# a) Requirement:

- Develop the architecture and working code of backend for sports management module.

- Make sure that the backend application mananges objects efficiently , eg ( ‘database connection’ object is created once and managed throughout the application when it is running.)

- As mentioned in the ‘Deliverable-2’, create API endpoints to manage the following

1. User registration
2. User login verification
3. Get all event data (UPCOMING,ONGOING)
4. Post a message in discussion forum
5. Retrieve all messages posted in the discussion forum

# Backend Environment Setup

- Extract the contents in the file sportsevent-backend.7z

- Make sure JDK ,JRE and MVN are installed in local environment

- Install STS from <https://spring.io/tools> to run the application as Spring Boot application

- Open the STS Eclipse application and Import the extracted project

- Run ‘mvn install’ for the project to download all the necessary dependency files.

# Backend Project setup

- Install Postman, Mysql and mysql workbench in local

- Run the ‘CreateTablesAndSchema’ file in src -> main -> java -> com -> sportsevent -> backend -> database path as a java application.

- After running the application make sure the schema ‘sports\_event’ is created in mysql.

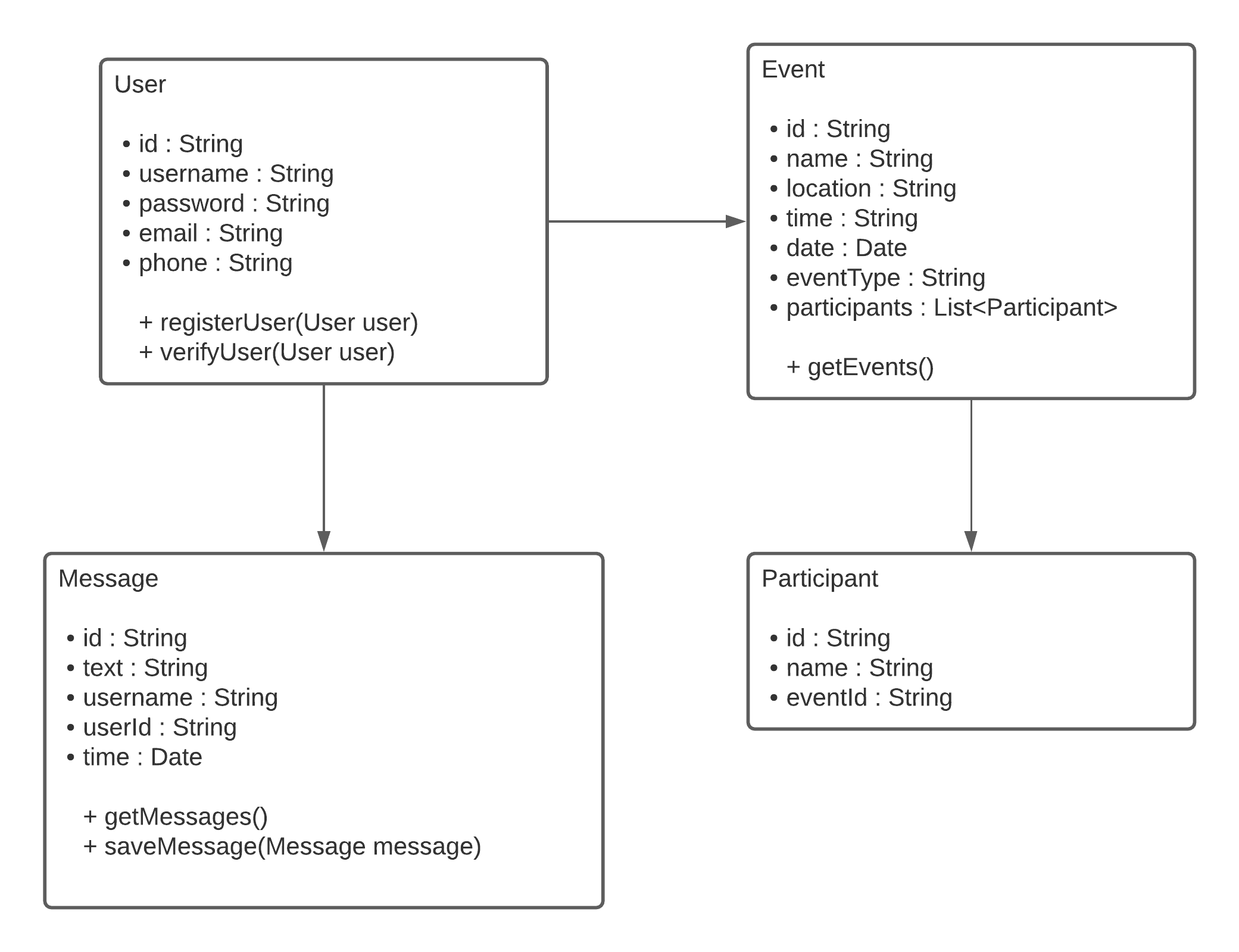
- Also check if the tables ‘user’, ‘event’, ‘participant’ and ‘message’ tables are created in the schema.

