2022

Project – Milestone #5

SOFTWARE ARCHITECTURE

GURPREET PUNJ 200003134

Change list:

1. Changed the design patterns.
2. Changed the verbiage of requirements as they were not clear
3. Added more details and clarity to the UMLs
4. Added the stakeholders and UML technique information as well

Milestone #1

Proposed software Name: WOW Lite

Briefdescriptionofpurpose:I am thinking of using the Blizzard API (World of warcraft game) to create something interesting. Below is the documentation. I always wanted to create small plugin/addon or web app which has something to do with WOW in general, as I play the game on regular basis.
https://develop.battle.net/documentation/world-of-warcraft/game-data-apis

Target Audience of the software: World of warcraft players Following

things can be done using the API and potentially the application:

1. Users will be able to sort and filter the data based on search parameters.
2. In Game Marketplace data shown outside which let users search and sort the data.
3. Report generation based on marketplace data (It is dependent on the server)
4. This will be for a single user who has the code running locally, the application will not implement login functionality. It will be tied to my account.

Design Patterns:

1. Strategy pattern for the sorting algorithms. Strategy pattern will use different algorithm based on the search parameter and categories. Implemented Bubble sort, selection sort and insertion sort which will sort the data based on itemId, name, and quantity respectively.
2. Singleton pattern is used for DB instance management. The reason I did that, and not implement observer pattern is due to DB sitting in cloud. I did not want to pay extra as I have limited credits available.
3. MVC pattern is implemented in this scenario to have separation of concerns.

Technologies used:

1. NodeJS (BackEnd and will be easier to use with API's)
2. FrontEnd could be just simple HTML using the .ejs templates)
3. Database is MySQLite sitting on Azure.
   Link: https://portal.azure.com/?quickstart=true#@mcmaster.ca/resource/subscriptions/e63156f3-f22a-458f-9534-31563597a73b/resourceGroups/wowlite/providers/Microsoft.DBforMySQL/flexibleServers/wowlite/overview

