# Software Requirement Specification (SRS)

Project Name: Unified Uber Web Platform (Uber, Uber School, Uber Eats)

University of Sciences and Arts in Lebanon (USAL)

Team Members:  
 - Mohamad Slim  
 - Mohamad Sweid  
 - Hassan Khalil

Date: 10 3 2025

Version: 1.0

By: Mohamad islim

## Table of Contents

1. Introduction

1.1 Product Scope

1.2 Product Value

1.3 Intended Audience

1.4 Intended Use

1.5 General Description

2. Functional Requirements

3. External Interface Requirements

3.1 User Interface Requirements

3.2 Hardware Interface Requirements

3.3 Software Interface Requirements

3.4 Communication Interface Requirements

4. Non-functional Requirements

4.1 Security

4.2 Capacity

4.3 Compatibility

4.4 Reliability

4.5 Scalability

4.6 Maintainability

4.7 Usability

4.8 Other Non-functional Requirements

5. Definitions and Acronyms

## 1. Introduction

### 1.1 Product Scope

This software aims to provide a unified web platform for Uber services: ride-sharing (Uber), school transportation (Uber School), and food delivery (Uber Eats). Users can book rides, schedule school pickups, and order meals—all from a single interface.

### 1.2 Product Value

The product simplifies life by integrating three vital services into one platform, providing convenience, centralized account management, and consistent UI/UX across services.

### 1.3 Intended Audience

- General users seeking transportation or food delivery  
 - Parents arranging school transport for their children  
 - Administrators managing user accounts and services  
 - Drivers and couriers using the platform for job management

### 1.4 Intended Use

- Book rides, schedule school pickups/drop-offs, and order food  
 - Manage payment methods, view history, and provide feedback  
 - Access via desktop and mobile browsers

### 1.5 General Description

The software is a web-based system supporting three integrated services:  
   
- Uber: Ride-hailing with location and time scheduling  
 - Uber School: School-specific routes, schedules, and child safety tracking  
 - Uber Eats: Restaurant browsing, ordering, and real-time delivery tracking

## 2. Functional Requirements

- User registration and authentication  
 - Role-based access (admin, parent, student, driver, customer)  
 - Booking system for rides, meals, and school transportation (with time, location, and pricing details)  
 - Real-time tracking via integrated map APIs  
 - Access to order/ride history and ability to submit ratings  
 - Notifications via SMS and email  
 - Admin dashboard for managing users, services, and data

## 3. External Interface Requirements

### 3.1 User Interface Requirements

- Responsive design for both desktop and mobile devices  
 - Intuitive navigation and layout  
 - Dashboards for orders, rides, and child tracking  
 - User-friendly forms for support and feedback

### 3.2 Hardware Interface Requirements

- Compatible with standard web-accessible devices (PCs, tablets, mobile phones)  
 - GPS-enabled devices required for tracking and navigation

### 3.3 Software Interface Requirements

- Frontend: HTML, CSS, JavaScript, React  
 - Backend: Node.js or PHP with MySQL database  
 - Integrations with third-party APIs for mapping, payment, and notifications

### 3.4 Communication Interface Requirements

- Backend communication via REST APIs  
 - Email delivery using SMTP services  
 - Real-time location updates via WebSockets

## 4. Non-Functional Requirements

### 4.1 Security

- HTTPS encryption for secure data transfer  
 - JWT-based user authentication  
 - Role-based access control to protect sensitive features

### 4.2 Capacity

- Supports 10,000+ concurrent users  
 - Designed to accommodate high user volumes and peak usage

### 4.3 Compatibility

- Works with major browsers: Chrome, Firefox, Safari, Edge  
 - Mobile compatibility with iOS and Android browsers

### 4.4 Reliability

- 99.9% uptime with cloud-based infrastructure  
 - Auto-recovery mechanisms in case of service failure

### 4.5 Scalability

- Scalable architecture based on microservices  
 - Elastic cloud resources to manage growing user demands

### 4.6 Maintainability

- Modular codebase for streamlined updates and debugging  
 - Integrated CI/CD pipeline for smooth deployments

### 4.7 Usability

- Designed with a minimal learning curve  
 - Clear labels, instructions, and tooltips for user assistance

