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

**P04: Manzil**

**<team member names & ids>**

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| **25100235** | **Shahrez Faisal** |
| **25100313** | **Muhammad Mehdi** |
| **25100015** | **Omar ibne sajjad** |
| **25100320** | **Muhammad Usman Arshid** |
| **24100199** | **Umer Inayat Khan** |

|  |  |  |
| --- | --- | --- |
| **Content** | **Totals** | **Obtained** |
| Software Process Selection | 20 | 20 |
| Project context analysis | 10 | 10 |
| Gantt chart | 25 | 25 |
| Development environment preparation | 10 | 10 |
| Deployment platform | 10 | 10 |
| Risk Management | 15 | 14 |
| 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** | **99** |
| **General Comments/Individual Grading:** | | |

**Table of Contents**

[1. Introduction 3](#_Toc179818092)

[2. Software Process Selection 4](#_Toc179818093)

[3. Gantt Chart 5](#_Toc179818094)

[4. Development Environment Preparation 6](#_Toc179818095)

[5. Deployment Platform 6](#_Toc179818096)

[6. Risk Management 7](#_Toc179818097)

[6.1 Potential Risks and Mitigation Strategies 7](#_Toc179818098)

[7. Who Did What? 8](#_Toc179818099)

[8. Review checklist 8](#_Toc179818100)

# Introduction

**Manzil** is a mobile application designed to assist users traveling to northern destinations and major cities in Pakistan. It aims to streamline the process by offering a one-stop solution for booking hotels, renting vehicles, checking weather updates, and finding local services and their reviews for better choice of places, such as restaurants, schools, and hospitals.

The app leverages machine learning (ML) to provide personalized recommendations and supports real-time services like live weather updates and itinerary management. The primary users include travelers, hotel management, and app administrators.

# Software Process Selection

Software development process is the set of structured steps involved in the development of any software. It is mainly a series of activities involving the following processes:

1. Specification – defining what the system should do.
2. Design - defining the organization of the system
3. Implementation – implementing/coding the system.
4. Validation – checking that it does what the customer wants.
5. Evolution – changing the system in response to changing customer needs.

In the process of selecting a software process, several things are required to be kept in consideration such as the adaptability of system to changes in software requirement during the process, the feasibility of developers to work together and being in touch with other stakeholders, software release requirements etc.

Based on the above criteria, a software development process is selected.

Mainly 2 software development processes are used widely:

1. Waterfall Model
2. Agile (Scrum) Model

Each model has its own pros and cons, and based on the software requirements and planning strategies, a development model is selected.

**Waterfall Model:**

This process model of development involves a step-by-step strategy, where you move to the next step once a step is completed.

The steps involve the following:

1. Requirements analysis and definition
2. Software design
3. Implementation and unit testing
4. Integration and system testing
5. Operation and maintenance

This software model is applicable and best suited to the environment where there is a defined and stable set of system requirements, and the requirements are unlikely to be changed in future. However, it also has its own disadvantages, which makes it unsuitable for certain development environments.

**Advantages:**

1. **Structured Approach:** The Waterfall model follows a clear, sequential process, which makes it easy to understand and manage.
2. **Well-Defined Stages:** Each phase in the model has specific deliverables and a review process, which helps to maintain focus and accountability.
3. **Easy to Measure Progress:** Progress can be easily tracked as each phase is completed, making it simple to manage timelines.
4. **Ideal for Smaller Projects:** The model is effective for projects with well-defined requirements that are unlikely to change.

**Disadvantages:**

1. **Inflexibility:** The model works in phases which makes it quite inflexible to adapt to changes in the system requirement at later stages.
2. **Late Testing:** As the process works in phases, the testing phase happens at the end, which can cause extensive costs if bugs are detected.
3. **Limited Customer Involvement:** Stakeholders may have little input during the development process, which can result in a final product that does not meet expectations.
4. **Assumes Predictable Requirements:** Waterfall models is not adaptable for dynamic environments, where changes might evolve during the development phase.

**Agile (Scrum) Model:**

The Agile (Scrum) model is an iterative and incremental approach to software development that focuses on flexibility, collaboration, and customer feedback.it involves iterations called sprints to deliver functional increments of the product, which allows for continuous improvement and adaptation to changes in requirements.

