**Documentation for GreenPioneers Hackathon project:**

ECOTRACK

# I. Introduction

EcoTrack is a web application designed to encourage eco-friendly habits by allowing users to log their sustainable activities, track their environmental impact, and share their progress with a community. The app aims to bridge the gap between individual actions and collective environmental change, making sustainability accessible and rewarding.

Key Features:

- User authentication (register/login).
- Log eco-friendly activities with descriptions and images.
- Track environmental impact (e.g., CO2 saved, waste reduced).
- PostgreSQL database for persistent storage.
- RESTful API for seamless frontend-backend communication.

*Name : Akoujan Ali*
*Contact : akoujanali37@gmail.com*

# II. Technologies Used

## Backend:

- Java 17: For its robustness and widespread use in enterprise applications.

- Spring Boot 3.8:  For rapid development and seamless integration with other Spring tools.

- Spring Security: For secure JWT-based authentication.

- Spring Data JPA: To simplify database interactions.

- Hibernate: As the ORM (Object-Relational Mapping) tool.

- PostgreSQL: A reliable and efficient relational database.

- Maven: For dependency management and project building.

## Frontend (Incomplete):

- Basic HTML/CSS/JS for the welcome, login, and register pages.

## Tools:

- GitHub: For version control and code hosting.

- Postman: For API testing and documentation.

**Documentation for GreenPioneers Hackathon project:**

ECOTRACK

# I. Backend Architecture

## Code Organization:

The backend is organized into the following packages:

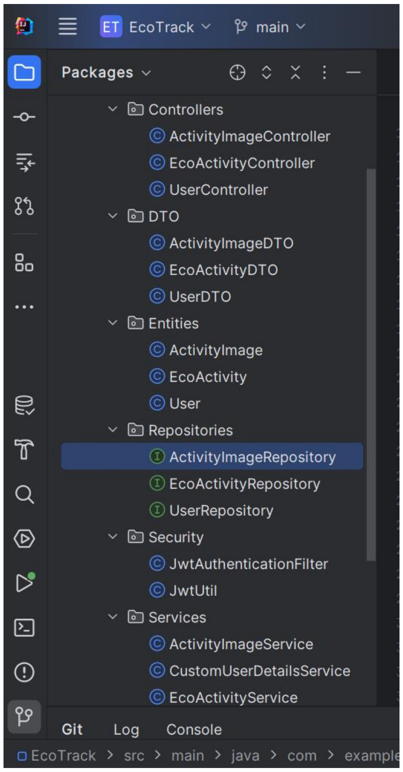
- `com.example.ecotrack.controllers`: Contains REST controllers for handling API requests.

- `com.example.ecotrack.services`: Implements business logic.

- `com.example.ecotrack.repositories`: Interfaces for database operations.

- `com.example.ecotrack.entities`: Defines entity classes (e.g., `User`, `Activity`).

- `com.example.ecotrack.security`: Handles JWT authentication and authorization.



```
Controllers
    ActivityImageController
    EcoActivityController
    UserController
DTO
    ActivityImageDTO
    EcoActivityDTO
    UserDTO
Entities
    ActivityImage
    EcoActivity
    User
Repositories
    ActivityImageRepository
    EcoActivityRepository
    UserRepository
Security
    JwtAuthenticationFilter
    JwtUtil
Services
    ActivityImageService
    CustomUserDetailsService
    EcoActivityService
```

Caption: *"Spring Boot project structure with controllers, services, repositories, and security configurations."*

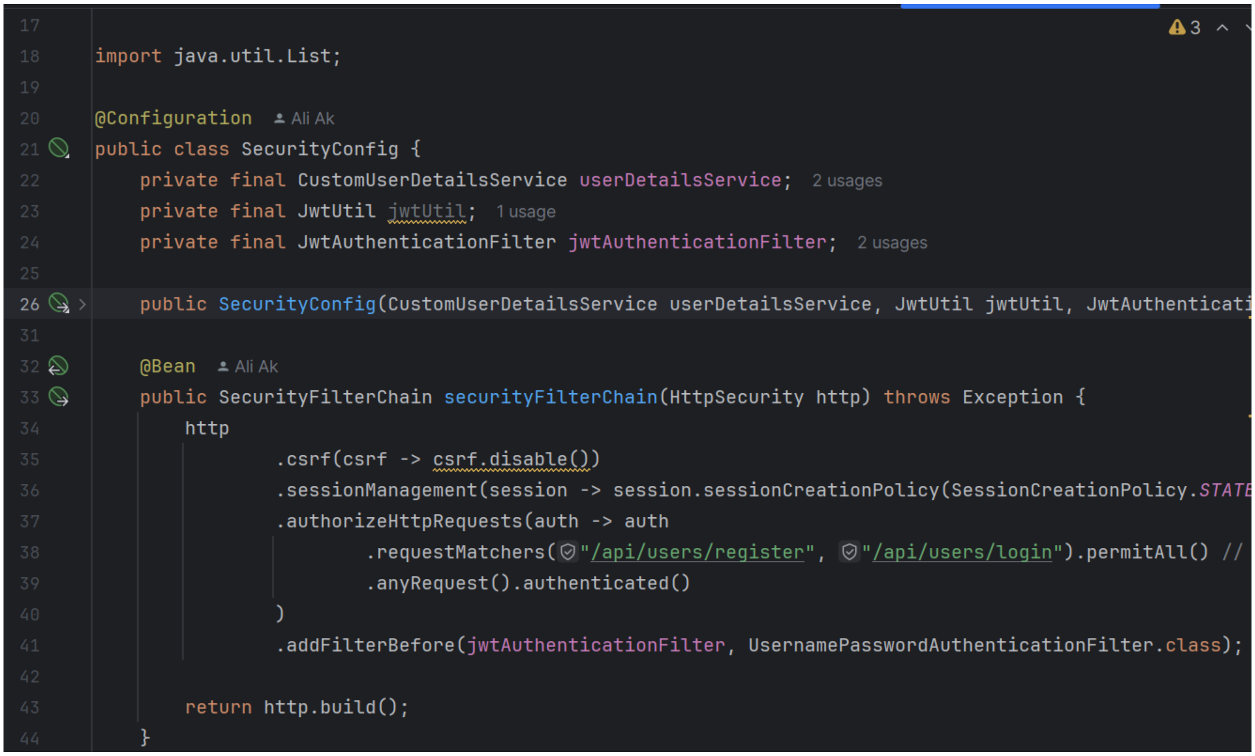
*Name : Akoujan Ali*
*Contact : akoujanali37@gmail.com*

