

Table of Contents

Introduction:	3
Motivation.....	3
Problem Definition:.....	3
Challenges:	3
Limitations of the project:	3
Background research:	3
Milestones:.....	4
RISKS:	5
Requirements:.....	6
Essential:	6
Recommended:	7
Optional:	7
Technical Specifications:	8
Visual studio code:	9
NodeJS.....	10
RESTful Web Services:.....	10
MongoDB	10
Project outcomes:	11
Login redirection page:	11
Login page:	12
Submit the preference page:	13
Admin Dashboard:	15
Add topics page:.....	15
View topics page:	16
View preferences page:	17
View allocations page:	18
Supervisor dashboard:	18
Submit supervising topic page:	19
APIs:	20
Endpoints:	20
Login module:	22
Student (create):	23

API for viewing approved topics:	25
API for selecting preferences	26
Admin	28
API for adding topics	29
API for viewing preferences	30
API for publishing allocations.....	31
Supervisor	31
Algorithm for allocation groups:	31
Following the web development process:	34
Planning:	35
Web site layouts:	35
Build and running of application:.....	36
Further updates:	37
Code snippets:.....	38
Code for taking students preferences:	38
Code for showing JSON data in a table with GET method:.....	39
Plugins and extras:	39
Particle JS:	39
Tailwind CSS:	40
Noah web development template:.....	41
SASS.....	41
BEM methodology:	43
Testing:.....	43
Functionality testing:	44
Usability testing:	45
Testing algorithm and its efficiency:	45

Introduction:

Motivation:

The allocation of project topics for students is always time taking and requires a person to dedicate his time to allocating projects/topics for the students. It'll be more complicated if there are many students and fewer topics or many students preferring the same topic or even in some cases where a single topic needs to be allocated to a group of students.

Problem Definition:

In our university, currently, we are facing problem for Personal and group skills module where a topic needs to be allocated to group of five students and all these processes is done manually by our professors or admin team by sending topics list to every student registered in this module and each student submits their preferred topics in an order and then admin needs to check the order of preference of each student to each project and carefully allocate project topic to each student by making sure they are allocated with their preferred topic and then sending emails to each student with topic allocated to them along with names of their other group members. The main objective of the project is to make this entire process automated with limited admin intervention.

Challenges:

Most of the project is not so complicated, as it is only a web application which stored user preferences and stored them in the database and then run a matching algorithm, which is the most challenging task in this project, as we have 5 preferences for each student and then we need to allocate 5 students for each topic and also make sure that each student is allocated with their top preferred topic.

Limitations of the project:

- Users have convenient access without major security requirements.
- It will not work if there is no internet connection.
- Less interaction between students and professors.

Background research:

During this phase, I found that admin/tutor currently sending emails to students with all the topics which are approved and asks all the students to submit their 5 preferences in order.

Admin then collects all these preferences of students and allocates them into groups.

As this is not once only work, the admin needs to do this process for every semester for students who choose the personal and group skills module.

Considering this, we have decided to do a web application that can be accessed online where admin can add topics and students submit their preferences, the admin runs an algorithm that allocated them to groups.

Milestones:

- Till November 7, we need to complete simple wireframes, finalizing application flows and designing the views. Also, create a project structure and a GitHub repository. This includes login pages for different roles (student, admin, supervisor), there will be 2 views for the student, one for submitting preference and one for viewing the allocated groups. 2 views for admin, one for managing the topics and one for viewing the submission. 2 views for supervisor, one for adding the topics and other for selecting the topics which they want to supervise and this view will be same for both admin and supervisor but admin just can see them, but supervisor can select one from them.
- By the end of November 3rd week, we need to complete a database for storing login details, topic details and preferences, build APIs for storing the topics into the database which are added by admin/supervisor and also integrate them.
- Till December 1st week, we need to complete total web application which includes admin adding topics and students must able to submit their preferences and at this stage, our project must be ready to implement the matching algorithm.
- By the end of December, we must complete implementing the matching algorithm, here we are using a bipartite weighted matching algorithm or any other algorithm similar which fits into our problem, for allocation of groups.
- In the above-mentioned plan, each step includes some part of writing report from the work done till date by which we can use time in the end for testing and also able to add some of the optional features which mentioned before.

The above plan mentioned is risk-free, as the time for each section is allocated by taking care of risks that may arise and cause a delay in developing the project.

RISKS:

Risk1: The author has no experience in developing a complete web application, Creating a database, Creating API's and integrating with front-end.

Action: In order to build a complete web application, one must know how to develop it in every phase.

Risk 2: While implementing the algorithm for allocating groups. We cannot guarantee them that we will allocate the, with their 1st preferred topic.

Action: We Implemented algorithms in such a way that, it tries to allocate their first preferred topic In most cases unless the topic has already allocated to 5 students and there is no chance of adding new students into that group. In this case, the algorithm tries to allocate with their second preferred topic and continues with the 3rd, 4th and 5th topics in case of the same situation where the topic is already allocated.

Risk 3: Implementing a weighted algorithm forces students to allocate them their lowest preferred topic even when the highest preferred topic is available. This situation occurs when a group is allocated with 4 students with their highest preferred topic as their group topic, but the 5th student is forced to allocate this group even if his highest preferred topic is available.

Action: We implemented an algorithm which first tries to allocate their highest preferred topic unless the group is allocated with 5 students.

Risk 4: Get values from Database and display them in the table.

Action: To make sure that all our values are displayed from the API with the GET method, we used a code from an online source to display values in the table dynamically from an API.

Requirements:

The main objective of the project is to make this entire process automated with limited admin intervention.

Features:

- Initially, admin adds topics by logging into the portal which we store in our database and submits them which sends emails to students registered to that module that topics are added and preference needs to be submitted.
- Students then log in to the portal and submit topics (let's say as 5 topics) as their preference. These topics are stored in our database.
- After all the students submitting their preferences, the admin checks them through a new page where all student's responses are displayed, and then he runs a matching algorithm that allocates topics and groups for students.
- In our case, matching algorithm is a program that added weights to the topics by their priority (example: first preferred topic is assigned with the highest weight and lowest to the least preferred and calculate some kind of algorithm which can make us allocate topics accordingly).
- After running a matching algorithm admin can view the allocated topics and groups, and if he is fine with the allocation, he/she can publish the allocation where students can check them by logging into the portal or by sending a mail with an allocated topic and group members. we are using a bipartite weighted matching algorithm or similar algorithm which is more efficient where every preference is allocated with some weight. This is a common approach which suits for many problems but the tricky part is we must allocate 1 topic to maximum of 5 students, if it is just assigning only 1 student a topic we can directly use this algorithm, but In our case, we need some more logic where we must allocate accordingly.

Essential:

- We have 2 main users, admin, and users. Here student needs two views, one for submitting the preferences and for viewing the allocation.

- Admin needs 3 views, one for adding the topics, one for viewing preferences of each student and final view which displays the allocation (it's the same view which is shown for students too).
- We need a database for storing login details, topics, preferences, and allocations.
- Login system for students and admin.
- API's also needs to be designed to submit the topics initially and store them into the database and also for submitting the preferences of students.
- One of the main requirements of this project is implementing the matching algorithm.

Recommended:

Admin must able to do all the actions of students and supervisors.

Admin must able to submit a preference on behalf of a student

Admin must able to submit a supervising topic on behalf of a supervisor

Admin must able to edit the preferences of student and supervisor

Admin must able to edit allocated groups.

Make the web application such a way that it meets WCAG requirements.

Making sure allocation is done properly when most of them choose the same topic as their first preference.

Students can make changes to their preferences even after submitting for a while.

