Adhish Malhotra (100761306)
Aiden Jolley-Ruhn (100745886)
Hima Paul (100753261)
Shreya Patel (100747036)

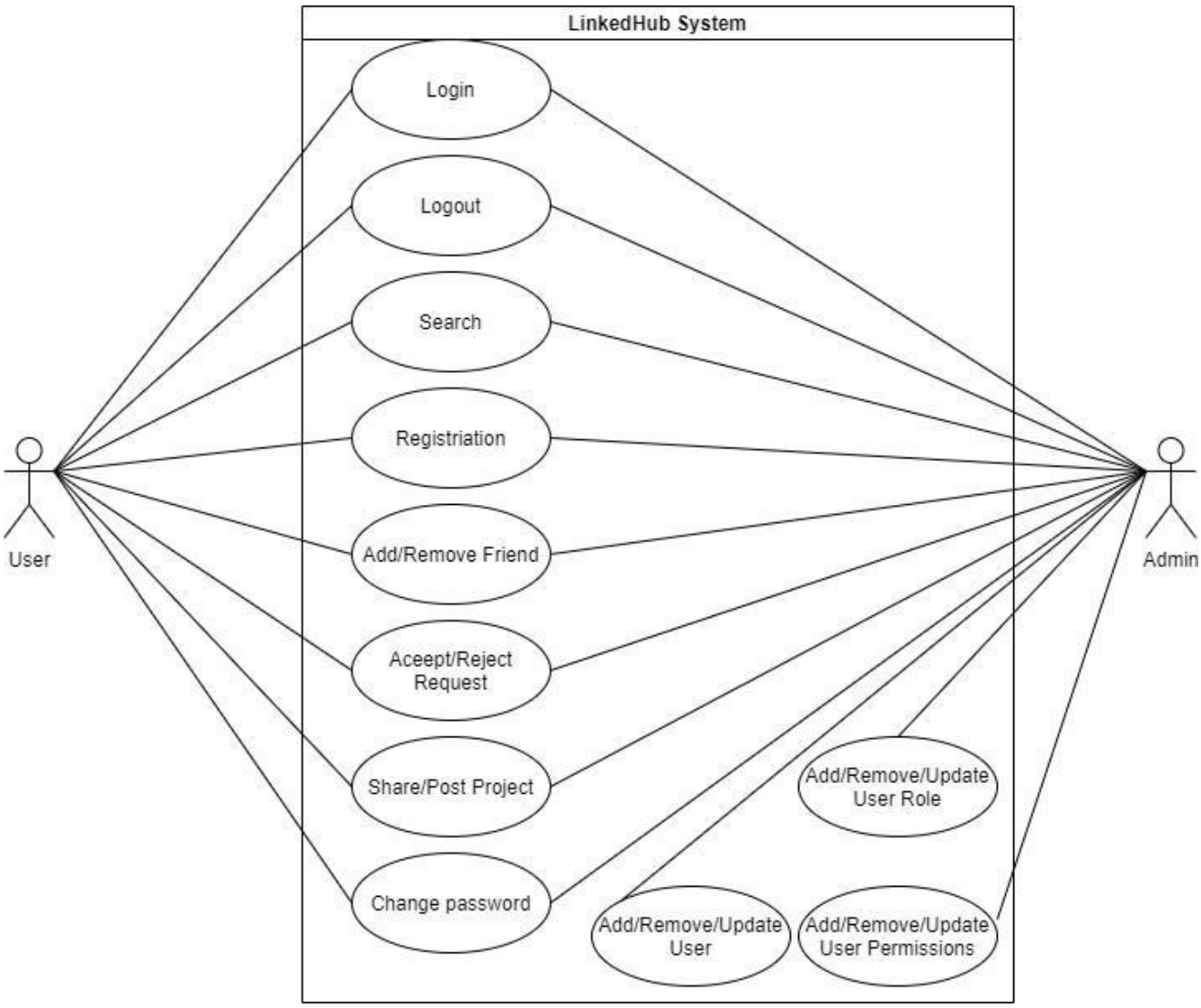
Project Report

LinkedHub

Table of Contents

Iteration 1

Use Case



Use Case Descriptions

Use Case	Details
UC-1 Registration	A user should be able to register an account within the Linkedhub database.
UC-2 Login	A user should be able to login to an existing account within the Linkedhub database.
UC-3 Logout	A user should be able to logout of a currently logged in account within the Linkedhub server.
UC-4 Add/Remove Friend	A user should be able to add or remove a friend from their personalized friends list for their account.
UC-5 Accept/Reject Request	A user should be able to accept or reject an incoming friend request from another user.
UC-6 Share/Post Project	A user should be able to upload any pre-existing projects to the Linkedhub database.
UC-7 Change Password	A user should be able to update their password within the Linkedhub database.
UC-8 Add/Remove/Update User	An admin user should be able to add, remove, or update a user within the Linkedhub database.
UC-9 Add/Remove/Update User role	An admin user should be able to add, remove, or update a user's role within the Linkedhub database.
UC-10 Add/Remove/Update User Permissions	An admin user should be able to add, remove, or update a user's permissions within the Linkedhub database.
UC - 11 News Feed	The news fed page should be able to show all the posts posted by the user who is logged in and all the posts of the user logged in user's friends
UC-12 Like/Comment	A user should be able to like/comment on their own post as well as on the posts of the people they're friends with.

Figure 1.1.1

Quality Attributes

ID	Quality Attribute	Scenario	Associated Use Case
QA-1	Performance	An error occurs within the system and causes it to crash. The server will continually attempt to restart on the last available backup and move onto the next if the one chosen still contains the error. The server should be able to resume operation within 1 minute.	All
QA-2	Availability	The hosting server is sent a heartbeat every 10 seconds and awaits the response from the server during normal operations. Depending on if the return time of the heartbeat is , the status of the server is determined to be responsive or not. This ensures little to no downtime on the server.	All
QA-3	Modifiability	The user updates their profile with new information and projects that they've done within the server during normal operation. After these updates are made, the server's database is updated as well to ensure consistency among the entire server.	UC-6 (Share Post/Project)
QA-4	Interoperability	When a user connects to the server, the server exchanges information with the user under normal operating conditions. The user must present an OS of either Windows, OSX, or Linux, along with a browser that is either (Chrome V3.0+, Firefox V4+, IE8+). If the server reads the user information and discovers that one of these conditions are not met, the user is told "Your operating system or web browser doesn't meet our specifications" and isn't granted access.	All
QA-5	Usability	A user is able to upload information about themselves to create a profile within the server during normal operation. The user will send the data to the server which will be checked for any inappropriate words or phrases, then uploaded to the database. Any other user should be able to see the changes made when reviewing the user's profile within 1 minute of uploading.	UC-6 (Share Post/Project)
QA-6	Security	A user contributes to a project that they are collaborating with others on within the server during normal operation. The ability to see who and when the change to the project happened is able to be seen any time by a certain user group.	All

QA-7	Security	When a user creates their account, their password should be encrypted before saving it to the database. The reason behind this QA is that the database maintainer should not be able to access any user's password in the database	UC-1 (registration)

Figure 1.1.2

Constraints

ID	Constraint
CON-1	A user isn't able to create a collaboration project with only themselves.
CON-2	The network connection to user's machines may be low bandwidth, but must be reliable.
CON-3	Any changes to projects, profiles, or any other user change on the machine must be recorded.
CON-4	The server must be accessed using one of the following web browsers (Chrome V3.0+, Firefox V4+, IE8+) and the only acceptable platforms being Windows,OSX, and Linux.
CON-5	Any event that takes place within the last 30 days must be recorded.

