22i-2391 , 21i-1521

Work Breakdown Structure

🔹 1. Identify Project Scope and Objectives

1.1 Define Project Objectives

- A smart home service platform.
- Ensure reliable, verified, and easily accessible service providers.
- Provide transparent booking, tracking, and payment functionalities.

1.2 Measures of Effectiveness

- Reduced booking time.
- Increased service provider reliability.
- Higher user satisfaction (ratings & feedback).

1.3 Establish Project Authority

- Course Coordinator

1.4 Identify Stakeholders

- Homeowners (users)
- Service Providers
- Admin
- Development Team

1.5 Communication Methods

- Weekly sprint meetings
- WhatsApp group for quick communication
- GitHub for version control
- Trello for task tracking

🔹 2. Identify Project Infrastructure

2.1 Project & Strategic Alignment

- Increasing service efficiency and customer trust.

2.2 Project Team Organization

- Backend Developer (Spring Boot)
- Frontend Developer (Thymeleaf, JS, CSS)
- Database Manager
- UI/UX Designer

🔹 3. Analyze Project Characteristics

3.1 Project Type

- Product-driven web platform with multiple modules (Admin, User, Service Provider).

3.2 Key Project Characteristics

- Multi-role access
- Real-time data interaction
- Security in payment and data handling

3.3 High-Level Risks

- Data privacy and payment security

- Lack of availability of reliable service providers
- System scalability

3.4 User Requirements

- User-friendly interface
- Mobile responsiveness
- Fast search and booking

3.5 Lifecycle Approach

- Agile methodology with bi-weekly sprints

3.6 Resource Estimates

- 2-4 weeks, 2-member team

🔹 4. Identify Project Products and Activities

4.1 Major Product Deliverables

- Web Application with three modules: User, Service Provider, Admin
- Responsive UI
- API for backend services
- Database Schema

4.2 Product Flow (High Level)

- Register/Login → Browse Services → Book → Track → Review

4.3 Project Phases & Checkpoints

- Phase 1: User & Provider Auth System
- Phase 2: Booking Engine
- Phase 3: Tracking & Notification
- Phase 4: Admin Dashboard
- Phase 5: Testing & Deployment

🔹 5. Estimate Effort for Each Activity

5.1 Backend (Spring Boot)
5.2 Frontend (Thymeleaf)
5.3 Database (MySQL)
5.4 Integration & Testing

🔹 6. Identify Activity Risks

6.1 Risks

- Payment gateway issues
- Thymeleaf dynamic rendering errors
- Poor service provider data quality

🔹 7. Allocate Resources

1. Planning & Management (Shared)

1.1 Define project goals and scope
1.2 Set milestones and deadlines
1.3 Divide responsibilities
1.4 Create and manage Trello/GitHub board
1.5 Weekly progress meetings

2. Frontend Development

2.1 Design wireframes and UI layout
2.2 Implement User Sign-Up/Login pages
2.3 Create booking and service listings UI
2.4 Build real-time tracking and notifications UI
2.5 Integrate feedback, reviews, and chat interface
2.6 Connect frontend with backend APIs

3. Backend Development

3.1 Setup database (users, bookings, providers, reviews)
3.2 Implement authentication and user management
3.3 Create booking and payment processing APIs
3.4 Setup notifications, chat, and real-time tracking backend
3.5 Admin dashboard logic and reporting tools

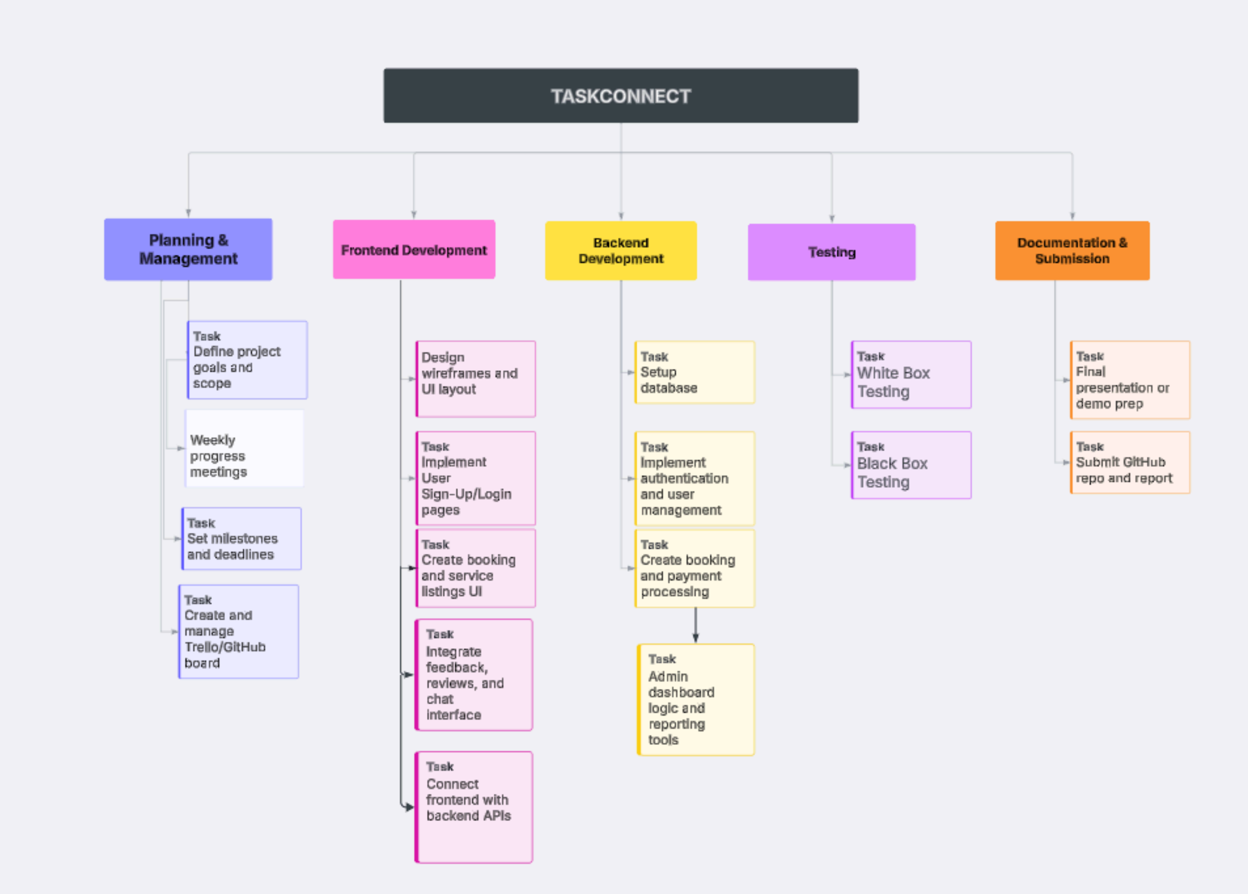
4. Testing (Both)

4.1 white box testing and black box testing

🔹 8. Review/Publicize Plan

8.1Final presentation or demo prep

8.2 Submit GitHub repo and report


TASKCONNECT
Planning & Management
Task Define project goals and scope
Weekly progress meetings
Task Set milestones and deadlines
Task Create and manage Trello/GitHub board
Frontend Development
Design wireframes and UI layout
Task Implement User Sign-Up/Login pages
Task Create booking and service listings UI
Task Integrate feedback, reviews, and chat interface
Task Connect frontend with backend APIs
Backend Development
Task Setup database
Task Implement authentication and user management
Task Create booking and payment processing
Task Admin dashboard logic and reporting tools
Testing
Task White Box Testing
Task Black Box Testing
Documentation & Submission
Task Final presentation or demo prep
Task Submit GitHub repo and report

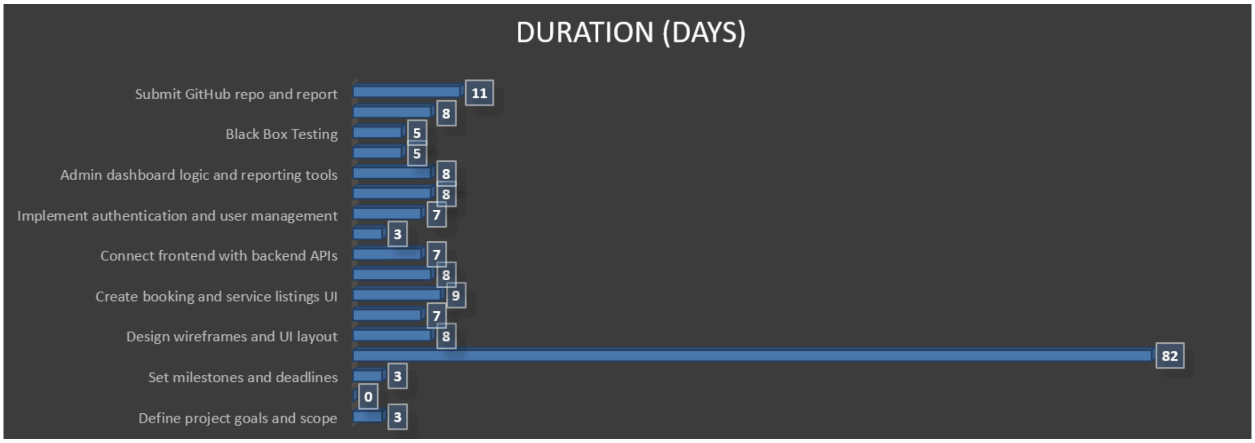
GANTT CHART

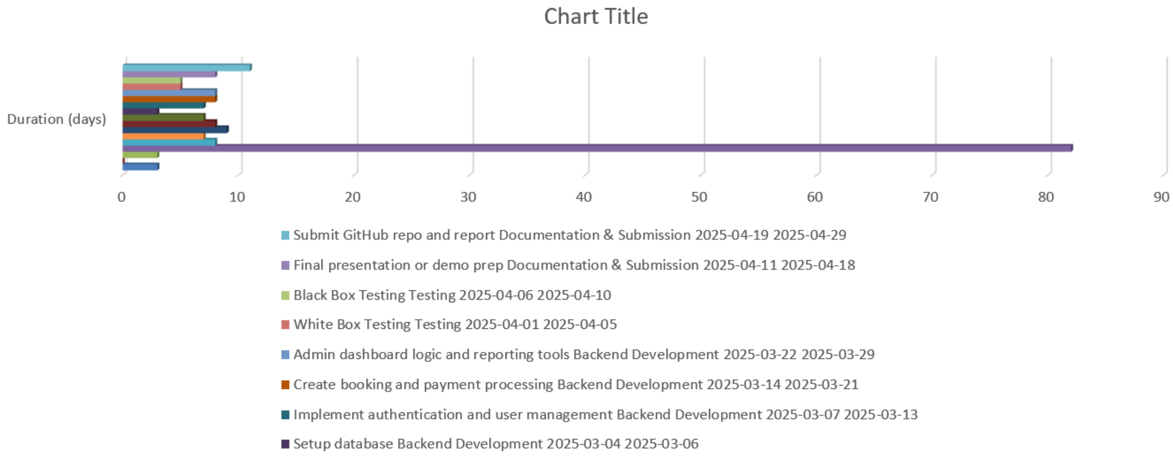
Excel

Task	Category	Start Date	End Date	Duration (days)

Define project goals and scope	Planning & Management	2025-02-01	2025-02-03	3
Weekly progress meetings	Planning & Management	2025-02-01	2025-04-26	86 (weekly)
Set milestones and deadlines	Planning & Management	2025-02-04	2025-02-06	3
Create and manage Trello/GitHub board	Planning & Management	2025-02-07	2025-04-29	82
Design wireframes and UI layout	Frontend Development	2025-02-08	2025-02-15	8
Implement User Sign-Up/Login pages	Frontend Development	2025-02-16	2025-02-22	7
Create booking and service listings UI	Frontend Development	2025-02-23	2025-03-02	9
Integrate feedback, reviews, and chat interface	Frontend Development	2025-03-03	2025-03-10	8
Connect frontend with backend APIs	Frontend Development	2025-03-11	2025-03-17	7
Setup database	Backend Development	2025-03-04	2025-03-06	3
Implement authentication and user management	Backend Development	2025-03-07	2025-03-13	7
Create booking and payment processing	Backend Development	2025-03-14	2025-03-21	8
Admin dashboard logic and reporting tools	Backend Development	2025-03-22	2025-03-29	8
White Box Testing	Testing	2025-04-01	2025-04-05	5
Black Box Testing	Testing	2025-04-06	2025-04-10	5
Final presentation or demo prep	Documentation & Submission	2025-04-11	2025-04-18	8

