Hayden Juday

Design Document

Contents

1. Project Selection

1.1. Description

The program, named GetHub (Name pending), will act as an online chat forum for users within an organization. Users can share code, images, and documents with other users. Other users will be able to view or download any shared files. Users will also be able to:

1.1.1. Rules

1) Leave comments on shared items
2) Report bugs on shared code.
3) Add text or file posts

1.1.2. Roles

Users can receive an additional administration or project manager role. These roles both include all previously mentioned functionality. The Project manager role will be assigned to a specific project. They gain the ability to remove posted content. The administration role inherits all functionality from the project manager role and adds the ability to create projects and add or remove users from projects.

1.2. Constraints

1.2.1. Competitors of the organization

Many competitors function similarly to GetHub. A few examples include GitHub, Stack Overflow, and Discord. The main difference is that GetHub operates within an organization. This scale allows for a more focused and streamlined experience, as users only see posts relevant to their work.

1.2.2. Management support

GetHub requires management to be willing to participate and moderate activity within the forums. Management will be given administration or project manager roles within GetHub, depending on their common areas of operation.

1.2.3. Employee Acceptance

Employee presence is essential to the proper functioning of GetHub. If GetHub is not more accessible or better to use than other alternatives, employees will likely use an alternative.

1.3. Solution Narrative

GetHub will be developed in Java. A database is used to store and retrieve users and posts. A frontend will be developed using JavaFX. GetHub will be completed over four weeks.

1.4. Scope

1.4.1. Main Components

The main components of the system include:

1) User login system that checks the username and password within the database and gives privileges to users according to their stored role in the database.
2) Code posting module. Allow all users to post code and text to the database.
3) Commenting module. Allow all to comment on posts.
4) Post navigation module. A convenient way to navigate and find posts relevant to the user. This will include a browse page and a MyProjects page.
5) Post viewing module. View a selected post and all its comments.
6) Create a project module. Allow Administrators to create projects.
7) Assign/remove the user to the project module. Allow project managers and administrators to add/remove users to a project.

1.4.2. Principal User(s)

The main users for this GetHub will be software development organizations with employee counts of between 10 and 500. Less than ten is not optimal, as GetHub will be developed with many users in mind. More than 500 users per organization may make the user experience cluttered and inefficient.

1.4.3. Organization

GetHub allows easy organization for projects. The MyProjects page allows users to sort posts by their assigned projects, ensuring a streamlined path to the files a user may find relevant.

1.5. Technical Feasibility

1.5.1. Hardware Required

Required hardware includes computers for initial development and servers for the database post-launch. Computers for development are not an issue. Servers will be outsourced to server farms.

1.5.2. Software Required

A Java IDE, as well as database software, will be required. The IDE being used will be Eclipse. MySQL will be used for the database development.

1.5.3. Knowledge and Expertise

The lead developer on this project has completed Database 1 and advanced programming topics at Florida Polytechnic University. The exemplary qualifications of the lead developer say the expertise required for developing this project is present.

2. Formal Requirements

2.1. Problem Synopsis

The goal of this project is to develop an online chat forum that will double as a file management system. Users will be able to upload files, code, and documents. The system will enable other users to view these uploaded items and download or comment on them. The system will include different roles, namely administrators and project managers. Both have their own responsibilities above normal users.

2.2. System Overview

The system consists of the following high-level components:

2.2.1. High-level components

2.2.1.1. User Interface:

Provides a comprehensive platform for users to upload, view, and manage files. Enables users to leave comments and report bugs in files.

2.2.1.2. File Storage:

Stores uploaded files securely on a database. It will support different file types and allow proper organization, upload, and access.

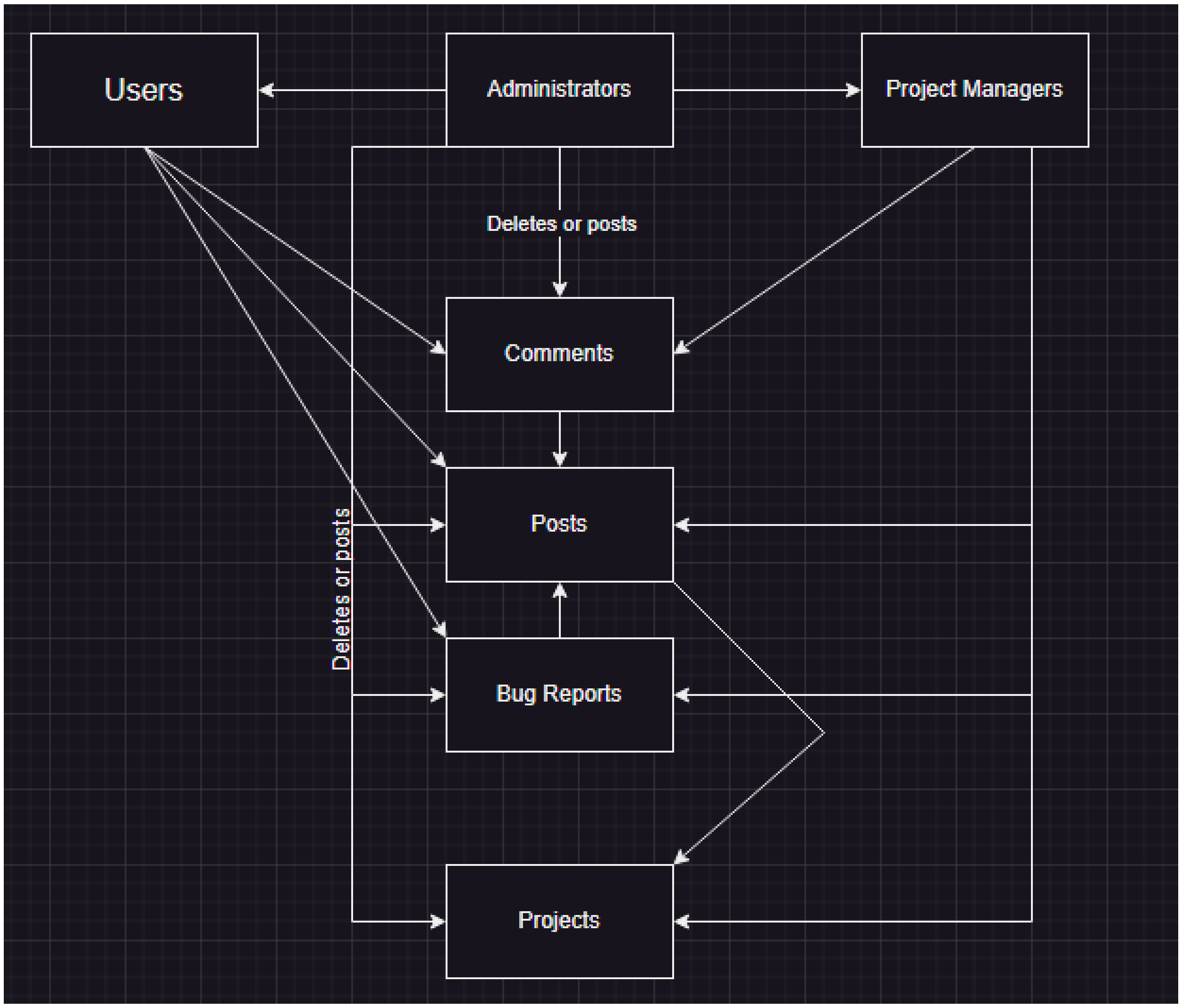
2.2.1.3. User Management:

