UMASS EVENTS BOARD

GROUP MEMBERS:
Muni Lohith Krishna Mohan Konidala
Muneer Ahmed
Shruti Ramesh Babu Mallavolu

GROUP NAME: Godfather

I. INTRODUCTION

There are many Clubs, Chapters and Communities responsible for conducting events in a huge campus like UMass. As a student, it is very hard to keep track of them. This application- 'UMass Events Board' aims to solve this problem by creating a one-stop for such events and allowing users(students, any other staff) to view and register for these events. The Events Board application consists of many functionalities that makes it user-friendly to view, navigate and register for new Events, and Associations.

Each of these functionalities are explained in detail below and the relevant test results are also attached.

II. FUNCTIONAL REQUIREMENTS

Stakeholders:

- Users (students, staff)
- Associations (Clubs, Chapters and other associations)
- Administrator

Normal User

- A Normal User can be any student, faculty or staff of UMass.
- They should be able navigate through their Profile, Home, Upcoming Events, Past Events, My Associations, Explore Associations, and Explore Events.
- The Profile tab should show the User Type, First Name and Email of the user.
- The Home tab should display the events not followed by the user. These events should belong to the Associations that the user follows.
- The Upcoming Events should display all the events that the user is going to attend in future.
- The Past Events tab should display all the events that the user has attended in the past.
- My Associations tabs should display all the associations that the user follows.
- The Explore Associations tab should display all the associations that the user doesn't follow.
- Explore Events should display all available events that are going to happen in the next 5 days.

Association

- An Association should be any user handling a Club, Community or an Association.
- A user cannot login as an association unless approved by the administrator.
- Once they are approved, they should be able to navigate through the Profile, Upcoming Events, Past Events, and Create an Event.
- The profile page should show the association details like User Type, First Name and Description.

- All the future events of that association should be displayed in the Upcoming Events tab.
- All the events of the association that have already occurred should be shown in the Past Events tab.
- The user should be able to post an event by navigating to the 'Create an Event' tab.
    - This tab should contain a form for the Association to create an event.
    - The form should contain details like Name, Description, Venue, Start Time and End Time of the event.
    - Name, Description, Start Time and End Time should be set as required fields.

Admin

- An admin should have access to approve/reject associations and hence admin only has the Home and Profile tabs
- Profile tab should show profile details like User Type, First Name and Email of the user.
- Home tab should show all the Associations that need to be approved. The admin should be able to either Accept or Reject a new Association.

Other Functionalities

- The user should be able to view the Login page first to enter the credentials. Proper authentication should be done in the back-end before accepting the login data. If not, the user should be provided with the option to either 'Register as a User' or 'Register as an Association'
- 'LOG OUT' option should be available in all the HomePages and it should redirect the user to the Login Page.
- 'BACK' button should be implemented in the Registration Pages and should redirect the user to the Login page when clicked.

III. NON-FUNCTIONAL REQUIREMENTS

Understandability

Understandability is one of the key Non-functional requirements of any software application system. It basically makes sure that the system can be easily understood even by a new user or developer. This becomes especially useful during an Incident Response or a bug resolution.
The application has separate folders for the Front-End and Back-End logic. In Front-End, each of the UI Pages are in the ui.pages folder, and the other components are present in their respective folders. The micro-services, request handlers and the routers are all in separate folders.

All the custom code written has associated comments attached to it and the variable names are easy to associate with their functionalities. Each function is written in a separate method which also follows proper naming conventions.

Testability

Testability shows how easy it is to verify or test any of the functional requirements or components in the application. It is important for the code to support this functionality as it becomes easy to identify faulty areas of code. It makes sure that the system is working as expected.

The Events Board application has substantial test cases written for all the Back-end logic that covers more than 65% of code. Alpha testing/Manual testing has been performed to test the Front-end design and functionality. Beta testing was also executed which documents feedback from 2-3 users about the interface and application.

Modularity

The functionalities of the software project can be divided into separate modules. These modules should be easy to understand, and loosely coupled such that they work efficiently when combined together.

The Events Board has modular components that work well on their own and also when combined. Each microservice is written in a separate Java class that can be run on its own. The UI pages are also divided into separate modules/classes that can be run on their own. This makes it very easy to make changes to a particular module in order to improve the functionality. It also makes the application more understandable.

