**Software Process Selection, Project Plan and Risk Management**

**P02:MinarMarket**

**<team member names & ids>**

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| **25100016** | **Abdul ahad bin ali** |
| **25100017** | **M. Umer Jamil** |
| **25100211** | **Hasan Malik** |
| **25100181** | **Saad Ilyas** |
| **25100257** | **Aniqa Aqeel** |

|  |  |  |
| --- | --- | --- |
| **Content** | **Totals** | **Obtained** |
| Software Process Selection | 20 | 19 |
| Project context analysis | 10 | 0 |
| Gantt chart | 25 | 15 |
| Development environment preparation | 10 | 10 |
| Deployment platform | 10 | 10 |
| Risk Management | 15 | 15 |
| Who did what | 3 | 3 |
| Review checklist | 2 | 2 |
| Overall formatting/template | 5 | 5 |
| GitHub folder structure penalty | -15 | - |
| Late submission penalty | -20 | - |
| **Grand Total** | **100** | **79** |
| **General Comments/Individual Grading:** | | |

**Table of Contents**

[1.](#_gjdgxs) Introduction 3

[2.](#_30j0zll) Software Process Selection 4

[3.](#_1fob9te) Gantt Chart 5

[4.](#_3znysh7) Development Environment Preparation 6

[5.](#_2et92p0) Deployment Platform 6

[6.](#_tyjcwt) Risk Management 7

[6.1 Potential Risks and Mitigation Strategies 7](#_3dy6vkm)

[7.](#_1t3h5sf) Who Did What? 8

[8.](#_4d34og8) Review checklist 8

# Introduction

<Give an overview of the project here. The overview must highlight the overall objectives of the project and its potential users and customers.>

In the rapidly evolving world of e-commerce, the relationship between buyers and sellers is continuously being redefined. Traditional online marketplaces tend to operate in a seller-centric manner, where sellers list their products, and buyers browse through these listings to make their purchases. While this model has proven effective in many scenarios, it often leaves buyers with limited options when they have specific needs that don’t align perfectly with the available listings. As consumers increasingly demand personalization and convenience, there is a growing need for a marketplace that addresses this limitation and fosters a more collaborative relationship between buyers and sellers.

Our project introduces an innovative approach to online marketplaces by creating a platform that not only allows sellers to list their products but also gives buyers the power to post their specific requirements. This dual functionality transforms the marketplace into a more dynamic and interactive ecosystem where buyers actively express their needs, and sellers can respond by offering products that meet those exact requirements. This model reduces the gap between supply and demand, enabling sellers to more effectively target interested buyers and ensuring buyers find the products that truly fit their preferences.

The core objective of this project is to design and implement a marketplace that enhances the traditional e-commerce experience. Buyers will no longer be confined to searching through predefined listings but can instead list the products or services they are looking for. Sellers, in turn, will have visibility into these buyer requests and can engage directly with potential customers by offering relevant products or negotiating terms that meet the buyer’s expectations. This two-way interaction fosters a marketplace that is more responsive, transparent, and efficient.

A key focus of our project is to simplify and streamline communication between buyers and sellers. The platform will feature an intuitive user interface that allows both parties to post, search, and communicate with ease. Buyers will be able to track the offers they receive in response to their requests, compare different sellers, and make informed purchasing decisions based on personalized recommendations. Sellers will benefit from real-time notifications of buyer requests that match their inventory, allowing them to act quickly to meet demand. This buyer-driven interaction introduces a new level of personalization and convenience, benefiting both parties involved in the transaction.

Our target audience for this platform includes individual consumers, small businesses, and larger enterprises. Individual consumers will appreciate the ability to request highly specific products, while businesses can leverage the platform to source bulk orders or specialized items. Additionally, the platform can serve niche markets where product availability may be limited, empowering buyers with greater choice and sellers with direct access to a motivated customer base.

