UNIVERSITY OF LEICESTER

Department of Informatics
University of Leicester
CO7201 Individual Project

Report

Web Application for Allocating Groups, Topics, and Supervisors for Group Discussions

Harshith Aiyannira Ganesh

hag5@student.le.ac.uk

219045972

Project Supervisor: Maryam Doostani

Second Marker: Dr Paula Severi

19/05/2023

(Word count 10314 excluding table of content, references, and appendix)

DECLARATION
All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Harshith Aiyannira Ganesh
Date: 19/05/2023

Abstract

As a fresher finding course mates with the same area of interest is one of the challenging tasks you face once you start university. Although the student may have friends, but their areas of interest would not coincide. Depending on the student, the area of interest might vary. Finding students with the same area of interest and forming groups for their topics would become problematic and time-consuming for the supervisors as a result. Similarly, as a fresher finding course mates with similar interests and organising groups would be difficult, and you might not find one.

Currently, the group allocation technique is established by letting students choose group members among themselves, later the groups choices would be submitted by each group according to the topics available.
However, the newly formed group would include students with varying opinions or areas of interest, which can cause conflict when the group would submit their preferences. Similarly, it would be challenging for the supervisor to identify students with common interest and organise them into a group. Even if the supervisor can put a group together later, it might not produce a satisfactory result in the coursework because students may have some inter-conflict owing to the different areas of interest.

As a result, the goal of this Web application is to give a graphical interface to students and supervisors, where the supervisor uploads the topic that will be supervised, and the students will enter their choices in the topic they would like to work. An algorithm would be developed to assist students and academic staff in group formation based on students' interest. This would be managed by the administrator.

This platform would make all tasks more fluid and faster, as well as automate the process with less human intervention. This automated function will save lot of time and effort of academic staffs and students by reducing the manual procedure required and it would be more probable to create a group with similar interests.

Contents

1. Introduction

1.1 Aims and Objectives

The aim and the objective of the project is to develop a website-based Application that assists academic staffs and students. The application is developed to automate the process of allocation of students and supervisors into groups for “Personal and Group Skill” course module, by allowing the supervisors to specify their topics that they are willing to supervise. Once the supervisor specifies the topic, the students are allowed to select four topic preference that they prefer to work on based on their interest and knowledge.

Administrator manages the allocation process. The application is designed in a way that administrator is allowed to view and edit the preferences given by the students. Administrator runs the algorithm to distribute the students and supervisors into a group, views the outcome of the algorithm (group allocation), and make the allocation visible to relevant users if it looks better.

The group that would be established using the algorithm would contain the students with the similar areas of interest. When several perspectives on the same topic are combined, the final product of the coursework will be more instructive and informative.

Once the outcome is made visible by the administrator for the relevant users, then students and supervisors would receive an email with the related details of the groups they have been allocated to, and additionally the allocation details would be displayed in the application for the students and the supervisors once they login to the application.

The overall goal of the website-based application is to improve accessibility and efficiency of the group allocation process for the academic staffs and students and make this process seamless.

1.2 Challenges

Developing an algorithm that distributes students into groups of four or five or group size specified by the administrator depending on the topics that they have given based on their interest is a challenge. Each student should ideally be paired with their first choice of topic via the algorithm, followed by their second choice, and so on. According to preferences listed by the students and the topics published by the supervisor; a group should be created. There should be four or five students in each group, or the group number specified by the administrator; no group should include more than specified number of students. One student should receive at most one topic, not necessarily all the student's top choices. If the students haven't given any preferences, then choose a group at random for them to be placed in. Notify the group details to the students and the supervisors through an email.

As this application is seeking an ideal algorithm, researching algorithm, and using the pros of the algorithm while bringing up the modifications in the algorithm as needed has become a difficult chore.

1.3 Literature Survey and Background Research

1.3.1 A Web-based System for Student-project Allocation

For all academic institutions, the software for assigning specific topic for the assignment for the students must be developed based on their interest. One of the reasons for suggesting an automated project assignment system is to prevent heated debates and disputes amongst students, especially when it comes to the assignment of tasks. (Jean-Pierre Gerval, 2020)

There are several phases during this process which has been utilized with few enhancements -

Phase 1:
Users can register themselves to the application. Assignment content must be proposed by the professor. Students could suggest the content that they are interested in. If the professor approves the content, he would be incharge of the content. Supervisor could edit the proposed topic and delete the topic. Administrator could remove or edit the topic. Administrator could register users or remove and monitor to see if all the professors have proposed topic where the group of students could be accommodated. Students' strength and topic required should be decided by the admin. (Jean-Pierre Gerval, 2020)

Phase 2:
Don't allow users to register. Removing topics should not be allowed. Students need to go through the topics proposed by the professors. (Jean-Pierre Gerval, 2020)

Phase 3:
Students are allowed to submit their preferences based on the topics proposed. Admin could view the preferences, edit it or set preferences on behalf of students. (Jean-Pierre Gerval, 2020)

Phase 4:

> Allocate the student based on the preferences they choose. Admin would allocate the student based on algorithm. If the result seems good, inform it to the students and supervisor. (Jean-Pierre Gerval, 2020)

These phases in the paper have considered the students getting allocated to the individual projects based on their preferences and few more work evaluations are taken up.

Above phases are utilized by the author to develop web app where the students with the similar areas of interest are put together where the group size would be 4 instead of individual allocation. Each topic proposed by the supervisor would get allocated to multiple students and these students are placed in a single group.

In the paper, in case a student can't suggest an individual topic, he will be allotted to a random project. The similar principle is used in the developed web app where the students who haven't set their preferences would be randomly assigned to a topic.

1.3.2 Final Year Project Allocation System Techniques

Integer Programming Algorithm: Students can rate the project using this approach by their preferences. If possible, the student will be given their first choices of projects. (Nik Intan Syahiddatul Ilani Jailani Faculty of Computing, Ali, & Ngah, 2022)

Linear-Time Algorithm: This algorithm considers the preferences of the professors and students. It enables lecturers to provide their preferences over the students and students should provide their individual preferences over available project topics when assigning tasks to students. (Nik Intan Syahiddatul Ilani Jailani Faculty of Computing, Ali, & Ngah, 2022)

Similarly, there are a few algorithms such as Hybrid Fuzzy Evolutionary Algorithm, Approximation Algorithm, Genetic Algorithm which works on the similar principle (Nik Intan Syahiddatul Ilani Jailani Faculty of Computing, Ali, & Ngah, 2022)

These algorithms are the few ones which I had referred in the allocation process of students and supervisors into the groups based on topic available and the preference of students on the topics. Based on the research in section 1.3.1, the author considers students' preference and the topics uploaded by the supervisor as primary inputs to the algorithm and I have incorporated the two algorithms.

1.3.3 iJRASET

As the project is based on developing a full stack application, it would require choosing a better technology for the speedy development of the software. The market offers a wide variety of full-stack development frameworks, such as the LAMP stack, MEAN stack, and MERN stack. One of the most popular web stacks is MERN. An open-source JavaScript library - React.js, JavaScript and JSX (JavaScript XML) are used. They offer better productivity with proper documentation. React.js enables us to construct any sophisticated application out of basic components. (Baiskar, 2022)

Since the application would be handling large set of data, mongo db, a NoSQL database, is used. This database will store the data in Json format where the data in the application would be flowing in Json from frontend to backend making application debugging easier. Even though the stack is new to the author, the author has chosen to develop the software since it takes less time to develop a feature in an efficient way with a booming technology. (Baiskar, 2022)

1.4 Risks

Risk 1: Learning technology called MERN stack which is new to the author and developing all the features would be a tedious task. The software development may be hampered by the necessity for further study before implementation, which would take more time.

Risk 2: To notify the users through the email in MERN stack nodemailer could be used. Since the google mail have disable to enable the less secure apps it would require using other mailing service or create a developer account.

Risk 3: Developing an algorithm which provides effective results with less time complexity might require research and more time spent on it might hamper the time plan.

Risk 4: Author is new to mongodb NoSQL database. To build good webapp it might require some knowledge on mongodb atlas.

Risk 5: Since react 18.2.0v, which has a materialUI problem that the developers are yet to fix. To use the functionality in the application development, one would thus need to discover some other libraries for the application or forcibly install the same and utilise the feature in the application development.

2. Requirements

2.1 High-Level Requirement

This application, there are three users: an administrator, a student, and a supervisor.

Administrator:

Once the administrator logs into the application he would be directed to the administrator's dashboard. Administrator handles the allocation process and runs the algorithm which distributes the students and the supervisor to the groups. Once the students and supervisors are distributed to the group and the allocation seems to be fine then the administrator will intimate the users through the email. The application also allows administrators to notify all these events through email.

Supervisor:

Once the supervisor logs into the application, he would be directed to the supervisor's dashboard. Supervisors can upload the subject matter in which they are knowledgeable or specify any topic which they are willing to supervise. Once the algorithm is run and the group information is made visible by the administrator. Details about the supervisor's group would be available in the application.

Students:

When the students log into the application, they would be directed to the student's dashboard. The topics uploaded by the supervisors are made visible to the students who

have registered in the application. The students are allowed to provide four preferences of their choices based on their interest and knowledge. Once the algorithm is run and the group information is made visible by the administrator, details about the student's group would be available in the application.

2.2 Detailed Requirement

2.2.1 Essential

- Users such as students and academic staffs would be provided with form in the web application to register themselves in the application where they need to input some of the basic information and get registered themselves.

- Users such as Administrator, student, and staff should have access to the application based on their user-type. Login as a student, supervisor, or administrator and should be able to perform necessary actions available for that user.

- The application would lead users to their dashboards based on the user credentials they provide during the login (for example, students would be directed to the student's dashboard, supervisors to the supervisor's dashboard, and administrator to the administrator dashboard).

- A straightforward database needs to be developed to store the required details, such as user details, preferences given by the students, and list of the topics uploaded by the supervisors (with title, description, and code).

- Design an algorithm that distributes the supervisors and students into the groups based on the topic specified by the supervisors and four preferences provided by the students based on their interest and knowledge.

    - Each student should ideally be assigned according to their first preference, then their second preference, and so on.

    - The group should include at least four to five students (ideally, there should be four students in every group and not more).

    - The students who haven't given their preferences should be randomly distributed to a topic into a group.

    - Students who have recently update their preferences would have a lower chance of receiving their desired topic because the algorithm would be designed based on the first come, first serve principle.

- Administrator, can perform the following:

    - Administrator should add, remove, and update details of the students and supervisors.

- Administrators can view, delete, and edit topics that supervisors have published and are willing to supervise.
- Administrator should view the preference which are submitted by the students and edit the same if needed.
- Administrator handles the allocation process. Execution of the algorithm (Allocation of students and supervisors to the group).
- After executing the algorithm, the administrator needs to check the allocation results and should be able to clear the current allocation and re-execute is the allocation if isn't as expected.
- If the administrator approves the allocation, it needs to be made available to the students and supervisors, or maybe even intimate through an email with the group information.

- Students, can perform the following:
  - Students should view topics that supervisors have published with the topic's details.
  - Students who are entitled to the course module should give four topics as their preference so that they decide which ones they prefer to work on based on their interests.
  - The group details should be accessible after student's login to the web application or receive an email notification after the administrator has run the algorithm and made it visible to the students.

- Supervisors, can perform the following:
  - Supervisors should publish the topic which they are knowledgeable or specify any topic which they are willing to supervise.
  - Supervisors are only allowed to update and remove topics that they have published. If the subject does not comply with the supervisor, editing and deleting should be disabled for the other topics.
  - The group details should be accessible after supervisor’s login to the web application or receive an email notification after the administrator has run the algorithm and made it visible to the supervisors.

- A Navigation bar to traverse the application should be provided to all users.
- The users should be able to reset their password using the forgotten password form when the user is created by the administrator.

2.2.2 Recommended

- The ability to edit and update one's profile information should be available to administrators, students, and supervisors. Only few details may be edited.
- A list view of the students and supervisors should be available to the administrator.
- The authority to activate or deactivate users account should rest with the administrator.
- The application would be recommended to using single sign-in and sign-up page and route to their dashboard based on the credentials used by the user to log into the application.
- The user such as students and supervisors would receive an email notification once the after the administrator has run the algorithm and made it visible to the users.
- A user's dashboard should display every announcement issued by the administrator for that user (e.g., students should see announcement made for students, and supervisors should see announcement made for supervisors).
- The user can easily explore the web application through a well-designed user interface.

2.2.3 Optional

- All Events should be reported by the administrator, such as announcement to all application users via email. (student/supervisors).
- Search the topic in the application for the students/ supervisor and administrator.
- Students who aren't given any topic will be sent an autogenerated email to contact the course Convenor for further aid.
- Students are allowed to provide their own topic and if any supervisor would be willing to supervise it gets published to the topic table.
- Administrator can search students and supervisors.

3. Technical Specification

Here the Web application uses JavaScript stack known as MERN Stack to develop a full-stack web application. The four technologies used in this stack are MongoDB, Express, React, and Node.js. The technologies, frameworks, and other libraries with their versions used in the development of the application are displayed in the table (Table 3.1 - Technical Specification).

With MERN stack, an application is developed, react as the client-side of the application which is currently popular in the market is maintained by META company. Mongodb atlas

is the cloud storage system which is NoSQL database which stores the data in the key value pair like Json. JavaScript is the Programming language has been used by the author throughout the application development.

Type	Technology	Version
Programming Language	JavaScript	ECMAScript6
	HTML	5
Frameworks and libraries	npm	v9.6.2
	Nodejs	v19.6.0
	React	v18.2.0
	Express	v4.18.2
	MaterialUI	v4.11.3
	FortAwesome	v6.3.0
	React Prime	v9.2.1
	Bootstrap	v2.7.2
Operating System	Macintosh	Venture 13.3.1
	windows	11
	Linux	Ubuntu 22.04.2 LTS
Database	MongoDB Atlas	6
IDE	Visual Studio Code	v1.7.4
Other technology and software	Amazon SES (Simple Email Service)	01/12/2010
	Postman(services)	v10.5.8
Application Management tool	Atlassian Trello	NA

Table 3.1 – Technical specifications

4. System design and Architecture

4.1 System Architecture

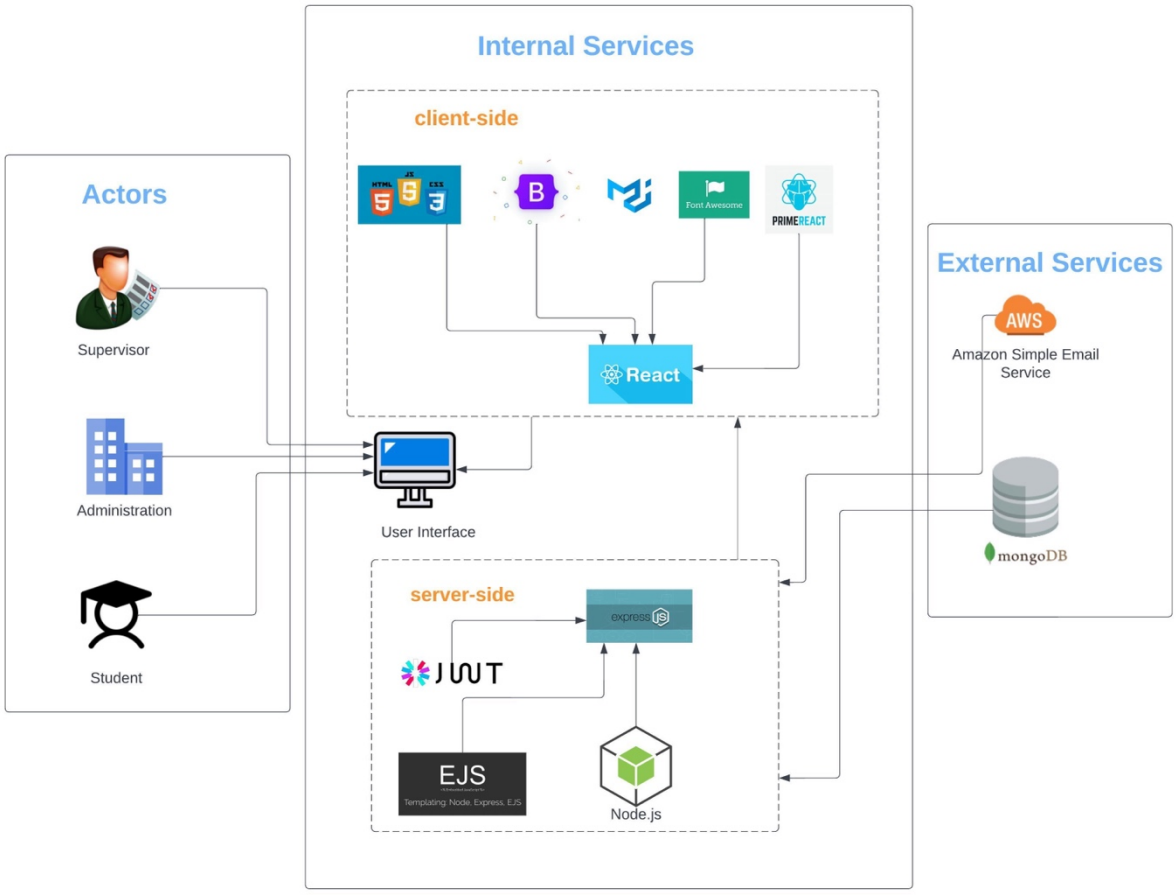


Figure 4.1.1 – System Architecture

Actors, internal services, and external services are the three main elements of the application architecture, as depicted in Figure 4.1.1 of the application's system architecture above.

Actors are the users of the application who are Administrator, Supervisors, and the students they would access the user-interface to engage with the application. Based on the role of the users, these actors are given access to their relevant functionalities.

With the aid of external services, such as the database mongo DB atlas used to store all the application-relevant data in the cloud and the Amazon web service cloud platform with feature such as Simple email service have been utilised in the application to notify the events through emails which is connected to the internal service, the internal services make up the client-side and the server-side of the application.
In the server-side of the application, express framework is used to build RESTful APIs and it provides several features for web applications development, with NodeJS, an open-source

server that serves as the back-end runtime environment, is part of the application's server-side architecture. The server-side templating is done using embedded JavaScript which use plain JavaScript which generates HTML markup.
JSON Web Token is used to communicate data securely between two parties as a JSON object. This information is digitally signed, making it more secure when transferred between two parties.

Reactjs is used to design the user interface, which is based on the components listed in the software requirement specification, allowing all the components to operate together. A few more user interfaces enhancing component libraries, such as materialUI, reactPrime, bootstrap, and fortawesome, have been utilised to enhance the user experience in the application and speed up development. Using the component approach, react integrates HTML and CSS and elevates them to a new level.

4.2 Layered Architecture

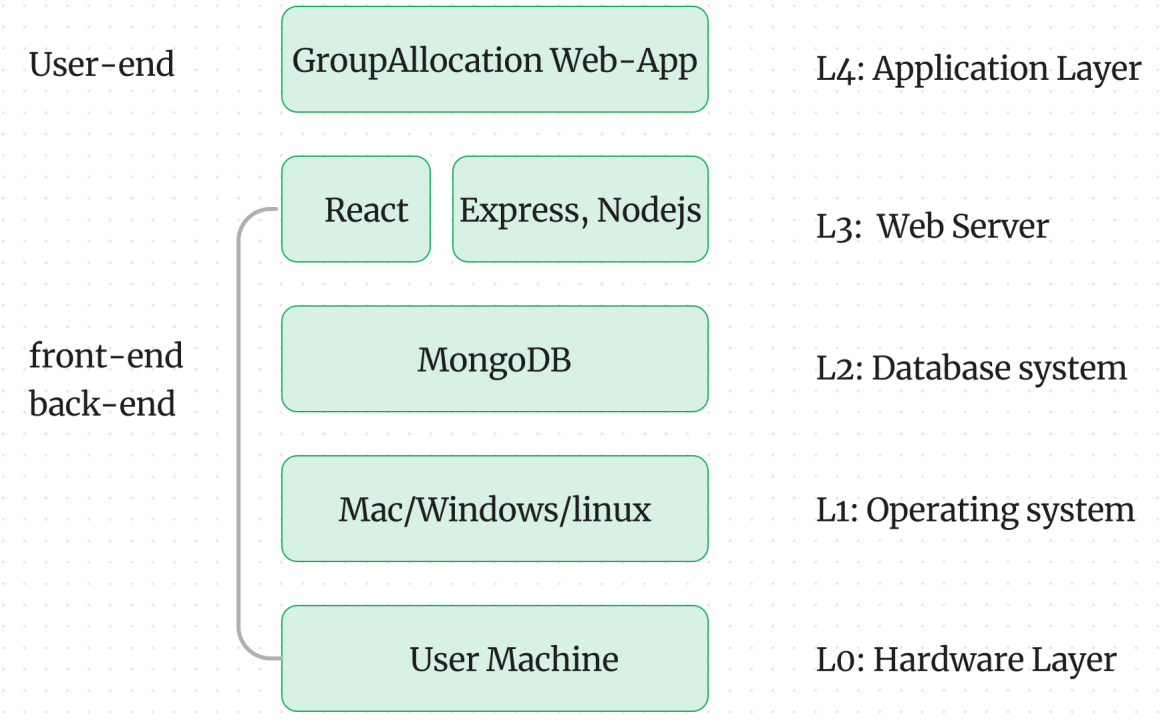


Figure 4.2.1 – Layered Architecture

The Layered Architecture diagram (Figure *4.2.1* – Layered Architecture) shows five layers, L0 – L4. The user machine's hardware layer, layer L0, and the hardware device's operating system, layer L1, are which allow the user and the software to communicate with one another.

