Report

Outline

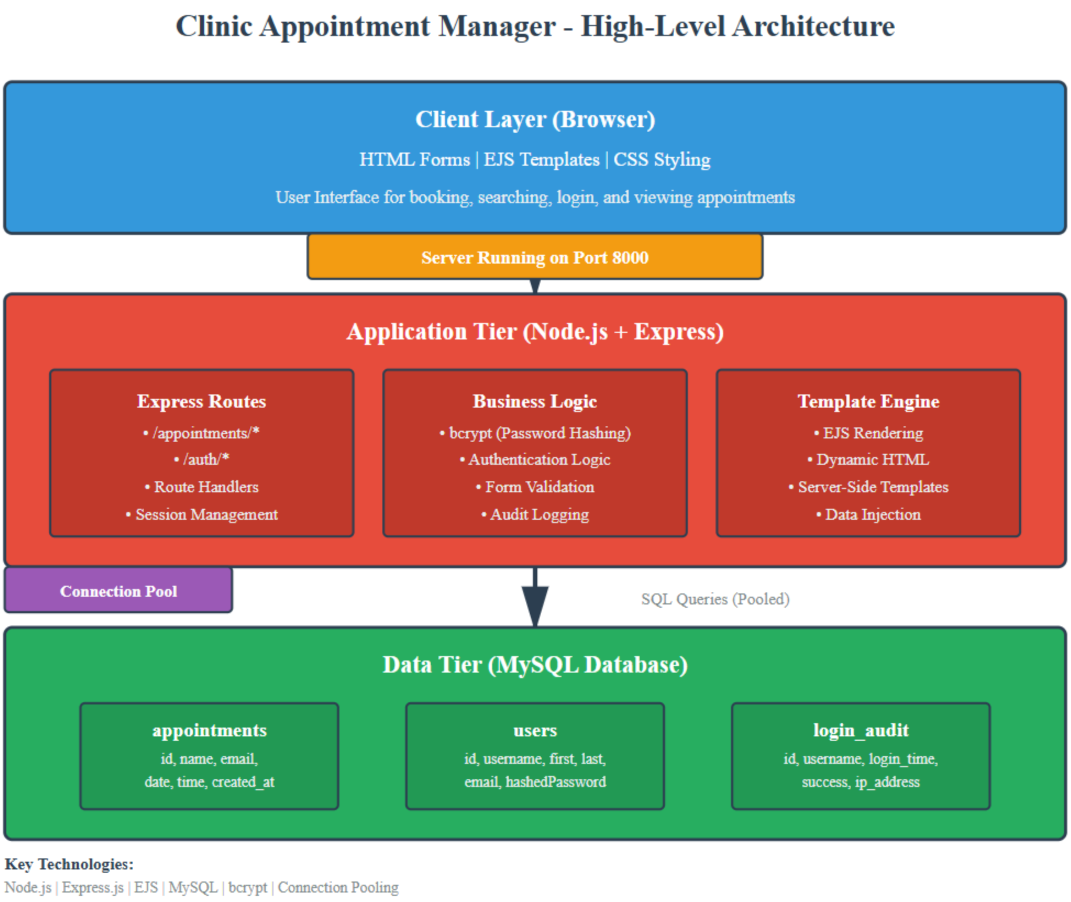
The Clinic Appointment Manager is a web application that allows patients to book, view, and search for appointments at a clinic. It also includes a user authentication system for staff members, tracking login attempts in an audit log. The application is built using Node.js with Express as the web framework, MySQL for the database, and EJS for server-side templates. Users can submit appointment requests via forms, while staff can manage users and view logs of login activity.

Architecture

Technologies & Components:

- Application tier: Node.js, Express, EJS templates, bcrypt for password hashing
- Data tier: MySQL database with tables for `appointments`, `users`, and `login_audit`

High-level Architecture Diagram:



Clinic Appointment Manager - High-Level Architecture
Client Layer (Browser)
HTML Forms | EJS Templates | CSS Styling
User Interface for booking, searching, login, and viewing appointments
Server Running on Port 8000
Application Tier (Node.js + Express)
Express Routes
• /appointments/*
• /auth/*
• Route Handlers
• Session Management
Business Logic
• bcrypt (Password Hashing)
• Authentication Logic
• Form Validation
• Audit Logging
Template Engine
• EJS Rendering
• Dynamic HTML
• Server-Side Templates
• Data Injection
Connection Pool
SQL Queries (Pooled)
Data Tier (MySQL Database)
appointments
id, name, email, date, time, created_at
users
id, username, first, last, email, hashedPassword
login_audit
id, username, login_time, success, ip_address
Key Technologies:
Node.js | Express.js | EJS | MySQL | bcrypt | Connection Pooling