**Advantages**:

1. **Flexibility and Adaptability:** This model allows for changes to be made at any stage of the project, which helps accommodate evolving requirements.
2. **Customer Collaboration:** All the stakeholders are kept in a loop during the development phase, and changes are made based on their input, which results in fulfilled customer expectations.
3. **Incremental Delivery:** Features are delivered in small and manageable increments, allowing for earlier detection of issues.
4. **Team Empowerment:** This model involves working in a group which promotes self-organizing teams, which can enhance motivation and creativity.
5. **Enhanced Communication:** Frequent meetings improve team communication and collaboration.

**Disadvantages:**

1. **Less Predictability:** Due to the flexibility, timelines and budgets can be harder to predict (New requirements can take more time and budget to be implemented), making planning difficult.
2. **Requires Cultural Shift:** Teams may struggle to adapt to Agile practices, especially in traditional environments.
3. **Scope Creep:** The openness to change can lead to continuous changes in scope, which may result in project overload.
4. **Documentation Challenges:** Agile often prioritizes working software over comprehensive documentation, which can create knowledge gaps.
5. **Dependency on Team Dynamics:** The success of Agile relies heavily on team members working well together and understanding Agile principles.

**Agile (scrum) model for our project:**

Having the pros and cons of the above process models in front, and analyzing the requirements and environment of our project, Agile (Scrum) model suits well in this scope. As the model provides the gap for changes in the requirements list, it will help us to better develop and incorporate changes at later stages in development phase.

