AI Investment Advisory

Software Design (SD)

(https://github.com/ChubsB/AI-Based-Stock-Investments)

BY

Team Lead: ***Syed Hussain Imam/19773***

Member(s):

***Ahmed Abdul Ghafoor 19735***

***Huzaifa Farooq 19766***

***Mohammad Razi Moosa 18663***

PROJECTS COMMITTEE (PC)

SUPERVISOR: ***Tahir Syed***

CO-SUPERVISOR: -

Member(s): -

**SUBMITTED TO**

pROJECTS Manager – FYP

**ON**

DATE (27/02/2023)

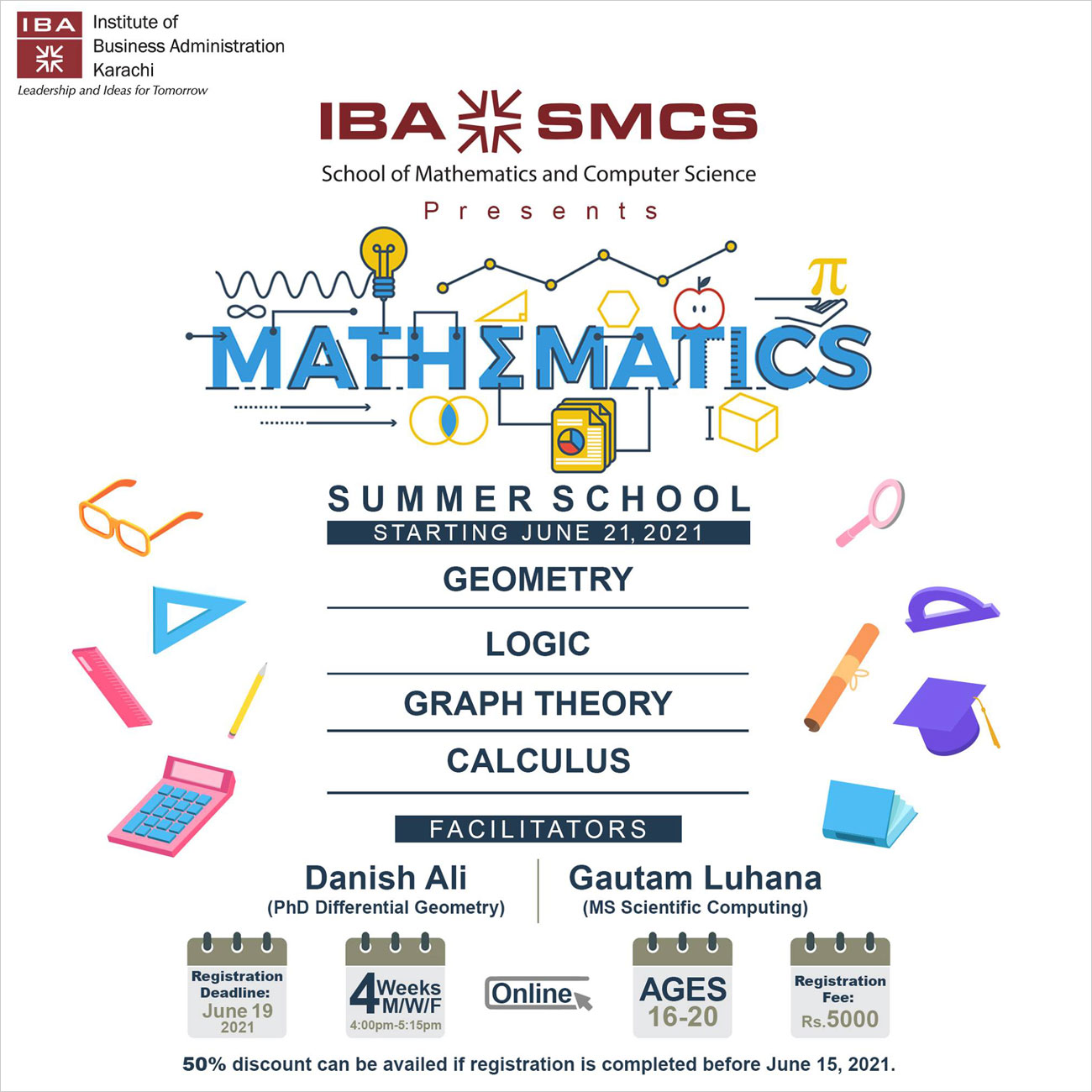


Table of Contents

[***PROJECTS COMMITTEE (PC)*** 1](#_Toc95986055)