Optional:

Asking students their preferred group members.

Linking this to the university website so that all the users can log in with their university IT accounts.

Making application more secure

Making application responsive to use them on any device, Currently, we are displaying the values of all the student names, allocations, groups as tables for which we cannot make them responsive.

More plugins and libraries to make better UI/UX

Can be able to merge with UOL application

Sending an email to students before the deadline, who are registered to this module, when they didn't send their response (which is currently done manually)

Taking the user's preference first and recommend topics according to their preferences.

Technical Specifications:

Type	Name	Version	Details
Programming language	<ul style="list-style-type: none">• HTML• JavaScript• J Query• AJAX• Node JS	<ul style="list-style-type: none">• 5.0• N/A• 3.3.1• N/A• 9.11.2	<ul style="list-style-type: none">• Mark-up language for our web pages• Scripting language for form validations, page navigations, etc.• JavaScript library for easy use of JS• For all API calls and loading the data onto the page• Server-side language for API's

Framework	<ul style="list-style-type: none"> • Laravel • Tailwind 	<ul style="list-style-type: none"> • 6.0 • 1.1.4 	<ul style="list-style-type: none"> • Framework for building web applications • Front end framework for styling webpages
Database	MongoDB	4.0.10	<ul style="list-style-type: none"> • cross-platform document-oriented database program
IDE	Visual studio code	1.27.0	<ul style="list-style-type: none"> • source-code editor developed by Microsoft for Windows, Linux, and macOS
Version Control	GIT	2.5	<ul style="list-style-type: none"> • Git is a distributed version control system for tracking changes in source code

Visual studio code:

In these days where many developers use different IDE's for developing, Visual studio code stands at the top for developing web-based applications

It provides many extensions for fast developing and coding by highlighting errors or accessibility errors while typing as these things small things make a huge change at the end and unable to point out them as the project scale increases day by day.

For writing HTML and CSS visual studio code is the best option as it has inbuilt features like closing tags, picking colors, selecting properties, selectors and folding the blocks of code.

More information for using visual studio code for web development can be found here(<https://code.visualstudio.com/docs/languages/overview>).

NodeJS:

In this project we use NodeJS for backend language for developing API's, which are application programming interface, in our project, these are the interfaces for collecting student's response and submitting them in a database and then also bring back them by admin for running a matching algorithm

RESTful Web Services:

"A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., communication between Java and Python, or Windows and Linux applications) is due to the use of open standards.

Web services based on REST Architecture are known as RESTful web services. These web services use HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI, Uniform Resource Identifier a service, which provides resource representation such as JSON and set of HTTP Methods." [https://www.tutorialspoint.com/nodejs/nodejs_restful_api.htm]

MongoDB:

"MongoDB is an open, non-tabular database developed by MongoDB, Inc. MongoDB stores data as documents in a binary representation called BSON (Binary JSON). Related information is stored together for fast query access through the MongoDB query language. Fields can vary from document to document; there is no need to declare the structure of documents to the system, as documents are self-describing. If a new field needs to be added to a document, then the field can be created without affecting all other documents in the collection, without updating a central system catalog,

and without taking the system offline. Optionally, schema validation can be used to enforce data governance controls over each collection.” [<https://www.mongodb.com/compare/mongodb-mysql>]

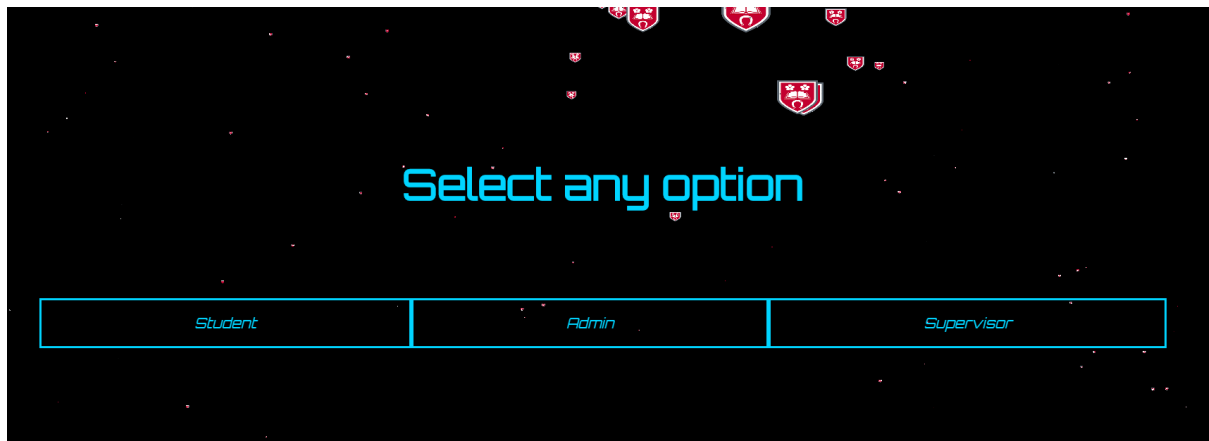
Project outcomes:

The project contains a web application for students, admin, and supervisors. Every user can access the application after successfully completing their login process. Each has different views to use the application. Views on the application include

- Login redirection page (For all the users)
- Login page (For all the users)
- Submit the preference page (Only for students)
- Admin dashboard
 - Add topics page (For admin and supervisors)
 - View and publish topics page (Only for admin)
 - View preferences page (Only for admin)
 - View allocations page (Only for admin)
- Supervisor dashboard
 - Add topics page (For admin and supervisors)
 - Supervising group page (Only for supervisors)

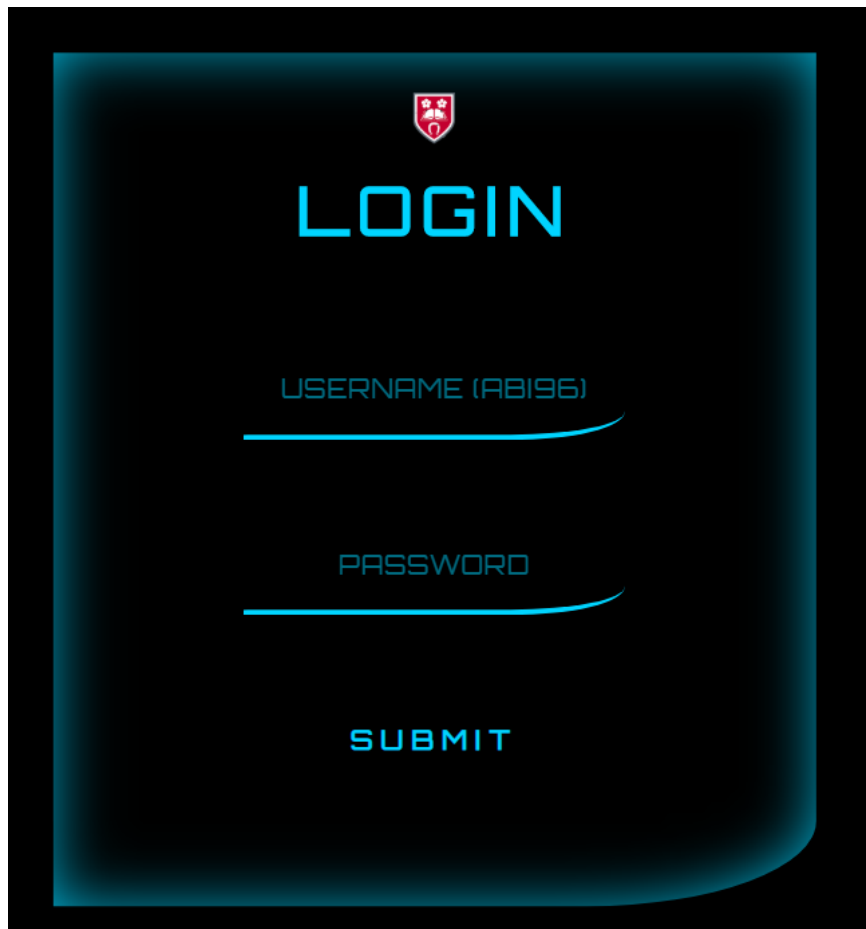
Login redirection page:

This is the first page where every user can see when they open the web application. In our case where we have 3 different types of users, we need to know which user wants to log in. This page has 3 buttons each for admin, student, and supervisor which redirects them to their required login pages.