Handles user authentication, registration, and role allocation. Allows administrators to manage user accounts and permissions.

2.2.1.4. Project Management:

Enables the creation and management of projects. Administrators can assign project managers to specific projects. Allows project managers to add or remove files, comments or bugs from their projects.

2.2.2. Diagram



2.3. System Components

2.3.1. User Interface

2.3.1.1. File Upload:

Allows users to upload files of various types.

2.3.1.2. File View and Download:

Allows users to view and download files.

2.3.1.3. Commenting System:
Allows users to leave comments on projects or posts.

2.3.1.4. Bug Reporting:
Allows users to report bugs on uploaded files.

2.3.2. File Storage
2.3.2.1. File Organization:
Ensures files are properly organized and categorized.

2.3.2.2. File Access Control:
Implements user permissions and access restrictions based on user role.

2.3.3. User Management
2.3.3.1. Authentication:
Authenticates users during login.

2.3.3.2. Role Assignment:
Assigns different roles to users. Only administrators may assign roles.

2.3.3.3. User Account Management:
Allows administrators to add or remove user accounts and roles.

2.3.4. Project Management
2.3.4.1. Project Creation:
Enables administrators to create or delete projects.

2.3.4.2. Project Assignment:
Enables administrators to assign project managers and regular users to specific projects.

2.3.4.3. File, Comment, and Bug Management:
Allows project managers to add or remove files, comments, and bugs within their projects.

2.4. Detailed Requirements
2.4.1. Functional Requirements:
1. Users must be able to register and log in to the system.
2. Users must be able to upload files of various types.
3. Users must be able to view and download files uploaded by other users.
4. Users must be able to post comments on files.

5. Users must be able to report bugs on files.
6. Users must be able to add files, comments, and bug reports to their assigned projects.
7. Administrators must be able to add and remove users.
8. Administrators must be able to create and delete projects.
9. Administrators must be able to add and remove project managers and users from projects.
10. Administrators must be able to add and remove files, comments, and bugs.
11. Administrators must be able to assign roles to other users.
12. Project managers must be able to add and remove files, comments, and bugs within their respective projects.

2.4.2. Non-functional Requirements:

1. The system should respond to the user's actions within 2 seconds.
2. User authentication and file storage will be secure and protected against unauthorized access.
3. The system should be able to handle a growing number of users and files without a significant performance decrease.
4. The user interface should be comprehensive and user-friendly.
5. The system should be functional 99.9% of the time.

2.5. Interface Specification

2.5.1. UI to File Storage:

The UI will show a user-friendly view of all files accessible to the user. User action can send file upload and download requests to the database.

2.5.2. UI to User Management:

The user interface will send authentication and user registration requests to the user management module

2.5.3. UI to Project Management:

The user interface will allow administrators to send requests to create projects and assign users to projects. Additionally, users will be able to retrieve project details and view project files, comments, and bugs.

2.6. Critical Constraints

2.6.1. Uptime:

The system will be accessible 24/7, except for a scheduled downtime of 0.1% for maintenance.

2.6.2. User count:

The system must be able to support at least 100 concurrent users.

2.6.3. Data Security:

The system must ensure data security and integrity to protect files from unauthorized access and accidental deletion.

3. Software Design

The database consists of 7 tables: User, Data, Bug Report, Comments, Project, User_has_Project, and Roles.

3.1. User table

The User table holds user information and has attributes:

- username VARCHAR.
- email VARCHAR.
- password VARCHAR.
- create_time TIMESTAMP.
- Userid INT.

Userid is the primary key and keeps a unique number for each registered user. The create_time attribute marks the time of account creation.

