Gregory Luna

CS 691

UML Diagram

3/8/2025

1. Use Case Diagram

Purpose: Show actors and their interactions with the system.

Actors: [Student] [Tutor] [Admin]

| | |

| | |

System: +--------------------------+

| Student Resource |

| Sharing Platform |

+--------------------------+

| + Register |<-----[Student,Tutor,Admin]

| + Login |<-----[Student,Tutor,Admin]

| + Upload Resource |<-----[Student,Tutor]

| + Search Resource |<-----[Student,Tutor]

| + Comment on Resource |<-----[Student,Tutor]

| + Review Content |<-----[Admin]

| + Manage Users |<-----[Admin]

+--------------------------+

Details:

* Actors:
  + Student: Primary user who uploads, searches, and comments.
  + Tutor: Similar to Student but with specific resource types (e.g., practice problems).
  + Admin: Manages content and users.
* Use Cases:
  + Register, Login: Authentication for all users (Technical Implementation: Backend).
  + Upload Resource: Students and Tutors share resources (Key Features: Resource Sharing).
  + Search Resource: Finding resources by keyword/subject (Key Features: Basic Search).
  + Comment on Resource: Interaction with resources (Key Features: Comments).
  + Review Content, Manage Users: Admin-specific tasks (User Roles: Administrators).
* Interactions: <----- indicates which actors interact with each use case.

2. Class Diagram

Purpose: Define the system’s structure with classes, attributes, methods, and relationships.

+----------------+ +------------------+ +-----------------+

| User | | Resource | | Comment |

+----------------+ +------------------+ +-----------------+

| - id: int |<>---->| - id: int |<>---->| - id: int |

| - username: | 1 \* | - title: string | 1 \* | - content: |

| string | | - description: | | string |

| - email: | | string | | - created\_at: |

| string | | - file\_path: | | datetime |

| - password\_hash| | string | | - user\_id: int |

| : string | | - user\_id: int | | - resource\_id: |

| - created\_at: | | - subject: | | int |

| datetime | | string | +-----------------+

| - role: string | | - created\_at: | | + create() |

+----------------+ | datetime | | + delete() |

| + register() | +------------------+ +-----------------+

| + login() | | + upload() |

| + logout() | | + delete() |

+----------------+ | + search() |

^ +------------------+

| inherits

|

+-------------------+-------------------+

| | |

+----------------+ +----------------+ +----------------+

| Student | | Tutor | | Admin |

+----------------+ +----------------+ +----------------+

| + uploadResource()| | + uploadPracticeProblems()| | + reviewContent() |

| + searchResource()| | + reviewResources()| | + manageUsers() |

+----------------+ +----------------+ +----------------+

Details:

* Classes:
  + User: Base class with Student, Tutor, Admin subclasses (inheritance via ^).
  + Resource: Core entity for shared materials.
  + Comment: Interaction entity linked to resources.
* Attributes: Match the MySQL schema (e.g., users, resources, comments tables).
* Methods: Reflect key functionalities (e.g., register(), upload(), search()).
* Relationships:
  + User to Resource: 1-to-many (1 \*).
  + Resource to Comment: 1-to-many (1 \*).

3. Sequence Diagram

Purpose: Illustrate the flow for two key functionalities: User Registration and Resource Upload.

a. User Registration

[Student] [Frontend] [Backend] [Database]

| | | |

|---register-->| | |

| |---POST /register-->| |

| | |---INSERT user-->|

| | |<--user\_id-------|

| |<--201 Created------| |

|<--Success----| | |

b. Resource Upload

[Student] [Frontend] [Backend] [Database]

| | | |

|---uploadResource-->| | |

| |---POST /resources-->| |

| | |---INSERT resource-->|

| | |<--resource\_id-------|

| |<--201 Created------| |

|<--Success----| | |

Details:

* Actors/Objects: Student (user), Frontend (React), Backend (Node.js), Database (MySQL).
* Flow:
  + Registration: Student submits registration form, frontend sends POST request, backend inserts user into database, returns success.
  + Upload: Student uploads a resource, frontend sends POST with file, backend stores it in database, returns success.
* Reflects RESTful API and full-stack integration (Technical Implementation).

4. Activity Diagram

Purpose: Represent the workflow for Document Upload (a core process from Key Features: Resource Sharing).

[Start]

|

v

+-------------+

| User Logs In |

+-------------+

|

v

+-------------+

| Select File |<----[Frontend: Upload Form]

+-------------+

|

v

+-------------+

| Enter Details| (title, description, subject)

+-------------+

|

v

+-------------+

| Submit Form |---->[Frontend: POST /resources]

+-------------+

|

v

+-------------+

| Backend |---->[Validate Input]

| Processes |

+-------------+

|

v

+-------------+

| Store File |---->[Save to File System]

+-------------+

|

v

+-------------+

| Database |---->[INSERT into resources]

| Update |

+-------------+

|

v

+-------------+

| Return |---->[201 Created]

| Success |

+-------------+

|

v

[End]

Details:

* Process: Document Upload.
* Steps:
  1. User logs in (prerequisite).
  2. Selects a file and enters metadata (frontend form).
  3. Submits form, triggering a POST request.
  4. Backend validates input, stores the file, and updates the database.
  5. Success response returned to the user.
* Annotations: [Frontend: ...] and [Backend: ...] indicate system components involved.