Lab 3 Deliverables

Introduction

This document outlines the key design issues, system architecture and the code skeleton for our project. Taking into account the nature of the project, our team has chosen to use the MERN stack to develop our web application; with React as our frontend library, Node.js as our server runtime environment, Express as our web framework and MongoDB as our database. The use of libraries and frameworks like React and Express has allowed the extensibility of some of the logic encapsulated within the library/framework, allowing us to not have to implement all logic stated in the class diagram.

3.1 System Architecture

Our team has chosen to adopt the Model-View-Controller (MVC) architectural pattern. The views will be our UI done with React while the Controllers will be the business logic contained in our RESTful APIs developed with Express. The models will then be the document type that we store in our database.

As we had developed our class diagram with the MVC pattern in mind in Lab 2, our class diagram could also serve as our architecture diagram. We segregated the class diagram into 3 sections, with the most left section being Views then the Controllers in the middle and Models on the right. Following the MVC pattern, we ensured that there is no direct communication between the View and Model. All communications are handled by the Controllers.

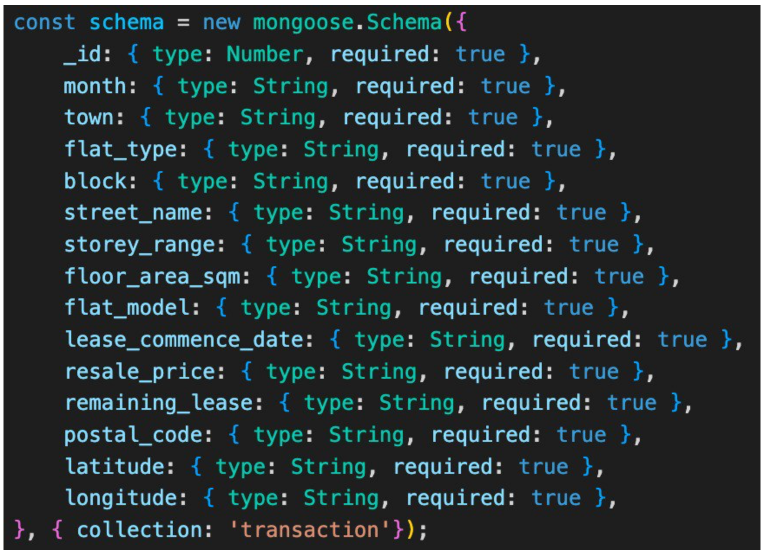
To showcase the design patterns adopted, we have created another class diagram, separate from our UML diagram, to include some additional classes. However, we did not implement these classes as they have been implemented by libraries and we can extend the use of it. An example would be HomePage observer can be performed by the useEffect hook in React to observe for changes in dependencies while the HomeCtrler can be performed by the Axios to make the necessary API calls.

3.2 Key design issues

<u>Data Persistence</u>

In our design, we identified that the persistent data in our design is the transaction data. The transaction data will help buyers and sellers to gauge how much they could potentially buy and offer a price relative to past transactions. The data is persistent as users are able to extract the data even after the webpage has been closed and restarted.

We have decided to use MongoDB Atlas. MongoDB is a document-oriented, noSQL database that stores data in documents similar to JSON. Due to the large data volume of our persistent data, MongoDB is suitable for handling our dataset due to its high scalability and fault-tolerant design.



```
const schema = new mongoose.Schema({
    _id: { type: Number, required: true },
    month: { type: String, required: true },
    town: { type: String, required: true },
    flat_type: { type: String, required: true },
    block: { type: String, required: true },
    street_name: { type: String, required: true },
    storey_range: { type: String, required: true },
    floor_area_sqm: { type: String, required: true },
    flat_model: { type: String, required: true },
    lease_commence_date: { type: String, required: true },
    resale_price: { type: String, required: true },
    remaining_lease: { type: String, required: true },
    postal_code: { type: String, required: true },
    latitude: { type: String, required: true },
    longitude: { type: String, required: true },
}, { collection: 'transaction'});
```

Example of how our data is stored.

Providing Acces Control
Despite having two actors, both buyers and sellers have access to the same functions and data. Therefore, there is no issue of access control in our design.

3.3 Code Skeleton

Our team has organised the code into 2 main folders:

- revaluate_client
- revaluate_server

The client folder contains the code for the development of the Views (UI) and the components within. The server contains the code for the development of the business logic and data persistence (Controllers and Models). Within the server folder also contains a Data_ETL folder which contains the code for the Data_Extract_Load pipeline to populate data into our database.

Folder name: revaluate_client

Sub-Folder/File Name	**Purpose**
public	Contains static files that form the base of the web application
src/components	Contains individual components that are nested and reused in views
src/screens	Contains individual views
src/App.js	Main parent component of the web application that will render the various views
package.json	Contains name and version of dependencies used

Folder name: revaluate_server

Sub-Folder/File Name	**Purpose**
Data_ETL	Data ETL pipeline to retrieve data from government API then transform data by adding latitude and longitude by calling OnePlace API and load into MongoDB
Models	Contains definition of models
index.js	Main file that performs server initialisation, connection to MongoDB and definition of controllers
package.json	Contains name and version of dependencies used