Implementation Document

for

mIIT-Kute

Version 1.0

prepared by

Group #: 8 **Group Name:Developing Decuple**

Name	Roll Number	Email id
Vedasree Tatimanu	201049	vedasreetatimanu@gmail.com
Gutta Raghavendra Chowdary	200396	grc1933@gmail.com
Akshat Garg	200084	akkigarg989@gmail.com
Janhvi Rochwani	200467	rochwani.janhvi@gmail.com
Deepak Sangle	200860	deepaksangleok@gmail.com
Yeginati Vinay Teja	201161	vinayteja686@gmail.com
Bugada Ashritha	200285	bugadaashritha2002@gmail.com
Jayaprakash Napa	200622	jayaprakashnapa2003@gmail.com
Kembasaram Nitin	200505	kembasaramnitin@gmail.com
Sai Charan Modem	200586	saicharansai2570@gmail.com

Course: CS253
Mentor TA: M. Sri Madhan
Date: 20-03-2022

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Version 1	Vedasree Tatimanu Gutta Raghavendra Chowdary Akshat Garg Janhvi Rochwani Deepak Sangle Yeginati Vinay Teja Bugada Ashritha Jayaprakash Napa Kembasaram Nitin Sai Charan Modem	–.	20/03/22

1 Implementation Details

We have used the following web development techniques.

Programming Languages:- HTML5, CSS3, JavaScript
Frameworks:- express.js, Mongoose
Runtime environments:- Node.js
Node.js middleware:- bcrypt.js, dotenv, ejs, express, -express-flash, express-session, jsonwebtoken ,method-override, mongoose, nodemailer, nodemon, passport, passport-local, toast-notification-alert
Database Systems:- MongoDB

For the frontend of our web application we have used HTML, CSS, JavaScript

For the back end of our web application we have used JavaScript Frameworks like Node.js, express.js, mongoDB..

For the database management we have used mongoDB

EJS is a simple templating language which is used to generate HTML markup with plain JavaScript. It also helps to embed JavaScript to HTML pages. It was convenient for us to include EJS in our code. However, Express.js can also be used for the same purpose.

We used ***bcryptjs*** to encrypt the passwords of our database as hashed passwords.

We employed ***DotEnv*** to load the environment variables from the .env file.

Different frameworks from ***Express*** such as ***Express Session*** (to maintain the details of a running user-session) and ***Express Flash*** (to display error messages and other prompts) have also been used.

JWT tokens are widely used for API authentication and are easy to implement. As an added plus, session information is not stored server-side which saves server memory consumption.

In object-oriented programming, ***method-override*** allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its superclasses or parent classes.

We chose a NoSQL database management system (***MongoDB***) because they are easily programmable, have well formed documentation, and availability of the option to update fields if we need to without worrying about a set “schema” or structure.

Nodemailer is the most popular and intuitive module for node.js applications for email sending.

Nodemon is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected.

2 Codebase

Provide the link to your github repository.
Mention briefly how to navigate the codebase.

Github Link: https://github.com/Deepak-Sangle/mIIT-Kute
Heruoku App Link: https://miit-kute.herokuapp.com

Codebase consists of the nodejs server app as backend along with basic ejs view engine as frontend.

1. The server consists of different routes mainly consisting of '/auth', '/event' and '/interest' route requests.
2. It also contains a public repository which contains basic images, css and js files for frontend.
3. The model folder consists of all the models required for the backend namely 'user', 'interest' and 'event' models.
4. The views folder consists of different ejs views which renders different views corresponding to the logged in user.
5. The middleware folder consists of all the middleware functions required for security purposes, restricting users from accessing different data and so on.

Firstly after running the app using a local server, the app.js script file runs which calls different routes, middleware function, ejs view, etc. In each of the routes different get, post, put, delete requests are called. Following are the route requests that are being called:

GET	'signin', 'signup', '/', 'verifying', '/:confirm' , 'allevents','find-events', 'myevents', 'createinterest', 'community' , 'myprofile'
POST	'signup', 'signin', 'createevents', 'liked/:' , 'edit-profile' ,
PUT	'join-events' , 'edit-profile'
DELETE	'signout'

After running the different route requests, the app looks into different ejs files which are present in the 'view' folder. It also renders the css static files which are present in the 'public/css/' folder.

After clicking on any event or button it again runs different route requests and the database gets updated accordingly.

Thus our complete app gets rendered.

3 Completeness

Provide the details of the part of the SRS that have been completed in the implementation.

● It allows the user to register/login using iitk credentials.
● Allows users to retrieve forgotten passwords using iitk mail id.
● Information about interested events is collected at the time of registration to improve the user experience.
● After logging in, users are provided with a list of ongoing events.
● Users can search for a specific event from the “find events” tab.
● Events can be filtered based on time.
● Once the user joins a particular event, it will appear in the “my events” section.
● Users can even request for the inclusion of an interest that is not already present, in the community section.
● Users can “like” a particular interest request by another person to express his/her interest about the same.

Provide the future development plan by listing down the features that will be added in the (may be hypothetical) future versions.

● Events can be divided into various categories , for e.g. sports, study groups, and filtered based on category.
● A chat server can be added to facilitate communication between interested participants.
● Existing services like the booking portal of the Games and Sports Council, IITK may be combined with our software to create a one-stop platform for sports.
● Various clubs and societies may join the software as super users to organise meetings/events at optimal times for their members.
● If a feedback/suggestion is sufficiently popular, the community page can be modified to notify super users regarding the same.
● Super users may be given access to remove participants in case they violate community guidelines.
● The user may be given access in order to add/edit their interests via their profile section.

Appendix A - Group Log

<*Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist in determining the effort put forth to implement your software*>

Date and time	**Work Done**
17 Feb 6:00 pm	Sharing ideas about the website creation and discussing the prerequisites for the website creation.
21 Feb 3:30 pm	Discussing about the languages to use for the website and checking the flexibility of the specific languages for our website.
25 Feb 4:30 pm	Division of work among the group mates according to the prerequisites required and the idea of the teammates in the corresponding work.
1 Mar 12:00 pm	Confirming about the amount of work done and discussing the convenience of the teammates in doing their respective jobs.
7 Mar 3:00 pm	Discussion about the problems faced in their corresponding work due to lack in some of the parts of their work.
12 Mar 10:30 am	Resolving the problems and again division of remaining work left with respect to their ideas in those parts that are yet to be done.
18 Mar 3:30 pm	Integration of code written by different people and checking the errors in different parts.
19 Mar 11:30 am	Discussion about the Implementation document and resolving the errors that were yet to be resolved.
20 Mar 3:00 pm	Finalisation & Submission of the code and the implementation document.