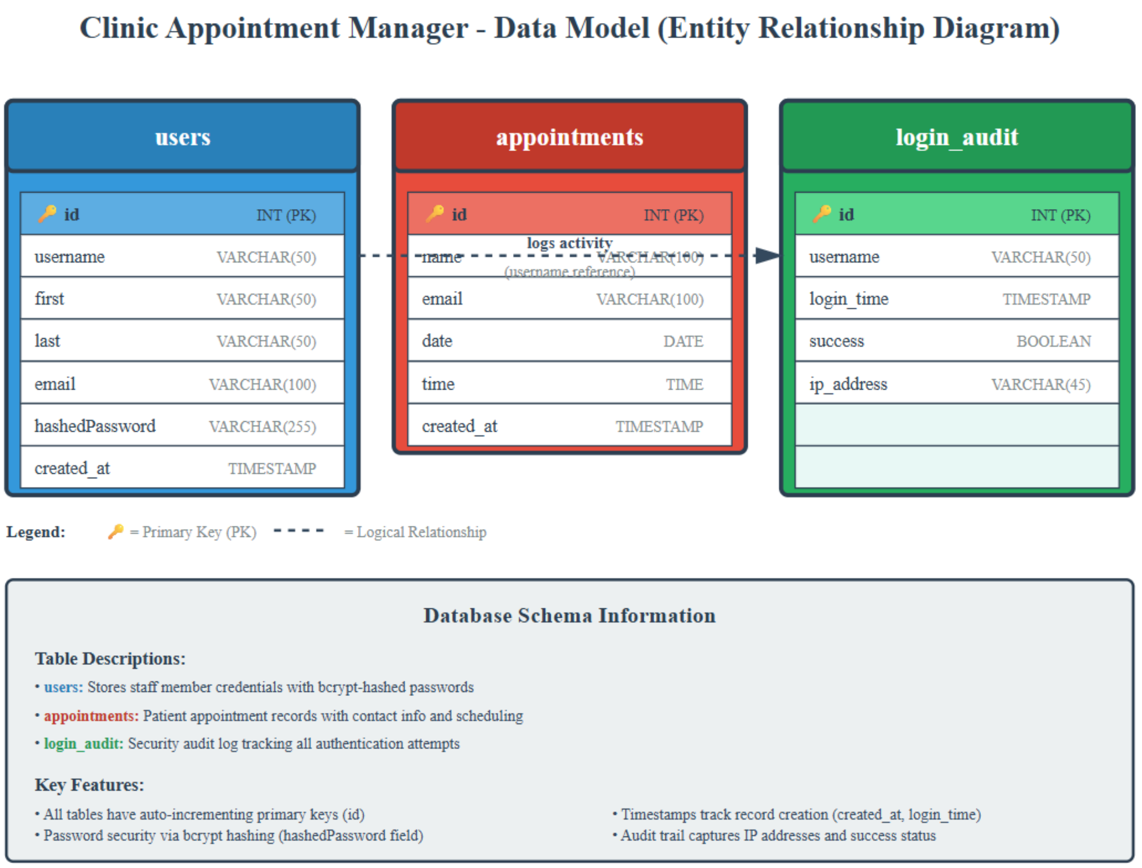
Data Model

The database has three main tables:

1. appointments – Stores patient appointments (id, name, email, date, time, created_at)

2. users – Stores staff user accounts (id, username, first, last, email, hashedPassword, created_at)
3. login_audit – Logs login attempts (id, username, login_time, success, ip_address)

<u>Data Model Diagram:</u>



Clinic Appointment Manager - Data Model (Entity Relationship Diagram)

users

🔑 id	INT (PK)
username	VARCHAR(50)
first	VARCHAR(50)
last	VARCHAR(50)
email	VARCHAR(100)
hashedPassword	VARCHAR(255)
created_at	TIMESTAMP

appointments

🔑 id	INT (PK)
name	VARCHAR(100)
email	VARCHAR(100)
date	DATE
time	TIME
created_at	TIMESTAMP

login_audit

🔑 id	INT (PK)
username	VARCHAR(50)
login_time	TIMESTAMP
success	BOOLEAN
ip_address	VARCHAR(45)

