**Lorem Ipsum – Assignment 2:**

**Student Conduct Tracker**

Skyler Almarales

Student ID: 816034038

GitHub Username: Zurun-Rikuthin

**–**

Kadie Bisnath

Student ID:816036090

GitHub Username: KeraGold

**–**

Jaheim Caesar

Student ID:816035438

GitHub Username: TheLegend1011101

**–**

Ronaldo Walker:

Student ID: 816036438

GitHub Username: ronaldowalker

St. Augustine Campus, University of the West Indies

COMP 3613: Software Engineering (Semester I – 2024/2025)

Mr. Nicholas Mendez

October 18, 2024

# 

[**Links**](#_c4hjr3pbiiad) **3**

[**Feature List**](#_jpin5jpz3e04) **3**

[**Use Case Diagram (Updated)**](#_m3nm29r8h8dg) **4**

[**Model Diagram (Updated)**](#_atfw8vuz3nya) **5**

[**Wireframes**](#_rgj8zl3lqh9g) **6**

[**Test Plans**](#_4xamvep7sdwj) **7**

[User Acceptance Tests](#_lzrleui6prmx) 7

[Account Creation](#_j6w8tevc167c) 7

[Login](#_ci5xp7lx0w5f) 9

[Logout](#_89nra733zguh) 10

[Add Student Profile](#_okwy9mysfpni) 11

[Search Student by ID](#_7uaen1410vio) 13

[Search Student by Filter](#_1dnsuolbbhse) 14

[View Student Profile](#_ym8ol12mcbnb) 15

[Review Student](#_7i4fe9mxmy18) 16

[Search Reviews](#_l2we6pb75vfv) 17

[List All Student Reviews](#_qggc5zx6k48q) 18

[View Specific Student Review](#_c6dwhjbikfjq) 19

[Unit Tests](#_17i7gsw8psew) 20

[Integration Tests](#_hnxy4rqoqlyd) 21

[User Operations](#_m59oq3pk14ny) 21

[Student Operations](#_gdx1knz0xttd) 21

[Review Operations](#_8fglc1dqootx) 22

[API Tests](#_5ch123qs8wqc) 23

[User Account Creation](#_j8cryaixtcsn) 23

[User Login](#_2xvzm82pm6dt) 24

[User Logout](#_2f62t2ljjlj6) 25

[Add Student Profile](#_939cjt9wh52d) 25

[Search Student by ID](#_5sz23x9q084u) 27

# 

# Links

GitHub Repo:

* https://github.com/Lorem-Ipsum-Scholars/COMP-3613-Assignment-2

# Feature List

Most Critical Features:

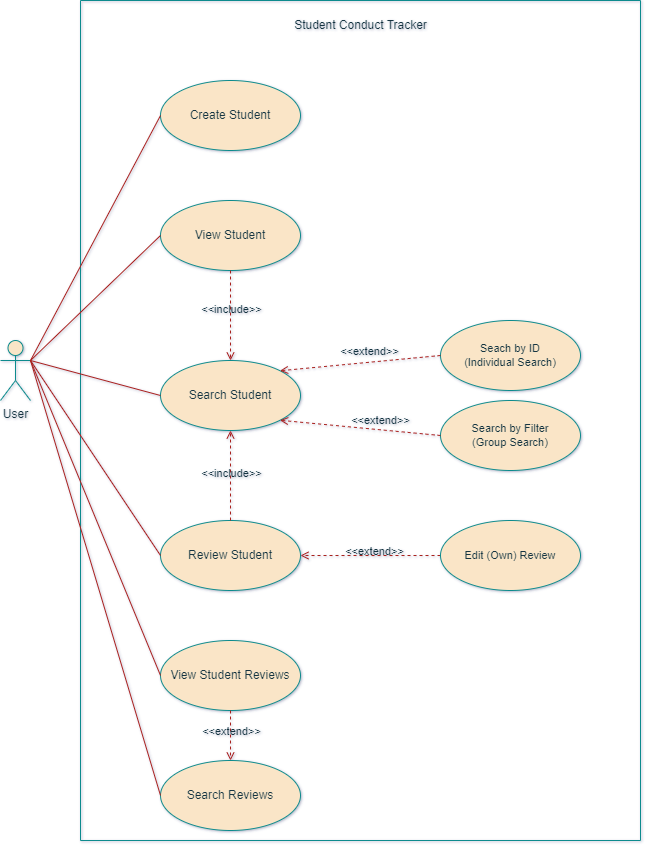
* Add Student
* View Student
* Review Student
* View Student Reviews

