Technical documentation

1. Database Schema Documentation

Introduction

This document describes the database schema for managing a platform with users, technicians, job postings, job bids, payments, reviews, and disputes. It covers the structure, relationships, and functionalities of the database tables.

---

1. Users Table

Table Name: users

This table stores the general information of users in the system. It serves as the core for user management, including admin, client, and technician roles.

Field	Type	Description
id	INT	Primary key, unique user identifier.
username	VARCHAR(100)	User's chosen username.
email	VARCHAR(100)	User's email address (unique).
password_hash	VARCHAR(255)	Hashed password for user authentication.
profile_image	VARCHAR(255)	Optional profile picture for the user.
phone_number	VARCHAR(20)	General phone number of the user.
mobile_phone	VARCHAR(20)	User's mobile phone number.
location	VARCHAR(255)	User's geographical location.
user_role	ENUM	Defines user role: admin, client, or technician.
is_verified	BOOLEAN	Flag indicating whether the user's email has been verified.
created_at	DATETIME	Timestamp for when the user was created.

updated_at	DATETIME	Timestamp for when the user was last updated.
deleted_at	DATETIME	Timestamp for when the user was soft deleted (nullable).

Relationships:

- **Foreign keys**: None directly in this table, but other tables like technicians, job_postings reference this table.

---

2. Technicians Table

Table Name: technicians

This table stores detailed information about technicians, including their skills, certifications, and job-related metrics.

Field	Type	Description
id	INT	Primary key, references users.id to associate technician with a user.
commercial_records	INT	Business documentation identifier.
identity_number	VARCHAR(255)	Technician's identity number (e.g., national ID or passport).
skills	TEXT	Comma-separated list of technician's skills.
hourly_rate	DECIMAL(10,2)	Technician's hourly rate.
certifications	TEXT	Comma-separated list of certifications held by the technician.
bio	TEXT	Short biography or description of the technician's background.
location	VARCHAR(255)	Technician's geographical location.
rating	DECIMAL(3,2)	Average rating based on user reviews.
available_from	DATETIME	Date and time when the technician is available to work.

created_at	DATETIME	Timestamp for when the technician profile was created.
updated_at	DATETIME	Timestamp for when the technician profile was last updated.
deleted_at	DATETIME	Timestamp for when the technician was soft deleted (nullable).

Relationships:

- **Foreign key**: user.id (References users.id)

---

3. Admins Table

Table Name: admins

This table stores information specific to admins, who have elevated privileges.

Field	Type	Description
id	INT	Primary key, references users.id to associate admin with a user.
additional_info	TEXT	Additional details about the admin.
created_at	DATETIME	Timestamp for when the admin was created.
updated_at	DATETIME	Timestamp for when the admin profile was last updated.
deleted_at	DATETIME	Timestamp for when the admin was soft deleted (nullable).

Relationships:

- **Foreign key**: user.id (References users.id)

---

4. Categories Table

Table Name: categories

This table stores job categories for organizing job postings.

Field	Type	Description
id	INT	Primary key.
category_name	VARCHAR(100)	Name of the job category (e.g., Plumbing, Electrical, etc.).
description	TEXT	Description of the category.
created_at	DATETIME	Timestamp for when the category was created.
updated_at	DATETIME	Timestamp for when the category was last updated.
deleted_at	DATETIME	Timestamp for when the category was soft deleted (nullable).

---

5. Job Postings Table

Table Name: job_postings

This table stores information about job postings made by clients.

Field	Type	Description
id	INT	Primary key.
client_id	INT	Foreign key referencing users.id for the client who posted the job.
title	VARCHAR(255)	Job posting title.
description	TEXT	Detailed description of the job.
category_id	INT	Foreign key referencing categories.id to categorize the job.
location	VARCHAR(255)	Job location.
budget_min	DECIMAL(10,2)	Minimum budget for the job.
budget_max	DECIMAL(10,2)	Maximum budget for the job.
duration	VARCHAR(100)	Estimated duration of the job.
status	ENUM	Current status of the job (open, in_progress, completed, cancelled, closed).

posted_at	DATETIME	Timestamp for when the job was posted.
updated_at	DATETIME	Timestamp for when the job was last updated.
deleted_at	DATETIME	Timestamp for when the job was soft deleted (nullable).