The Layer L2 is the database in the system that stores all the application's necessary information; the data may be user information, topic that the supervisor uploaded to supervise, or the preference of the students mentioned etc.

Client-side of the application is developed with the React. It is front-end JavaScript library that develops client-side user interfaces. It may be used to create server-rendered or single-page applications.
Server-side of the application is developed with express framework and NodeJS environment.
Express is back-end framework for Nodejs, instead of using NodeJS to create different nodes by using express, it makes the code in the back-end easier. Express supports many libraries which intern make the API to be developed in an seamless way (geeksforgeeks, n.d.).
The layer L4 is the application layer. This application can help the users such as students, supervisor and the administration which make the allocation process seamless by minimising the manual process needed.

4.3 Use case Diagram

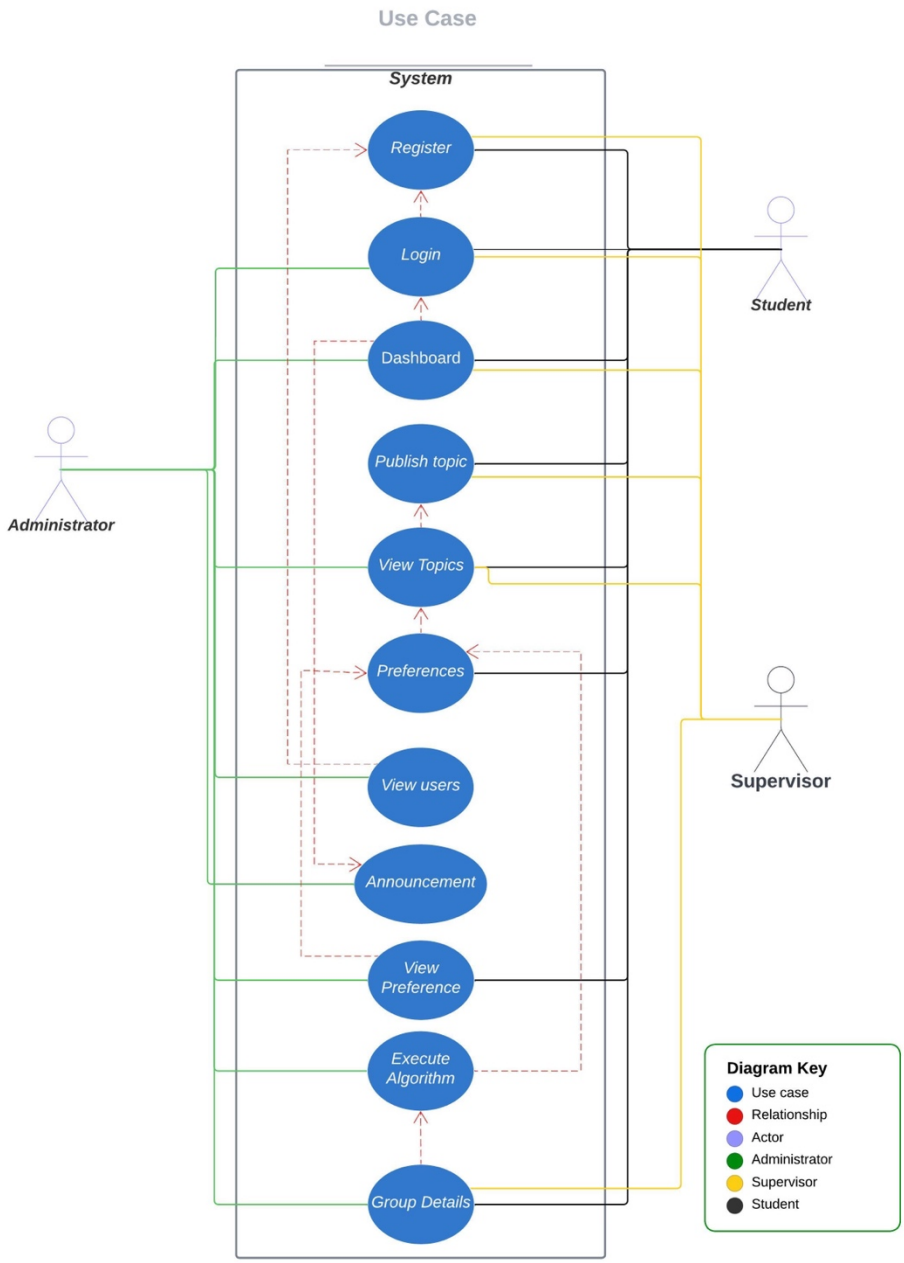


Figure 4.3.1 – Use Case Diagram

The (Figure *4.3.1* – Use Case Diagram) visualizes the high-level overview of the functionality of the web application.
This application, there are three users (actors): an administrator, a student, and a supervisor.

List of main functionalities of the application developed are,

- Users of the application, such as students and supervisors, can register themselves by filling out a form and by inputting a few basic information. This allows them to become registered users of the application.
- The system will authenticate user credentials based on those generated during registration, after which it directs users to the appropriate dashboards. (For example, Students would be directed to the student's dashboard, supervisors to the supervisor's dashboard, and administrator to the administrator dashboard).
- Any announcements conveyed by the administrator to that specific user will appear in their dashboards. (For example, Students would be seeing student's announcement, supervisors to the supervisors).
- Supervisors should specify any topic they are willing to supervise or upload the topic in which they are knowledgeable and expertise in.
- All users will have access to the topics that are uploaded by the supervisor.
- The students should choose four topics as their choice of preference based on the topics that have been posted.
- Once the choice is made, the administrator runs the algorithm, which divides the students into groups according to the choice, supervisor to the topic they are supervising and assign topics in a random manner to the students who haven't made a choice.
- The application will display the group details once the administration has made the allocation public.

4.4 Sequence Diagram

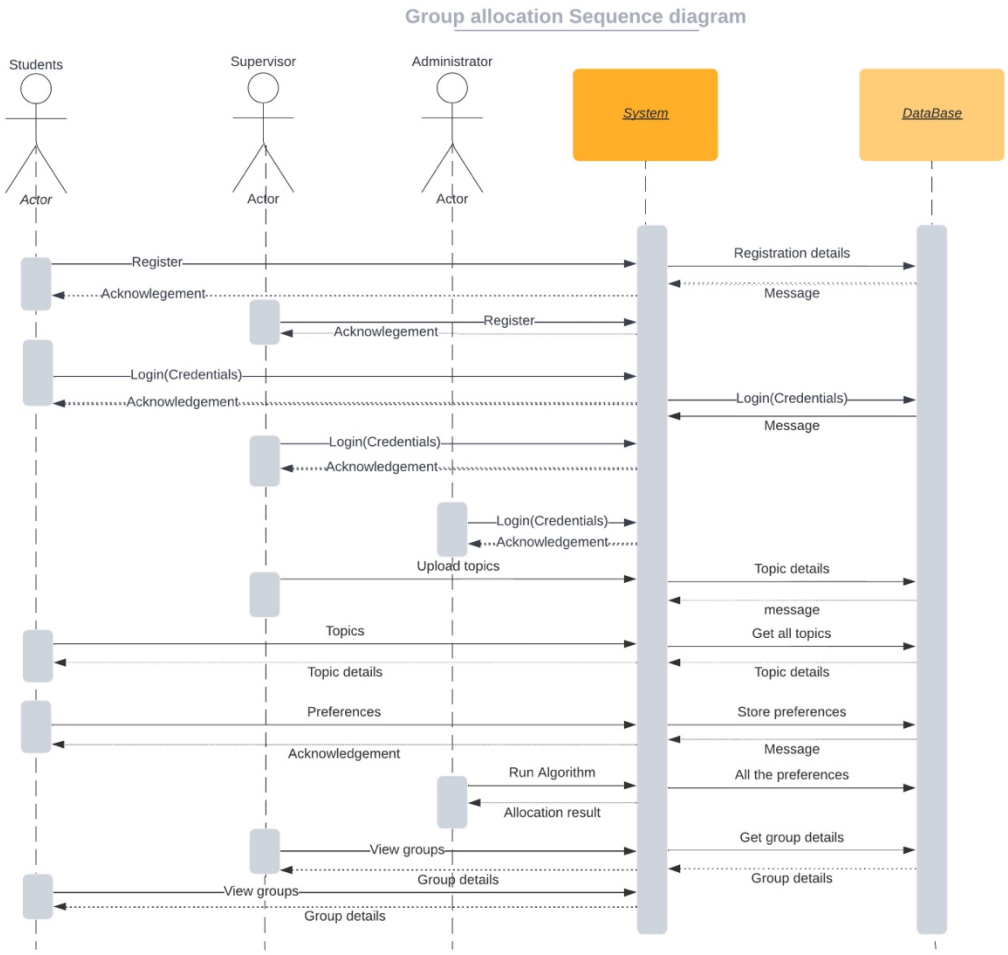


Figure 4.4.1 – Sequence Diagram

The (Figure 4.4.1 – Sequence Diagram) visualizes the high-level overview of the functionality data flow in the web application.

The user is allowed to register themselves for the application. Once registered, the user enters the credentials that are registered, and the application's backend verifies that the entered credentials are valid. If they are, the user gets directed to their dashboard.
Depending on their user-type, users like administrators, students, and supervisor should have access to the application functionalities. If you log in as a student, supervisor, or administrator, you should be able to access all the essential activities supporting to that user.

The supervisor uploads the topics once they receive notification from the administrator to do before the deadline. Then that specific topic details would get stored in the database. Students may select from the same topics that the supervisors had posted and then submit their preferences, according to which topics they are most interested in. These preferences would then be stored in the database and used as input by an algorithm that would allocate the students and supervisors into groups. When the preference deadline is reached, the administrator runs the algorithm. If the results appear to be satisfactory, the administrator

makes the results visible to all users. Now all the users would be able to view the allocation results.

4.5 User Activity Diagram

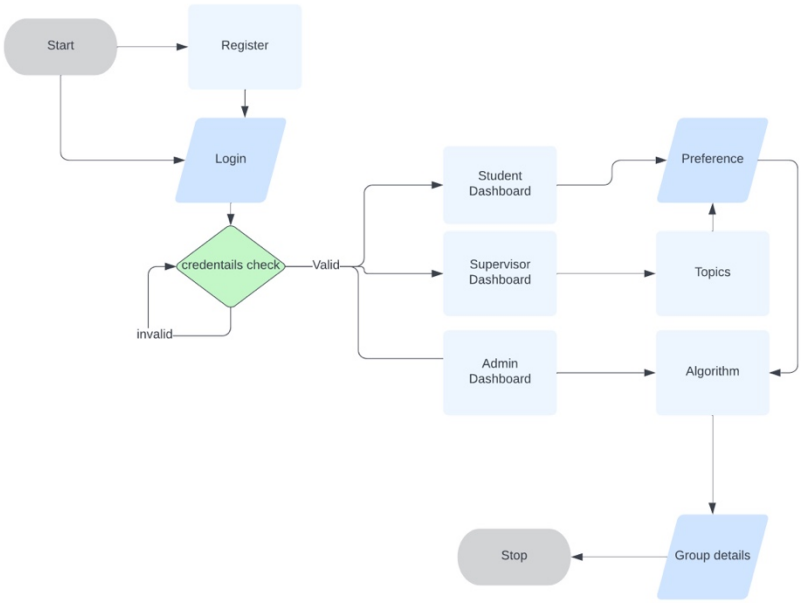


Figure 4.5.1 – Activity diagram

The activity diagram in Figure 4.5.1 above, which shows the overall flow of the key features of the application and visualises the flow of application for different users.
Once the user registers himself to the application with the generated credentials user attempts to log into the application based on the credentials entered, they are sent to the user's dashboard with the functions made accessible to that user. Further rest of the functionalities which is shown in the (*Figure 4.3.1 – Use Case Diagram)* would be carried out.

5. Algorithm Analysis

5.1 Algorithm with N^2 complexity based on the topic preferred.

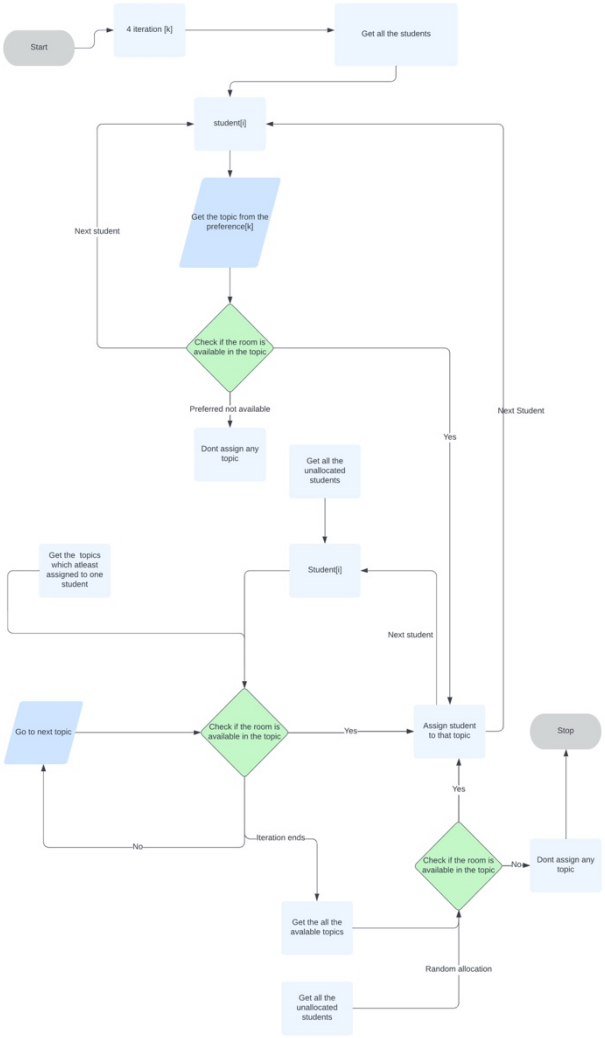


Figure 5.1.1 – Algorithm with N^2 time complexity

The flowchart above (Figure 5.1.1 - Algorithm Flowchart) depicts the method used in the software to divide students and supervisors into groups where the algorithm's time complexity is N^2. A list of students is obtained, and this list is iterated for each student since each student has four preferences. In the first iteration, the student's first preference is taken into consideration; if the topic has a room in it, the student gets the topic; otherwise, it moves on to the next student's first preferred topic. If after four iterations the students still doesn't get their preferred topic, then do not allocate them to any topic.

The database would then be used to extract a list of students who have not yet been allocated to any topics while topics having at least one student would be gathered from the database and students would then be randomly assigned to topics. Still, if any students aren't allocated to a topic, the programme will at random assign them to one of the available topics.

The groups which have been formed with this algorithm would be the students with similar interest and this allocation is based on the weight of each students' preference. As n increases, the quadratic term will be dominated by this, which has a time complexity of $O(4n) + O(n^2) + O(n^2)$ will be $O(n^2)$.
However, this algorithm makes sure the goup has a specified number of students in it. Until there is no more topic where the students need to place in the group.

5.2 Algorithm with N^2 complexity based on first come first serve.

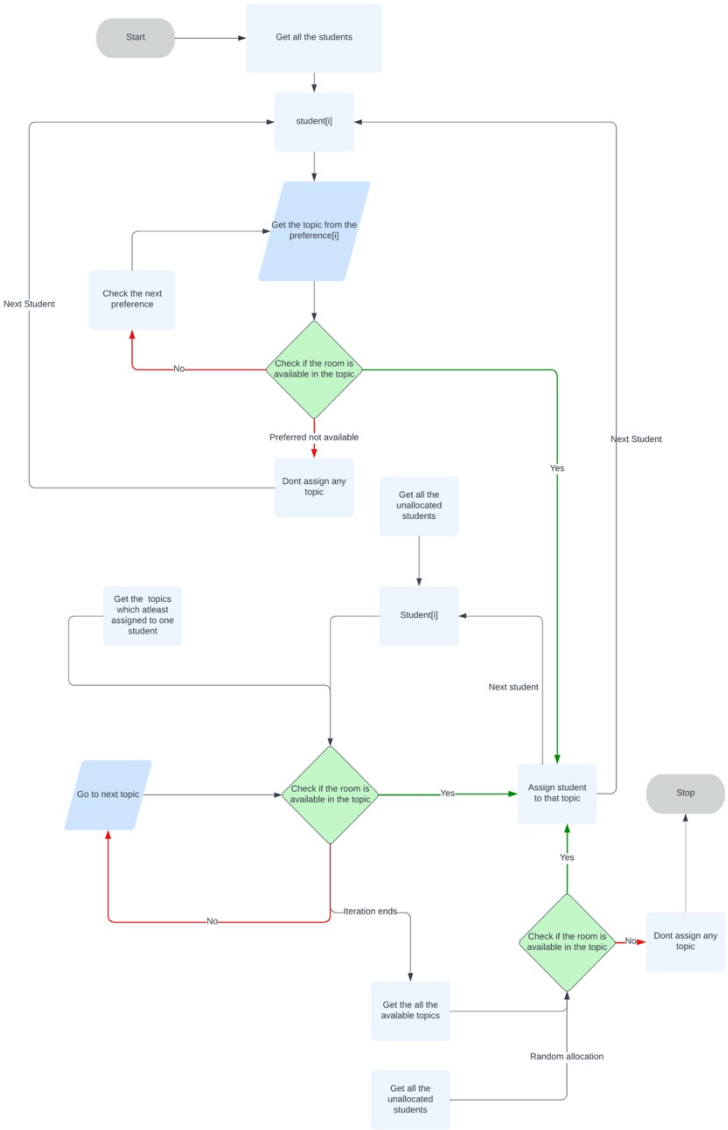


Figure 5.2.1 – Algorithm with N^3 time complexity

The flowchart above (Figure 5.1.2 - Algorithm Flowchart) depicts the method used in the software to divide students and supervisors into groups where the algorithm time complexity is N^2.
All students who have enrolled for the application are first taken into consideration, together with the preferences listed by each student. If the desired topic is available after going through the list of students, the student is assigned to it; if not, look for the next preference. This procedure is repeated; if none of the student's preferences are accessible in the first iteration, the program will not assign that student to any topics.

Now, only topics having at least one student assigned in them and room for more students are retrieved from the updated database, allocate students to the topic if the room is available; otherwise, check for the next topic. This ensures that the group would be full for the topics where the students are distributed based on the choices in the first iteration.

In the last iteration, gather the students who haven't been assigned to any topics and only choose the topics that still have space for students. Then, distribute each student at random by going through each topic at a time.

A group of students with a similar field of interest and a supervisor assigned to it would now be the algorithm's output. However, this method will ensure that every group is full, until there are no more students to distribute into groups.
The algorithm's time complexity is $O(n^2)$. This is because there are two nested loops with n iterations each. For every time the outer loop iterates, the inner loop iterates n times, totalling n^2 iterations.

For each iteration, the nested loop executes multiple O (1) (process block as indicated) operation. Consequently, the total time complexity would be $n^2 * O(1) = O(n^2)$.

6. Methodology

6.1 Software Development Approach

Agile approach has been followed in the software development life cycle process. It is the methodology where the project is split into distinct phases where these distinct phases are developed and tested in an incremental manner. (Atlassian, 2023).

Initially, all the major requirements of the application that need to be developed were gathered, and a plan was developed in a linear sequential way based on the priorities of the software requirement specification (SRS). From the software Requirement specification, the requirements (SRS) were broken into several sprints based on priority. Once the sprint was planned, sprint feature would be developed and tested.

This developed feature was demonstrated to the supervisor and the feedback was gathered. In the next planned sprint these improvements were added and developed. During this instance, the supervisor acted as a scrum master and the author as developer and tester in the application development.

The application life cycle management is taken care by the application Atlassian Trello. The information regarding the sprint plan and the task updates are available in this link.

Link to the Scrum task board:

https://trello.com/b/VDpP6kJJ/msc-project-groupallocationsystem

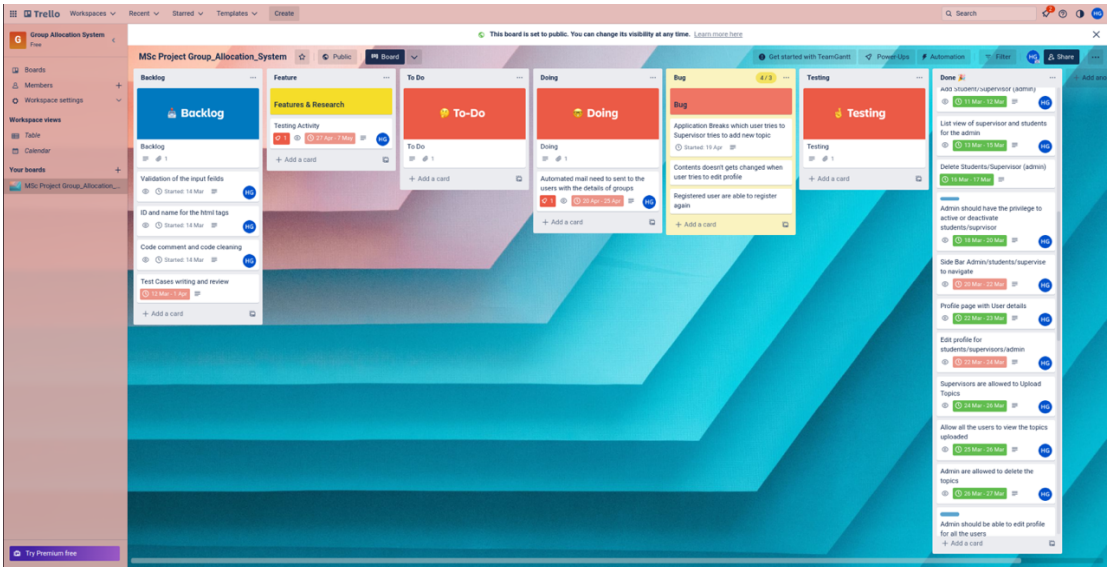


Figure 6.1.1 – Scrum task board

6.2 Client-Server Architecture

The Web Application is developed using Mongodb, Express, ReactJs and NodeJs which is also called as MERN Stack which is the JavaScript stack used for the development and deployment of full stack application (Baiskar, 2022).
All the files related to the web application is in the folder called branches where client-side, and the server-side of the application are maintained.