Extensibility

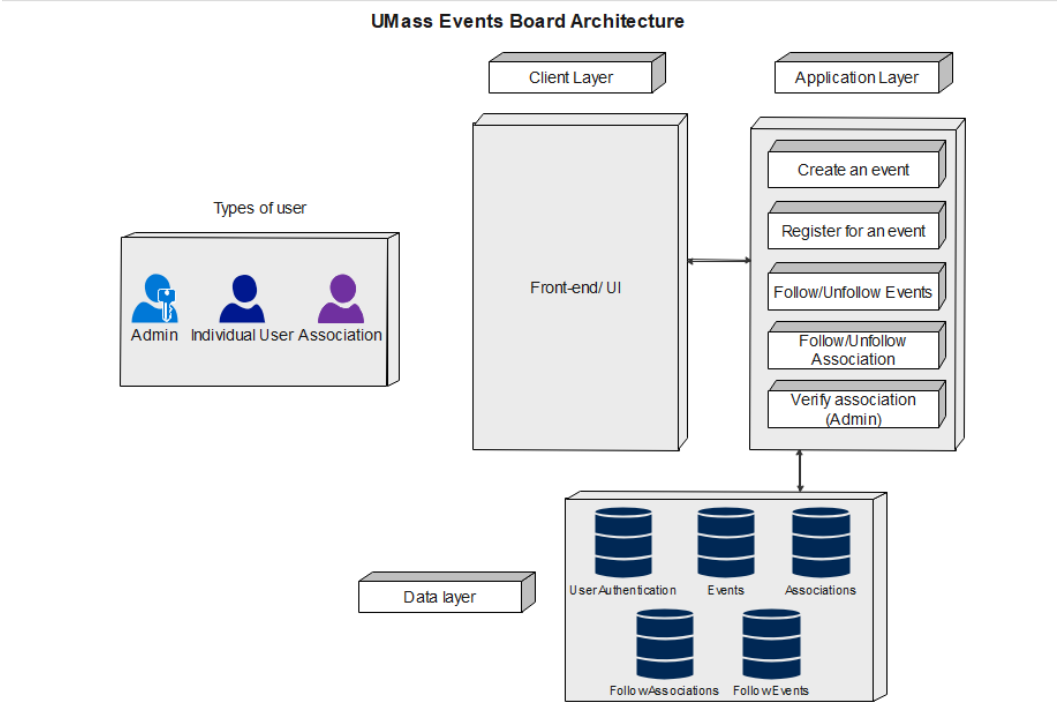
Extensibility measures how easily the designed application allows adding new functionality or components. The new functionality should primarily be extending the application already present. This is especially necessary when the user needs/requirements are dynamic and also when there's a need to align with the competitive market.

Events Board application is structured in such a way that it makes it very easy to add new features and build on top of the existing system. A new page can be easily created and integrated to the HomePage. The components folder can be easily extended to include new features or components that can be used in the HomePage. In the backend, a new microservice can be easily added to introduce a new table and extend the database.

IV. SYSTEM REQUIREMENTS

- Front-End, Back-End - Apache Netbeans IDE 15, JDK 19
- Database - MySQL Server 8.0 must be up and running. It should be connected to the application server after authenticating the credentials.
- A 32-bit or 64-bit system that supports these softwares is necessary to run the application.
- The tech stack used here is Java Swing for front-end, MySQL database, and Java for back-end logic. The test cases are developed using JUnits.

V. ARCHITECTURE



Client-Server Architecture has been implemented in this project; depicted in the diagram above. As the name suggests, the Client-Server architecture consists of a Client and a Server. The Client could be a computer that basically receives data or service from the server. The Server consists of services or data that it gives access to a Client on authentication. The client communicates with the server through REST API calls.

The Client layer has the code for the front-end and it is connected to the back-end through appropriate REST APIs. The Application Layer consists of functionalities like ‘Create an Event’, ‘Register for an Event’, ‘Follow/Unfollow an Association’, ‘Follow/Unfollow an Event’, and ‘Verify Association’. The server processes all the REST API calls that are requested by the client. The Data layer consists of Users, Events, Associations, Follow Associations and Follow Events tables.

VI. IMPLEMENTATION

Client

The Client layer keeps frequently communicating with the Server layer through REST API calls. It makes a GET request to fetch the details of the user on login. After the credentials are validated, the user is redirected to their Home Page.