Legend: 🔑 = Primary Key (PK) - - - - = Logical Relationship

Database Schema Information

Table Descriptions:

- **users:** Stores staff member credentials with bcrypt-hashed passwords
- **appointments:** Patient appointment records with contact info and scheduling
- **login_audit:** Security audit log tracking all authentication attempts

Key Features:

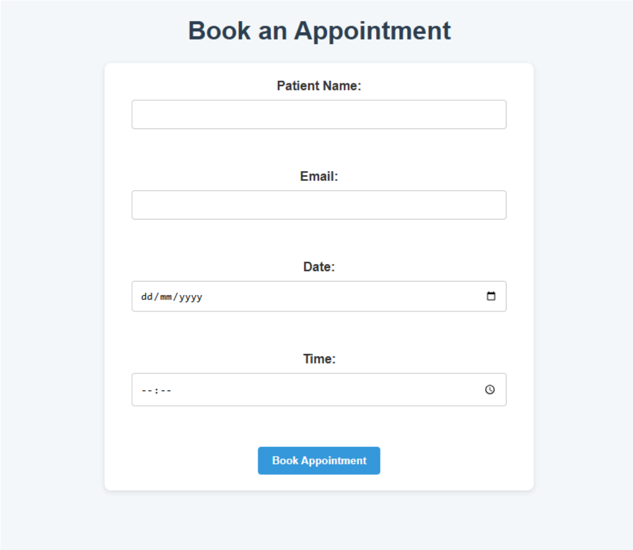
- All tables have auto-incrementing primary keys (id)
- Password security via bcrypt hashing (hashedPassword field)
- Timestamps track record creation (created_at, login_time)
- Audit trail captures IP addresses and success status

User Functionality

1. Book Appointment – Users fill a form (` /appointments/book`) with name, email, date, and time. On submission, the appointment is stored in the database.
2. List Appointments – Users or staff can view all appointments (` /appointments/list`) in a table sorted by date and time.
3. Search Appointments – Users can search for appointments by patient name (` /appointments/search`).
4. User Authentication – Staff can register (` /auth/register`) and log in (` /auth/login`). Passwords are hashed with bcrypt.
5. Audit Log – All login attempts are stored in ` login_audit` , accessible via ` /auth/audit` . This logs username, IP address, success status, and timestamp.

<u>Example Screenshot Descriptions:</u>

Booking form submission



Book an Appointment

Patient Name:

Email:

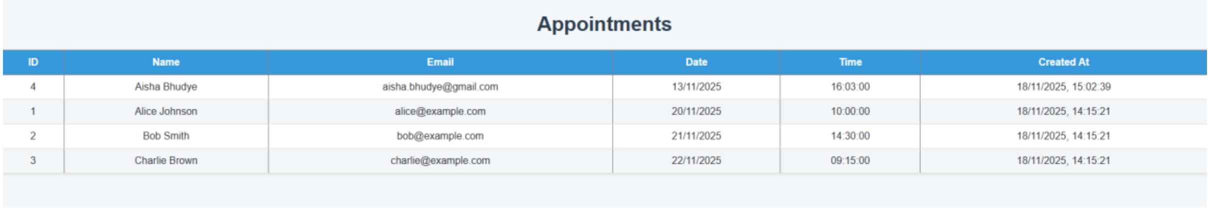
Date:

dd/mm/yyyy

Time:

--:--

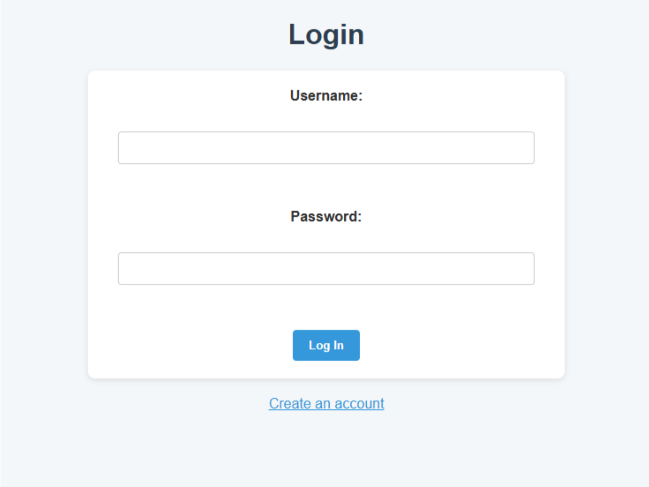
Appointment list table



Appointments

ID	Name	Email	Date	Time	Created At
4	Aisha Bhudye	aisha.bhudye@gmail.com	13/11/2025	16:03:00	18/11/2025, 15:02:39
1	Alice Johnson	alice@example.com	20/11/2025	10:00:00	18/11/2025, 14:15:21
2	Bob Smith	bob@example.com	21/11/2025	14:30:00	18/11/2025, 14:15:21
3	Charlie Brown	charlie@example.com	22/11/2025	09:15:00	18/11/2025, 14:15:21

