

Table 8 Extended use case send message.

3.6.5: Entity Relationship Diagram

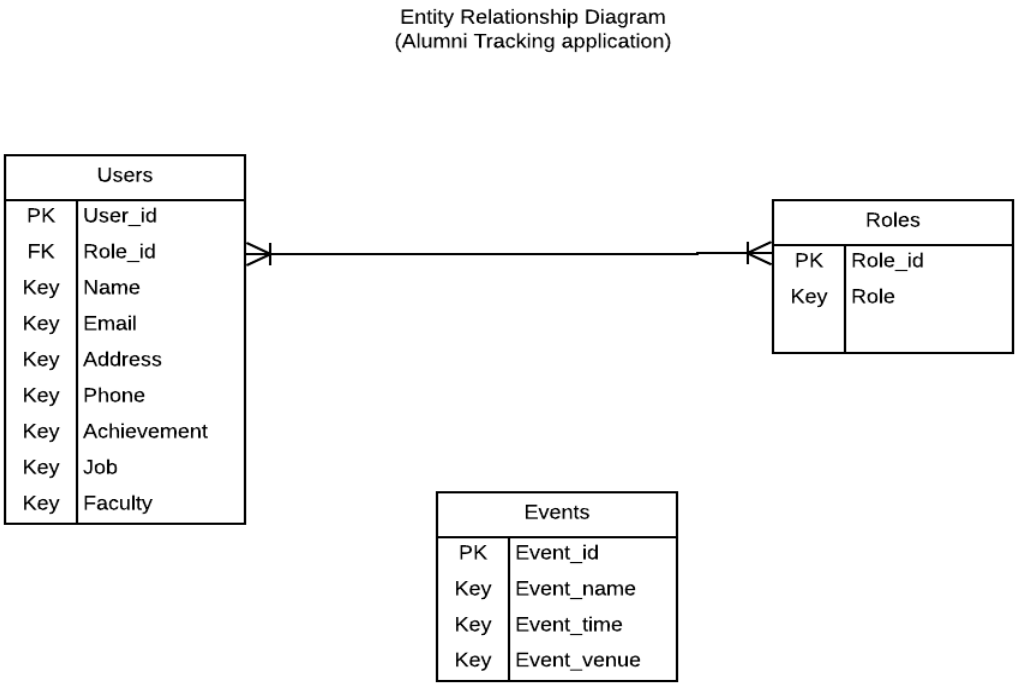


Figure 33 Entity relationship diagram

(tutorials point, 2018)

Description of Entity Relationship Diagram:

Here, after the normalization up to 2nf three tables are separated. Users table include all data of alumni, student, college and admin. Roles table includes the role and role id where alumni, student, college are separated with the code. A user can have multiple role because a student can become alumni in future in that case a user has multiple role, so role table is inherited with users table.

As we don't have any connection with event table because everybody gets event notifications and no necessary of having links between tables.

3.6.6: Sequence diagram:

Register user:

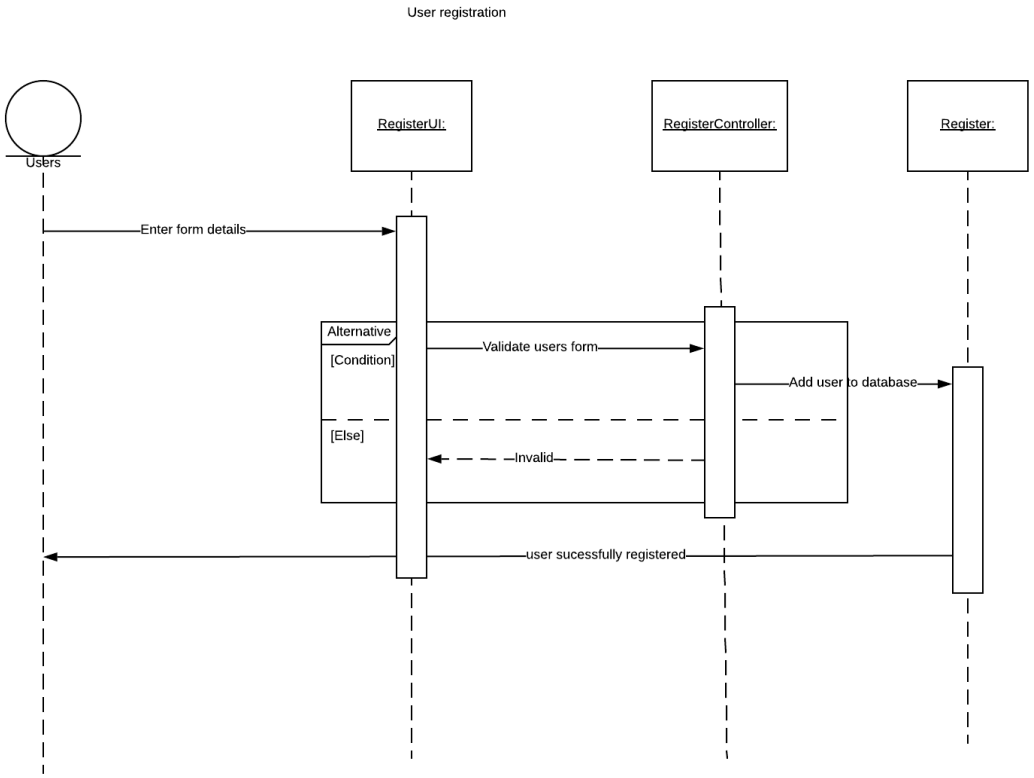


Figure 34 sequence diagram of register user.

(smartdraw, 2017)

Above sequence diagram shows the sequential description of how the user and system registers to the system.

Login User:

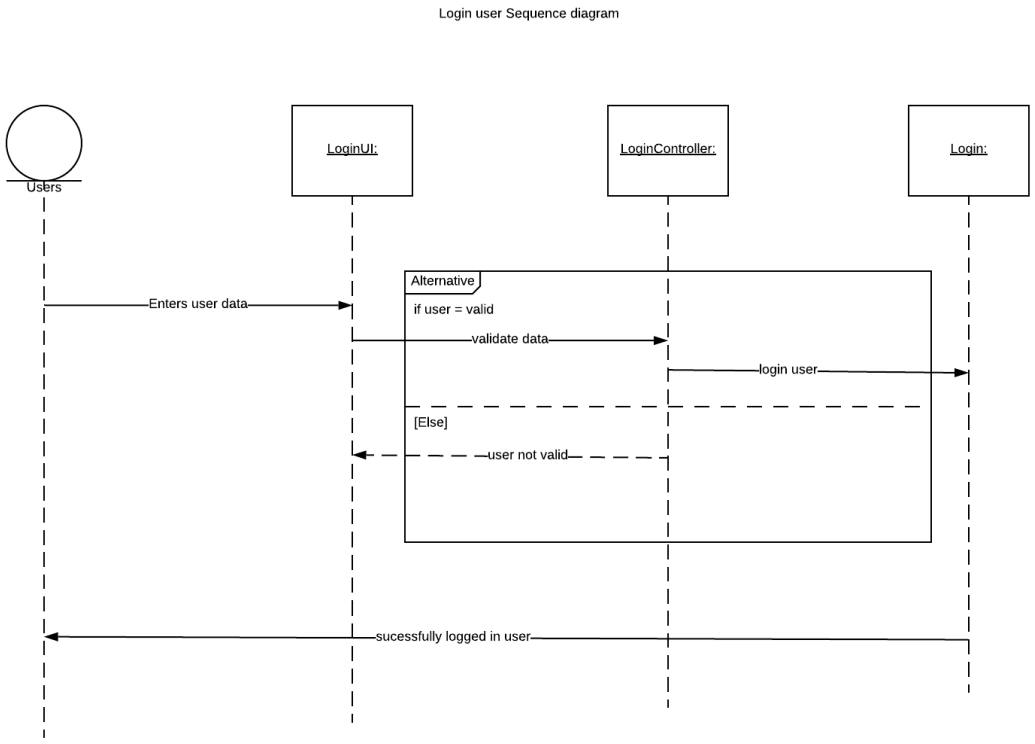


Figure 35 sequence diagram of login user.

Above sequence diagram shows the sequential description of how the user login themselves to the system.

Create Events:

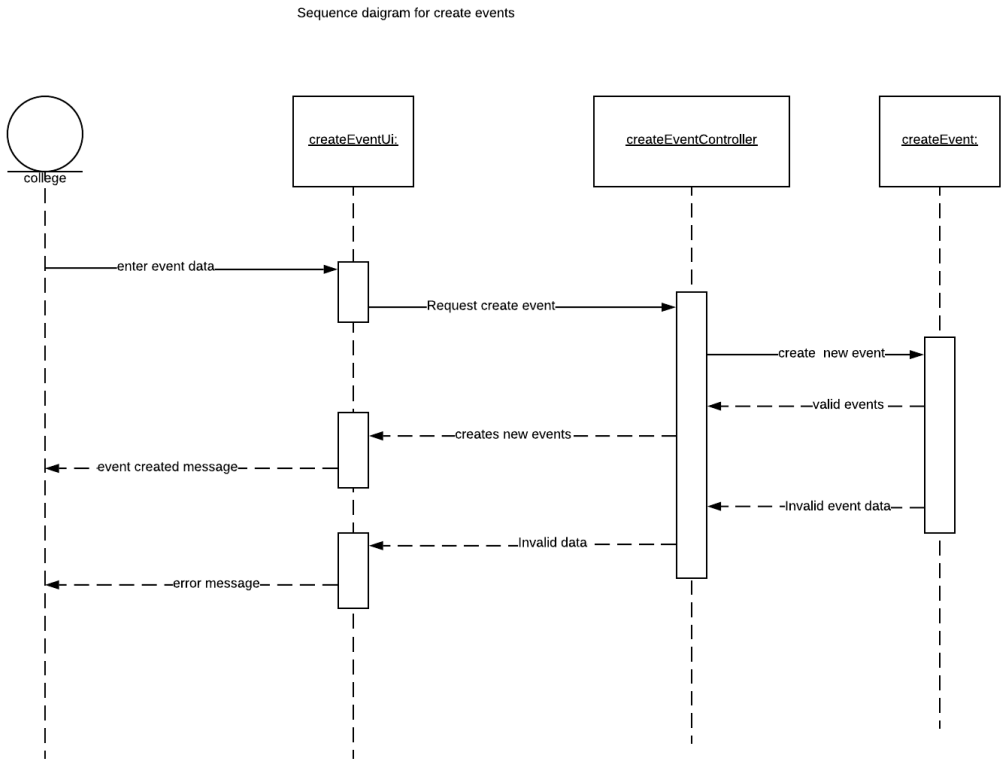


Figure 36 sequence diagram of create events.

Above sequence diagram shows the sequential description of how the college creates a new event from the system.

Search User:

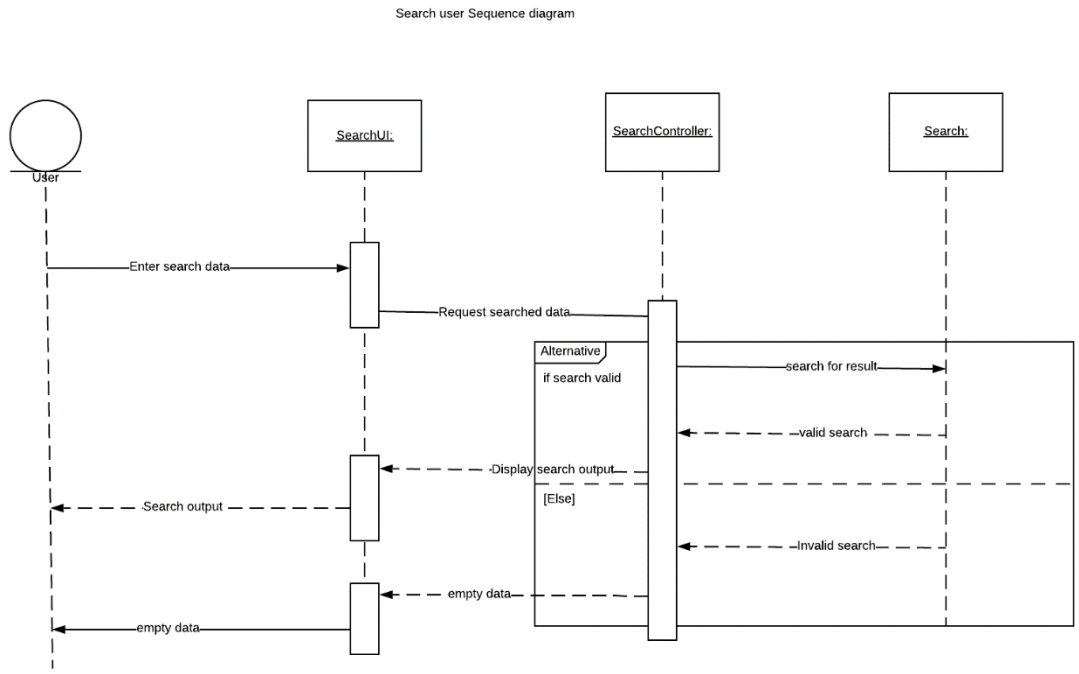


Figure 37 sequence diagram of search user.

Above sequence diagram shows the sequential description of how the user search other users from the system.

Update User:

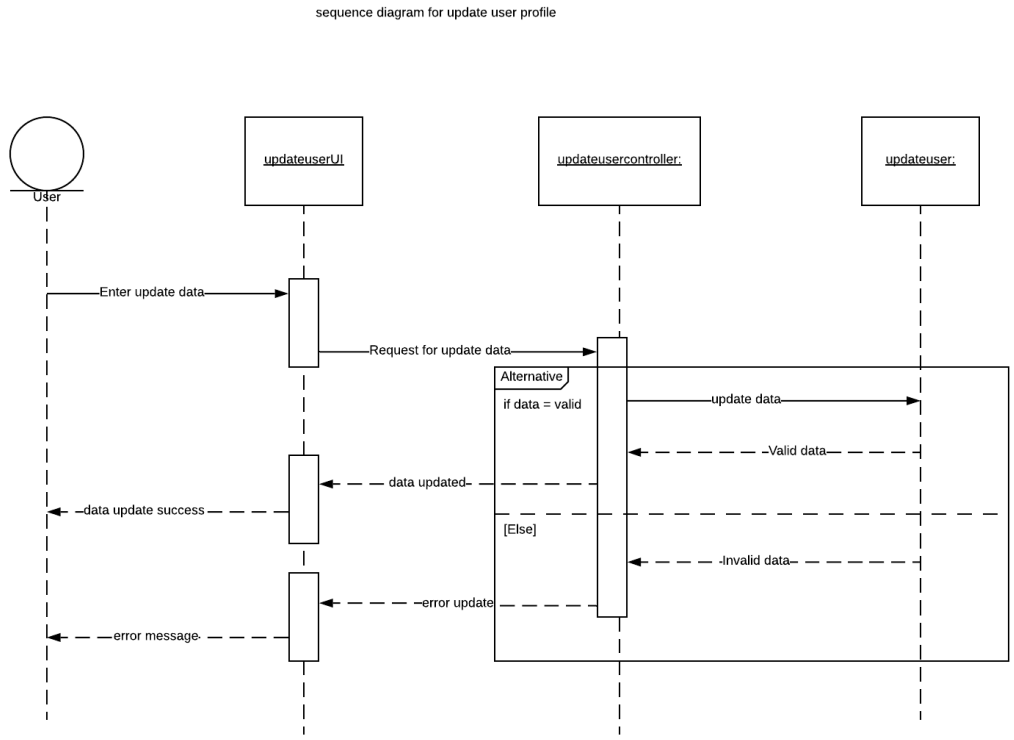


Figure 38 sequence diagram of update user.

Above sequence diagram shows the sequential description of how the user updates their information once logged in to the system.

3.6.7: Collaboration diagram:

Register User:

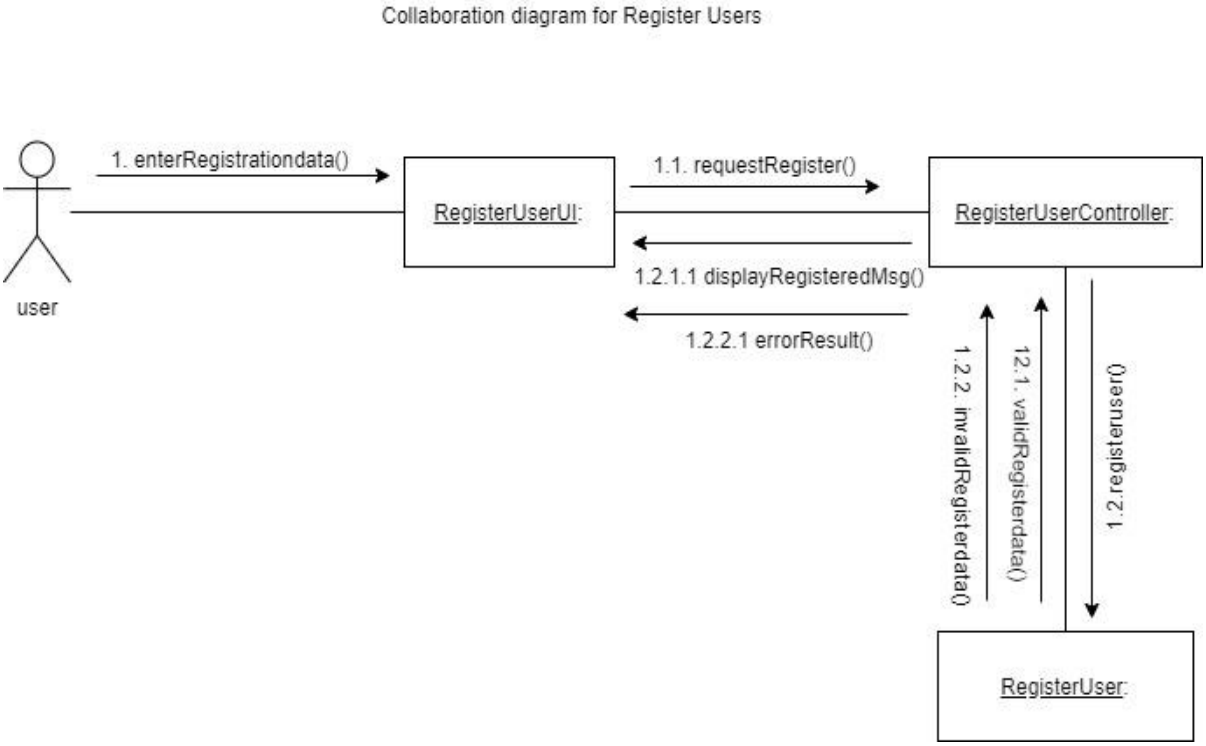


Figure 39 collaboration diagram of register user.

(Visual paradigm, 2020)

Here, the above figure shows the collaboration diagram of how the user and system interacts when registering a new user to the system.

Login User:

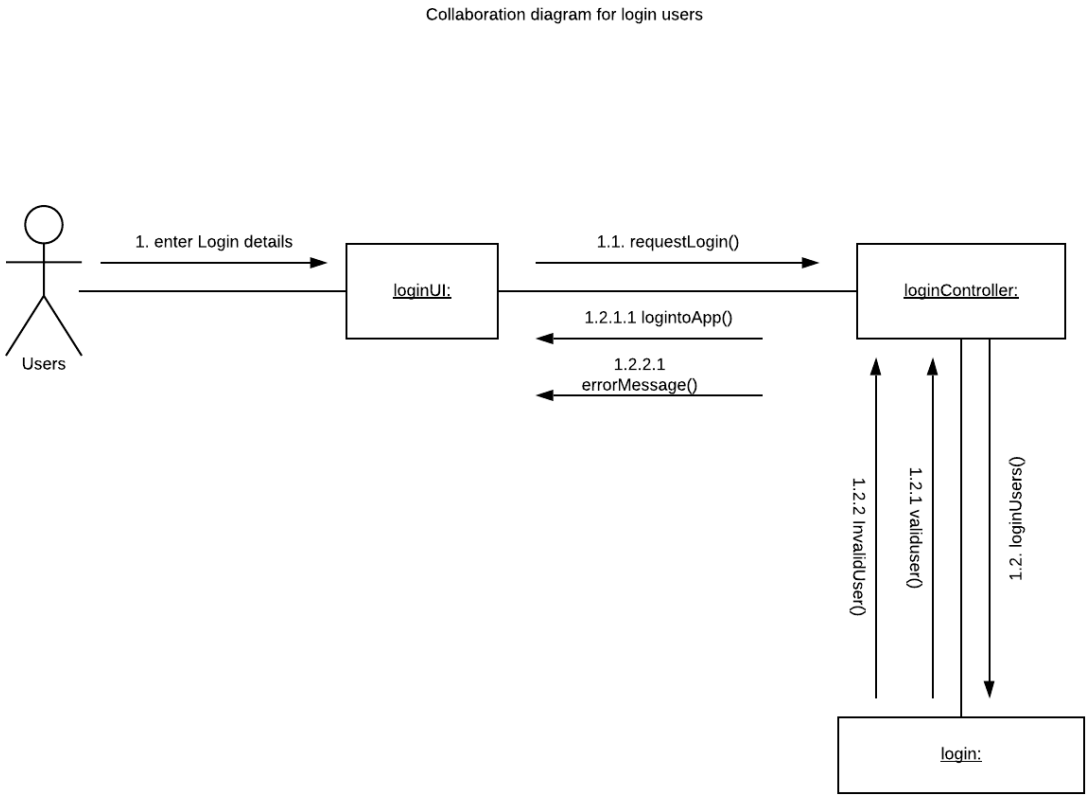


Figure 40 collaboration diagram of login user.

Here, the above figure shows the collaboration diagram of how the user login to the system. Here actors are all the users.

Search User:

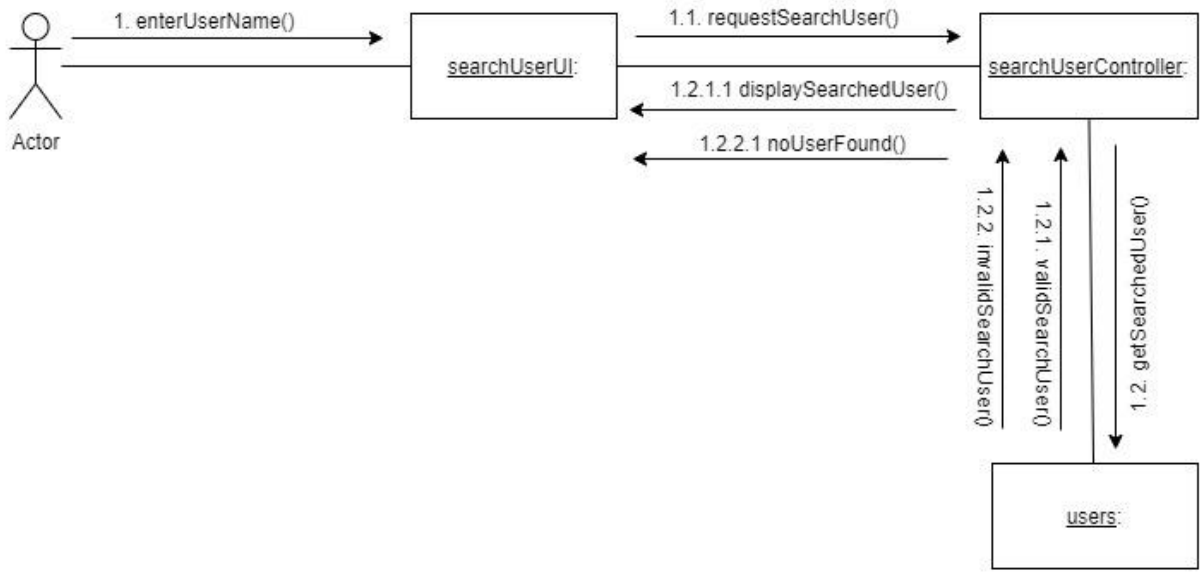


Figure 41 collaboration diagram of search user.

Here, the above figure shows the collaboration diagram of how the user search another user from the system.

Update user:

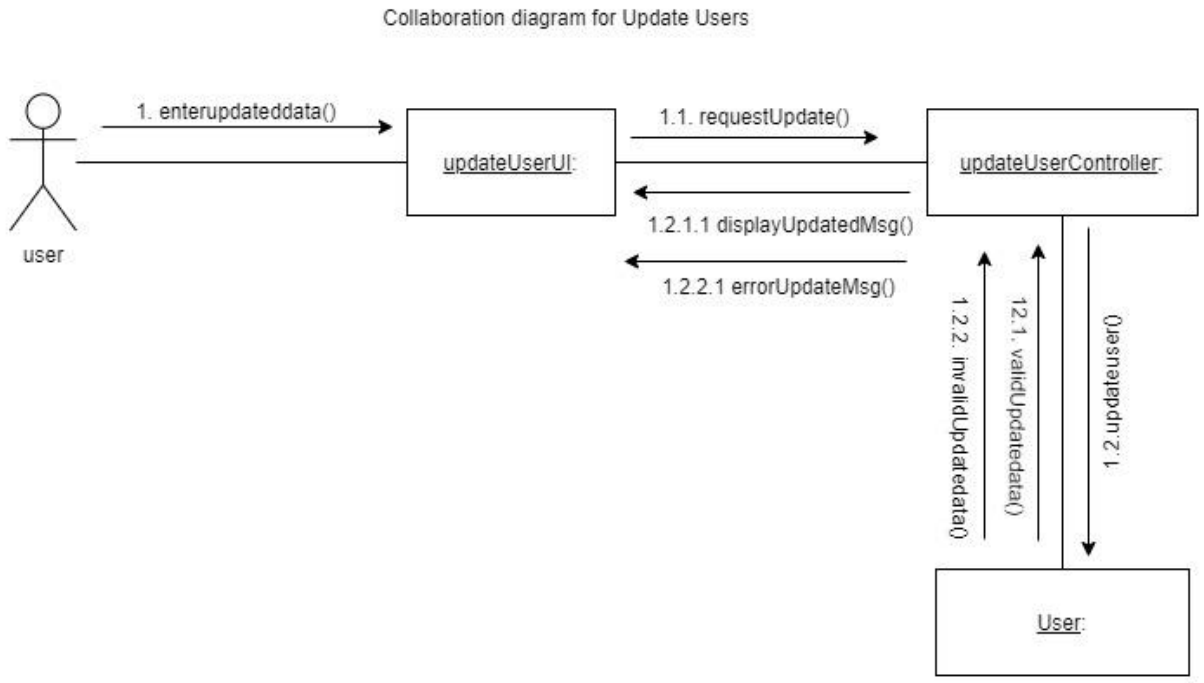


Figure 42 collaboration diagram of update user.

Here, the above figure shows the collaboration diagram of how the user updates their information after logged in the system.

Create Event:

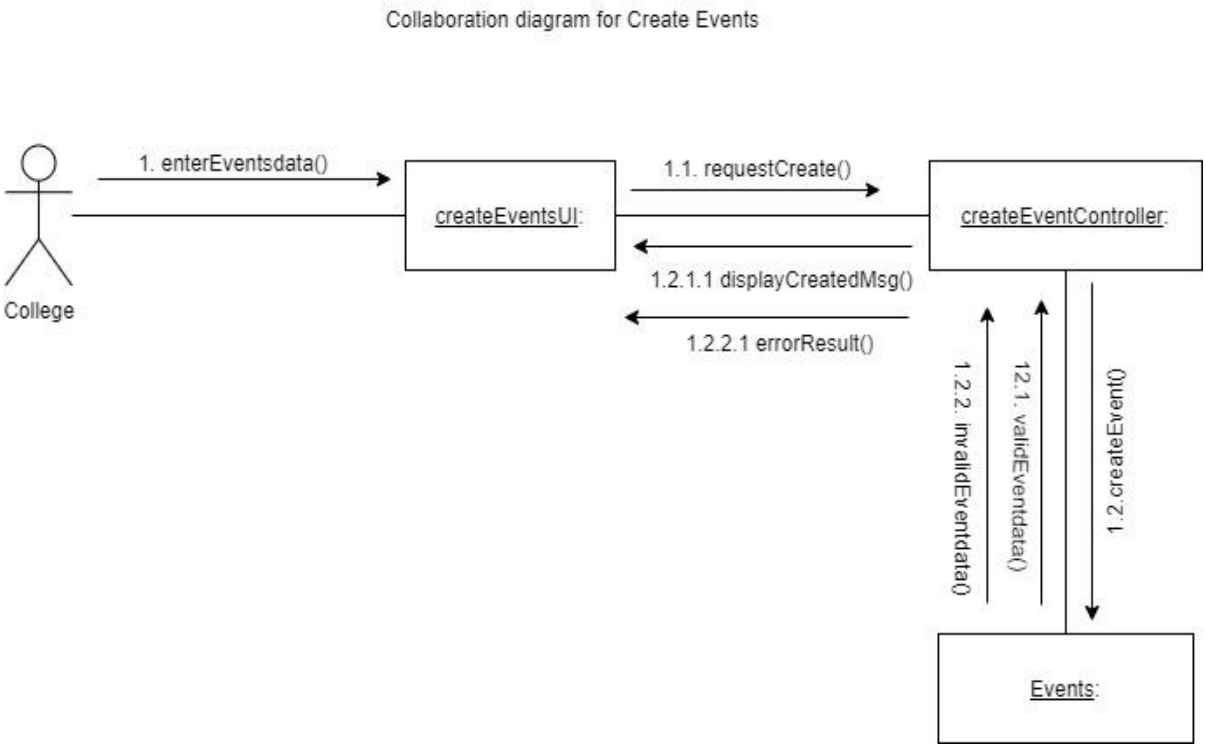


Figure 43 collaboration diagram of create events.

Here, the above figure shows the collaboration diagram of how the college creates new events. College is the only actor who interact to this part of the system.

Search Event:

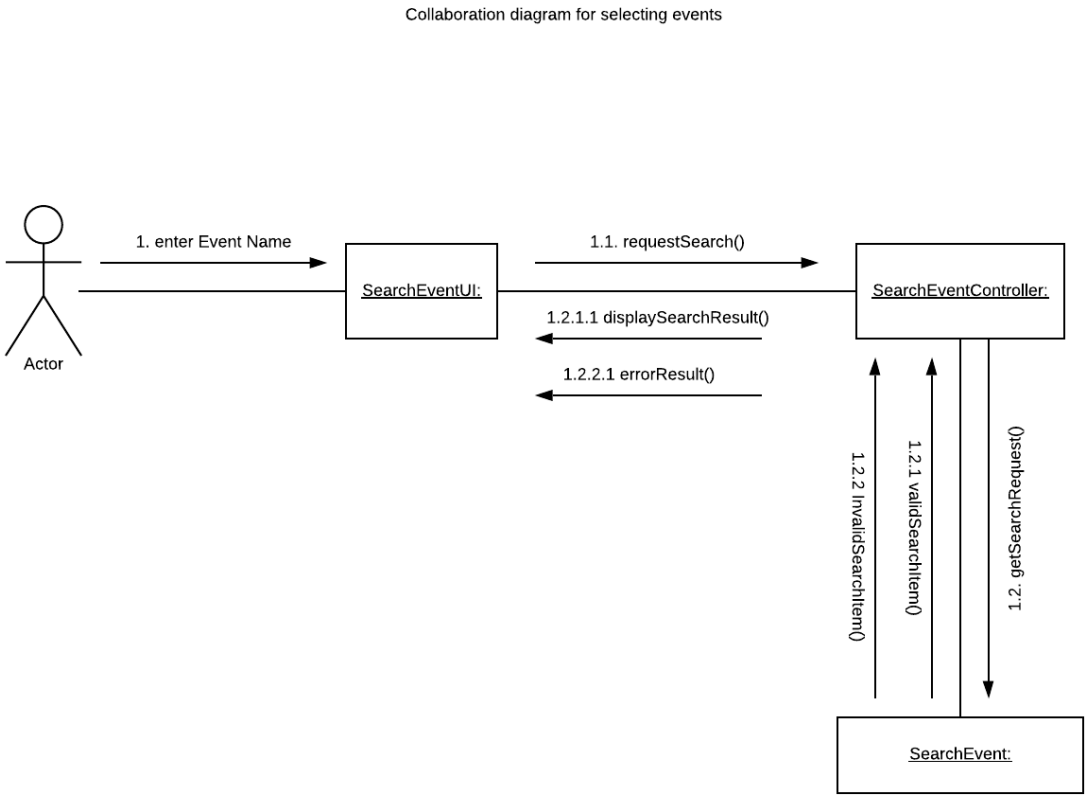


Figure 44 collaboration diagram of search events.

Here, the above figure shows the collaboration diagram of how the user search events created by college from events page.

3.6.8: Activity diagram:

Register User:

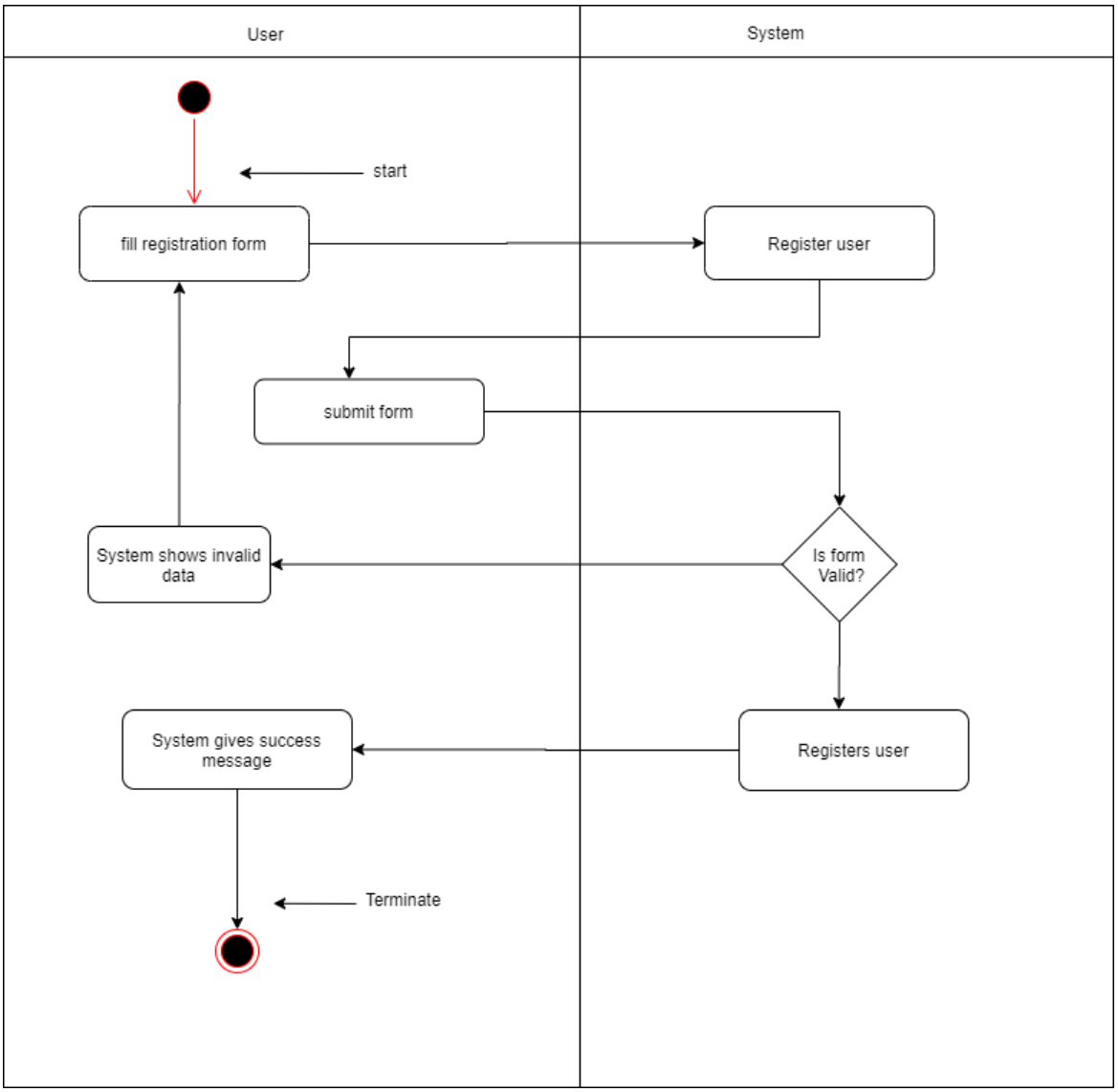


Figure 45 activity diagram of register user.

(guru99, 2019)

Figure above shows the activity diagram of registering user to the system. The diagrams show how the user interact to the system and system gives response while a new user is being registered.

Login User:

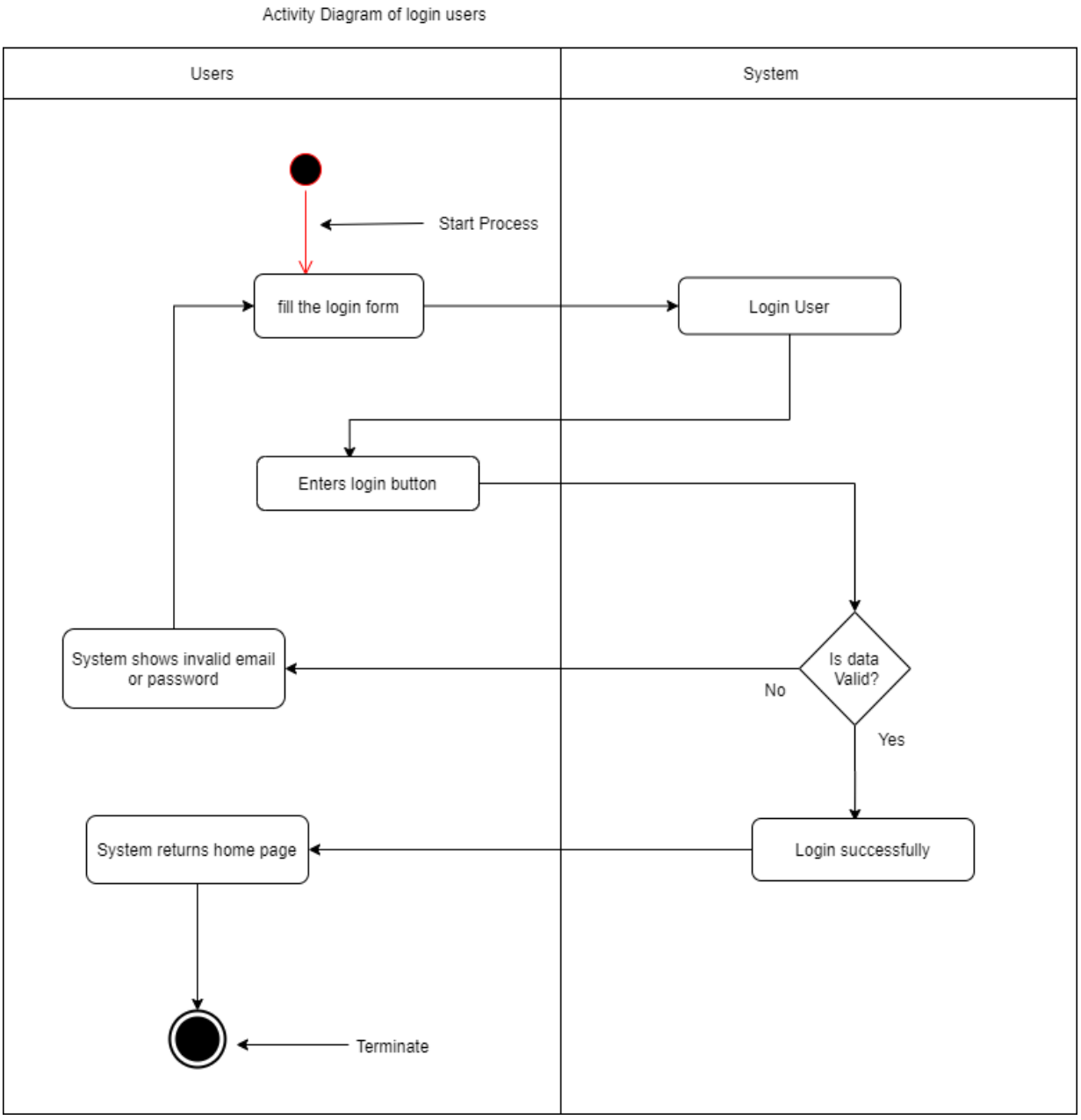


Figure 46 activity diagram of login user.

Figure above shows the activity diagram of login user of the system. The diagrams show how the user interact to the system and system gives response while the user login to the system.

Update user:

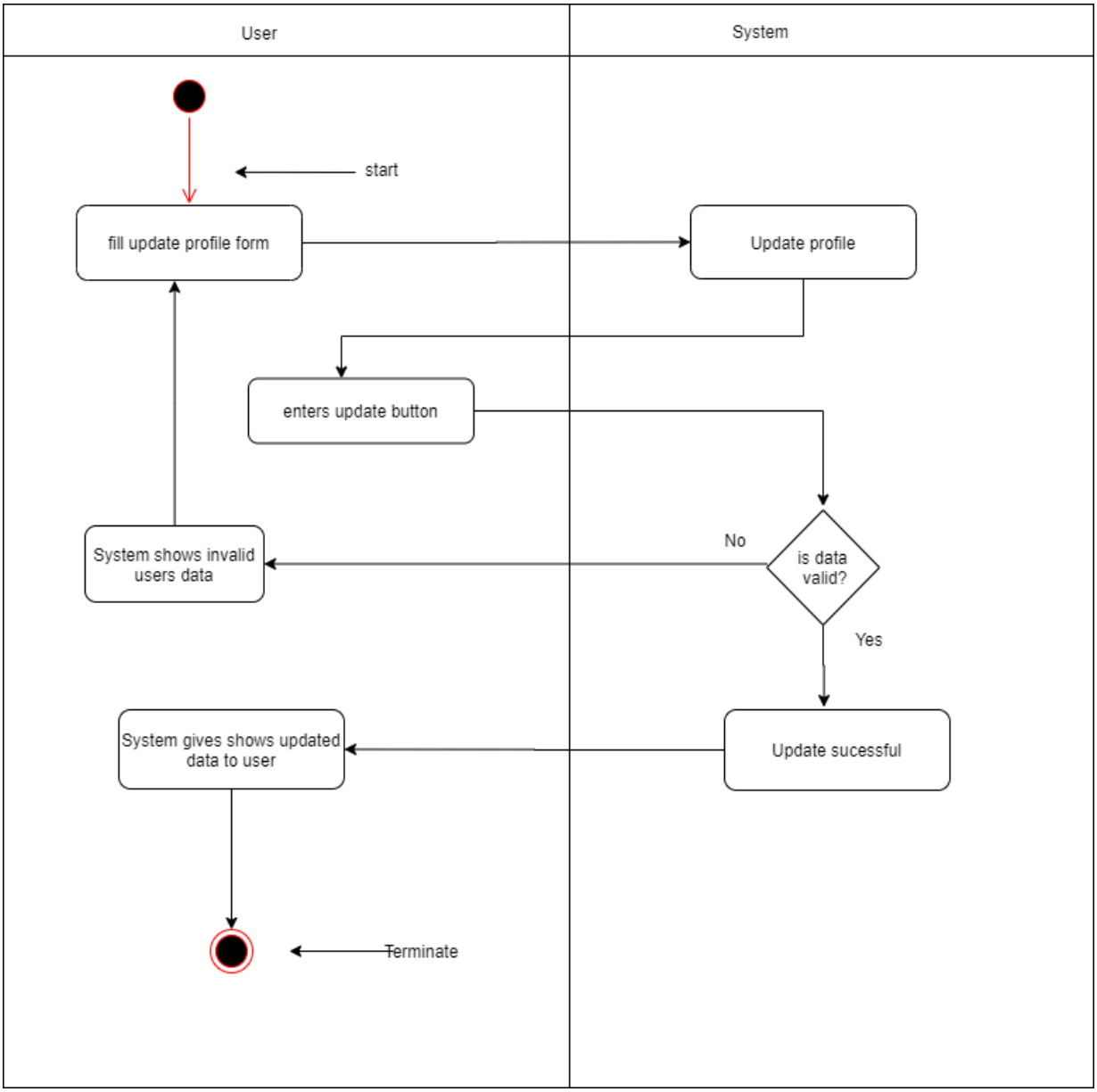


Figure 47 activity diagram of update user.

Figure above shows the activity diagram of updating users' profile. Above diagram shows how the user interact to the system and how the system response to him while he/she updates his profile and information.

Search user:

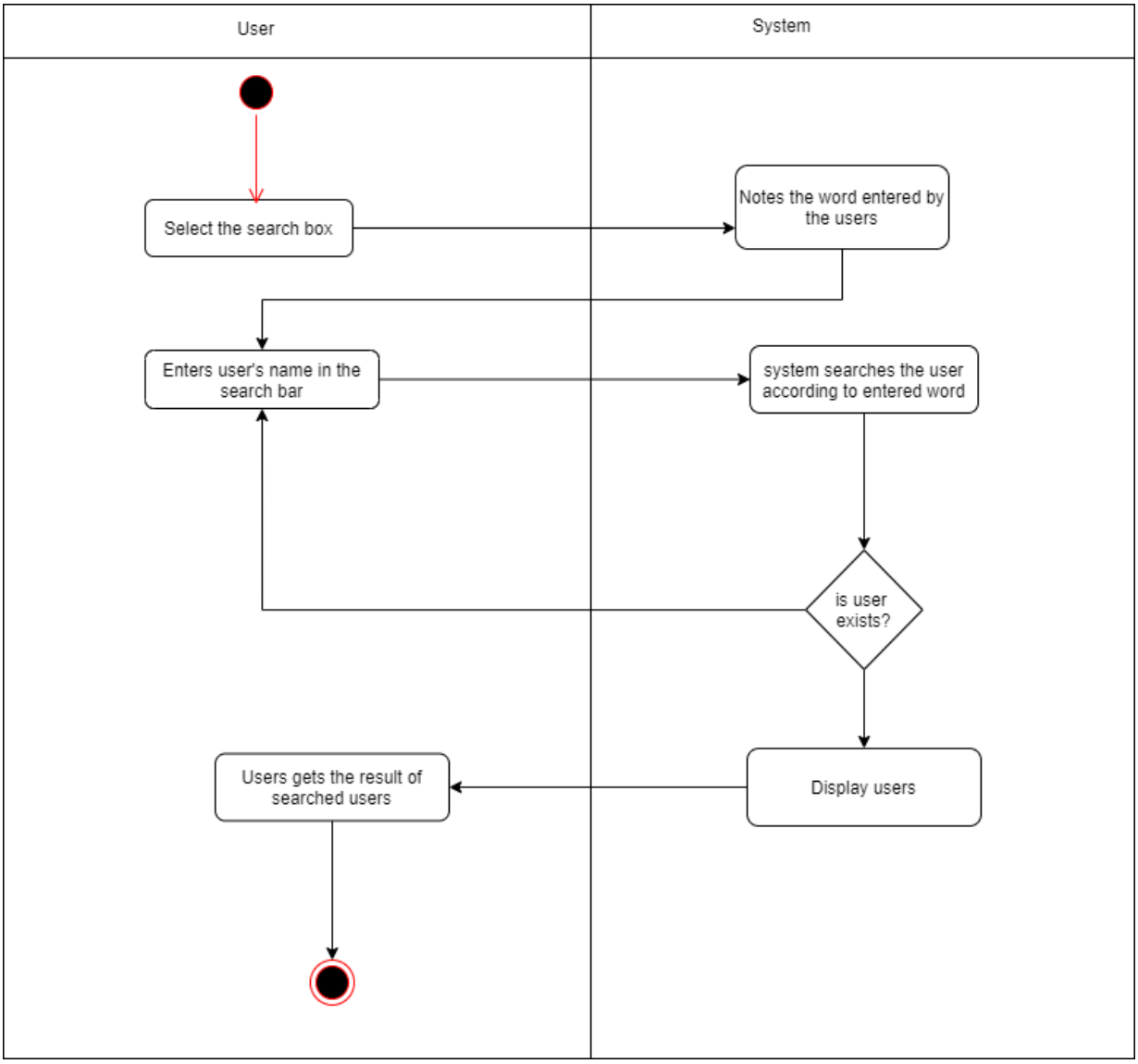


Figure 48 activity diagram of search user.

Figure above shows the activity diagram of searching user by a logged in user. Above diagram shows how the user interact to the system and how the system response to them while he/she searches a specific user from the search user UI of the application.

Create Event:

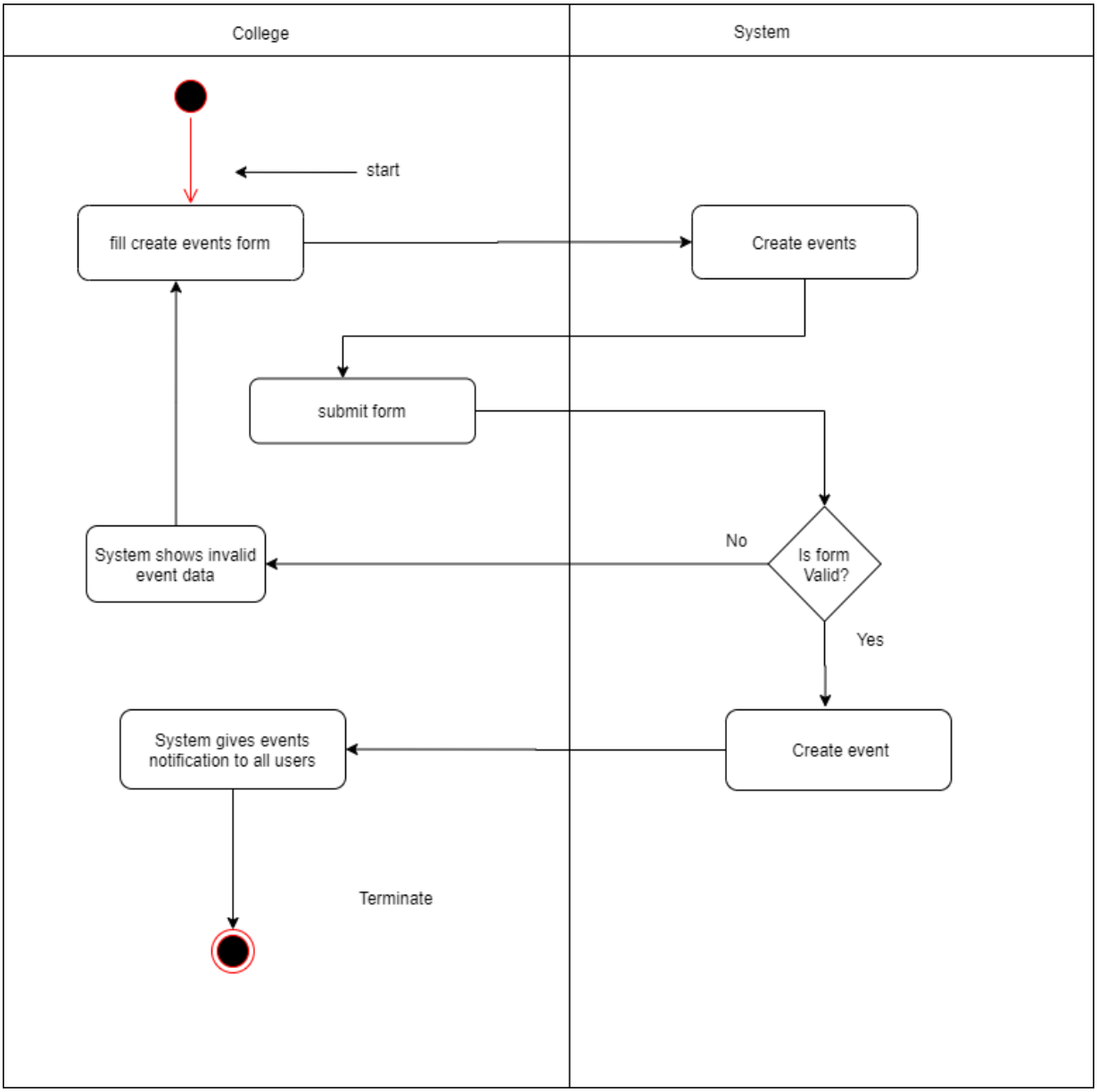


Figure 49 activity diagram of create events.

Figure above shows activity diagram of creating events by a college. The diagram shows how a college interact to the system and what system responses while creating a new event to the system.

Search Event:

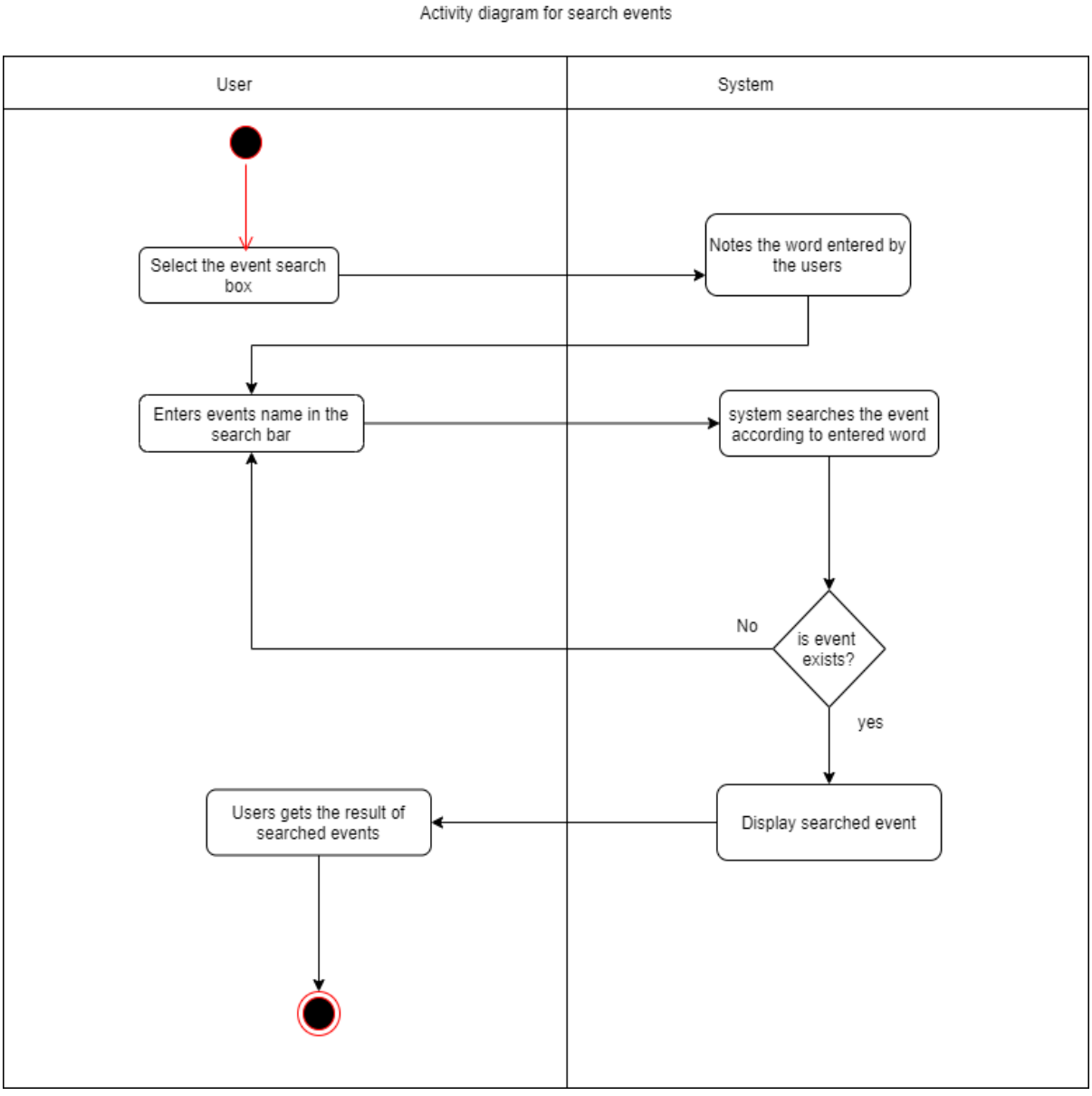


Figure 50 activity diagram of search events.

Figure above shows the activity diagram of searching events by any users from the system. The diagrams show how the user interact to the system and what system gives response while searching for events.

3.6.9: Data Flow diagram:

DFD level 0:

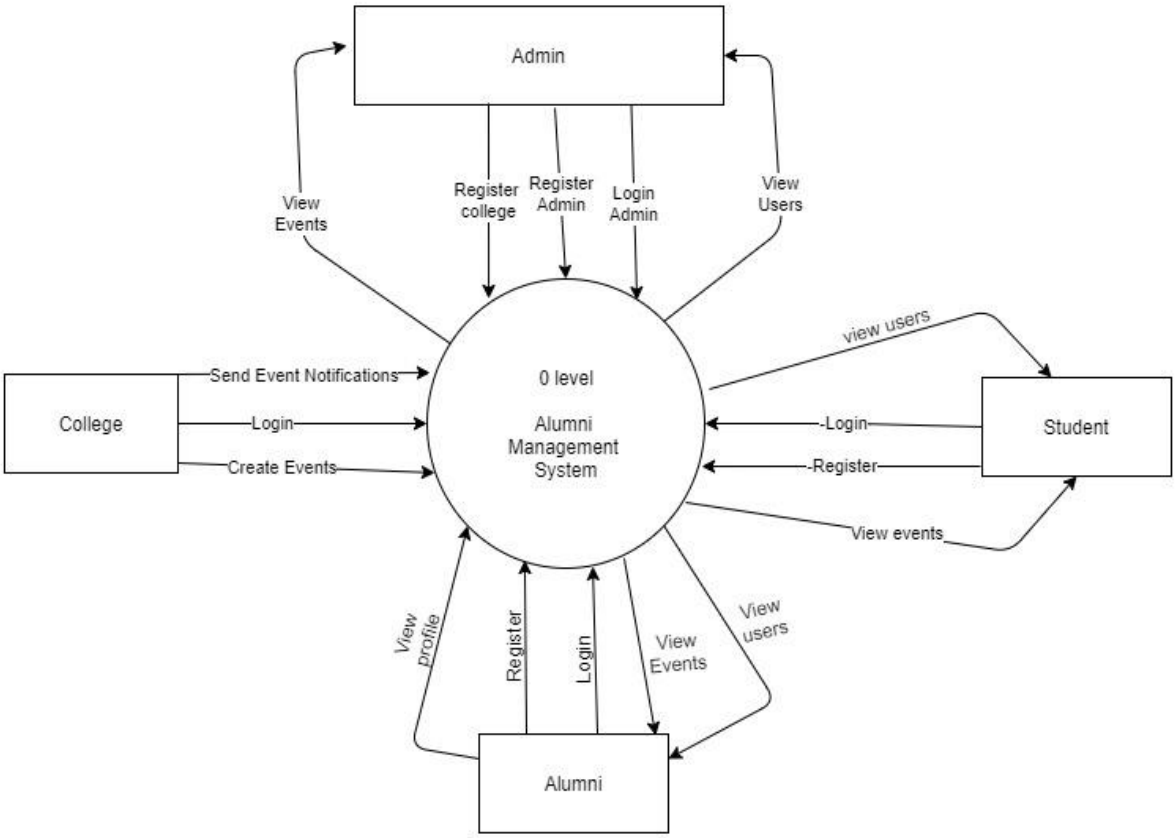


Figure 51 DFD level 0 of system.