Milestone #2

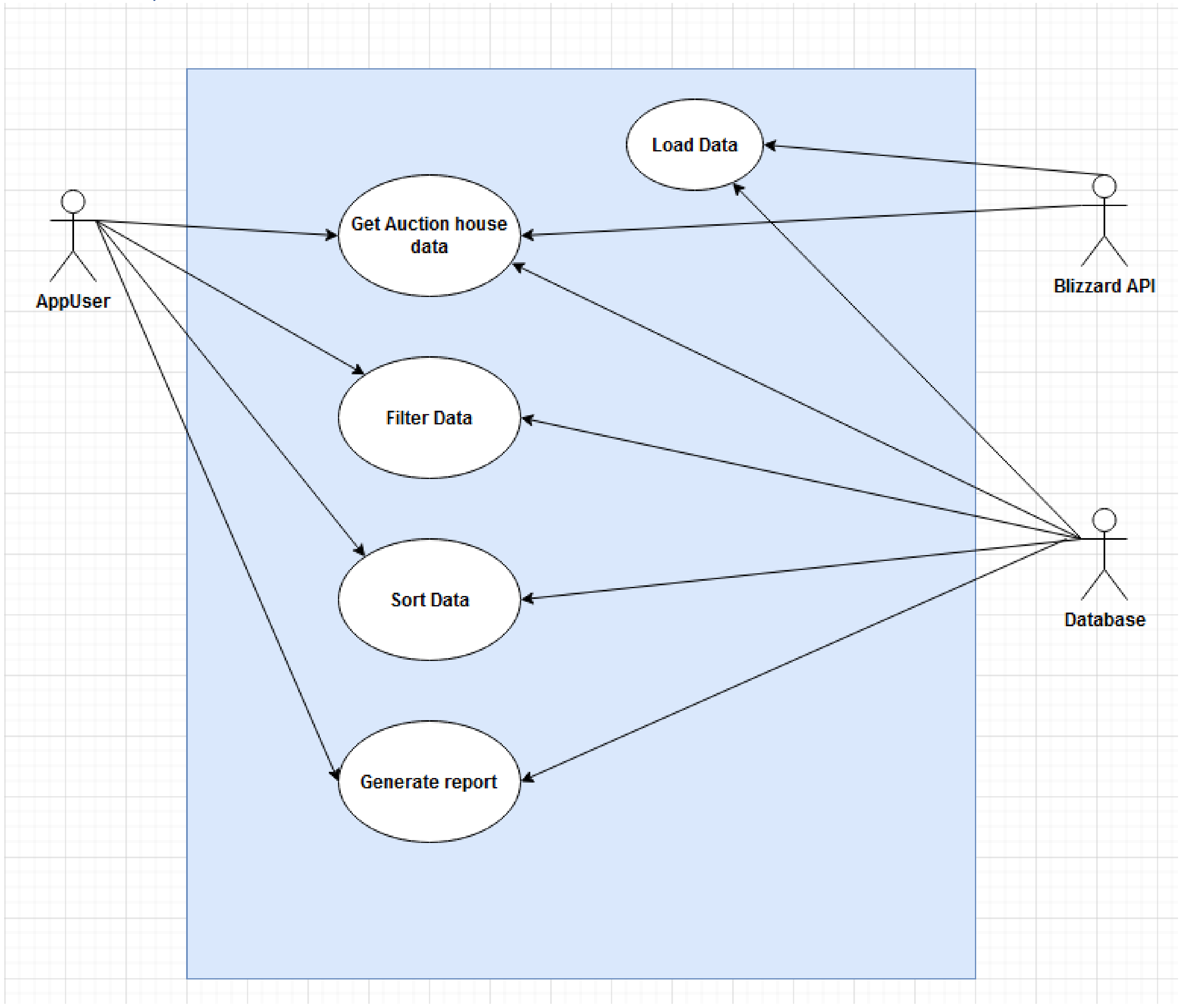
Requirements

1.0	Users will be able to search the auction house data from 1 realm in world of warcraft game which will be updated on demand.	✓
1.1	On demand data will only refresh every hour from server side.	✓
2.0	Users will be able to sort the data based on item name, itemID and quantity	✓
3.0	Users will be able to generate a report using the auction data. The report will show the data of itemID along with the respective quantities.	✓
4.0	User search and filter will return results in less than 5 seconds.	✓
5.0	The application will be compatible with any operating system running node JS and supports JavaScript.	✓
5.1	Users need to have a Blizzard account to make new API requests.	✓

DB Queries

Auctions SQL	Items SQL
DROP TABLE IF EXISTS auction; CREATE TABLE auction(id INTEGER NOT NULL PRIMARY KEY ,itemid INTEGER NOT NULL ,quantity INTEGER NOT NULL ,unit_price INTEGER ,time_left VARCHAR(5) NOT NULL ,itemcontext INTEGER ,itembonus_lists0 INTEGER ,itembonus_lists1 INTEGER ,itemmodifiers0type INTEGER ,itemmodifiers0value INTEGER ,itemmodifiers1type INTEGER ,itemmodifiers1value INTEGER ,buyout INTEGER ,itempet_breed_id INTEGER ,itempet_level BIT ,itempet_quality_id INTEGER ,itempet_species_id INTEGER);	DROP TABLE IF EXISTS items; CREATE TABLE items(itemid INTEGER NOT NULL PRIMARY KEY ,typename VARCHAR(17) NOT NULL ,value INTEGER NOT NULL ,is_negated VARCHAR(4) ,displaydisplay_string VARCHAR(20) NOT NULL ,displaycolorr INTEGER NOT NULL ,displaycolorg INTEGER NOT NULL ,displaycolorb INTEGER NOT NULL ,displaycolora BIT NOT NULL);

