Part-A)

- My web system is based on a Saskatoon based blood donation camp which helps the camp organizers to add/delete staff members as well as customers.
- The main architecture of the web system is that I used HTML and CSS to create webpages and using the power of NodeJS to route them. For storing staff and customer credentials, I used MySQL. There are 2 tables in the *project* database,
    - Staff - to store staff members
    - Customer - to store customers
- On a test run when the user first visits the home page of the web system, the login webpage appears. The user puts in their credentials and the form data is sent to the server where it is queried in the MySQL database. If the user is found, the user is redirected to the staff webpage where he/she can perform administrative steps like adding/deleting existing staff and customer. If not, then the user is redirected to the login page with an error message. NOTE, I had to create an admin staff member through MySQL command-line to login for the first time.
- On the staff webpage, there is a side menu bar from which the user can decide which tasks to perform.
- To add a new staff member, the suer can fill in the staff register form, and the data is sent to the server where it is validated. Upon successful validation, a new staff member is added to the MySQL database. The newly created staff member is added to the staff list which is then updated.
- If the user wants to delete a staff member, he/she just must fill in the username of the staff member which is sent to the server where is queried in the database and deleted if found.
- Similar tasks can be performed in the customer webpage.
- If the user wishes to logout, he/she can just press logout from the side menu bar upon which they are redirected to the login page.
- To test the NodeJS server, I used loadtest package. I also performed some dummy runs to test the CRUD tasks.

Part-B)

- I decided to use MongoDB as my new technology.
- Following are the reasons why I chose it:
    1. It is highly and easily scalable
    2. Beginner friendly syntax
    3. Stores data as JSON-documents which are easier to understand if coming from a Javascript background

4. We can store large volumes of data with no structural requirement

Part-C)

1. Stores records in a document database
    - MongoDB stores data as JSON document.
    - The document data model maps naturally to objects in application code, making it simple for developers to learn and use.
    - The document model provides flexibility to work with complex, fast-changing, messy data from numerous sources. It enables developers to quickly deliver new application functionality.

2. Collections: grouping documents
    - In MongoDB, a collection is a group of documents.
    - If you are familiar with relational databases, you can think of a collection as a table. But collections in MongoDB are far more flexible.
    - Collections do not enforce a schema, and documents in the same collection can have different fields.
    - Each collection is associated with one MongoDB database.

3. Data models
- Data in MongoDB has a flexible schema. Collections do not enforce document structure by default. This flexibility gives us data-modeling choices to match us application and its performance requirements.
- MongoDB also provides schema validation during insertions and updates.

Part-D)

- As the new technology deals with data storage, the front-end has not changed much except for the required changes to accommodate the new technology.
- The NodeJS server experienced most of the changes.
- I used a utility called Mongoose to communicate between NodeJS server and the MongoDB database. Using MongoDB Atlas, I created a new cluster and added the database *project*. Inside the database, I created 2 collections:
    1. Staffs - to store the documents containing staff member credentials

2. Customers - to store the documents containing customer credentials

- When the user tries to login from the login page, the data is sent over to the server where is queried in the staffs collection. If found, the user is redirected to the staff webpage. In here, the user can perform similar CRUD functions as seen in Part-A.
- Even though it was my first time trying with MongoDB, making a transition from MySQL to MongoDB was very smooth.
- The API documentation was thorough and very helpful.
- The syntax is very easy to catch up on and the JSON document syntax is also a relief and queried data can be easily sent to the client end to update the webpage.