Blog-pipe technical design

CPSC 319 (Team Zenith)

Document Version History

Version 1.0, February 8, 2023

Stakeholder Sign-Off

Evan Seabrook / Alvin Madar (Project Sponsor):

Asem Ghaleb (TA):

Andrew Liu (Project Manager):

Introduction

We will be building a containerized blog application that allows for authentication, commenting, creating, updating, and deleting blog posts.

Additionally, we will build a CI/CD pipeline in the Google Cloud Platform that runs the blog application through an automated testing and deployment process.

The main goal of the system is to use the blog application to showcase the core CI/CD implementation of our solution.

Front End Environment

Framework

React v17.xx will be used. Although React v18.xx is available, support for it is somewhat limited in our CSS frameworks/libraries.

Package Manager

Node Package Manager (npm) 6.14.4

Libraries

React Bootstrap or MUI for CSS. Redux Toolkit for state management. Redux Toolkit provides simplified, opinionated APIs for working with Redux, including creating and managing the store, writing asynchronous logic, and handling action creators. By using Redux Toolkit, we can streamline the development process, reduce boilerplate code, and maintain the integrity and predictability of our application’s state management.

Front End Environment - Continued

IDE

Visual Studio Code 1.75.0 and WebStorm 2022.3.2

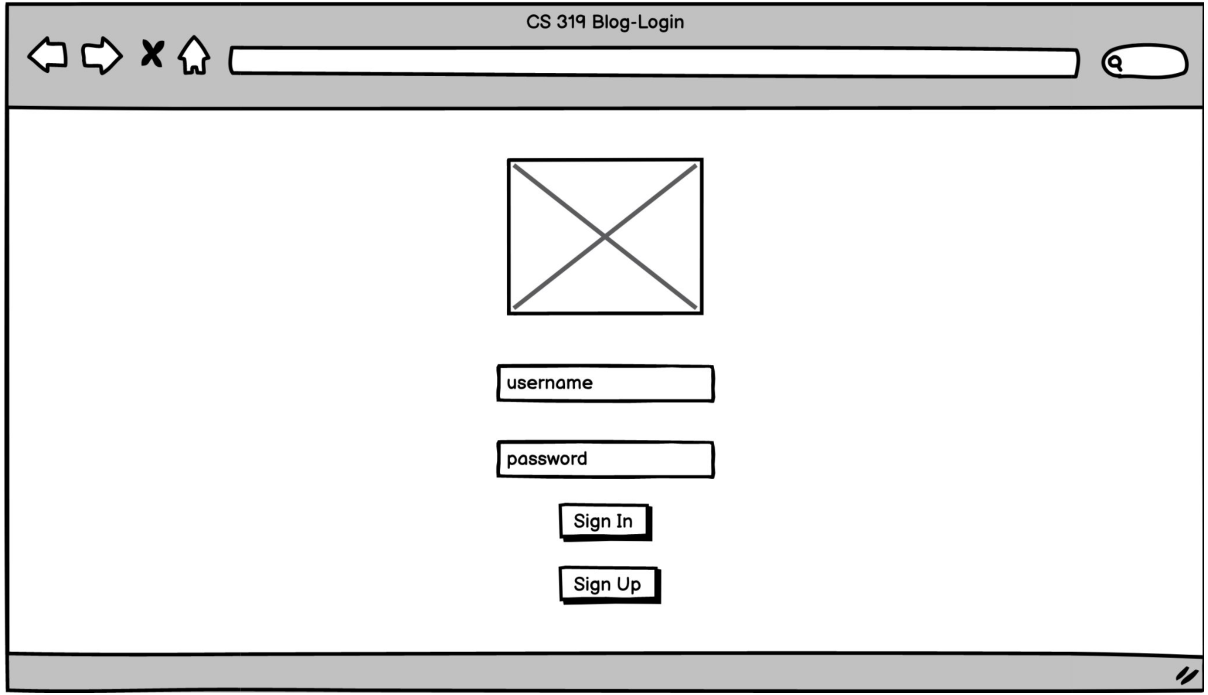
Linter

A linter will help keep code clean and consistent with multiple developers working on the codebase.
ESLint 8.33.0. We will be using the Airbnb configuration (19.0.4)