Scenario viewpoint



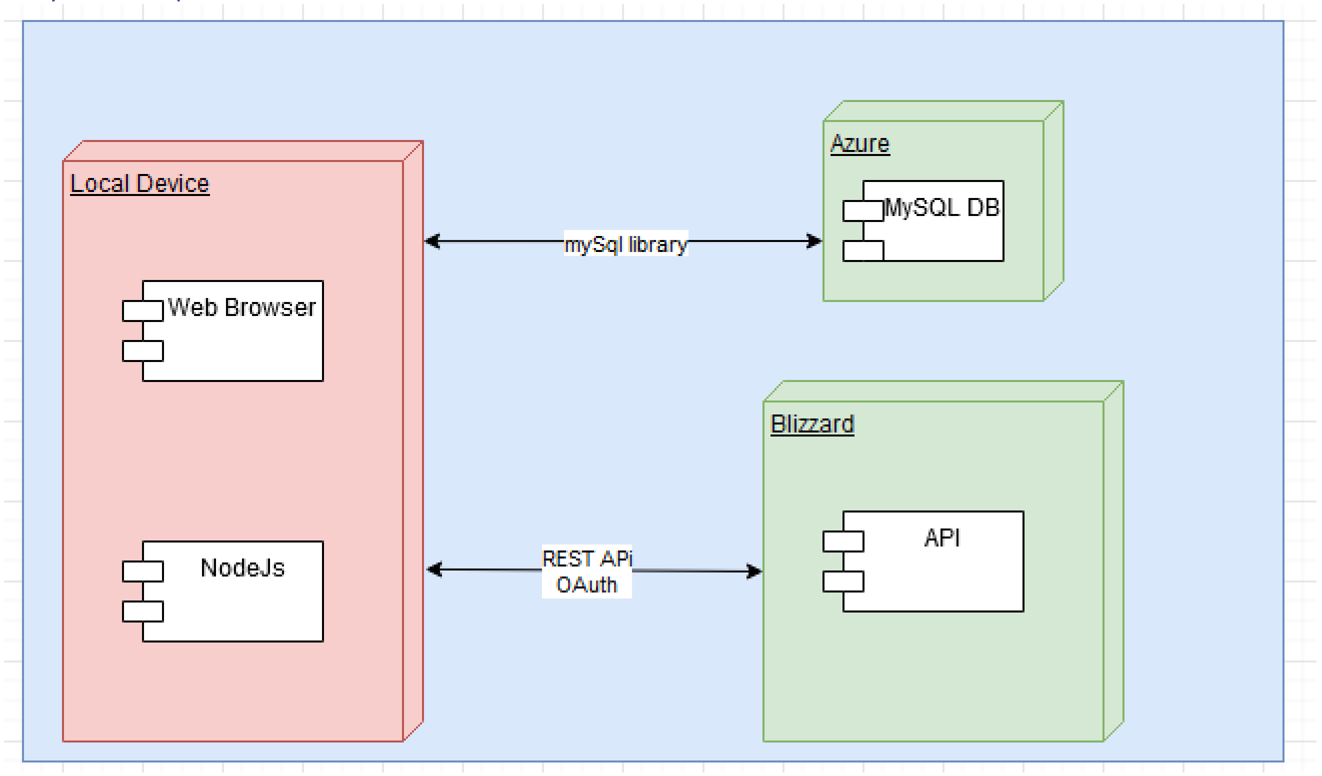
The above viewpoint is showcasing the high-level functionality of the application. It includes load the data into the database by making the API call, and the capabilities user will have in the app. The

Concerns: (Assisting with functionalities and use cases of the project)