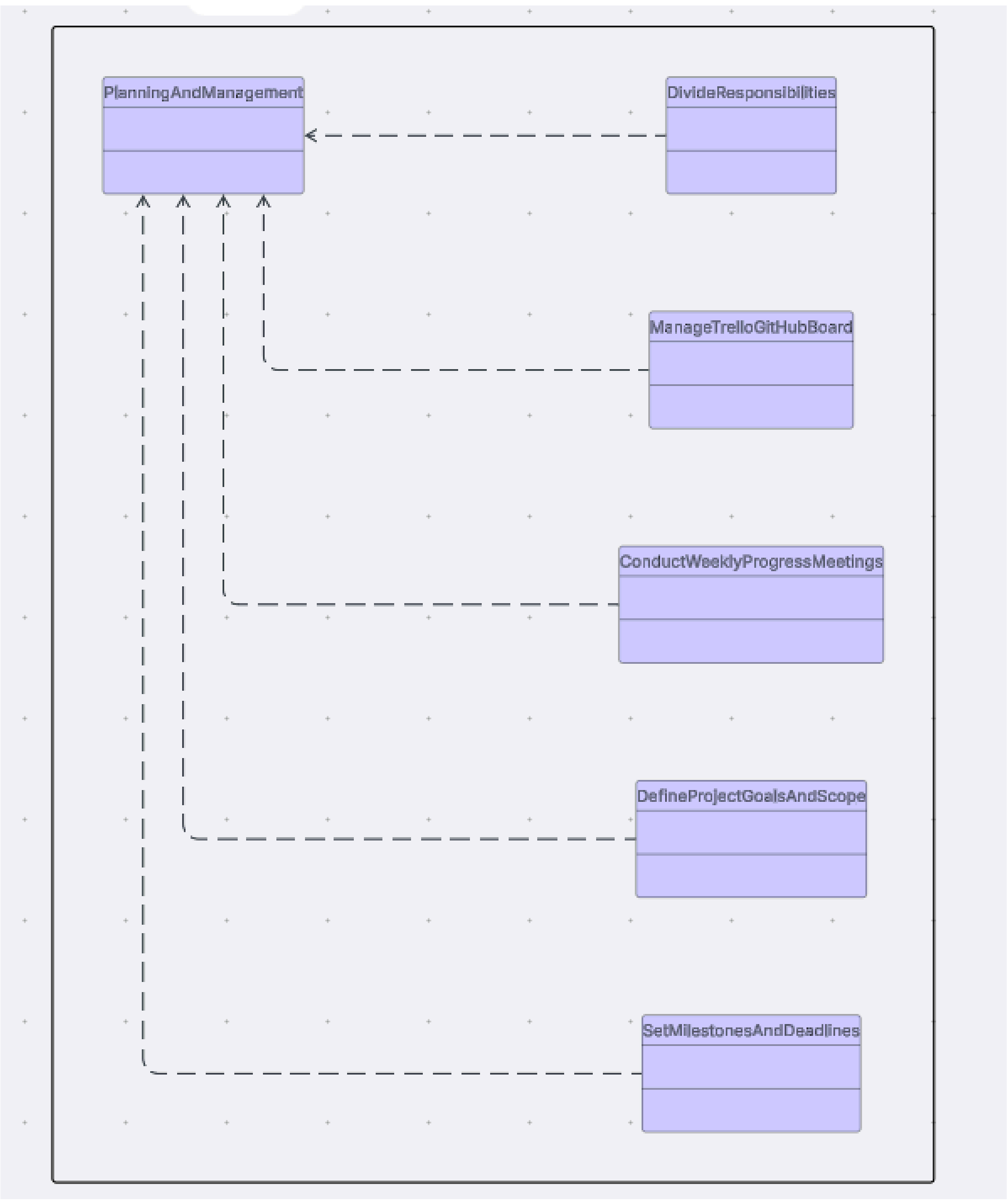
Submit GitHub repo and report	Documentation & Submission	2025-04-19	2025-04-29	11



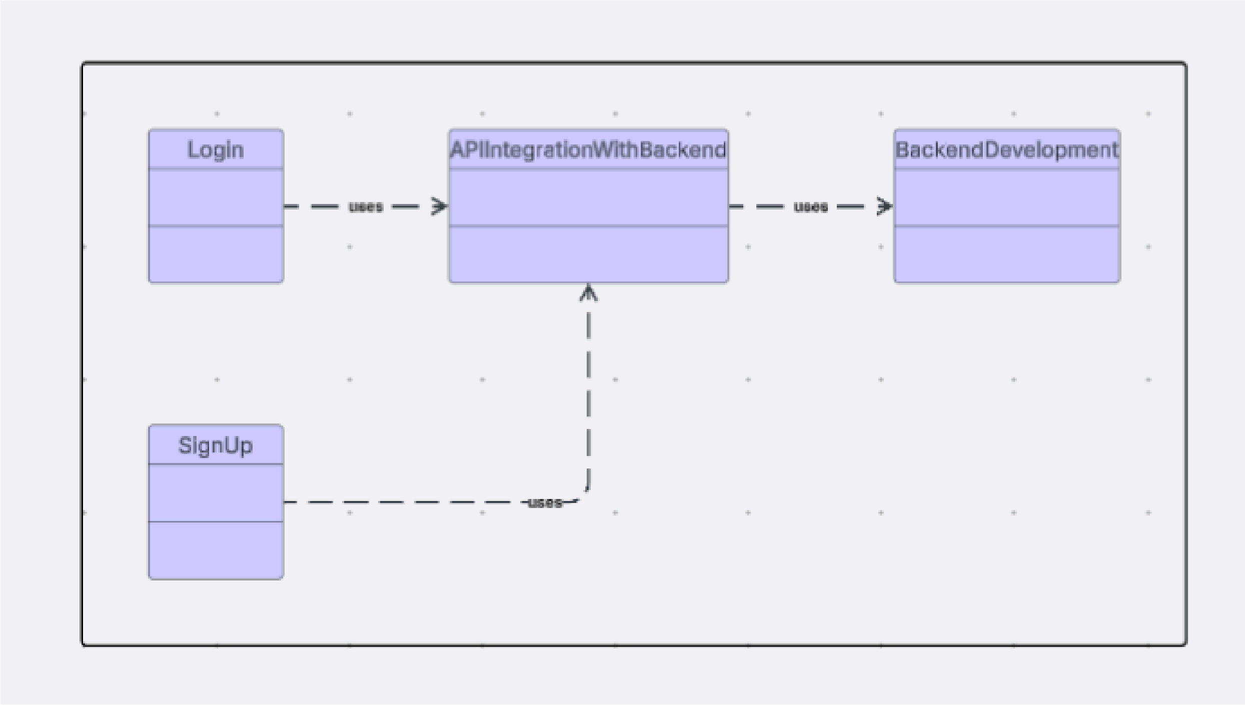


B- System Architecture [15 Marks]

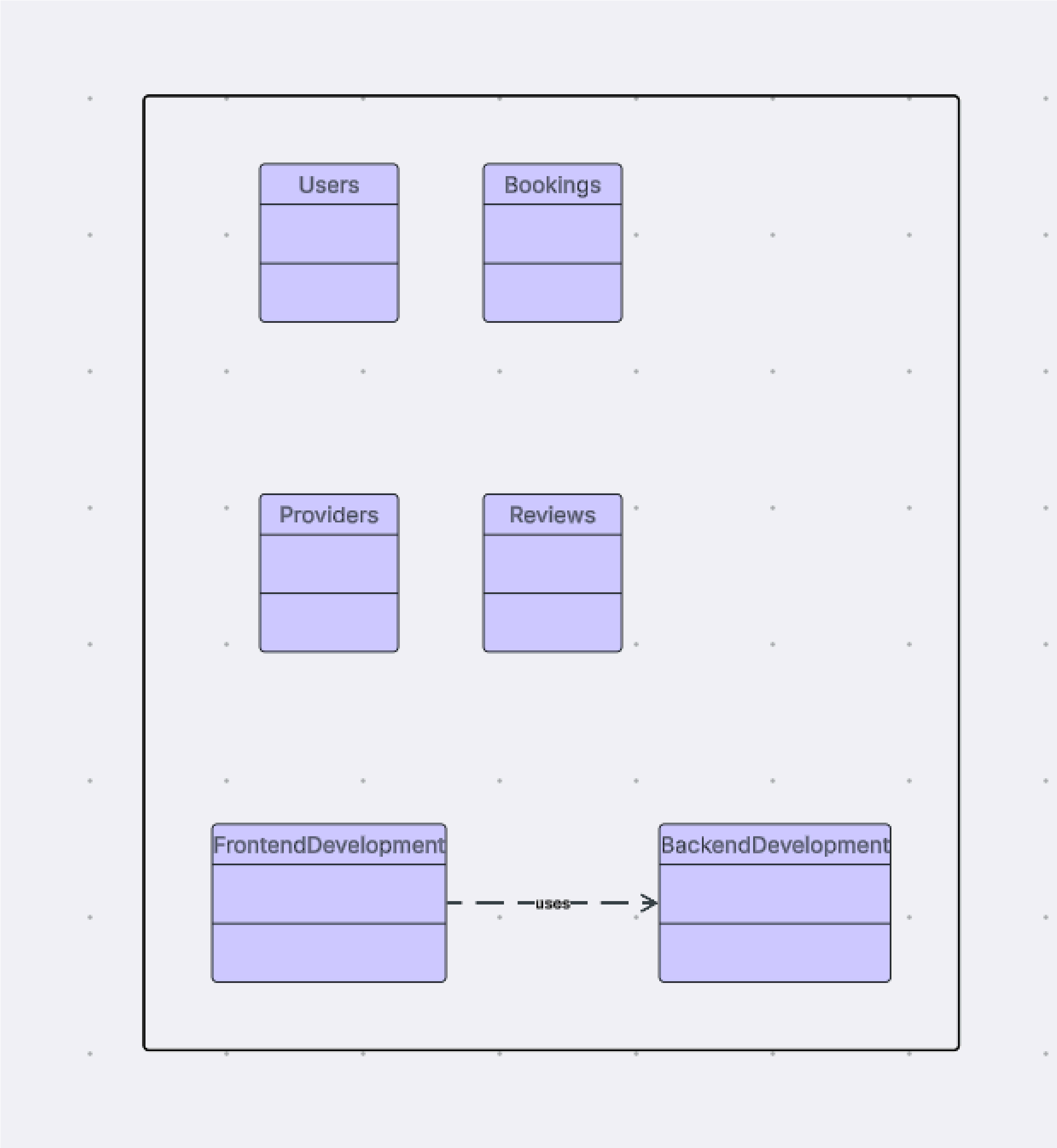
- Identifying Subsystems**[3 Marks]**
    1. **PlanningAndManagement**


PlanningAndManagement
DivideResponsibilities
ManageTrelloGitHubBoard
ConductWeeklyProgressMeetings
DefineProjectGoalsAndScope
SetMilestonesAndDeadlines

