Software Design Document

StudyBoard

Team Members

Pyungkang Hong	UI/UX Design	pyungkang.hong@stonybrook.edu
Hawon Park	PM	hawon.park@stonybrook.edu
Jeong Ho Shin	LP	jeongho.shin@stonybrook.edu

Last Modified: October 20, 2021

Table of Contents

1. Introduction

1.1 Product description

People often have questions that they need answering. There might be a specific term that they don't understand or a topic that they are trying to review. More often than not, these people turn to the internet to try and find an answer. They might be able to find what they are looking for in five minutes, or they might also end up spending hours without much success. Even if they can find answers to their questions, these answers might be vague, factually incorrect, or too poorly explained to understand.

StudyBoard aims to prevent such problems by providing a web platform where anyone with a question in mind can post their thoughts and receive clear and instructional answers. Users would also be able to interact with one another by leaving replies and sharing posts. Users would also be able to like content that they see, be it question or reply, that would feed into a ranking system where replies with higher ratings would be displayed first. Naturally, users would also be able to flag any inappropriate or incorrect questions and replies. Such posts or their replies would be examined and removed if needed. This would ensure that no one leaves the platform with the wrong information in hand. Having a unified search bar would also help users peruse through any subject of their choosing.

These features, when combined, would facilitate active discussion and improved learning by enabling each person to leave with the exact information that they were looking for.

1.2 Scope

StudyBoard is a web platform that aims to help users get replies to their questions. This application allows users to post, reply to, and share questions. This platform also allows users to vouch for specific responses by liking the replies. Users can increase in rank via the number of likes that their replies have.

When a piece of content seems inappropriate, users may flag the content to be reviewed by the moderators. While StudyBoard does not automatically verify the factuality of a reply, the user interactions of liking and flagging content would sift out inaccurate replies over time. Moderators would review any reported content and decide whether to accept or reject the report as valid.

StudyBoard is not limited to any specific group of people but is open to anyone with a question in mind. StudyBoard would be optimized to be a responsive, cross-platform web application supported by any modern web browser.

1.3 Users

StudyBoard's primary target group is anyone with a question that they would like answered. While there is no specific limitation in terms of educational level or experience in who can ask a question, StudyBoard assumes that the user has moderate experience in browsing the modern internet to search for information on platforms such as Stack Overflow or Quora.

StudyBoard has three types of users. First, there are the normal users who can post questions, reply to questions, and interact with one another. Then, there are users who can be elevated to an endorsed status where their word has more weight than normal users. Users are upgraded based on a combination of the number of likes they have and the number of flags that they have received. Last but not least are the moderators whose job is to manage flagged content to keep the platform safe from incorrect or inappropriate content.

1.4 User feedback

StudyBoard has received preliminary feedback from three users of varying technological expertise who were shown the SRS document and the most recent UI mock-ups. They brought up the issue of how to deal with questions with duplicate or similar content, which we decided to allow. They also asked if users would be able to have a list of tags to which the platform would filter newly generated questions for a more personalized question feed. We decided that this was a missing key feature that we wanted to add to our platform.

StudyBoard plans to receive further feedback at least two more times. At the very least, we want to receive feedback after the alpha and beta deployments for StudyBoard. We plan to get feedback from a wide variety of users so that we can optimize the experience for as many users as possible.

1.5 Existing alternatives

There are a few existing alternatives to this problem of finding an answer on the internet. A list of these platforms includes Stack Overflow, Naver Knowledge iN, and Quora. All three of these platforms mentioned above provide a space where anyone can ask a question and receive a response. One key strength shared by all three platforms is that they all have a relatively large active user base that steadily asks and answers questions online. Another key strength that these platforms share is that they all implement their respective versions of a ranking/rating system for users and the content they post.

However, there are some weaknesses to these platforms that StudyBoard aims to overcome. Naver Knowledge iN is a strictly Korean website, limiting usage to just Korean users. Quora requires users to be logged in to search for content. Stack Overflow is highly specialized in the field of Computer Science.

