Orange and white text on a black background

Description automatically generated

|  |  |
| --- | --- |
|  | **ASSESSMENT 1 – Group Assessment** |
| **Student Name** | Mohammaduzair Mohammadyahya Shaikh | A00126661 |
| Student Name | Mohammed Isa | A00118725 |
| **Subject Code and Title** | PBT205—Project-based Learning Studio: Technology |
| **Assessment** | Prototype Development |
| Learning Facilitator | Nyasha Shoniwa |

# Task – 2 Trading System

### Application Overview

This System was created using RabbitMQ and Node.js to simulate a stock exchange. Orders are processed, matched, and trades are carried out between buyers and sellers. With various components for handling orders, order books, and trade execution, the system is modular.

### Functionality

Exchange Management: The order books for different stocks are initialized and managed by the exchange.js code. To receive incoming orders and process them appropriately, it creates RabbitMQ queues.

Order handling: Buy and sell orders are sorted and matched for trade execution by the orderBook.js code. The order is entered into the book if no matches are discovered.

Trading Execution: Users can send buy or sell orders using the trader.js file. Orders are published to RabbitMQ, where they are processed and consumed by the exchange. Orders that are successful are reported by the system and are shown in a table.

Configuration: The config.js file manages stocks and RabbitMQ configuration settings, making sure everything is configured appropriately before operations start.

RabbitMQ Integration: Reliable messaging and queue management are provided via the rabbitmq.js file, which controls connections and channels with RabbitMQ.

Validation: To guard against mistakes and guarantee data integrity, the validation.js file verifies configuration parameters and user input.

### Running the Application

To execute this application, we follow these steps

Install require Dependencies: Install Node.js Package

Terminal: npm Install

Execute the Exchange: Run the Exchange to start processing orders

Terminal: node src/exchange/exchange.js

Place the Order: Execute the trader.js to place the order

Terminal: node src/trader/trader.js <username> <stock> <BUY|SELL> <price>

Terminal: node src/trader/trader.js Uzairr XYZ BUY 100

### Error Handling

There are several levels of robust error handling in the program. To prevent inconsistent states, thorough error messages are logged, and the process terminates gracefully if there are problems with RabbitMQ connections, invalid user inputs, or configuration issues.

### Usefulness

This application is a useful tool for learning and testing trading algorithms, market behaviours, and order matching processes since it replicates a real-world stock exchange environment. It is dependable and instructive for users interested in financial technology and system architecture because of its modular design and mistake handling.

### References:

RabbitMQ Documentation Team. (n.d.). *RabbitMQ documentation* [Website]. Retrieved from <https://www.rabbitmq.com/docs>

[Node.js](https://node.js/)Foundation. (n.d.). [*Node.js*](https://node.js/)*documentation* [Website]. Retrieved from <https://nodejs.org/en/docs/>

Docker Inc. (n.d.). *Docker documentation* [Website]. Retrieved from <https://docs.docker.com/>

# Project Charter Document

**Project Overview**

**Project Name:** Trading Application Using RabbitMQ and Node.js

**Project Manager:** Mohammaduzair & Mohammed Isa

**Start Date: 3 October**

**End Date: 26 October**

### Purpose

The project's goal is to use Node.js and RabbitMQ to create a simulated stock exchange application. This application will demonstrate message brokering and order matching while enabling users to manage order books, place buy and sell orders and execute trades in real-time.

### Objectives

* Build a simulation system for stock trading that allows buy and sell orders.
* Integrate RabbitMQ for order handling and messaging.
* Enable users to interact with the application via terminal commands.

### Scope

Developing the trading application, configuring order books, setting up RabbitMQ for order and trade queues, and thoroughly testing the system via the terminal interface are all included in the scope.

### Deliverables

* Trading application code for executing trades and processing orders.
* Detailed report detailing the usefulness, error handling, and summary of the application.
* Documentation pertaining to user instructions, functionality, and setup.

### Stakeholders

Mohammaduzair: Project Manager and Reviewer

Mohammed Isa: Developer and Designer

### Risks

* Dependency Issues: Possible problems setting up RabbitMQ.
  + Mitigation: Verify that environment and Docker variables are set up appropriately.
* Connectivity Issues: RabbitMQ may fail to connect if the server is not running. Mitigation: Document steps to troubleshoot RabbitMQ setup.

### Team Members

|  |  |
| --- | --- |
| Name | Role |
| Mohammaduzair Mohammadyahya Shaikh | Project Manager and Reviewer |
| Mohammed Isa | Developer and Designer |

Ui and Screenshots of functionality

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated