**Software Requirement Specifications**

**University Project Display Platform**



**Submitted by**

Iqra Fayyaz (1033)

**Submitted to**

Mr. Zaman Ali

**Department of Information Technology**

**Faculty of Computing**

**The Islamia University of Bahawalpur**

**Summary**

This document outlines the specifications for a **University Project Display Website**, designed to streamline the sharing, grading, review and archiving process for student projects within the university. To ensure academic integrity, the platform will integrate the **Turnitin API** for plagiarism detection.

The university currently lacks an online repository for student projects, which limits access to high-quality, completed projects for learning purposes. This website will provide an online platform for students to upload their project documents, description and links to their associated repositories.

Table of Contents

[1. Introduction 3](#_Toc185287138)

[1.1. Purpose 3](#_Toc185287139)

[1.2. Scope 3](#_Toc185287140)

[1.3. Product Perspective 3](#_Toc185287141)

[1.4. User Characteristics 4](#_Toc185287142)

[1.5. Similar apps and systems/Literature Review 4](#_Toc185287143)

[1.6. Proposed Technologies 4](#_Toc185287144)

[2. Requirements 5](#_Toc185287145)

[2.1. Functional Requirements 5](#_Toc185287146)

[2.2. Non Functional Requirements 10](#_Toc185287147)

[2.2.1. Performance Requirements 10](#_Toc185287148)

[2.2.3. Software Quality Attributes 11](#_Toc185287149)

[3. Use Cases and Flow of Processes 12](#_Toc185287150)

[3.1. Use Case 1 13](#_Toc185287151)

[3.2. Use Case 2 13](#_Toc185287152)

[3.3. Use Case 3 14](#_Toc185287153)

[3.4. Use Case 4 15](#_Toc185287154)

[3.5. Use Case 5 17](#_Toc185287155)

[3.6. Use Case 6 18](#_Toc185287156)

[3.7. Use Case 7 19](#_Toc185287157)

[3.8. Use Case 8 20](#_Toc185287158)

[4. References 22](#_Toc185287159)

# **Introduction**

The **University Project Display** is being developed for Islamia University of Bahawalpur, which serves a large community of students and faculty, offering diverse academic programs and opportunities for research and development. Despite its extensive academic resources, the university currently lacks a centralized platform for managing and showcasing student projects.

The proposed system, **University Project Display**, aims to fill this gap by allowing students to upload their projects.

## Purpose

The **University Project Display** aims to completely change how student projects are uploaded, managed, and displayed in the university. It offers an easy-to-use platform where everything happens in one place, making the whole process more efficient and organized compared to the old, manual methods.

## Scope

The **University Project Display** will serve as a platform for students to upload, showcase, and manage their projects, and for tutors to review, grade, and provide feedback. The platform will include features such as project submissions, a description section, links to code repositories, and image uploads. Additionally, the Turnitin API will be integrated to ensure plagiarism checks for all submitted projects. The website will also allow students to search and view projects uploaded by their peers for learning and inspiration.

## Product Perspective

The **University Project Display** will be a standalone web-based platform focusing on project submissions, grading, and feedback. It will provide facility to tutors to check projects for plagiarism.

## User Characteristics

The **University Project Display** will be used by different types of users, each with distinct roles and responsibilities:

* **Students**: The primary users who will upload and manage their project submissions. Students will also have access to a repository of public projects uploaded by peers for inspiration and learning.
* **Tutors**: Tutors will review and evaluate the submitted projects. Tutors will also provide feedback to students through the platform and use the Turnitin API to check for plagiarism.
* **Administrators**: Admins will manage the overall platform, including user accounts, permissions, and system maintenance.
* **System**: The system itself, including the backend infrastructure, will be responsible for user management, file storage, plagiarism detection, and overall platform functionality. It will interact with users through a web interface and external services like the Turnitin API.