Figure 1.1.3

Concerns

ID	Concern
CRN-1	Accomplish completion of the final system in the allotted time and resources.
CRN-2	Evenly distribute the workload within the project across the development team to fully utilize the team's capabilities.
CRN-3	Creating the first prototype model of the system's structure.

Step 1: Review Inputs

<table>
<tr><th>Category</th><th>Details</th></tr>
<tr><td>Design purpose</td><td>This is a Social Networking website. The purpose of this is to produce sufficiently detailed design to support the functionality of the website.
The system should provide the user the ability to create an account, login into their account, search for different users in the database or different posts in the system, add/remove friends, share/delete posts, modify their profile, etc.</td></tr>
<tr><td>Primary Functional requirements</td><td>Of the use cases presented in Figure 1.1.1, the primary ones are:
- UC -1 because it is the first step in accessing the website
- UC - 2 because it directly supports the core functionality
- UC - 6 because it directly supports the core functionality</td></tr>
<tr><td>Quality attribute scenarios</td><td>The scenarios which are explained above are prioritized in the following table:
<table>
<tr><th>Scenario ID</th><th>Importance to Customer</th><th>Difficulty of Implementation according to the architect</th></tr>
<tr><td>QA-1</td><td>High</td><td>High</td></tr>
<tr><td>QA-2</td><td>High</td><td>High</td></tr>
<tr><td>QA-3</td><td>High</td><td>Low</td></tr>
<tr><td>QA-4</td><td>Medium</td><td>Medium</td></tr>
<tr><td>QA-5</td><td>High</td><td>Low</td></tr>
<tr><td>QA-6</td><td>Medium</td><td>High</td></tr>
<tr><td>QA-7</td><td>High</td><td>Medium</td></tr>
</table></td></tr>
<tr><td>Constraints</td><td>All constraints are included as drivers.</td></tr>
<tr><td>Architectural Concerns</td><td>
<table>
<tr><td>ID</td><td>Concern</td></tr>
<tr><td>CRN-1</td><td>Accomplish completion of the final system in the allotted time and resources.</td></tr>
<tr><td>CRN-2</td><td>Evenly distribute the workload within the project across the development team to fully utilize the team's capabilities.</td></tr>
<tr><td>CRN-3</td><td>Creating the first prototype model of the system’s structure.</td></tr>
</table>
CRN-2 was selected as the driver.</td></tr>
</table>

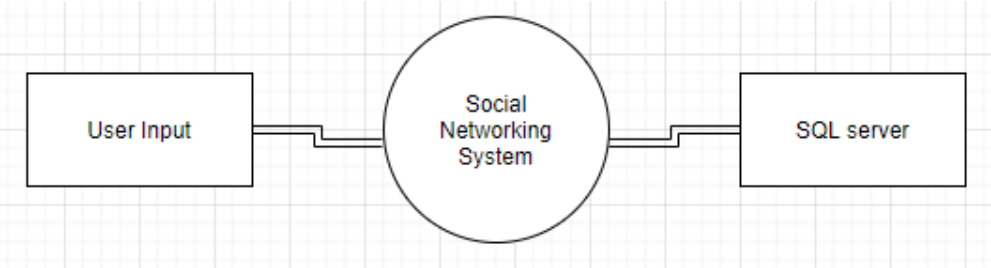
Step 2: Establish Iteration Goal by Selecting Drivers

The drivers that will be focused on in this iteration are:

- QA-1: Performance
- QA-2: Availability
- QA-5: Usability
- QA-6: Security
- QA-7: Security
- CON-1: A user isn't able to create a collaboration project with only themselves.
- CON-3: Any changes to projects, profiles, or any other user change on the machine must be recorded.
- CON-4: The server must be accessed using one of the following web browsers (Chrome V3.0+, Firefox V4+, IE8+) and the only acceptable platforms being Windows,OSX, and Linux.

Step 3: Choose One or More Elements of the System to Refine

In this iteration, we will be refining the server side of the application. It's important for the system to support connecting to both client and the server easily. The modules which are required are located in the server.
The following components will be refined in this step:



Step 4: Choose One or More Design Concepts that Satisfy the Selected Drivers

Design Design and Locations	**Rationale**
Logically structure the client part of the system using the	**Web application** reference architecture supports deploying and running our application on a web browser. It is fairly easy to create views for a web application as compared to any other type of

<table>
<tr><td>Web application reference architecture</td><td>application. Choosing this architecture itself supports QA-3, QA-4, QA-5. Moreover, it also supports CON-3 and CON-7.

<u>Discarded Alternatives:</u>

<table><tr><th>Alternative</th><th>Reason for discarding</th></tr><tr><td>Mobile Application</td><td>This alternative was discarded because this type of device was not considered for accessing the system.</td></tr></table></td></tr>
<tr><td>Logically structure the server part of the system similar to a web application</td><td>The database which is created for the purpose of this application is primarily used for storing different types of data that would be essential to our application. Most of the processing is done before-hand, meaning the data which is supplied by the user is first checked to make sure whether it’s relevant or not, and once it's established that the data is relevant, it’s stored in the database. The application follows a client-side scripting pattern.
All of the team members are familiar with the MySQL database system and hence we decided to proceed with this architecture specifically.</td></tr>
<tr><td>Physically structure the application using the three-tier deployment pattern</td><td>The three tier development pattern is used because the web application requires a browser that the user connects from for the client, a server that is used for the logic, and finally a dedicated server only for the data that is inputted on the application.

Discarded Alternatives:
A two tier development pattern was considered but was found to be insufficient due to the data having to remain on its own server. Furthermore, a tier development pattern over 3 would be too many as it’s not required.</td></tr>
<tr><td>Build the user interface of the client application using HTML, CSS, bootstrap, JQuery, JavaScript and PHP</td><td>Building the user interface on the foundational blocks of HTML, CSS, bootstrap, Jquery, Javascript, and PHP provides a great outlook for the project as each one provides some aspect of workability upon the user interface.

