*Florida International University*

*School of Computing and Information Sciences*

Software Engineering Focus

Feature Document

User Story #**199**

**Team Member:**

Gabriel Fernandez

**Product Owner(s)**:

Mohsen Taheri

**Mentor(s)**:

Mohsen Taheri

**Instructor**: Masoud Sadjadi

**User Story** Create login & register backend endpoints

* As a developer, I want to have endpoints for the backend to create new users and to retrieve user authentication credentials, so that the mobile app can support users.

Acceptance Criteria

* Endpoint URLs are accessible and take json requests
* The endpoints return useful information or error messages
* A token is returned when an user is successfully logged in
* Passwords are securely stored

**Use Case** #**199a – Create new users**

**Actors**

Front-end developer

**Entry Conditions**

Front-end developer has access to a terminal or an api to send http requests

Front-end developer has read the documentation and understands which api endpoints to hit and what data to send

**Flow of Events**

1. Use case starts when front-end developer sends an http post request to the user endpoint with the required json-formatted information: username, email, password, profile…
2. The backend receives the request and validates the data received against the database
3. The backend sends a response to the front-end developer saying that the new user was created
4. The backend creates a new user instance and sends a response to the front-end developer with the new user information and the use case ends

**Alternate Flow of Events**

* 3a.
  + The backend sends a response to the front-end developer saying the user already exists and the use case ends
* 4a.
  + The database remains unchanged and the use case ends
* 3b.
  + The backend sends a response to the front-end developer saying the information given is incorrect and the use case ends
* 4b.
  + The database remains unchanged and the use case ends

**Use Case** #**199b – Authenticate users**

**Actors**

Front-end developer

**Entry Conditions**

Front-end developer has access to a terminal or an api to send http requests

Front-end developer has read the documentation and understands which api endpoints to hit and what data to send

**Flow of Events**

1. Use case starts when front-end developer sends an http post request to the user endpoint with the required json-formatted information: username, password
2. The backend receives the request and validates the data received against the database
3. The data received matches the data in the database: The backend creates a token and returns it to the front-end developer as a response, and the use case ends

**Alternate Flow of Events**

* 3a.
  + The data received does not match the data in the database: The backend sends the front-end developer a response saying that the information provided is incorrect, and the use case ends

**Use Case Diagram**



**Sequence Diagram**

Authenticate Users



Authenticate Users



**Class Diagram**



**Unit Test**

**Test Case 1 (Sunny Day)**

**Purpose**

* Ensure login backend endpoint works correctly

**Precondition**

* Http request is made to the server, using the login URL

**Input**

* '{"username":"admin","password":"password123"}'

**Expected Result**

* Token

**Actual Result**

* Token

**Test Case 2 (Sunny Day)**

**Purpose**

* Ensure register backend endpoint works correctly

**Precondition**

* Http request is made to the server, using the register URL

**Input**

* '{"username":"newuser","email":"email@gmail.com","password":"password123","profile":{"grade":"12","age":"23","gender":"male"}}'

**Expected Result**

* User created

**Actual Result**

* User created

**Test Case 3 (Rainy Day)**

**Purpose**

* Ensure login backend endpoint errors work correctly when given incorrect information

**Precondition**

* Http request is made to the server, using the login URL

**Input**

* '{"username":"admin2","password":"password123"}'

**Expected Result**

* Error; User does not exist.

**Actual Result**

* Error; User does not exist.

**Test Case 4 (Rainy Day)**

**Purpose**

* Ensure register backend endpoint errors work correctly when given incorrect information

**Precondition**

* Http request is made to the server, using the register URL

**Input**

* '{"username":"newuser","email":"email@gmail.com","password":"password123","userInfo":{"grade":"12","age":"23","gender":"male"}}'

**Expected Result**

* Error; User already exists.

**Actual Result**

* Error; User already exists.

**Visual User Guide**