Login page:

This page is also available for all the users. This page allows all users to log in to the web application. Every user can log in with their username/email and their password and submit them to complete their login process. Based on the type of user, further pages are shown to them.



Submit the preference page:

This page is only available for students to submit their preferences. This page contains all the available topics once the approved topics are published by the admin. Students need to submit their top 5 preferred topics by dragging and dropping them in the required order. A video popup that shows how to submit preferences in the required order is also added after each login so that students can submit them without any doubts and also avoids submitting wrong preferences. For further understanding of their priority colors from green to red are allocated for their preferences so students can know their top preferred topic while submitting. If the student submits the topics without selecting them in their preferred order, then default topics which are from 1 to 5 are submitted as their top 5 preferred topics. If the student has changed their mind and wants to reorder the preferences after submitting them, he/she can log in and submit their preferences again to update them in the database. Here we are not showing their previous preferences when they login back to submit again.

Topic link is added for every topic so students when on clicking it opens a new tab in the browser which redirects them to the topic's webpage where students can find more about the topic before selecting it as one of their preferences.

Topics list

TOPIC ID	TOPIC TITLE	TOPIC LINK
1	abhi	testing.com
2	Testing1	testing1232.com
3	abhihi	asd.com
4	AI	AI.com
5	informatics	informatics.com
6	physics	physics.com
7	science	science.com

Submit Preferences



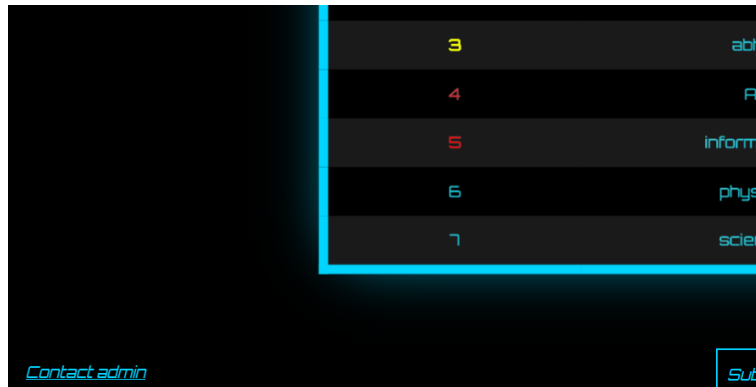
Five colors on the page which show students each of their preferences in the following order:

- Green as preference 1
- Light green as preference 2
- Yellow as preference 3
- Light red as preference 4
- Red as preference 5

If the student logs in before admin publishing the topics, then it shows them that there are no topics published and the submit button is disabled with its href attribute value as javascript: void(0).

IMAGE OF NO TOPICS NEED TO BE ADDED HERE

On every page, at the left bottom, we have an anchor tag that redirects them to send mail to the admin in case they are unable to login to the application or unable to submit their preferences, etc.



Admin Dashboard:

Admin dashboards have 4 buttons which redirect them to pages which include

- Add topics
- View topics
- View preferences
- View allocations

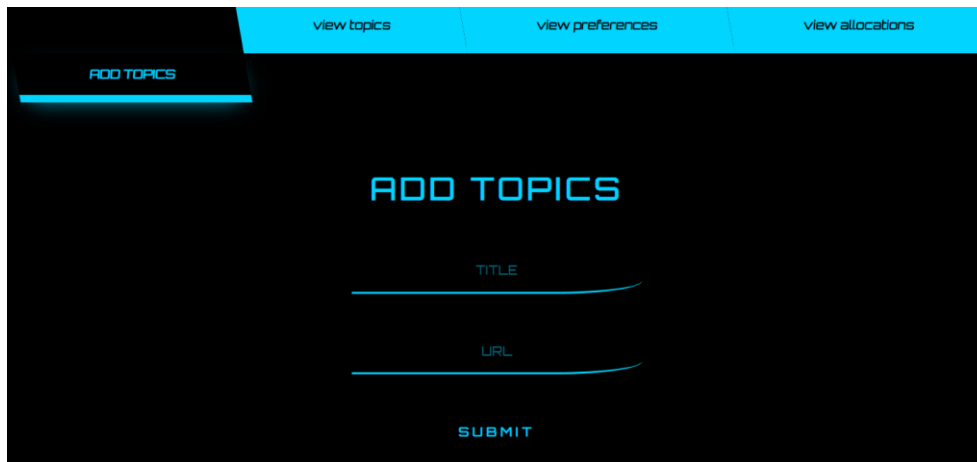


Add topics page:

This page is available for admin and supervisors to add a topic.

To add a topic, admin or supervisor needs to click add topic button on top of the page, which shows them a form with two input fields. One for the topic title and other for topic URL. After submitting the form by clicking add topic, the topic gets added into the database and it'll be updated on view

topics page so admin can later publish them once they are approved. An alert message showing 'topics added' will be shown after each successful submission of the topic.

The image shows a web application interface for adding topics. At the top, there is a navigation bar with three links: 'view topics', 'view preferences', and 'view allocations'. Below this, on the left, is a sidebar with a button labeled 'ADD TOPICS'. The main content area has a dark background and features the title 'ADD TOPICS' in large, bold, white letters. Below the title are two input fields: 'TITLE' and 'URL', each with a light blue border and a light blue underline. At the bottom of the form is a 'SUBMIT' button in white text on a dark background.

View topics page:

This page is available only for admin. Here admin can view the topics which are in the database and publish the topics to students.

We also have an option for deleting the topics and also hide them from students so only selected topics are shown to students. To delete a topic, admin can simply click the x button which is added beside every topic and to temporarily hide the topic from the student, there is a checkbox added for each topic which on clicked hides them from the students.

After deleting or hiding the topics, the admin needs to click the "publish topics button" to submit them into the database and to also show them for students.

Topics list

		TOPIC ID	TOPIC TITLE	TOPIC LINK
<input type="checkbox"/>	<input type="checkbox"/>	5de405183d2f163c240c18f3	abhi	link to topic
X	<input checked="" type="checkbox"/>	5de4059f3d2f163c240c18f6	Testing1	link to topic
X	<input checked="" type="checkbox"/>	5de411293d2f163c240c18f9	abhi1	link to topic
X	<input type="checkbox"/>	5e1c81f13b7df46944887b6	AI	link to topic
X	<input type="checkbox"/>	5e1c83483b7df46944887b7	informatics	link to topic
X	<input type="checkbox"/>	5e1c83713b7df46944887b8	physics	link to topic
X	<input type="checkbox"/>	5e1c83bc3b7df46944887b9	science	link to topic
X	<input type="checkbox"/>	5e1c98f53b7df46944887f6	topic 6	link to topic

[Publish Topics](#)

View preferences page:

After publishing the topics, all the students need to submit their top 5 preferred topics than on this page admin can view all the preferences of all the students. If everything is fine, then admin clicks the run algorithm button which is at the bottom of the page. This runs the algorithm and allocates students into a group of 5 students with their top preferred topic as the group topic.

