*Florida International University*

*School of Computing and Information Sciences*

Software Engineering Focus

Feature Document

User Story #**229**

**Team Member:**

Gabriel Fernandez

**Product Owner(s)**:

Mohsen Taheri

**Mentor(s)**:

Mohsen Taheri

**Instructor**: Masoud Sadjadi

**User Story** Create endpoints for returning data to front-end

* As a front-end developer, I want to be able to fetch specific information from the database, so that I can provide a demo of our connected application

**Acceptance Criteria**

* All models in the database are created and updated to mirror the information to be stored and fetched
* All required endpoints are provided to fetch and update database models through http requests

**Use Case** #**229a – Fetch User Assignment Progress**

**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

Front-end developer has the id of the assignment

**Flow of Events**

1. Use case starts when front-end developer sends an http request to the assignment progress endpoint with a user token as a header and the id of the assignment
2. The backend authenticates the user
3. The backend queries the database for the assignment
4. The backend queries the database for the user’s tasks in the assignment
5. The backend calculates the user’s task completion percentage and returns the data to front-end developer as a response, and the use case ends

**Alternate Flow of Events**

* 2a.
  + The backend determines the token does not belong to an existing user and the use case ends.
* 3a.
  + The backend determines the assignment does not exist or does not belong to the user, and returns an error to front-end developer as a response. The use case end.

**Use Case** #**229b – Fetch User Info**

**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

Front-end developer has a user token

**Flow of Events**

1. Use case starts when front-end developer sends an http request to the user endpoint with a user token as a header
2. The backend authenticates the user
3. The backend queries for the user’s profile
4. The backend returns the user information to front-end developer as a response, and the use case ends

**Alternate Flow of Events**

* 2a.
  + The backend determines the token does not belong to an existing user and the use case ends.

**Use Case** #**229c – Add User To Assignment**

**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

Front-end developer has the id of the assignment and the user

**Flow of Events**

1. Use case starts when front-end developer sends an http request to the assignment roster endpoint with a user token as a header, the id of the assignment, and the id of the user
2. The backend authenticates the user
3. The backend queries the database for the assignment
4. The backend queries the database for the user
5. The backend adds the user and the assignment to the assignment roster model
6. The backend returns the assignment roster information to front-end developer as a response, and the use case ends

**Alternate Flow of Events**

* 2a.
  + The backend determines the token does not belong to an existing user and the use case ends.
* 3a.
  + The backend determines the assignment does not exist and returns an error to front-end developer as a response. The use case end.
* 4a.
  + The backend determines the user does not exist and returns an error to front-end developer as a response. The use case end.

**Use Case Diagram**



**Sequence Diagrams**

#**229a – Fetch User Assignment Progress**



#**229b – Fetch User Info**



#**229c – Add User to Assignment**



**Class Diagram**



**Unit Test**

**Test Case 229a-1 (Sunny Day)**

**Purpose**

* Ensure backend returns the user’s percentage per assignment

**Precondition**

* Http request is made to the server at the assignment progress endpoint

**Input**

* User token, assignment pk, user pk

**Expected Result**

* The user’s completion percentage for the assignment

**Actual Result**

* The user’s completion percentage for the assignment

**Test Case 229a-2 (Rainy Day)**

**Purpose**

* Ensure backend returns a proper error message when the assignment doesn’t exist

**Precondition**

* Http request is made to the server at the assignment progress endpoint

**Input**

* User token, assignment pk, user pk

**Expected Result**

* Error message saying the assignment given doesn’t exist

**Actual Result**

* Error message saying the assignment given doesn’t exist

**Test Case 229b-1 (Sunny Day)**

**Purpose**

* Ensure backend returns the user info by token

**Precondition**

* Http request is made to the server at the user endpoint

**Input**

* User token

**Expected Result**

* The token’s user information

**Actual Result**

* The token’s user information

**Test Case 229b-2 (Rainy Day)**

**Purpose**

* Ensure backend returns no information with the wrong token

**Precondition**

* Http request is made to the server at the user endpoint

**Input**

* Invalid user token

**Expected Result**

* An empty array

**Actual Result**

* An empty array

**Test Case 229c-1 (Sunny Day)**

**Purpose**

* Ensure backend can enroll a user in an assignment

**Precondition**

* Http request is made to the server at the assignment roster endpoint

**Input**

* User token, user url, assignment, url

**Expected Result**

* User enrolled & assignment roster data

**Actual Result**

* User enrolled & assignment roster data

**Test Case 229c-2 (Rainy Day)**

**Purpose**

* Ensure backend returns a proper error when the user is already enrolled

**Precondition**

* Http request is made to the server at the assignment roster endpoint

**Input**

* User token, user url, assignment, url

**Expected Result**

* The fields assignment, user must make a unique set

**Actual Result**

* The fields assignment, user must make a unique set

**Visual User Guide**