If the user is not already registered, they can either register as a User, or an Association if they are handling any of the clubs/associations. The user needs to fill the form and click the Register button which performs a POST request to insert the new user in the Users table. Similarly, associations have a different register page, which upon registering does a post call to add a new row to Users and the Associations table. To register as an Association, the Admin's approval is required. When the Admin approves the request for a new Association, a POST request is called to update the approval_status flag in the Associations table. If the Admin rejects the request, the corresponding rows in Users and Associations table will be deleted.

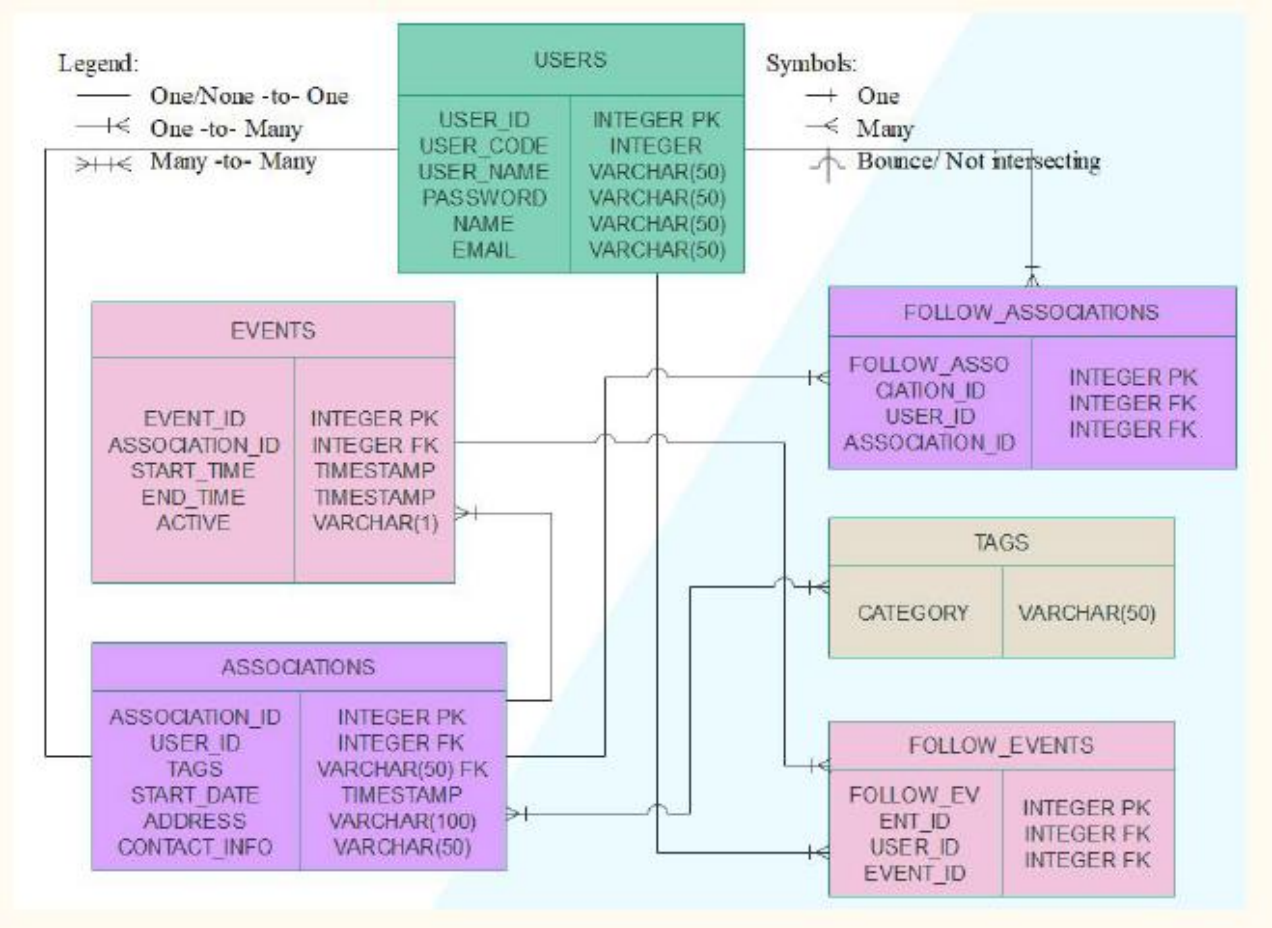
The User can follow or unfollow any of the Associations and the Events. These operations are carried by POST calls to Associations and Events microservices respectively. Each Association can create an event by making a POST call to the Events microservice. When the user follows an Association, all the events of those Associations are displayed in the HomePage. These events are basically fetched through a GET request call to the Events microservice.