3.1.1. User Relationships

- User has a 1:1 relationship with roles
- User has a 1:n relationship with Data.
- User has a m:n relationship with Project(through user_has_project).
- User also has a 1:n relationship with project for Project_managers.

3.2. Data table

The Data table holds all the posted files and has attributes:

- idData INT.
- file VARBINARY().
- user_Userid INT.
- Project_ProjectID INT

Attribute idData is the primary key and keeps a unique id for each uploaded piece of data. Attribute File stores the uploaded file in binary. Attribute user_Userid is a foreign key holds the Userid of whoever posted the file. Attribute Project_ProjectID is a foreign key from the project table. It allows each file to be assigned to a project and can be null to allow for independent files.

3.2.1. Data Relationships

- Data has a n:1 relationship with User.
- Data has a 1:n relationship with Bug Report.
- Data has a 1:n relationship with Comments.

- Data has a n:1 relationship with Project.

3.3. Bug Report table

The Bug Report table contains bug reports posted on different files and has attributes:

- Data_idData INT
- Bug_report LONGTEXT

Attribute Data_idData is the primary key and is referenced from the Data table so each file can have many bug reports. Attribute Bug_report will store the report in a longtext file.

3.3.1. Bug Report Relationships

- Bug Report has a n:1 relationship with User.

3.4. Comments table

The comments table stores comments made on files by users and has attributes:

- Idcomments INT
- Data_idData INT
- Comment LONGTEXT

Attribute Idcomments provides a unique id for each comment. Attribute Data_idData is a foreign key from the data table to track which file the comment is on. Attribute Comment will store the comment in a longtext file.

3.4.1. Comments Relationships

- Comments has a n:1 relationship with Data.

3.5. Project table

The project table allows for data to be sorted by projects and has attributes:

- ProjectID INT
- Project_manager INT

Attribute ProjectID is a primary key that sorts files into groups. It is referenced in the Data table. Attribute Project_manager is a foreign key taken from Userid in User table. It allows for one project manager to be assigned to each project.

3.5.1. Project Relationships

- Project has a 1:n relationship with Data
- Project has a n:1 relationship with user.
- Project has a m:n relationship with user(through user_has_project).

3.6. User_has_Project table

The User_has_Project table acts as a middleman for the project and user tables and has attributes:

- User_Userid INT
- Project_ProjectID INT

Both attributes are foreign keys from the user and project tables respectively. This allows for one user to be assigned to multiple projects, and for one project to be assigned to multiple users.

3.6.1. User_has_Project relationships

- User_has_project has a n:1 relationship with User.
- User_has_project has a n:1 relationship with Project.

3.7. Roles table

The roles table stores the level of authority each user is assigned and has attributes:

- user_Userid INT
- Roles INT

Attribute user_Userid is referenced from the user table and stores which user the role is assigned to. Roles is the level of authority. A value of 1 is a default user, 2 is a project manager, 3 is an administrator.

3.7.1. Roles Relationships

- Roles has a 1:1 relationship with User.

3.8. Entity-Relationship Diagram

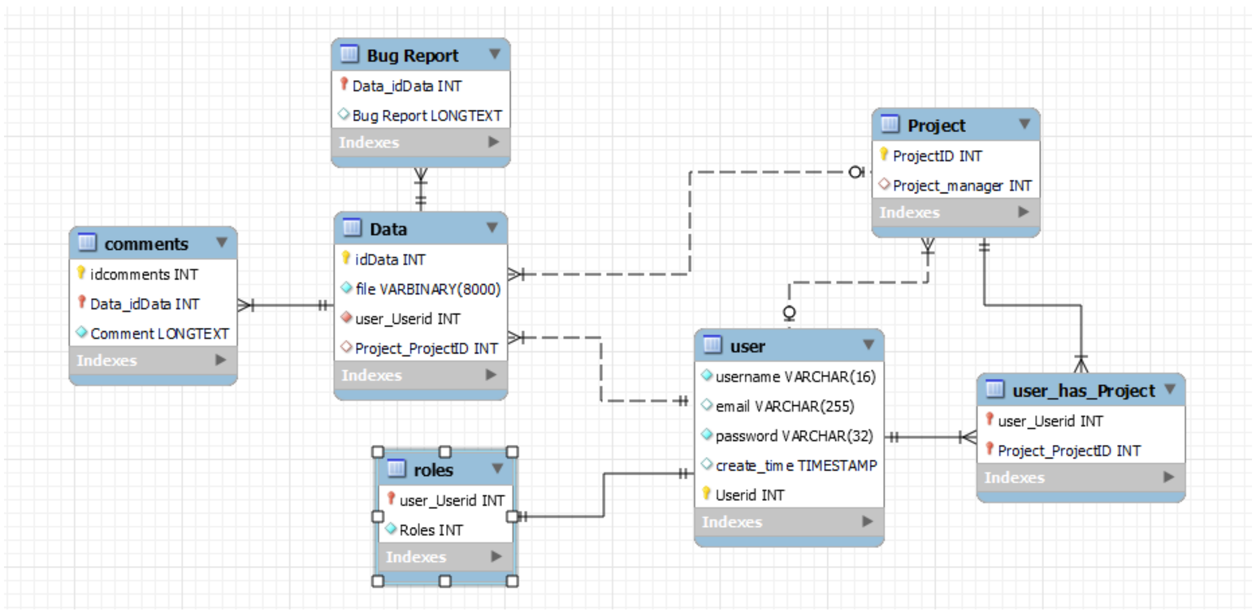
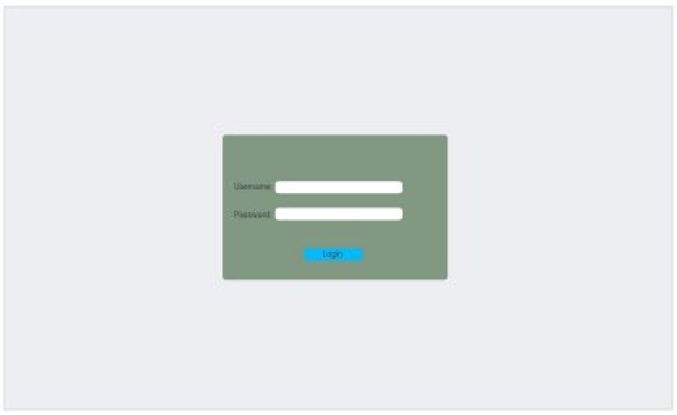
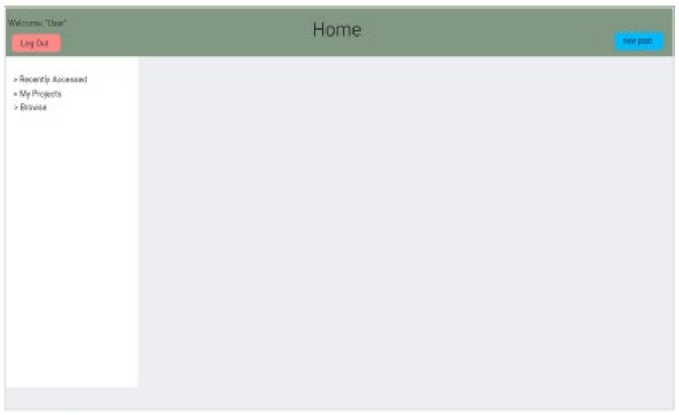