### 4.8 Other Non-Functional Requirements

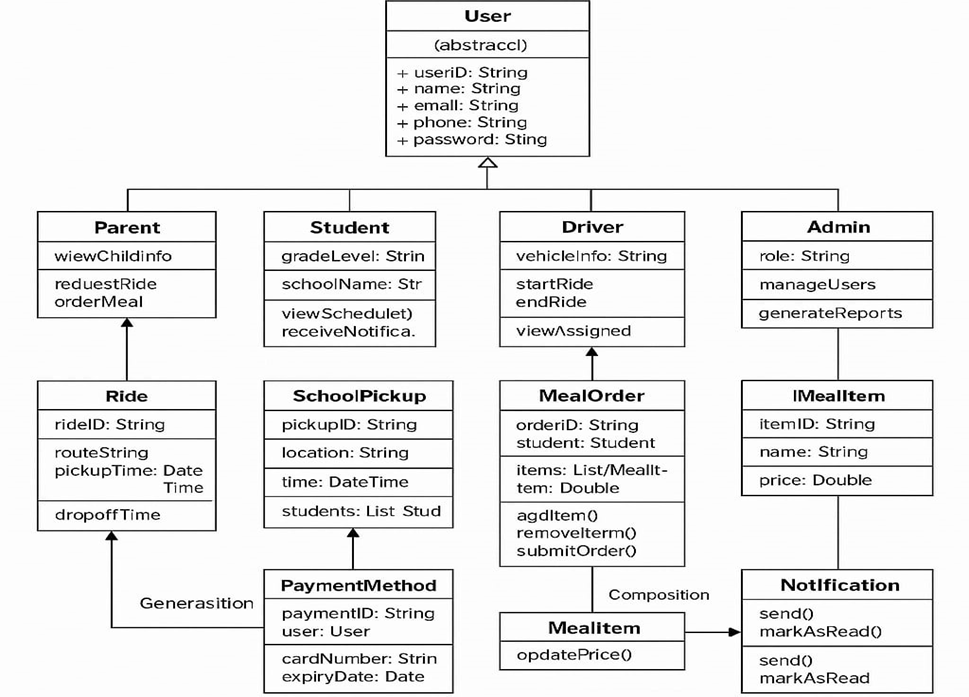
- Integrated logging and analytics for monitoring  
 - Multilingual support (e.g., English, Arabic)

## 5. Definitions and Acronyms

|  |  |
| --- | --- |
| Term | Definition |
| UI/UX | User Interface / User Experience |
| JWT | JSON Web Token |
| CI/CD | Continuous Integration / Continuous Deployment |
| API | Application Programming Interface |