[add topics](#)
[view topics](#)
[view allocations](#)

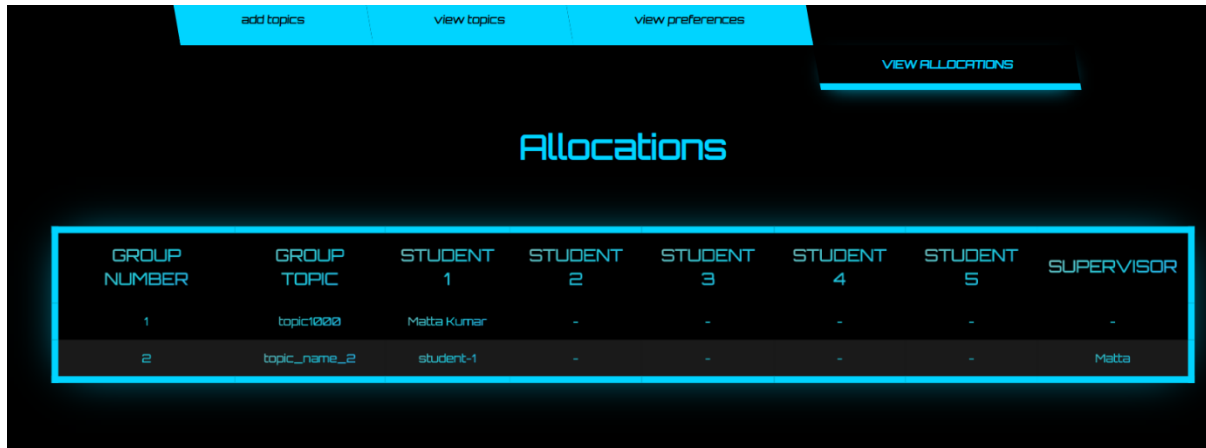
Students Preferences

STUDENT ID	PREFERENCE 1	PREFERENCE 2	PREFERENCE 3	PREFERENCE 4	PREFERENCE 5
student-1	topic_name_2	topic_name_4	topic_name_5	topic_name_7	topic_name_1

[Run Algorithm](#)

View allocations page:

Once the preferences are submitted by the students and implementing the algorithm, all the allocated groups can be seen by the admin and he can publish them so supervisors can see them and select a topic for which they want to supervise.



GROUP NUMBER	GROUP TOPIC	STUDENT 1	STUDENT 2	STUDENT 3	STUDENT 4	STUDENT 5	SUPERVISOR
1	topic1000	Matta Kumar	-	-	-	-	-
2	topic_name_2	student-1	-	-	-	-	Matta

Supervisor dashboard:

Here, the supervisor has 2 buttons which redirect them to pages which include

- Add topics page
- Submit supervising topic page (available only for supervisor)

Add topics page:

This page is the same for both admin and supervisor.

To add a topic, admin or supervisor needs to click add topic button on top of the page, which shows them a form with two input fields. One for the topic title and other for topic URL. After submitting the form by clicking add topic, the topic gets added into the database and it'll be updated on view topics page so admin can later publish them once they are approved. An alert message showing 'topics added' will be shown after each successful submission of the topic.



Allocations

ADD TOPICS

ADD TOPICS

TITLE

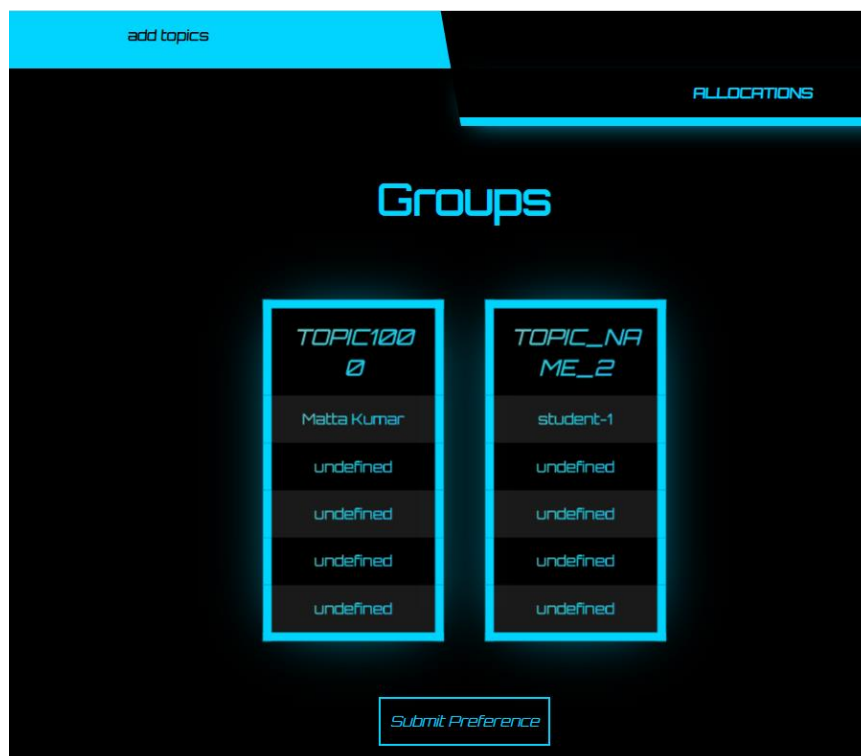
URL

SUBMIT

Submit supervising topic page:

This page is only available for the supervisor. By clicking the allocations button on the supervisors' dashboard, the supervisor can access this page.

Once the admin runs algorithm and groups are allocated, the supervisor can see those allocated groups on this page.



add topics

ALLOCATIONS

Groups

TOPIC100

0

Matta Kumar

undefined

undefined

undefined

undefined

TOPIC_NA

ME_2

student-1

undefined

undefined

undefined

undefined

Submit Preference

The supervisor can select one topic from the list of groups and submit it by clicking the submit preference button on the bottom of the page. After selecting the preferred group, the group color

changes from blue top green and a smiley is shown as a cursor showing the supervisor that the group he wants to supervise is selected.



APIs:

Endpoints:

API Name	URL Endpoint	Method	Input fields
API for creating a topic by admin from admin panel by adding topic URL and TITLE	api/v1/topics	POST	Title and URL
API for showing topics to students to submit their preference	api/v1/topics	GET	Title and URL
Update a topic once it is published to the Students or added into the Database	api/v1/topics/5de29b84b474a5290b5aa4c2	UPDATE	Title and URL

Delete a topic once it is published to the Students or added into the Database	api/v1/topics/5de29b84b474a5290b5aa4c2	DELETE	
Create student preference	api/v1/preferences	POST	Student ID, Student Name, Preference 1, Preference 2, Preference 3, Preference 4, Preference 5
Update student's preference after it's submission	api/v1/preferences/5dada2d4cd45eb99939aecc8	UPDATE	Student ID, Student Name, Preference 1, Preference 2, Preference 3, Preference 4, Preference 5
Get students preferences to run the algorithm and allocate them into groups	api/v1/preferences	GET	Student ID, Student Name, Preference 1, Preference 2, Preference 3, Preference 4, Preference 5
Creating a user(Student/ admin/ supervisor)	api/v1/users	POST	Email and password
Creating groups	api/v1/groups	POST	Name, Topic, Student 1, Student 2, Student 3, Student 4, Student 5
Get all allocated groups	api/v1/groups	GET	

Update the group with supervisor	api/v1/groups/5de3c9763270323561b9fb6f	UPDATE	Supervisor

Login module:

```
/**
 * TODO:
 * Implement User LOGOUT
 */
'use strict';

/**
 * Module dependencies.
 */
const User = require('../models/User');
const loginRouter = require('express').Router();

/**
 * User Login
 */
const login = async (req, res) => {
  try {
    const { email, password } = req.body;
    const userDetails = await User.findByCredentials(email, password);
    const token = await userDetails.generateAuthToken();
    res.send({ data: { user: userDetails.removeUnwantedFields(), token } });
    // res.send('Logged In');
  } catch (err) {
    res.status(400).send(JSON.stringify(err, ['stack'], 4));
  }
};

loginRouter.post('/login', login);
```

```
module.exports = loginRouter;
```

The login module recalls the user details for the already existing users from the database. Users will be required to type in their email and their password after which the try and catch block will be executed to validate the credentials. If both the email and the passwords match with the existing user details, then the user is allowed to log in successfully.