StudyBoard aims to adapt the best features found in these platforms to create a streamlined, comprehensive platform. Specifically, StudyBoard takes inspiration from Stack Overflow's question and reply model and Naver Knowledge iN's ranking system to create an easily accessible platform with a wider content scope. Furthermore, StudyBoard would allow users to freely search for content without having to log in which is contrary to Quora's model. Users would only be required to log in to submit a reply and access related features.

1.6 Definitions

Moderator	A moderator is someone who is in charge of deleting inappropriate content from the platform.
Event	An event is when there is an update to the website (newly posted reply, newly posted question, like counts going up, etc).
UI	User Interface.
Nginx	Nginx is a web server that can also be used as a reverse proxy, load balancer, mail proxy, and HTTP cache.
I.E.	Internet Explorer; Stock web browser provided with the Windows operating system.
Instance	An instance is a singular web server that contains the platform's source code.
UX	User Experience.
PM	Project Manager.
LP	Lead Programmer.
CDN	Content Delivery Network.
Google OAuth	Google OAuth is an authentication service used to implement Google Sign-In functionality in web and mobile applications.
SDD	Software Design Specification.
Pagination	Pagination is a web design technique in which a document is divided into pages to fit the content in the browser.
Thread	A thread is a series of sub-posts attached to a single post.

1.7 References

Stack Overflow	https://stackoverflow.com/
Naver Knowledge iN	https://kin.naver.com/
Quora	https://www.quora.com/

2.1 Functional Requirements

Stretch Goals are preceded by an asterisk (*)

Login Page

- Users can log into the website using Google OAuth.
- * Users can select/enter tags that they want their question feed to be personalized around during their first login attempt.

Post Question

- Users can post questions with optional media attached (Images).
- Users can edit and delete their questions.
- Users can add tags to their posts to further describe what kind of question they are asking.
- * Users can format their questions within a text editor.

Question Feed

- Users can view the question feed in pagination style.
- Users can also navigate to a question's individual page with all the respective replies
- Users can sort the feed by most recently posted or the highest number of user interactions (endorsed likes, likes, favorites, replies).
- Users can filter the question feed with their preferred tags.

Question Page

- Users can add replies to a question which are displayed thread style underneath the question.
- Users can add replies to other replies which are also displayed thread style underneath the reply.
- Users can like a question or reply
    - System upgrades users who have enough likes to 'endorsed' status.
    - Likes made by an endorsed user are counted and displayed separately.
- Users can flag a question or reply as inappropriate.
    - * Endorsed users that have received enough reports are demoted to normal status.
- Users can get the link to a specific question by clicking on the share button.
- Users can add a specific question to their favorites list
- Replies can be most recently posted or by the number of user interactions (endorsed likes, likes, favorites, replies).

Unified Search Bar

- Users can search for existing questions using the title, content, or tags.
- * Users can see and delete search history.

Rank System

- System displays a special mark next to the names of endorsed users.
- Users can see the highest-ranking endorsed users in a chart.

Profile Page

- Users can view and edit their nicknames.
- Users can see their list of favorite questions.
- Users can see the replies that they have made to other posts.
- Users can see their uploaded questions.
- Users can see and edit their list of preferred tags.

Moderator User

- Moderators can see a list of flagged content that they can review.
- Moderators can delete a post that they think is inappropriate.
- Moderators can delete a reply that they think is inappropriate.

- Moderators can mark a post as appropriate.
- Moderators can mark a reply as appropriate.
- Moderators can blacklist a user from accessing the platform.

Notifications

- Users are notified when their question or reply is liked.
- Users are notified when their question or reply is reported by another user.
- * Users are notified when their question or reply is deleted by a moderator.

