**Hands On Session**

**JWT Authentication Service**

This is a Spring Boot application that provides JWT (JSON Web Token) authentication for RESTful web services.

**## Project Structure**

```

src/main/java/com/example/jwtauthservice/

├── JwtAuthServiceApplication.java    # Main application class

├── config/

│   ├── SecurityConfig.java          # Spring Security configuration

│   └── DataInitializer.java         # Data initialization

├── controller/

│   ├── AuthController.java

│   └── AuthenticationController.java

├── entity/

│   └── User.java

├── filter/

│   └── JwtAuthenticationFilter.java

├── repository/

│   └── UserRepository.java

├── service/

│   └── CustomUserDetailsService.java

└── util/

    └── JwtUtil.java

```

**## Features**

- JWT token generation and validation

- HTTP Basic Authentication for login

- H2 in-memory database integration (default)

- Spring Security configuration

- Stateless authentication

**## Prerequisites**

- Java 11 or higher

- Maven 3.6 or higher

**## Database Setup**

By default, the application uses an H2 in-memory database.

No setup is required.

If you want to use MySQL, update the `src/main/resources/application.properties` file accordingly.

**## Running the Application**

1. Build the project:

   ```bash

   mvn clean package

   ```

2. Run the application:

   ```bash

   mvn spring-boot:run

   ```

The application will start on port 8090.

## API Endpoints

### Authentication

\*\*POST\*\* `/authenticate`

Authenticate a user and receive a JWT token.

\*\*Request (x-www-form-urlencoded):\*\*

- `username`: user

- `password`: pwd

\*\*Example using curl:\*\*

```bash

curl -X POST http://localhost:8090/authenticate -d "username=user&password=pwd"

```

\*\*Response:\*\*

```json

{

  "token": "eyJhbGciOiJIUzI1NiJ9..."

}

```

### Protected Endpoint

\*\*GET\*\* `/hello`

Access a protected endpoint using the JWT token.

\*\*Example using curl:\*\*

```bash

curl -H "Authorization: Bearer YOUR\_JWT\_TOKEN" http://localhost:8090/hello

```

**## Testing with Postman**

1. \*\*Authenticate:\*\*

   - Method: POST

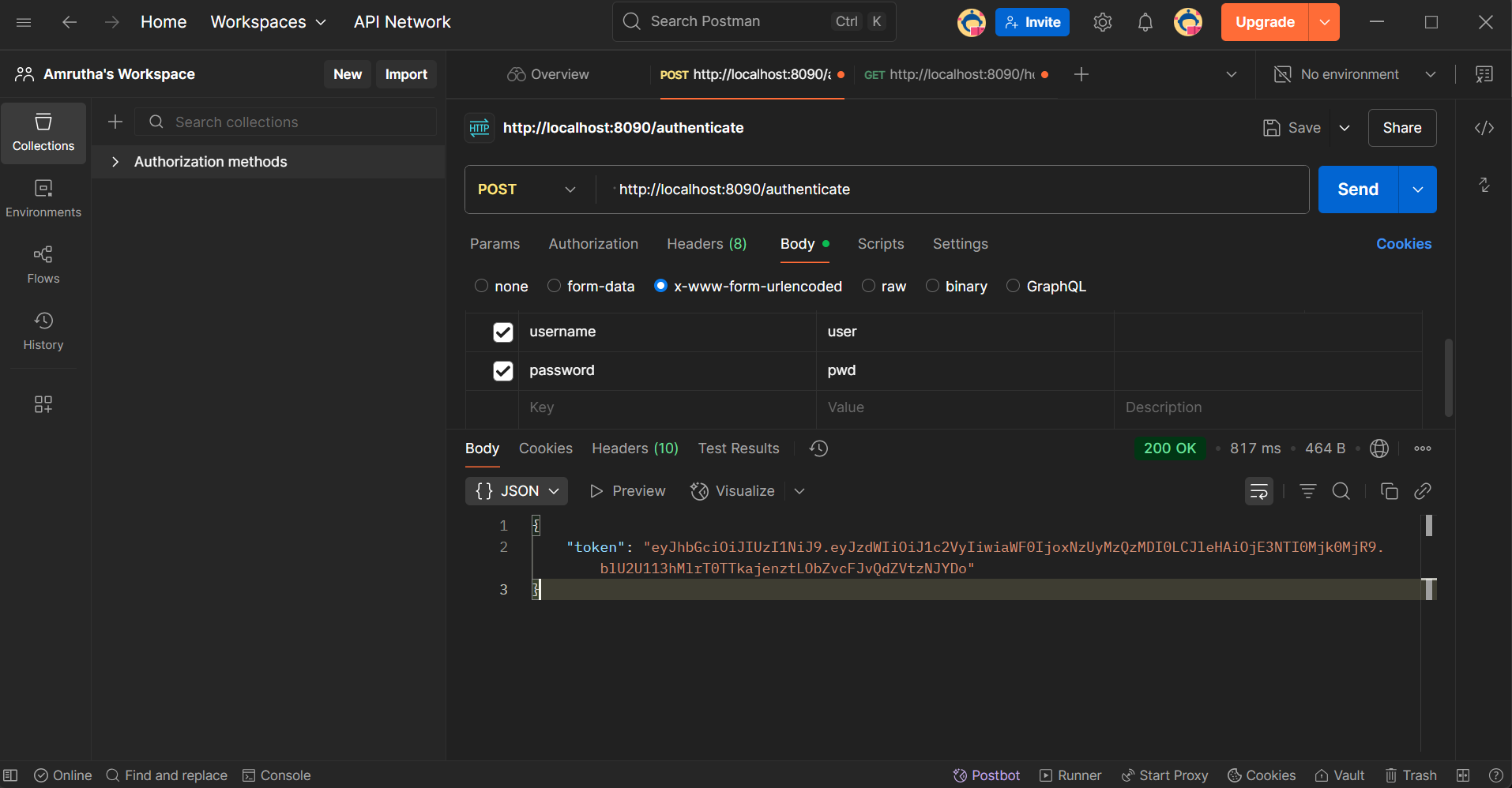
   - URL: `http://localhost:8090/authenticate`

   - Body: `x-www-form-urlencoded`

     - `username`: `user`

     - `password`: `pwd`

   - Click \*\*Send\*\* and copy the `token` from the response.



