Developing a Spring Boot REST API application for a Stock Trading Management System is a complex task that involves several components and functionalities. Below are the detailed implementation steps for such a project:

Step 1: Project Setup

1. Create a new Spring Boot project using a build tool like Maven or Gradle.

2. Add necessary dependencies such as Spring Web, Spring Data JPA, and any other dependencies you might need for your database, security, and testing.

Step 2: Define Entities

1. Create entity classes to represent the main objects in your system like `User`, `Portfolio`, `Stock`, `Transaction`, and `TradingAccount`.

2. Annotate these classes with appropriate JPA annotations to define the database schema, relationships, and constraints.

Step 3: Repository Layer

1. Create repositories for each entity using Spring Data JPA. These repositories will handle database operations.

2. Define custom queries in repositories to retrieve data based on specific criteria, like fetching a user's portfolio or transaction history.

Step 4: Service Layer

1. Create service classes for each major functionality, such as `PortfolioService`, `TradingAccountService`, and `StockService`.

2. Implement the business logic for managing portfolios, trading accounts, and stock transactions in these service classes.

Step 5: Controller Layer

1. Create REST controllers for each functionality, such as `PortfolioController`, `TradingAccountController`, and `StockController`.

2. Define REST endpoints (URL mappings) for creating, updating, deleting, and retrieving portfolios, trading accounts, and stock transactions.

3. Use appropriate request and response DTOs (Data Transfer Objects) to handle data exchange between the client and server.

4. Implement validation and error handling in the controllers.

Step 6: Security

1. Implement security features to protect sensitive operations. Spring Security can be used for authentication and authorization.

2. Configure roles and permissions for users and restrict access to certain endpoints based on roles.

Step 7: Maintenance and Updates

1. Regularly update dependencies and libraries to keep the application secure and up to date.

2. Address user feedback and add new features as necessary.

Remember to follow best practices in software development, including code organization, code reviews, and version control (e.g., Git). Additionally, consider using a robust error tracking system like Sentry or New Relic to identify and fix issues in production.