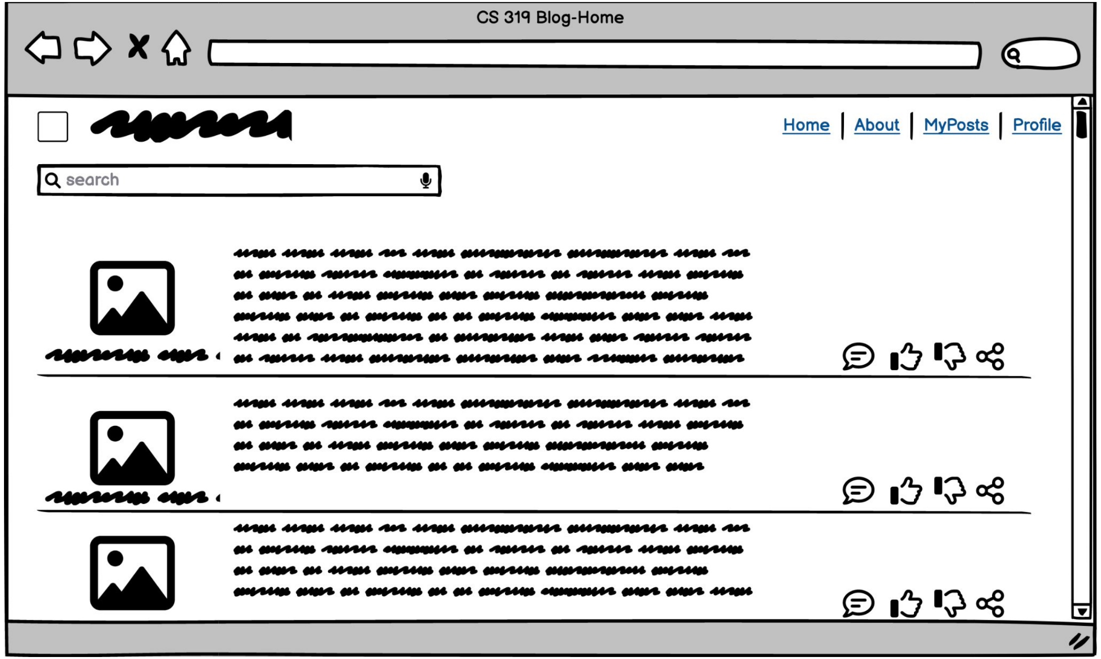
Frontend Testing Framework

Cypress 12.1.0 is an end-to-end testing framework that provides a comprehensive suite of APIs for interacting with a web application, simulating user behaviour, and asserting expected outcomes. Cypress tests run directly in the browser, providing a fast and reliable testing environment.

