## 1. Document Control

- **Document Title**: Home Service Platform—Product Requirement Document
- **Version**: 1.0
- **Date**: October 10, 2024
- **Prepared by**: []
- **Reviewed by**: []
- **Approved by**: []
- **Task Board Link**: https://trello.com/b/JdAlrqOL/home-service

---

## 2. Executive Summary

- **Overview**: The Home Service Platform is a web and mobile application that connects users with verified technicians who provide home maintenance services like AC repair, plumbing, electrical work, and more. The platform will offer users the convenience of booking qualified technicians for on-demand home services while giving service providers an efficient way to connect with customers and manage their jobs.
- **Goals and Objectives**:
    - Provide a seamless user experience for customers to book home services.
    - Ensure timely service delivery through a real-time booking system.
    - Enable technicians to manage bookings, payments, and user reviews.
    - Create a reliable and trustworthy service ecosystem.
- **Success Criteria**:
    - 1,000+ registered users within the first 3 months.
    - Successful booking completion rate of over 85%.
    - Average customer satisfaction rating of 4.5/5.
    - Increase the number of active technicians by 20% every quarter.

---

## 3. Scope

- **In-Scope**:
    - User registration, profile management, and service booking.
    - Technician onboarding and verification.
    - Real-time notifications, GPS tracking, and service status updates.
    - Offline payments (only offline payments supported initially).
    - Review and rating system for users and technicians.
    - Admin dashboard for managing users, services, and technicians.
- **Out-of-Scope**:
    - In-app messaging between users and technicians (for this version).
    - Payment gateway integration for online transactions.

- International support (initial launch in a single city/region).

---

## 4. Stakeholders

- **Product Owner**:
- **Project Manager**:
- **Development Team**: Web developers, mobile app developers, backend developers, UI/UX designers.
- **End Users**: Homeowners, renters, technicians (AC repair, plumbing, electricians, etc.)

---

## 5. Functional Requirements

### 5.1. User Stories and Use Cases

- **User Story 1**: As a homeowner, I want to browse and book technicians for home services like AC repair to resolve my maintenance issues quickly.
  - **Acceptance Criteria**:
    1. Users should be able to search for technicians based on service type and location.
    2. Users should see a list of available technicians with ratings and reviews.
    3. Users can schedule a service at a specific time and date.
- **User Story 2**: As a technician, I want to receive job requests and manage my schedule on the platform so that I can grow my business.
  - **Acceptance Criteria**:
    1. Technicians should receive notifications for new service requests.
    2. They should be able to accept or decline jobs.
    3. They can manage their service availability and view earnings.

### 5.2. System Features

- **Feature 1**: **User Registration & Authentication**
  - **Description**: Users and technicians must be able to sign up and log in using email, phone, or social media accounts.
  - **Priority**: High
  - **Dependencies**: user profile and verification processes.
  - **Assumptions**: Authentication will support password reset and account verification.
- **Feature 2**: **Technician Verification**
  - **Description**: Technicians must be verified via ID and certification documents before being able to accept jobs.
  - **Priority**: High

    - **Dependencies**: Admin approval system.
    - **Assumptions**: Documents will be manually reviewed by the admin team.
- **Feature 3**: **Service Booking and Scheduling**
    - **Description**: Users can book a service by selecting the type of job, technician, and preferred time.
    - **Priority**: High
    - **Dependencies**: Technician availability.
    - **Assumptions**: Technicians have real-time availability updates.
- **Feature 4**: **Payment Gateway Integration**
    - **Description**: Secure payment processing for service bookings via credit card or digital wallets.
    - **Priority**: High
    - **Dependencies**: Payment provider APIs
    - **Assumptions**: The payment system will comply with security standards.
- **Feature 5**: **Rating and Review System**
    - **Description**: Users can rate technicians, and technicians can view their feedback.
    - **Priority**: Medium
    - **Dependencies**: Completed service tracking.
    - **Assumptions**: Only users who completed a service can leave a review.

## Detailed Features

### A. User Registration and Authentication

1. **User Registration**:
    - Users (both customers and technicians) can register via email, phone number, or third-party services (Google, Facebook, etc.).
    - For technicians, an additional verification step (ID and skill certification upload) is required.
2. **User Login**:
    - Users can log in using email, phone number, or third-party services.
    - Implement 2-factor authentication (optional) for additional security.
3. **Password Recovery**:
    - Provide password reset via email or SMS.
    - Technicians and users can request a password reset link.

