# Low Level Design (LLD) - Login & JWT Authentication

## Revision History

|  |  |  |
| --- | --- | --- |
| Version | Changes | Date |
| 1.1 | Added flowchart and use case diagrams for Signup and Signin. Updated test cases for JWT token validation. | 2025-08-14 |

## Use Case Diagrams

Below are the use case diagrams for Sign-up and Sign-in processes. They show how JWT token is generated, stored, and validated.

### Sign-up Flow

1. User enters username, email, and password.  
2. Backend validates inputs.  
3. Password is hashed.  
4. User data stored in MySQL database.  
5. JWT token is generated using user ID and secret key.  
6. Token is sent back to the frontend.

### Sign-in Flow

1. User provides username and password.  
2. Backend verifies credentials.  
3. If valid, generates JWT token and sends to frontend.  
4. Frontend stores the token (in session/local storage).  
5. For any subsequent API request, token is sent in Authorization header.  
6. Backend verifies token before processing request.

## Test Cases for JWT Token Authentication

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output |
| TC\_JWT\_001 | Access API with valid JWT token | Authorization: Bearer <valid\_token> | API returns 200 OK with expected data |
| TC\_JWT\_002 | Access API with invalid JWT token | Authorization: Bearer <invalid\_token> | API returns 401 Unauthorized |
| TC\_JWT\_003 | Access API without JWT token | No Authorization header | API returns 401 Unauthorized |
| TC\_JWT\_004 | Access API with expired JWT token | Authorization: Bearer <expired\_token> | API returns 401 Unauthorized |