- Open mysql workbench and import the ‘event.csv’ and ‘participant.csv’ files to their respective tables

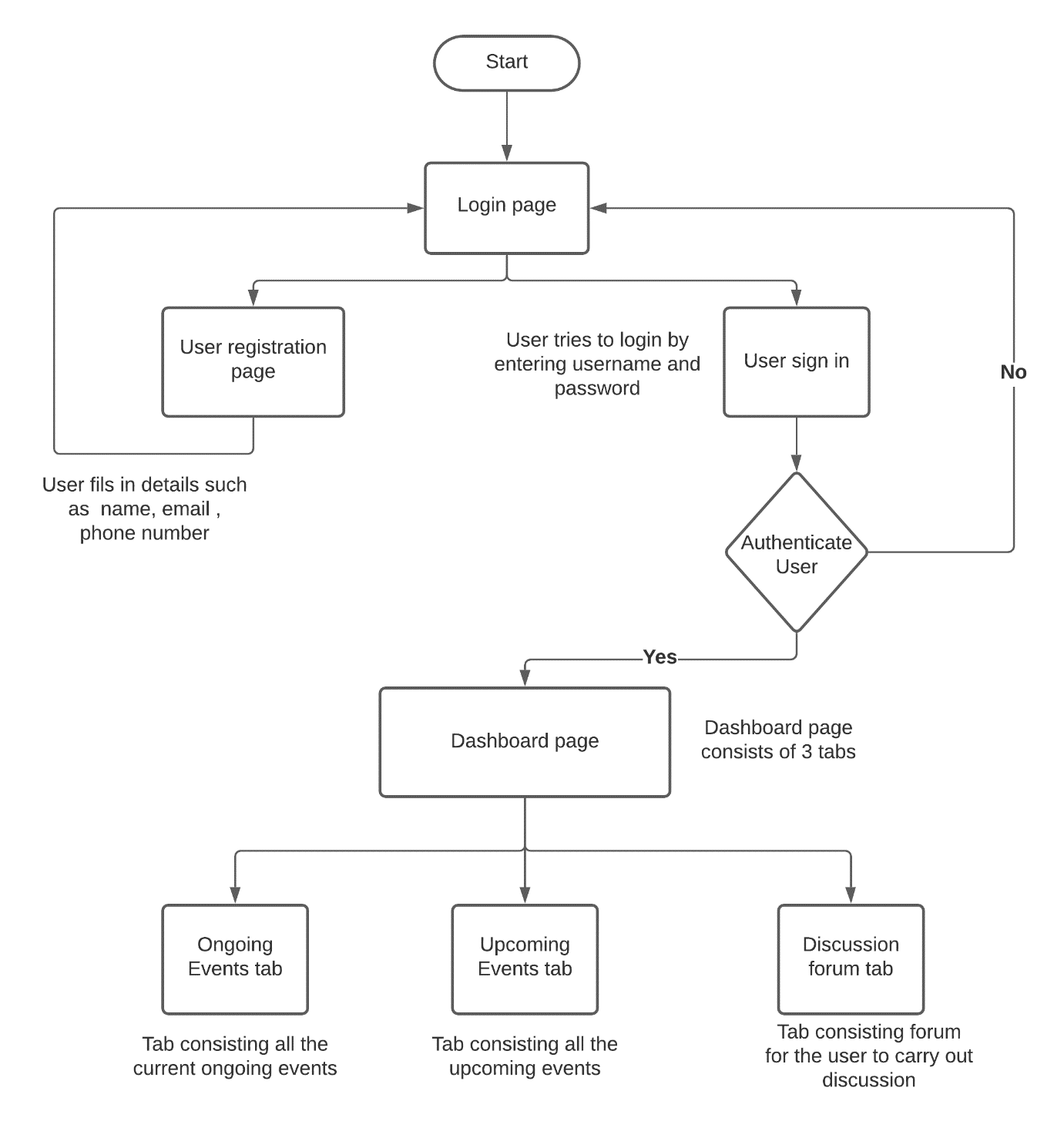
- Now run the project as spring boot application

- Open the postman application and check if the backend Rest endpoints are working by calling the endpoints (Check below for API information)

# b) Class Diagram



# Sequence Diagram



Verify user (API to be called when user Signs In)

endpoint – <http://localhost:8080/user/login>

method – POST

requestBody example -

{

“username” : “root”,

“password” : “root”