# Class Diagram:

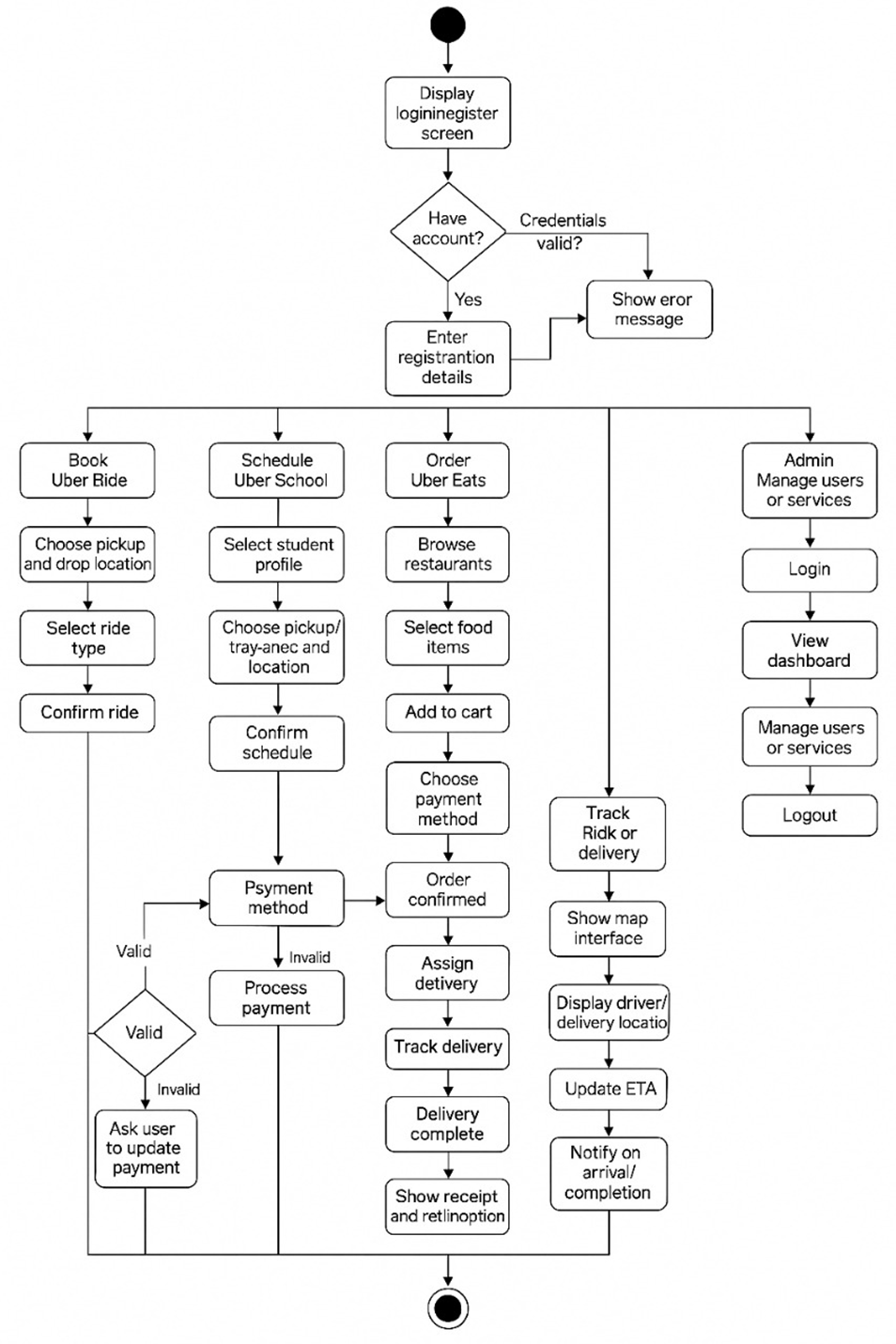
* 1. This diagram should illustrate the static structure of your application.
  2. Identify the main classes (e.g., User, Admin, Parent, Student, Driver, Customer, Ride, SchoolPickup, MealOrder, PaymentMethod, Notification, etc.).
  3. Define the attributes (data) and methods (functions) for each class.
  4. Show the relationships between classes, such as:
     1. Associations (e.g., a Customer *places* an Order).
     2. Generalizations/Inheritance (e.g., Parent, Student, Driver, Customer might*inherit from* a User class).
     3. Aggregations/Compositions (e.g., an Order *contains* MealItems).
  5. Refer to Sections 1.5 (General Description), 2 (Functional Requirements), and 3 (External Interface Requirements) in the SRS for details on entities and their interactions.



|  |  |
| --- | --- |
|  |  |
|  |  |

# Activity Diagram:

* 1. This diagram should model the dynamic flow of activities within the application.
  2. Focus on key processes outlined in the functional requirements (Section 2). Examples include:
     1. User registration and login.
     2. Booking a standard Uber ride.
     3. Scheduling an Uber School pickup.
     4. Ordering from Uber Eats.
     5. Processing payments.
     6. Tracking a ride or delivery in real-time.
     7. Admin managing users or services.
  3. Show the sequence of steps, decisions (branches), and parallel activities involved in completing these tasks.

