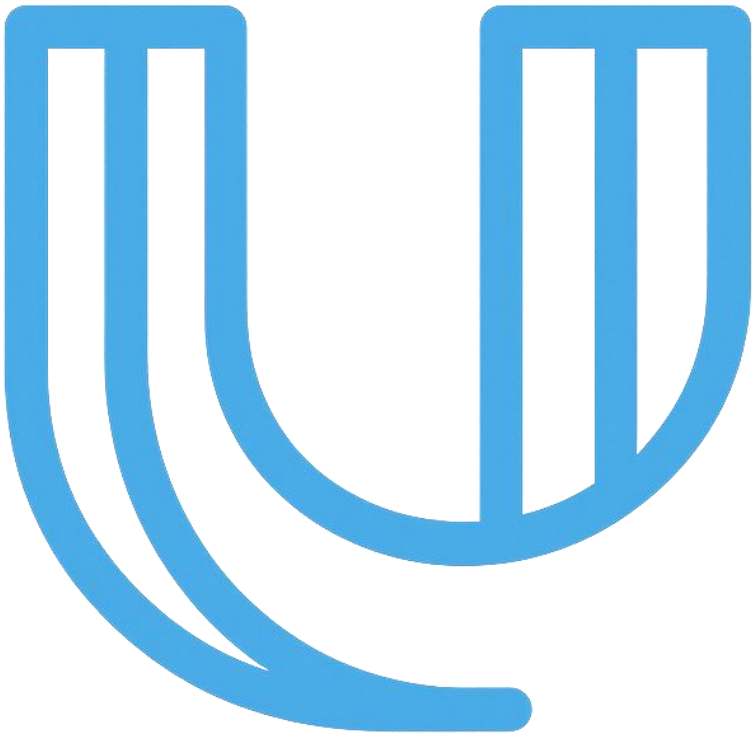
U-COLLAB

Research project conduction, management and collaboration made easier for everyone.



University of the Witwatersrand
COMS3009A - Software Design

Table of Contents

About the Dev Team

Behind every groundbreaking solution is a team that dared to dream bigger — below are the trailblazers who turned your research needs into a thrilling reality.

- Samuel Amoah
- Tshiamo Madikgetla
- Odirile Mthimunye
- Lesedi Seganoe
- Monare Selahle
- Ndivho Shilling

Executive Summary

This document outlines the development journey and foundational architecture of our university research collaboration platform — a digital solution designed to streamline academic collaboration, centralize project management, and enhance research efficiency across departments and institutions. For simplicity, this platform shall be referred to by its intuitive name, U-Collab.

The platform was developed with a strong focus on usability, aiming to support the diverse and evolving needs of the academic research community. It enables users to coordinate projects, share resources, and track progress through an intuitive and robust interface.

Key areas covered in this document include, but are not limited to:

- Technology Stack
- System Architecture
- User Stories
- User Acceptance Tests
- Models
- Project Management Practices
- Strategic Considerations

This summary reflects the collaborative effort and thoughtful planning invested in building a platform that not only meets current research demands but is also adaptable to the changing landscape of academic collaboration.

Tech Stack – MERN

What is MERN?

"The MERN stack is a popular set of technologies used to build full-stack web applications using JavaScript for both front-end and back-end development."

The acronym stands for:

- **M** – MongoDB (Database layer)
- **E** – Express.js (APIs and server-side logic)
- **R** – React.js (Front-end)
- **N** – Node.js (Back-end)

Why MERN?

We have a full-stack JavaScript project:

- MongoDB stores data in BSON documents (Binary JSON) meaning there is no need to convert data formats when sending data between our frontend, backend, and database. This unified language approach simplifies development, improves team collaboration, and speeds up debugging and feature integration.
- MongoDB being schema-less means that data can be stored in a flexible format. Data is kept in "documents" where each document in a "collection" can have different fields from the next. We can store things like messages, collaborators, and documents all in one collection, instead of spreading them across different tables like in an SQL database.
- React uses a Virtual DOM (a copy of the real DOM which exists in memory) to detect what parts of the page need to change and only updates those, not the whole page. This allows users a seamless, responsive experience where data appears and updates in real time. Perfect for features such as sending messages, project updates, adding new collaborators and milestones.
- React lets us break up our UI into reusable components, making it easier to maintain and scale. This is beneficial to us, considering the complex components we must implement (user profiles, project pages, messaging etc.). It also allows us to build features incrementally and adapt as requirements change, which is a core part of Agile.
- Node's non-blocking, event-driven architecture is ideal for handling multiple simultaneous users, which is perfect for a collaboration platform where researchers are constantly interacting, updating projects, and sending messages in real time.
- Express is a lightweight and flexible web framework that runs on top of Node, making it easier to build and manage backend logic and APIs. It simplifies routing, handles requests and responses efficiently, and integrates smoothly with databases

like MongoDB. For our research collaboration platform, Express allows us to quickly set up endpoints for tasks like project creation, user management, and messaging.

System Architecture

To ensure a robust and maintainable system architecture, we modelled our platform according to the 4+1 architectural view model, covering the following perspectives:

- Logical View
- Development View
- Process View
- Physical View
- Scenarios

Each of these views is supported by UML diagrams, which we'll see later in this document, enabling clear communication between technical and non-technical stakeholders and aiding future system evolution.