Client-side refers to the application's front-end, which is built with the React library and the node package manager.

The server-side is the application's backend, which is built with the express framework and the NodeJS runtime environment.

For data storage, a NoSQL database named mongodb has been used, which stores all application-related data in tabular format depending on the schema that has been defined.

Simple email service, an Amazon webservice functionality, has been used to notify application users through email.

- MongoDB Cross-platform document-oriented database which is NoSQL database records each document in a key value pair as JSON. Because of its adaptability, MongoDB enables its users to design schema, databases, tables, etc. Documents which are stored in the MongoDB are identified by its primary key (geeksforgeeks, n.d.).
- Express is back-end framework for Nodejs. Instead of using NodeJS to create different nodes by using express, it makes the code in the back-end easier. Express supports many libraries making the API to be developed in an effortless way (geeksforgeeks, n.d.).
- React is a JavaScript library for front-End for building user-interface(client-side). It can be used to develop single-page or server-rendered application (Baiskar, 2022).
- Node.js is the runtime environment which allows to run the code in the server-side. Node package manager provides the developer to install the packages required and helps in developing easier and the shorter code (geeksforgeeks, n.d.).
- Without an onsite Simple Mail Transfer Protocol (SMTP) infrastructure, you may reliably communicate with clients using Amazon Simple Email Service (SES) (Amazon web service, n.d.).

6.3 Milestone

The application life cycle management platform, Atlassian Trello, has been utilised throughout the project to plan each weekly sprint making use of the specific requirements that were previously acquired before the start of the project.
The table (Table – 6.3.1 Milestone) gives the brief information regarding the software development life cycle. Feature development detail is given in the task details section in the table for the related modules.

Module	Duration	Task Details
Research and Software Requirement Specification (SRS)	1-2 week	1.Collect the software requirements required for developing the application, such as software requirement specifications, research components, tools, and plugins. 2. Learn MERN Stack 3.Preliminary Report
Environment Setup and server-side	3-4 week	1.Create a functioning environment in the MERN stack using Visual Studio to run the client and server sides on the local system. MongoDB for the database, react for the client, and nodejs for the server. 2. Create sprint plan and user stories. 3. Build API endpoint for registration and login 4.Mongo Database schema to store admin, student, supervisor details 5.Build different routes for each user so that they may be easily differentiated. (All of the necessary API is written within the appropriate user routes.) 6. Route user to homepage based on their login credentials. Using the user's login information, direct them to their homepage based on the email ID they login with. 7.JWT secret authentication and user logout. Create a logout method for all users and utilise JWT tokens to authenticate user login

Client-side and server-side	5-6 week	1.The admin must be given a form to register the students and the supervisor. 2. Get a list of the supervisors and students who are registered with the application. 3.Administrators should have the authority to remove users like supervisors and students. 4.The feature to allow or deny a user's ability to log in to an application should belong to the administrator. Schema changes is required for the students and supervisor (Add status in the schema). 5.Build a side bar to allow users to click through the Web application's tabs. 6.Depending on the user login, retrieve the user's information, authenticate the user with a token, and show the user's information in the profile page. 7.Enable users to change their profile information for a few fields. 8.Create a schema to record topic details and allow the supervisor to enter the topic title and description. 9. Students, supervisors, and administrators should be able to access the topic that the supervisor has posted.

Client-side and server-side	7-8 week	1.Enable the admin to remove the subjects that the supervisors have uploaded. 2. Enable Admin to modify users' profile information; for a few fields 3.If a supervisor creates a subject, the application should only allow that supervisor to remove the topic he has created. Other supervisors cannot delete topics posted by this supervisor. 4.Enable the supervisor and administrator to modify the subjects. 5.Receive the announcement made by the admin to the students and supervisors depending on their roles. 6.Students may propose their own subjects, which are visible to supervisors. 7.Students can specify their preferences, which are stored in the database depending on their interests and needs.
Client-side and server-side	9-10 week	1.Enable the administrator to view the students' preferences. 2. Admin should be able to change the preferences set by students. 3.An algorithm must be constructed to form a group of four students with a supervisor and the topic provided.
Client-side and server-side	11-12 week	1.When the administrator runs the algorithm, students and supervisors are permitted to view the group allocation.
Software quality activity	13th week	1. Thoroughly test the application with the testcases written for each feature. 2. Bug fixing

Table – 6.3.1 Milestone

The Gantt chart, as illustrated in Figure 6.4.1 - Gantt Chart below, was used to track the progress and completion status of the software development life cycle. Gantt chart for software development were maintained based on the Scrum task board displayed in section 6.1 Software Development Approach.

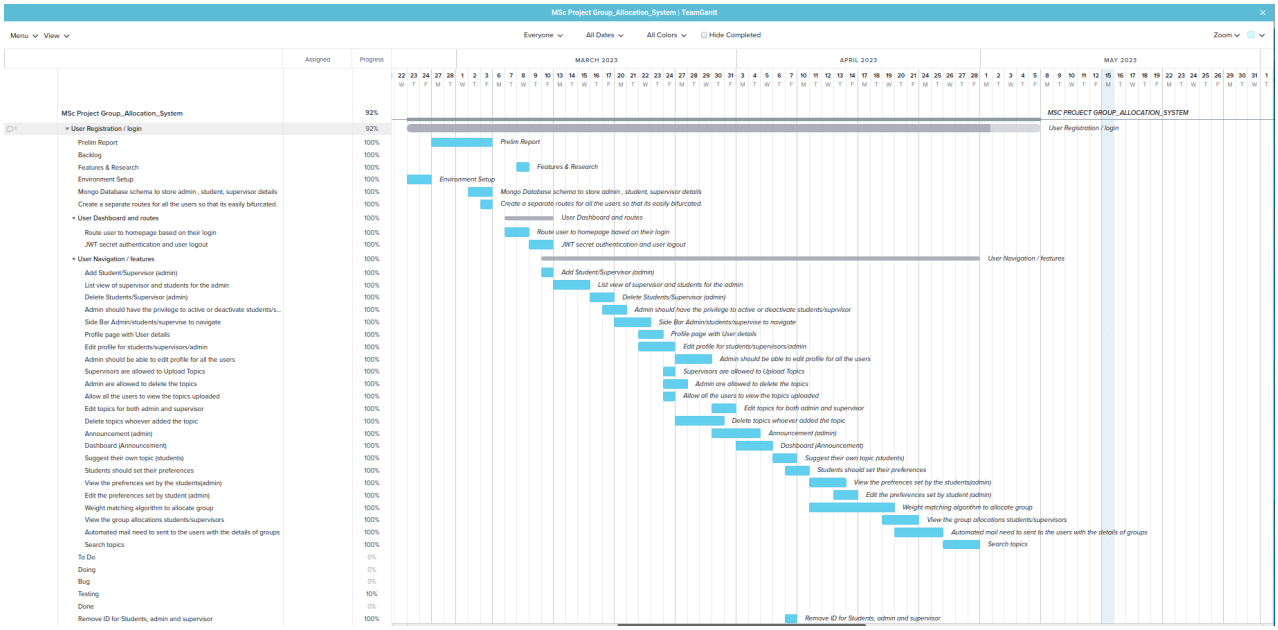


Figure 6.4.1 – Gantt Chart

7. Project Outcome

7.1 Client-side Modules

Client-side refers to the portion of a programme where the user interface is made available to them so they can interact with the web application.

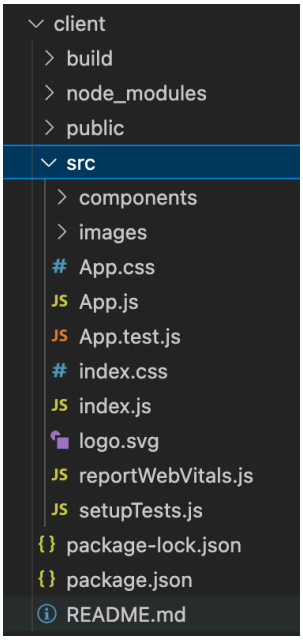


Figure 7.1.a – Client-side project structure

The project's client-side structure is depicted in the above picture, where all the application-related components are maintained in src/components, the application's images are saved in src/images, and the package.json contains all the application-related dependencies. The routes of every component created for the application are maintained in the App.js file.

7.1.1 Application Components

1. Landing page:

The landing page is the component that a user would initially land on while launching client as seen in the Figure 7.1.1. This page features login and register buttons. Clicking on the login button will take the user to the login page, while clicking the register button will take them to the application's registration page. Furthermore, the user can sign in or register themselves with the application.

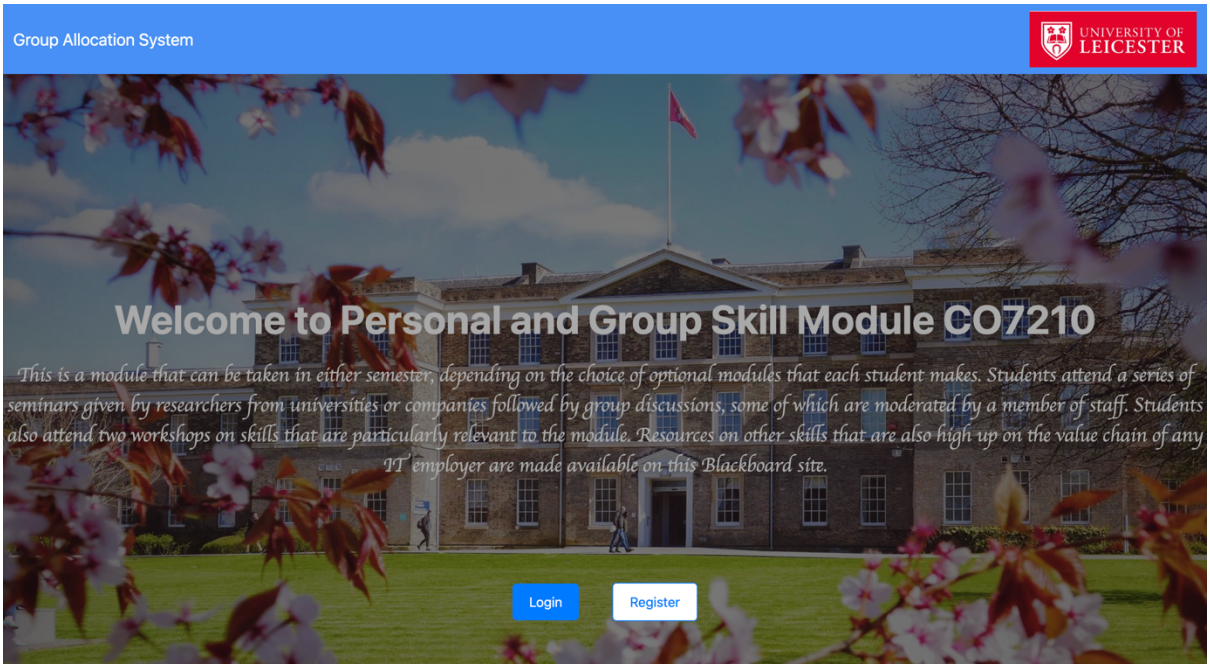


Figure 7.1.1 – Landing Page

2. Registration and Login module:

This module allows users to sign up for the application; they can be either students or supervisors. Once a user has registered with an application, they are able to log in using the credentials they had provided, and the application will direct them to their dashboards according to the login information entered in the software application.

a. Student Registration

The registration form for the students is used to sign up with the application; Students need to provide their first and last names, email address, password, and department to which they belong to, and the information furnished gets stored.
Once the student registers with the application he would be allowed to login to the application with valid credentials he had provided for the application. Further information and the snapshot are provided in the appendix A.

b. Supervisor Registration

The registration form shown in the Figure 7.1.2 below is what supervisors use to sign up with the application; they require to furnish their first name, surname, email ID, password, department, and secret key provided by the administrator to prevent the registration of any fraudulent profiles.
Once the supervisor hits "sign up," the information is registered, and the supervisor can log in using the credentials they have generated.

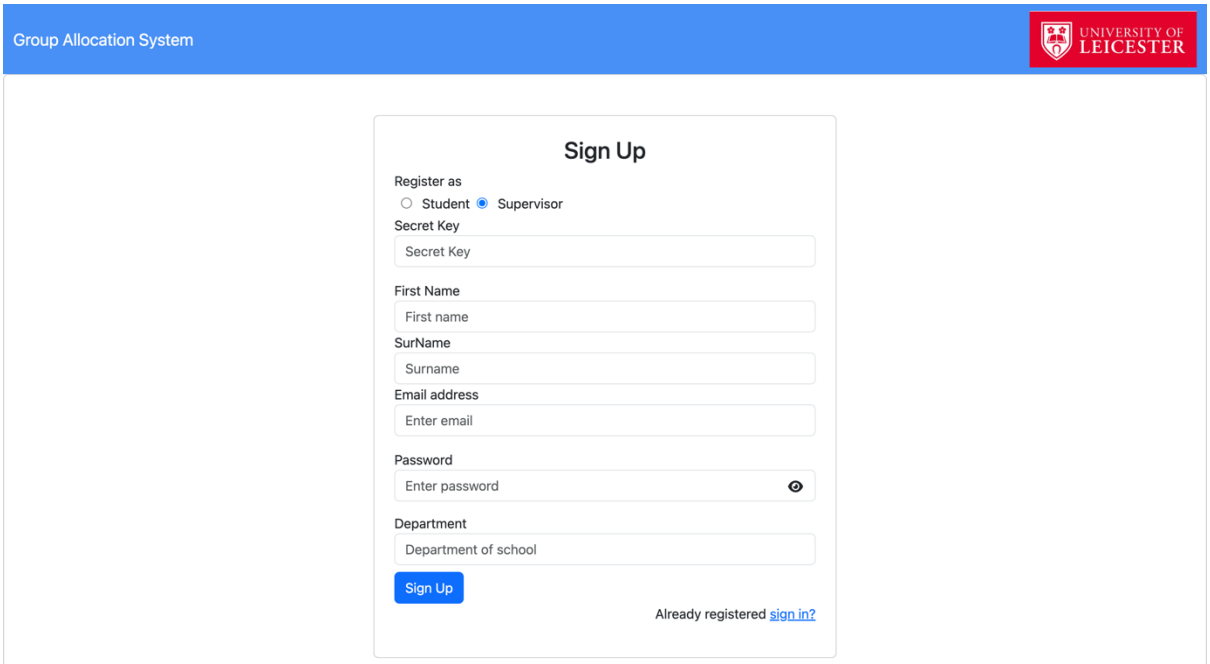


Figure 7.1.2 – Supervisor Registration Page

c. User Login Page

Based on the credentials provided, the application will direct users to their individual dashboards using the login form. The author has developed a single login component for all the users in the application, Once the user has enters his credential, a token is generated based on that credential and is sent to the backend. This JWT token is then checked against the data stored in the database, and verifies it. If the verification is

sucessfull the user is directed to the appropriate page where user-specific features are displayed. If unsucessful, the user will be asked to enter a valid credential. Further information and the snapshot are provided in the appendix A.

d. Forgotten password

The user has a separate component to reset the password and re-login to the application if they forget it or if the account is created by the administrator. Once the user clicks forgotten password, the user is asked to input the email address for which they have forgotten their password. Now they will receive an email with a link to reset their password. If user clicks the link, he will be redirected to a page to reset the password. Further information and the snapshot are provided in the appendix A.

3. Dashboard:

The system will verify the user's credentials upon logging into the application based on the credentials created during registration. The dashboard would display the notifications or the information that needs to be conveyed by the administrator to the specific users. Further information and the snapshot are provided in the appendix A.

This is carried out when the user enters the credentials. A part of the email ID such as @admin.leicester.ac.uk for administrator, "@student.le.ac.uk" for students and "@leicester.ac.uk" will be taken and it would route to the appropriate users login api which verifies the token against the data stored in the database.

a. Admin Dashboard

As soon as a user logs in as an administrator, the programme takes the user to the administrator dashboard where the announcement intended for administrators is displayed.

b. Student Dashboard

When a user enters in as a student, the software takes him to the student dashboard, where they may see the announcement that is only for students. And the latest announcements from the administration would show up.

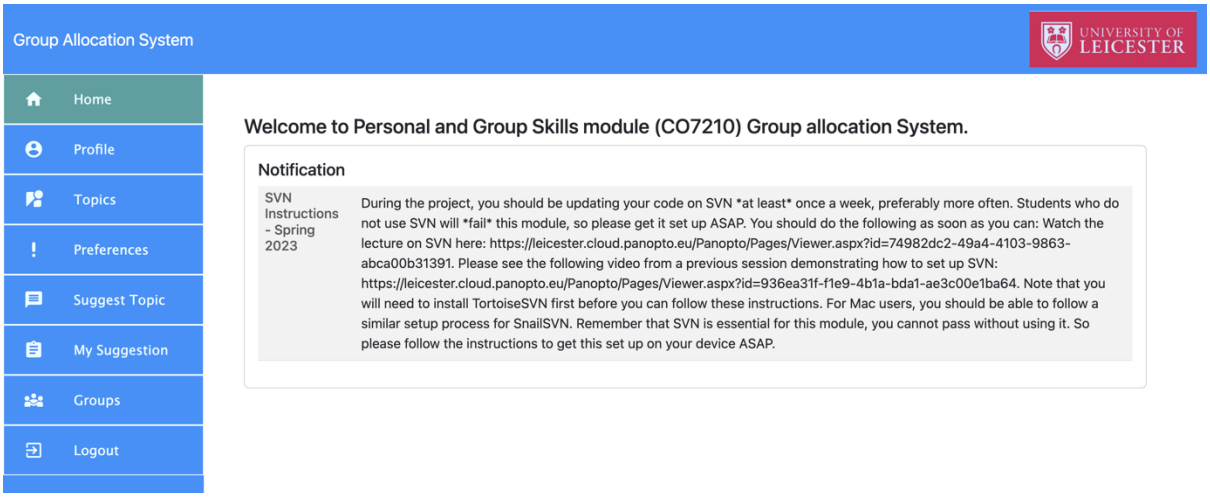


Figure 7.1.3 – Student Dashboard

c. Supervisor Dashboard

When a supervisor logs in to the application, the software directs him to the supervisor dashboard, where they may view an announcement that is intended exclusively for supervisors. And the latest announcements from the administration would also show up.

4. User Profile

As the application has three distinct actors, each actor's profile would be displayed as shown in figure 7.1.10, The user information that was initially submitted for registration will be shown, with the role and user would have the opportunity to edit only a few fields of the profile information. The role and the profile icon help to distinguish the profiles.

a. User Profile

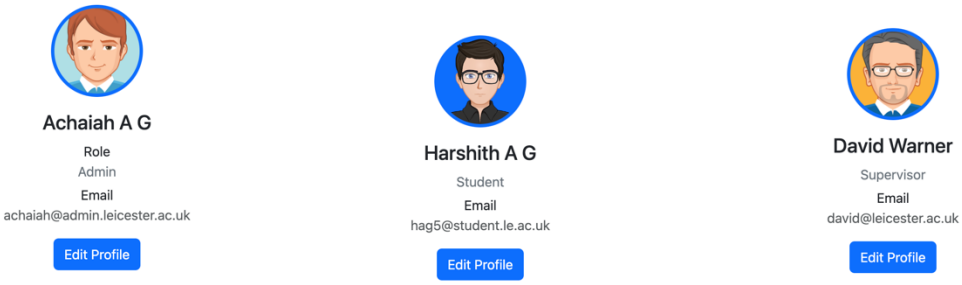


Figure 7.1.4 – User profile

b. Edit Profile

Users can update their own information, additionally administration also can modify supervisor and student information. Further information and the snapshot are provided in the appendix A.

c. Users Sidebar

A sidebar is a column with many rows that assists in application navigation. Each user has his own sidebar The sidebar changes depending on the user's role. For instance, students will have a sidebar exclusively with the features they need and those would not be allowed by other actors to access those features. The image below displays the components relating to different user roles.

