Test 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: **Mr. Sri Madhan**

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Vedasree Tatimanu Gutta Raghavendra Chowdary Akshat Garg Janhvi Rochwani Deepak Sangle Yeginati Vinay Teja Bugada Ashritha Jayaprakash Napa Kembasaram Nitin Sai Charan Modem	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	04/04/22

1 Introduction

Mention the test strategy

The main test strategy used in the testing process was manual testing. The major reason for this was that manual testing to a great extent removes nearly all the bugs and errors present in any web apps. Basically we tested different API's that we created in parallel with the development and then after integration, we tried our web app on different input cases. In this way, most of the bugs were removed from it.

When was the testing conducted?

The testing part of the code was done in parallel with the development. Basically, whenever a certain API or function was developed, those were tested at the same time using unit testing. Also whenever different units were integrated, the different systems were also manually tested along with it parallelly.

Who were the testers?

The testers were the developers themselves. The major reason for this was that since the developers know their code, they all know for which kind of input cases the function may lead to errors and thus it saves the testers time exponentially.

What coverage criteria were used?

The coverage criteria that were primarily used were that all the different test cases were taken manually and the testing was implemented until all the different inputs are exhausted. In this fashion, most of the bugs were eliminated.

Have you used any tool for testing?

Unfortunately, no. We didn't used any tool for testing since manual testing proved to be quite efficient as it discovered most of the bugs and errors present in our web app.

2 Unit Testing

1. Sign in API

Unit Details: *The SignIn API is used to render the sign in page of the app. It takes the input as the username and password of the already registered user.*

Test Owner: *Deepak*

Test Date: *01/04/2022 - 01/04/2022*

Test Results: The user can successfully sign in if the user has given proper credentials details. The system sends an error message if the details don't match or the user is not registered.

Structural Coverage: Given valid username and wrong password, invalid username and wrong password and also left any of the field empty. In all these cases the system throws an error message.

Additional Comments: *None*

2. Sign up API

Unit Details: *The Signup API is used to render the register page of the app. It takes the input as the username, IITK email and password to register user.*

Test Owner: *Vedasree*

Test Date: *01/04/2022 - 01/04/2022*

Test Results: The system sends confirmation email and user can successfully sign up given proper credential details.The system sends an error message if the mail-id is not IITK mail id or if password is not strong.

Structural Coverage: Given invalid IITK mail id system throws error message 'invalid email-id'. Given valid IITK mail id and weak password system throws an error message 'password must contain alteast 8 characters'.

Additional Comments: *None*

3. Homepage API

Unit Details: *The 'Homepage' API is used to fetch the top 10 ongoing events currently present in the database. It fetches the data from the databases and renders it in the frontend homepage.*

Test Owner: *Vedasree*

Test Date: *01/04/2022 - 01/04/2022*

Test Results: It fetched top 10 events correctly to the frontend with its correct data if there are more than 10 events on that day in the database, If there are less than 10 events all the events were fetched, this bug was removed later.

Structural Coverage: Initially if there were less than 10 events in that day all events were fetched but later this bug was fixed by fetching the events on the same day and not whole database, bug was removed successfully.

Additional Comments: *None*

4. 'All Events' API

Unit Details: *The ‘all-events’ API is used to fetch all the events currently present in the mIIT-Kute app. It fetches all the data from the databases and renders it in the frontend homepage.*

Test Owner: *Deepak*

Test Date: *02/04/2022 - 02/04/2022*

Test Results: All the events were correctly fetched to the frontend with its correct data. For eg. Events where some or many fields were empty were fetched and it fetches the data incorrectly. This bug was removed after testing various times

Structural Coverage: Initially different possible events were tried to render. After various attempts, all the bugs were removed.

Additional Comments: *None*

5. Create Event API request

Unit Details: *The ‘create-event’ API POST request was created exclusive to admin usage where only super user can create different event with respect to different interest of the users.*

Test Owner: *Deepak*

Test Date: *03/04/2022 - 03/04/2022*

Test Results: Different Events were created using different combination of the data and they all satisfactorily stored all these events in the database.

Structural Coverage: Different possible events were created and were tried to render. Such events include those types which have one or many incomplete or invalid field. After few attempts all the input cases were exhausted.

Additional Comments: *None*

6. Find Events Request API

Unit Details: *User can search different events in which he/she has interest in. User can search events through either entering the name of the event/s or the date and time he/she is interested in.*

Test Owner: *Deepak*

Test Date: *01/04/2022 - 02/04/2022*

Test Results: All different keywords were used to search events which includes many different input cases. After testing it thoroughly, all the bugs were removed from it.

Structural Coverage: Different input cases include keywords in which the events name has different case sensitivity(for eg. hall and HaLl), if a certain name comes in between two words (for eg. hall in Interhall), if there is simple spelling changes (for eg. circket and cricket) and the list goes on forever. Many of the bugs were removed, however some of them proved quite difficult to be removed by us(for eg the bug related to spelling changes).