Figure 1 ERD diagram

4. UI Design

4.1. Logging in successfully and moving to home screen.



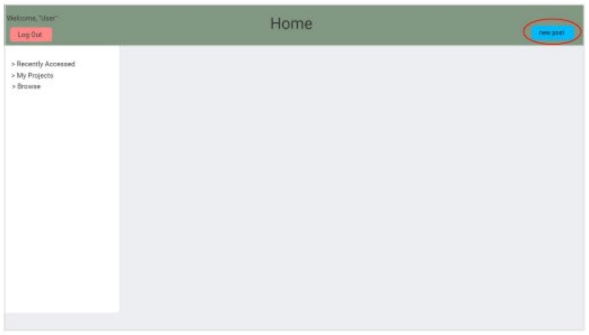


1a Login Screen

Click on login

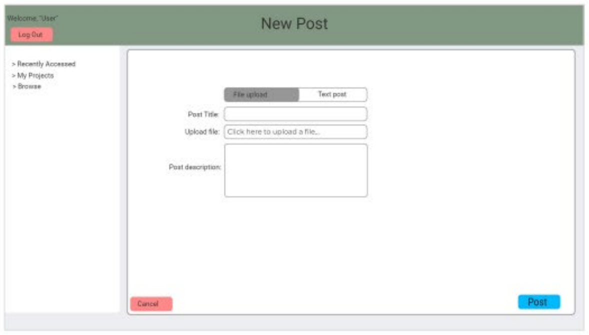
1b Home Screen

4.2. Creating a post



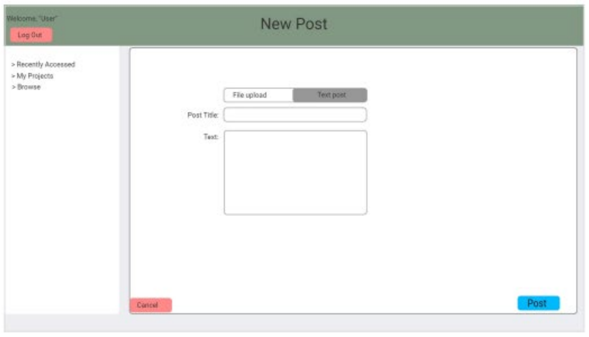
2a Home screen

Create post



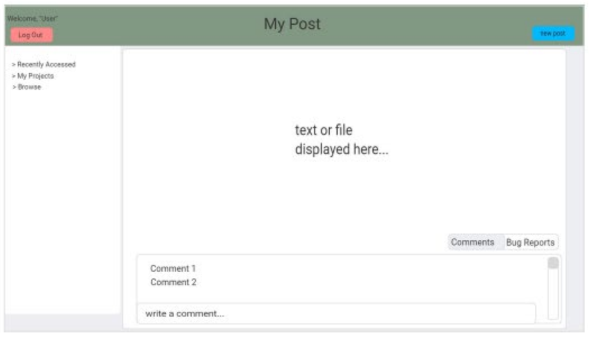
2b New post screen(File upload)

oryboard V1



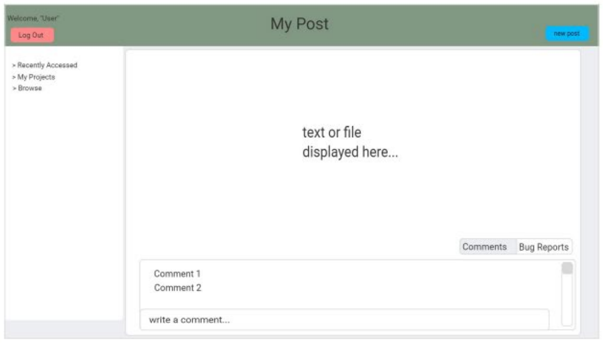
2c New post screen(Text post)

Post clicked

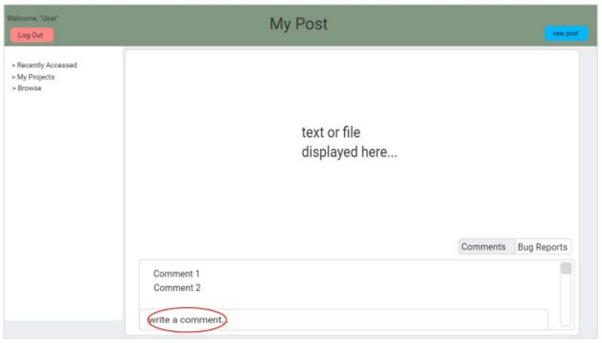


2d Posted!