---

### B. Profile Management

1. **Customer Profile**:

    - Customers can update their personal details (name, phone number, email, address).
    - Manage saved addresses for service requests (e.g., home, office).
2. **Technician Profile**:
    - Technicians can add/edit personal information, service details (category of service, hourly rate, years of experience).
    - Upload identification, certifications, and images of previous work.
    - Set availability (hours/days they can accept service requests).

---

**C. Search and Service Discovery**

1. **Service Search**:
    - Users can search for available services by categories (AC repair, plumbing, electrical work, etc.).
    - Search can be filtered by technician rating, price, and location.
    - Users can view technicians on a map or in a list format.
2. **Location-Based Service Suggestions**:
    - Use GPS or location data to suggest nearby technicians based on user location.
    - Provide estimated service arrival time based on technician proximity.

---

**D. Booking and Scheduling**

1. **Service Request**:
    - Users can request a service by selecting the desired category, date, and time.
    - Option to select a preferred technician or allow the system to assign one based on availability.
2. **Real-Time Technician Availability**:
    - Show available time slots based on the technician's calendar.
    - Users receive instant confirmation upon booking a service.
3. **Booking Modifications**:
    - Allow users to reschedule or cancel bookings with predefined rules (e.g., no penalty if canceled 24 hours before the service).
4. **Booking Notifications**:
    - Send automated notifications via SMS, email, or app notifications for service confirmation, reminders, and updates.

---

**E. Payment System**

1. **Payment Options**:

    - Support multiple payment methods: credit card, debit card, PayPal, Stripe, or cash on delivery.
    - Option to add and save multiple payment methods.
    - Users can view payment history in their profile.
2. **Payment Split for Technicians**:
    - Technicians receive payment after service completion minus the platform's commission (if applicable).
    - Implement automatic payouts to technicians' linked bank accounts or wallets.
3. **Invoice Generation**:
    - Automatically generate invoices upon service completion for customers, sent via email and available in the app.

---

**F. Real-Time GPS Tracking**

1. **Technician Location Tracking**:
    - After a booking is confirmed, users can track the technician's real-time location via GPS.
    - Display estimated arrival time on the map.
2. **Route Optimization for Technicians**:
    - Technicians can get route suggestions for optimal travel to the customer's location using integrated map services.

---

**G. Service Management for Technicians**

1. **Service Status Updates**:
    - Technicians can mark services as "Accepted", "In Progress", "Completed", or "Cancelled".
    - Customers are notified of these status changes.
2. **Availability Management**:
    - Technicians can update their service availability (e.g., holidays, weekends).
    - Option to mark themselves as "Busy" or "Available" in real-time.
3. **Service History**:
    - Technicians can view their previous service jobs, earnings, and ratings.

---

**H. Rating and Review System**

1. **Post-Service Feedback**:
    - Customers can rate technicians on a scale of 1-5 stars and leave detailed feedback.
    - Technicians can also rate customers based on their experience (optional).

2. **Review Moderation**:
    - Admins can moderate and manage reviews, including reporting and blocking inappropriate content.
3. **Average Rating Display**:
    - Display technician ratings and reviews on their profile, affecting their ranking in search results.

---

**I. Admin Dashboard**

1. **User Management**:
    - Admins can view, edit, or deactivate user (customer and technician) accounts.
    - Handle technician verification and document approval.
2. **Service Management**:
    - View active, pending, and completed services.
    - Manage service categories and pricing recommendations.
3. **Payment Management**:
    - Track and manage platform transactions, technician payouts, and commissions.
    - Generate financial reports.
4. **Dispute Resolution**:
    - Admins can manage disputes between customers and technicians, including issuing refunds or partial credits.
5. **Analytics and Reports**:
    - View platform analytics (e.g., number of users, services booked, most popular services).
    - Generate weekly/monthly reports on platform performance.

---

**J. Notifications System**

1. **Push Notifications**:
    - Notify users of booking confirmations, reminders, and service updates.
    - Send notifications for special promotions or offers.
2. **SMS/Email Notifications**:
    - Send booking confirmations, payment receipts, and reminders via SMS or email.
3. **In-App Messaging**:
    - Allow customers and technicians to communicate within the app after booking a service.
    - Notifications for new messages and updates within the chat.

---

**K. Platform Security**

1. **Data Encryption**:
    - Encrypt sensitive user data, including payment information and personal details.
2. **Role-Based Access Control**:
    - Implement role-based access to ensure only admins have access to critical management features.
3. **Activity Logs**:
    - Track user actions (logins, profile updates, bookings, etc.) to provide an audit trail for security and troubleshooting.