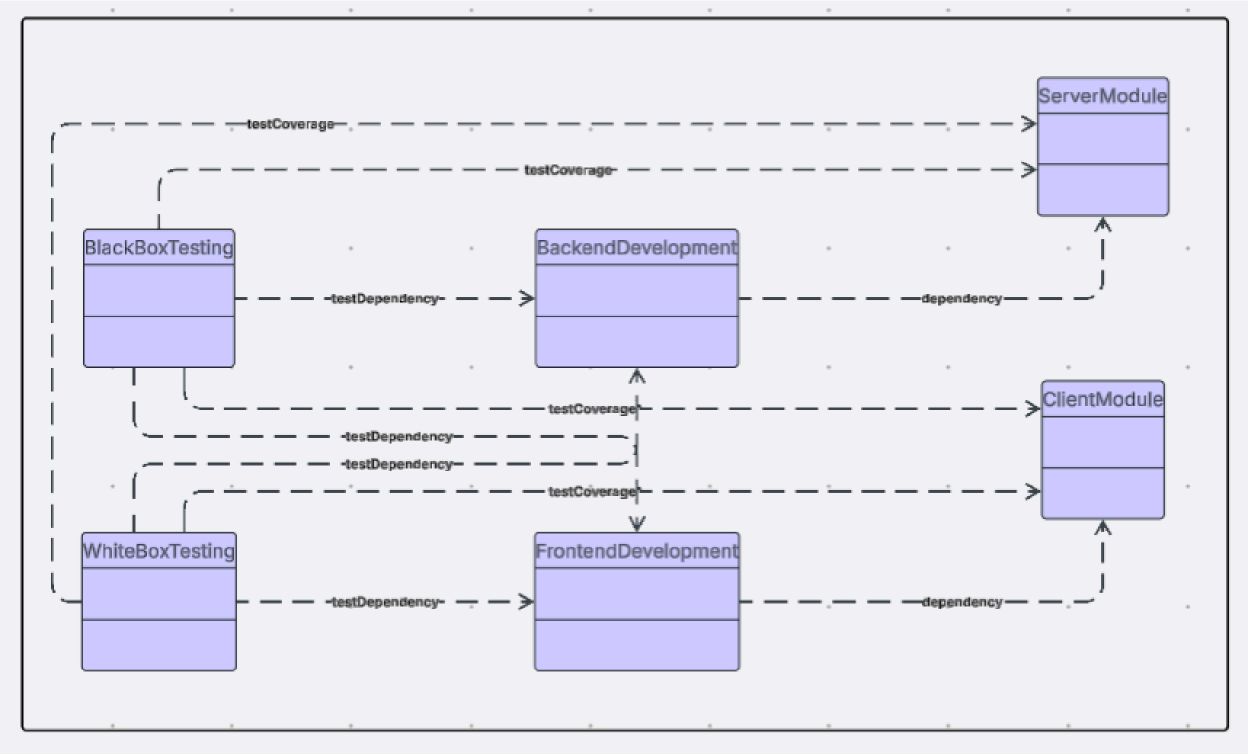
2. FrontendDevelopment



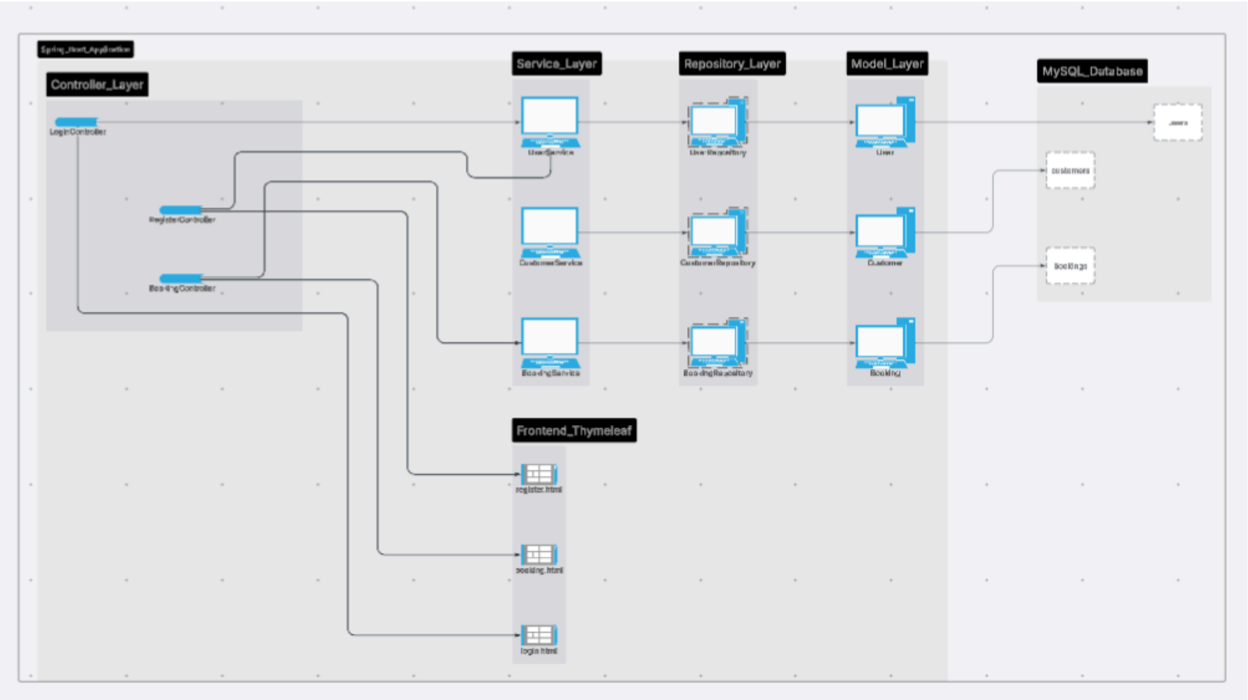
3. **BackendDevelopment**


Users
Bookings
Providers
Reviews
FrontendDevelopment
uses
BackendDevelopment

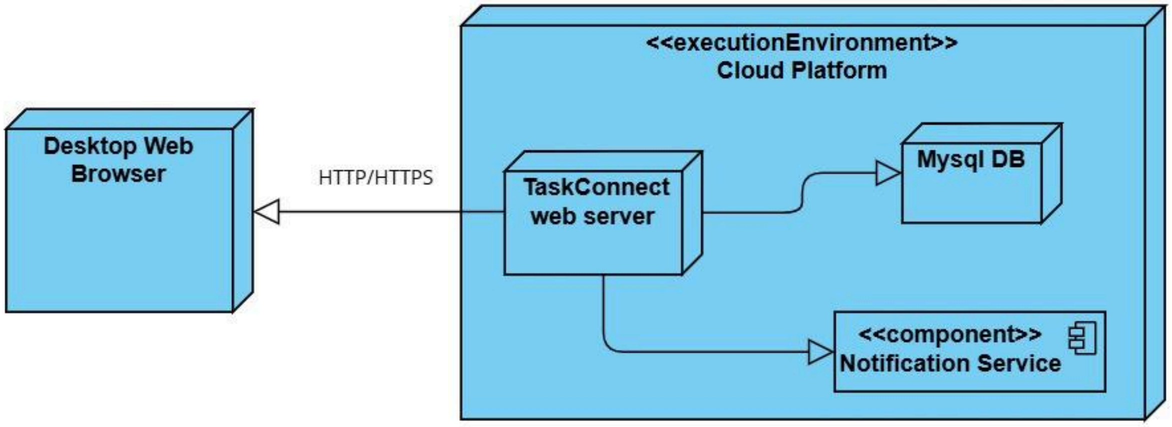
4. Testing



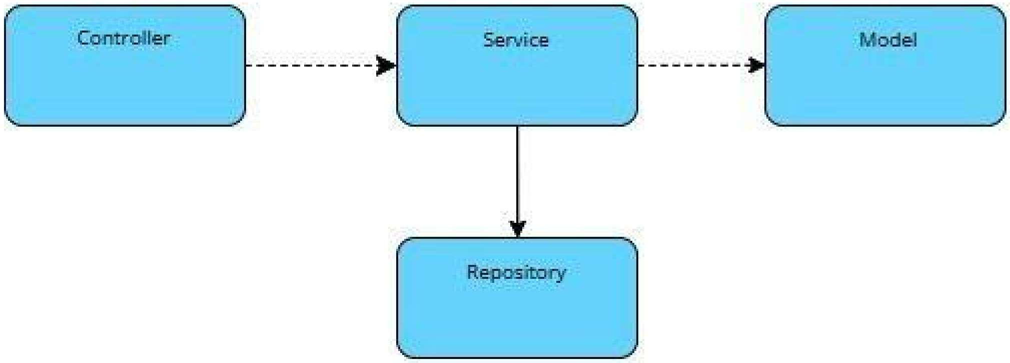
- Architecture Styles**[4 Marks]**
  - Describe the architectural styles used in your design

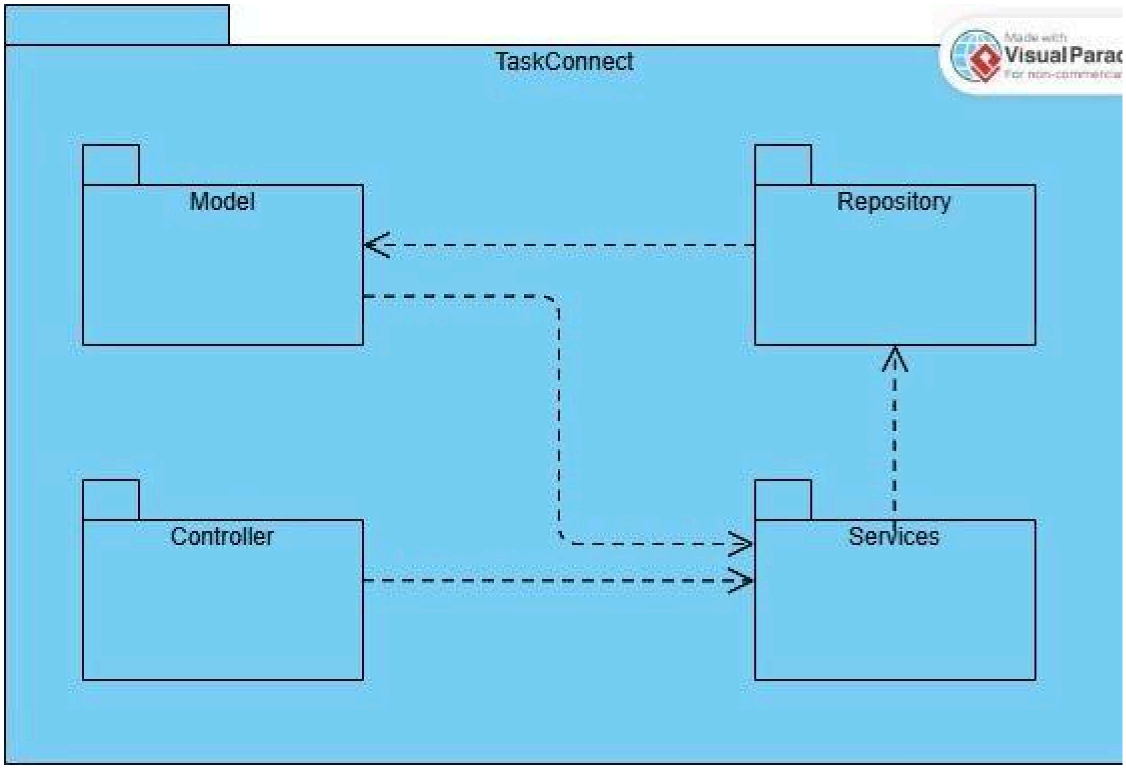


- Deployment diagram for client deployments**[4 Marks]**



- Component diagrams showing which services/components/subsystems you expect to be interfacing with or modifying in order to implement your enhancements.**[4 Marks]**





C- TestCases BlackBox[20 Marks]

1. Equivalence Class Partitioning (10 Marks)

Input: User registration (email, password, role).

Input Field	Valid Classes	Invalid Classes
Email	Valid format (a@b.com)	Missing @ (ab.com), Empty
Password	Length ≥ 8, has special chars	Length < 8, no special chars
Role	"client", "worker"	"admin", empty

Test Cases:

TC#	Input (email, password, role)	Expected Result	Class Tested
1	a@b.com, Pass@123, client	Success	Valid all fields

2	ab.com, Pass@123, client	Fail (invalid email)	Invalid email
3	a@b.com, pass, client	Fail (short password)	Invalid password
4	a@b.com, Pass@123, admin	Fail (invalid role)	Invalid role

2. Boundary Value Analysis (10 Marks)

Input: Booking price (valid range:
10
–
10–1000).