[1. Interaction Diagrams 3](#_Toc95986056)

[2. Class Diagram and Interface Specification 3](#_Toc95986057)

[3. System Architecture and System Design 3](#_Toc95986058)

[3.1. Architectural Style 3](#_Toc95986059)

[3.2. Identifying Subsystems 3](#_Toc95986060)

[3.3. Mapping Subsystems to Hardware 3](#_Toc95986061)

[3.4. Persistent Data Storage 3](#_Toc95986062)

[3.5. Network Protocol 3](#_Toc95986063)

[3.6. Global Control Flow 3](#_Toc95986064)

[3.7. Hardware/Sofware Requirements 3](#_Toc95986065)

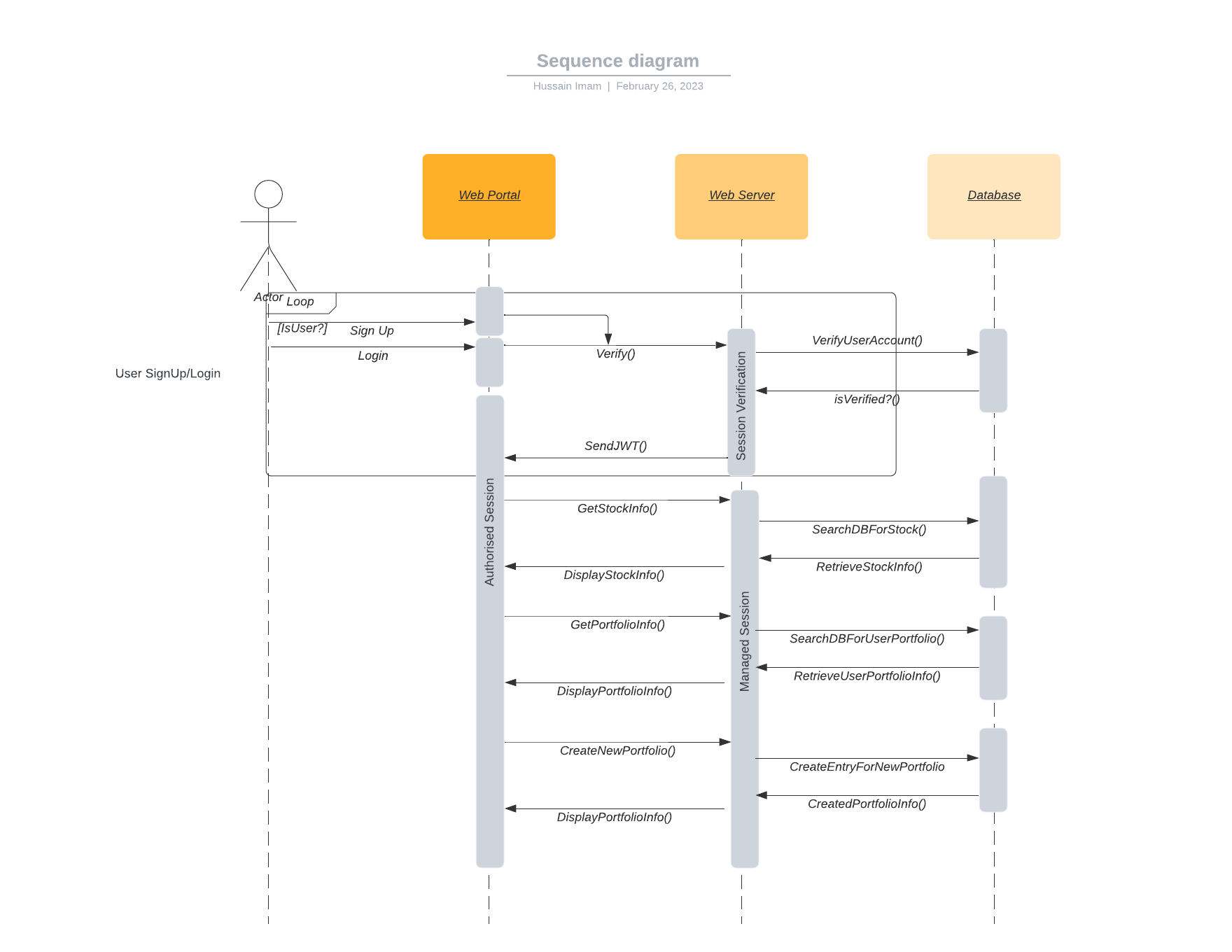
[4. Algorithms and Data Structures 4](#_Toc95986066)