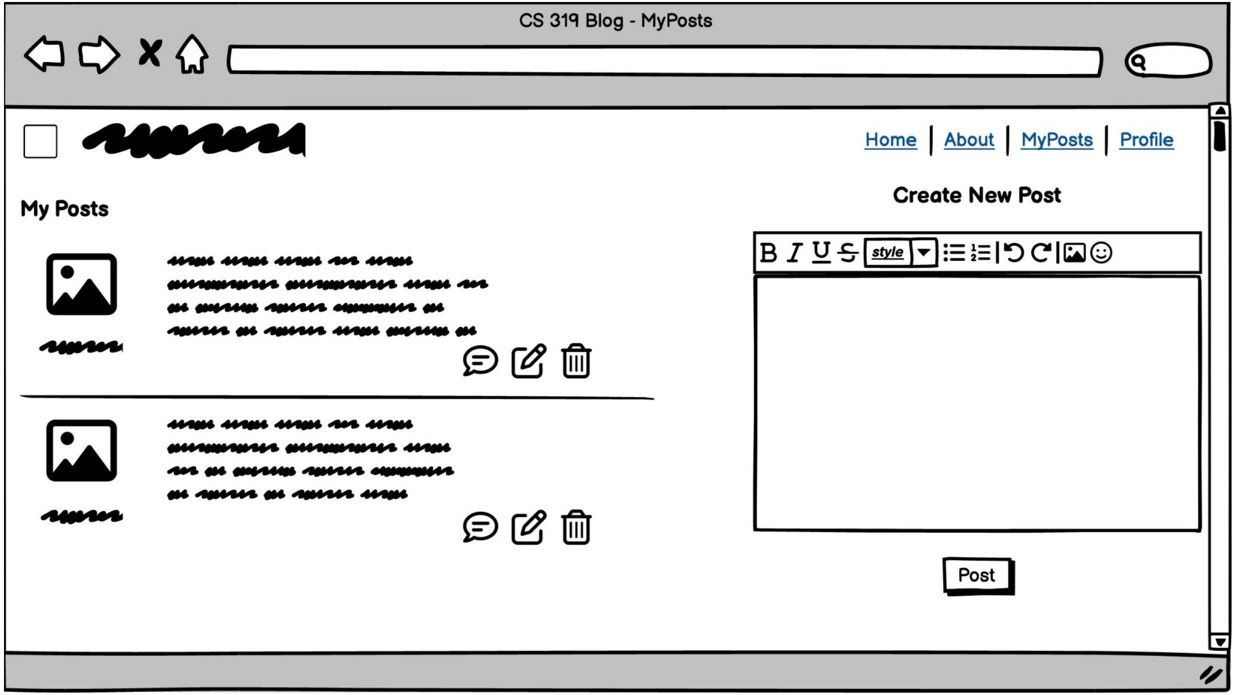
Blog Application Wireframe



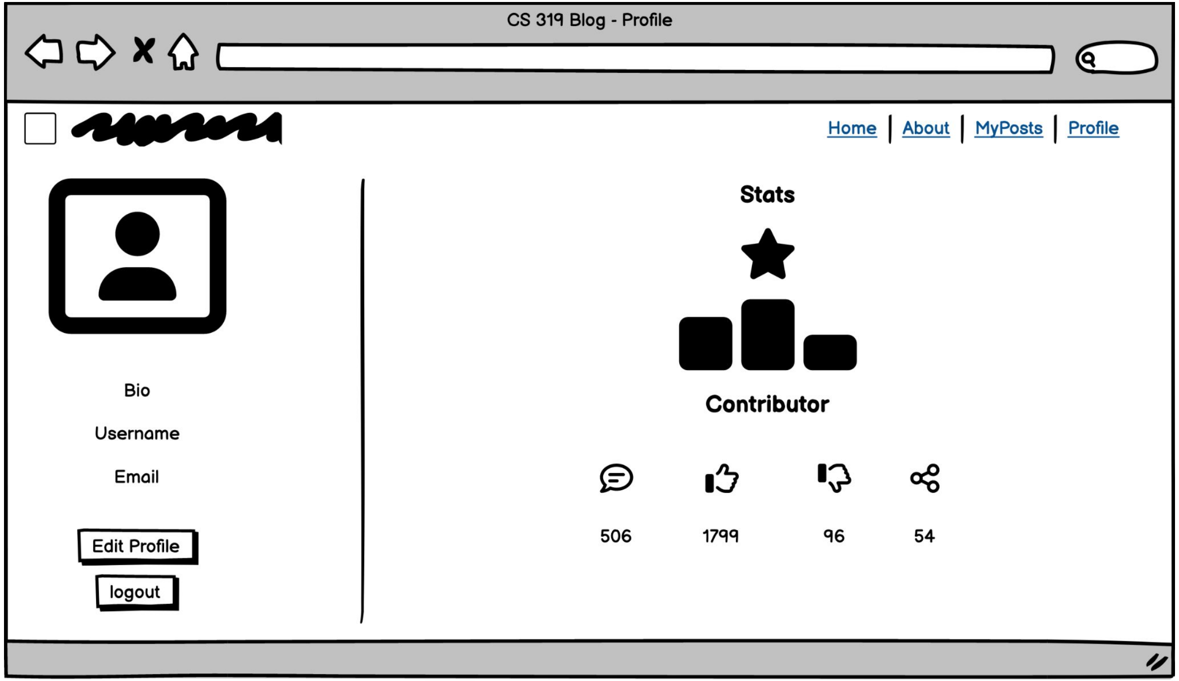
Blog Application Wireframe - Continued



Blog Application Wireframe - Continued



Blog Application Wireframe - Continued



CI/CD Pipeline Environment

Source Controller – git 2.25.1

- Will be utilized to manage source code. Git provides features such branching and merging of source code, allowing the segregation of different versions of the source code and features/bug fixes to be isolated.
- Git also provides a revision history of changes to the source code which allows rollbacks to stable versions in the case of bad deployments
- Benefits of using git are that most developers will be familiar with it due to its widespread adoption and ease of use

Dependency Manager – maven 3.9

- Manages modules/libraries the source code relies on to run.
- Benefits of using maven are that it is compatible with java, which the back end of the project will be built upon. Similarly, to git, maven is one of the most widely adopted dependency managers for Java, so developers are more likely to be familiar with it

Build Tool – JDK 19

- Manages compilation of source code so that it can be run
- Benefits of using JDK include that it is compatible with Java, which the back end will be built upon. Also, JDK is much easier to use than creating make files, which will make developers working on the project more likely to build and test their code incrementally

CI/CD Pipeline Environment - Continued

Test Tools – JUnit 5.7.1

- Framework upon which tests may be built. Supports Unit testing and integration testing for the back-end of the project
- Benefits of using JUnit include developers not needing to create their own testing frameworks from scratch, which will make it easier and more likely for developers to write tests for their code. Also, JUnit is among the most widely adopted testing frameworks for the Java language, so it is likely for Java developers to already be familiar with and is easy to learn if they do not.

