CSE370 : Database Systems
Project Report
Project Title : Project Title Here

Group No : 6, CSE370 Lab Section : 12, Spring 2025		
ID	**Name**	**Contribution**
22299546	N M Saif Kabir	All

Table of Contents

Introduction

A web platform to help agencies efficiently manage multiple projects, tasks, resources, and budgets in one place.

Project Features

1. Admin Dashboard

A central hub that pulls together up-to-date counts and tables for clients, projects, tasks, resources, staff, payments and financial summaries, giving you a single snapshot of agency operations.

2. Client & Project Management

- **Client intake:** capture name, contact details and contract terms; view an auto-generated client list.

- **Project creation & tracking:** create projects against a chosen client, set start/end dates and a status (Planned, Active, etc.); view an "Ongoing Projects" table that tracks every project's timeline and state.

3. Staff Management

- **Staff onboarding:** add staff with name, email and role.

- **Roster view:** a full staff list showing IDs, contact details and positions for quick assignment or lookup.

4. Resource Management

- **Resource logging:** record software licences, ad spend, supplies and any other cost items, optionally linking each one to a project.

- **Cost ledger:** running resource table with auto‑calculated dollar amounts so you can see where money is being spent.

5. Task Management

- **Task creation:** define tasks for a project, set deadlines, priority and status.

- **Progress board:** list of all tasks with real‑time status (Not Started, In Progress, Completed).

- **Assignment tool:** match each task to specific staff members and keep a visible "Task ⇄ Staff" matrix.

6. Payments

- **Payment recording:** log payment date, amount and notes, tied to a project.

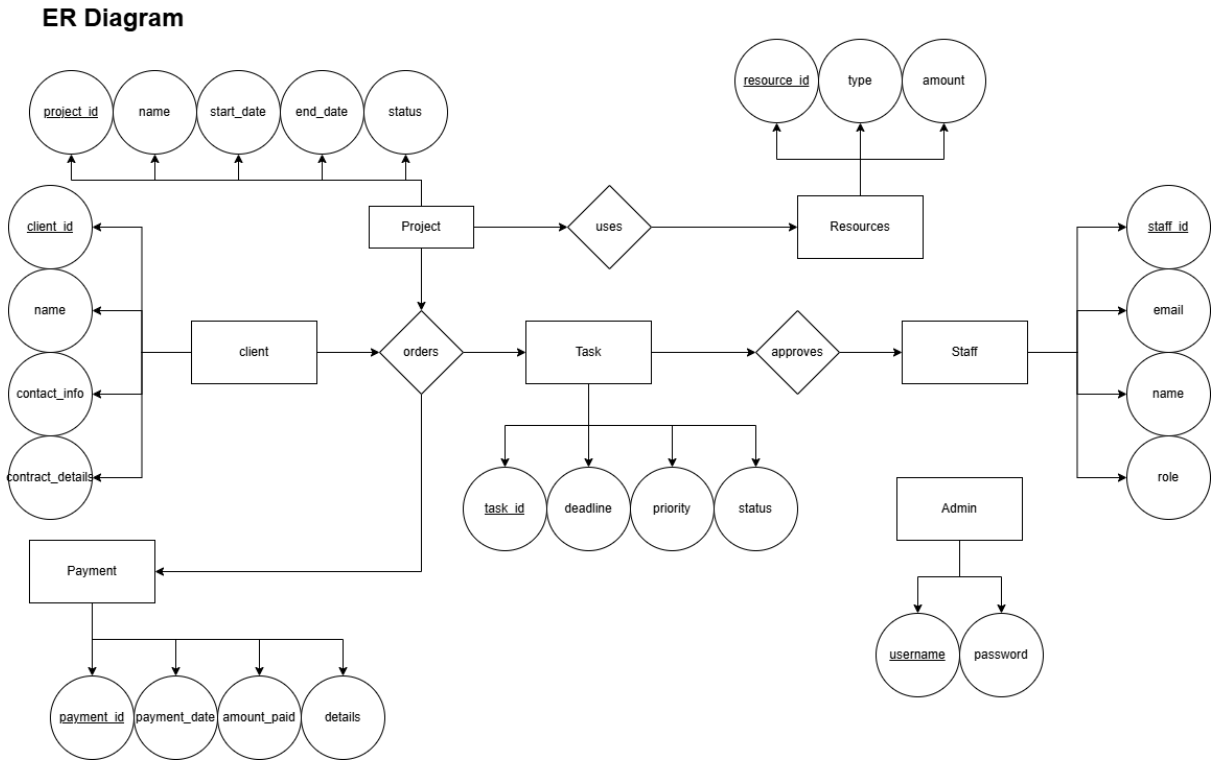
- **Payment history:** chronological ledger per project to check which milestones or retainers have been paid.

