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

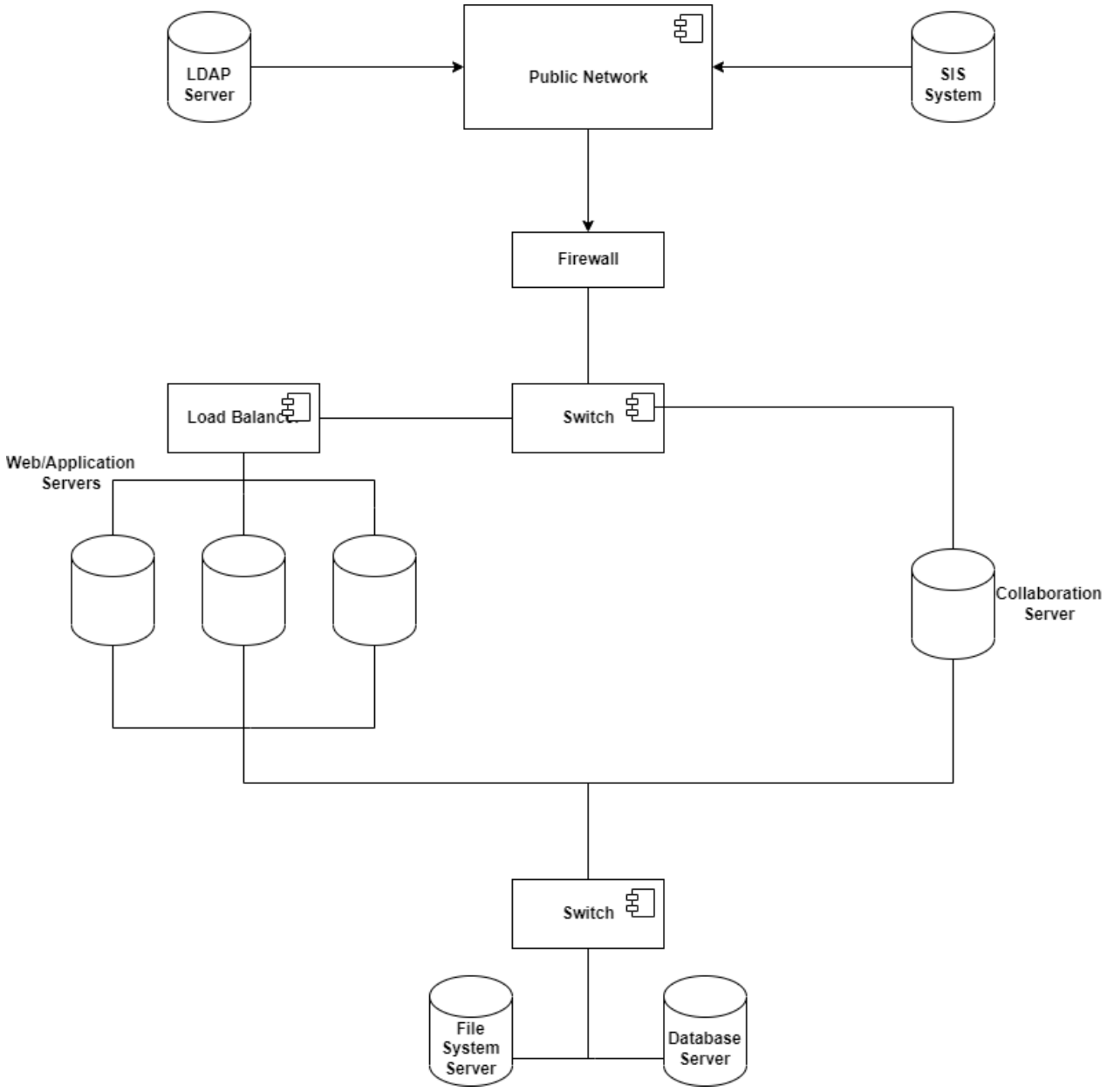
Design Decisions and Location	**Rationale and Assumptions**
Implement a **sampling management system** for the rate of data from the database.	Reducing the stream of data will greatly increase the performance of the database, thus reducing delay on requests for crucial operations such as adding a new 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.