The Association has the option to 'Create an Event'. The Association must fill all the required fields and the Action Performed event is called on clicking the 'Post' button. All the fields are validated and on success, a new Event is created for this Association.

Server

We have three microservices Users, Events and Associations that are running on different ports. They are HttpServers, which can take rest calls as inputs and process them and send the response to the client. Users microservice, processes all the requests related to creation and login of a user. Associations microservice processes all requests related to an Association. For example: Create an Association, Approve/Reject an association, Follow/Unfollow an Association etc. Similarly Events microservice processes all requests related to an Event. For Example: Create an Event, Follow/Unfollow an Event, get Events feed etc.

Database



A database contains data in a structured format. All the data is stored in the MySQL database. MySQL is a relational database management system that lets the user access, add, update, and delete the records stored in the database tables. SQL (Structured Query Language) is used to access the MySQL database.

There are 6 tables in the database - Users, Associations, Events, Follow_Associations, and Follow_Events, Tags. Tags is an additional table that we are currently not using. This table was created to link associations with their type. The User table consists of all the user records whether it be a Normal User, an Association or an Admin. The Associations table consists of Association data, along with an extra column approval_status, which takes the values ‘Y’ or ‘N’. The Events table consists of all the Events posted by the Associations. The Follow_Associations table consists of all the Associations that each of the Users is following. Similarly, the Follow_Events table consists of all the Events that each of the Users is following.

The **Database Schema** for each of these tables is given as follows:

1. Users -
    - It consists of an Auto-Incremental user_id field which gives an integer value. It is set to Not Null and it is the Primary key in this table.

- The user_code is a varchar field which is also set to Not Null. It is set to 0 for Administrator, 1 for an Association and 2 for a Normal User.
- The user_name is the input collected from the User Registration Page under the User Name field. It is a required field as it is set to Not Null. The user_name should always be unique.
- The Password field in the User Registration Page is also required as it is a Not Null field in the User table.
- The first_name field is also required and it is collected from the First Name text field in the Registration Page.
- The email field is set to Not Null and accepted via the Email field in the User Registration page.
- last_name is a varchar field whose data is collected from the input given in the Last Name textfield from the Registration page.

2. Associations -
   - It consists of an Auto-Incremental association_id that cannot be empty. It is set to a Primary Key.
   - user_id field is also set to Not Null and is retrieved from the User table. It is set as a Foreign Key.
   - An association_name must be given as a varchar and it is retrieved from the Name field in the Association Registration page.
   - A description field is also set to Not Null. It is retrieved from the Description, a text area field in the Association Registration page and it is used to add important information about the Association.
   - Three more fields - address, contact_info, and email can also be given optionally and they are set as varchar fields.
   - The approval_status field is marked true if the Association has been approved by the Administrator.

3. Events -
   - The field event_id is a Not Null Auto-Increment field that is set as the Primary Key.
   - It has a field association_id that is associated with it and taken from the Association table. This field is set as a Foreign Key.
   - Two timestamps are present - start_time and end_time which are taken from their respective fields in the Association Registration page.
   - Further, it has Name and Description fields that are set to Not Null.
   - A venue field is set as varchar and it can be optionally given by the user.

4. Follow_Associations -
   - The follow_association_id is set to Not Null and it is Auto-Incremental. It is set to be a Primary key.
   - The user_id is set to Not Null and it is taken from the User table. It is set as a Foreign Key.
   - Other required fields are association_id which is taken from the Association table.

- An extra Constraint is created to make sure that there is only one set of (user_id, association_id). It basically handles the case when more than one row can be created with the User associated with the same Association.