Sub-Features:

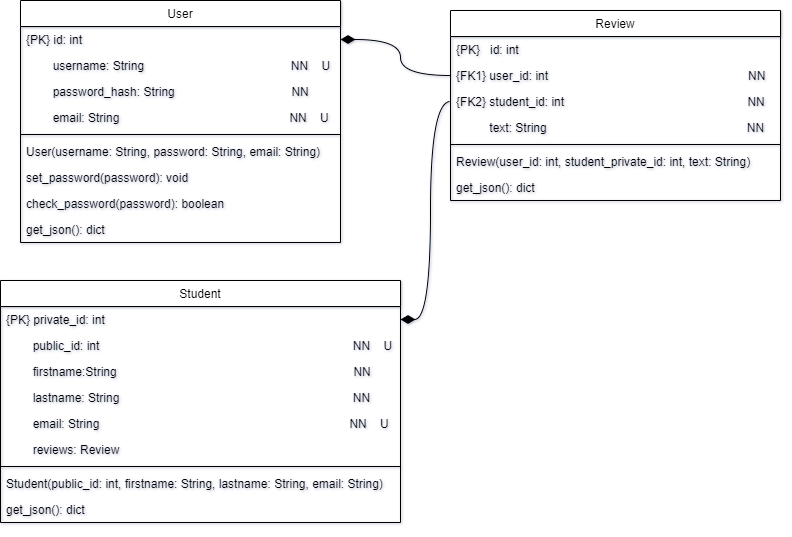
* Add Student
  + Edit Student
* View Student
  + Search Student
    - Search by ID
    - Filter search
  + View the profile of a specific student
  + List all students in the database
* Review Student
  + Edit Review
* View Student Reviews
  + List all reviews in the database
    - Sort reviews
  + List all reviews for an individual student
    - (Also falls under *View Student* as a given student’s reviews can be listed directly under their profile.)
  + View a specific student review from the list

(Assume an authentication page as the landing page for logged out users.)

# Use Case Diagram (Updated)



# Model Diagram (Updated)



# Wireframes

e

# Test Plans

## User Acceptance Tests

### Account Creation

| **Test Case** | **Sub-Case** | **Pre-conditions** | **Test Steps** | **Test Criteria** |
| --- | --- | --- | --- | --- |
| **Test Account Creation** | ***Successful*** | * None | * Access the authentication page via URL or redirect. * Fill in the sign-up form’s required fields, including a unique username and email. * Submit the form. | * Alert the user to a successful account creation. * Redirect the user back to the authentication page to login to their new account. |
| ***Missing required fields*** | * None | * Access the authentication page via URL or redirect. * Partially fill in the sign-up form’s required fields. * Submit the form. | * Alert the user to a failed sign up due to incomplete data. * Redirect the user back to the authentication page to try again. |
| ***Invalid username format*** | * None | * Access the authentication page via URL or redirect. * Use a non-alphanumeric username for the sign-up form. * Submit the form. | * Alert the user to a failed sign up due to an invalid username. * Redirect the user back to the authentication page to try again. |
| ***Invalid email format*** | * None | * Access the authentication page via URL or redirect. * Use an invalid email address for the sign-up form. * Submit the form. | * Alert the user to a failed sign up due to an invalid email address. * Redirect the user back to the authentication page to try again. |
| ***Duplicate username*** | * A valid user profile that shares the target username must already exist. | * Access the authentication page via URL or redirect. * Use a pre-existing username for the sign-up form. * Submit the form. | * Alert the user to a failed sign up due to a duplicate username. * Redirect the user back to the authentication page to try again. |
| ***Duplicate email address*** | * A valid user profile that shares the target email address must already exist. | * Access the authentication page via URL or redirect. * Use a pre-existing email for the sign-up form. * Submit the form. | * Alert the user to a failed sign up due to a duplicate email. * Redirect the user back to the authentication page to try again. |