Container Manager – Docker 20.10.22

- Saves images of built source code and its dependencies, allowing the project to run similarly on any kind of machine
- Allows project to only need to be built once throughout the entire pipeline
- Benefits of Docker include that it is widely adopted, so most developers are familiar with it

Communication Medium – Slack

- Messaging platform teams within a business to communicate
- Will be used as a destination for the results of build tests

CI/CD Pipeline Environment - Continued

Test Runner/ Deployment Manager – Jenkins 2.375.3

- Provides a standardized build and test environment to assure changes to the source code run correctly, independent of their local machines
- Conditionally deploys source code to the respective environment conditionally, depending on the results of the testing phase which both lowers the risk of deployment and allows for deployment to occur at a much faster rate
- Benefits of Jenkins include that it is free to use because it is open source. Jenkins also offers vast customization
- Cons of Jenkins is that it is more difficult and time consuming to set up

Or

Test Runner/ Deployment Manager – Google Cloud Platform (GCP)

- Provides a standardized build and test environment to assure changes to the source code run correctly, independent of their local machines
- Conditionally deploys source code to the respective environment conditionally, depending on the results of the testing phase which both lowers the risk of deployment and allows for deployment to occur at a much faster rate
- Benefits of GCP as a test runner / deployment manager include direct integration into GCP, which is already being used as a deployment environment

Cloud Technology – Google Cloud Platform (GCP)