2.2 Use cases

Use Case 1: User selects Tags for Personalized Question Feed After First Login

Primary actor:	User
Goal in context:	The user selects the tags to personalize their question feed.
Preconditions:	User has logged into the platform using Google OAuth.
Scenario:	1. User clicks Sign-In with Google. 2. System displays the ‘Sign Up’ page. 3. User clicks/enters at least one tag to follow (required). 4. User clicks/enters as many additional tags as the user wants for a personalized feed. 5. User clicks on the ‘Sign In’ to finalize the Sign-In process. 6. System verifies the data and directs the user to the ‘Question Feed’ page.
Extensions:	5a. User clicks on the ‘Cancel’ button. 5a.1System deletes user information.
Priority:	Expected

Use Case 2: User Posts a Question

Primary actor:	User
Goal in context:	The user wants to ask a question that they need answering.
Preconditions:	User has successfully logged into the platform.
Scenario:	1. User clicks on the ‘Post Question’ textbox to type out their question. 2. User submits the question by clicking on the ‘Post’ button. 3. System updates the feed with the new post at the top of the list.
Extensions:	1a. User wants to add an image to the question. 1a.1 User clicks on the ‘Add Image’ button. 1a.2 System shows the file selector window. 1a.3 User selects the image to add and clicks on the ‘ok’ button. 1b. User wants to add tags to the question. 1b.1 User types ‘#’ and a word to add tags. 2a.1 User wants to cancel the posting process. 2a.1 User clicks on the ‘Cancel’ button. 2a.2 System displays an alert asking for confirmation. 2a.3 User clicks on the ‘ok’ button. 2a.3 The process is cancelled.
Priority:	Essential

Use Case 3: User replies to a question

Primary actor:	User
Goal in context:	User replies to a question posted by another user.
Preconditions:	The question exists in the database.
Scenario:	1. User clicks on the ‘Reply’ button. 2. System prompts the user to type the response. 3. User clicks on the ‘Submit’ button to post their response to a question. 4. System updates the post with the added reply.
Extensions:	3a. User clicks on the ‘Cancel’ button. 3a.1 Text input by the user is deleted and the process is cancelled.
Priority:	Essential

Use Case 4: User adds a question to their favourites list

Primary actor:	User
Goal in context:	User adds a question so that it appears in the user’s profile page.
Preconditions:	There exists a question in the database
Scenario:	1. User clicks on the Favourite button. 2. System appends the question to the user’s list of Favourite questions. 3. Systems informs that the current question was added to their favourites list
Extensions:	1a. User clicks on the Favourite button again to remove the question from their favourites list. 1a.1 System removes the current question from the user’s list of favourite questions.
Priority:	Essential

Use Case 5: User searches for content using Unified Search Bar

Primary actor:	User
Goal in context:	User wants to search for a question or a person.
Preconditions:	The user has opened the StudyBoard website.
Scenario:	1. User clicks on the search bar. 2. User input the content that they want to see. 3. User clicks on the search button. 4. System displays the list of posts or people that matched the search keyword.
Extensions:	4a. There is no content with the given keyword. 4a.1 The system displays a text stating that nothing was found for the current search request.
Priority:	Essential

Use Case 6: User likes a reply

Primary actor:	User
Goal in context:	User thinks a reply is helpful and wants to promote the reply.
Preconditions:	The question has received one or more replies and the user is on the question page.
Scenario:	1. User clicks on the like button. 2. System updates the ‘like’ count for the respective reply.
Extensions:	1a. User has already ‘liked’ the reply. 1a.1 System negates the number of ‘likes’ for the respective reply. 2a. User is an endorsed user. 1a.1 System increments two to the like count.
Priority:	Essential

Use Case 7: User flags a reply as inappropriate

Primary actor:	User
Goal in context:	User finds a reply inappropriate and demands that the moderator removes it.
Preconditions:	User is viewing a specific thread.
Scenario:	1. User clicks on the 'flag’ button. 2. The system displays a pop-up window prompting the user to type in the reason for the flag. 3. User clicks on the 'Ok’ button. 4. System informs the user that their flag was received via system alert. 5. System updates the list of flagged posts on the moderator’s page.
Extensions:	3a. User cancels the flagging process. 3a.1 User clicks the ‘Cancel’ button. 3a.2 The system cancels the process.
Priority:	Essential

Use Case 8: Moderator deletes a flagged reply

Primary actor:	Moderator
Goal in context:	Moderator deletes a flagged response in order to keep the platform safe.
Preconditions:	Moderator is logged in and is on the ‘flagged replies page.
Scenario:	1. Moderator clicks on his/her own profile page. 2. Moderator clicks on the flagged replies column. 3. System displays the list of flagged replies. 4. Moderator reviews a specific reply and clicks delete. 5. System adds a notification to the writer of the deleted post.
Extensions:	3a. There are no replies to review 3a.1 System displays that there is no reply flagged as inappropriate.