### 

### Login

| **Test Case** | **Sub-Case** | **Pre-conditions** | **Test Steps** | **Test Criteria** |
| --- | --- | --- | --- | --- |
| **Test Login** | ***Successful*** | * A valid user profile must already exist. | * Access the authentication page via URL or redirect. * Fill in the required fields for the login form. * Submit the form. | * Alert the user to a successful login. * Navbar now displays user profile and logout buttons. * Redirect the user to the home page. |
| ***Missing required fields*** | * A valid user profile must already exist. | * Access the authentication page via URL or redirect. * Partially fill in the form’s required fields. * Submit the form. | * Alert the user to a failed login due to incomplete data. * Redirect the user back to the authentication page to try again. |
| ***Invalid username or password*** | * A valid user profile must already exist. | * Access the authentication page via URL or redirect. * Use an invalid username/password for the login form. * Submit the form. | * Alert the user to a failed login due to invalid data. * Redirect the user back to the authentication page to try again. |
| ***User already logged in*** | * The user is currently logged in to their account. | * Attempt to access the authentication page via URL. | * Redirect the user to the home page. |

### Logout

| **Test Case** | **Sub-Case** | **Pre-conditions** | **Test Steps** | **Test Criteria** |
| --- | --- | --- | --- | --- |
| **Test Logout** | ***Successful*** | * The user is currently logged in to their account. | * Press the logout button in the navbar. | * Alert the user to a successful logout. * Navbar no longer displays user profile and logout buttons. * Redirect the user to the authentication page. |
| ***Still logged in*** | * The user is currently logged in to their account. | * Press the logout button in the navbar. | * Alert the user to a failed logout. * Redirect the user to the home page.   ***(I’m not exactly sure how to handle this…)*** |
| ***User not logged in*** | * The user is ***not*** currently logged in to their account. | * Attempt to access the logout page via URL. | * Redirect the user to the authentication page to log in. |

### 

### Add Student Profile

| **Test Case** | **Sub-Case** | **Pre-conditions** | **Test Steps** | **Test Criteria** |
| --- | --- | --- | --- | --- |
| **Test Add Student Profile** | ***Successful*** | * The user is currently logged in to their account. | * Access the student record creation page via the navbar or URL. * Fill in the form’s required fields including both a unique student ID and email. * Submit the form. | * Alert the user to a successful student record creation. * Redirect the user to the new student’s profile page. |
| ***User not logged in*** | * The user is ***not*** currently logged in to their account. | * Attempt to access the student record creation page via URL. | * Redirect the user to the authentication page to log in. |
| ***Missing required fields*** | * The user is currently logged in to their account. | * Access the student record creation page via the navbar or URL. * Partially fill in the form’s required fields. * Submit the form. | * Alert the user to a failed student record creation due to incomplete data. * Redirect the user back to the student record creation page. |
| ***Invalid student ID format*** | * The user is currently logged in to their account. | * Access the student record creation page via the navbar or URL. * Use an invalid student ID for the form. * Submit the form. | * Alert the user to a failed student record creation due to an invalid student ID. * Redirect the user back to the student record creation page. |
| ***Invalid name format*** | * The user is currently logged in to their account. | * Access the student record creation page via the navbar or URL. * Use a non-alphabetic name for the form. * Submit the form. | * Alert the user to a failed student record creation due to an invalid name. * Redirect the user back to the student record creation page. |
| ***Duplicate student ID*** | * The user is currently logged in to their account * A valid user profile that shares the target username must already exist. | * Access the student record creation page via the navbar or URL. * Use a pre-existing student ID for the form. * Submit the form. | * Alert the user to a failed student record creation due to an invalid student ID. * Redirect the user back to the student record creation page. |
| ***Duplicate email*** | * The user is currently logged in to their account * A valid user profile that shares the target email address must already exist. | * Access the student record creation page via the navbar or URL. * Use a pre-existing email for the form. * Submit the form. | * Alert the user to a failed student record creation due to an invalid email address. * Redirect the user back to the student record creation page. |