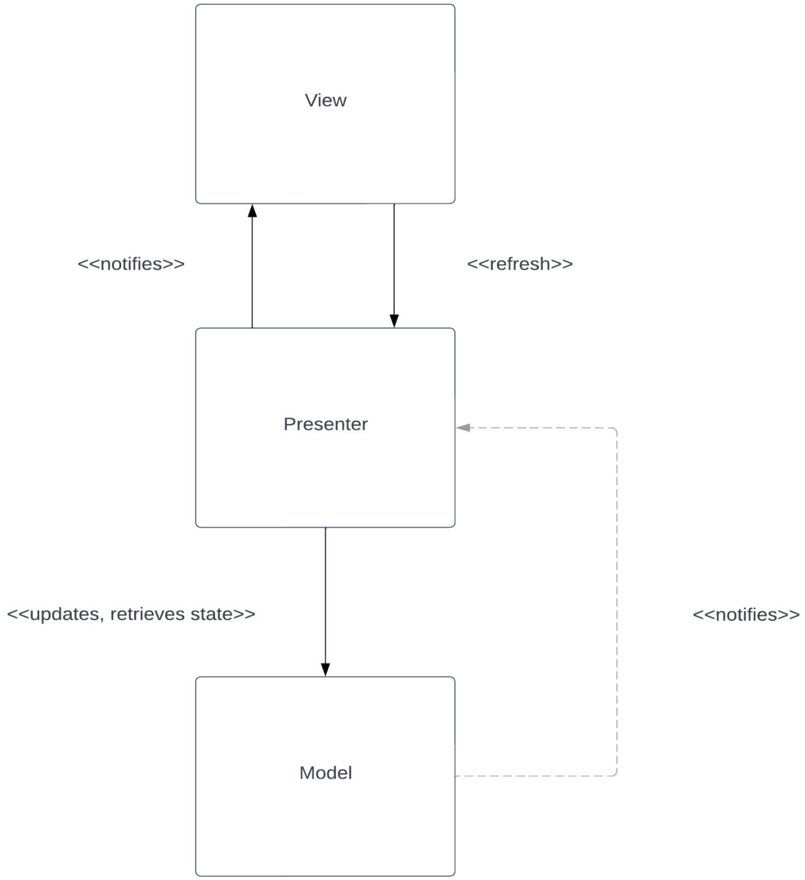
- Offers globally accessible servers as a deployment environment
- Benefits of GCP include low cost to maintain, security and reliability of Google services, and built in tools to keep track of analytics for the deployment environment

Production and Testing Environment

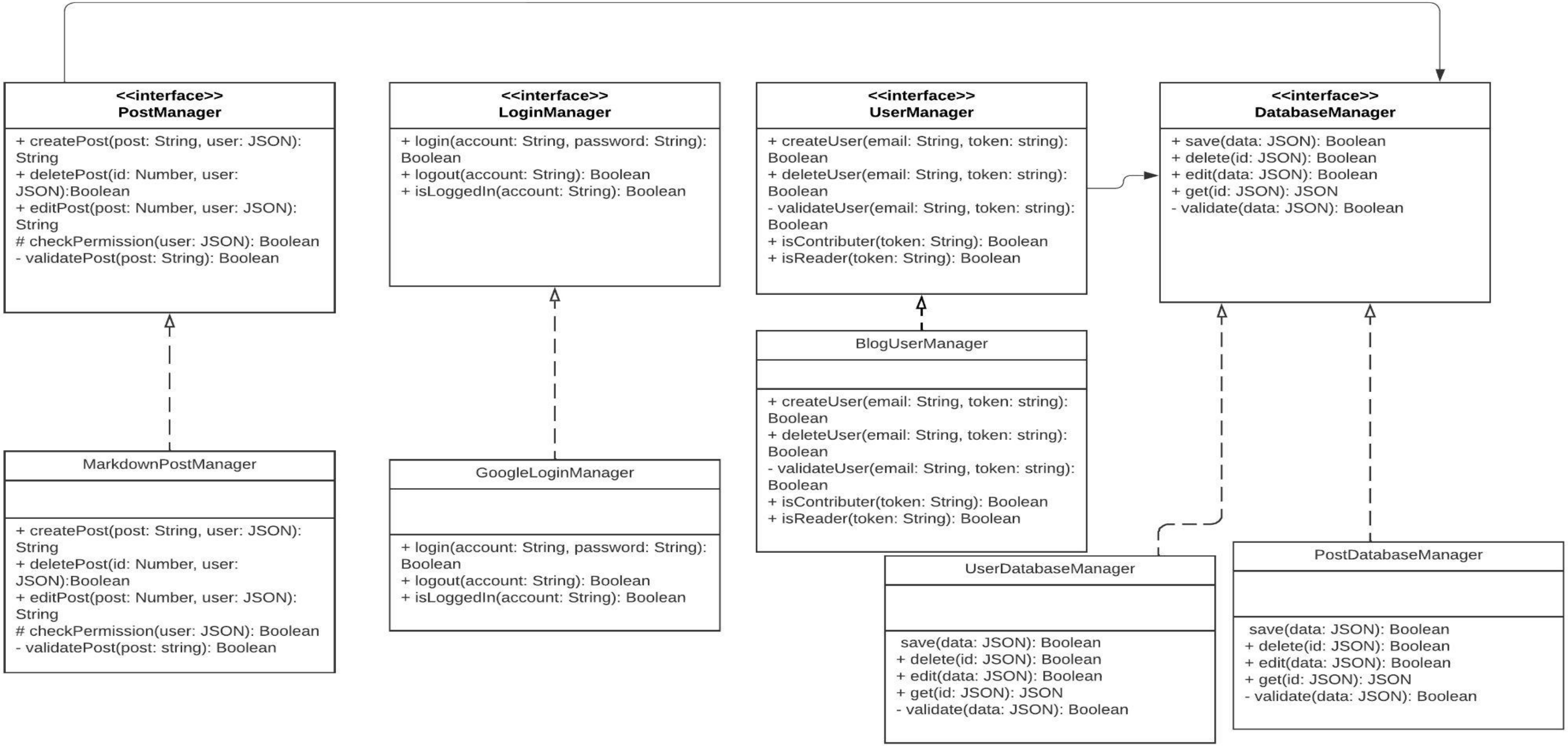
Production: Two websites and a Slack bot are included.
Blog: accessible to the public and serves live traffic, providing functionalities such as creating blogs, calling tests by commands, interacting with other developers, and deleting blogs.
Testing: This environment is only available to developers and mainly serves the purpose of conducting tests through its interface. It will also be responsible for visualizing the CI/CD progress and pipeline.
Slack bot: The Slack bot is for developers and will print out the developer and changes made every time there is a major update.