DFD level 1:

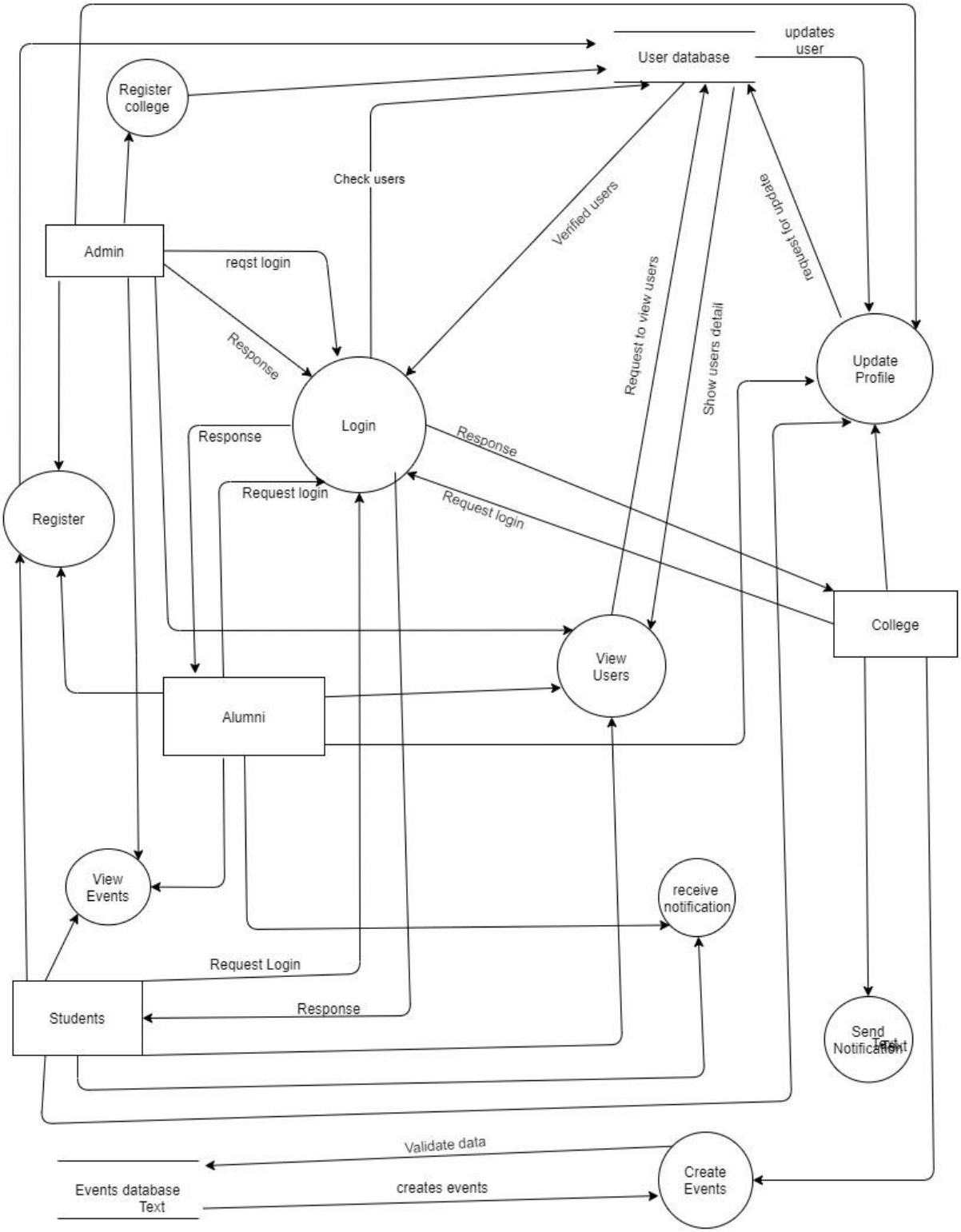


Figure 52 DFD level 1 of System.

3.7: Implementation:

Mobile Application development:

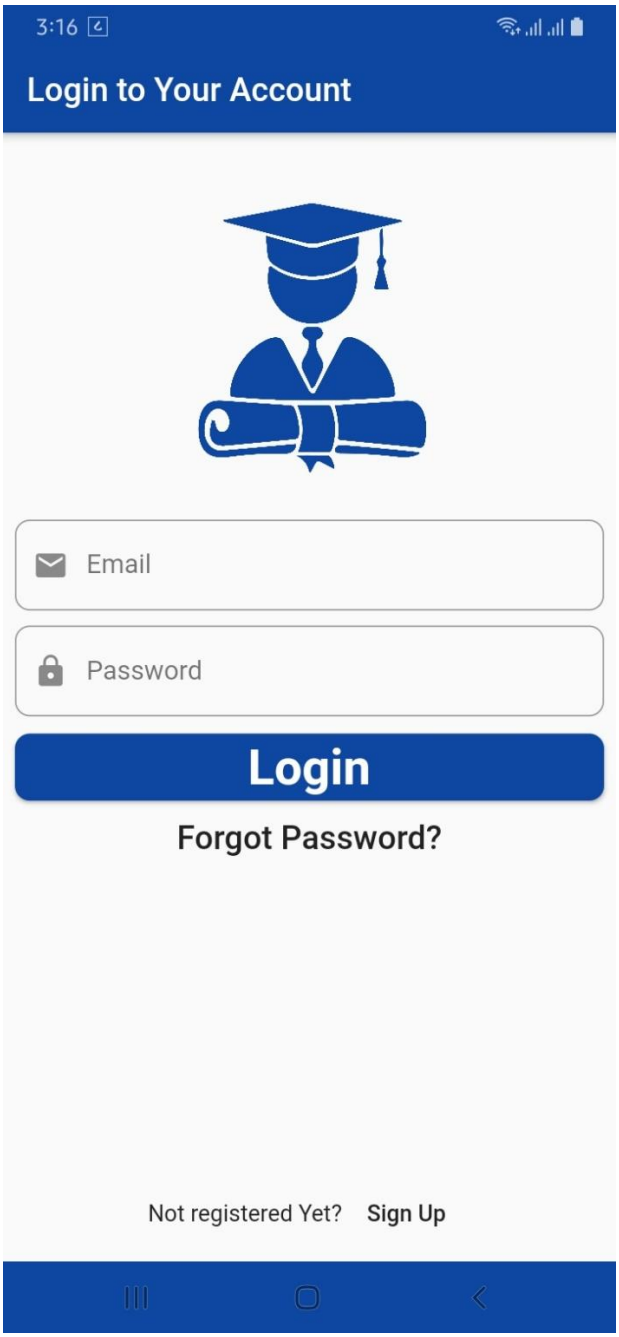


Figure 53 Login page UI.

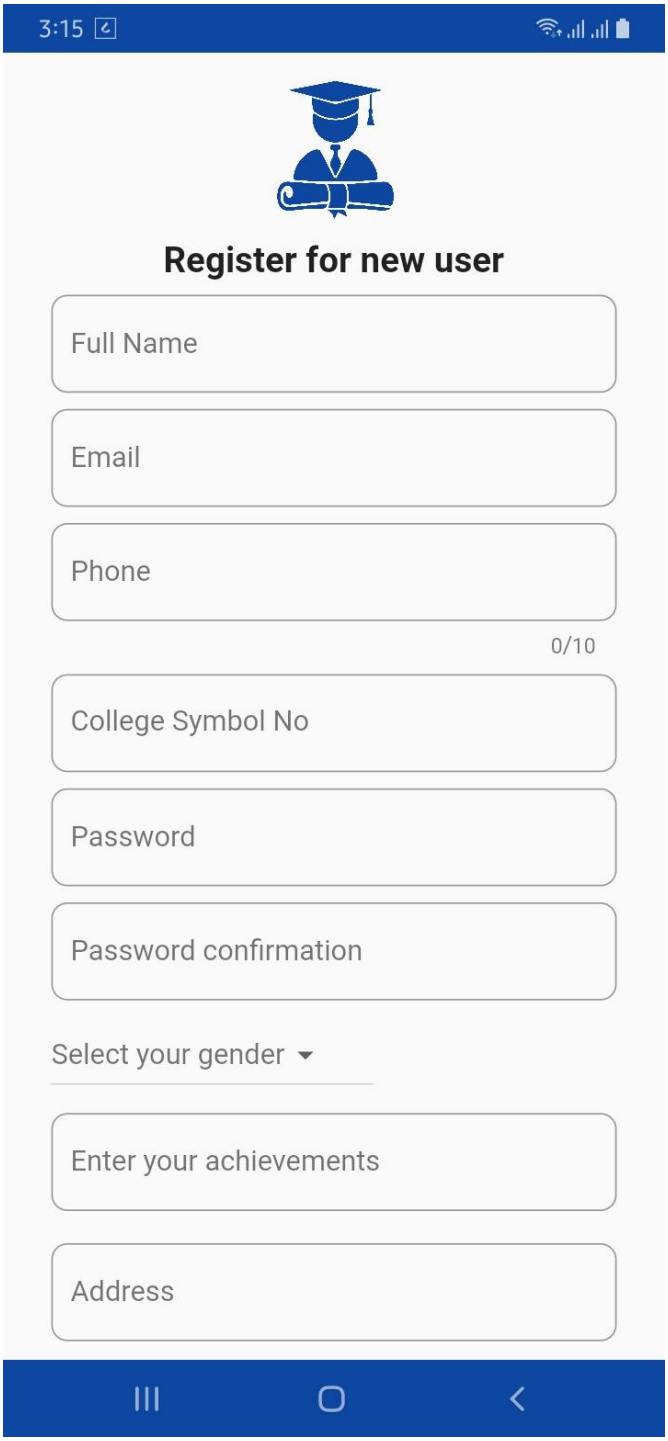


Figure 54 Registration UI

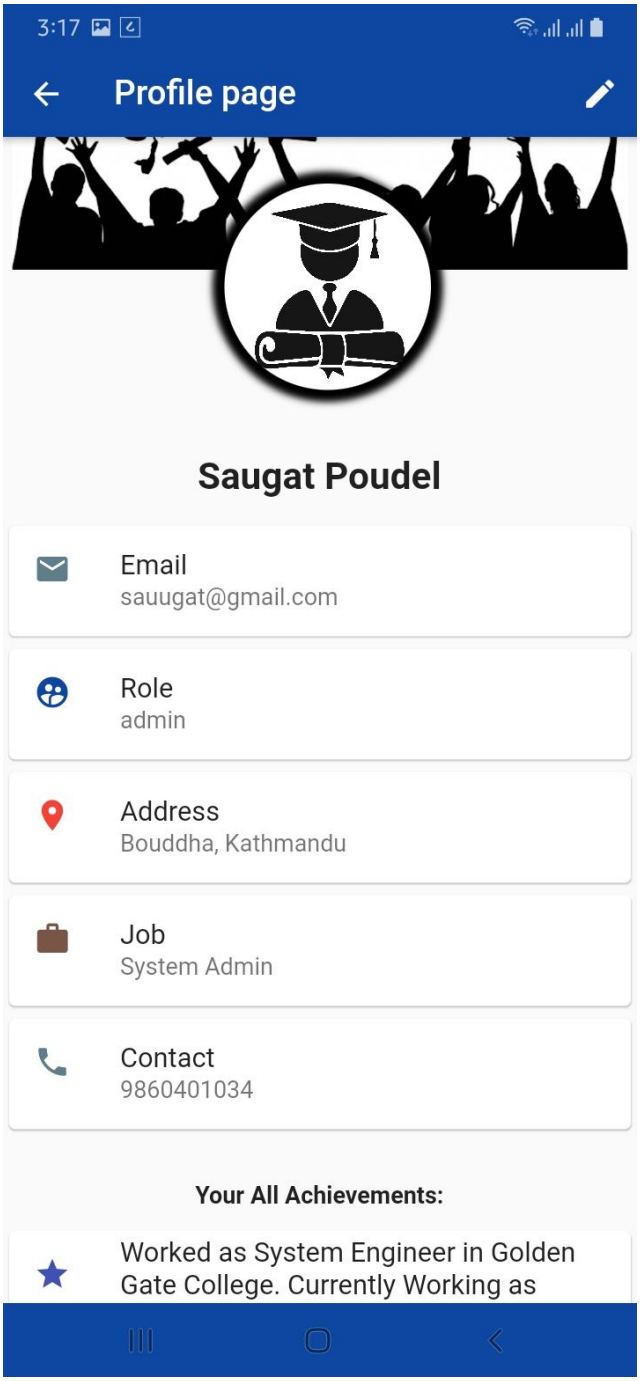


Figure 55 Profile UI.

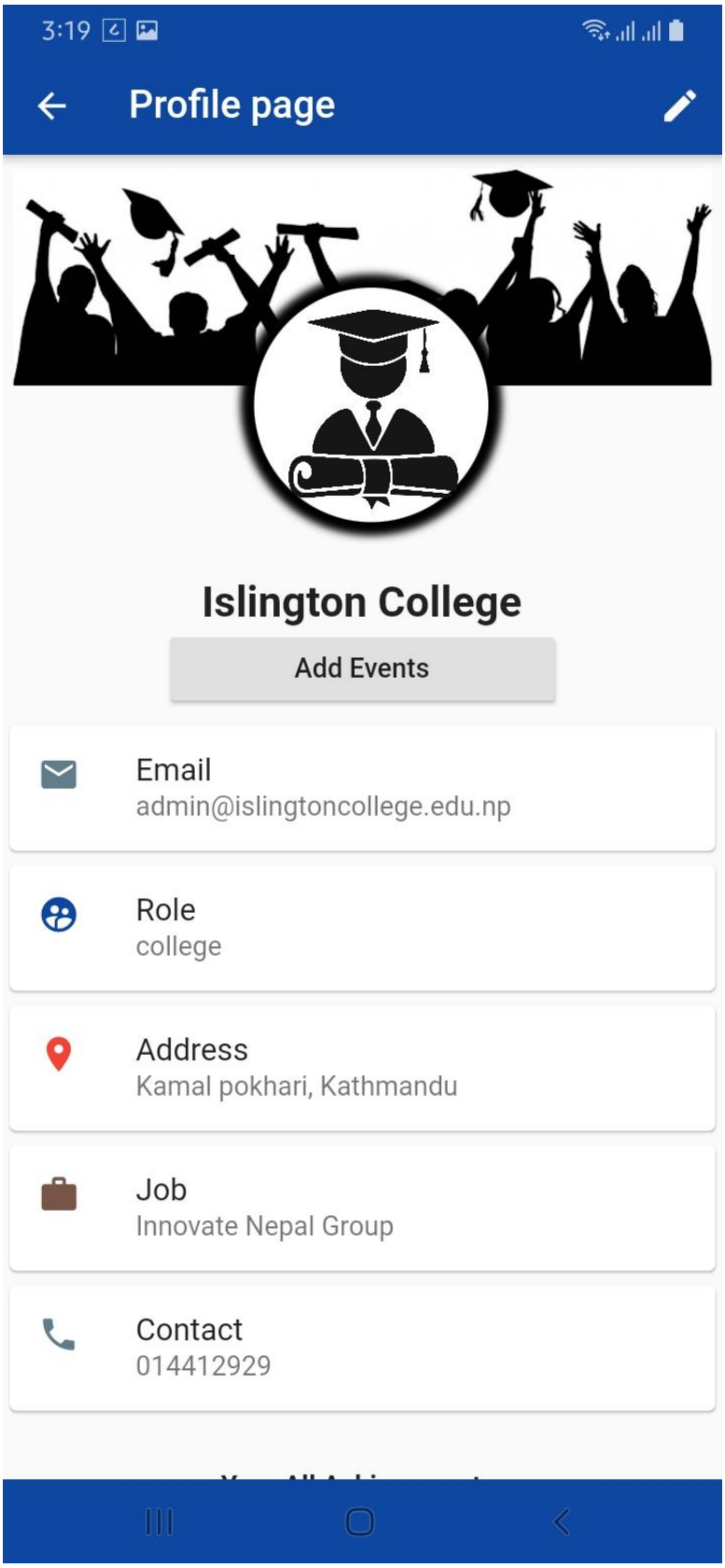


Figure 56 college profile UI.

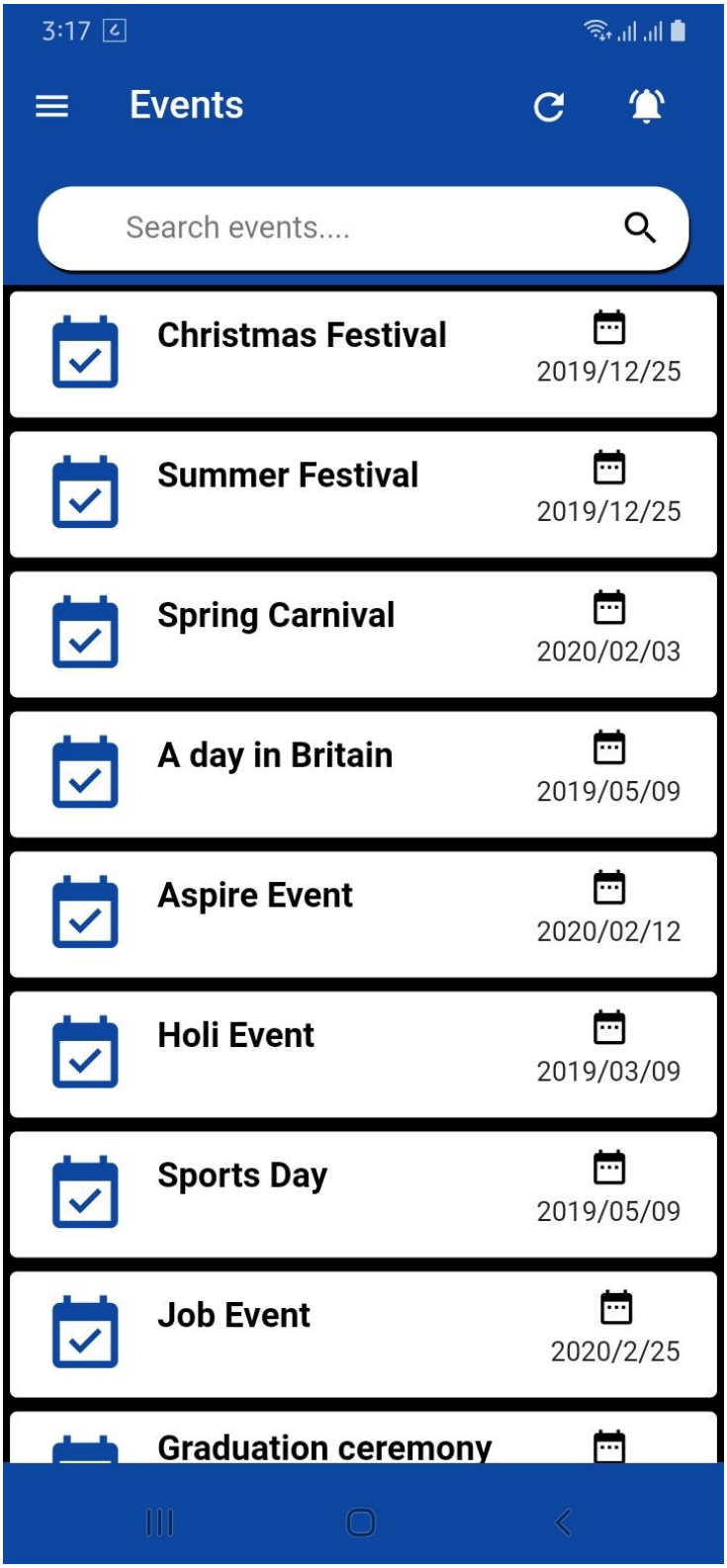


Figure 57 Home page UI.

Figure 58 event detail UI.

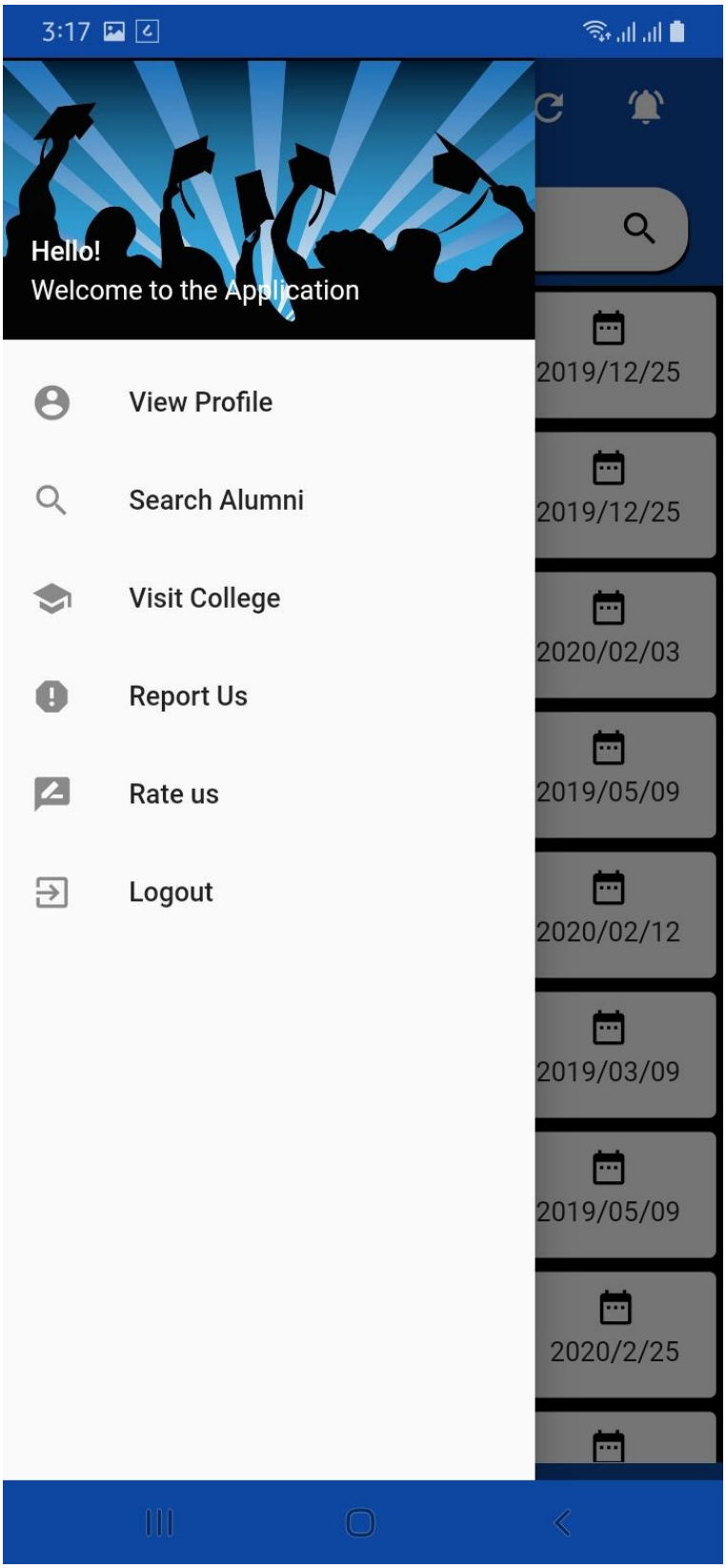


Figure 59 homepage navigation UI.

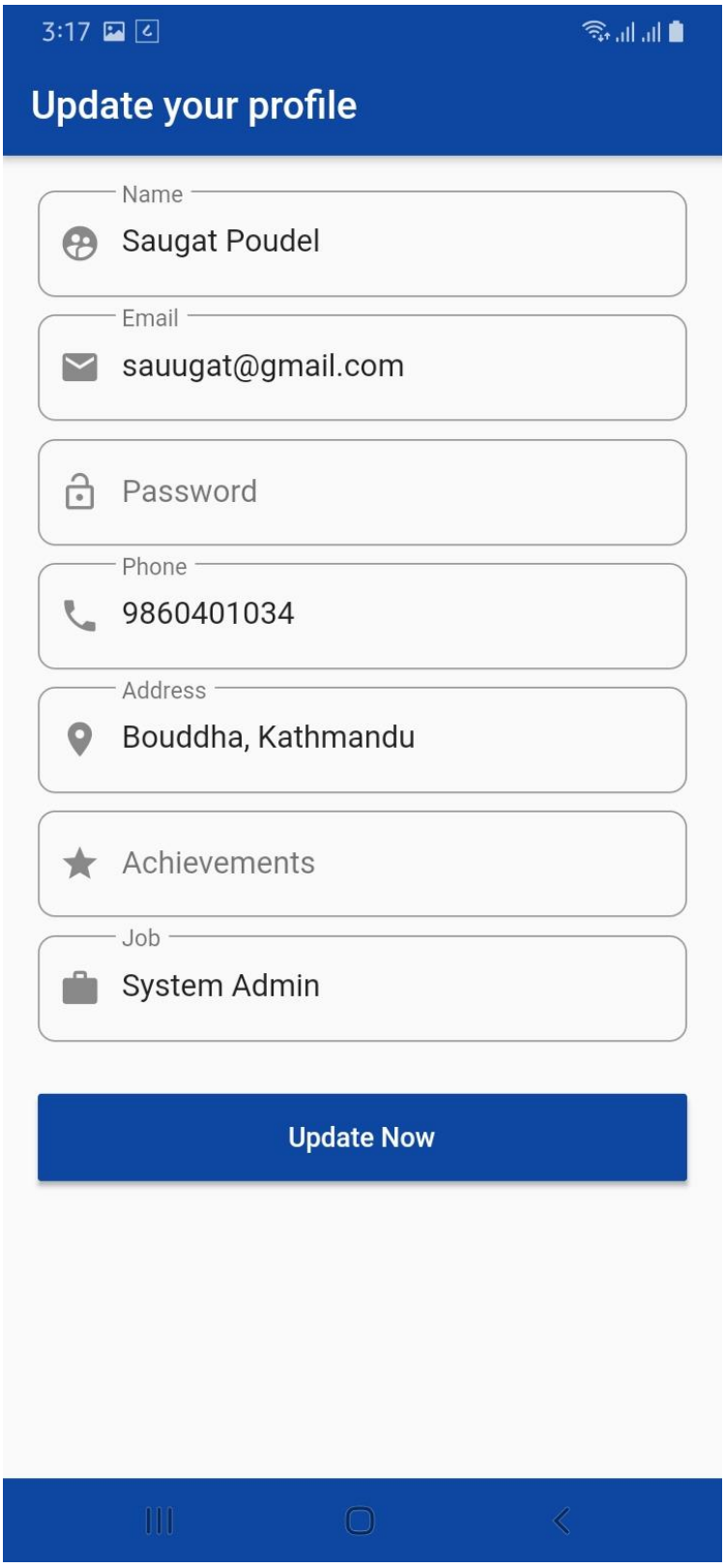


Figure 60 edit profile UI.

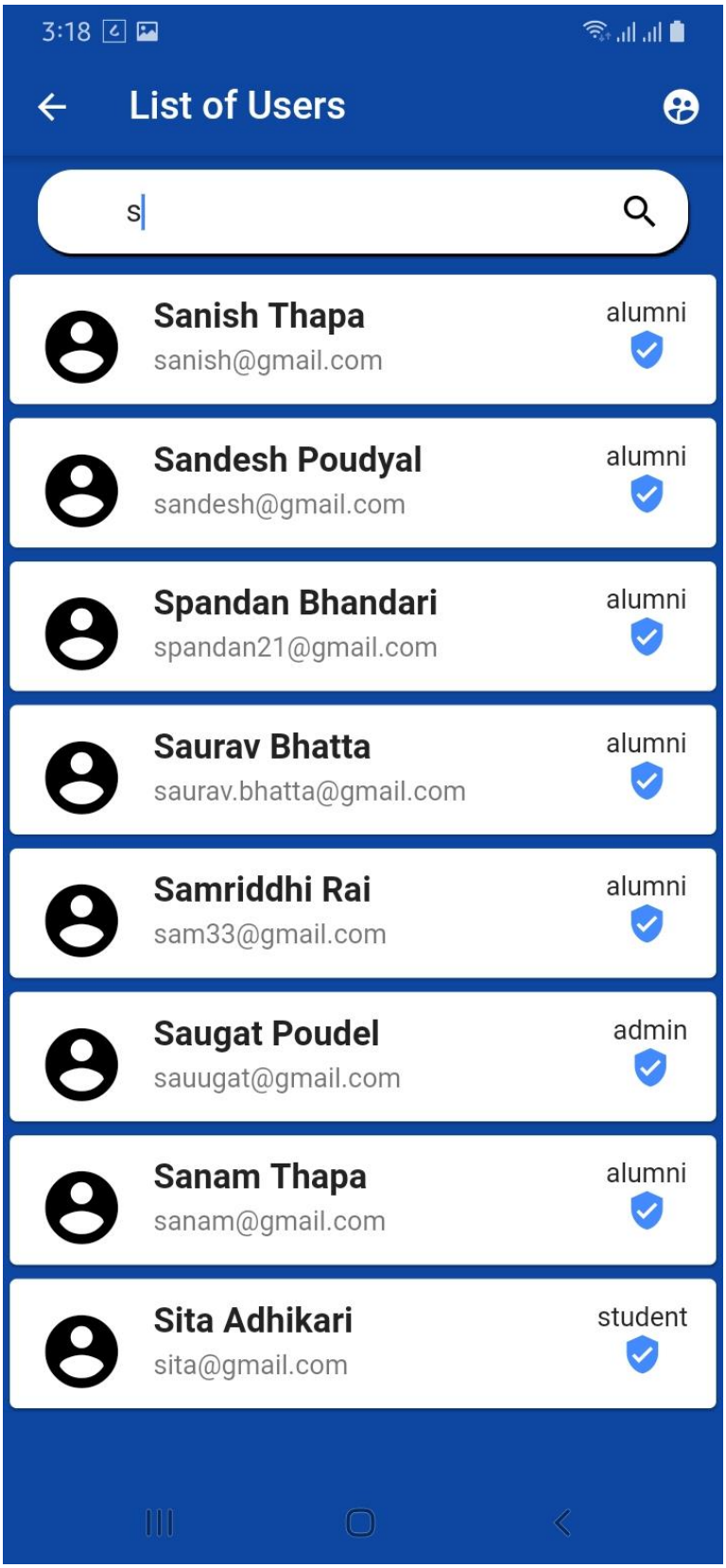


Figure 61 search user UI.

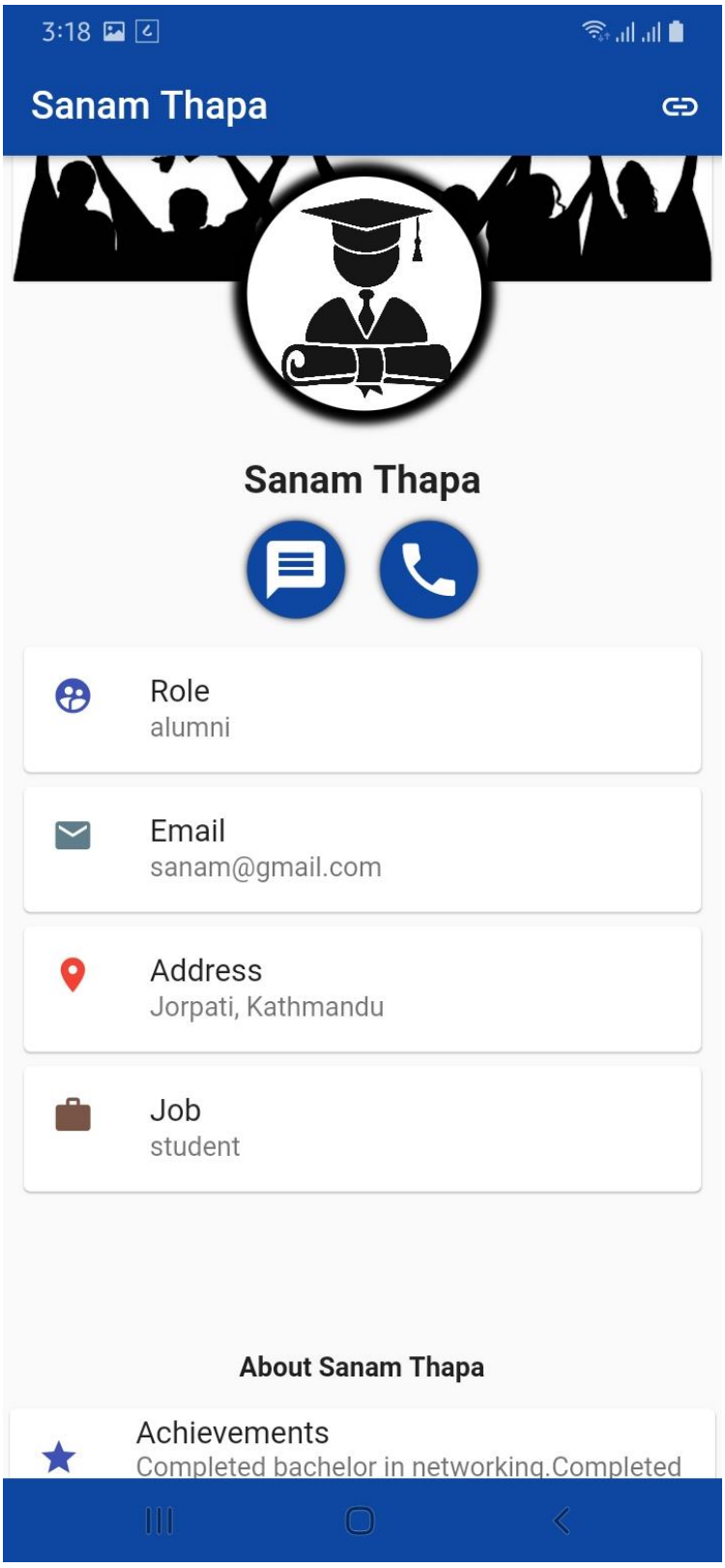


Figure 62 Alumni profile UI.

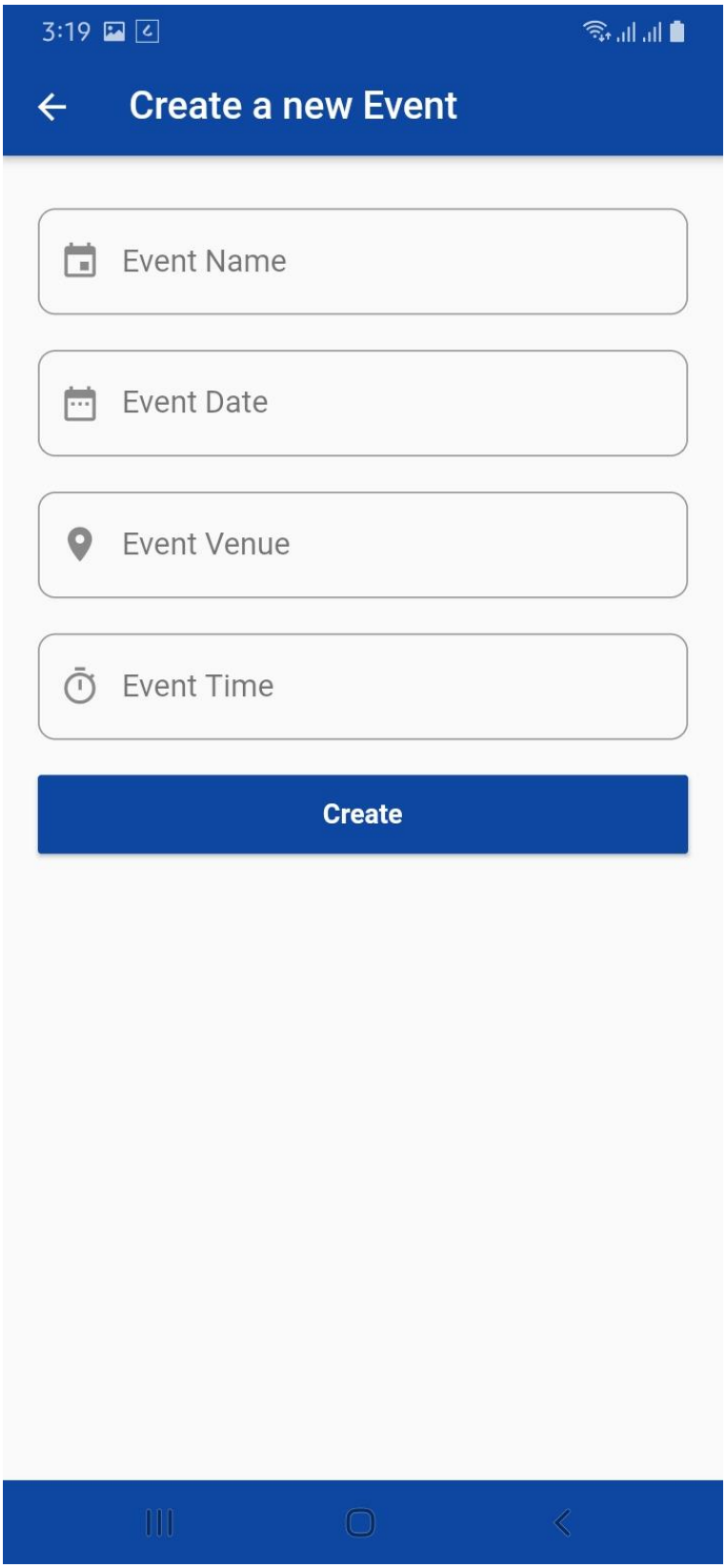


Figure 63 create event UI.

Admin Webpage:

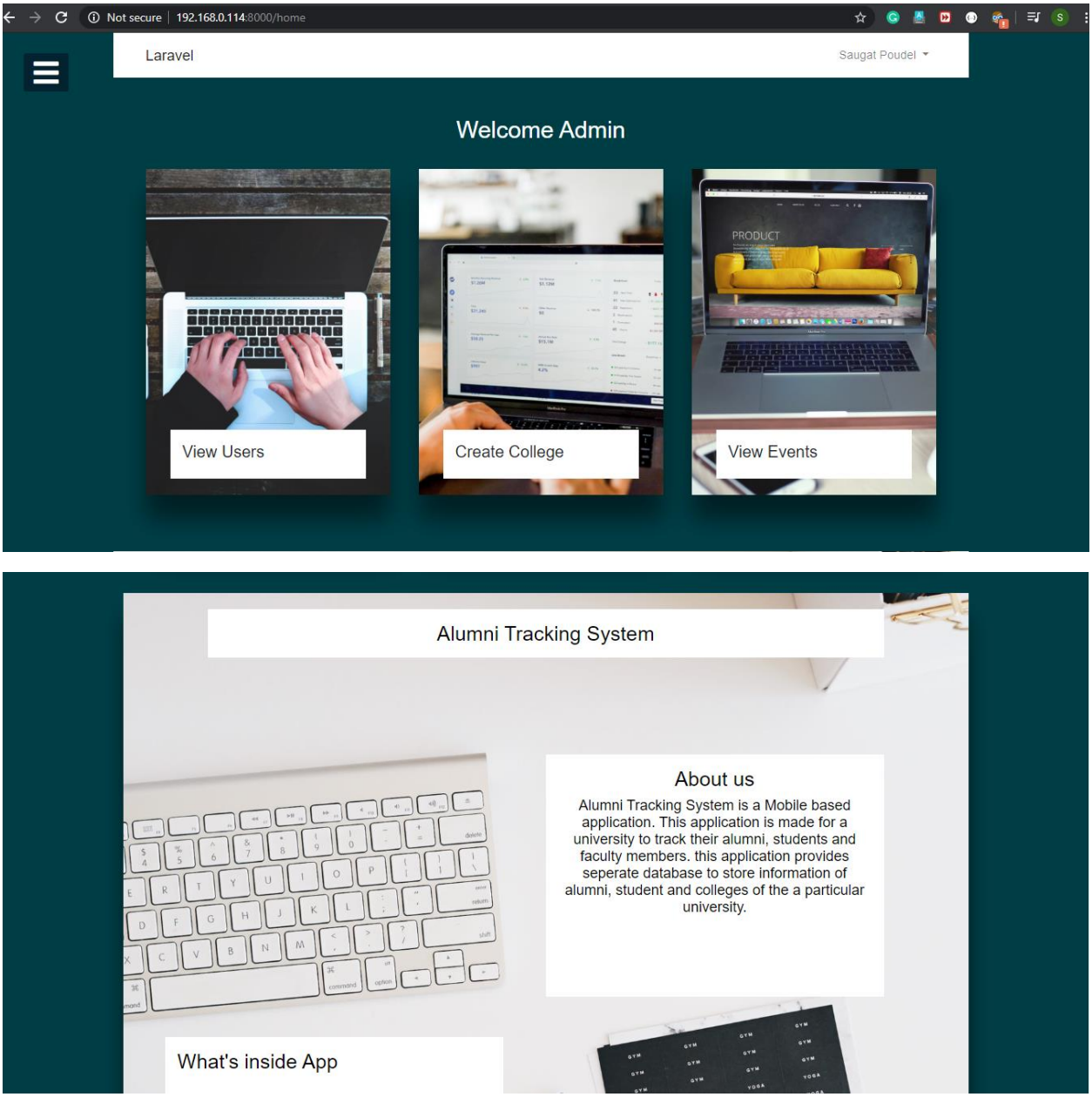


Figure 64 Admin web homepage.

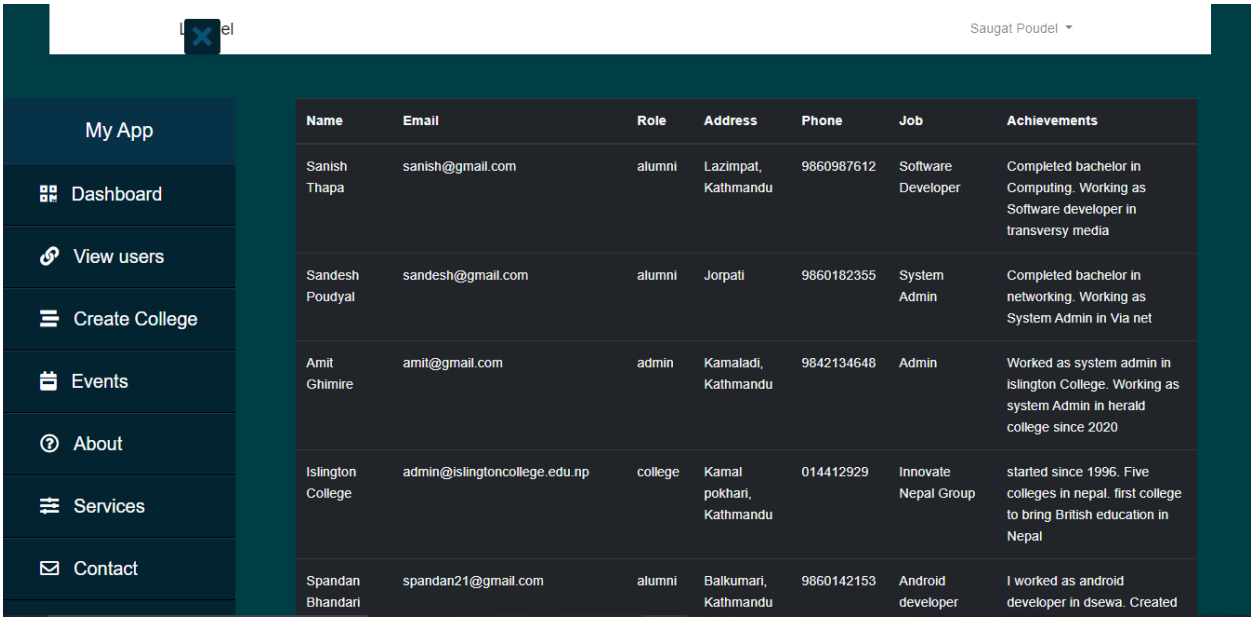


Figure 65 Admin web User list page.

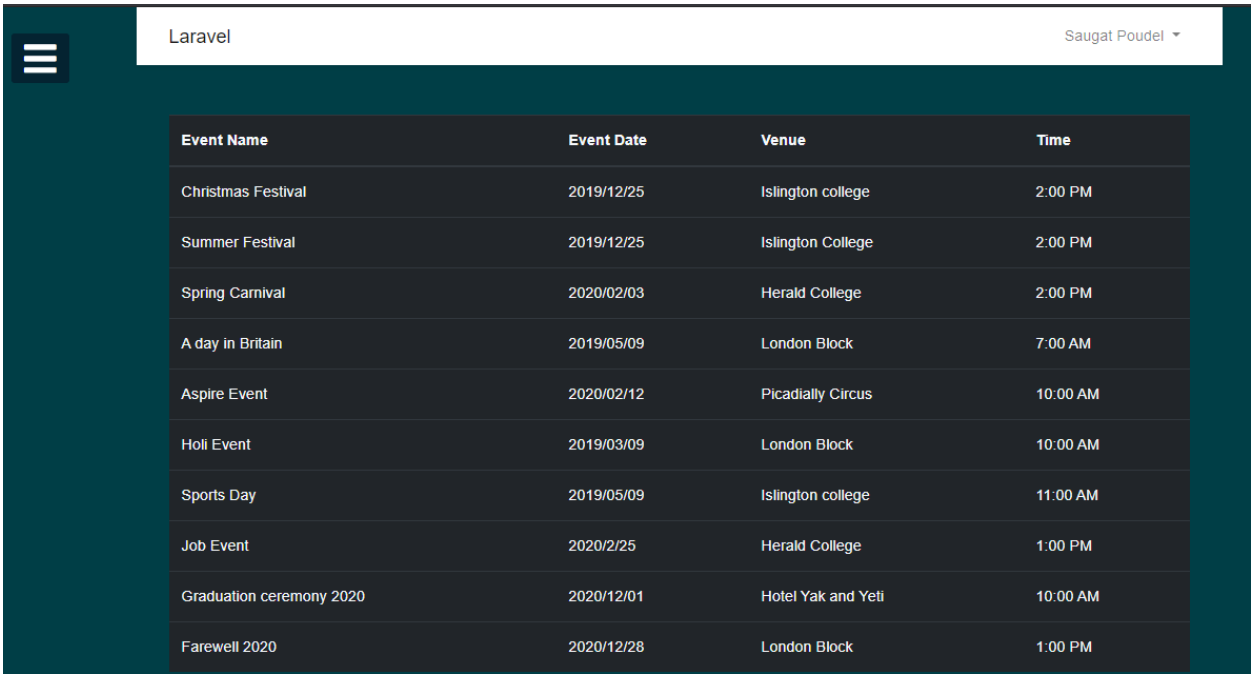


Figure 66 event page web App.

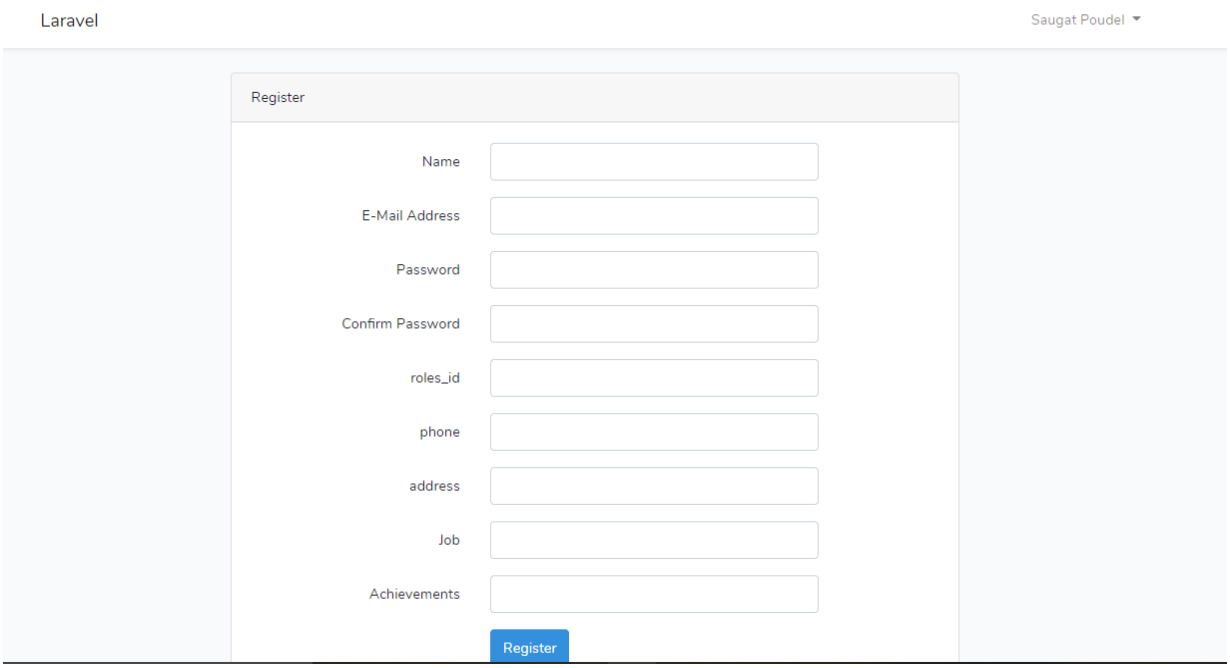


Figure 67 Register user Admin web.

Chapter 4: Testing and analysis:

4.1: Test Plan:

4.1.1: Unit Testing, test plan:

This section of the report is about the planning of unit test cases done in the testing section of the project. Various test cases were done to make the system bugs free and usable. Unit testing is the testing of every unit (function) of the system. I have used Laravel unit testing properties for checking URL of login register of the user. I used postman (an api testing tool) for showing the request and response of the api. I have also shown the failed cases in the unit testing.

The test plan for unit testing is elaborated in the table below.

Unit testing cases	Objective
Test case 1	To send api request to register a new user and add data to database.
Test case 2	To send api request to login a new user, create token and save in database and login user to system.
Test case 3	To send api request to create a new event and save in database.
Test case 4	To make an URL testing on http web login route.
Test case 5	To make an URL testing on http web register route failed case.
Test case 6	To make an URL testing on http web register route.
Test case 7	Fetching api data in mobile success.

Table 9 unit testing test plans.

4.1.2: System Testing test plans:

System testing generally means to test the overall system validation and system functionality through user interface. The user interface in this application is very simple and easily understandable. In this section of testing I have checked each functionality of the system like login authentication and validation, Registration validation and success cases, navigation routing test through one page to another, updating user data and validity etc. the detailed objectives and test cases of system test plan are shown in the table below.

System testing cases	Objective
Test case 1	To reset password of user from web.
Test case 2	To check empty validation of login page mobile.
Test case 3	To check authorized user for login to system.
Test case 4	To login user to system and show home page.
Test case 5	To register a new user checking form validation.
Test case 6	To register a new user after valid data and show in database.
Test case 7	To search events from home page.
Test case 8	To view profile of the logged in user.
Test case 9	To change user role from student to alumni.
Test case 10	To update profile of logged in user failed case.
Test case 11	To update profile of logged in user success.
Test case 12	To search a specific user from user list.
Test case 13	To view profile of specific user.
Test case 14	To view details of specific events.
Test case 15	To create a new event by college validation error.
Test case 16	To add new events by college and send real time notification to all users.

Test case 17	To logout user from the application.
Test case 18	To configure local host server for admin web page.
Test case 19	To check login authentication of admin web failed case.
Test case 20	To login admin to the system after authentication success case.
Test case 21	To view list of all users from admin panel.
Test case 22	To create new college from admin panel.
Test case 23	To display all events created by college in admin panel.

Table 10 system testing test plans.

4.2: Unit testing:

Unit testing registration of user api and saving to database.

Objective	To send api request to register a new user and add data to database.
Expected Result	Api should check the validation and save data to user database.
Actual Result	User enters valid data and send data through post method to the api. A new user is registered and added to database.
Conclusion	The test was Successful.

Table 11 unit testing of register user api.

```php
public function signup(Request $request)
{
    You, 3 months ago • commited after managing update feature
    $request->validate([
        'name' => 'required|string',
        'email' => 'required|string|email|unique:users',
        'password' => 'required|string|confirmed',
        'roles_id' => 'required|integer',
        'phone' => 'required|string',
        'address' => 'required|string',
        'Achievements' => 'required|string',
        'Job' => 'required|string',
    ]);
    $user = new User([
        'name' => $request->name,
        'email' => $request->email,
        'password' => bcrypt($request->password),
        'roles_id' => $request->roles_id,
        'phone' => $request->phone,
        'address' => $request->address,
        'Achievements' => $request->Achievements,
        'Job' => $request->Job,
    ]);
    $user->save();
    return response()->json(
        'Successfully created user!'
    , 201);
}
```

Figure 68 function testing of register user.

```php
Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});

Route::group([
    'prefix' => 'auth'
], function () {
    Route::post('login', 'AuthController@login');
    Route::post('signup', 'AuthController@signup');
```

Figure 69 route of user signup.

Sending data for register user api via postman.

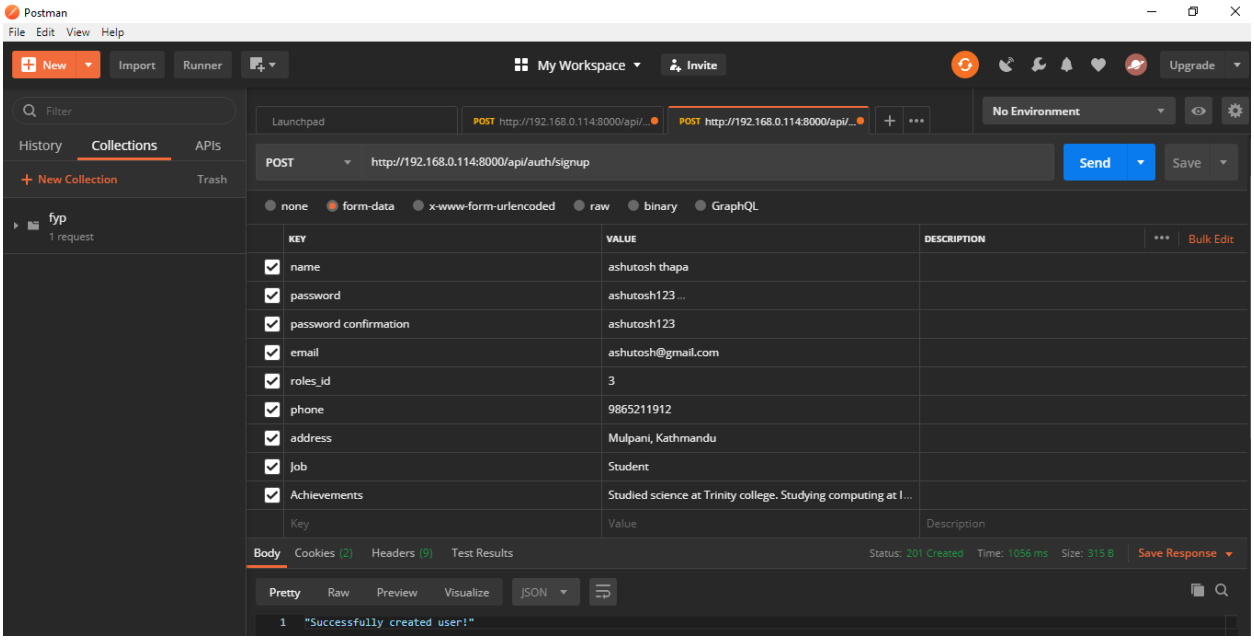


Figure 70signup user postman unit testing.

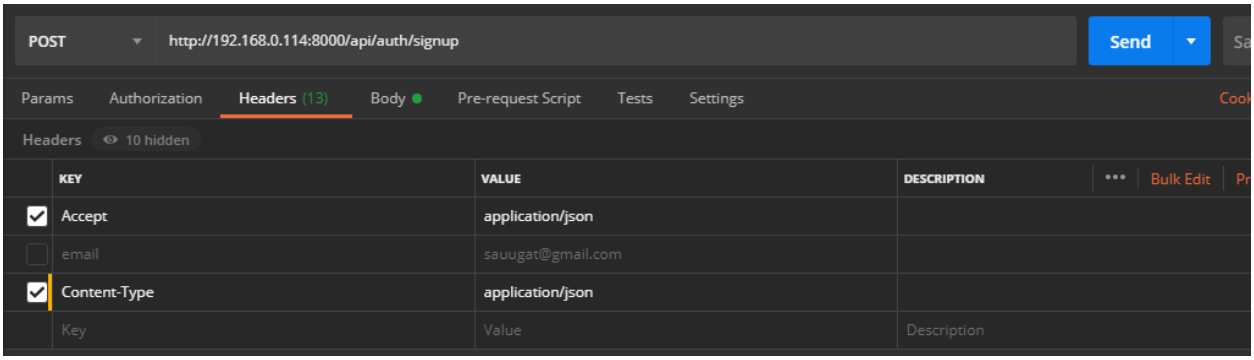


Figure 71 signup user postman headers data for signup.

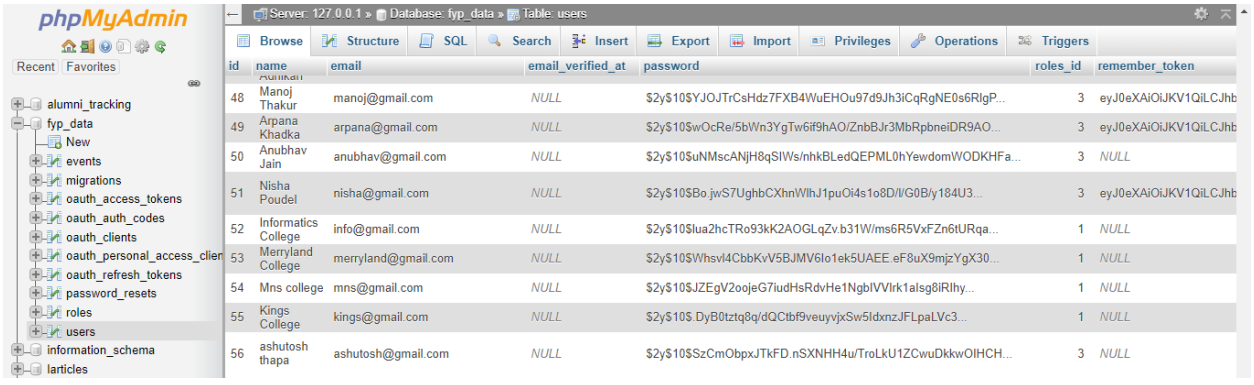


Figure 72 Saving new user to database via api unit testing.

Unit testing for login user via login api:

Objective	To send api request to login a new user, create token and save in database and login user to system.
Expected Result	Api should check the validation and create token and save in database. login should be success.
Actual Result	User is logged in and token is created and saved to database.
Conclusion	The test was Successful.

Table 12 unit testing of login user api.

```
 */
public function login(AuthRequest $request)
{
    $credentials = request(['email', 'password']);
    if(!Auth::attempt($credentials))
        return response()->json(
            'Invalid Username or Password'
        , 401);
    $user = $request->user();
    $tokenResult = $user->createToken('Personal Access Token');
    $token = $tokenResult->token;

    if ($request->remember_me)
        $token->expires_at = Carbon::now()->addWeeks(1);
        $user = User::where('email', '=', $request->email)->firstOrFail();
        $user->remember_token = $tokenResult->accessToken;
        $user->save();
    return response()->json([
        'access_token' => $tokenResult->accessToken,
        'token_type' => 'Bearer',
        'expires_at' => Carbon::parse(
            $tokenResult->token->expires_at
        )->toDateTimeString()
    ]);


}
```

Figure 73 login function for api creation.

```
Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});

Route::group([
    'prefix' => 'auth'
], function () {
    Route::post('login', 'AuthController@login');
```

Figure 74 login route for api.