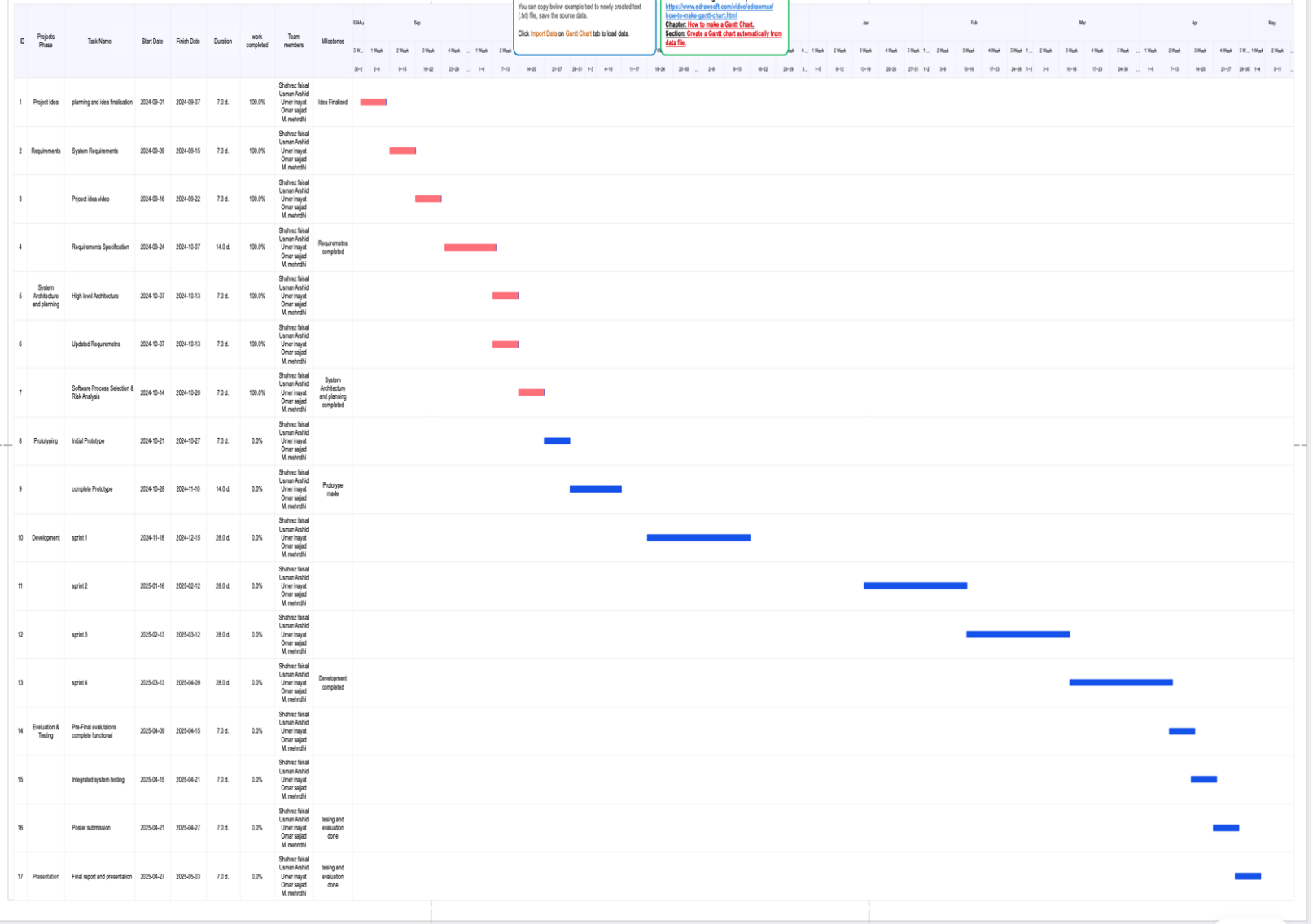
**Reasoning:**

While keeping the project context in mind, and the development environment in front, the following points resonate more to the Agile (Scrum) development model.

1. As our project has a limited budget and time constraints, there could be a high potential loss in case of bugs or issues found in the system. Thus, Agile model suits best due to early testing feature.
2. The development team is medium skilled developers but are capable of learning and incorporating changes effectively.
3. The team size is less (5), which makes it effective for meetings (Agile model).
4. The organization culture is adaptive to change which makes it suitable for Agile model.
5. As the course project requires prototype releases, and the project needs to be tested well before the final deadlines, agile model gives the space for early software release.
6. Also, the biweekly meetings with the project supervisors and keeping them in loop, helps the development team not lose track and continue smoothly.

The above points about the team environment and project requirements resonate with the agile development model, which makes it best suited for the project scope.

# Gantt Chart



# Development Environment Preparation

< (1) List down tools and technologies that you will use for prototype development.

(2) Setup the development environment on your machines and mention here that you have actually setup the environment. Include three snapshots of the tool(s) that you are going to use for development. These snapshots must be taken from the tool(s) while they are actually running on your system.>

**Node.js**: JavaScript runtime for backend development, running the server-side logic.

**React Native**: Framework for developing cross-platform mobile applications using JavaScript and React.

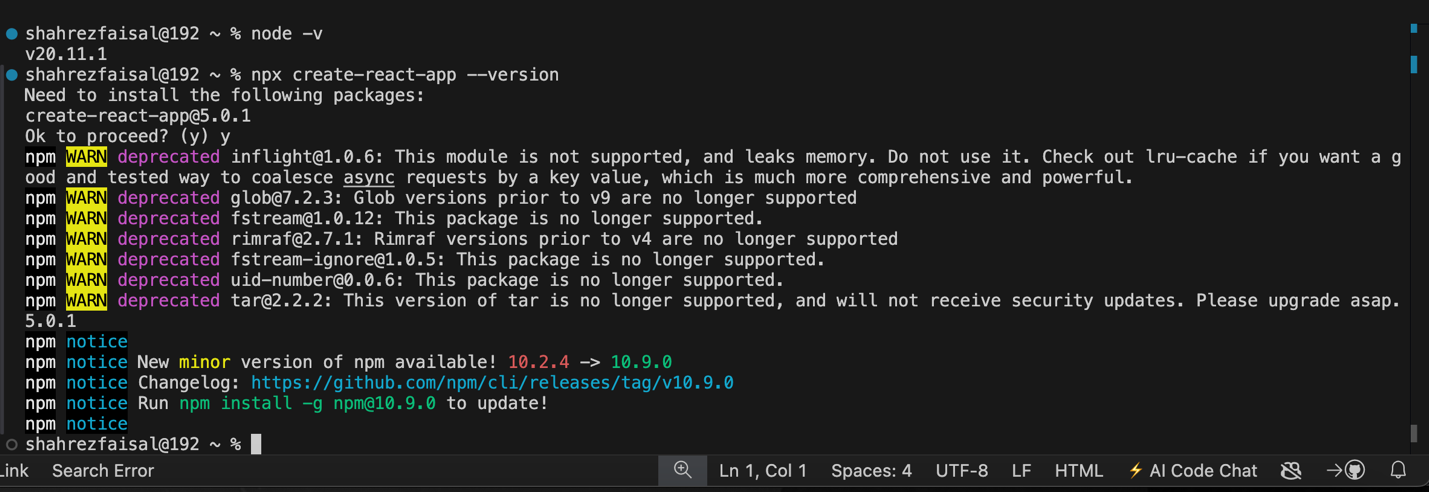
**AWS** (Amazon Web Services): Cloud service provider for hosting, databases, and backend services (e.g., EC2, S3, Lambda).