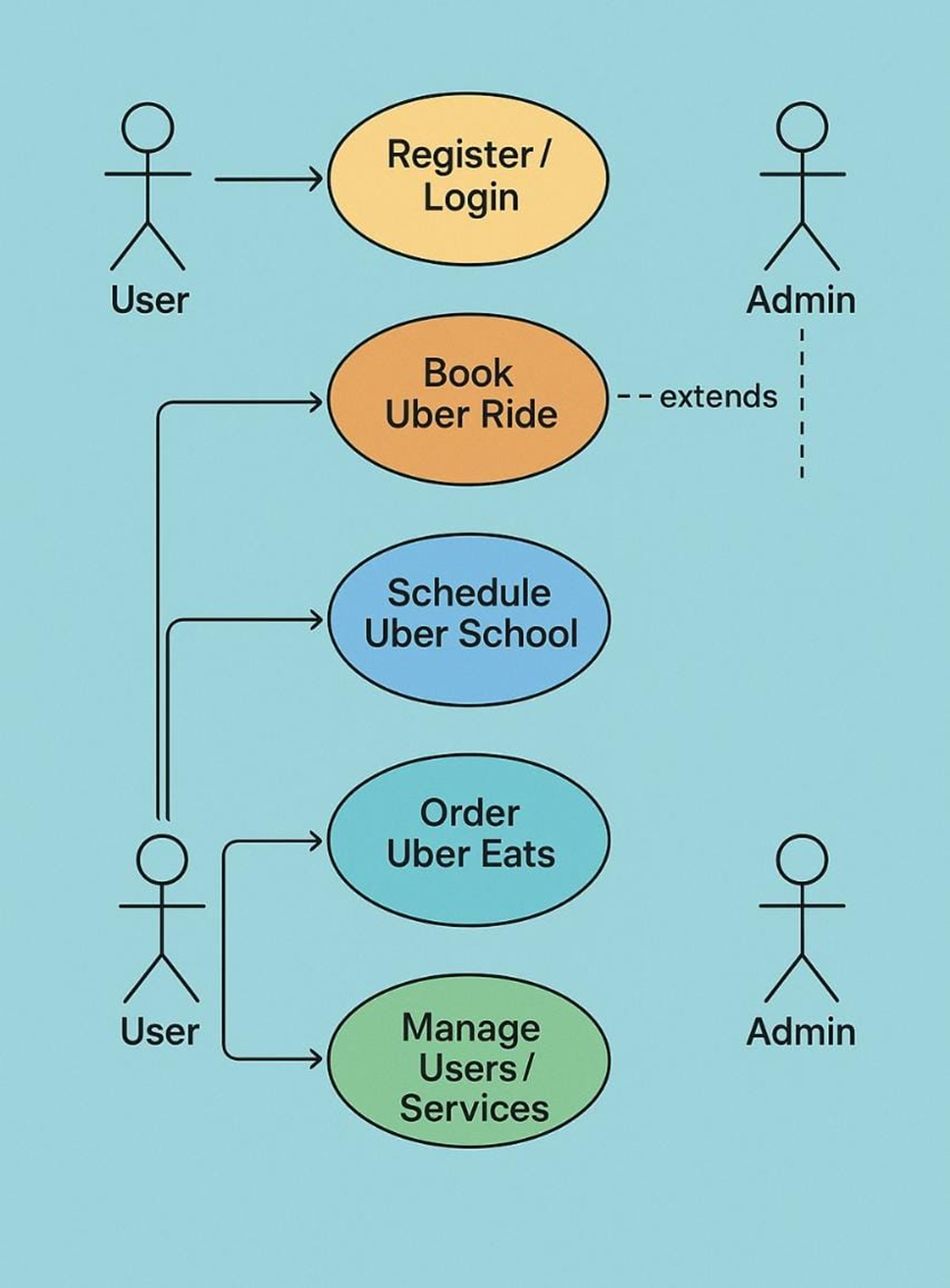


|  |  |
| --- | --- |
|  |  |
|  |  |

# Use Case Diagram:

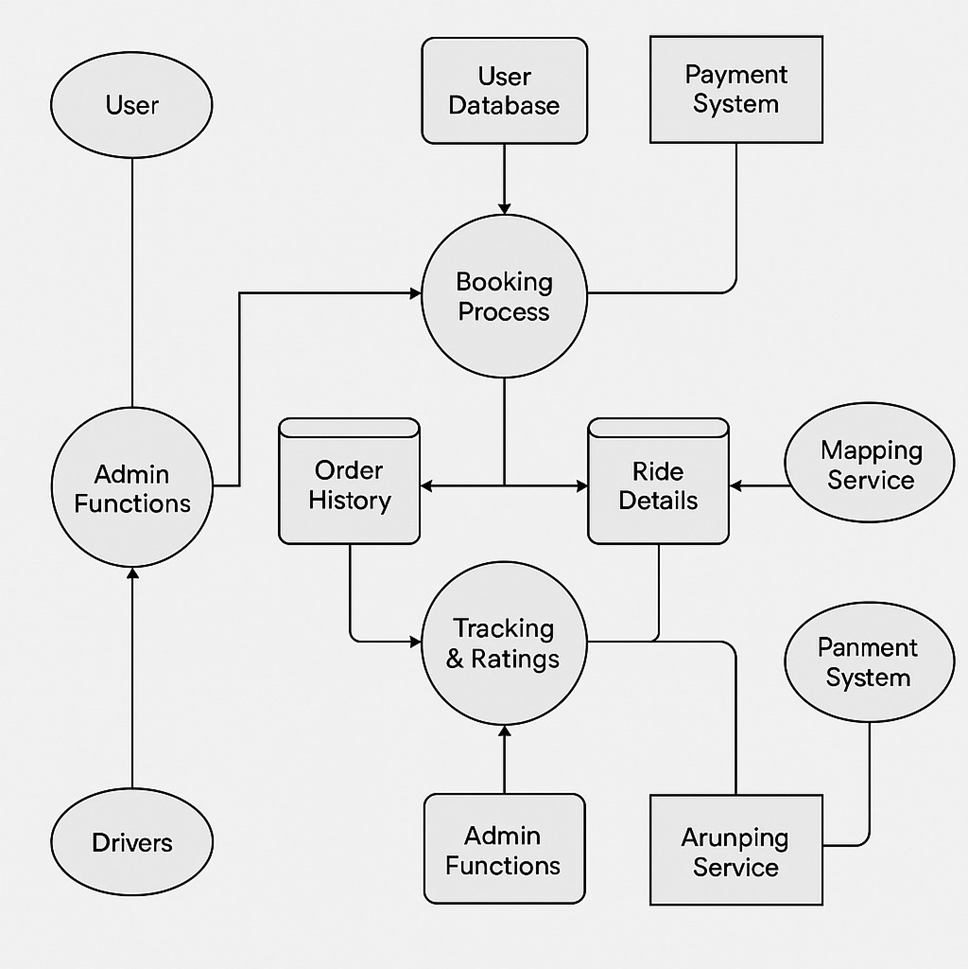
* 1. This diagram focuses on the interactions between users (actors) and the system.
  2. Identify the actors based on the intended audience (Section 1.3): General users, Parents, Administrators, Drivers/Couriers. Note that 'Student' is mentioned as a role but might interact indirectly via Parents or the school system, which needs clarification based on specific requirements not fully detailed in the summary.
  3. Define the use cases based on the functional requirements (Section 2), such as:
     1. Register Account
     2. Log In
     3. Book Ride
     4. Schedule School Transport
     5. Order Food
     6. Track Ride/Order
     7. Manage Profile/Payments
     8. View History
     9. Provide Rating/Feedback
     10. Manage Users (Admin)
     11. Manage Services (Admin)
  4. Show the relationships between actors and the use cases they perform.

|  |  |
| --- | --- |
|  |  |
|  |  |



# Data Flow Diagram (DFD):

* 1. This diagram models how data moves through your system.
  2. Identify the main processes (functions that transform data, derived from functional requirements like "Book Ride," "Process Payment").
  3. Identify data stores (where data is held, e.g., User Database, Order History, Ride Details, likely within the MySQL database mentioned in Section 3.3 ).
  4. Identify external entities (actors interacting with the system, like Users, Admins, Drivers, Payment Systems, Mapping Services).
  5. Show the data flows (arrows indicating data movement) between processes, data stores, and external entities (e.g., booking details flowing from User to Booking Process, ride data stored in Ride History).
  6. Consider data inputs and outputs for features like booking, tracking, payments, ratings, and admin functions.