## Similar apps and systems/Literature Review

**Devpost**  
**Features:** Devpost is a platform for hackathons and coding projects where users can submit projects, showcase their work, and connect with other developers. It includes project descriptions, media uploads, and project evaluation by judges.  
**Shortcomings**: While Devpost excels in showcasing projects, it lacks direct plagiarism detection and a structured grading system for academic purposes. It is designed for hackathons and open-source communities rather than university-specific project evaluation and feedback.

## Proposed Technologies

The key proposed technologies include:

* Django
* HTML, CSS, and JavaScript
* Turnitin API
* Git & GitHub

# **Requirements**

## Functional Requirements

* + 1. **Sign Up**
* **Name:** FR001
* **Purpose:** To become member of system
* **User(s):** Student, Supervisor, Admin
* **Input:** 
  + *Name:* Actual name of the user
  + *Username:* Must be the username provided by university
  + *Password:* Must be 8 characters long and must include number
* **Output:**
  + User will be the member of system and able to login
    1. **Sign In**
* **Name:** FR002
* **Purpose:** To gain access of the system
* **User(s):** Student, Supervisor, Admin
* **Input:**
  + *Username:* Must be the username provided by university
  + *Password:*Must be the password used during Sign Up
* **Output:**
  + User have access to the system.
    1. **View Projects**
* **Name:** FR003
* **Purpose:** To view projects of students
* **User(s):** Student, Supervisor, Admin
* **Input:**
  + *Search Query:* A keyword or project title to filter projects.
  + *Pagination Controls:* Inputs for navigating between multiple pages of project listings.
* **Output:**
  + A list of projects
    1. **Search or filter projects**
* **Name:** FR004
* **Purpose:** Allow users to search or filter projects based on specific criteria to quickly locate relevant projects.
* **User(s):** Student, Supervisor
* **Input:**
  + *Search Query:* A keyword or phrase to search
  + *Filter Options:*
    - *Category:* Project domain or category
    - *Department:* The department associated with the project
    - *Supervisor*
    - *Year:* Year of submission or completion of the project.
    - *Status:* Current project status
  + *Pagination Controls:* For navigating through filtered or search results.
* **Output:**
  + User have access to the system.
    1. **Upload Projects**
* **Name:** FR005
* **Purpose:** Upload project to the system and check for plagiarism
* **User(s):** Student, Supervisor, Admin
* **Input:**
  + Project title
  + Project Description
  + SRS document
  + SDD document
  + UI/UX document
  + Repository link
* **Output:**
  + Plagiarism Report is generated
  + If there’s no plagiarism detected, project is uploaded in the system else user is alerted.
    1. **Review Projects**
* **Name:** FR006
* **Purpose:** To review student’s projects
* **User(s):** Supervisor
* **Input:**
  + Status changed from “Review Pending” to “Reviewed”
  + Feedback is provided (optional)
* **Output:**
  + Project status changed to “Reviewed”
  + Feedback on project
    1. **Grade Projects**
* **Name:** FR007
* **Purpose:** To grade student’s projects
* **User(s):** Supervisor
* **Input:**
  + Status set to “Graded”
  + *Evaluation criteria:* Set be supervisor or admin
* **Output:**
  + Project status Updated to “Graded”
  + Assigned grade is stored in system
    1. **Add Comments**
* **Name:** FR008
* **Purpose:** Add comments to student’s projects
* **User(s):** Supervisor, Student
* **Input:**
  + *Comment Text:* Text of the comment, with a maximum length of 500 characters
  + *User ID:* Automatically fetched ID of the logged-in user adding the comment.
  + *Project ID:* Automatically fetched ID of the project being commented on.
  + *Timestamp:* Automatically captured date and time of the comment submission.
* **Output:**
  + The comment is successfully added and displayed under the project in the comments section.
  + The comment includes:
    - User Name
    - Comment Text
    - Timestamp
    1. **Remove Comments**
* **Name:** FR009
* **Purpose:** Remove comments from student’s projects
* **User(s):** Supervisor, Admin
* **Input:**
  + *Comment ID:* The unique identifier of the comment to be removed.
  + *Confirmation:* A confirmation prompt requiring the user to verify the deletion action
* **Output:**
  + The comment is successfully removed under the project in the comments section.
    1. **Manage Users**
* **Name:** FR010
* **Purpose:** Allow administrators to manage user accounts, including creating, updating, viewing, and deleting user information.
* **User(s):** Admin
* **Input:**
  + *Search Query:* Keyword or user attribute (e.g., name or email) to filter users
  + *User Information:*
    - Name: Full name of the user.
    - Email: A valid email address for the user.
    - Role: Role assigned to the user
    - Status: Active, Inactive, or Suspended.
  + *Update Details:* New values for editable fields such as name, email, or role.
  + *Action Inputs:*
    - *Create User:* Form input fields for new user details.
    - *Update User:* Editable user details and confirmation of changes.
    - *Delete User:* Confirmation to remove a user from the system
* **Output:**
  + User account successfully created, updated, or deleted.
  + A list of users, including the following details:
    - User ID: A unique identifier for each user.
    - Name: Full name of the user.
    - Email: Registered email address.
    - Role: User's assigned role.
    - Status: Active, Inactive, or Suspended
  + Notifications for the admin.