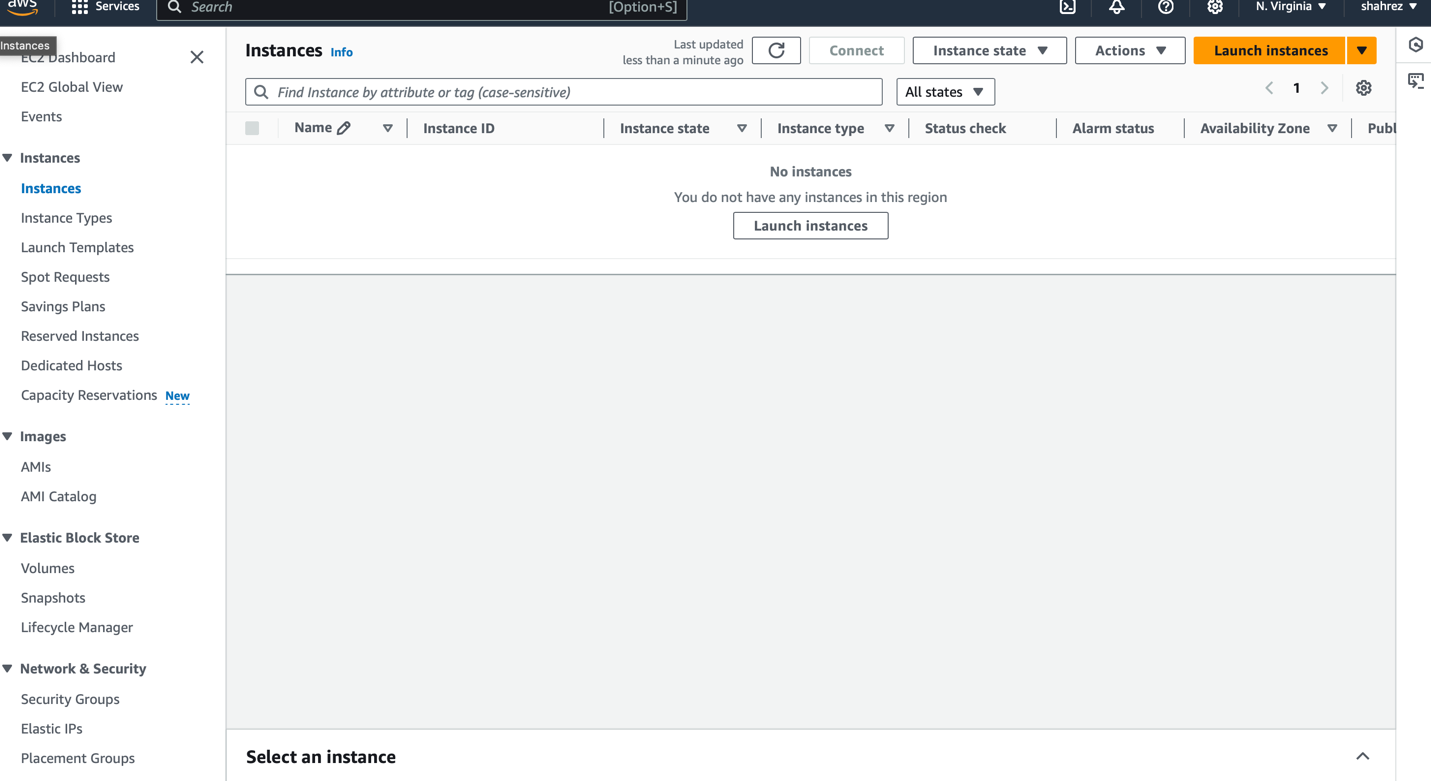
**MongoDB**: NoSQL database for storing application data.

