

Project 02: Web-Based Application

Final Documentation Report

CSC 530: Software Engineering Methodology
Professor Kim

Group Members:
Angel Bajracharya
Giovanny Joseph
Derek Feliu
Nicole Magpantay
Christopher Miles

Due: 12/6/23

Table of Contents

Project Scope	3
Requirements Specification	3
Access Levels	3
Guest	3
User	3
Verified User	3
Admin	3
Pages and Functionality Requirements	4
login.html	4
resetpassword.html	4
createacct.html	4
menu.html	4
searchresults.html	4
question.html	4
ask.html	4
updateAccount.html	5
admin_panel.html	5
Product Backlog	6
Initial Wireframes	7
Page Control Flow	9
Database Schema Design	10
System Architecture Design	11
API Documentation	11
Test Plan	11
How To Use	12
Github Repository	12
Demo	12

Project Scope

This project involves creating a web-based Q&A application. Users are able to ask, answer, and view questions with varied levels of permissions based on their roles (e.g. guest, user, verified user, admin). The end product is a dynamic interface that integrates with a secure database that allows CRUD operations (e.g. add, edit, request). The website draws inspiration from its predecessors: Stack Overflow, Quora, and Reddit. The target audience for this application is broad and diverse, supporting questions from various topics (e.g. technology, science, art, history, fitness, etc.).

Requirements Specification

Access Levels

Guest

- Can view posts.

User

- Can view posts.
- Can create posts.
- Can edit their own account.

Verified User

- Can view posts.
- Can create posts.
- Can edit their own account.
- Can delete posts.
- Can delete comments.

Admin

- Can view posts.
- Can create posts.
- Can edit their own account.
- Can delete posts.
- Can delete comments.
- Can delete other accounts.
- Can edit the access level of other accounts.

Pages and Functionality Requirements

login.html

Login as guest: Binds to guest account.

Login as user: Checks if user exists in the database and redirects to a role-specific view.

Button: resetpassword.html

Button: createacct.html

resetpassword.html

Parameter expected is username to patch account and update new password.

createacct.html

Parameters expected: first name, last name, email, password, phone number, username

Fill form to make an account

Redirects to menu.html bound with the account.

Validity check: if already logged in, an alert will pop up.

menu.html

Buttons redirecting to:

HOME→searchresults.html (Guest, User, Verified User, Admin)

ASK A QUESTION→ask.html (User, Verified User, Admin)

ACCOUNT INFO→updateAccount.html (User, Verified User, Admin)

ADMIN PANEL→admin_panel.html (Verified User, Admin)

searchresults.html

Loads all posts.

Search bar: allows filtering posts by ID, Title, Genre, Author ID, Posted Date, and Order

Button: question.html (allows user to view post in its full context)

question.html

Loads post in context.

Adding comments allowed for everyone except guests.

ask.html

Allows users with proper access levels (every role except guest) to submit questions.

Fill in the form to ask a question.

Parameters expected: question title, question description, genre, image (optional)

When a post is made, redirects back to searchresults.html.

updateAccount.html

Parameters expected: first name, last name, email, phone number, username
Fill form and press button to modify account details

admin_panel.html

Delete account (Admin).

Delete post (Admin, Verified User).

Delete comment (Admin, Verified User).


Change access level (Admin).

Users with access are able to search for entities and delete or modify.

Product Backlog

PRODUCT BACKLOG				
User Story ID	User Story	Priority	Sprint	Complete?
US001	As an unauthorized user I want to try to login to create a new account or login	1	1	Complete
US002	As a user who forgot their password, I want to reset my password	2	1	Complete
US003	As an authorized user I want to view my own page to see my forums	3	1	Complete
US004	As an authorized user I want to post a question in the forum	4	1	Complete
US005	As an administrator, i want to manage user accounts and forums to ensure security	7	2	Complete
US006	As a user i want to view profiles of other users	5	1	Complete
US007	As an authorized user, I want to upvote and downvote forums that I like/dislike	8	1	Overdue/Incomplete
US008	As a unauthorized or a guest user, I want to be informed of the restrictions on upvoting, downvoting, and replying	9	1	Complete
US009	As an authorized user, I want to reply to forums and ask questions	6	1	Complete
US010	As a user, I want to see multiple forums in a compact form	12	2	Complete
US011	As a user, I want to expand on a forum	13	2	Complete
US012	As a user, I want to be able to edit my account	11	2	Complete
US013	As a verified user, I want to have the same access/features as an authorized user	14	3	Complete
US014	As a verified user, I want to be able to delete posts and comments	15	3	Complete
US015	As admin, I want to have the same access level and features as the verified user and authorized user	16	4	Complete
US016	As an admin I want to be able to delete other accounts	17	4	Complete
US017	As an admin, I want to be able to change the access level of all users	18	4	Complete

Initial Wireframes


 [About Us](#) [Support](#)

Username

Email Address

Password

Already have an account?