Stakeholders: All the stakeholders and the end user

Modelling technique: UML Use case diagram

Physical viewpoint



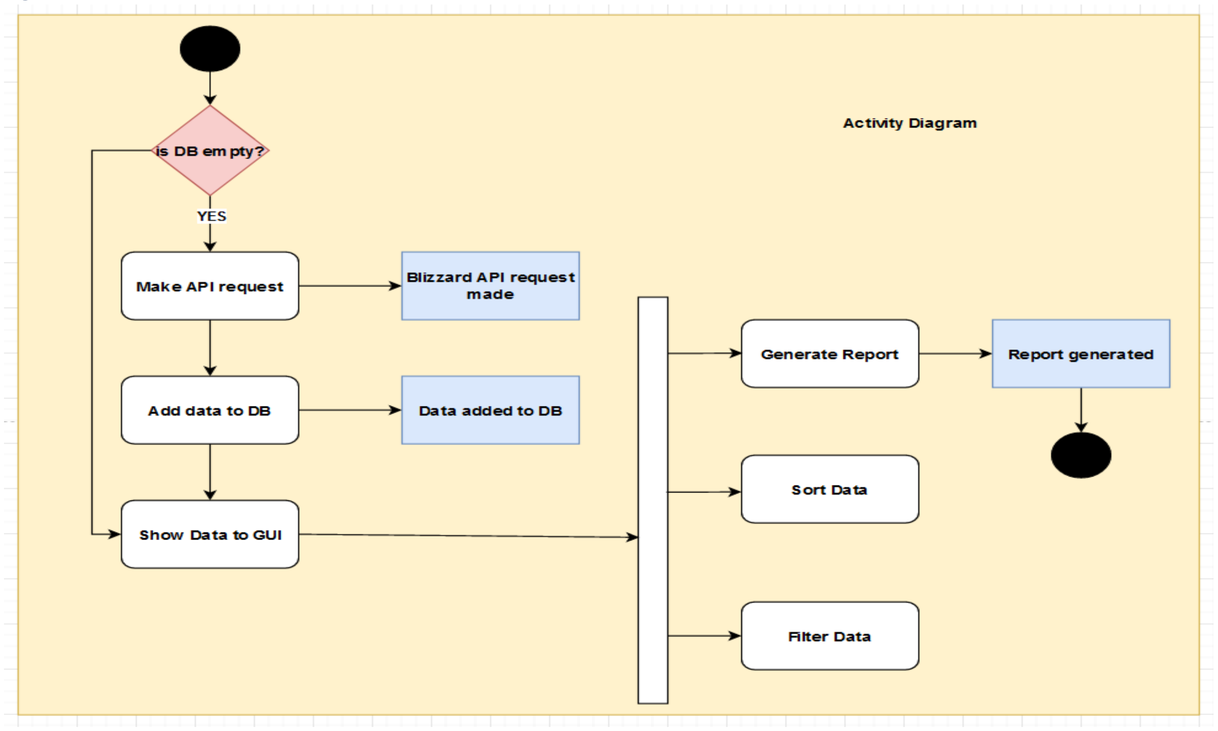
The cloud database is sitting in Azure service. The NodeJS app is running locally on client's computer and making calls to the API and DB.

Concerns: (Assisting with the deployment activities)/ Mapping the data from software to hardware components