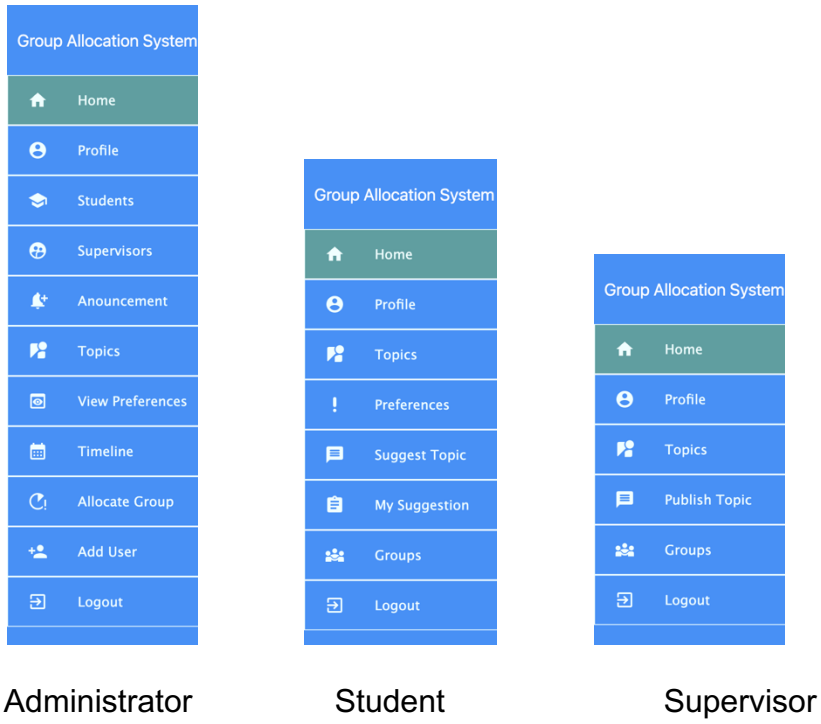


Figure 7.1.5 – Users Sidebar

5. Submit Topics.

This feature, which is primarily for supervisors, allows them to indicate the topic they would want to supervise. This topic may be the one in which they have expertise or the one they had chosen to carry out over their own interests. Once the supervisor uploads the topic, it is saved in the database and used as an input for the algorithm. The author has also added another feature to the application which allow the students to suggest the topics in which they are interested in. This topic list will be displayed to the supervisors, and if they interested, they may supervise the topics that the students have suggested.

a. Publish Topics by supervisors

To use this function, a supervisor must first upload the topic they wish to supervise. To upload a topic, the supervisor must first create a unique code by clicking the

generate code button, enter the code, and then upload the topic's details such as the title and the description.

The supervisor also has the option of taking up a topic that the students have submitted for supervision as their suggestion. These topics would be displayed in a table with the title, a brief explanation, the student's name, and their email address. If the supervisor decides to pursue the topic given by the students, they must produce the code by selecting the generate code button, pasting it into the topic they want to supervise, and then clicking "Approve." After the supervisor clicks the approve button, the professor will be in charge of the topic and the student would be notified through an email.

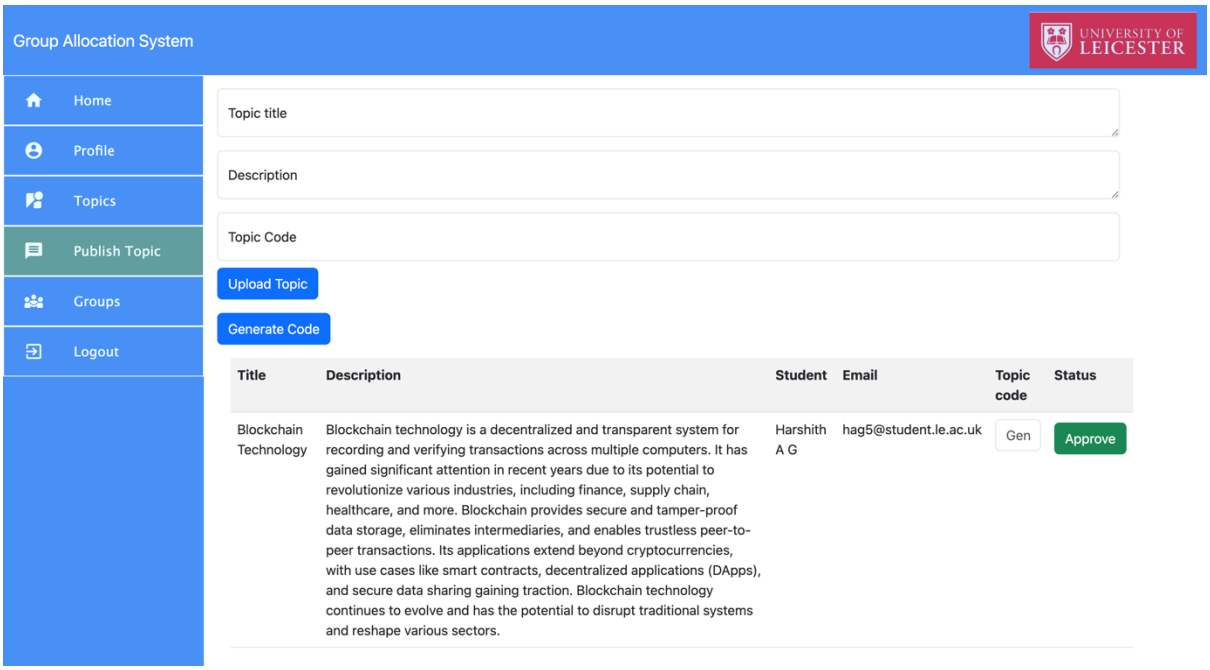


Figure 7.1.6 – Publish Topics (supervisors)

b. Suggest Topics by the students.

Students have the option to suggest a topic that they would want to work on. The title and a succinct explanation of the topic that the students would like to suggest should be included. Thereafter the topic suggested would be displayed to the supervisor and if they are interested, they would take up the topic.

Once a student has suggested a topic, the topic will be shown for that student in a tabular format under the “My Suggestion” page of the student profile. Students have the option to modify or remove a topic under this tab. The students would be notified through email if any supervisor approves the topic. Further information and the snapshot are provided in the appendix A.

6. Topics table for users:

Displaying the topics that are available in the course module is this component's primary function. By entering information in the search area and clicking the search icon, a user can look for a relevant topic. Depending on the user's role, a few more features may be offered. For instance, an administrator may be able to remove any topic, while a student will not have that specific feature. Further information and the snapshot are provided in the appendix A.

a. Topics view for Administrator

The administrator can search, edit, or remove any topic by clicking on the icons displayed in the tabular view, which is available for the course module as seen in the Figure 7.1.7 below. The topics suggested by students are also included on the administrators' Topic view, where admins have the option to modify or remove them if its repeated or if they find it irrelevant.

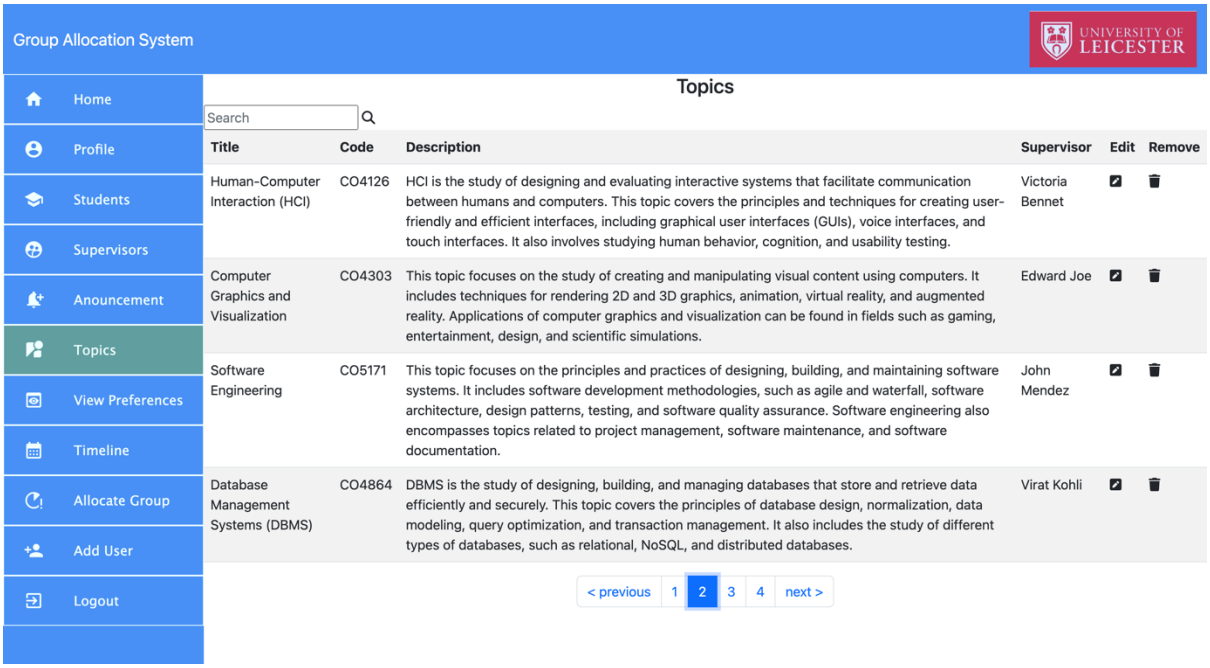


Figure 7.1.7 – Topics Table (Administrator View)

b. Topics view for Supervisor

The supervisor can search for any topic but can only edit or remove topics that they have uploaded themselves by clicking on the icons in the tabular view. This functionality is specific to individual supervisors and one supervisor cannot modify the other supervisor's topic. Students suggested topic will appear in this table only if a supervisor has approved the topic.

c. Topics view for Students

Students can look up topic table and search for any topic. The icon such as delete and edit won't be available for the students as they don't have the privilege to edit

or delete them. Topics which are suggested by the students would be displayed until they are approved by one of the supervisors.

7. Submit Student Preference

The students will be able to submit their preferences on the topics that have been uploaded or approved by supervisors. Students may submit their choices here depending on their areas of interest or on which they would like to work.
For students, four preferences would be accepted and recorded in the database. Before the deadline starts, students are free to change their selections.
These students' preferences serve as the primary input for the algorithm towards allocation process.

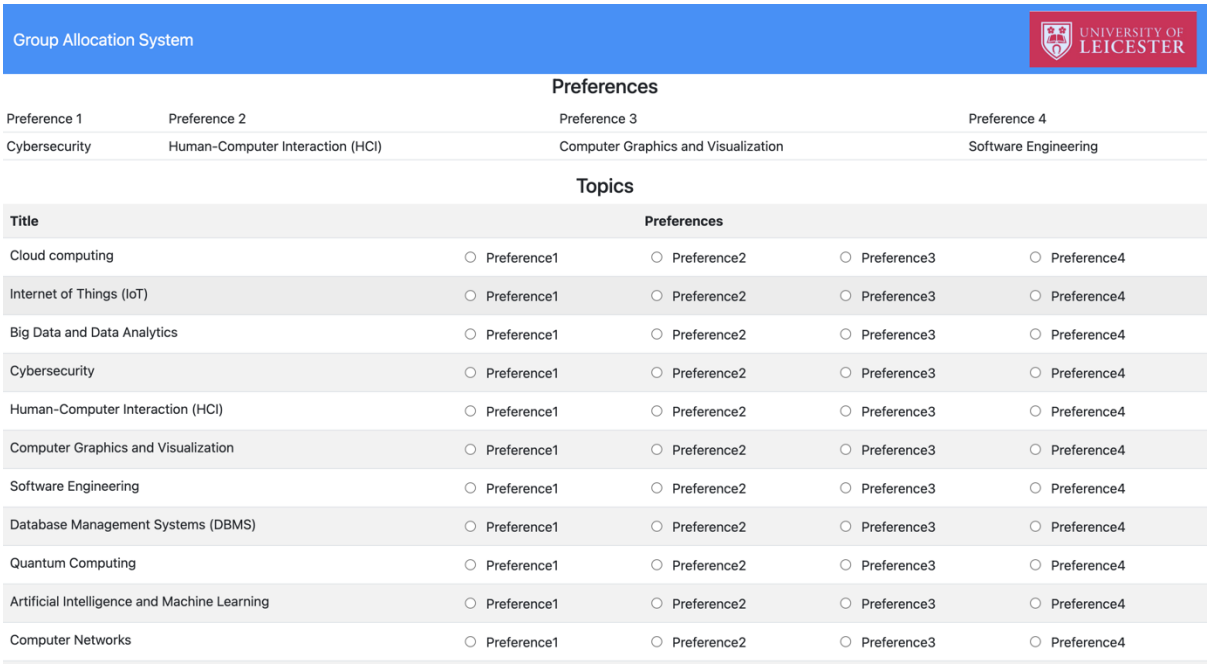


Figure 7.1.8 – Submit preferences (Student View)

8. View all registered users

Administrators can see the list of students and supervisors who have registered for the application by selecting the Students and Supervisors option from the sidebar of the application as shown in the below Figure 7.1.9.
Here, the administrator can change the user's information - they can remove, temporarily disable the user's account, and edit few profiles information whether they are students or supervisors.

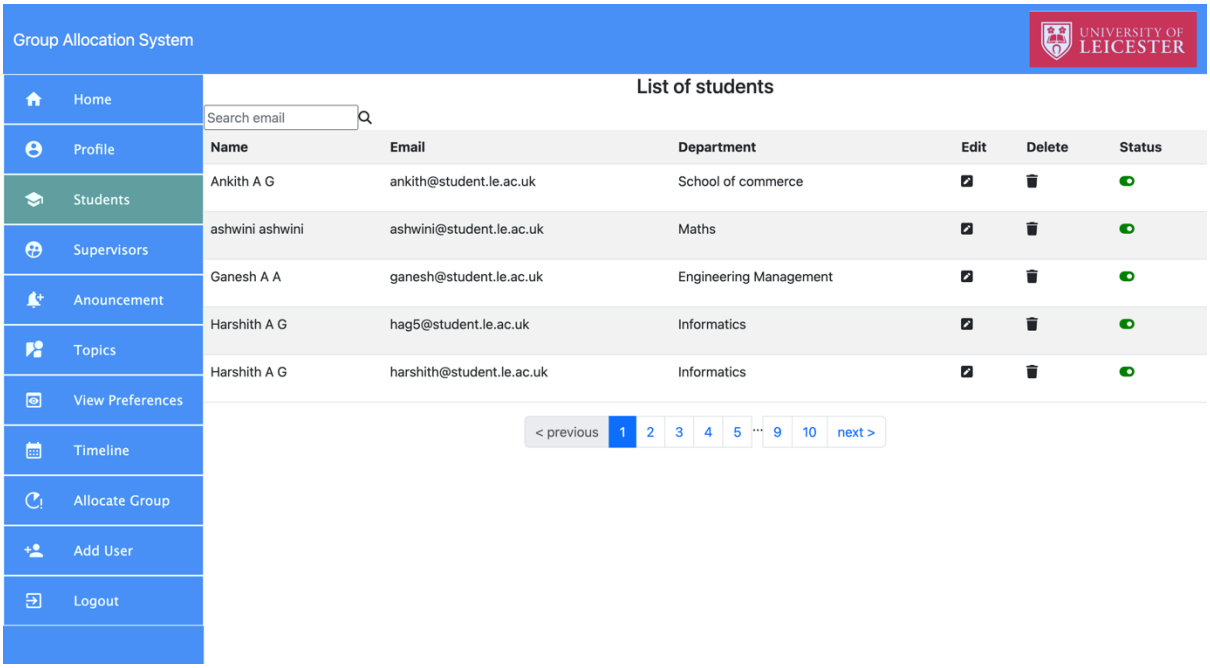


Figure 7.1.9 – Registered Users

9. Students Preferences

Administrators have the option of viewing and editing the preferences submitted by students or setting preferences on their behalf by clicking the edit icon as shown in the Figure 7.1.19. Further information and the snapshot are provided in the appendix A.

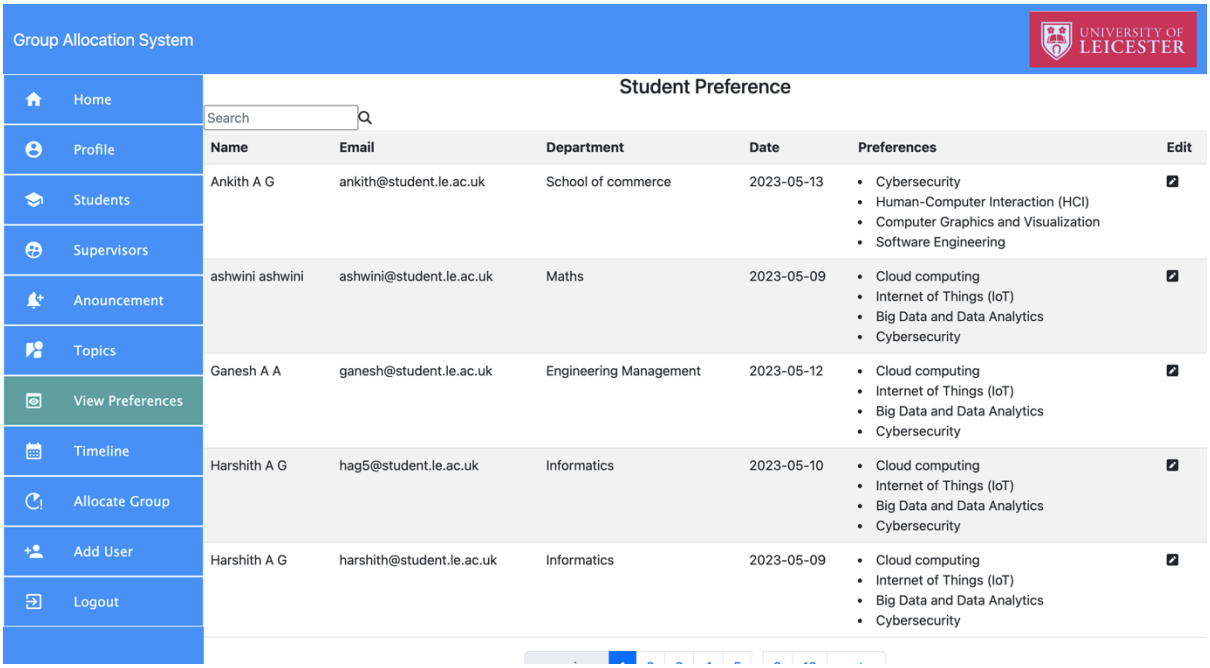


Figure 7.1.11– Edit Students Preferences (admin)

10. Dashboard Announcements

All the announcements that are presented on the user dashboard, as described in Section 3, have been announced in this component. When an admin sends out an announcement to a specific user, the message is shown on that user's dashboard and is also delivered through email.

By clicking on the icons in the table for the current announcements, the administrator can edit and remove the announcement here. There is an announce icon which allows the administrator to hide and show the announcement to the users on clicking it. It would change to red if the announcement is disabled. Further information and the snapshot are provided in the appendix A.

11. Timeline and Email notifications

The administrator may use this feature to activate or disable the page's visibility by clicking the button. The colour of the button changes as they are clicked; they become blue if they are enabled and red otherwise.

The section "Notify Events" allows the admin to select whom to email by selecting a radio button, filling in the subject and the announcement that needs to be made, and then clicking "Notify." This will enable the admin to send email to the professors or students.

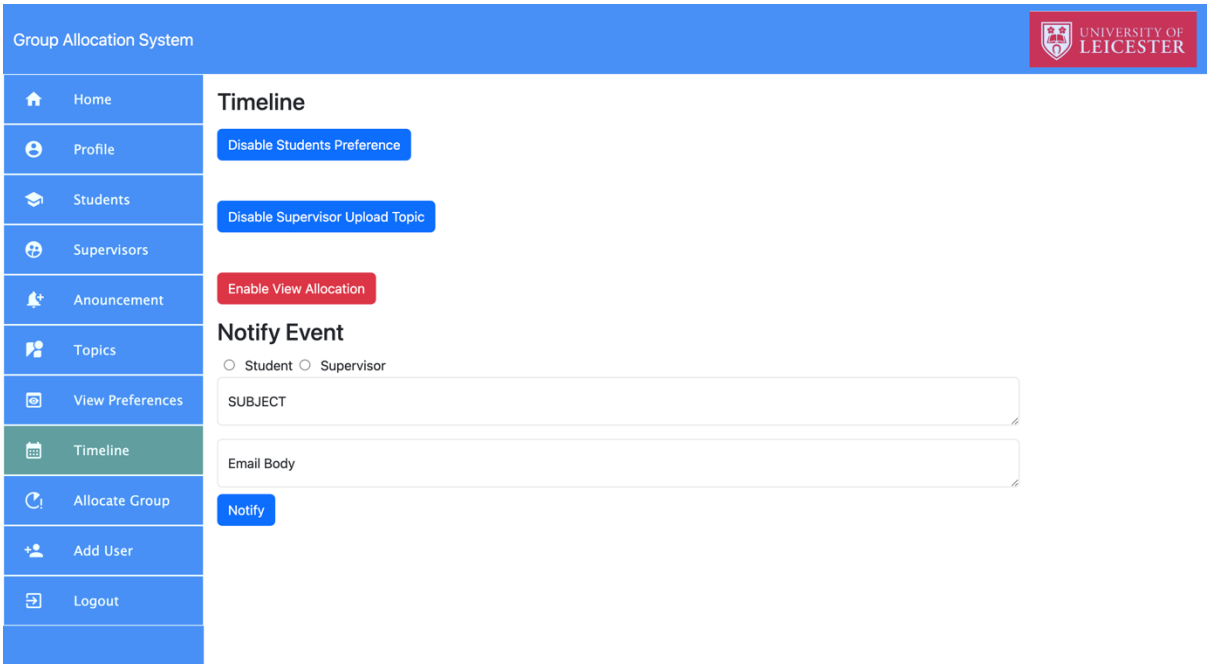


Figure 7.1.12 – Timeline (admin)

12. Register User by admin

Administrators can register users to the application who might be supervisors or students. In this case, the password area is not available, but by default, the user's email address is used as the password so that they can login to the application temporarily. Later, if required, users can change their password.

13. Allocate Group

This is the administrator's primary area, where they run the algorithm to divide students across the topics while taking their individual preferences into consideration for the algorithm. The algorithm may be used by the administrator to see the allocation results. The administrator can clear the allocation result and re-run the process if the allocation results don't seem to be satisfactory.

