BookTable- A Cloud based Restaurant Reservation Application

Team Name: Code Catalysts

Team Members:

Sanjushree Golla - 017528433

Chanukya Vejandla - 018298215

Harsha Vardhan Badithaboina - 018280210

Jayanth Yarlagadda - 018291819

1. Introduction

1.1 Background & Motivation

The use of technology has increasingly changed business operations, especially in the hospitality business sector. Internet reservation systems have become part of the smooth functioning of restaurants because a customer can make a reservation, check restaurant details, and leave reviews. The growing interest in gaining access to digital solutions that will facilitate higher-level customer experience, operational efficiency improvement, and restaurant management has given rise to different reservation platforms. These systems automate manual work, offer more flexibility to customers to pick the timing of tables, a variety of table sizes, and special requests, all from the comfort of a device.

Motivation

The motive for implementing this reservation system is the inherent challenges posed to customers and restaurant managers. Customers are frequently frustrated by the inability to locate appropriate restaurants with available seats. Managers of restaurants are frustrated with their manual reservation management and overbooks, and poor optimization of seating arrangements.

1.2 Problem Statement

The reservation system is essentially paper based in most cases, and it is telephone based, off the street walking in, or even rudimentary software, which does not integrate well. Such methods lead to inefficiencies such as double or over-bookings or poor experience from poor management. Restaurant managers, in particular, may find it costly and cumbersome to keep a manual of this dynamically changing information.

1.3 Objectives of the Project

- To create an intuitive and simple interface for customers, whereby they can quickly browse, book, and find restaurant details and write reviews on responsive pages.
- To develop a responsive management system for restaurant managers to be able to monitor, alter, and refine reservations while examining customer preferences and feedback for adjustments to a restaurant operation.
- To deliver administrative tools to guide the management of user accounts, restaurant approval, and moderation of the customer review system to guarantee data accuracy and integrity of the system.

- To create a scalable backend architecture that can support high traffic, the real-time statuses of reservation need to be updated, and database consistency to be maintained on the interaction of different users.
- To incorporate secure ways of authenticating a user and protecting data while securing the user's access and safeguarding sensitive information, such as passwords and reservation information.

1.4 Target Users

The reservation system has had three major user constituencies that have well-defined roles and responsibilities.

Customers: System users, Customers, are likely to access the platform to search for restaurants, reserve dishes, view menus, identify restaurant details, & leave their reviews. They will benefit from an intuitive and responsive user interface that will enable easy navigation through options and also allow them to make good decisions.

Restaurant managers: Restaurant managers must be in control of the operational affairs of the restaurant, including control of reservations, customer demands, and availability of the restaurant (Punia *et al*., 2024). They will have access to an admin dashboard, which gives them a trend in reservations control and availability of tables and customer feedback, which will help them to optimize their restaurant performance and increase customer satisfaction.

Admins: The Admin will handle the general system, take care of user accounts (Customers and Restaurant manager) and accountability that restaurant data is correct, and moderate reviews. Admins will also have to approve listings and restaurant listings as well keep the system integrity.

1.5 Technologies Used

The development of the reservation system is dependent on the marriage of modern technologies for the frontend and backend activities. The central technologies applied for this project are:

Frontend technologies

- **React.js:** A strong JavaScript library for the development of user interfaces, mainly for single-page applications. React's component design architecture guarantees that React's components are modular, reusable, and easy to maintain.
- **CSS & Styled Components:** Both CSS and styled components are used to style the UI so as to have responsiveness and optimal viewing from all devices.

Backend technologies

- **Node.js:** Runtime environment for execution of JavaScript code on a server which allows for scalable and efficient backend services development.
- **Express.js:** Web application framework for Node.js to facilitate the development of sturdy APIs to process requests, data fetch, and user logins.
- **MySQL:** A relational database management system in which to store user, restaurants, reservations, reviews, and other system component data. Structured query language (SQL) used in MySQL executes the operations in the database.

2. System Requirements & Functional Specifications

2.1 Functional Requirements

Table 1: Functional Requirements for Customer

ID	Functionality	Description
FR1	User Registration	Customers should be able to create a new account by providing the necessary details such as email, password, and name.
FR2	User Login	Customers should be able to log in using their email and password.
FR3	Browse Restaurants	Customers should be able to view a list of restaurants, including their details, photos, and ratings.
FR4	Search Restaurants	Customers should be able to search for restaurants based on location, cuisine type, or rating.
FR5	Make Reservations	Customers should be able to select a restaurant, table, date, and time, and make a reservation.
FR6	View Reservation Details	Customers should be able to view and manage their existing reservations.

FR7	Cancel Reservations	Customers should be able to cancel an existing reservation.
FR8	Leave Reviews	Customers should be able to write reviews for restaurants they visited, including a rating and comments.
FR9	Update Profile	Customers should be able to update their personal information, such as email and phone number.
FR10	View Reservation History	Customers should be able to view their past reservation history.

Table 2: Functional Requirements for Restaurant Manager

ID	Functionality	Description
FR1	Manager Login	Restaurant managers should be able to log in using their credentials (email and password).
FR2	View and Manage Restaurant Information	Managers should be able to update restaurant details such as name, description, contact information, and operating hours.
FR3	Approve/Reject Restaurant Listing	Managers should have the ability to approve or reject a new restaurant listing before it goes live.
FR4	Manage Reservations	Managers should be able to view, modify, or cancel reservations made by customers for their restaurant.
FR5	Manage Tables	Managers should be able to add, remove, or update tables available in the restaurant, including table numbers and capacities.

FR6	Monitor Customer Feedback	Managers should be able to view and respond to customer reviews to enhance service quality.
FR7	View Reservation Statistics	Managers should be able to analyse reservation trends and statistics such as most booked times, party sizes, etc.
FR8	Set Operating Hours	Managers should be able to update the restaurant’s working hours, including holidays and special timings.
FR9	Update Restaurant Photos	Managers should be able to upload or remove restaurant photos, including primary and secondary images.
FR10	Access Reporting Tools	Managers should have access to reporting tools that provide insights into business performance, including occupancy rates.

Table 3: Functional Requirements for Admin

ID	Functionality	Description
FR1	Admin Login	Admins should be able to log in using their credentials (email and password).
FR2	Manage User Accounts	Admins should be able to create, update, and delete customer, manager, and restaurant manager accounts.
FR3	Manage Restaurant Listings	Admins should have the ability to add, modify, or remove restaurants from the platform.

FR4	Approve/Reject Restaurant Listings	Admins should be able to approve or reject restaurant listings submitted by managers or owners.
FR5	Monitor User Activity	Admins should be able to track and monitor user activity on the platform, including login times and actions.
FR6	Review and Manage Customer Feedback	Admins should be able to moderate, approve, or remove customer reviews and ratings.
FR7	Assign Roles and Permissions	Admins should be able to assign or update roles (e.g., customer, restaurant manager, admin) and permissions.
FR8	Access All Reservations	Admins should have access to all reservations made by customers across all restaurants.
FR9	Generate Reports	Admins should be able to generate and download various reports, such as user activity, reservations, and restaurant performance.
FR10	Manage System Settings	Admins should be able to configure and manage system-wide settings such as platform features and user access controls.

2.2 Non-functional Requirements

Table 4: Non-functional Requirements

ID	Requirement	Description
NFR1	Scalability	The system should be able to handle an increasing number of users (both customers and managers) without performance degradation.
NFR2	Security	The system must ensure the confidentiality, integrity, and availability of user data.
NFR3	Performance	The system should respond to user actions (such as reservations, search queries, and logins) within 2 seconds.
NFR4	Availability	The system should have 99.9% uptime, ensuring that it is always accessible for users.
NFR5	Backup and Disaster Recovery	Data backups should be performed regularly to ensure system recovery in case of failure.
NFR6	Usability	The system should provide a user-friendly interface for all user types (customers, managers, and admins).
NFR7	Maintainability	The system's codebase should be modular and well-documented, allowing for easy maintenance, bug fixes, and updates.
NFR8	Compatibility	The system should be compatible with all major browsers (Chrome, Firefox, Safari, Edge) and support different operating systems (Windows, macOS, Linux) to ensure that users have a consistent experience (Zhang *et al.*, 2022).

NFR9	Accessibility	The system should be accessible to users with disabilities, adhering to the WCAG 2.1 guidelines for accessibility.
NFR10	Compliance	The system should comply with relevant data privacy laws such as GDPR or CCPA.
NFR11	Load Handling	The system should be able to handle at least 1,000 concurrent users without any noticeable performance degradation.
NFR12	Localization and Internationalization	The system should support multiple languages and regional settings (such as time zones and currencies) to cater to a global audience.
NFR13	Auditability	The system should maintain logs of all significant actions (e.g., user logins, reservations made, reviews posted) for auditing purposes (D'Agati *et al*., 2024).
NFR14	Session Management	The system should implement secure session management to ensure that users are logged out after a period of inactivity (e.g., 15 minutes).
NFR15	Error Handling	The system should provide clear and user-friendly error messages for users.

3. System Architecture

3.1 High-Level Architecture Overview

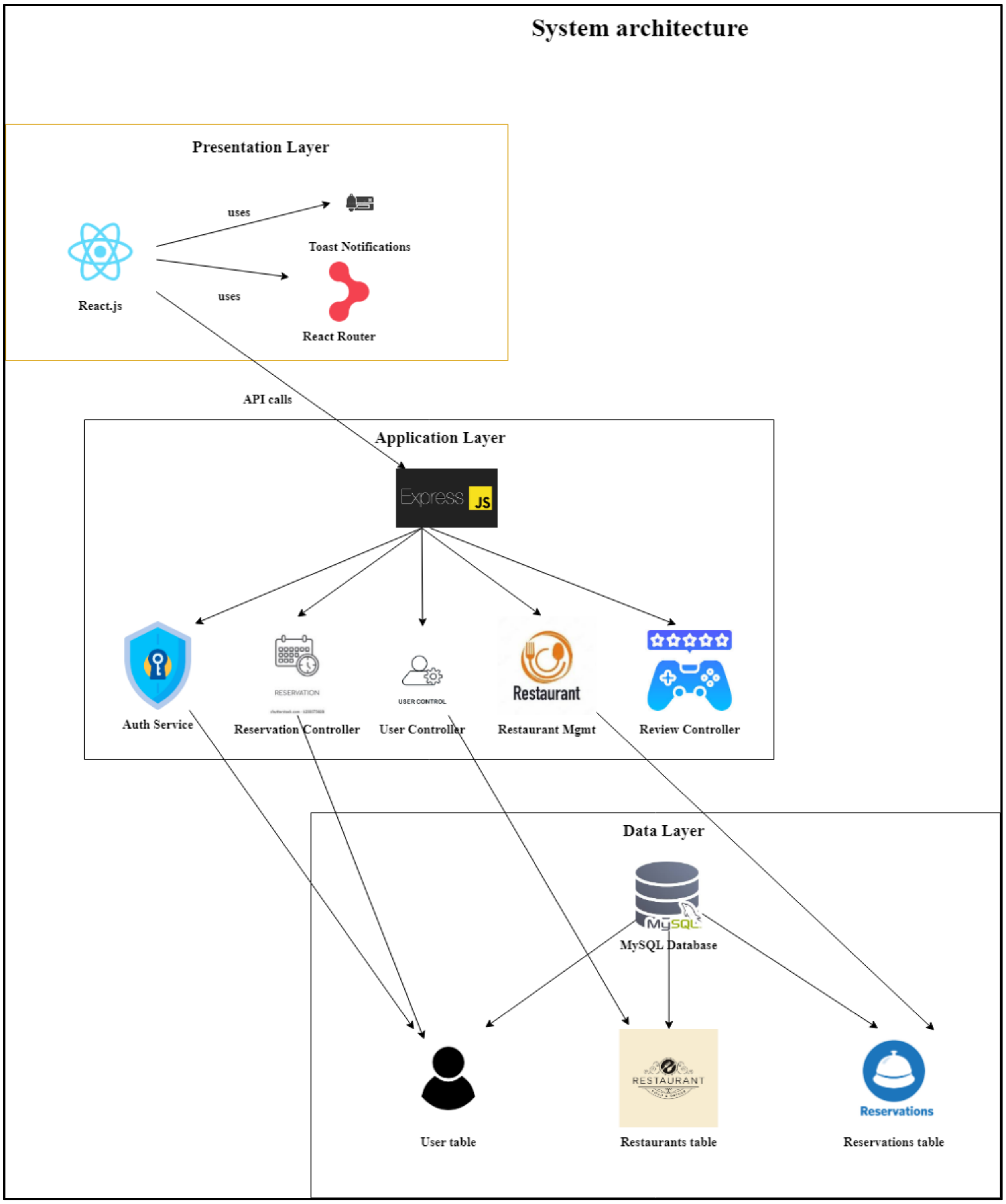


Figure 1: System architecture

The system architecture diagram illustrates the reservation platform's overall structure and interaction between various layers. It also shows the stream of data and requests from the user interfaces to the frontend, backend services, and the database. The architecture is described as a 3-tier model. presentation (browser/React.js), application (Node.js API), and data (MySQL database). This design prevents a messy tangle of responsibilities and allows for updates on separate tiers to be performed independently. RESTful communication patterns, robust security approach, and responsive interfaces are also supported by the architecture (Singh *et al*., 2021). In general, it guarantees a coherent system, has high performance, and is adaptable for further improvements and integration.