Stakeholders: Software developers, designers, and architects

Modelling technique: UML Deployment diagram

Logical ViewPoint



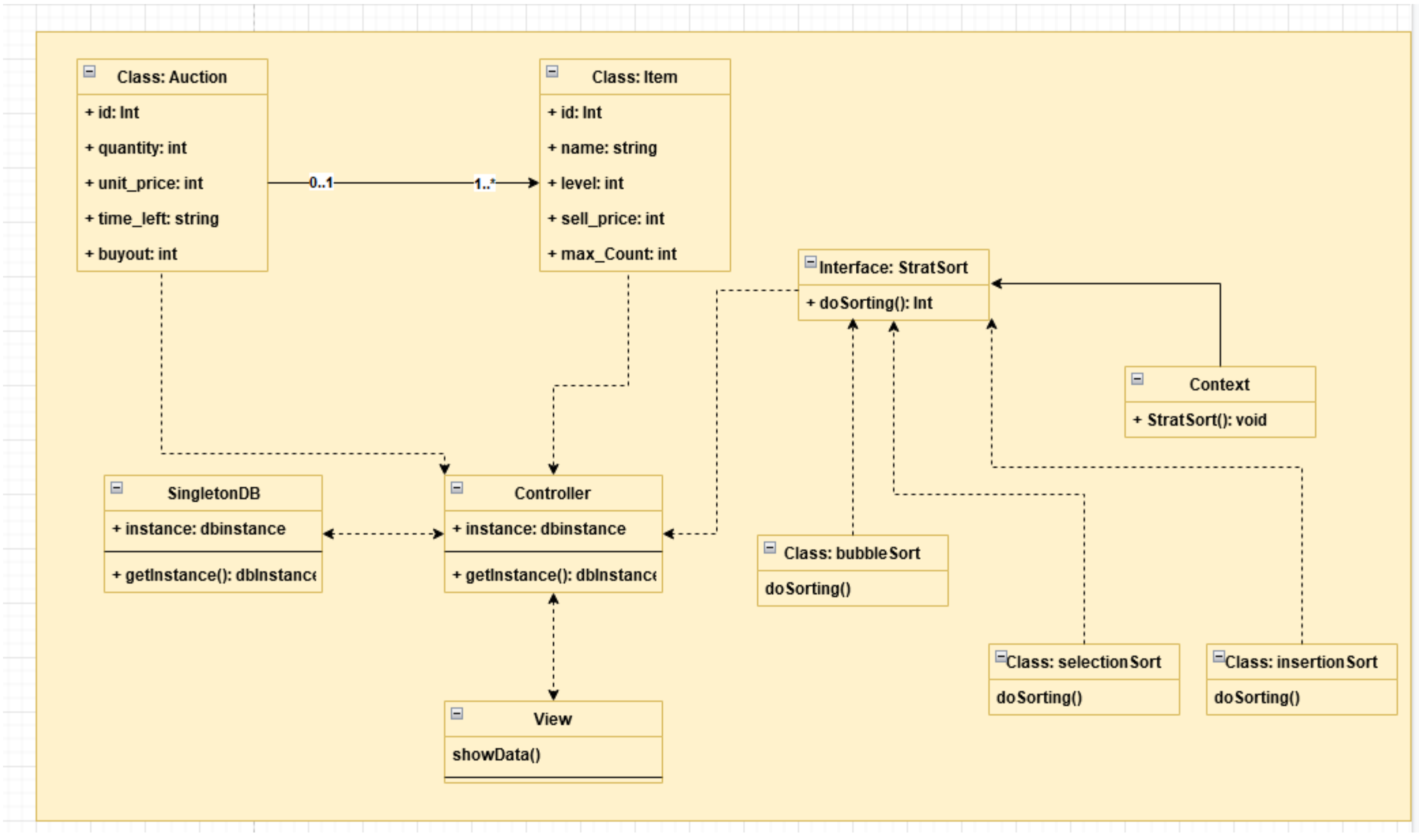
The above activity diagram is showcasing how flow of the overall application and different activities that take place.

Concerns: Functional requirements (Assisting to understand the logic and flow of the project)