TC#	Input (price)	Expected Result	Boundary Tested
5	10	Success	Lower boundary
6	1000	Success	Upper boundary
7	9	Fail	Below lower bound
8	1001	Fail	Above upper bound

3. User Story Validation

User Story: "As a user, I want to book a service so that I can hire workers."

TC#	Action	Expected Result
9	Submit booking with valid details	Booking created, status="pending"
10	Submit booking with invalid price	Error message, no booking

D- White Box Testing [20 Marks]

1. Unit Testing (JUnit Example)

Test Class: BookingServiceTest.java

java

```
@Test
public void testSaveBooking_ValidPrice() {
   Booking booking = new Booking();
   booking.setPrice(50.0); // Valid price
   when(bookingRepository.save(booking)).thenReturn(booking);
   Booking savedBooking = bookingService.saveBooking(booking);
   assertEquals(50.0, savedBooking.getPrice());
}

@Test
public void testSaveBooking_InvalidPrice() {
   Booking booking = new Booking();
   booking.setPrice(5.0); // Invalid price
   assertThrows(IllegalArgumentException.class, () -> {
      bookingService.saveBooking(booking);
   });
```

}

E - Final Report [20 Marks]

1. Introduction

TaskConnect:Smart Service for Homes

Problem Statement

The problem of	Difficulty in finding reliable and skilled home maintenance and repair service providers
Affects	Homeowners and service providers
The impact of which is	Inefficiency in service discovery, booking delays, uncertainty in pricing, and lack of reliability
A successful solution would be	A digital platform, **TaskConnect: Smart Service for Homes**, that connects homeowners with verified professionals for home services. The platform would allow easy booking, display verified professionals with ratings, provide transparent pricing, offer real-time tracking, secure payment options, and customer reviews to ensure service quality. It would streamline the entire process, making home service management more efficient, reliable, and convenient.

2. Functional Requirements

<u>User Management</u>

- Users can register as either a Customer or a Service Provider
- Users can log in / log out securely
- Profile information includes:
  - Name, Email, Phone, Address
  - Role (Customer/Provider)
  - Password with encryption
- Users can edit/update their profile
- Email and/or phone verification upon registration

Service Booking

- Customers can:
  - Select service category (e.g., plumber, electrician)
  - Choose date & time for booking
  - Enter location & service description
- Bookings can be:
  - Immediate (emergency) or Scheduled
- Bookings receive unique IDs and are stored in DB
- Booking status: Pending, Completed, Cancelled

Service Provider Profiles

- Each provider has:
  - Name, Skills, Experience, Ratings
  - Service area (zip codes or city)
  - Availability calendar

- Customers can view service provider profiles before booking
- Providers can update their availability and service rates
- Admin can review and approve providers

Payments & Price Estimation

- App provides estimated pricing based on:
  - Service type
  - Duration or complexity
- Optional discount codes or promotions
- Integration with payment gateway (e.g., Razorpay, PayPal)
- Users receive invoice upon successful payment
- Payment status tracked (Pending, Paid, Failed)

Feedback & Ratings

- After job completion, customers can:
  - Rate service provider (1 to 5 stars)
  - Leave written feedback
- Providers can also rate customers (optional)
- Average ratings are displayed on profiles
- Admin can flag/remove inappropriate content

Notifications & Reminders

- Email/SMS notifications for:
  - Booking confirmations and updates
  - Payment receipts

    - Profile changes

- Reminders before a scheduled booking
- Admin or system can send mass notifications (e.g., promotions or policy changes)

Emergency Services

- Users can mark a booking as “Emergency”
- System prioritizes:
    - Nearest available service provider
    - Fastest estimated arrival
- Emergency bookings may have:
    - Additional charges
    - Limited cancellation/refund options

Service History & Invoices

- Users can view:
    - Past bookings, including date, time, provider, price, and rating
- Invoices include:
    - Service details
    - Provider name
    - Total cost and tax breakdown
- Export/download invoice as PDF
- Admin has access to all transaction records

3. Non Functional Requirements

Product Requirements

1) Service booking transactions should be processed within 3 seconds.

2) The web app should have an intuitive UI/UX to ensure ease of use for both customers and service providers.

3) The system should handle at least 500 concurrent users without performance degradation.

4) The system should have 99.9% uptime, ensuring it is available at all times.

5) The system should be compatible with Android and iOS browsers.

6) Logs and error reports should be maintained for at least 6 months for debugging purposes.

7) The system must comply with GDPR (General Data Protection Regulation) for handling user data.

Organizational Requirements

1) The system must be developed using Spring Boot (backend) and MySQL (database) following the MVC architecture.

2) The system should follow agile development methodology with bi-weekly sprints.

3) A version control system (GitHub/GitLab) should be used for code management.

4) Regular code reviews and testing should be conducted before deployment.

External Requirements

1) Payment transactions should follow financial regulations and taxation laws in the operating regions.

2) The system must integrate with Google Maps API for service location tracking.

3) The platform should support SMS and email notifications.

4) The system should offer competitive service fees to attract more users.

4. User Stories

Story ID: 1	**Story Title:** *User Sign-Up*
User Story: As a new user,	

I want to sign up using my phone number or email,
so that I can create an account and access services.

Acceptance Criteria:

- Users can enter phone/email for registration.
- System sends a verification code.
- User receives a confirmation message.

Type:
Workflow

Story ID: 2 **Story Title:** *User Login*

User Story:

As a registered user,
I want to log in using my credentials,
so that I can access my account and book services.

Acceptance Criteria:

- Users can log in via phone/email and password.
- System provides a password reset option.
- Successful login redirects users to the dashboard.

Type:
[ ] Search
[✓] Workflow
[ ] Manage Data
[ ] Payment
[ ] Report/View

Story ID: 3 **Story Title:** *Book a Service*

User Story:

As a customer,
I want to book a service online,
so that I can get professional assistance at my home.

Acceptance Criteria:

- User selects a service.
- User provides location and preferred time.
- Booking confirmation is sent to the user.

Type:
[ ] Search
[✔] Workflow
[ ] Manage Data
[ ] Payment
[ ] Report/View

Story ID: 4 **Story Title:**Track Service Provider

User Story:

As a customer,
I want to track my assigned service provider in real-time,
so that I know when they will arrive.

Acceptance Criteria:

- User receives a tracking link.
- Provider's live location and estimated arrival time are displayed

Type:
[ ] Search
[ ] Workflow
[ ] Manage Data
[ ] Payment
[✔] Report/View

Story ID: 5 **Story Title:**Make a Payment

User Story:

As a customer,
I want to pay securely through digital wallets or bank transfers,
so that I can complete transactions without cash.

Acceptance Criteria:

- User selects payment methods (Easypaisa, JazzCash, etc.).
- Payment is processed securely.
- Confirmation is sent after a successful transaction.

Type:
[ ] Search
[ ] Workflow
[ ] Manage Data
[✔] Payment
[ ] Report/View

Story ID: 6 **Story Title:** Rate Service Provider

User Story:

As a customer,
I want to rate my service provider,
so that I can help improve service quality.

Acceptance Criteria:

- Users give a 1-5 star rating.
- Users can add a review.
- Ratings are stored and displayed on provider profiles.

Type:
[ ] Search
[ ] Workflow
[ ] Manage Data
[ ] Payment
[✔] Report/View

Story ID: 7 **Story Title:** View Past Bookings

User Story:

As a customer,
I want to view my past bookings,
so that I can keep track of my service usage

Acceptance Criteria:

- User sees a list of previous bookings.

- Each entry includes date, time, provider, and service type.

Type:
[ ] Search
[ ] Workflow
[ ] Manage Data
[ ] Payment
[✔] Report/View

Story ID: 8 **Story Title:** Cancel a Service

User Story:

As a customer,
I want to cancel a booked service,
so that I can manage my schedule flexibly.

Acceptance Criteria:

- Users can cancel before service start time.
- User receives a cancellation confirmation.

Type:
[ ] Search
✔ Workflow
[ ] Manage Data
[ ] Payment
[ ] Report/View

Story ID: 9 **Story Title:** Request a Refund

User Story:

As a customer,
I want to request a refund after canceling a service,
so that I can recover my money.

Acceptance Criteria:

- User submits a refund request.
- System processes and updates refund status.

Type:
[ ] Search
[ ] Workflow
[ ] Manage Data
✔ Payment
[ ] Report/View

Story ID: 10 **Story Title:** Live Chat with Support

User Story:

As a user,
I want to chat with customer support,
so that I can resolve my queries.

Acceptance Criteria:

- Users can start a chat with support.
- Support team responds to queries in real time.

Type:
[ ] Search
[ ] Workflow
✓ Manage Data
[ ] Payment
[ ] Report/View

Story ID: 11 **Story Title:** Service Provider Profile

User Story:

As a service provider,
I want to manage my profile,
so that customers can see my expertise and experience.

Acceptance Criteria:

- Provider can update their skills, experience, and availability.
- Profile displays ratings and reviews.

Type:
[ ] Search
[ ] Workflow
✓ Manage Data
[ ] Payment
[ ] Report/View

Story ID: 12 **Story Title:** Find Nearby Providers

User Story:

As a customer,
I want to search for nearby service providers,
so that I can book local professionals.

Acceptance Criteria:

- User enters a location.
- Nearby providers are listed.

Type:
✓ Search
[ ] Workflow
[ ] Manage Data
[ ] Payment
[ ] Report/View

Story ID: 13 **Story Title:** Receive Notifications

User Story:

As a customer,
I want to receive notifications for bookings and service updates,
so that I stay informed.

Acceptance Criteria:

- Users receive notifications for service status updates.
- Notifications appear in the app and via SMS.

Type:
[ ] Search
[ ] Workflow
[ ] Manage Data
[ ] Payment
✔ Report/View

Story ID: 14 **Story Title:** Apply Discounts

User Story:

As a customer,
I want to use promo codes,
so that I can get discounts on my services.

Acceptance Criteria:

- User enters a valid promo code at checkout.
- Discount is applied before payment.

Type:
[ ] Search
[ ] Workflow
[ ] Manage Data
[ ] Payment
✔Report/View

Story ID: 15 **Story Title:** Book Multiple Services

User Story:

As a customer,
I want to book multiple services at once,
so that I can save time.

Acceptance Criteria:

- User selects multiple services.
- System schedules and confirms all bookings.

Type:
[ ] Search
✓ Workflow
[ ] Manage Data
[ ] Payment
[ ] Report/View

Story ID: 16 **Story Title:** Referral Program

User Story:

As a customer,
I want to refer a friend and earn rewards,
so that I can get discounts on future bookings.

Acceptance Criteria:

- User shares a referral link.
- Both referrer and new user receive discounts after signup.

Type:
[ ] Search
[ ] Workflow
✓Manage Data
[ ] Payment
[ ] Report/View

Story ID: 17 **Story Title:** Request Urgent Service

User Story:

As a customer,
I want to book emergency services,

so that I can get urgent repairs.

Acceptance Criteria:

- User selects an emergency option.
- Service provider is assigned within 30 minutes.

Type:
[ ] Search
✔ Workflow
[ ] Manage Data
[ ] Payment
[ ] Report/View

Story ID: 18 **Story Title:** Submit Feedback

User Story:

As a customer,
I want to submit feedback,
so that I can share my experience.

Acceptance Criteria:

- User fills out a feedback form.
- System records and displays feedback.

Type:
[ ] Search
[ ] Workflow
[ ] Manage Data
[ ] Payment
✔ Report/View

Story ID: 19 **Story Title:** Personalized Service Suggestions

User Story:

As a customer,
I want AI to recommend services based on my past bookings,

so that I can find what I need quickly.

Acceptance Criteria:

- AI analyzes user history.
- System suggests relevant services.

Type:
✓Search
[ ] Workflow
[ ] Manage Data
[ ] Payment
[ ] Report/View

Story ID: 20 **Story Title:** Adjust Service Pricing Based on Demand

User Story:

As an admin,
I want to adjust service prices dynamically,
so that I can optimize revenue based on demand.