4.3. Adding a comment to the post



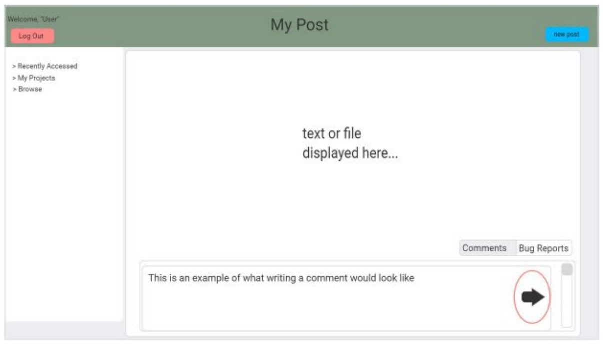
3a Posted!



3b Click to comment

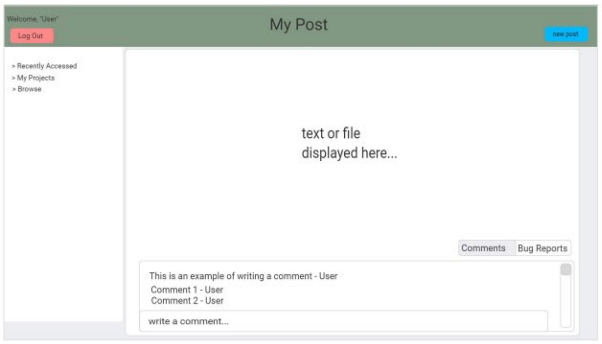
click "write a comment"

yboard V1



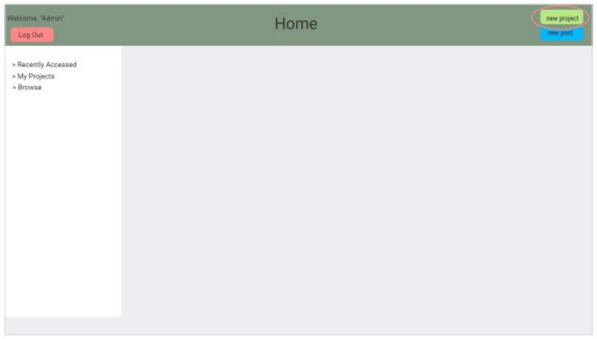
3c Comment writing popup

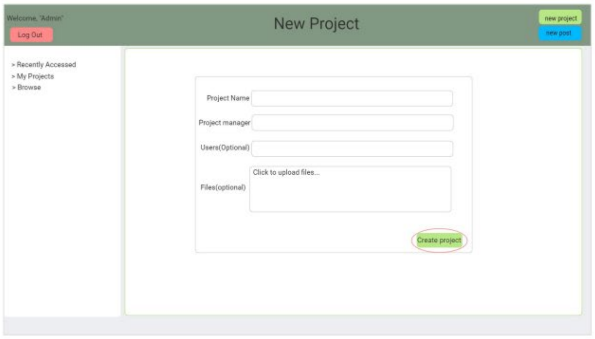
Click arrow to post comment



3d Comment Posted

4.4. Creating a project as an administrator





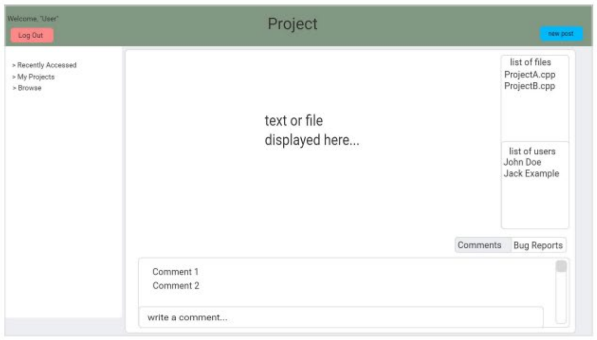
4a Logged in as admin, creating project.