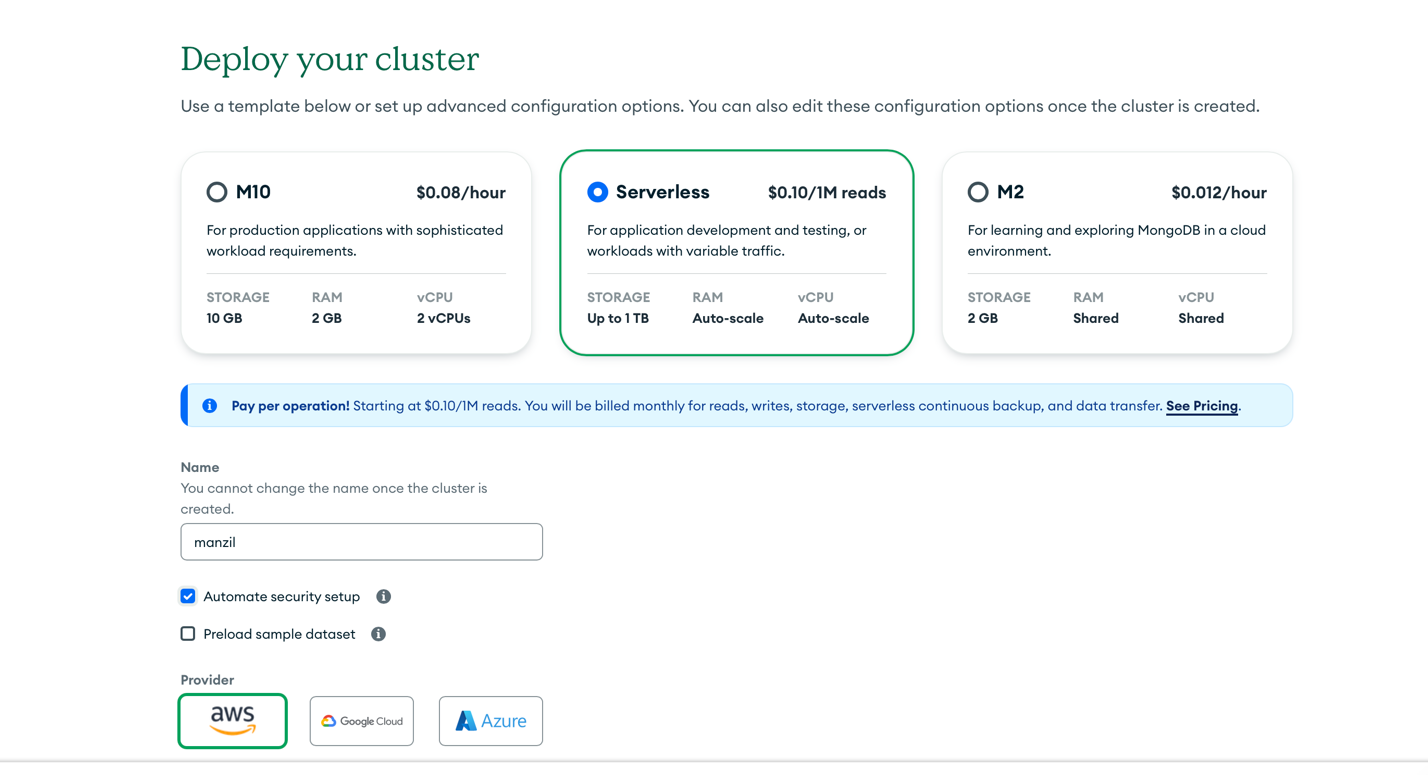
**VS Code**: Code editor with built-in tools for debugging, extensions, and version control.

**Postman**: For API testing and validation.

**Git/GitHub**: For version control.

****





A screenshot of a computer

Description automatically generated

# Deployment Platform

For the deployment of the Manzil application, we have chosen Amazon EC2 (Elastic Compute Cloud 2) as our deployment platform. Amazon’s EC2 provides resizable virtual servers (compute capacity) in the cloud, that can host backend services for any type of application, including mobile applications.