[5. User Interface Design and Implementation 4](#_Toc95986067)

[6. Design of Tests 4](#_Toc95986068)

System Design

# Interaction Diagrams



# Class Diagram and Interface Specification

Show all classes and their associations. If cannot fit on a single-page, make an overview diagram showing all classes and their associations with just name. Details with attributes can be depicted in partial diagrams.

Use UML notation for class specification with datatypes for each class.

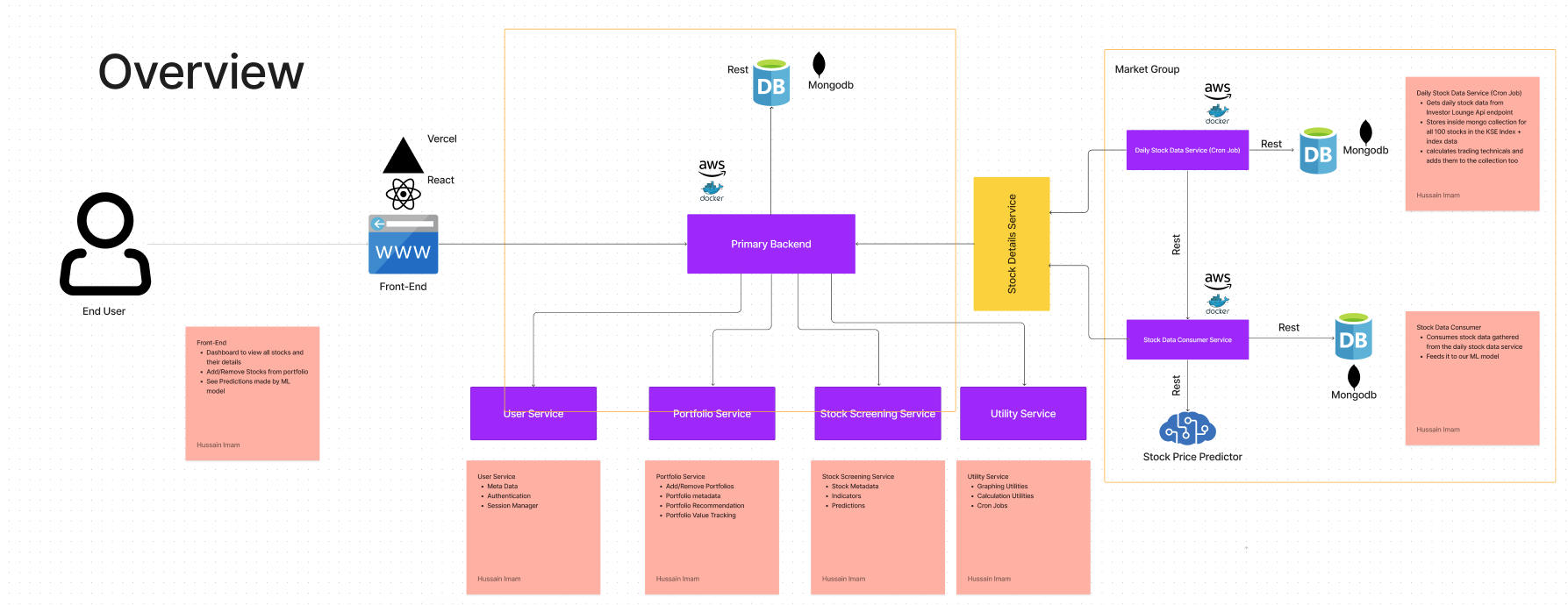