Priority:	Essential

2.3 Non-functional Requirements

Performance Requirements

- The system should be able to handle an average of 100 concurrent users with a per-page response time under 2 seconds.
- The number of corresponding replies, likes, and flags should be updated across all elements in the UI as an event occurs.

Operating Constraints

- The platform should run on any modern browser (I.E. may be an exception).

Modifiability

- It should take less than five minutes to get an updated, error-free source code running in the web instance.

Reliability

- In case of any server failure, the average downtime should be less than five minutes.
- The server should be able to automatically restart the latest backup version of the web instance via NGINX upon failure.

Security

- Personal data should only be accessible by that specific user.

Usability

- Users with moderate experience with social networking services such as Facebook and Twitter should take less than 10 minutes to learn how to use the web platform comfortably.

3. System Architecture

3.1 Overview

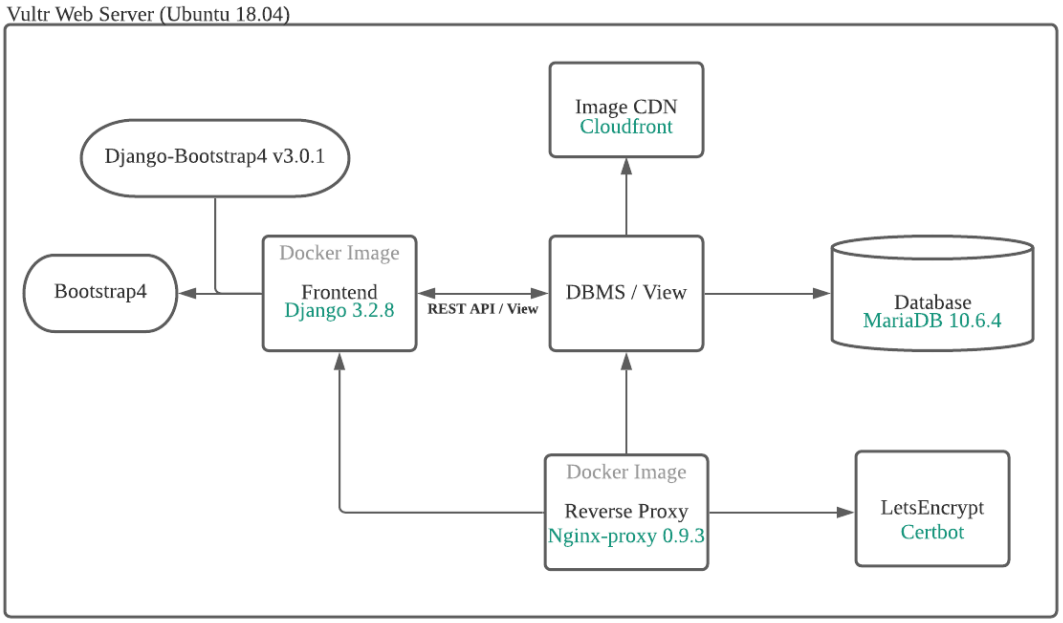

Figure 3.1 System Architecture Diagram

3.1.1 Programming Languages and DBMS Used.

Name	Purpose	Rationale
HTML5/CSS	Frontend	● Used for making templates to be deployed by Django
Python 3.7	Frontend, Backend	● Programming language used for Django. ● 3.9.8 is the latest version of Python that is compatible with Django 3.2.8.
MariaDB 10.6.4	DBMS	● Fork of MySQL DBMS, which is free and open-source. ● 10.6.4 is the latest version of MariaDB. ● Relational database allows for normalization and higher data integrity.