7. Join Events Request API

Unit Details: *This API is used to allow user to join a particular event which he/she is interested in .*

Test Owner: *Bugada Ashritha*

Test Date: *01/04/2022 - 01/04/2022*

Test Results: The user will be successfully able to join the event and event in which he/she joined will be showed in the MY EVENTS section.

Structural Coverage: By clicking the join event once , the user will be able to join the event and by further clicking the user will automatically be withdrawn from the event and the event will be disappeared from his/her MY EVENTS section.

Additional Comments: *None*

8. My Events Get Request API

Unit Details: *This API is used to display the events the user has already joined in .*

Test Owner: *Bugada Ashritha*

Test Date: *01/04/2022 - 01/04/2022*

Test Results: It shows all the events joined by the user, time, venue at which it is the event is taking place and number of participants that already registered in the event.

Structural Coverage: All the events present in the database in which you have have joined were tried to rendered and fetched to fronted.

Additional Comments: *None*

9. Function isAlreadyExists().

Unit Details: *This function checks whether the event that user want to create already exists or not. It runs when user clicks on create event*

Test Owner: *Vedasree*

Test Date: *01/04/2022 - 01/04/2022*

Test Results: If the event already exists it prompts a message 'event already exists. *Join event?'.* Else it allows the program to continue.

Structural Coverage: *None*

Additional Comments: *None*

10. Function searchFunction().

Unit Details: This function allows the user to search for an existing event.

Test Owner: *Deepak*

Test Date: *01/04/2022 - 01/04/2022*

Test Results:

Structural Coverage:

Additional Comments: *None*

11. Create Interest Post Request API().

Unit Details: *The Create Interest Post Request API() is used to allow the user request his/her interest to be added in the events database.*

Test Owner: *Bugada Ashritha*

Test Date: *01/04/2022 - 01/04/2022*

Test Results: The user can successfully enter the interest he like to be added to the events database and would be able to see his is interest also.

Structural Coverage: After submitting the interest in the community page , the interest would be visible along with the other user's interest and other users would be able to like his/her interest.

Additional Comments: *None*

12. Community Get Request API().

Unit Details: This displays the community page which contains the existing interests section, a section to post the user's interests and also a my profile section.

Test Owner: *Deepak*

Test Date: *01/04/2022 - 01/04/2022*

Test Results: the community page was correctly displayed without any bugs.

Structural Coverage:

Additional Comments: *None*

13. Liked PUT Request API.

Unit Details: *This allows the user to like an existing interest.*

Test Owner: *Nitin Kembasaram*

Test Date: *01/04/2022 - 01/04/2022*

Test Results: The number of likes for the existing interest increases by one.

Structural Coverage: The like button under the existing event changes to "liked" and the number of likes increases by one thereby registering the user's like and storing it in the database.

Additional Comments: *None*

14. My Profile GET Request API

Unit Details: *The My Profile GET Request API is used to fetch the name , roll no ,username , Email id of the user along with the top 5 interests he is interested in from the database and renders it to the fronted Profile page.*

Test Owner: *Bugada Ashritha*

Test Date: *01/04/2022 - 01/04/2022*

Test Results: It successfully displays the name,roll no,username and Emailid of the user along with his/her profile photo and interests he/she is mostly interested in.

Structural Coverage: *None*

Additional Comments: *None*

15. Edit Profile API().

Unit Details: This allows the user to edit personal details like name, Roll number , email id and profile picture.

Test Owner: *Nitin Kembasaram*

Test Date: *01/04/2022 - 01/04/2022*

Test Results: The user will successfully be able to change their details.

Structural Coverage: The user will be able to edit his personal information and the changes will be reflected in the database accordingly.

Additional Comments: *None*

….

16. Function sendConfirmationEmail().

Unit Details: This function runs when a new user registers by filling in the username , email id and password.This function sends a confirmation mail to the user.

Test Owner: *Nitin Kembasaram*

Test Date: *01/04/2022 - 01/04/2022*

Test Results: The user receives a confirmation mail to the registered email id which contains a link upon clicking which the user will be successfully registered.

Structural Coverage: During the registration after filling in the details a confirmation email will be sent to the email id and a message "please check your email address to complete your registration process" will be displayed on the screen .

Additional Comments: *None*

….

3 Integration Testing

1. **User Signup and Email Verification**

Module Details: In this integration testing we integrated two majority different units present in our web app which is user registration and their email verification. Just after the user registers with their email id, they have to wait until their email id is verified. Only after that the email id is verified, the user can be allowed to sign in using their details.