---

**L. Promotions and Discounts**

1. **Promo Codes**:
    - Users can apply promo codes during checkout for discounts.
    - Admins can create and manage promotional offers through the admin dashboard.
2. **Referral Program**:
    - Users can refer others to the platform and receive discounts or credits.

---

## 6. Non-Functional Requirements

### 6.1. Performance

- **Response Time**: The platform should respond to user actions (e.g., search, booking) within 2 seconds.
- **Load capacity**: must support up to 10,000 concurrent users during peak times.
- **Scalability**: Designed to scale to multiple cities and countries in the future.

### 6.2. Security

- **Authentication**: Secure login with OAuth or JWT.
- **Data Protection**: Encrypt all personal and payment data (SSL).
- **Auditing and Logging**: Log all user and technician activities for compliance and dispute resolution.

### 6.3. Usability

- **UI Guidelines**: Design must follow material design principles for web and mobile.
- **UX Principles**: Ensure simple and intuitive user flows, especially during booking and payment.

### 6.4. Reliability

- **Availability**: 99.9% uptime required.
- **Disaster Recovery**: Daily backups and automated recovery processes.

### 6.5. Maintainability

- **Code Modularity**: Ensure the application is built in a modular manner for easy updates and maintenance.
- **Documentation**: Provide comprehensive documentation for both frontend and backend.

---

## 7. UI/UX Design Requirements

- **Wireframes**: Attached wireframes for the user booking process and technician profile page.
- **Mockups**: Provide high-fidelity mockups showing the key flows (search, booking, payment, service confirmation).
- **Design Principles**: responsive design, intuitive navigation, and accessibility compliance (WCAG 2.1).
- **Navigation Flow**:  user can move from home screen to service search, booking, and confirmation in no more than 4 steps.

---

## 8. Technical Requirements

- **Platform**:
    - Web: Supported browsers include Chrome, Firefox, Safari, and Edge.
    - Mobile: Android and iOS support via a cross-platform framework ( Flutter).
- **Technology Stack**:
    - Frontend: React for web, Flutter for mobile.
    - Backend: Spring boot
    - Database:  MySQL.
- **API Integration**:
    - Payment APIs (e.g., Chapa, telebirr).
    - SMS or Email notification service (e.g.,EthioTel).
    - GPS tracking for technicians.

---

## 9. Assumptions and Dependencies

- **Assumptions**:
    - Technicians will update their availability regularly.
    - Payment integration will follow local financial regulations.

- **Dependencies**:
    - Integration with third-party payment gateways and notification services.
    - Reliable internet access for real-time booking and tracking.

---

## 10. Risks

- **Technician Availability**: Technicians might not always be available, leading to delays or cancellations.
- **Payment Issues**: Payment gateway failures may disrupt the booking process.
- **User Trust**: Ensuring that technicians are trustworthy and verified can impact user retention.

---

## 11. Timeline and Milestones

- **Phase 1**: Prototype (Wireframes and Design) – 2 weeks
- **Phase 2**: Backend and Frontend Development – 5 weeks
- **Phase 3**: Testing (UAT, performance) – 1 weeks
- **Phase 4**: Beta Launch
- **Phase 5**: Full Launch – After successful Beta phase

---

## 12. Testing and Validation

- **Test Plan**: Conduct unit testing for all components and integration testing for booking and payment.
- **User Acceptance Testing (UAT)**: Gather feedback from selected users and technicians.
- **Performance Testing**: Simulate peak load to ensure stability.
- **Security Testing**: Perform vulnerability scans and penetration testing before launch.

---

## 13. Post-launch Support and Maintenance

- **Support Plan**: 24/7 customer support via in-app chat and email.
- **Bug Fixes**: Regular updates and bug fixes every 2 weeks after launch.
- **Product Updates**: Roll out new features based on user feedback quarterly.