Furthermore, our marketplace is designed with scalability and flexibility in mind. As the platform grows, we plan to incorporate advanced features such as AI-driven product matching, where algorithms analyze buyer requests and suggest potential matches from a seller’s inventory. This feature will streamline the offer-making process for sellers and make it easier for buyers to receive relevant product suggestions. The system will also allow for future integrations with payment gateways, shipment tracking, and customer review systems to create a comprehensive and seamless e-commerce experience.

Ultimately, this project aims to revolutionize the traditional marketplace model by making it more buyer-driven, interactive, and efficient. By bridging the gap between buyer needs and seller offerings, our platform will create a more engaging and fulfilling experience for all users. This approach will not only improve transaction success rates but also foster stronger relationships between buyers and sellers, setting a new standard for online commerce in the modern digital era.

# Software Process Selection

< (1) Discuss the pros and cons of waterfall and agile (scrum) processes in your own words.

(2) Select one of the above processes for your project development.

(3) Justify your selection with clear reasoning. Refer to “Project Context Analysis” in the slides to get help for writing this section.>

#### **2.1 Pros and Cons of Waterfall and Agile (Scrum) Processes**

**2.1.1 Waterfall Process:**

* Pros:
* **Structured Approach**: The waterfall model is linear and sequential, making it easy to manage and understand. Each phase has specific deliverables and a review process, ensuring clarity in project progress.
* **Defined Requirements**: Since requirements are gathered at the beginning, there’s a clear understanding of the project’s scope from the start, which can help in planning and resource allocation.
* Cons:
* **Lack of Flexibility**: Once a phase is completed, it is difficult to go back and make changes. If the client or development team identifies issues or new requirements later, the rigid structure does not allow for easy modification.
* **Limited Client Visibility**: Clients don’t see the final product until the end of the development cycle, which increases the risk of the product not meeting their expectations.
* **Delayed Feedback**: Since there is no incremental development or regular testing with clients, any issues or misalignments are only identified towards the end, making them harder and more expensive to fix.

**2.1.2 Agile (Scrum) Process:**

* Pros:
* **Iterative Development**: Agile operates on short, iterative cycles (sprints), allowing teams to focus on manageable tasks and continuously improve based on client feedback.
* **Frequent Feedback**: Clients are involved at the end of each sprint, providing feedback that can be incorporated into the next iteration. This ensures the product is more aligned with their needs and expectations.
* **Flexibility**: Agile allows teams to adapt to changing requirements and make modifications during development, which is beneficial when project needs are uncertain or evolving.
* **Transparency and Collaboration**: With daily stand-ups and weekly checkpoints (scrum), there is clear visibility of the project’s progress, enhancing collaboration within the team and with stakeholders.
* Cons:
* **Demanding for Team and Client**: Agile requires active participation from both the team and the client. Regular meetings and feedback sessions can be time-consuming and may require resources that some teams or clients might not be able to commit.
* **Potential for Scope Creep**: Since Agile is flexible and open to changes, there is a risk of scope creep if the project team does not manage changes carefully. This could lead to extended timelines or additional costs if not controlled properly.

#### **2.2 Selection of Agile (Scrum) Process**

For the development of our MinarMarket e-commerce marketplace project, we have chosen the Agile (Scrum) process with a one-week sprint cycle.

#### **2.3 Justification for Agile (Scrum) Process Selection**

Given the dynamic nature of our project and the need to build a platform that actively engages buyers and sellers, the Agile (Scrum) methodology is the most suitable approach. Our marketplace aims to foster an interactive and responsive environment, which requires continuous iterations and client feedback to refine and improve the user experience.

By adopting Agile, we can work in short, focused sprints, ensuring that we receive feedback frequently from users and stakeholders. This is critical because our platform's success depends on meeting user needs and expectations, and their feedback will help us validate our features and functionality early on. Moreover, Agile’s flexibility will allow us to adjust our plans if new requirements or challenges arise, ensuring that the final product is adaptable and optimized for the evolving needs of our target audience.