Test Owner: Deepak

Test Date: 01/04/2022 - 01/04/2022

Test Results: This integrated system works perfectly fine with all different user inputs. Some bugs were found but they were also removed quickly. Some simple Structural coverage used to find different errors were inputting invalid email id and waiting it to verify, closing the window and trying to access the sign in page directly and so on.

Additional Comments: In this integration testing some other users who are not part of the team were also required since we all have exhausted all our email ids and thus required other people's email ids to test our app.

2. User Sign in and Rendering that user's data from Database

Module Details: In this top module, users were allowed to sign in, and thus all the user data specific to that user only, the user was fetched from the database to the backend and finally it was rendered in the frontend using ejs engines. The basic data specific to a particular user includes all the events he/she has joined and not joined, all the interests he/she has created, and all the interests which he/she has liked or not liked.

Test Owner: Deepak

Test Date: 02/04/2022 - 02/04/2022

Test Results: All the user data were fetched correctly and thus rendered correctly as well in the frontend. Since all unit testings were already done, the integration testing of this section did not create new bugs and thus the page worked flawlessly.

Additional Comments:

3. Integrating the fronted with the backend as well as the database

Module Details: This module required a significant amount of work as we had to integrate the three major components of our software. The different groups that worked on Database, Frontend, and Backend were involved and had to collaborate for this testing.

Test Owner: Janhvi

Test Date: 02/04/2022-02/04/2022

Test Results: There were a few issues with the design of the frontend and backend which rendered some features incompatibly, and our team immediately began work to resolve them.

Additional Comments: This was the part where the team collaboration was most needed and was carried out smoothly.

4. **Joining different events and rendering it in frontend Section**

Module Details: This module was designed to update the participants of an ongoing event. It takes care to achieve two main objectives - adding the event data to the user's events and adding the user data to the events database.

Test Owner: Janhvi

Test Date: 02/04/2022-02/04/2022

Test Results: The tests worked without any major bugs because the code was already written in such a way to reflect these changes before proceeding further.

Additional Comments:

5. **Creating Interest and rendering it in Community Page**

Module Details: This module takes in feedback from the user in the form of an interest name and a text feedback. The rendering was done in such a way that the list of feedback is visible on the community page without distorting the creating interest/add feedback section of the tab.

Test Owner: Janhvi

Test Date: 03/04/2022-03/04/2022

Test Results: The testing showed scrolling and rendering issues which were then fixed.

Additional Comments:

4 System Testing

1. **Requirement:** *External Interface Requirement*

Test Owner: Deepak

Test Date: 04/04/2022-04/04/2022

Test Results: The UI Interface was also one of the main aspects of any webapps. The interface was testing repeatedly many times on different devices such as I-phones, Android devices, I-pads, Laptop and computer systems. We also tested using the responsive feature of the chrome developer tools.

Additional Comments: *None*

2. **Requirement:** *Interacting with different Events*

Test Owner: *Deepak*

Test Date: 02/04/2022 - 02/04/2022

Test Results: This requirement includes all the different interactions related to the events present in our web app. It includes different methods like joining, creating, finding an event, showing all the events, showing only the user-specific events (which he/she has joined), etc.

Since unit testing of all these independent methods was done already, integrating it and thus running this system gives no error or bugs in our program.

Additional Comments: None

3. **Requirement:** *Interacting with IITK Community*

Test Owner: *Deepak*

Test Date: 02/04/2022 - 02/04/2022

Test Results: This system interact with the IITK community which has registered in our webapp. It includes many different functions like creating interest, liking and disliking others and their own interest, updating their profile picture,etc. Some of this functions created some bugs, for eg. edit-profile section used to give errors if the profile picture was not uploaded (since it was tying to access the null value). Thus kind of errors were removed after a thorough testing of this system.

Additional Comments: None

4. **Requirement:** *Performance Requirement*

Test Owner: *Deepak*

Test Date: *03/04/2022 - 03/04/2022*

Test Results: *Performance issue is one of the most important problem we faced since it was our first time developing any complete web app. Since we used ejs view engines, the loading time were significantly reduced. Also the data stored in the database was in json string so during parsing of this data we used json.stringify() option so to reduced the fetching amount required in the process.*

Additional Comments:

6. **Requirement:** *Safety and Security Requirement*

Test Owner: *Deepak*

Test Date: 02/04/2022 - 04/04/2022

Test Results: The security and safety testing was one of the most important non-functional requirements in our web app. We tried many different approaches to reduce the security concerns related to this app. In this system testing, we tried to access different URLs without signing in, tried accessing the super-user page by accessing its URL, tested verify-email system once again thoroughly, etc. We passed a number of middleware functions in our all the API's so that all the system remains safe.

Additional Comments:

….

5 Conclusion

How Effective and exhaustive was the testing?

Which components have not been tested adequately?

What difficulties have you faced during testing?

How could the testing process be improved?

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 produce this document*>