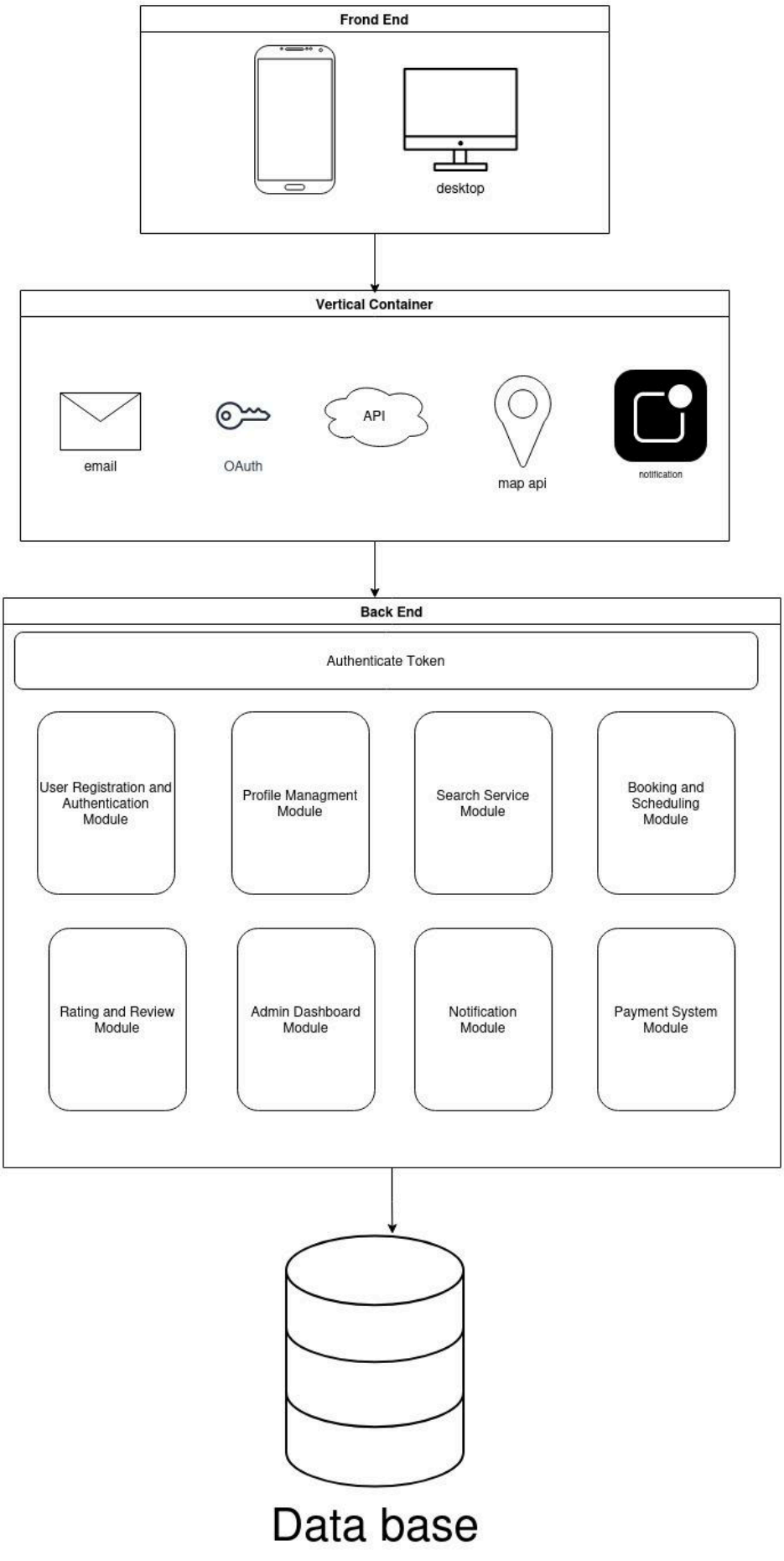
---

## 14. Glossary

- **OAuth**: Open standard for access delegation, commonly used for token-based authentication.
- **UAT**: User Acceptance Testing, a phase where users test the product before release.
- **GPS**: Global Positioning System, used for location tracking.

---

## System Design


Frond End
desktop
Vertical Container
email
OAuth
API
map api
notification
Back End
Authenticate Token
User Registration and Authentication Module
Profile Managment Module
Search Service Module
Booking and Scheduling Module
Rating and Review Module
Admin Dashboard Module
Notification Module
Payment System Module
Data base

## User Registration & Authentication

### 1. Customer Registration

- **Endpoint:** `POST /auth/customer/signup`
- **Request Parameters:**
  - Name: Customer's full name
  - Email: Customer's email address
  - Phone Number: Customer's phone number
  - Password: Customer's password
- **Response:**
  - Token: JWT Token for authentication
  - User: User object containing customer details

---

### 2. Customer Login

- **Endpoint:** `POST /auth/customer/login`
- **Request Parameters:**
  - Email: Customer's email address
  - Password: Customer's password
- **Response:**
  - Token: JWT Token for authentication
  - User: User object containing customer details