Sending response form postman for login api.

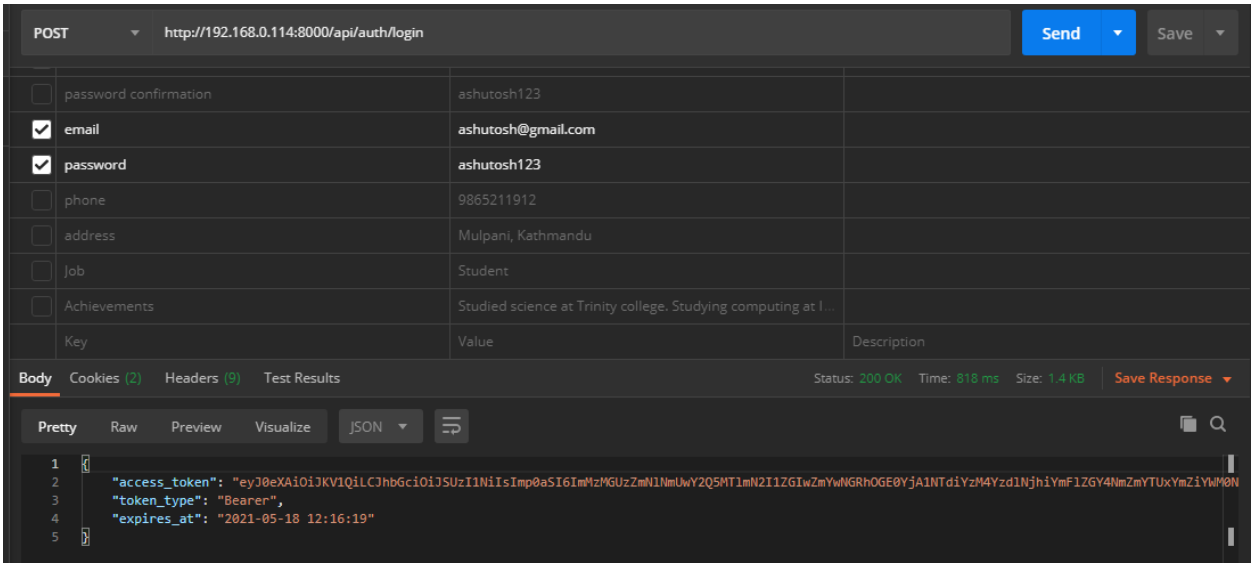
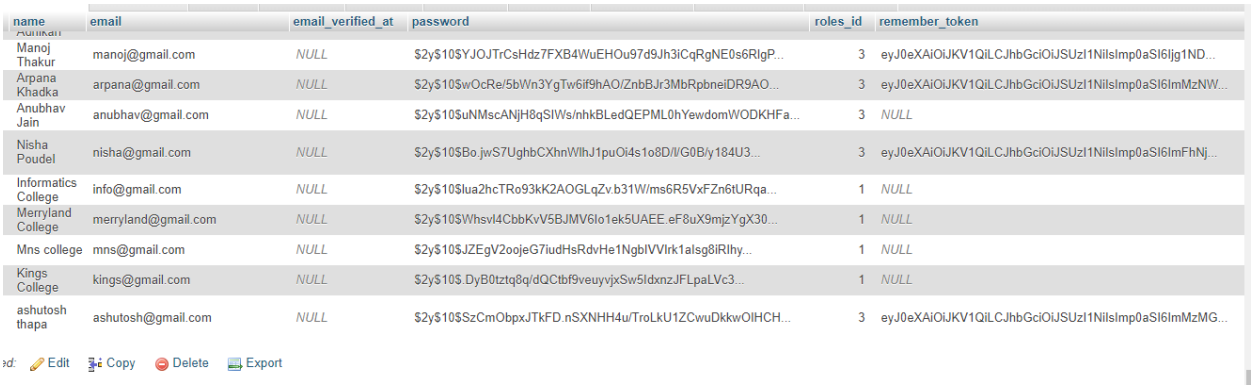


Figure 75 login api request.



name	email	email_verified_at	password	roles_id	remember_token
Manoj Thakur	manoj@gmail.com	NULL	$2y$10$YJOJTrCsHdz7FXB4WuEHOu97d9Jh3iCqRgNE0s6RlgP...	3	eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6Ijg1ND...
Arpana Khadka	arpana@gmail.com	NULL	$2y$10$wOcRe/5bWn3YgTw6if9hAO/ZnbBJr3MbRpbneiDR9AO...	3	eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6ImMzNW...
Anubhav Jain	anubhav@gmail.com	NULL	$2y$10$uNMscANjH8qSIWs/nhkBLedQEPML0hYewdomWODKHFa...	3	NULL
Nisha Poudel	nisha@gmail.com	NULL	$2y$10$Bo.jwS7UghbCXhnWlhJ1puOi4s1o8D/l/G0B/y184U3...	3	eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6ImFhNj...
Informatics College	info@gmail.com	NULL	$2y$10$lua2hcTRo93kK2AOGLqZv.b31W/ms6R5VxFZn6tURqa...	1	NULL
Merryland College	merryland@gmail.com	NULL	$2y$10$WhsvI4CbbKvV5BJMV6Io1ek5UAEE.eF8uX9mjzYgX30...	1	NULL
Mns college	mns@gmail.com	NULL	$2y$10$JZEgV2oojeG7iudHsRdvHe1NgbIVVIrk1alsg8iRIhy...	1	NULL
Kings College	kings@gmail.com	NULL	$2y$10$.DyB0tztq8q/dQCtbf9veuyvjxSw5ldxnzJFLpaLVc3...	1	NULL
ashutosh thapa	ashutosh@gmail.com	NULL	$2y$10$SzCmObpxJTkFD.nSXNHH4u/TroLkU1ZCwuDkkwOlHCH...	3	eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6ImMzMG...

Edit Copy Delete Export

Figure 76 token creation and save after login success.

Unit testing for creating events:

Objective	To send api request to create a new event and save in database.
Expected Result	Api should check the validation and create new events and save in database.
Actual Result	New event is created and saved in database.
Conclusion	The test was Successful.

Table 13 unit testing of events api..

```
    */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $event = $request->isMethod('put') ? Event::findOrFail
        ($request->event_id): new Event;

        $event->id = $request->input('event_id');
        $event->event_name = $request->input('event_name');
        $event->event_time = $request->input('event_time');
        $event->event_date = $request->input('event_date');
        $event->event_venue = $request->input('event_venue');

        if($event->save()){
            return new EventResource($event);

        }
    }
```

Figure 77 function for creating events.

```
47  Route::get('events', 'EventController@index');
48
49  Route::get('allevents', 'EventController@allevents');
50
51  Route::get('search', 'EventController@search');
52
53  Route::get('event/{id}', 'EventController@show');
54
55  Route::post('event', 'EventController@store');
56
57  Route::put('event', 'EventController@store');
58
59  Route::delete('event/{id}', 'EventController@destroy');
```

Figure 78 route for creating events.

Create events api testing from postman.

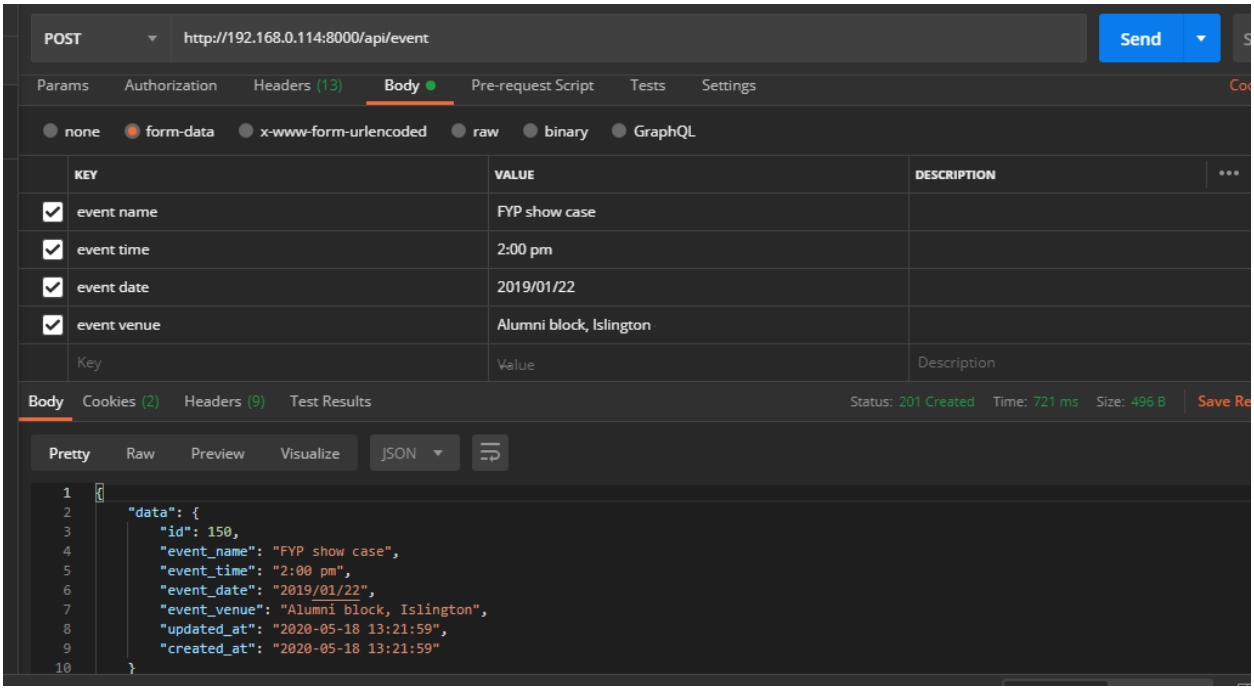


Figure 79 Api request for create events.

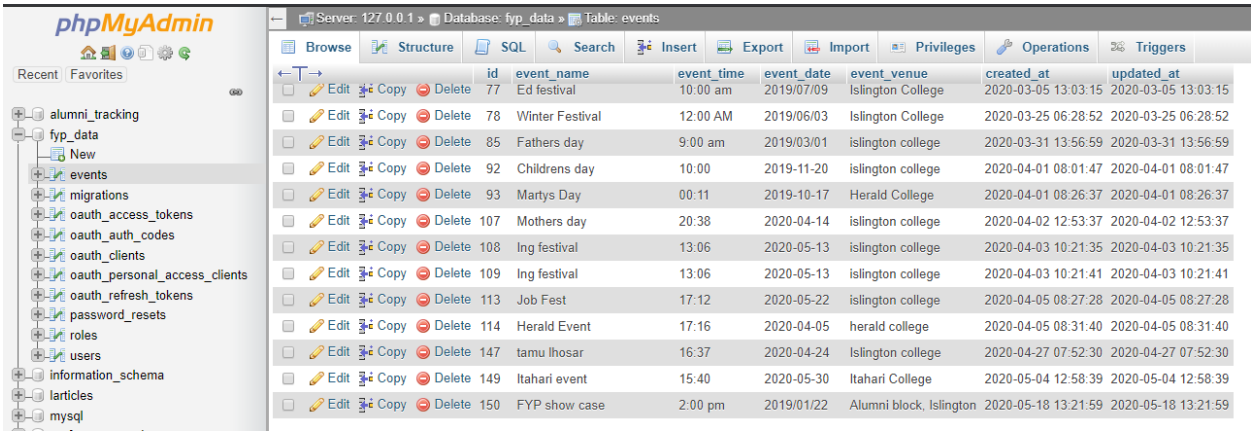


Figure 80 event database for new data.

Unit testing Laravel http URL for web login:

Objective	To make an URL testing on http web login route.
Expected Result	Should check the entered URL from the testing function and show the test results.
Actual Result	Here the "/login" URL was a valid URL for web login, so the test was successful.
Conclusion	The test was Successful.

Table 14 unit testing of Laravel web login URL.

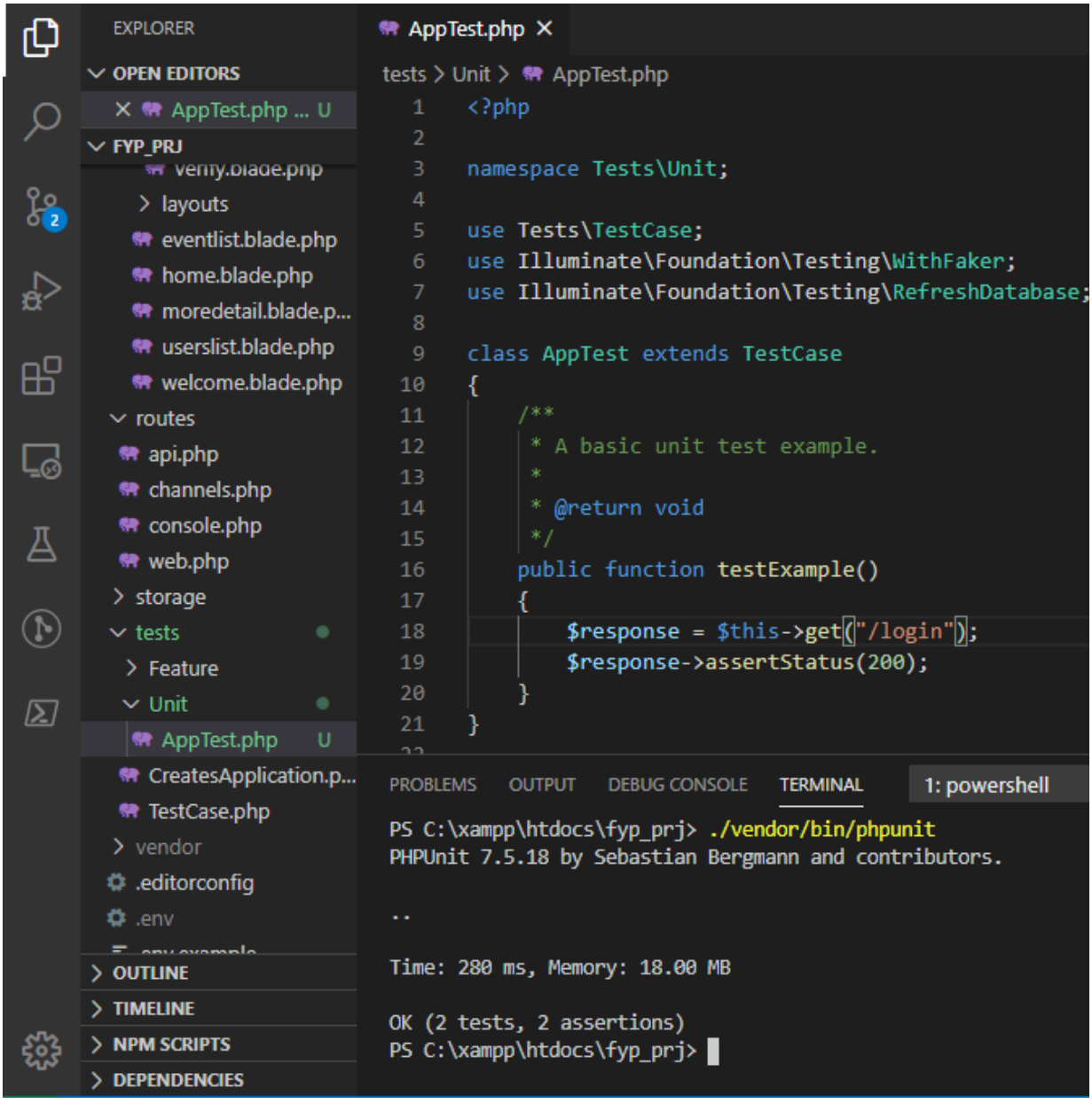


Figure 81 unit testing of login web URL.

Unit testing Laravel http URL for web register failed:

Objective	To make an URL testing on http web register route.
Expected Result	Should check the entered URL from the testing function and show the test results.
Actual Result	Here the "/register" URL was a valid URL for web register, but the expected status we entered was 404 that is page not found. So, the test was failed.
Conclusion	The test was not successful.

Table 15 unit testing of failed test cases Laravel register URL.

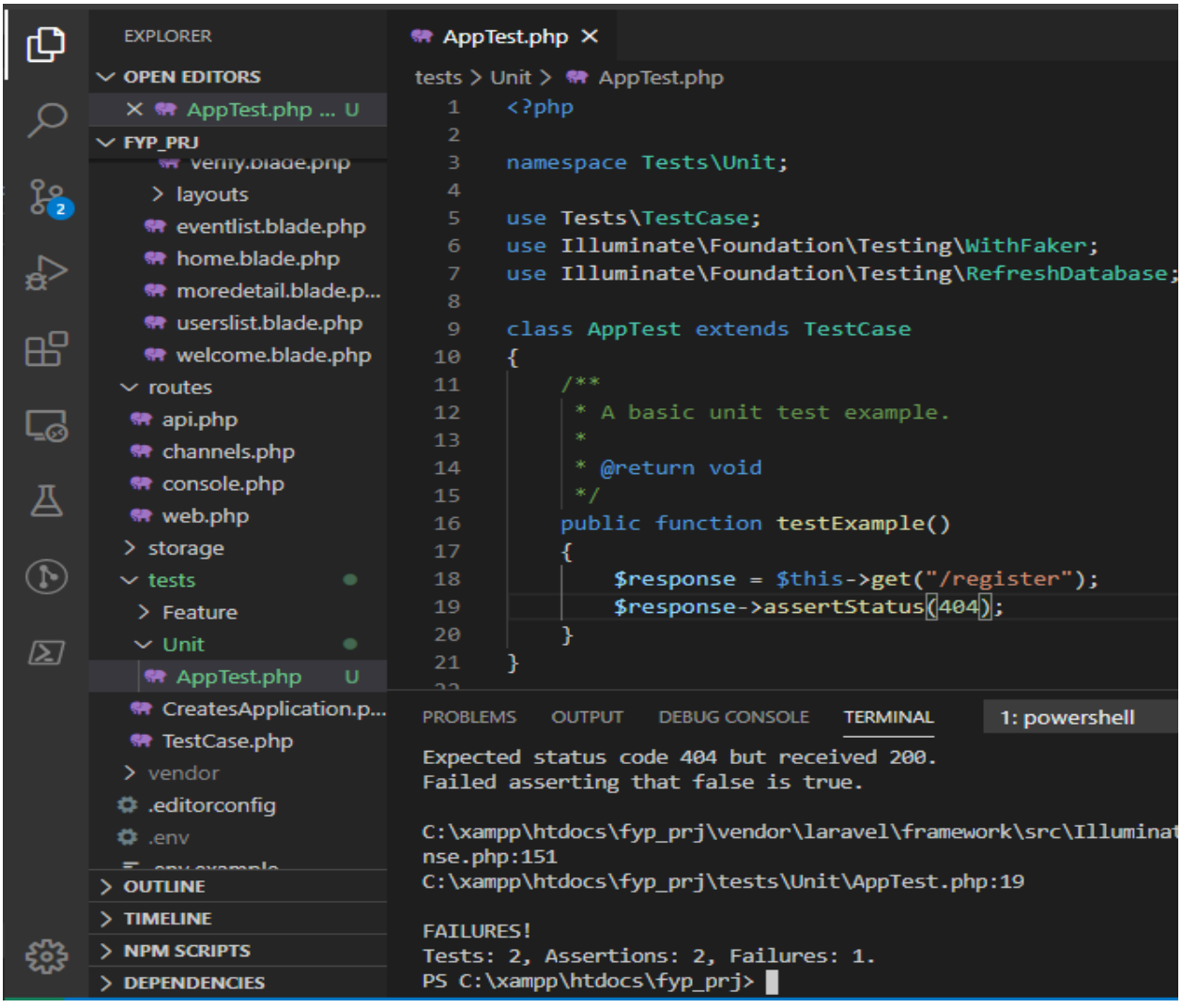


Figure 82 unit testing of register URL failed case.

Fetching event api to mobile home page:

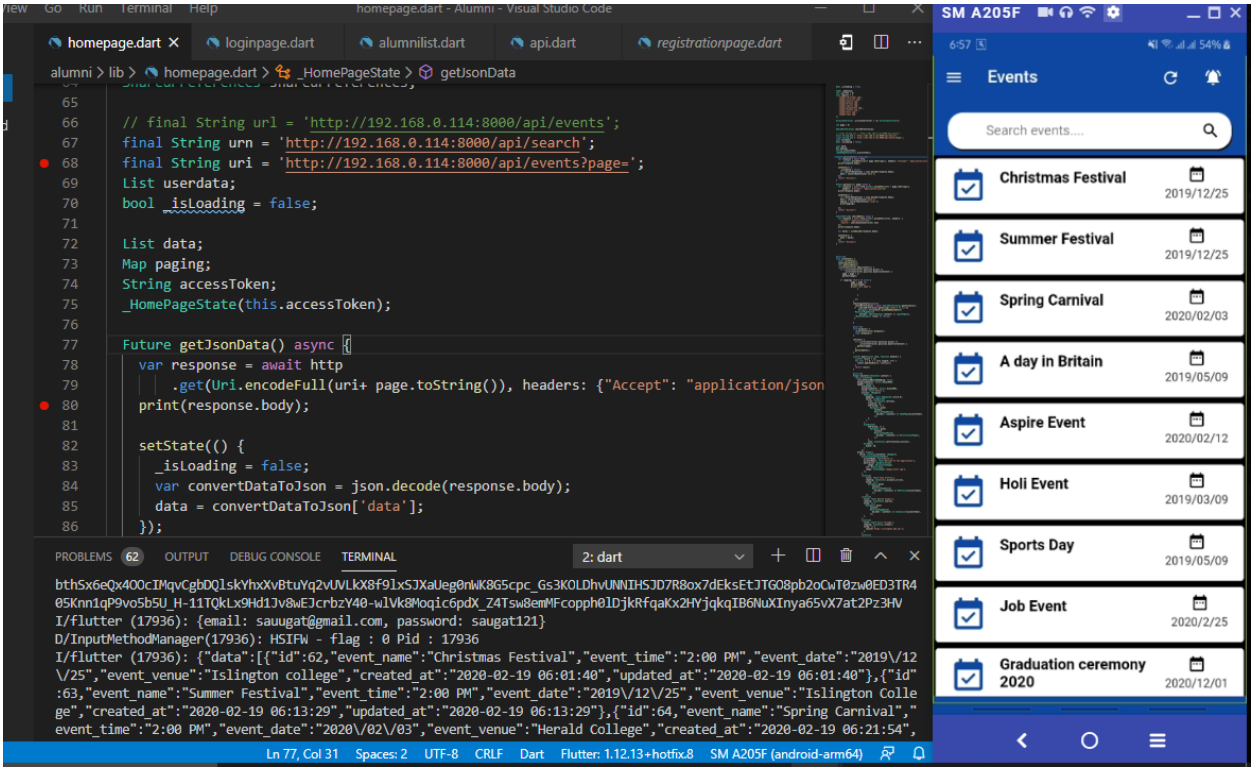


Figure 83 unit testing of fetching data from api.

4.3: System testing:

Login Validation testing:

Objective	To login into the system.
Action	User clicks login button without inserting login credentials.
Expected Result	Should show error message.
Actual Result	Shows email and password cannot be empty.
Conclusion	The test was Successful.

Table 16 testing for login validation empty data.

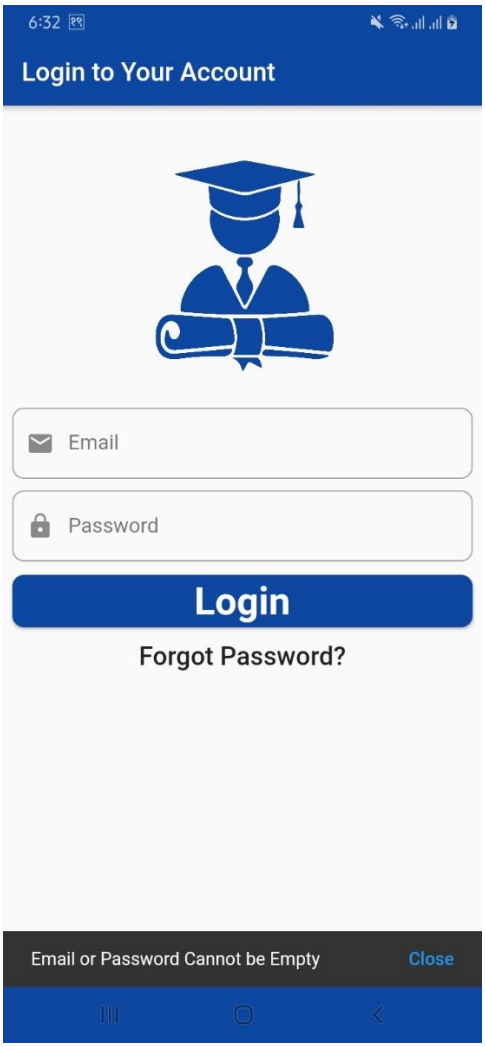


Figure 84 testing for empty data in login.

Login validation unauthorized user:

Objective	To login user into the application.
Action	User inserts incorrect login credentials and press login button.
Expected Result	Should show error message.
Actual Result	The username or password is invalid.
Conclusion	The test was successful.

Table 17 testing for unauthorized user validation.

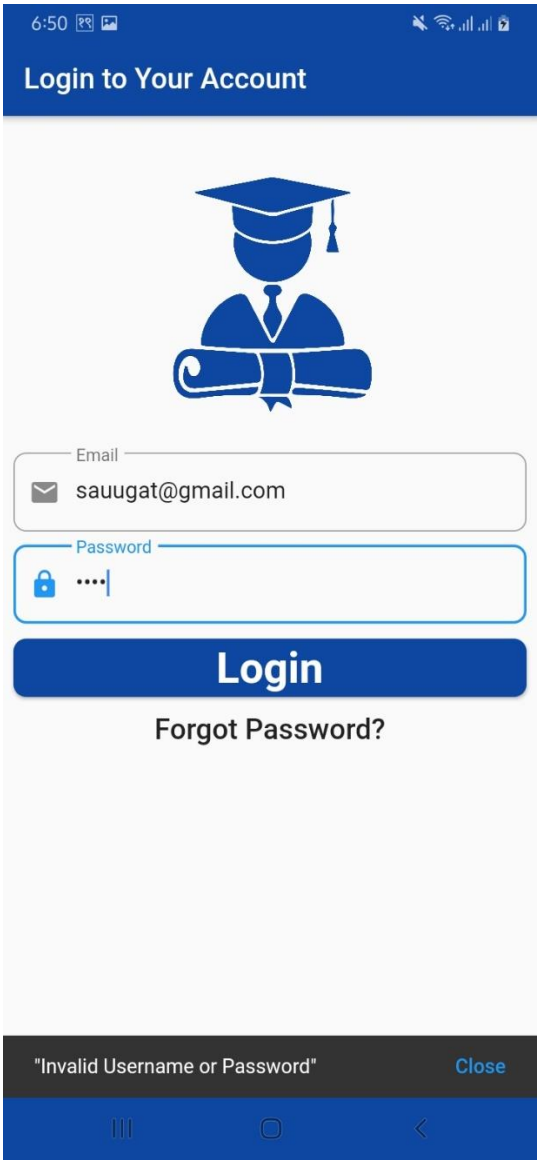


Figure 85 testing for invalid username and password.

Login validation successful:

Objective	To login users to the application.
Action	User enters correct login credential and press login button.
Expected Result	Should redirect user to the homepage of application.
Actual Result	User is redirected to the home page of application.
Conclusion	The test was successful.

Table 18 testing for login success.

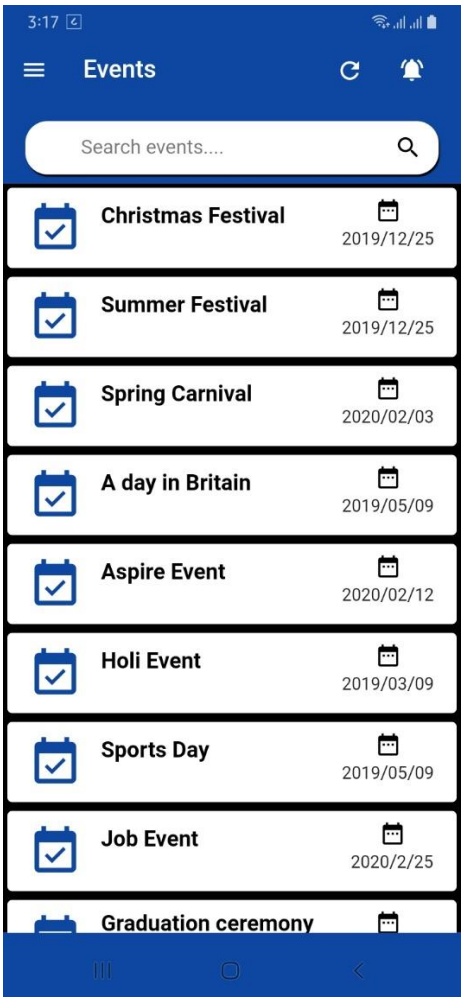


Figure 86 testing for login success.

Register User empty data Validation:

Objective	To register a new user.
Action	User doesn't enter any data and clicks submit button.
Expected Result	Should validate every fields and show error message.
Actual Result	Shows error message "all fields are required".
Conclusion	The test was successful.

Table 19 testing for register user form validation.

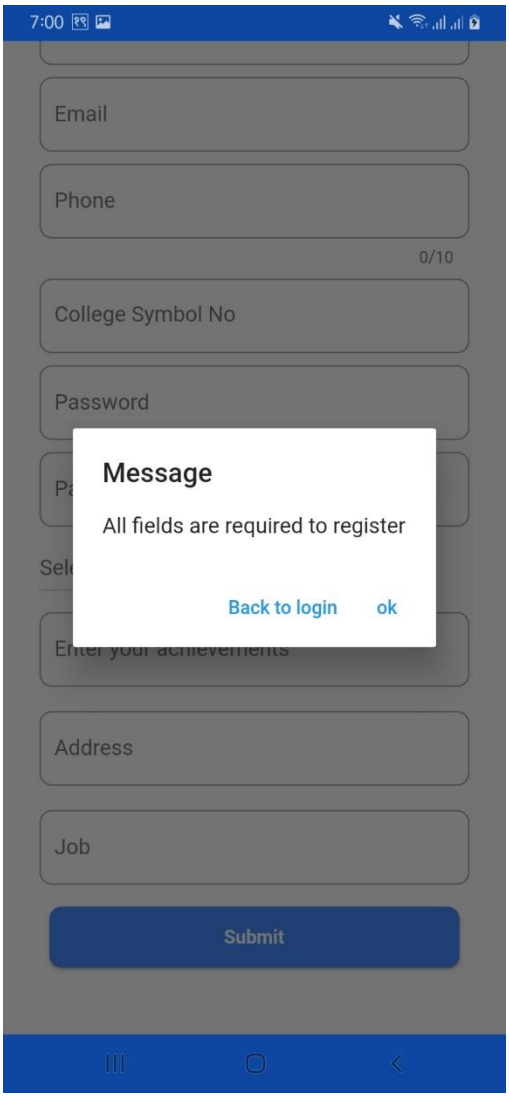


Figure 87 testing for register user form validation.

Register user validation Success:

Objective	Should create a new user.
Action	User enters all required data and press submit button.
Expected Result	Should register a new user to the system.
Actual Result	Registration successful message is shown.
Conclusion	The test is Successful.

Table 20 testing for register user success.

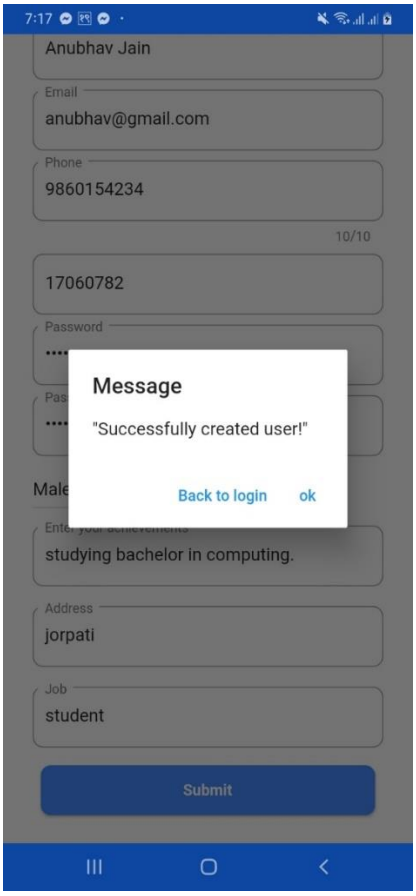


Figure 88 testing for register user success.

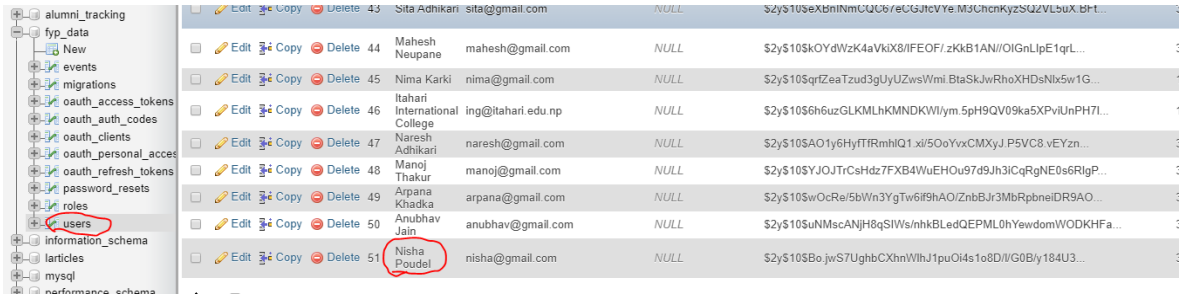


Figure 89 testing for new user saved in database.

Testing for Search events:

Objective	Searching events from event name.
Action	User enters “S” word in the search bar of event page.
Expected Result	Should shows all the events whose first letter is S.
Actual Result	List of events starts from S is shown in List view.
Conclusion	The test was successful.

Table 21 testing for searching events from mobile.

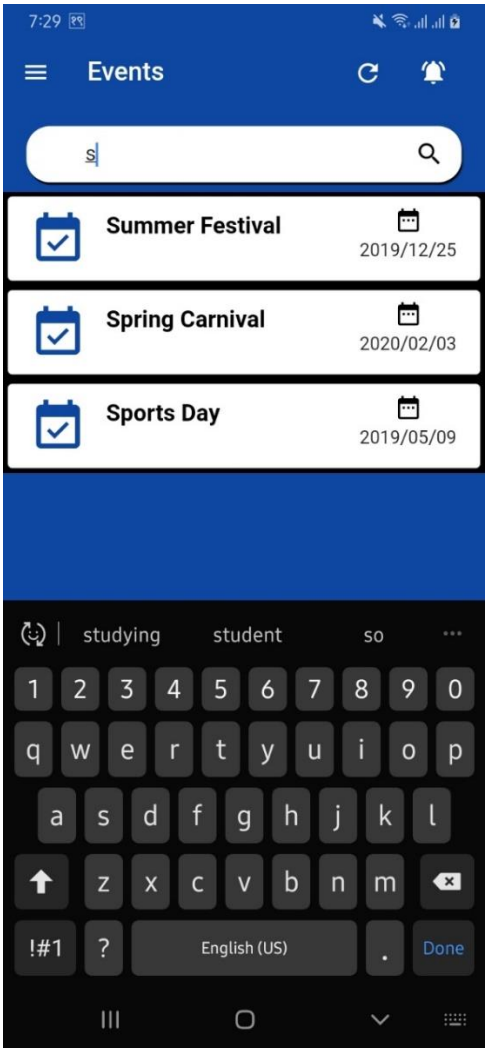


Figure 90 testing for search events.

Testing for Searching users:

Objective	To search the users from the application.
Action	User selects search user button from home page and enters “a” in the search bar.
Expected Result	Should shows list of users whose first name starts with a with their general information.
Actual Result	System shows list of all users and their general information whose name starts with a.
Conclusion	The test was successful.

Table 22 testing for search users.

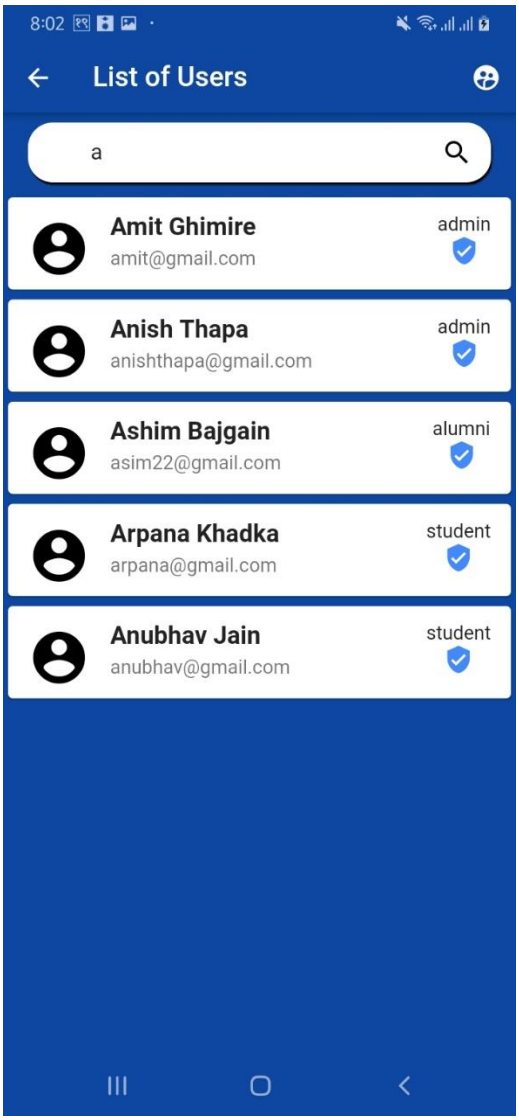


Figure 91 testing for search users.

Testing for creating events validation success:

Objective	To add the new events by the college and to give real time notification to all devices and users.
Action	College add events data and press create button.
Expected Result	System should create a new event and show in event page, also should give real time notification to all devices.
Actual Result	New event name “itahari event” is created and notification is received by all devices with alumni app.
Conclusion	The test was successful.

Table 23 testing for add new events success.

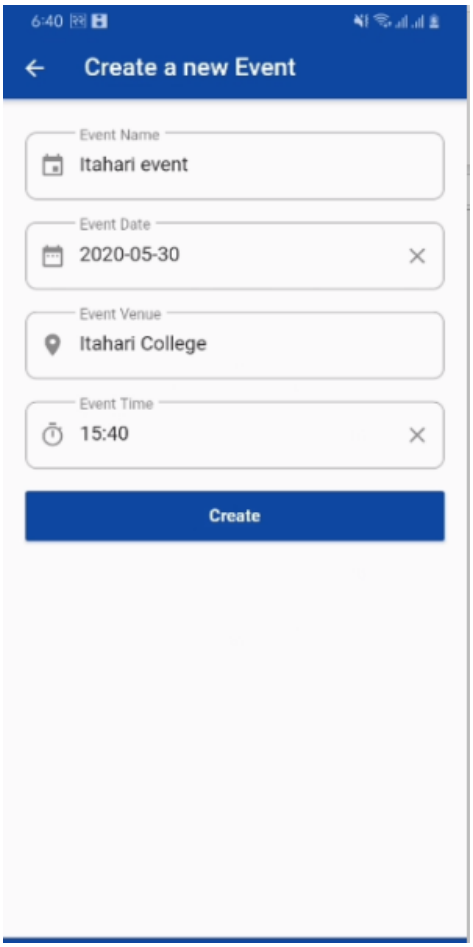
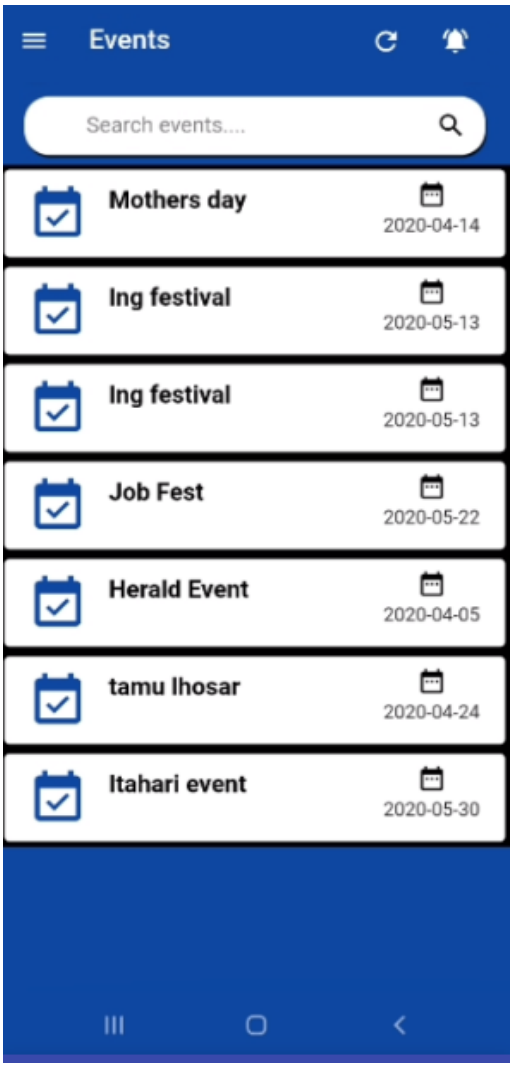


Figure 92 testing for add new events.

Edit Copy Delete	108	Ing festival	13:06	2020-05-13	islington college	2020-04-03 10:21:35	2020-04-03 10:21:35
Edit Copy Delete	109	Ing festival	13:06	2020-05-13	islington college	2020-04-03 10:21:41	2020-04-03 10:21:41
Edit Copy Delete	113	Job Fest	17:12	2020-05-22	islington college	2020-04-05 08:27:28	2020-04-05 08:27:28
Edit Copy Delete	114	Herald Event	17:16	2020-04-05	herald college	2020-04-05 08:31:40	2020-04-05 08:31:40
Edit Copy Delete	147	tamu lhosar	16:37	2020-04-24	Islington college	2020-04-27 07:52:30	2020-04-27 07:52:30
Edit Copy Delete	149	Itahari event	15:40	2020-05-30	Itahari College	2020-05-04 12:58:39	2020-05-04 12:58:39

Figure 93 testing for add new events to database.



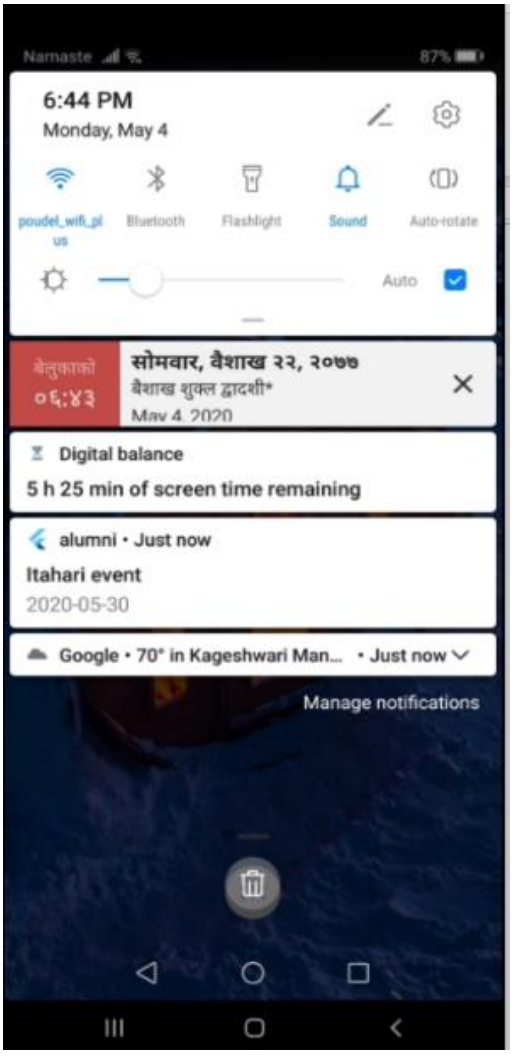


Figure 94 testing for send and show event notification to users.

Testing for Admin web panel (Login Validation)

Objective	To check user's validation for admin website.
Action	User enters wrong user credential and clicks on login button.
Expected Result	System should validate the user from database and show error message.
Actual Result	Error message was shown as "The credential does not match our records."
Conclusion	The test was successful.

Table 24 testing for admin web login validation error.

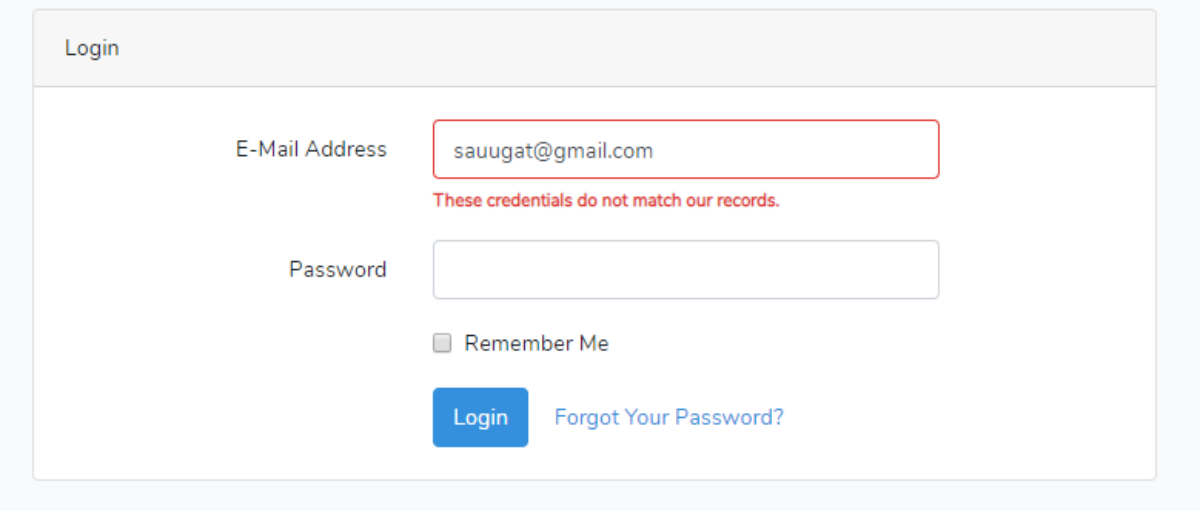


Figure 95 testing for admin web login validation error.

Testing for Admin web Panel (Login):

Objective	To login admin to the web application and show details of application in website.
Action	Admin enters the login details and press login button.
Expected Result	Application should validate the user and redirects admin to the home page of website.
Actual Result	User was verified and redirected to the home page of web application.
Conclusion	The test was successful.

Table 25 testing for admin web panel login success.

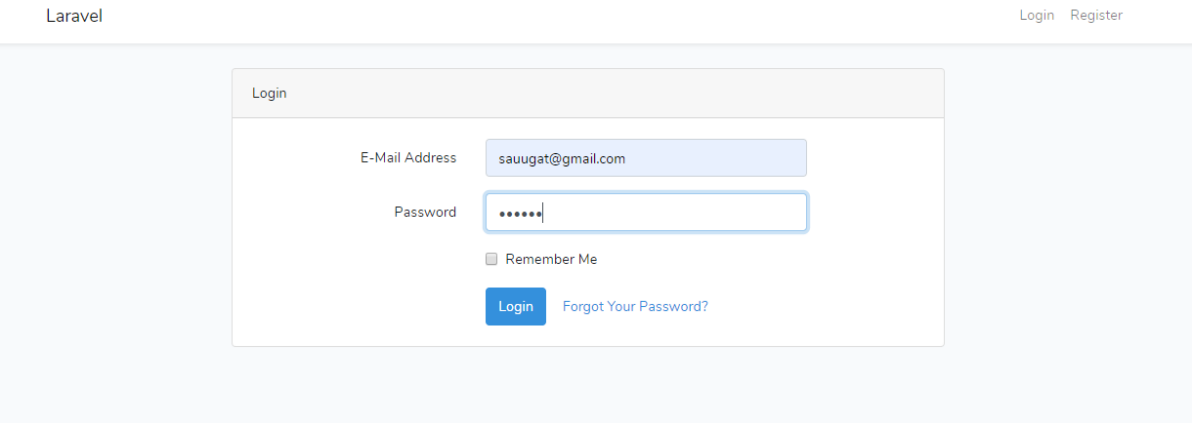


Figure 96 testing for admin web panel login success.

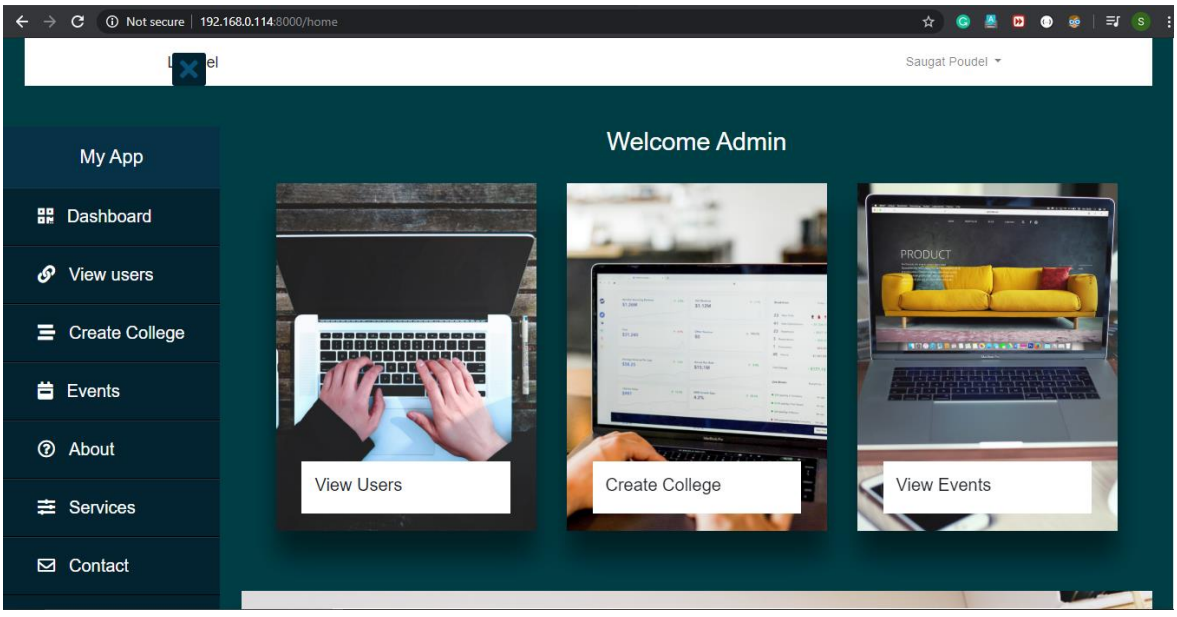


Figure 97 testing for admin home page.

Testing for view users Admin web panel:

Objective	To show all users and their details in admin web panel.
Action	Admin clicks on view users' button from home navigation bar.
Expected Result	All list of users who are registered to the alumni application with their details should be shown.
Actual Result	All users are shown in a list with their details.
Conclusion	The test was successful.

Table 26 testing for view users admin panel.

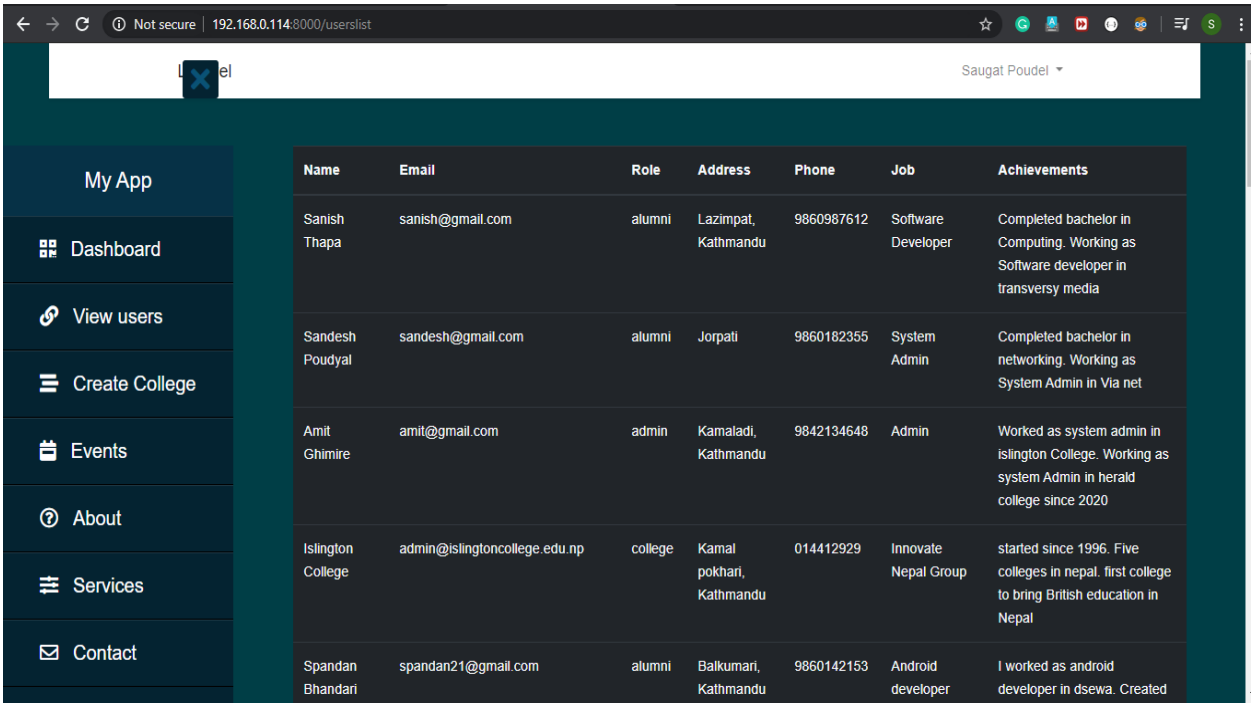


Figure 98 testing for admin web panel view users.

Testing for creating new college from admin web panel:

Objective	To register a new college by admin to the system.
Action	User enters all details for new college and press on register button.
Expected Result	A new college should be created to the system.
Actual Result	A new college is created to the system.
Conclusion	The test was successful.

Table 27 testing for register new college or admin.

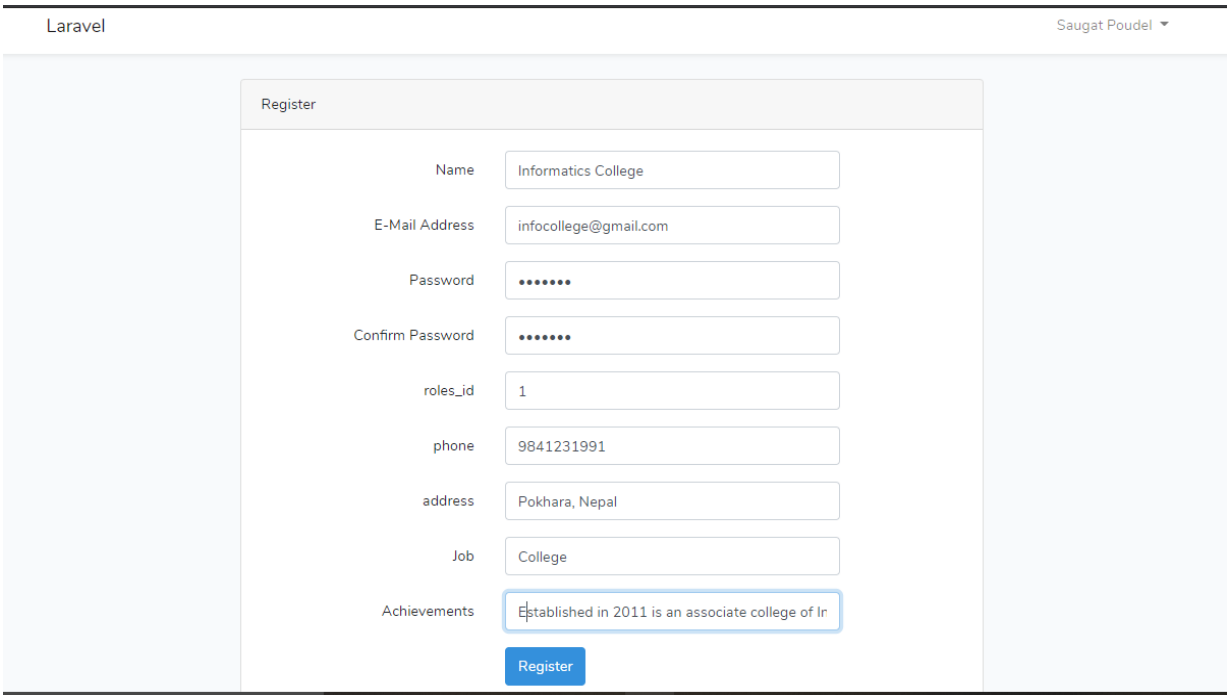


Figure 99 testing for enter college add for registration.

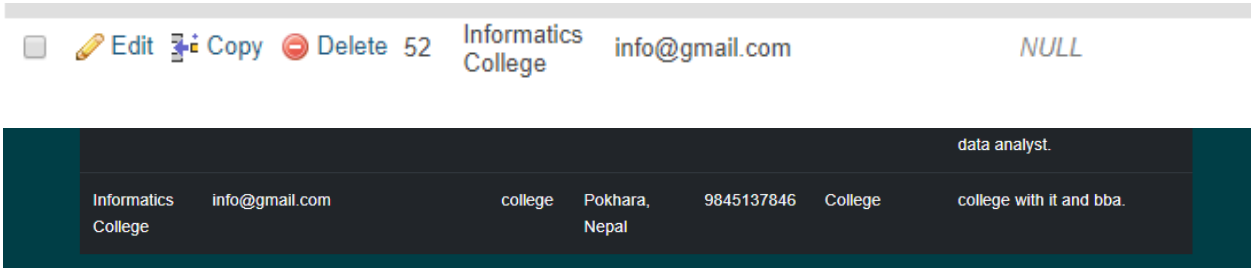


Figure 100 testing for showing new college data in user list.

Testing for Admin web panel (Show Events):

Objective	To show all events created by college to admin.
Action	Admin clicks on Events from navigation bar.
Expected Result	All events should be shown to the admin.
Actual Result	All events with their details is shown to the admin webpage.
Conclusion	The test was successful.

Table 28 testing for show events admin web panel.