Click new project

4b New project screen

click create project

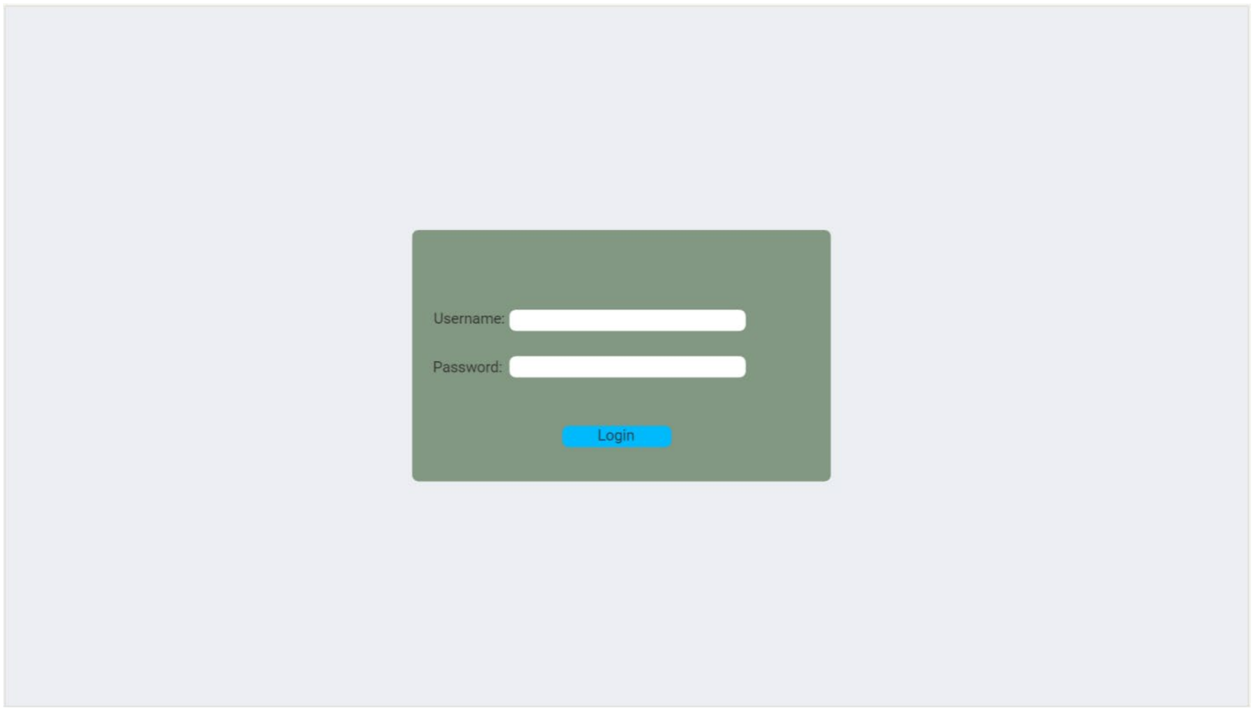
yboard V1



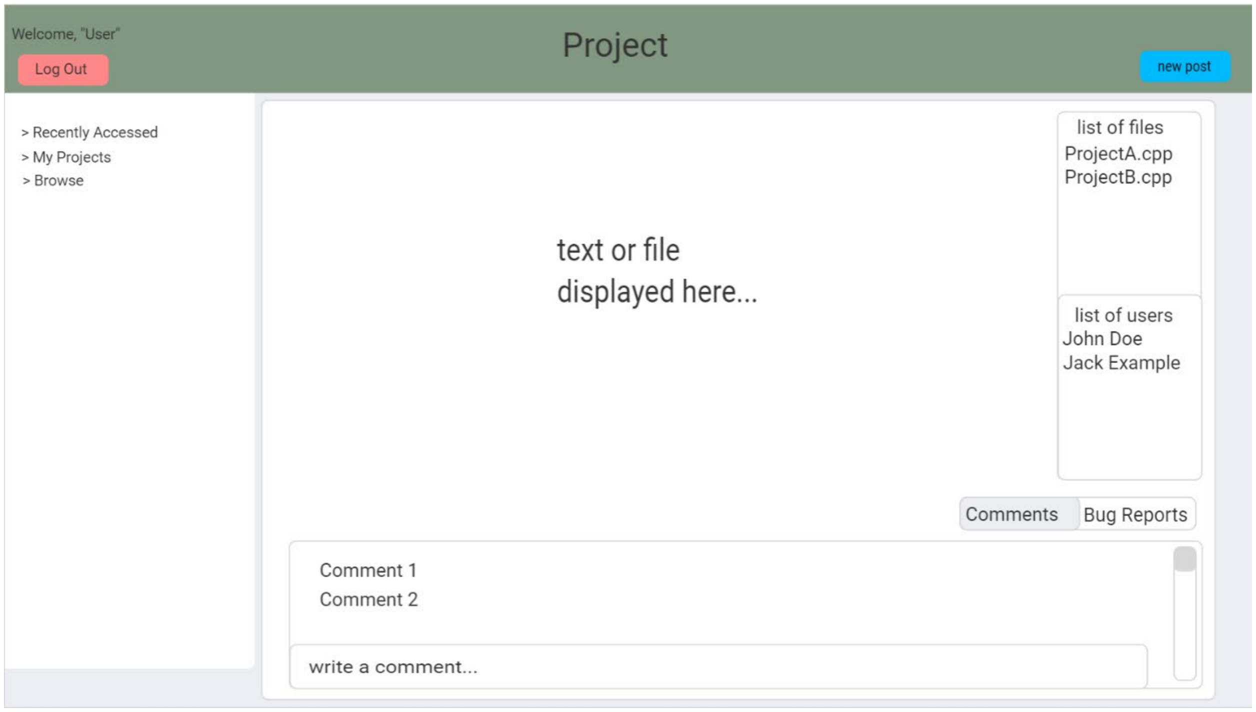
4c Project Screen

5

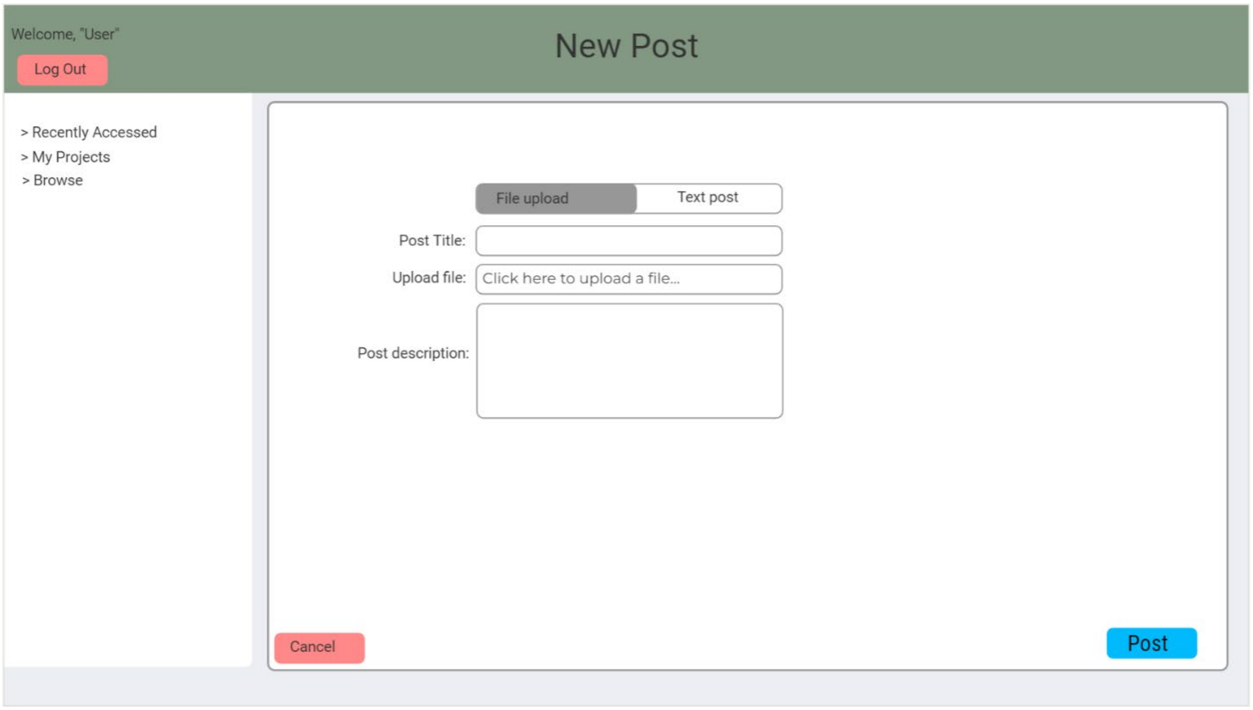
4.5. Detailed Login page



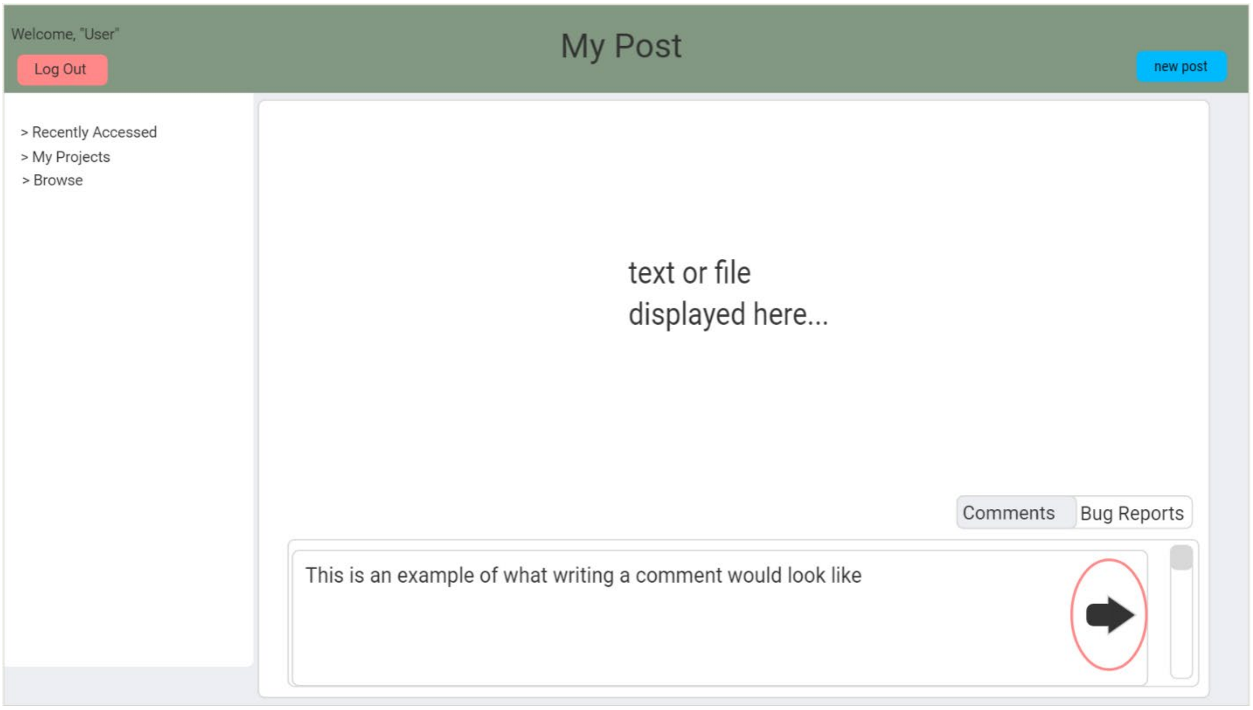
4.6. Detailed Home page



4.7. Detailed Create post page



4.8. Detailed Create comment page.



5. Project Planning

5.1. List of Tasks

Item	Prerequisites
A : User Authentication	-
B : Role System	A
C : File Upload functionality	A
D : File Download functionality	C
E : Commenting System	C
F : Bug Reporting	E
G : Posting System	C,E,F
H : Project Creation functionality	D,G,I
I : Project Assignment functionality	B
J : UI Implementation	H

User authentication is the starting point of our project. Without user authentication, testing any other system would be difficult. The posting system requires file upload, commenting, and bug reports to be fully working. The UI will be the last thing to be implemented and will allow users to easily navigate to files.