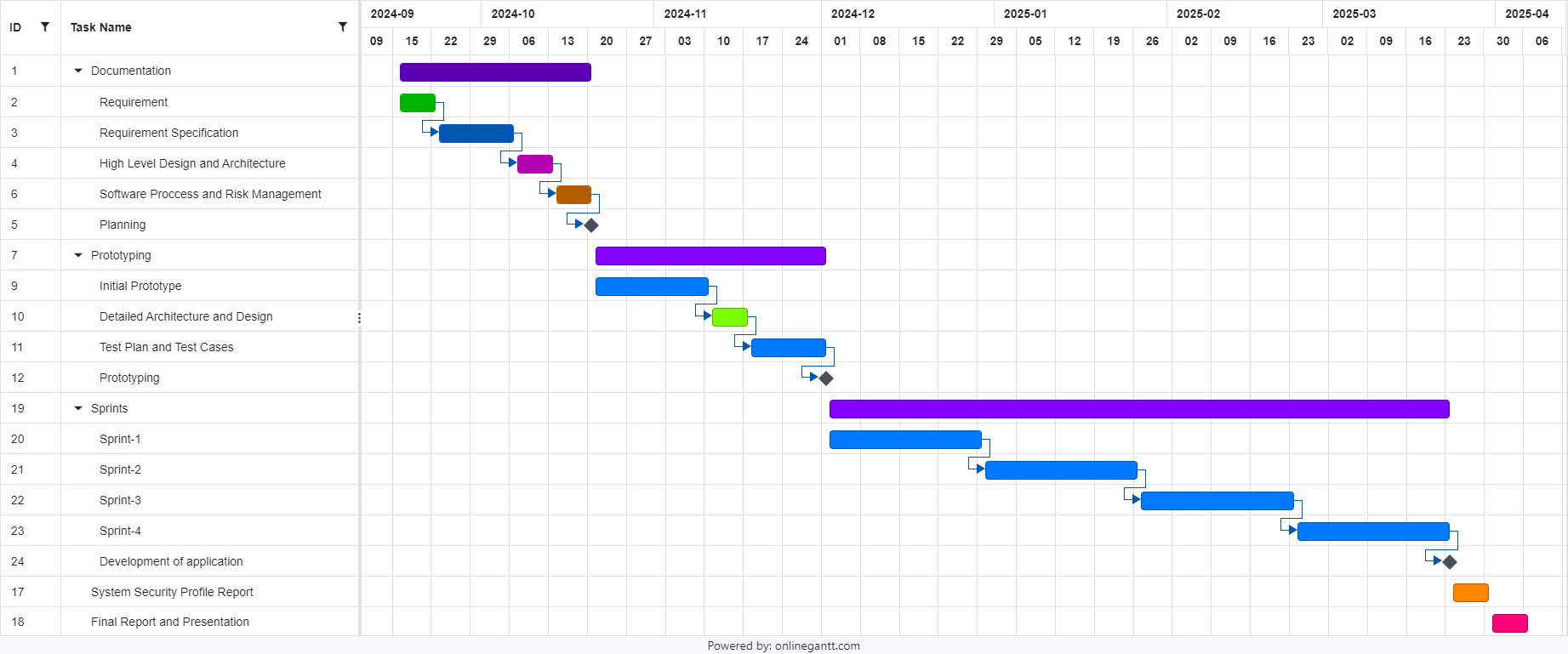
Additionally, the weekly sprint structure in Scrum aligns perfectly with our Agile approach, allowing us to break down tasks into manageable components that can be completed and reviewed within a short timeframe. This provides transparency to all stakeholders and helps keep the team focused on delivering specific functionalities efficiently.

In contrast, the waterfall approach would not be suitable for our project since it lacks the flexibility required for an evolving and user-driven platform. The risk of reaching the final stages of development only to find that the product does not align with user expectations is too high, making Agile the preferred choice for a dynamic, interactive marketplace.

[Missing discussion as required by “project context analysis” in the lecture slides.]