5. Follow_Events -
   - The follow_event_id field is a Primary Key that is set to Not Null and it is Auto-Incremental.
   - The user_id is a Foreign key and it is retrieved from the User table.
   - The event_id is another Foreign key retrieved from the Event table.
   - An extra Constraint is created to make sure that there is only one set of (user_id, event_id). It basically handles the case when more than one row can be created with the User associated with the same Event.

VII. DESIGN

The Design of the User Interface (UI) is sleek, modern and uses the UMass colors maroon, and white.

Login Page

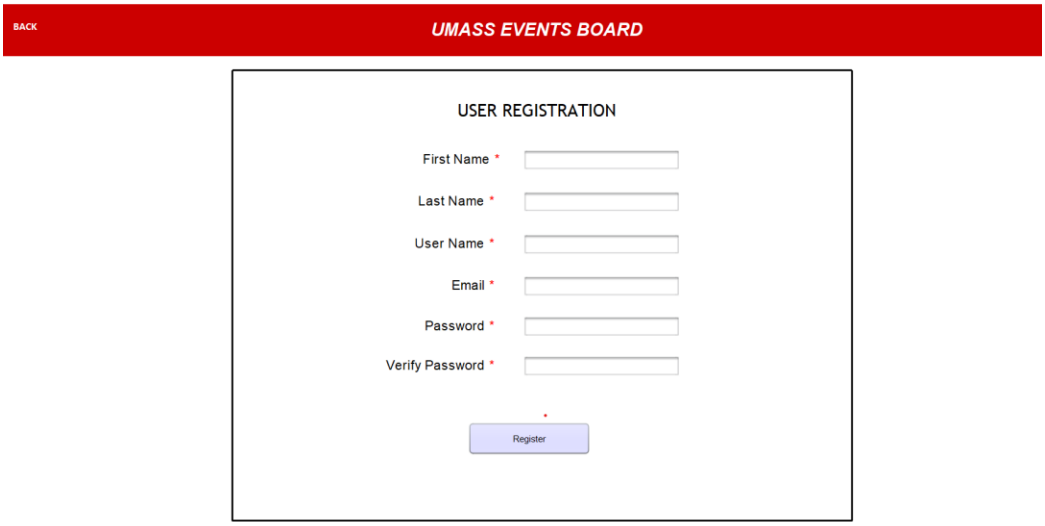
- Login page consists of User Id and Password that must be entered by the user.
- The credentials must be authenticated by the user details present in the database.
- The user is logged in their respective Pages based on their type - User, Association, Admin.
- It also provides buttons to 'Register as a User' or to 'Register as an Association'. The buttons navigate the user to either the User or the Association Registration Pages respectively.
- Need to enter valid UserName, and Password. If valid Username and Password are not entered, the system throws an error message.

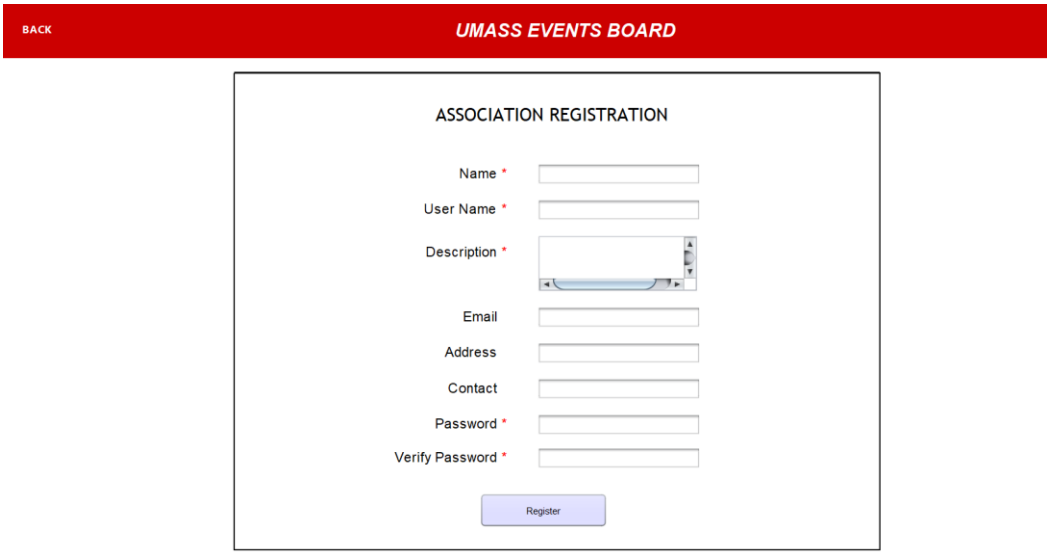
Registration Page

There are two Registration pages - one is for Users and the other one is for Association.