Login page

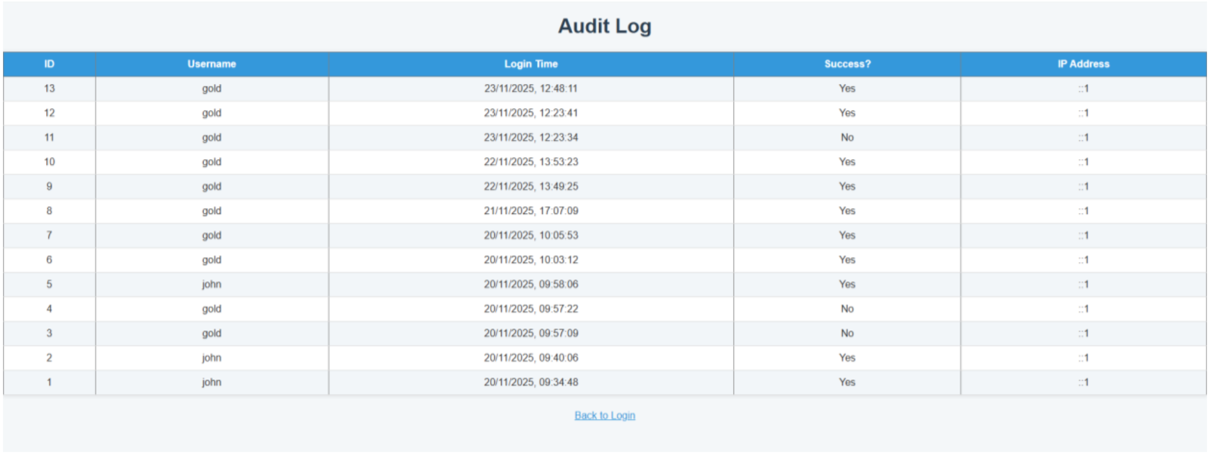


Login

Username:

Password:

Audit log showing successful and failed login attempts



Audit Log

ID	Username	Login Time	Success?	IP Address
13	gold	23/11/2025, 12:48:11	Yes	::1
12	gold	23/11/2025, 12:23:41	Yes	::1
11	gold	23/11/2025, 12:23:34	No	::1
10	gold	22/11/2025, 13:53:23	Yes	::1
9	gold	22/11/2025, 13:49:25	Yes	::1
8	gold	21/11/2025, 17:07:09	Yes	::1
7	gold	20/11/2025, 10:05:53	Yes	::1
6	gold	20/11/2025, 10:03:12	Yes	::1
5	john	20/11/2025, 09:58:06	Yes	::1
4	gold	20/11/2025, 09:57:22	No	::1
3	gold	20/11/2025, 09:57:09	No	::1
2	john	20/11/2025, 09:40:06	Yes	::1
1	john	20/11/2025, 09:34:48	Yes	::1

Advanced Techniques

1. Password Hashing with bcrypt– All passwords are stored securely using bcrypt. Example from `auth.js`:

js

```
const bcrypt = require('bcrypt');

const saltRounds = 10;

bcrypt.hash("smiths", saltRounds, (err, hash) => console.log(hash));
```

2. Database Connection Pooling – Using `mysql2.createPool` to manage multiple simultaneous connections efficiently in `index.js`:

js

```
const db = mysql.createPool({

  host: 'localhost',

  user: 'clinic_app',

  password: 'qwertyuiop',

  database: 'clinic_db',

  connectionLimit: 10

});

global.db = db;
```

3. Audit Logging – Every login attempt is logged with SQL insertions directly in the `/auth/loggedin` route:

js

```
db.query(

 "INSERT INTO login_audit (username, success, ip_address) VALUES (?, ?, ?)",
 [username, match, req.ip]
);
```

4. Dynamic Templating – EJS is used to render appointment lists and login pages dynamically with server-side data (`views/appointment_list.ejs`).