3.2 Component Diagram (UML)

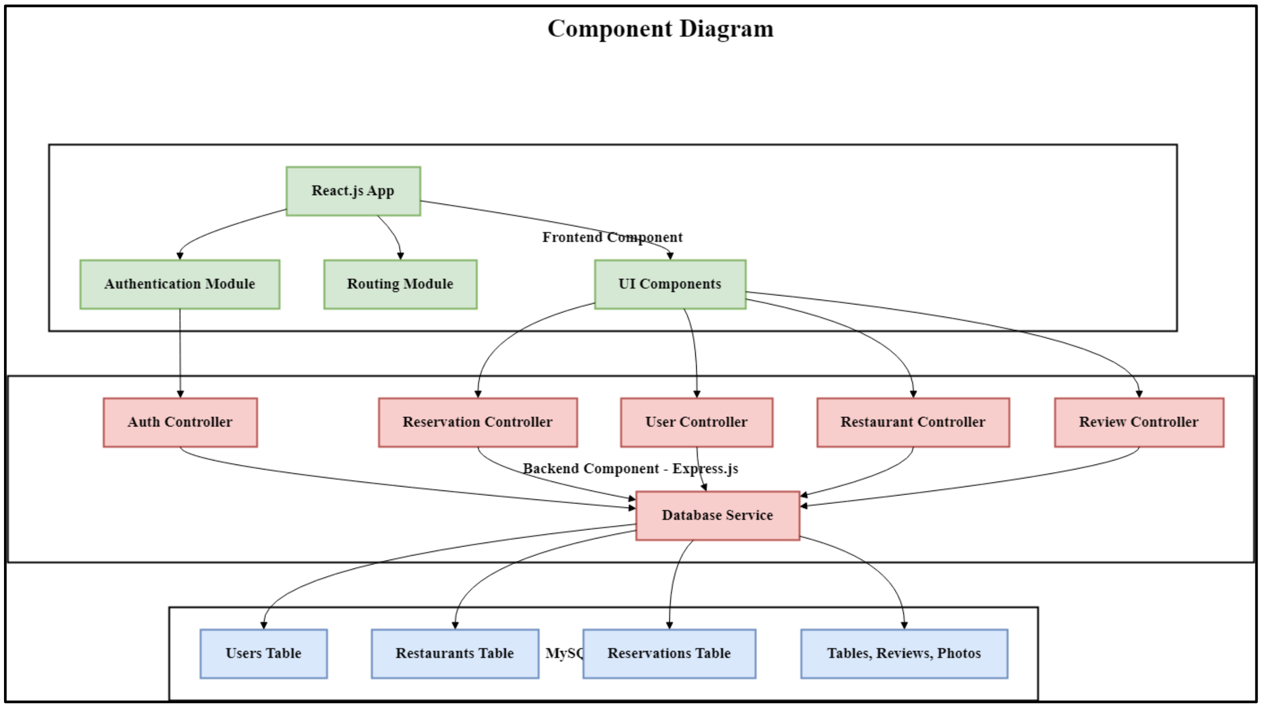


Figure 2: Component diagram

The component diagram shows how the restaurant reservation system is modular. It minimizes a single location where concerns are tangled by separating concerns such as the user interface (frontend), backend API services, and database layer. Each significant functionality, such as authentication, reservation management, and restaurant operation, is implemented as an individual component to allow maintainability and scalability. The diagram demonstrates which communication method is utilized for the frontend and backend to communicate through APIs and how these, in turn, access and modify the relational database's data (Henry *et al*., 2023). Such abstracting facilitates code reuse, supports the debugging process, and is beneficial for collaboration within a team during the process of system development and testing.

3.3 Deployment Diagram (UML)

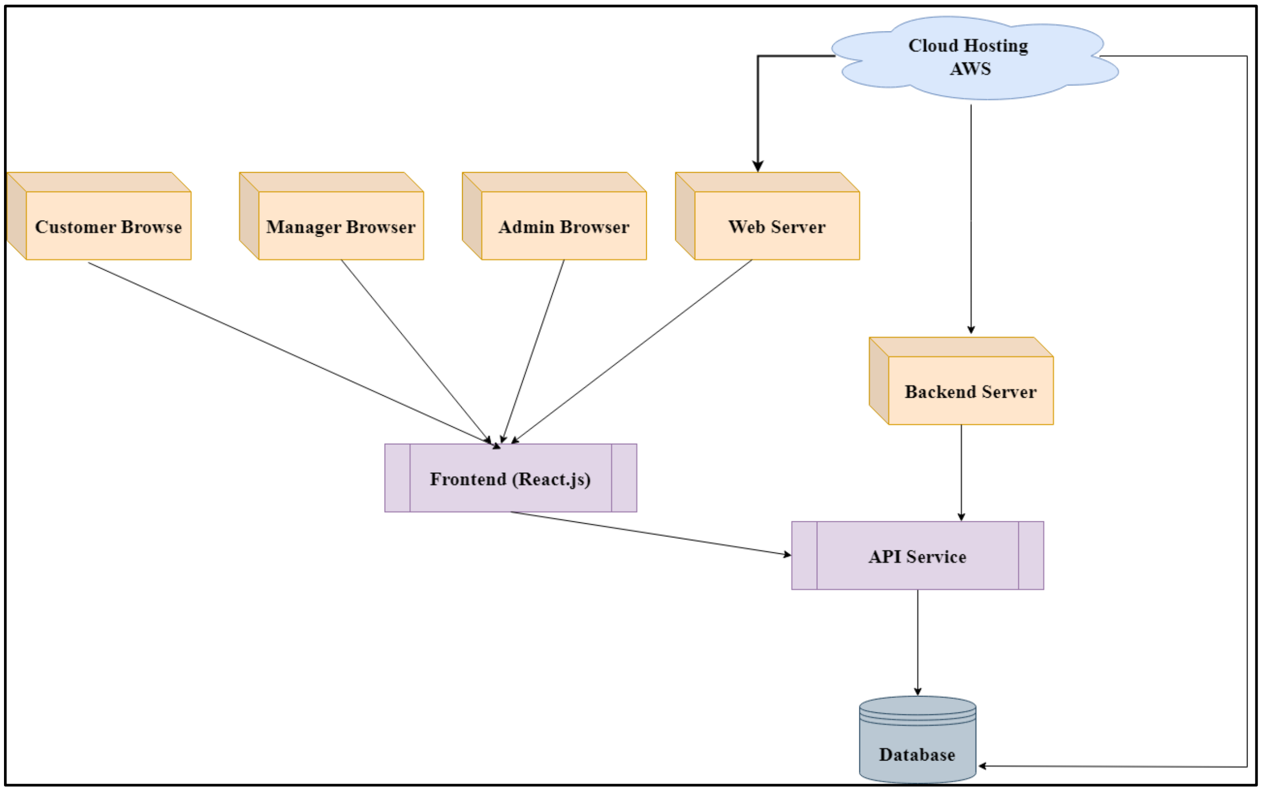


Figure 3: Deployment diagram

The deployment diagram gives a big picture of the physical architecture of the system on how it deploys the software artifacts to different nodes. It also identifies components such as frontend (React.js) and backend (Node.js/Express), and placements in host environments, such as cloud platforms (e.g,. AWS). Browsers that connect to the web server and backend services are used to interact with the system by user roles such as customers, managers, and administrators. The database stays in an environment where security is guaranteed, which maintains the integrity of data and access control (Deep *et al.*, 2023). This setup provides availability of the system, fault tolerance, and the ability to dynamically scale out the resources based on user demand.

4. User Interface Design

4.1 Purpose of UI for each role

The reservation system makes restaurant management easier by offering an intuitive platform to customers, managers, and admins. It has efficient reservation management, real time updates and smooth user experiences from different roles.

Customer

The UI of customers acts as the main interface to the reservation system. It is meant to provide a smooth and simple-to-use application where the customer can search for restaurants, check tables available, make a booking, and read restaurant offerings (Rahman *et al*., 2023). The ease of use should be the priority in the design so that customers can easily browse options by cuisine, location, and availability.

Restaurant Manager

The UI should allow ease of management in running restaurant operations for restaurant managers. It enables them to enable them to update information about restaurant information, reserve reservations, and tables, and monitor customer reviews. The UI should support an effective interface for editing operating hours, adding or deleting tables, and approval of or rejection of reservation requests.

Admin

The admin UI functions like the hub control panel to manage the whole system. It should enable admins to control users, approve/reject restaurant listings control all restaurant activities. The platform's UI needs to implement administrative functions for tracking system performance, resolving disputes, and maintaining data integrity across all parts of the platform (Kertész *et al*., 2021).

4.2 Attach UI Wireframes

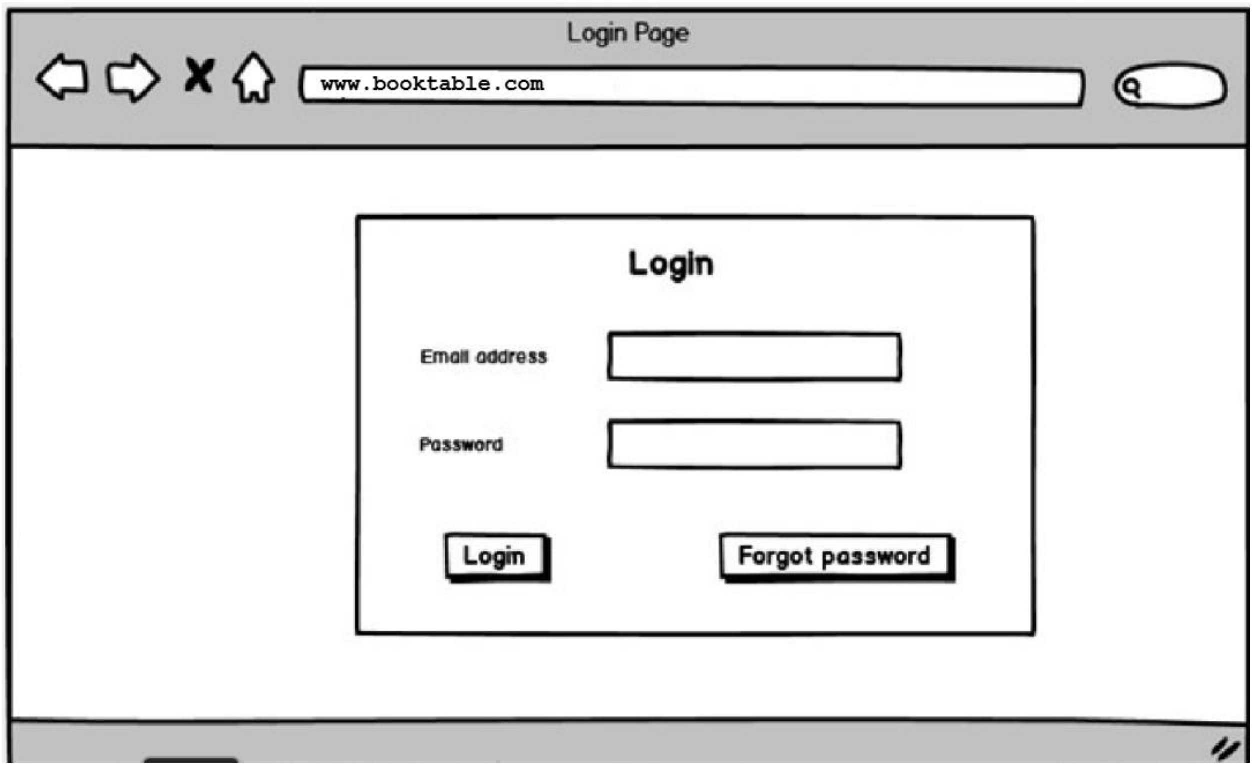


Figure 4: Login page

The secure access point for all users, customers, managers, and admins is the login page. It allows users to enter their email and password using fields, and has essential features such as “Forgot Password” and “Sign Up”. The usability and accessibility are given primary importance behind the page, which allows a hassle-free process of authenticating to the page. Its neat structure guarantees that users can easily log in without getting confused. Security best practices, including password masking and form validation, are incorporated to act as a shield to user credentials and to regulate proper access control.

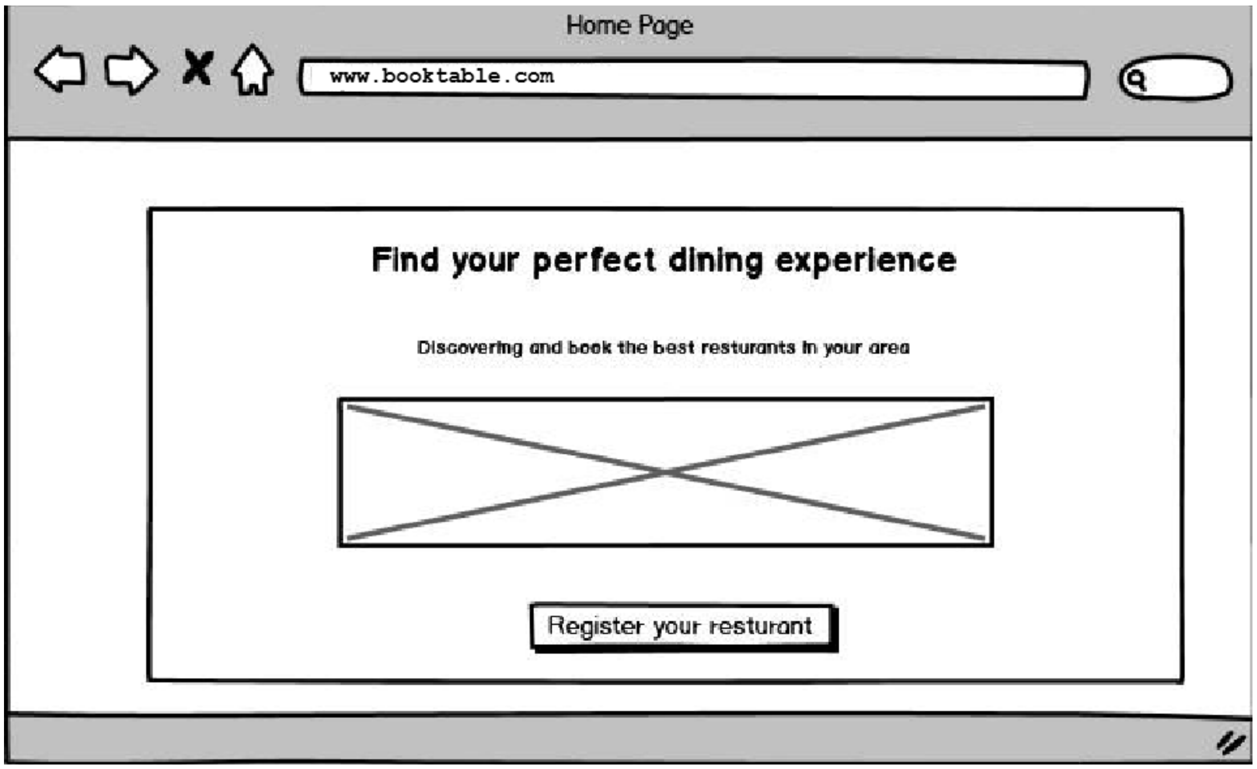

Home Page
www.booktable.com
Find your perfect dining experience
Discovering and book the best resturants in your area
Register your resturant

Figure 5: Home page

An intuitive customer home page is configured for customer interaction to search restaurants and tables, and book tables easily. A navigation bar is convenient for having quick access to account settings, with the possibility to log out. The user interface focuses on simplicity, responsiveness, and making the user experience uniform from any device. Such personalized elements like reservation history and recommended restaurant(s) improve engagement too. The layout enhances efficiency, directing customers through reservation protocols with the least possible steps, providing real-time feedback from the backend.

