# OtpAuthSystem Project Documentation

## Overview

OtpAuthSystem is a full-stack OTP-based authentication system with a Java Spring Boot backend and a React (Vite) frontend. It demonstrates secure user registration, OTP login, and best practices for password hashing, CORS, and temporary OTP storage using Redis.

## Technologies Used

### Backend

* Java 17+
* Spring Boot 3+
* Spring Data JPA
* MySQL 8 (local, via MySQL Workbench)
* Spring Mail (Mailtrap.io for email testing)
* Spring Security (for password hashing and endpoint security)
* Spring Data Redis (for OTP storage)
* Maven
* Lombok

### Frontend

* React 18+
* Vite
* Axios
* React Router DOM

## Folder Structure

OtpAuthSystem/  
├── backend/  
│ ├── pom.xml  
│ ├── README.md  
│ ├── src/  
│ │ ├── main/java/com/otp/auth/  
│ │ │ ├── App.java  
│ │ │ ├── MailTestRunner.java  
│ │ │ ├── config/  
│ │ │ │ ├── AppConfig.java  
│ │ │ │ └── SecurityConfig.java  
│ │ │ ├── controller/  
│ │ │ │ └── AuthController.java  
│ │ │ ├── entity/  
│ │ │ │ ├── User.java  
│ │ │ │ └── Otp.java (legacy, not used with Redis)  
│ │ │ ├── repository/  
│ │ │ │ ├── UserRepository.java  
│ │ │ │ └── OtpRepository.java (legacy)  
│ │ │ ├── service/  
│ │ │ │ ├── EmailService.java  
│ │ │ │ ├── OtpService.java  
│ │ │ │ └── UserService.java  
│ │ └── resources/  
│ │ └── application.properties  
│ └── test/java/com/otp/auth/AppTest.java  
├── frontend/  
│ ├── package.json  
│ ├── vite.config.js  
│ ├── src/  
│ │ ├── App.jsx  
│ │ └── pages/  
│ │ ├── Register.jsx  
│ │ ├── Login.jsx  
│ │ ├── VerifyOtp.jsx  
│ │ └── Dashboard.jsx  
│ └── public/  
└── ...

## Backend Setup & Configuration

### 1. MySQL (Local)

* Installed MySQL 8 via Homebrew.
* Managed and visualized with MySQL Workbench.
* Created database: otp\_auth\_db.
* User: root (password set during installation).

### 2. Redis (for OTPs)

* Installed Redis via Homebrew: brew install redis
* Started Redis: brew services start redis
* No GUI required; used redis-cli for inspection.

### 3. Spring Boot Configuration

* application.properties:
* spring.datasource.url=jdbc:mysql://localhost:3306/otp\_auth\_db  
  spring.datasource.username=root  
  spring.datasource.password=YOUR\_PASSWORD  
  spring.jpa.hibernate.ddl-auto=update  
  spring.jpa.show-sql=true  
    
  # Email (Mailtrap)  
  spring.mail.host=sandbox.smtp.mailtrap.io  
  spring.mail.port=2525  
  spring.mail.username=YOUR\_MAILTRAP\_USER  
  spring.mail.password=YOUR\_MAILTRAP\_PASS  
  spring.mail.properties.mail.smtp.auth=true  
  spring.mail.properties.mail.smtp.starttls.enable=true  
    
  # Redis  
  spring.redis.host=localhost  
  spring.redis.port=6379
* **CORS**: Configured in AppConfig.java to allow requests from http://localhost:3000.
* **Security**: SecurityConfig.java allows unauthenticated access to /api/auth/\*\* endpoints.
* **Password Hashing**: Uses BCrypt via Spring Security.

### 4. Email Testing

* Used Mailtrap.io for safe email delivery.
* Configured SMTP credentials in application.properties.
* All OTPs are sent to the Mailtrap inbox for testing.