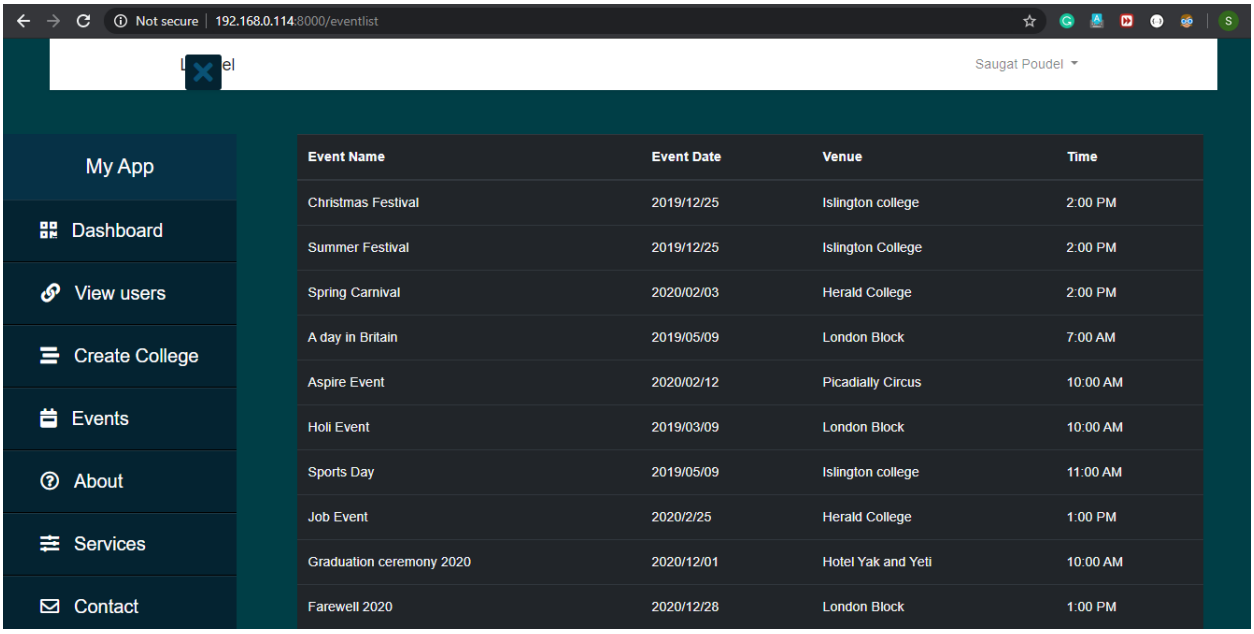


Figure 101 testing for show events admin panel.

(Note: More test cases are shown in appendix section J of the report.)

4.4: Critical Analysis:

During testing various test cases was performed like unit testing and system testing. In unit testing the api was tested with postman, also using unit test properties in Laravel the api route was tested and shown. Unit testing was a bit hard to perform as it requires actual case and actual output using coding logic. During unit testing in Laravel api route of admin panel was tested. Also, the failed cases were shown in the unit testing section.

Similarly, while discussing about system testing it was easily to perform test cases because it was based on testing through User Interface. During this testing form validation of each pages like login, register, creating events, updating profile was checked. Also, there were some failed cases in this testing which was also shown in the test case. Each functionality of the system including admin website was tested during this phase.

All the testing of the system was completely successful. The system was working without any bugs and all the features was working properly. With the final sample of Alumni Tracking System users were given chance to interact with the application and their feedback was collected.

The system will be updated time to time and will also get new features according to the user's suggestion. Again, the test cases will be performed after adding new features to make sure the new features are not affecting the old one.

Chapter 5: Conclusion:

5.1: Legal, Social and Ethical Issues:

The demand for mobile application is increasing widely in the world day by day. As the "Statista portal" states, global mobile app revenue is forecasted to reach 188.9 billion U.S. dollars by 2020 via app stores and in-app ads (Clement, 2019).

Due to such high demand developer must face many types of issues inorder to launch their product in the market. This application can also face many kinds of issues when it will be launched in the market. Some of the issues like legal issues, social issues and ethical issues are discussed below:

5.1.1: Legal Issues:

This application will also face some of the legal issues and one of them is intellectual property about copyrights because this application is not registered, and patient and idea of this application can be stolen. Due to such mistake any one can copy the idea and violet intellectual property of this application. Trademark or copyrights of this application is also not made so it is easy for anyone to thief whole application and own it.

The application has collected many user's information. Collecting user's information in a software is not crime because we first must ask the users that their information is being collected. But keeping their information private is the duty of software owners. Here in this project the user's information may not be secure because one user can see others private information which may cause legal issues about privacy. (openGeeksLab, 2018)

5.1.2: Social Issues:

The problems held in a society which influences any individuals living in that society is

called social issues or problems. Social issues do not concern about court decision but mostly

concerns with individual well-being, moral values, cultures and rights.

This application can also face social issues because of the user's data. Users can collect other user information and can bully them or black mail them by misusing their private information.

Also, inside this application there is features like events collection and invitation. Events in this application subjects to any religion or culture. For example, in case of Christmas event when

people from Hindu's and Christianity is being called then there can be caste discrimination inside the events due to activities held in the program which may cause social issues due to this feature of application.

5.1.3: Ethical issues:

Taking about ethical issues this project also has some issues in it. here while an alumni's post their achievements, they may post some personal information of the company they are working on which may cause privacy issues and violets code of conducts of a company. (ASHA, 2020)

5.2: Advantages:

There are many advantages of this application some of them are mentioned below:

- As there were no any separate database for alumni in the university now with this application university will get database of alumni also.
- Inviting the alumni to any kind of events held by college was a big challenge now with this application college can create events and notify all the alumni about it.
- There was no any link between college, alumni and students to see each other achievements now with this application sharing their own achievements is possible.
- Students can now get information of Alumni and see their achievements easily through this application.
- College can always be in contact with their alumni when they need each other.
- Student can be in touch with alumni and get solutions by emailing them through this application.
- Every user's will be able to update their profile and add their fresh data.
- Students can get event notification in real time with this application.
- Admin will have separate web-based application, so he doesn't have to open mobile while working.
- Admin can register new college to the application and add more students of same community.
- All data are normalized and free of redundancy so there will not be any problem in future for adding, updating and deleting information to the database.

5.3: Limitations:

The limitations of this project are as follows:

- Users should provide their personal information to the application which may cause privacy issues.
- As there is no proper verification of users during registration some users may violet the rules of the application.
- Admin cannot delete any users from the application which may occur difficulty to manage data.
- Users may not update their information in time which may leads to misunderstanding while getting information.
- Users should know the name of user whom he/she is searching for inorder to view their profile.
- Teachers are not engaged in this application so if any teacher want to communicate with alumni, he/she should either go through college or through student for the alumni information.

5.4: Future work:

In this frequently upgrading era of technology it is very important to add new features and update the application time to time. The future works for this application depends upon the feedback of the users of this application. An application can have many features which depends upon the user's feedback. Every user has their own perspective of interacting in the application, some of them just opens the applications once in a week, when some of them opens daily.

The main objectives of future work for this application will be according to the user's desire and feedback. Last time a post survey was taken about this application where it is found that some of the users are suggesting adding news feed where users can watch every user daily progress without visiting their profile. The next work will be adding a news feed section in the application.

Also, some of the users are suggesting making a chatbot in this application so that they can chat with users without using mail. Here recently in this application uploading photos of users is not available. In the future work those features will be added. these features were not added because some people don't want to share their pictures for their own privacy but now, as many of the users are suggesting for it those features will be added in future.

Chapter 6: References

References

Burke, W.W. & Noumair, D.A. (2015) *Organization Development: A Process of Learning and Changing*. 3rd ed. USA: FT Press, 2015.

Charvat, J. (2003) *Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects*. illustrated ed. Canada: John Wiley & Sons, 2003.

Chi, C. (2018) *Everything You Need to Know About Using the Waterfall Methodology* [Online]. Available from: https://blog.hubspot.com/marketing/waterfall-methodology [Accessed 03 June 2020].

Clement, J. (2019) *Statista* [Online]. Available from: https://www.statista.com/statistics/269025/worldwide-mobile-app-revenue-forecast/ [Accessed 24 April 2020].

Firebase. (2020) *Firebase* [Online]. Available from: https://firebase.google.com/ [Accessed 07 January 2020].

Flutter.dev. (2020) *Fetch data from the internet* [Online]. Available from: https://flutter.dev/docs/cookbook/networking/fetch-data [Accessed 02 March 2020].

flutter. (2020) *flutter.dev* [Online]. Available from: https://flutter.dev/ [Accessed 07 January 2020].

Gour, B., Srivastava, M. & Richhariya, V. (2018) *Database management system concept and normalization.* 1st ed. New Delhi: EDUCREATION.

Guru99. (2020) *MVC Tutorial for Beginners: What is, Architecture & Example* [Online]. Available from: https://www.guru99.com/mvc-tutorial.html [Accessed 03 June 2020].

Guru99. (2020) *What is Spiral Model? When to Use? Advantages & Disadvantages* [Online]. Available from: https://www.guru99.com/what-is-spiral-model-when-to-use-advantages-disadvantages.html.

Hegde, S. (2019) *meduim.com* [Online]. Available from: https://medium.com/communityzapp/how-to-reconnect-solving-the-key-challenges-of-alumni-engagement-4ef31ff38b33 [Accessed 02 June 2020].

Hugley, A.W. (2019) New direction in student services. *The role of the alumni associations in student life*, (2020/06/02), p.19.

IBM. (2003) Rational Unified Process. *Rational Unified Process: A Best Practicles Approach*, 1(2020/1/7), p.82.

IBM. (1998) *Rational Unified Process* [Online]. ibm Available at: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf [Accessed 07 January 2020].

infostride.com. (2019) *What is RUP?* [Online]. Available from: https://blog.infostride.com/blog/what-is-rup [Accessed 9 May 2020].

Kilicdagi, A. & Yilmaz, i.H. (2014) UK: Packt Publishing.

Knoernschild, K. (2002) *Java Design objects, uml and process*. 1st ed. Boston: Pearson Education Corporate Sales Division.

Kumar, P. (2009) *Rational Unified Process(Rup)* [Online]. Available from: https://www.slideshare.net/pawanonline83/rational-unified-processrup [Accessed 07 January 2020].

Kumar, A. (2012) *Sencha MVC Architecture*. 1st ed. UK: PCKT.

laravel. (2011) *Introduction to laravel* [Online]. Available from: https://laravel.com/docs/4.2/introduction.

Morse, A.P. (2017) *Rational Unified Process: What Is It And How Do You Use It?* [Online]. Available from: https://airbrake.io/blog/sdlc/rational-unified-process [Accessed 14 March 2017].

openGeeksLab. (2018) *Top 7 Legal Issues to Consider in Mobile App Development* [Online]. Available from: https://medium.com/@openGeeksLab/top-7-legal-issues-to-consider-in-mobile-app-development-c76192f281f3 [Accessed 24 April 2020].

oracle. (2020) MYSQL services. *MYSQL database service*, p.1.

ourcodeworld. (2019) *Mobile App database SQLite* [Online]. Available from: https://ourcodeworld.com/articles/read/737/everything-you-need-to-know-about-sqlite-mobile-database [Accessed 09 May 2020].

Pakhhira, M.K. (2013) *Database Management System*. 1st ed. Delhi: PHI learning.

pub.dev. (2020) *Firebase Cloud Messaging for Flutter* [Online]. Available from: https://pub.dev/packages/firebase_messaging [Accessed 03 January 2020].

Rajgor, U. (2017) *Create REST API in Laravel with authentication using Passport* [Online]. Available from: https://medium.com/techcompose/create-rest-api-in-laravel-with-authentication-using-passport-133a1678a876 [Accessed 02 March 2020].

salesforceforeducation. (2000) *Salesforce for Education Reviews & Product Details* [Online]. Available from: https://www.g2.com/products/salesforce-for-education/reviews [Accessed 06 January 2020].

Shavani, H. (2019) *Build RESTful API In Laravel 5.8 Example* [Online]. Available from: https://www.itsolutionstuff.com/post/build-restful-api-in-laravel-58-exampleexample.html [Accessed 09 Augustus 2019].

Sinha, S. (2019) *Beginning Laravel: Build Websites with Laravel 5.8*. 2nd ed. India: Apress.

smartdraw. (2017) *Sequence Diagram* [Online]. Available from: https://www.smartdraw.com/sequence-diagram/ [Accessed 08 April 2020].

SQL Lite. (2020) *About SQLite* [Online]. Available from: https://www.sqlite.org/about.html [Accessed 09 May 2020].

Tansey, J. (2008) *Tracking Alumni*. research. Washinton: The advisory board company Student Affairs Leadership Council.

Windle, D.R. & Abreo, L.R. (2003) *Software requirements using the unified process.* 1st ed. USA: Prentice Hall PTR.

Windle, D.R. & L, A.R. (2003) *Software Requirements Using the Unified Process.* 1st ed. USA: Prentice hall ptr.

www.getalma.com. (2012) *Alma | Student Information* [Online]. Available from: https://sourceforge.net/software/product/Alma/ [Accessed 07 January 2020].

Chapter 7: Bibliography:

Sinha, S. (2019) *Beginning Laravel: Build Websites with Laravel 5.8*. 2nd ed. India: Apress.

www.getalma.com. (2012) *Alma | Student Information* [Online]. Available from: https://sourceforge.net/software/product/Alma/ [Accessed 07 January 2020].

Tansey, J. (2008) *Tracking Alumni*. research. Washinton: The advisory board company Student Affairs Leadership Council.

Flutter.dev. (2020) *Fetch data from the internet* [Online]. Available from: https://flutter.dev/docs/cookbook/networking/fetch-data [Accessed 02 March 2020].

flutter. (2020) *flutter.dev* [Online]. Available from: https://flutter.dev/ [Accessed 07 January 2020].

Shavani, H. (2019) *Build RESTful API In Laravel 5.8 Example* [Online]. Available from: https://www.itsolutionstuff.com/post/build-restful-api-in-laravel-58-exampleexample.html [Accessed 09 Augustus 2019].

pub.dev. (2020) *Firebase Cloud Messaging for Flutter* [Online]. Available from: https://pub.dev/packages/firebase_messaging [Accessed 03 January 2020].

tutorials point. (2018) *ER Diagram Representation* [Online]. Available from: https://www.tutorialspoint.com/dbms/er_diagram_representation.htm [Accessed 13 March 2020].

USlegal. (2019) *Legal Issue Law and Legal Definition* [Online]. Available from: https://definitions.uslegal.com/l/legal-issue/ [Accessed 24 April 2020].

Velasquez, M. (2010) *What is Ethics?* [Online]. Available from: https://www.scu.edu/ethics/ethics-resources/ethical-decision-making/what-is-ethics/ [Accessed 24 April 2020].

Visual paradigm. (2020) *What is UML Collaboration Diagram?* [Online]. Available from: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml-collaboration-diagram/ [Accessed 21 April 2020].

Lucid Chart. (2017) *UML Use Case Diagram Tutorial* [Online]. Available from: https://www.lucidchart.com/pages/uml-use-case-diagram [Accessed 09 October 2019].

guru99. (2019) *UML Activity Diagram: What is, Components, Symbol, EXAMPLE* [Online]. Available from: https://www.guru99.com/uml-activity-diagram.html [Accessed 04 May 2020].

Rational Software White Paper. (1998) Rational Unified Process. *Best Practices for Software Development Teams* , (2020/04/22), p.21.

ASHA. (2020) *Issues in Ethics: Ethical Use of Social Media* [Online]. Available from: https://www.asha.org/Practice/ethics/Ethical-Use-of-Social-Media/ [Accessed 24 April 2020].

Bagui, S. & Earp, R. (1964) *Database Design Using Entity-Relationship Diagrams*. illustrated ed. London: Auerbach publications.

Bittner, K. & Spence, I. (2003) *Use Case Modeling*. Illustrated ed. New york: Addison-wesley.

Chapter 8: Appendix

8.1. Appendix A: Pre-Survey:

8.1.1 Pre-Survey Form:

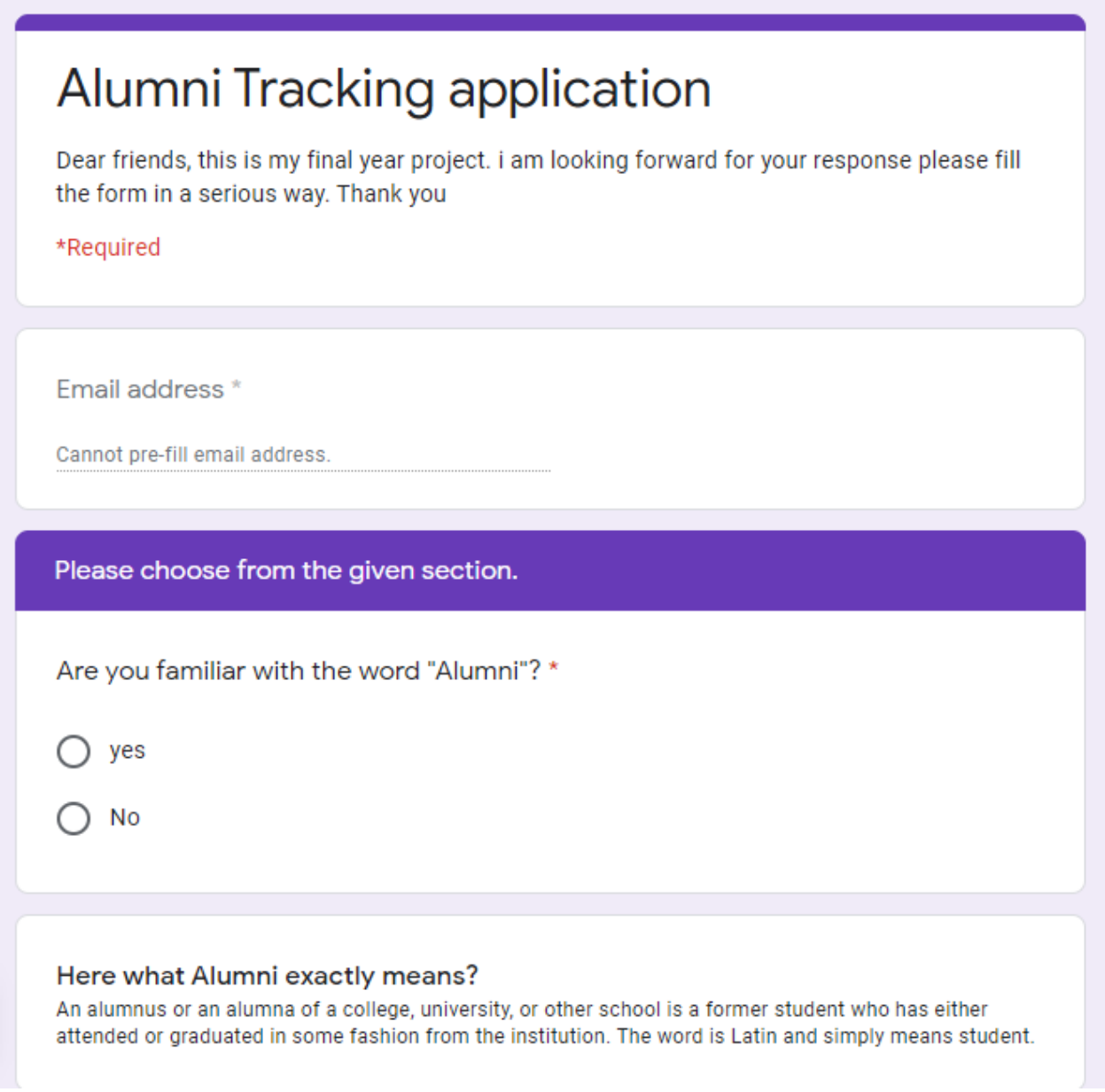


Figure 102 pre survey form 1.

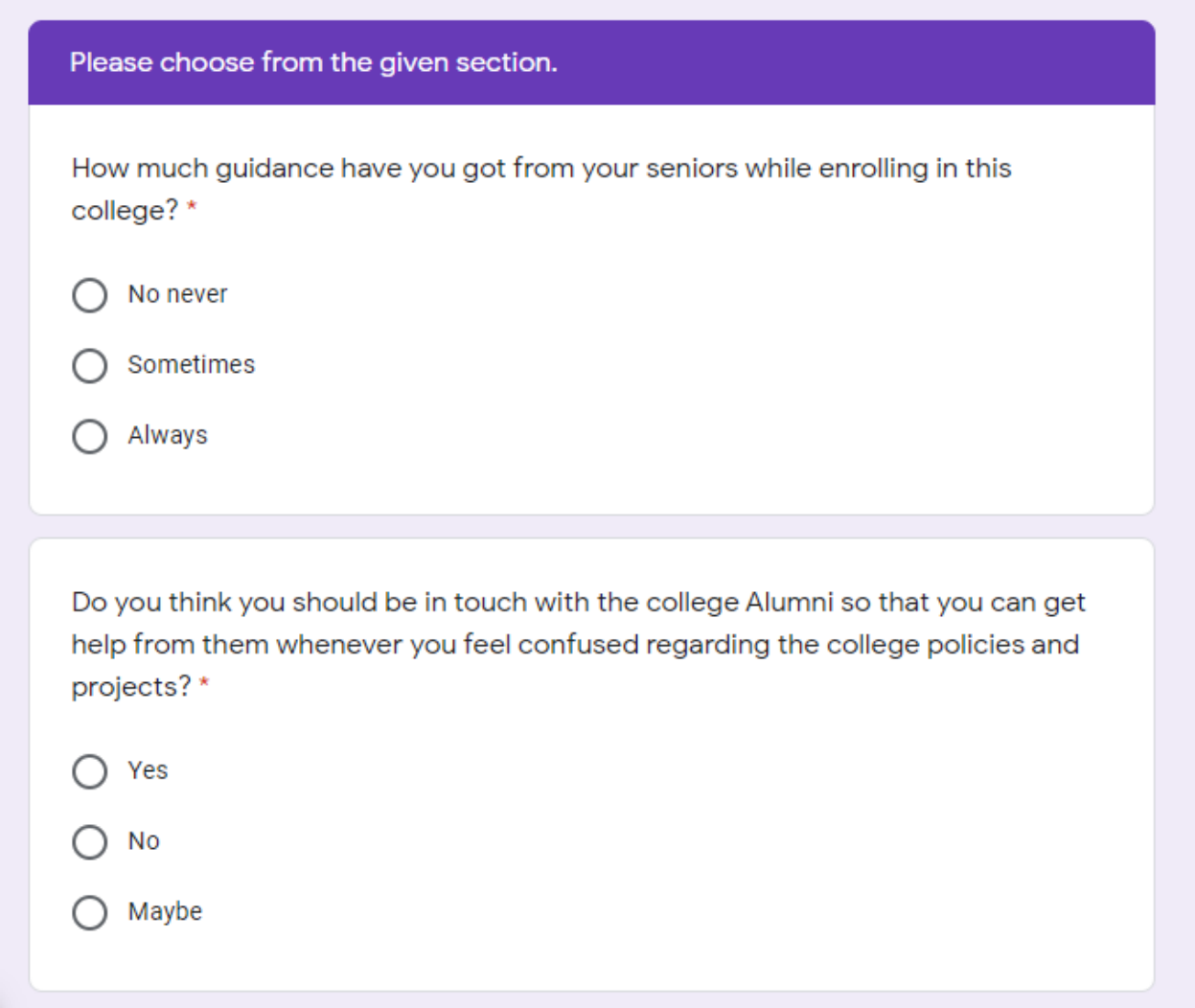


Figure 103 pre survey form 2.

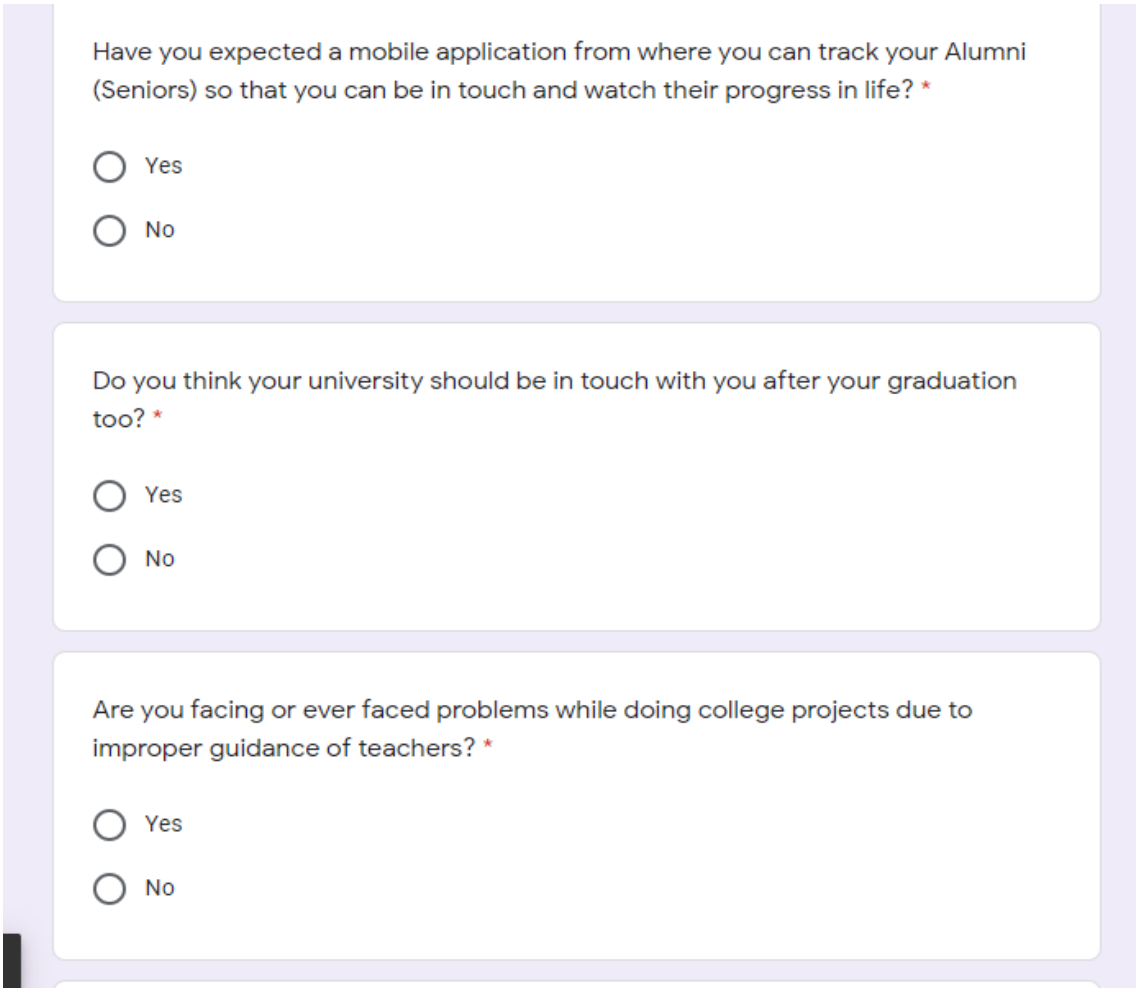


Figure 104 pre survey form 3.

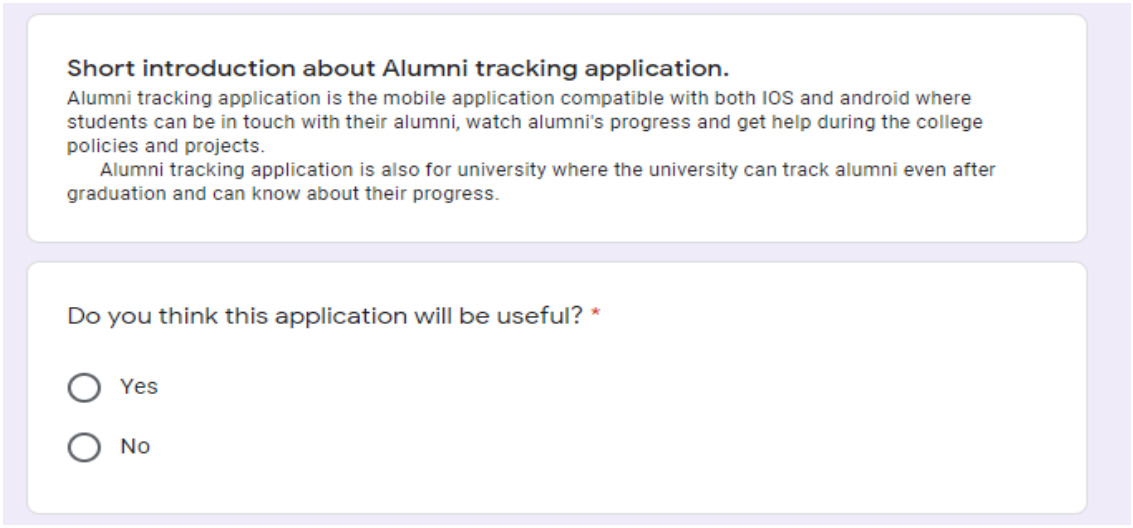


Figure 105 pre survey form 4.

8.1.2: Sample of filled pre survey form:

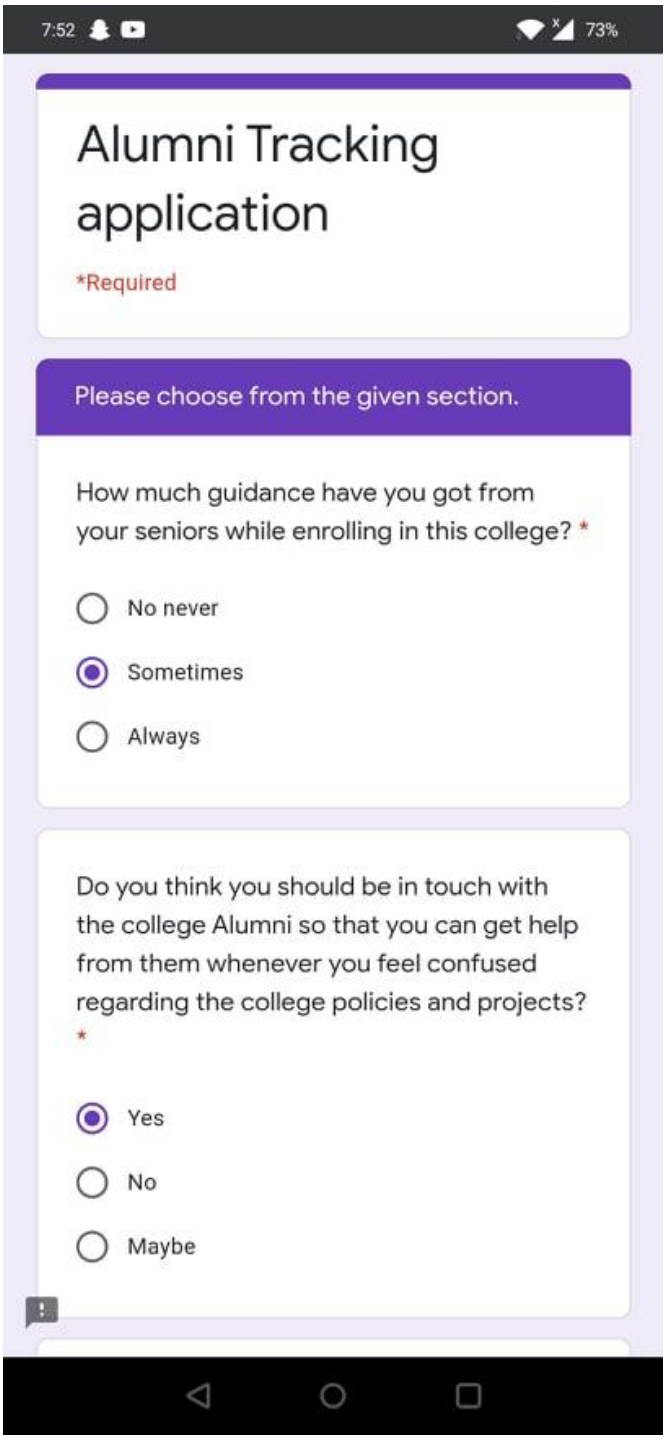


Figure 106 filled pre survey form 1.

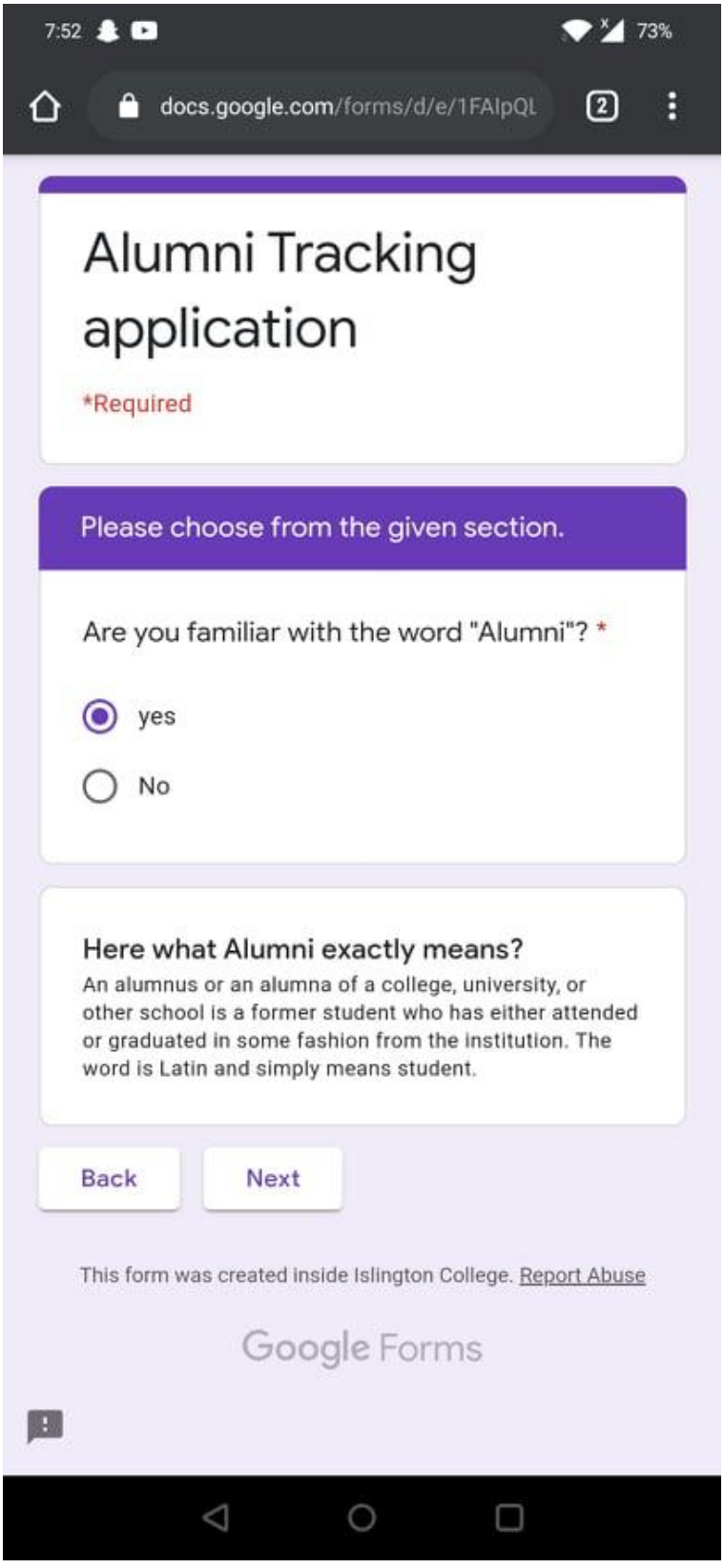


Figure 107 filled pre survey form 2.

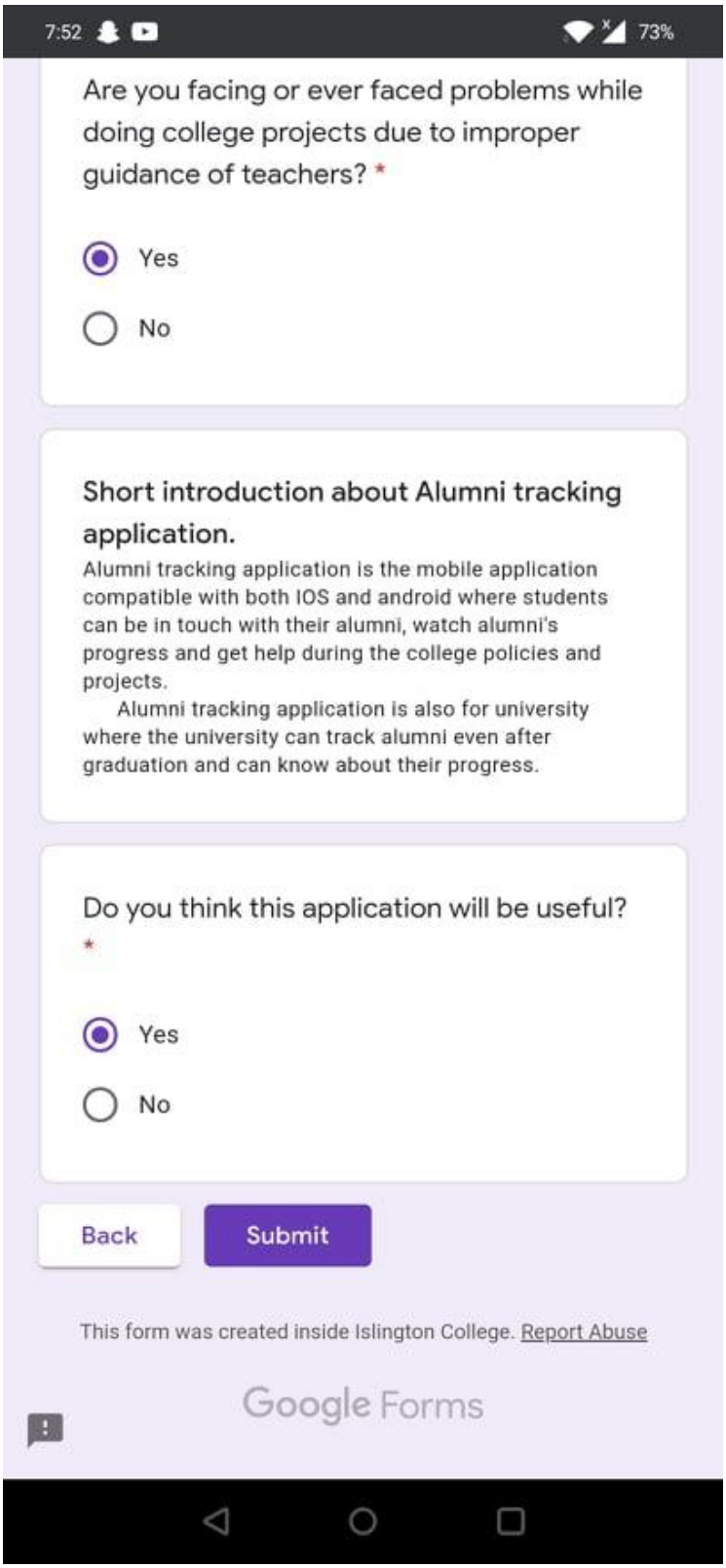


Figure 108 filled pre survey form 3.

8.1.3 Pre-Survey Result:

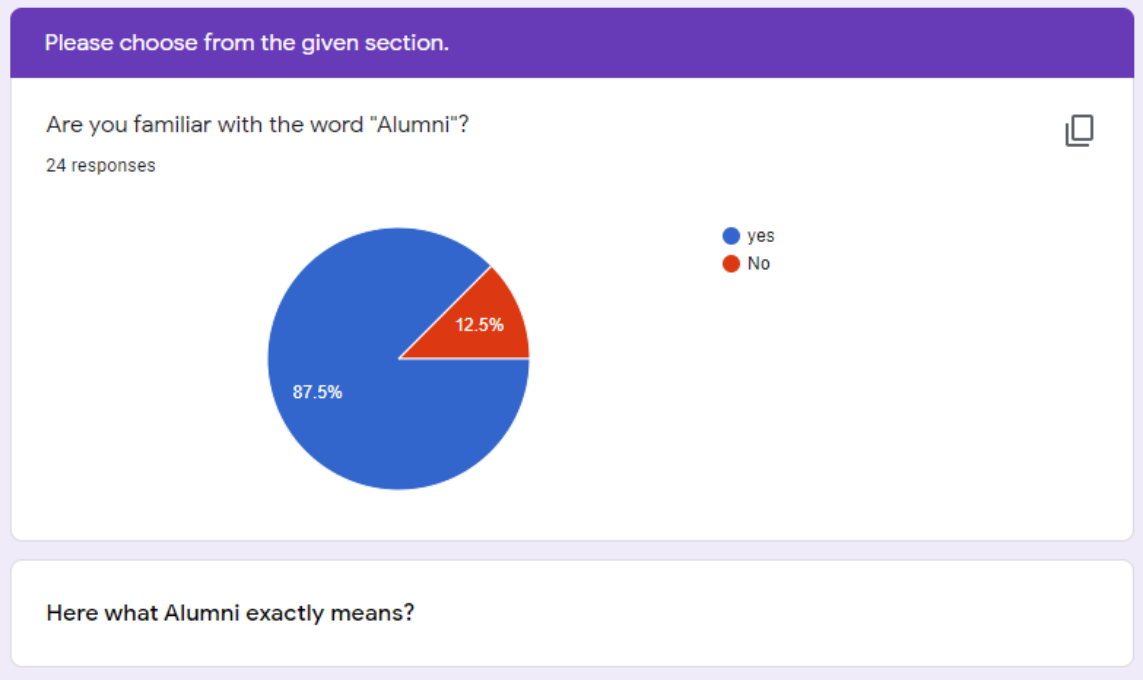


Figure 109 pre survey result 1.

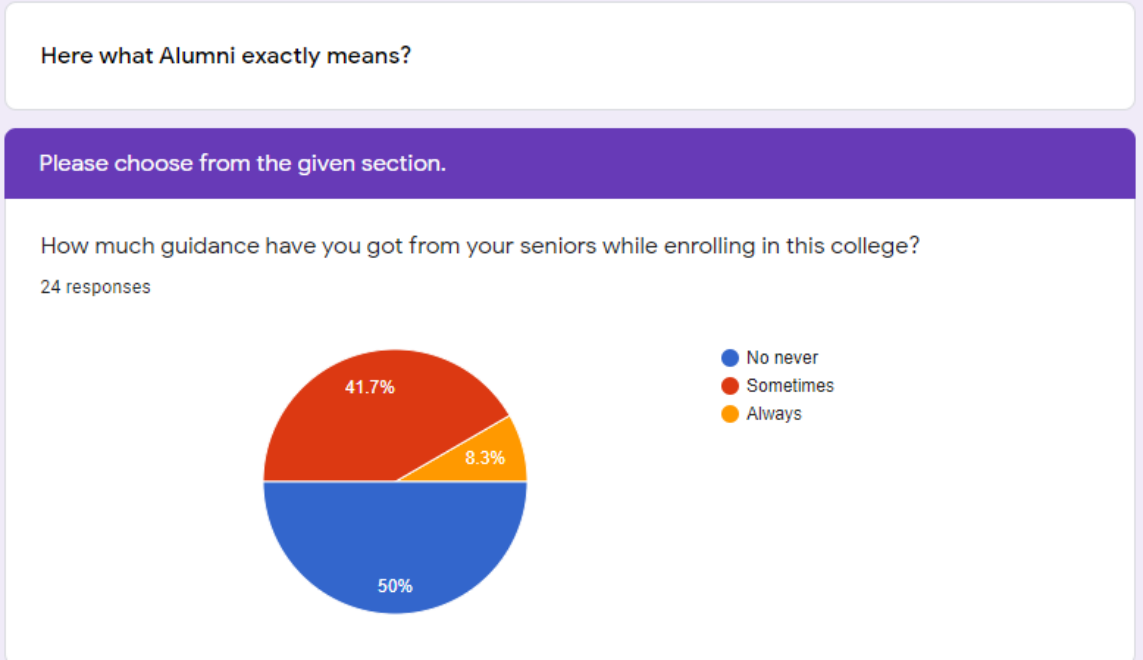


Figure 110 pre survey result 2.

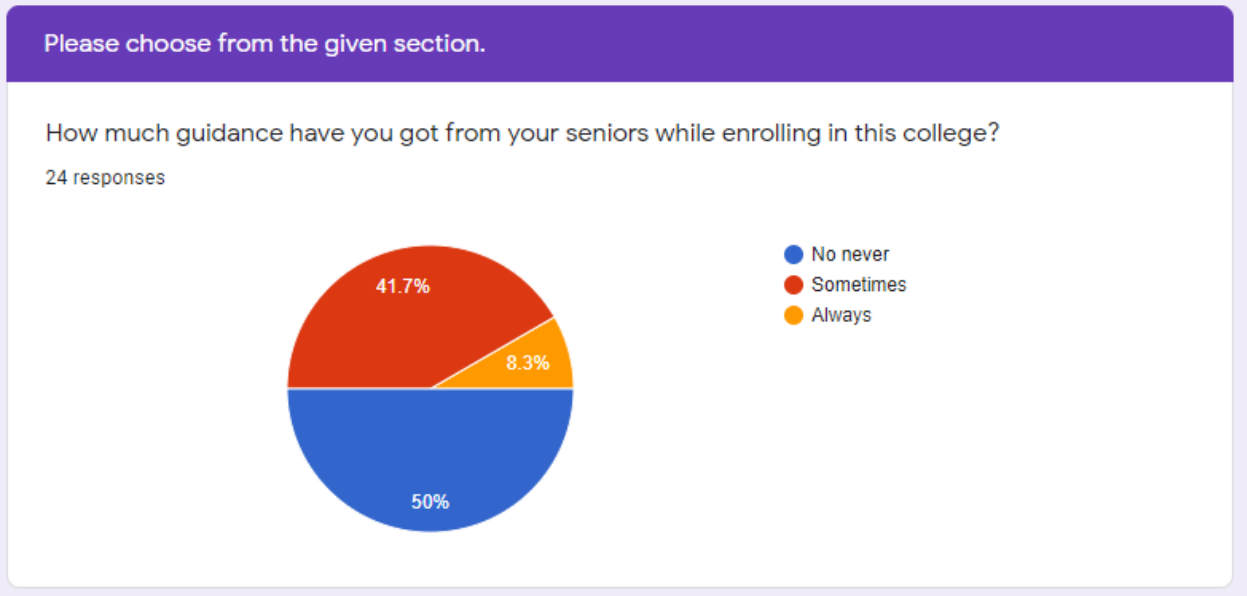


Figure 111 pre survey result 3.

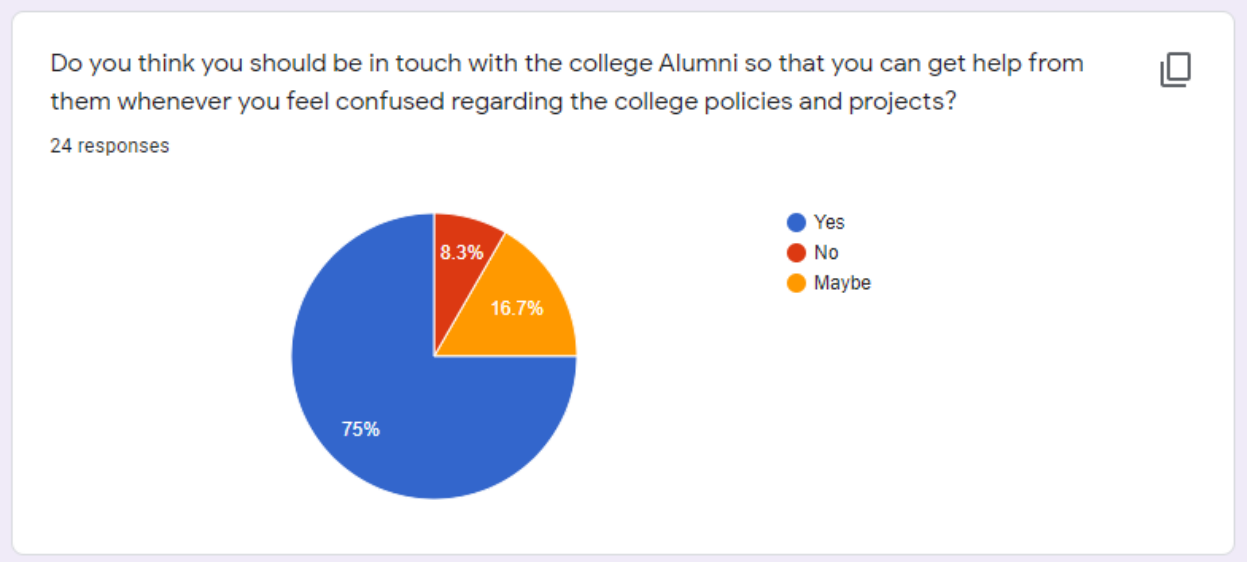


Figure 112 pre survey result 4.

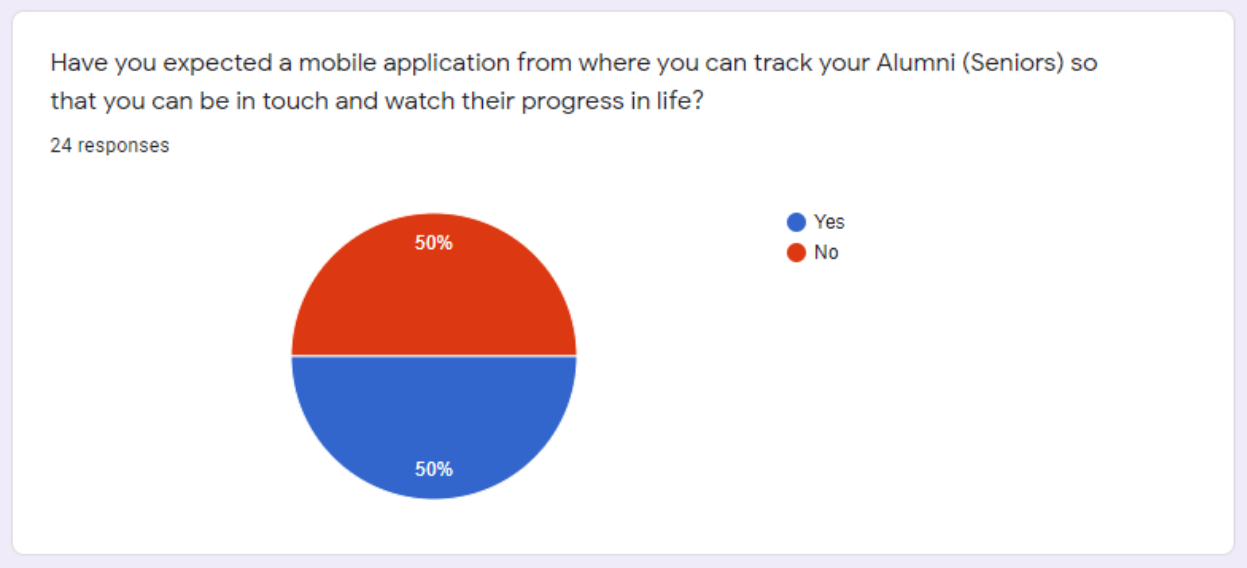


Figure 113 pre survey result 5.

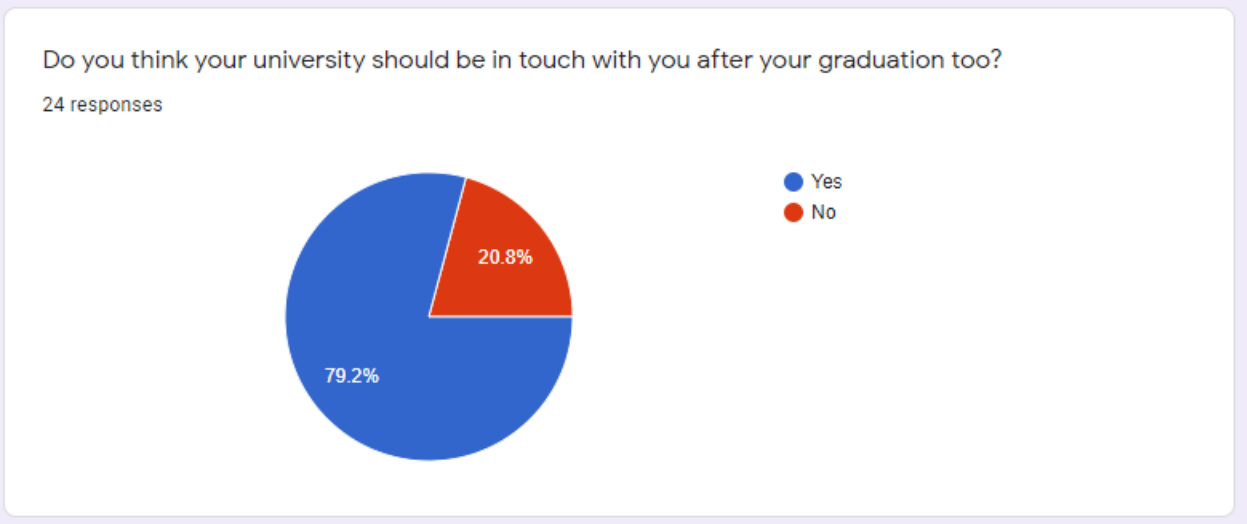


Figure 114 pre survey result 6.

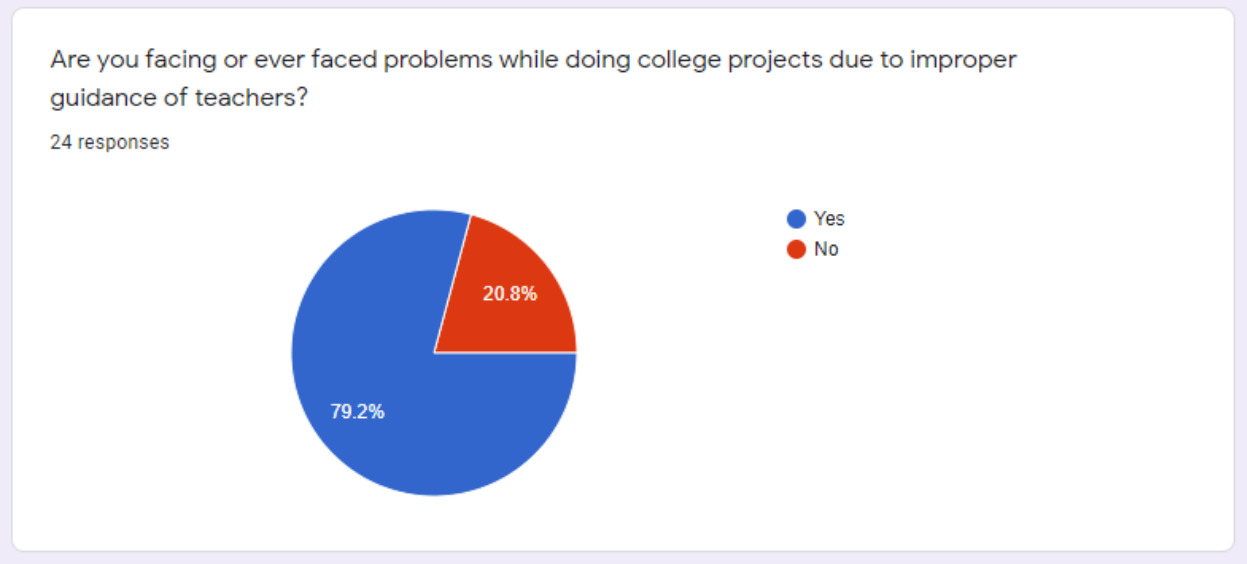


Figure 115 pre survey result 7.

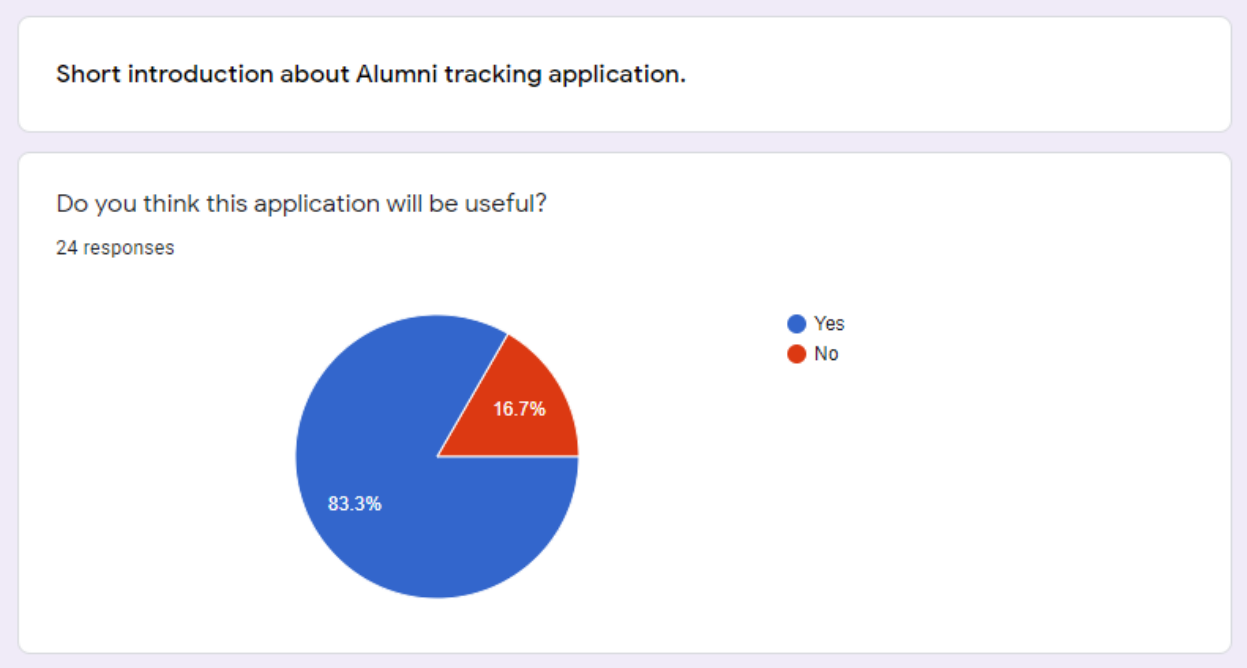


Figure 116 pre survey result 8.

8.2. Appendix B: Post – Survey:

8.2.1: Post Survey Form:

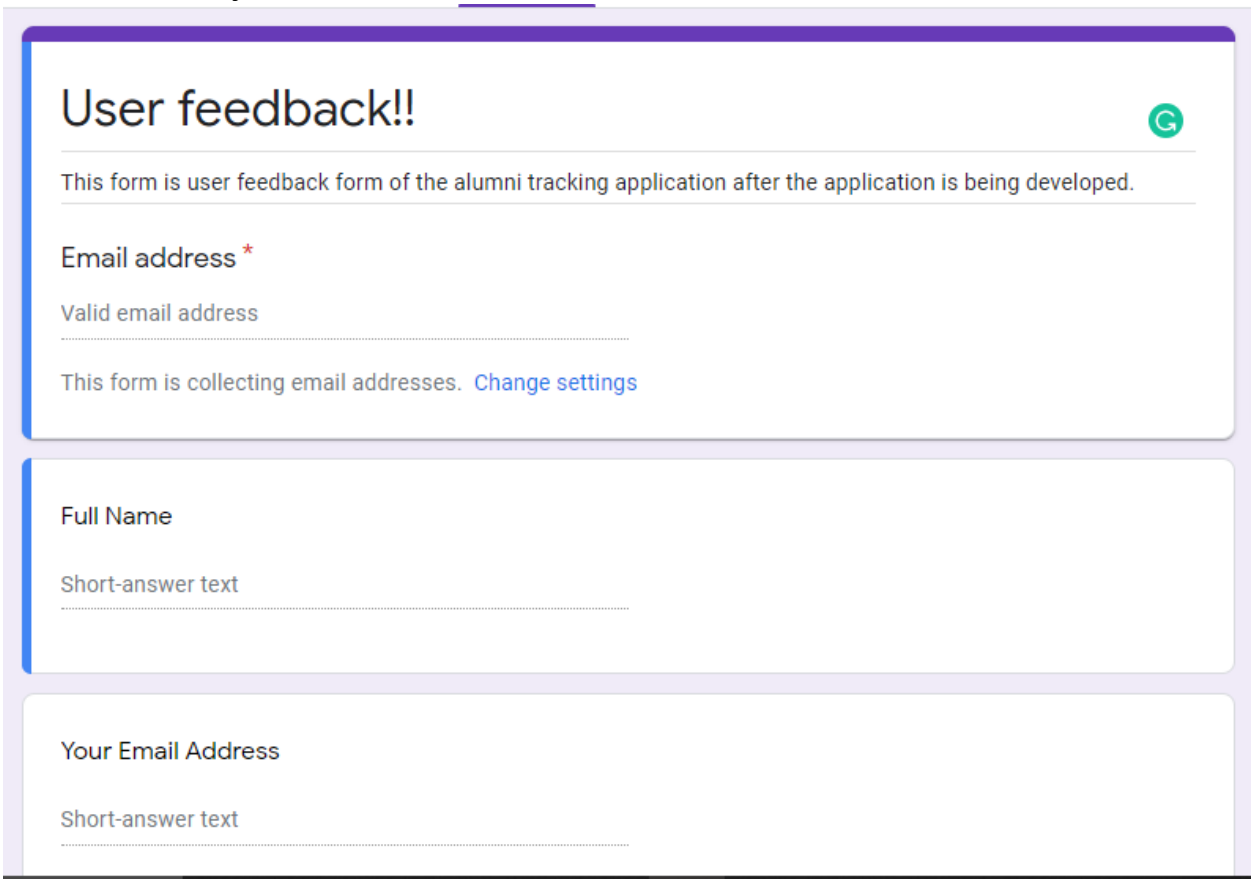


Figure 117 post survey form 1.