### 

### Search Student by ID

| **Test Case** | **Sub-Case** | **Pre-conditions** | **Test Steps** | **Test Criteria** |
| --- | --- | --- | --- | --- |
| **Test Search Student by ID** | ***Successful - Empty Search*** | * The user is currently logged in to their account. | * Access the student search page via the navbar or URL. * Submit the form without providing a student ID. | * Redirect the user to the student search result page listing ***all*** students in the database. |
| ***Successful - Student not found*** | * The user is currently logged in to their account. * The target student profile does ***not*** exist. | * Access the student search page via the navbar or URL. * Fill in the ID search form with a valid student ID. * Submit the form. | * Redirect the user to the student search result page stating there were no results found. |
| ***Successful - Student found*** | * The user is currently logged in to their account. * The target student profile must already exist. | * Access the student search page via the navbar or URL. * Fill in the ID search form with a valid student ID. * Submit the form. | * Redirect the user to the student search result page listing the target profile. |
| ***User not logged in*** | * The user is ***not*** currently logged in to their account. | * Attempt to access the student search/search results page via URL. | * Redirect the user to the authentication page to log in. |
| ***Invalid student ID format*** | * The user is currently logged in to their account. | * Access the student record creation page via the navbar or URL. * Use an invalid student ID for the form. * Submit the form. | * Alert the user to a failed student record creation due to an invalid student ID. * Redirect the user back to the student search page. |

### 

### Search Student by Filter

| **Test Case** | **Sub-Case** | **Pre-conditions** | **Test Steps** | **Test Criteria** |
| --- | --- | --- | --- | --- |
| **Test Search Student by Filter** | ***Successful - Empty Search*** | * The user is currently logged in to their account. | * Access the student search page via the navbar or URL. * Submit the form without providing any filters. | * Redirect the user to the student search result page listing ***all*** students in the database. |
| ***Successful - Student found*** | * The user is currently logged in to their account. * At least one student profile matching the provided filters must already exist. | * Access the student search page via the navbar or URL. * Fill in the desired fields of the filter search form. * Submit the form. | * Redirect the user to the student search result page listing all student profiles that match the provided filters. |
| ***Successful - Student not found*** | * The user is currently logged in to their account. * No student profiles matching the provided filters currently exist. | * Access the student search page via the navbar or URL. * Fill in the desired fields of the filter search form. * Submit the form. | * Redirect the user to the student search result page stating there were no results found. |
| ***User not logged in*** | * The user is ***not*** currently logged in to their account. | * Attempt to access the student search/search results page via URL. | * Redirect the user to the authentication page to log in. |
| ***Invalid student ID format*** | * The user is currently logged in to their account. | * Access the student record creation page via the navbar or URL. * Use an invalid student ID for the form. * Submit the form. | * Alert the user to a failed student record creation due to an invalid student ID. * Redirect the user back to the student search page. |
| ***Invalid name format*** | * The user is currently logged in to their account. | * Access the student record creation page via the navbar or URL. * Use a non-alphabetic name for the form. * Submit the form. | * Alert the user to a failed student record creation due to an invalid name. * Redirect the user back to the student search page. |

### 

### View Student Profile

| **Test Case** | **Sub-Case** | **Pre-conditions** | **Test Steps** | **Test Criteria** |
| --- | --- | --- | --- | --- |
| **Test View Student Profile** | ***Successful*** | * The user is currently logged in to their account. * The target student profile must already exist. | * Access a student’s profile (e.g., from search results, student listings, URL). | * Redirect the user to the target student’s profile page. |
| ***User not logged in*** | * The user is ***not*** currently logged in to their account. | * Attempt to access a student’s profile page (e.g., from search results, student listings, URL). | * Redirect the user to the authentication page to log in. |
| ***Student not found*** | * The user is currently logged in to their account. | * Attempt to directly access a non-existent student profile page via URL. | * Redirect the user to the “404 - Not Found” page. |

### Review Student

| **Test Case** | **Sub-Case** | **Pre-conditions** | **Test Steps** | **Test Criteria** |
| --- | --- | --- | --- | --- |
| **Test Review Student** | ***Successful*** | * The user is currently logged in to their account. * A valid student profile must already exist. | * Access the student review creation page via the given student’s profile or URL. * Fully complete the form with valid data (including both a unique student ID and email). * Submit the form. | * Alert the user to a successful review creation. * Redirect the user to the new review’s page. |
| ***User not logged in*** | * The user is ***not*** currently logged in to their account. | * Attempt to access the student review creation page via URL. | * Redirect the user to the authentication page to log in. |
| ***Missing required fields*** | * The user is currently logged in to their account. | * Access the student record creation page via the navbar or URL. * Partially fill in the form’s required fields. * Submit the form. | * Alert the user to a failed student record creation due to incomplete data. * Redirect the user back to the student record creation page. |
| ***Student not found*** | * The user is currently logged in to their account. | * Attempt to access the review page for a non-existent student via URL. | * Redirect the user to the “404 - Not Found” page. |