Test: The testing environment will be wrapped in the testing website. The fundamental testing setup will reference JUnit testing framework. JUnit 5.7.1 will be used for the backend (Java). Tests with Cypress 12.1.0 and exploratory testing will be included for frontend (JavaScript).

Meta Design Pattern (UML)



Backend Design (UML)



<<interface>>
PostManager

+ createPost(post: String, user: JSON): String
+ deletePost(id: Number, user: JSON):Boolean
+ editPost(post: Number, user: JSON): String
# checkPermission(user: JSON): Boolean
- validatePost(post: String): Boolean

<<interface>>
LoginManager

+ login(account: String, password: String): Boolean
+ logout(account: String): Boolean
+ isLoggedIn(account: String): Boolean

<<interface>>
UserManager

+ createUser(email: String, token: string): Boolean
+ deleteUser(email: String, token: string): Boolean
- validateUser(email: String, token: string): Boolean
+ isContributer(token: String): Boolean
+ isReader(token: String): Boolean

<<interface>>
DatabaseManager

+ save(data: JSON): Boolean
+ delete(id: JSON): Boolean
+ edit(data: JSON): Boolean
+ get(id: JSON): JSON
- validate(data: JSON): Boolean

MarkdownPostManager

+ createPost(post: String, user: JSON): String
+ deletePost(id: Number, user: JSON):Boolean
+ editPost(post: Number, user: JSON): String
# checkPermission(user: JSON): Boolean
- validatePost(post: string): Boolean

GoogleLoginManager

+ login(account: String, password: String): Boolean
+ logout(account: String): Boolean
+ isLoggedIn(account: String): Boolean

BlogUserManager

+ createUser(email: String, token: string): Boolean
+ deleteUser(email: String, token: string): Boolean
- validateUser(email: String, token: string): Boolean
+ isContributer(token: String): Boolean
+ isReader(token: String): Boolean

UserDatabaseManager

save(data: JSON): Boolean
+ delete(id: JSON): Boolean
+ edit(data: JSON): Boolean
+ get(id: JSON): JSON
- validate(data: JSON): Boolean

PostDatabaseManager

save(data: JSON): Boolean
+ delete(id: JSON): Boolean
+ edit(data: JSON): Boolean
+ get(id: JSON): JSON
- validate(data: JSON): Boolean

Cloud SQL Data model for blog

TABLE User
user_ID VARCHAR(255)
username VARCHAR(255)
creation_date CHAR(30)
last_active CHAR(30)
profile_picture VARCHAR(255)
bio VARCHAR(1000)
user_level TINYINT
is_deleted BOOLEAN