Acceptance Criteria:

- System suggests price changes for high-demand services.
- Admin approves or rejects price updates.

Type:
[ ] Search
[ ] Workflow
✓Manage Data
[ ] Payment
[ ] Report/View

Story ID: 21 **Story Title:**Detect Suspicious Activities

User Story:

As an admin,
I want to detect fraudulent activities,
so that I can prevent misuse of the platform.

Acceptance Criteria:

- System flags multiple failed payments.
- System alerts admin for suspicious accounts.

Type:
[ ] Search
[ ] Workflow
✓ Manage Data
[ ] Payment
[ ] Report/View

Story ID: 22 **Story Title:** Manage User Complaints

User Story:

As an admin,
I want to review and resolve customer complaints,
so that I can ensure high service quality.

Acceptance Criteria:

- Admin sees a list of submitted complaints.
- System assigns complaints to support agents.
- User receives a resolution status update.

Type:
[ ] Search
[ ] Workflow
✓ Manage Data
[ ] Payment
[ ] Report/View

Story ID: 23 **Story Title:** Track Popular Services

User Story:

As an admin,
I want to track the most frequently booked services,
so that I can optimize marketing and resource allocation

Acceptance Criteria:

- System generates reports on top services.
- Admin can filter reports by location and time.

Type:
[ ] Search
[ ] Workflow
[ ]Manage Data
[ ] Payment
✓ Report/View

Story ID: 24 **Story Title:** Approve or Reject Service Providers

User Story:

As an admin,
I want to approve or reject new service providers,
so that only verified professionals can join.

Acceptance Criteria:

- Admin reviews provider applications.
- System updates provider status (Approved/Rejected).

Type:
[ ] Search
[ ] Workflow
✓Manage Data
[ ] Payment
[ ] Report/View

Story ID: 25 **Story Title:** Service Information

User Story:

As a user,
I want to view detailed information about a service,
so that I can decide whether to book it.

Acceptance Criteria:

- Users can see descriptions, images, pricing, and reviews.
- Contact information of the service provider is available.

Type:
[ ] Search
[ ] Workflow
[ ]Manage Data
[ ] Payment
✓ Report/View

5. Product Backlog

High-Priority User Stories

1) User Sign-Up
2) User Login
3) Book a Service
4) Make a Payment
5) Rate Service Provider
6) Cancel a Service
7) Service Provider Profile
8) Receive Notifications
9) Submit Feedback
10) Approve or Reject Service Providers
11) Service Information

Medium-Priority User Stories

1) View Past Bookings
2) Request a Refund
3) Adjust Service Pricing Based on Demand
4) Book Multiple Services
5) Request Urgent Service
6) Apply Discounts
7) Manage User Complaints

Low-Priority User Stories

1) Live Chat with Support
2) Personalized Service Suggestions
3) Referral Program
4) Find Nearby Providers
5) Detect Suspicious Activities
6) Track Popular Services

6. Sprint-1 Backlog

1) User Sign-up
2) User-Login
3) User Dashboard
4) Service Provider Dashboard
5) Book a service
6) Add Services

7. Sprint-2 Backlog

1) View Past Bookings
2) Request a Refund
3) Track Popular Services
4) Rate Service Provider
5) Cancel a Service
6) Receive Notifications
7) Submit Feedback
8) Approve or Reject Service Providers
9) Service Information

8. Project Plan

Project Objectives

TaskConnect is a web-based platform designed to connect customers with verified service providers (e.g., plumbers, electricians). The project aims to streamline service bookings, manage profiles, enable secure payments, and ensure reliable service delivery.

Scope of the Project

1) Web-based system with customer, provider, and admin interfaces
2) User registration and login

3) Service booking and scheduling
4) Profile management
5) Notification system (email/SMS)
6) Feedback and ratings
7) Emergency request feature

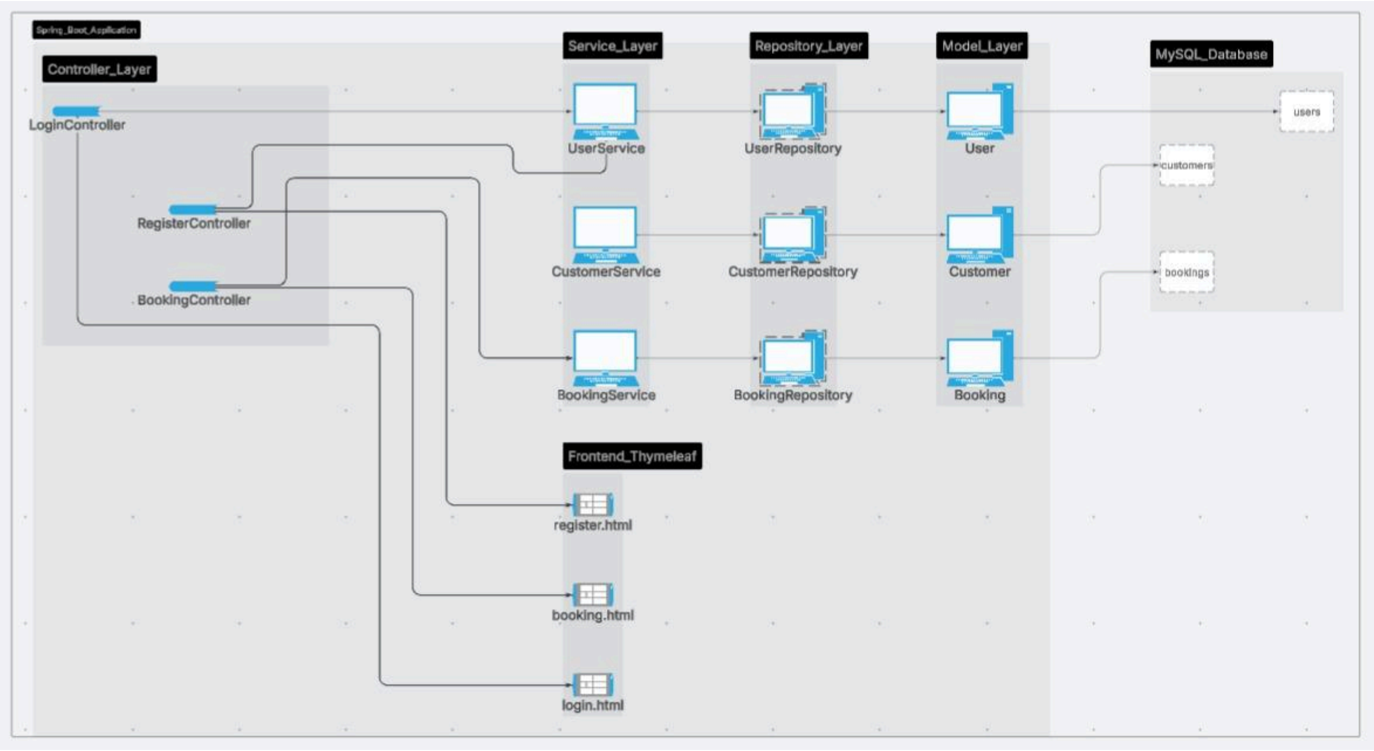
Team Roles and Responsibilities

Team Member	**Role**	**Responsibilities**
Imra Tariq	Scrum Master	Planning, backend, coordination, Schema design, MySQL setup, queries, Testing.
Abdulmanan	Developer	UI/UX using HTML/CSS, Thymeleaf. Java Model Classes, Testing.

Tools and Technologies

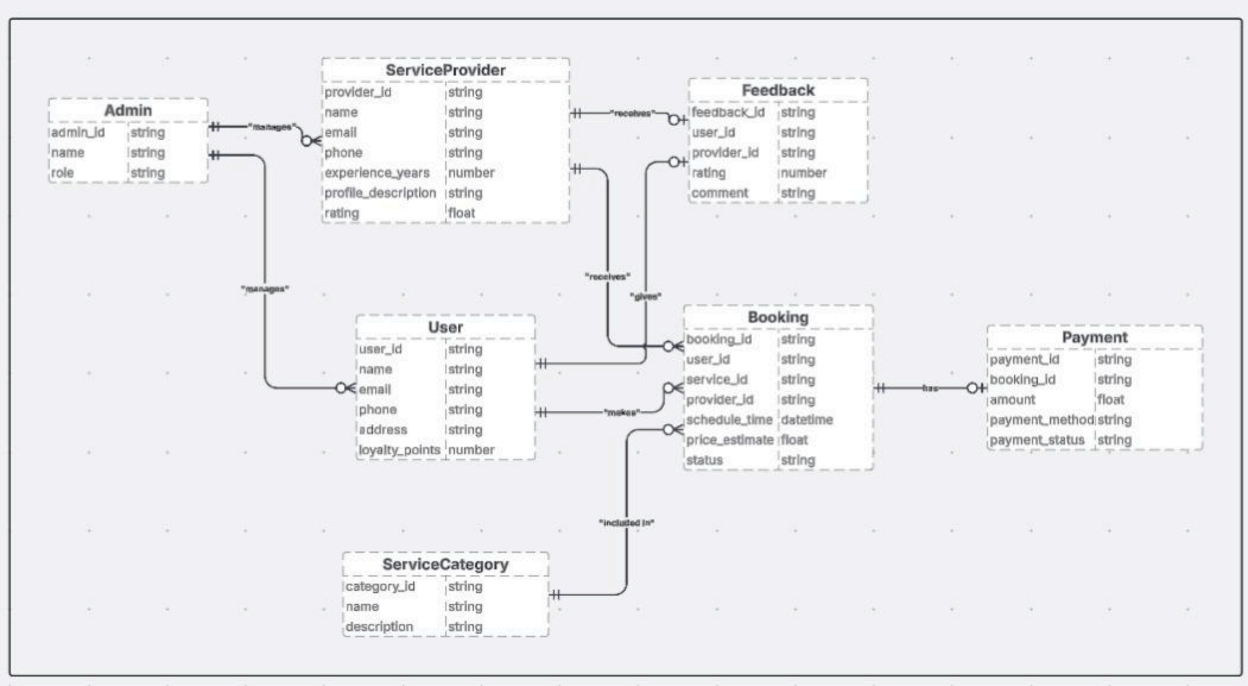
1) Java, Spring Boot, Thymeleaf
2) MySQL, JPA/Hibernate
3) HTML/CSS, Bootstrap
4) Eclipse IDE
5) GitHub for version control