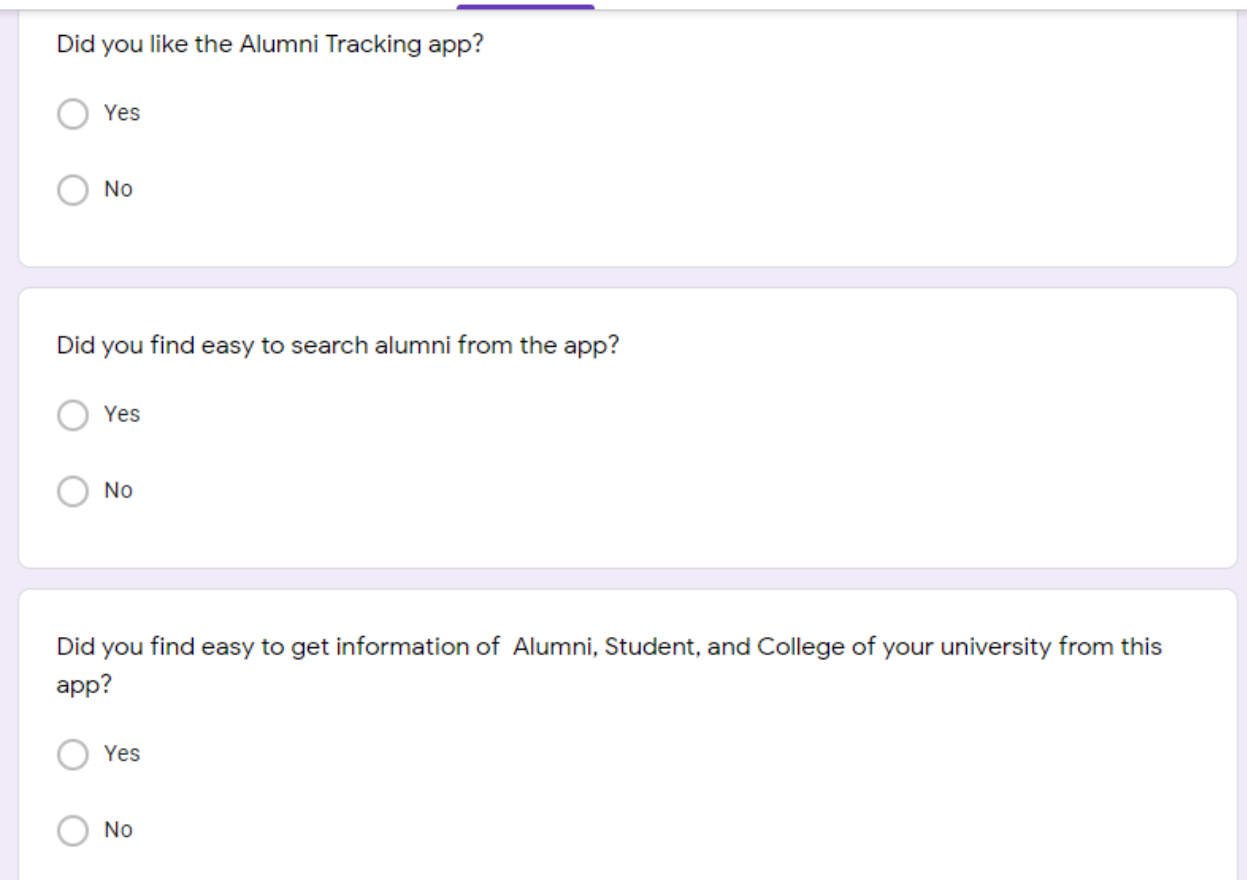


Figure 118 post survey form 2.

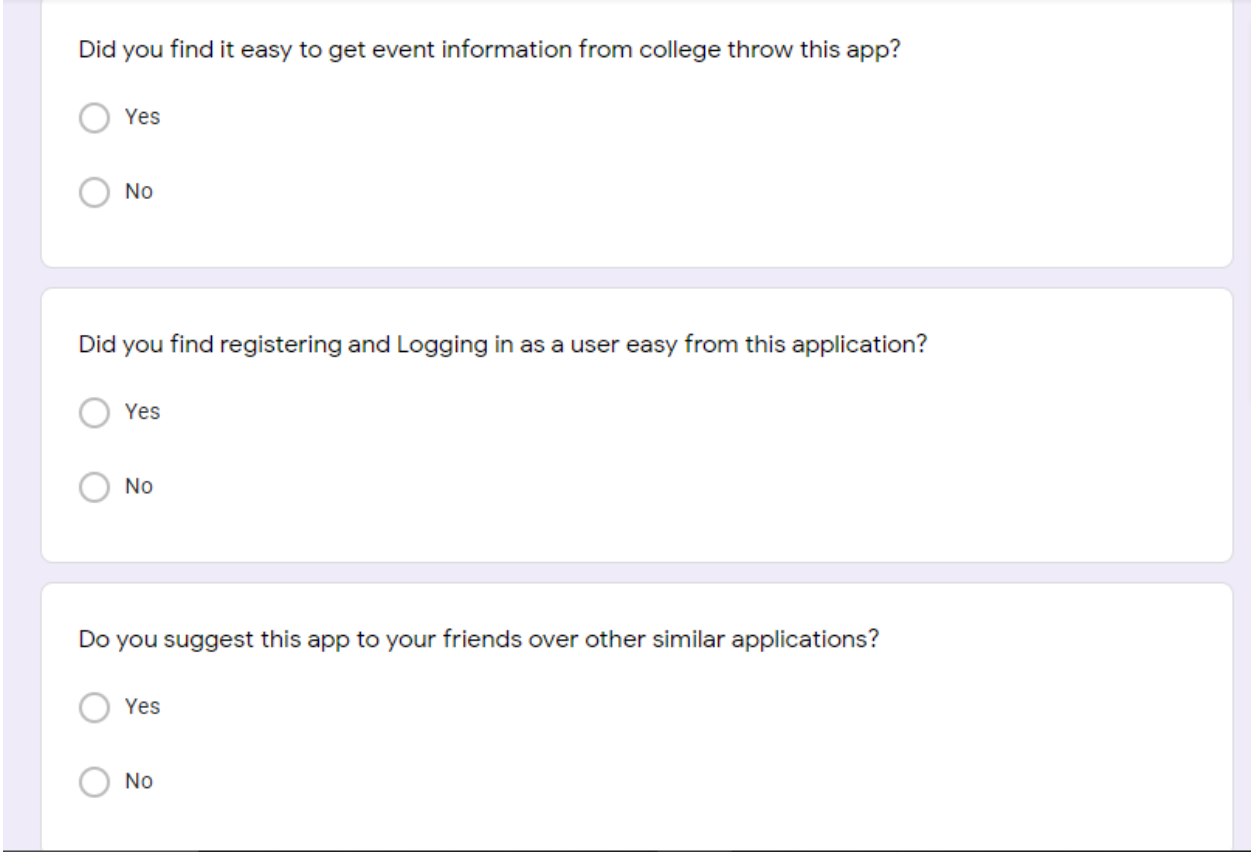


Figure 119 post survey form 3.

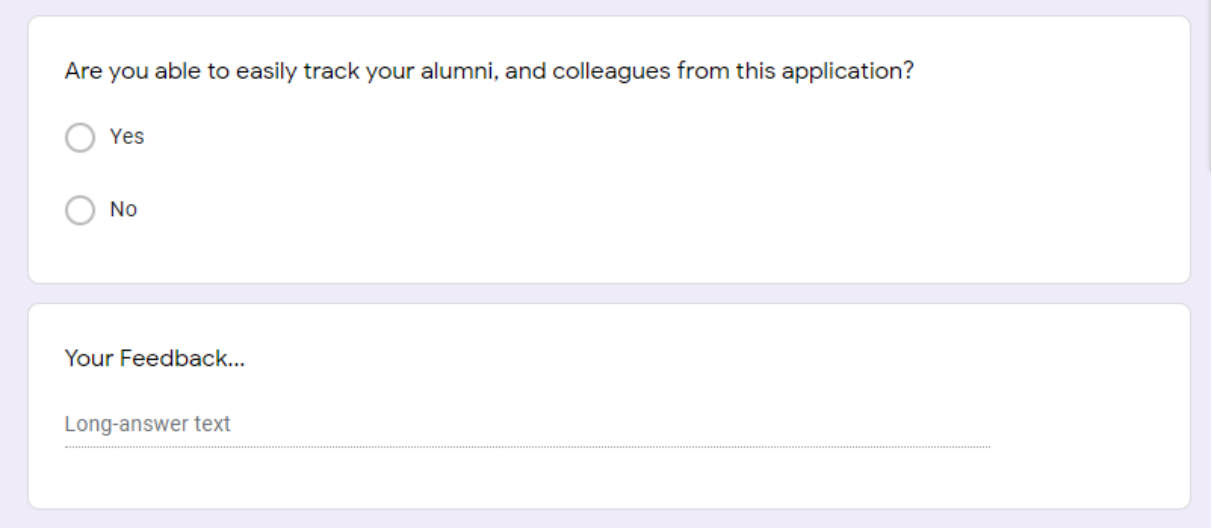


Figure 120 post survey form 4.

8.2.2: Sample of filled post survey form:

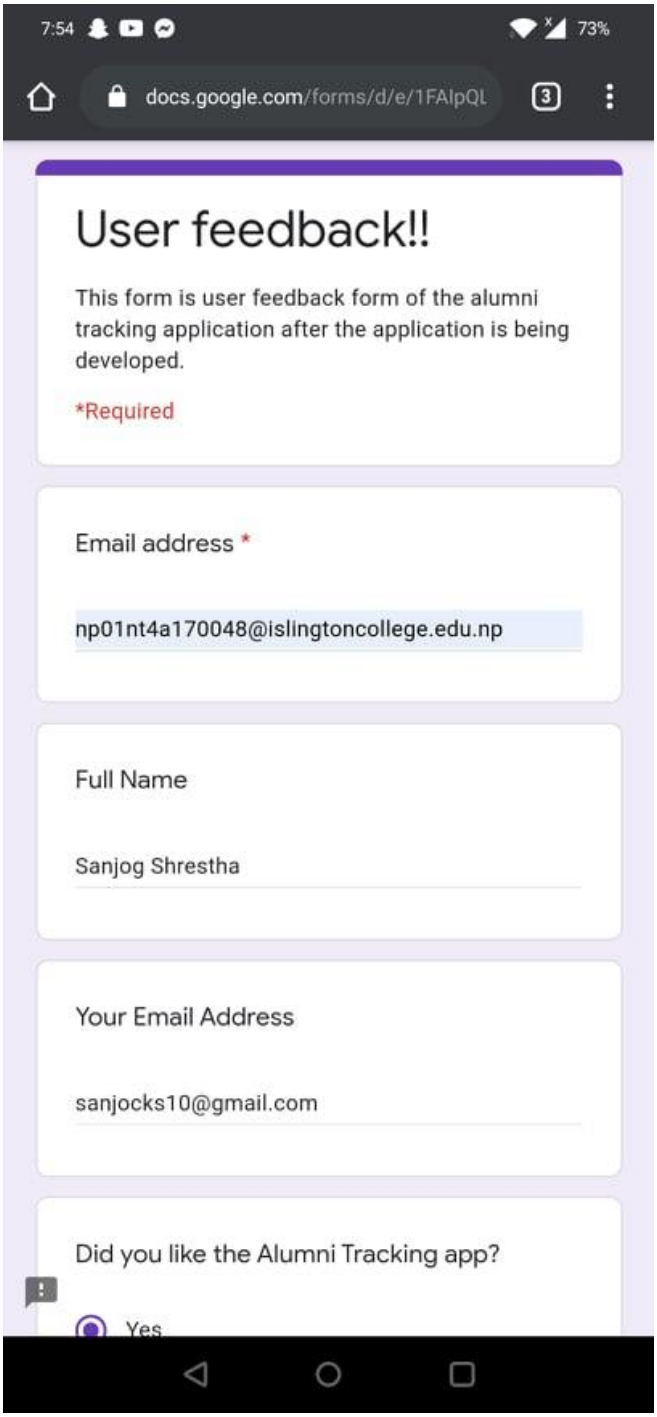


Figure 121 filled post survey form 1.

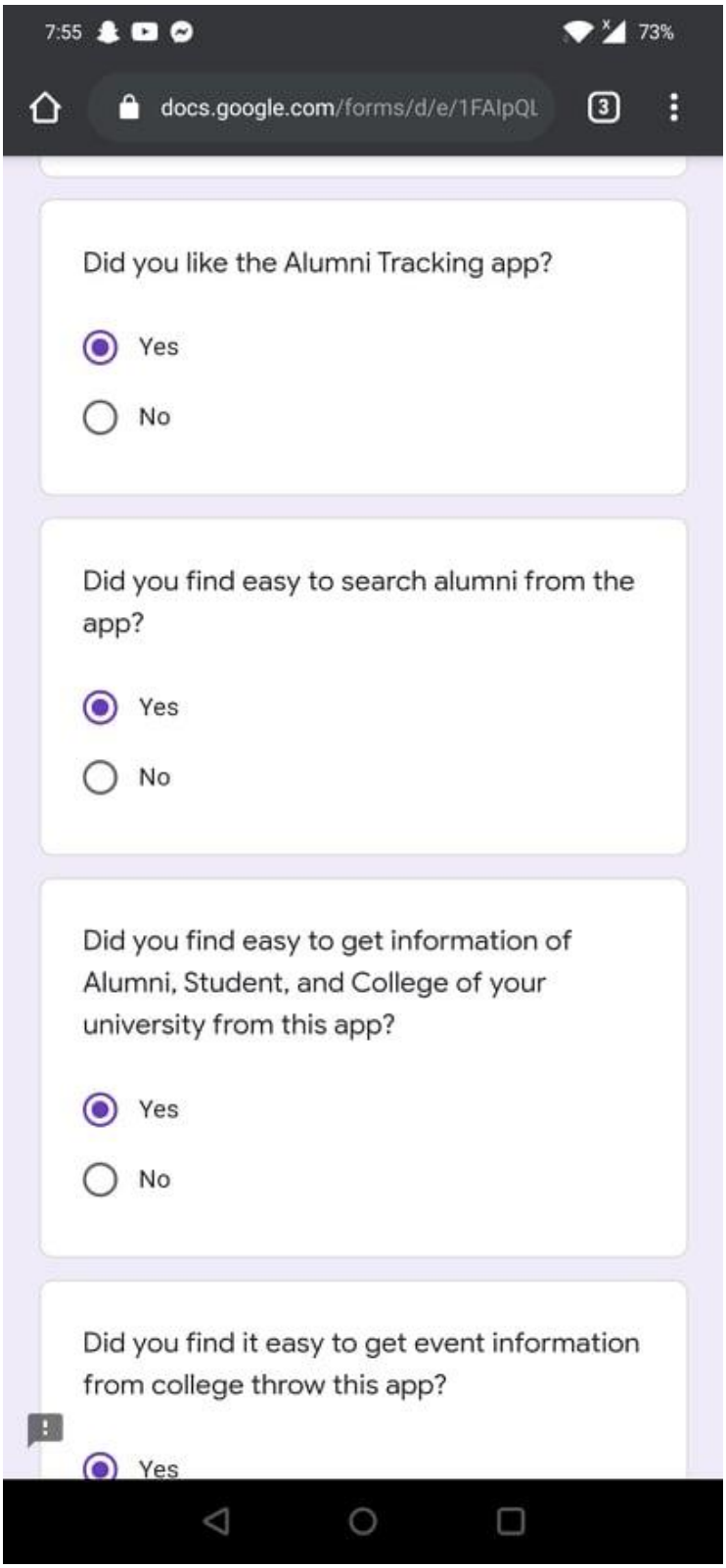


Figure 122 filled post survey form 2.

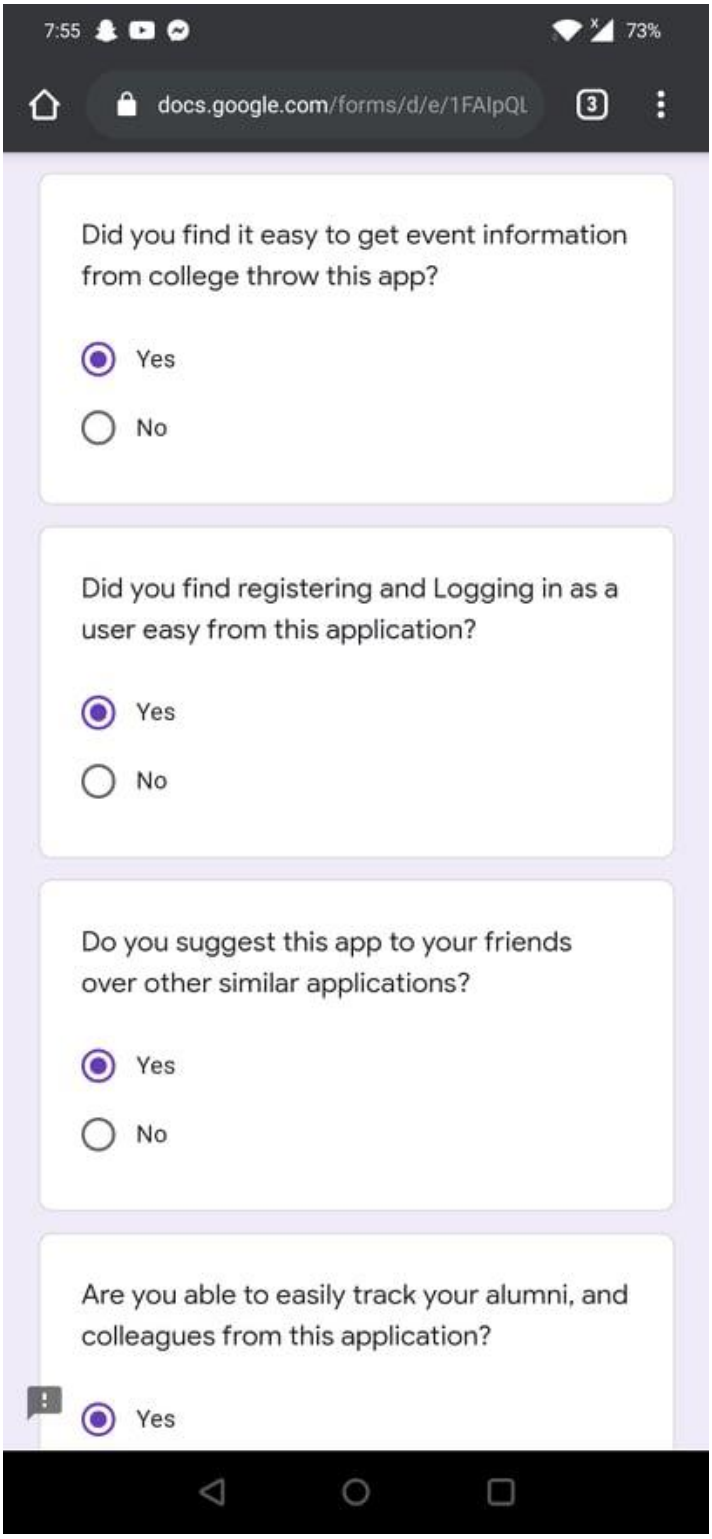


Figure 123 filled post survey form 3.

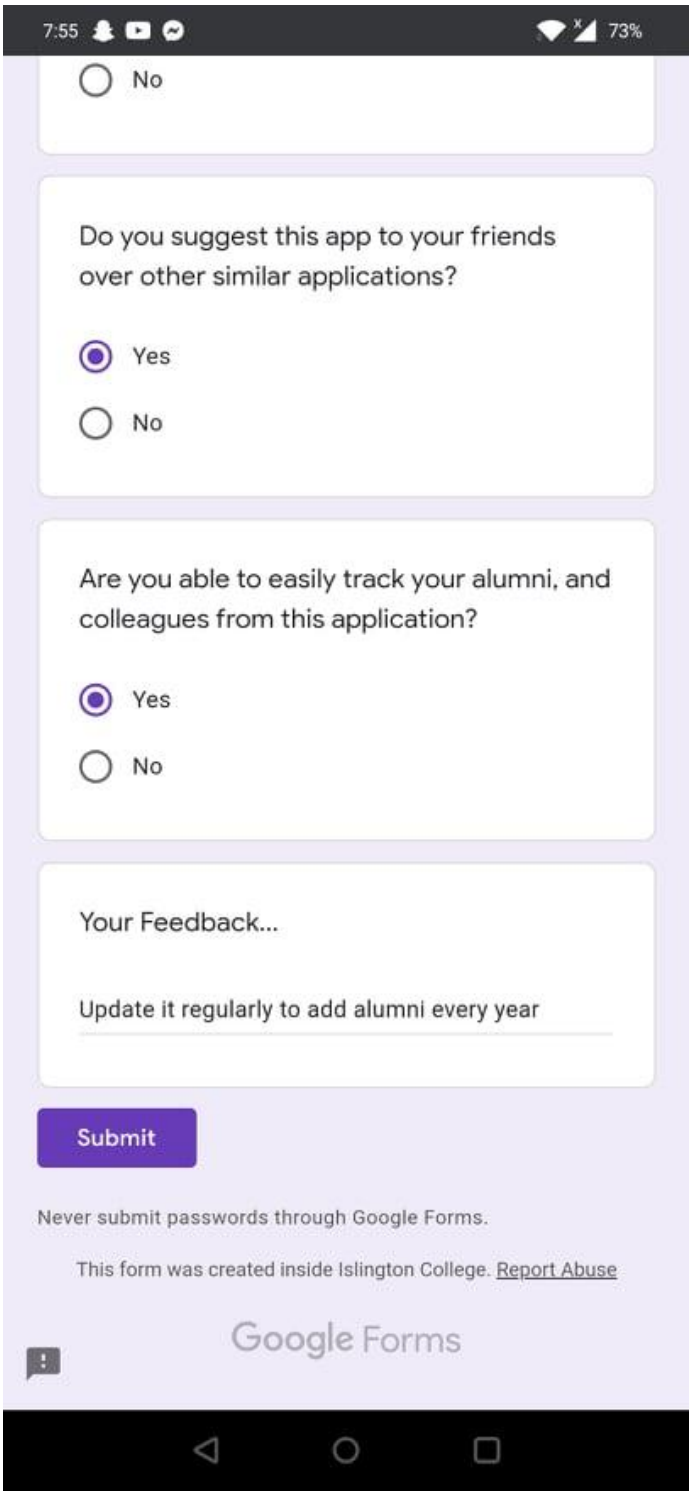


Figure 124 filled post survey form 4.

8.2.3: Post-Survey Result:

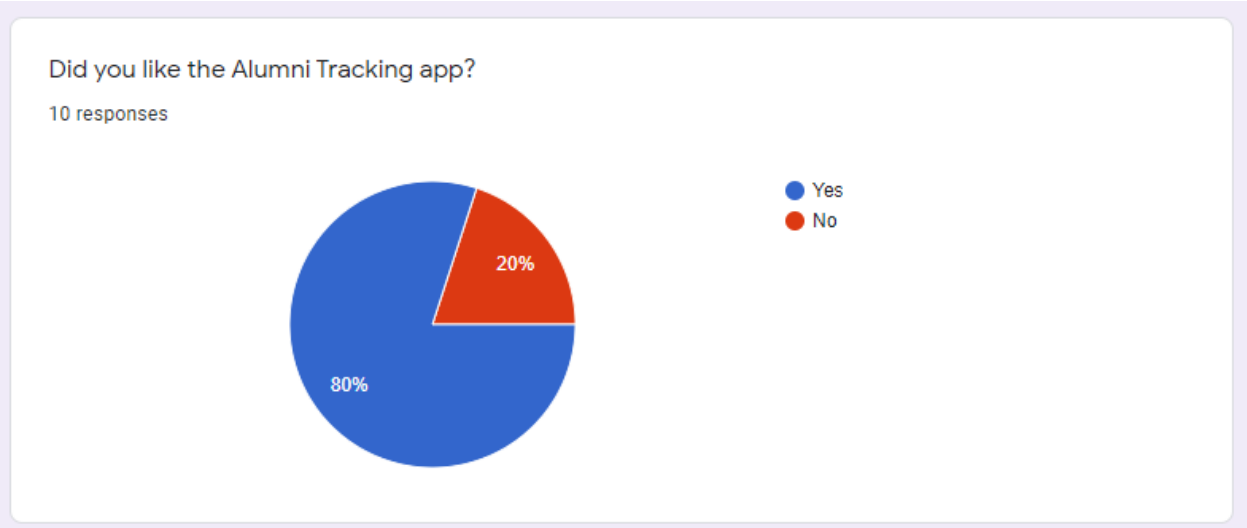


Figure 125 post survey results 1.

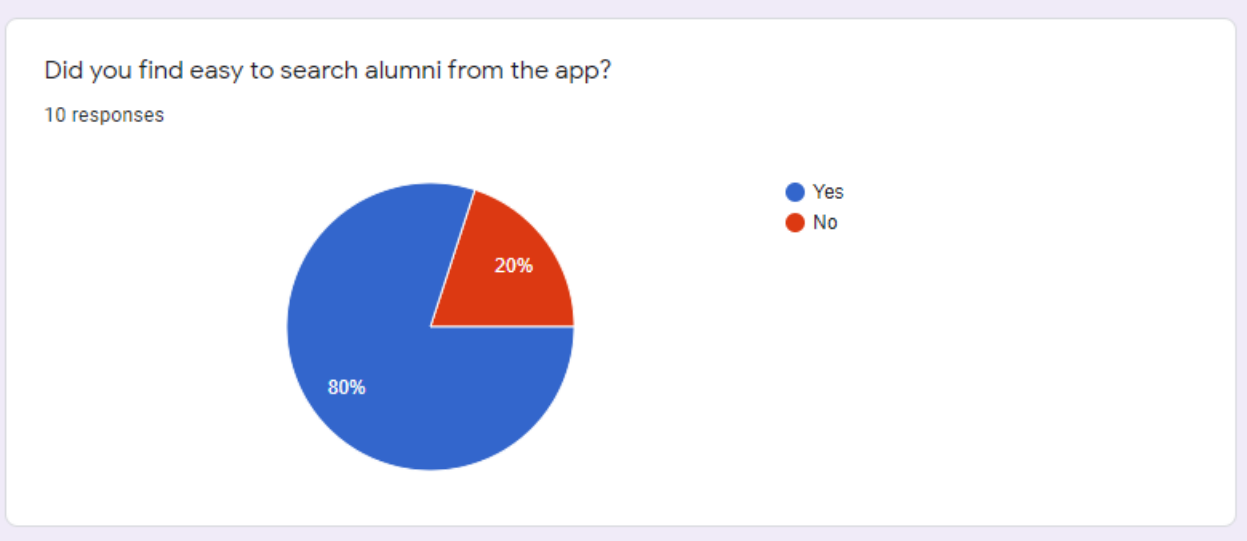


Figure 126 post survey results 2.

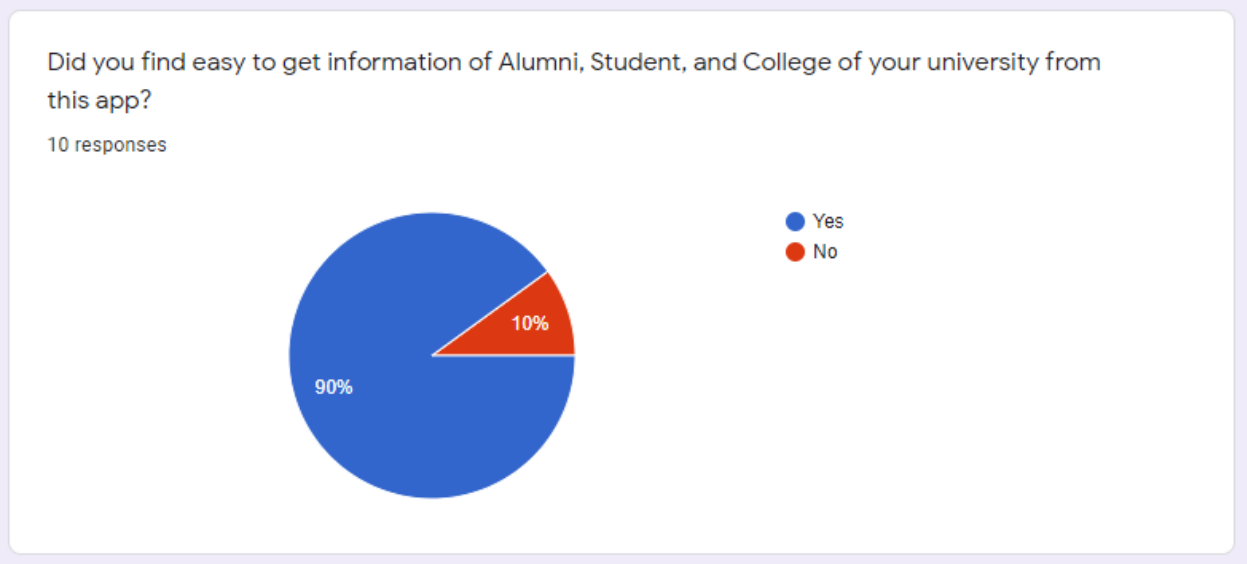


Figure 127 post survey results 3.

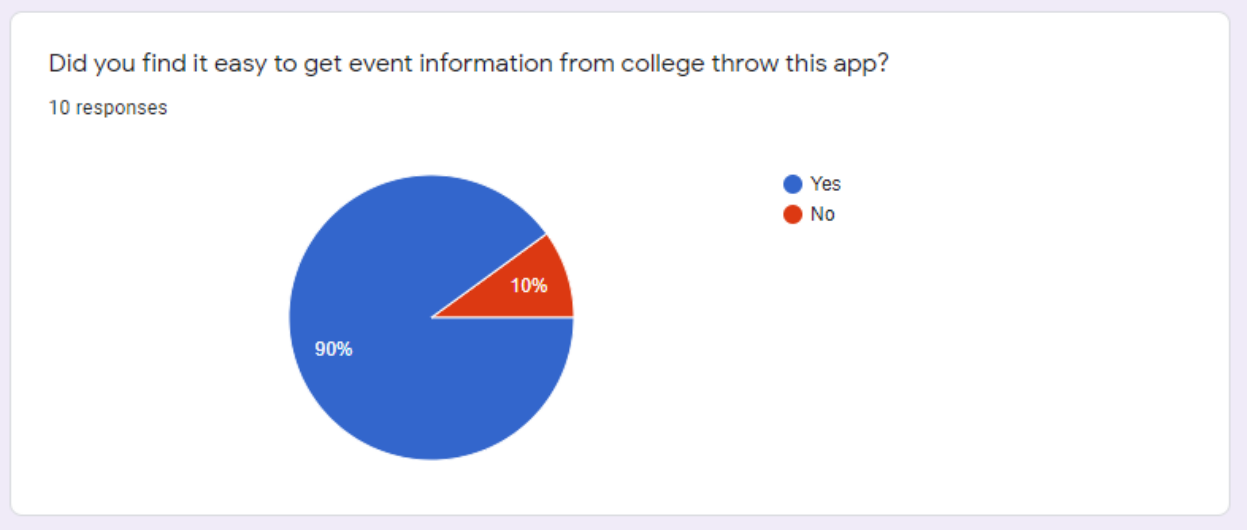


Figure 128 post survey results 4.

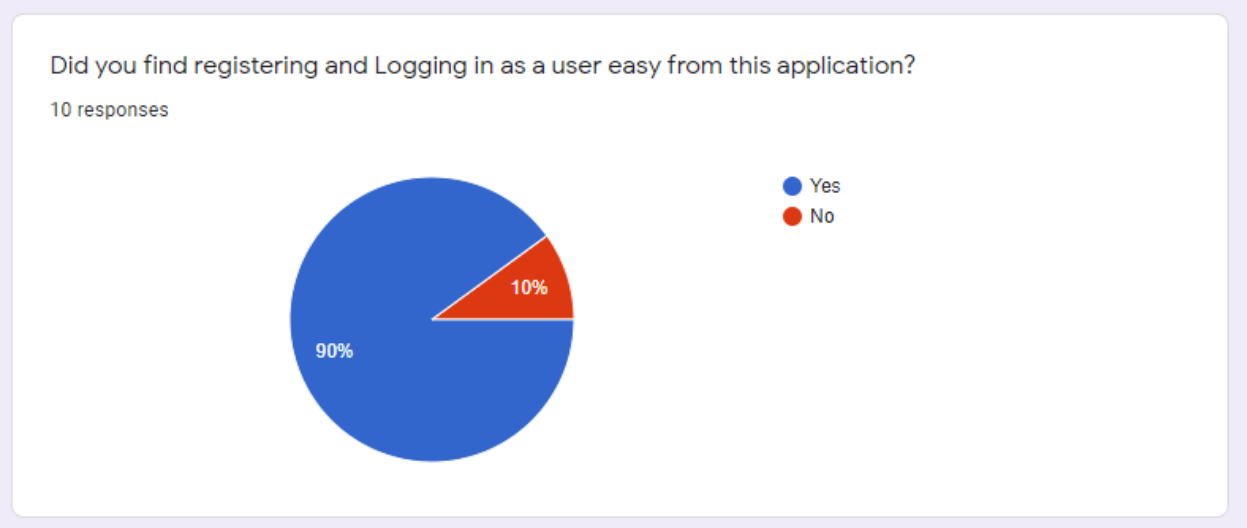


Figure 129 post survey results 5.

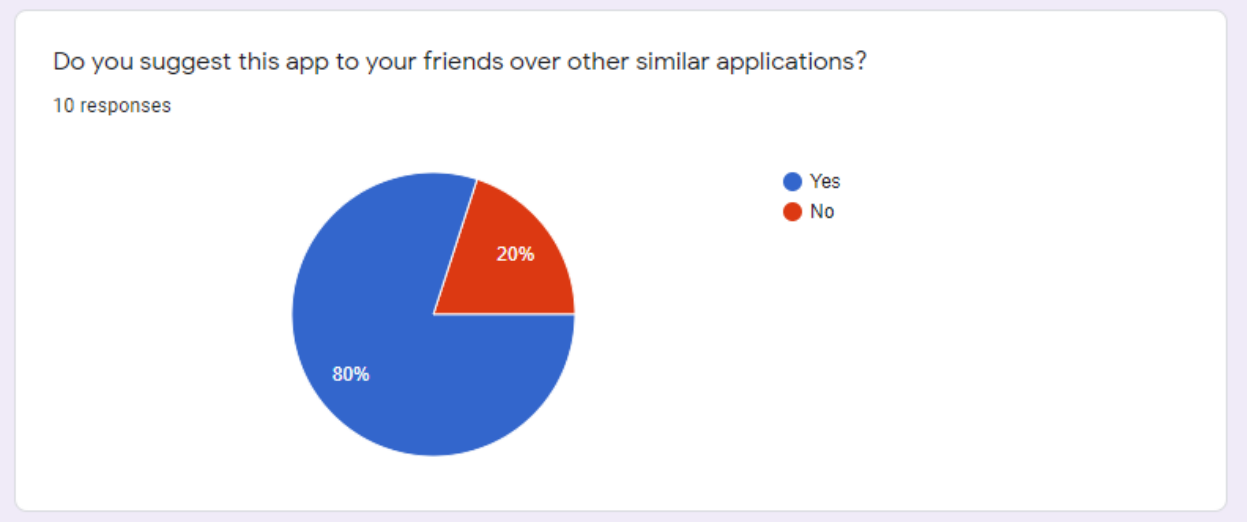


Figure 130 post survey results 6.

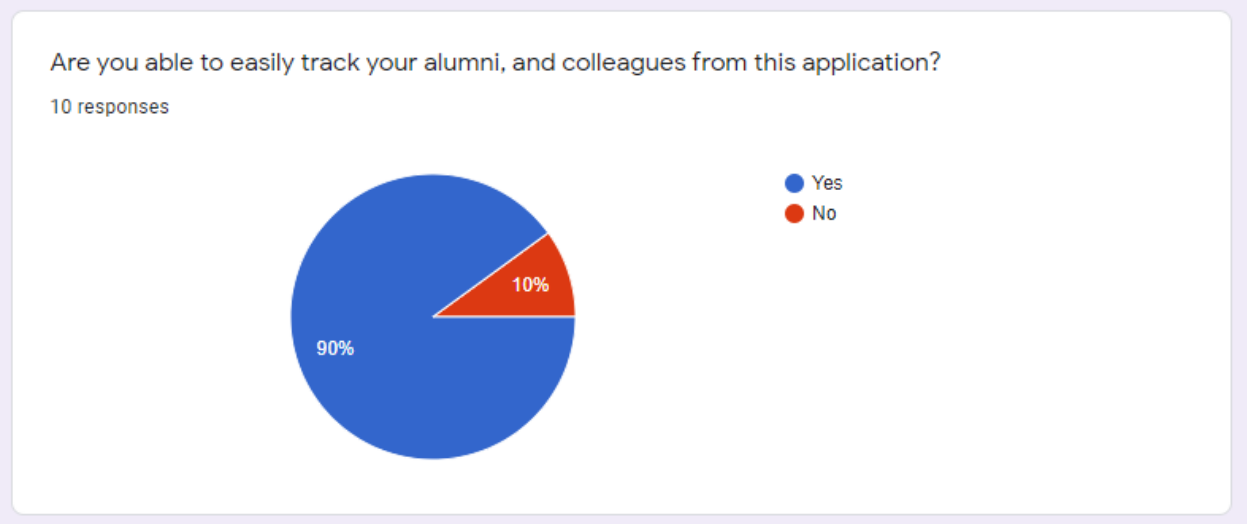


Figure 131 post survey results 7.

Amazing features and user friendly

Nice application

I found the application is very useful, but I am expecting a chat-bot in the near future.
#superapp #alumnitrackingapp

application is overall very useful but i expect an news feed page where users can view other users daily achievements instead of visiting their individual profile.

Figure 132 post survey feedback.

8.3: Appendix C: Sample Codes:

8.3.1: Sample Codes of the Mobile App:

Main page:

```
import 'package:alumniapp/loginpage.dart';
import 'package:alumniapp/notificationPage.dart';
import 'package:alumniapp/userprofile.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:firebase_messaging/firebase_messaging.dart';


void main() {
  runApp(Main()); // this is the main method of the application
  SystemChrome.setSystemUIOverlayStyle(SystemUiOverlayStyle(
    systemNavigationBarColor: Colors.blue[900], // navigation bar color
    statusBarColor: Colors.blue[900], // status bar color
  ));
}

class Main extends StatelessWidget {
  @override
  Widget build(BuildContext context) {

    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Alumni App',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        body: LoginPage(), //this calls loginpage as a body while launching the a
pp
      ),
    );
  }
}
```

Login page:

```
import 'package:alumniapp/homepage.dart';
import 'dart:async';
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:alumniapp/registrationpage.dart';
import 'package:url_launcher/url_launcher.dart';
import 'package:url_launcher/url_launcher.dart';

import 'api.dart';

class LoginPage extends StatefulWidget {
  @override
  State createState() => new LoginPageState();
}

class LoginPageState extends State<LoginPage> {
  // Create a text controller and use it to retrieve the current value
  // of the TextField.

  final emailController = TextEditingController();
  final passwordController = TextEditingController();

  bool _isLoading = false;

  ScaffoldState scaffoldState;

  _showMsg(msg) { // this method shows message in snackbar of login page.

    final snackBar = SnackBar(
      content: Text(msg),
      action: SnackBarAction(
        label: 'Close',
        onPressed: () {},
      ),
    );
    Scaffold.of(context).showSnackBar(snackBar);
  }

  @override
  void dispose() {
    // Clean up the controller when the widget is disposed.
```

```
  emailController.dispose();
  passwordController.dispose();
  super.dispose();
}

final _formKey = GlobalKey<FormState>();
var sizebox = SizedBox(
  height: 10,
);

Widget build(BuildContext context) { //this widget will run at first
  return Scaffold(
    resizeToAvoidBottomPadding: false,
    appBar: AppBar(
      title: Text('Login to Your Account'), // this is the app bar
      backgroundColor: Colors.blue[900],
    ),
    body: Padding(
      padding: const EdgeInsets.all(8.0),
      child: Center(
        child: Form(
          key: _formKey,
          autovalidate: true,
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              alumnilogo(context),
              TextFormField( // text field for entering email
                controller: emailController,
                keyboardType: TextInputType.emailAddress,
                decoration: InputDecoration(
                    prefixIcon: Icon(Icons.email),
                    labelText: "Email",
                    border: OutlineInputBorder(
                        borderRadius: BorderRadius.circular(10.0))),
              ),
              sizebox,
              TextFormField( //text field for entering password
                controller: passwordController,
                obscureText: true,

                decoration: InputDecoration(
                    prefixIcon: Icon(Icons.lock),
                    labelText: "Password",
                    border: OutlineInputBorder(
```

```
                        borderRadius: BorderRadius.circular(10.0))),
                ),
                sizebox,
                ButtonTheme( // button for login
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(10.0)),
                  minWidth: MediaQuery.of(context).size.width - 10,
                  height: 45,
                  child: RaisedButton(
                    color: Colors.blue[900],
                    child: Text(
                      _isLoading ? 'Logging in...' : "Login",
                      style: TextStyle(
                          fontWeight: FontWeight.bold,
                          fontSize: 30,
                          color: Colors.white),
                    ),
                    onPressed: () {


                      setState(() {
                        _isLoading = true;



                      });

                      signIn(emailController.text, passwordController.text); // t
his will call signin method and pass values on it.
                    },
                  ),
                ),
                MaterialButton(
                  splashColor: Colors.blue[900],
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(10.0)),
                  onPressed: () {
                    launch('http://192.168.0.114:8000/password/reset'); // this w
ill run password reset page
                  },
                  child: Text(
                    "Forgot Password?",
                    style: TextStyle(fontSize: 20.0),
                  ),
                  minWidth: MediaQuery.of(context).size.width,
```

```
                ),
                Expanded( // this is for registering new users
                  child: Column(
                    mainAxisAlignment: MainAxisAlignment.end,
                    children: <Widget>[
                      Row(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: <Widget>[
                          Text("Not registered Yet?"),
                          MaterialButton(
                            splashColor: Colors.blue[900],
                            shape: RoundedRectangleBorder(
                                borderRadius: BorderRadius.circular(5.0)),
                            minWidth: 5,
                            height: 5,
                            onPressed: () {
                              showDialog( // this will show roles option while re
gistering users
                                  context: context,
                                  builder: (BuildContext context) {
                                    return AlertDialog(
                                      title: Text('Choose Your role',style: TextS
tyle(fontWeight:FontWeight.bold),),
                                      content: Text('Register As'),
                                      actions: <Widget>[
                                        Column(
                                          children: <Widget>[
                                            Container(
                                              width: 400,
                                              child: RaisedButton(
                                                color: Colors.blue[900],
                                                onPressed: () { // this will call
 registration page and pass role as parameter
                                                  Navigator.push(
                                                    context,
                                                    MaterialPageRoute(
                                                        builder: (context) =>
                                                            Registrationpage(0)),
                                                  );
                                                },
                                                child: Text('Admin'),
                                              ),
                                            ),
                                            Container(
                                              width: 400,
```

```
                child: RaisedButton(
                  color: Colors.blue[900],
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                          builder: (context) =>
                              Registrationpage(1)),
                    );
                  },
                  child: Text('College'),
                ),
              ),
              Container(
                width: 400,
                child: RaisedButton(
                  color: Colors.blue[900],
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                          builder: (context) =>
                              Registrationpage(2)),
                    );
                  },
                  child: Text('Alumni'),
                ),
              ),
              Container(
                width: 400,
                child: RaisedButton(
                  color: Colors.blue[900],
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                          builder: (context) =>
                              Registrationpage(3)),
                    );
                  },
                  child: Text('Student'),
                ),
              ),
            ],
          )
```

```
                                    ],
                                  );
                                });
                            },
                            child: Text("Sign Up"),
                          ),
                        ],
                      ),
                    ],
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }

  void signIn(String email, password) async { // this function will handle all ap
i login authentication

    setState(() {
      _isLoading = true;
    });
    Map data = {'email': email, 'password': password}; // mapps api data with inp
ut data
    // try{
    var jsonData = null;
    SharedPreferences sharedpreferences = await SharedPreferences.getInstance();

    var response = await CallApi().postData(data, 'login'); // post data of login
 to api and checks the validaty
    if (response.statusCode == 200) { // checks if user is valid or not
      jsonData = json.decode(response.body);

      setState(() {

      sharedpreferences.setString('data', data.toString());
        sharedpreferences.setString("token", jsonData['access_token']); // saves
token in system shared prefernces for login
        var accessToken = jsonData['access_token'];

        print(accessToken);
        print(data);
```

```
       Navigator.of(context).pushAndRemoveUntil(
           MaterialPageRoute(
               builder: (BuildContext context) => HomePage(accessToken)),
           (Route<dynamic> route) => false); // if user is valid returns user to
homepage also pass token as parameter.
     });

   } else {
     // var convertDataToJson = response.body;
     // datas = convertDataToJson['message'];
     if (emailController.text == "" || passwordController.text == ""){ // checks
validity
       _showMsg('Email or Password Cannot be Empty');

     }
     else{
     _showMsg(response.body); // shows error messages
     }
     // print(response.body);
   }
   // }catch (e){
   //   showDialog(
   //     context: context,
   //     builder: (context) {
   //       return AlertDialog(
   //         title: Text('No Internet Connected'),
   //         content: Text('Please turn on wifi or mobile data'),
   //         actions: <Widget>[
   //           RaisedButton(
   //             child: Text(
   //               'ok',
   //               style: TextStyle(color: Colors.white),
   //             ),
   //             onPressed: () {
   //               Navigator.of(context).pop();
   //             },
   //           ),
   //         ],
   //       );
   //     },
   //   );


   // }
```

```
    setState(() {
       _isLoading = false;
     });

  }
}

// String validateEmail(String value) {
//   Pattern emailPattern =
//       r'^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0
-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-
Z]{2,}))$';
//   RegExp emailRegExp = RegExp(emailPattern);

//   if (!emailRegExp.hasMatch(value)) {
//     return 'Please enter your valid Email';
//   } else {
//     return null;
//   }
// }

/*
 *
 */
Widget alumnilogo(context) { // this widget creates logo of alumni
  return Expanded(
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.center,
      children: <Widget>[
        Expanded(
          child: Padding(
            padding: const EdgeInsets.symmetric(vertical: 20.0),
            child: Material(
              child: Image(
                image: AssetImage('images/alumnilogo.png'),
                color: Colors.blue[900],
              ),
            ),
          ),
        )
      ],
    ),
  );
}
```

Registration page:

```
import 'dart:convert';

import 'package:alumniapp/api.dart';
import 'package:flutter/material.dart';
import 'package:flutter/widgets.dart';
import 'package:alumniapp/main.dart';
import 'dart:async';
import 'package:http/http.dart' as http;

class Registrationpage extends StatefulWidget {
  int getrole; // gets role from login page
  Registrationpage(this.getrole);

  @override
  State<StatefulWidget> createState() {
    return RegistrationpageState(this.getrole);
  }
}

class RegistrationpageState extends State<Registrationpage> {
  int getrole;
  RegistrationpageState(this.getrole);

  TextEditingController nameController = TextEditingController(); // text editing
 controller for registration form
  TextEditingController emailController = TextEditingController();
  TextEditingController phoneController = TextEditingController();
  TextEditingController symbolController = TextEditingController();
  TextEditingController addressController = TextEditingController();
  TextEditingController passwordController = TextEditingController();
  TextEditingController passwordConfController = TextEditingController();

  TextEditingController achievementController = TextEditingController();
  TextEditingController jobController = TextEditingController();

  bool _isLoading = false;

  List<DropdownMenuItem<int>> listDrop = []; // drop down for gender selection
  int selected = null;

  final _formKey = GlobalKey<FormState>(); /// form key for form data validation

  void loadData() { // function for seelcting gender from dropdown
```

```
    listDrop = [];
    listDrop.add(
      new DropdownMenuItem(
        child: new Text('Male'),
        value: 1,
      ),
    );
    listDrop.add(new DropdownMenuItem(
      child: new Text('Female'),
      value: 2,
    ));
  }


  @override
  Widget build(BuildContext context) {
    loadData();
    return Scaffold(
      resizeToAvoidBottomInset: false,
      body: ListView( // list view for showing registration form in scrolling vie
w
        children: <Widget>[
          Form(
            key: _formKey,
            autovalidate: true,
            child: Container(
              margin: EdgeInsets.only(top: 10),
              child: Padding(
                padding: const EdgeInsets.all(30.0),
                child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: <Widget>[
                      Center(
                        child: Container(
                          child: Image(
                            height: 100,
                            width: 100,
                            image: AssetImage('images/alumnilogo.png'),
                            color: Colors.blue[900],
                          ),
                        ),
                      ),
                      Padding(
                        padding: const EdgeInsets.all(8.0),
                        child: Center(
```

```
          child: Text(
        'Register for new user',
        style: TextStyle(
            fontWeight: FontWeight.bold, fontSize: 20),
      )),
    ),

    /*

    Text field for entering registration data of new users
    */
    TextFormField(
      controller: nameController,

      decoration: InputDecoration(
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(10),
        ),
        labelText: "Full Name",
        hintText: "Please enter your full name",
      ),

    ),
    SizedBox(
      height: 10,
    ),
    TextFormField(
      controller: emailController,
      keyboardType: TextInputType.emailAddress,

      decoration: InputDecoration(
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(10),
          ),
          labelText: "Email",
          hintText: "Please enter your valid email"),

    ),
    SizedBox(
      height: 10,
    ),
    TextFormField(

      controller: phoneController,
      keyboardType: TextInputType.phone,
```

```
                maxLength: 10,
                maxLengthEnforced: true,
                decoration: InputDecoration(
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(10),
                  ),
                  hintText: "Please enter your valid mobile number",
                  labelText: "Phone",
                ),

              ),
              SizedBox(
                height: 10,
              ),
              TextFormField(

                decoration: InputDecoration(

                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(10),
                  ),
                  hintText: "College Symbol No",

                ),

              ),
              SizedBox(
                height: 10,
              ),
              TextFormField(
                obscureText: true,

                controller: passwordController,
                decoration: InputDecoration(
                    border: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(10),
                    ),
                    labelText: "Password",
                    hintText: "Please create a Password"),
              ),
              SizedBox(
                height: 10,
              ),
              TextFormField(
```

```
                    controller: passwordConfController,
                    obscureText: true,
                    decoration: InputDecoration(
                        border: OutlineInputBorder(
                          borderRadius: BorderRadius.circular(10),
                        ),
                        labelText: "Password confirmation",
                        hintText: "Password should match"),

                  ),

                  SizedBox(
                    height: 10,
                  ),
                  Container(
                    height: 50,
                    width: 200,
                    child: DropdownButton(
                      value: selected,
                      items: listDrop,
                      hint: Text('Select your gender'),
                      onChanged: (value) {
                        selected = value;
                        setState(() {});
                      },
                    ),
                  ),
                  SizedBox(
                    height: 10,
                  ),
                  TextFormField(

                    controller: achievementController,
                    decoration: InputDecoration(
                      border: OutlineInputBorder(
                        borderRadius: BorderRadius.circular(10),
                      ),
                      labelText: "Enter your achievements",
                      hintText: 'Achievement',
                    ),
                  ),
                  SizedBox(
                    height: 10,
                  ),
                  SizedBox(
```

```
                  height: 10,
                ),
                TextFormField(

                  controller: addressController,
                  decoration: InputDecoration(
                      border: OutlineInputBorder(
                        borderRadius: BorderRadius.circular(10),
                      ),
                      labelText: "Address",
                      hintText: "Enter your Recent Address"),
                ),
                SizedBox(
                  height: 10,
                ),
                SizedBox(
                  height: 10,
                ),
                TextFormField(

                  controller: jobController,
                  decoration: InputDecoration(
                      border: OutlineInputBorder(
                        borderRadius: BorderRadius.circular(10),
                      ),
                      labelText: "Job",
                      hintText: "Enter your profession"),
                ),
                SizedBox(
                  height: 10,
                ),
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: Center(
                    child: Container(
                      width: 430,
                      height: 50,
                      child: RaisedButton(
                        shape: RoundedRectangleBorder(
                            borderRadius: BorderRadius.circular(10)),
                        onPressed: () {
```

```
                                _handleSubmit(); // will handle the registration
by launching this function

                              },
                              color: Colors.blueAccent,
                              child: Text(
                                _isLoading ? 'Registering..' : "Submit",
                                style: TextStyle(
                                  fontWeight: FontWeight.bold,
                                  color: Colors.white,
                                ),
                              ),
                            ),

                          ),
                        ),

                      ),

                      /*
                                            * For password textfield
                                            */
                    ]),
                ),
              ),
            ),
            SizedBox(
                          height: 300,
                      ),
          ],
        ),
      );
  }

  void _handleSubmit() async { // this method will post data of new user to regis
ter user api
    setState(() {
      _isLoading = true;
    });

    var data = { // maps data of user from textfield to json data
      'name': nameController.text,
      'email': emailController.text,
      'password': passwordController.text,
      'password_confirmation': passwordConfController.text,
```

```
      'roles_id': getrole,
      'phone' : phoneController.text,
      'address' : addressController.text,
      'Achievements' : achievementController.text,
      'Job' : jobController.text,
    };

    print(data);

    var res = await CallApi().postData(data, 'signup'); // calls registration api
 from call api class and sends data

    var body = jsonDecode(res.body); // prints response of created users data
    showDialog(context: context,
                    builder: (BuildContext context){
    return AlertDialog( // checks the form validation and shows messages
          title: Text('Message'),
          content: nameController.text == "" || emailController.text == "" || pas
swordController.text == "" || passwordConfController.text == "" ||
          phoneController.text == "" || addressController.text == "" || achieveme
ntController.text == "" || jobController.text == ""? Text('All fields are require
d to register'): Text(res.body),
          actions: <Widget>[
            Row(
              children: <Widget>[
                FlatButton(onPressed: (){
                  Navigator.pop(context);
                  Navigator.pop(context);
                  Navigator.pop(context);
            }, child: Text('Back to login')),
            FlatButton(onPressed: (){
                  Navigator.pop(context);
            }, child: Text('ok'))

              ],
            )

          ],
        );
                    },
    );
    print(body);


    setState(() {
```

```
      _isLoading = false;
    });
  }
}
```