(**bold** for primary keys,
italics for foreign keys)

TABLE Post
post_ID INTEGER
user_ID VARCHAR(255)
title VARCHAR(255)
ontent VARCHAR(20000)
creation_date CHAR(30)
last_modified CHAR(30)
upvotes INTEGER
downvotes INTEGER
views INTEGER
is_deleted BOOLEAN
allow_comments BOOLEAN
thumbnail_url VARCHAR(255)

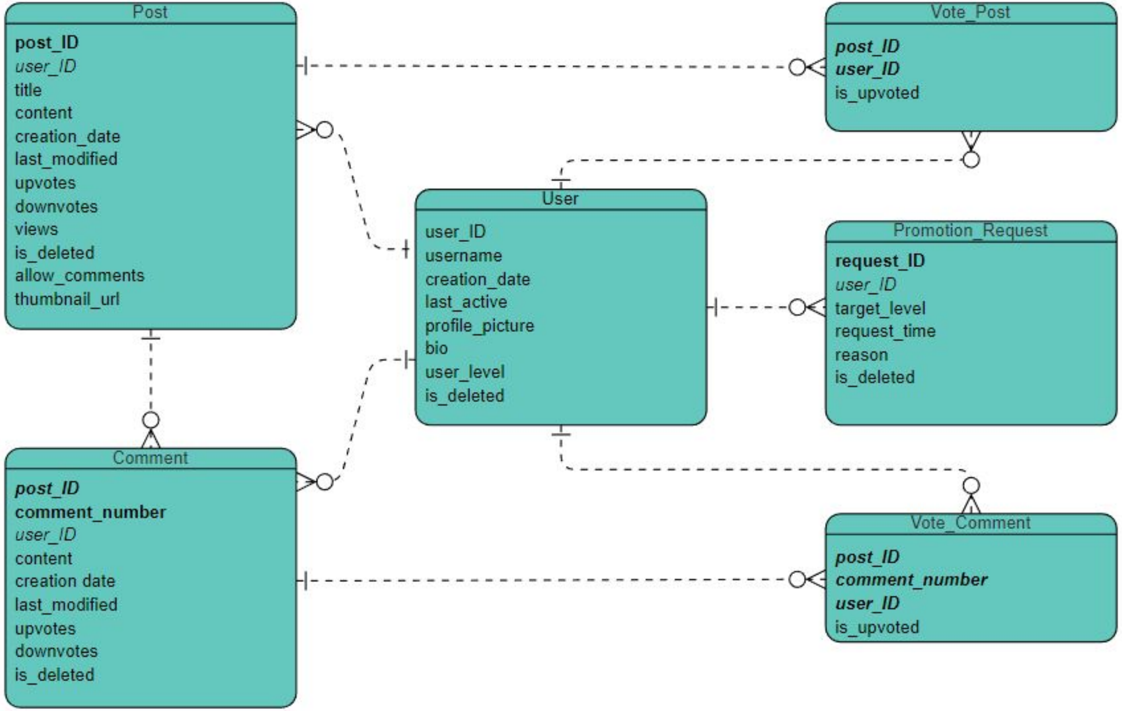
TABLE Comment
post_ID INTEGER
comment_number INTEGER
user_ID VARCHAR(255)
content VARCHAR(10000)
creation_date CHAR(30)
last_modified CHAR(30)
upvotes INTEGER
downvotes INTEGER
is_deleted BOOLEAN

TABLE Vote_Post
post_ID INTEGER
user_ID VARCHAR(255)
is_upvoted BOOLEAN

TABLE Vote_Comment
post_ID INTEGER
comment_number INTEGER
user_ID VARCHAR(255)
is_upvoted BOOLEAN

TABLE Promotion_Request
request_ID INTEGER
user_ID VARCHAR(255)
target_level TINYINT
request_time CHAR(30)
reason VARCHAR(500)
is_deleted BOOLEAN

Cloud SQL Data model for blog (ER Diagram)



Post

post_ID
user_ID
title
content
creation_date
last_modified
upvotes
downvotes
views
is_deleted
allow_comments
thumbnail_url