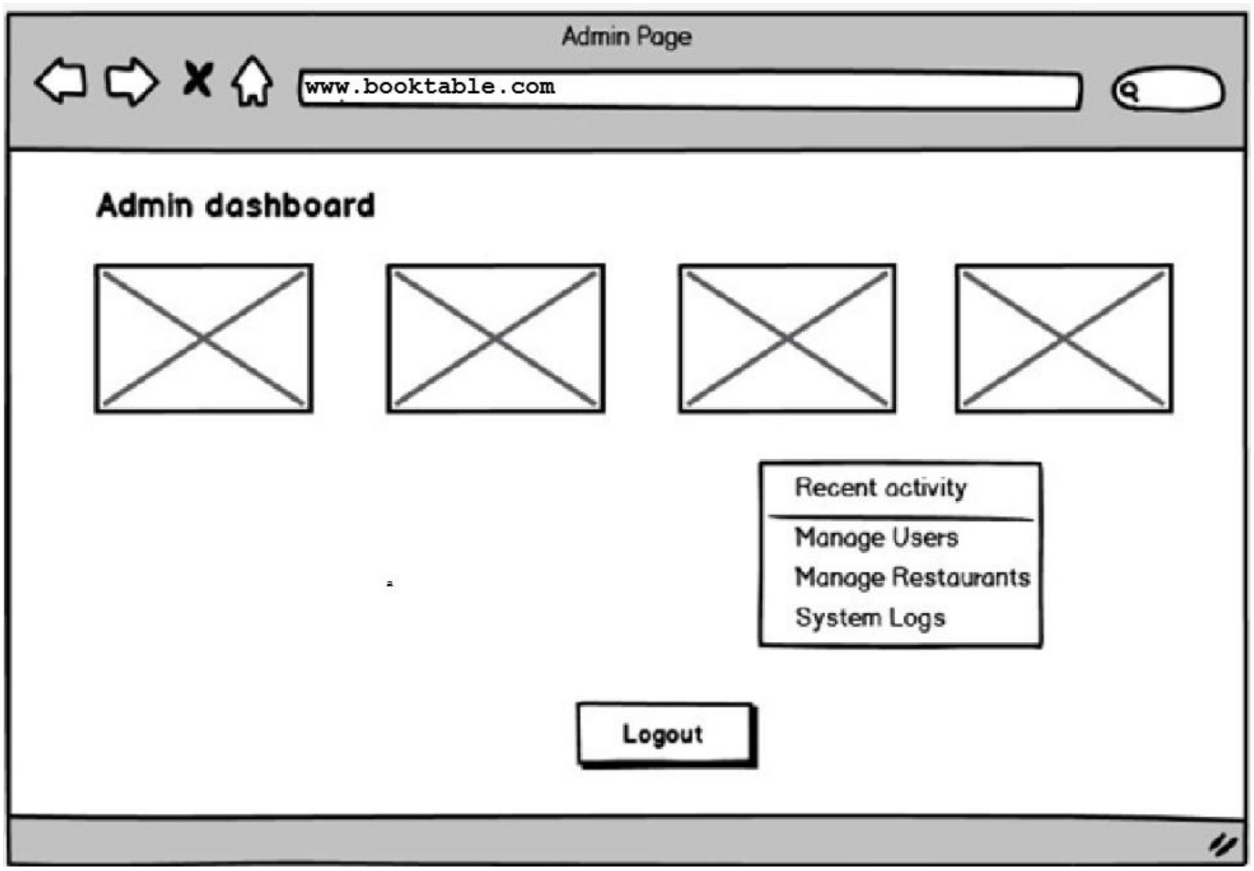


Figure 6: Admin page

The admin home page provides a single point that enables easy monitoring and management at the system but all levels. It incorporates widgets for user analytics, restaurant statistics, and system logs. Admins can have control over users, approve or block out restaurants, and run the operations of the platform. Design emphasizes clarity, control, as administrators can easily spot problems or anomalies instantly. Security aspects are made to prevent sensitive functionalities from being accessed by unauthorized users. The interface is responsive and scalable, which fits a growing platform without loss of operational oversight.

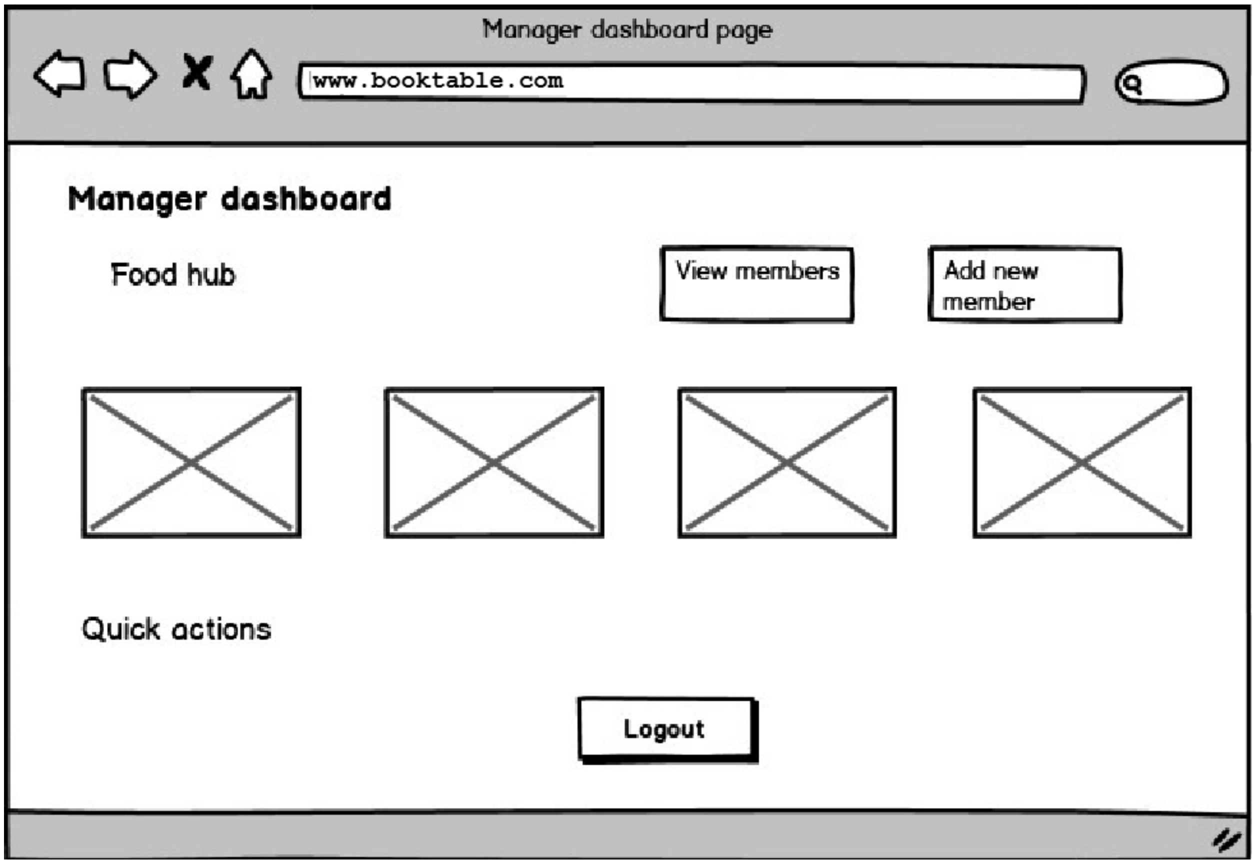
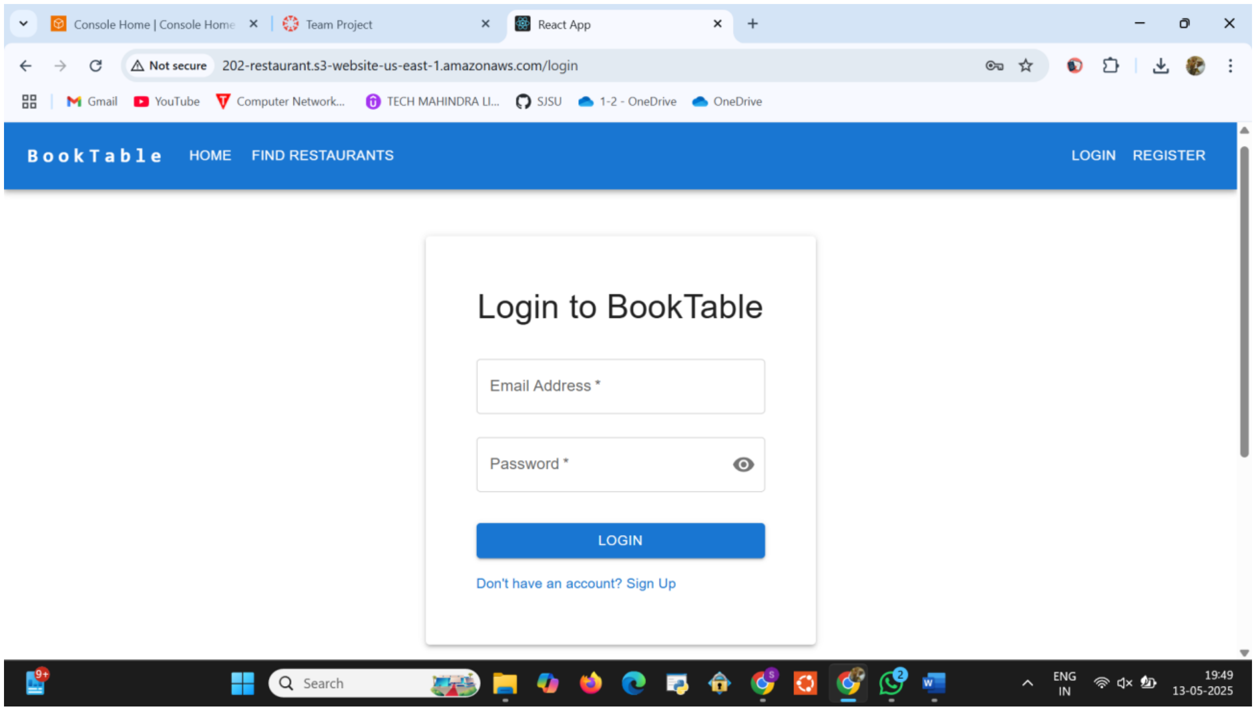


Figure 7: Manager dashboard page

The manager's home page equips restaurant owners with tools to manage their establishment's bookings, availability, and feedback from customers. It provides reservation data, editable restaurant profiles, and scheduling controls. Managers can update open slots, answer customer reviews, and manage internal notifications. The design is user-oriented, so that efficient task completion is possible without deep system navigation. It makes sure that functionalities that are specific to the restaurant are readily available, making day-to-day operations smooth. The UI provides performance tracking, assisting managers to enhance the quality of services by observing real-time interactions from customers.

4.3 Describe major screens and navigation


Console Home | Console Home
Team Project
React App
Not secure 202-restaurant.s3-website-us-east-1.amazonaws.com/login
Gmail YouTube Computer Network... TECH MAHINDRA LI... SJSU 1-2 - OneDrive OneDrive

BookTable HOME FIND RESTAURANTS LOGIN REGISTER

Login to BookTable

Email Address *

Password *

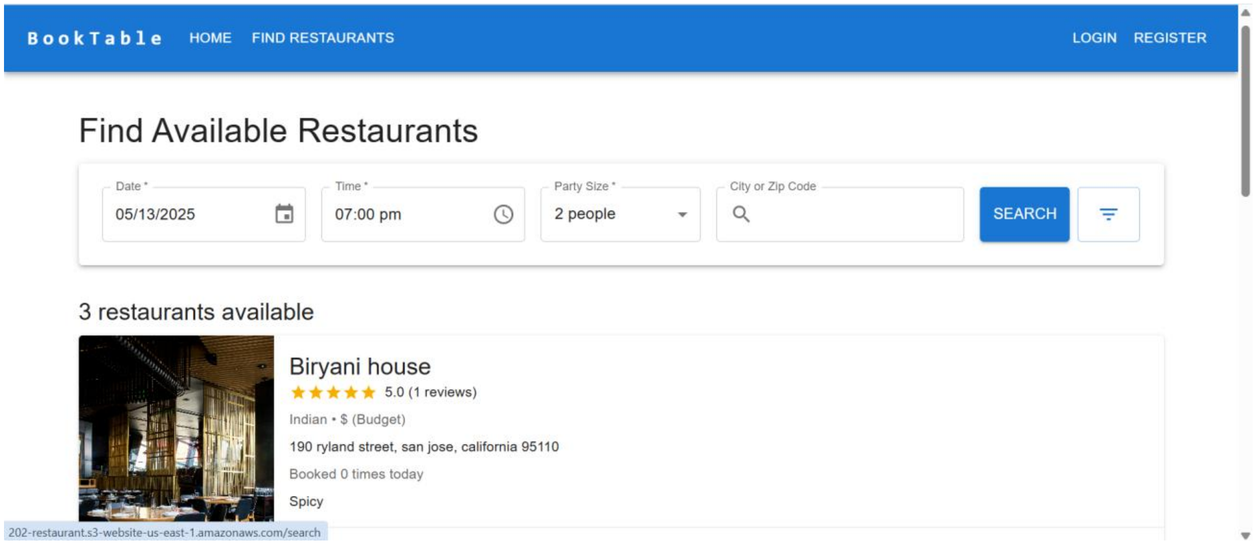
LOGIN

Don't have an account? Sign Up

Search ENG IN 19:49 13-05-2025

Figure 8: Login page

The login page includes a slim login form consisting of email and password fields, a login button, and a sign-up link. The blue header is gone with the DineEase logo in front of it, with HOME and FIND RESTAURANTS navigation, and LOGIN/REGISTER options. The footer has the company description, quick links, contact information and also social media icons.


BookTable HOME FIND RESTAURANTS LOGIN REGISTER

Find Available Restaurants

Date * 05/13/2025
Time * 07:00 pm
Party Size * 2 people
City or Zip Code
SEARCH

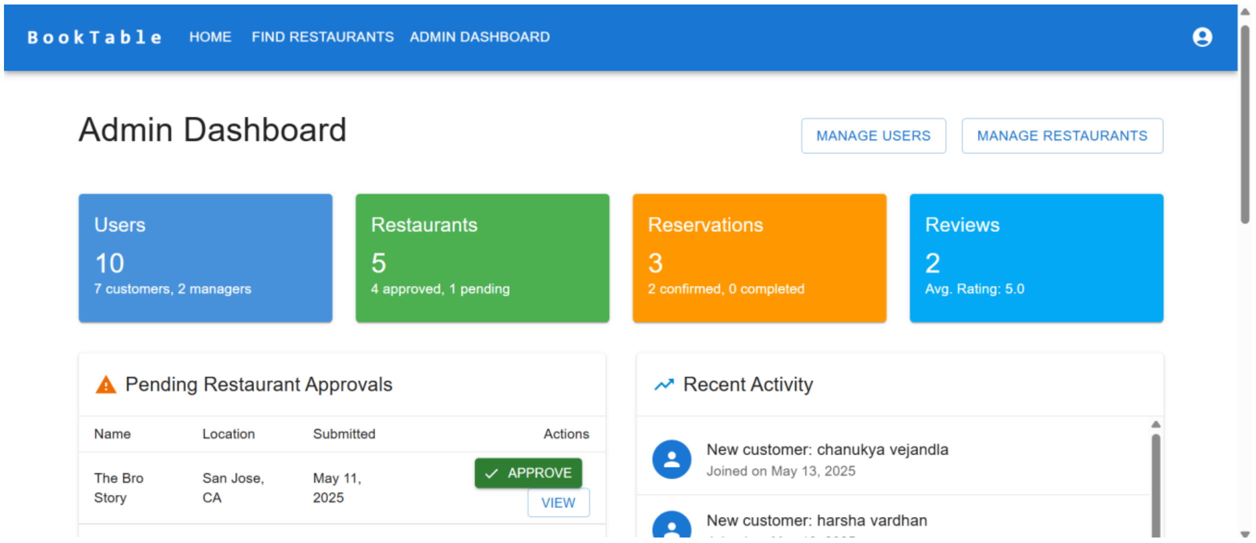
3 restaurants available

Biryani house
★★★★★ 5.0 (1 reviews)
Indian • $ (Budget)
190 ryland street, san jose, california 95110
Booked 0 times today
Spicy

202-restaurant.s3-website-us-east-1.amazonaws.com/search

Figure 9: Home page

Features a search for restaurants in terms of location, date, time, and number of members in the party. Below, they have sections for featured restaurants and the platform is described in a three-step process. It has a call to action for restaurant owners and the same footer as the login page.