Relationships:

- **Foreign key**: client_id (References users.id)
- **Foreign key**: category_id (References categories.id)

---

6. Job Bids Table

Table Name: job_bids

This table records bids made by technicians on job postings.

Field	Type	Description
id	INT	Primary key.
job_id	INT	Foreign key referencing job_postings.id for the job being bid on.
technician_id	INT	Foreign key referencing technicians.id for the technician making the bid.
bid_amount	DECIMAL(10,2)	Bid amount offered by the technician.
bid_message	TEXT	Message accompanying the bid.
status	ENUM	Bid status (pending, accepted, declined, withdrawn).
bid_date	DATETIME	Timestamp for when the bid was made.
updated_at	DATETIME	Timestamp for when the bid was last updated.
deleted_at	DATETIME	Timestamp for when the bid was soft deleted (nullable).

Relationships:

- **Foreign key**: job_id (References job_postings.id)
- **Foreign key**: technician_id (References technicians.id)

---

7. Escrow Payments Table

Table Name: escrow_payments

This table manages the escrow payments for completed jobs.

Field	Type	Description
id	INT	Primary key.
job_id	INT	Foreign key referencing job_postings.id for the related job.
client_id	INT	Foreign key referencing users.id for the client making the payment.
technician_id	INT	Foreign key referencing technicians.id for the technician receiving the payment.
amount_min	DECIMAL(10,2)	Minimum agreed payment amount.
amount_max	DECIMAL(10,2)	Maximum agreed payment amount.
status	ENUM	Payment status (hold, released, refunded, disputed).
created_at	DATETIME	Timestamp for when the escrow payment was created.
updated_at	DATETIME	Timestamp for when the escrow payment was last updated.
deleted_at	DATETIME	Timestamp for when the escrow payment was soft deleted (nullable).

Relationships:

- **Foreign key**: job_id (References job_postings.id)
- **Foreign key**: client_id (References users.id)
- **Foreign key**: technician_id (References technicians.id)

---

8. Payments Table

Table Name: payments

This table records the payments for completed jobs.

Field	Type	Description
id	INT	Primary key.
job_id	INT	Foreign key referencing job_postings.id for the related job.
client_id	INT	Foreign key referencing users.id for the client making the payment.
technician_id	INT	Foreign key referencing technicians.id for the technician receiving the payment.
amount	DECIMAL(10,2)	Total amount paid for the job.
payment_method	ENUM	Payment method used (credit_card, bank_transfer, paypal, other).
payment_status	ENUM	Status of the payment (pending, completed, failed, refunded).
transaction_id	VARCHAR(255)	Unique transaction ID from the payment gateway.
payment_date	DATETIME	Timestamp for when the payment was made.
created_at	DATETIME	Timestamp for when the payment was created.
updated_at	DATETIME	Timestamp for when the payment was last updated.

Relationships:

- **Foreign key**: job_id (References job_postings.id)
- **Foreign key**: client_id (References users.id)
- **Foreign key**: technician_id (References technicians.id)

---

9. Reviews Table

Table Name: reviews

This table records user reviews and ratings of technicians or clients.

Field	Type	Description
id	INT	Primary key.

job_id	INT	Foreign key referencing job_postings.id for the related job.
reviewer_id	INT	Foreign key referencing users.id for the user giving the review.
reviewee_id	INT	Foreign key referencing users.id for the user being reviewed.
rate	INT	Rating (1-5) given by the reviewer.
review_message	TEXT	Review content written by the reviewer.
created_at	DATETIME	Timestamp for when the review was created.
updated_at	DATETIME	Timestamp for when the review was last updated.
deleted_at	DATETIME	Timestamp for when the review was soft deleted (nullable).

Relationships:

- **Foreign key**: job_id (References job_postings.id)
- **Foreign key**: reviewer_id (References users.id)
- **Foreign key**: reviewee_id (References users.id)

---

10. Disputes Table

Table Name: disputes

This table tracks disputes between clients and technicians for jobs.