Vote_Post

post_ID
user_ID
is_upvoted

User

user_ID
username
creation_date
last_active
profile_picture
bio
user_level
is_deleted

Promotion_Request

request_ID
user_ID
target_level
request_time
reason
is_deleted

Comment

post_ID
comment_number
user_ID
content
creation date
last_modified
upvotes
downvotes
is_deleted

Vote_Comment

post_ID
comment_number
user_ID
is_upvoted

DEPRECATED Blog API design

Endpoint	Method	Description	Request Body	Success Response	Failed Response
/posts	GET	Retrieve a list of all blog posts	N/A	Array of blog post objects	N/A
/posts/{id}	GET	Retrieve a single blog post by ID	N/A	Blog post object	400 for bad request 404 not found
/posts	POST	Create a new blog post	Blog post object	Blog post object with ID	400 for bad request 401 for unauthorized user 403 for insufficient permissions
/posts/{id}	PUT	Update an existing blog post	Blog post object	Blog post object	400 for bad request 401 for unauthorized user 403 for insufficient permissions
/posts/{id}	DELETE	Delete a blog post	N/A	N/A	400 for bad request 401 for unauthorized user 403 for insufficient permissions
/posts/{id}/comments	GET	Retrieve a list of comments for a blog post	N/A	Array of comment objects	400 for bad request 404 not found
/posts/{id}/comments	POST	Create a new comment for a blog post	Comment object	Comment object with ID	400 for bad request 401 for unauthorized user 403 for insufficient permissions

BLOG API

See *Blog API.pdf*.

CI/CD pipeline API design

Endpoint	Method	Description	Request Body	Success Response	Failed Response
/build	POST	Triggers the build process.	repository URL, branch, and build configuration.	202 accepted	400 for bad request 401 for unauthorized user 403 for insufficient permissions
/test	POST	Triggers the test process.	build artifacts and test configuration.	202 accepted	400 for bad request 401 for unauthorized user 403 for insufficient permissions
/deploy	POST	Triggers the deployment process.	configuration.	202 accepted	400 for bad request 401 for unauthorized user 403 for insufficient permissions
/status/{id}	GET	Retrieves the status of the CI/CD pipeline. Returns information about the current state	Build_id/test_id/deploy/id	200 ok	400 for bad request 401 for unauthorized user 403 for insufficient permissions
/config	POST	Manages the configuration of the CI/CD pipeline.	environment, build, test, and deployment configurations.	200 ok	400 for bad request 401 for unauthorized user 403 for insufficient permissions

Trade-off

Flexibility vs Scalability: the blog content as a formatted HTML string in a database can make it more difficult to read and modify. When the content is retrieved from the database, it must be parsed in order to be displayed or modified, which can require more effort and processing power.

Additionally, storing formatted HTML in a database can make it more difficult to search and manipulate the content, as special characters and formatting tags must be taken into account when performing database operations.

Concrete vs Abstract: If we utilize the interface, the code will be more abstract and hard to read. In return, the code will be more extensible

Speed vs Quality: trivial, because we aim to achieve the best quality in this project.

User Experience vs Security: Since we will not reference any of PWC data, the quantity of test from PWC side the final product will be impacted to compromise the protection of sensitive data, so that makes the application less usable.

Potential Risks of not meeting system requirements

Potential Risks of not meeting system requirements:

Not meeting the system requirements for a blog application can result in various potential risks, including:

1. **Performance Issues**: The application may run slowly or not function as expected, leading to a poor user experience.
2. **Security Vulnerabilities**: The application may be more susceptible to security threats and data breaches if it is not configured properly.
3. **Compatibility Problems**: The application may not work properly on certain devices or platforms if the system requirements are not met.
4. **Technical Support Difficulties**: If the system requirements are not met, technical support may not be able to resolve issues with the application effectively.
5. **User Dissatisfaction**: If the application does not perform as expected, users may become frustrated and stop using the app, leading to a loss of traffic and potential customers.

It is therefore important to carefully review and meet the system requirements for a blog application to minimize these risks and ensure the best possible experience for the users.