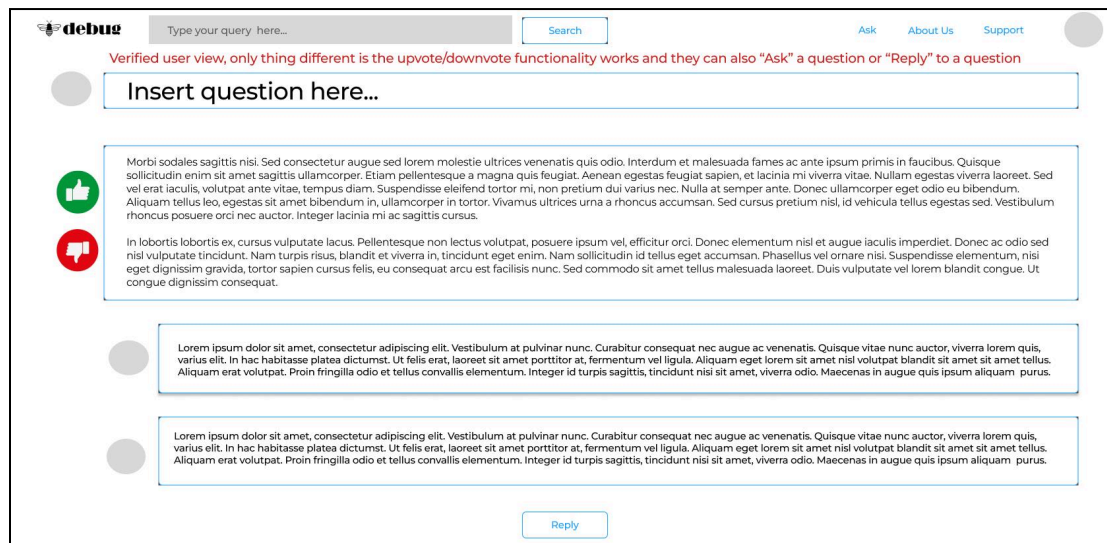
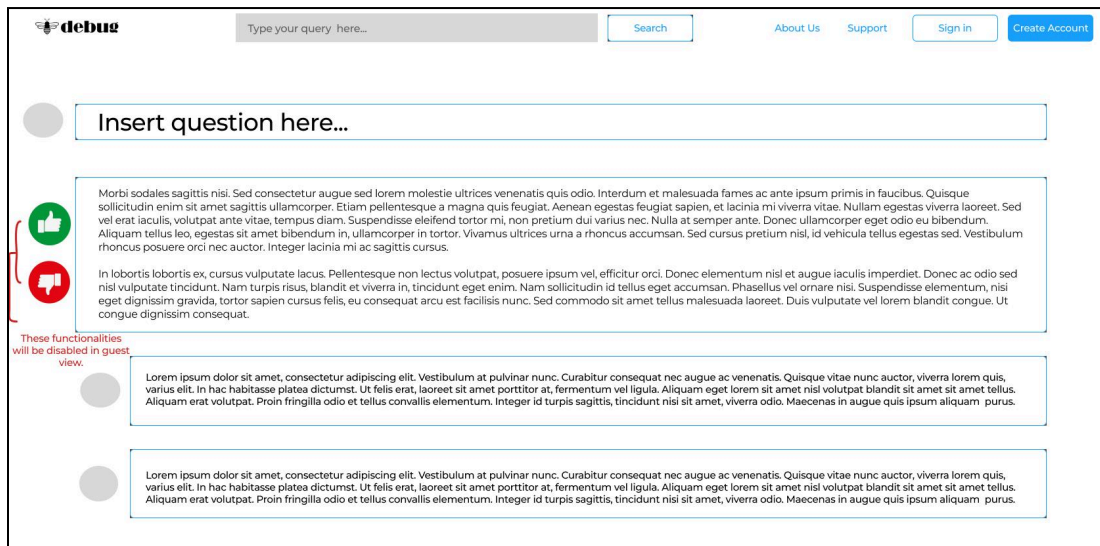
 [About Us](#) [Support](#)