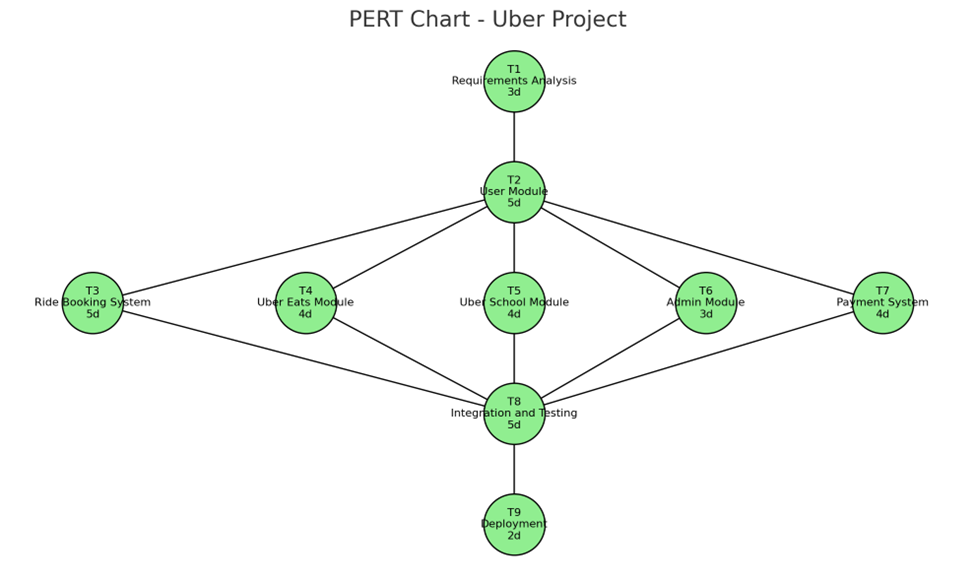
Extract Tasks & Subtasks from Functional Requirements

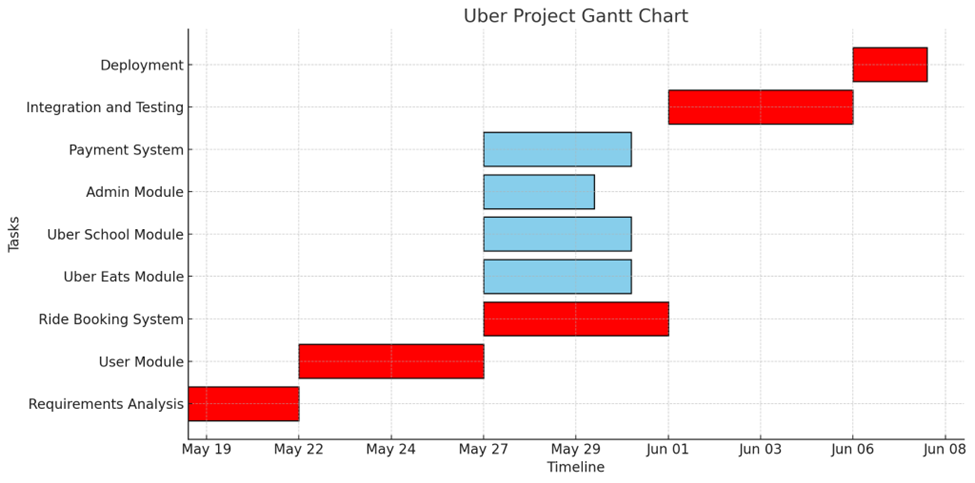
|  |  |  |
| --- | --- | --- |
| **TaskID** | **Task Name** | **Subtasks** |
| T1 | Requirements Analysis | Functional, Non-functional, Use Case Modeling |
| T2 | User Module Development | Registration, Login, Profile |
| T3 | Ride Booking System | Book Ride, Ride Matching, Tracking |
| T4 | Uber Eats Module | Restaurant List, Order Placement, Delivery Tracking |
| T5 | Uber School Module | Schedule Ride, Manage Student Profiles |
| T6 | Admin Module | User Management, Service Monitoring |
| T7 | Payment System | Add Payment Method, Process Payment |
| T8 | Integration and Testing | Functional Testing, Module Integration |
| T9 | Deployment and Documentation | Hosting, Final Reports |