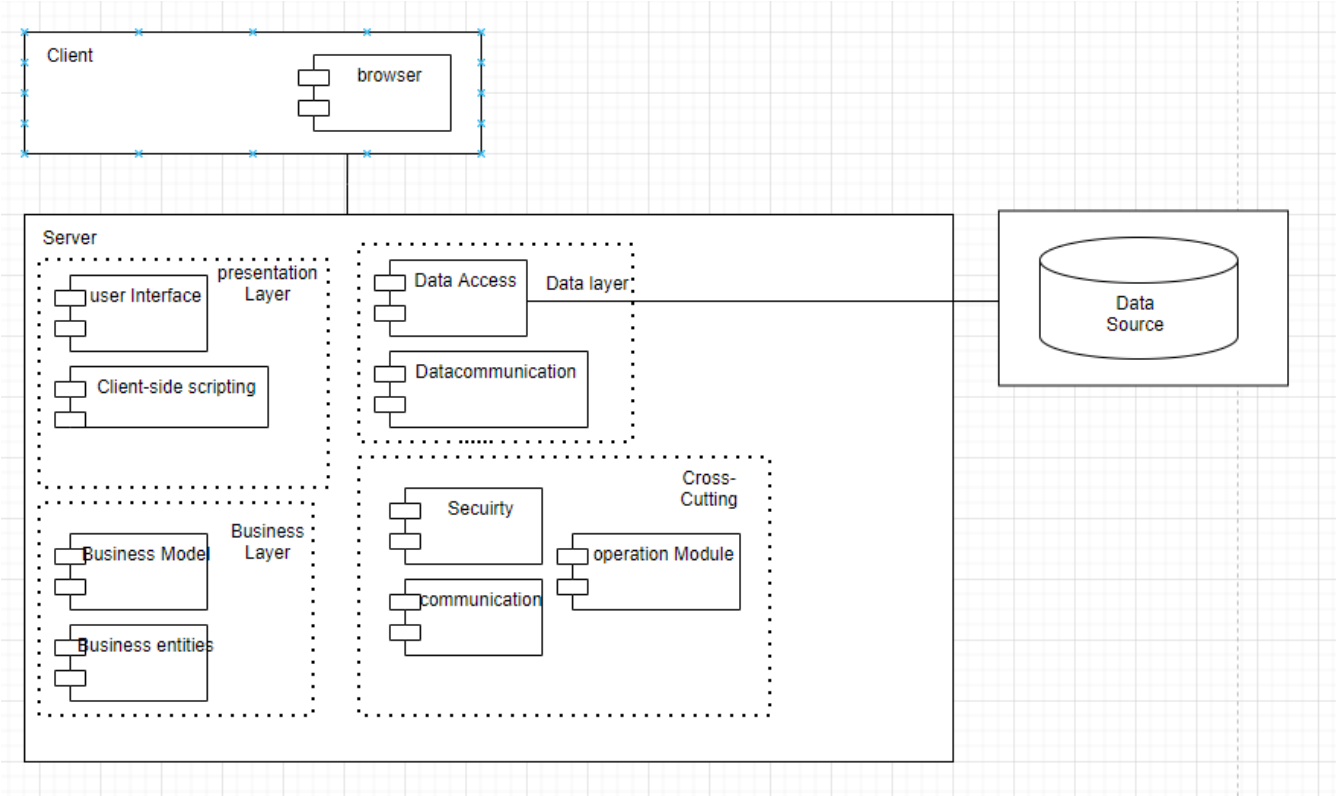
Discarded Alternatives:
React was discarded due to its complexity and learning curve. Angular was also discarded for the same reason of its complexity and the learning curve that comes along with it. The team of development felt more comfortable working with the frameworks listed above.</td></tr>
<tr><td>Deploy the application using GitHub Pages</td><td>GitHub Pages allows you to turn GitHub repositories to create websites.
All the changes are made and pushed in real time, which will allow</td></tr>
</table>

	changes to projects, profiles, or any other user change on the machine to be recorded. (CON-3) GitHub pages support more than 750 MME types hence, the deployed website will be accessible on many different browsers including Chrome V3.0+, Firefox V4+, IE8+. (CON-4)

Step 5: Instantiate Architectural Elements, Allocate Responsibilities and Define Interfaces

Design decisions and Location	Rationale
Create initial domain	Initial domain model helps understand working of the system
Define modules based on Use case model	Business logic made using the reference from the use case model
Incorporate a separate layer to hold the data on server side	Adding a third layer to the server side that will manage the data ensuring that it is consistent with the data on the dedicated server.
Connecting the database access module to data source	The data layer that exists on the server side will have access to the database through an access module. The purpose of this module is to provide access to the SQL database server.

Step 6: Include all the Rough Diagrams



Step 7: Design the Kanban Board

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
	UC-1		Selected reference architecture establishes the modules that will support this functionality.
	UC-2		Selected reference architecture establishes the modules that will support this functionality.
	UC-6		Selected reference architecture establishes the modules that will support this functionality.
QA-1			No relevant decision made, as it is necessary to identify the elements that participate in the use case that is associated with the scenario.
		QA-2	The use of GitHub Pages as a hosting platform will allow for minimal down time.
		QA-5	The use of a browser as a client in the web application architecture is sufficient to fulfill this quality attribute
QA-6			No relevant decisions have been made regarding the elements that participate in this quality attribute.
QA-7			No relevant decisions have been made regarding the elements that participate in this quality attribute.
		CON-1	Allowing users to collaborate on projects with others is sufficient to fulfill this concern
CON-2			No relevant decisions have been made regarding the elements that participate in this constraint.
		CON-3	Changes to projects, profiles, or any other user change on the machine is recorded
		CON-4	The server can be accessed using specific web browsers and the acceptable platforms are Windows, Linux and OSX.
	CON-5		No relevant decisions have been made regarding the elements that participate in this constraint.
		CRN-1	The final system is successfully completed in the allotted time
		CRN-2	The workload is evenly distributed within the project across the development team to fully utilize the team's capabilities.
		CRN-3	The first prototype of the system was successfully completed.

Iteration 2

The outcomes of the actions undertaken in each of the ADD phases in the second iteration of the design process for the LinkedHub system are shown in this section. We go from the general and coarse-grained functionality descriptions used in iteration 1 to more comprehensive decisions that will drive implementation and, as a result, the establishment of development teams in this iteration.

Step 2: Establish Iteration Goal by Selecting Drivers

The primary use cases and concerns for this iteration are:

- UC - 1
- UC - 2
- UC - 6
- CRN - 3

Step 3: Choose One or More Elements of the System to Refine

In this iteration, we will be refining the server side of the application. It's important for the system to support connecting to both client and the server easily. The modules which are required are located in the server since we are utilizing the web Web Application Architecture.