---

### 3. Technician Registration (Online)

- **Endpoint:** `POST /auth/technician/signup`
- **Request Parameters:**
  - Name: Technician's full name
  - Email: Technician's email address
  - Phone Number: Technician's phone number
  - CV: Technician's CV as a file
  - ID Card: Technician's ID card as a file
  - Services: List of services the technician provides
  - Address: Technician's address information
  - Password: Technician's chosen password

    - Image: Technician's profile picture as a file
- **Response:**
    - Token: JWT Token for authentication
    - User: User object containing technician details

---

**4. Technician Registration (In-Person)**

- **Endpoint:** `POST /auth/technician/signup/in-person`
- **Request Parameters:**
    - Name: Technician's full name
    - Email: Technician's email address
    - Phone Number: Technician's phone number
    - CV: Technician's CV as a file
    - ID Card: Technician's ID card as a file
    - Services: List of services the technician provides
    - Address: Technician's address information
    - Image: Technician's profile picture as a file
- **Response:**
    - Password: Auto-generated password for technician's account
    - User: User object containing technician details

---

**5. Technician Pays for Registration**

- **Endpoint:** `POST /auth/technician/pay`
- **Request Parameters:**
    - Payment Method: Technician's payment information
- **Response:**
    - Token: JWT Token for authentication
    - User: User object containing technician details

---

**6. Technician Login**

- **Endpoint:** `POST /auth/technician/login`
- **Request Parameters:**
    - Email: Technician's email address
    - Password: Technician's password
- **Response:**
    - Token: JWT Token for authentication
    - User: User object containing technician details

---

**7. Operator Registration (In-Person)**

- **Endpoint:** `POST /auth/operator/signup`
- **Request Parameters:**
  - Name: Operator's full name
  - Email: Operator's email address
  - Phone Number: Operator's phone number
  - CV: Operator's CV as a file
  - ID Card: Operator's ID card as a file
  - Services: List of services the operator manages
  - Address: Operator's address information
- **Response:**
  - Password: Auto-generated password for the operator
  - Username: Operator's username

---

**8. Admin Login**

- **Endpoint:** `POST /admin/auth`
- **Request Parameters:**
  - Username: Admin's username
  - Password: Admin's password
- **Response:**
  - Token: JWT Token for authentication
  - User: Admin object containing admin details

---

## Password Recovery

**1. Customer & Technician Password Recovery**

- **Step 1: Send Recovery Email**
  - **Endpoint:** `POST /auth/user/email`
  - **Request Parameters:**
    - Email: User's email address
  - **Response:**
    - Message confirming that a recovery email has been sent.
- **Step 2: Validate Recovery Code**
  - **Endpoint:** `POST /auth/user/code`
  - **Request Parameters:**
    - Code: Recovery code received in the email

    - **Response:**
        - Message confirming that the recovery code is valid.
- **Step 3: Set New Password**
    - **Endpoint:** `POST /auth/user/newpassword`
    - **Request Parameters:**
        - Password: New password
    - **Response:**
        - Token: JWT Token for authentication
        - User: User object containing user details

---

## Profile Management

### 1. Customer Profile

- **Endpoint:** `GET /user/customer/profile`
- **Response:**
    - User: User object containing customer details
    - History: List of previous bookings
    - Pending: List of pending bookings
    - Rating: Customer's overall rating

### 2. Technician Profile

- **Endpoint:** `GET /user/technician/profile`
- **Response:**
    - User: User object containing technician details
    - History: List of previous bookings
    - Rating: Technician's average rating
    - Address: Technician's address details
    - Services: List of services provided by the technician
    - Schedule: Technician's available schedule

---

## Service Search

### 1. Search for Service

- **Endpoint:** `GET /search/service`
- **Request Parameters:**
    - Service: Service object with search parameters
- **Response:**
    - Technicians: List of available technicians matching the search criteria

**2. Search for Technician**

- **Endpoint:** `GET /search/tech`
- **Request Parameters:**
  - Query: Search keyword
  - Price: Price range
  - Rating: Technician's rating
  - Location: Location filter
- **Response:**
  - Technicians: List of available technicians matching the search criteria

---

## Booking Management

**1. Request Booking**

- **Endpoint:** `POST /booking/request`
- **Request Parameters:**
  - Customer ID: ID of the customer making the booking
  - Technician ID: ID of the technician being booked
  - Service ID: ID of the service requested
  - Description: Description of the requested service
- **Response:**
  - Status: 201 Created
  - Message confirming the booking request