Admin Dashboard

Users
10
7 customers, 2 managers

Restaurants
5
4 approved, 1 pending

Reservations
3
2 confirmed, 0 completed

Reviews
2
Avg. Rating: 5.0

Pending Restaurant Approvals

Name	Location	Submitted	Actions
The Bro Story	San Jose, CA	May 11, 2025	APPROVE VIEW

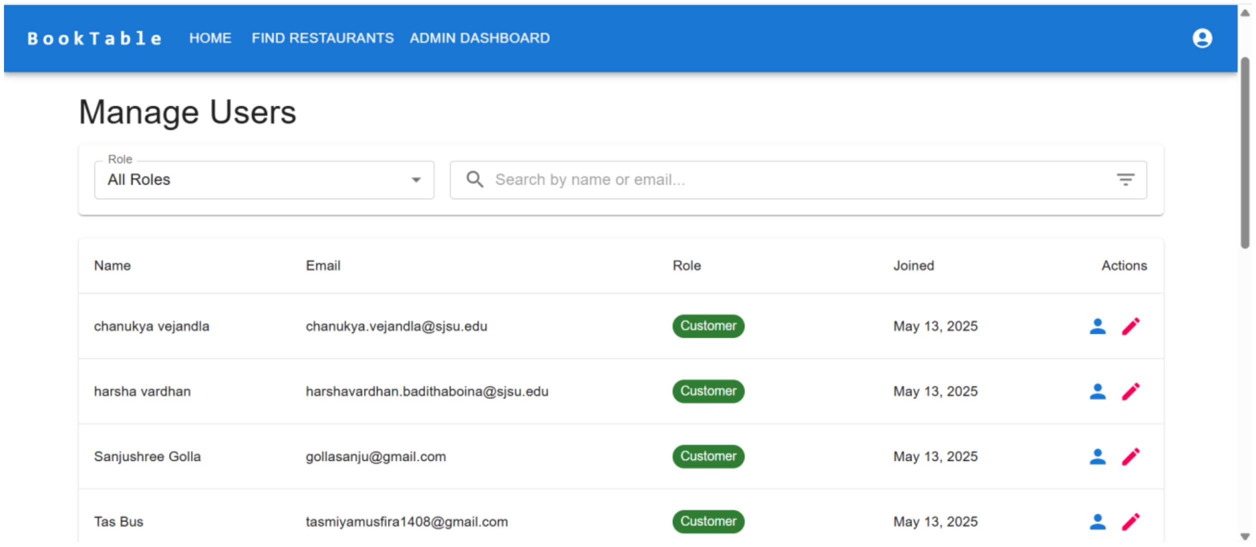
Recent Activity

New customer: chanukya vejandla
Joined on May 13, 2025

New customer: harsha vardhan

Figure 10: Admin dashboard

The admin dashboards show statistics about users, restaurants, reservations and reviews within color-coded cards. It indicates restaurant approvals pending, activity log from recent times and reservation trends. The interface has management buttons and navigation choices for administrators.



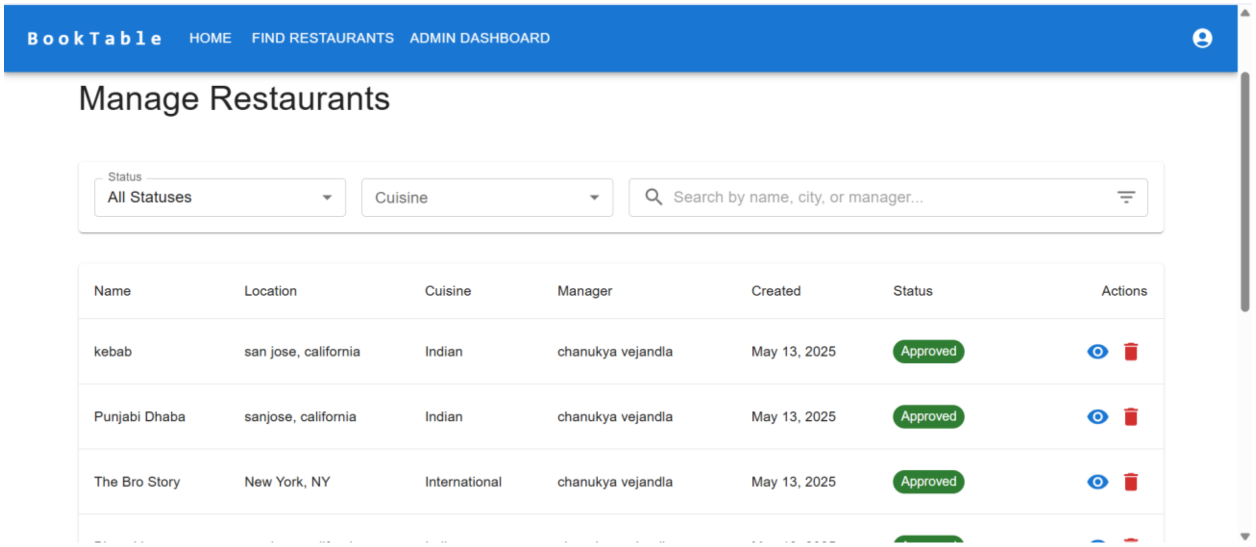
Manage Users

Role: All Roles
Search by name or email...

Name	Email	Role	Joined	Actions
chanukya vejandla	chanukya.vejandla@sjsu.edu	Customer	May 13, 2025	
harsha vardhan	harshavardhan.badithaboina@sjsu.edu	Customer	May 13, 2025	
Sanjushree Golla	gollasanju@gmail.com	Customer	May 13, 2025	
Tas Bus	tasmiyamusfira1408@gmail.com	Customer	May 13, 2025	

Figure 11: Admin user management page

The admin user management page displays a list of all users having name, email, role, joining date, and action buttons. There's also a search function and a "filter by role" drop-down menu for the users. There are pagination controls and the same footer elements as in other pages on the page.



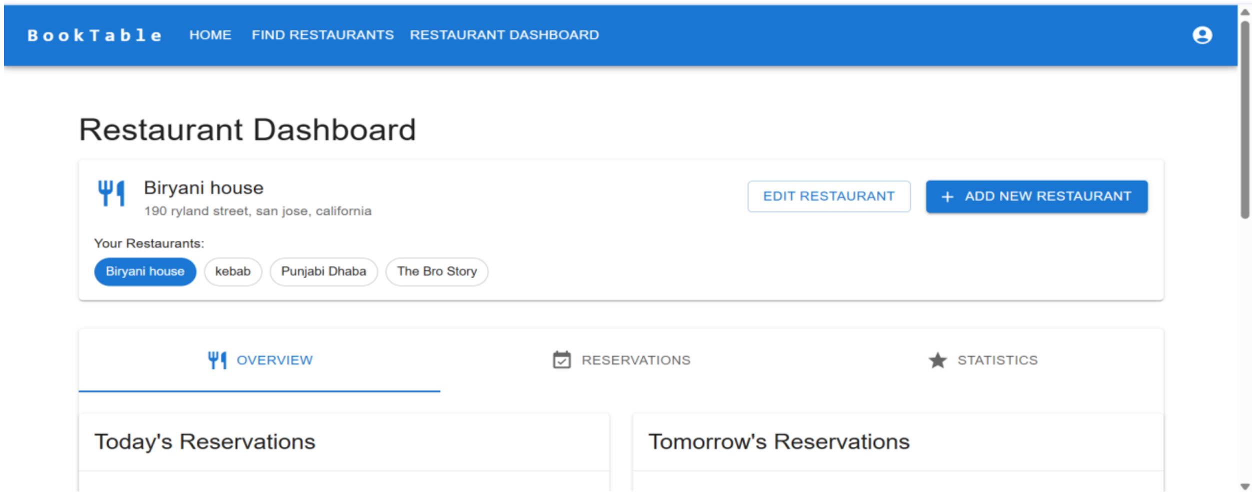
Manage Restaurants

Status: All Statuses | Cuisine | Search by name, city, or manager...

Name	Location	Cuisine	Manager	Created	Status	Actions
kebab	san jose, california	Indian	chanukya vejandla	May 13, 2025	Approved	
Punjabi Dhaba	sanjose, california	Indian	chanukya vejandla	May 13, 2025	Approved	
The Bro Story	New York, NY	International	chanukya vejandla	May 13, 2025	Approved	

Figure 12: Admin Restaurant management page

The admin Restaurant management page displays a table of registered restaurants with selections available for statuses and cuisine types. A restaurant, "Foody Hub," is also listed with pending status with its current location, cuisine, manager, and date of creation. Action buttons enable admins to see, approve, or delete restaurants. The page has pagination controls and also retains the standard DineEase header and footer elements.



Restaurant Dashboard

Biryani house
190 ryland street, san jose, california

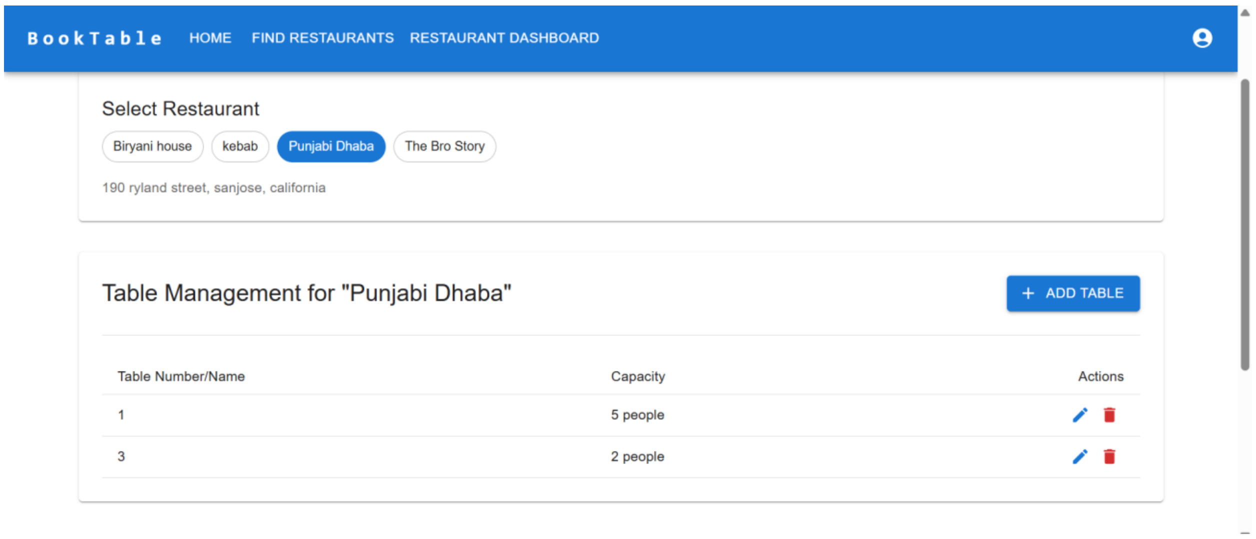
Your Restaurants: Biryani house, kebab, Punjabi Dhaba, The Bro Story

Today's Reservations

Tomorrow's Reservations

Figure 13: Manager dashboard page

The manager dashboard page delivers to restaurant managers a summary of their restaurant's activity. In the dashboard of "Foody Hub," there is no current booking, but figures for booking today, next reserve, tables, and approval status. Navigation tabs consist of Overview, Reservations, and Statistics. Quick action buttons help managers to view the public page, reservations and configure table settings.



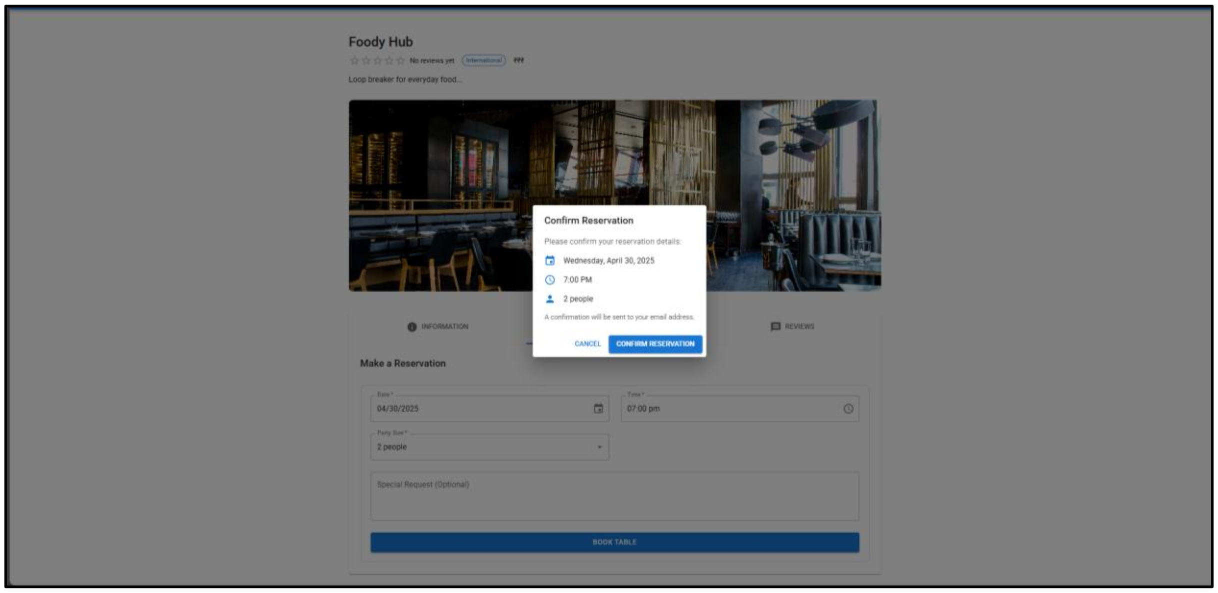
190 ryland street, sanjose, california

Table Management for "Punjabi Dhaba"

Table Number/Name	Capacity	Actions
1	5 people	
3	2 people	

Figure 14: Manager restaurant table management page

Manager restaurant table management page enables restaurant managers to build a seating pattern. A modal dialog displays how to add a new table with the choice to specify the table number and capacity (currently 4 people). A success message tells us a table was added recently, and the past table configuration for "Foody Hub" is in the background.