Student (create):

```
const UserSchema = new mongoose.Schema(  
  {  
    email: {  
      type: String,  
      required: true,  
      trim: true,  
      minlength: 4,  
      unique: true  
    },  
    password: {  
      type: String,  
      required: true,  
      minlength: 6  
    },  
    role: {  
      type: Number,  
      minlength: 1,  
      required: true  
    }  
  },  
  { timestamps: true },  
  { usePushEach: true }  
);
```

The above snippet is the code that enables students to create their account and login after creating an account. Then all the data types that will be required are initialized and their conditions such as the minimum length of the data and whether it is required or it can be skipped are determined. For this case, the email is a string data type and only a min of four characters are allowed. The password, on the other hand, must be unique for every user, a string data type, and only a minimum of six characters are accepted. The system uses timestamps to be able to know at what time the student logged in the system.

```
UserSchema.methods = {  
  // Generating jwt after creating a user and after login  
  async generateAuthToken() {  
    try {  
      const token = jwt.sign(  
        { _id: this._id.toHexString(), email: this.email, role: this.role },  
        'secret_guna',  
        { expiresIn: '7d' }  
      );  
      return token;  
    } catch (err) {  
      throw err;  
    }  
  },  
  removeUnwantedFields() {  
    let { password, __v, ...rest } = this.toObject();  
    return rest;  
  }  
};
```

For the above code, a jwt is generated after creating an account and login in. After setting up the account with an email and a password, the above code uses a try, catch and throw method of error handling. As indicated earlier, the throw statement allows the program to notify the student using customized error messages. In addition, the try statement tests the code for errors while the catch statement handles it.

An if-else statement has been used in the above code to ensure that the code executed as long as the password and the email match with the data that is stored in the database. If the password and

the email are collected, the student logs in the system successfully. The throw statement is constantly used in this particular segment of the code to ensure that the system remains user-friendly, thus improving usability¹. When a student or any other user for that matter is notified of what the system is doing along the way, then they feel more comfortable while using the system.

API for viewing approved topics:

The admin is responsible for creating new topics, sending them to students for them to pick their preferred topics and then the admin approves those topics using computer algorithms. After the approval, the students need to view their approved topics.

This API will use a try and catch block as it is meant to give the approved topic to the student. The try statement will examine the code for any errors while the catch statement will handle any errors found.

If the admin had approved a topic for a specific student, the block would be true and thus, the student will be able to view their approved topics. Otherwise, the student will get an error message.

¹ Renee Garrett et al., "A Literature Review: Website Design and User Engagement," *Online Journal of Communication and Media Technologies* 6, no. 3 (July 2016): 1–14.

API for selecting preferences

```
const createPreference = async (req, res) => {
  try {
    //TODO: Bulk Pref insertions
    let pArray = data.map(pref => {
      const temp = new Preference(pref);
      return temp.save();
    });
    const r = await Promise.all(pArray);
    res.send(r);
    // let preferences = new Preference(req.body);
    // await preferences.save();
    // res.send(preferences);
  } catch (err) {
    logger.error(err.stack);
    res.status(400).send(err);
  }
};
```

The above snippet has the same code structure as the admin API for creating and viewing users. The code snippet uses module dependencies that are imported from the existing libraries. The second part of the code is where the student creates their preferences after which, a try and catch error handling method is used so that they don't choose a preference that they had chosen before. The catch statement is used to handle any errors that might be found when testing the code using the try block. After the student has created their preferences, they are able to query the system and get their preference details. At this point, the system also employs another try and catch block as a way of handling the errors that might arise.

```
const getPreferences = async (req, res) => {
  try {
    const {studentId} = req.query;
    const preferences = await Preference.find({studentId});
    res.send(preferences);
  } catch (err) {
    logger.error(err.stack);
    res.status(404).json(err);
  }
};
```

```
let preferences = await Preference.findOneAndUpdate(
  {studentId: id},
  req.body,
  {
    new: true
  }
);
```

Following the creation and the viewing preferences, the students are now able to update their preference details while using the same API for selecting preferences. A try catch-and-throw block is used to help the student in dealing with any system error that might arise. In this case, the throw statement is used to notify the student when they make the wrong choice. For instance, when they input an invalid preferenceID. An if a statement is also used to execute the code as long as the student has inputted wrong details, the system will always give them the same message of invalid preference else the system will update the preference details.

After successfully updating the preference details, the API for selecting preferences also allows the students to delete these preferences in case they are not happy with their choices.

6.2.2.3 API for showing allocated topics and groups

```
const getGroups = async (req, res) => {  
  try {  
    // const {studentId} = req.query;  
    // console.log(studentId);  
    const groups = await Group.find({}).lean();  
    res.send(groups);  
  } catch (err) {  
    logger.error(err.stack);  
    res.status(404).json(err);  
  }  
};
```

The administrator is tasked with creating the topics, sending them to the students via an email notification, viewing the student's preferred selections, and then allocating these preferences to students by use of an algorithm.

Once a student has been allocated their preferred topic, then they can view them through the API for showing allocations. The API initializes the data types to be used, for instance, the data type to be used both strings and Boolean type. If the student has been allocated a topic, the system will return true and show them their allocated topic; else, the student will not be shown any topic. The timestamps have been used so that they can be able to monitor when the student saw the topic to avoid having students who might say that they had never seen their allocations.

Admin

The admin is responsible for overseeing the whole system as well as maintenance². For this particular web application, the admin has roles such as adding topics, publishing approved users, viewing student's topic preferences, and viewing and publishing topic allocations. The following sections explain the various APIs for the admin.

² Charles C. Snow, Øystein Devik Fjeldstad, and Arthur M. Langer, "Designing the Digital Organization," *Journal of Organization Design* 6, no. 1 (June 2017): 7, doi:10.1186/s41469-017-0017-y.

² Charles C. Snow, Øystein Devik Fjeldstad, and Arthur M. Langer, “Designing the Digital Organization,” *Journal of Organization Design* 6, no. 1 (June 2017): 7, doi:10.1186/s41469-017-0017-y.