9. Architecture Diagram

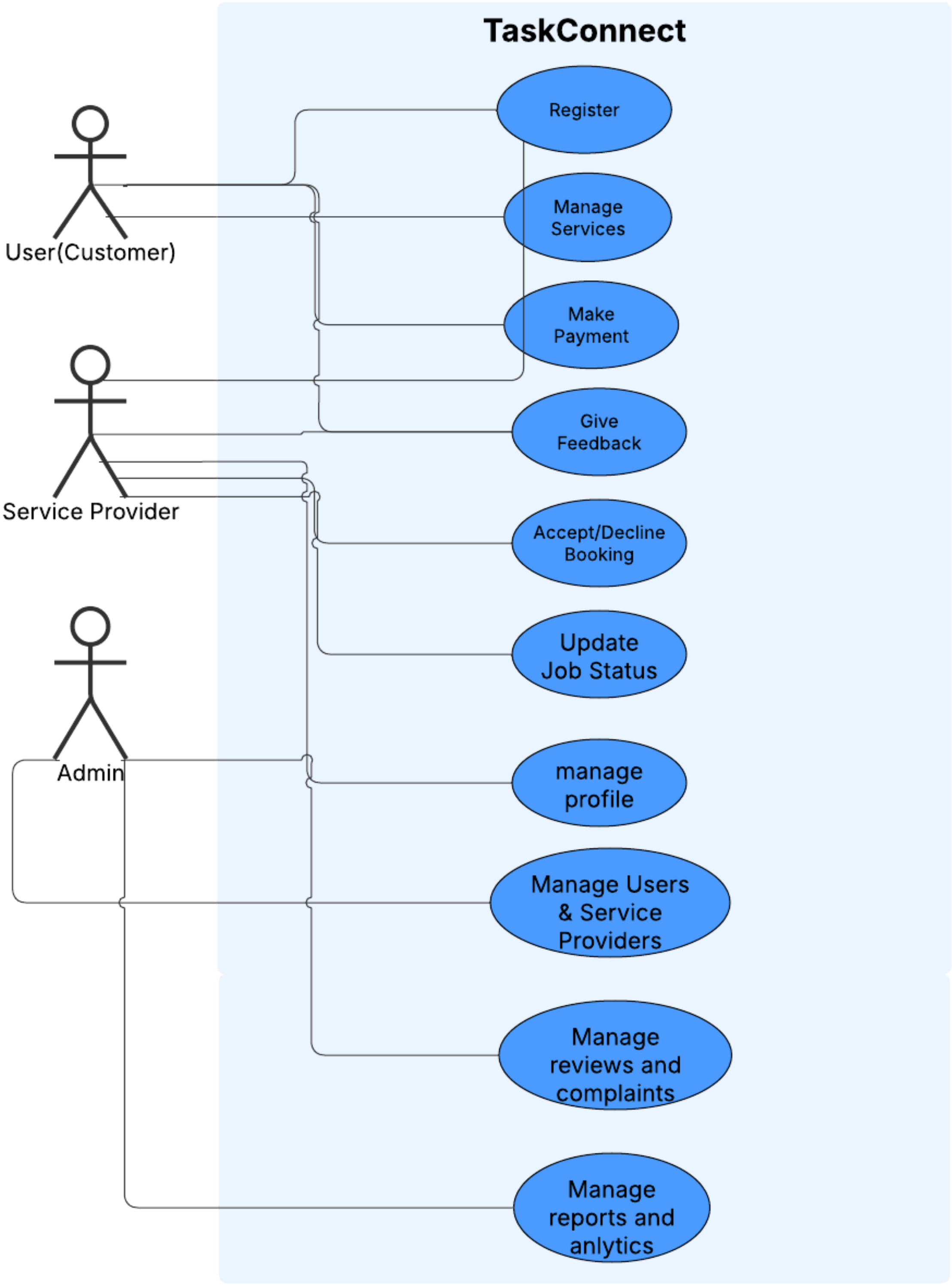


10. Design

Class Diagram

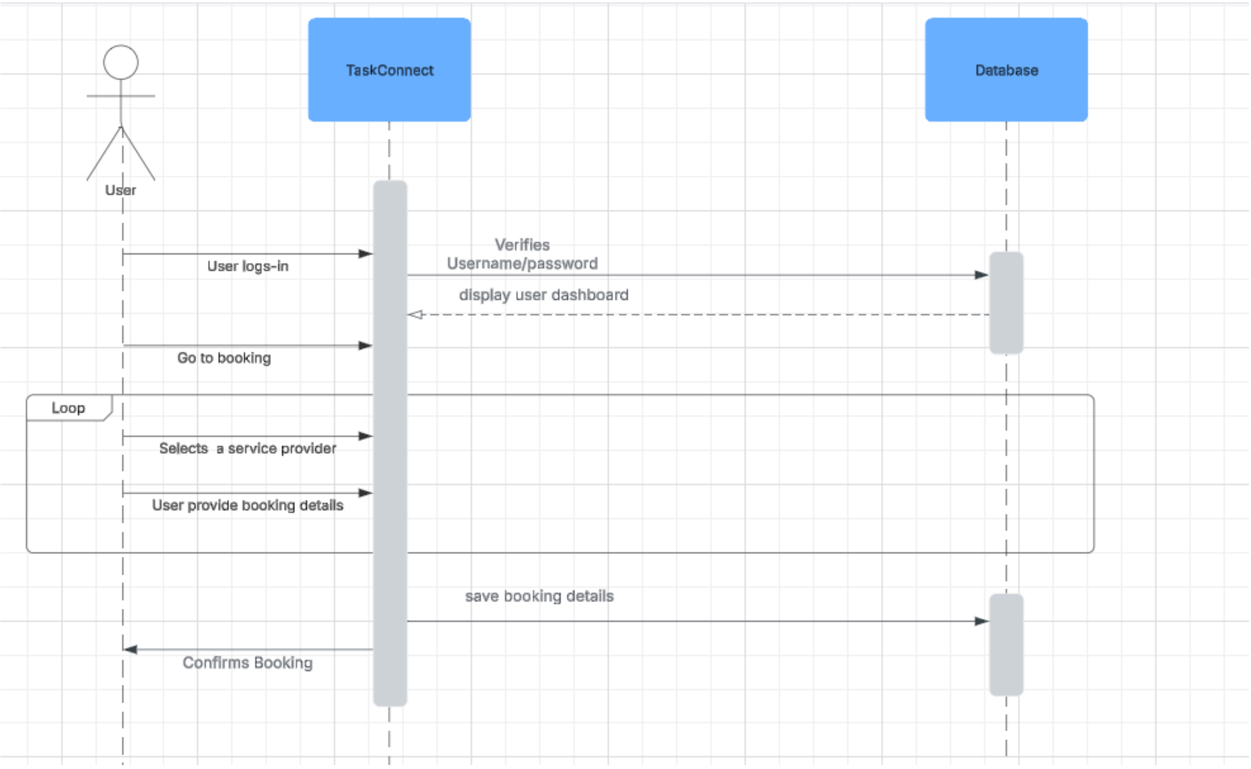


Use Case Diagram

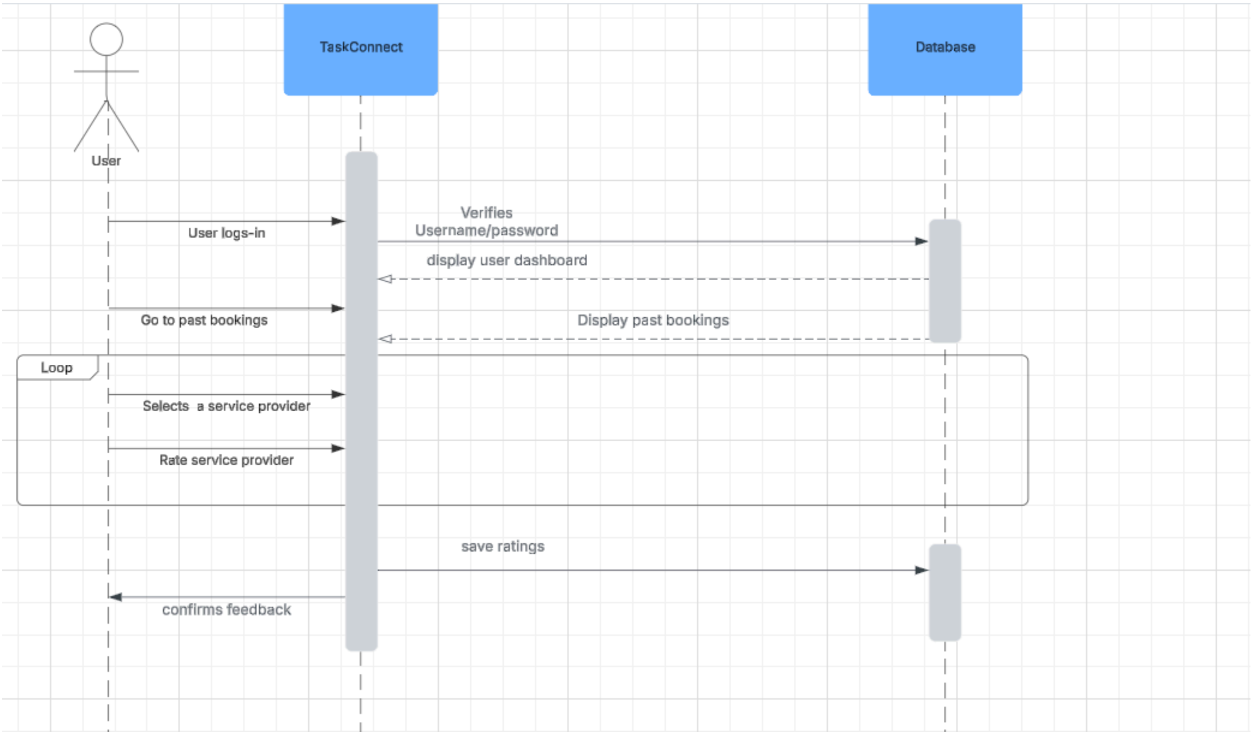


Sequence Diagram

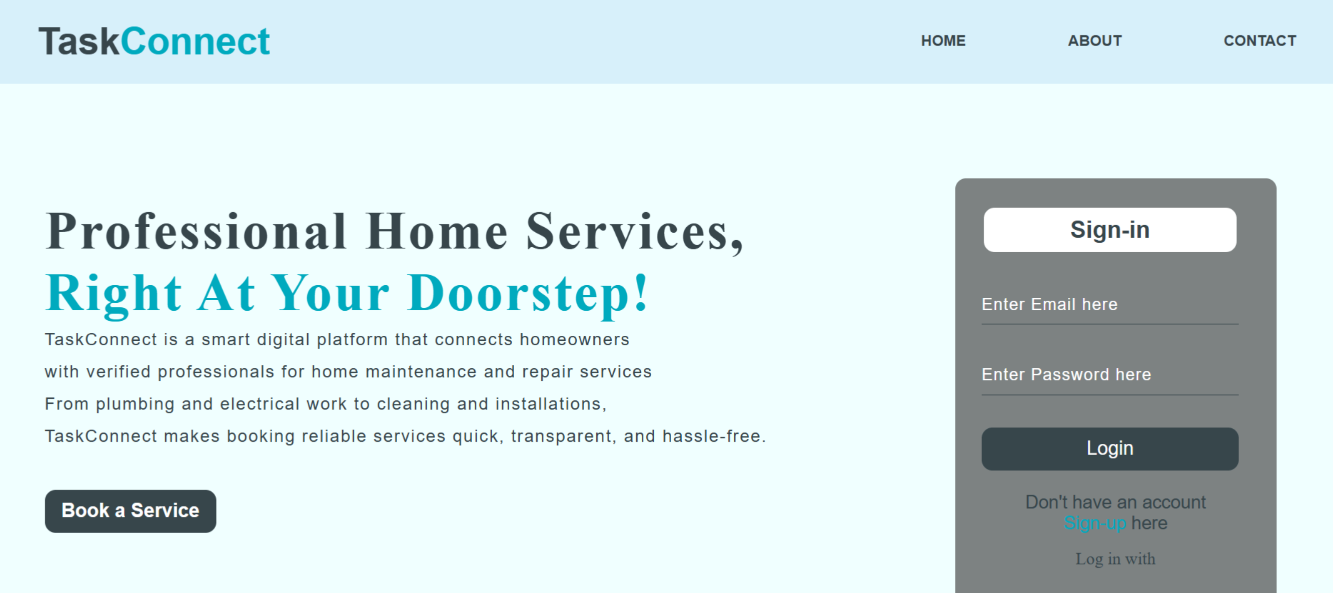
1) Book a Service



2) Rate Service Provider



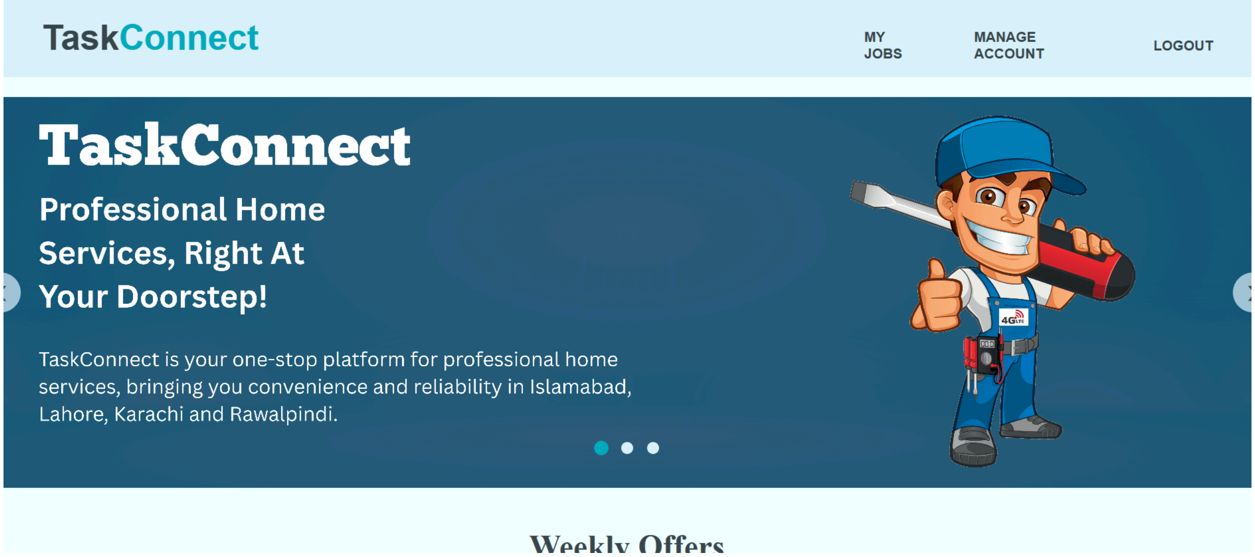
11. Implementation Screenshots



Professional Home Services,
Right At Your Doorstep!