# Gantt Chart

# [Missing individual assignments. Missing plan for final integration testing. ]



# Development Environment Preparation

**1.Tools and Technologies for Prototype Development**

* Programming Languages:

JavaScript: Frontend interactivity

Node.js: Backend development

* Frontend Framework:

React.js: Build dynamic user interfaces

* Backend Framework:

Express.js: Manage API routes and server-side logic

* Database:

MongoDB: NoSQL database to store product listings and buyer requests

* Version Control:

Git: For code collaboration and version management

* Development Tools:

Visual Studio Code (VS Code): Code editor

Postman: API testing

Docker: Containerization of the application

* Task Management and Collaboration:

Google Meat: Communication within the team

**Environment Setup Details**

We have successfully set up the development environment on our machines. Below are the steps we followed:

Step 1: Installed Node.js and verified it by running node -v and npm -v.

Step 2: Initialized a new Git repository and connected it with GitHub for collaboration.

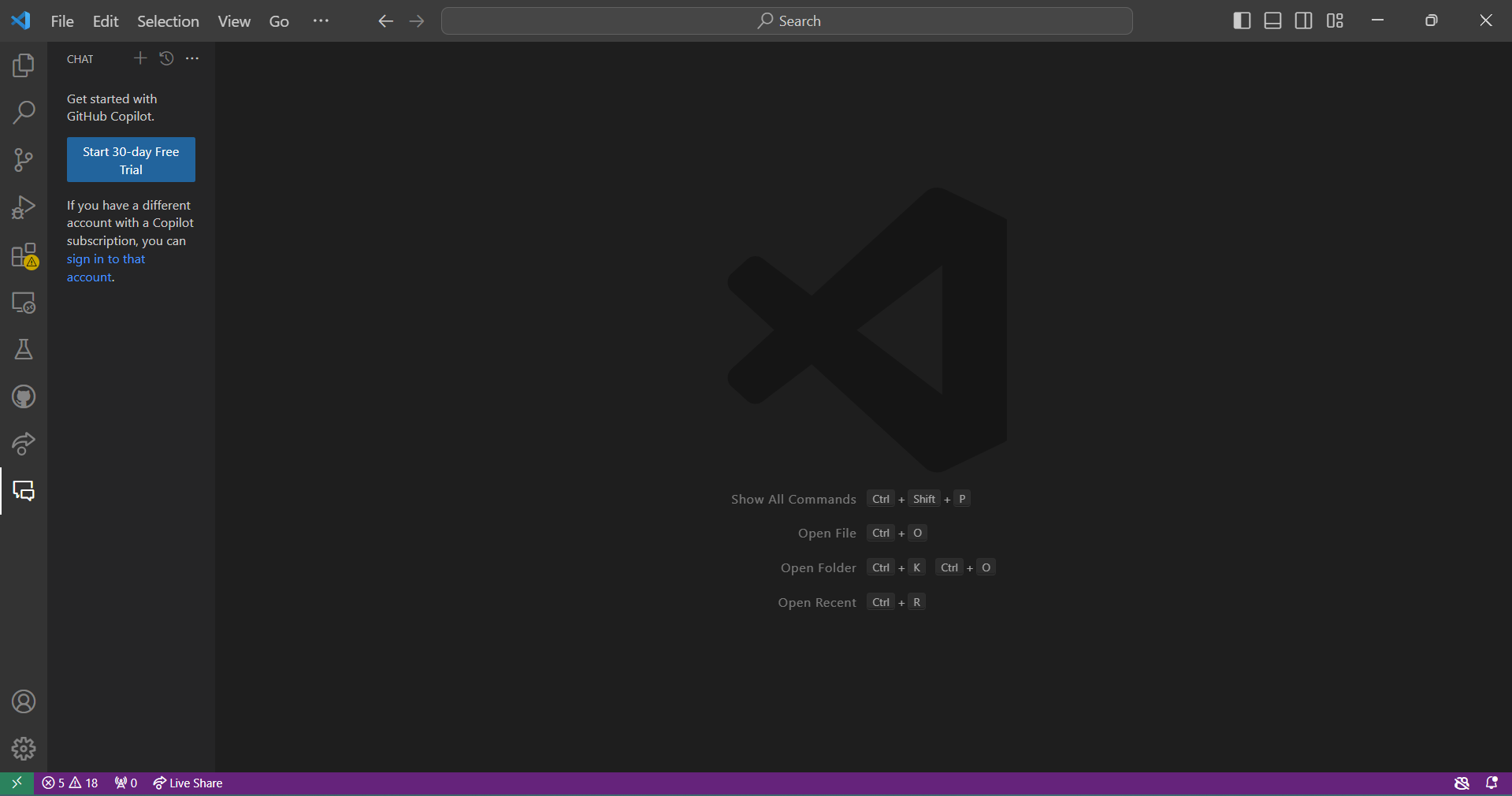
Step 3: Installed Visual Studio Code with relevant extensions (Prettier, ESLint, etc.).