- The User Registration page needs details like First Name, Last Name, User Name, Email and Password.

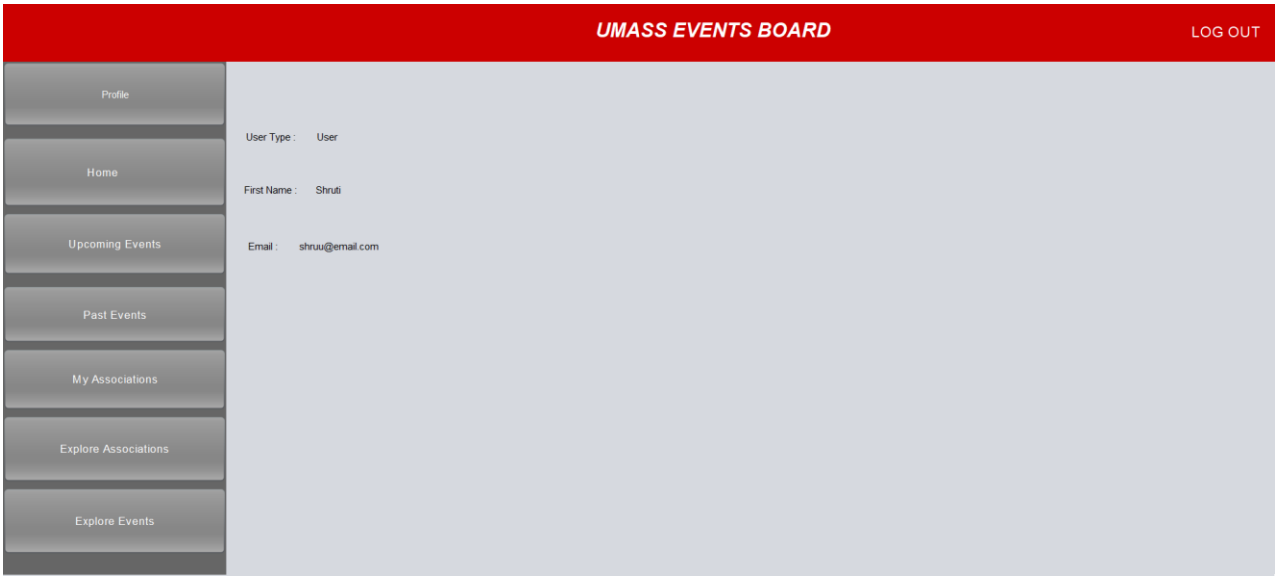
- All of the fields are required to populate the data in the User table.
- The Association Registration page needs details like Name, User Name, Description, Email, Address, Contact, and Password.
- Name, User Name, Description and Password are required fields to populate the data in the User table. Once it is approved by the Admin, the data is populated in the User table.