7. Financial Overview & Reporting

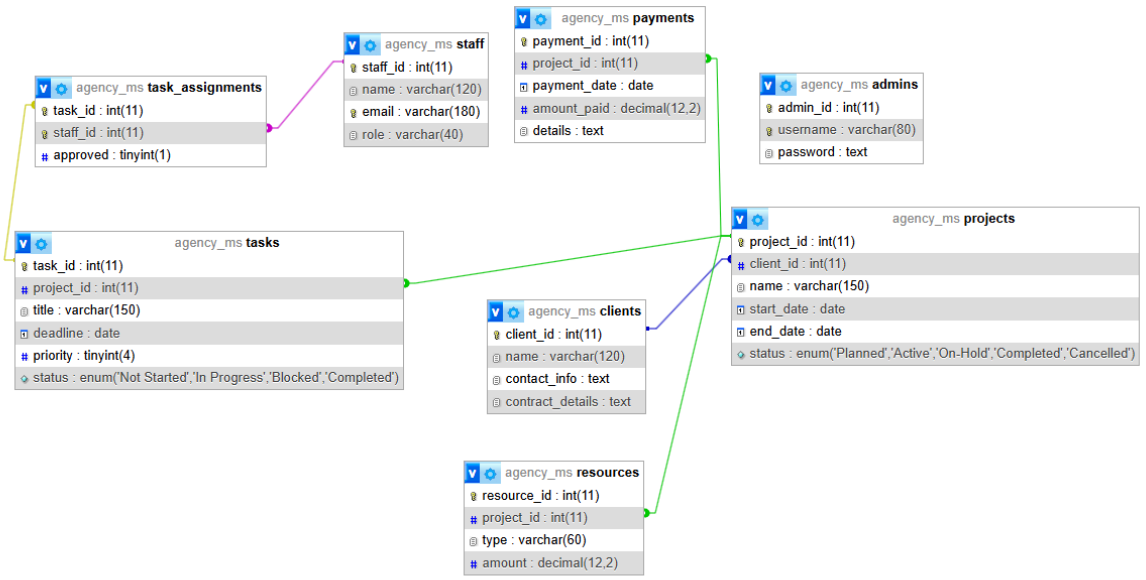
- **Filterable P&L:** choose a project status (e.g., Not Started) and see revenue, cost and profit/loss for each matching project.

- **Aggregated totals:** automatic roll‑up of total revenue, total cost and net profit/loss for the selected slice.

ER/EER Diagram



Schema Diagram



Frontend Development

There are only 2 pages. Login and Admin pages.

The frontend of the Agency Management Website Tool is built using React.js, a popular JavaScript library for creating user interfaces.

The core of the frontend is the AdminDashboard component (admin.jsx). This component utilizes React Hooks (useState and useEffect) extensively for managing the application's state. This includes:

Storing data fetched from the backend (e.g., lists of clients, projects, staff).
Holding the values of input fields for creating new entries (e.g., new client details, new project information).
Tracking loading and error states for asynchronous operations like fetching or submitting data.
Data is fetched from and sent to a backend API (presumably running on http://localhost:3000) using the browser's fetch API. Asynchronous JavaScript (async/await) is used to handle these network requests.

The user interface is structured to allow administrators to view lists of existing entities (clients, projects, staff, resources, payments, tasks) and to add new entities through dedicated forms. It also includes a financial overview section that calculates and displays revenue, costs, and profits based on project data.

Styling is managed through CSS Modules (admin.module.css), which helps in scoping CSS locally to the component, preventing style conflicts.

Backend Development

The backend for the Agency Management Website Tool is developed using Node.js and the Express.js framework. It serves as an API that the frontend React application communicates with.

Key aspects of the backend include:

API Endpoints: It defines various HTTP routes (e.g., /clients, /projects, /staff) to handle requests for different resources.
GET routes are used to fetch data (e.g., app.get('/projects', ...) retrieves a list of projects).
POST routes are used to create new data entries (e.g., app.post('/projects', ...) adds a new project).
Database Interaction: It connects to a MySQL database named agency_ms using the mysql Node.js driver.
Raw SQL queries are embedded within the route handlers to perform operations like SELECT (to fetch data) and INSERT (to add new data).
For example, when a POST request is made to /projects, the backend constructs an INSERT INTO projects ... SQL query with the data received in the request body.
Request Handling:
It uses the express.json() middleware to parse incoming JSON request bodies.
The cors middleware is used to enable Cross-Origin Resource Sharing, allowing the frontend (likely running on a different port) to make requests to this backend.
Data Validation & Error Handling:
Basic validation is performed on incoming data (e.g., checking for required fields like name and client_id when creating a project).
It includes error handling for database operations and request processing, sending appropriate HTTP status codes (e.g., 400 for bad requests, 500 for server errors) and error messages back to the client. It also checks for specific database error codes like ER_NO_REFERENCED_ROW_2 (foreign key constraint violation) to provide more informative error messages.
Response: Successful operations typically respond with a JSON object, often including a success message and the ID of the newly created or affected resource.
The server.js file shows the setup of the Express application, database connection, and the definition of these API routes for managing clients, projects, staff, resources, payments, tasks, and task assignments.

Source Code Repository

https://github.com/saif-kabeer/agency_ms.git

Conclusion

Completing this system deepened my understanding of full‑stack development. Designing the schema clarified normalisation and foreign‑key constraints; building REST endpoints strengthened my Express and SQL skills; integrating React hooks honed state‑management practices. Debugging, version control and faculty reviews also reinforced disciplined workflows. Overall, the project turned abstract database concepts into tangible, working solutions.

References