Project Scheduling Table

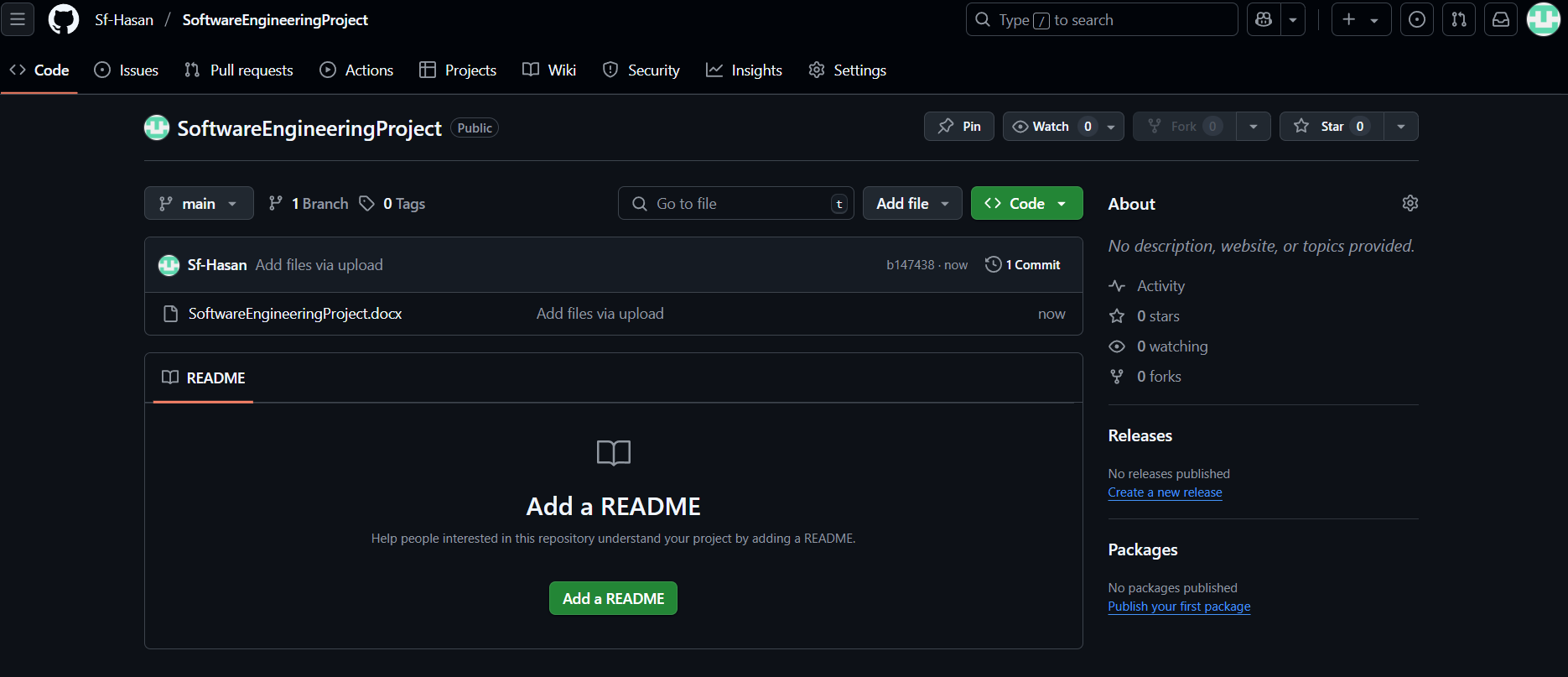
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Task ID** |  | **Task Name** | **Predecessor(s)** | **Duration (Days)** | **ES** | **EF** | **LS** | **LF** | **Critical Path** | **Slack** |
| T1 |  | Requirements Analysis | — | 3 | 0 | 3 | 0 | 3 | ✔️ | 0 |
| T2 |  | User Module | T1 | 5 | 3 | 8 | 3 | 8 | ✔️ | 0 |
| T3 |  | Ride Booking System | T2 | 5 | 8 | 13 | 8 | 13 | ✔️ | 0 |
| T4 |  | Uber Eats Module | T2 | 4 | 8 | 12 | 10 | 14 |  | 2 |
| T5 |  | Uber School Module | T2 | 4 | 8 | 12 | 10 | 14 |  | 2 |
| T6 |  | Admin Module | T2 | 3 | 8 | 11 | 13 | 16 |  | 5 |
| T7 |  | Payment System | T2 | 4 | 8 | 12 | 13 | 17 |  | 5 |
| T8 |  | Integration and Testing | T3, T4, T5, T6, T7 | 5 | 13 | 18 | 14 | 19 | ✔️ | 1 |
| T9 |  | Deployment | T8 | 2 | 18 | 20 | 19 | 21 | ✔️ | 1 |