Step 4: Choose One or More Design Concepts that Satisfy the Selected Drivers

Design Design and Locations	**Rationale and Assumptions**
Create a domain model for the application	Before starting the functional decomposition, we need to create an initial domain model of the system that identifies the key entities in the domain and their relationships. There are no good options. Finally, it is necessary to build a domain model. Otherwise, it won't happen in an optimal way, resulting in specialized architectures that are difficult to understand and maintain. After a series of brainstorming sessions, we were able to create a domain model for our project using er diagrams and schema diagrams.
Identify Business Logic and map it to the use cases	In order to appropriately segregate concerns and specify data processing, use cases must be mapped to business logic. Alternatively, functional needs might be mapped to business entities,

	but certain requirements could be overlooked, or the business workflow could be riddled with duplication.
Decompose Business Logic into Business Entities	The entities for the business domain and their business logic are represented by combined business entities. Decomposing business logic into business entities has no viable alternatives.
Use Ajax and PHP	Ajax stands for Asynchronous Javascript and XML. AJAX is basically a technique which reloads certain contents of the page without actually reloading the entire page. Whereas on the other hand, PHP is a scripting language that stands for hypertext processor. We'd be using PHP for most of our client-side processing and to connect the business logic to the server.

Step 5: Instantiate Architectural Elements, Allocate Responsibilities and Define Interfaces

Design decisions and Location	**Rationale**
Create an initial domain model	We make an initial domain model and first prototype to better understand the working of the system in our primary use cases (CRN-3)
Map modules based on Use Case model	Taking reference from the use case model, a finally business logic model is created after defining the business entities.
Create a component diagram to map modules to the workflow	Modules that support the primary use cases must be defined, and the responsibilities for these modules must be defined as well.
Create Sequence Diagrams for primary use cases	Once the modules have been defined, sequence diagrams are required for the primary use cases to define interfaces for the modules.

Step 6: Sketch Views and Record Design Decisions

As a result of all the decisions made in step 5 of our iteration 2, several diagrams have been created.

- Figure 1.5 highlights the initial domain model of our system.
- Figure 1.6 marks the business entities diagram necessary for our system.
- Figure 1.7 highlights all the modules that support the primary use case.

And, in the end, we've defined business entities in the form of a business entity element and their requirements.

Initial Domain Model

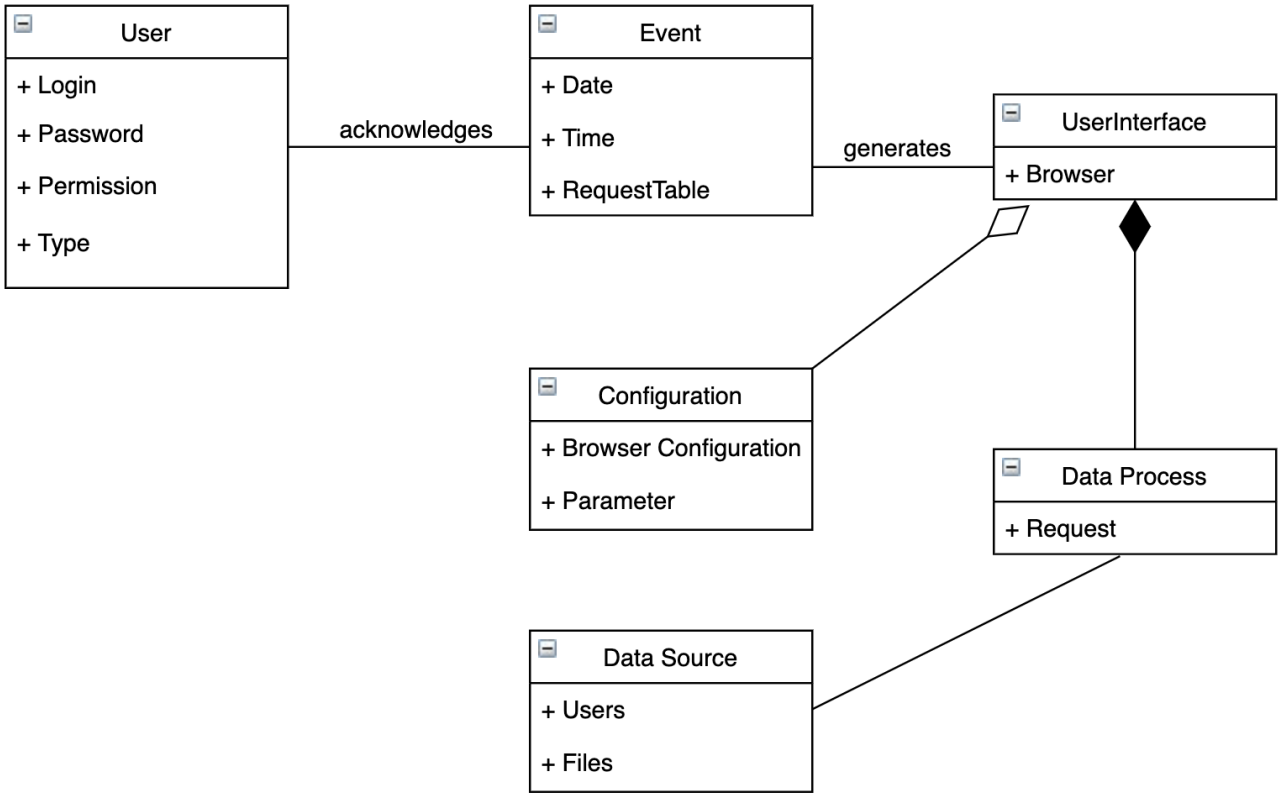


Figure 1.5 Initial Domain Model (Key: UML)