3.1.2 Frameworks and Libraries Used

Name	Purpose	Rationale
Django 3.2.8	Frontend, Backend	● Full stack framework. ● 3.2.8 is the latest version of Django. ● High scalability. ● Myriad of Python libraries available. ● Team members familiar with framework
Django-REST Framework 3.12.4	Frontend, Backend	● 3.12.4 is the latest version of the framework. ● Highly compatible with Vultr.
Django-bootstrap4 3.0.1	Frontend	● 3.0.1 is the latest version of the library. ● Blends Django and Bootstrap v4 seamlessly
Bootstrap 4.6	Frontend	● Greater compatibility with commonly used web browsers than bootstrap5. ● Extensive documentation to easily create clean UI.
Boto3 1.18	Image storing	● Amazon Web Services software development kit for Python. ● Used to store images on Amazon Cloudfront on Django.

3.1.3 Services and API used

Name	Purpose	Rationale
Vultr	Server	● Includes Ubuntu 18.04 which allows for the use of Docker. ● Ease of use in deploying servers. ● Enables Swift error handling.
Docker	Containerization	● Consistent and isolated environments ● Flexible, Scalable, Modular
Amazon Cloudfront	Image storing	● Secure and scalable ● Free use available for 12 months.
Nginx-Proxy	Reverse proxy	● 0.9.3 is the latest version ● Easy setup of nginx and docker-gen ● Automated Nginx reverse proxy config / reload
Google OAuth 2.0	Authentication	● Secure authentication solution provided by Google. ● Ease of use for users and developers due to widespread use of Google Mail.

3.1.4 Design Patterns

Studyboard follows the Model View Template design pattern which is specified in Django's documentation. The rationale of using Django, and thus the MVT design pattern, is mentioned above.

3.2 Data Design

Notification	
notification_id	INTEGER
user_id	INTEGER
notification_text	VARCHAR
notification_date	DATETIME
notification_seen	BOOLEAN
PRIMARY KEY	notification_id
FOREIGN KEY	user_id REFERENCES User(user_id)

User	
user_id	INTEGER
user_email	VARCHAR
user_nickname	VARCHAR
user_is_endorsed	BOOLEAN
user_is_mod	BOOLEAN
user_flags_received	INTEGER
user_likes_received	INTEGER
PRIMARY KEY	user_id

Tag	
tag_id	INTEGER
tag_name	VARCHAR
PRIMARY KEY	tag_id

User_Tag	
user_id	INTEGER
tag_id	INTEGER
PRIMARY KEY	(user_id, tag_id)
FOREIGN KEY	user_id REFERENCES User(user_id) tag_id REFERENCES Tag(tag_id)

Post_Tag	
post_id	INTEGER
tag_id	INTEGER
PRIMARY KEY	(post_id, tag_id)
FOREIGN KEY	post_id REFERENCES Post(post_id) tag_id REFERENCES Tag(tag_id)

Post	
post_id	INTEGER
user_id	INTEGER
post_title	VARCHAR
post_text	VARCHAR
post_like_count	INTEGER
post_reply_count	INTEGER
post_favourite_count	INTEGER
post_date	DATETIME
PRIMARY KEY	post_id
FOREIGN KEY	user_id REFERENCES User(user_id)

Post_Image	
post_id	INTEGER
image_id	INTEGER
image_url	VARCHAR
PRIMARY KEY	(post_id, image_id)
FOREIGN KEY	Post_id REFERENCES Post(post_id)

User_Post_Like	
user_id	INTEGER
post_id	INTEGER
PRIMARY KEY	(user_id, post_id)
FOREIGN KEY	user_id REFERENCES User(user_id) post_id REFERENCES Post(post_id)

Post_Report	
report_id	INTEGER
post_id	INTEGER
user_id	report_id
report_text	VARCHAR
report_date	DATETIME
PRIMARY KEY	report_id
FOREIGN KEY	post_id REFERENCES Post(post_id) user_id REFERENCES User(user_id)