## Key Features:

* Scalability and Flexibility: EC2 offers the ability to adjust the computer resources based on the needs of our application. This allows Manzil to handle varying traffic loads effectively through features like auto-scaling and load balancing because we don’t know how much load there will be at a given time. We can ensure high availability by hosting the application across multiple data centers (regions).
* 12-Month Free Tier: We can utilize Amazon’s 12-month free tier, which includes 750 hours/month of free usage for a t2.micro or t3.micro instance (depending on the region) for both Linux and Windows platforms.
* Integration with Databases: EC2 can easily integrate with external databases like MongoDB Atlas, which we are using for our app to store user data, book history, and reviews. This ensures efficient data handling and good communication.
* Security: EC2 provides secure API endpoints, SSL/TLS encryption, and the ability to configure role-based access control (RBAC), ensuring data privacy and protection.
* Backend: we’re using AI-based recommendations, the backend support will handle the processing of these models which require a lot of computational power.

## Advantages:

1. Cost-Effectiveness: The free tier offering for the first 12 months enables cost savings during the development and early deployment stages where we will have no initial investment in our hands.
2. Flexibility: EC2 supports a wide range of instance types and operating systems, giving us the flexibility to optimize performance for specific use cases across multiple.

## Disadvantages:

1. Complexity: Managing EC2 instances, setting up load balancers, and configuring auto-scaling can be complex and requires a steep learning curve. Which would also take up our time
2. Cost Overruns: Improper scaling or underestimation of instance requirements could lead to unexpected costs if the project grows beyond the free tier limits.

# Risk Management

## Potential Risks and Mitigation Strategies

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

|  |  |  |
| --- | --- | --- |
| **Sr.** | **Risk Description** | **Mitigation Strategy** |
|  | **System Downtime**: The app may experience downtime due to server overload, particularly during peak tourist seasons. | Implement load balancing and auto-scaling on AWS EC2 to dynamically allocate resources based on demand, ensuring high availability. |
|  | **Data Breach or Security Flaws**: Sensitive user information (e.g., payment details) may be compromised. | Use encryption for data storage and transmission (e.g., SSL/TLS, AES-256), and implement multi-factor authentication (MFA) for user logins. |
|  | **Inaccurate Results from ML Models**: Lack of real data may result in poor recommendations. | Use publicly available datasets for initial training and test the model on sample use cases to refine and validate the ML algorithms. |
|  | **Outdated Data from Hotels or Users**: Users may see stale hotel listings or reviews. | Use a manual update system where hotels/administrators can update records easily and periodically notify users when data may be outdated. |
|  | **Team Coordination Issues**: Team members may face difficulty collaborating, leading to miscommunication. | Use free project management tools like Trello, GitHub Projects, or Google Sheets to track tasks, assign roles, and regularly update project progress. |
|  | **Slow Response Time**: The app may load slowly due to inefficient code or large data transfers. | Optimize data structures, reduce image sizes, and minimize API calls by requesting only necessary data. |
|  | **Hardware Limitations**: Limited processing power on personal machines may hinder development, especially for ML models. | Use cloud services (e.g., Google Colab, AWS free tier) for resource-heavy tasks like training models or running large datasets, especially for ML components. |
|  | **Insufficient User Feedback**: The app may not meet user needs due to a lack of real user input. | Conduct early user testing by sharing prototypes with peers or target users and gather feedback to refine the design and features before the final version. |
|  | **Scope Creep**: New features or changes in requirements may overwhelm the team and increase workload. | Clearly define project scope early on and prioritize features. Implement a change request process where every new feature is discussed before being approved. Focus on better functionality over wider scope. |
|  | **Inconsistent Team Contribution**: Some team members may not contribute equally due to other commitments or lack of motivation. | Set clear individual responsibilities and track progress using project management tools. Regular check-ins ensure accountability. |

# Who Did What?

|  |  |
| --- | --- |
| **Name of the Team Member** | **Tasks done** |
| Shahrez Faisal | 3(Gantt chart),4 |
| Muhammad Usman Arshid | 5 (Deployment Platform) |
| Muhammad Mehdi | 2 (Software process selection) |
| Omar Ibne Sajjad | 6 (Risk Management) |
| Umer Inayat Khan | 1 |

# 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)** |
| Deployment Platform | Muhammad Mehdi |
| Sofware process selection | Umer Inayat |
| Risk Managament | Shahrez Faisal |
|  |  |