Business Entities Diagram

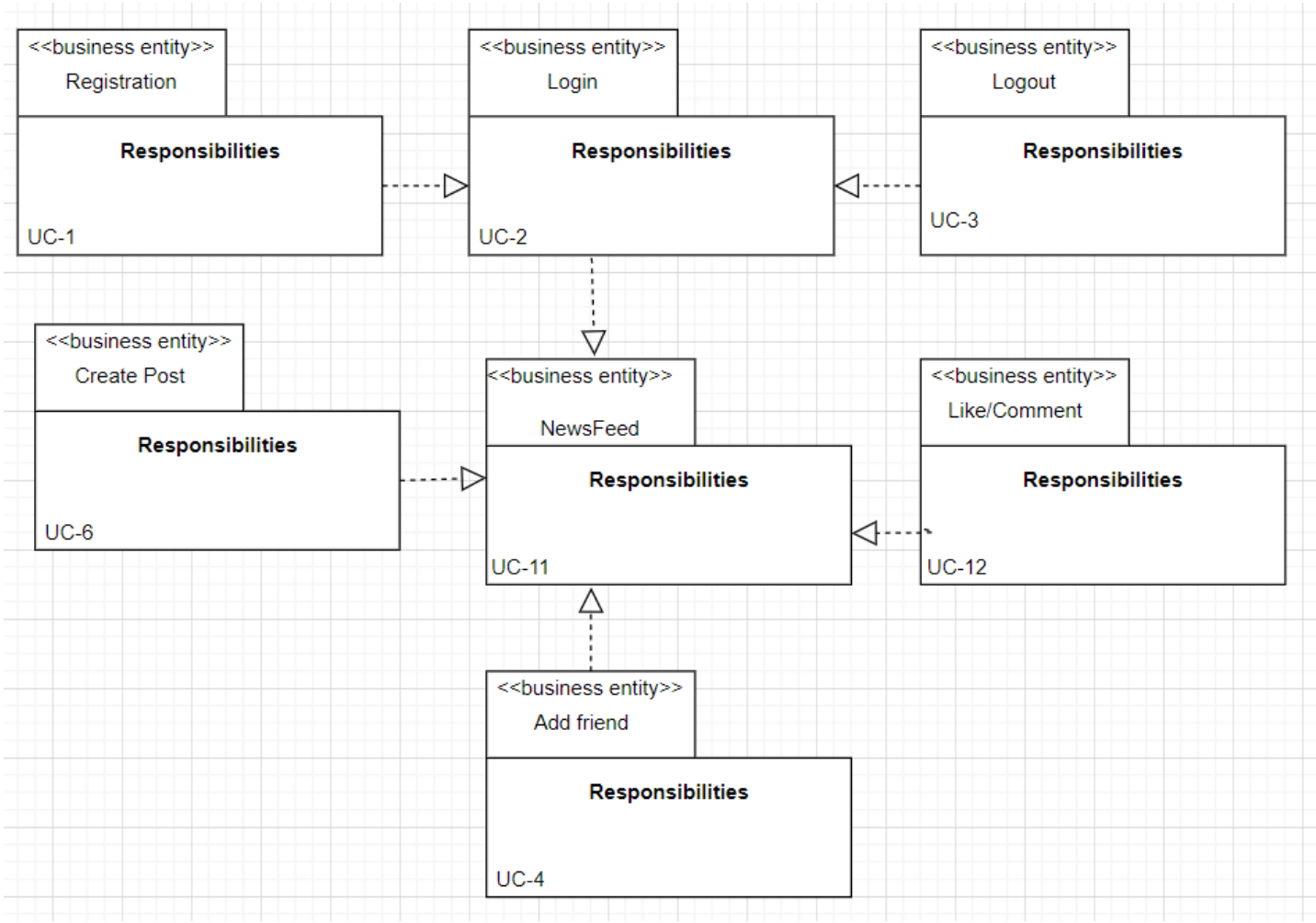


Figure 1.6 Business Entities Model (Key: UML)

https://drive.google.com/file/d/1VcSxMyd7nkKYuhw3jhtD1simL4KMDnmi/view?usp=sharing

Modules that support Primary Use Case

https://drive.google.com/file/d/117FWQD67li133vnXDCpP_ud8RsibB-OB/view?usp=sharing

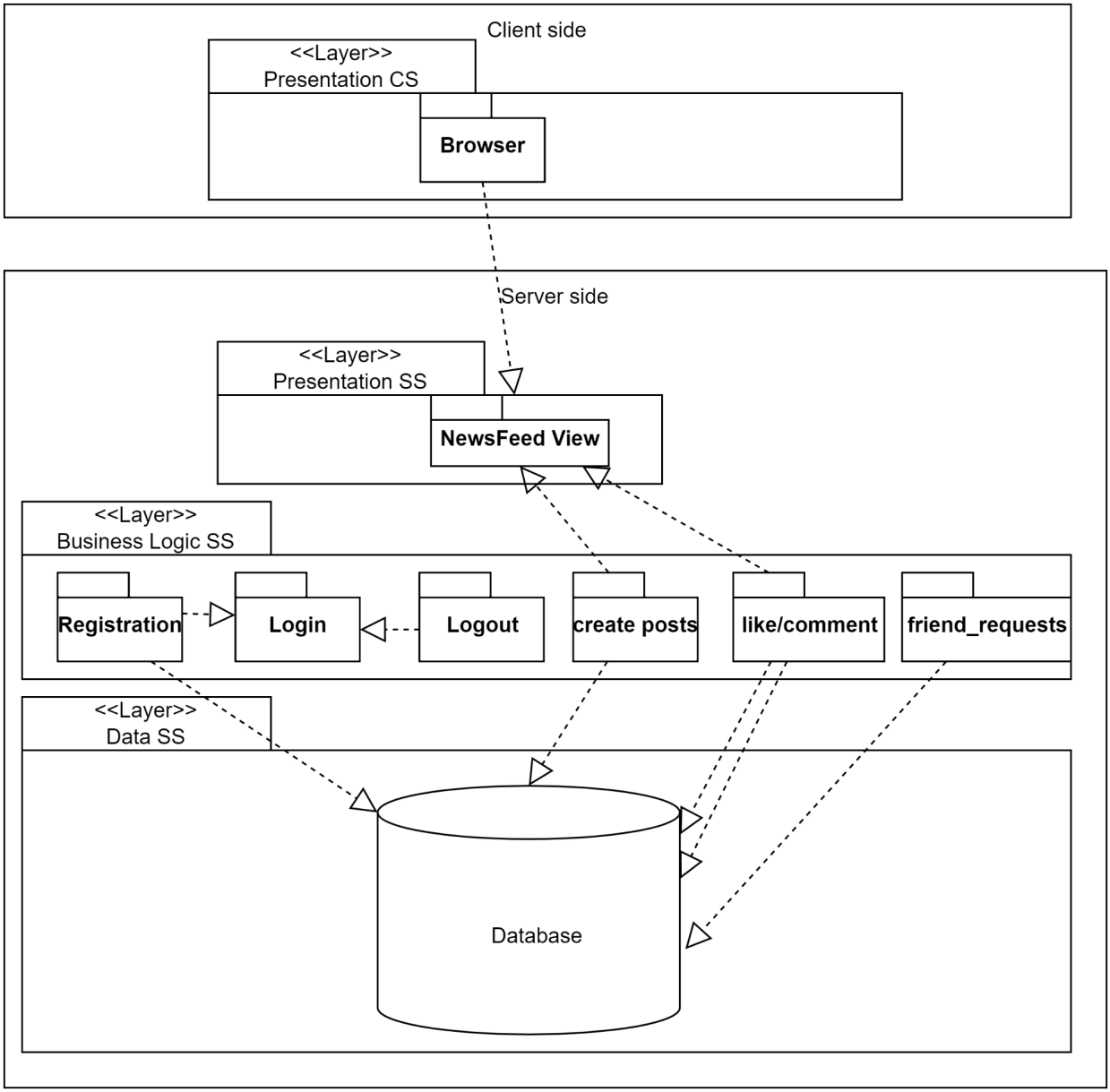


Figure 1.7 Module View (Key: UML)

Define Business Entities.