Foody Hub

No reviews yet International ₹₹₹

Loop breaker for everyday food...

Confirm Reservation

Please confirm your reservation details:

Wednesday, April 30, 2025

7:00 PM

2 people

A confirmation will be sent to your email address.

Make a Reservation

Date * 04/30/2025

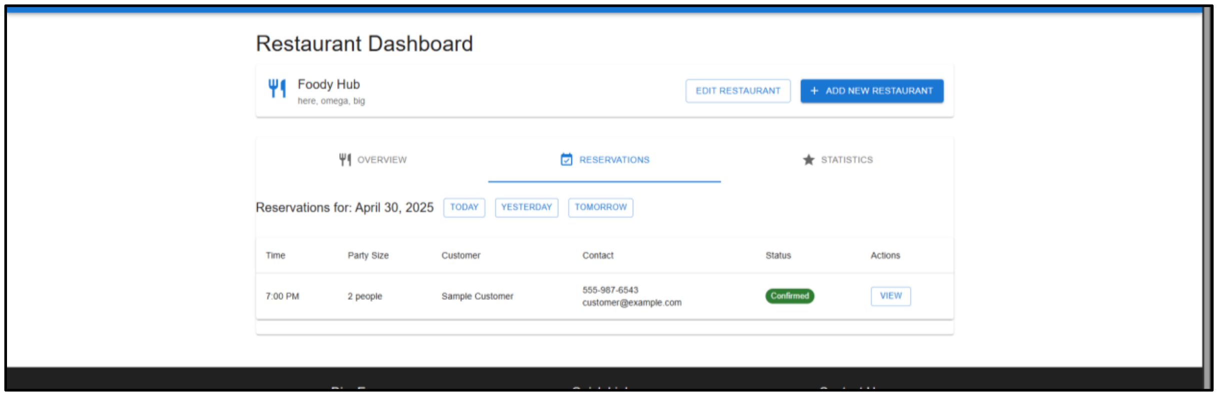
Time * 07:00 pm

Party Size * 2 people

Special Request (Optional)

Figure 15: Restaurant booking page

The restaurant booking page demonstrates the customer-facing reservation interface of “Foody Hub” restaurant. A dialog confirmation is in the centre encouraging users to confirm the details for April 20, 2025 Wednesday at 7:00 PM for 2 people. The hidden page shows the restaurant description, score, a reservation form with options for date, time and party size selectors as well as a special request field.


Restaurant Dashboard

Foody Hub

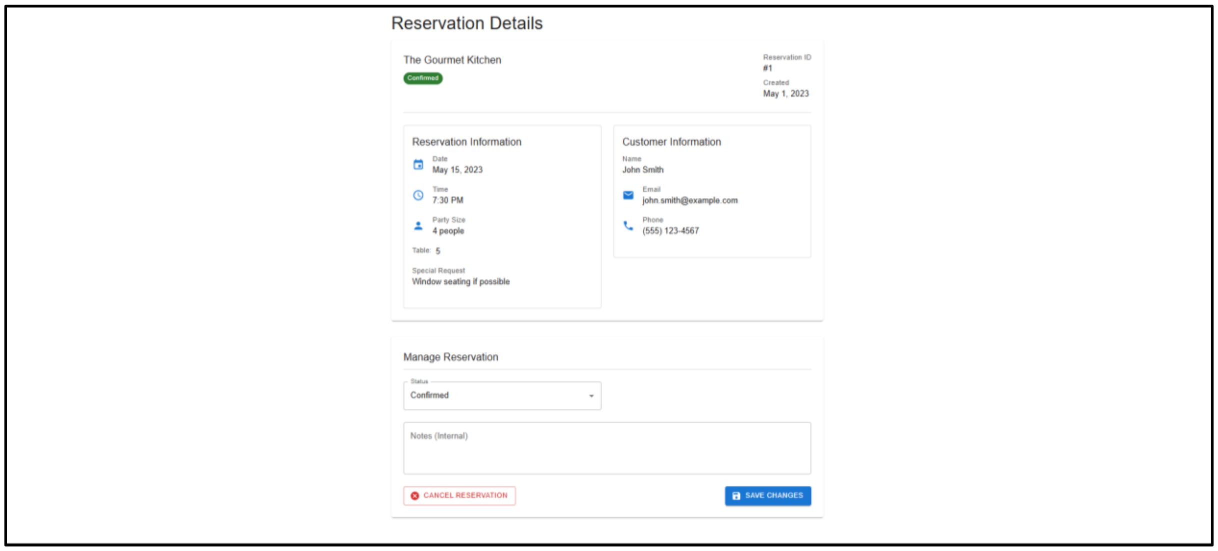
here, omega, big

Reservations for: April 30, 2025

Time	Party Size	Customer	Contact	Status	Actions
7:00 PM	2 people	Sample Customer	555-987-6543 customer@example.com	Confirmed	VIEW

Figure 16: Manager reservation management page

Manager reservation management page, reservations for Foody Hub restaurant as of April 30, 2025. It displays one confirmed reservation of Sample Customer on 7:00 PM for 2 people, including details of contact (phone and email). Under tab navigation, there is a functionality for managers to see reservation schedules for today’s, yesterday’s, or tomorrow’s bookings. The page contains edit and view action buttons to handle each reservation while preserving the same DineEase header and footer elements throughout the page.



Reservation Details

The Gourmet Kitchen
Confirmed

Reservation ID
#1
Created
May 1, 2023

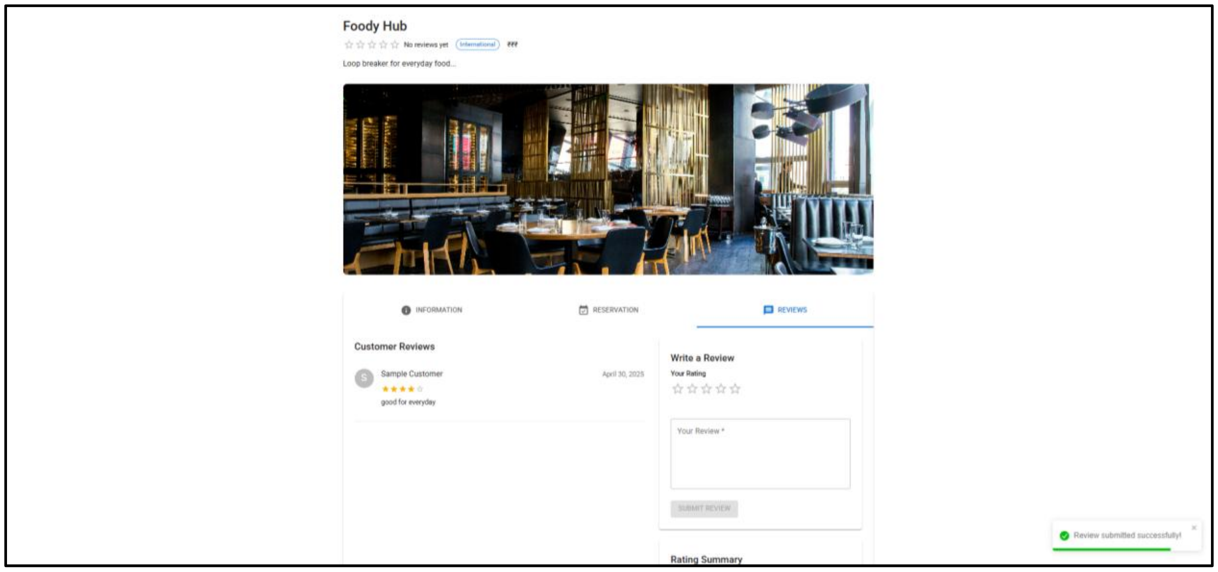
Reservation Information
Date
May 15, 2023
Time
7:30 PM
Party Size
4 people
Table: 5
Special Request
Window seating if possible

Customer Information
Name
John Smith
Email
john.smith@example.com
Phone
(555) 123-4567

Manage Reservation
Status
Confirmed
Notes (Internal)

Figure 17: Reservation view page for manager

Reservation view page for the manager provides detailed information about the reservation at The Gourmet Kitchen. Sections are available on the page: Reservation information (May 15, 2025, 7:30 PM, 4 people with a special request for window seating), Customer information and John Smith's contact details, and Reservation management options. Managers can multiple markers to change the status (currently confirmed), add notes, and save or cancel the reservation.



Foody Hub
No reviews yet International $$$
Loop breaker for everyday food...

Customer Reviews
Sample Customer April 30, 2025
★★★★☆
good for everyday

Write a Review
Your Rating
☆☆☆☆☆
Your Review *

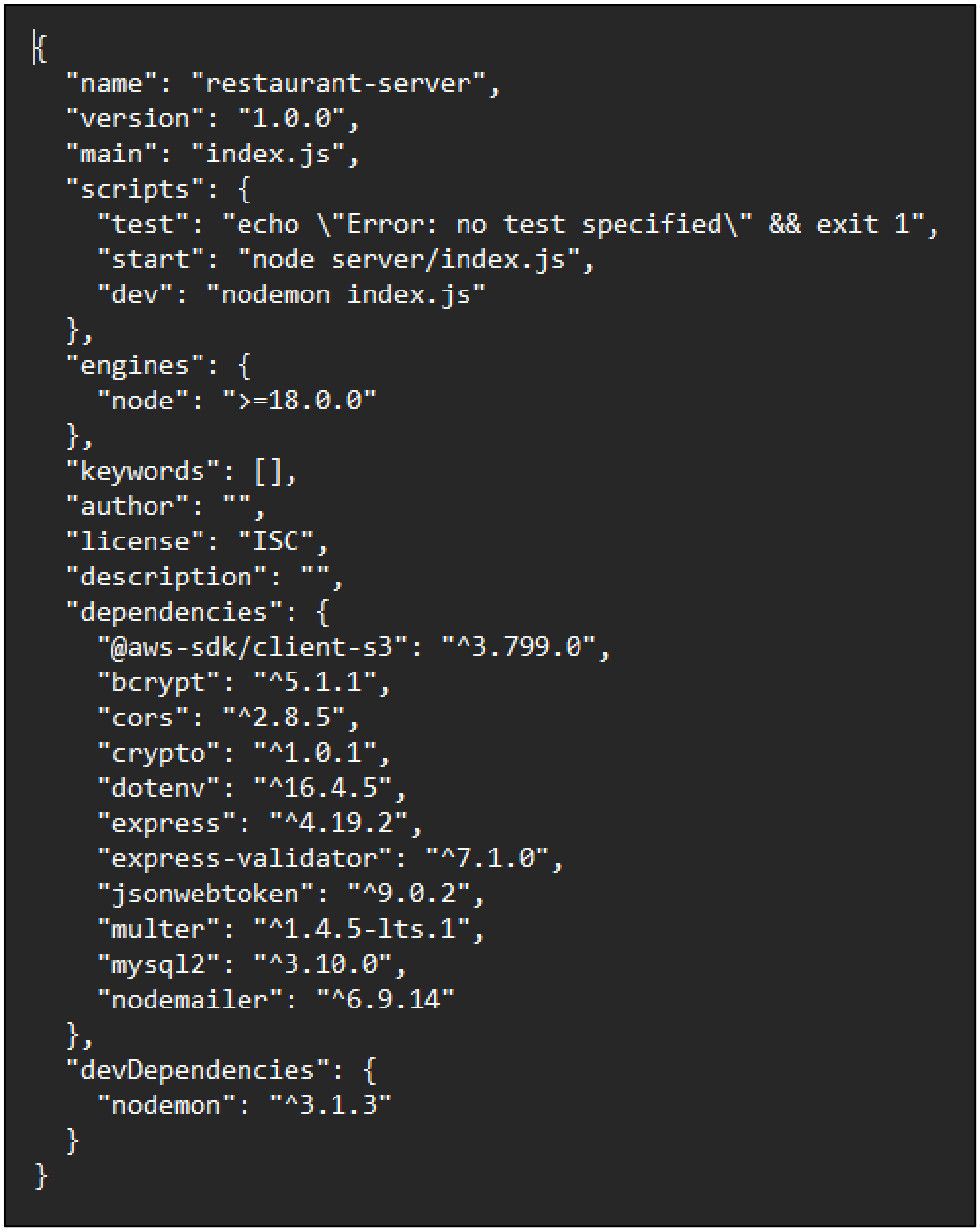
Review submitted successfully!

Rating Summary

Figure 18: Restaurant reviews page

Restaurant reviews page includes the public-facing reviews section of the Foody Hub restaurant. The page shows an excellent interior picture of the modern dining space with stylish decor. Navigation tabs are Information, Reservation, and Reviews (and these are currently selected). The page reveals one of the existing 4-star reviews done by a sample customer on April 25, 2025, and has a form for customers to complete to write and submit other reviews. A review was recently submitted is indicated by a success message.