## Non Functional Requirements

## Performance Requirements

**Scalability:** The platform should be able to handle 5000 users at once without experiencing performance issues.

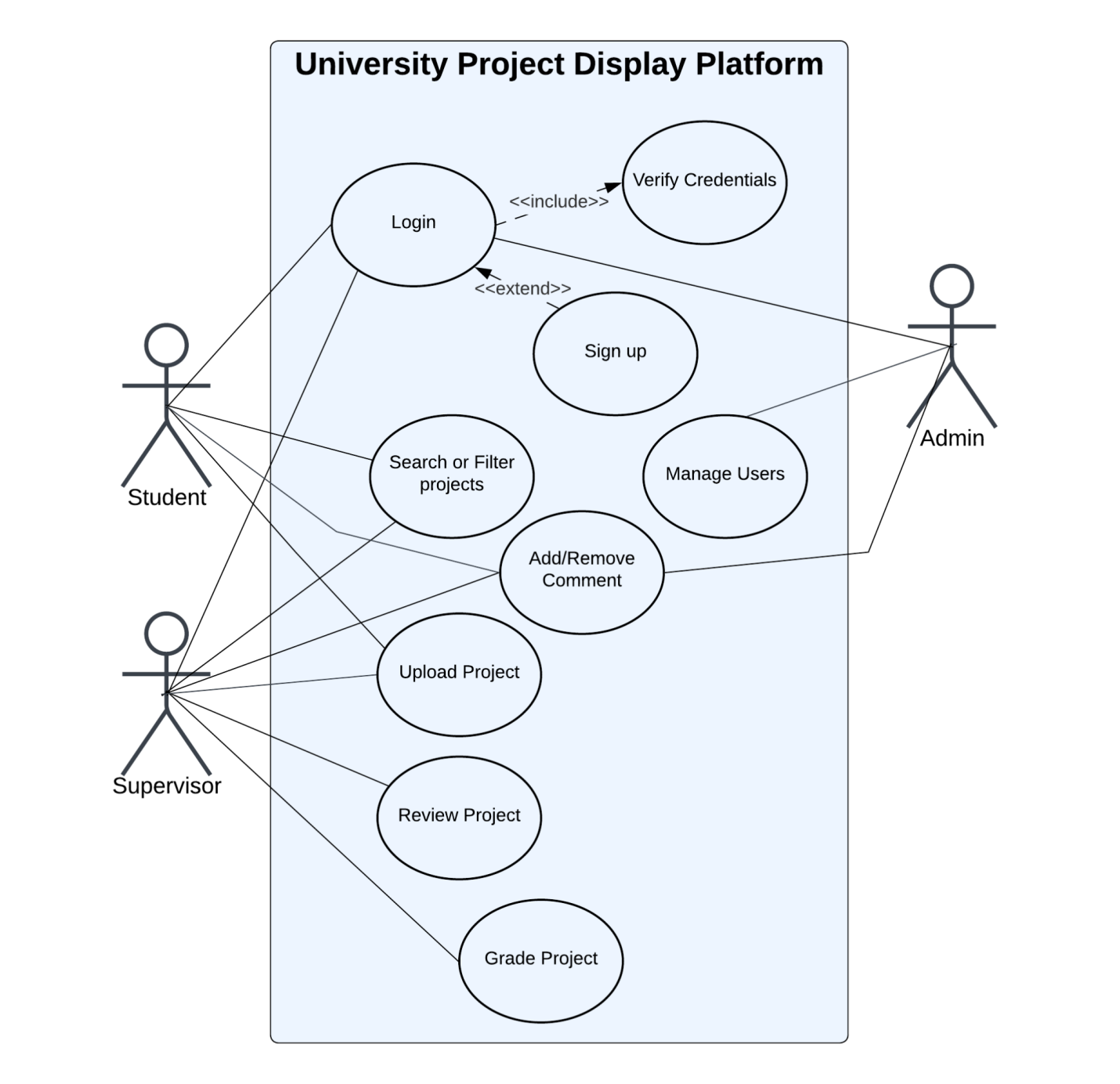
**Reliability:** The platform should be accessible at all times of the day with minimal downtime or service disruptions. It should be responsive at least 99% of the time.