User Stories

We now make a recall of the user stories that have been scheduled for implementation in the system for each sprint. Each user story has its own acceptance criteria, which provides a detailed description of how the success of the system will be measured upon implementation. These criteria are then summarized into a User Acceptance Test, which outlines the expected behaviour of the system given a specific action or behaviour from the user.

Sprint 1

Below are the user stories that were scheduled for prototyping in Sprint 1:

- As a researcher, I should be able to log in so that I can have access to the system.
- As a researcher, I should be able to define requirements so that I can show what is needed for the project.
- As a researcher, I should be able to track milestones so that I can assess the progress within projects.
- As a researcher, I should be able to track spending so that I can see how much money has been utilized.

Sprint 2

Inclusive of the ones listed above, the following user stories were scheduled for implementation by the end of Sprint 2:

- As a reviewer, I should be able to log in so that I can have access to the system.
- As a researcher, I should be able to create project listings so that I can show active research that's open for collaboration.
- As a researcher, I should be able to add collaborators so that they can participate in active projects.
- As a researcher, I should be able to share documents so that collaborators can have access to them.
- As a researcher, I should be able to track grants so that I can see what funds are available for use.
- As a researcher, I should be able to track funding requirements so that I can ensure that a given project has enough funding to sustain it till completion.

Sprint 3

The following user stories, in addition to all previously indicated ones, were scheduled for implementation by the end of Sprint 3:

- As a researcher, I should be able to add collaborators so that they can contribute to a given project.
- As a researcher, I should be able to send and receive messages so that I can communicate with collaborators.

- As a researcher, I should be able to share and view documents so that collaborators and reviewers can see them.
- As a researcher, I should be able to request access to be a reviewer so that I can provide feedback to projects.
- As a reviewer, I should be able to make reviews for projects so that I can provide comments and suggestions.
- As a reviewer, I should be able to send and receive messages so that I can communicate with researchers.
- As a reviewer, I should be able to view documents so that I can access them.
- As an admin, I should be able to approve or deny reviewer requests so that access to projects is authorized.
- As a researcher, I should be able to view dashboard reports so that I can use the overall data to make meaningful decisions.
- As a researcher, I should be able to export dashboard reports as a .csv or .pdf file so that I can share them with stakeholders efficiently.
- As a reviewer, I should be able to view dashboard reports so that I can use the overall data to make meaningful decisions.
- As a reviewer, I should be able to export dashboard reports as a .csv or .pdf file so that I can share them with stakeholders efficiently.

Sprint 4

No new user stories were introduced in Sprint 4. However, the backlog from Sprint 3 was attended to during this sprint, allowing maximized feature functionality. The actors for the last four user stories from the previous sprint were also refined after consultation with the client. It was initially determined that an administrator would view and export dashboard reports, but later clarified that the feature is meant for researchers and reviewers. These changes are already noted in the section above and need not be restated, thus eliminating any confusion.

User Acceptance Tests

Sprint 1

Below are the UATs associated with the user stories introduced in Sprint 1.

1. *"As a researcher, I should be able to log in to access the system."*

   Acceptance Criteria:

   - The user has a registered account in the database.
   - The user can access the login page.
   - The user can put their details in the required fields and tap "login."
   - After tapping "login", the user is taken to the homepage.

UAT:

Given that the user has a registered account, **when** they access the login page and input their login details, **then** they should be taken to the homepage and have access to the system.

2. *"As a researcher, I should be able to define requirements so that I can show what is needed for the project."*

   Acceptance Criteria:

   - When creating or accessing a project, the user should be able to access the section "Project Requirements".
   - The user should be able to add multiple requirements.
   - Each requirement can be written as free text and edited or deleted before submission.
   - Upon the user saving the project, requirements should be displayed on the project detail page, visible to the user.

UAT:

Given the researcher is logged in and is on defined requirements, **when** they enter details for project requirements and save, **then** the details should appear on the project details page.

3. *"As a researcher, I should be able to track spending so that I can see how much money has been utilized."*

Acceptance Criteria:

- The user can check out the budget details for every project.
- The system shows you the total budget, how much has been spent, and what's left in your balance.
- The user can view a breakdown of spending by category and date.
- The data updates as new expenses are added.

UAT:
Given that the user is logged into the system and selects a project, **when** they navigate to the spending section, **then** they should see a detailed breakdown of total spending, including categories and remaining budget.

4. *"As a researcher, I should be able to track milestones so that I can assess the progress within projects."*

Acceptance Criteria:

- Users can easily check out a list of the projects they're currently working on.
- Each project displays associated milestones.
- Milestones show current status.
- The user can update the status of a milestone.
- The milestone updates are reflected immediately in the project overview.

UAT:
Given that the user is logged into the system and working on a project, **when** they view the project's milestones and update their statuses, **then** the updated statuses should be visible in real time and reflect the current progress on the project.

Sprint 2

Below are the UATs associated with the user stories that were introduced in Sprint 2.

5. *"As a reviewer, I should be able to log in so that I can have access to the system."*

   Acceptance Criteria:

   - The reviewer is presented with a login page.
   - Login accepts valid credentials and grants access.
   - Invalid credentials show an appropriate error message.

UAT:

Given that the user has a registered account, **when** they access the login page and input their login details, **then** they should be taken to the homepage and have access to the system.