Reply	
reply_id	INTEGER
user_id	INTEGER
reply_text	VARCHAR
reply_like_count	INTEGER
reply_date	DATETIME
PRIMARY KEY	reply_id
FOREIGN KEY	user_id REFERENCES User(user_id)

Reply_To_Post	
post_id	INTEGER
reply_id	INTEGER
PRIMARY KEY	(post_id, reply_id)
FOREIGN KEY	reply_id REFERENCES Reply(reply_id) post_id REFERENCES Post(post_id)

Reply_To_Reply	
source_id	INTEGER
reply_id	INTEGER
PRIMARY KEY	(source_id, reply_id)
FOREIGN KEY	source_id REFERENCES Reply(reply_id) reply_id REFERENCES Reply(reply_id)

User_Reply_Like	
user_id	INTEGER
reply_id	INTEGER
PRIMARY KEY	(user_id, post_id)
FOREIGN KEY	user_id REFERENCES User(user_id) reply_id REFERENCES Reply(reply_id)

Reply_Report	
report_id	INTEGER
reply_id	INTEGER
user_id	report_id
report_text	VARCHAR
report_date	DATETIME
FOREIGN KEY	reply_id REFERENCES Reply(reply_id) user_id REFERENCES User(user_id)

Favourite_Question	
user_id	INTEGER
post_id	INTEGER
PRIMARY KEY	(user_id, post_id)
FOREIGN KEY	user_id REFERENCES User(user_id) post_id REFERENCES Post(post_id)

Search_History	
search_id	INTEGER
user_id	INTEGER
search_text	VARCHAR
search_date	DATETIME
PRIMARY KEY	search_id
FOREIGN KEY	user_id REFERENCES User(user_id)

Blacklisted_User	
user_id	INTEGER
PRIMARY KEY	user_id
FOREIGN KEY	user_id REFERENCES User(user_id)