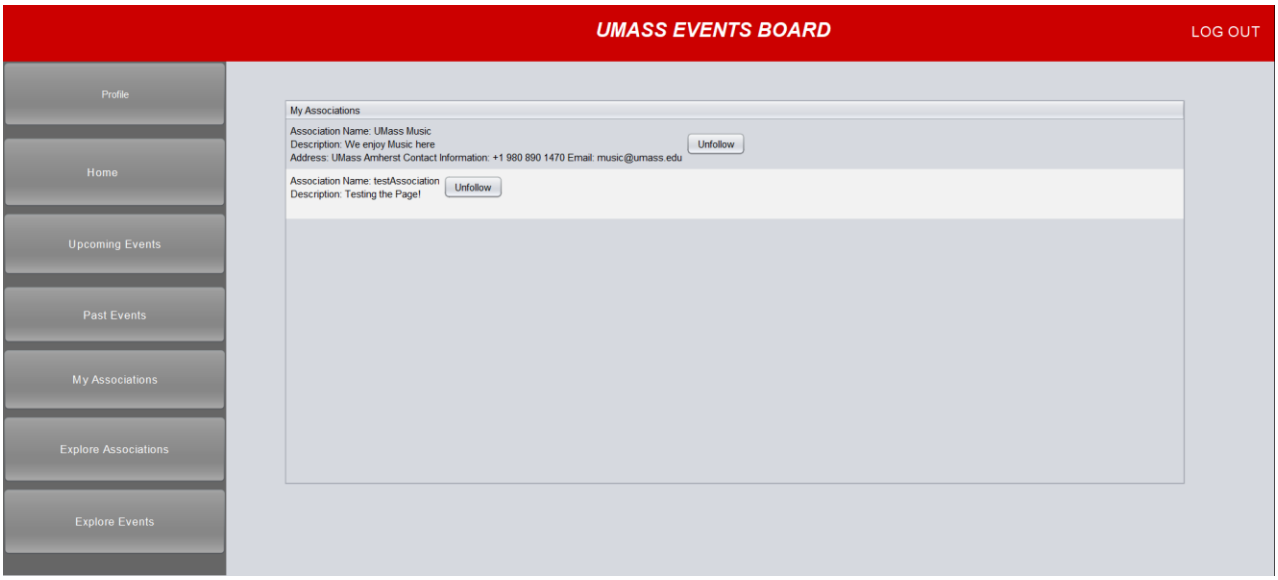


Other Design Choices

- A Navigation Bar is fixed on the side of the page to navigate through the Pages. This bar is a fixed panel and has relevant buttons corresponding to the type of User. Only the panel integrated with the Button on the NavBar is displayed on click. The other panels are set to hidden.



- Custom JTables have been reformatted to display the Events and Associations in the Home, Upcoming Events, and Past Events panels.



VIII. TESTING AND EVALUATION

Line and Branch Coverage

We have achieved around 66% of Branch coverage and 55% of Line Coverage.

We used JUnits 4 and JACOCO framework to extensively test the backend and generate test reports. These are the following line and branch coverages we achieved in various packages:

EventBoardBackEnd

Sessions

EventBoardBackEnd

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
get_requests_handlers		62%		82%	18	59	116	350	0	8	0	3
post_requests_handlers		42%		44%	30	51	148	265	0	8	0	3
microservices		0%		n/a	6	6	45	45	6	6	3	3
url_processor		95%		100%	0	7	2	31	0	3	0	1
routers		100%		75%	3	12	0	39	0	6	0	3
Total	1,268 of 2,867	55%	69 of 208	66%	57	135	311	730	6	31	3	13

Created with JaCoCo 0.8.8.202204050719

EventBoardBackEnd > get_requests_handlers

get_requests_handlers

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
EventsGetRequestHandler		59%		77%	11	27	52	149	0	3	0	1
AssociationsGetRequestHandler		70%		87%	6	27	36	143	0	3	0	1
UsersGetRequestHandler		48%		83%	1	5	28	58	0	2	0	1
Total	568 of 1,514	62%	18 of 102	82%	18	59	116	350	0	8	0	3

EventBoardBackEnd > post_requests_handlers

post_requests_handlers

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
AssociationsPostRequestHandler		37%		45%	16	27	91	147	0	3	0	1
EventsPostRequestHandler		56%		57%	8	16	34	80	0	3	0	1
UsersPostRequestHandler		31%		8%	6	8	23	38	0	2	0	1
Total	551 of 959	42%	48 of 86	44%	30	51	148	265	0	8	0	3

EventBoardBackEnd > url_processor

url_processor

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
ProcessURL		95%		100%	0	7	2	31	0	3	0	1
Total	5 of 116	95%	0 of 8	100%	0	7	2	31	0	3	0	1

EventBoardBackEnd > routers

routers

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
EventsRequestRouter		100%		75%	1	4	0	13	0	2	0	1
AssociationsRequestRouter		100%		75%	1	4	0	13	0	2	0	1
UsersRequestRouter		100%		75%	1	4	0	13	0	2	0	1
Total	0 of 134	100%	3 of 12	75%	3	12	0	39	0	6	0	3

User Acceptance Testing

Project Name:	Event Board
Team Member	Muni Lohith Krishna Mohan Konidala
Team Member	Shruti Ramesh Babu Mallavolu
Team Member	Muneer Ahmed
Release Date:	