API for adding topics

In the first segment of the above code snippet, the topic is initialized and all the required libraries imported. JavaScript models are also used in reference to the libraries. The second part of the code is where the admin creates a new topic. At this point, the system uses a try and catch block to validate and make sure that only topics that are new can be added to the system. The last segment in the above snippet is the code that the admin uses when checking for topic details to ensure they have been added to the system successfully.

```
const createTopic = async (req, res) => {
  try {
    let topic = new Topic(req.body);
    await topic.save();
    res.send(topic);

  } catch (err) {
    logger.error(err.stack);
    res.status(400).send(err);
  }
};

const getTopics = async (req, res) => {
  try {
    let options = {active: true};
    if (
      req.headers['referrer']
        ? req.headers['referrer'].split('/').pop() === 'admin1.html'
        : true
    ) {
      options = {};
    }
    const topics = await Topic.find(options);
```

```

    res.send(topics);
  } catch (err) {
    logger.error(err.stack);
    res.status(404).json(err);
  }
};

```

This part of the API has two segments where the admin is able to both update and delete topics from the system. While updating the topics, the admin a try, throw, and catch block have been used the try block tests the code segment for any errors. The throw block allows the admin to receive customized error messages from the system. The same procedure has been used for the deleted segment of the code where a throw statement has been added to notify the admin when the select an invalid topicID. Additionally, the try and catch statement tests the code and handles it respectively. An if the statement has been used for both segments of the code and allows either segment to be executed as long as certain conditions hold true.

API for viewing preferences

```

*/
const getPreferences = async (req, res) => {
  try {
    const {studentId} = req.query;
    const preferences = await Preference.find({studentId});
    res.send(preferences);
  } catch (err) {
    logger.error(err.stack);
    res.status(404).json(err);
  }
};

```

The admin is responsible for creating new topics, sending them to students for them to pick their preferred topics, and then the admin approves those topics using a computer algorithm. After the approval, the admin needs to view these student's preferred topics.

This API will use a try and catch block as it is meant to give the student preferred topics to the admin. The try statement will examine the code for any errors while the catch statement will handle any errors found.

If the admin has queried the system for preferences for a specific student, the block will be true, and thus the admin will be able to view their preferred topics. Otherwise, the admin will get an error message.

API for publishing allocations

This API helps the admin to publish the topics that the students have been allocated. The supervisor will then take over from when the publishing of allocation is done. The try statement is necessary to test this code segment in the API for publishing approved users for any possible errors. The catch block ensures that any of these errors that are identified when testing are well handled³. And finally, the throw statement ensures that the admin is able to receive personalized messages telling them of what the system is doing at any particular instance. Timestamps have been used to allow the admin to know the time when they published the approved users.

³ Elishai Ezra Tsur, "Rapid Development of Entity-Based Data Models for Bioinformatics with Persistence Object-Oriented Design and Structured Interfaces," *BioData Mining* 10, no. 1 (March 2017): 11, doi:10.1186/s13040-017-0130-z.

Supervisor

The supervisor checks groups that the students have been allocated and, at the same time, needs to select one group from the list and submit it, which confirms that he/she is ready to supervise that particular group

Algorithm for allocation groups:

³ Elishai Ezra Tsur, "Rapid Development of Entity-Based Data Models for Bioinformatics with Persistence Object-Oriented Design and Structured Interfaces," *BioData Mining* 10, no. 1 (March 2017): 11, doi:10.1186/s13040-017-0130-z.

```
const ds = data.reduce((acc, pref) => {  
  const {name, ...rest} = pref;  
  acc[name] = rest;  
  return acc;  
}, {});
```

Sample JSON data to run an algorithm:

```
const data = [  
  {  
    name: 'student-1',  
    p1: 'topic_name_2',  
    p2: 'topic_name_4',  
    p3: 'topic_name_5',  
    p4: 'topic_name_7',  
    p5: 'topic_name_1'  
  },  
  {  
    name: 'student-2',  
    p1: 'topic_name_5',  
    p2: 'topic_name_7',  
    p3: 'topic_name_1',  
    p4: 'topic_name_3',  
    p5: 'topic_name_4'  
  }  
]
```

JSON data which comes from the database, give us names and their five preferences of all the students.

First, to run the implemented algorithm, we need to convert the data which we get from the database to the required format. Data we get from the DB is in JSON format. We need a slight

modification to it, so we can use this in our algorithm. In the above code, we created a new function called ds which returns the array 'acc' of student's preferences.

```
const groups = {};

for (let i = 1; i <= 5; i++) {
    const priority = `p${i}`;
    for (let key of Object.keys(ds)) {
        const x = ds[key][priority];
        if (!groups[x]) {
            groups[x] = [];
            groups[x].push(key);
            delete ds[key];
        } else if (groups[x] && groups[x].length < 5) {
            groups[x].push(key);
            delete ds[key];
        } else {
        }
    }
}
```

```
const groups = {};
```

We created an empty array names groups so we can add the allocated groups into it.

```
for (let i = 1; i <= 5; i++) {
```

```
    const priority = `p${i}`;
```

We need to allocate no more than five students in each group, so we have created a main outer loop that runs five times. We have declared a variable named priority and for every time the loop runs, it acts as the preference of each student from first preference to fifth preference.

```
        groups[x].push(key);
```

```
        delete ds[key];
```

```
for (let key of Object.keys(ds))
```

Now we have an inner loop which runs for every student. We get each student's priority, key (which is unique for every student) and name from ds variable which we created initially.

```
const x = ds[key][priority];
```

We created a new variable X for every student with all their preferences. (Example: ds[student-1][Preference-1], ds[student-1][Preference-2], ds[student-1][Preference-3], ds[student-1][Preference-4], ds[student-1][Preference-5]).

```
if (!groups[x])
```

```
groups[x].push(key);
```

```
delete ds[key];
```

We have an if condition to check if we already have a group with x, if not, we will push the student with a selected key to that array and delete them from ds array.

```
} else if (groups[x] && groups[x].length < 5) {
```

```
groups[x].push(key);
```

```
delete ds[key];
```

```
else{
```

```
}
```

we have an else if block where we check if there is already a group[x] and it has less than five students in them and deletes the student from the array ds. In the end, we have an empty else block which allocated student to any other group if all the groups if allocated with all their preferred topic.

Following the web development process:

"Any software development project, a methodology should be followed to ensure project consistency and completeness.

The Web development life cycle includes the following phases: planning, analysis, design and development, testing, and implementation and maintenance.”

[http://cs.tsu.edu/ghemri/CS117/ClassNotes/Web%20Development%20Life%20Cycle_small.htm]

Planning:

We need to identify the goals first so that we can make a clear plan for developing the application.

“The question to ask is: What is the purpose of this Web site?

In addition to understanding the Web site purpose, you should also ask: Who will use the Website? or knowing the target audience in terms of age, gender, computer literacy, etc.

Understanding the computing environment will allow the designer to know what type of Technologies to use.”

[http://cs.tsu.edu/ghemri/CS117/ClassNotes/Web%20Development%20Life%20Cycle_small.htm].

Then I created a timetable for the project. We have 3 months deadline for submission, considering university as a client and as an employee of a company all I need to do is deliver quality final product within the agreed deadline (In our case it is the deadline for submission of the project). In these 3 months, each month is allocated to one of the main tasks in our development process. The first month is for designing and developing front end views. The second month is for developing an entire backend part of the application which includes design and creating a database, developing APIs and integrate it with the front end. 3rd month is for implementing of the algorithm, testing and report writing

Web site layouts:

Websites are designed using any of several different types of layouts, including linear, hierarchical, and Webbed [].

Image of wireframe/XD.

This project doesn't have any complicated views, so I decided to implement a linear layout. A linear Web site layout connects Web pages in a straight line. [].

After knowing about the project, who will use it and what is the purpose of the project, I've finally decided to show data in form of tables so that students can clearly view topics rather than getting confused by providing the list of topic links.

For taking preferences from students, initially, the plan was to add a drop-down menu with 4 numbers from 1 to 5 in it for every topic. Students must select number 1 as top preference and 5 as least preference but this seems to more confusing for a student to check the order of preference. The final design has a list of topics where a student can manually drag them with the cursor and place them in their preferred order. Along with those, colors from green to red are added for the first 5 topics so the user can easily notice the topic's order before submitting them.