Username/Email

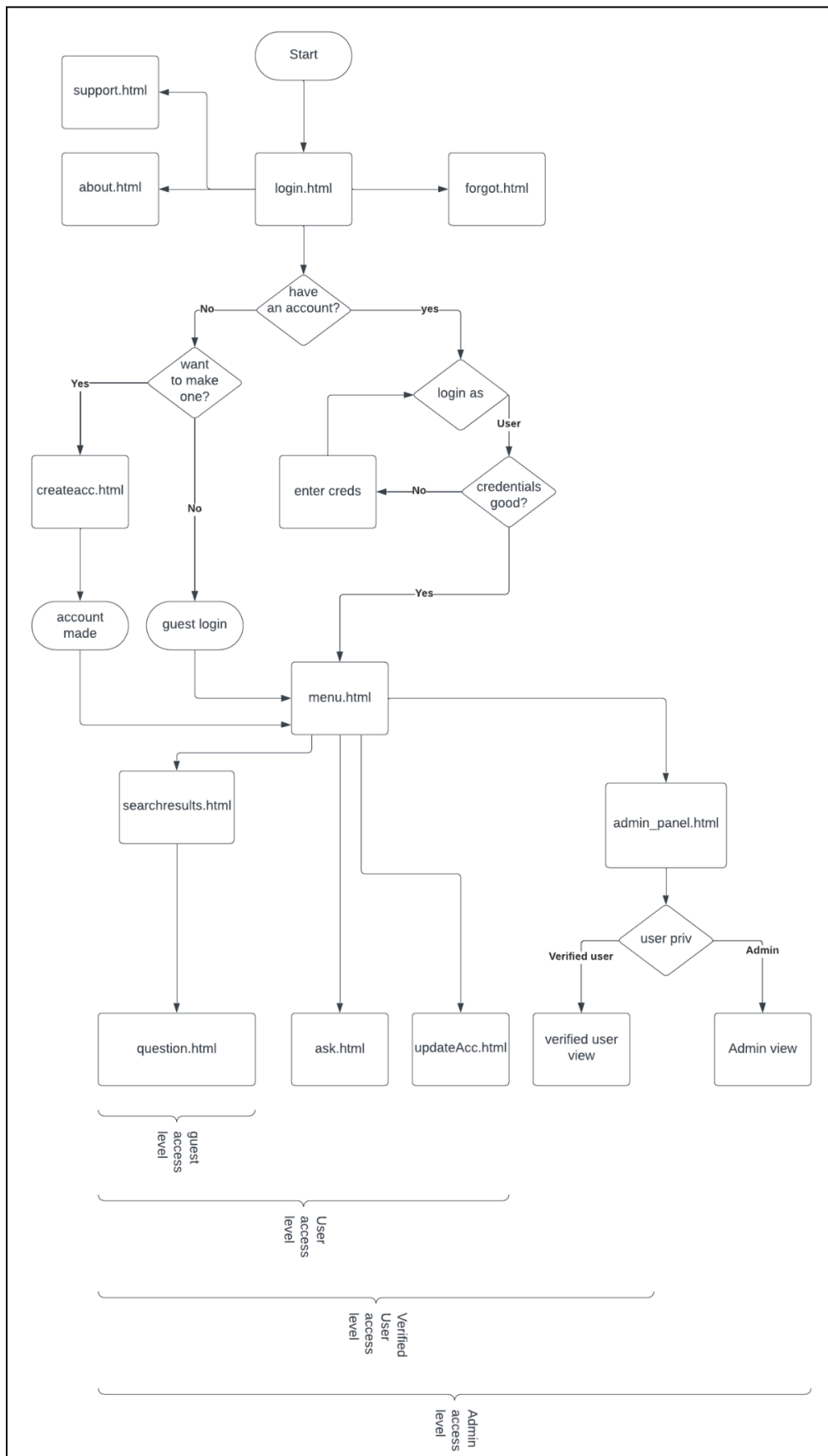
Password

Don't have an account yet?

