

Infosys Internship 5.0 -BATCH 02

Sports Performance Monitoring and Event Management Platform

-BY Legends Team

Milestone 1: Week (1 to 3)

Module 1: User Authentication and Role Management

- **Description:** Implement user authentication using JWT and role-based access control for athletes and administrators.

- **Features:**

- User registration and login for athletes and administrators.
- Secure access using JWT tokens.
- Role-based access control to restrict access to specific functionalities (event creation, result publishing).

Week 1 Tasks for Our Team:

- Develop a homepage with registration and login functionality for athletes, coaches, and administrators.
- Integrate the database to store user registration data.
- Implement successful authentication on the login page.
- Ensure secure access using JWT tokens.

Week 1 Tasks Completed by Our Team Members:

Task 1: Develop a homepage with registration and login functionality for athletes, coaches, and administrators.

Completed by: **Jui Mistry & Siva Sai**

Task 2: Integrate the database to store user registration data.

Completed by: **Maria Joesilin**

Task 3: Implement successful authentication on the login page.

Completed by: **Gokul & Kumaresan**

Task 4: Ensure secure access using JWT tokens.

Completed by: **Kumaresan & Gokul**

Task 1:

Create a responsive home page



Create a signup page with user details below

- Username
- Email
- Create password
- Choose your role (Athlete/Coach/Admin)

Register

Username

Email

Create password

Choose your role

Submit

Already have an account? Login

Create a login page to login with user details below

- > Username / Email
- > password

Login

Username or Email

Password

☐ Remember me

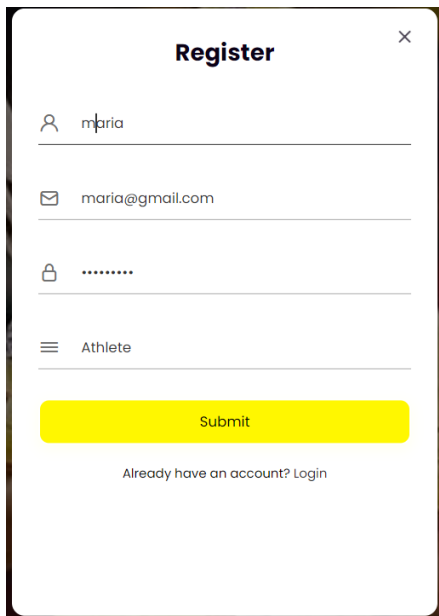
Forgot password?

Login

Don't have an account? Register

Task 2: Integrate the database to store user registration data.

Frontend Register

A screenshot of a web application's 'Register' form. The form is titled 'Register' and has a close button (X) in the top right corner. It contains four input fields: a username field with 'm'aria', an email field with 'maria@gmail.com', a password field with '.....', and a role field with 'Athlete'. Below the input fields is a yellow 'Submit' button. At the bottom, there is a link that says 'Already have an account? Login'.

Database storage

id	email	password	username
1	mahesh@gmail.com	\$2a\$10\$TUacVO1mBqA53er0pGr8Muhqy/YBU.g...	mahesh
2	reddy@gmail.com	\$2a\$10\$WFL8mRAze5gEfbdvXLoZb.GwFNcjpCJ...	ravi
3	gokul@gmail.com	\$2a\$10\$mSmYprdN/ay6kEhe5gyq3.bOPbcPMOP...	gokul
4	kumar@gmail.com	\$2a\$10\$wRLi3uIA31A/DoQE1XWcMuY0mTwdic....	kumar
5	maria@gmail.com	\$2a\$10\$MwsO..gK94Rb4g2zNOO6wus.20gTKJ...	maria
6	maria1@gmail.com	\$2a\$10\$ifUcszSzk81TFg/nry4exd4nwyMOU4...	mam
7	kumaresan.22it@sonatech.ac.in	\$2a\$10\$PMNuDXCfhtYyhlQz9g.deVxgNoHxw...	kumaresan98
HULL	HULL	HULL	HULL

The user **registration data** is sent to the backend **controller** in **JSON** format via **JavaScript**. The controller receives this data at the **/api/auth/signup** endpoint, maps it to the **User** entity class, and stores the user details in the **MySQL Workbench** database using **JPA**.

WORKFLOW:

Frontend Data Submission

- User inputs (username, email, password, role) are sent as a JSON POST request to the **/api/auth/signup** endpoint.

Controller Mapping

- AuthController maps the incoming data to a RegisterRequest object and forwards it to UserService.