Build and running of application:

Build and running of application:

We need to make sure to install python to run the project and NPM must be set to the system path. In the command prompt (running as administrator and having fully read and write permissions for the folder), install all the dependencies and plugins by typing “npm install”.

We need to install robo3t and MongoDB and import our DB into it.

In the backend folder, install node modules by typing “npm install”. This installs all the node modules which are needed. After installing it, type “npm start” which runs the server and connects to Database. This makes the APIs work on the browser and shows all the data from the database on the browser.

Make sure that no other page or folder is accessing port 3000, If you are running the site with a different web server change this to the URL of the site e.g. localhost:1234 in webpack.mix.js file on assets folder.

Then type “npm run watch” command to watch the changes in the files and initialize browser-sync, this runs the application on the browser. Now complete web application must be working.

All styles must be written in the sass folder. All styles written here are compiled into plain CSS and saved in style.css in the site folder.

All JS code must go into the assets/js folder which compiles and saved in script.js file in the site folder.

All Markup is written in HTML files in the site folder.

Further updates:

- In the designed web application, students can only submit their top five preferred topics, but we can still make it in more useful way by taking students preferred group members and supervisor, for this, we need to first get all the students details from the DB and show as a list for the student so they can select their preferred group members. Right now, we don't have any student details in our database as we are only creating a new user every time a student's login with any username and password
- Make tables responsive. We can see all the topics, student's preferences, groups all are in the form of tables which takes more pixels than of any other data while rendering on the web page. We need to make a new design and implement it so that the application can be easily used and accessed on mobiles and tablets too.
 - Responsive Website Improves Your SEO Efforts.
 - Adapts Easily To Any Screen Size
 - Provides a Seamless User Experience
 - Increased Traffic From Mobile Users

[\[https://www.forbes.com/sites/brianrashid/2017/06/13/5-essential-reasons-and-benefits-why-you-should-be-using-a-responsive-website-design-now/#1694a8ae17c9\]](https://www.forbes.com/sites/brianrashid/2017/06/13/5-essential-reasons-and-benefits-why-you-should-be-using-a-responsive-website-design-now/#1694a8ae17c9)
- Make total web application according to A/AA/AAA web standards. "The Web must be accessible to provide equal access and equal opportunity to people with diverse abilities. Indeed, the UN Convention on the Rights of Persons with Disabilities recognizes access to information and communications technologies, including the Web, as a basic human right." [<https://www.w3.org/standards/webdesign/accessibility>], Accessibility focuses on how a disabled person accesses or benefits from a site, system or application. Accessibility is an important part of designing your site and should be considered throughout the development process. [<https://www.usability.gov/what-and-why/accessibility.html>]
- By merging the web application with the UoL application, students can easily access the site and can use them on their mobile.
- Can send emails for students once the approved topics are posted, sending reminder emails for students who didn't submit their preferences, sending emails for students with their group name, supervisor and allocated group members.
- Ask students for their interested topic areas and use appropriate tags for each topic so that when a student logs into the application, he can see topics related to his interests first and other topics next. "Tagging work items help you quickly filter the product backlog or a work

item query by categories that you define. A tag corresponds to a one or two keyword phrase that you define and that supports your needs to filter a backlog or query, or define a query.”

[<https://docs.microsoft.com/en-us/azure/devops/boards/queries/add-tags-to-work-items>]

Code snippets:

Code for taking students preferences:

```
function dragit(event){  
    shadow=event.target;  
}  
  
function dragover(e){  
    let children=Array.from(e.target.parentNode.parentNode.children);  
    if(children.indexOf(e.target.parentNode)>children.indexOf(shadow))  
        e.target.parentNode.after(shadow);  
    else e.target.parentNode.before(shadow);  
}
```

Here we are calling the function dragit and dragover and storing the event and setting the target location where the user wants to drop it. We are showing topics in the form of a table in which every <tr> tag has a topic in it. So, we are adding these two functions on every <tr> so the student can drag and drop them in their preferred order.

This code is only for taking student’s input in an easy way rather than student typing his preferred number (1,2...5) in the input field for every topic or selecting the value of preference from a dropdown is more confusing and nor user-friendly. So, I have used this code from this source so can allocate more of my time in developing web application and algorithm.

[<https://codepen.io/nabildroid/pen/ZPwYvp>].

Code for showing JSON data in a table with GET method:

```
var tableHTML = "";
tableHTML +=
  '<tr><th>Topic ID</th><th>Topic Title</th><th>Topic link</th><th></th></tr>';
for (var i = 0; i < response.length; i++) {
  //alert(response[i].title + response[i].link);
  tableHTML +=
    '<tr draggable="true"ondragstart="dragit(event)"ondragover="dragover(event)" id=' +
    response[i].title +
    '><td>' +
    (i + 1) +
    '</td><td>' +
    response[i].title +
    '</td><td>' +
    response[i].link +
    '</td><td class="preference"></td></tr>';
}
```

To clearly show the values in the form of a table, in this code we get topics as a response from the database and runs length of the response times and adds every topic in a new row and also adds our drag function for every topic. This snippet of code is used from this source

[<https://codepen.io/codingsite/pen/sdSDeffs>].

Plugins and extras:

Particle JS:

The login page on the web application uses particle JS, which is a light weighted JavaScript library for creating particles. This is an NPM package and can be installed with below command

```
npm install particles.js
```

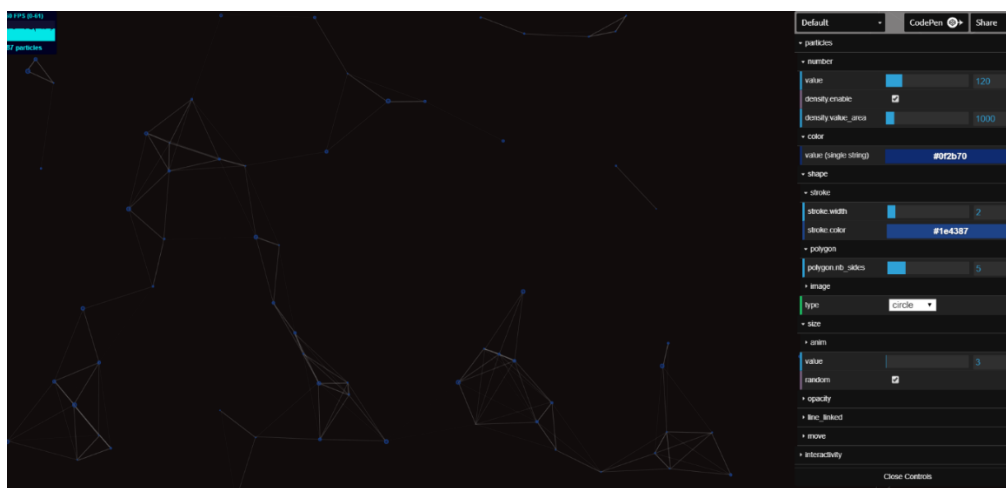
This library is only just for a good UI and it can be customized based on our requirements like a number of particles, type of interaction, color, etc.

```

"particles": {
  "number": {
    "value": 150,
    "density": {
      "enable": true,
      "value_area": 700
    }
  },
  "color": {
    "value": "#00d4ff"
  },
  "shape": {
    "type": "image",
    "image": {
      src: "./img/uol-logo-png.png",
      width: 60,
      height: 50
    }
  },
},

```

I have used university of Leicester logo here instead of predefined particles for better UI.