* + 1. **Security Requirements**
* User Identification:
  + There should be proper user identity authentication for
    - Professors
    - Students
    - Administrative Users
* Role-Based Access Control: The product must implement a role-based access control system to ensure that users only have access to the data and functions that they are authorized to access.
* User Identity Verification: The product must verify the identity of all users, such as professors and students, to ensure that only authorized users are allowed to access the system.
* Security Testing: The product must undergo regular security testing, such as penetration testing and vulnerability scanning, to identify and address any security vulnerabilities.
* Students should not be able to access other students’ accounts.
* Students should not be able to access administrative users’ accounts.
  + Students should not be able to access Administrative User functions.
  + Students should not be able to access administrative user data.
* Administrative users should have access to professor data if given permission.
* Administrative users should be able to access student data.

## Software Quality Attributes

* Availability: The platform must be available to users at all times, and there should be safeguards in place to prevent downtime.
* Maintainability: The platform should be easy to maintain and update, with clear documentation and good coding practices.
* Usability: The platform should be easy to use and navigate, with an intuitive user interface and clear features.

# **Use Cases and Flow of Processes**



**[LucidChart Link](https://lucid.app/lucidchart/ecf9e22b-fdcd-47ae-ab79-3389ff1f61a7/edit?viewport_loc=-502%2C-239%2C1295%2C587%2C0_0&invitationId=inv_325cbf52-5c0a-41e5-9c8f-df398a82d66d):** https://lucid.app/lucidchart/ecf9e22b-fdcd-47ae-ab79-3389ff1f61a7/edit?invitationId=inv\_325cbf52-5c0a-41e5-9c8f-df398a82d66d&page=0\_0#

* 1. **Use Case 1**

|  |  |
| --- | --- |
| **ID** | UC001 |
| **Name** | Sign Up |
| **Description** | This use case describes the flow of signing up of a new user |
| **Requirement** | FR001 |
| **Actor(s)** | Student, Supervisor, Admin |
| **Precondition** | Actors must not be a registered user of the Website. |
| **Postcondition** | The actor views a confirmation message stating that the registration is successful |
| **Basic Flow** | 1. Actor enter credentials 2. System validates the inputs    1. Following fields are required       1. Name       2. Username       3. Password    2. Check the proper inputs for following       1. **Username:** Must not already exist in the system.       2. Password must be 8 character including number    3. On successful validation the system will create new user profile   ***Exceptions***   1. Invalid Inputs 2. System is unable to respond |