5.2. Pert Diagram

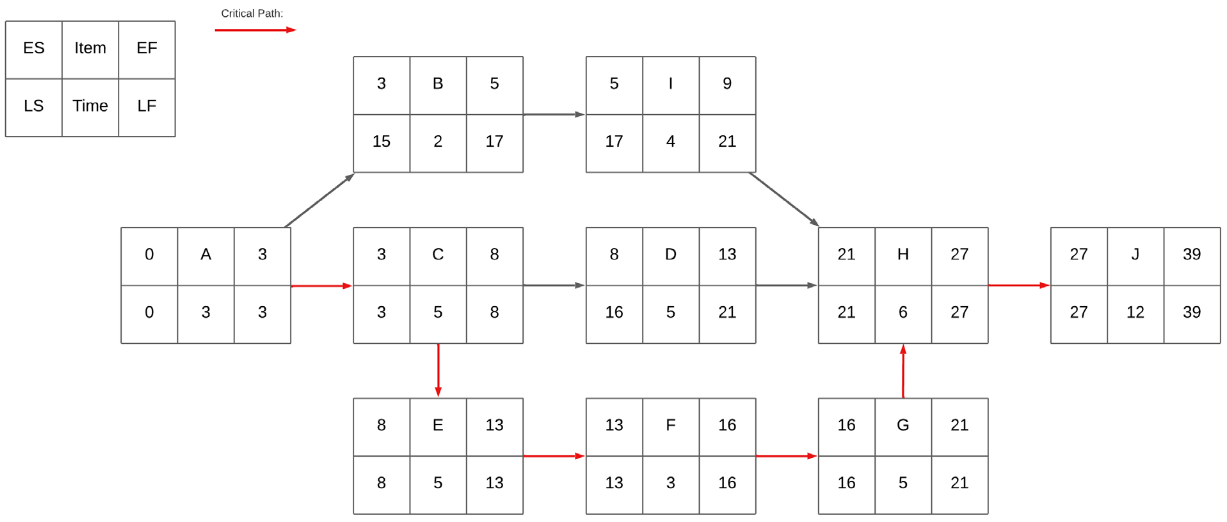
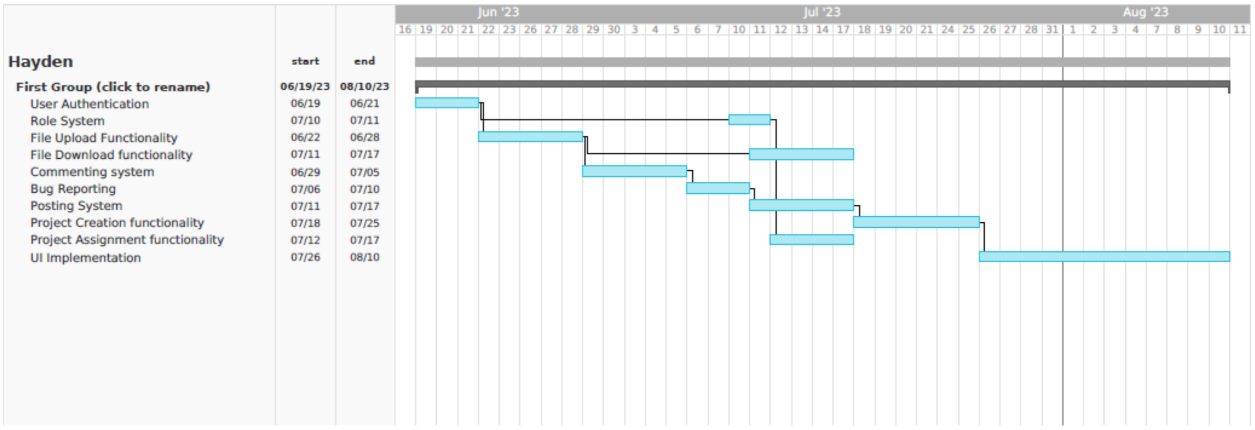


Figure 2: Pert Diagram

The pert diagram shows the order and expected time of development. The Item letter in each box corresponds with the letters in the list of tasks. The critical path is marked by a red arrow. Non-critical paths are marked by black arrows. The total development time is 39 days. UI implementation notably takes up the largest amount of time at 12 days.

5.3. Gantt Chart



Hayden	start	end
First Group (click to rename)	**06/19/23**	**08/10/23**
User Authentication	06/19	06/21
Role System	07/10	07/11
File Upload Functionality	06/22	06/28
File Download functionality	07/11	07/17
Commenting system	06/29	07/05
Bug Reporting	07/06	07/10
Posting System	07/11	07/17
Project Creation functionality	07/18	07/25
Project Assignment functionality	07/12	07/17
UI Implementation	07/26	08/10

Figure 3: Gantt Chart

The Gantt chart gives a visual representation of the project schedule. Note that the chart includes Monday-Friday as working days. The chart exposes a potential issue with the project schedule. The development of components: Items Role system, File Download functionality, Posting system, and Project Assignment functionality would be simultaneous. This could become an excessive workload for the developer. It will be crucial to address this concern to ensure the project stays on track.