Home page:

```
import 'dart:async';
import 'package:alumniapp/alumnilist.dart';
import 'package:alumniapp/createEvent.dart';
import 'package:alumniapp/eventdetail.dart';
import 'package:alumniapp/loginpage.dart';
import 'package:alumniapp/notificationPage.dart';
import 'package:alumniapp/profile.dart';
import 'package:carousel_slider/carousel_slider.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'package:alumniapp/logout.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:url_launcher/url_launcher.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:flutter/cupertino.dart';

class Homepage extends StatelessWidget {
  String accessToken; //get acess token from login page
  Homepage(this.accessToken);
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Home page',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: HomePage(this.accessToken),
    );
  }
}

class HomePage extends StatefulWidget {
  String accessToken;
  HomePage(this.accessToken);
  @override
  _HomePageState createState() => _HomePageState(this.accessToken);
}
```

```
class _HomePageState extends State<HomePage> {
  TextEditingController searcheventController = TextEditingController(); // contr
oller for event search box


  bool _isloading = true;

  Timer _debounce;
  int _current = 0;
  List imgList = [
    'images/christmas.jpg',
    'images/carnival.jpg',
    'images/aspire.jpg',
    'images/britan.jpg',
    'images/sports.jpg',
    'images/graduation.jpg',
    'images/lhosar.png',
    'images/holi.jpg',
    'images/teej.jpg'
  ];

  ScrollController _scrollController = new ScrollController(); // scroller for ev
ents page

  int page = 0;

  SharedPreferences sharedPreferences;


  final String urn = 'http://192.168.0.114:8000/api/search'; // api for searching
 events
  final String uri = 'http://192.168.0.114:8000/api/events?page='; // api for dis
playing events with page number
  List userdata; // maps json data to list
  bool _isLoading = false;

  List data;
  Map paging; // maps paging from api
  String accessToken;
  _HomePageState(this.accessToken);

  Future getJsonData() async { // this function will get json data of events and
prints.
    var response = await http
```

```
        .get(Uri.encodeFull(uri+ page.toString()), headers: {"Accept": "applicati
on/json"}); // here header is given as parameter
    print(response.body);

    setState(() {
      _isLoading = false;
      var convertDataToJson = json.decode(response.body);
      data = convertDataToJson['data']; // this will maps data format of api and
decode json
    });
    return "Success";
  }

  Future getnext(int page) async {  // this function will get more event data fro
m next page while user scrolls
    var response = await http.get(Uri.encodeFull(uri + page.toString()),
        headers: {"Accept": "application/json"});
    print(response.body);

    setState(() {
      var convertDataToJson = json.decode(response.body);
      data = convertDataToJson['data'];
      paging = convertDataToJson['links'];
      print(paging);

    });
    return "Success";
  }

  Future<String> searchData() async { // this function handles search api and fet
ch data according to search reqst
    var response = await http.get(Uri.encodeFull(urn), headers: {
      "Accept": "application/json",
      "search": searcheventController.text
    });
    print(response.body);

    var datas = jsonDecode(response.body);

    setState(() {
      data = datas;
    });
    return "Success";
  }
```

```
  @override
  void initState() {
    super.initState();
    checkLoginStatus();
    this.getJsonData();
    _scrollController.addListener(() { // this will checks the scroll of page
      if (_scrollController.position.pixels ==
          _scrollController.position.maxScrollExtent) {
        page = page + 1;
        getnext(page);

      if (paging['next'] == null) {
                page = page -1;
                getnext(page);
                print('last page');   // will check last page and stops scrolling

                    }


                    }

                  });
                }
                checkLoginStatus()async{ // checks login status and allow user
to login again if logged out
                  sharedPreferences = await SharedPreferences.getInstance();
                 if  (sharedPreferences.getString("token") == null){
                     Navigator.of(context).pushAndRemoveUntil(
                  MaterialPageRoute(
                      builder: (BuildContext context) => LoginPage()),
                  (Route<dynamic> route) => false);
                 }

                }

                @override
                void dispose() {
                  _scrollController.dispose();
                  super.dispose();
                }
```

```
                refresh() { // will refresh event page after user pull the indi
cator
                  if (_scrollController.position.pixels ==
                      _scrollController.position.maxScrollExtent) {
                    getnext(page);
                  }
                  getJsonData();
                }

                List<T> map<T>(List list, Function handler) {
                  List<T> result = [];
                  for (var i = 0; i < list.length; i++) {
                    result.add(handler(i, list[i]));
                  }
                  return result;
                }

                @override
                Widget build(BuildContext context) { // this widget contains al
l ui properties of home page
                  return Scaffold(
                    resizeToAvoidBottomPadding: false,
                    backgroundColor: Colors.blue[900],
                    appBar: AppBar(
                        elevation: 0,
                        backgroundColor: Colors.blue[900],
                        title: Text('Events'),
                        actions: <Widget>[
                          Padding(
                            padding: const EdgeInsets.all(8.0),
                            child: IconButton(
                              icon: Icon(Icons.refresh),
                              iconSize: 25,
                              onPressed: () { // this will refresh the home pag
e after having some changes
                                Navigator.push(
                                    context,
                                    MaterialPageRoute(
                                      builder: (context) => HomePage(accessToke
n),

                                    ));
                              },
                            ),
                          ),
                          IconButton(
```

```
                                onPressed: () {
                                  Navigator.push(
                                      context,
                                      MaterialPageRoute(
                                        builder: (context) => NotificationPage(),
                                      ));
                                },
                                icon: Icon(Icons.notifications_active)),
                            SizedBox(
                              width: 20,
                            )
                          ]),
                      drawer: Drawer( // this will show navigation drawer of the
application
                        child: ListView(children: <Widget>[
                          UserAccountsDrawerHeader(
                            accountName: Text("Hello!"),
                            accountEmail: Text('Welcome to the Application'),
                            decoration: BoxDecoration(
                                image: DecorationImage(
                              fit: BoxFit.fill,
                              image: AssetImage('images/cover.jpg'),
                            )),
                          ),
                          ListTile( // will display profile page
                              title: Text('View Profile'),
                              leading: Icon(Icons.account_circle),
                              onTap: () {
                                Navigator.push(
                                    context,
                                    MaterialPageRoute(
                                      builder: (context) => MyProfile(accessToken
),
                                    ));
                              }),
                          ListTile( // will display alumnilist page
                            title: Text('Search Alumni'),
                            leading: Icon(Icons.search),
                            onTap: () {
                              Navigator.push(
                                  context,
                                  MaterialPageRoute(
                                    builder: (context) => AlumniList(accessToken)
,
                                  ));
```

```
                            },
                          ),
                          ListTile( // will display islington college website
                            title: Text('Visit College'),
                            leading: Icon(Icons.school),
                            onTap: () {
                              launch('https://islington.edu.np/');
                            },
                          ),
                          ListTile(
                            title: Text('Report Us'),
                            leading: Icon(Icons.report),
                          ),
                          ListTile(
                            title: Text('Rate us'),
                            leading: Icon(Icons.rate_review),
                          ),
                          ListTile( // will logout user from the system
                            title: Text('Logout'),
                            leading: Icon(Icons.exit_to_app),
                            onTap: () {
                              showDialog( // confirms user for logging out
                                          context: context,
                                          builder: (BuildContext context) {
                                            return AlertDialog(
                                              title: Text('Are you sure you want
to logout?',style: TextStyle(fontWeight:FontWeight.bold),),
                                              content: Text('Logout from applicat
ion?'),

                                              actions:<Widget>[
                                                Row(
                                                  children: <Widget>[
                                                    RaisedButton(onPressed: (){
                                                        Navigator.push(
                                    context,
                                    MaterialPageRoute(
                                      builder: (context) => Logout(accessToken, con
text),

                                    ));
                                                    }, child: Text('Yes'),
                                                    ),
                                                    SizedBox(
                                                      width: 20,
                                                    ),
                                                    RaisedButton(onPressed: (){
```

```
                                                Navigator.pop(context);
                                              },child: Text('no'),),

                                            ],
                                          )

                                      ]
                                    );

                                },
                            );
                          },
                        ),
                      ]),
                    ),
                    body: Column(
                      children: <Widget>[
                        SizedBox(height: 10),
                        Container( // UI for search box in top
                          margin:
                              EdgeInsets.only(left: 20.0, right: 20.0, bottom:
8.0, top: 8.0),
                          decoration: BoxDecoration(
                            color: Colors.white,
                            boxShadow: <BoxShadow>[
                              BoxShadow(
                                offset: Offset(
                                  1,
                                  2,
                                ),
                              ),
                            ],
                            borderRadius: BorderRadius.circular(20),
                          ),
                          child: Stack(
                            children: <Widget>[
                              TextField(
                                onChanged: (String text) {
                                  if (_debounce?.isActive ?? false) _debounce.c
ancel();

                                  _debounce = Timer(const Duration(milliseconds
: 1000), () {

                                    searchData(); // calls searchData function
and displays data in list

                                  });
```

```
                                },
                                controller: searcheventController,
                                decoration: InputDecoration(
                                  hintText: 'Search events....',
                                  contentPadding: EdgeInsets.only(left: 50.0),
                                  border: InputBorder.none,
                                ),
                              ),
                              Padding(
                                padding: const EdgeInsets.only(left: 320),
                                child: IconButton(
                                  icon: Icon(
                                    Icons.search,
                                    size: 25,
                                  ),
                                  color: Colors.black,
                                  onPressed: () {
                                    searchData();
                                  },
                                ),
                              ),
                            ],
                          ),
                        ),

                        Expanded(
                          child: RefreshIndicator(
                            onRefresh: getJsonData,
                            child:  data == null ?
                                   Center(
                                    child: CircularProgressIndicator(),
                                   )

                                :ListView.builder( // will returns list view if
 data is not empty
                              controller:  _scrollController,
                              itemCount: data.length+1,
                              itemBuilder: (BuildContext context, int index) {

                              if (index == data.length ) { // shows activity in
dicator while scrolling
                                return Center(
                                  child: CupertinoActivityIndicator
                                  ()
                                );
```

```
                                }




                                return Container(
                                  color: Colors.black,
                                  child: new Center(
                                    child: new Column(
                                      crossAxisAlignment: CrossAxisAlignment.st
retch,

                                      children: <Widget>[
                                        Card(
                                          child: Padding(
                                            padding: EdgeInsets.only(left: 2, r
ight: 4),

                                            child: ListTile( // calls api metho
d and shows the return data in list view

                                              onTap: () {
                                                Navigator.push(
                                                    context,
                                                    MaterialPageRoute(
                                                      builder: (context) => Eve
ntdetail(

                                                          data[index]['event_name
'], // fetching json data and decoding it to show in List view UI
                                                          data[index]['event_date
'],

                                                          data[index]['event_time
'],

                                                          data[index]['event_venu
e'],

                                                      ),
                                                    ));
                                              },
                                              leading: Icon(
                                                Icons.event_available,
                                                size: 50,
                                                color: Colors.blue[900],
                                              ),
                                              title: Text(
                                                data[index]['event_name'],
                                                style: TextStyle(
                                                  fontWeight: FontWeight.bold,
```

```
                                                color: Colors.black,
                                                fontSize: 18,
                                              ),
                                            ),
                                            subtitle: Row(
                                              children: <Widget>[
                                                Text(
                                                  data[index]['event_venue'],
                                                  style: TextStyle(
                                                      color: Colors.white, fo
ntSize: 1),
                                                ),
                                                SizedBox(
                                                  width: 5,
                                                ),
                                                Text(
                                                  data[index]['event_time'],
                                                  style: TextStyle(
                                                      color: Colors.white, fo
ntSize: 1),
                                                ),
                                              ],
                                            ),
                                            trailing: Column(
                                              children: <Widget>[
                                                Icon(
                                                  Icons.date_range,
                                                  color: Colors.black,
                                                ),
                                                SizedBox(height: 5),
                                                Text(data[index]['event_date'
]),
                                              ],
                                            ),
                                          ),
                                        ),
                                      ),
                                    ),
                                  ],
                                ),
                              ),
                            );
                          },
                        )

                    ),
```

```
                        ),
                        SizedBox(
                          height: 10.0,
                        ),

                      ],
                    ),
                  );
                }

              }
```

8.3.2. Back end Api creation Laravel Code:

Auth Controller (Api function codes)

```
<?php
namespace App\Http\Controllers;
use Illuminate\Support\Facades\Input;
use Illuminate\Http\Request;
use App\Http\Requests\AuthRequest;
use Illuminate\Support\Facades\Auth;
use Carbon\Carbon;
use App\User;
class AuthController extends Controller // this controller is self created contro
ller for writing api function of users
{
    /**
     * Create user
     *
     * @param  [string] name
     * @param  [string] email
     * @param  [string] password
     * @param  [string] password_confirmation
     * @return [string] message
     */
    public function signup(Request $request) // this function creates new user to
 the system
    {
```

```
        $request->validate([ // checks form validity while creating new user
            'name' => 'required|string',
            'email' => 'required|string|email|unique:users',
            'password' => 'required|string|confirmed',
            'roles_id' => 'required|integer',
            'phone' => 'required|string',
            'address' => 'required|string',
            'Achievements' => 'required|string',
            'Job' => 'required|string',
        ]);
        $user = new User([ // crate new user and save in database.
            'name' => $request->name,
            'email' => $request->email,
            'password' => bcrypt($request->password),
            'roles_id' => $request->roles_id,
            'phone' => $request->phone,
            'address' => $request->address,
            'Achievements' => $request->Achievements,
            'Job' => $request->Job,
        ]);
        $user->save();
        return response()->json( // shows sucess message after creating user
            'Successfully created user!'
        , 201);
    }

    /**
     * Login user and create token
     *
     * @param  [string] email
     * @param  [string] password
     * @param  [boolean] remember_me
     * @return [string] access_token
     * @return [string] token_type
     * @return [string] expires_at
     */
    public function login(AuthRequest $request) // this function will log in user
 to the system
    {
        $credentials = request(['email', 'password']); // asks for email and pass
word and check validity
        if(!Auth::attempt($credentials))
            return response()->json(
                'Invalid Username or Password'
            , 401);
```

```
        $user = $request->user();
        $tokenResult = $user-
>createToken('Personal Access Token'); // creates token after user logs in
        $token = $tokenResult->token;

        if ($request->remember_me) // checks remeber me option
            $token->expires_at = Carbon::now()->addWeeks(1);
            $user = User::where('email', '=', $request->email)->firstOrFail();
            $user->remember_token = $tokenResult->accessToken;
            $user->save();
        return response()->json([
            'access_token' => $tokenResult-
>accessToken, // return response as token after user logs in
            'token_type' => 'Bearer',
            'expires_at' => Carbon::parse(
                $tokenResult->token->expires_at
            )->toDateTimeString()
        ]);


    }

    /**
     * Logout user (Revoke the token)
     *
     * @return [string] message
     */
    public function logout(Request $request) // this function log out user from t
he system and deletes token
    {
        $token = $request->user()->token();
        $token->revoke();

        $response = 'You have been succesfully logged out!';
        return response($response, 200);

    }

    /**
     * Get the authenticated User
     *
     * @return [json] user object
     */
```

```
    public function user(Request $request){ // this function will show details of
 logged in users

        $user = $request->user();
        $user->load('role');
        return response()->json($user);
    }

    public function users(Request $request ){ // this function shows all register
ed user in app

        $search = $request->header('search');
        $search = $search ."%";
        if($search){
            return User::where("name","like",$search)->with('role')->get();
        }
        return User::with('role')->get();
    }




    public function getUserById($id){ // this functon shows user by id

        $user = User::find($id);
        return response()->json($user);

        }

    public function updatebyid(Request $request, $id) // this function is created
 to update the data of user
        {

            $user = User::find($id);
            $user->name = $request->input('name');
            $user->email = $request->input('email');
            $user->password = bcrypt($request->password);
            $user->roles_id = $request->input('roles_id');
            $user->phone = $request->input('phone');
            $user->address = $request->input('address');
            $user->Achievements = $request->input('Achievements');
            $user->Job = $request->input('Job');


            $user->save(); // saves updated data to database
```

```
            return response()->json($user); // returns updated data



        }




}
```

Api route coding:

```
<?php

use Illuminate\Http\Request;
use App\User;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/
// These are the api routes for the users table

Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});

Route::group([
    'prefix' => 'auth'
], function () {
    Route::post('login', 'AuthController@login'); // api route for login user
    Route::post('signup', 'AuthController@signup'); // api route for register
```

```
    Route::post('Api/password/email', 'Api\ForgotPasswordController@sendResetLink
Email'); // api route for forgot passsword
    Route::post('Api/password/reset', 'Api\ResetPasswordController@reset');

    Route::group([
      'middleware' => 'auth:api'
    ], function() {
        Route::get('logout', 'AuthController@logout'); // api route for logout us
er
        Route::get('user', 'AuthController@user'); // api route for display logge
d in user

        Route::get('users', 'AuthController@users'); // api route for displaying
all user
        Route::get('user/{id}', 'AuthController@getUserById'); // api route for d
isplaying 1 user by id

        Route::put('userupdate/{id}', 'AuthController@updatebyid'); // api route
for update user data


    });
});

// this is the api routes for events table:

Route::get('events', 'EventController@index'); // api route for showing events

Route::get('allevents', 'EventController@allevents');

Route::get('search', 'EventController@search'); // api route for searching events

Route::get('event/{id}', 'EventController@show'); // api route for showing events
 by id

Route::post('event', 'EventController@store'); // api route for create event

Route::put('event', 'EventController@store'); // api route for update events

Route::delete('event/{id}', 'EventController@destroy'); // api route for delete e
vents
```

Event controller (Contains function to create event api):

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Http\Requests;
use App\Event;
use App\Http\Resources\Event as EventResource;

class EventController extends Controller // this is self created controller which
 have functionality of CRUD events
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request) // displays events by pagination to 1
0
    {
        $events = Event::paginate(10);

        return EventResource::collection($events);

        $search = $request->header('search');
        $search = $search ."%";
        if($search){
            return Event::where("event_name","like",$search)->get();
        }




    }
    public function allevents(Request $request) // displays events by pagination
to 100
    {
        $events = Event::paginate(100)->sortBy("event_date");

        return EventResource::collection($events);
```

```
    }
    public function search(Request $request)  // search events by event name take
 event name as request
    {

        $search = $request->header('search');
        $search = $search ."%";
        if($search){
            return Event::where("event_name","like",$search)-
>get(); // using where query to search events
        }







    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request) // this function creates new events a
nd updates old events
    {
        $event = $request-
>isMethod('put') ? Event::findOrFail // checks if requuest is update or create
        ($request->event_id): new Event;

        $event->id = $request->input('event_id');
        $event->event_name = $request->input('event_name');
        $event->event_time = $request->input('event_time');
        $event->event_date = $request->input('event_date');
```

```
        $event->event_venue = $request->input('event_venue');

        if($event->save()){
            return new EventResource($event); // saves event in database afte cre
ating or updating it

        }
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id) /// shows events by id
    {
        $event = Event::findOrFail($id);

        return new EventResource($event);
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        //
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        //
    }
```