TaskConnect is a smart digital platform that connects homeowners
with verified professionals for home maintenance and repair services
From plumbing and electrical work to cleaning and installations,
TaskConnect makes booking reliable services quick, transparent, and hassle-free.

Don't have an account
Sign-up here

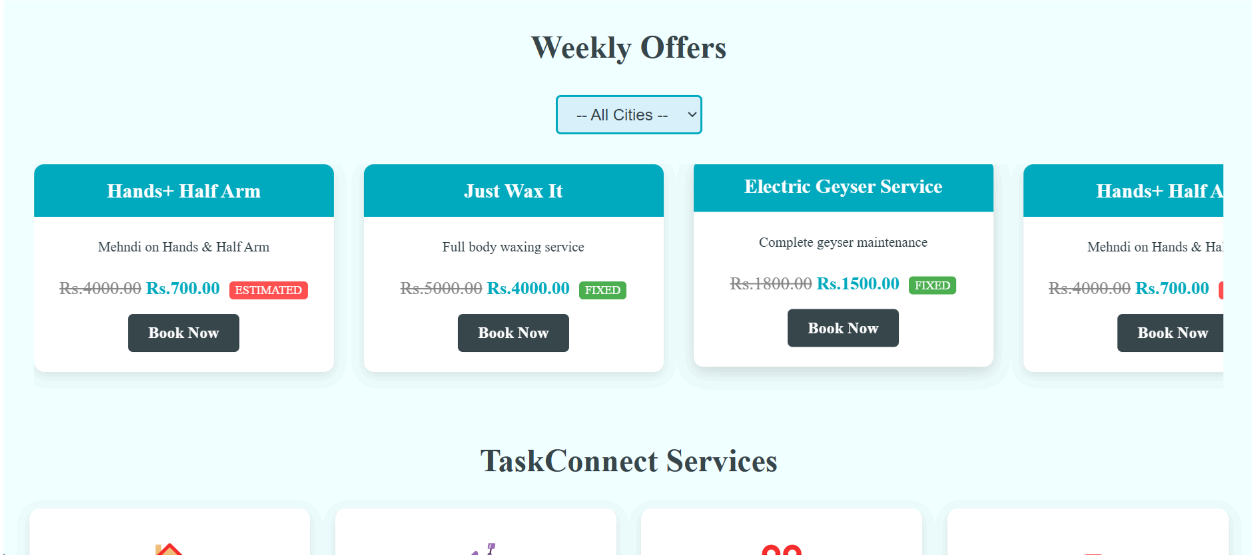


TaskConnect

Professional Home Services, Right At Your Doorstep!

TaskConnect is your one-stop platform for professional home services, bringing you convenience and reliability in Islamabad, Lahore, Karachi and Rawalpindi.

Weekly Offers



Weekly Offers

-- All Cities --

Hands+ Half Arm

Mehndi on Hands & Half Arm

~~Rs.4000.00~~ Rs.700.00 ESTIMATED

Book Now

Just Wax It

Full body waxing service

~~Rs.5000.00~~ Rs.4000.00 FIXED

Book Now

Electric Geyser Service

Complete geyser maintenance

~~Rs.1800.00~~ Rs.1500.00 FIXED

Book Now

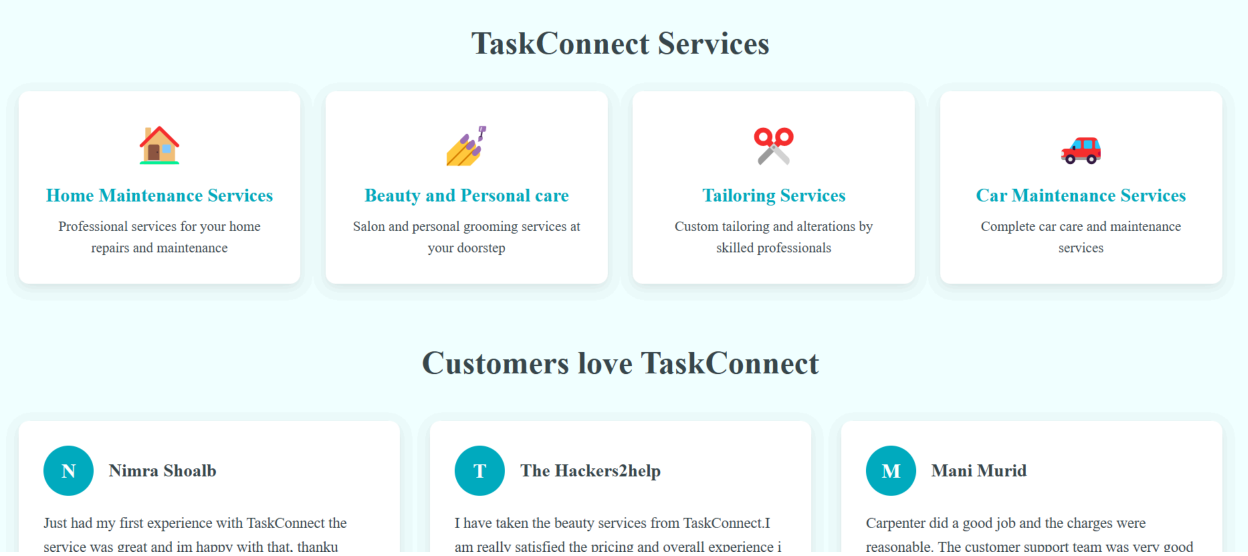
Hands+ Half A

Mehndi on Hands & Ha

~~Rs.4000.00~~ Rs.700.00

Book Now

TaskConnect Services



TaskConnect Services

🏠 **Home Maintenance Services**

Professional services for your home repairs and maintenance

💅 **Beauty and Personal care**

Salon and personal grooming services at your doorstep

✂ **Tailoring Services**

Custom tailoring and alterations by skilled professionals

🚗 **Car Maintenance Services**

Complete car care and maintenance services

Customers love TaskConnect

N **Nimra Shoaib**

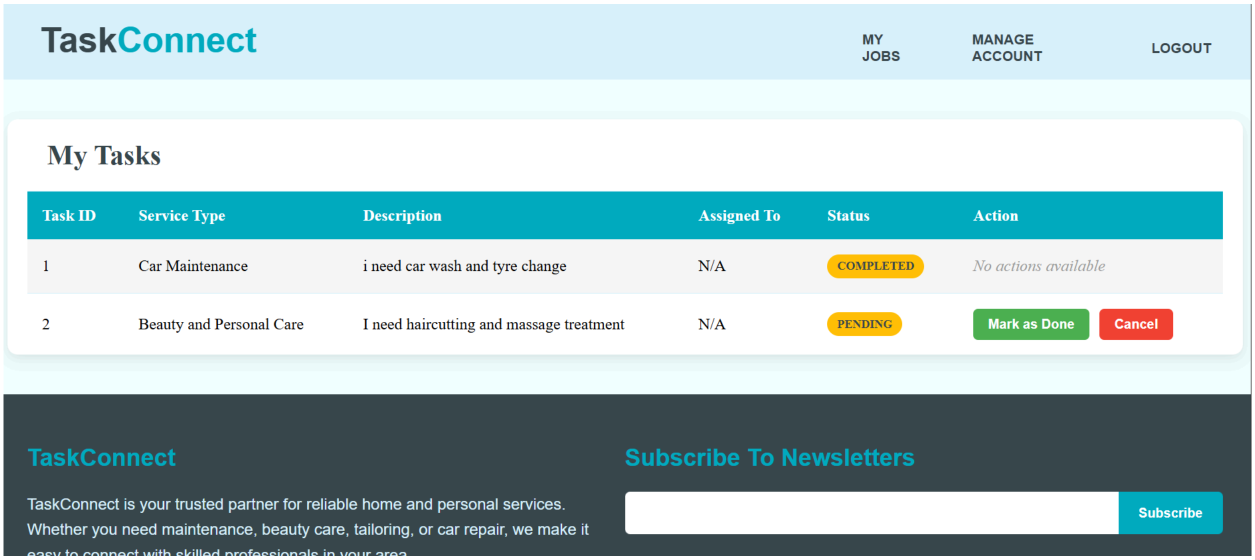
Just had my first experience with TaskConnect the service was great and im happy with that. thanku

T **The Hackers2help**

I have taken the beauty services from TaskConnect.I am really satisfied the pricing and overall experience i

M **Mani Murid**

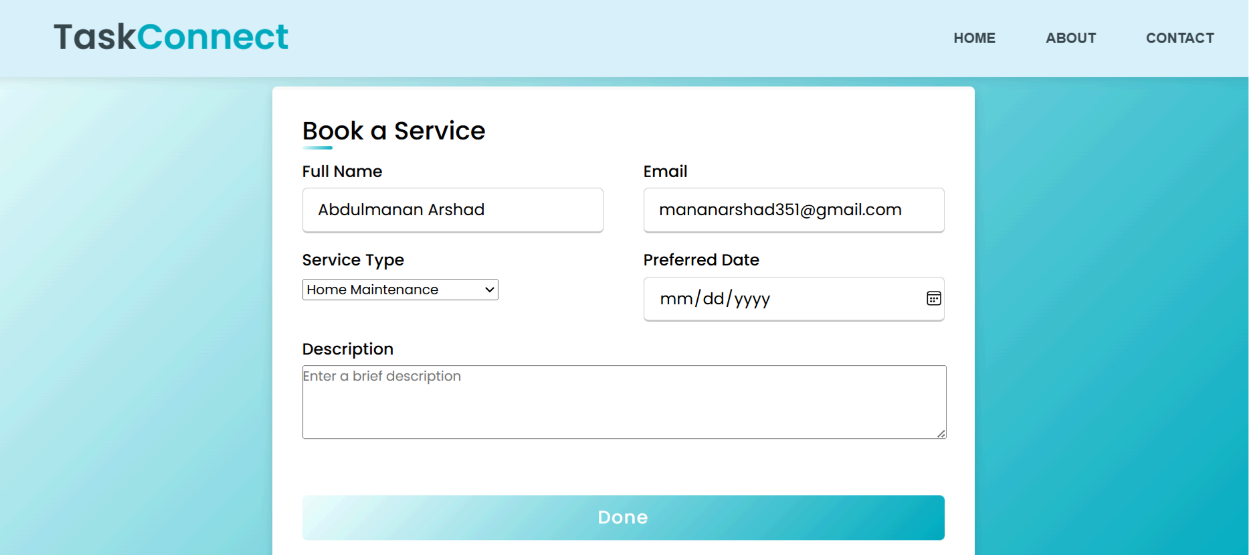
Carpenter did a good job and the charges were reasonable. The customer support team was very good



My Tasks

Task ID	Service Type	Description	Assigned To	Status	Action
1	Car Maintenance	i need car wash and tyre change	N/A	COMPLETED	*No actions available*
2	Beauty and Personal Care	I need haircutting and massage treatment	N/A	PENDING	Mark as Done Cancel

TaskConnect is your trusted partner for reliable home and personal services. Whether you need maintenance, beauty care, tailoring, or car repair, we make it easy to connect with skilled professionals in your area.