**2. Cancel Booking**

- **Endpoint:** `POST /booking/cancel`
- **Request Parameters:**
  - Booking ID: ID of the booking to be canceled
- **Response:**
  - Message confirming the cancellation

**3. Accept/Reject Booking**

- **Endpoint:** `POST /booking/response/accept` or `POST /booking/response/deny`
- **Request Parameters:**
  - Booking ID: ID of the booking
- **Response:**
  - Status: 201 Created
  - Message confirming the booking response

#### 4. Complete Booking

- **Endpoint:** `POST /booking/completion`
- **Request Parameters:**
  - Booking ID: ID of the completed booking
- **Response:**
  - Message confirming the booking completion

---

## Rating & Review

#### 1. Submit Rating & Review

- **Endpoint:** `POST /review`
- **Request Parameters:**
  - Booking ID: ID of the booking
  - Technician ID: ID of the technician
  - Customer ID: ID of the customer
  - Rating: Rating (1-5)
  - Review: Written review
- **Response:**
  - Message confirming the review submission

---

## Admin Dashboard

#### 1. Manage Entities (CRUD Operations)

- **Endpoint:** Admin has CRUD operations for:
  - Customers
  - Technicians
  - Operators
  - Services

#### 2. Dispute Resolution

- **Endpoint (Customer Dispute):** `POST /user/customer/dispute`
- **Request Parameters:**
  - Technician ID: ID of the technician
  - Customer ID: ID of the customer
  - Title: Title of the dispute
  - Description: Detailed description of the dispute
- **Endpoint (Technician Dispute):** `POST /user/technician/dispute`

- **Request Parameters:**
    - Technician ID: ID of the technician
    - Customer ID: ID of the customer
    - Title: Title of the dispute
    - Description: Detailed description of the dispute
- **Endpoint (Admin Dispute Resolution):** `GET /admin/dispute`
- **Response:**
    - Customer: Object containing customer details
    - Technician: Object containing technician details
    - Dispute: Object containing dispute details

---

## Security Considerations

1. **JWT Token Expiry:** Ensure that tokens have an expiration time, and implement token refreshing mechanisms.
2. **Encryption:** Passwords and sensitive data should be encrypted before storing them in the database.
3. **File Upload Validation:** Uploaded files (e.g., CVs, ID cards) should be validated to prevent malicious file uploads.
4. **Role-based Access Control (RBAC):** Access to certain endpoints must be restricted based on user roles (Customer, Technician, Operator, Admin).

## Entities and Attributes

### 1. User (Base Entity)

- `user_id` (PK): Unique identifier for each user.
- `name`: Full name of the user.
- `email`: Email address.
- `phone_number`: Phone number.
- `password`: Encrypted password.
- `role`: Role of the user (e.g., Customer, Technician, Operator, Admin).
- `profile_image`: URL to the profile picture.
- `created_at`: Timestamp when the user account was created.
- `updated_at`: Timestamp of the last update.
- `status`: Account status (active, suspended, etc.).

### 2. Address

- `address_id` (PK): Unique identifier for each address.
- `user_id` (FK): Foreign key referencing `User.user_id`.
- `street`: Street address.
- `city`: City.
- `state`: State/Province.
- `country`: Country.
- `zip_code`: Postal code.
- `latitude`: Geolocation latitude.
- `longitude`: Geolocation longitude.
- `created_at`: Timestamp when the address was added.

**3. Payment Method**

- `payment_method_id` (PK): Unique identifier for each payment method.
- `user_id` (FK): Foreign key referencing `User.user_id`.
- `card_number`: Masked card number.
- `cardholder_name`: Name on the card.
- `expiry_date`: Expiration date of the card.
- `billing_address_id` (FK): Foreign key referencing `Address.address_id` for the billing address.
- `payment_type`: Type of payment (credit card, PayPal, etc.).
- `is_default`: Indicates whether this is the default payment method for the user.

**4. Customer**

- `user_id` (FK): Foreign key referencing `User.user_id`.
- `service_history`: List of services previously booked.
- `saved_addresses`: List of additional saved addresses (linked to the Address entity).

**5. Technician**

- `user_id` (FK): Foreign key referencing `User.user_id`.
- `services`: The type of service(s) the technician offers.
- `bio`: Technician's short description.
- `location`: Technician’s operating location.
- `availability`: Days and hours when the technician is available.
- `rating`: Average customer rating.
- `completed_jobs`: Number of jobs completed.
- `documents`: URLs to uploaded documents (e.g., certifications).
- `ID_card_image`: Image for Technician’s Identification card.

