# ABSTRACT

JusticeDesk is a web-based platform built during a 24-hour Hackverse hackathon to provide categorized legal information and a bridge between users seeking legal help and registered lawyers. The platform supports structured legal content (chapters/sections), user-submitted case forms, lawyer profiles, and an admin panel to manage the ecosystem. The project is implemented using HTML, CSS, Bootstrap, JavaScript, PHP and MySQL.

**PROBLEM STATEMENT**

Many users face difficulty finding reliable legal information and connecting to lawyers quickly. Physical visits are time-consuming and costly, and legal content is often scattered. JusticeDesk addresses these problems by centralizing legal content, allowing users to submit cases and enabling lawyers to view submissions and offer consultations.

**OBJECTIVES**

- Provide categorized, searchable legal content.

- Allow users to submit legal issue forms and request consultations.

- Enable lawyers to register, create profiles, and access submissions.

- Implement an Admin panel for moderation and management.

- Deliver a secure authentication system for users and lawyers.

- Demonstrate a working MVP within 24 hours.

**SCOPE AND LIMITATIONS**

Scope:

- Web-based MVP focusing on essential features: legal content pages, user registration, lawyer registration, form submission, lawyer dashboard, admin panel.

Limitations:

- Not a substitute for formal legal advice.

- Advanced features (secure payments, fully encrypted messaging, AI legal advice) are out-of-scope for the 24-hour build.

**HIGH-LEVEL ARCHITECHTURE**

Frontend: HTML, CSS, Bootstrap, JS

Backend: PHP

Database: MySQL

Hosting: LAMP stack (local or cloud)

Data Flow: Users -> Submit Form -> Stor ed in DB -> Lawyers view via Dashboard -> Lawyer contacts user.

Admin monitors and manages users, lawyers, and submissions.

# SYSTEM MODULES

1. Public (Landing, About, Legal Content Browsing)

2. User Module (Register/Login, Submit Case Form, View Status)

3. Lawyer Module (Register/Login, Profile, View Forms, Contact User)

4. Admin Module (Manage users/lawyers/forms, Approvals)

5. Authentication & Authorization

6. Database Layer

7. (Optional) Real-time Chat — MVP: poll-based or AJAX-driven messaging

**Database Design (Summary)**

- users(id, name, email, password\_hash, phone, role('user'/'lawyer'/'admin'), created\_at)

- lawyers(id, user\_id, qualifications, experience, specialties, profile\_summary, verified, created\_at)

- legal\_chapters(id, title, description)

- legal\_sections(id, chapter\_id, section\_no, title, content)

- submissions(id, user\_id, title, description, category, status, assigned\_lawyer\_id, created\_at)

- messages(id, submission\_id, sender\_id, receiver\_id, message, timestamp)

- admin\_actions(id, admin\_id, action\_type, target\_type, target\_id, remark, timestamp)

**UML & Diagrams (Descriptions)**

Use-case Diagram: Actors — Guest, Registered User, Lawyer, Admin. Use-cases include Browse Laws, Register, Submit Case, View Submissions, Manage Users.

Sequence Diagram (Submit Case): User fills form -> Backend validates -> Store in DB -> Notify Lawyers (in dashboard) -> Admin logs action if required.

ER Diagram: Entities listed in Database Design above. Relationships: users-1..1->lawyers, chapters->sections (1..\*), users->submissions (1..\*), submissions->messages (1..\*).

**Admin Panel Features**

- Dashboard metrics: total users, lawyers, submissions (open/closed)

- Approve/Reject lawyer registrations

- View and search submissions

- Assign/transfer submissions to lawyers

- Manage legal content (add/edit chapters & sections)

- View audit logs and admin actions

- Manage site settings

# Security Considerations

- Passwords hashed using bcrypt/argon2

- Use prepared statements to prevent SQL injection

- Input validation and server-side checks

- Access control: role-based checks for admin/lawyer/user

- Basic rate limiting for form submissions

- HTTPS enforced in production

**Implementation Plan for 24-hour Hackathon**

Hour 0-1: Project setup, repo, basic UI skeleton

Hour 1-4: Authentication (user/lawyer/admin), DB schema

Hour 4-8: Legal content pages and admin CRUD for chapters/sections

Hour 8-12: Submission form, submissions listing for lawyers

Hour 12-16: Admin panel features and dashboards

Hour 16-20: Messaging MVP (AJAX polling) and search

Hour 20-22: Testing, bug fixes

Hour 22-24: Final polish, README, packaging, demo preparation

**Testing Strategy**

- Unit tests for PHP functions where feasible

- Manual functional testing for registration, login, submission flow

- Cross-browser checks for UI (Chrome, Firefox)

- Security checks: test for SQL injection/basic XSS

- Acceptance tests: Admin approves lawyer, Lawyer sees assigned submission, User receives status update

# Sample Code Snippets

1) Database connection (PHP - PDO):

<?php

$dsn='mysql:host=localhost;dbname=justicedeck;charset=utf8';

$pdo=new PDO($dsn,'root','password', [PDO::ATTR\_ERRMODE=>PDO::ERRMODE\_EXCEPTION]);

?>

2) Prepared statement (insert submission):

<?php

$stmt=$pdo->prepare('INSERT INTO submissions (user\_id,title,description,category,status,created\_at) VALUES (?, ?, ?, ?, ?, NOW())');

$stmt->execute([$user\_id,$title,$desc,'general','open']);

?>

**Deployment & Setup (Quick)**

1. Install Apache, MySQL, PHP (LAMP)

2. Place project files in /var/www/html/justicedeck

3. Import database schema (provided in SQL file)

4. Configure config.php with DB credentials

5. Enable mod\_rewrite if friendly URLs required

6. Use HTTPS (Let's Encrypt) in production

**Deliverables (in ZIP)**

- JusticeDesk\_Project\_Report.docx

- JusticeDesk\_Project\_Report.pdf

- README.md (setup instructions)

- SQL schema file (schema.sql)

- Sample demo data (sample\_data.sql)

- Admin credentials (example)

**Future Enhancements**

- Real-time chat with WebSockets

- Payment gateway for paid consultations

- Role verification via KYC

- Mobile-friendly responsive PWA

- AI-powered legal document summarization

# Conclusion

JusticeDesk is an MVP designed to address the need for fast access to legal information and connection to legal professionals. The 24-hour Hackverse build focuses on core features and a workable admin panel to manage the ecosystem. The project provides a solid foundation for future expansion.

# Appendix: SQL Schema (Key Tables DDL)

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(150),

email VARCHAR(150) UNIQUE,

password\_hash VARCHAR(255),

phone VARCHAR(30),

role ENUM('user','lawyer','admin') DEFAULT 'user',

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE lawyers (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

qualifications TEXT,

experience VARCHAR(50),

specialties VARCHAR(255),

profile\_summary TEXT,

verified TINYINT DEFAULT 0,

FOREIGN KEY (user\_id) REFERENCES users(id)

);

CREATE TABLE submissions (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

title VARCHAR(255),

description TEXT,

category VARCHAR(100),

status VARCHAR(50) DEFAULT 'open',

assigned\_lawyer\_id INT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id)

);