6. *"As a researcher, I should be able to create project listings so that I can show active research."*

   Acceptance Criteria:

   - Researcher sees an option to create a new project.
   - Project form allows input of title, description, status, and collaboration options.
   - Project appears in the project listing after submission.

UAT:

Given that I am a researcher, **when** I create a new project with valid data, **then** it should appear in the project listings and be marked as open for collaboration.

7. *"As a researcher, I should be able to add collaborators so that they can contribute to a given project."*

   Acceptance Criteria:

   - Researchers can add collaborators to their projects.
   - Collaborators are linked to specific projects.
   - Collaborators can contribute to the project by adding or editing content.

UAT:

Given that I manage a project, **when** I collaborator, **then** the system should allow them to contribute to the project.

8. *“As a researcher, I should be able to share documents so that collaborators can have access to them.”*

Acceptance Criteria:

- Researcher can upload and attach documents to projects.
- Collaborators can view/download shared documents.

UAT:

Given that I am a researcher, **when** I upload a document to a project, **then** collaborators should be able to view or download it from the project’s document section.

9. *“As a researcher, I should be able to track grants so that I can see what funds are available for use.”*

Acceptance Criteria:

- Researcher can input grant details (source, amount, duration).
- Grants are linked to specific projects or researchers.
- Available funds are calculated and displayed.

UAT:

Given that I manage a project, **when** I enter grant details, **then** the system should show the available funds and link them to the project.

10. *“As a researcher, I should be able to track funding requirements so that I can ensure that a given project has enough funding to sustain it till completion.”*

Acceptance Criteria:

- Researcher can set estimated total funding needed for a project.
- System compares required funding with available funds.
- Alerts are shown if there's a shortfall.

UAT:

Given that I am reviewing a project, **when** I set the funding requirement, **then** the system should compare it to current funding and notify me if there's a shortfall.

Sprint 3

Below are the UATs associated with the user stories that were introduced in Sprint 3.

11. *"As a researcher, I should be able to add collaborators so that they can contribute to a given project."*

    Acceptance Criteria:

    - Researchers can add collaborators to their projects.
    - Collaborators are linked to specific projects.
    - Collaborators can contribute to the project by adding or editing content.

UAT:

Given that I manage a project, **when** I collaborator, **then** the system should allow them to contribute to the project.

12. *"As a researcher, I should be able to send and receive messages so that I can communicate with collaborators."*

    Acceptance Criteria:

    - Researchers can send messages to collaborators.
    - Researchers can receive messages from collaborators.

UAT:

Given that I have collaborators, **when** I send or receive a message, **then** the system should support communication within the project.

13. *"As a researcher, I should be able to share and view documents so that collaborators and reviewers can see them."*

    Acceptance Criteria:

    - Researchers can upload and share documents with collaborators and reviewers.
    - Collaborators and reviewers can view the documents.

UAT:

Given that I am working on a project, **when** I share or view a document, **then** the system should allow access to collaborators and reviewers.

14. *"As a researcher, I should be able to request access to be a reviewer so that I can provide feedback to projects."*

Acceptance Criteria:

- Reviewers can submit a request to review a project.
- Admins are notified of the request.

UAT:

Given that I am a researcher, **when** I request to become a reviewer, **then** the system should notify the admin for approval.

15. *"As a reviewer, I should be able to make reviews for projects so that I can provide comments and suggestions."*

Acceptance Criteria:

- Reviewers can submit reviews for specific projects.
- Reviews include comments and suggestions.

UAT:

Given that I am approved to review, **when** I make a review, **then** the system should capture and display my comments and suggestions.

16. *"As a reviewer, I should be able to send and receive messages so that I can communicate with researchers."*

Acceptance Criteria:

- Reviewers can send messages to researchers.
- Reviewers can receive messages from researchers.

UAT:

Given that I am a reviewer, **when** I send or receive a message, **then** the system should support communication with researchers.

17. *"As a reviewer, I should be able to view documents so that I can access them."*

Acceptance Criteria:

- Reviewers can view shared project documents.
- Reviewers can share comments on documents.
- Document sharing is linked to specific projects.

UAT:

Given that I am a reviewer, **when** I share or view documents, **then** the system should allow me to access and review them.

18. *"As an admin, I should be able to approve or deny reviewer requests so that access to projects is authorized."*

    Acceptance Criteria:

    - Admins can review and approve/decline reviewer requests.
    - Access to reviewing projects is granted based on admin approval.

UAT:

Given that I am an admin, **when** a request to become a reviewer is submitted, **then** the system should allow me to approve or decline it.

19. *"As a researcher, I should be able to view dashboard reports so that I can use the overall data to make meaningful decisions."*

    Acceptance Criteria:

    - Researchers can view a dashboard with project and review data.
    - Data includes overall project statuses, reviewers, and milestones.
    - The dashboard is updated in real-time.

UAT:

Given that I am a researcher, **when** I open the dashboard, **then** the system should display reports with overall project data.

20. *"As a researcher, I should be able to export dashboard reports as a .csv or .pdf file so that I can share them with stakeholders efficiently."*

    Acceptance Criteria:

    - Researchers can export dashboard reports in .csv or .pdf format.
    - Reports can be shared via email or downloaded.
    - Exported files contain the latest project and review data.

UAT:

Given that I am viewing dashboard reports, **when** I export them, **then** the system should provide the file in .csv or .pdf format.