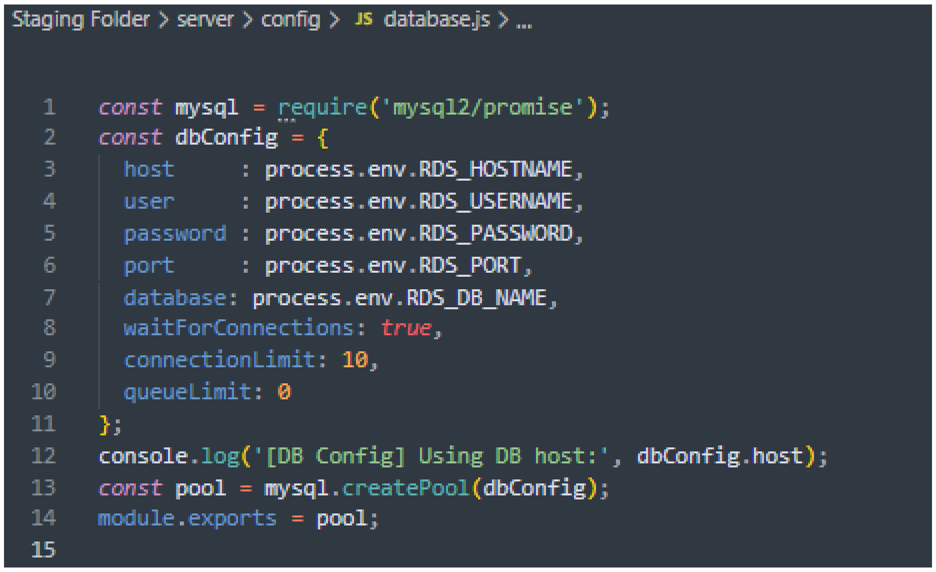
5. API Design



```json
{
  "name": "restaurant-server",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server/index.js",
    "dev": "nodemon index.js"
  },
  "engines": {
    "node": ">=18.0.0"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "@aws-sdk/client-s3": "^3.799.0",
    "bcrypt": "^5.1.1",
    "cors": "^2.8.5",
    "crypto": "^1.0.1",
    "dotenv": "^16.4.5",
    "express": "^4.19.2",
    "express-validator": "^7.1.0",
    "jsonwebtoken": "^9.0.2",
    "multer": "^1.4.5-lts.1",
    "mysql2": "^3.10.0",
    "nodemailer": "^6.9.14"
  },
  "devDependencies": {
    "nodemon": "^3.1.3"
  }
}
```

Figure 19: Package.json

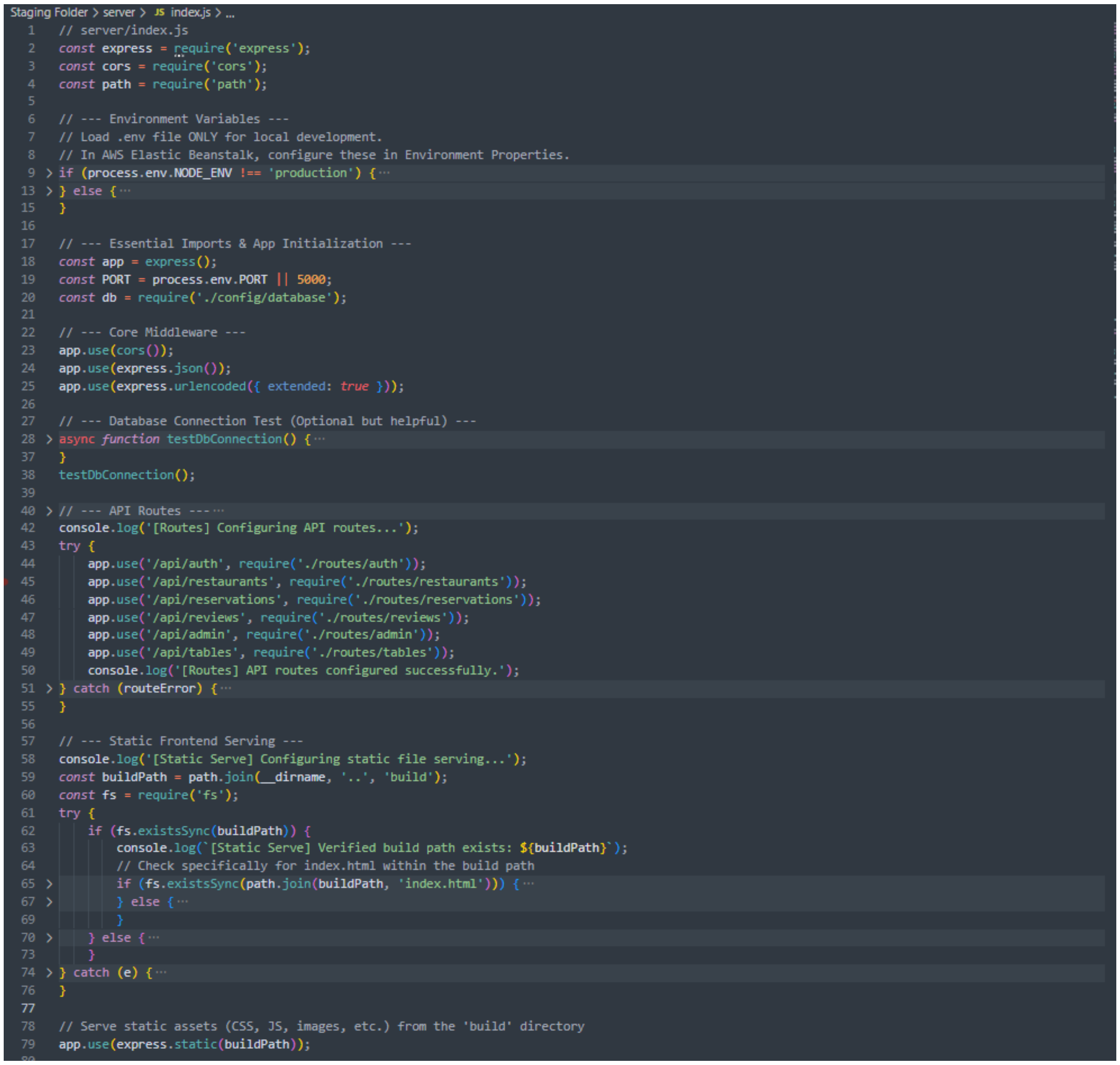
Restaurant Server Node.js Application Configuration file. It identifies the project as version 1.0.0, which has scripts for testing, starting the server, and development using nodemon. Major dependencies: AWS SDK, security libraries (bcrypt, crypto), DB connections (mysql2), Express framework, and email functions. The package needs Node.js v18.0.0 and above and uses nodemon for development purposes. The ISC license is written to be an open-source project.


Staging Folder > server > config > JS database.js > ...

```
const mysql = require('mysql2/promise');
const dbConfig = {
  host     : process.env.RDS_HOSTNAME,
  user     : process.env.RDS_USERNAME,
  password : process.env.RDS_PASSWORD,
  port     : process.env.RDS_PORT,
  database: process.env.RDS_DB_NAME,
  waitForConnections: true,
  connectionLimit: 10,
  queueLimit: 0
};
console.log('[DB Config] Using DB host:', dbConfig.host);
const pool = mysql.createPool(dbConfig);
module.exports = pool;
```

Figure 20: Database connection

The implementation code is to connect to a MySQL database with a setup of credentials through environment variables for a safe credential handling process. It sets up the database pool with parameters of host, user, password, port, and name of database connection settings. The code sets up waitForConnections being true, limits connections to 10 and disables queuing. The connection pool is exported for use in other application parts.

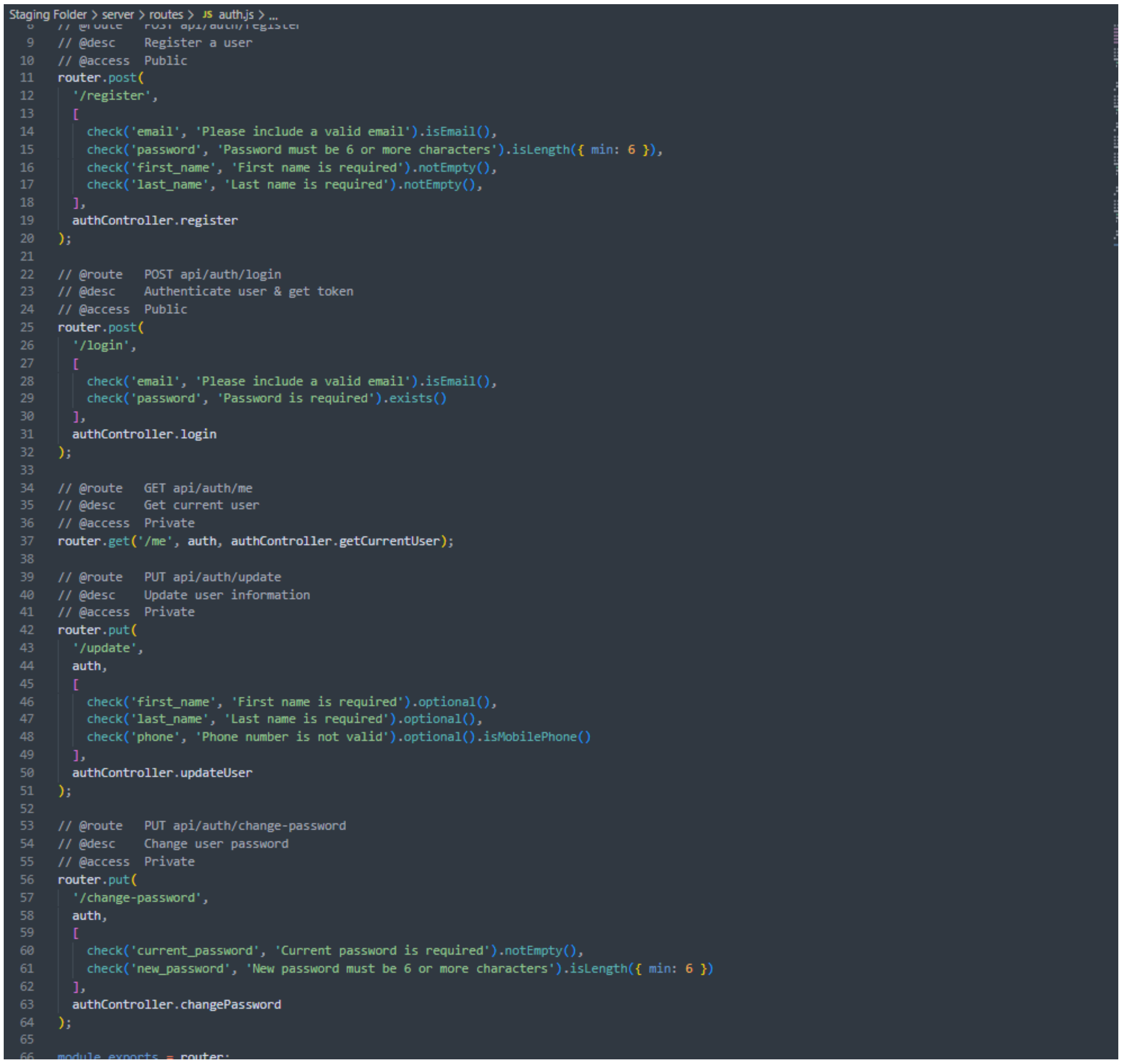


Staging Folder > server > JS index.js > ...

```javascript
// server/index.js
const express = require('express');
const cors = require('cors');
const path = require('path');

// --- Environment Variables ---
// Load .env file ONLY for local development.
// In AWS Elastic Beanstalk, configure these in Environment Properties.
if (process.env.NODE_ENV !== 'production') {⋯
} else {⋯
}

// --- Essential Imports & App Initialization ---
const app = express();
const PORT = process.env.PORT || 5000;
const db = require('./config/database');

// --- Core Middleware ---
app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// --- Database Connection Test (Optional but helpful) ---
async function testDbConnection() {⋯
}
testDbConnection();

// --- API Routes ---⋯
console.log('[Routes] Configuring API routes...');
try {
    app.use('/api/auth', require('./routes/auth'));
    app.use('/api/restaurants', require('./routes/restaurants'));
    app.use('/api/reservations', require('./routes/reservations'));
    app.use('/api/reviews', require('./routes/reviews'));
    app.use('/api/admin', require('./routes/admin'));
    app.use('/api/tables', require('./routes/tables'));
    console.log('[Routes] API routes configured successfully.');
} catch (routeError) {⋯
}

// --- Static Frontend Serving ---
console.log('[Static Serve] Configuring static file serving...');
const buildPath = path.join(__dirname, '..', 'build');
const fs = require('fs');
try {
    if (fs.existsSync(buildPath)) {
        console.log(`[Static Serve] Verified build path exists: ${buildPath}`);
        // Check specifically for index.html within the build path
        if (fs.existsSync(path.join(buildPath, 'index.html'))) {⋯
        } else {⋯
        }
    } else {⋯
    }
} catch (e) {⋯
}

// Serve static assets (CSS, JS, images, etc.) from the 'build' directory
app.use(express.static(buildPath));
```

Figure 21: Server app

The configuration of the Main Express application for the restaurant server. The code sets up Express, loads environment variables, configures middleware (CORS, JSON parsing, and timeout), and connects to the database. There are API routes for authentication, restaurants, reservations, users, admin, and tables. The bottom part takes care of static frontend serving, checking whether build paths exist or not, and serving HTML/CSS/ JS assets from the build repo, which completes the full-stack implementation.



```javascript
// @route   POST api/auth/register
// @desc    Register a user
// @access  Public
router.post(
  '/register',
  [
    check('email', 'Please include a valid email').isEmail(),
    check('password', 'Password must be 6 or more characters').isLength({ min: 6 }),
    check('first_name', 'First name is required').notEmpty(),
    check('last_name', 'Last name is required').notEmpty(),
  ],
  authController.register
);

// @route   POST api/auth/login
// @desc    Authenticate user & get token
// @access  Public
router.post(
  '/login',
  [
    check('email', 'Please include a valid email').isEmail(),
    check('password', 'Password is required').exists()
  ],
  authController.login
);

// @route   GET api/auth/me
// @desc    Get current user
// @access  Private
router.get('/me', auth, authController.getCurrentUser);

// @route   PUT api/auth/update
// @desc    Update user information
// @access  Private
router.put(
  '/update',
  auth,
  [
    check('first_name', 'First name is required').optional(),
    check('last_name', 'Last name is required').optional(),
    check('phone', 'Phone number is not valid').optional().isMobilePhone()
  ],
  authController.updateUser
);

// @route   PUT api/auth/change-password
// @desc    Change user password
// @access  Private
router.put(
  '/change-password',
  auth,
  [
    check('current_password', 'Current password is required').notEmpty(),
    check('new_password', 'New password must be 6 or more characters').isLength({ min: 6 })
  ],
  authController.changePassword
);

module.exports = router;
```

Figure 22: Authentication route

Router implementation of user authentication for the DineEase application. It defines four key endpoints: registration (needs email, password, first/last name), login (needs – email/password), current user details (private route requires auth), update profile details (private route allows changes in first/last name and phone), change password (private route requires current/new passwords). Each route has validations defined, and routes to relevant controller functions for the business logic of the system.