# System Architecture and System Design

## Architectural Style

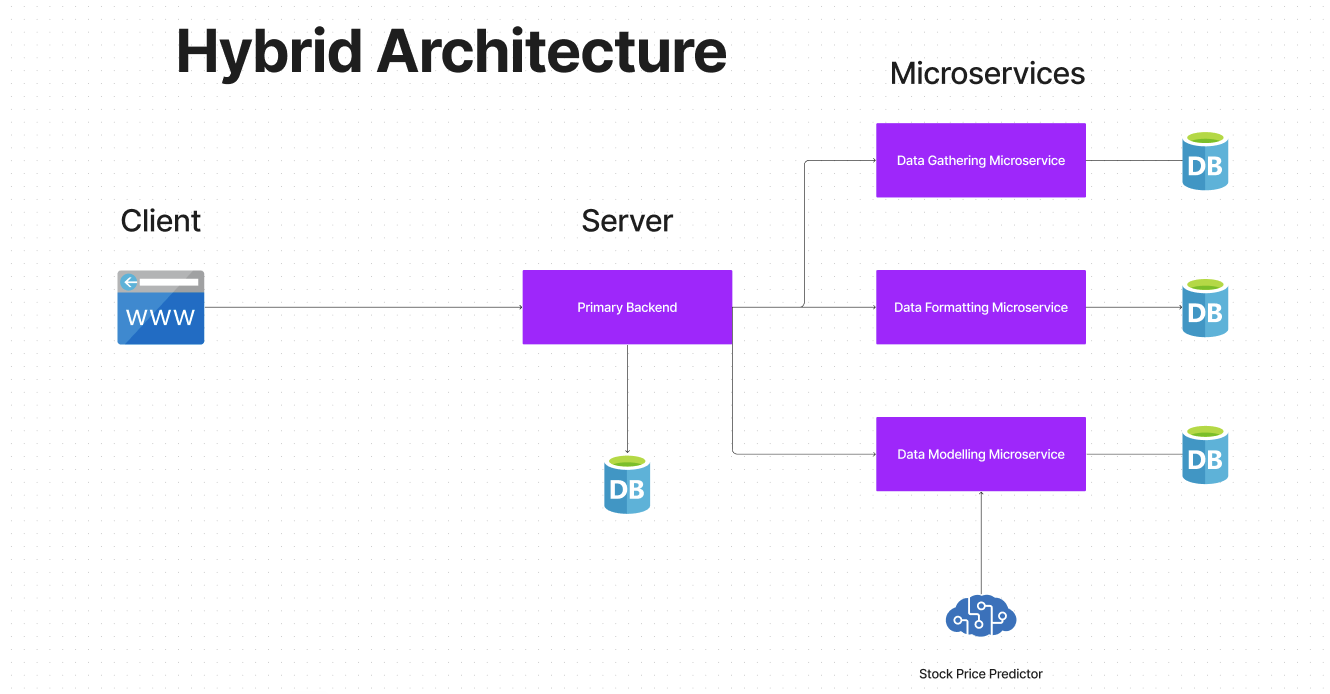
Client – Server Architecture

Our system is segregated into two major components our client (Browser Portal) and our server (Nodejs Backed) + Database (MongoDB/MySQL). Although for the most part our architecture is heavily centered around Client – Server fundamentals. Some of our services fall outside our monolithic backend and follow the microservice pattern i.e. Data collection, Data Formatting and Data modelling services