21. *"As a reviewer, I should be able to view dashboard reports so that I can use the overall data to make meaningful decisions."*

Acceptance Criteria:

- Reviewers can view a dashboard with project and review data.
- Data includes overall project statuses, reviewers, and milestones.
- The dashboard is updated in real-time.

UAT:

Given that I am a reviewer, **when** I open the dashboard, **then** the system should display reports with overall project data.

22. *"As a reviewer, I should be able to export dashboard reports as a .csv or .pdf file so that I can share them with stakeholders efficiently."*

Acceptance Criteria:

- Reviewers can export dashboard reports in .csv or .pdf format.
- Reports can be shared via email or downloaded.
- Exported files contain the latest project and review data.

UAT:

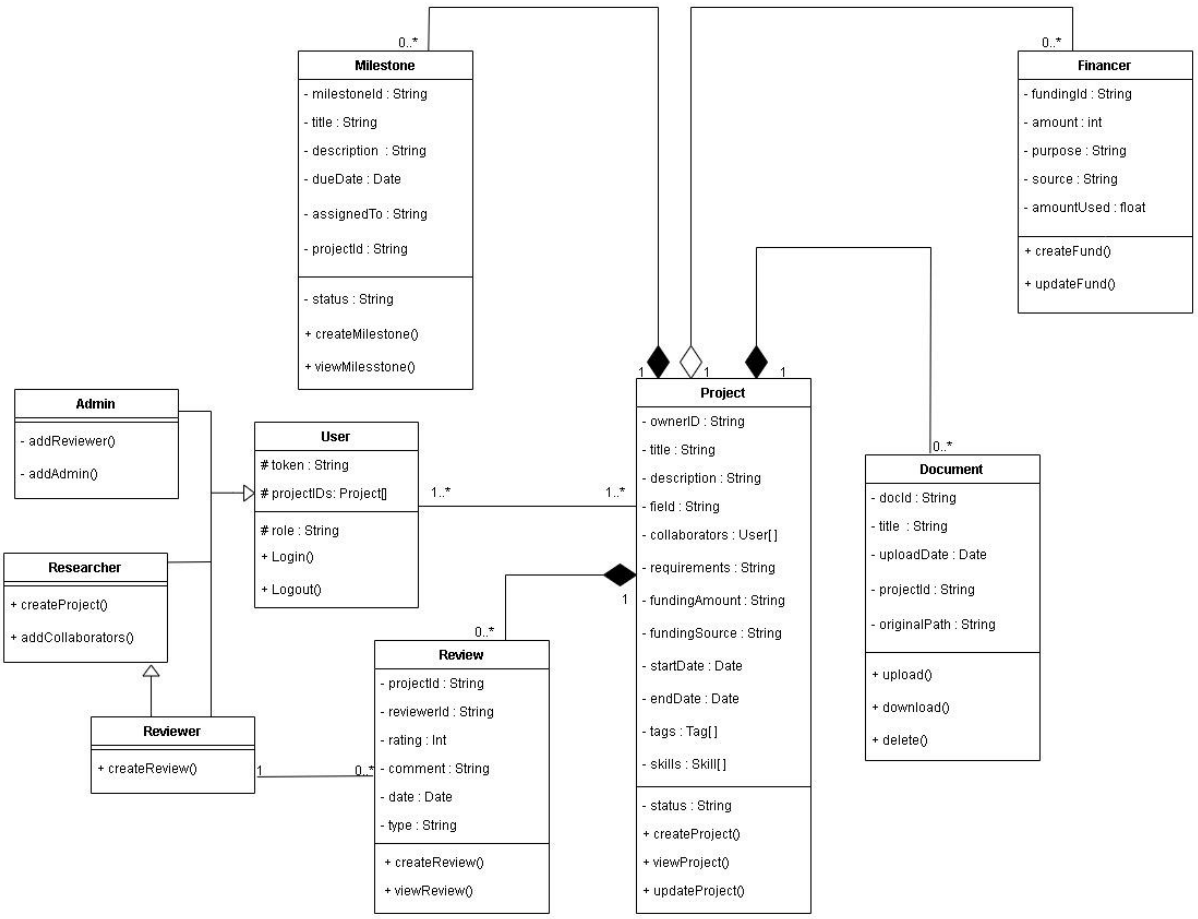
Given that I am viewing dashboard reports, **when** I export them, **then** the system should provide the file in .csv or .pdf format.