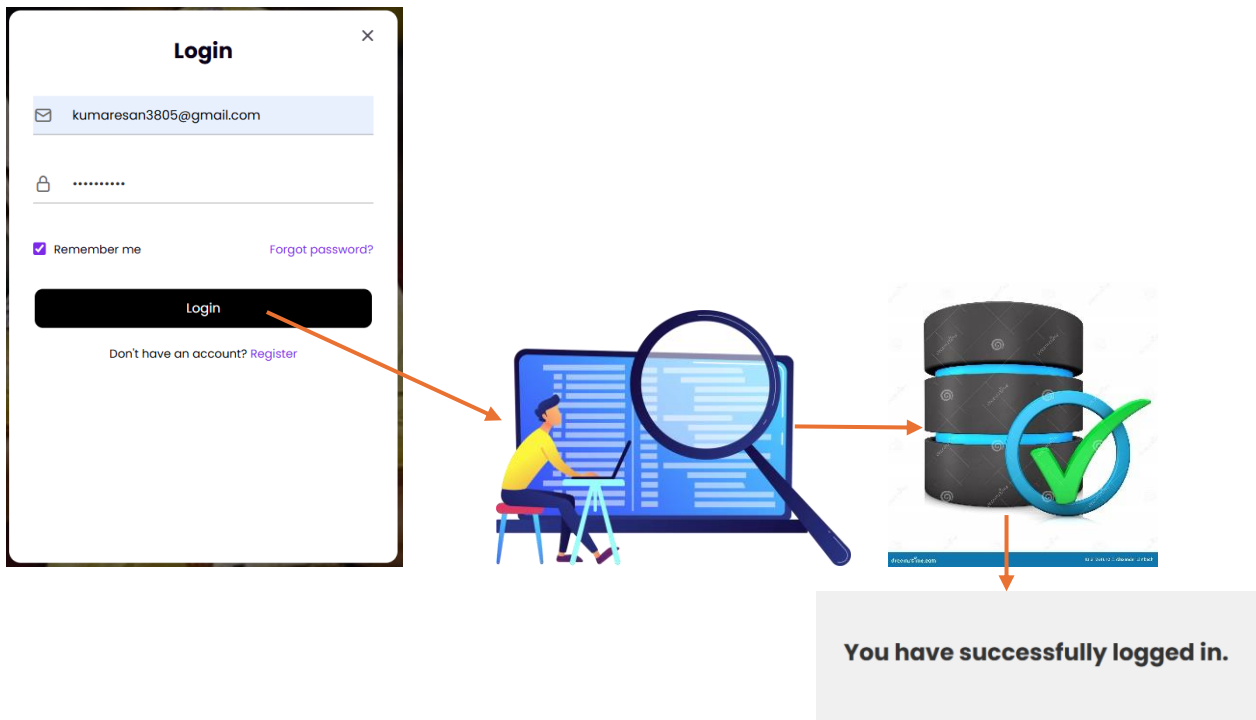
Data Persistence

- UserService processes the data, encodes the password, and uses UserRepository to save the User entity in the MySQL database.

? Database Configuration

- MySQL is configured in application.properties, enabling seamless JPA interaction for storing user data.

Task 3: Implement successful authentication on the login page.



Workflow:

1. User Login Request:

- The user submits username and password via the login form.
- A POST request is sent to the /login endpoint with these credentials.

2. Authentication Logic:

- AuthController handles the request and delegates it to UserService.
- UserService retrieves the user from the database.
- Password validation is performed using PasswordEncoder to compare the stored hash with the input password.

3. Session Management:

- On successful authentication, a server-side session is created.
- A session ID is sent to the client in a secure, HttpOnly cookie to maintain the authenticated state.

4. Error Handling:

- If authentication fails, an appropriate error (e.g., 401 Unauthorized) is returned, prompting the user to retry.

Task 4: Ensure secure access using JWT tokens.



Objective:

To implement secure access by generating and validating JSON Web Tokens (JWT) for user authentication and authorization.

Workflow:

1. User Authentication and Token Generation:

- When a user successfully logs in, the backend authenticates the credentials by verifying the username and password.
- Upon successful authentication, a JWT token is generated using a secret key. This token contains the user's information (e.g., username, roles) as claims.

2. Token Distribution:

- The server sends the JWT token in the response body (typically in the Authorization header) to the client. The client stores this token (usually in local storage or cookies).

3. Secure Access to Protected Resources:

- For subsequent requests to protected resources, the client includes the JWT token in the Authorization header using the Bearer prefix.
- The server extracts the token from the request header, validates it by verifying the signature using the secret key, and checks the token's expiration.

4. Access Control and Authorization:

- If the token is valid, the server grants access to the requested resource and extracts the user's roles for role-based access control.
- If the token is invalid or expired, the server responds with an error (e.g., 401 Unauthorized), requiring the user to authenticate again.

```
String SECRET_KEY = "a7df60475bdacd0fb1969757fb682f71e8c0d3949ccfdc1667585671e1d6312c";
```

Conclusion of weak1 Task:

Milestone 1 focused on implementing user authentication with JWT. Users can register with secure password encryption, and log in to receive a JWT token for secure access to protected resources. Spring

Security was integrated to manage authentication and role-based access, laying the groundwork for future project functionality.