Ticketing System Backend Team Roles & Instructions

Overview

This document outlines the responsibilities of each backend developer for the ticketing system project, along with instructions and best practices to help them succeed.

Developer 1: Database & Models

Responsibilities: - Design and implement database schema using Sequelize or Prisma. - Define models: User, Ticket, Comment, Attachment, TicketLog, LoginLog. - Create migrations for table creation and modifications. - Seed initial data: roles, admin user. - Ensure relationships between models are properly defined (e.g., User → Tickets).

Instructions to Succeed: - Collaborate with Dev2 to understand required fields for controllers. - Use Sequelize CLI or Prisma Studio to verify database setup. - Write clear and consistent naming for all models and columns. - Document any complex relationships or constraints. - Test migrations locally before pushing.

Developer 2: Controllers & Business Logic

Responsibilities: - Implement CRUD operations for Tickets, Users, Comments, Attachments. - Handle authentication logic (login, registration, JWT token management). - Implement ticket management features (status updates, assignment, logs). - Write unit tests for controller functions.

Instructions to Succeed: - Work closely with Dev1 to ensure model fields match controller logic. - Follow RESTful API best practices. - Validate inputs to prevent security issues. - Use try/catch blocks and proper error handling. - Communicate any missing data requirements to Dev1 immediately.

Developer 3: Routes & Middleware

Responsibilities: - Define Express routes and link them to controller functions. - Implement authentication middleware (authMiddleware.js). - Implement role-based access middleware (roleMiddleware.js). - Implement error handling middleware (errorMiddleware.js). - Ensure API endpoints are secure and properly structured.

Instructions to Succeed: - Coordinate with Dev2 to understand all available controller methods. - Keep routes organized by resource (users, tickets, comments, attachments). - Test all middleware thoroughly (JWT validation, role restrictions). - Document API endpoints and expected inputs/outputs. - Ensure error messages are informative but secure.

Collaboration Guidelines

- **Git Workflow:**
- Branch from `develop` for all tasks.
- Use descriptive feature branch names (e.g., `feature/db-setup`).
- Commit frequently with clear messages.
- Create Pull Requests to `develop` and conduct code reviews.

- **Daily Sync:**

- Merge and test `develop` branch at least once a day.
- Communicate blockers via Slack/Teams.
- Update task boards (Trello/Jira) daily.

- **Testing:**

- Use Postman or Insomnia for endpoint testing.
- Write unit tests for controllers.
- Ensure database migrations are up-to-date before running tests.

- **Best Practices:**

- Keep code modular and well-documented.
- Follow consistent naming conventions.
- Handle errors gracefully.
- Maintain secure handling of sensitive data (passwords, JWTs, etc.).

Outcome: By following these roles and instructions, the backend team can efficiently collaborate, reduce merge conflicts, and deliver a stable ticketing system API.