* 1. **Use Case 2**

|  |  |
| --- | --- |
| **ID** | UC002 |
| **Name** | Sign In |
| **Description** | This use case describes the flow of signing in of a registered user |
| **Requirement** | FR002 |
| **Actor(s)** | Student, Supervisor, Admin |
| **Precondition** | Actor must be a registered user |
| **Postcondition** | Actor views Homepage of Website |
| **Basic Flow** | 1. Actor enter credentials 2. System validates the inputs    1. Following fields are required       1. Username       2. Password    2. Validate the input credentials 3. On successful validation the system shows Home Screen 4. On unsuccessful validation system    1. Shows invalid login Message    2. Shows Signup Button   ***Exceptions***   1. Invalid Inputs 2. System is unable to respond |

* 1. **Use Case 3**

|  |  |
| --- | --- |
| **ID** | UC003 |
| **Name** | Search or Filter Projects |
| **Description** | This use case describes the flow of searching and filtering of students' projects. |
| **Requirement** | FR004 |
| **Actor(s)** | Student, Supervisor |
| **Precondition** | Actors must view the list of all projects sorted in the most recent to oldest manner. |
| **Postcondition** | The actor views a list of projects that match the search or filter criteria. |
| **Basic Flow** | 1. The actor can perform the following actions: 2. Enter a keyword to search. 3. Apply one or more filters, such as:    1. Project Status    2. Department    3. Project Category    4. Year 4. The system validates the search or filter inputs. 5. The system queries the database for matching projects. 6. Database returns results based on the search query. 7. The system displays the results to the actor in a paginated or scrollable format.   ***Exceptions***   1. Invalid Inputs 2. No results found 3. System is unable to respond |

* 1. **Use Case 4**

|  |  |
| --- | --- |
| **ID** | UC004 |
| **Name** | Upload Project |
| **Description** | This use case describes the flow of uploading projects and checking plagiarism using the Turnitin API before submission.  Actors must have access to the Turnitin plagiarism check feature. |
| **Requirement** | FR005 |
| **Actor(s)** | Student, Supervisor |
| **Precondition** | Actors must be directed to fillable form once they click on the Upload Project button. |
| **Postcondition** | 1. Actors view the uploaded project on the “View Project” page. 2. A plagiarism report is generated for review before final submission. |
| **Basic Flow** | 1. Actors are directed to the fillable form page once they click on the Upload Project Button. 2. Actors view the fillable form which should consist of following:    1. Project Title    2. Description    3. SRS Document    4. SDD Document    5. UI/UX Document    6. Any Media file(Can include images or video) 3. Actors click on "Check for Plagiarism" button. 4. The system sends the attached documents to the Turnitin API. 5. The system retrieves and displays the plagiarism report to the actor, including:    1. Overall Similarity Index    2. Highlighted Similarities with Sources 6. Actors review the plagiarism report and make any necessary changes to their documents. 7. Actors click "Submit" to finalize the upload. 8. After submission, the system saves the uploaded project details and files. 9. The system redirects the actors to the "View Project" page, where they can view their uploaded project.   ***Exceptions***   1. Invalid attached file formats 2. Turnitin API failure 3. System is unable to respond |