```
    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id) // deletes events by id
    {
        $event = Event::findOrFail($id);
        if($event->delete()){
        return new EventResource($event);
        }
    }
}
```

8.4: Appendix D: Designs:

8.4.1: Gantt Chart:

ID	Task Mode	Task Name	Duration	Start	Finish
1		**Inception**	**55 days**	**Tue 8/20/19**	**Sat 11/2/19**
2		Determine features of project	10 days	Sat 8/24/19	Thu 9/5/19
3		initiating milestone	0 days		
4		Requirement Gathering	20 days	Sat 8/24/19	Thu 9/19/19
5		Resource management	17 days	Thu 10/10/19	Fri 11/1/19
6		Risk management	52 days	Sun 8/25/19	Sat 11/2/19
7		Inception phage completed	55 days	Tue 8/20/19	Sat 11/2/19
8		**Elaboration**	**6.15 mons**	**Sun 9/22/19**	**Tue 3/10/2**
9		executing milestone	0 days		
10		Design architectural prototype	15 days	Sun 9/22/19	Thu 10/10/19
11		Design use case model	7 days	Sun 11/24/19	Sun 12/1/19
12		Revised risk management and requirement collection	81 days	Sun 9/22/19	Fri 1/10/20
13		Elaboration phase completed	10 days	Sun 9/22/19	Thu 10/3/19
14		**Construction**	**106 days**	**Sun 12/1/19**	**Fri 4/24/20**
15		Development milestone	0 days		
16		Develop UI for the application	60 days	Sun 12/1/19	Thu 2/20/20
17		Develop application features	104 days	Sun 12/1/19	Wed 4/22/20
18		Testing all features	106 days	Sun 12/1/19	Fri 4/24/20
19		Construction phase completed	106 days	Sun 12/1/19	Fri 4/24/20
20		**Transition**	**31 days**	**Sat 4/25/20**	**Fri 6/5/20**
21		Feedback	31 days	Mon 4/20/20	Mon 6/1/20
22		controlling milestone	0 days		
23		Correction	40 days	Fri 4/10/20	Thu 6/4/20
24		Submit project	1 day	Fri 6/5/20	Fri 6/5/20

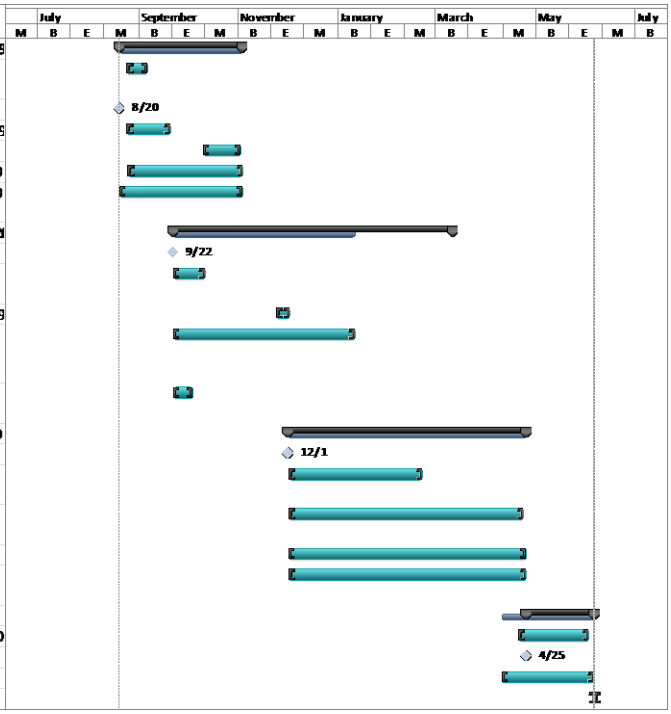


Figure 133 Gantt chart.

8.4.2: Work break down structure:

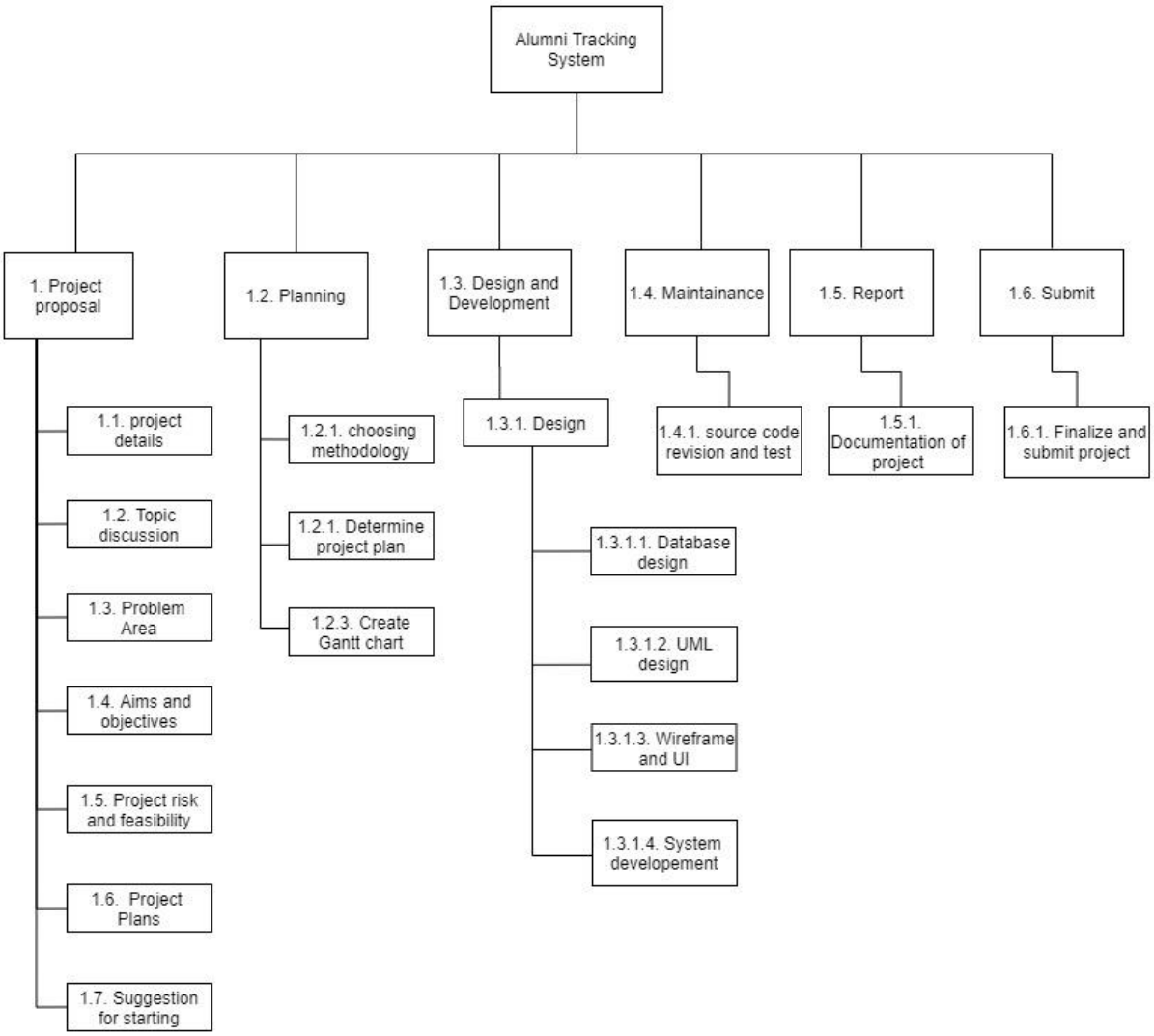


Figure 134 Work break down structure.

8.4.3: Data Flow Diagrams (DFD):

8.4.3.1: DFD level 0:

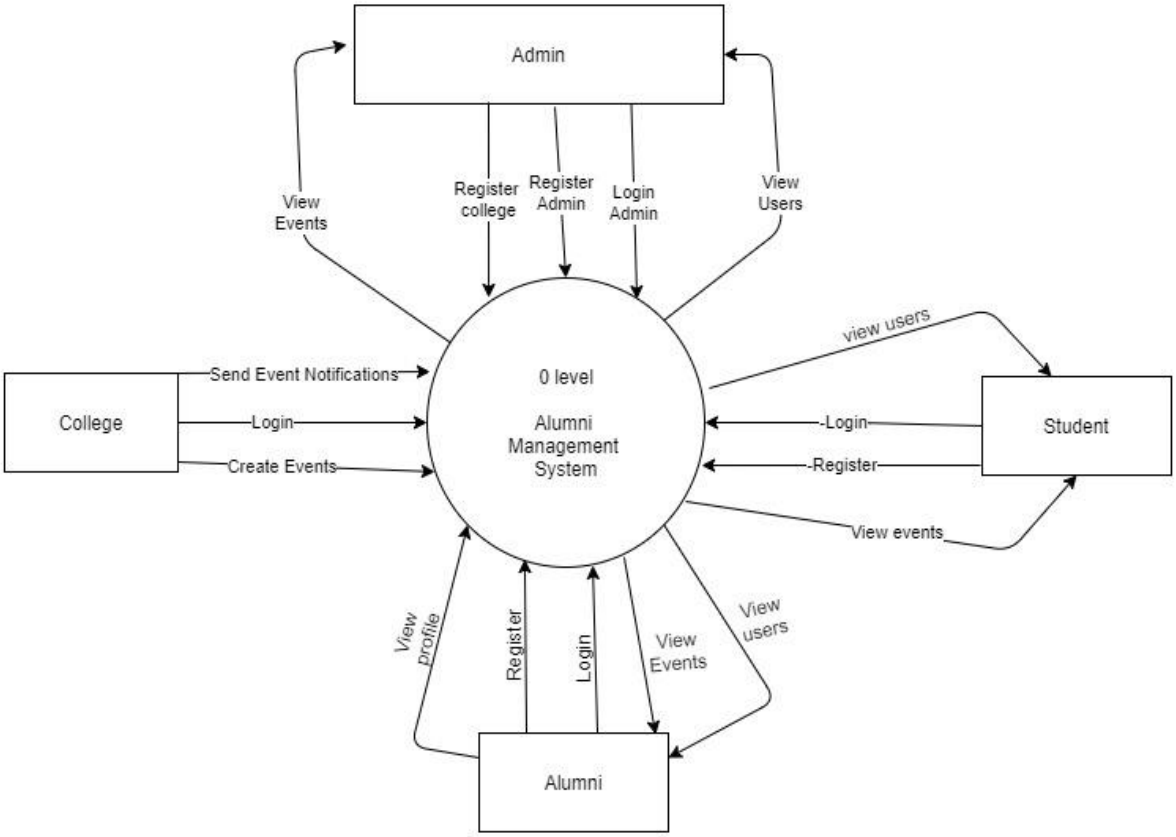


Figure 135 Dfd level 0 appendix.

8.4.3.2: DFD level 1:

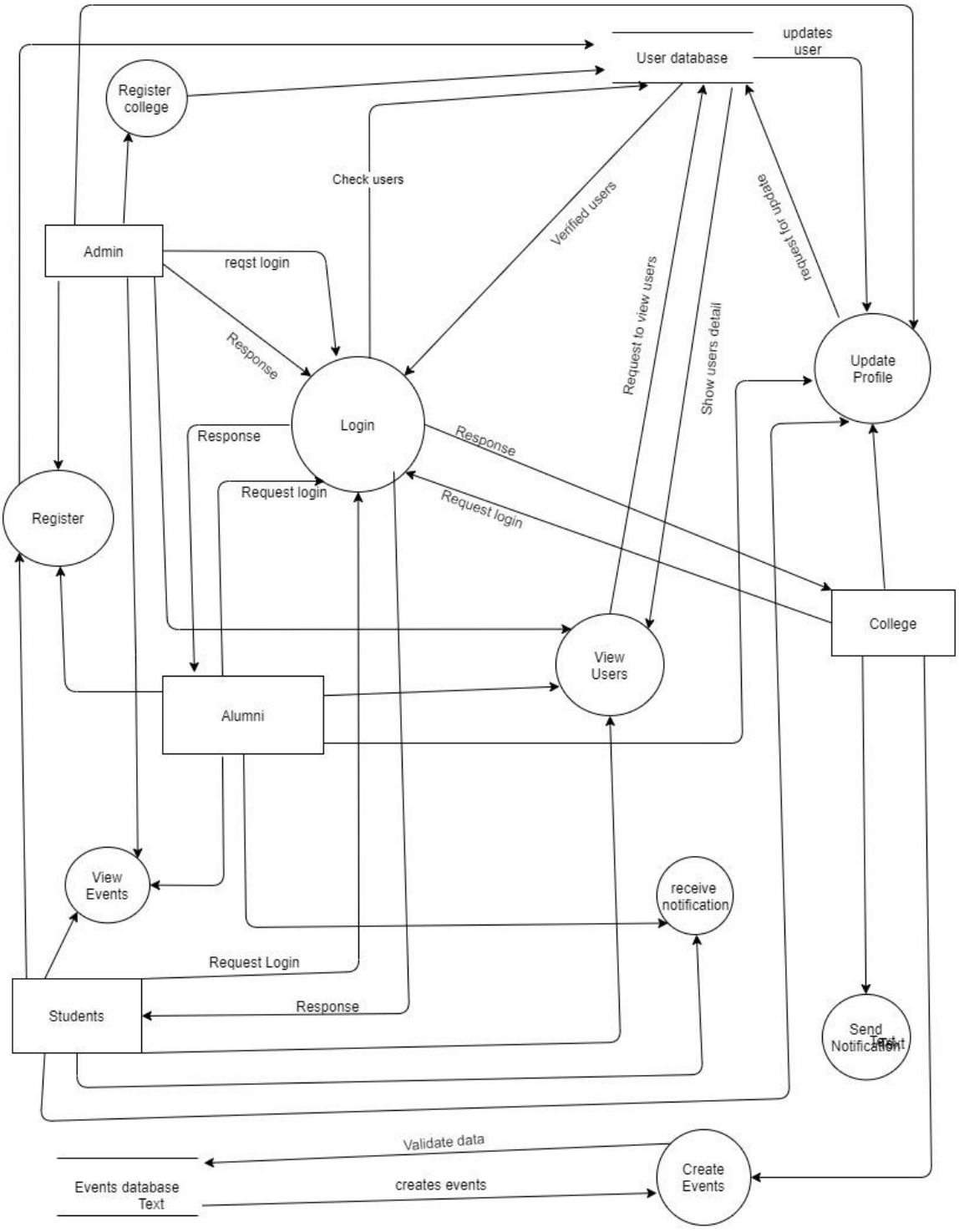


Figure 136 Dfd level 1 appendix.

8.4.4: Use case diagram:

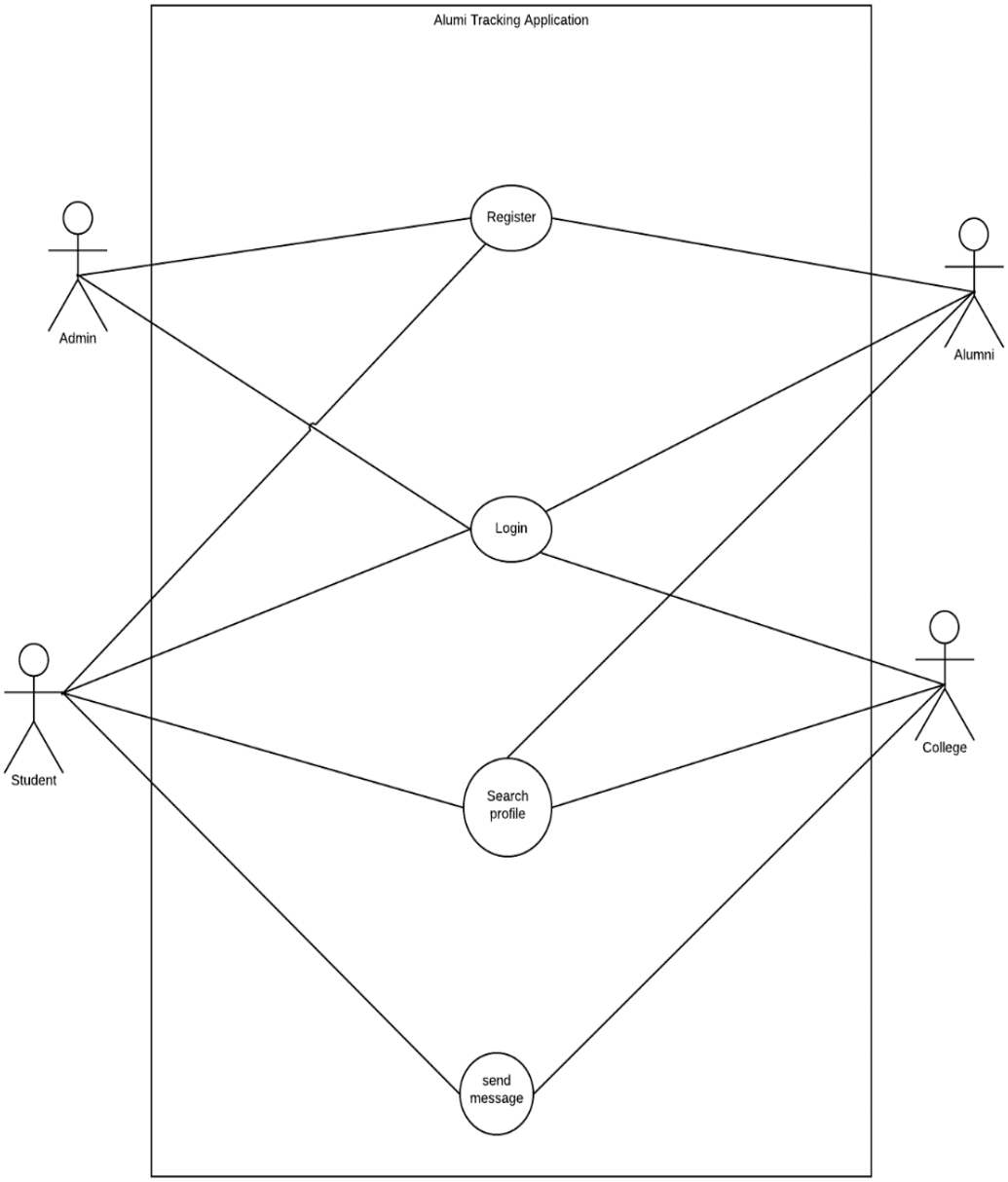


Figure 137 Use case diagram appendix.

8.4.5: Wireframe:
Wireframes and Prototypes:

login wireframe:

this is the wireframe of login page where two text boxes are placed for username and password of the user. There is also a button for login. Two links are attached in it for forgot password and signup.

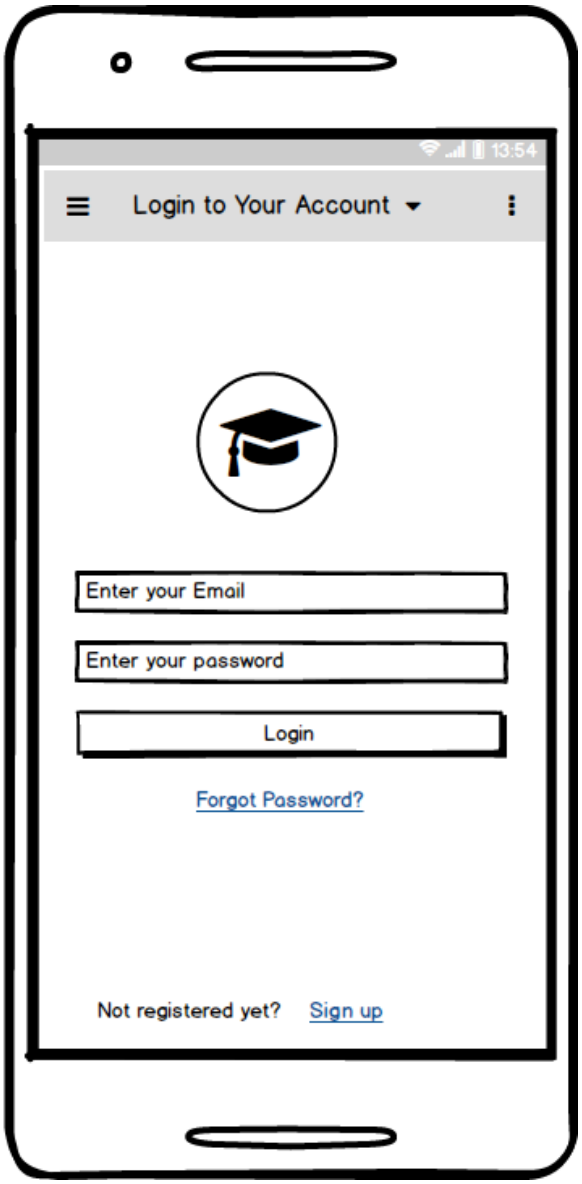


Figure 138 Login page wireframe.

Registration wireframe

This is the wireframe of registration page of the application. This includes all the user's details some of the field are required where some are optional. There is also a button for register now when the users can submit their details.

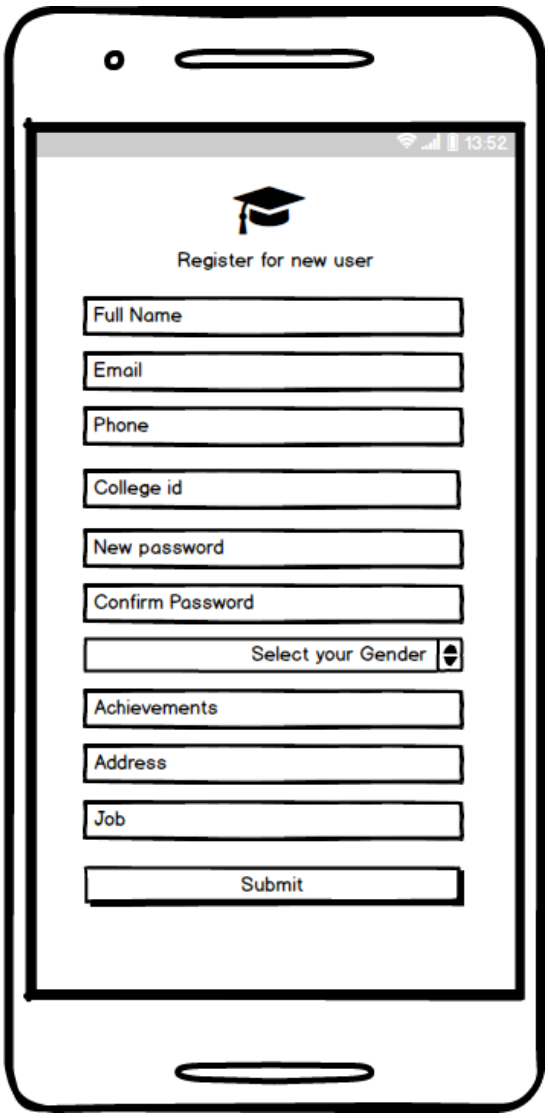


Figure 139 Registration page wireframe.

Profile wireframe:

This is the wireframe of profile page. Here the profile will be shown according to the user's detail. User can either be alumni student or college. The below boxes are for adding achievements. The first-round shaped field is for image user can put image of theirs in it.

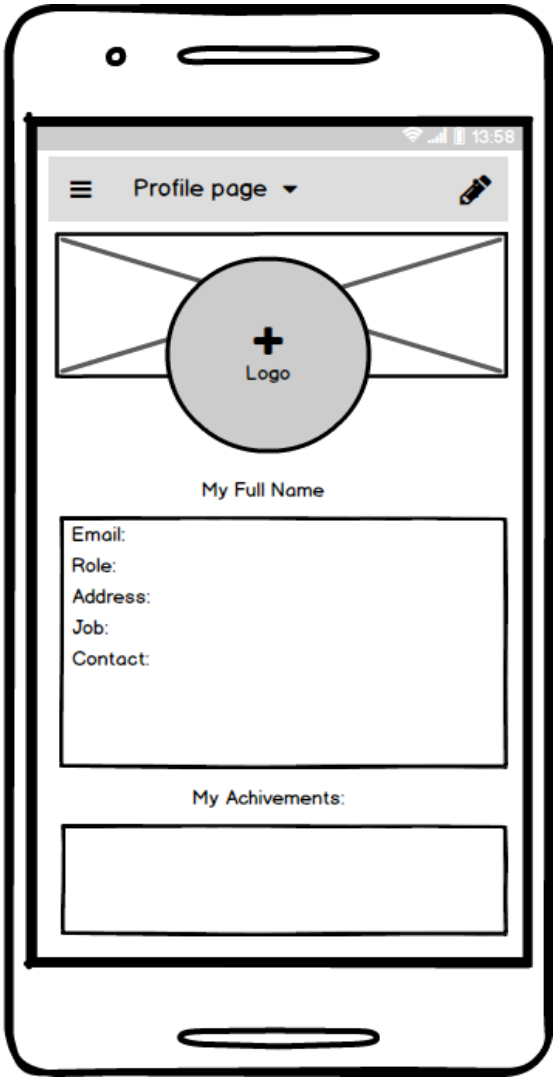


Figure 140 Profile wireframe.

Home page:

This is the wireframe of homepage of all the users. This page contains list of all the events created by college. User will also be able to search the events by the name. Newly added events will be automatically updated here.

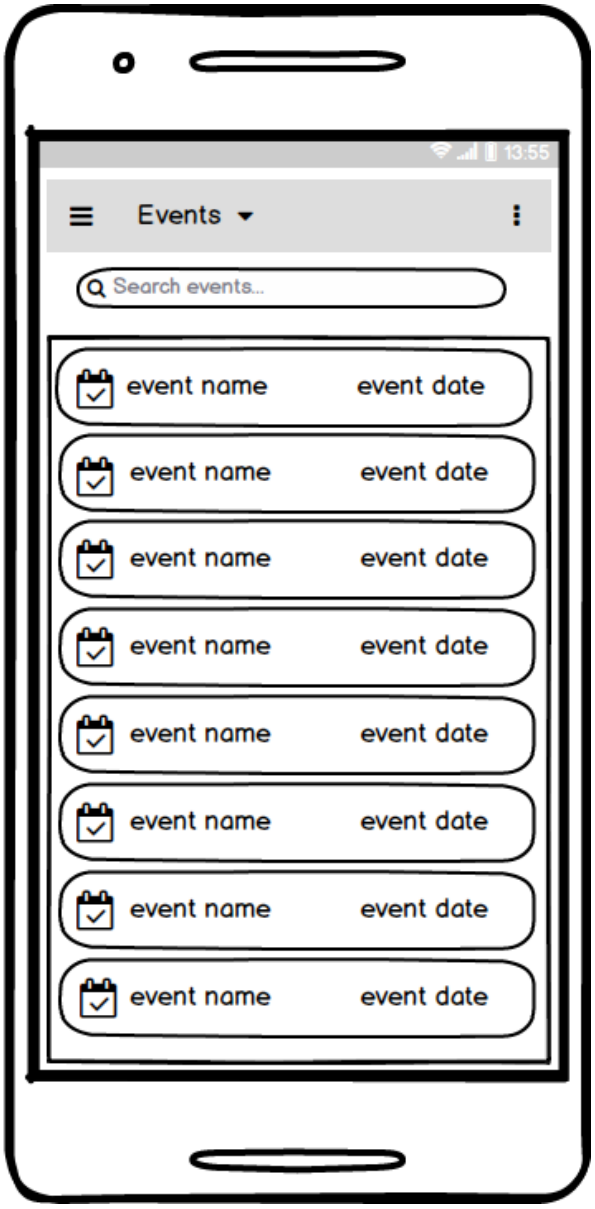


Figure 141 Homepage wireframe.

Event details page:

This page will be open once the user clicks on any events from the homepage. This contains the detail information such as event name, date, venue and time of that events.

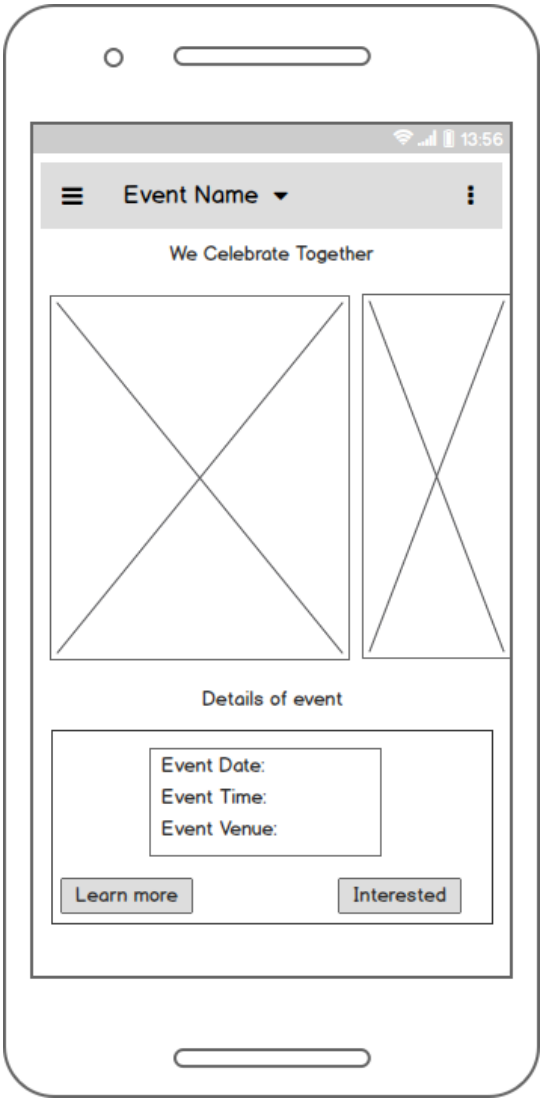


Figure 142 event detail wireframe mobile.

Navigation page:

This page is a part of homepage from where user will be able to navigate to different pages of the application such as profile page, search users page, visit college website, logout and others.

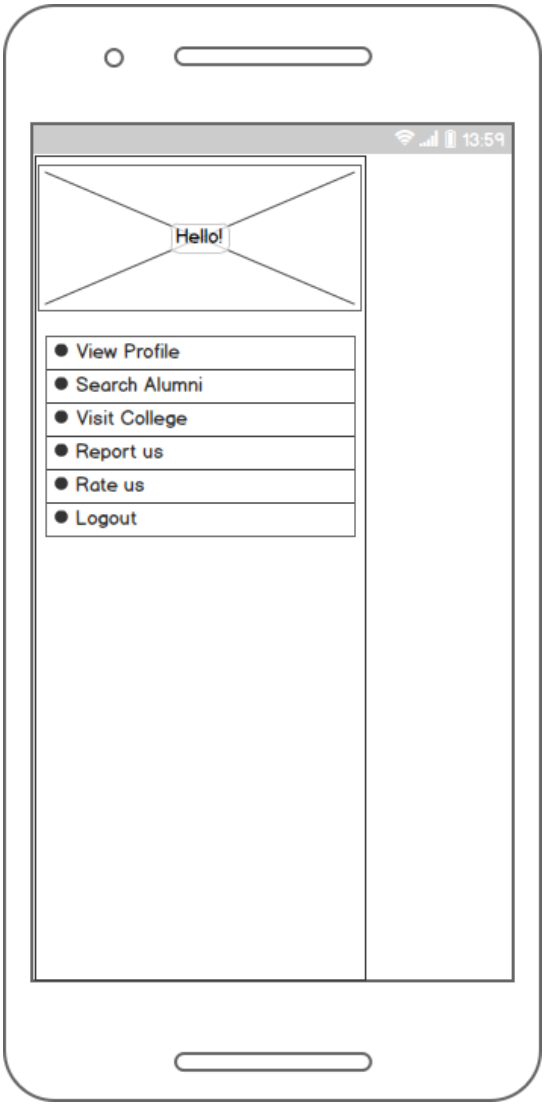


Figure 143 Navigation page wireframe mobile.

My profile page:

This page contains the information of the users which is logged in this application. From this page can see their own details edit their information and add achievements.

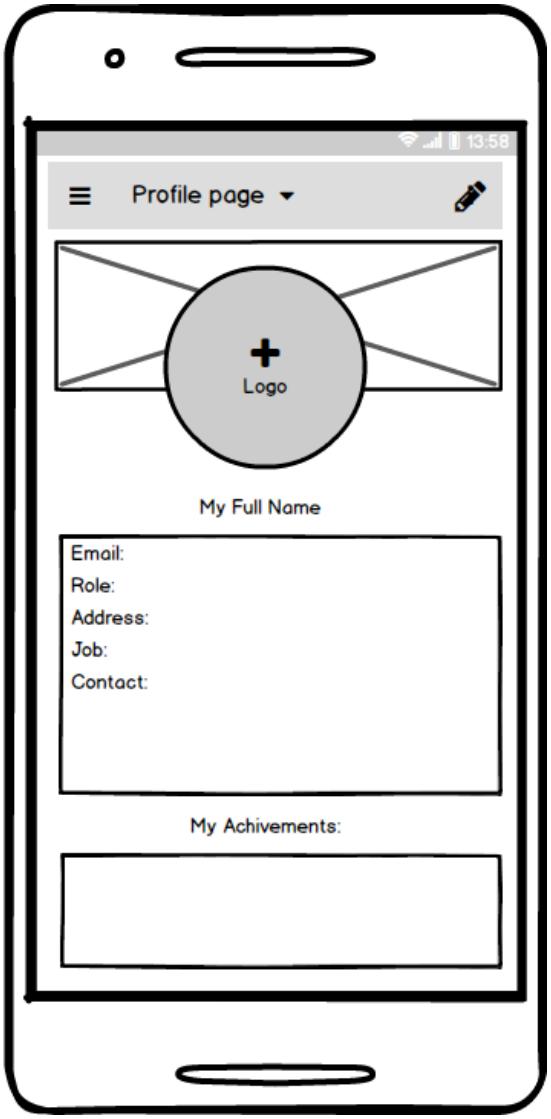


Figure 144 My profile mobile wireframe.

Search users:

This page contains the information of all the users i.e. alumni, students, college and admin who has registered in this application as a user. User should enter name of the user whom they want to search and will get the information of that users in details.

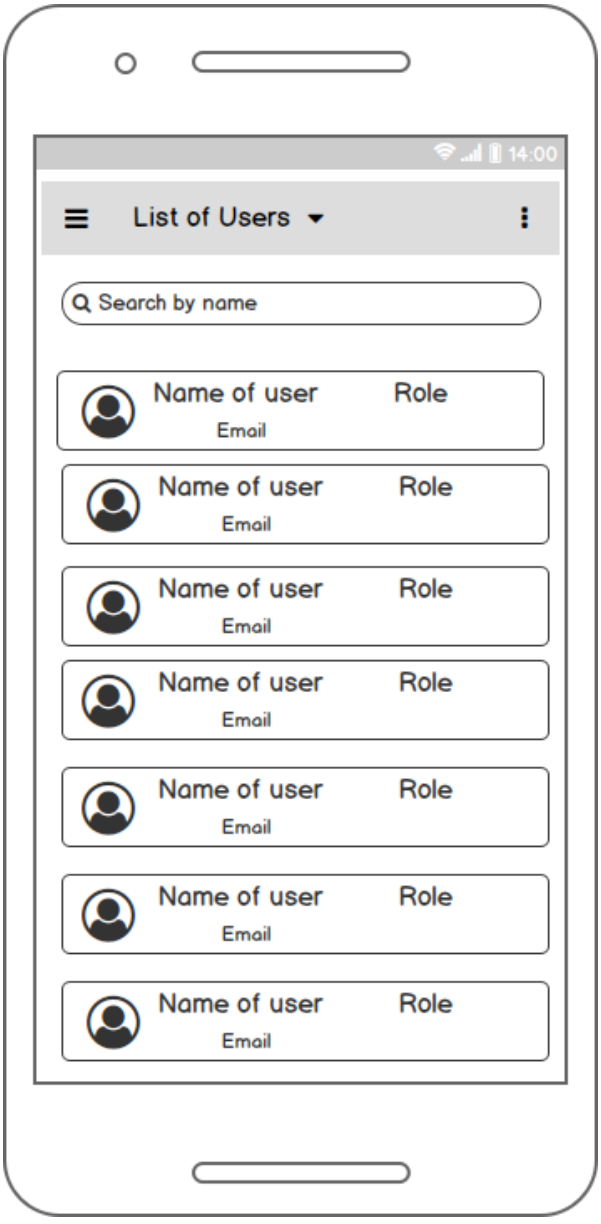


Figure 145 search Users wireframe.

User profile:

After user selects a name of user by search them, they click on it and they will be navigated to that user's profile page which contains detail information of that users. User can view their information, achievements and can mail and call the user directly from this application.

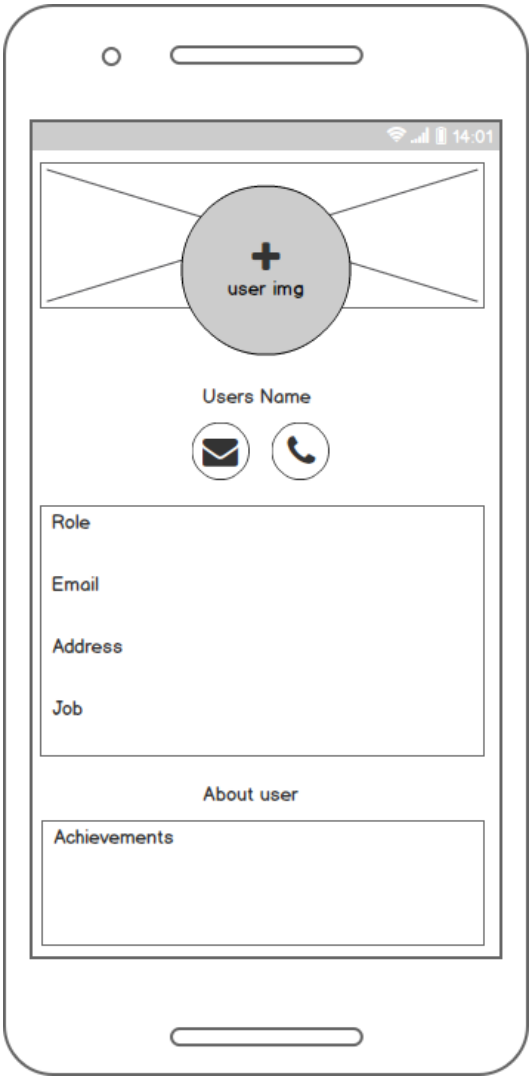


Figure 146 user profile mobile wireframe.

create event page:

From this page college will be able to create new events and send notification to all the users from the application.

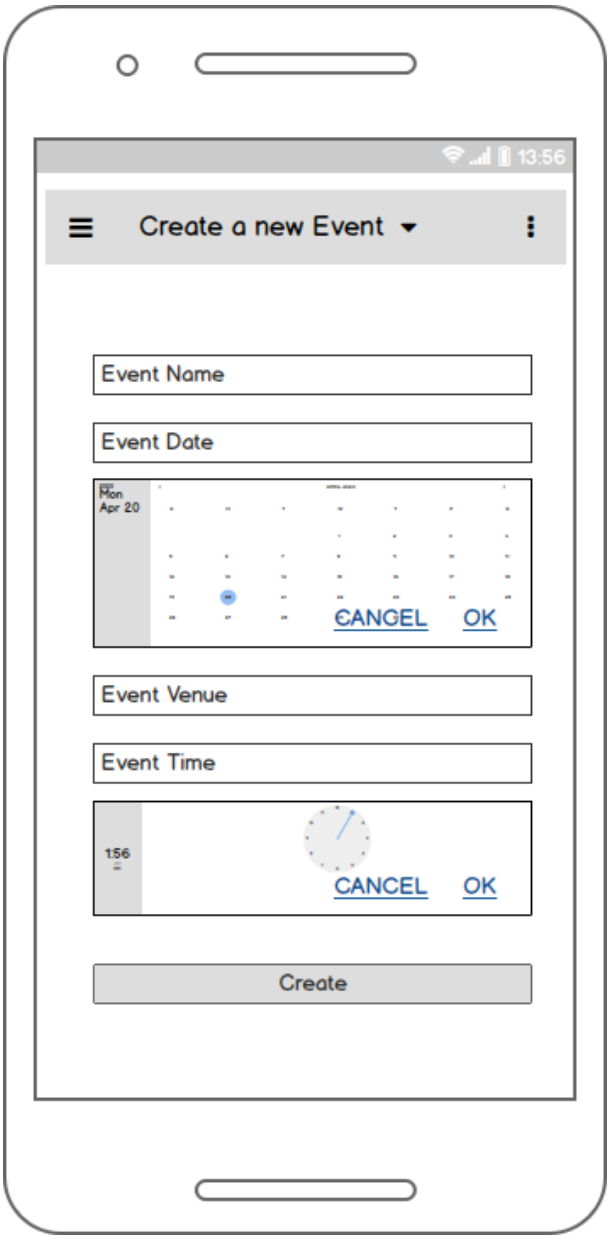


Figure 147 create events wireframe.

Admin web login wireframe:

This page is admin web login page. From this page admin will be logged in to the web control of the alumni tracking system website.

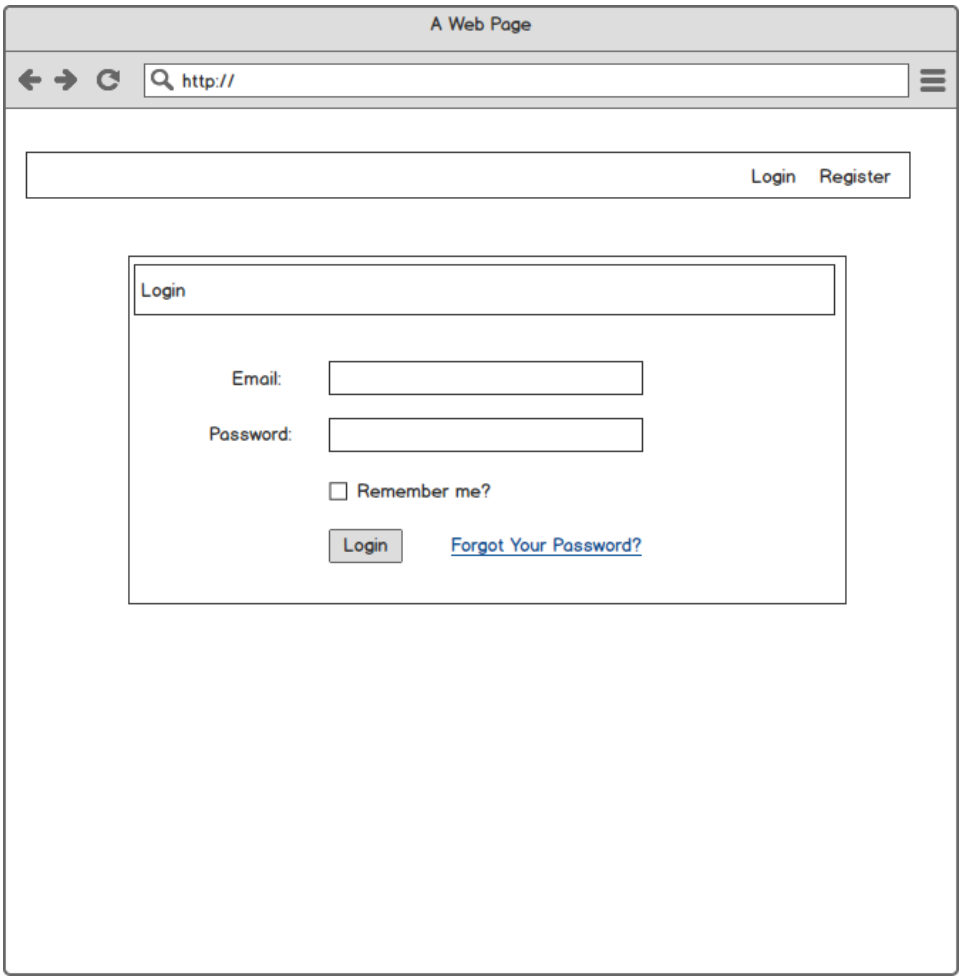


Figure 148 Login page Admin web wireframe.

Admin web Register wireframe:

From this page admin will be able to register new college. Create more admins and users from web.

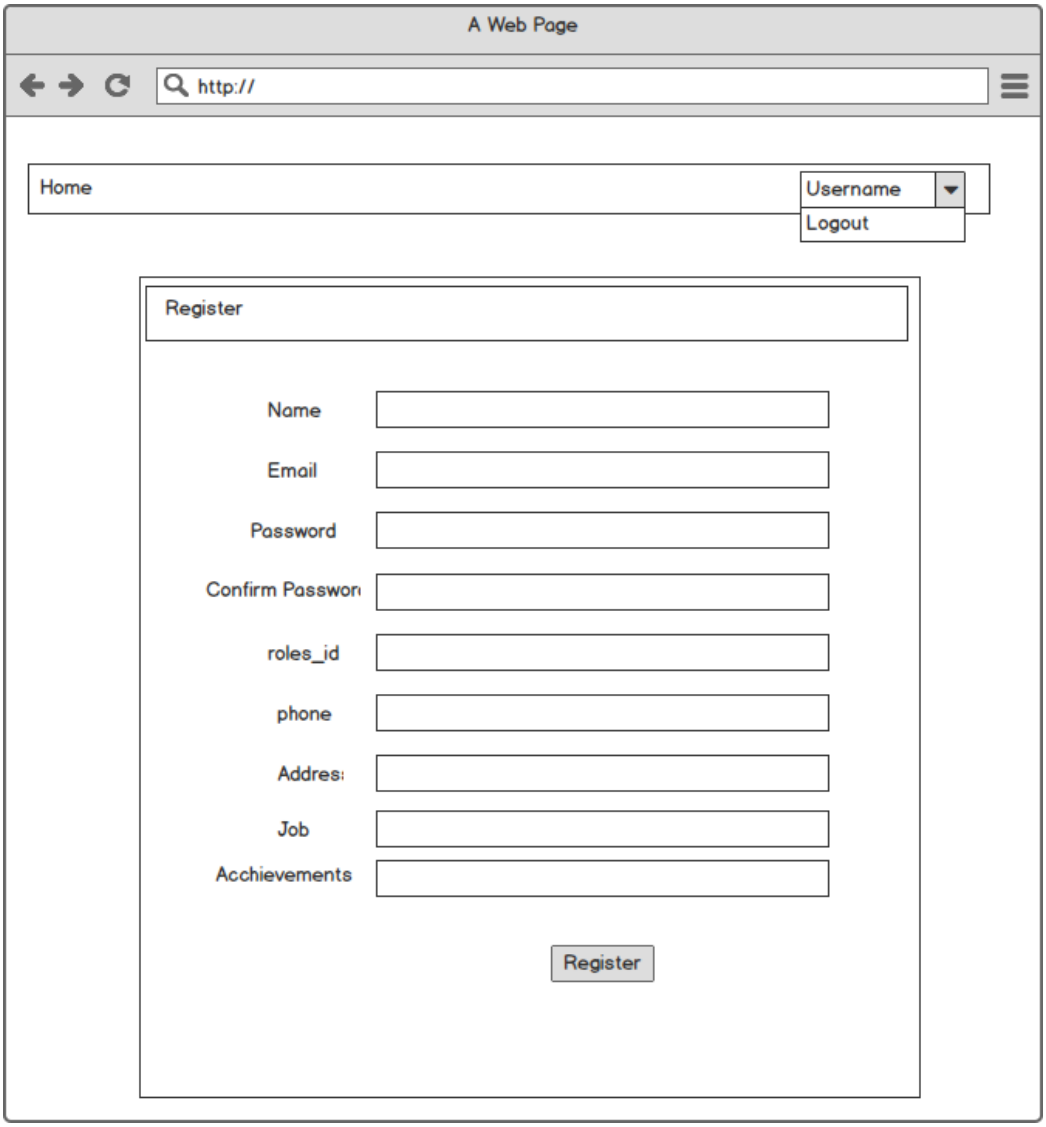


Figure 149 Web admin registration wireframe.

Admin web home page:

This page is the home page of Admin where general information is provided, and admin can navigate to many other pages of the website.

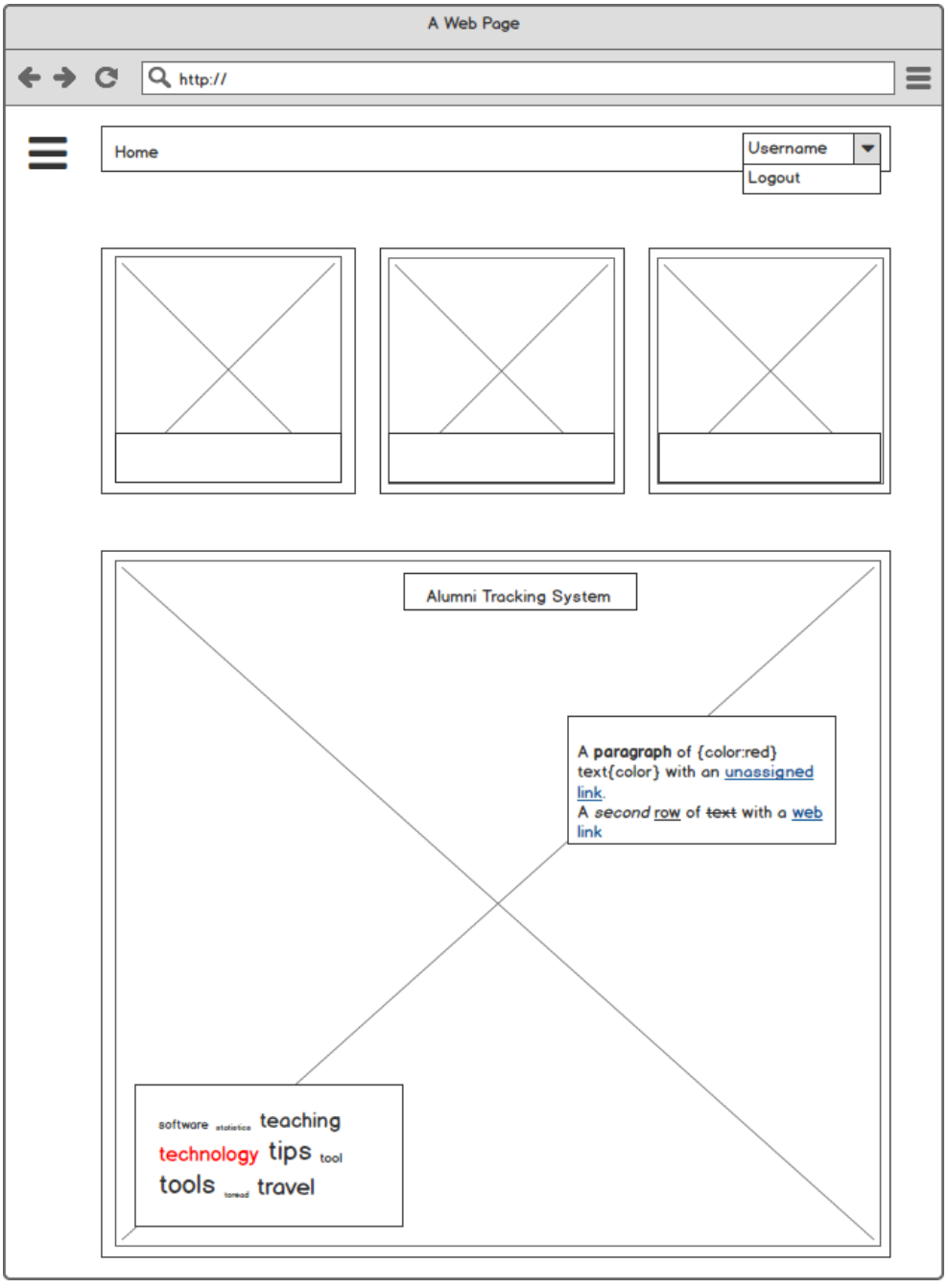


Figure 150 Admin web home page wireframe.

Admin website Users list page:

This page contains all the list of users and their information who are registered to the alumni application.

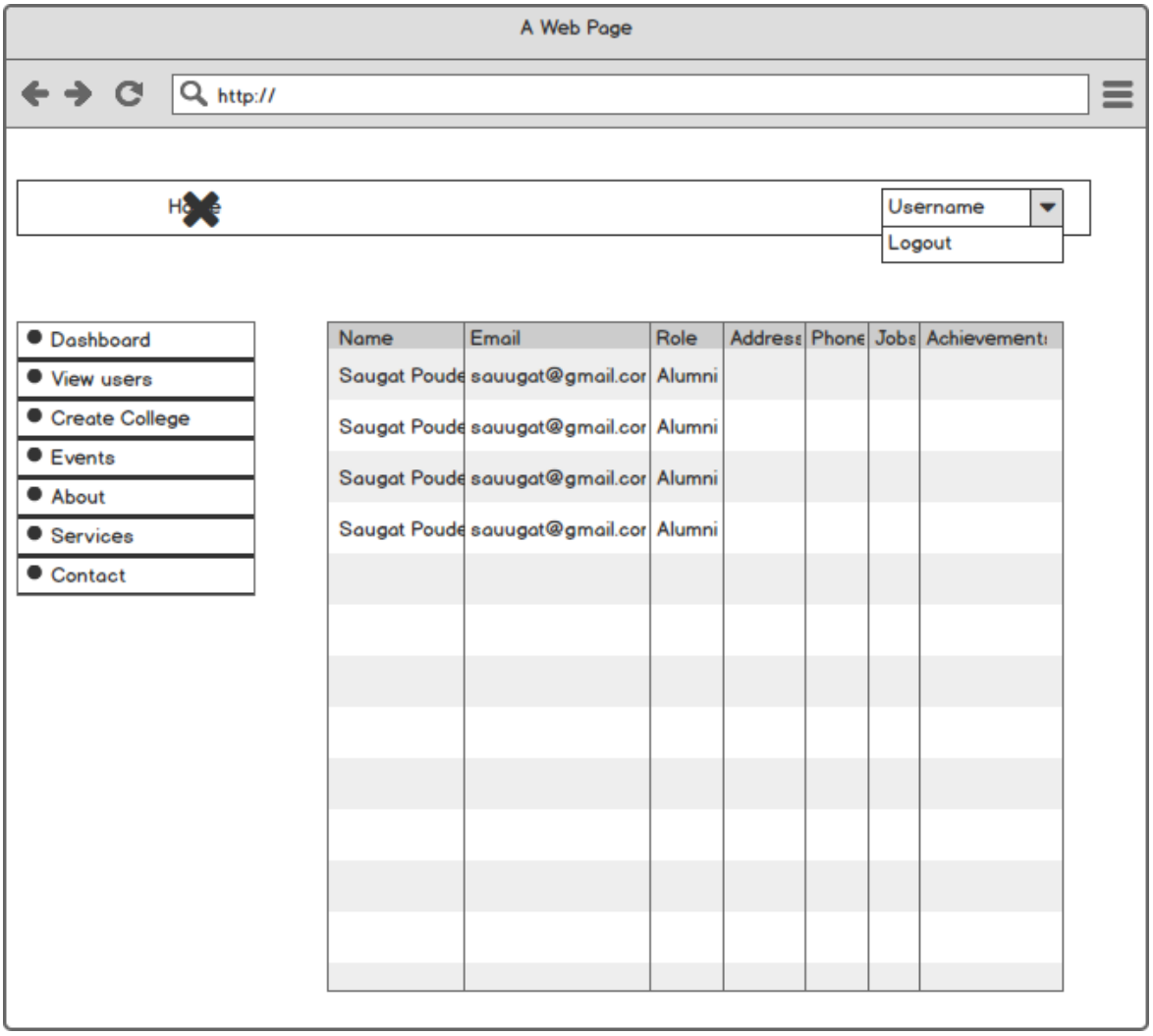


Figure 151 admin web users list page wireframe.

Admin website event list page:

This page contains all the list of events created by college with their details.

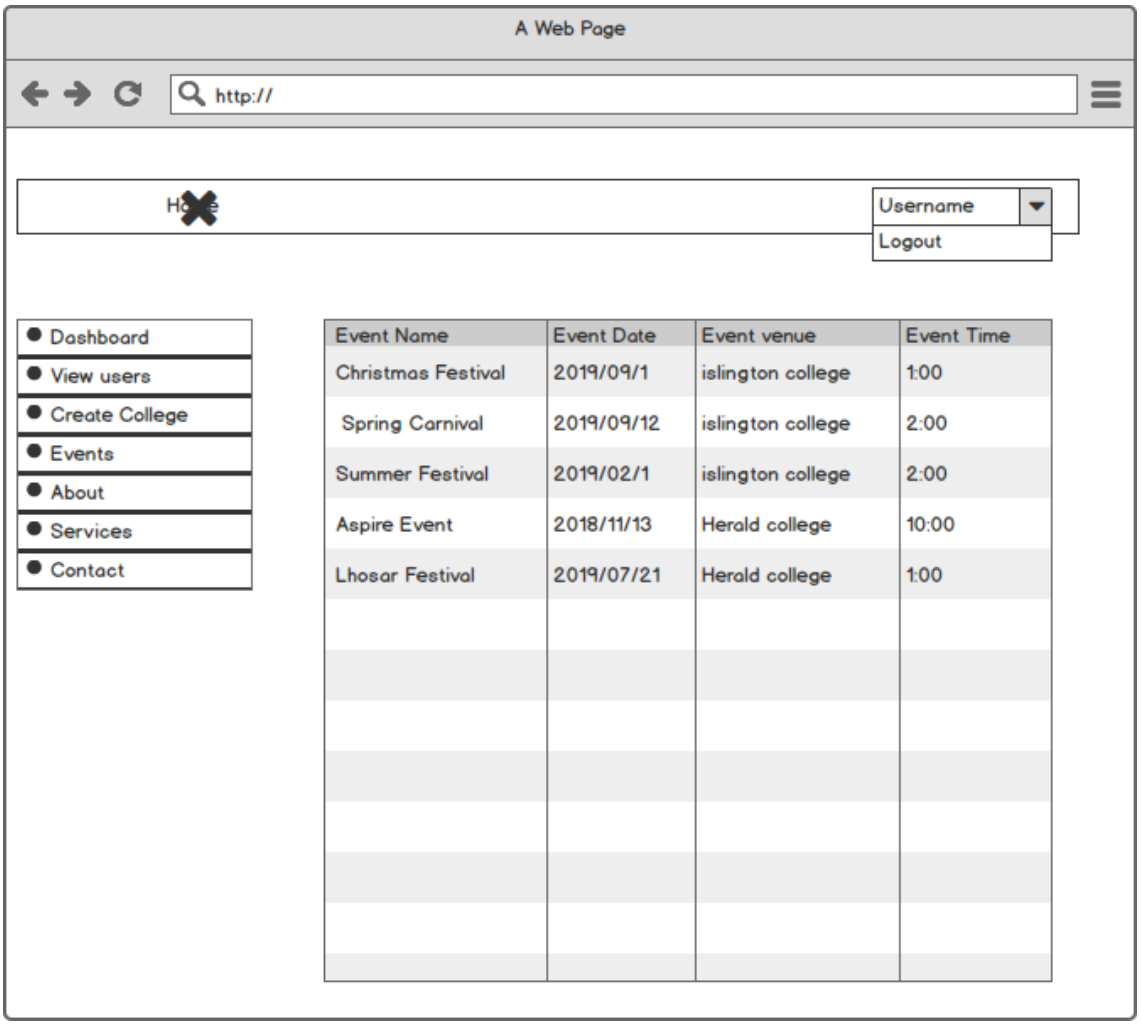


Figure 152 admin web event list wireframe.

8.5. Appendix E: Screenshot of the system:

Mobile Application development:

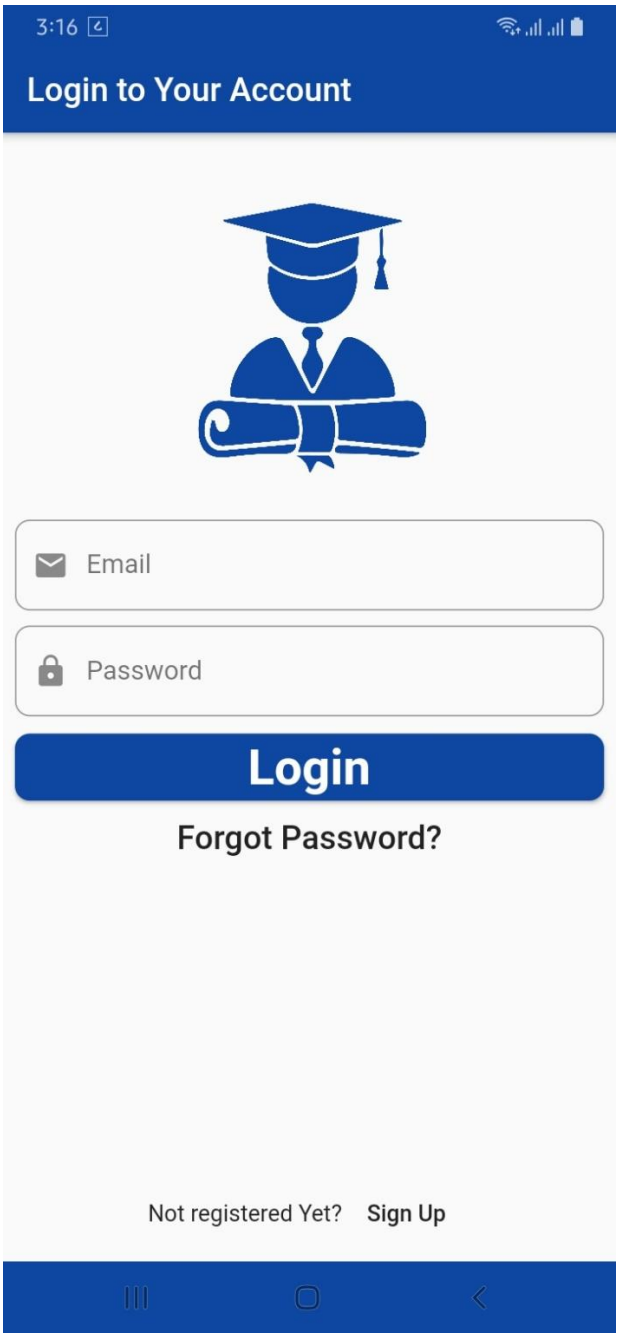


Figure 153 Login page UI.

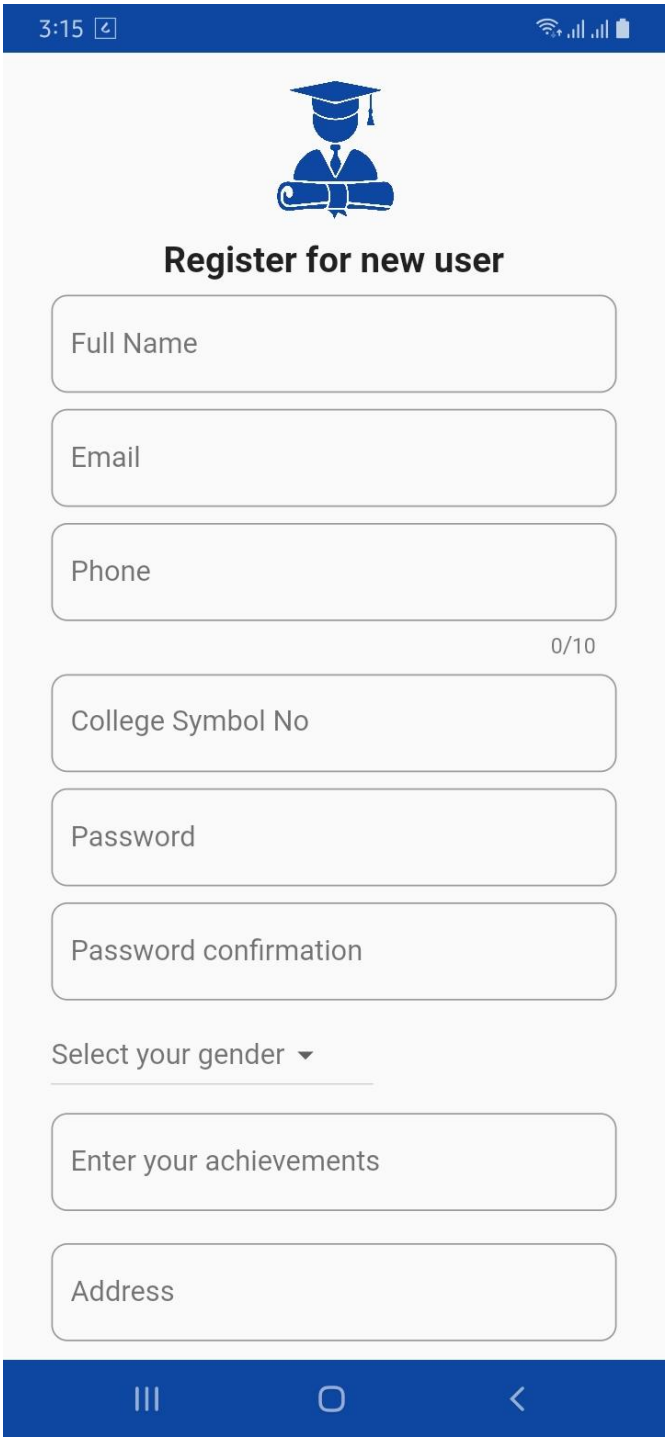


Figure 154 Registration UI

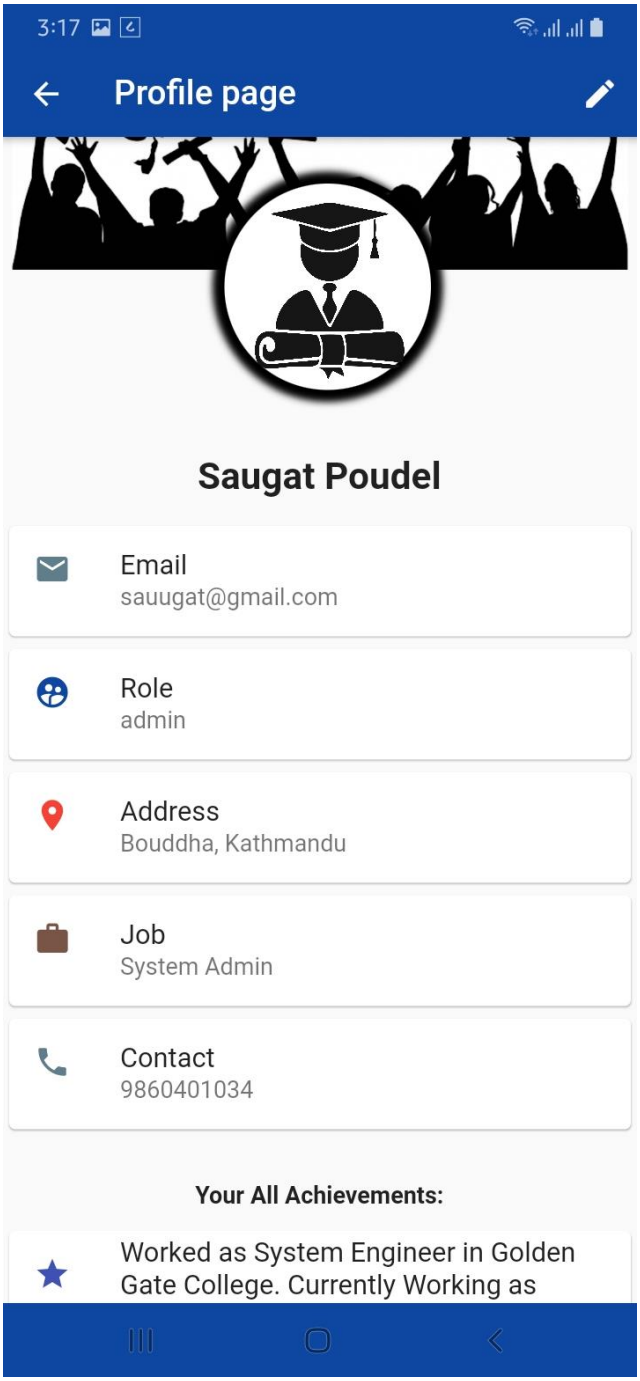


Figure 155 Profile UI.

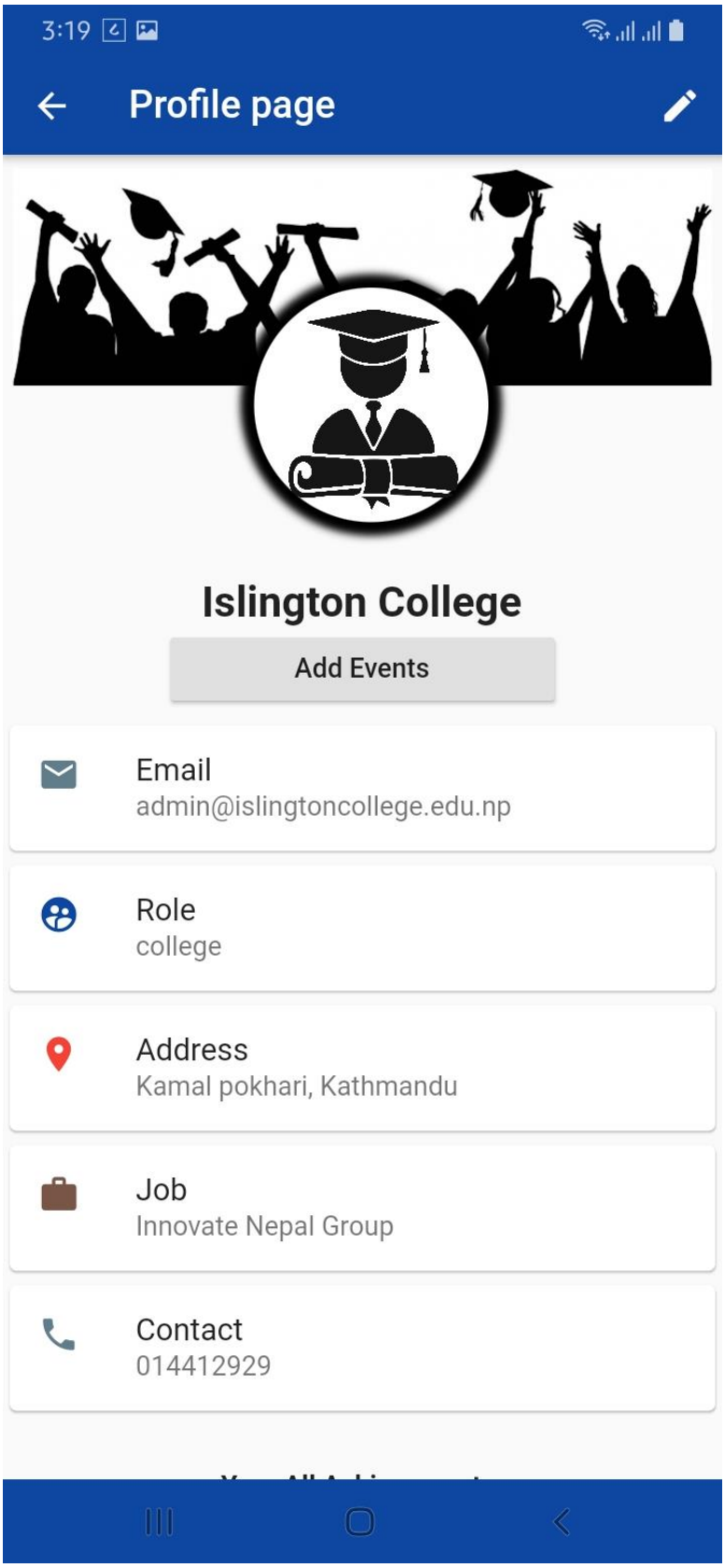


Figure 156 college profile UI.

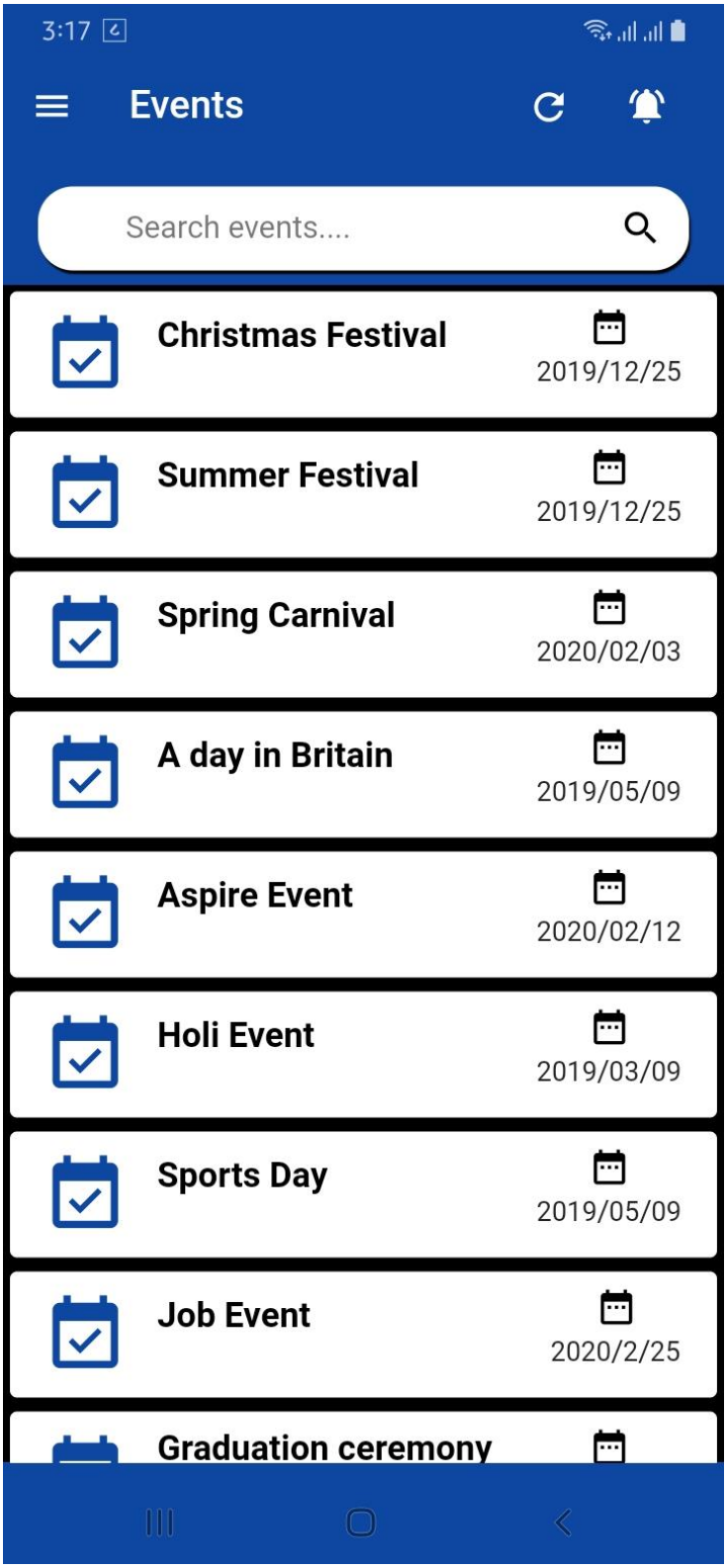


Figure 157 Home page UI.

Figure 158 event detail UI.

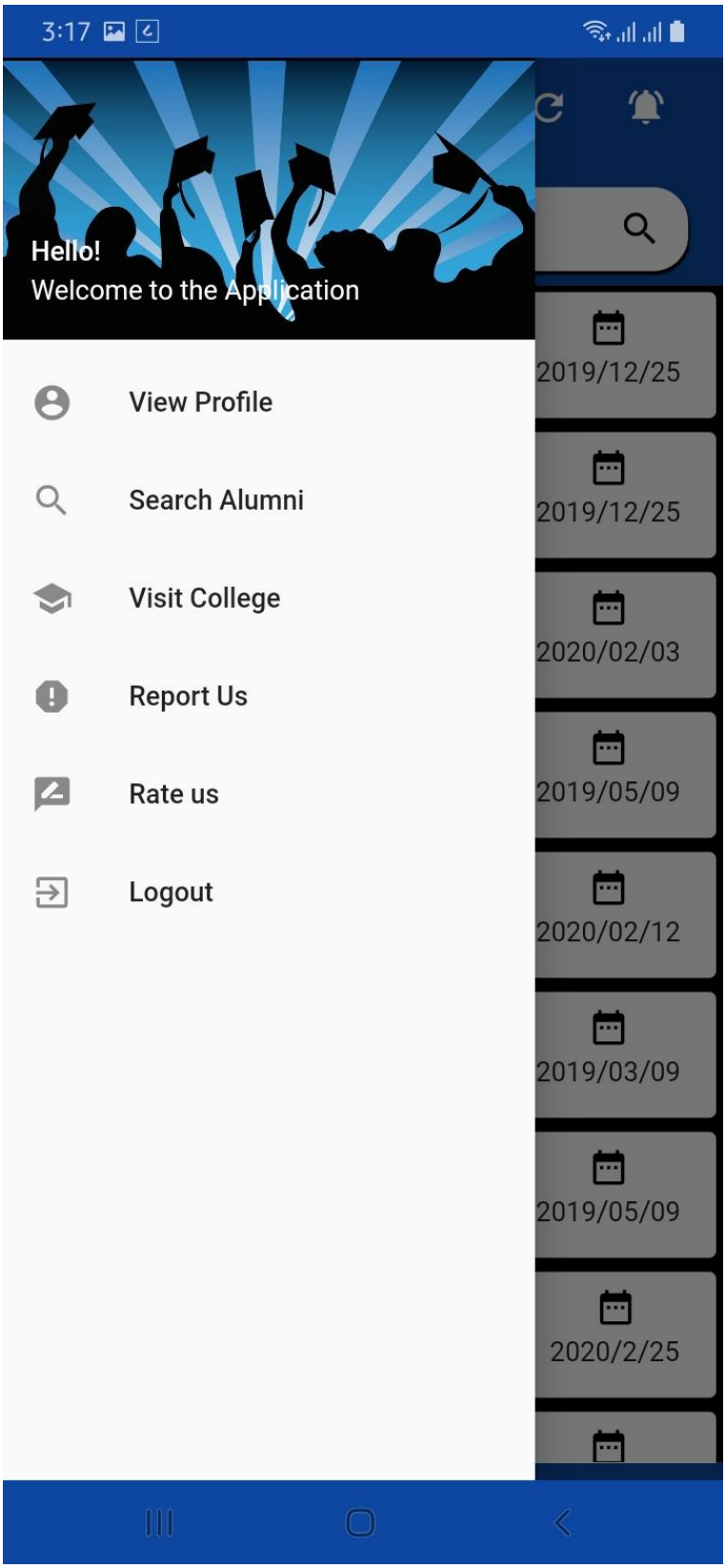


Figure 159 homepage navigation UI.

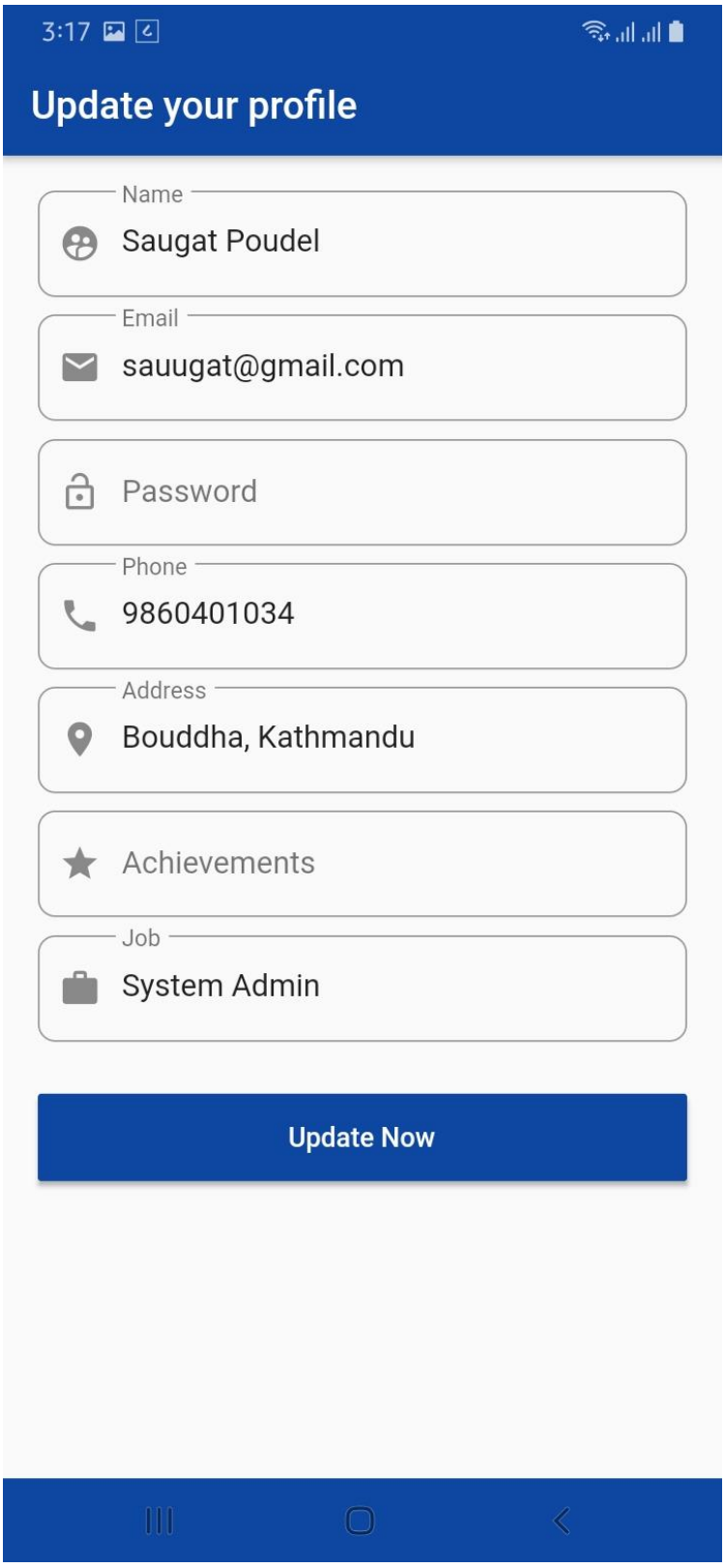


Figure 160 edit profile UI.

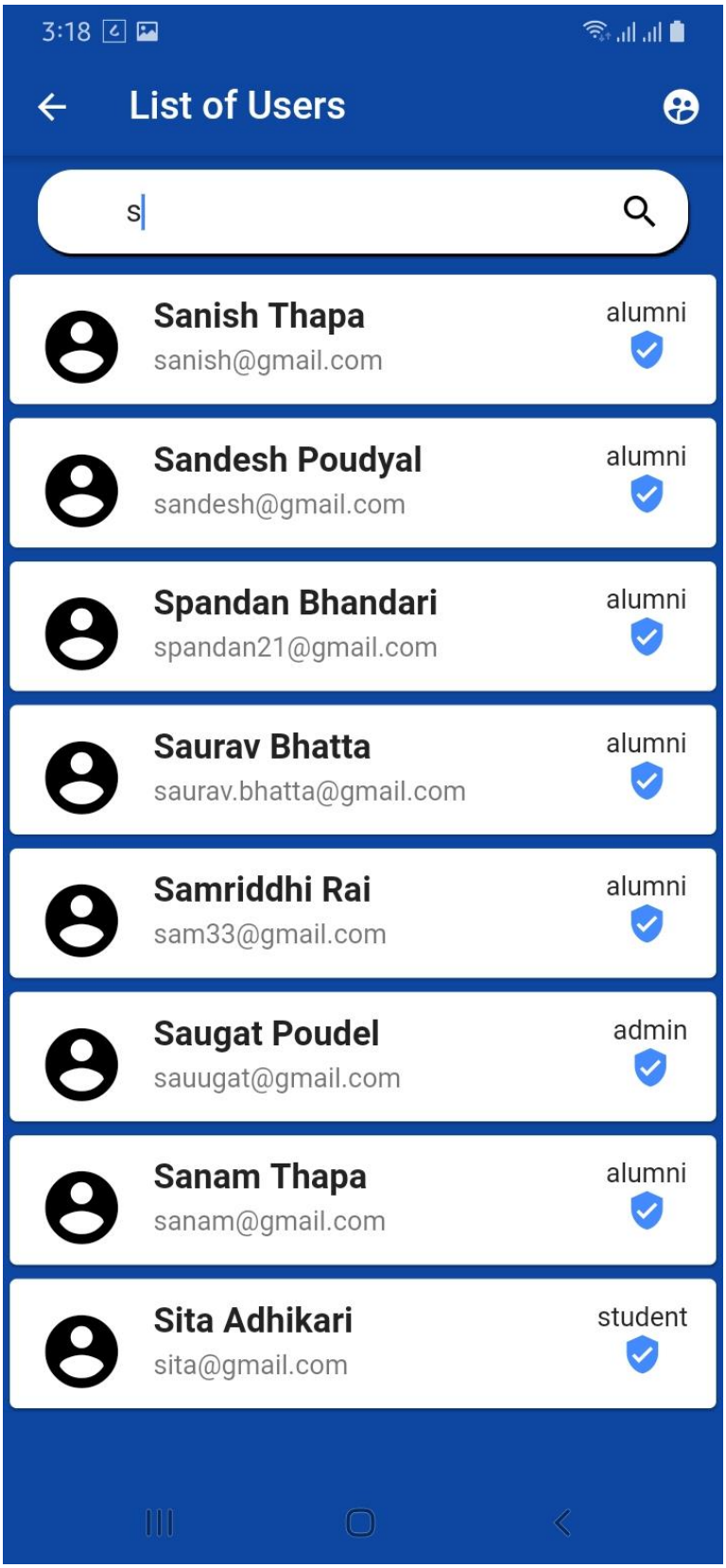


Figure 161 search user UI.

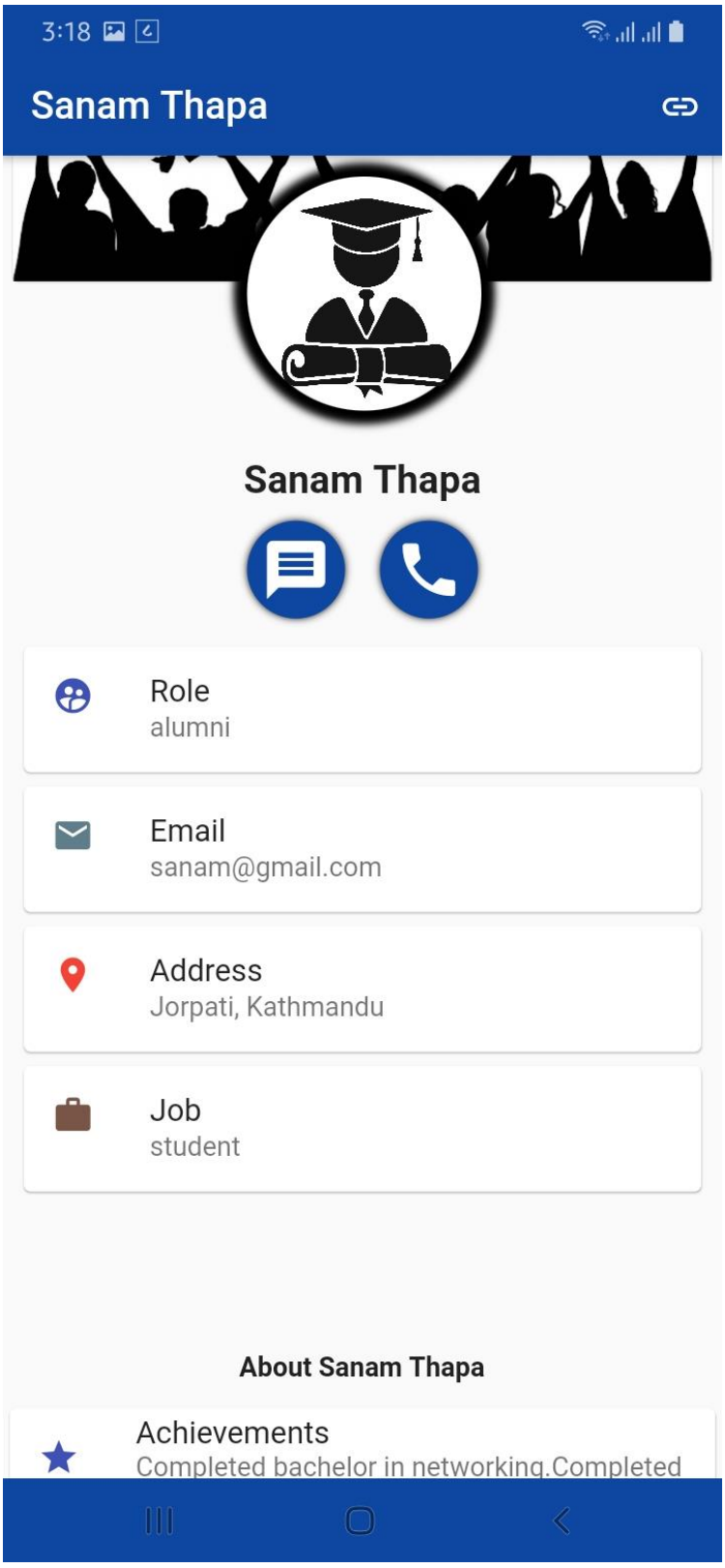


Figure 162 Alumni profile UI.

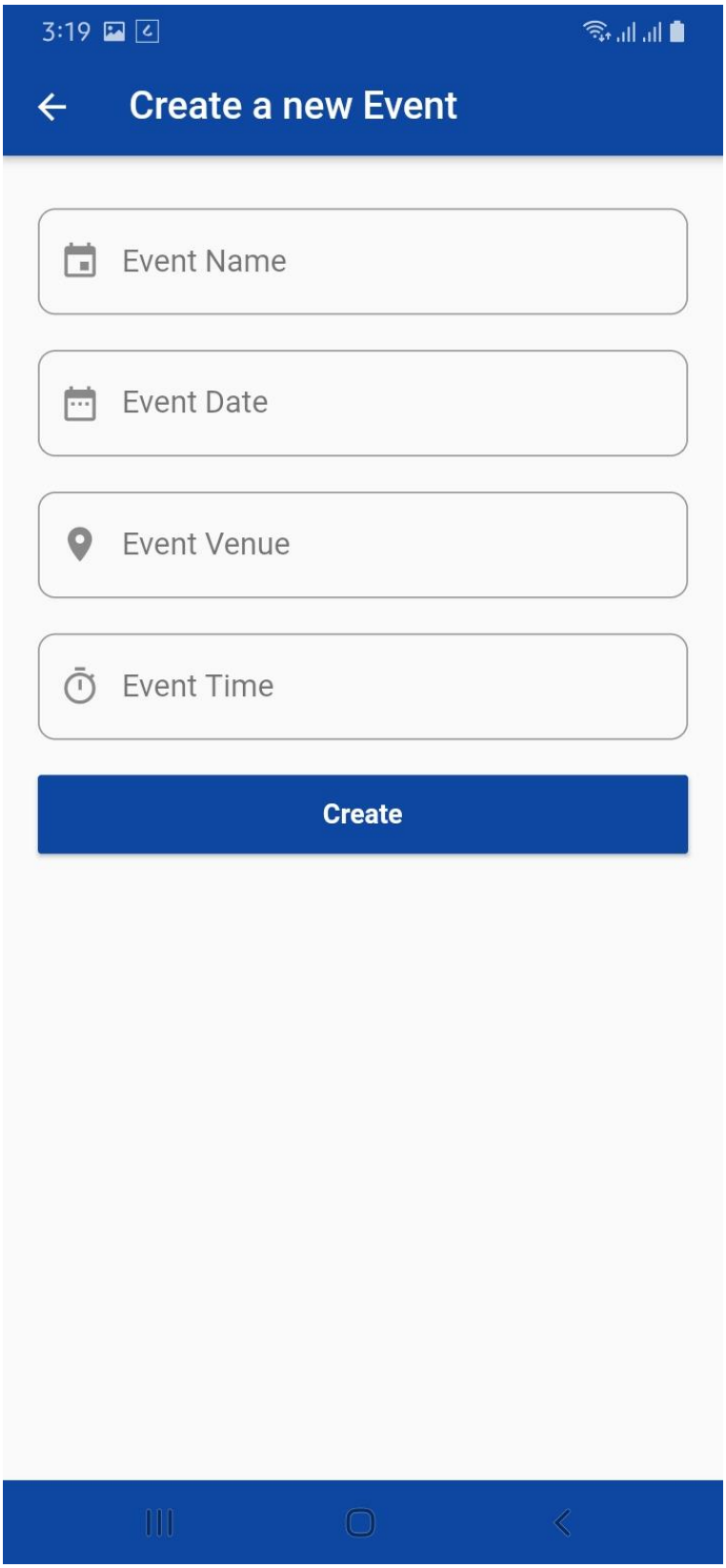


Figure 163 create event UI.

Admin Webpage:

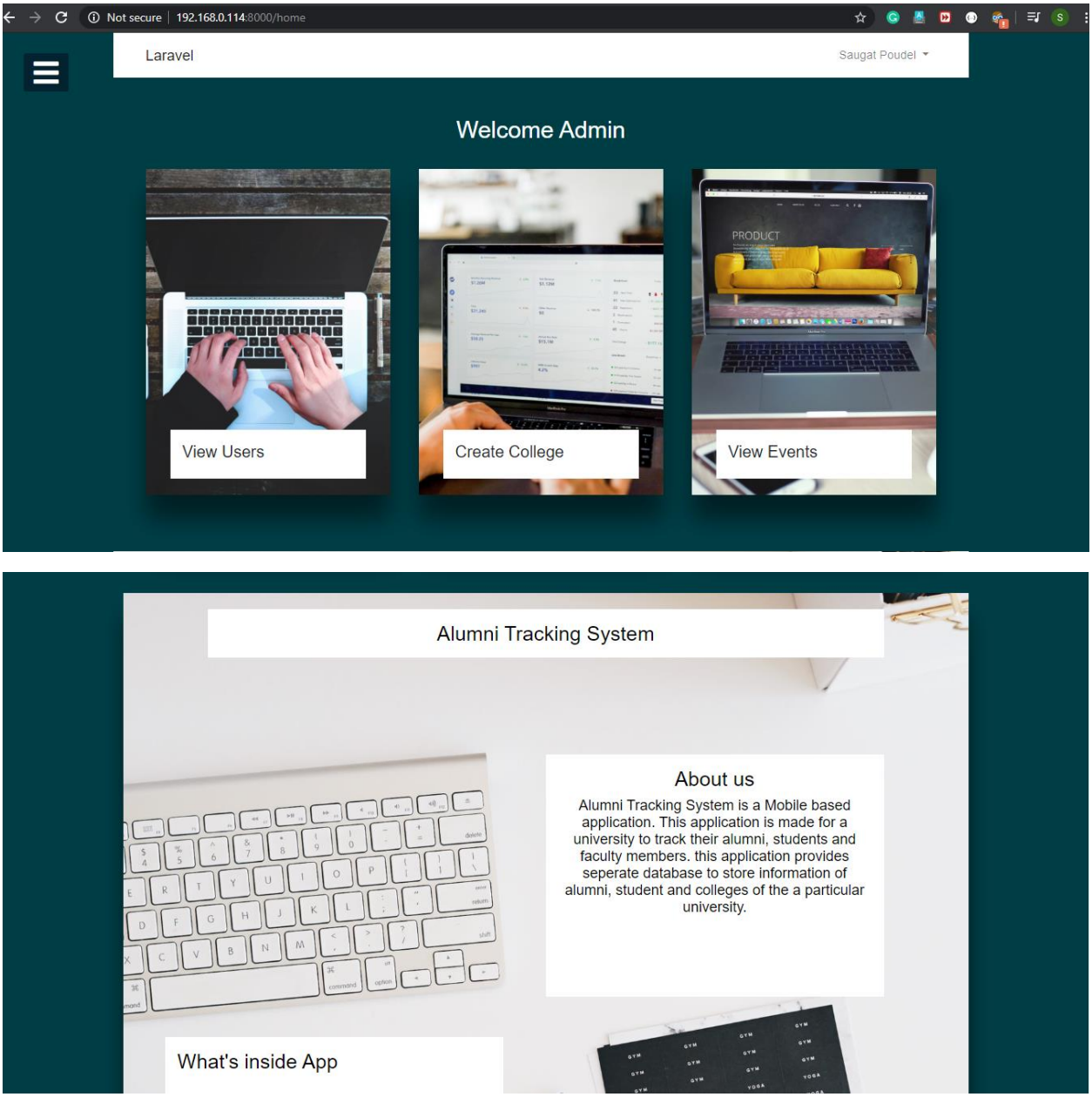


Figure 164 Admin web homepage.

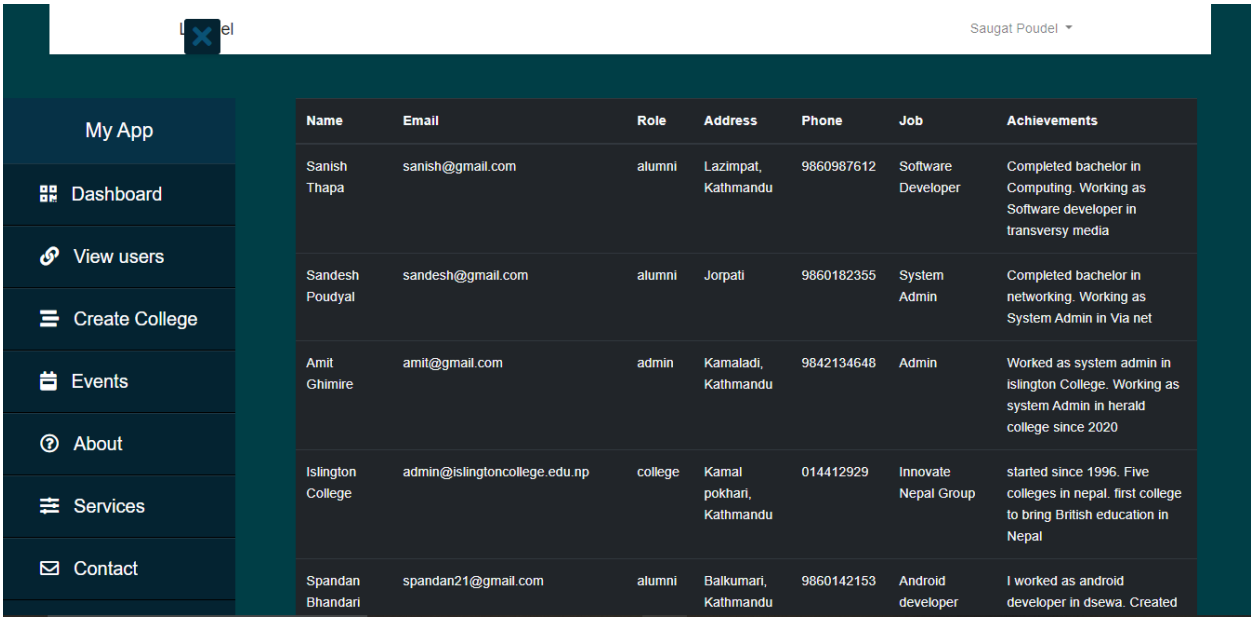


Figure 165 Admin web User list page.

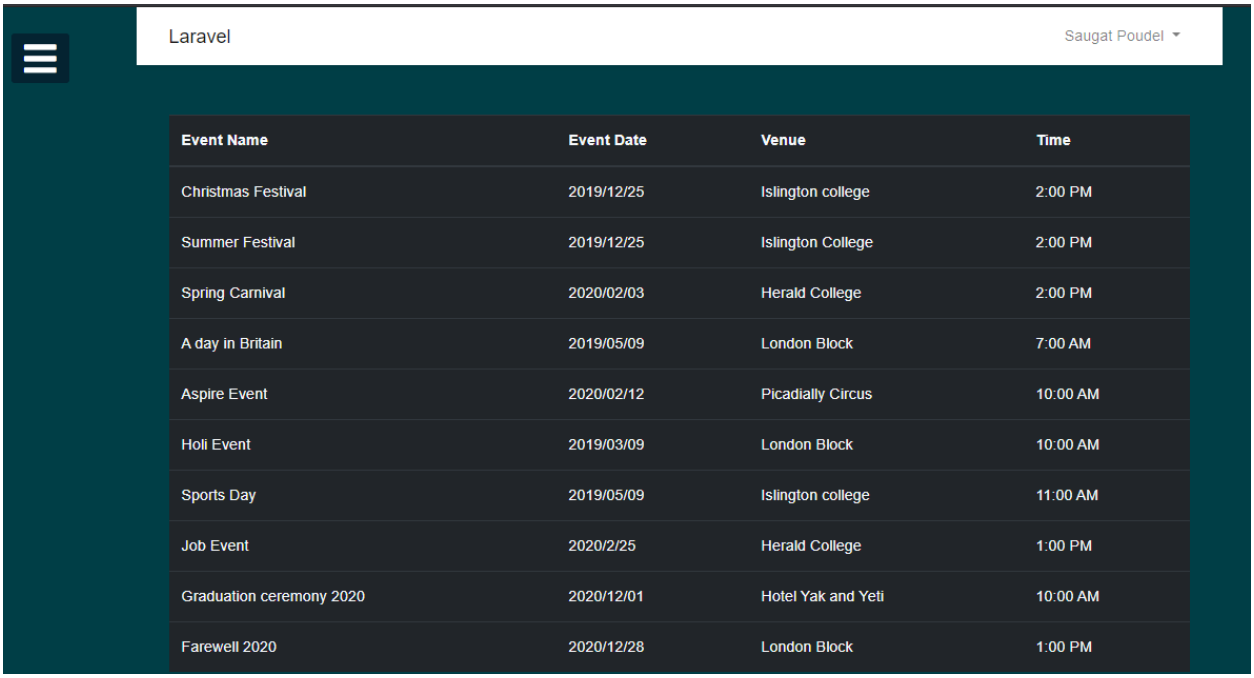


Figure 166 event page web App.

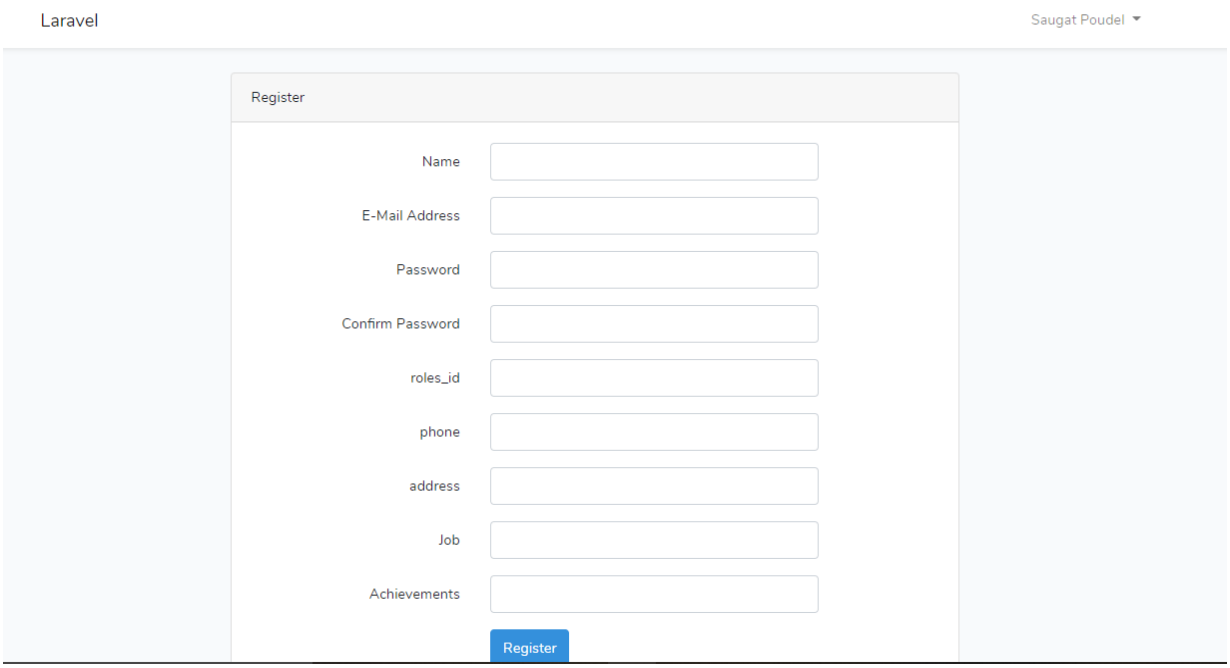


Figure 167 Register user Admin web.

8.6: Appendix F: User Feedback Form:

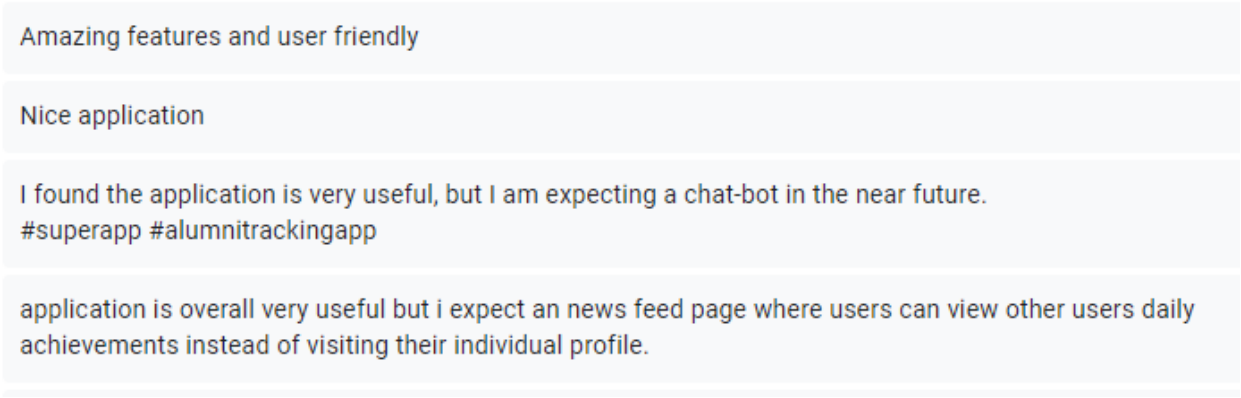


Figure 168 user feedback after using app.

8.7: Appendix G: Future work:

In this frequently upgrading era of technology it is very important to add new features and update the application time to time. The future works for this application depends upon the feedback of the users of this application. An application can have many features which depends upon the user's feedback. Every user has their own perspective of interacting in the application, some of them just opens the applications once in a week, when some of them opens daily.

The main objectives of future work for this application will be according to the user's desire and feedback. Last time a post survey was taken about this application where it is found that some of the users are suggesting adding news feed where users can watch every user daily progress without visiting their profile. The next work will be adding a news feed section in the application.

Also, some of the users are suggesting making a chatbot in this application so that they can chat with users without using mail. Here recently in this application uploading photos of users is not available. In the future work those features will be added. these features were not added because some people don't want to share their pictures for their own privacy but now, as many of the users are suggesting for it those features will be added in future.

8.8: Appendix H: More About Rational Unified Process:

Rational Unified Process Best Practices:

- Develop Software Iteratively: Encourages iterative development by locating and working on the high-risk elements within every phase of the software development life cycle.
- Manage Requirement describes how to organize and keep track of functionality requirements, documentation, tradeoffs and decisions and business requirements.
- Use Component-Based Architectures: Emphasizes development that focuses on software components which are reusable through this project and most importantly within future projects.
- Visually Model Software: Based on the Unified Modeling language (UML) the Rational Unified Process provides the means to visually model software including the components and their relationship with one another.
- Verify Software Quality: Assists with design, implementation and evaluation of all manner of tests throughout the software development life cycle.
- Control Changes to Software: Describes how to track and manage all the forms of change that will inevitably occur throughout development, inorder to produce successful iterations from one build to the next. (IBM , 2003)

Static Structure of Rational Unified Process:

Static Structure of RUP describes who is doing what, how, and when. It is represented using four primary modeling elements and they are as follows:

1. Workers (Who): A workers in Rup is defined as the behavior and responsibilities of an individual, or a group of individuals working together as a team. Here workers are the part of project which an individual is assigned. The responsibilities assigned to a worker

includes both to perform a certain set of activities as well as being owner of set of artifacts. Here in this project the worker is the part of project which me as a developer should develop. Part of projects like designing use cases, developing system adding features and others.

2. Activities (How): Activities in Rup is defined as a work assigned to a specific worker. The activity has a clear purpose, usually expressed in terms of creating or updating some artifacts, such as model, a class, a plan. Every activity is assigned to a specific worker and affects one or only a small number of artifacts.

   Example of activities are:

   Plan an iteration, for the Worker: Project Manager

   Find use cases and actors, for the Worker: System Analyst

   Review the design, for the Worker: Designer

   Execute performance test, for the Worker: Tester

3. Artifacts (What): An artifact is the piece of information that is produced, modified, or used by a process. It is the part of system which produced within a project to make a complete system. Artifacts may take various shapes or forms. It can also be considered as a model, such as use case model, design model, classes, software architecture documents, Source code of the system and so on. Here artifacts can be considered as alumni data model, events data model, diagrams of the system and can also be called as source code to design the application.
4. Workflows (When): Workflow in Rup describes a meaning sequence of activities that produce some results and shows interaction between workers. In UML terms, a workflow can also be defined as sequence, collaboration diagram, or an activity diagram. The workflow of this project is shown in design section of this project. In the following diagram the workflow of the system is shown about how the users interact to the application and performs the activity. (Rational Software White Paper, 1998)

8.9: Appendix I: Details of Iterations in Rational Unified Process:

Iterations in each phase of Rup:

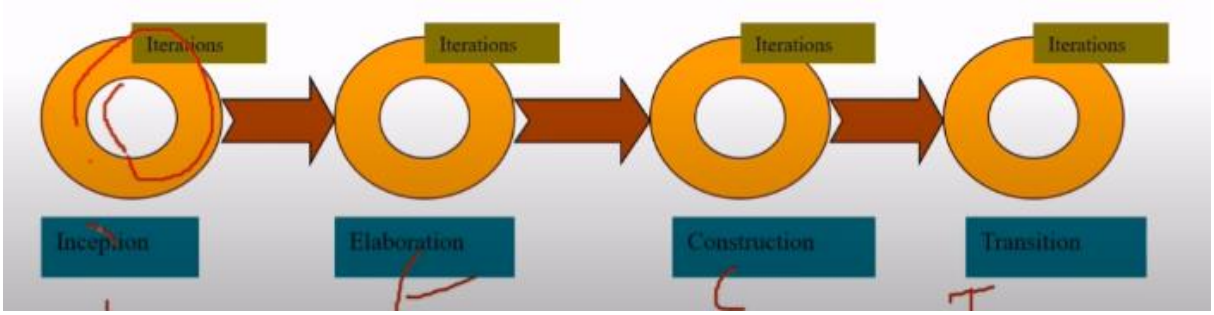


Figure 169 Iteration in Rup.

Iteration is the most important and frequently coming phase of the rational unified process.

All above phases have multiple iterations in each. The duration of iteration may vary from two weeks or less up to six months. As shown in the above figure Iteration occurs in each of the phase of this methodology where customers are collaborated with the developers and each phases of development are shown to them so that whenever they need some changes in the software the system can be iterated.

The RUP also recommends that each of the four above phases be further broken down into iterations, a concept taken from agile and other common iterative development models. Just as with other models, in the context of RUP an interatom simply represents full cycle of the core phases, until a product is released in some form internally or externally. From this baseline the next iteration can be modified as necessary until, finally, a full and complete product is released to customers (Morse, 2017). This phase was the most important phase to my system because with the help of this phase I was able to iterate through my system and implemented some basic changes in a proper way. While working on this phase I collected lots of feedback from my supervisor about the overall working mechanism and development process. According to my supervisor I did many iterations so that I can make the system as better as possible. The iterations are divided into multiple parts. I have shown the detail of each iteration below.

First Iteration:

As this was the first iteration of the system, there were so many changes to be made in this system.

As we know every iteration contains all Rup phases.

Inception phase of the first iteration was to discuss about project scenario and feasibility of the alumni tracking system. Before the first iteration I was planning to implement the backend system of the application in SQL lite database. After a research it was found that SQL lite database has many drawbacks if It will be used for this application. Some of them are as follows:

SQLite fails when it comes to user management as this project have multiple users Alumni, College, Students and Admin in this system. Any user can read/write the data without any special access. Any activity or process in the application can have access to the stored data which results in big security issue (ourcodeworld, 2019).

Due to such drawback the database implementation was changed from SQLITE to MySQL.

MySQL database is the most popular open source database. It is easy, secure, and enterprise ready. we can perform various operations like data normalization to make the data free of anomaly and redundancy (oracle, 2020). Choosing MySQL as the database was the best approach in this system because it can solve all above problems like security of data, managing roles of users, data anomaly.

Storing data to database directly from application is not the best approach so inorder to solve this issue Application Programming Interface was used as middleware between database and mobile application. Here in this figure below the database implementation before and after first iteration is shown.

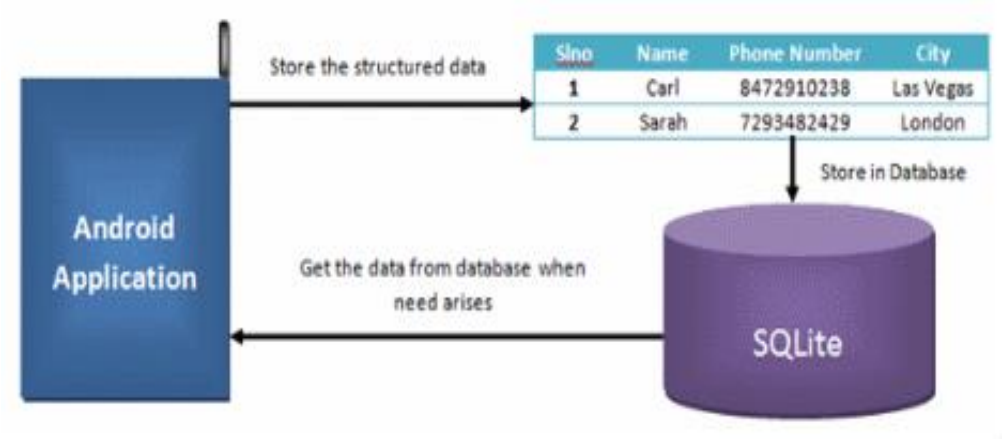


Figure 170 Database implementation before.

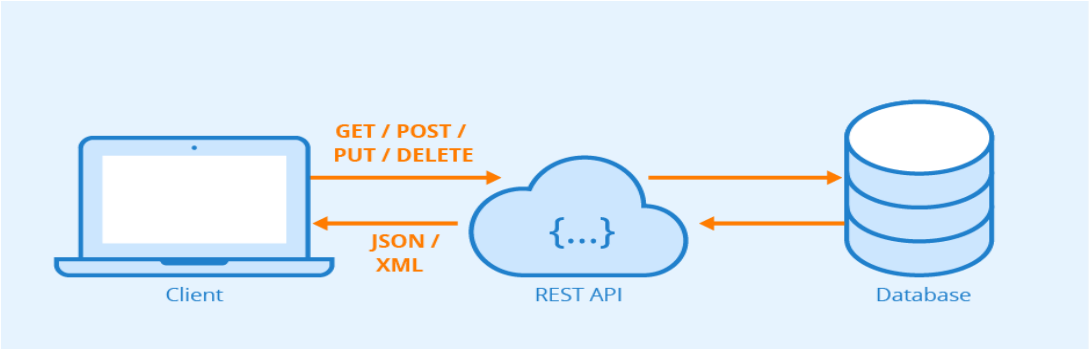


Figure 171 Database implementation after.

Elaboration phase of first iteration was based on creating system designs and architecture in the basis of inception phase of first iteration. In this phase diagrams and prototypes were made according to first iteration. All entities and attributes of the users and events were collected and normalized to 3NF in this phase. (Normalization of the system is shown in normalization section of this document).

Construction phase of first iteration was based on ideas of inception and elaboration phase of first iteration. As per the planning of first iteration the next step was to create rest api for user with the help of Laravel framework. This phase was based on creating api of users and database tables. (Rajgor, 2017).

AppTest.php api.php

routes > api.php