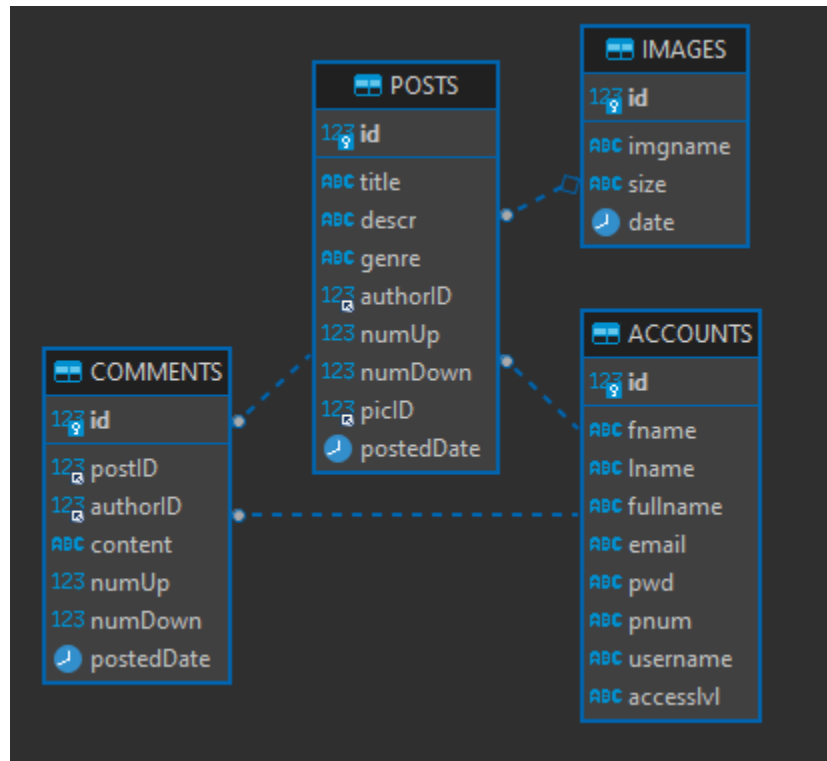
Forgot Password?



Page Control Flow



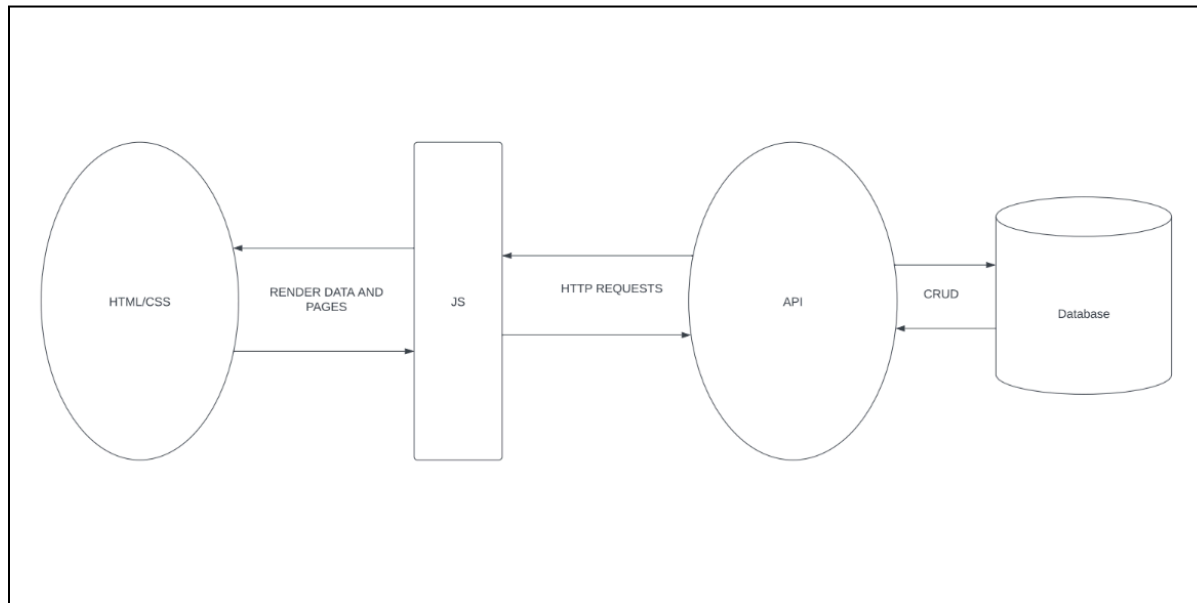
Database Schema Design



Relationships Between Entities:

Users are represented by the accounts table. Users are able to make posts and entities, which are both respective entities. Posts can include images.

System Architecture Design



1. Client-Side (Frontend):
 - a. HTML/CSS:
 - i. Provides scaffolding and styling to display the user interface.
 - ii. Allows user input.
 - b. Javascript:
 - i. Communicates with the backend API to handle dynamic rendering of different data and pages depending on the role of the user.
2. Server-Side (Backend):
 - a. API Server(Go):
 - i. Responds to HTTP requests.
 - ii. API endpoints support functionality for CRUD operations.
 - b. MySQL Database
 - i. Utilizes a database implemented in MySQL.
 - ii. Stores and retrieves data.

API Documentation

[API Doc Link](#)

Test Plan

All of the API endpoints were tested using the Postman application.

How To Use

1. Clone the [repository](#).
2. Open the project in VSCode terminal
3. Start the server by going to the API directory and running main.go
4. Start the webserver at SE-Project2/Frontend/src/login.html using the [live server](#). extension

Github Repository

[Github Link](#)

Demo

<https://youtu.be/sVJt0uni0gQ>