6. Database Design

6.1 Entity-Relationship Diagram (ERD)

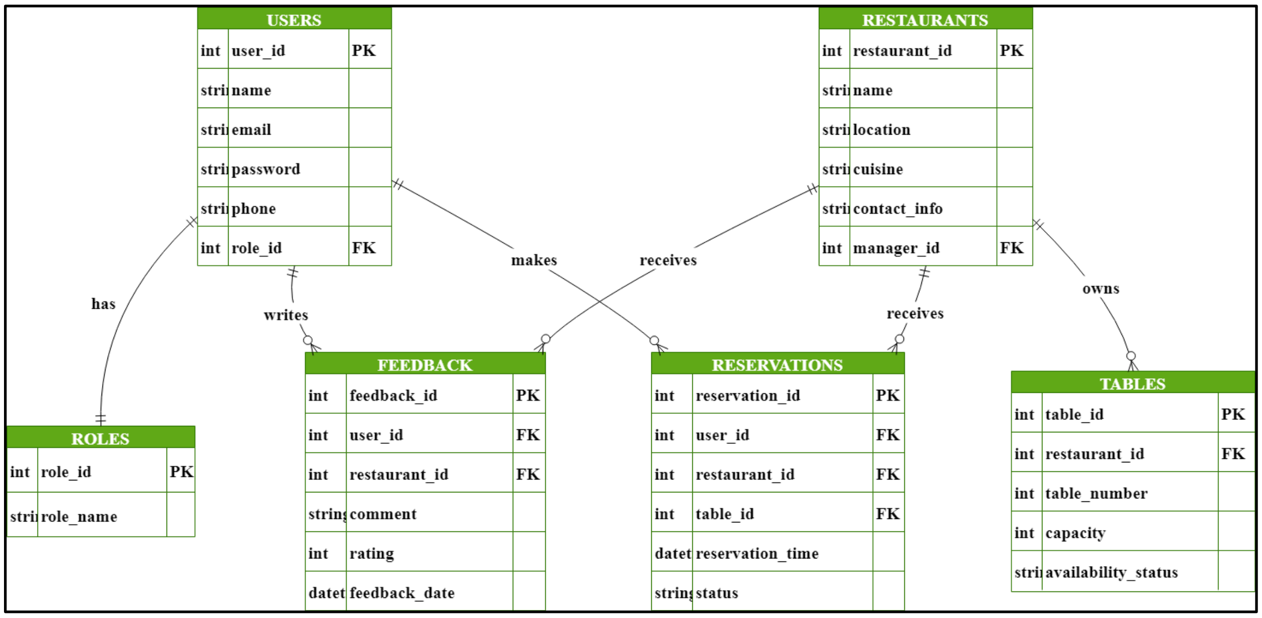


Figure 23: ER diagram

An ERD diagram shows the data structure for a restaurant reservation system where relationships between users, roles, restaurants, reservations, tables, and feedback are visible. Access to the system is determined by users' roles: customer, manager, or admin. Customers can book and give orders for restaurants. Each reservation is attached to a certain restaurant and table, and the tables based on which restaurants are formed have such attributes as capacity and availability. Managers are related to the restaurants that they run. Feedback consists of ratings and comments connected to users and restaurants (Andreoli *et al*., 2021). This model maintains effective data handling, user-role control, as well as scalable operations in the restaurant establishment and the customer interaction.

6.2 Description of key tables

Users table

The basic details that belong to all users such as customers, the restaurant managers and the admins are saved in the users table. It includes personal details such as names, email, phone number, encrypted password and assigned roles (Manca *et al*., 2023). This table facilitates role based access control and helps trace each of the user's activity in the system as well as it can be properly secured.

Bookings table

All reservations made by customers to the restaurant are made in the reservations table. It references to the customer, restaurant, table, date, time, party size, special requests, and booking status (Nufus *et al*., 2025). This framework guarantees optimal table allocation management and reservation history monitoring for analysis and service enhancement.

Restaurants table

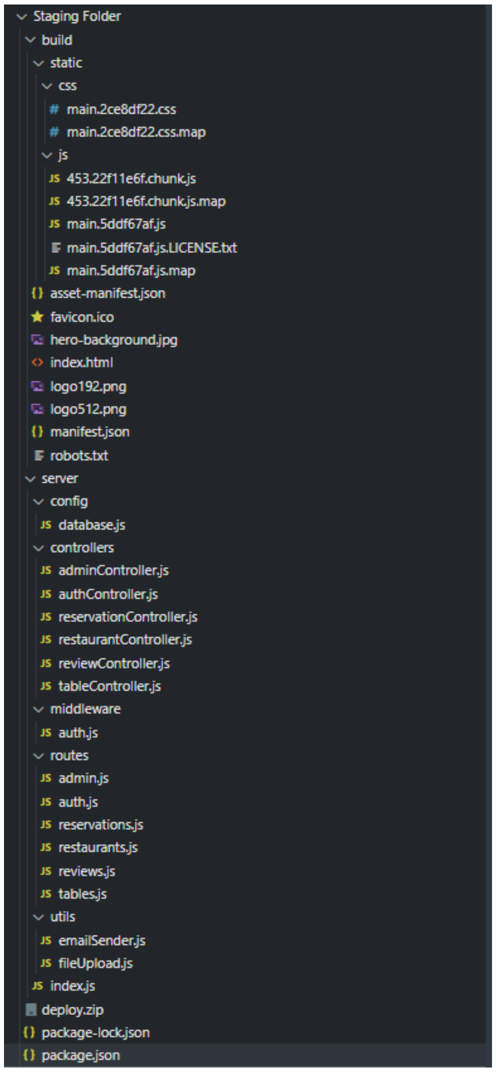
The restaurant's table stores information about every restaurant as name, description, cuisine type, contact details, location, and cost. It also points to the manager user accountable for that restaurant (Rajapaksha *et al*., 2022). This table underlies discovery, management, and customer review capacities and serves as a hub for other aspects.

Reviews table

The customer’s feedback on the restaurant experience has been stored on the reviews table. Every review contains a rating, a comment (optional), and a link to the user and restaurant reviewed (Martinez-Romero *et al*., 2025). This table gives a road map to acquire customer satisfaction information, to be reflected through the restaurant and its visibility and reputations across the platform.

7. AWS Deployment

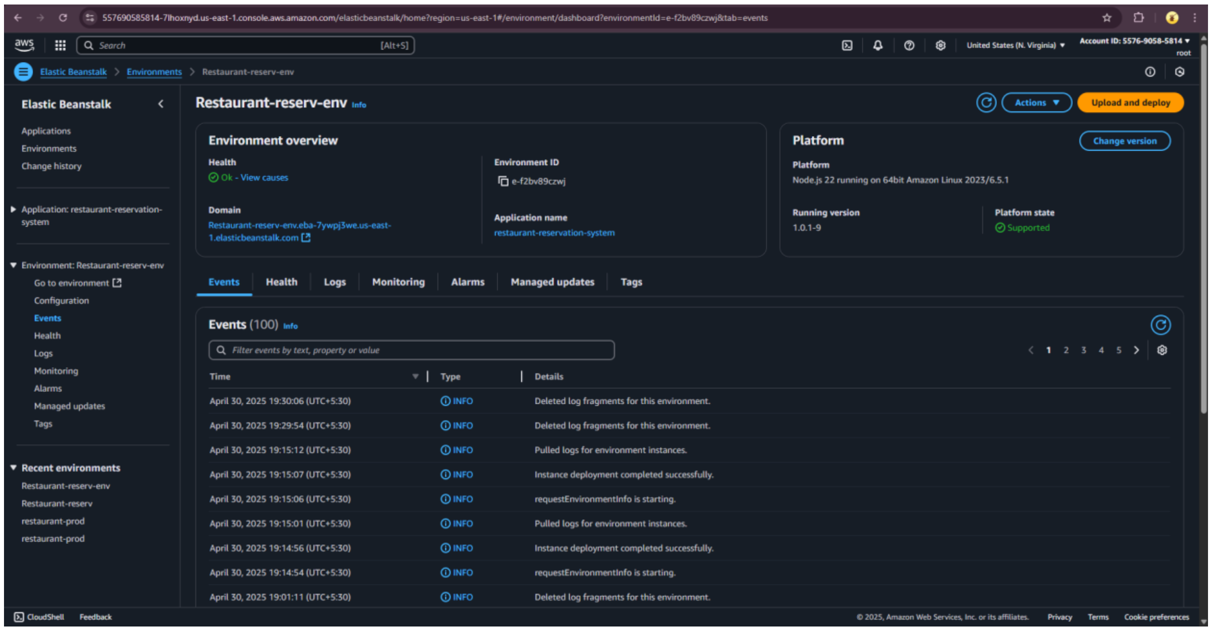
7.1 How the app is deployed (AWS EC2 Auto Scaling Group, Load Balancer)



```
Staging Folder
  build
    static
      css
        main.2ce8df22.css
        main.2ce8df22.css.map
      js
        453.22f11e6f.chunk.js
        453.22f11e6f.chunk.js.map
        main.5ddf67af.js
        main.5ddf67af.js.LICENSE.txt
        main.5ddf67af.js.map
    asset-manifest.json
    favicon.ico
    hero-background.jpg
    index.html
    logo192.png
    logo512.png
    manifest.json
    robots.txt
  server
    config
      database.js
    controllers
      adminController.js
      authController.js
      reservationController.js
      restaurantController.js
      reviewController.js
      tableController.js
    middleware
      auth.js
    routes
      admin.js
      auth.js
      reservations.js
      restaurants.js
      reviews.js
      tables.js
    utils
      emailSender.js
      fileUpload.js
    index.js
  deploy.zip
  package-lock.json
  package.json
```

Figure 24: Complete file structure for deployment

The figure displays a structure of the directory for the deployment of a web application. The project seems to be organized so that it is divided into folders such as “build” (having static assets like CSS); “server” (controllers for authentication, reservations, restaurants, etc.); “routes”, “middleware”, and “utils”. < JavaScript files denote a Node.js application, probably of a restaurant reservation system. The structure is built like a standard MVC, having separate folders for controllers, routes, and middleware. Configuration files such as package.json and package-lock.json can be seen. Such an organized structure makes it easy to deploy to AWS EC2, as in the title.


557690585814-7lhoxnyd.us-east-1.console.aws.amazon.com/elasticbeanstalk/home?region=us-east-1#/environment/dashboard?environmentId=e-f2bv89czwj&tab=events
Search [Alt+S]
United States (N. Virginia)
Account ID: 5576-9058-5814
root
Elastic Beanstalk > Environments > Restaurant-reserv-env
Elastic Beanstalk
Applications
Environments
Change history
Application: restaurant-reservation-system
Environment: Restaurant-reserv-env
Go to environment
Configuration
Events
Health
Logs
Monitoring
Alarms
Managed updates
Tags
Recent environments
Restaurant-reserv-env
Restaurant-reserv
restaurant-prod
restaurant-prod
Actions
Upload and deploy
Change version
Events
Health
Logs
Monitoring
Alarms
Managed updates
Tags
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Restaurant-reserv-env Info

Environment overview

Health: Ok - View causes

Environment ID: e-f2bv89czwj

Domain: Restaurant-reserv-env.eba-7ywpj3we.us-east-1.elasticbeanstalk.com

Application name: restaurant-reservation-system

Platform

Platform: Node.js 22 running on 64bit Amazon Linux 2023/6.5.1

Running version: 1.0.1-9

Platform state: Supported

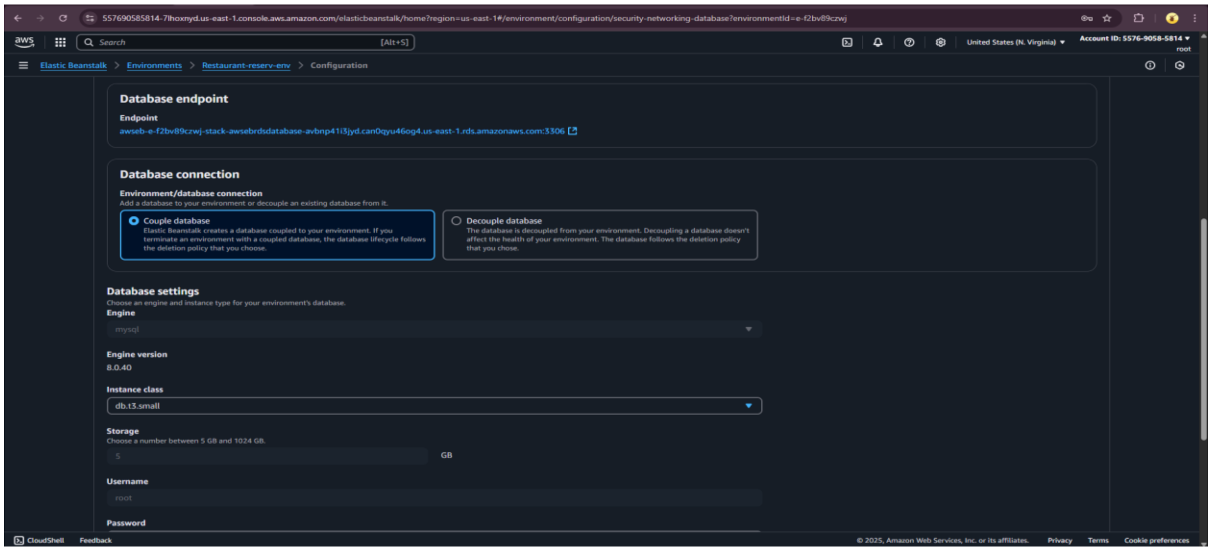
Events (100) Info

Filter events by text, property or value

Time	Type	Details
April 30, 2025 19:30:06 (UTC+5:30)	INFO	Deleted log fragments for this environment.
April 30, 2025 19:29:54 (UTC+5:30)	INFO	Deleted log fragments for this environment.
April 30, 2025 19:15:12 (UTC+5:30)	INFO	Pulled logs for environment instances.
April 30, 2025 19:15:07 (UTC+5:30)	INFO	Instance deployment completed successfully.
April 30, 2025 19:15:06 (UTC+5:30)	INFO	requestEnvironmentInfo is starting.
April 30, 2025 19:15:01 (UTC+5:30)	INFO	Pulled logs for environment instances.
April 30, 2025 19:14:56 (UTC+5:30)	INFO	Instance deployment completed successfully.
April 30, 2025 19:14:54 (UTC+5:30)	INFO	requestEnvironmentInfo is starting.
April 30, 2025 19:01:11 (UTC+5:30)	INFO	Deleted log fragments for this environment.

Figure 25: AWS deployment environment

The AWS Elastic Beanstalk console for "restaurant–reservation–system" application. The environment, named "Restaurant-reserv-env", is running smoothly on Node.js 22 with Amazon Linux 2023/5.5.1 (Green health status). The application version is 1.0.1-9. The events panel displays successful deployment activities of April 30, 2025, including instance deployments and log management operations. The environment instance ID is "f2be89czwj" and the instance has the application accessible under a domain that ends with ".elasticbeanstalk.com" based on the US-East-1 region.


557690585814-7lhoxnyd.us-east-1.console.aws.amazon.com/elasticbeanstalk/home?region=us-east-1#/environment/configuration/security-networking-database?environmentId=e-f2bv89czwj
Search [Alt+S]
United States (N. Virginia)
Account ID: 5576-9058-5814
root
Elastic Beanstalk > Environments > Restaurant-reserv-env > Configuration
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Database endpoint

Endpoint: awseb-e-f2bv89czwj-stack-awsebrdsdatabase-avbnp41i3jyd.can0qyu46og4.us-east-1.rds.amazonaws.com:3306

Database connection

Environment/database connection
Add a database to your environment or decouple an existing database from it.

[✓ Couple database] Elastic Beanstalk creates a database coupled to your environment. If you terminate an environment with a coupled database, the database lifecycle follows the deletion policy that you choose.

Decouple database: The database is decoupled from your environment. Decoupling a database doesn't affect the health of your environment. The database follows the deletion policy that you chose.

Database settings
Choose an engine and instance type for your environment's database.