* 1. **Use Case 5**

|  |  |
| --- | --- |
| **ID** | UC005 |
| **Name** | Add or Remove Comments |
| **Description** | This use case describes the flow of adding or removing comments in a user's project. |
| **Requirement** | FR008, FR009 |
| **Actor(s)** | Student, Supervisor, Admin |
| **Precondition** | The project must be accessible to the actor (based on permissions).   1. For Adding Comments (Student and Supervisor Only):    1. The comment input field must be available and active. 2. For Removing Comments (Admin Only):    1. The comment to be deleted must exist. |
| **Postcondition** | 1. For Adding Comments (Student and Supervisor Only):   a. A new comment is successfully added to the project.   1. For Removing Comments (Admin Only):   a. The comment is successfully deleted from the project. |
| **Basic Flow** | 1. Adding Comments (Student, Supervisor)   a. The actor (Student or Supervisor) navigates to the project details page.  a. The actor enters a comment in the provided text field.   * 1. The actor clicks the "Add Comment" button.   2. The system validates the input for:      1. Non-empty content.      2. Allowed character limits and format.      3. The system saves the comment in the database and associates it with the project and the actor.  1. Removing Comments (Admin)    1. The Admin identifies the comment to be removed and clicks the "Delete" button/icon.    2. The system verifies the Admin's permissions.    3. The system removes the comment from the database.   ***Exceptions***   1. Invalid Inputs 2. System unable to save comment 3. System unable to delete comment |

* 1. **Use Case 6**

|  |  |
| --- | --- |
| **ID** | UC006 |
| **Name** | Review Project |
| **Description** | This use case describes the flow of reviewing a student's project. |
| **Requirement** | FR006 |
| **Actor(s)** | Supervisor |
| **Precondition** | The project must be available for review and it must not have been reviewed already. |
| **Postcondition** | The system saves the review (including any feedback or ratings provided) associated with the project. |
| **Basic Flow** | 1. The actor views a list of projects assigned to them or available for review. 2. The actor selects a project marked as "Pending Review". 3. The system presents the full project details submitted by the student. 4. The actor enters feedback in a provided text box. 5. If applicable, the actor assigns a rating. 6. The actor clicks the "Submit Review" button. 7. The system checks that all required fields (e.g., feedback, rating) are completed and valid. 8. The system saves the review.   ***Exceptions***   1. Project not found 2. Validation errors    1. Required fields are left empty or invalid inputs are provided 3. System is unable to save review 4. Project already reviewed |

* 1. **Use Case 7**

|  |  |
| --- | --- |
| **ID** | UC007 |
| **Name** | Grade Project |
| **Description** | This use case describes the flow of grading a student's project based on predefined criteria. |
| **Requirement** | FR007 |
| **Actor(s)** | Supervisor |
| **Precondition** | The project must be accessible and in “Pending Grading” status. |
| **Postcondition** | The project is successfully graded, and the grade is stored in the system.  The project status is updated to “Graded”. |
| **Basic Flow** | 1. The Supervisor views a list of projects awaiting grading and selects a specific project. 2. The system presents project details, including the title, description, submission files, plagiarism report, and any associated reviews or comments. 3. The Supervisor enters a grade or marks in the provided input field. 4. The Supervisor clicks the "Submit Grade" button. 5. The system checks the grade for validity. 6. The system saves the grade in the database, associates it with the project, and updates the project status to "Graded". 7. The system displays a success message to the Supervisor.   ***Exceptions***   1. Invalid Grade Inputs 2. Project Not Found 3. System is unable to respond |

* 1. **Use Case 8**

|  |  |
| --- | --- |
| **ID** | UC008 |
| **Name** | Manage User |
| **Description** | This use case describes the process by which an Admin manages user accounts, including viewing, updating, activating, deactivating, or deleting user profiles. |
| **Requirement** | FR010 |
| **Actor(s)** | Admin |
| **Precondition** | The users to be managed must exist in the system. |
| **Postcondition** | The user account is successfully updated, activated, deactivated, or deleted as per the Admin's action. |
| **Basic Flow** | 1. The Admin selects the "Manage Users" option from the menu. 2. The system displays a list of all users, including their details such as name, role, status (active/inactive), and date of registration. 3. The Admin selects a user and chooses one of the following actions:    1. View User Details    2. Update User Information    3. Activate/Deactivate User    4. Delete User 4. The system updates the database with the changes.   ***Exceptions***  Invalid Input  User Not Found  System Error |

# **4.** **References**

(About us - Devpost, n.d.)