3.3 UML Sequence Diagrams

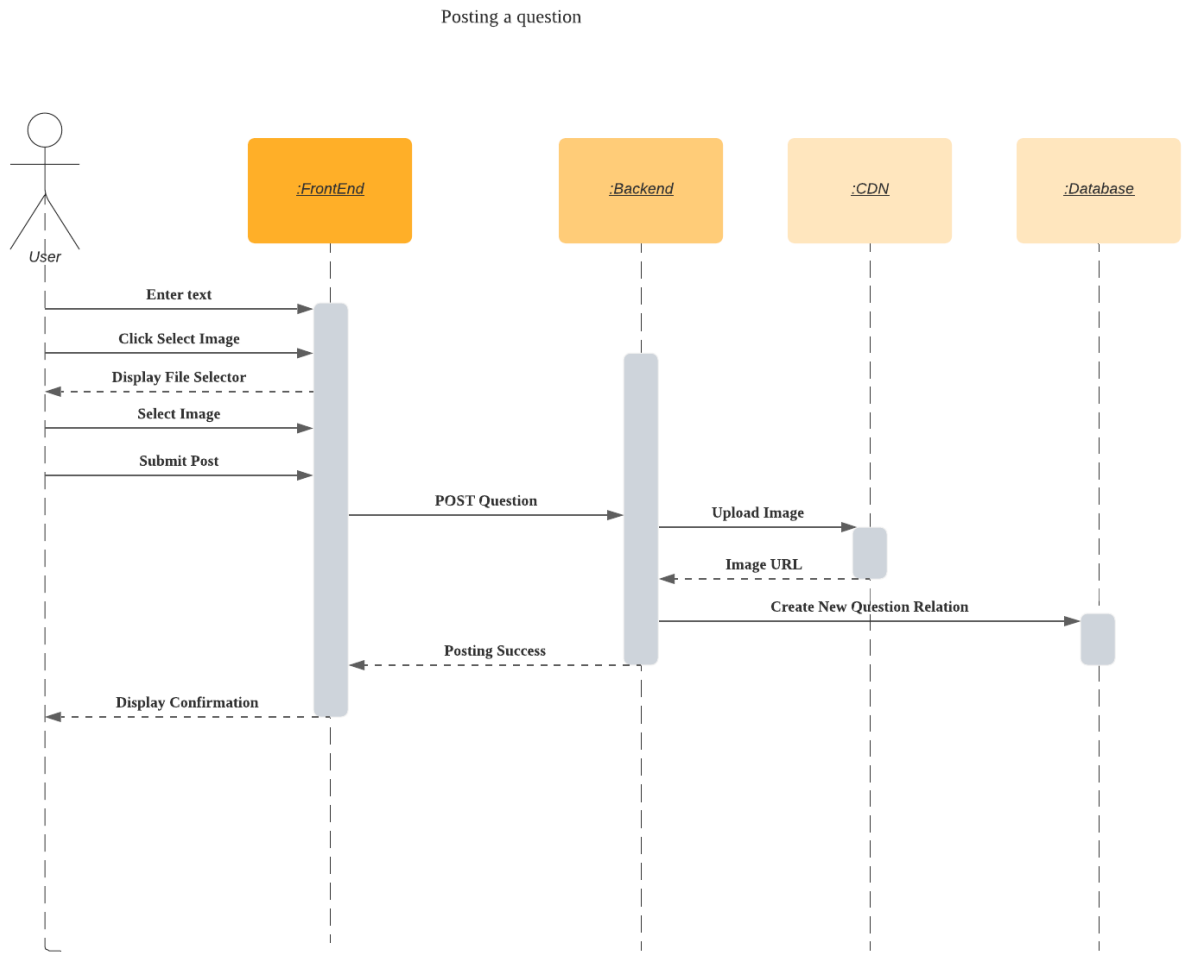

Figure 3.3.1 User posting a Question

The figure above details the user's web request in UML Sequence format for posting a new question on StudyBoard. This UML Sequence diagram assumes the user has successfully logged in to the system.