Beta Testers	Sri Charan
Beta Testers	Manish Reddy
Beta Testers	Rohan

Item	Acceptance criteria or Requirement	Test Type	Owner or Responsible	Priority	Test Result	Comments
1	Null Input	All pages	Alpha Testing	High	Accept	Popup's up "Please enter valid username and password"
2	Invalid Input	Login Page, User, Association registeration page	Alpha Testing	High	Accept	Popup's up "Please enter valid username and password"
3	Correct input for user	Login Page	Alpha Testing	High	Accept	Refer Alpha_testing_report
4	Correct input for approved association	Login Page	Alpha Testing	High	Accept	
5	Correct input for unapproved association	Login Page	Alpha Testing	High	Accept	
6	Correct input for admin	Login Page	Alpha Testing	High	Accept	Refer Alpha_testing_report
7	Logging in as association	Login Page	Alpha Testing	High	Reject	Refer Alpha_testing_report
8	Already existing userid	User, Association Registration Page	Alpha Testing	Moderate	Accept	Refer Alpha_testing_report
9	Mismatch password and verify password	User, Association Registration Page	Alpha Testing	Moderate	Accept	Refer Alpha_testing_report
10	Back button	User, Association Registration Page	Alpha Testing	Moderate	Accept	
11	Invalid character inputs	User Registration Page	Alpha Testing	Moderate	Reject	Refer Alpha_testing_report
12	Invalid character inputs	Association Registration Page	Alpha Testing	Moderate	Reject	Refer Alpha_testing_report
13	Button clicks	All homepages	Alpha Testing	Low	Accept	Refer Alpha_testing_report
14	Related panels and pages(like Profile, events panel etc	All homepages	Alpha Testing	Low	Accept	
15	Initial Test	All homepages	Alpha Testing	Low	Accept	Popup's up the Home Panel
16	Following an event/association	User home page	Alpha Testing	Low	Accept	Gives popup "Followed the association: Umass SASA"
17	Unfollowing an event/association	User home page	Alpha Testing	Low	Accept	Gives popup "Unfollowed the association: Umass SASA"
18	Creating a new event	Association home page	Alpha Testing	Moderate	Accept	Gives popup "Succesfully created an event!"
19	Approving an association	Admin home page	Alpha Testing	Moderate	Accept	Gives popup "Approved associationName"
20	Rejecting an association	Admin home page	Alpha Testing	Moderate	Accept	Gives popup "Rejected associationName"
21	Exiting the application	All pages	Alpha Testing	Moderate	Accept	Exits the application and closes all the jFrame's
22	Logout	All pages	Alpha Testing	High	Accept	Logs the user out and returns back to the login page
23	User Charan		Beta Testing			
24	User Rohan		Beta Testing			
25	User Manish		Beta Testing			

As part of User Acceptace testing we have performed both alpha and beta testing (we have documented this in '**TestReports**' folder as excel file named '**UserAcceptanceTesting**'):

Alpha Tests

Alpha testing is performed initially when an application has been developed. It is used to test if an application is performing as expected. They are carried out by the developers, testers or other people within the organization responsible for developing the application. These tests are generally followed up by beta tests.

For Alpha testing, we have performed manual testing on the Front-end UI and have also attached a file to show all the tests performed and the relevant screenshots. It is in the **'TestReports'** folder and named '**AlphaTestingReport'.**

Beta Tests

Beta Testing is carried out by end users and is generally performed for understanding the market value and to check other parameters like reliability and robustness.

We have had beta testing performed by three independent users, namely:

- Charan
- Manish
- Rohan

Their reports are present in the project folder **'TestReports'** under the name '**BetaTesting**'.

IX. DEBUGGING EXPERIMENTS

We encounter various bugs and errors during the course of our project. Some were related to UI whereas others were related to the REST calls or integration.

We used different methods and tools to debug like making use of print statements and Postman to methodically perform debugging experiments and fix our code.

X. CONCLUSION

Our project consists of many complex UI functionalities and tables in the database. We have implemented Microservices, integrations, test cases, UI components using the best Software Design Principles. Our project also follows all the relevant functional and non-functional requirements. There is definitely room for improvement by adding new features like a search bar and event tags associated with each event. Overall, our implementation consists of all the concepts that were taught in class which helped us extend our knowledge beyond the theoretical aspect of this course.

XI. APPENDIX

Link to Github repository

Link to Demonstration Video