### 

### Search Reviews

| **Test Case** | **Sub-Case** | **Pre-conditions** | **Test Steps** | **Test Criteria** |
| --- | --- | --- | --- | --- |
| **Test Search Reviews** | ***Successful -***  ***Empty Search*** | * The user is currently logged in to their account. | * Access the review search page from the navbar or via URL. * Submit the form without providing any filters. | * Redirect the user to the reviews list page, listing all reviews in the database. |
| ***Successful -***  ***(At least one review has been made)*** | * The user is currently logged in to their account. * At least one (1) valid student profile must already exist. * At least one (1) valid student review for any given student must already exist. | * Access the review search page from the navbar or via URL. * Fill in the desired fields of the filter search form. * Submit the form. | * Redirect the user to the reviews list page, listing all reviews in the database that match the given filters. |
| ***Successful -***  ***(No reviews have been made)*** | * The user is currently logged in to their account. * No valid student reviews have yet been made. | * Access the review search page from the navbar or via URL. * Fill in the desired fields of the filter search form. * Submit the form. | * Redirect the user to the reviews list page, with a message saying no reviews currently exist. |
| ***User not logged in*** | * The user is ***not*** currently logged in to their account. | * Attempt to access the reviews search/search results page via URL. | * Redirect the user to the authentication page to log in. |

### List All Student Reviews

| **Test Case** | **Sub-Case** | **Pre-conditions** | **Test Steps** | **Test Criteria** |
| --- | --- | --- | --- | --- |
| **Test List All Student Reviews** | ***Successful -***  ***(At least one review has been made)*** | * The user is currently logged in to their account. * A valid student profile must already exist. * A valid review for the given student must already exist. | * Perform a filter search for reviews of a specific student.   + Can be done manually via the review search page or automatically via the given student’s profile page. | * Redirect the user to the reviews list page, filtering reviews to only those for the given student. |
| ***Successful -***  ***(No reviews have been made)*** | * The user is currently logged in to their account. * A valid student profile must already exist. * The target student has no reviews yet. | * Perform a filter search for reviews of a specific student.   + Can be done manually via the review search page or automatically via the given student’s profile page. | * Redirect the user to the reviews listing page, with a message saying no reviews currently exist with the current filter. |
| ***User not logged in*** | * The user is ***not*** currently logged in to their account. | * Attempt to access the review search/search results page via URL. | * Redirect the user to the authentication page to log in. |
| ***Student not found*** | * The user is currently logged in to their account. | * Attempt to access the review page for a non-existent student via URL. | * Redirect the user to the “404 - Not Found” page. |

### View Specific Student Review

| **Test Case** | **Sub-Case** | **Pre-conditions** | **Test Steps** | **Test Criteria** |
| --- | --- | --- | --- | --- |
| **Test View Specific Student Review** | ***Successful*** | * The user is currently logged in to their account. * A valid student profile must already exist. * A valid review for the given student must already exist. | * Access a specific student review from the student’s profile, search results, or via URL. | * Redirect the user to review for a given student. |
| ***User not logged in*** | * The user is ***not*** currently logged in to their account. | * Attempt to access the review page via URL. | * Redirect the user to the authentication page to log in. |
| ***Review not found*** | * The user is currently logged in to their account. | * Attempt to access a non-existent review page via URL. | * Redirect the user to the “404 - Not Found” page. |

# 

## Unit Tests

| **Unit Test** | **Module Tested** | **Expected Output** |
| --- | --- | --- |
| **test\_create\_user** | User(“bob”, “bobpass”, bob.smith@mail.com) | * user.username == “bob” * user. email == “bob.smith@mail.com” |
| **test\_user\_check\_password** | User.check\_password(“bobpass”) | * user.check\_password(“bobpass”) == True * user.check\_password(“FAKEpass”) == False |
| **test\_user\_set\_password** | User.set\_password(“newpass”) | user.password\_hash != generate\_password\_hash(“bobpass”) |
| **test\_user\_to\_json()** | User.to\_json() | {  “id”: 1,  “username”: “bob”  “email”: “bob.smith@mail.com”  } |
| **test\_create\_student()** | Student(816012345, “jane”, “smith”, jane@mail.com) | * student.public\_id == 816012345 * student.firstname == “jane” * student.lastname == “smith” * student.email == “jane@mail.com” |
| **test\_ student\_to\_ json()** | Student.to\_json() | {  “private\_id”: 1,  “public\_id”: 816012345  “firstname”: “jane”,  “lastname”: “smith”,  “email”: “jane@mail.com”  “reviews”: []  } |
| **test\_create\_review()** | Review(1, 1, “Test Review”, 1) | * review.user\_id == 1 * review.student\_private\_id == 1 * review.text == “Test Review” |
| **test\_review\_to\_json()** | Review.to\_json() | {  user\_id: 1,  student\_private\_id: 1,  'text': "Test Review"  } |