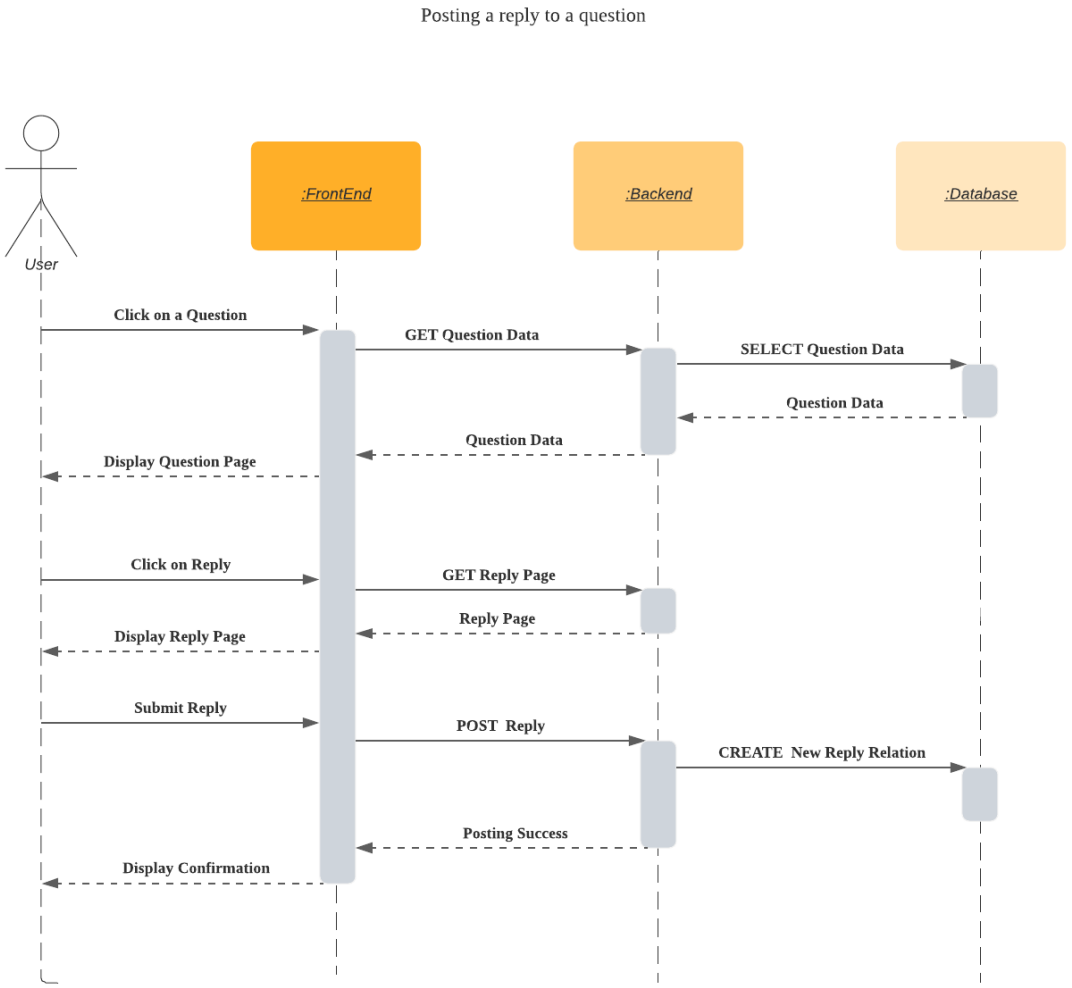

Figure 3.3.2 User reply to a question

The figure above details the user's web request in UML Sequence format for replying to a question. This UML Sequence diagram assumes the user has successfully logged in to the system.

3.4 API Design

Link to API Design

Attached below is a table of common HTTP Response Status Codes that is used by StudyBoard. It should be noted that this table is not inclusive of every status code that may be used by StudyBoard but rather indicates the most commonly used status codes for StudyBoard.

HTTP Response Status Codes	
Status Codes	Meaning
200 OK	Operation was successful
201 CREATE	When a POST request creates a new resource
202 ACCEPTED	Acknowledge that request was sent to the server
400 BAD REQUEST	Client side input validation failure
401 UNAUTHORIZED	Authentication unsuccessful
403 FORBIDDEN	User is not authorized to perform the request
500 INTERNAL SERVER ERROR	Server cannot handle the request

3.5 Deployment

StudyBoard will be deployed on a Ubuntu 18.04 machine using the Vultr cloud service. StudyBoard will use a singular Django instance containerized in a Docker image for both the frontend and backend applications. The backend will serve as a DBMS for the MariaDB database and send populated template views to the frontend. Any dynamic update request in the frontend will be sent via Django-Rest to the backend.

StudyBoard will use a reverse proxy to increase flexibility, security, maintenance, and load balancing. Using nginx-proxy will allow for automatic reverse proxy configurations and reload nginx whenever containers are started and stopped. LetsEncrypt (Certbot) will be used to add HTTPS certification to StudyBoard for better security.

Images will be stored on a AWS Cloudfront CDN to separate static image files from both the web server and the database. Using a CDN will allow for faster loading of images while eliminating any strain on the web server itself in terms of file storage.

3.6 Code Conventions

StudyBoard complies to the following coding conventions for the following languages used in its development and deployment.

Language	Adherence	Link
HTML/CSS	Google HTML/CSS Style Guide	URL
JavaScript	Google JavaScript Style Guide	URL
Python	PEP 8	URL

In the event that there are conflicting naming schemes in the coding conventions mentioned above, the following naming conventions will be used instead. All other coding conventions for each language will adhere to their respective styles.

What	Style
Variables/Methods	lowerCamelCase
Class Names	UpperCamelCase
Constants	SCREAMING_SNAKE_CASE

Attached below is a table describing the code conventions used in the MariaDB database itself for the tables and their attributes/triggers.

MariaDB Code Conventions	
Attributes, Keys	lower_snake_case
Tables	Upper_Snake_Case
Triggers	Upper_Snake_Case

4. Schedule

Link to Schedule

5. Contributions

Hawon Park	Jeong Ho Shin	Pyungkang Hong
Data Design	Introduction	Overview
Coding Conventions	Requirements	Deployment
Schedule	UML Sequence Diagrams	
API Design		