UML Diagrams

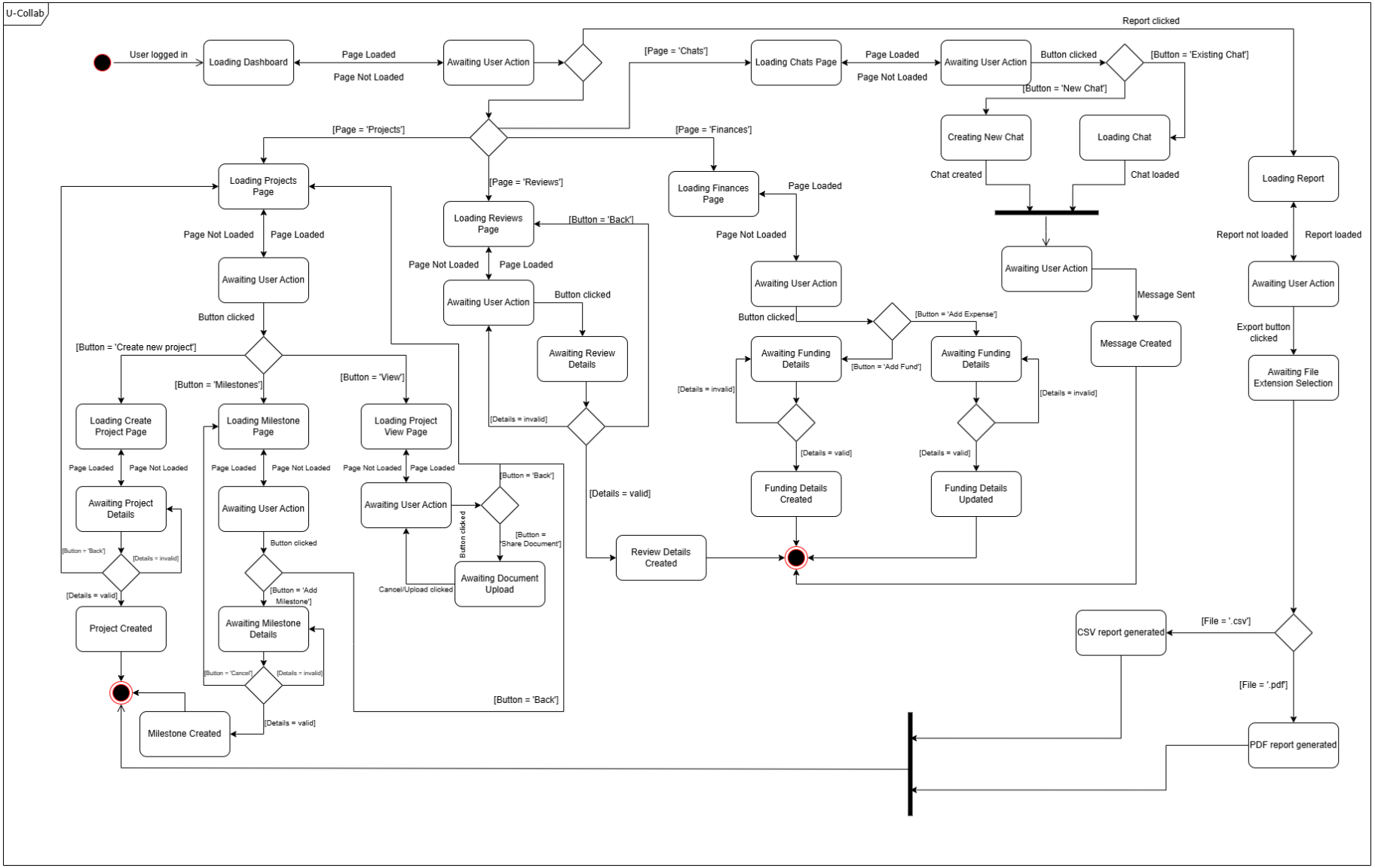
Recall that this document previously mentioned the system architecture, wherein the 4+1 architectural view was introduced. Below are the associated diagrams for each of the views in this architecture model. These are available here with better display resolutions, for convenience.