## Integration Tests

### User Operations

| **Test** | **Dependencies** | **Description** |
| --- | --- | --- |
| **test\_create\_user** | * create\_user() * get\_user() | Creates a user record in the database and ensures it has the correct values. |
| **test\_get\_user()** | * create\_user() * get\_user\_by\_id() | Ensures that a user record with a specific ID can be successfully retrieved from the database. |
| **test\_get\_user\_by\_username()** | * create\_user() * get\_user\_by\_username() | Ensures that a user record *with a specific username* can be successfully retrieved from the database. |
| **test\_user\_to\_json()** | * create\_user() * get\_user\_json()   + User.to\_json() | Ensures that the data of a given user record can be properly converted to JSON format. |
| **test\_get\_all\_users()** | * create\_user() * get\_all\_users() | Ensures that the data of all user records in the database can be retrieved. |
| **test\_get\_all\_users\_json()** | * create\_user() * get\_all\_users\_json()   + User.to\_json() | Ensures that the data of all user records in the database can be properly converted to JSON format. |

### Student Operations

| **Test** | **Dependencies** | **Description** |
| --- | --- | --- |
| **test\_create\_student()** | * create\_user() * create\_student() | * public\_id == 816012345 * student.firstname == “jane” * student.lastname == “smith” * student.email == “jane@mail.com” |
| **test\_ student\_to\_ json()** | * get\_student\_json()   + Student.to\_json() | {  “private\_id”: 1,  “public\_id”: 816012345  “firstname”: “jane”,  “lastname”: “smith”,  “email”: “jane@mail.com”  “reviews”: []  } |
| **test\_search\_student\_no\_parameters()** | * create\_student() * search\_student() | Verifies that searching with no parameters returns all students in the system. |
| **test\_search\_student\_by \_public\_id()** | * create\_student() * search\_student() | Ensures that searching by public ID returns the correct student. |
| **test\_search\_student\_by\_public\_id\_no\_results()** | * create\_student() * search\_student() | Verifies that when there are no ID matches, no results are returned. |
| **test\_search\_student\_by\_first\_name\_only()** | * create\_student() * search\_student() | Verifies that searching with only the first name returns all matching students, ignoring the last name. |
| **test\_search\_student\_by\_last\_name\_only()** | * create\_student() * search\_student() | Ensures that searching with only the last name returns all matching students, ignoring the first name. |
| **test\_search\_student\_by\_first\_and\_last\_name()** | * create\_student() * search\_student() | Ensures that searching with both first and last name returns the correct student(s). |
| **test\_search\_student\_partial\_first\_and\_last\_name()** | * create\_student() * search\_student() | Ensures that searching with partial first and last names returns all relevant matching students. |
| **test\_search\_student\_no\_results\_for\_either\_name()** | * create\_student() * search\_student() | Verifies that when neither the first nor last name matches, no results are returned. |

### Review Operations

| **Test** | **Dependencies** | **Description** |
| --- | --- | --- |
| **test\_create\_review** | * create\_user() * create\_student() * get\_student() * create\_review() * get\_review() | Ensure the review record in the database has the correct values. |
| **test\_review\_to\_json()** | * get\_review\_json()   + Review.to\_json() | {  user\_id: 1,  student\_private\_id: 1,  'text': "Test Review"  } |

