Criterion C: Development

CrossBox Gymnasium Management System

Technology Stack

The system is built using React.js version 18.2 for frontend development, Node.js with Express.js version 4.18 for backend API, and MySQL version 8.0 for database management. Additional technologies include bcrypt for password hashing, jsonwebtoken for authentication, node-cron for scheduled tasks, nodemailer for email functionality, and CSS3 for responsive styling. Development tools included Visual Studio Code, Postman for API testing, and MySQL Workbench for database design.

Database Implementation

MySQL tables were created using CREATE TABLE statements with appropriate data types: VARCHAR for text, INT for numeric values, DATE for dates, DATETIME for timestamps, DECIMAL for monetary values, and ENUM for constrained options. Primary keys use AUTO_INCREMENT for automatic ID generation. Foreign key constraints link related tables with ON DELETE CASCADE for member_workouts and workout_logs to maintain referential integrity. Connection pooling configured with maximum ten concurrent connections optimizes database performance. Indexes added to frequently queried fields like email improve lookup speed.

Backend API Development

Express.js routes organized into modular files handle authentication, members, trainers, workouts, and schedules endpoints. Authentication route receives login credentials, queries appropriate table based on role, compares bcrypt-hashed passwords, and generates JWT tokens containing user ID and role claims. Middleware functions verify tokens by extracting from authorization headers, validating signatures, and attaching user data to request objects. CRUD operations implement async/await pattern with try-catch error handling. Input validation middleware checks required fields, data types, email formats using regex patterns, and phone number lengths before database operations.

Frontend React Implementation

React components use functional components with hooks. Dashboard component employs useState for managing statistics state and useEffect for fetching data on mount. MemberList implements controlled form inputs with onChange handlers updating state variables. Calendar component calculates grid positions based on datetime values and renders event blocks with CSS positioning. Fetch API makes HTTP requests to backend endpoints with authorization headers containing JWT tokens. Error handling displays user-friendly messages for failed requests. Styling uses CSS Flexbox for layouts, grid for calendar views, and media queries for responsive breakpoints.

Key Features

Automated Renewal Reminders: Node-cron schedules daily task querying members table for expiring memberships, constructing HTML email templates with member names and dates, sending via nodemailer with SMTP configuration.

Conflict Detection: Before schedule insertion, queries existing events for same trainer/location, checks datetime overlaps using SQL BETWEEN clauses, returns 400 error with conflict details if overlaps detected.

Progress Tracking: Members submit workout logs storing sets, reps, weights in workout_logs table. Progress view fetches logs, groups by workout, displays trends using chart components showing weight progression over time.

Testing

Unit tests using Jest verified individual functions like validation utilities. Integration tests with Postman confirmed API endpoints return correct HTTP status codes and JSON response formats. User acceptance testing involved client using system with real member data identifying usability improvements. Browser testing across Chrome, Firefox, Safari ensured cross-platform compatibility. Responsive testing with DevTools verified layouts adapt properly to mobile and tablet screen sizes.

Word Count: 481