The default group size is 4, but the administrator can input a different number if they need to modify group member's count. For instance, the algorithm will establish a group of 7 if the administrator specifies "group size" as "7" and clicks "allocate group."

If the allocation appears to be acceptable, the administrator may make it accessible by activating the page for viewing in the timeline or sending it through an email by clicking on the send email button once the algorithm has been run.

Here, the author has created two distinct algorithms, one with a N^2 time complexity with first come first serve and the other with a N^2 time complexity based on the preferred topic.

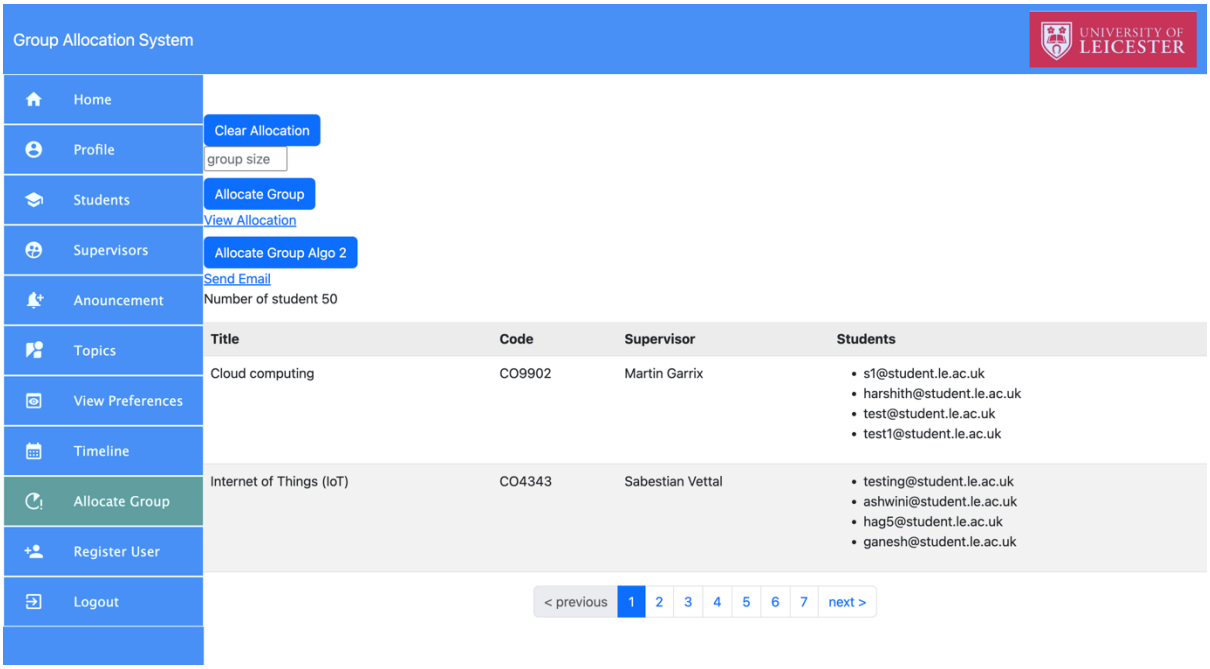


Figure 7.1.13 – Allocate Group(admin)

14. Group Information

Here, details about the group that was created after the administrator ran the algorithm will be shown.

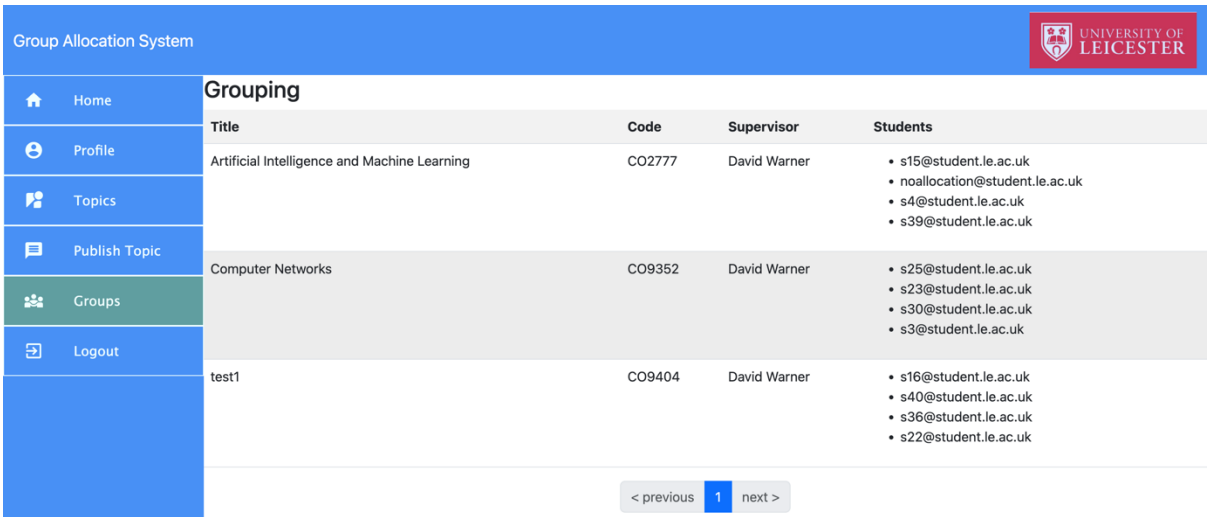


Figure 7.1.14 – Group Information (Supervisor)

15. Logout

When a user hits the logout button, the programme deletes all the user's information stored in the local storage and closes the current session. The login page will be presented to the user.

7.2 Server-side Modules

The server-side of the application uses CORS which handles all the interactions between the client-side and the server-side. Express with NodeJS environment are user to build restful api's and the application related data are stored in mongodb with mongoose to connect with the backend.

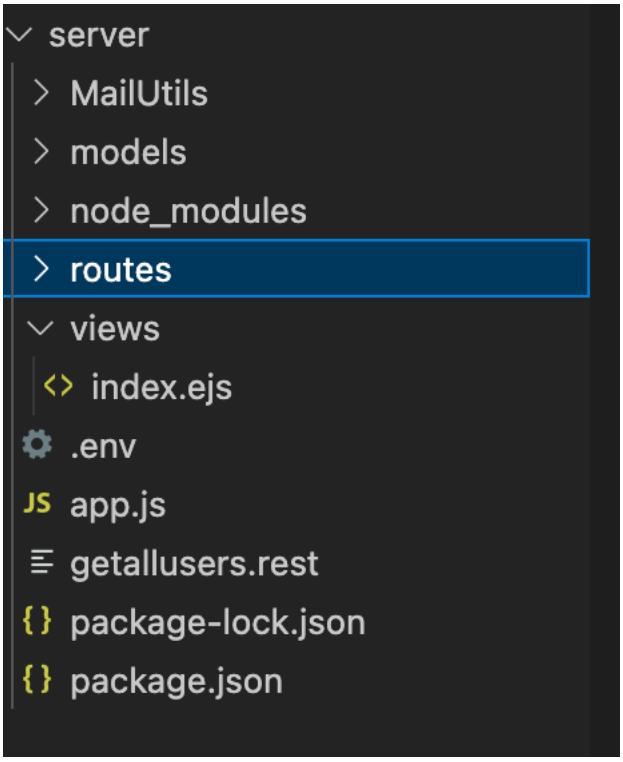


Figure 7.2.1– Server-side project structure

The server-side project structure is shown in the above Figure 7.2.1, where all server-side-related schema will be kept under server/models. Since there are three actors in the application, each actor has a unique route that has been maintained in index.js. It contains all the application's related APIs. Back-end required dependencies are in the package.json file and a dotenv file is maintained to store the credentials. All other connections and configurations are maintained in app.js file.

7.2.1 Application programming interfaces

Express and nodeJS are utilised to provide a RESTful API for the application. Based on the functionality that is utilised for; the author has categorised the endpoint.

a. router.post('/allocate-stud-preferred-topic', async (req, res) => {}

The above mentioned is the POST method in an api to allocate students and supervisor into the topics, where the administrator would send group size as an input, which would be taken from the request body in the above code. The data from the StudentInformation table which are the preference of the students and the topics from the TopicInformation table would be used as the input for the algorithm.

b. router.post('/login', async (req, res) => {}

This above mentioned api is common for all the users but the application checks the type of user and decides which route to select. The application has 3 main routes /admin,

/student and /supervisor. For example, if admin logs in then the end point would be admin/login and if student it would be student/login.

c. router.post('/search-student/:content', async (req, res) => {}

This api is to search the contents in the database and returns the relevant result where the content that needs to be searched is passed in the query string. This is further taken as the input for the api to search the content in the database where data is fetched and sent as a Json.

d. router.get('/reset-password/:id/:token/:email', async (req, res) => {}

The above api is to reset the password of the user which considers multiple query parameter to authenticate the user such is the id, email and the token generated during the login. Based on this query input the user is authenticated and if it verifies successfully, it directly routes the user to api to change the password.

Some important RESTful Api's are given in the below table.

Module	Method	Function	EndPoints		
			Administrator	Student	Supervisor
Registration	post	Register the user to the application	/admin/registration	/student/registration	/supervisor/registration
Login	post	Login the user to the application	/admin/login	/student/login	/supervisor/login
User Information	post	Get users data	/admin/data	/student/data	/supervisor/data
Topics	get	1.Get all the topics 2.Get specified number of topics	1./admin/gettopics 2./admin/gettopics?page=${currentPage.current}&limit=${limit}	1./student/gettopics 2./student/gettopics?page=${currentPage.current}&limit=${limit}	1./supervisor/gettopics 2./supervisor/gettopics?page=${currentPage.current}&limit=${limit}
	patch	Update the existing topics	/admin/updatetopic	na	/supervisor/updatetopic
Allocation	post	Allocate students and supervisor into the groups based on their preferences	/admin/allocate-stud	na	na

	post	Allocate students and supervisor into the groups based on their preferences	/admin/allocate-stud-preferred-topic	na	na
	put	Reset all the allocation made	/reset-allocation	na	na
Preferences	patch	Edit and save the preferences	/admin/savepreferences	/student/savepreferences	na
	post	Get individual preference	na	/student/getstudentpreference	na
	get	Get all students preferences	na	/student/getallstudentpreference	na
Upload topic	post	Add topics	na	na	/supervisor/uploadtopic
	post	Delete topics	na	na	/supervisor/deletetopic
	post	Update topics	na	na	/supervisor/updatetopic
Email Notification	post	Notify events through email	/admin/email-notify	na	na
Student Suggestion	post	Add student suggest topics	na	/student/suggest-topic	na
	post	Get all student suggested topics	na	/student/stud-suggestion	na
	post	Delete student suggestion	na	/student/delete-suggested-topic	na
	post	Update student suggestion	na	/student/update-suggested-topic	na
	get	Get all the suggested topic by student	na	na	/supervisor/stud-suggestion
	patch	Approve suggestion	na	na	/supervisor/approve-suggestion

Table 7.2.2– API's

7.3 Database and Schema

Using mongodb and its library named mongoose is used to establish connections with the client-side and the database. Figure 7.3.1 below shows all the schema with their fields and type which have been maintained in the application.

Since the application has three different actors, there are separate tables to store the information about the users, such as SupervisorInformation to store details about supervisors, AdminInformation to store details about administrators, and StudentInformation to store details about students. The fields in these tables may differ depending on the type of user in the application.
Each student's preference and the assigned topic are being stored in StudentInformation table.
The AnnoucementInformation table is used to hold information about user dashboards.
TimelineInformation have been utilised in the programme to store the information to enable or disable the page visibility.

TopicInformation is a table that stores information about topics when a supervisor uploads it. Once the administrator has run the algorithm, this table is then utilised to store group members.

The topic's table and the student tables' preferences are used as the algorithm's input, and the topic table is afterwards utilised to record the allocation result so that all the data from that table may be obtained and shown to users.

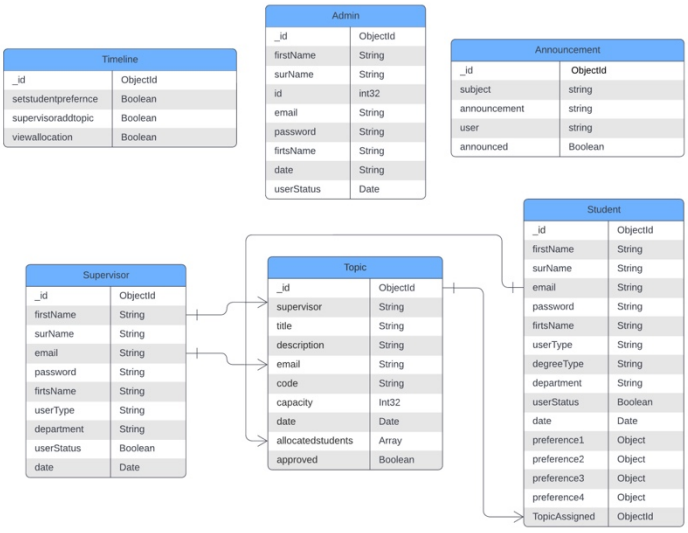


Figure 7.3.1 Database Schema

app.js

```
1   const express = require("express")
2   const app = express()
3   const mongoose = require("mongoose")
4   const dotenv = require('dotenv')
5   const cors = require("cors");
6   app.set("view engine", "ejs");
7   app.use(express.urlencoded({ extended: false }))
8
9   // Load ENV variables
10  dotenv.config()
11
12  // Database connection URL
13  const mongodbUrl = process.env.DATABASE_ACCESS;
14
15  const allRoutes = require("./routes");
16
17
18  app.use(cors(
19      {
20
21      }
22  ));
23
24  // Connect to Database
25  mongoose.connect(mongodbUrl, { useNewUrlParser: true })
26      .then(() => { console.log("Database Connected") })
27      .catch(e => console.log(e))
28
29
30
31  app.use(express.json());
32
33  app.use("/", allRoutes)
34
35  // Localhost port 4000 backend
36  app.listen(4000, () => {
37      console.log("Server is up and running");
38  })
```

Snipped

Figure 7.3.2 app.js server-side (Database Connection)

7.4 Code Snippets

The code snippet 1 is the portion of the weight matching algorithm. The below code has two for loops one which iterates 4 time as there are four weights(preferences).
The other for loop is to iterate through the students, in the first iteration the code checks the first preference of all the students, and if the student's preferred topic has room to accommodate student, then the code will allocate the student to that topic. If not, it would go to the next student. Once the iteration is completed for first preference for all students the code increments to the next preference. Now this time the list of students is obtained who haven't been allocated any topics and check for their other preferences.

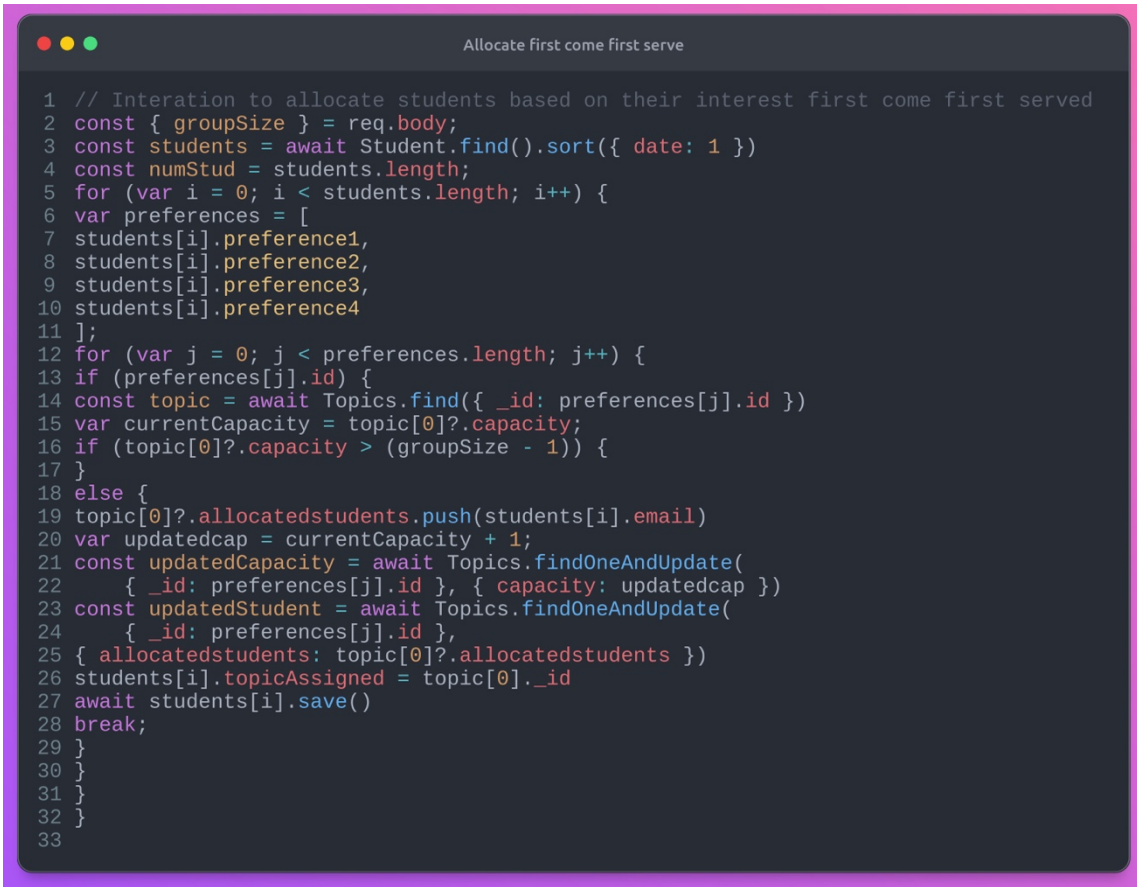
Assign preferred topic

```
// Interation to allocate students based on their preferred topics
for (var k = 0; k < 4; k++) {
        const students = await Student.find({ topicAssigned: { $exists: false } })
        const numStud = students.length;
        for (var i = 0; i < students.length; i++) {
            //store student preferences in an array
            var preferences = [
                students[i].preference1,
                students[i].preference2,
                students[i].preference3,
                students[i].preference4
            ];
            if (preferences[k].id) {
                const topic = await Topics.find({ _id: preferences[k].id })
                var currentCapacity = topic[0]?.capacity;
                if (topic[0]?.capacity > (groupSize - 1)) {
                }
                else {
                    topic[0]?.allocatedstudents.push(students[i].email)
                    var updatedcap = currentCapacity + 1;
                    const updatedCapacity = await Topics.findOneAndUpdate(
                        { _id: preferences[k].id }, { capacity: updatedcap })
                    const updatedStudent = await Topics.findOneAndUpdate(
                        { _id: preferences[k].id },
                        { allocatedstudents: topic[0]?.allocatedstudents })
                    students[i].topicAssigned = topic[0]._id
                    await students[i].save()
                }
            }
        }
    }

```

Code snippet 1

In Code snippet 2 group size that needs to be formed is obtained through the request body in the group allocation portion of the code sample below. A list of students would be collected using a for loop to go over the list of students who have registered for the application and the student's recorded preferences are retrieved. The author checks the student's preferred topics in the second for loop and determines if there is a space for the student to be assigned to that topic.
If the room is available in the topic, the code updates the topic table with that student’s email and the topic would be assigned to that student.



Code snippet 2

The code snippet 3 is the server-side code of the application, where the application generates a JWT token based on the credentials entered when the user logs in. The token is kept in the application's local storage, and then it is sent to the application's backend for verification against the data maintained in the database. If the data entered is found to be correct, the user data is then sent as a response as a Json file to the front end of the application.

When the application receives data from its backend, it may determine the user type from the information and then utilise that information to make that user's preferred functionality available and renders it. Additionally, the programme keeps track of the token until it expires; If it does, it logs the user out of the programme.

User authentication and route

```
1
2  //API for UserData
3  router.post("/data", async (req, res) => {
4      const { token } = req.body;
5      const studentEmail = "student.le.ac.uk";
6      const superviserEmail = "leicester.ac.uk";
7      const adminEmail = "admin.leicester.ac.uk";
8      try {
9          const user = jwt.verify(token, JWT_secret, (err, res) => {
10             if (err) {
11                return "token expired";
12             }
13             return res;
14         })
15         if (user == "token expired") {
16             return res.send({ status: "error", data: "token expired" });
17         }
18         const userEmail = user.email;
19         if (userEmail.includes(adminEmail)) {
20             Admin.findOne({ email: userEmail })
21                 .then((data) => {
22                     res.send({ status: "ok", data: data })
23                 })
24                 .catch((error) => {
25                     res.send({ status: "error", data: error })
26                 })
27         }
28         else if (userEmail.includes(studentEmail)) {
29             Student.findOne({ email: userEmail })
30                 .then((data) => {
31                     res.send({ status: "ok", data: data })
32                 })
33                 .catch((error) => {
34                     res.send({ status: "error", data: error })
35                 })
36         }
37         else if (userEmail.includes(superviserEmail)) {
38             Supervisor.findOne({ email: userEmail })
39                 .then((data) => {
40                     res.send({ status: "ok", data: data })
41                 })
42                 .catch((error) => {
43                    res.send({ status: "error", data: error })
44              })
45         }
46     } catch (error) {
47         res.send({ status: "error" })
48     }
49 })
```

User authentication and route

```
//Page that renders users profile based on the credentials entered

const UserDetails = () => {
    const [userData, setuserData] = useState('')
    const [superviser, setSupervisor] = useState("")
    const [admin, setAdmin] = useState('')
    useEffect(() => {
        fetch("http://localhost:4000/user/data", {
            method: "POST",
            crossDomian: true,
            headers: {
                "Content-Type": "application/json",
                Accept: "application/json",
                "Access-Control-Allow-Origin": "*"
            },
            body: JSON.stringify({
            token: window.localStorage.getItem("token"),
            }),
        }).then((res) => res.json())
            .then((data) => {
                console.log(data, "User Data");
                if (data.data.userType === "Supervisor") {
                    setSupervisor("Supervisor")
                }
                else if (data.data.userType === "Admin") {
                    setAdmin("Admin")
                }
                setuserData(data.data);
                if (data.data === "token expired") {
                    alert("Token expired Login again")
                    window.localStorage.clear();
                    window.location.href = "/sign-in"
                }
            })
    }, [])
    return (
        admin === "Admin" ? <AdminHomepage userData={userData} />
        : superviser === "Supervisor"
        ? <SupervisorHomepage userData={userData} />
        : <StudentHomepage userData={userData} />
    )
}
export default UserDetails

```

Code snippet 3

The applications navigation is taken care by the below code snippet where the component SideBar is used to render the pages based on the icons clicked by the user. These icons and routes for that page have been kept in the SideBarData, which has a title, icon, and the page route stored as objects. By importing these data into the SideBar component, it is rendered as a column with multiple rows, where each row will display the title with the icon once the user clicks on this icon and then the application will render the page that the user clicked on. Since the application has three different actor distinct side bar are maintained based on the code shown in Code snippet 3 relevant sidebar for the user would be rendered.