Element	Responsibility
Registration Controller	Enables a user to register their account on the LinkedHub server.
Login Controller	Validates whether the login information entered by the user is correct or not. If the login information is correct, the user is directed to the main page of the website. You can also logout
News Feed view	This is the main page of our social networking website. It consists of all the posts posted by the different users who are your friends.
Create Post Controller	This function allows the user who is logged into the system to create a post. This post shall be visible on the user's newsfeed page along with the posts of other users who are friends of the user who is logged in.
Comment Controller	This function allows a user who is logged into the system to comment on their post or their friend's post.
Like Controller	This function allows a user who is logged into the system to like their own post or their friend's post. You can also unlike a liked post.
Add friend controller	Enables a user who is logged into the system to add other users. Just like the like button controller you can also remove a friend.

UC-1 Registration

https://drive.google.com/file/d/190HxkFeZkljTQHXCPoCUDwlTqUCgH4-t/view?usp=sharing

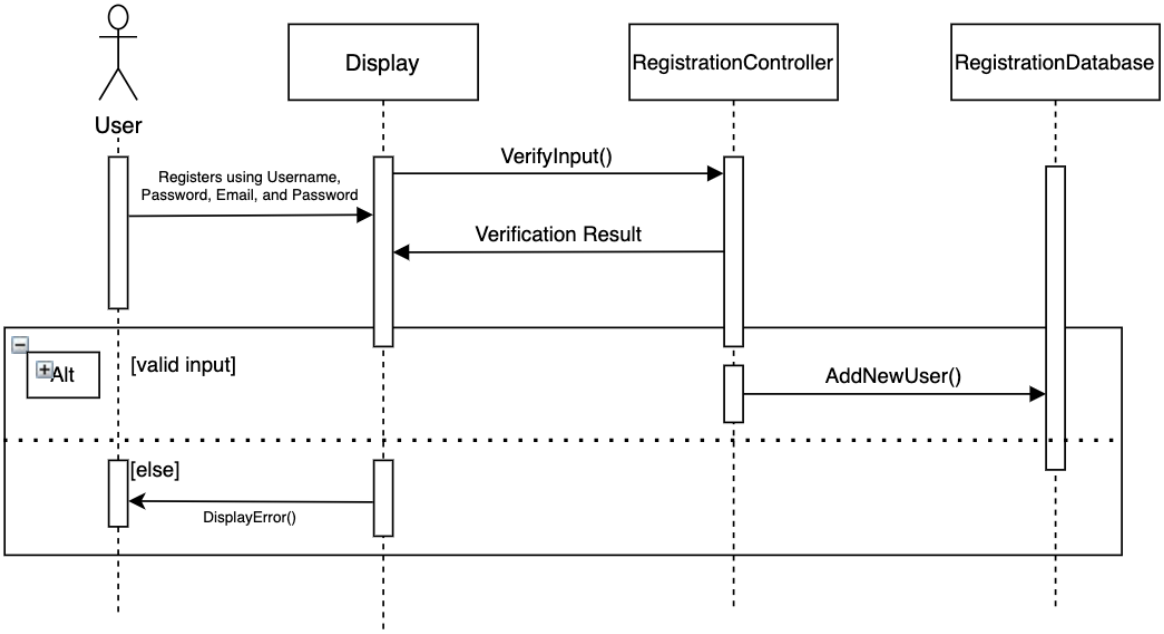


FIgure 1.8 Sequence Diagram for UC-1

Function	**Description**
VerifyInput ()	This method is part of the RegistrationController and verifies whether the information added is in a valid format so that it can be stored properly in the Database.
AddNewUser()	This method is part of the RegistrationController and handles the event of the user addition of the RegistrationDatabase.
DisplayError()	This method outputs an error if the input is invalid.

UC-2 Login

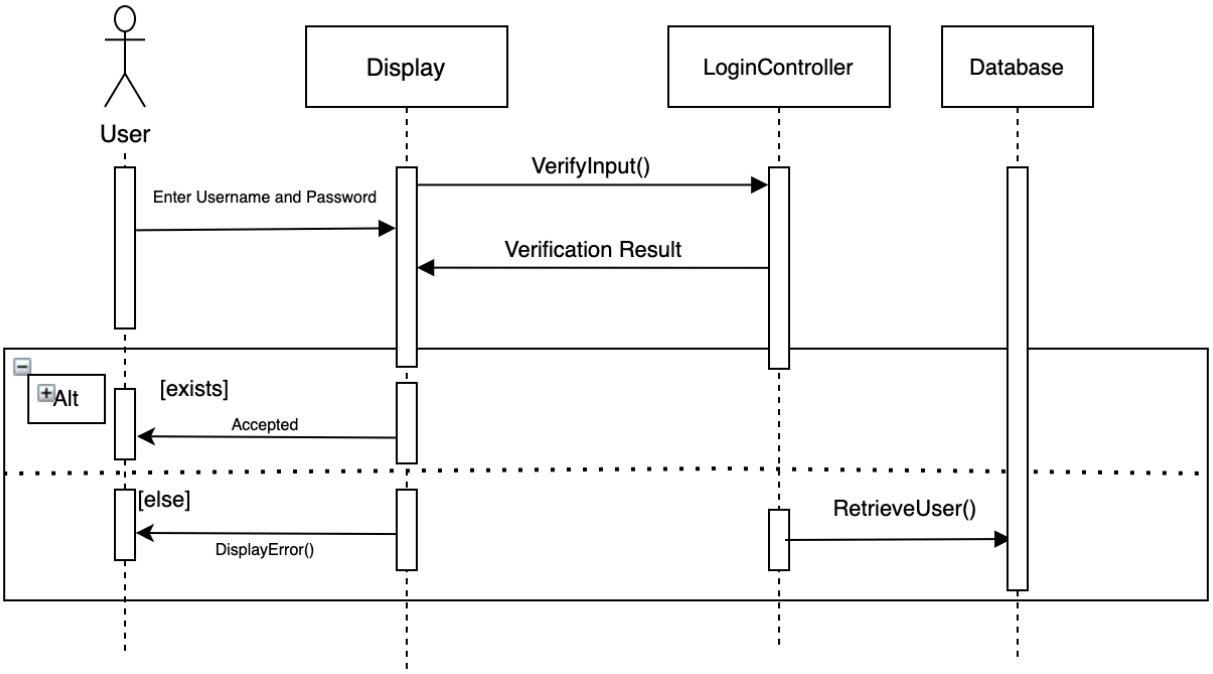
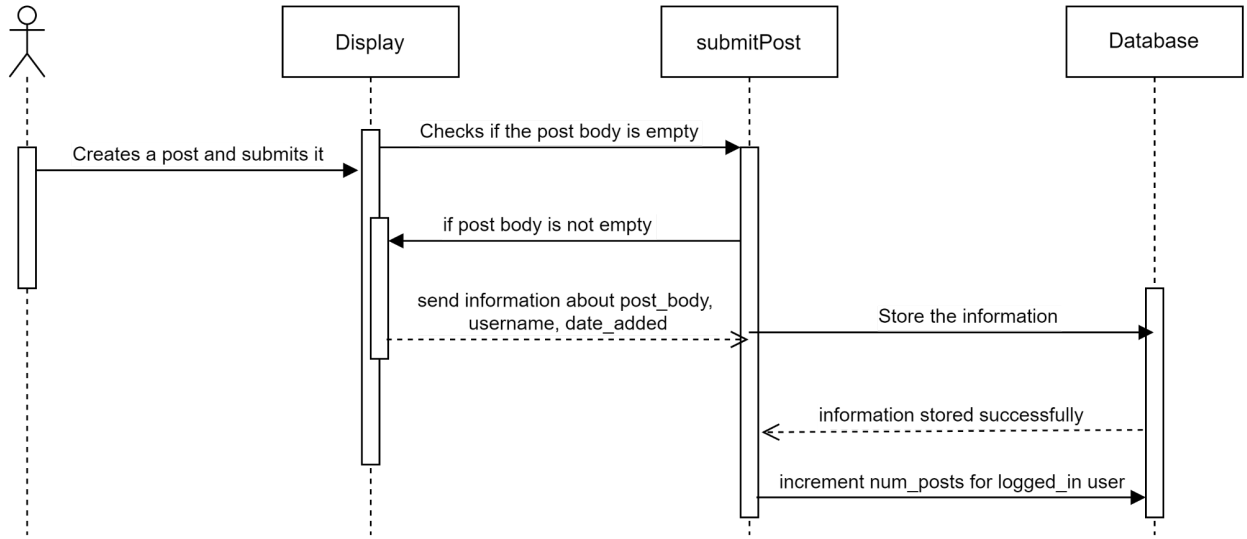