## 

## API Tests

### User Account Creation

| **Test Case** | **Method** | **URL** | **Request Headers, Body** | **Response Body, Status** |
| --- | --- | --- | --- | --- |
| **Test Account Creation: *Successful*** | POST | /auth | **Body:**  {  “username”: “bob”,  “password”: “bobpass”,  “confirm\_password”: “bobpass”,  “email”: “bob.smith@mail.com”  } | **Body:**  {  “message”: “account created”  }, 201  **Redirect:**  User is redirected to the **/home** page. |
| **Test Account Creation: *Missing Required Fields*** | POST | /auth | **Body:**  {  “username”: “bob”,  “password”: “”,  “confirm\_password”: “bobpass”,  “email”: “bob.smith@mail.com”  } | **Body:**  {  “error”: “missing required fields”  }, 400  **Redirect:**  User is redirected back to the **/auth** page. |
| **Test Account Creation: *Invalid Username*** | POST | /auth | **Body:**  {  “username”: “b%-21f{{}ob}”,  “password”: “bobpass”,  “confirm\_password”: “bobpass”,  “email”: “bob.smith@mail.com”  } | **Body:**  {  “error”: “invalid username”  }, 400  **Redirect:**  User is redirected back to the **/auth** page. |
| **Test Account Creation: *Invalid Email*** | POST | /auth | **Body:**  {  “username”: “bob”,  “password”: “bobpass”,  “confirm\_password”: “bobpass”,  “email”: “2bob.mal”  } | **Body:**  {  “error”: “invalid email”  }, 400  **Redirect:**  User is redirected back to the **/auth** page. |
| **Test Account Creation: *Duplicate Username*** | POST | /auth | **Pre-condition:** A valid user with the target username exists.  **Body:**  {  “username”: “bob”,  “password”: “bobpass”,  “confirm\_password”: “bobpass”,  “email”: “bob.smith@mail.com”  } | **Body:**  {  “error”: “username already taken”  }, 400  **Redirect:**  User is redirected back to the **/auth** page. |
| **Test Account Creation: *Duplicate Email*** | POST | /auth | **Pre-condition:** A valid user with the target email exists.  **Body:**  {  “username”: “jane”,  “password”: “janepass”,  “confirm\_password”: “janepass”,  “email”: “bob.smith@mail.com”  } | **Body:**  {  “error”: “email already taken”  }, 400  **Redirect:**  User is redirected back to the **/auth** page. |

### User Login

| **Test Case** | **Method** | **URL** | **Request Headers, Body** | **Response Body, Status** |
| --- | --- | --- | --- | --- |
| **Test Login: *Successful*** | POST | /auth | **Pre-condition:** A valid user exists.  **Body:**  {  “username”: “bob”,  “password”: “bobpass”  } | **Body:**  {  “message”:”login successful”,  “access\_token”: <jwt\_token>  }, 200  **Redirect:**  User is redirected to the **/home** page. |
| **Test Login: *Missing Required Fields*** | POST | /auth | **Pre-condition:** A valid user exists.  **Body:**  {  “username”: “”,  “password”: <string>  } | **Body:**  {  “error”: “missing required fields”  }, 400  **Redirect:**  User is redirected back to the **/auth** page. |
| **Test Login: *Invalid Username or Password*** | POST | /auth | **Pre-condition:** A valid user exists.  **Body:**  {  “username”: bob,  “password”: janepass  } | **Body:**  {  “error”: “invalid credentials”  }, 401  **Redirect:**  User is redirected back to the **/auth** page. |
| **Test Login: *User Already Logged In*** | GET | /auth | **Pre-condition:** User is already logged in.  **Headers:**  Authorization: Bearer <jwt\_token> | **Redirect:**  User is redirected to the **/home** page with status 302. |

### User Logout

| **Test Case** | **Method** | **URL** | **Request Headers, Body** | **Response Body, Status** |
| --- | --- | --- | --- | --- |
| **Test Logout: *Successful*** | POST | /auth/logout | **Pre-condition:** User is logged in.  **Headers:**  Authorization: Bearer <jwt\_token> | **Body:**  {  “message”:”login successful”,  “access\_token”: <jwt\_token>  }, 200  **Redirect:**  User is redirected to the **/auth** page for login. |
| **Test Logout: *Still Logged In*** | POST | /auth/logout | **Pre-condition:** User is logged in but the server fails to log them out.  **Headers:**  Authorization: Bearer <jwt\_token> | **Body:**  {  “error”: “logout failed”  }, 500  **Redirect:**  User is redirected back to the **/home** page. |