Side Bar

```
//Component that renders the side bar based on the user loggedin

function Sidebar() {
return (
    <div className="sidebarapp" >
        <ul className='SideBarlist'>
            {
            SidebarData.map((val, key) => {
            return (
            <li className="row" key={key}
             id={window.location.pathname === val.link ? "active" : ""}
            onClick={() => { window.location.pathname = val.link }}>
            <div id="icon">{val.icon}</div><div id='title'> {val.title} </div>
            </li>
            )
            })
            }
        </ul>
    </div>
)
}

```

Side Bar Data

```

import ExitToAppIcon from '@material-ui/icons/ExitToApp';
import PersonAddIcon from '@material-ui/icons/PersonAdd';
import StreetviewRoundedIcon from '@material-ui/icons/StreetviewRounded';
import NotificationAddIcon from '@mui/icons-material/NotificationAdd';
import MessageRoundedIcon from '@material-ui/icons/MessageRounded';
import PreviewIcon from '@mui/icons-material/Preview';
import CalendarMonthIcon from '@mui/icons-material/CalendarMonth';
import RunningWithErrorsIcon from '@mui/icons-material/RunningWithErrors';
import React from 'react';
export const SidebarData = [


    {
        title: "Home",
        icon: <HomeIcon />,
        link: "/admindashboard"
    },
    {
        title: "Profile",
        icon: <AccountCircleIcon />,
        link: "/user-details"
    },
    {
        title: "Students",
        icon: <SchoolIcon />,
        link: "/viewstudstaff/students"
    },
    {
        title: "Supervisors",
        icon: <SupervisedUserCircleIcon />,
        link: "/viewstudstaff/supervisors"
    },
    {
        title: "Anouncement",
        icon: <NotificationAddIcon />,
        link: "/announcement"
    },];
```

Code snippet 4

7.5 Libraries and plugins

The below table shows all the libraries and plugins installed for the speedy development of software development. Few of the inbuilt feature have been utilised in the application with the help of these plugins.

Library	Usage
FontAwesome	Used in the Frontend for the icons such as trashcan, edit, announce and toggle etc.
MaterialUI	Used in the SideBar of the application icons to navigate.
Bootstrap	Used for the table rendering and components of the application in the frontend.
jQuery	With bootstrap.
Prime React	Popups and Models have been used for the application
React-Pagination	Navigate across the data displayed in the table.
Nodemon	Detects the changes in the backend and restarts the server.
Bcryptjs	Used in hashing the password in the database
Cors	It is Cross origin resource sharing used to communication with backed and frontend.
Dotevn	Used to store connection string and few secret keys.
JWT web token	Used to authenticate user
Ejs Embedded JavaScript	Used to render page in the backend.
Mongoose	Used to establish connection between backend and database.

Table 7.5.1 Libraries and plugins

8. Conclusion

8.1 Challenges faced

All though there was quick start in the software development in the first week of the project, there was a need in changing the database schema for users, which was created initially for the students, supervisors, and admin. as while developing the software felt that the users should have few more details to make the validation stronger to include more user related data. Hence, there was a slight delay in the plan. In the beginning, the user details were stored in the single table where the roles were separated by the field provided with the information called “userType”.

However, the plan was to complete most of the features within the two sprints and spend the remaining time in developing an optimized algorithm. There was a lag in plan since new features in the software development was included as suggested by the supervisor in between the software development lifecycle and author made sure that it was completed.

As this programme is seeking an ideal algorithm, researching an algorithm, and using the pros of the algorithm while bringing up the modifications in the algorithm as needed has become a difficult chore.

Identifying the bugs discovered during the application testing was difficult, but the one-by-one feature development and testing(parallel) made everything run well.

8.2 Software Testing

Here in the software development life cycle, the development and the testing were done parallelly, which means when one of the requirements is developed, the same part of the development would be tested with different test strategies.

Initially, the requirement was categorized based on the ease of development and in linear sequential manner. Based on the requirements, manual testcases have been developed for testing the application which includes the happy flow and negative flow of the application. Few of the important testcases is shown in Appendix B.

The application would be tested against the testcases which have been developed for each functionality the outcome would be noted.

Below test strategy has been taken as main part of measuring the quality of the application:

- Functional testing also known as component testing which will check if the functionality of each component is well implemented or not. In the group allocation application, the author has made sure that the components and its functional would

work as expected. For example, the email input field should accept only the email in this application, admin should contain @admin.leicester.ac.uk and student should contain @students.le.ac.uk and supervisor should contain @leicester.ac.uk exclusively else system should not accept it.

- Integration testing is a kind of test done on the software to check the flow of the data between two features/modules. Once a module is developed, the data flow between two modules has been verified. For instance, if the supervisor uploads a topic, it should be visible to the administrator and the students, if he removes that topic should not be displayed in the table.

- System testing is the end-to-end testing of one flow of scenario and checks whether the application works as designed. Starting from the very beginning, such as creating a user and assigning them a topic, the author investigates several situations.

- Few other test methodologies would be also implemented such as path testing, condition testing, equivalence portioning etc.

- Regression testing have been done on the application where the new functionality is developed and tested to confirm that there is no impact due to new feature. A few bugs were fixed during the regression testing.

- API developed in the application have been tested using the tool postman and a collection where all the api's have been maintained. Every time an API is created, it is tested to ensure that the intended result will be achieved.

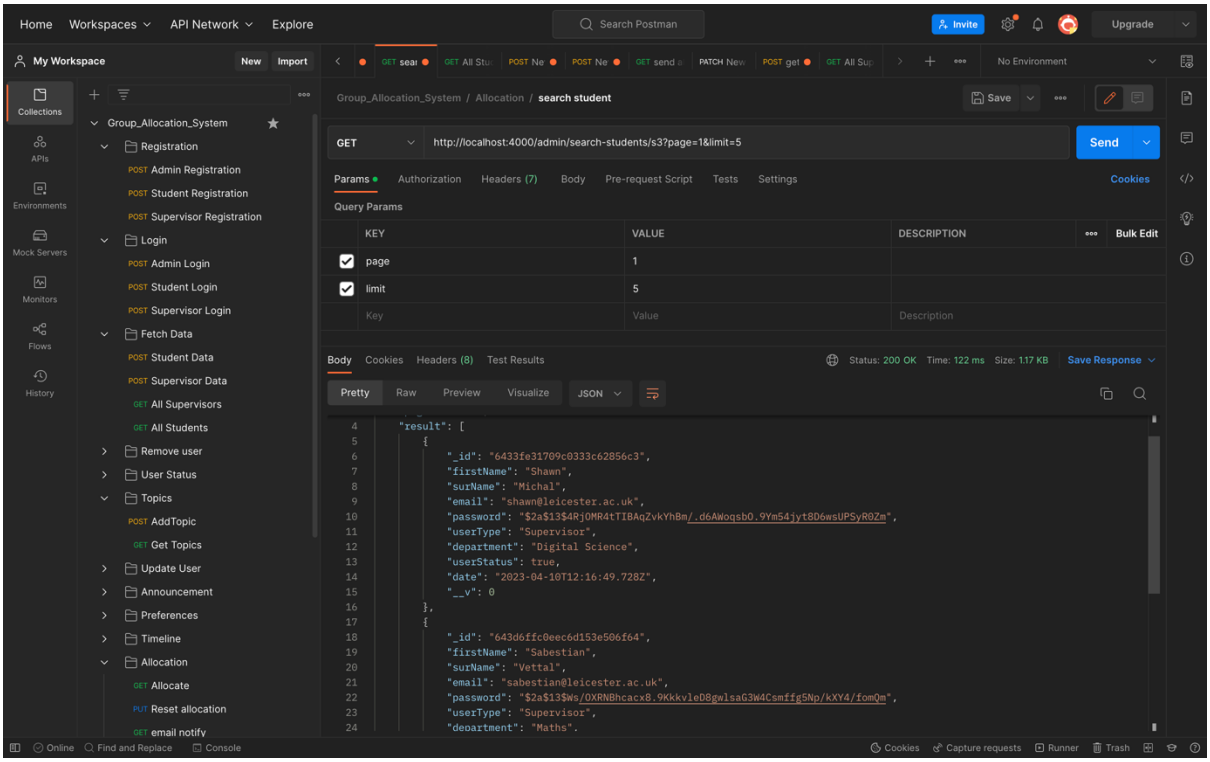


figure 8.2.1 API collections (Postman)

8.3 Feedback

The application developed was presented to the students of bachelor's in computer science in (Percy Gee Lab). As they got to know the aim of the software development, they explored the application and provided a few suggestions. Some of their suggestions were then incorporated in the application and the rest has been considered for the future improvement of the application. Similarly, this application was tested based on the student's suggestion and monitored the results for its working.

Furthermore, while gathering the requirement author contacted a few students and did a short survey regarding the software required by the students.

8.4 Future Improvements

Few things that could be improved in the future are,

- Rather than maintaining three separate tables for administrators, students, and supervisors, it would be preferable to have a single table in the database to have user information and differentiate them by adding a property called "usertype".
- User experience in the application can be improved as currently it is developed with basic UI.
- Automatic reminder to notify if the students haven't submitted the preferences and supervisors if they haven't uploaded any topics.
- A chat box can be provided to interact with supervisors and students to discuss regarding the topics.
- Inform students if the topic isn't approved after the topic submission deadline.
- Email body structure can be well defined.

9. References

1. Atlassian. (2022). *What is Agile?* Atlassian. https://www.atlassian.com/agile#:~:text=Agile%20is%20an%20iterative%20approach

2. *MERN Stack*. (2019, October 11). GeeksforGeeks. https://www.geeksforgeeks.org/mern-stack/

3. Wan-hong, L. (2017). *A Web-based Project Allocation System Interim Report*. Www.semanticscholar.org. https://www.semanticscholar.org/paper/ A-Web- based-Project-Allocation-System-Interim-Wan-hong/c827d370e0affa4b76b5a3a0a789dcb12a5aefa8#citing-papers

4. Modi, S., Shagari, N. M., & Wadata, B. (2018). Implementation of Stable Marriage Algorithm in Student Project Allocation. *Asian Journal of Research in Computer Science*. https://www.semanticscholar.org/paper/Implementation-of- Stable-Marriage-Algorithm-in-Modi- Shagari/2674182d8a7613dd07a84ba859e57fe362766626

5. Jean-Pierre Gerval, Yann Le Ru. A WEB-BASED SYSTEM FOR STUDENT-PROJECT ALLOCA- TION. International Conference on Computers and Advanced Technology in Education, Jun 2012, Napoli, Italy. hal-02998714

6. AWS. (2018). *Amazon Simple Email Service (Amazon SES)*. Amazon Web Services, Inc. https://aws.amazon.com/ses/

7. Nik Intan Syahiddatul Ilani Jailani, Al-Fahim Mubarak Ali, & Syahrulanuar Ngah. (2022). Final Year Project Allocation System Techniques: A Systematic Literature Review. *2022 IEEE 12th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, *10.1109/ISCAIE54458.2022.9794501*. https://doi.org/10.1109/iscaie54458.2022.9794501

8. Srinivasan, D., & Rachmawati, L. (2008). Efficient Fuzzy Evolutionary Algorithm-Based Approach for Solving the Student Project Allocation Problem. *IEEE Transactions on Education*, *51*(4), 439–447. https://doi.org/10.1109/te.2007.912537

9. El-Atta, A. H. A., & Moussa, M. I. (2009, December 1). *Student Project Allocation with Preference Lists over (Student, Project) Pairs*. IEEE Xplore. https://doi.org/10.1109/ICCEE.2009.63

10. Porter, P., Yang, S., & Xi, X. (2019). The Design and Implementation of a RESTful IoT Service Using the MERN Stack. *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*, *10.1109/MASSW.2019.00035*. https://doi.org/10.1109/massw.2019.00035

11. IJRASET, Y. J. B. (2022, January 18). *MERN: A Full-Stack Development*. Www.ijraset.com. https://www.ijraset.com/research-paper/mern-full-stack-development

10. Appendix

10.1 Appendix A - Features

In this section the author has provided snapshots and short explanation of each component of the software developed.

The Figure 10.1.1 is the registration form provided for the students to login to the application.

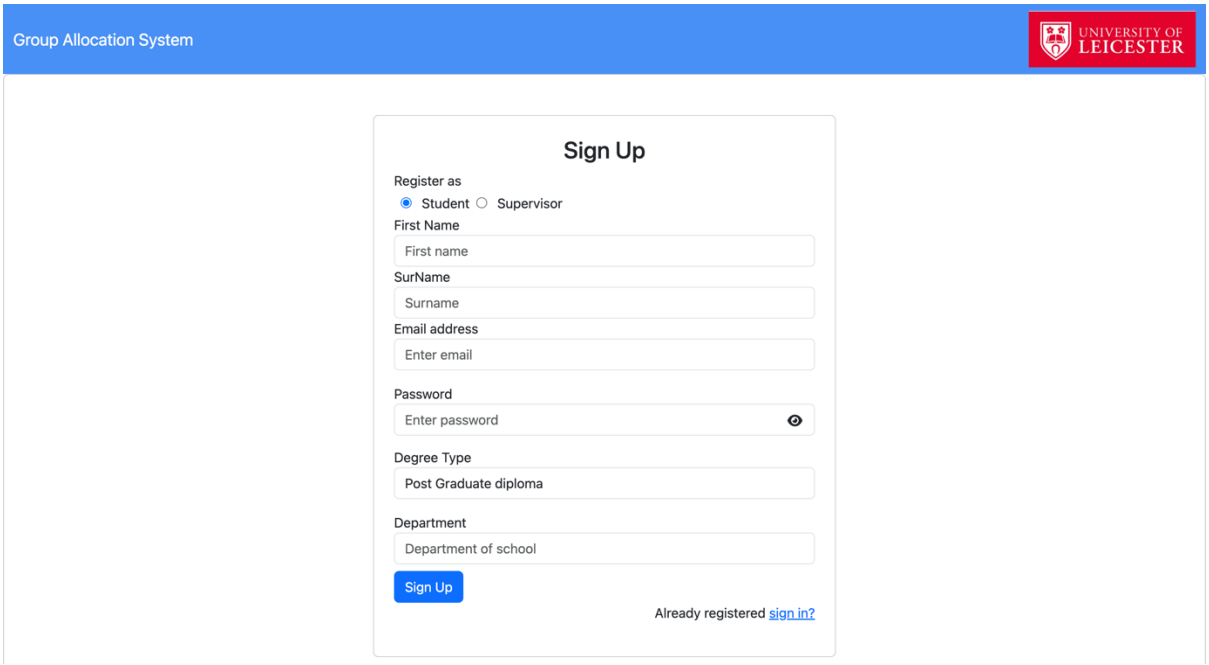


Figure 10.1.1 – Student Registration Page

Common login page is provided to all the user in the application as shown in the Figure 10.1.2. Once the credential is entered a token is generated and verified.

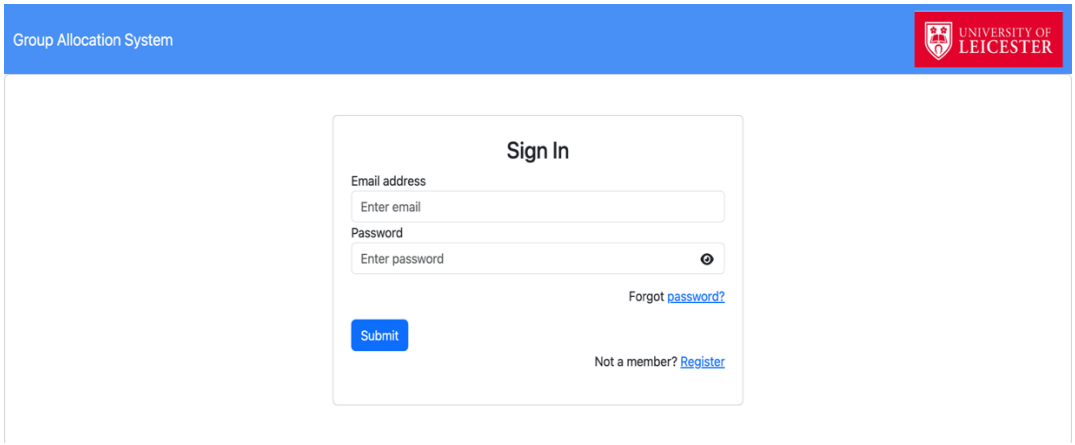


Figure 10.1.2 – User Login Page (Common for students/supervisor and administrator)

If the user foregets password or if he is the new user the below Figure 10.1.3 is the page where they can change the password.

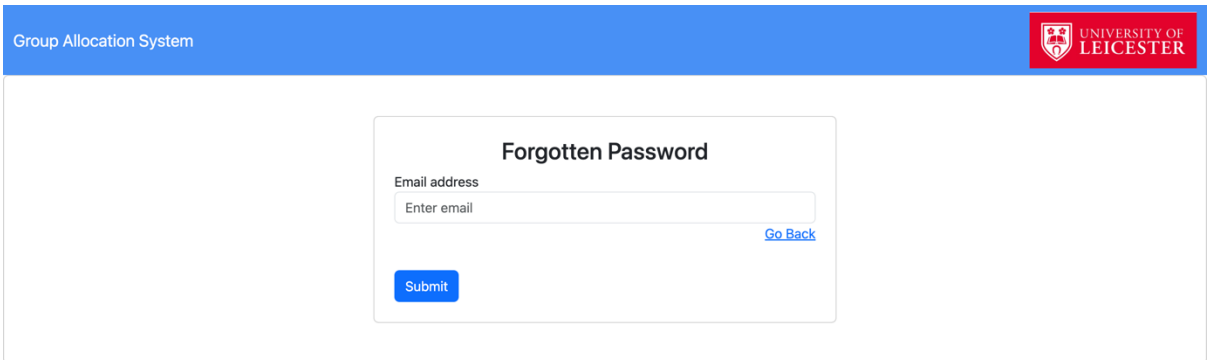


Figure 10.1.3– Forgotten password

Figure 10.1.4 – Reset password

Administrator dashboard would some information which provided by the developer.

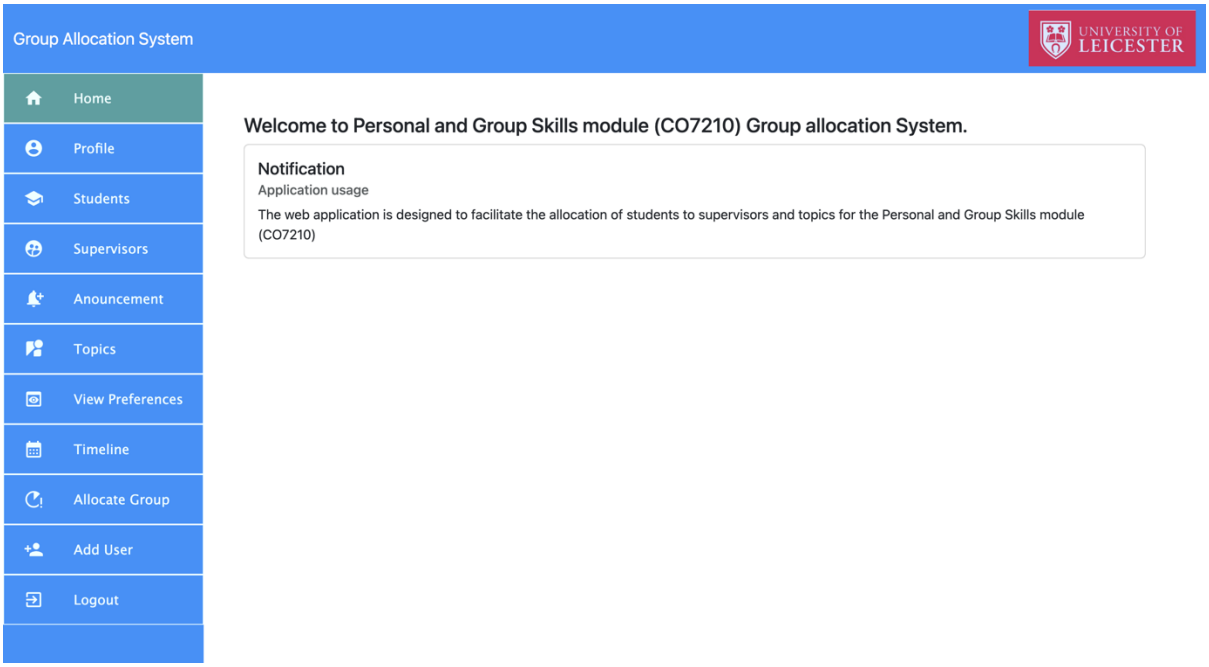


Figure 10.1.5 – Administrator Dashboard

Recent announcement made by the administrator for the supervisor would be display inn supervisor's dashboard.