Logical View

Class Diagram

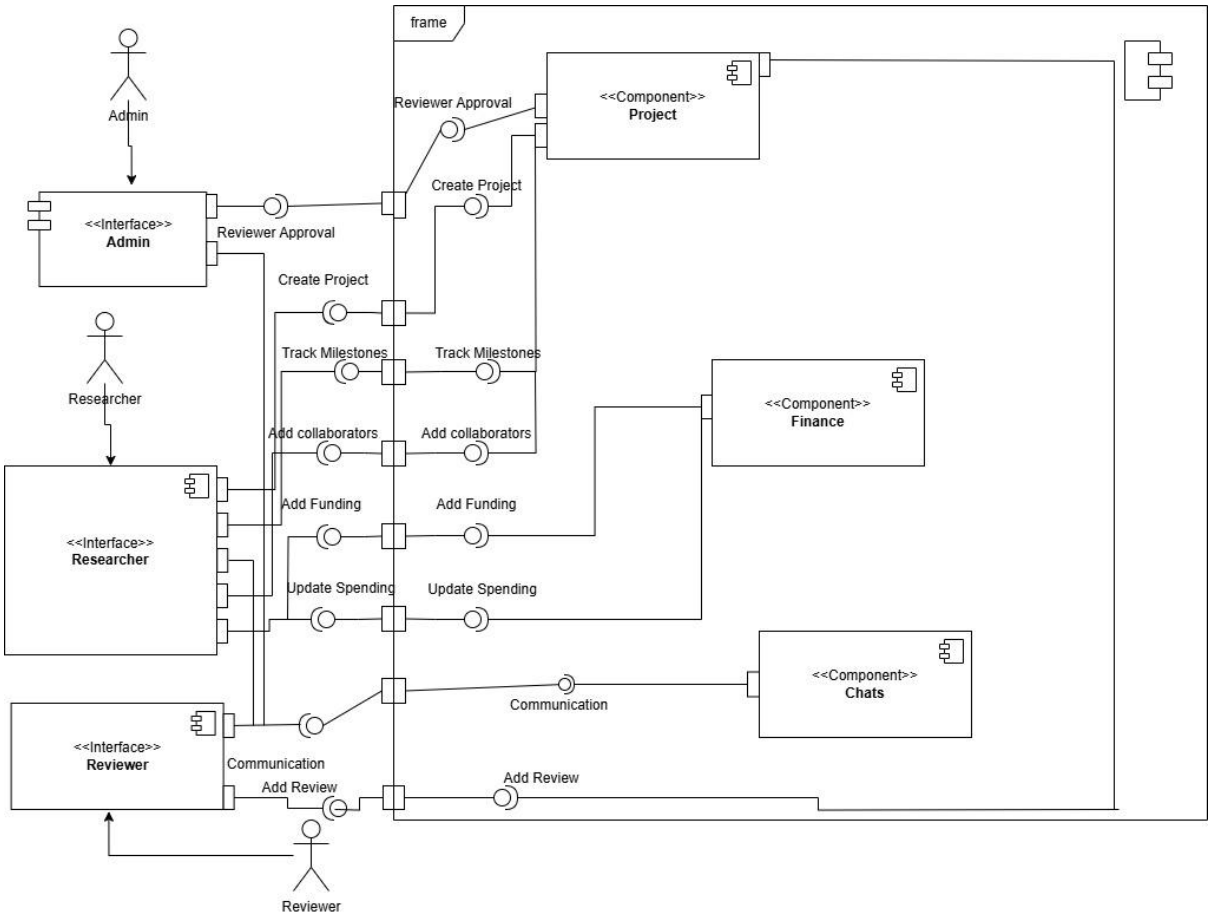


State Machine Diagram



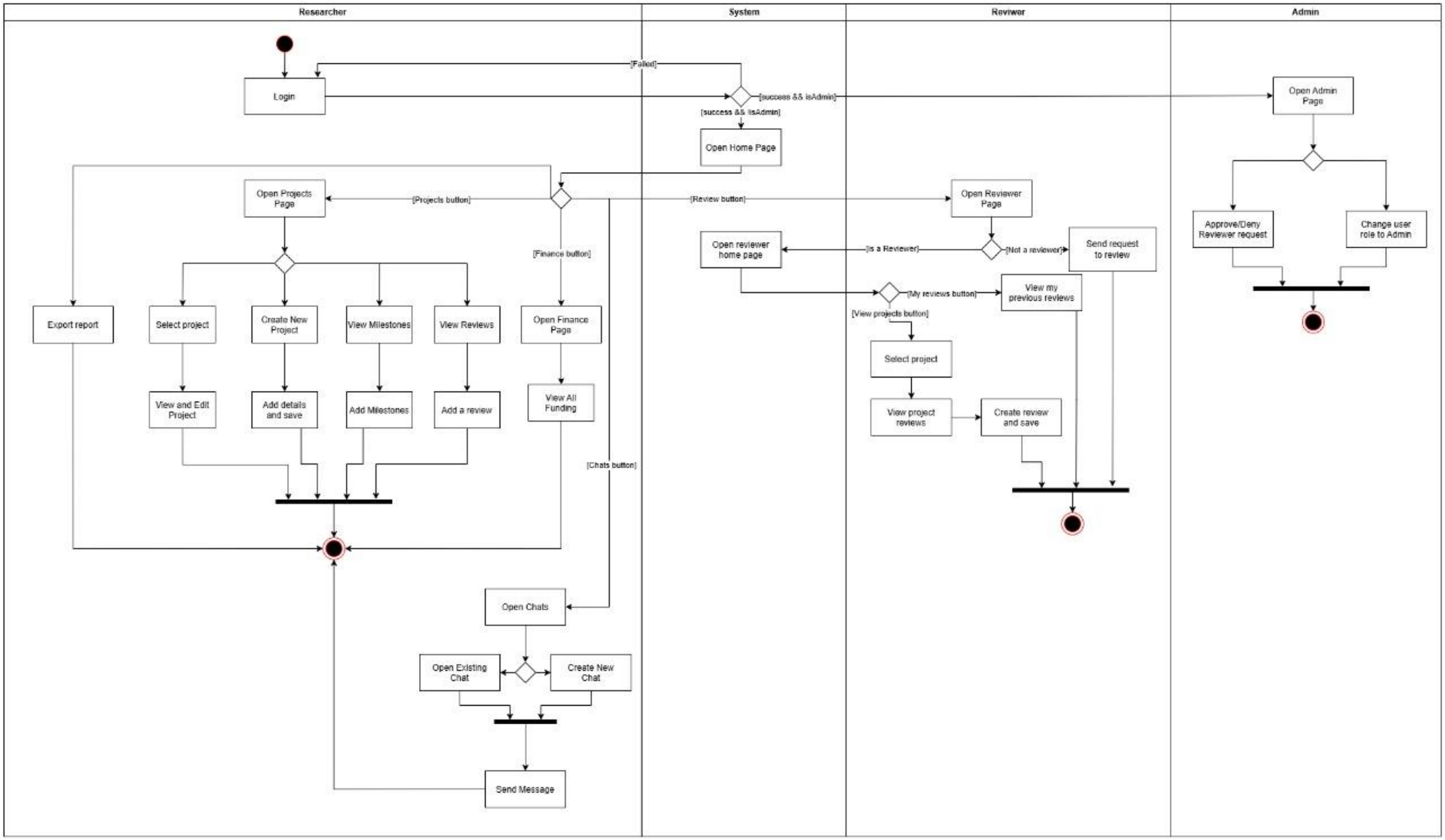
Development View

Component Diagram

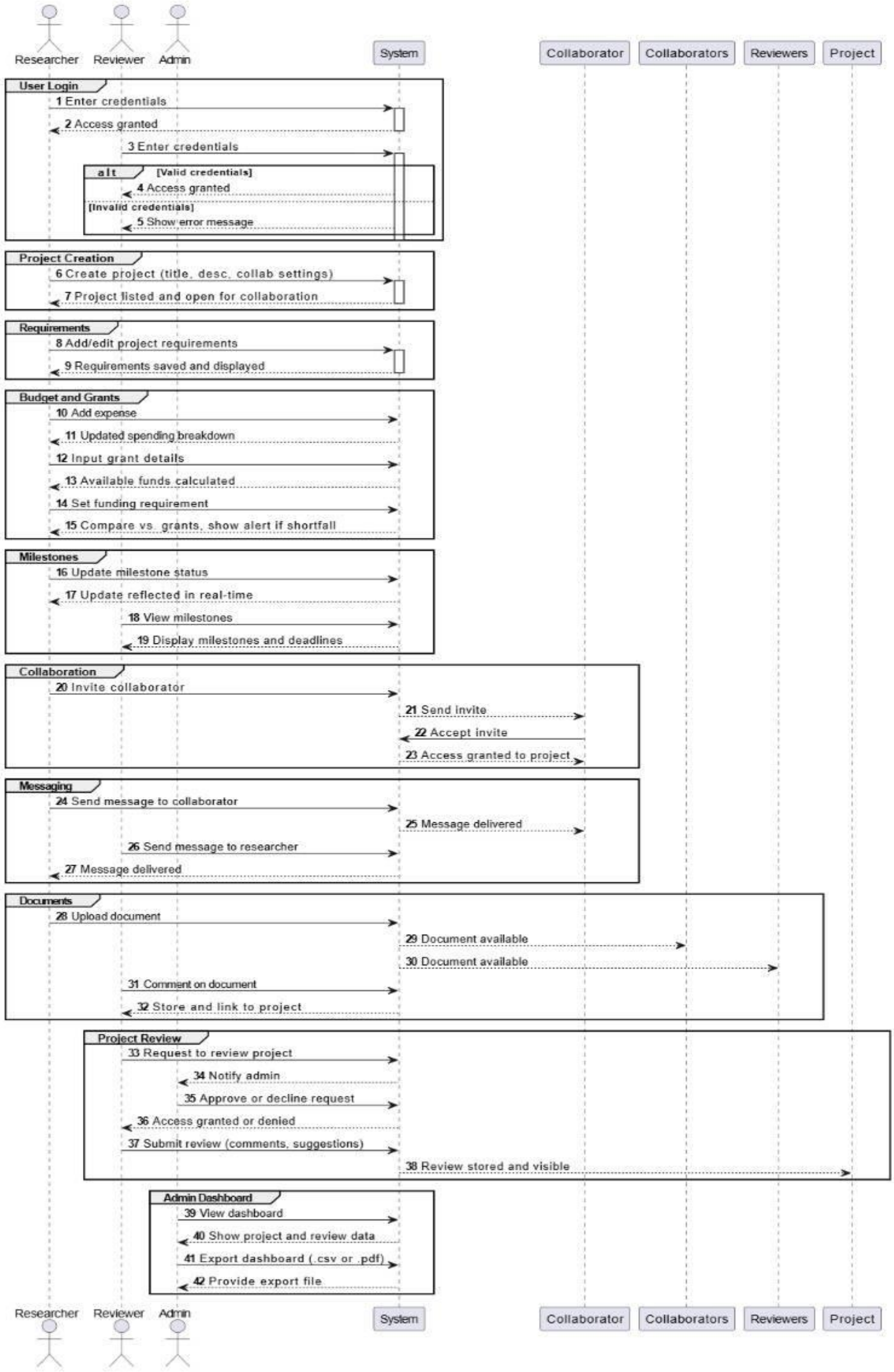


Process View

Activity Diagram

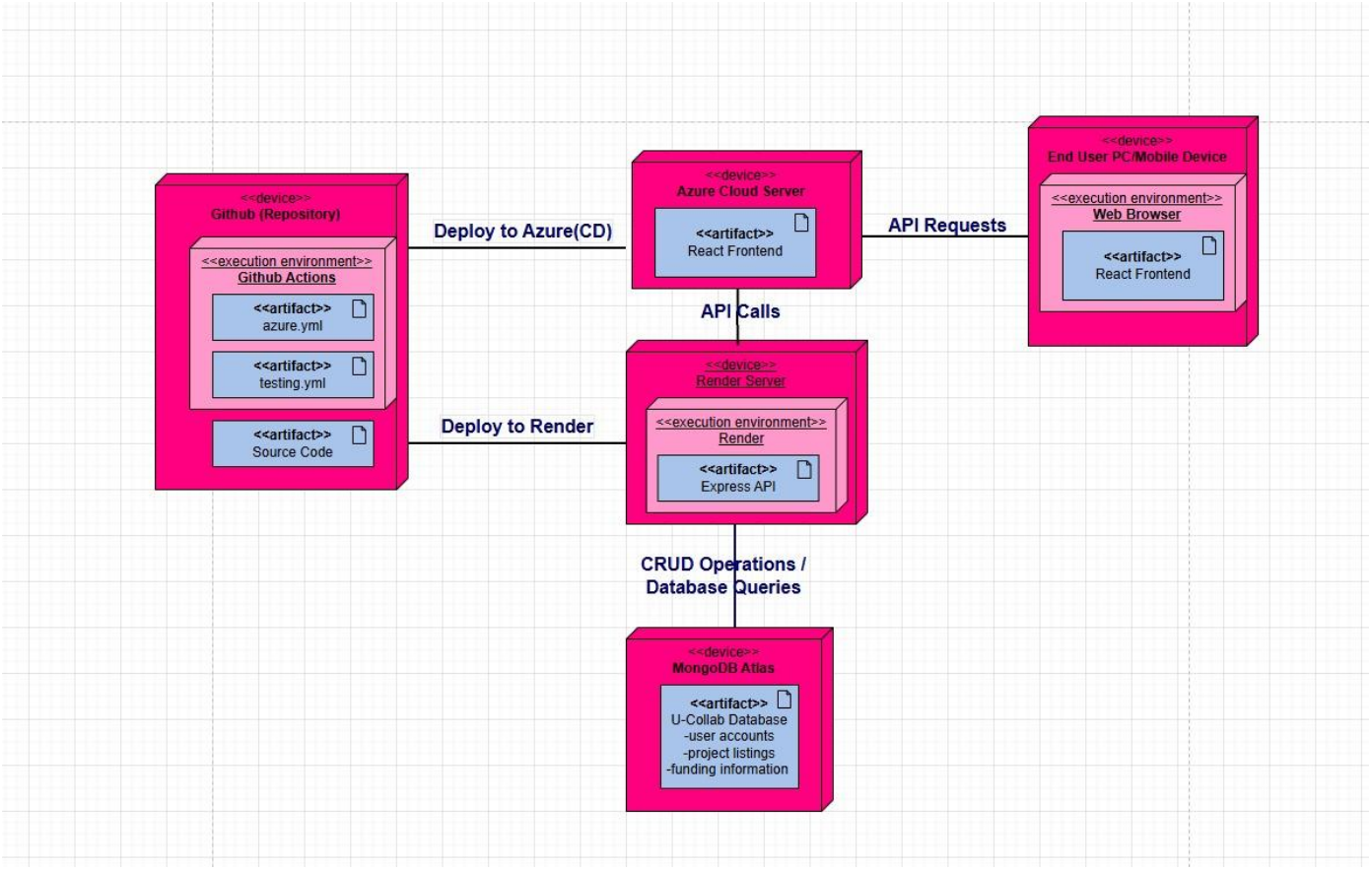


Sequence Diagram



Physical View

Deployment Diagram



Scenarios

Use Case Diagram

Strategic Decisions

During the development of the U-Collab system, a few high-level decisions needed to be put in place. These decisions serve the purpose of aligning with business rules to foster an environment that is manageable, justifying system behaviour where necessary. Some of these decisions are:

- A user may create a project at a later date than the project's start date. The logic behind this is that a researcher who is new to the system may want to add an ongoing project to the system. Restricting the start date of a project to present and future dates will, in this case, result in the data on the system not reflecting the same data as that of the project itself. The team has therefore not included this restriction.
- A user may not upload a document that exceeds 16MB. This helps to reasonably restrict the size of content added to the database, which improves the usability and scalability of the system.