}

sample response 1(If user accout is registered)

- true

sample response 2(If user accout is not registered)

- false

Register user (API to be called when user registers)

endpoint – [http://localhost:8080/user](http://localhost:8080/user/login)/register

method – POST

requestBody example -

{

“username” : “root”,

“password” : “root”,

“email” : “[sample@gmail.com](mailto:sample@gmail.com)”,

“phone” : “+19829982891”

}

sample response 1(If user registration is successful)

- true

sample response 2(If user registration is not successful)

- false

Get EVENTs (API to get all the EVENTS)

endpoint – [http://localhost:8080/s](http://localhost:8080/user/login)ports/events

method – GET

sample response :

{[ {

“id” : 1,

“name” : “Long Jump”,

“Location” : “USA”,

“time” : “9:00 – 10:00”

“date” : “11-10-2021”,

“eventType” : “ONGOING”,

“participants” : [

{“id” : 1,

“name” : “John”,

“eventId” : 1},

{“id” : 2,

“name” : “wick”,

“eventId” : 1}

]},

{

“id” : 2,

“name” : “High Jump”,

“Location” : “USA”,

“time” : “9:00 – 10:00”

“date” : “12-10-2021”,

“eventType” : “UPCOMING”,

“participants” : [

{“id” : 3,

“name” : “David”,

“eventId” : 2},

{“id” : 3,

“name” : “Charles”,

“eventId” : 2}

]}

]}

Get Messages (API to get all the Messages for discussion forum)

endpoint – [http://localhost:8080/](http://localhost:8080/user/login)forum/discussions

method – GET

sample response -

{[

{

“id” : 1,

“text” : “Hello”,

“userId” : 1,

“time” : “2021-10-11 20:35:43.110”,

“username” : “John”

},

{

“id” : 2,

“text” : “Good morning”,

“userId” : 2,

“time” : “2021-10-11 20:36:43.110”,

“username” : “Wick”

}

]}

Get Messages (API to get all the Messages for discussion forum)

endpoint – [http://localhost:8080/](http://localhost:8080/user/login)forum/discussions

method – GET

sample RequestBody -

{

“text” : “I would like to know the timing of the High Jump event”,

“userId” : 1

}

sample response 1(If message is saved in DB successfully)

- true

sample response 2(If message is not saved in DB successfully)

- false

**c) Test cases:**

Login page

Student register page

**Test Cases for Login Page:**

**Scenario1:** Login with correct credentials.

1. Verify that on page load “User Name” text box should contain the cursor.

2. Verify that Login page should contain all the elements like “User Name”, “Password”, “Login Button”, “Forgot Password Link” and “Sign In link”.

3. Verify that Password field is masked.

4. Verify the functionality of Login button.

5. Verify the functionality of tab key.

6. Verify the functionality of enter key.

7. Verify that the User Name entered is correct or not.

8. Verify that the Password is correct or not.

9. Verify that the correct page is redirected after logging in.

**Scenario2:** Login with Incorrect credentials.

1. Verify that on page load “User Name” text box should contain the cursor.

2. Verify that Login page should contain all the elements like “User Name”, “Password”, “Login Button”, “Forgot Password Link” and “Sign In link”.

3. Verify that Password field is masked.

4. Verify the functionality of Login button.

5. Verify the functionality of tab key.

6. Verify the functionality of enter key.

7. Verify that the User Name entered is displayed as Incorrect.

8. Verify that the error message to be displayed on page load (“401 Error”).

**Test Cases for Registration Page:**

**Scenario1:** Sign In with the Student Details.

1. Verify that on page load “First Name” text box should contain the cursor.

2. Verify that all the elements are present in the sign in page like “First Name”, “Middle Name”, “Last Name”, “StudentID”, “Sex Radio Button”, “Mobile No.”, “Email”, “Password”, “Verify Password”, and “Sign In Button”.

3. Verify that Password field is masked.

4. Verify the functionality of Login button.

5. Verify the functionality of tab key.

6. Verify the functionality of enter key.

7. Verify that the email entered is valid or not.

8. Verify that the password fields are matched.

9. Verify that all the fields are mandatory fields.

10. Verify that student ID Entered is matching with the data in the database.

| Team members | Contribution | Technical stack |
| --- | --- | --- |
| Aditya | SQL Developer | Repository for backend |
| Anudeep | Backend Developer | Discussion forum |
| Abhiram | Backend Developer | Discussion forum |
| Charan | Backend Developer | Events backend |
| Nitesh Reddy | Backend Developer | User registration |
| Chaitanya | Front end Developer | Events backend |
| Manvitha | Front end Developer | SQL backend |
| Sai Krishna | Front end Developer | user authentication |