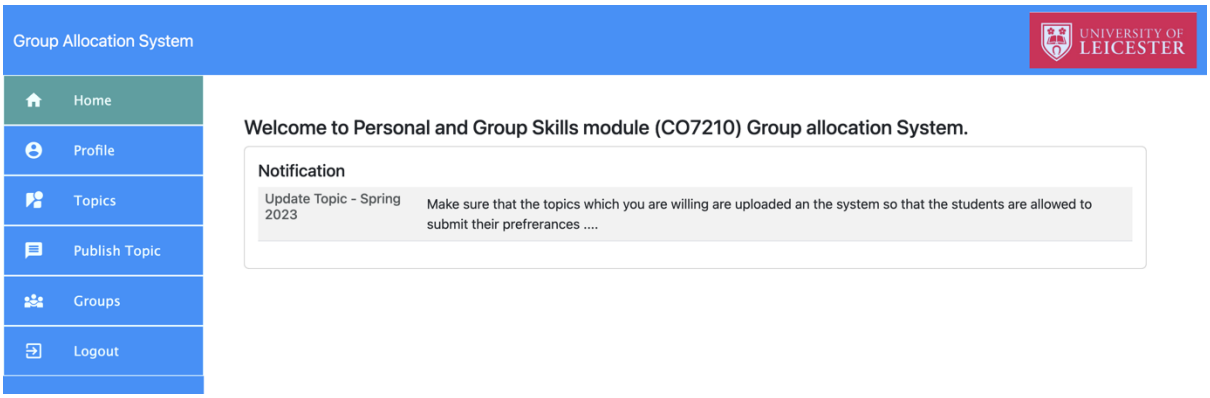


Figure 10.1.6 – Supervisor Dashboard

Details would be edited by the user, but a few fields are allowed to modify as shown below email would be allowed to edit and the user role as well.

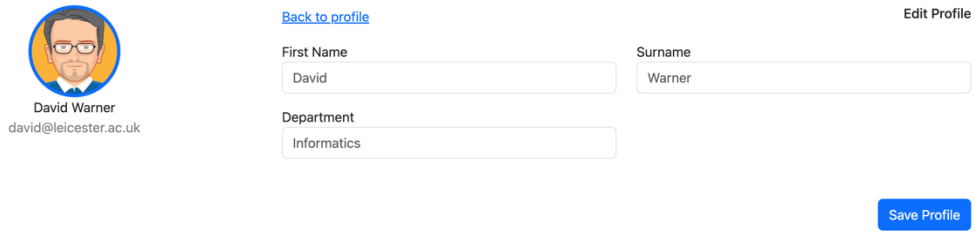


Figure 10.1.7 – Edit profile

An additional feature of the application is to allow should submit the topic if any supervisors approve, he would be in charge. The Figure 10.1.7 show is the page where could submit their topic.

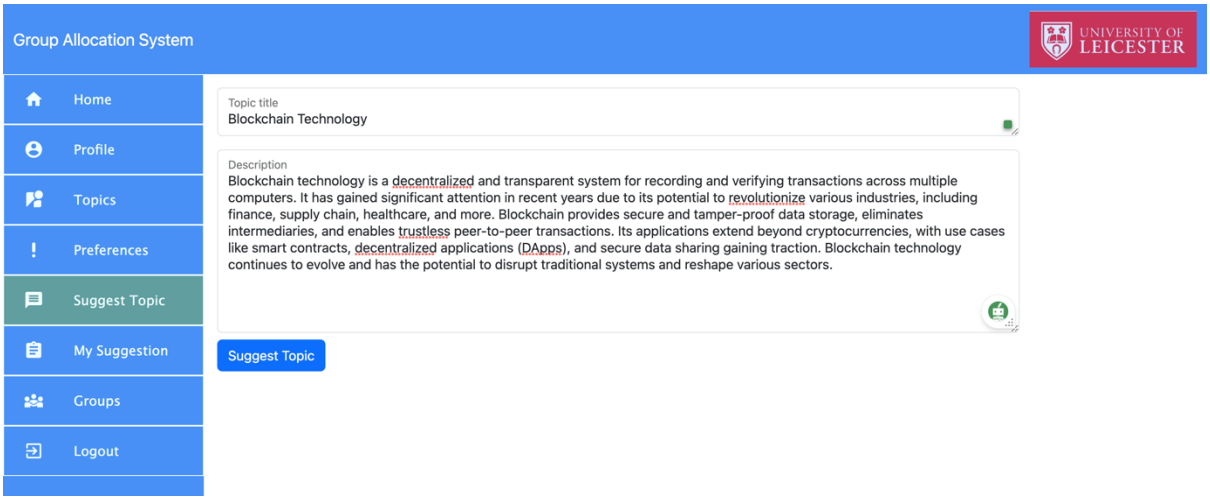


Figure 10.1.8 – Suggest Topics (students)

There is also an opportunity for the students to edit or delete the topic which is available in their application as shown in the below Figure 10.1.9.

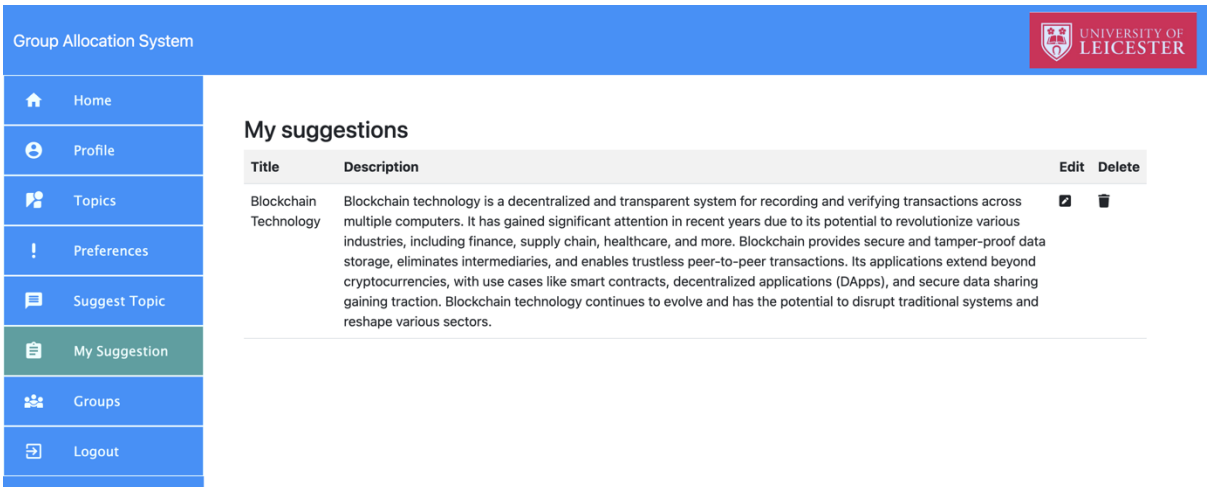


Figure 10.1.9 – Suggested Topics (students)

Topics which are uploaded and available in the module would be displayed as shown in the Figure 10.1.10 for the supervisor, as seen the edit icon for the supervisor Louis Hamilton is disabled, since user the logged to application is David warner the feature is developed where one could only edit or remove his own topic.

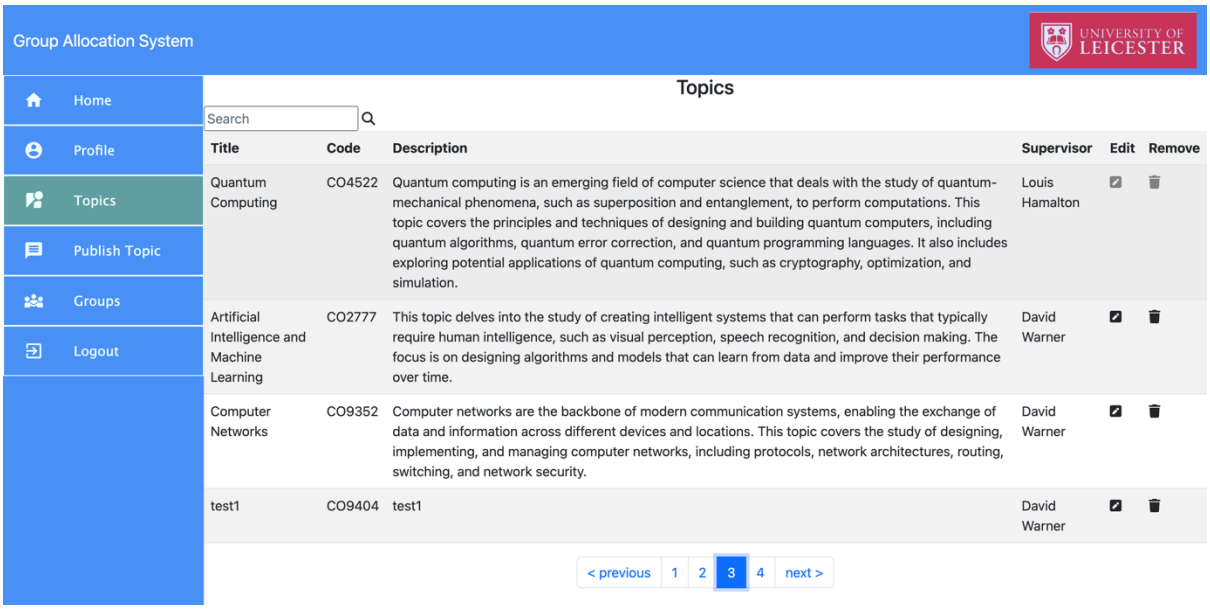


Figure 10.1.10 – Topics Table (Supervisor view)

Similarly for the students only view access is provided and the topic available in the module would be as shown in Figure 10.1.11.

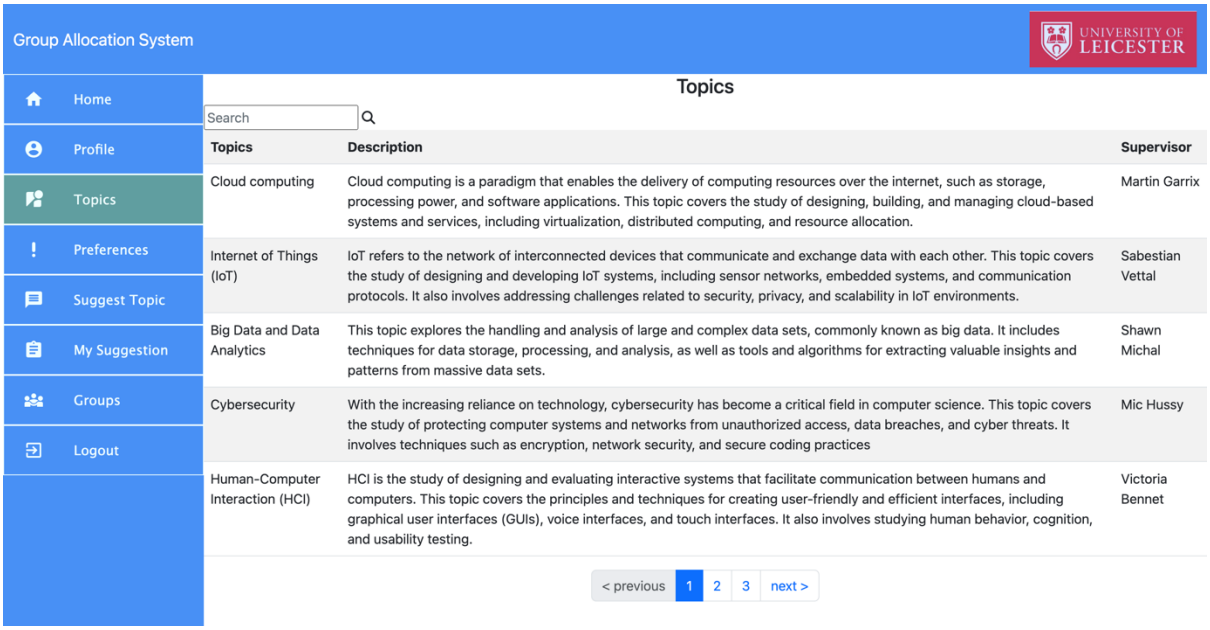


Figure 10.1.11 – Topics Table (Student View)

If any of the student have issue in submitting the preference administrator could be able to do on behalf of them or any changes in the preference required would be done in the view preference tab available for the administrator. The page would render with topics and preferences as shown in Figure 10.1.12

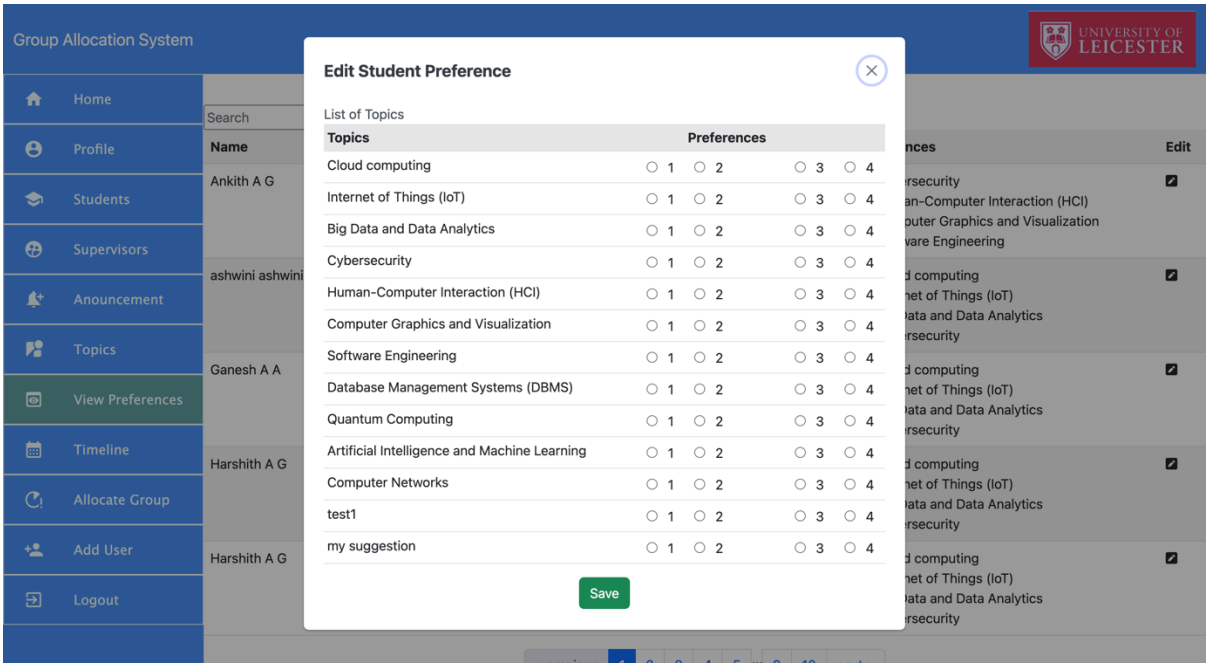


Figure 10.1.12 – Edit Students Preferences (admin)

A feature announces the notification, or an occurrence would be done by the administrator for users of the application which would appear on users dashboard. This announcement could be edited, remove and temporally disabled. As seen in Figure 10.1.13.

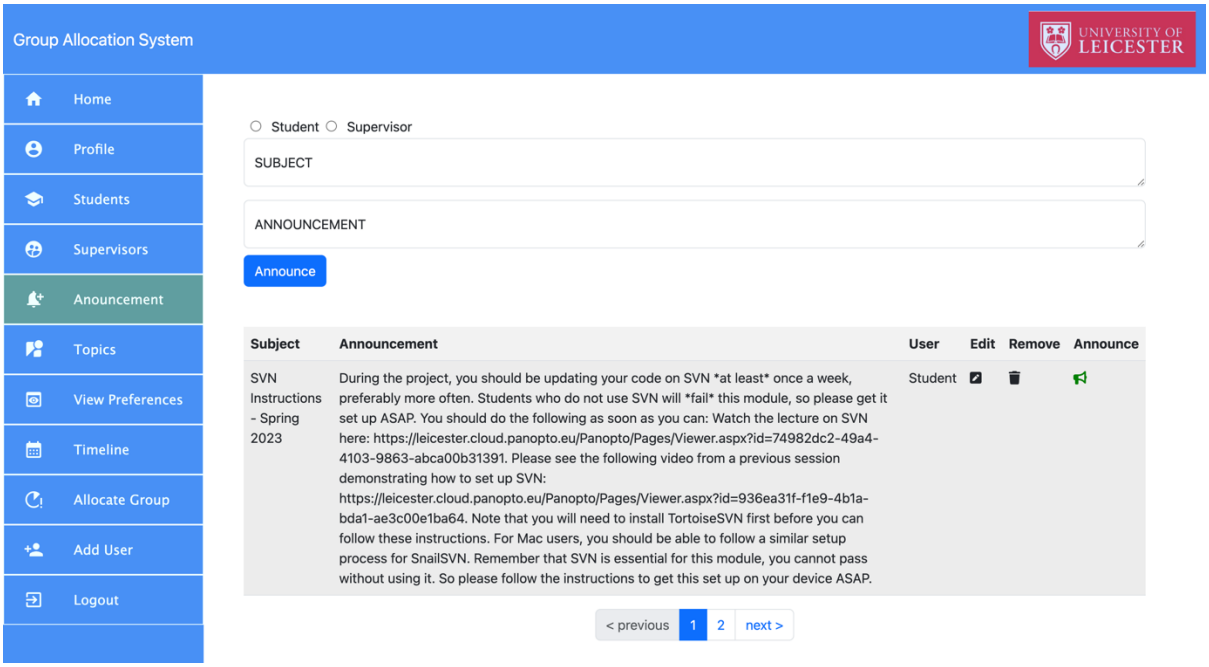


Figure 10.1.13– Announcement (admin)

Users in the application could be register by the administrator as shown in the below Figure 10.1.14.

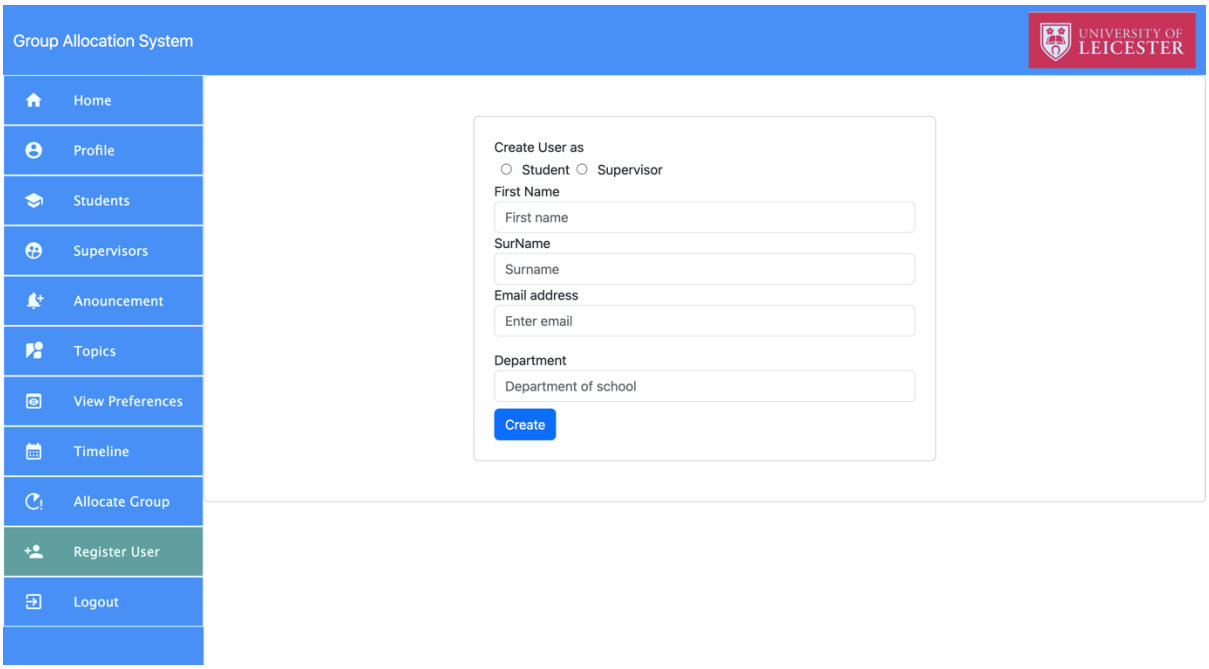


Figure 10.1.14– Register User (admin)

Once the administrator announces the allocation is students is assigned to the groups would be displayed with the group information else it would say “You haven’t been assigned to any groups”.