Stakeholders: Architects and end users

Modelling technique: UML Activity diagram

Development Viewpoint

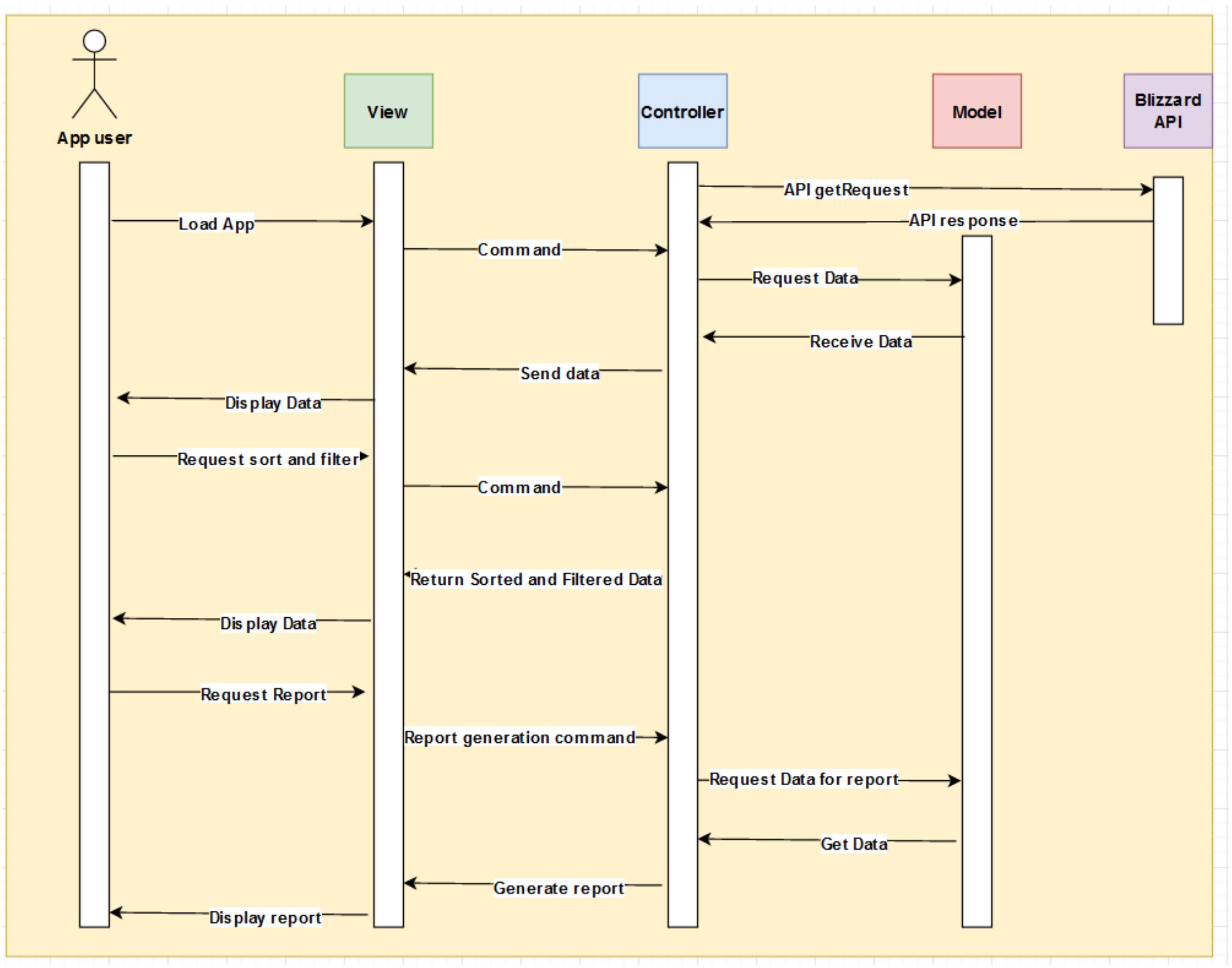


1. Auction and Item are the models in this diagram.
2. Controller is handling the communication between models and other design pattern clases.
3. There are three design patterns in above implementation. Strategy, MVC and Singleton.
4. Finally, view is loaded after all the interactions are complete.

Concerns: Classes and software module organization (Assisting with the development of the project)
Stakeholders: Developers