FIgure 1.9 Sequence Diagram for UC-2

Function	**Description**
VerifyInput ()	This method is part of the LoginController and verifies whether the credentials added is in a valid format so that it can be used to retrieve the user info.
DisplayError()	This method outputs an error if the input is inavlid
RetrieveUser()	This method retrieves users from a database where the users are registered

UC-6 Share/Post/Project



https://drive.google.com/file/d/1sp1syv5W7Qj-uOsJAyG23H3CmkzIgFUk/view?usp=sharing

FIgure 2.0 Sequence Diagram for UC-6

Function	**Description**
submitPost ()	This function is activated when the logged-in user fills out the information in the html form and hits the post button. The function checks whether the post body is empty or not. If the post body is not empty then it stores the session information in the database. The database responds that the information has been successfully saved and the submitPost () function requests the database to increment the total number of posts of the logged-in user.

Sequence Diagrams for UC-1, UC-2, and UC-6 were created to define interfaces.

Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
		UC-1	A user is able to register themselves onto the application's database. An HTML form is used to collect the data from the user and PHP is used to establish connection to the server and PHP is also used for storing all the data.
		UC-2	A user is able to login so that they have access to our website's features. An HTML form is used to collect the user's login credentials. And, PHP is used to connect to the database and make sure that the login information inputted by the user is correct.
		UC-3	There is a logout button on the home-page which allows the user to logout of the session. PHP is used to achieve this.
		UC-4	A user can add another user in the system by searching for them using their username and a user can remove another user as their friend. This is achieved using PHP as well.
		UC-5	Every user has a friend_array. When a logged-in user accepts another user's friend request, then that user is added into the logged-in user's friend-array and vice versa. This is achieved using PHP. Moreover, a user can remove another user as their friend. When this happens, the user is removed from the logged-in user's friend_array and vice versa.
		UC-6	A logged-in user can create a post. An HTML form is used to take the post-body from the logged-in user and once the user posts the post, it is visible to the logged-in user's friend
		UC-11	When the user logs into the system, they're led to our application's home page/ newsfeed page. This page shows the number of likes, posts and friends the logged-in user has. The logged-in user can also see their own posts as well as the posts shared by the user's friends. This is achieved using PHP as well.

		UC-12	A user can like or comment on a post that is either shared by himself or the posts that are shared by the user's friends. This is achieved using PHP
CON-1			This concern will be addressed within a frontend constraint that requires the user to define at least one other user that exists within the MySql database that will collaborate with them.
		QA-7	When a user creates an account, their credentials for their login will be encrypted and then it will be stored to ensure security of the user's personal information.
UC-7			When a user wishes to change their password, a form will be presented to the user asking them for their previous password, then asking for a new password that they wish to replace their old one with.
		QA-4	The constraint of up to date browsing software will be required by the frontend of the server, if the user doesn't have one of the listed browsers, they will be told to reconnect on an up to date version.
		CON-5	This will be done through the selected database which is SQL. Each user's actions on the server are recorded for up to 30 days.
		CRN-1	The final-system was finished in the allotted time.
		CRN-2	The work was evenly distributed amongst the team-members.
		CRN-3	The first functional prototype was successfully rendered.

Iteration 3:

Step 1

<table>
<tr><th>Category</th><th>Detail</th></tr>
<tr><td>Design purpose</td><td>To create a social networking site from a database domain. This system must be constructed to manage various contacts, posts, making comments and like/dislike comments. The main server must be able to generate, store, and retrieve specific information to specific users</td></tr>
<tr><td>Primary functional requirements</td><td>From the use cases above, the primary ones were:
UC-1 It directly supports the core functionality
UC-2 It is also one of the essentials since a user needs to login to access our application’s features
UC-6 It directly supports core functionality</td></tr>
<tr><td>Quality attribute scenarios</td><td>
<table>
<tr><th>Scenario ID</th><th>Importance to customer</th><th>Difficulty of implementing according to architecture</th></tr>
<tr><td>QA-1</td><td>High</td><td>Medium</td></tr>
<tr><td>QA-2</td><td>Low</td><td>Medium</td></tr>
<tr><td>QA-3</td><td>Low</td><td>Medium</td></tr>
<tr><td>QA-4</td><td>Medium</td><td>Low</td></tr>
<tr><td>QA-5</td><td>High</td><td>Medium</td></tr>
<tr><td>QA-6</td><td>High</td><td>High</td></tr>
<tr><td>QA-7</td><td>Medium</td><td>Medium</td></tr>
</table>
</td></tr>
<tr><td>Constraints</td><td>All constraints are important but would not be considered drivers for this iteration.</td></tr>
<tr><td>Architectural concerns</td><td>
<table>
<tr><th>ID</th><th>Concern</th></tr>
<tr><td>CRN-1</td><td>The completion of the selected system architecture within the allotted amount of time.</td></tr>
</table>
</td></tr>
</table>

Step 2

The goal of this iteration is to focus on QA-5.A user is able to upload information about themselves to create a profile within the server during normal operation. The user will send the data to the server which will be checked for any inappropriate words or phrases, then uploaded to the database. Any other user should be able to see the changes made when reviewing the user's profile within 1 minute of uploading.