Field	Type	Description
id	INT	Primary key.
job_id	INT	Foreign key referencing job_postings.id for the job in dispute.
initiator_id	INT	Foreign key referencing users.id for the user initiating the dispute.
technician_id	INT	Foreign key referencing technicians.id for the technician involved.
client_id	INT	Foreign key referencing users.id for the client involved in the dispute.

dispute_reason	TEXT	Description of the dispute.
status	ENUM	Dispute status (open, resolved, cancelled).
created_at	DATETIME	Timestamp for when the dispute was created.
resolved_at	DATETIME	Timestamp for when the dispute was resolved (nullable).
deleted_at	DATETIME	Timestamp for when the dispute was soft deleted (nullable).

Relationships:

- **Foreign key**: job_id (References job_postings.id)
- **Foreign key**: initiator_id (References users.id)
- **Foreign key**: technician_id (References technicians.id)
- **Foreign key**: client_id (References users.id)

---

11. Contacts Table

Table Name: contacts

This table stores communication between users and technicians regarding job postings.

Field	Type	Description
id	INT	Primary key.
job_id	INT	Foreign key referencing job_postings.id for the job in question.
technician_id	INT	Foreign key referencing technicians.id for the technician involved.
name	VARCHAR(255)	Name of the sender.
email	VARCHAR(255)	Email of the sender.
subject	VARCHAR(255)	Subject of the contact message.
message	TEXT	Content of the contact message.
created_at	DATETIME	Timestamp for when the contact message was created.
deleted_at	DATETIME	Timestamp for when the contact message was soft deleted (nullable).

Relationships:

- **Foreign key**: job_id (References job_postings.id)
- **Foreign key**: technician_id (References technicians.id)

---

12. Admin Actions Table

Table Name: admin_actions

This table tracks administrative actions taken by admins.

Field	Type	Description
id	INT	Primary key.
admin_id	INT	Foreign key referencing admins.id for the admin who performed the action.
action_type	ENUM	Type of action (ban_user, approve_profile, resolve_dispute).
description	TEXT	Detailed description of the action.
target_user_id	INT	Foreign key referencing users.id for the affected user.
created_at	DATETIME	Timestamp for when the action was taken.
deleted_at	DATETIME	Timestamp for when the action was soft deleted (nullable).

Relationships:

- **Foreign key**: admin_id (References admins.id)
- **Foreign key**: target_user_id (References users.id)

---

2. Software Design

System Architecture

The system follows a client-server architecture, where:

- Frontend: The user interface (UI) is web or mobile-based. It allows interaction with the system through forms, dashboards, and reports.
- Backend: The backend consists of Laravel 10 MVC that communicates with the database, performs business logic, and serves data to the frontend.
- Database: The relational database (MySQL) stores all user, job, payment, and other relevant data.

Key Components

1. User Management Module:
    - User Registration & Authentication: A user can register an account, log in, and manage their profile (for clients, technicians, and admins).
    - User Role Management: Differentiates users into clients, technicians, and admins with role-specific permissions.
2. Job Management Module:
    - Job Posting: Clients can post jobs with details such as title, description, location, category, and budget.
    - Job Bidding: Technicians can browse open jobs, submit bids with proposals and amounts.
    - Job Status Management: Track job progress with statuses like 'open', 'in progress', 'completed', etc.
3. Escrow and Payments Module:
    - Escrow Payments: Payments are held in escrow when a job is accepted, and released to technicians upon job completion.
    - Payment Processing: Payments are processed when a job is completed, and can be managed through multiple payment methods.
4. Review and Rating Module:
    - Client and Technician Ratings: Clients and technicians can rate each other after a job is completed.
    - Review Management: Reviews include ratings (1-5) and textual feedback.
5. Dispute Resolution Module:
    - Dispute Management: Allows clients and technicians to submit disputes related to a job. Admins can resolve these disputes.
    - Case Logging: Dispute cases are logged with status updates and resolutions.
6. Admin Dashboard:
    - Admins can manage users, job postings, payments, and disputes, with the ability to view reports and logs.
7. Notification System:
    - Notifies users about job updates, bids, payment confirmations, dispute resolutions, etc.

Database Design

The database is designed to support:

- Normalization: Ensures efficient data storage and avoids data redundancy (1NF, 2NF, 3NF).
- Referential Integrity: Ensures that relationships between tables are consistent and valid (foreign key constraints).
- Scalability: Database can handle high volumes of users, jobs, bids, and transactions with efficient indexing.