Tailwind CSS:

Tailwind CSS -A utility-first CSS framework for rapidly building custom designs.

“They come with all sorts of predesigned components like buttons, cards, and alerts that might help you move quickly at first, but cause more pain than they cure when it comes time to make your site stand out with a custom design.

Tailwind is different.

Instead of opinionated predesigned components, Tailwind provides low-level utility classes that let you build completely custom designs without ever leaving your HTML.”

[<https://tailwindcss.com/#what-is-tailwind>].

By default tailwind imports all its predefined classes into final stylesheet which increases the file size and also makes page loading very slow

“Why do we use PurgeCSS? Tailwind includes a lot of bloat by default so we need to remove anything that isn't being used.

CSS file size before purge -> 414KB

After purge for this default template -> 3KB” [<https://github.com/SteinIAS/Noah>]

. To overcome this, we have installed a plugin called purgeCSS. This will check all the classes used in out markup and add only them to the final output and makes file size to 1/10th of the initial file size. If in case we have some classes defined in CSS and not included in the markup which was supposed to add through Javascript, in such case we need to add ‘pa-’ before the class name. Example: pa-active, which means purge allows and it makes sure to include this class in final outputted CSS.

Noah web development template:

This is Stein IAS' Web Development Template, it has all the plugins added to it and ready to use a web development template for any sized application. [<https://github.com/SteinIAS/Noah>]

SASS:

On this project, we are using sass for styling the web pages. SASS stands for Syntactically Awesome Style Sheets. This is a stylesheet language that will be converted to CSS.” It allows you to use variables, nested rules, mixins, functions, and more, all with a fully CSS-compatible syntax. Sass helps keep large stylesheets well-organized and makes it easy to share design within and across projects.” [\[https://sass-lang.com/documentation\]](https://sass-lang.com/documentation).

This can be added through NPM with this command: **Install node-sass**.

SCSS SYNTAX:

```
$font-stack: Helvetica, sans-serif;
```

```
$primary-color: #333;
```

```
body {
```

```
    font: 100% $font-stack;
```

```
    color: $primary-color;
```

```
}
```

CSS OUTPUT:

```
body {
```

```
    font: 100% Helvetica, sans-serif;
```

```
    color: #333;
```

```
}
```

BEM methodology:

This method is used while naming classes in our style sheet. BEM stands for Block, Element, Modifier. When we are creating a module on a web page, we need to divide them into blocks and use a single class name for each block and divide it into elements. Now the class name for the element will be BLOCKNAME__ELEMENT-NAME, by which we can know that the element belongs to that particular block. In case we need some more styling to the same class or using the same class with some extra styles, we need to use a modifier. The syntax for using modifier is BLOCKNAME__ELEMENT-NAME—MODIFIER.

A block can have only block name and can have any number of elements and modifiers, but nested naming is not allowed in this. Example: 'BLOCKNAME__ELEMENT1__ELEMENT2—MODIFIER'.

“BEM Avoids inheritance and provides some sort of scope by using unique CSS classes per element (like .my-component__list-item).

Reduces style conflicts by keeping CSS specificity to a minimum level.”

[<https://blog.decaf.de/2015/06/24/why-bem-in-a-nutshell/>].

Testing:

There are many types of testing but some of the main steps in testing phase include:

- Functionality testing
- Usability testing
- Interface testing
- Compatibility testing
- Performance testing
- Security testing

Functionality testing:

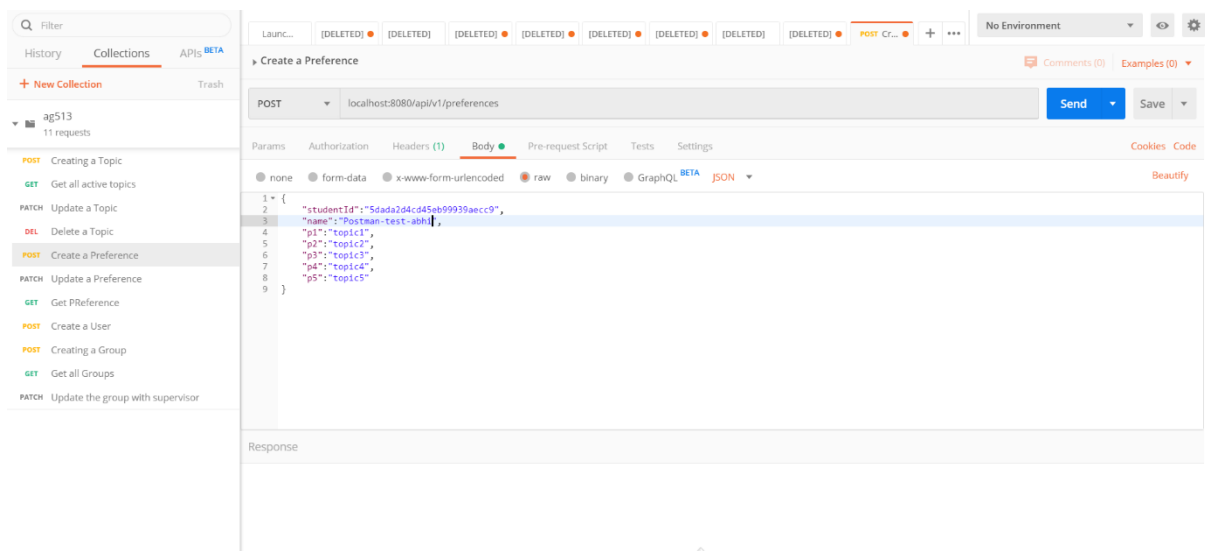
In this phase, we need to check the functioning of every module, database connection, etc. In our project, we tested this phase by checking whether or not all the data is getting submitted into the database correctly and all the data is showing exactly how it should be shown from the database.

This process tests the working on Front-end views, DB and API.

Testing with Postman tool:

By submitting different inputs manually to the database and checking them on the postman, "A collaboration platform for API development" [<https://www.getpostman.com/>].

In this, we need to open the postman tool and add the endpoint of our API and run it. We can view all the responses of users on our page if everything in the code works fine.



On the left side, we can see all our APIs and their methods.

In the body section, we can see student ID, name and five preferences, This is the body of API for submitting the student's preferences into the database.

Usability testing:

“Usability testing refers to evaluating a product or service by testing it with representative users. Typically, during a test, participants will try to complete typical tasks while observers watch, listen and takes notes. The goal is to identify any usability problems, collect qualitative and quantitative data and determine the participant's satisfaction with the product.”

[<https://www.usability.gov/how-to-and-tools/methods/usability-testing.html>]

In this phase, I have asked some of my fellow students to test the site while running it on my desktop as it is not hosted anywhere. I've taken their feedback and most of them where optional requirements of the project.

Testing algorithm and its efficiency:

Developing an algorithm is not a hard task, but making if making it work more efficiently is the main problem. In our case efficiency means the allocation of the top preferred topic to all the students. So, to test the efficiency of the algorithm I have used many test cases with different student's preferences. The algorithm in this project works with data that it receives from the database but we don't have all student's preferences in our database and its waste of time adding preferences manually from the web application into our database. So, I have created test data in JSON format and sent it to the algorithm to test the data. The first version of the algorithm allocated students' preferences in such a way that its efficiency was nearly 75 -80%, after modifying the algorithm many times and ended up with this algorithm which has nearly 95% efficiency when tested with sample data. (efficiency is calculated by Testing data containing 35 students with 6 different data. In all cases 30-32 students are allocated with their first preferred topic, 2-4 students are allocated with their second preferred topic and 1-2 students are allocated with the fourth or fifth preferred topic).