**Complete Architecture:**

****

**Client – Server + Microservices:**

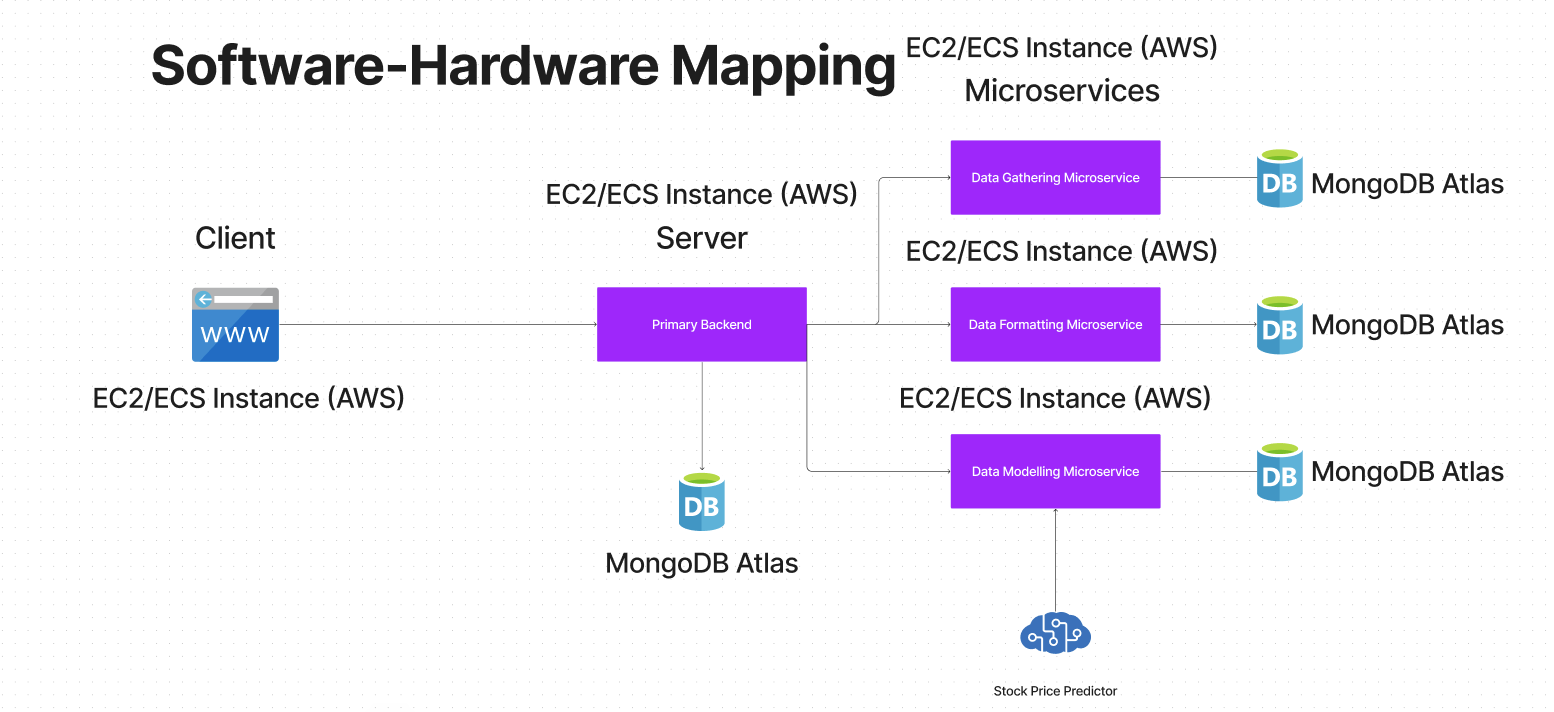


## Identifying Subsystems

## Mapping Subsystems to Hardware

Components in our architecture:

* Client (Reactjs Frontend): EC2 instance (AWS)
* Primary Backend (Nodejs): EC2 instance (AWS)
* Microservices: EC2 instance (AWS)
* Database: EC2 instance (AWS) / MongoDB Atlas