Technologies

- Frontend: HTML, CSS, JavaScript (for web).
- Backend: PHP LARAVEL 10 MVC with Breeze.
- Database: MySQL,WAMP.
- Payment Gateway: Stripe, PayPal.
- third-party services for processing payments.
- Authentication: JWT (JSON Web Token) for user authentication.

Functional Requirements

1. User Management

- User Registration & Login:
    - Users (clients, technicians, admins) can create an account by providing required details (email, username, password).
    - Passwords should be hashed for security.
    - Users can log in using their credentials.
    - Forgotten passwords can be reset via email.
- Profile Management:
    - Users can update their profile information (contact details, location, etc.).
    - Technicians can list their skills, certifications, and hourly rates.
    - Admins can manage users, including deactivating or deleting accounts.
- User Roles & Permissions:
    - Admins have full access to the system (can manage jobs, users, disputes, payments).
    - Clients can post jobs, make payments, and leave reviews.
    - Technicians can bid on jobs, complete work, and receive payments.

2. Job Management

- Job Posting:

    - Clients can create job postings with title, description, budget range, and job category.
    - Job postings should be visible to technicians matching the job category.
- Job Bidding:
    - Technicians can view job listings and submit bids.
    - Technicians must provide a bid amount and a proposal message.
    - Clients can review and accept or reject bids.
- Job Status:
    - Job status must be tracked (e.g., open, in progress, completed, cancelled).
    - Technicians must mark jobs as completed when done.

3. Escrow & Payments

- Escrow System:
    - When a job is accepted, the client's payment is placed in escrow.
    - Payment is only released to the technician when the job is completed and approved by the client.
- Payment Processing:
    - Clients can make payments for completed jobs via integrated payment gateways.
    - Payment information (transaction IDs, amounts, methods) should be stored in the database.

4. Reviews & Ratings

- Technician & Client Reviews:
    - After job completion, clients can rate technicians (1-5 stars) and leave reviews.
    - Technicians can rate clients based on job experience.
    - Reviews should be stored with timestamps and linked to the relevant job.
- Review Moderation:
    - Admins can moderate reviews to prevent inappropriate content.

5. Dispute Management

- Dispute Submission:
    - Either the client or technician can submit a dispute for unresolved issues (payment disputes, quality of work).
    - A dispute must have a clear reason and evidence (e.g., photos, descriptions).
- Dispute Resolution:
    - Admins can review and resolve disputes based on provided information.
    - Admins can make decisions on refunds or penalties, and update the dispute status (resolved, ongoing).

6. Admin Functionality

- User & Job Management:

    - Admins can view and manage users, job postings, and job bids.
    - Admins can assign or cancel jobs, approve/reject job postings, and deactivate users.
- Report Generation:
    - Admins can generate reports related to user activity, job completions, payment statuses, and disputes.
    - Admins should be able to view user feedback, ratings, and financial summaries.

7. Notification System

- Job Status Notifications:
    - Notify clients when a bid is submitted or accepted.
    - Notify technicians when their bid is accepted, or job status changes.
- Escrow & Payment Notifications:
    - Notify clients when payments are received and placed in escrow.
    - Notify technicians when payments are released.
- Dispute Notifications:
    - Notify users when a dispute is filed or resolved.

Non-Functional Requirements

1. Security:
    - Use HTTPS for secure communication.
    - Passwords must be encrypted (e.g., using bcrypt).
    - Use JWT for user authentication and session management.
    - Implement role-based access control for admin and user permissions.
2. Scalability:
    - The system should be able to handle thousands of simultaneous users, job postings, and transactions.
    - Database design must include indexing for fast query performance (especially for jobs and payments).
3. Usability:
    - The frontend should be user-friendly with easy navigation for job posting, bidding, and payment processing.
    - Mobile and desktop versions should be responsive.
4. Reliability:
    - The system should have a 99.9% uptime.
    - Backup and recovery procedures must be in place for data integrity.
5. Performance:

    - The system should be able to handle job posting, bidding, and payment actions in under 3 seconds.
    - High-traffic times (e.g., holidays) should be handled efficiently without performance degradation.
6. Compliance:
    - The system must comply with relevant data protection regulations (e.g., GDPR, CCPA) for user data.
    - Payment processing must comply with PCI DSS standards.