### Add Student Profile

| **Test Case** | **Method** | **URL** | **Request Headers, Body** | **Response Body, Status** |
| --- | --- | --- | --- | --- |
| **Test Add Student: *Successful*** | POST | /students/add | **Pre-condition:** User is logged in.  **Headers:**  Authorization: Bearer <jwt\_token>  **Body:**  {  “public\_id”: 816012345,  “firstname”: “jane”,  “lastname”: “smith”  “jane  } | **Body:**  {  “message”: “student created”  }, 201  **Redirect:**  User is redirected to the new student’s **/students/<public\_id>** page. |
| **Test Add Student: *Missing Required Fields*** | POST | /students/add | **Pre-condition:** User is logged in.  **Headers:**  Authorization: Bearer <jwt\_token>  **Body:**  {  “public\_id”: 816012345,  “firstname”: “”,  “lastname”: “smith”  } | **Body:**  {  “error”: “missing required fields”  }, 400  **Redirect:**  User is redirected back to the **/students/add** page. |
| **Test Add Student: *Invalid Student ID Format*** | POST | /students/add | **Pre-condition:** User is logged in.  **Headers:**  Authorization: Bearer <jwt\_token>  **Body:**  {  “public\_id”: “bad\_id”,  “firstname”: “jane”,  “lastname”: “smith”,  “email”: “jane.smith@mail.com”  } | **Body:**  {  “error”: “invalid student id format”  }, 400  **Redirect:**  User is redirected back to the **/auth** page. |
| **Test Add Student: *Invalid Name Format*** | POST | /students/add | **Pre-condition:** User is logged in.  **Headers:**  Authorization: Bearer <jwt\_token>  **Body:**  {  “public\_id”: 816012345,  “firstname”: “14jane”,  “lastname”: “s}mi\_+th”,  “email”: “jane.smith@mail.com”  } | **Body:**  {  “error”: “invalid name format”  }, 400  **Redirect:**  User is redirected back to the **/auth** page. |
| **Test Add Student: *Duplicate Student ID*** | POST | /students/add | **Pre-condition:** User is logged in.  **Headers:**  Authorization: Bearer <jwt\_token>  **Body:**  {  “public\_id”: 816012345,  “firstname”: “alice”,  “lastname”: “smith”,  “email”: “alice.smith@mail.com”  } | **Body:**  {  “error”: “student id already taken”  }, 400  **Redirect:**  User is redirected back to the **/auth** page. |
| **Test Add Student: *Duplicate Email*** | POST | /students/add | **Pre-condition:** User is logged in.  **Headers:**  Authorization: Bearer <jwt\_token>  **Body:**  {  “public\_id”: 816054321,  “firstname”: “alice”,  “lastname”: “smith”,  “email”: “jane.smith@mail.com”  } | **Body:**  {  “error”: “email already taken”  }, 400  **Redirect:**  User is redirected back to the **/auth** page. |

### Search Student by ID

| **Test Case** | **Method** | **URL** | **Request Headers, Body** | **Response Body, Status** |
| --- | --- | --- | --- | --- |
| **Test Search Student by ID: *Empty Search*** | GET | /students/search | **Pre-condition:** User is logged in; no student ID is provided.  **Headers:**  Authorization: Bearer <jwt\_token> | **SuccessBody:**  (List all students)  **Redirect:**  User is redirected to the **/students/results** page.  (Provide link for each listed student’s full profile page at **/students/profile/<public\_id>**) |
| **Test Search Student by ID: *Student Not Found*** | GET | /students/search | **Pre-condition:** User is logged in.  **Headers:**  Authorization: Bearer <jwt\_token>  **Body:**  {  “public\_id”: 816012345  } | **Body:**  {  “message”: “no results found”  }, 200  **Redirect:**  User is redirected to the **/students/results** page. |
| **Test Search Student by ID: *Student Found*** | GET | /students/add | **Pre-condition:** User is logged in.  **Headers:**  Authorization: Bearer <jwt\_token>  **Body:**  {  “public\_id”: 816012345  } | **Body:**  (List found student with status 200)  **Redirect:**  User is redirected to the **/students/results** page.  (Provide link to full profile page at **/students/profile/<public\_id>**) |
| **Test Search Student by ID: *User Not Logged In*** | GET | /students/add | **Pre-condition:** User is not logged in. | **Body:**  {  “error”: “authentication required”  }, 302  **Redirect:**  User is redirected to the **/auth** page. |
| **Test Search Student by ID: *Invalid Student ID Format*** | GET | /students/add | **Pre-condition:** User is logged in.  **Headers:**  Authorization: Bearer <jwt\_token>  **Body:**  {  “public\_id”: “bad\_id”  } | **Body:**  {  “error”: “invalid student id format”  }, 400  **Redirect:**  User is redirected back to the **/auth** page. |