```
| is assigned the "api" middleware group. Enjoy building your API!
|
*/
// These are the api routes for the users table

Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});

Route::group([
    'prefix' => 'auth'
], function () {
    Route::post('login', 'AuthController@login');
    Route::post('signup', 'AuthController@signup');
    Route::post('Api/password/email', 'Api\ForgotPasswordController@sendResetLinkEmail');
    Route::post('Api/password/reset', 'Api\ResetPasswordController@reset');

    Route::group([
      'middleware' => 'auth:api'
    ], function() {
        Route::get('logout', 'AuthController@logout');
        Route::get('user', 'AuthController@user');

        Route::get('users', 'AuthController@users');
        Route::get('user/{id}', 'AuthController@getUserById');

        Route::put('userupdate/{id}', 'AuthController@updatebyid');


    });
});
```

Figure 172 api creation during first iteration.

Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	name	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
3	email	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
4	email_verified_at	timestamp			Yes	NULL			Change Drop More
5	password	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
6	roles_id	int(11)			No	None			Change Drop More
7	remember_token	longtext	utf8mb4_unicode_ci		Yes				Change Drop More
8	Phone	varchar(10)	utf8mb4_unicode_ci		No	None			Change Drop More
9	Achievements	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
10	Address	varchar(100)	utf8mb4_unicode_ci		No	None			Change Drop More
11	Job	varchar(100)	utf8mb4_unicode_ci		No	None			Change Drop More
12	created_at	timestamp			Yes	NULL			Change Drop More
13	updated_at	timestamp			Yes	NULL			Change Drop More

Figure 173 users table creation during first iteration.

Second Iteration:

Second Iteration was based on front end implementation and admin panel.

Inception phase of second iteration was based on planning about user interface and system control. After creating api for user the next step was to implement it into front end mobile application. register and login page of users for mobile application was created in this phase. While registering users we have different roles. A user should choose one of the roles and should register themselves with the system As all users must use mobile application an admin panel website was necessary for user control which should be web based so that the admin of the system can do their basic work from the web without using mobile application. Also, in most of the cases admin of the system use web to control users so in this phase an admin panel was also created.

In the elaboration phase of second iteration prototype for admin website was created. The old UML diagrams was updated and added admin as a new user for website. In this phase wireframes of login, registration, event page, profile page of the mobile application was designed.

(Note: UI wireframe creating during this phase is in design section of the document.)

Construction phase of second iteration was based on creating admin website. Laravel Authentication system was used for creating basic login and registration and using html css and java script different webpages in admin panel was created. Webpages contains information of all users who are registered to the system and contains events created by college through application. As the main feature of admin was to create a new college. Admin can register a new college from this website too. Login and registration were implemented in mobile with api authentication. (Flutter.dev, 2020)

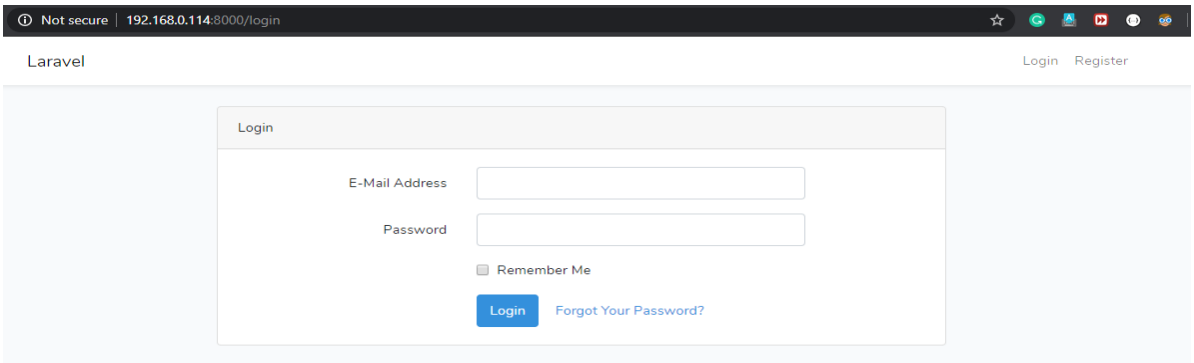


Figure 174 construction phase second iteration login admin.

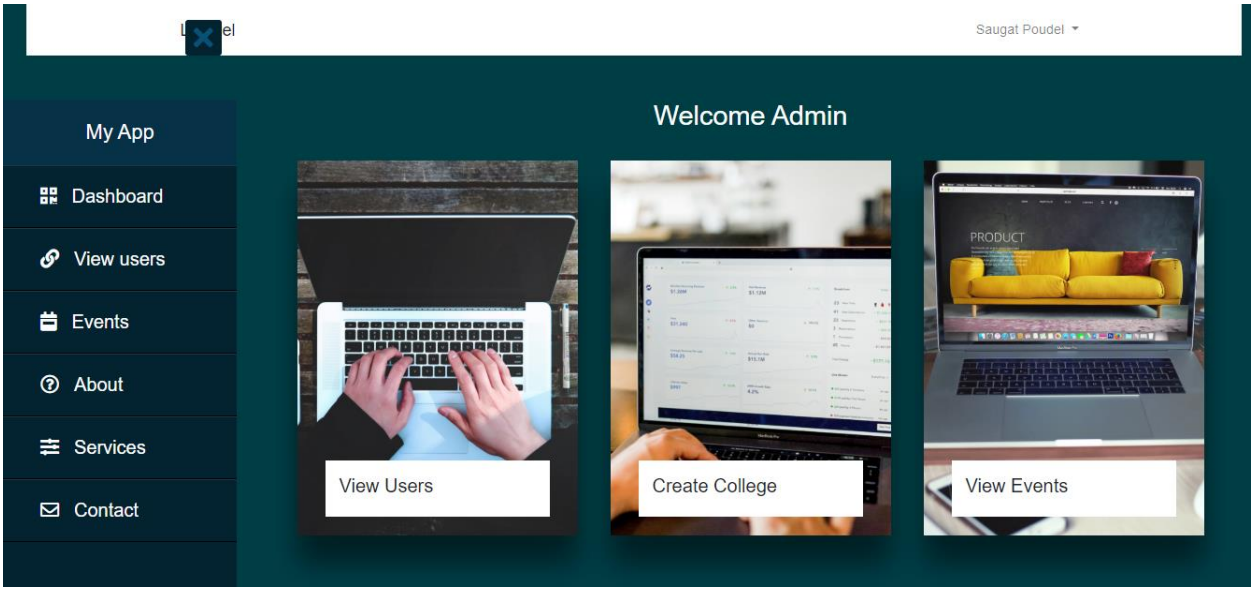


Figure 175 construction phase second iteration admin home page.

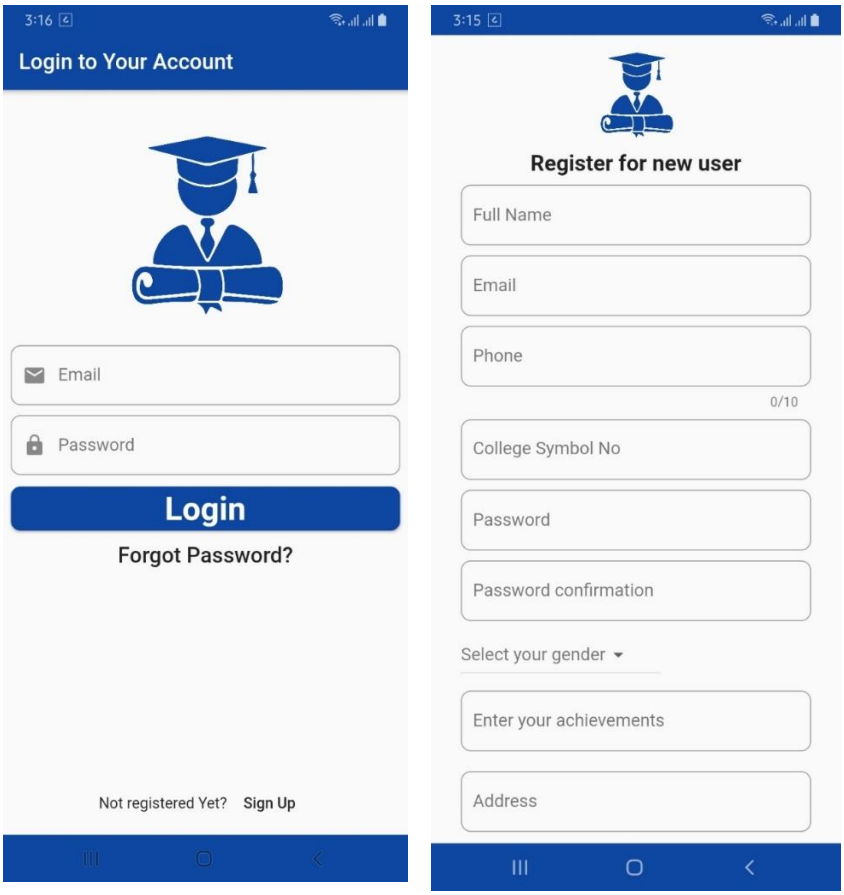


Figure 176 login register user interface creating in second iteration with api authentication.

Note: more designs created during this phase is shown in the design section of the report.

As the application is not completed the transition phase doesn't occur in this iteration. After completing the second iteration again according to the feedback of user third iteration was done.

Third Iteration:

Till this iteration the basic work of the application was completed for both mobile and website. While coming to this iteration user can be registered through mobile application. Login and view their profiles. Also, the work of creating events from mobile application and showing them in home page was completed. Users were able to search another users and events created by college and see the details. The search functionality implemented in second iteration was not the best approach as it was a front-end search. Also, the validation of the user registration and login was based on frontend and was implemented in a wrong way. Creating backend searching system instead of searching in front end was the best approach.

In second iteration the system was collecting all the list of users from backend and was searching from the list in front end from which when a user wants to search a specific user all users was fetched. Which is the bad approach in case of thousands of users thousands of data are fetched just to get one data. Also, about the user's validation is planned to implement it in a better approach. In case of showing list of events all events were shown in second iteration but in this iteration the data was paginated so that the application can only fetch specific amount of data at a time.

Elaboration phase of third iteration was based on updating UML diagrams and wireframes just to make it better than before. Basic changes like showing relation between users and entities and other details.

In construction phase of third iteration separate api was created for searching users and events, also the pagination property in events data was used. Later, data was fetched according to backend search and validated the login and registration form in a proper way. The events list is now paginated and are in proper structure. In this phase tasks like updating users' profile, creating new events by college and changing role by students to alumni was implemented.

```php
// this is the api routes for events table:

Route::get('events', 'EventController@index');

Route::get('allevents', 'EventController@allevents');      You, 10 days ago • xa

Route::get('search', 'EventController@search');

Route::get('event/{id}', 'EventController@show');

Route::post('event', 'EventController@store');

Route::put('event', 'EventController@store');

Route::delete('event/{id}', 'EventController@destroy');
```

Figure 177 events api creation in third iteration.

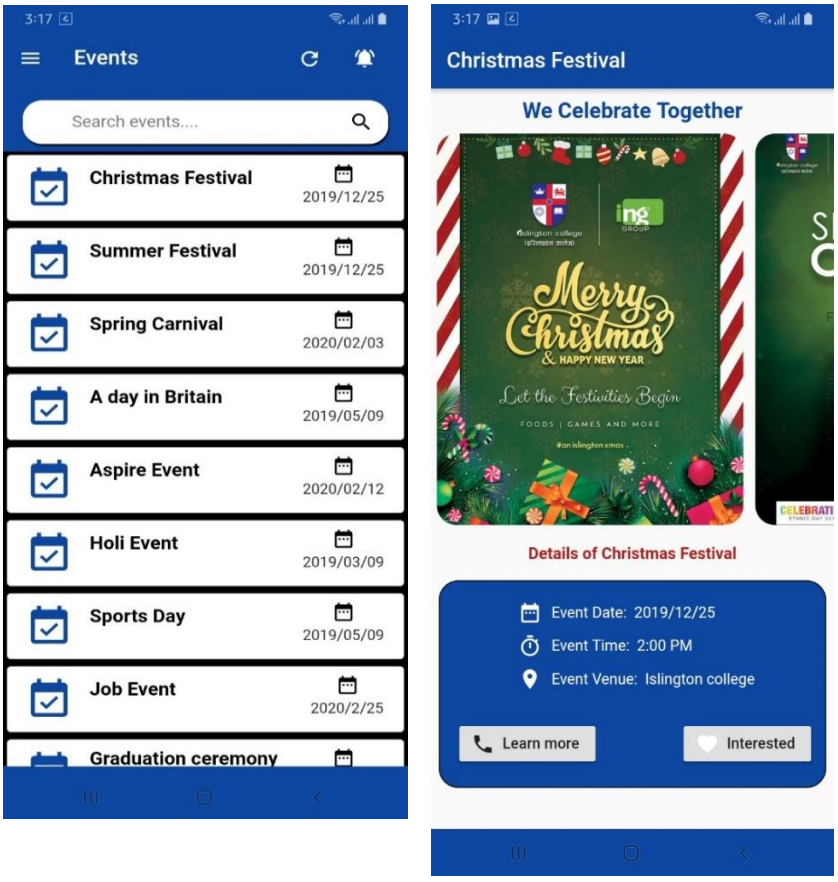


Figure 178 event page api implementation in third iteration.

Another feature of the application was to create notification to all the user about the events created by college.

Fourth Iteration:

Fourth iteration was based on managing notifications and overall system workflow. Here in third iteration the implementation of event notification was just a local notification to the specific device but while knowing more about it the notification was not real time. Only a specific device was able to get notification. Then to overcome this problem and make notification real time firebase push notification was used (pub.dev, 2020).

In elaboration phase of this iteration UML diagram such as DFD, sequence, collaboration and activity diagram were created. All the remaining documentation was completed in this phase, so this was the time-consuming phase of the project.

Construction phase of Fourth iteration was based on creating real time notification from firebase and checking overall system and fixing minor errors and features.

Also, each test cases were performed in this phase. Testing like unit testing and system testing was done. In unit testing the minimum validation and authentication was checked. In system testing the overall features of the application was checked and shown in the documentation.

Transition phase has occurred in this phase of iteration because the overall system was completed, tested and ready for transition. After this phase the system was submitted.

In Rational unified process the iteration never ends. An application can have many features to be added in the future. Likewise, in this application also many features can be added in future so after collecting the user's feedback again another iteration will take place and bugs will be fixed and new features will be added.

8.10: Appendix J: More test cases of System:

Testing section:

Testing for reset password:

Objective	To reset password of user from web.
Expected Result	Should change password of user by emailing the link.
Actual Result	New password was reset, and the link is sent to mail trap.
Conclusion	The test was Successful.

Table 29 reset password testing.

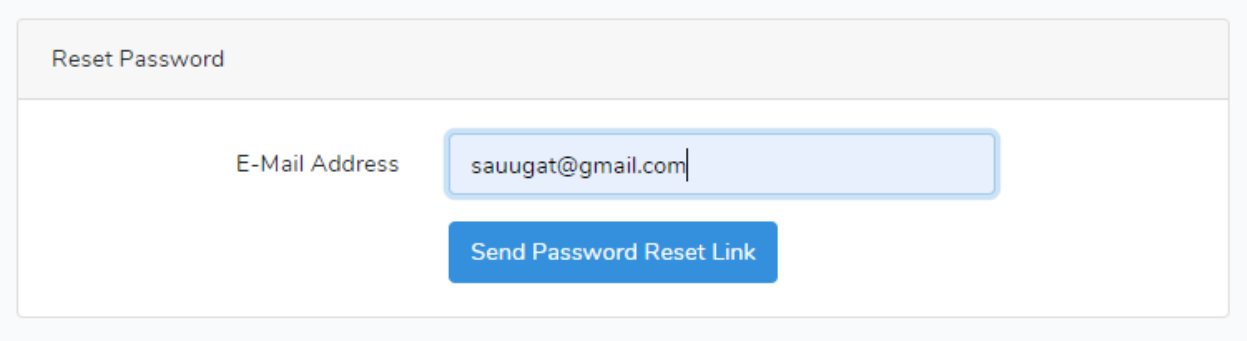


Figure 179 forgot password enter email.

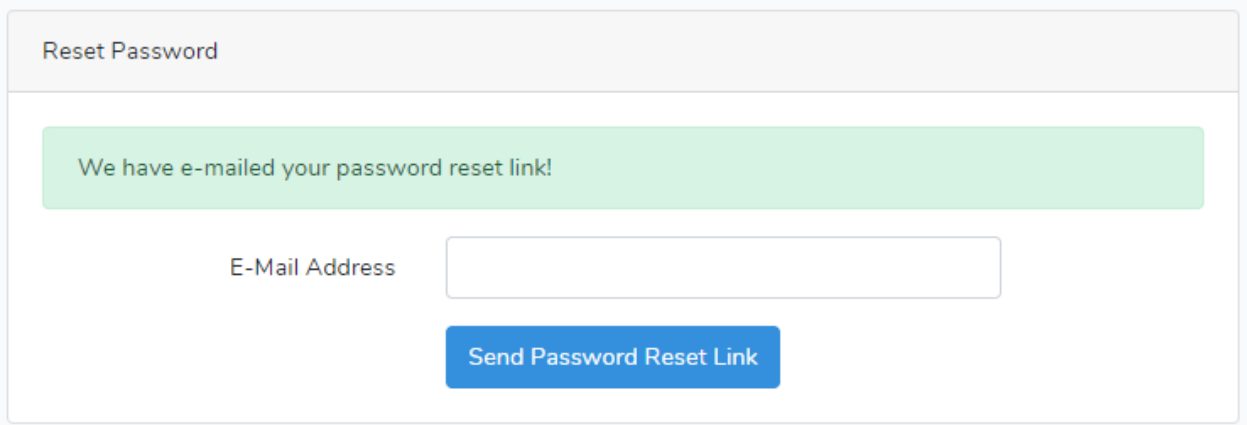


Figure 180 forgot password email send success.

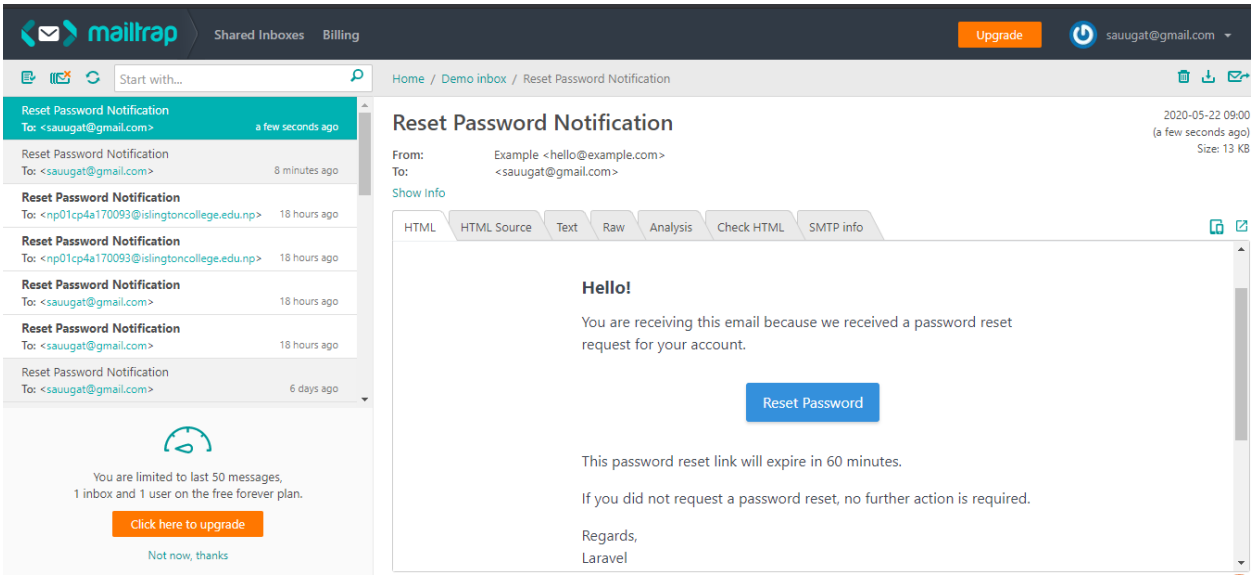


Figure 181 forgot password mail trap link.

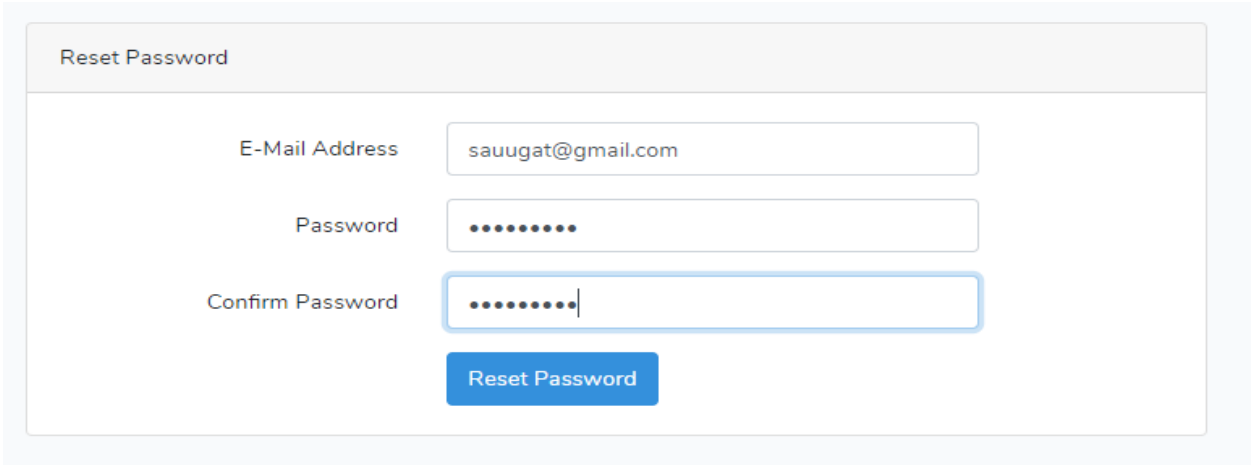


Figure 182 forgot password set new password.

Testing for View profile:

Objective	To view the profile of user who is logged into the system.
Action	User selects view profile options from home page navigation.
Expected Result	Should shows the detail information of user who is logged into the system.
Actual Result	All details of user were shown who is logged into the system.
Conclusion	The test was successful.

Table 30 testing for view profile.

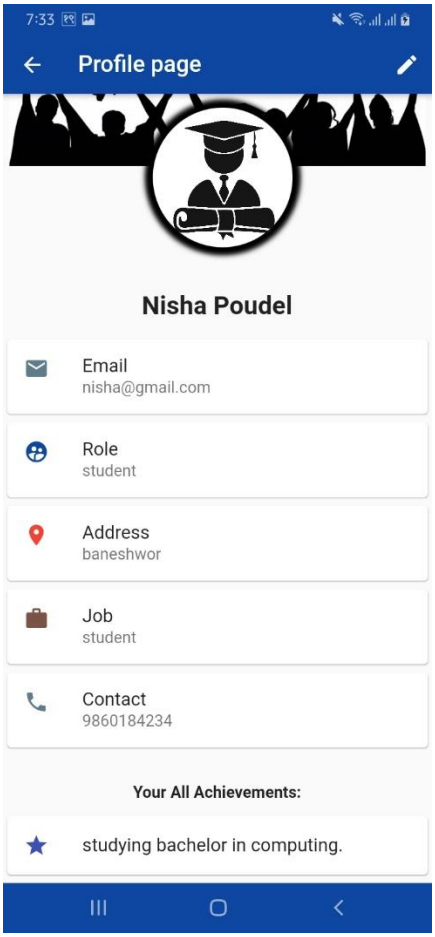


Figure 183 testing for view profile.

Testing for updating user's role:

Objective	To change the user role from student to alumni.
Action	User with role of student clicks on update button of top right button.
Expected Result	Should checks the user's role and ask to update role only if the user is student.
Actual Result	As the user was student the system asks to either update the roles or not.
Conclusion	The test was successful.

Table 31 testing for update user role.

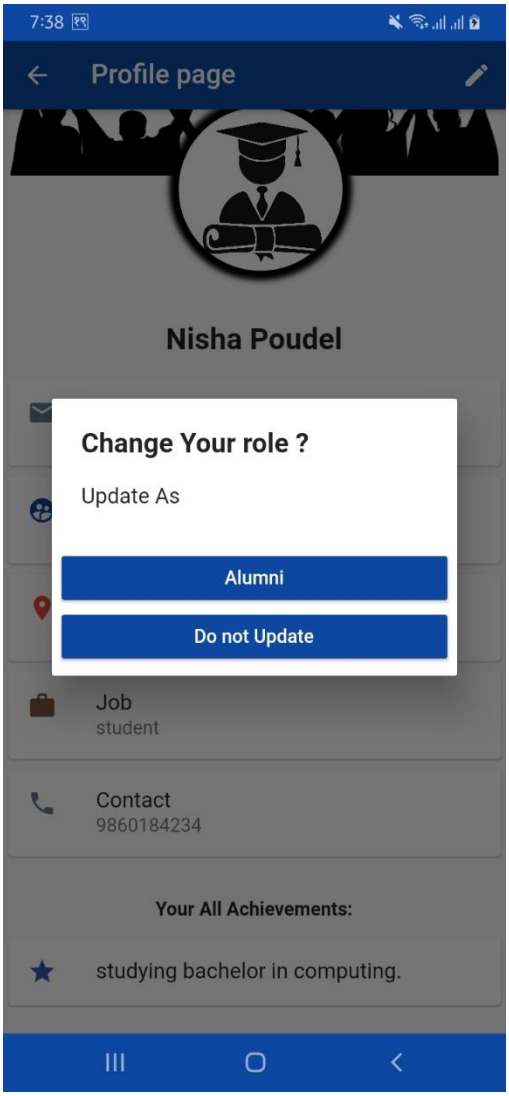


Figure 184 testing for update user role.

Testing for Update profile of users:

Objective	To update profile of logged in user.
Action	User clicks update button without entering password.
Expected Result	Should validate password field and avoid user to update data.
Actual Result	Error message was shown for verifying password.
Conclusion	The test was successful.

Table 32 testing for update user error.

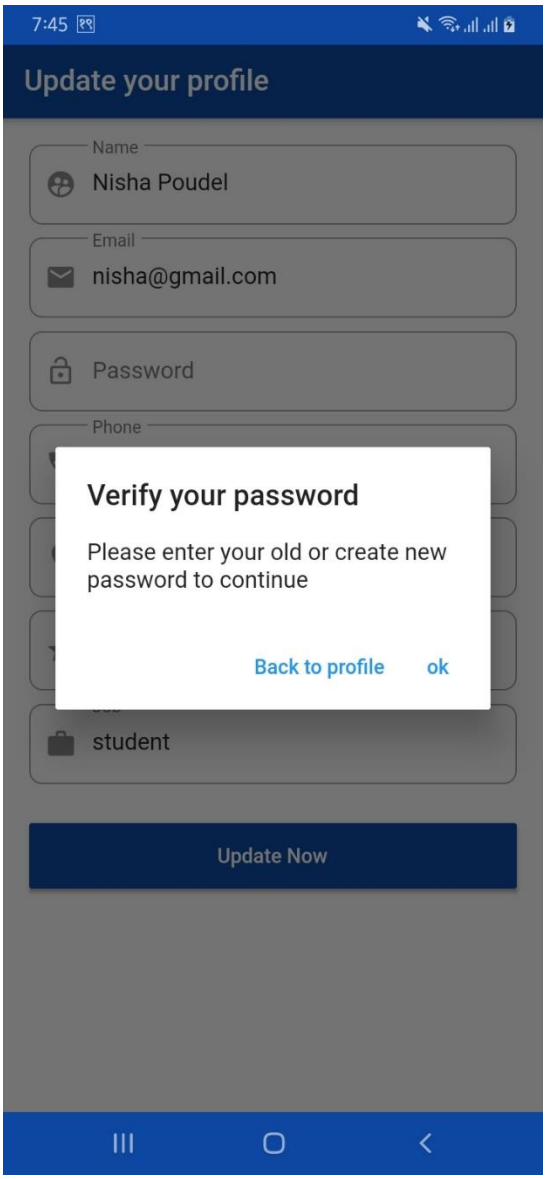


Figure 185 testing for update user password error.

Testing of Update profile successful:

Objective	Updating the profile of logged in user.
Action	User adds new achievements and enters password and press update button.
Expected Result	Should update user profile with new data.
Actual Result	Success message was shown as profile updated successfully.
Conclusion	The test was successful.

Table 33 testing for update profile success.

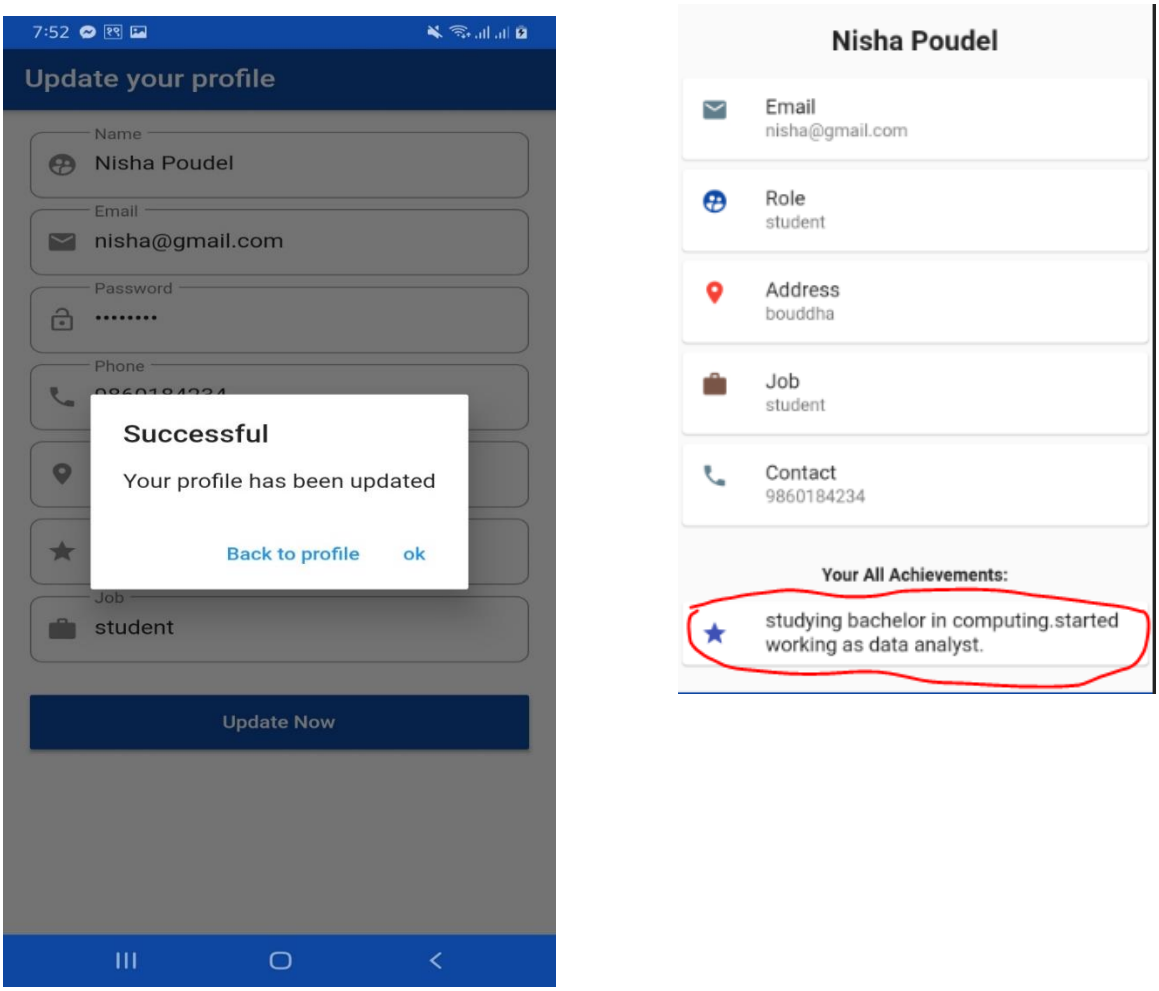


Figure 186 testing for update user profile success.

Testing for View profile of users:

Objective	To view users' profile and their achievements by searching them from application.
Action	User enters "Spandan Bhandari" in the search bar and clicks on it.
Expected Result	Should show profile of "Spandan Bhandari" including this full details and achievements.
Actual Result	Profile of Spandan is shown with his all achievements and details.
Conclusion	The test was successful.

Table 34 testing for view user profile.

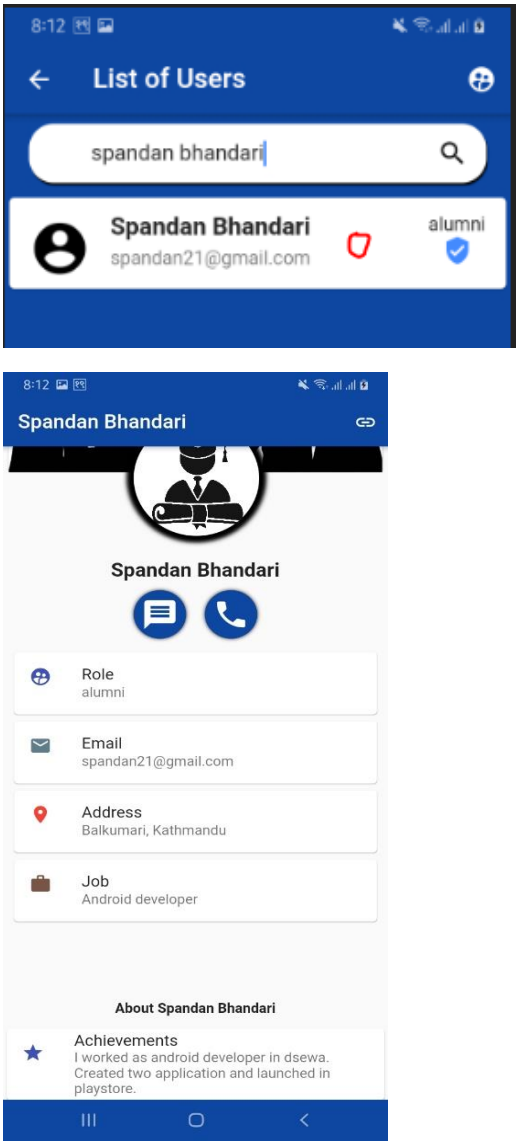


Figure 187 testing for view user profile success

Testing for view event details:

Objective	To view details of events by searching from application.
Action	User enters “Christmas” in the event search bar and clicks on it.
Expected Result	Should shows Christmas events and its details.
Actual Result	Events name Christmas and its detail was shown in different page.
Conclusion	The test was successful.

Table 35 testing for search and view events.

Figure 188 testing for search and view events.

Testing for adding Events validation:

Objective	To create a new event by the college.
Action	College clicks on Add events button from his profile and press create events without adding data on it.
Expected Result	System should validate the events form and show error message.
Actual Result	An error message is shown as “All fields are required to add new events.”
Conclusion	The test was successful.

Table 36 testing for adding new events error.

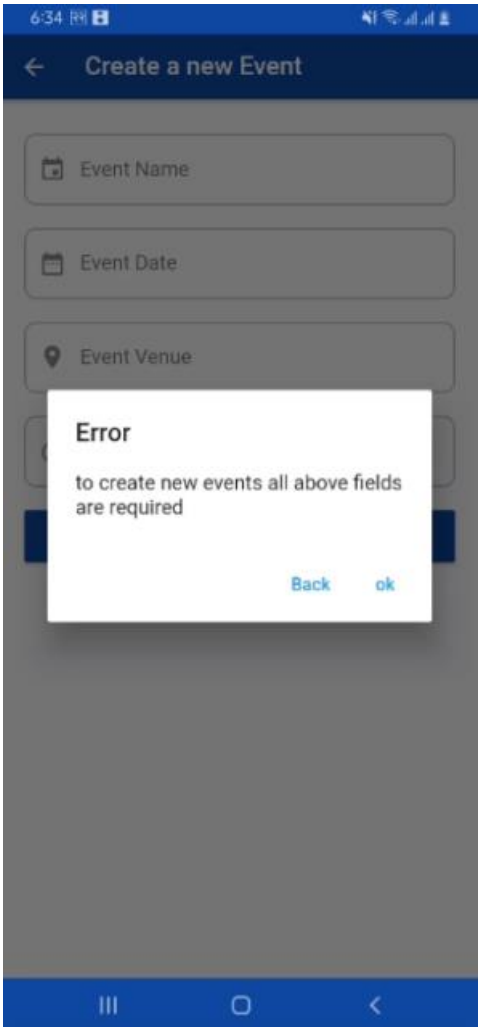


Figure 189 testing for add new events error.

Unit testing Laravel http URL for web register success:

Objective	To make an URL testing on http web register route.
Expected Result	Should check the entered URL from the testing function and show the test results.
Actual Result	Here the "/register" URL was a valid URL for web register, and the status was 200 so the test was successful.
Conclusion	The test was successful.

Table 37 unit testing of Laravel http register URL.

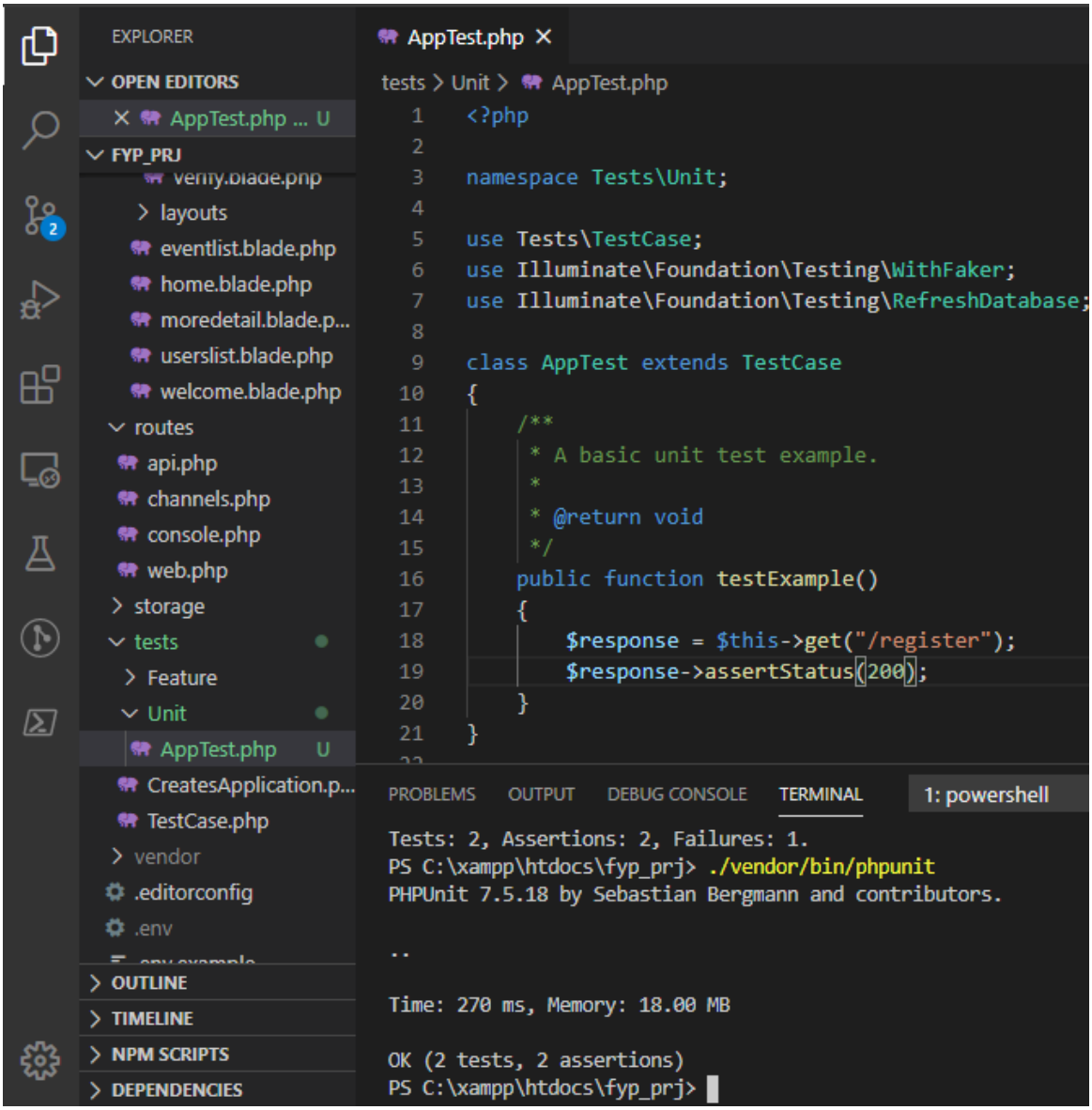


Figure 190 unit testing Laravel web URL register user.

Logout user testing:

Objective	To logout user from the application.
Action	User press logout button from navigation bar and clicks on ok.
Expected Result	System should logout user from the application and returns user to login page.
Actual Result	A user is logged out successfully and redirected to login page of application.
Conclusion	The test was successful.

Table 38 testing for logout user.

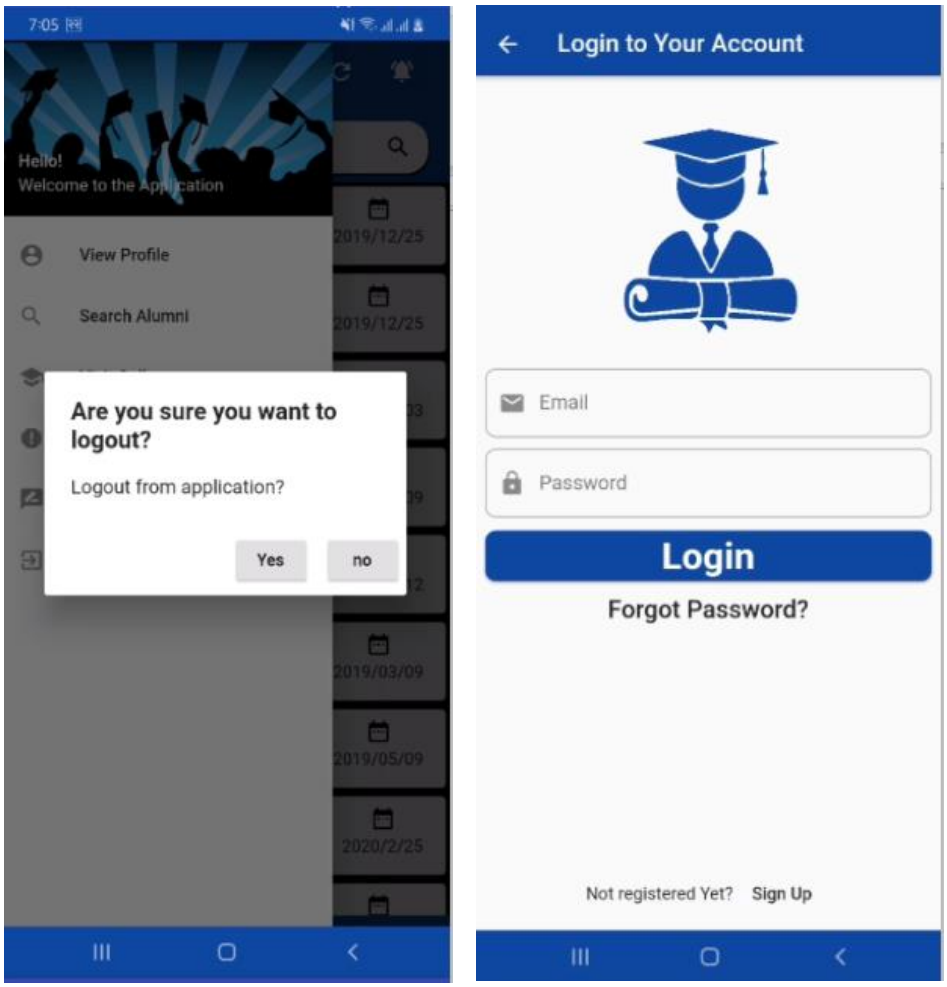


Figure 191 testing for logout user success.

Testing for Admin web Panel (Connecting local host):

Objective	To connect the application with local server and read MySQL database to web and mobile application.
Action	User has hosted the application to the local server and connection was made.
Expected Result	The application should be connected to the local server and shows the data.
Actual Result	The application is running in the local server and data of application was shown.
Conclusion	The test was successful.

Table 39 testing for connect web application to server success.

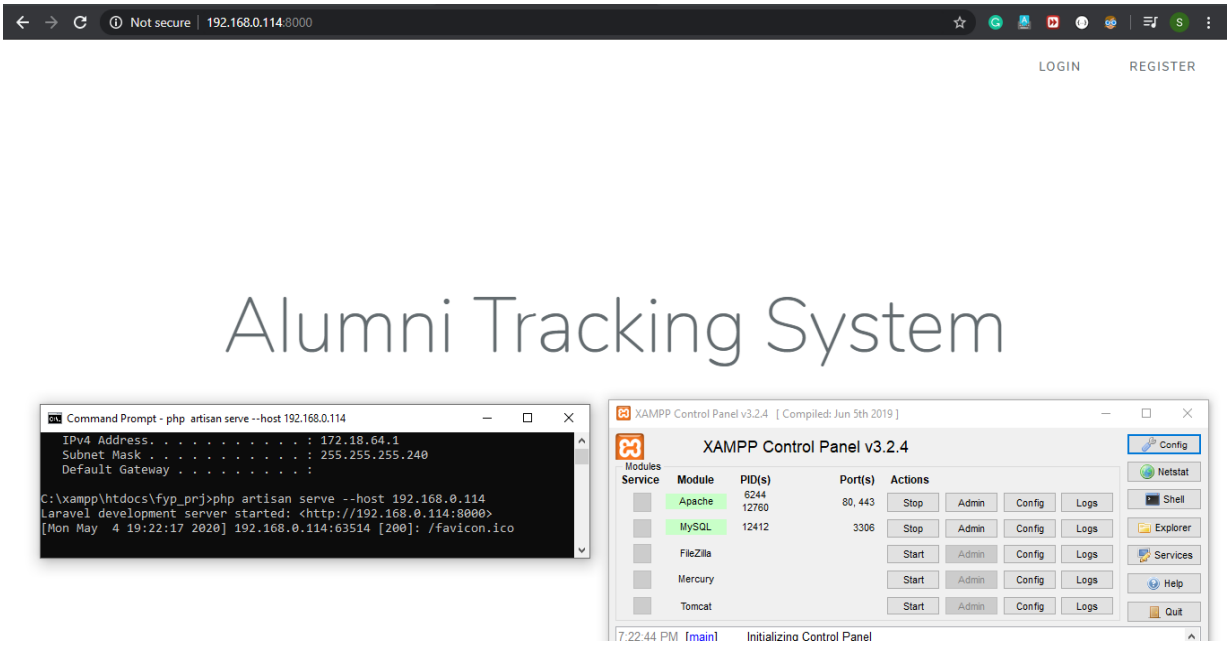


Figure 192 testing for server configuration success.