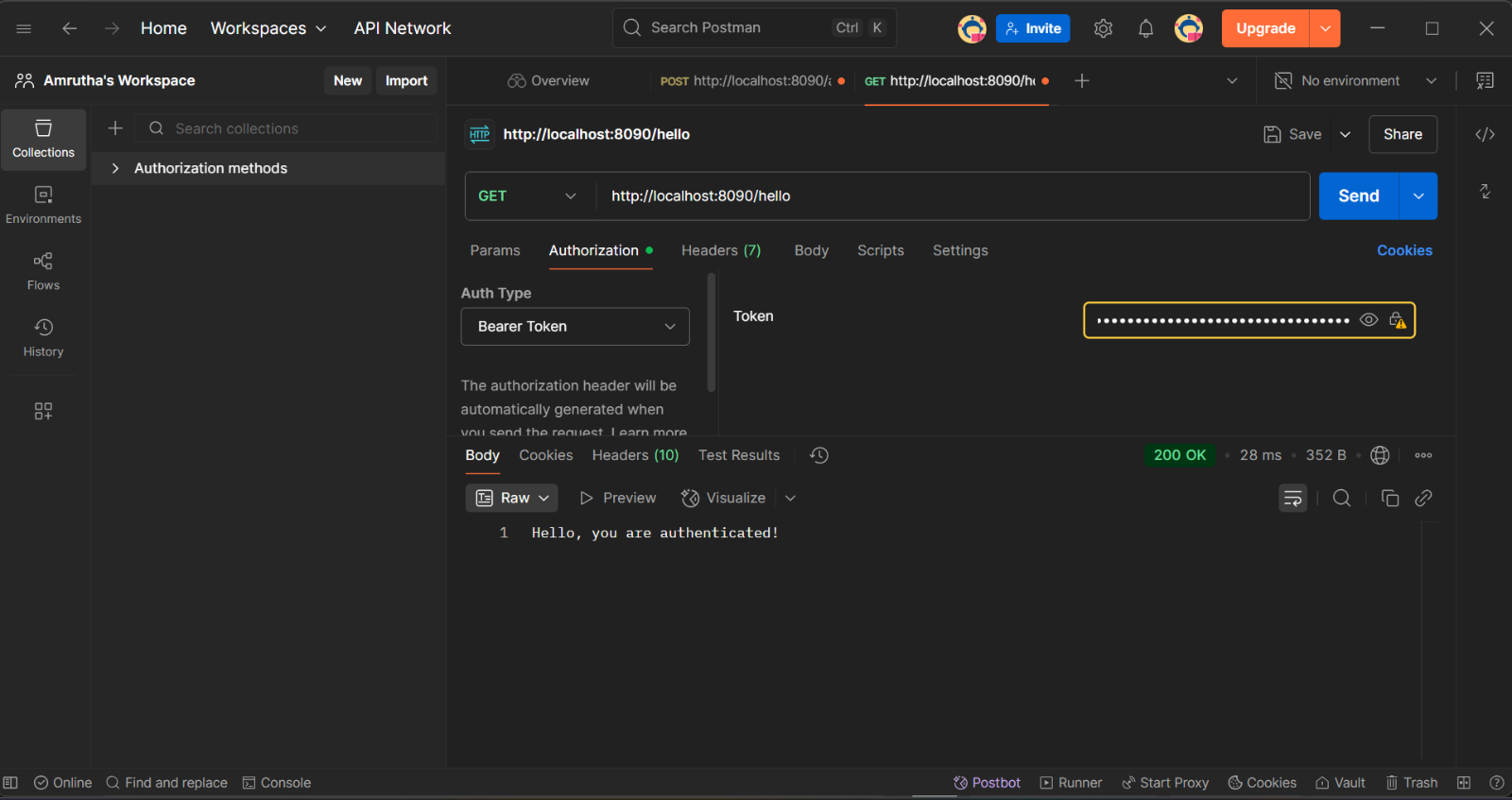
2. \*\*Access Protected Endpoint:\*\*

   - Method: GET

   - URL: `http://localhost:8090/hello`

   - Authorization: Bearer Token (paste the token from step 1)

   - Click \*\*Send\*\* to see the protected message.



**## Default Test User**

The application creates a test user automatically:

- Username: `user`

- Password: `pwd`

**## JWT Configuration**

JWT settings can be configured in `application.properties`:

```properties

jwt.secret=mySecretKey123456789012345678901234567890123456789012345678901234567890

jwt.expiration=86400000

jwt.header=Authorization

jwt.prefix=Bearer

**## Security Features**

- HTTP Basic Authentication for initial login

- JWT token generation with expiration

- Stateless session management

- Password encryption using BCrypt

- Role-based access control

**## H2 Console**

You can access the H2 database console at:

[http://localhost:8090/h2-console](http://localhost:8090/h2-console)

JDBC URL: `jdbc:h2:mem:testdb`

Username: `sa`

Password: `password`

**## Testing**

You can test the application using the provided curl commands or any REST client like Postman.