Rail Watch Backend Documentation

Developed by: Akarshan Ghosh

July 6, 2025

1 Overview

Rail Watch is a backend system developed using Node.js, Express, and MongoDB to manage railway divisions, coach status reporting, and user authentication. The backend includes features such as:

- User and Admin Authentication (JWT-based)
- OTP verification for secure access
- Division and train data management
- Email alerts for chain pulled status
- RESTful APIs

2 Technologies Used

- Node.js
- Express.js
- MongoDB + Mongoose
- Nodemailer (SMTP via Brevo)
- JSON Web Tokens (JWT)
- dotenv

3 Project Structure

```
RAIL_WEB_SERVER/
 config/              → Nodemailer configuration
    nodemailer.js
 conn/                → MongoDB connection setup
    conn.js
 controller/          → All route logic
    AdminController.js
```

```
   AuthController.js
   DivisionController.js
   TrainController.js
   UserController.js
middleware/          → JWT token validator
   UserAuth.js
models/              → Mongoose schemas
   Division.js
   Train.js
   User.js
routes/              → Route definitions
   authRoute.js
   divisionRoute.js
   trainRoute.js
   userRoute.js
app.js               → Main entry file
.env                 → Environment variables
package.json
```

4 Environment Variables (.env)

```
PORT=1000
URI="mongodb+srv://<credentials>"
JWT_SECRET='your_secret'
NODE_ENV='development'
SMTP_USER='smtp_user@brevo.com'
SMTP_PASS='smtp_password'
SENDER_EMAIL='youremail@example.com'
```

5 API Endpoints

Base URL: `http://localhost:1000`

Method & URL	Description	Auth Required
`POST /api/auth/register`	Register a new user	No
`POST /api/auth/login`	Login user via email/phone	No
`POST /api/auth/logout`	Logout current user	No
`POST /api/auth/forgot-password`	Send OTP to reset password	No

Continued on next page

Method & URL	Description	Auth Required
`POST /api/auth/reset-password`	Reset password via OTP	No
`POST /api/auth/verify-email`	Verify user email via OTP	No
`GET /api/auth/getuserbyid`	Get current user info	Yes
`POST /api/auth/is-auth`	Check authentication validity	Yes
`POST /api/auth/admin/login`	Admin login (generate OTP)	No
`POST /api/auth/admin/verify-otp`	Verify admin OTP	No
`GET /api/auth/admin/data`	Get admin data (header: `id`)	Yes
`POST /api/division/add-division`	Add a new division	Admin Only
`POST /api/division/delete-division`	Delete division (header: `division-id`)	Admin Only
`GET /api/division/get-all-division`	Get all divisions	No
`GET /api/division/recent-division`	Get 4 recent divisions	No
`GET /api/division/division-id/:id`	Get division by ID	No
`POST /api/coach/add-coach-data`	Add train/coach sensor data	No
`GET /api/coach/get-coach-data`	Get train details via query params	No
`POST /api/coach/get-coach`	Get available coaches for a train	No
`GET /api/user/data`	Get current logged-in user data	Yes

6 Security Notes

- JWT-based access protection
- OTP verification for password reset and admin login
- Role-based route protection (admin vs user)

7 Deployment Tips

- Host MongoDB on Atlas or managed cluster
- Use platforms like Render, Railway, or Heroku for deployment

- Store all environment variables securely