Step 4: Installed MongoDB locally and tested the connection using MongoDB Compass.

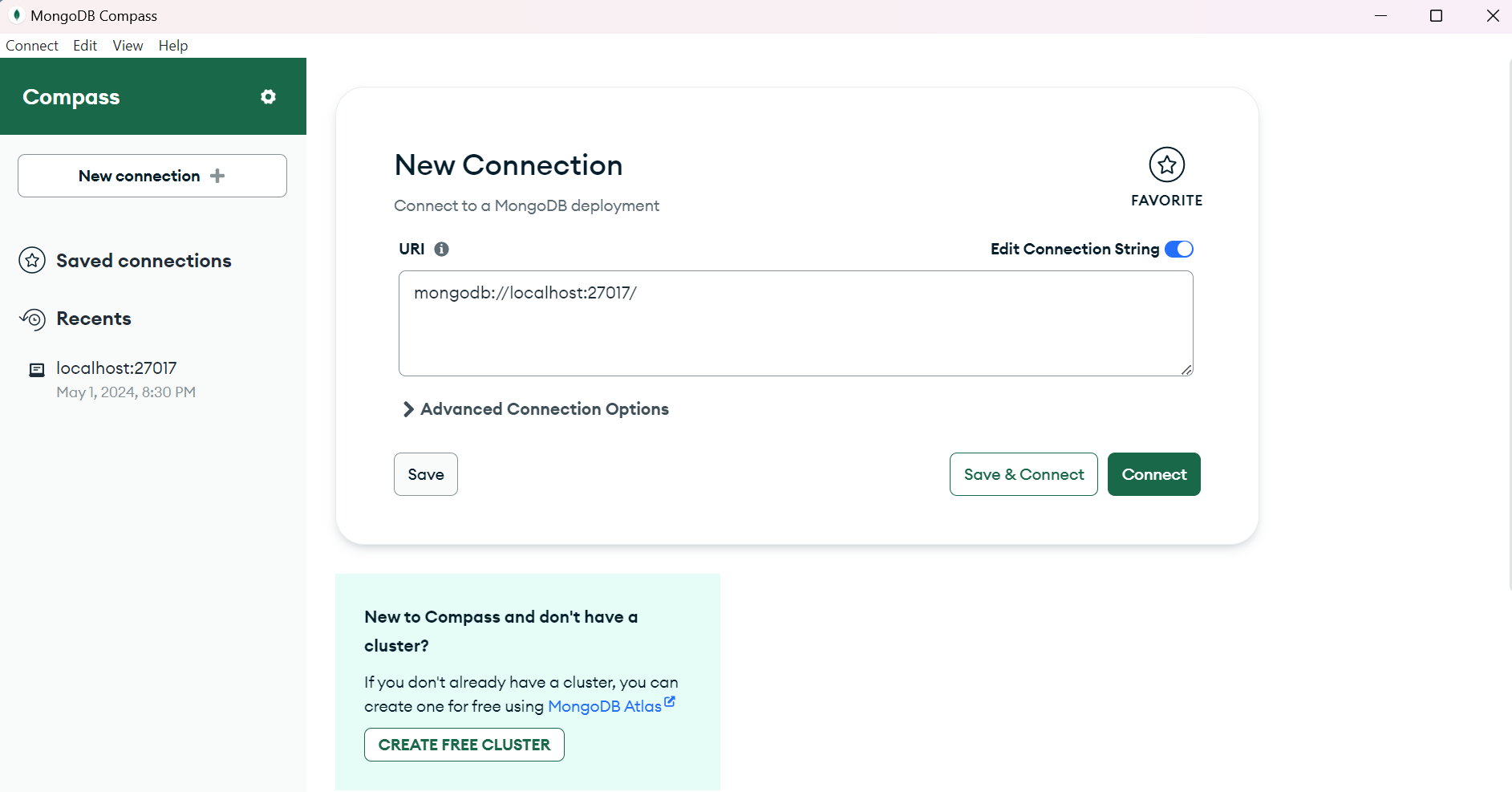
Step 5: Set up Docker and created a Dockerfile to containerize the backend service.