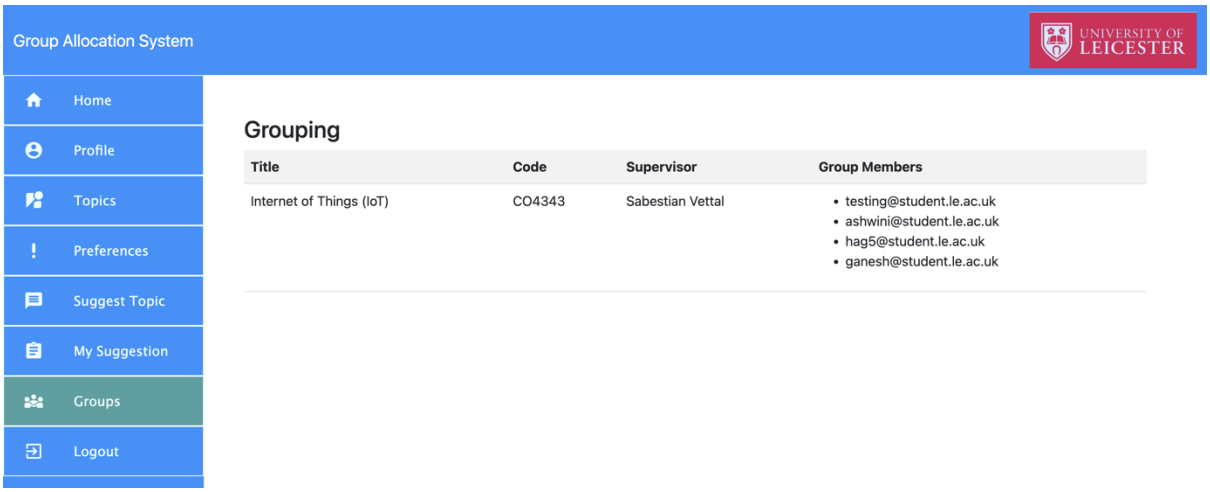


Figure 10.1.15 – Group Information (Student)

10.2 Appendix B – Testcase

Here in this section the author has provided some important testcase which is developed against the software application and tested these scenarios in the software. These testcase have been developed for all the modules in the application and considered for all users.

TestCase ID	Module	Description	Prerequsites	Steps	Expected Result	Actual Result	Status
T_01	Regitration	Student Should be able to register to the app with required details	Student should have an email id and few other details to registered to the application.	1. Fill all the required details to register students to the application. 2. Click the submit button	Create credentials with the entered details, Student should login with the created credentials.	Same as Expected result	PASS
T_02	Regitration	Supervisor Should be able to register to the app with required details	Supervisors should have an email id and few other details to registered to the application and the secret code provided by the admin to register.	1. Fill all the required details to register Supervisors to the application. 2. Click the submit button	Create credentials with the entered details, Supervisor should login with the created credentials.	Same as Expected result	PASS
T_03	Regitration	Application should not allow supervisor to get registered without secret code	Supervisors should have an email id and few other details to registered to the application.	1. Fill all the required details to register Supervisors to the application. 2. Enter invalid code in sceret code. 3. click on submit button.	App should not allow supervisor to register.	Same as Expected result	PASS
T_04	Regitration	User Should not be able to register to the app with invalid details	Invalid data	1. Fill all the required with invalid details not to register to the application. 2. Click the submit button	App should not create credentials with the entered details.	Same as Expected result	PASS
T_05	Login	Student Should be able to login to the app with created credentials	Student must have register to the application	1.Enter valid details 2.click on login	Should display students dashboard	Same as Expected result	PASS
T_06	Login	Supervisor Should be able to login to the app with created credentials	Supervisor must have register to the application	1.Enter valid details 2.click on login	Should display supervisor dashboard	Same as Expected result	PASS
T_07	Login	User not should be able to login to the app with created credentials	User must have register to the application	1.Enter invalid details 2.click on login	Should display Invaild credentials	Same as Expected result	PASS
T_08	Login	Admin Should be able to login to the app with	Admin must have register to the application	1.Enter valid details 2.click on login	Should display Admin dashboard	Same as Expected result	PASS

Login and registration | User Profile | View students and supervisors | Topics | Allocate Group | Upload topics | View student preference | Timeline | Admin Annnouncement

Figure 10.2.1 – Login and registration module

T_01	User Profiles	Verify whether the name , role email of the user logged in to application is visble	User Must be registered to the application.	1. Enter the registered Credentials in the loginpage 2.click on login 3. Click on profile	Should display the details of the user who logged in	Same as Expected result	PASS
T_02	User Profiles	Verify whether the user is able to edit the profile data	User Must be registered to the application.	1. Enter the registered Credentials in the loginpage 2.click on login 3. Click on profile 4. Click on edit button 5.Enter the details which need to be edited 6.Click on save 7.Verify wheather the details is updated	The entered details needs to be edited.	Same as Expected result	PASS
T_03	User Profiles	Verify whether the user is able to navigate back by clicking	User Must be registered to the application.	1. Enter the registered Credentials in the loginpage 2.click on login 3. Click on profile 4. Click on edit button 5.click on go back	The user should direct back to profilr page from the edit profile page	Same as Expected result	PASS
T_04	User DashBoard	Verify whether the user is able to View the announcement in the dashboard	User Must be registered to the application.	1. Enter the registered Credentials in the loginpage 2.click on login	Students should be displayed with the students announcements or notification. Supervisors should be displayed with the supervisor announcements or notification and admin with admin notification	Same as Expected result	PASS
T_05	User DashBoard	Verify whether the dashboard is updated with the new announcement made by the admin	User Must be registered to the application. 1.Need to login as admin and student and supervisor	1. Login as admin and make an announcement to a students 2. login as studenst and check the announce ment made is displayed in the student dashboard. 3. Repeat the same for the supervisor and login as the supervisor and check whteher the right annoncement is displayed.	Students should be displayed with the students announcements or notification. Supervisors should be displayed with the supervisor announcements or notification and admin with admin notification	Same as Expected result	PASS
T_06	Logout	Verify whether the application gets logged out when the user click on the logout. Verify it for admin, student and supervisor.	User Must be registered to the application.	1. Enter the registered Credentials in the loginpage 2.click on login 3.click on logout	User should be logged out of the application	Same as Expected result	PASS

Login and registration | User Profile | View students and supervisors | Topics | Allocate Group | Upload topics | View student preference | Timeline | Admin

Figure 10.2.2 – Users profile

TestCase ID	Module	Description	Prerequsites	Steps	Expected Result	Actual Result	Status
T_01	View Students and supervisor for admin	Admin should be able to view all the students registered to the application	User should be admin	1.Login as an admin 2.Click on students 3. Verify all the students are displayed in the application in the compare with the database	All the details of the students in the database should be displayed in the table format	Same as Expected result	PASS
T_02	View Students and supervisor for admin	Admin should be able to view all the supervisors registered to the application	User should be admin	1.Login as an admin 2.Click on supervisors 3. Verify all the supervisors are displayed in the application in the compare with the database	All the details of the supervisors in the database should be displayed in the table format	Same as Expected result	PASS
T_03	View Students and supervisor for admin	Admin should be able to edit details of students registered to the application	User should be admin	1.Login as an admin 2.Click on students 3.click on edit icon 4.Enter the details that needs to be edited 5.click on save	Details entered that requires to be edited should be updated with new entered details.	Same as Expected result	PASS
T_04	View Students and supervisor for admin	Admin should be able to delete the students registered to the application	User should be admin	1.Login as an admin 2.Click on students 3.click on delete icon	The student should be removed form the table and database.	Same as Expected result	PASS
T_05	View Students and supervisor for admin	Admin should be able to disable the students registered to the application	User should be admin	1.Login as an admin 2.Click on students 3.click on toggle buttons	The student account should be disabled. Try logging into the disabled account.	Same as Expected result	PASS
T_06	View Students and supervisor for admin	Admin should be able to enable the students registered to the application.	User should be admin	1.Login as an admin 2.Click on students 3.click on toggle buttons	The student account should be enable. Try logging into the enable account. Users should be logged into the account.	Same as Expected result	PASS

Login and registration | User Profile | View students and supervisors | Topics | Allocate Group | Upload topics | View student preference | Timeline | Admin Annnou

Figure 10.2.3 – View students and supervisors

T_07	View Students and supervisor for admin	Admin should be able to disable the students registered to the application	User should be admin	1.Login as an admin 2.Click on students 3.click on toggle buttons	The student account should be disabled and the toggle icon should be changed to red colour icon.	Same as Expected result	PASS
T_08	View Students and supervisor for admin	Admin should be able to enable the students registered to the application.	User should be admin	1.Login as an admin 2.Click on students 3.click on toggle buttons	The student account should be disabled and the toggle icon should be changed to green colour icon.	Same as Expected result	PASS
T_09	View Students and supervisor for admin	All the details of the should be displayed in a tabular format and should load the next set of data	User should be admin	1. Login as an admin 2. Click on students 3. Click on next page or prev to check the pagination	Student data from the database should be displayed.	Same as Expected result	PASS
T_10	View Students and supervisor for admin	All the details of the should be displayed in a tabular format and should load the next set of data	User should be admin	1. Login as an admin 2. Click on supervisor 3. Click on next page or prev to check the pagination	Supervisor data from the database should be displayed.	Same as Expected result	PASS
T_11	View Students and supervisor for admin	Admin should be able to disable the supervisor registered to the application	User should be admin	1.Login as an admin 2.Click on supervisor 3.click on toggle buttons	The supervisor account should be disabled and the toggle icon should be changed to red colour icon.	Same as Expected result	PASS
T_12	View Students and supervisor for admin	Admin should be able to enable the supervisor registered to the application.	User should be admin	1.Login as an admin 2.Click on supervisor 3.click on toggle buttons	The supervisor account should be disabled and the toggle icon should be changed to green colour icon.	Same as Expected result	PASS
T_13	View Students and supervisor for admin	Admin should be able to disable the supervisor registered to the application	User should be admin	1.Login as an admin 2.Click on supervisor 3.click on toggle buttons	The supervisor account should be disabled. Try logging into the disabled account.	Same as Expected result	PASS

Login and registration | User Profile | View students and supervisors | Topics | Allocate Group | Upload topics | View student preference | Timeline | Admin Annou

Figure 10.2.4 – View students and supervisors

	for admin	application.		3.click on toggle buttons	should be changed to green colour icon.		
T_13	View Students and supervisor for admin	Admin should be able to disable the supervisor registered to the application	User should be admin	1.Login as an admin 2.Click on supervisor 3.click on toggle buttons	The supervisor account should be disabled. Try logging into the disabled account.	Same as Expected result	PASS
T_14	View Students and supervisor for admin	Admin should be able to enable the supervisors registered to the application.	User should be admin	1.Login as an admin 2.Click on supervisor 3.click on toggle buttons	The supervisor account should be enable. Try logging into the enable account. Users should be logged into the account.	Same as Expected result	PASS
T_15	View Students and supervisor for admin	Admin should be able to search the supervisors registered to the application.	User should be admin	1.Login as an admin 2.Click on supervisor 3.Enter the student name to be searched 4.Click on search icon	Should display the data if that student is available.	Same as Expected result	PASS
T_16	View Students and supervisor for admin	Admin should be able to search the supervisors registered to the application.	User should be admin	1.Login as an admin 2.Click on supervisor 3.Enter the supervisor name to be searched 4.Click on search icon	Should display the data if that supervisor is available.	Same as Expected result	PASS
T_17	View Students and supervisor for admin	Admin should not be able to search the invalid supervisors registered to the application.	User should be admin	1.Login as an admin 2.Click on supervisor 3.Enter an invalid supervisor name to be searched 4.Click on search icon	Should display the data not available.	Same as Expected result	PASS
T_18	View Students and supervisor for admin	Admin should be not able to search the invalid students registered to the application.	User should be admin	1.Login as an admin 2.Click on students 3.Enter an invalid student name to be searched 4.Click on search icon	Should display the data not available.	Same as Expected result	PASS

Login and registration | User Profile | View students and supervisors | Topics | Allocate Group | Upload topics | View student preference | Timeline | Admin Annou

Figure 10.2.5 – View students and supervisors

TestCase ID	Module	Description	Prerequsites	Steps	Expected Result	Actual Result	Status
T_01	Topics table	Admin should be able to see all the topics uploaded by stupervisors and students	User should be admin	1.Login as an admin 2.Click on topics	Table of topics should be listed	Same as Expected result	PASS
T_02	Topics table	Admin should be able to see all the topics uploaded by stupervisors and students and edit, delete and search	User should be admin	1.Login as an admin 2.Click on topics 3.Enter the text to search the topic 4.Edit the topic 5. click on delete	Admin should be able to search the topic , edit , delete it	Same as Expected result	PASS
T_03	Topics table	supervisor should be able to see all the topics	User should be supervisor	1.Login as an supervisor 2.Click on topics	Table of topics should be listed	Same as Expected result	PASS
T_04	Topics table	supervisor should be able to see all the topics delete or edit own topic	User should be supervisor	1.Login as an supervisor 2.Click on topics 3. click on edit and delete	Topic from the table should be deleted	Same as Expected result	PASS
T_05	Topics table	students should be able to see all the topics	User should be student	1.Login as an student 2.Click on topics	Table of topics should be listed and can search the topics	Same as Expected result	PASS
T_06	Preference	Should should be able to set their preference by selecting the radio button	User should be student	1.Login as an student 2.Click on prefernces 3. slect preferences and save	Set preference should be displayed in the same page and should be update in the admin prefernce also	Same as Expected result	PASS
T_07	Preference	Should should be able to set their preference by selecting the radio	User should be student	1.Login as an student 2.Click on prefernces	Set preference should be displayed in the same page	Same as Expected result	PASS

Login and registration | User Profile | View students and supervisors | Topics | Allocate Group | Upload topics | View student preference | Timeline | Admi

Figure 10.2.6 – Topics

TestCase ID	Module	Description	Prerequsites	Steps	Expected Result	Actual Result	Status
T_01	Allocate Group	Group of students with preferred topic should be formed	User Should be admin	1. Log in as admin 2. click on allocate group 3. By default group size should be 4	Group of 4 students should be formed	Same as Expected result	PASS
T_02	Allocate Group	Group of students with preferred topic should be formed	User Should be admin	1. Log in as admin 2. click on allocate group 3. enter the group size click on allocate group	Group of enter size should be formed students should be formed	Same as Expected result	PASS
T_03	Allocate Group	Group of students should be cleared	User Should be admin	1. Log in as admin 2. click on allocate group 3. click on clear allocation	Allocation made previsously should be cleared	Same as Expected result	PASS
T_04	Allocate Group	Group of students with preferred topic should be formed	User Should be admin	1. Log in as admin 2. click on allocate group 3. click on allocate group	Verify wheather the group created is right and as expected	Same as Expected result	PASS
T_05	Allocate Group	Group of students with preferred topic should be formed	User Should be admin	1. Log in as admin 2. click on allocate group 3. click on allocate group 4. Verify the grop size entered is great than 2	Pop should be displayed if user enters invalid data	Same as Expected result	PASS
T_06	Allocate Group	Group of students with preferred topic should be formed and email should be sent	User Should be admin	1. Log in as admin 2. click on allocate group 3. click on allocate group 4. Click on sne demail	Email should be senty	Same as Expected result	PASS

Login and registration | User Profile | View students and supervisors | Topics | Allocate Group | Upload topics | View student preference | Timeline | Ad

Figure 10.2.7 – Allocate Group

				3. click on clear allocation	cleared		
T_04	Allocate Group	Group of students with preferred topic should be formed	User Should be admin	1. Log in as admin 2. click on allocate group 3. click on allocate group	Verify wheather the group created is right and as expected	Same as Expected result	PASS
T_05	Allocate Group	Group of students with preferred topic should be formed	User Should be admin	1. Log in as admin 2. click on allocate group 3. click on allocate group 4. Verify the grop size entered is great than 2	Pop should be displayed if user enters invalid data	Same as Expected result	PASS
T_06	Allocate Group	Group of students with preferred topic should be formed and email should be sent	User Should be admin	1. Log in as admin 2. click on allocate group 3. click on allocate group 4. Click on sne demail	Email should be senty	Same as Expected result	PASS
T_07	Allocate Group	Group of students with preferred topic should be formed for same preference	User Should be admin	1. Log in as admin 2. click on allocate group 3. click on allocate group	Verify wheather the group created is right and as expected	Same as Expected result	PASS
T_08	Allocate Group	Group of students with preferred topic should be formed for different preference	User Should be admin	1. Log in as admin 2. click on allocate group 3. click on allocate group	Verify wheather the group created is right and as expected	Same as Expected result	PASS

Login and registration | User Profile | View students and supervisors | Topics | Allocate Group | Upload topics | View student preference | Timeline | Ad

Figure 10.2.8 – Allocate Group

T_01	Publish topic	Supervisor should be able to submit the topics	User should be the supervisor	1.Login as supervisor 2.clcik on publish topic 3.Enter the details of the topic that need to be added 4.click on upload	New topic need to be added	Same as Expected result	PASS
T_02	Publish topic	Supervisor should be able tosee topics suggested by students	User should be the supervisor	1.Login as supervisor 2.clcik on publish topic 3.A table with topic details should be displayed	A table with student details should be displayed	Same as Expected result	PASS
T_03	Publish topic	Supervisor should be able tosee topics suggested by students and approve	User should be the supervisor	1.Login as supervisor 2.clcik on publish topic 3.A table with topic details should be displayed 4. Enter the code which supervispr would like to talke up	The topic the supervisor have approved should be allocation under that supervisors name	Same as Expected result	PASS
T_04	Suggest topics	Students should be able to submit the topics	User should be the students	1.Login as supervisor 2.clcik on publish topic 3.Enter the details of the topic that need to be added 4.click on upload	New topic need to be added to supervisor to view.	Same as Expected result	PASS
T_05	my suggestion	Students should be able to submit the topics and see it in my suggestion	User should be the students	1.Login as supervisor 2.clcik on suggest topic 3.Enter the details of the topic that need to be added 4.click on upload 5. Should be displayed in my section tab	New topic need to be added should be displayed	Same as Expected result	PASS
T_06	my suggestion	Students should be able to submit the topics and see it in my suggestion able to delete /edit	User should be the students	1.Login as supervisor 2.clcik on suggest topic 3.Enter the details of the topic that need to be added 4.click on upload	New topic need to be added should be displayed those topics need to be edited and deleted	Same as Expected result	PASS

Login and registration | User Profile | View students and supervisors | Topics | Allocate Group | Upload topics | View student preference | Timeline | Ad

Figure 10.2.9 – Upload Topics

TestCase ID	Module	Description	Prerequsites	Steps	Expected Result	Actual Result	Status
T_01	Student preference table table	Admin should be able to see all the preferesnce students	User should be admin	1.Login as an admin 2.Click on preferences	Table of preference should be listed	Same as Expected result	PASS
T_02	Student preference table table	Admin should be able to edit the preferesnce students	User should be admin	1.Login as an admin 2.Click on preferences 3. click on search and edit	Table of preference should be listed should be edited	Same as Expected result	PASS
T_03	Student preference table table	Admin should be able to add the preferesnce students for students who have not set the preference	User should be admin	1.Login as an admin 2.Click on preferences 3. click on edit	Table of preference should be listed should be edited	Same as Expected result	PASS

Login and registration | User Profile | View students and supervisors | Topics | Allocate Group | Upload topics | View student preference | Timeline

Figure 10.2.11 – Student view preferences(admin)

TestCase ID	Module	Description	Prerequsites	Steps	Expected Result	Actual Result	Status
T_01	Timeline	Admin should be able to see all the button too enable the page and disable the page	User should be admin	1.Login as an admin 2.Click on Timline	List of buttons need to displayed	Same as Expected result	PASS
T_02	Timeline	Admin should be able to click the button and diable and anable the pages as per the button	User should be admin	1.Login as an admin 2.Click on Timline 3. click on students preference/supervisor upload topic/ disable view allocation	Button with the relavent page should be disabled/enabled	Same as Expected result	PASS
T_03	Timeline	Admin should be able to send an email	User should be admin	1.Login as an admin 2.Click on Timline 3. Select the typr user to send email 4.Enter the subject 5.clcik on notify	An email should be sent.	Same as Expected result	PASS

Login and registration | User Profile | Timeline | View students and supervisors | Topics | Allocate Group | Upl

Figure 10.2.12 – Timeline

TestCase ID	Module	Description	Prerequsites	Steps	Expected Result	Actual Result	Status
T_01	Admin Announcement	Admin should be able to Announce to the details which need to be displayed in the users dashboard.	User should be admin	1.Login as an admin 2.Click on Announcement 3.Check the radio button which the details needs to be announced to. 4.Enter the subject 5.Enter the Announcement 6. Click on Announce	The announcement which is made by the admin needs to be stored in the database and dispalyed to the relevent users once after they loggin to the application.	Same as Expected result	PASS
T_02	Admin Announcement	Admin should be able to Announce to the details to the students	User should be admin	1.Login as an admin 2.Click on Announcement 3.Check the student radio button . 4.Enter the subject 5.Enter the Announcement 6. Click on Announce	The announcement which is made by the admin needs to be stored in the database and dispalyed to the to students once after they loggin to the application.	Same as Expected result	PASS
T_03	Admin Announcement	Admin should be able to Announce to the details to the supervisor	User should be admin	1.Login as an admin 2.Click on Announcement 3.Check the supervisor radio button . 4.Enter the subject 5.Enter the Announcement 6. Click on Announce	The announcement which is made by the admin needs to be stored in the database and dispalyed to the to supervisors once after they loggin to the application.	Same as Expected result	PASS
T_04	Admin Announcement	Verify whether all the mandatory fields are are filled and application should not allow to annonce without mandatory fields.	User should be admin	1.Login as an admin 2.Click on Announcement 3.Check the radio button which the details needs to be announced to. 4.Enter the subject 5.Enter the Announcement 6. Click on Announce	Should not allow application to announce without filling the required fields.	Same as Expected result	PASS

Login and registration | User Profile | Timeline | View students and supervisors | Topics | Admin Annnouncement | Allocate Group | Upload top

Figure 10.2.12 – Announcement(admin)