Project Branch Management Strategy

We follow a structured GitHub workflow to maintain code quality, support distributed feature development, error isolation and debugging as well as ensure safe deployments. Our branching model consists of:

1. Main Branch (main)

- Represents the production-ready version of the application.
- Only updated through approved pull requests from the testing branch.
- Protected with branch rules (e.g., no direct pushes, PR reviews required).

2. Testing Branch (testing)

- Serves as the staging area for integration testing.
- New features and bug fixes are merged here before being promoted to main.
- Pull requests into this branch allow us to catch conflicts and run test builds.

3. Feature Branches

- Each major feature/module is developed in its own isolated branch:
    - feature/research-postings
    - feature/collaboration-tools
    - feature/funding-tracking
    - feature/reporting
- Created from the testing branch.

4. Pull/Push Request Workflow

- Developers push code to their respective feature branches.
- Once ready, developers merge into the testing branch.
- Periodically, stable updates from testing are promoted to main.

Conclusion

This project showcased a dynamic, fast-paced Agile development cycle, blending strategic planning with adaptive execution across four intensive sprints. The team demonstrated resilience, learning agility, and a collaborative spirit, overcoming challenges like deployment roadblocks, testing complexities, and evolving client expectations.

Key achievements included the successful deployment of a fully functional system with critical features such as secure authentication, collaborative project management tools, document sharing capabilities, and comprehensive reporting functionalities. The team also achieved an impressive 80% code coverage target, integrated CI/CD pipelines, and leveraged feedback from external users to refine the user experience.

Beyond the technical accomplishments, the project fostered a culture of open communication, continuous improvement, and shared ownership. Each sprint built upon the last, culminating in a Minimum Viable Product that not only met technical requirements but also aligned with the client's vision.

As the project concludes, the team stands proud of its journey—having delivered a robust system while honing valuable skills in software architecture, testing practices, and client collaboration. The lessons learned will undoubtedly inform and empower future endeavours.

The journey may end here, but the spirit of teamwork, innovation, and growth continues—ready to fuel the next big challenge.