**6. Operator**

- `user_id` (FK): Foreign key referencing `User.user_id`.
- `assigned_region`: The region the operator manages.
- `ID_card_image`: Image for Technician's Identification card.

**7. Admin**

- `user_id` (FK): Foreign key referencing `User.user_id`.
- `permissions`: List of allowed actions.

**8. Service**

Represents the various services offered on the platform, including the newly requested ones.

- `service_id` (PK): Unique identifier for each service.
- `category_id` (FK): Foreign key referencing `ServiceCategory.category_id`.
- `description`: Detailed description of the service.
- `estimated_duration`: Estimated time to complete the service.
- `service_fee`: Base cost for the service.
- `created_at`: Timestamp when the service was added.
- `updated_at`: Timestamp of the last update.

**9. Service Category**

List of service categories, which include the requested ones.

- `category_id` (PK): Unique identifier for each category.
- `category_name`: Name of the category (e.g., "AC Repair & Service", "Plumbing").
- `description`: Description of the category.

**10. Booking**

Represents a service booking made by a customer with a technician.

- `booking_id` (PK): Unique identifier for each booking.
- `customer_id` (FK): Foreign key referencing `Customer.user_id`.
- `technician_id` (FK): Foreign key referencing `Technician.user_id`.
- `service_id` (FK): Foreign key referencing `Service.service_id`.
- `scheduled_date`: Date and time for the service.
- `status`: Status of the booking (pending, confirmed, completed, canceled).
- `service_location_id` (FK): Foreign key referencing `Address.address_id` for the service location.

- `total_cost`: Total cost of the service.

**11. Payment**

- `payment_id` (PK): Unique identifier for each payment.
- `booking_id` (FK): Foreign key referencing `Booking.booking_id`.
- `customer_id` (FK): Foreign key referencing `Customer.user_id`.
- `technician_id` (FK): Foreign key referencing `Technician.user_id`.
- `payment_method_id` (FK): Foreign key referencing `PaymentMethod.payment_method_id`.
- `amount`: Total payment amount.
- `commission_fee`: Platform's commission.
- `payment_status`: Status of the payment (pending, completed).
- `payment_date`: Date the payment was made.

**12. Review**

- `review_id` (PK): Unique identifier for each review.
- `booking_id` (FK): Foreign key referencing `Booking.booking_id`.
- `customer_id` (FK): Foreign key referencing `Customer.user_id`.
- `technician_id` (FK): Foreign key referencing `Technician.user_id`.
- `rating`: Rating (1-5 stars).
- `review_text`: Feedback provided by the customer.

**13. Notification**

- `notification_id` (PK): Unique identifier for each notification.
- `user_id` (FK): Foreign key referencing `User.user_id`.
- `message`: Notification content.
- `sent_at`: Timestamp when the notification was sent.
- `status`: Status (sent, delivered).

**14. Promotion**

- `promotion_id` (PK): Unique identifier for each promotion.
- `promo_code`: Code used to claim the promotion.
- `discount_percentage`: Discount offered.
- `start_date`: Promotion start date.
- `end_date`: Promotion end date.

**15. Dispute**

- `dispute_id` (PK): Unique identifier for each dispute.
- `booking_id` (FK): Foreign key referencing `Booking.booking_id`.
- `customer_id` (FK): Foreign key referencing `Customer.user_id`.
- `technician_id` (FK): Foreign key referencing `Technician.user_id`.
- `dispute_reason`: Reason for the dispute.
- `dispute_status`: Status (open, resolved).

---

## Updated Service Categories and Examples

Below are some of the requested services and their categories:

1. **Home Appliances Repair**
    - Air Cooler Repair
    - AC Repair & Service
    - Washing Machine Repair
    - Refrigerator Repair
    - Water Purifier Repair & Service
    - Gas Stove Repair & Service
    - Television Repair
2. **Home Maintenance**
    - Electrician
    - Fan Installation
    - Plumber
    - Furniture Assembly
    - Wall Panels Installation
    - Mounting
    - Painting
3. **Cleaning Services**
    - Full Home Cleaning
    - Part-time Cleaners
    - Sofa & Carpet Cleaning
    - Water Tank Cleaning
4. **Real Estate Services**
    - Home On Rent
    - House Sell Service
    - Broker Service
5. **Personal Services**
    - Home Care
    - Home Servant
    - Security Guard Service
    - Hotel Servant Service

6. **Miscellaneous**
    - Ambulance Service
    - Talk To Expert
    - Cockroach, Ant & Pest Control

