# University Management Application Documentation

# Introduction

Creating documentation outlining the University Management Application, a mobile platform connecting students, faculty, and administrators. Built to streamline university operations with features like authentication, profile photo management, task assignment, ticket systems, resume building, attendance tracking, and real-time chat functionality. This concise document explains each feature and database structure for client understanding.

## System Architecture

Describing the system architecture by detailing the following components:

* **Frontend**: Utilizing a cross-platform mobile framework to create a seamless experience across devices.
* **Backend**: Managing API requests and server-side logic.
* **Database**: Storing all user data, communications, and academic records efficiently.
* **Real-time Communication**: Enabling instant messaging between users.

## Installation and Setup

Explaining the installation and setup process by covering:

* **Frontend**: Requiring installation of necessary mobile app dependencies and configuration to connect with the backend server.
* **Backend**: Involving setting up a server environment with required dependencies and configuring environment variables for secure operation.
* **MySQL Setup**: Requiring installation of MySQL and creation of the university database with the provided schema.

## Database Structure

Defining the database structure organizing data efficiently with role-specific tables and relational integrity. Presenting SQL queries for creating all tables:

* **Students Table**  
  Creating a table for students with the following schema:
* CREATE TABLE students (
* id INT AUTO\_INCREMENT PRIMARY KEY,
* name VARCHAR(255) NOT NULL,
* email VARCHAR(255) NOT NULL UNIQUE,
* password VARCHAR(255) NOT NULL,
* branch VARCHAR(100) NOT NULL,
* photo VARCHAR(255),
* otp VARCHAR(6),
* profile\_edit BOOLEAN DEFAULT FALSE,
* created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,
* updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP
* );
* **Faculty Table**  
  Creating a table for faculty with the following schema:
* CREATE TABLE faculty (
* id INT AUTO\_INCREMENT PRIMARY KEY,
* name VARCHAR(255) NOT NULL,
* email VARCHAR(255) NOT NULL UNIQUE,
* password VARCHAR(255) NOT NULL,
* branch VARCHAR(100) NOT NULL,
* photo VARCHAR(255),
* otp VARCHAR(6),
* profile\_edit BOOLEAN DEFAULT FALSE,
* created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,
* updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP
* );
* **Admin Table**  
  Creating a table for admin with the following schema:
* CREATE TABLE admin (
* id INT AUTO\_INCREMENT PRIMARY KEY,
* name VARCHAR(255) NOT NULL,
* email VARCHAR(255) NOT NULL UNIQUE,
* password VARCHAR(255) NOT NULL,
* photo VARCHAR(255),
* otp VARCHAR(6),
* created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,
* updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP
* );
* **Messages Table**  
  Creating a table for messages with the following schema:
* CREATE TABLE messages (
* id INT AUTO\_INCREMENT PRIMARY KEY,
* sender\_id INT NOT NULL,
* sender\_role ENUM('student', 'faculty', 'admin') NOT NULL,
* receiver\_id INT NOT NULL,
* receiver\_role ENUM('student', 'faculty', 'admin') NOT NULL,
* content TEXT NOT NULL,
* timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,
* is\_read BOOLEAN DEFAULT 0,
* created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,
* updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,
* INDEX (sender\_id),
* INDEX (receiver\_id)
* );
* **Resumes Table**  
  Creating a table for resumes with the following schema:
* CREATE TABLE resumes (
* id INT AUTO\_INCREMENT PRIMARY KEY,
* student\_id INT NOT NULL,
* objective TEXT,
* education TEXT,
* skills TEXT,
* experience TEXT,
* projects TEXT,
* certifications TEXT,
* achievements TEXT,
* languages TEXT,
* references\_info TEXT,
* additional\_info TEXT,
* created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,
* updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,
* FOREIGN KEY (student\_id) REFERENCES students(id)
* );
* **Attendance Table**  
  Creating a table for attendance with the following schema:
* CREATE TABLE attendance (
* id INT AUTO\_INCREMENT PRIMARY KEY,
* faculty\_id INT NOT NULL,
* student\_id INT NOT NULL,
* status ENUM('present', 'absent') NOT NULL,
* date DATE NOT NULL,
* semester VARCHAR(50),
* subject VARCHAR(100),
* created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,
* FOREIGN KEY (faculty\_id) REFERENCES faculty(id),
* FOREIGN KEY (student\_id) REFERENCES students(id),
* INDEX (faculty\_id),
* INDEX (student\_id)
* );
* **Tasks Table**  
  Creating a table for tasks with the following schema:
* CREATE TABLE tasks (
* id INT AUTO\_INCREMENT PRIMARY KEY,
* title VARCHAR(255) NOT NULL,
* description TEXT,
* assigned\_to INT NOT NULL,
* role ENUM('student', 'faculty', 'admin') NOT NULL,
* status ENUM('pending', 'completed') DEFAULT 'pending',
* due\_date DATETIME,
* assigned\_by INT NOT NULL,
* created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,
* updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,
* link TEXT,
* FOREIGN KEY (assigned\_to) REFERENCES students(id),
* FOREIGN KEY (assigned\_by) REFERENCES faculty(id),
* INDEX (assigned\_to),
* INDEX (assigned\_by)
* );
* **Tickets Table**  
  Creating a table for tickets with the following schema:
* CREATE TABLE tickets (
* id INT AUTO\_INCREMENT PRIMARY KEY,
* subject VARCHAR(255) NOT NULL,
* description TEXT,
* raised\_by INT NOT NULL,
* role ENUM('student', 'faculty', 'admin') NOT NULL,
* status ENUM('open', 'completed', 'requested\_updates') DEFAULT 'open',
* response TEXT,
* created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,
* updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,
* FOREIGN KEY (raised\_by) REFERENCES students(id)
* );

## Authentication

Explaining authentication features:

* **Registration**: Allowing users to create an account with basic details like name, email, password, and branch. Ensuring no duplicate emails.
* **Login**: Authenticating users with credentials and assigning a role-specific token for secure access to dashboards.
* **Forgot Password & OTP**: Sending a one-time password to the user's email for password reset initiation.
* **Reset Password**: Enabling users to update their password using the OTP, ensuring security.

## Profile Photo

Describing profile photo management by allowing users to upload and manage profile photos, which are stored on the server and linked to their database records.

## Dashboards

Detailing dashboards for different roles:

* **Student Dashboard**: Providing access to personal profile, resume, attendance records, assigned tasks, chat, and ticket submissions.
* **Faculty Dashboard**: Allowing management of students, attendance marking, task assignment, chat, and ticket resolution.
* **Admin Dashboard**: Offering user management, system configuration, report generation, and ticket handling.

## Real-time Chat

Explaining chat functionality facilitating instant messaging between students, faculty, and admins with persistent message storage and read status tracking.

## Resume Management

Describing resume building enabling students to create, update, and download resumes with customizable sections like education, skills, and experience.

## Attendance Tracking

Detailing attendance management where faculty can mark attendance for students, which is recorded with details like date, semester, and subject. Students can view their attendance statistics.

## Task Management

Explaining task assignment where faculty assign tasks to students with titles, descriptions, due dates, and links. Students can update task status.

## Ticket System

Describing ticket submission where users can raise tickets with subjects and descriptions, which can be resolved by faculty or admins with optional updates or responses.

## Security

Outlining security measures implementing token-based authentication with expiry, password hashing, input validation, restricted file uploads, and secure environment variable management.

## Conclusion

Concluding that the application improves university operations by providing a secure, scalable, and user-friendly platform for managing various academic and administrative tasks.