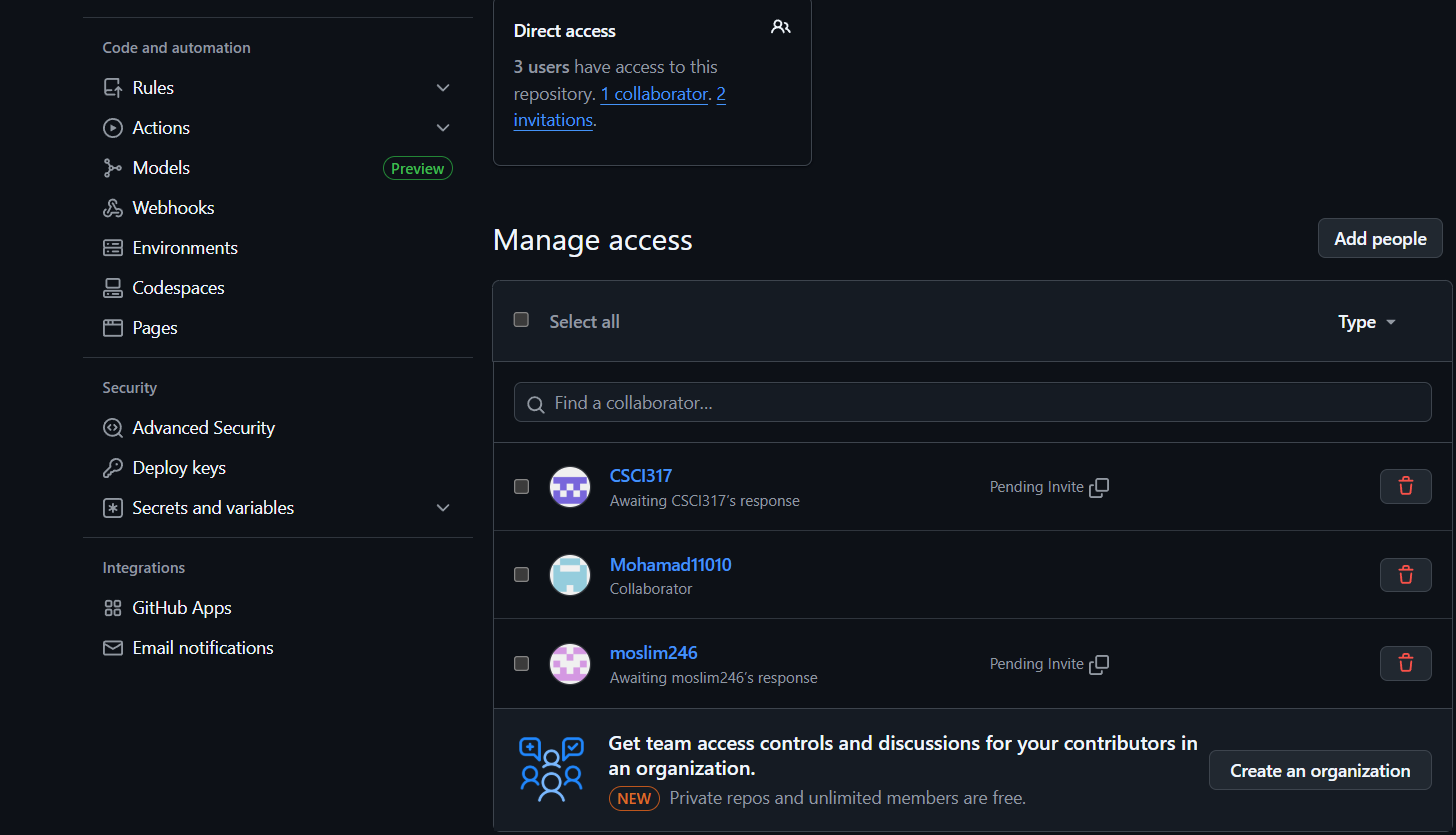
PERT Chart



Gantt Chart 

**Phase5:Github**





|  |  |
| --- | --- |
|  |  |
|  |  |