Step 3

For this iteration, the elements that have been chosen to be refined are:

- Database server
- Application server

Step 4

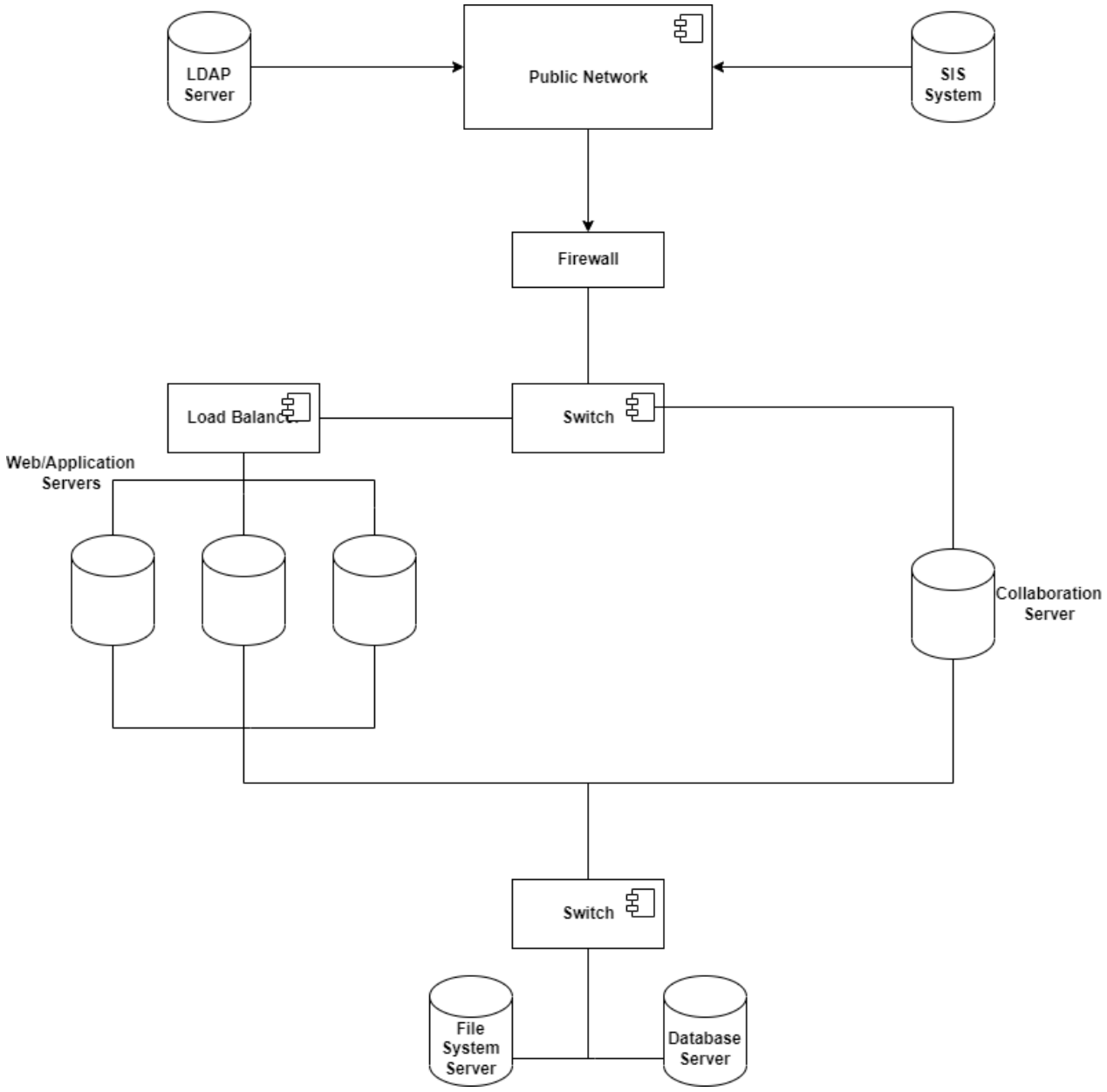
Design Decisions and Location	**Rationale and Assumptions**
Implement a **sampling management system** for the rate of data from the database.	Reducing the stream of data will greatly increase the performance of the database, thus reducing delay on requests for crucial operations such as adding a new registered user.
Introduce the **load balancing switches** tactic within the application server.	HTTP requests can become a mess for a server if it's attempting to handle many at one time. For the solution, load balancing switches are implemented to handle where the request should be redirected thus improving performance

Step 5

Design Decisions and Location	**Rationale**
Initializing load balancing switches tactic on separate server.	Initializing the load balancing tactic upon separate servers, external from the database and application server and allow for proper request redirection that doesn't affect the performance of data requests.
Use sampling management system tactic for application and database servers.	Implementing this tactic upon our servers will increase performance greatly as it reduces the amount of requests for data due to the frequency of data being decreased. Overall, the performance of the servers would be increased.

Step 6

Refined deployment diagram



Step 7

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
		UC-1	A user is able to register themselves onto the application's database. An HTML form is used to collect the data from the user and PHP is used to establish connection to the server and PHP is also used for storing all the data.
		UC-2	A user is able to login so that they have access to our website's features. An HTML form is used to collect the user's login credentials. And, PHP is used to connect to the database and make sure that the login information inputted by the user is correct.
		UC-3	There is a logout button on the home-page which allows the user to logout of the session. PHP is used to achieve this.
		UC-4	A user can add another user in the system by searching for them using their username and a user can remove another user as their friend. This is achieved using PHP as well.
		UC-5	Every user has a friend_array. When a logged-in user accepts another user's friend request, then that user is added into the logged-in user's friend-array and vice versa. This is achieved using PHP. Moreover, a user can remove another user as their friend. When this happens, the user is removed from the logged-in user's friend_array and vice versa.
		UC-6	A logged-in user can create a post. An HTML form is used to take the post-body from the logged-in user and once the user posts the post, it is visible to the logged-in user's friend
		UC-11	When the user logs into the system, they're led to our application's home page/ newsfeed page. This page shows the number of likes, posts and friends the logged-in user has. The logged-in user can also see their own posts as well as the posts shared by the user's friends. This is achieved using PHP as well.
		UC-12	A user can like or comment on a post that is either shared by himself or the posts that are shared by

			the user's friends. This is achieved using PHP
CON-1			This concern will be addressed within a frontend constraint that requires the user to define at least one other user that exists within the MySql database that will collaborate with them.
		QA-7	When a user creates an account, their credentials for their login will be encrypted and then it will be stored to ensure security of the user's personal information.
UC-7			When a user wishes to change their password, a form will be presented to the user asking them for their previous password, then asking for a new password that they wish to replace their old one with.
		QA-4	The constraint of up to date browsing software will be required by the frontend of the server, if the user doesn't have one of the listed browsers, they will be told to reconnect on an up to date version.
		CON-5	This will be done through the selected database which is SQL. Each user's actions on the server are recorded for up to 30 days.
		CRN-1	The final-system was finished in the allotted time.
		CRN-2	The work was evenly distributed amongst the team-members.
		CRN-3	The first functional prototype was successfully rendered.