Engine: mysql

Engine version: 8.0.40

Instance class: db.t3.small

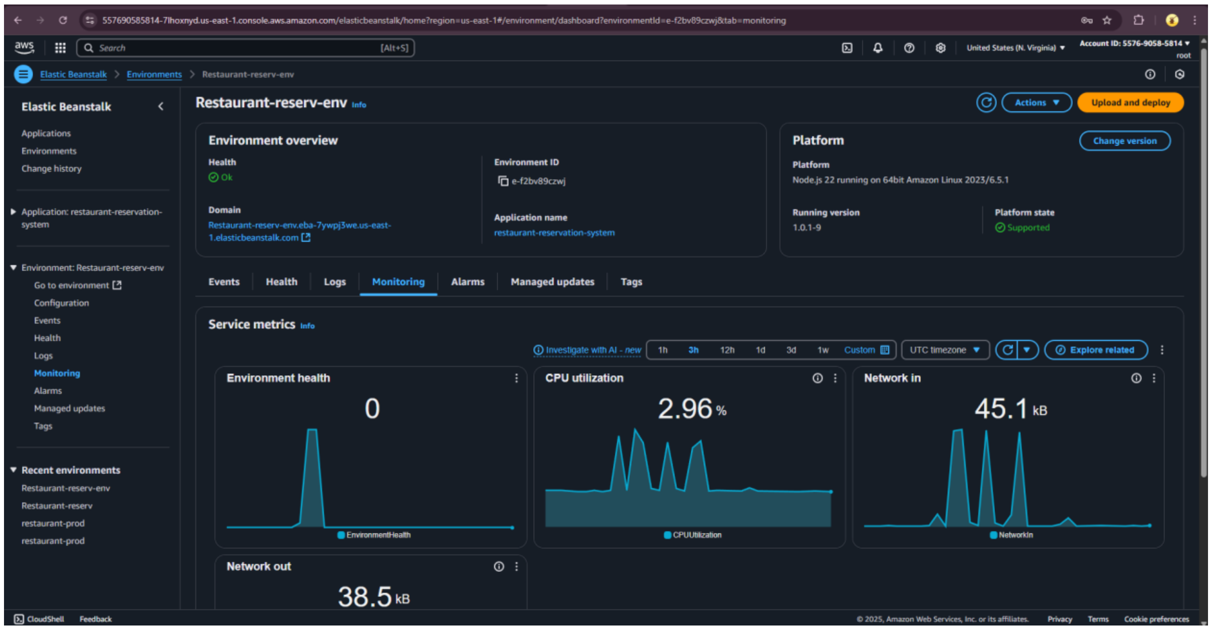
Storage: Choose a number between 5 GB and 1024 GB.
5 GB

Username: root

Password

Figure 26: AWS database endpoint

Figure portrays the database setup in the restaurant reservation app in AWS Elastic Beanstalk. From the endpoint URL, the Database is visible, connecting to an Amazon RDS instance. MySQL 8.0.40 is chosen as the database engine on db.t3.small instance class and with 5GB storage. It's set up in the "Coupled database" mode, which means it's coupled with the lifecycle of the Elastic Beanstalk environment. Such an arrangement allows the database and application environment to be run in unison for consistency of both deployment and operations.


557690585814-7lhoxnyd.us-east-1.console.aws.amazon.com/elasticbeanstalk/home?region=us-east-1#/environment/dashboard?environmentId=e-f2bv89czwj&tab=monitoring

Restaurant-reserv-env

Environment overview
Health: Ok
Environment ID: e-f2bv89czwj
Domain: Restaurant-reserv-env.eba-7ywpj3we.us-east-1.elasticbeanstalk.com
Application name: restaurant-reservation-system

Platform
Platform: Node.js 22 running on 64bit Amazon Linux 2023/6.5.1
Running version: 1.0.1-9
Platform state: Supported

Service metrics
Environment health: 0
CPU utilization: 2.96 %
Network in: 45.1 kB
Network out: 38.5 kB

Figure 27: AWS environment monitoring

The monitoring dashboard for the restaurant reservation application in AWS Elastic Beanstalk is shown in the above figure. The "Restaurant-reserv-env" environment demonstrates key performance figures with graphs illustrating the CPU utilization (which averages at 2.96%), network inbound traffic (45.1 KB), and network outbound traffic (38.5 KB). The environment looks healthy (green "OK" status) and it is running on Node.js 22 with Amazon Linux 2023/5.5.1, version 1.0.1-9. The monitoring interface also supports any custom time frames (1h, 3h, 12h, etc) and metric exploration through CloudWatch. Using this dashboard, applications are monitored for performance in real-time, which is of assistance to developers to keep operations stable as well as detect issues that may arise before they affecting the users.

7.2 Database hosting (RDS)

The Amazon Web Services (AWS) hosting on Amazon Relational Database Service (RDS) offers the restaurant reservation platform's MySQL database a very scalable, secure, and managed setting

(Bidikar *et al*., 2025). RDS makes it easier to manage database operations like provisioning, patching, backup, recovery, and scaling the platform's database is available with increased performance.

1. Managed Service for MySQL Databases

Amazon RDS enables the platform to leverage the MySQL RDBMS with little admin burden. RDS will automate all the maintenance processes, such as the setup of databases, the patch management, and the back-ups that developers will be interested in the application logic not the database maintenance.

2. High Availability and Scalability

Multi-AZ deployments can be configured by RDS to ensure High Availability for the database with availability of the database across multiple Availability Zones for fault tolerance and to tackle disaster situations (Ocran *et al*., 2025). RDS facilitates read replicas to distribute read traffic and enhance the platform's performance in high-demand periods for scalability. Horizontal scaling is also possible, which can do the needful as the number of users and restaurants increases.

3. Security and Data Protection

RDS is linked up with the AWS Identity and Access Management (IAM), for the purposes of access control, that this is a way to ensure that only authenticated end users and applications are able to connect to the database. Encryption of data in transit as well as at rest enhances the protection to customers and business data (Barros *et al*., 2025). Automated backups and snapshots add another segment on the level of data protection, which makes assure that valuable data is recoverable in case of failure.

4. Monitoring and Performance Optimization

Amazon RDS provides tools for monitoring performance through Amazon CloudWatch, which give ideas on such database performance indicators that include the number of CPU, storage and execution of queries (Fortino *et al*., 2022). These metrics empower administrators to fine-tune query performance and make the system conform to the performance requirement.

RDS for database hosting offers a strong, scalable and secure solution for hosting the platform's relational database with features aimed to reduce overhead in operation and increase system performance & availability respectively.

8. Agile Process & Project Management

8.1 XP Core Values

Extreme Programming (XP) is an approach to the development of software, emphasizing high-quality software using a set of core values. Such values promote cooperation, continuous feedback, and multiple releases to enhance the development process.

Core Value 1: Communication

- **Definition:** Communication means the clear sharing of thoughts, decisions and information between members of a team, stakeholders, and customers.
- **Importance in XP:** In XP, effective communication actually is vital as it means that all member will be on the same page and that they are working towards one vision. It minimises misunderstandings and supports fast decision making.

How the Team Applied Communication

Daily Standups: The team had daily standup meetings (Scrum) where everyone would describe what they've worked on, what they will work on next, and blockers. This guaranteed that the team members were in alignment with the project status and they could have enhanced collaboration.

Pair Programming: In some situations, team members collaborated on the tasks (pair programming), thus they communicated immediately, shared knowledge learned, and provided immediate feedback.

Core Value 2: Feedback

- **Definition:** Feedback is the process of both giving and receiving input on the work completed, and so the team has the chance to know if they are progressing in the direction of expectancies and improve their work as they go.
- **Importance in XP:** Because it makes for quick iterations and corrections, XP requires continuous feedback. It is a mechanism to discover problems early and satisfy the desires of the customers.

How the Team Applied Feedback

Test-Driven Development (TDD): The team practiced TDD, first coding up tests, then actual code. Such an approach returned immediate feedback regarding the quality of the code as well as its feasibility, concerning the given requirements.

Continuous Integration (CI): The team had CI pipeline, where any code changes were tested, and integrated to the system on regular intervals. This enabled the team to receive a rapid feedback on quality of the code and possible troubles.

Sprint Reviews: Towards the conclusion of every sprint, the team held sprint review sessions to present what they have accomplished.

Retrospectives: The team used a retrospective at the end of each sprint to reflect on what worked, what didn't, and how to do better. This feedback loop gave the team the opportunity and space to correct their working practices towards better performance.

9. Individual Contributions

Sanjushree Golla – Backend Developer

Driven in designing and designing and implementing RESTful APIs in JavaScript, including input validation and error handling. Participated in development and tested the user authentication mechanisms, as well as ensuring that the application, the API endpoints worked properly for all user roles (Mubeen, 2024).

Harsha Vardhan Badithaboina – Frontend Developer

Headed the development of the web interface based on React. Developed and implemented responsive UI components for Customers, Restaurant Managers, and also Admins. Funnelled closely with backend APIs and integrated Google Maps to give real-time viewing of restaurant locations.

Chanukya Vejandla – Deployment & DevOps

AWS deployment was handled on Elastic Beanstalk. Deployed auto-scaling and load-balancing for application reliability. Set up environment variables, provided secure cloud deployment, and did post-deployment testing for performance and uptime in different user interfaces (Martseniuk *et al*., 2024).

Jayanth Sai Yarlagadda – Frontend and Database Designer

Designed the DB schema and worked with task distribution using Agile tools. Wireframes and documentation that had been kept up (Sonawane *et al*., 2020). Developed component and deployment diagrams and maintained team coordination, weekly scrum updates, and effective communication practices.

Scrum Sheet and TaskBoard Google Sheet Links:

Link1:

https://docs.google.com/spreadsheets/d/19llil24bXxkkQ9kxDbmTfY8rllQxSErUM3GAfFTOp7c/edit?usp=sharing

Link2:

https://docs.google.com/document/d/1T8ebPMvMR_i4Hgn8QFiI1tmCeKMMcX3MhrxIDDxT7BQ/edit?usp=sharing

10. Conclusion

The restaurant reservation and management system effectively manages the need to have an efficient flow, user user-friendly system with scalability that benefits not only customers but also the restaurant managers and administrators. Through the core functionalities such real real-time booking, profile management, table allocation, and review management, the system facilitates a proper dining experience and management. Modern tech solutions such as React.js for frontend, Node.js with Express for backend, and MySQL for data storage, along with fitting web standards, promise responsiveness and scalability of the site and full maintainability. Deployment on as well as database hosting in RDS through AWS adds to the reliability and security of the system. The high-level architecture is complemented with well-designed interfaces and API structures providing for unproblematic interaction between various system components. In general, the project is a practical example of full-stack development and cloud deployment adapted to hospitality business. With subsequent improvements, this platform can develop into an all-round system for the functioning of a restaurant and customer relation management within a dynamic digital environment.

References

Andreoli, R., Cucinotta, T. and Pedreschi, D., 2021. RT-MongoDB: A NoSQL database with differentiated performance. In *Proceedings of the 11th International Conference on Cloud Computing and Services Science-CLOSER* (pp. 77-86). Science and Technology Publications (SciTePress).

Barros, S., 2025. Trusted Identities for AI Agents: Leveraging Telco-Hosted eSIM Infrastructure. *arXiv preprint arXiv:2504.16108*.

Bidikar, A., 2025. Flagship: Reimagining Stateful Serverless Architectures at the Edge: An edge-first stateful serverless feature flag service.

D'Agati, L., 2024. Compute Continuum in Bioengineering: Enhanced Motion Capture and Real-Time Data Processing on Cloud-Edge Infrastructures.

Deep, G., Sidhu, J.S. and Mohana, R., 2023. *Access Control Mechanism for Prevention of Insider Threat in Distributed Cloud Environment* (Doctoral dissertation, Jaypee University of Information Technology, Solan, HP).

Fortino, G., Guerrieri, A., Pace, P., Savaglio, C. and Spezzano, G., 2022. Iot platforms and security: An analysis of the leading industrial/commercial solutions. *Sensors*, *22*(6), p.2196.

Henry, B., 2023. *Development and Implementation of a Web-based Hostel Management System: Kigali Independent University ULK as Case Study* (Doctoral dissertation, ULK).

Kertész, D.R., Farkas, K. and Szabó, G., 2021. Best Practices of Cloud Native Application Development. *Bachelor of profession's thesis, Budapest University of Technology and Economics, Budapest*.

Manca, D., 2023. *Study, design and implementation of infrastructure as code libraries for the provisioning of a resilient cloud infrastructure model in a multi-cloud context* (Doctoral dissertation, Politecnico di Torino).

Martinez-Romero, M., Horridge, M., Mistry, N., Weyhmiller, A., Yu, J.K., Fujimoto, A., Henry, A., O'Connor, M.J., Sier, A., Suber, S. and Akdogan, M.U., 2025. RADx Data Hub: A Cloud Repository for FAIR, Harmonized COVID-19 Data. *arXiv preprint arXiv:2502.00265*.

Martseniuk, Y., Partyka, A., Harasymchuk, O. and Korshun, N., 2024. Automated Conformity Verification Concept for Cloud Security. *Cybersecurity Providing in Information and Telecommunication Systems 2024*, *3654*, pp.25-37.

Mubeen, M., 2024. Zero-Trust Architecture for Cloud-Based AI Chat Applications: Encryption, Access Control and Continuous AI-Driven Verification.

Nufus, U., Qadriah, L. and Niazi, A., 2025. Library Management System Application Development at Jabal Ghafur University Using Laravel Framework. *Big Data: Journal of Informatics and Computing*, *1*(1), pp.9-16.

Ocran, A., 2025. Design and Implementation of an Attribute-Based Encryption to Enhance Privacy in Federated Identity Management (FIM) Systems for Cloud Computing. *International Journal of Innovative Science and Research Technology*, *10*(3), pp.2384-2421.

Punia, A., Gulia, P., Gill, N.S., Ibeke, E., Iwendi, C. and Shukla, P.K., 2024. A systematic review on blockchain-based access control systems in cloud environment. *Journal of Cloud Computing*, *13*(1), p.146.

Rahman, M., 2023. Serverless cloud computing: a comparative analysis of performance, cost, and developer experiences in container-level services.

Rajapaksha, C.I., 2022. Machine Learning-Driven Anomaly Detection Models for Cloud-Hosted E-Payment Infrastructures. *Journal of Computational Intelligence for Hybrid Cloud and Edge Computing Networks*, *6*(12), pp.1-11.

Singh, P., Kaur, A., Gupta, P., Gill, S.S. and Jyoti, K., 2021. RHAS: robust hybrid auto-scaling for web applications in cloud computing. *Cluster Computing*, *24*(2), pp.717-737.

Sonawane, V.R., Arage, C.S., Suryawanshi, J.R., Jadhav, S.P., Palve, Y.H. and Gunjal, M.B., 2020. Enhanced Cauchy Matrix Reed-Solomon Codes and Role-Based Cryptographic Data Access for Data Recovery and Security in Cloud Environment.

Zhang, S., Pandey, A., Luo, X., Powell, M., Banerji, R., Fan, L., Parchure, A. and Luzcando, E., 2022. Practical adoption of cloud computing in power systems—Drivers, challenges, guidance, and real-world use cases. *IEEE Transactions on Smart Grid*, *13*(3), pp.2390-2411.