IRCTC API Architecture Documentation

1 Overview

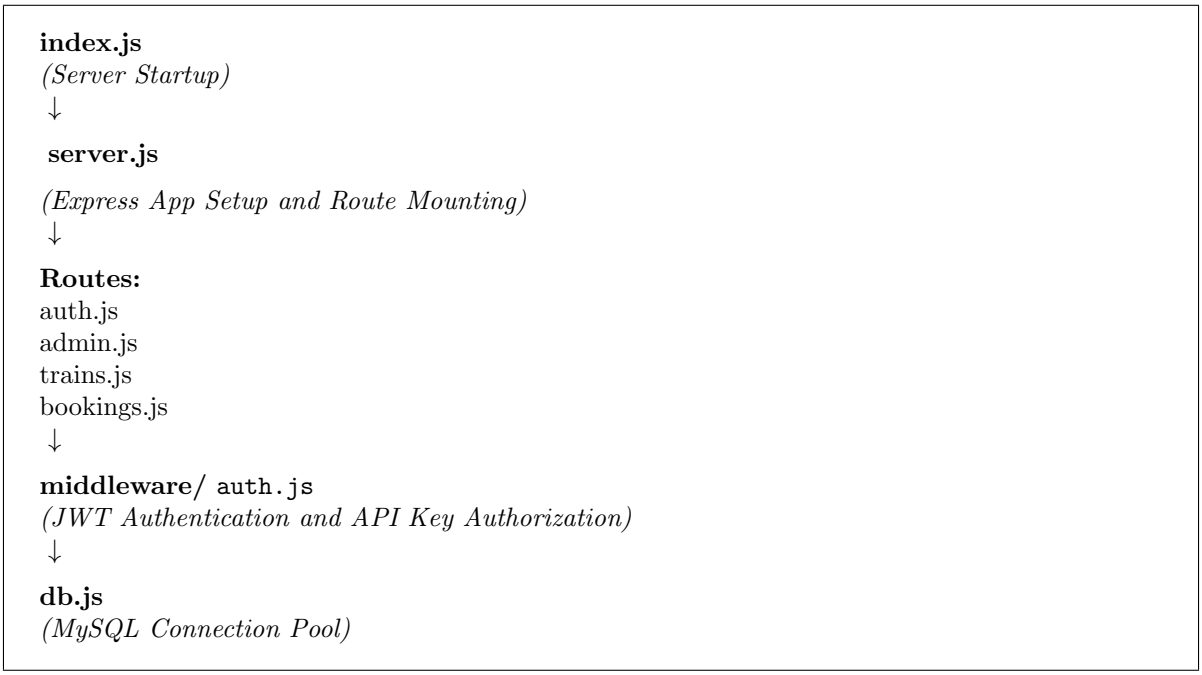
This document describes the architecture of the IRCTC API, built using Node.js, Express, and MySQL. The API is designed following modular design principles and separation of concerns to ensure scalability, maintainability, security, and testability.

2 File Structure

The project is organized as follows:

```
SDE API Round - IRCTC/
 .env                   % Environment variables
 package.json           % Dependencies, scripts, etc.
 README.md              % Project instructions
 index.js               % Entry point for server startup
 server.js              % Express app configuration and route mounting
 db.js                  % MySQL connection pool setup
 middleware/
    auth.js            % Authentication and authorization middleware (JWT, API key)
 routes/
     auth.js            % User registration and login endpoints
     admin.js           % Admin-specific endpoints (e.g., add train)
     trains.js          % Train information and booking endpoints
     bookings.js        % Booking details retrieval endpoints
```

3 Architecture Diagram



4 Design Principles

4.1 Separation of Concerns

- **Routing:** Endpoints are split into separate modules (auth, admin, trains, bookings) based on functionality.
- **Middleware:** Authentication and authorization logic is encapsulated in its own module.
- **Database Connectivity:** All database interactions are centralized in `db.js`.

4.2 Modularity

- The Express application is defined in `server.js` and exported for testing.
- The server startup logic is kept in `index.js`, keeping the application logic separate from deployment.

4.3 Scalability and Maintainability

- **RESTful API Design:** Clear and predictable endpoints.
- **Database Pooling:** Efficient management of concurrent database connections.
- **Concurrency Handling:** Use of transactions and row-level locking prevents race conditions during critical operations (e.g., seat booking).

4.4 Security

- **JWT Authentication:** Secure endpoints require valid JWT tokens.
- **Admin API Key:** Admin endpoints are protected by an additional API key.
- **Environment Variables:** Sensitive data (e.g., JWT secrets, DB credentials) are stored in `.env` and not hardcoded.

4.5 Testability

- The separation of app configuration from server startup allows the application to be imported and tested without launching the server.
- Modular routes and middleware facilitate unit and integration testing using tools like Mocha, Chai, and Supertest.

5 Additional Details

- **Concurrency Management:** MySQL transactions and row-level locking ensure that simultaneous booking requests do not lead to inconsistencies.
- **RESTful Principles:** The API follows REST conventions for resource-based URLs and HTTP method usage.