Book a Service

Full Name: Abdulmanan Arshad

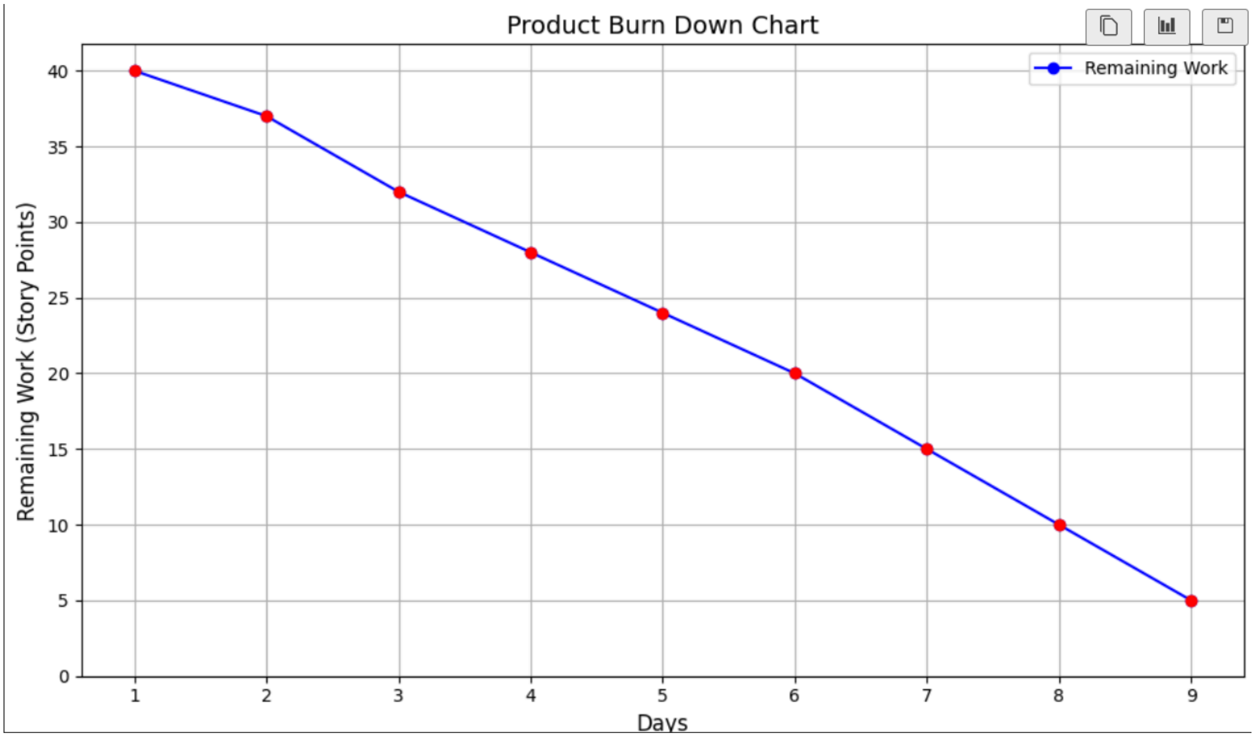
Email: mananarshad351@gmail.com

Service Type: Home Maintenance

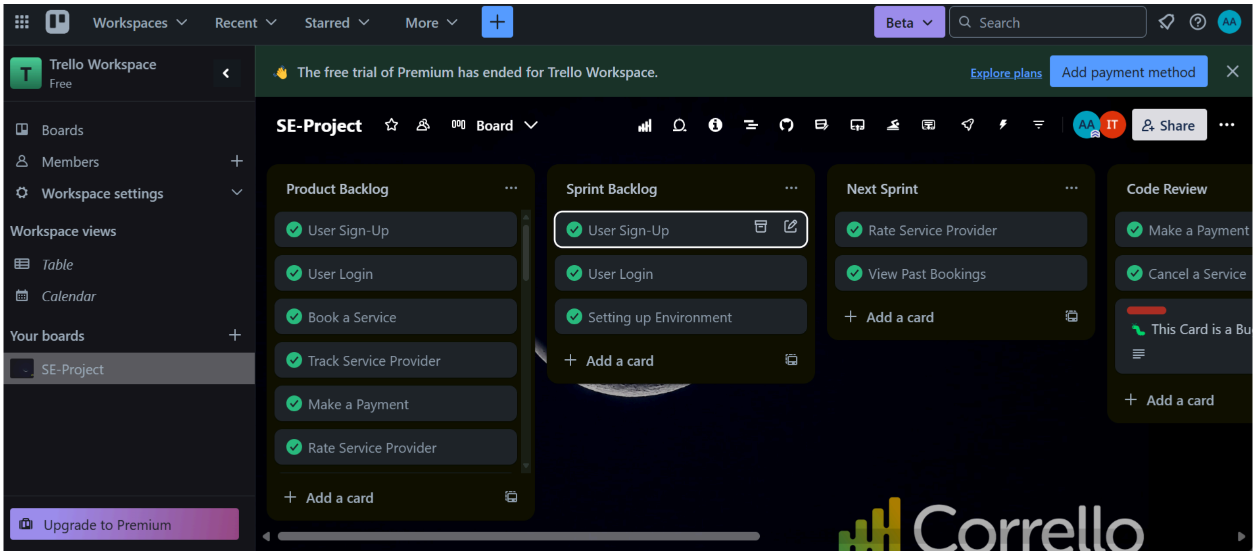
Preferred Date: mm/dd/yyyy

Description: Enter a brief description

12. Product Burndown chart



13. Trello board screenshots



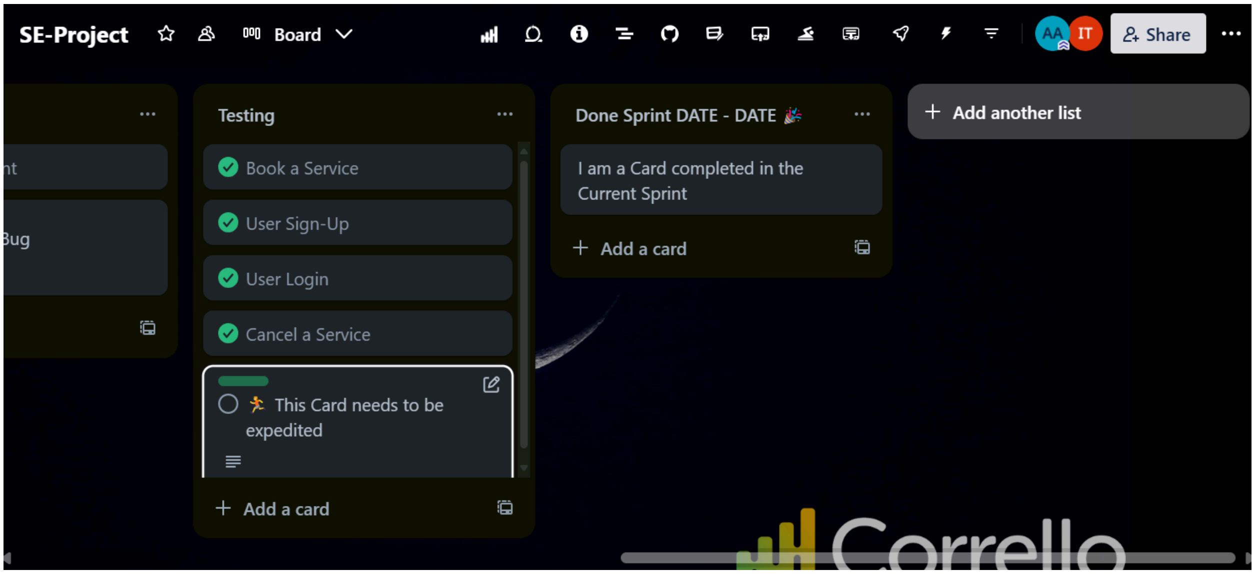
👋 The free trial of Premium has ended for Trello Workspace.

Product Backlog
- User Sign-Up
- User Login
- Book a Service
- Track Service Provider
- Make a Payment
- Rate Service Provider
- + Add a card

Sprint Backlog
- User Sign-Up
- User Login
- Setting up Environment
- + Add a card

Next Sprint
- Rate Service Provider
- View Past Bookings
- + Add a card

Code Review
- Make a Payment
- Cancel a Service
- This Card is a Bu
- + Add a card



14. Test Cases -Black box

1. Equivalence Class Partitioning (10 Marks)
Input: User registration (email, password, role).

Input Field	Valid Classes	Invalid Classes
Email	Valid format (a@b.com)	Missing @ (ab.com), Empty
Password	Length ≥ 8, has special chars	Length < 8, no special chars
Role	"client", "worker"	"admin", empty

Test Cases:

TC#	Input (email, password, role)	Expected Result	Class Tested
1	a@b.com, Pass@123, client	Success	Valid all fields
2	ab.com, Pass@123, client	Fail (invalid email)	Invalid email
3	a@b.com, pass, client	Fail (short password)	Invalid password
4	a@b.com, Pass@123, admin	Fail (invalid role)	Invalid role

2. Boundary Value Analysis (10 Marks)
Input: Booking price (valid range:
10
–
10–1000).

TC#	Input (price)	Expected Result	Boundary Tested
5	10	Success	Lower boundary
6	1000	Success	Upper boundary
7	9	Fail	Below lower bound
8	1001	Fail	Above upper bound

3. User Story Validation
User Story: "As a user, I want to book a service so that I can hire workers."

TC#	Action	Expected Result
9	Submit booking with valid details	Booking created, status="pending"
10	Submit booking with invalid price	Error message, no booking

15. Test Cases -White box

1. Unit Testing (JUnit Example)
Test Class: BookingServiceTest.java

java

```
@Test
public void testSaveBooking_ValidPrice() {
    Booking booking = new Booking();
    booking.setPrice(50.0); // Valid price
    when(bookingRepository.save(booking)).thenReturn(booking);
    Booking savedBooking = bookingService.saveBooking(booking);
    assertEquals(50.0, savedBooking.getPrice());
}

@Test
public void testSaveBooking_InvalidPrice() {
    Booking booking = new Booking();
    booking.setPrice(5.0); // Invalid price
    assertThrows(IllegalArgumentException.class, () -> {
        bookingService.saveBooking(booking);
    });
}
```

16. Work Division between group members

Team Member	**Role**	**Responsibilities**
Imra Tariq	Scrum Master	- Project planning and coordination - Backend development using Spring Boot - Database schema design and implementation (MySQL) - Writing and optimizing SQL queries - Unit and integration

		testing - Managing sprints and tracking progress
Abdulmanan	Developer	- Front End development using HTML, CSS, and Thymeleaf - Implementing Java model and controller classes - UI/UX design and validation- Supporting integration with backend APIs - Writing and executing test cases

17. Lesson learnt by group

During the development of **TaskConnect**, our team gained valuable insights into both technical and collaborative aspects of building a real-world web application. We learned the importance of planning tasks clearly within sprints, maintaining regular communication, and integrating frontend and backend components early to avoid delays. Working with Spring Boot, MVC architecture, Thymeleaf, and MySQL gave us hands-on experience with full-stack development, while implementing testing strategies like equivalence partitioning and boundary value analysis helped us improve quality. Deploying the application to the cloud introduced us to real-world challenges like environment configuration and service integration. Overall, the project strengthened our teamwork, problem-solving skills, and understanding of building scalable, user-focused applications.

G- Powerpoint presentation [10 Marks]

a) Project Introduction

b) Functional Requirements

c) Nonfunctional requirements (2 NFR)

d) Design - Class Diagram

e) Architecture Diagram

f) Actual implementation screenshots

g) Work Division between group members

h) Lesson learnt