Step 6: Configured Postman for testing backend API endpoints.

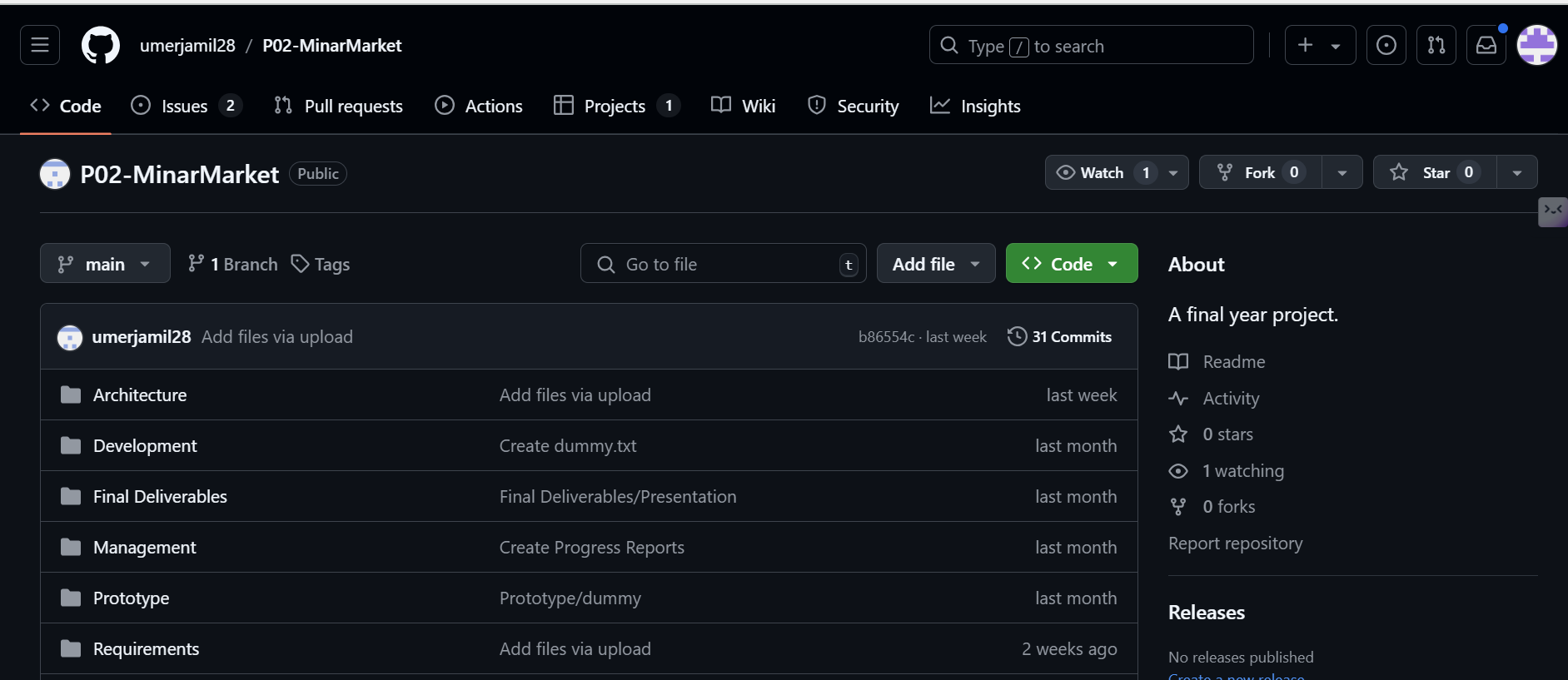
VS Code:

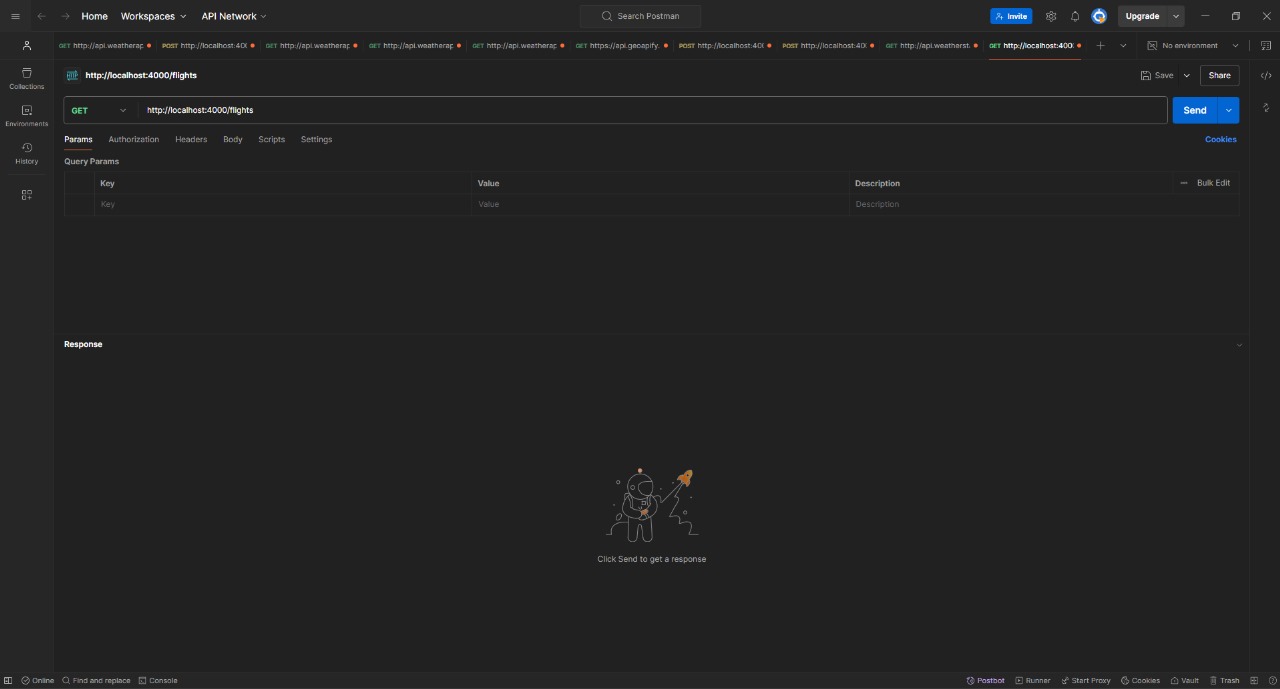


MongoDB:

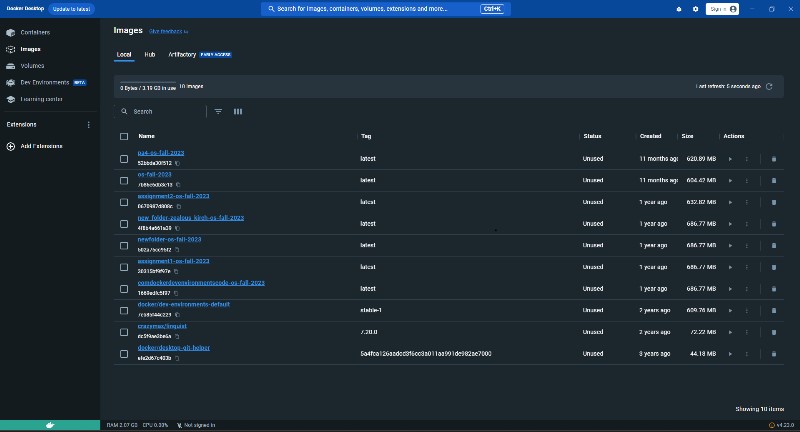


Github:



PostMan:  


Docker:



# Deployment Platform

We will be using VPS-based hosting from Hostinger. We plan to pilot-test our product in LUMS after completion, and for that, we are opting for the KVM 1 package provided by Hostinger.  
  
The specs this package offers us are:   
One vCPU core

4 GB RAM

50 GB NVMe disk space

4 TB bandwidth

Data centers worldwide

Linux operating systems  
  
These specs will be able to handle approximately 100-150 simultaneous users since our app is a dynamic web app.   
  
Moreover, if our pilot testing goes well, we can scale it up and use greater packages such as KVM 8, or KVM 9, which provides us greater resources to tackle with greater traffic on our website.  
  
The KVM-1 Package will cost around Rs.1500/month

# Risk Management

## Potential Risks and Mitigation Strategies

<List down top 10 potential risks and their mitigation strategies>

|  |  |  |
| --- | --- | --- |
| **Sr.** | **Risk Description** | **Mitigation Strategy** |
|  | **Deployment Issues**: Errors during deployment or incorrect environment setup. | Perform a spike and research (RnD) on deployment tools beforehand. Use Docker to ensure consistency in environments. |
|  | **Scope Creep**: Uncontrolled changes that increase project scope. | Regularly review project goals and adhere to Agile sprint planning to limit changes. Prioritize tasks and features. |
|  | **Inadequate Testing**: Bugs or issues discovered late in the process. | Implement continuous integration with automated unit and integration testing. Perform testing at each sprint review. |
| 4. | **Security Vulnerabilities**: Potential for data breaches or malicious attacks. | Conduct security audits and implement encryption and best security practices (e.g., HTTPS, secure database storage). |
| 5. | **Poor User Adoption**: Users finding the platform difficult to use or not meeting expectations. | Conduct user research and usability testing to gather feedback and adjust the user interface based on user behavior. |
| 6. | **Performance Issues**: The platform becomes slow or unresponsive with increasing users. | Optimize backend queries, implement load balancing, and conduct performance testing under simulated high traffic. |
| 7. | **Data Loss**: Accidental loss of data due to technical or human error. | Regularly back-up databases and set up recovery strategies using automated cloud backups and version-controlled environments. |
| 8. | **Insufficient Scalability**: The system architecture may not handle higher-than-expected loads or scaling to a larger user base. | Design the system with scalability in mind, using cloud-based services and load testing tools. Incorporate microservices and containerization (e.g., Docker). |
| 9. | **Hardware/Software Compatibility Issues**: Compatibility problems between different development environments, software, or hardware. | Use containerization (e.g., Docker) and ensure standardized development environments. Conduct early testing on all target environments. |
| 10. | **Time Management**: Inability to meet deadlines due to mismanagement of time or underestimation of task complexity. | Create a detailed Gantt chart and regularly monitor progress. Break down tasks into smaller milestones with clear deadlines. |

# Who Did What?

|  |  |
| --- | --- |
| **Name of the Team Member** | **Tasks done** |
| Aniqa Aqeel | Risk Management |
| Muhammad Umer Jamil | gantt chart |
| Abdul Ahad Bin Ali | Deployment Platform |
| Hasan Malik | Development Environment Preparation |

# Review checklist

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

|  |  |
| --- | --- |
| **Section** **Title** | **Reviewer Name(s)** |
| gantt chart | Aniqa Aqeel |
| Development Environment Preparation | Abdul Ahad Bin Ali |
| Deployment Platform | Hasan Malik |
|  |  |