Modelling technique: UML Class diagram

Process Viewpoint



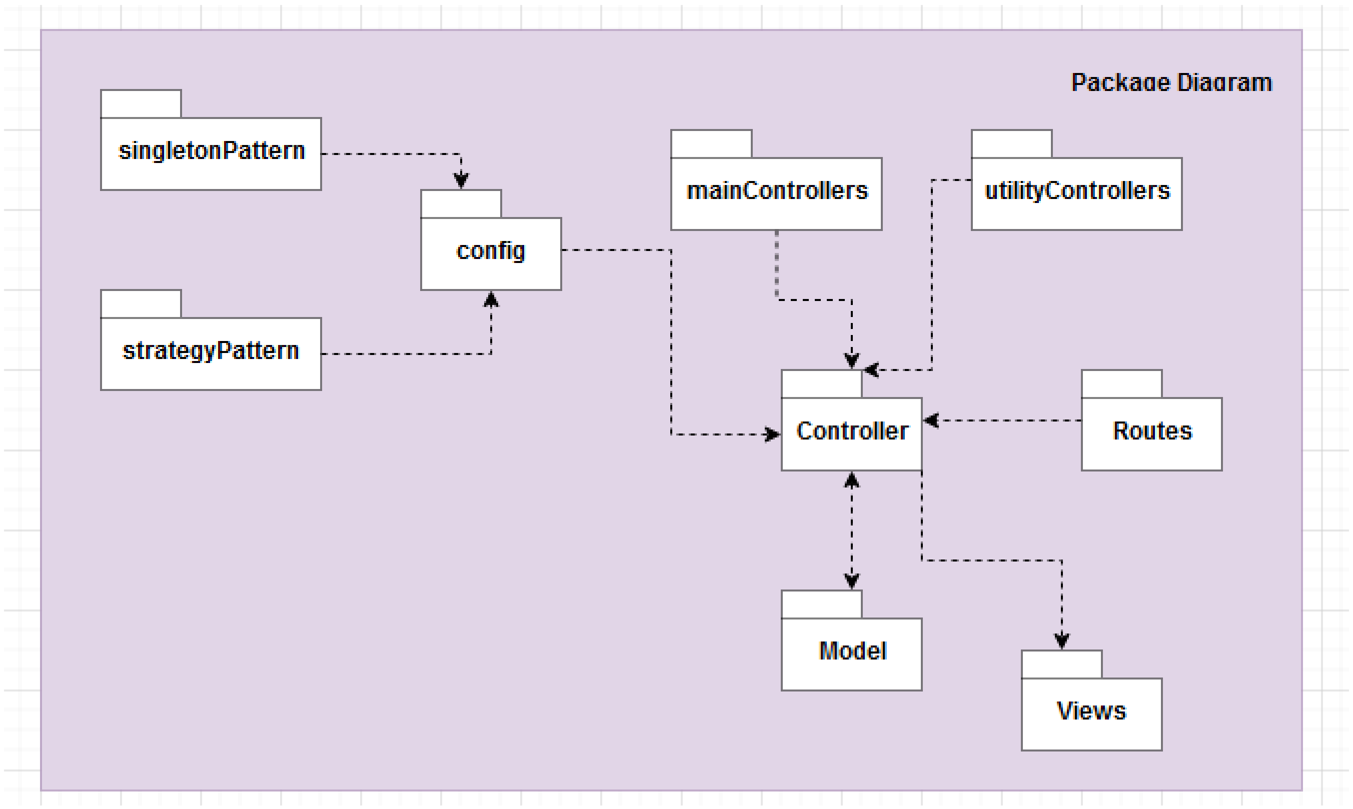
The process viewpoint showcases how MVC pattern is used (model-View controller), and how the interactions happen between the user, API and MVC pattern itself. The UML technique is used. It starts with loading the app by making the API calls, and inserting the data into the DB. After DB request is done using the Model queries and objects, controller and routes handle different requests and pass it to the right methods. View is finally loaded after the controller work is done, which is used by the user to view the data.

Concerns: Runtime communication (Assisting to understand the process and call flows in the project)

Stakeholders: Architects and Designers

Modelling technique: UML Sequence diagram

Package diagram



The above diagram is UML package diagram showcasing the folder structure. Below is the summary of each folder.

1. Config folder is implementing the functionalities of design patterns.
2. Controllers are using the config and implementing them in the methods. mainControllers handles the controllers based on models, and utility controllers handle controllers from config.
3. Routes are just an extension of controllers to handle all the routing logic of the application
4. Model and views folder are typical MVC folders to handle DB queries and HTML data respectively.

Concerns: (Assisting to understand the structure of the project in a quick glance)

Stakeholders: Software developers and managers

Modelling technique: UML Package diagram