## Persistent Data Storage

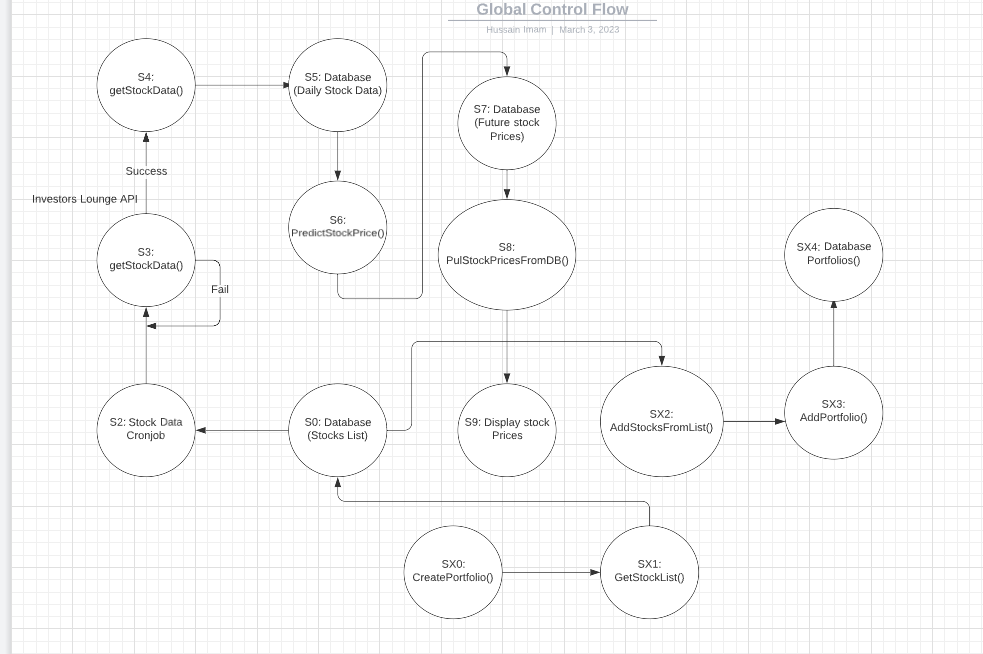
Attach the description of the file format and/or database schema (format of database tables, printed by the command description)

## Network Protocol

Our application will be following the principles of RESTful web architectures, hence all communication will use HTTP/HTTPS.

## Global Control Flow

You’ll handle execution orderness (linear vs event-driven), time dependency (use of timers) and concurrency (multiple threads) issues here in this section



## Hardware/Sofware Requirements

**Recommended Operating Systems:**

Windows: 7 or newer

MAC: OS X v10.7 or higher

Linux: Ubuntu

**Hardware Requirements:**

We strongly recommend a computer fewer than 5 years old.

* Processor: Minimum 1 GHz; Recommended 2GHz or more
* Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)
* Hard Drive: Minimum 32 GB;Recommended 64 GB or more
* Memory (RAM): Minimum 1 GB; Recommended 4 GB or above

**Supported Browsers:**

People often ask what browser they should use. There is no single answer for this. Use whichever browser works best on your computer. However, we recommend downloading Firefox and/or Chrome in addition to having Internet Explorer or Safari.

* FireFox
* Chrome
* Edge
* Safari
* Opera
* Brave

# Algorithms and Data Structures

Describe the complete algorithms your system use. Also specify and complex data structure (arrays, link lists, hash tables, trees etc.) your system uses.

# Design of Tests

Tests will be conducting using the Nodejs testing library called Jest (Testing Framework)

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Test Case | Input Value | Expected Output |
| 1 | Sign Up | Email: [xyz@gmail.com](mailto:xyz@gmail.com)  Password: 1234Admin | Auth Token: JWT \*\*\*\* |
| 2 | Login | Email: [xyz@gmail.com](mailto:xyz@gmail.com)  Password: 1234Admin | Auth Token: JWT \*\*\*\* |
| 3 | Get stock data | Ticker: ABL | ABL Stock Data Object |
| 4  5 | Get Portfolio  Create Portfolio | Portfolio\_id: 3A21  PortfolioName: XYZ  Stocks: {  ABL: 25%  ENGRO: 25%  PIBTL: 50%  } | User Portfolio Object  Success: 200 (Reponse) |
| 6 | Edit Portfolio | Portfolio\_id: 3A21  Stocks: {  ABL: 55%  PIBTL: 45%  } | Success: 203 (Reponse) |

\*Frontend testing will consist of manual end-end testing according to specified user journeys.