## Permissions Matrix

| Entity | Customer | Technician | Operator | Admin |
|---|---|---|---|---|
| **1. User Profile** | Update profile | Update profile | Update profile | CRUD on all users |
| **2. Address** | CRUD own | CRUD own | View, Edit customers' | CRUD all |
| **3. Payment Method** | CRUD own | CRUD own | No access | CRUD all |
| **4. Service Booking** | Create, View own | Accept, View assigned | View, Update bookings | CRUD all bookings |
| **5. Service** | Request services | Offer services | View, Assign | CRUD all |
| **6. Service Category** | View all | View all | View, Edit | CRUD all |

| | | | | |
|---|---|---|---|---|
| **7. Payment** | View own payments | View own payments | View all payments | CRUD all payments |
| **8. Review** | Create, View own | View, Respond | View, Moderate | CRUD all |
| **9. Notification** | View own | View own | Send to customers/technicians | CRUD all |
| **10. Promotion** | View, Apply | No access | Create, Edit | CRUD all |
| **11. Dispute** | Create, View own | Respond to disputes | Resolve disputes | CRUD all |

---

## Detailed Role-Based Permissions

### 1. Customer Permissions

- **Profile Management**: Can update their own profile details (e.g., name, contact information).
- **Address Management**: Can add, edit, delete, or view their own saved addresses.
- **Payment Methods**: Can add, update, delete, and view their own payment methods.
- **Booking Services**: Can view available services, create new bookings, view their own bookings, and cancel upcoming bookings.
- **Service Reviews**: Can provide ratings and reviews for services they've completed.
- **Notifications**: Can view notifications sent to them by the platform.
- **Promotions**: Can view and apply promotional codes during the booking process.
- **Disputes**: Can open a dispute for any unresolved or problematic service.

### 2. Technician Permissions

- **Profile Management**: Can update their own profile details, including uploading documents (e.g., certifications).
- **Service Offering**: Can view service requests that match their expertise and accept/reject job assignments.

- **Service Bookings**: Can view and manage their assigned bookings. Mark services as completed once done.
- **Payment Overview**: Can view payment history and invoices related to completed jobs.
- **Reviews**: Can view reviews and respond to feedback left by customers.
- **Notifications**: Receive notifications about new job assignments or updates.
- **Disputes**: Respond to any disputes raised by customers related to their completed services.

**3. Operator Permissions**

- **Profile Management**: Can update their own profile.
- **Service Monitoring**: View and manage service requests, assign services to available technicians based on location, availability, and expertise.
- **Address**: View and update customers' addresses if needed for a service.
- **Payments**: Can view payments related to bookings in their assigned region.
- **Customer and Technician Management**: View and assist in resolving issues for both customers and technicians in their region.
- **Disputes**: Review, moderate, and resolve disputes between customers and technicians.
- **Promotions**: Can create or edit promotions for the platform, depending on assigned region or service category.

**4. Admin Permissions**

- **Full User Management**: Admins can create, read, update, and delete any user (customer, technician, operator, other admins).
- **Address Management**: CRUD operations for all addresses.
- **Payment Methods**: Full CRUD access to payment methods across all users.
- **Service Management**: Admins can create, update, and delete services. They can also categorize services, modify pricing, and oversee all technician profiles.
- **Service Categories**: CRUD operations for all service categories.
- **Booking Management**: Admins can view, update, and delete any booking. They can reassign bookings between technicians if necessary.
- **Payments**: Full access to all payments, commissions, and payouts to technicians.
- **Reviews and Ratings**: Admins can view and moderate all reviews on the platform, remove inappropriate feedback, and resolve conflicts.
- **Notifications**: CRUD operations for sending notifications to all users.
- **Promotions**: Admins can create, update, and delete promotions for the platform.
- **Disputes**: Admins can fully manage all disputes, including resolving escalated cases.

---

## Access Summary for Each User Role

1. **Customer**:

    - **Primary focus**: Interact with the platform for service booking, managing their profile, addresses, payments, and disputes. Limited access to view promotions and their own bookings.
2. **Technician**:
    - **Primary focus**: Manage job requests, mark them as complete, and handle payments related to services. They can interact with customers through reviews and disputes but have limited control over the platform's features.
3. **Operator**:
    - **Primary focus**: Act as an intermediary between customers and technicians. They manage bookings, handle disputes, and maintain the smooth functioning of services within their assigned region. Limited admin-like permissions for certain operations.
4. **Admin**:
    - **Full access**: Manage all aspects of the platform, from user roles to service management and bookings, payments, disputes, reviews, and promotions.