### 5. OTP Storage

* **Redis** is used for OTP storage with 5-minute expiry.
* No OTPs are stored in MySQL after migration to Redis.
* Verified with redis-cli using get otp:<phoneNumber> and ttl otp:<phoneNumber>.

## Sample Configuration Files

### Backend: application.properties.example

A sample config file is provided at backend/src/main/resources/application.properties.example. Copy it to application.properties and fill in your credentials:

spring.datasource.url=jdbc:mysql://localhost:3306/otp\_auth\_db  
spring.datasource.username=YOUR\_DB\_USER  
spring.datasource.password=YOUR\_DB\_PASSWORD  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
  
# Email (Mailtrap example)  
spring.mail.host=sandbox.smtp.mailtrap.io  
spring.mail.port=2525  
spring.mail.username=YOUR\_MAILTRAP\_USER  
spring.mail.password=YOUR\_MAILTRAP\_PASS  
spring.mail.properties.mail.smtp.auth=true  
spring.mail.properties.mail.smtp.starttls.enable=true  
  
# Redis configuration  
spring.redis.host=localhost  
spring.redis.port=6379

## Backend API Endpoints

* POST /api/auth/register — Register user (phoneNumber, name, password, email)
* POST /api/auth/send-otp — Send OTP to email (phoneNumber)
* POST /api/auth/verify-otp — Verify OTP (phoneNumber, otp)

## Frontend Setup & Configuration

### 1. Project Setup

* Scaffolded with Vite: npm create vite@latest frontend -- --template react
* Installed dependencies: axios, react-router-dom

### 2. Routing & Pages

* /register — User registration form
* /login — Send OTP form
* /verify-otp — OTP verification form
* /dashboard — Welcome page after login

### 3. API Integration

* All API calls use Axios to http://localhost:8080/api/auth/...
* Handles success/error messages and redirects.

### 4. Running the Frontend

cd frontend  
npm install  
npm run dev

Open <http://localhost:3000> in your browser.

## Testing & Validation

* Registered users and sent OTPs via Postman, curl, and the React UI.
* Verified OTPs in Mailtrap and Redis.
* Confirmed no OTPs are stored in MySQL after migration.
* Used redis-cli to inspect and confirm OTP expiry and deletion.

## Key Learnings & Best Practices

* Use Redis for short-lived, sensitive data like OTPs.
* Use BCrypt for password hashing.
* Use Mailtrap or similar for safe email testing.
* Separate backend and frontend for clean architecture.
* Use CORS and security configuration for safe API exposure.

## Troubleshooting

* MySQL connection issues: Ensure correct user, password, and port.
* Redis not running: Start with brew services start redis.
* Mail not received: Check Mailtrap credentials and inbox.
* 401 errors: Ensure security config allows unauthenticated access to /api/auth/\*\*.

## Authors & Credits

* Built by following best practices in Java Spring Boot, React, and modern authentication workflows.

## How to Set Up This Project from GitHub

### 1. Clone the Repository

git clone https://github.com/your-username/otp-auth-system.git  
cd otp-auth-system

### 2. Backend Setup

* Install Java 17+, Maven, MySQL 8, and Redis (see above for details).
* Configure your MySQL and Mailtrap credentials in backend/src/main/resources/application.properties.
* Start MySQL and Redis services.
* Build and run the backend:

cd backend  
mvn clean install  
mvn spring-boot:run

### 3. Frontend Setup

* Install Node.js (v18+ recommended).
* Install dependencies and start the frontend:

cd frontend  
npm install  
npm run dev

* Open <http://localhost:3000> or <http://localhost:5173> in your browser.

### 4. Test the Application

* Register a user, send OTP, and verify OTP using the UI or Postman.
* Check Mailtrap for OTP emails.
* Use redis-cli to inspect OTP keys if needed.

## Repository Naming Suggestion

A good public repo name for this project could be: - otp-auth-system - springboot-react-otp-auth - otp-auth-fullstack - otp-authentication-demo

Choose a name that is clear, concise, and describes the tech stack and purpose. otp-auth-system is a great choice!

## End of Documentation