# Documentation for GreenPioneers Hackathon project:

ECOTRACK

## Authentication Flow:

Spring Security is configured to use JWT for authentication. When a user logs in, the backend generates a JWT, which is then included in the `Authorization` header for subsequent requests.



```
import java.util.List;

@Configuration  Ali Ak
public class SecurityConfig {
    private final CustomUserDetailsService userDetailsService;  2 usages
    private final JwtUtil jwtUtil;  1 usage
    private final JwtAuthenticationFilter jwtAuthenticationFilter;  2 usages

    public SecurityConfig(CustomUserDetailsService userDetailsService, JwtUtil jwtUtil, JwtAuthenticati

    @Bean  Ali Ak
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
                .csrf(csrf -> csrf.disable())
                .sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.STATE
                .authorizeHttpRequests(auth -> auth
                        .requestMatchers("/api/users/register", "/api/users/login").permitAll() //
                        .anyRequest().authenticated()
                )
                .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class);

        return http.build();
    }
```

*Caption: "Spring Security configuration enabling JWT authentication."*

*Name : Akoujan Ali*
*Contact : akoujanali37@gmail.com*

**Documentation for GreenPioneers Hackathon project:**

ECOTRACK

# IV. Database Design

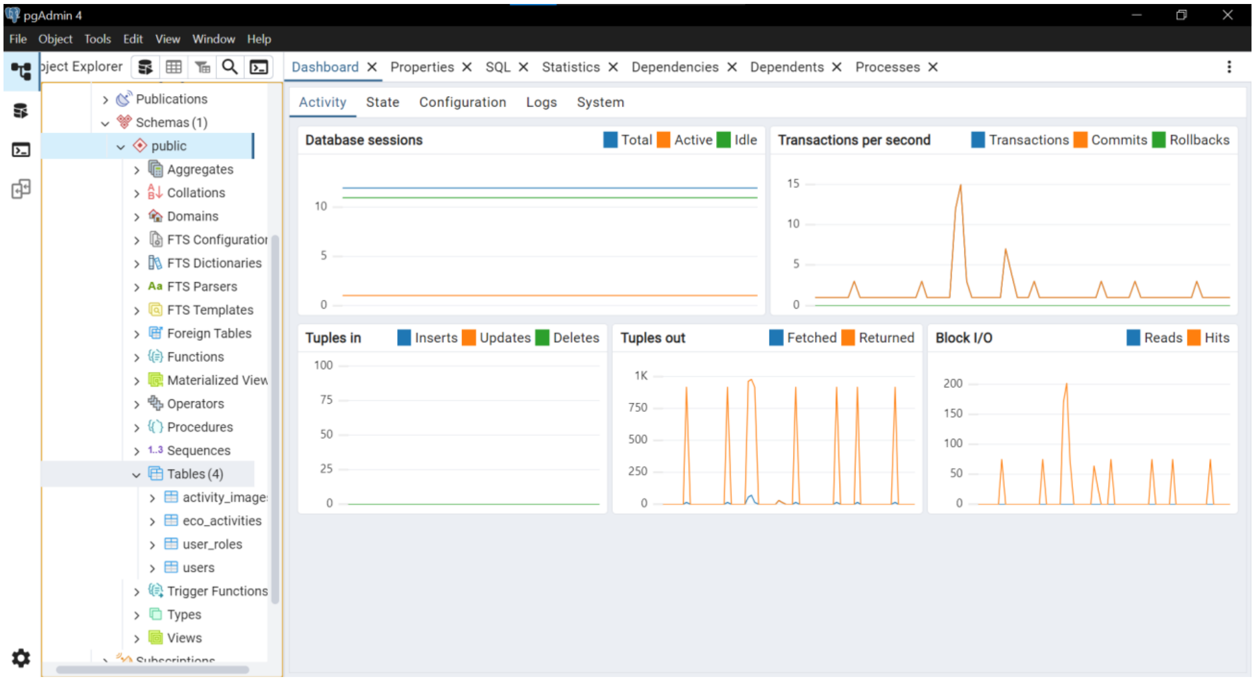
## Entity-Relationship Diagram (ERD)

The database consists of the following tables:

- `users`: Stores user information (e.g., email, password).

- `activities`: Stores eco-friendly activities (e.g., description, impact metrics).

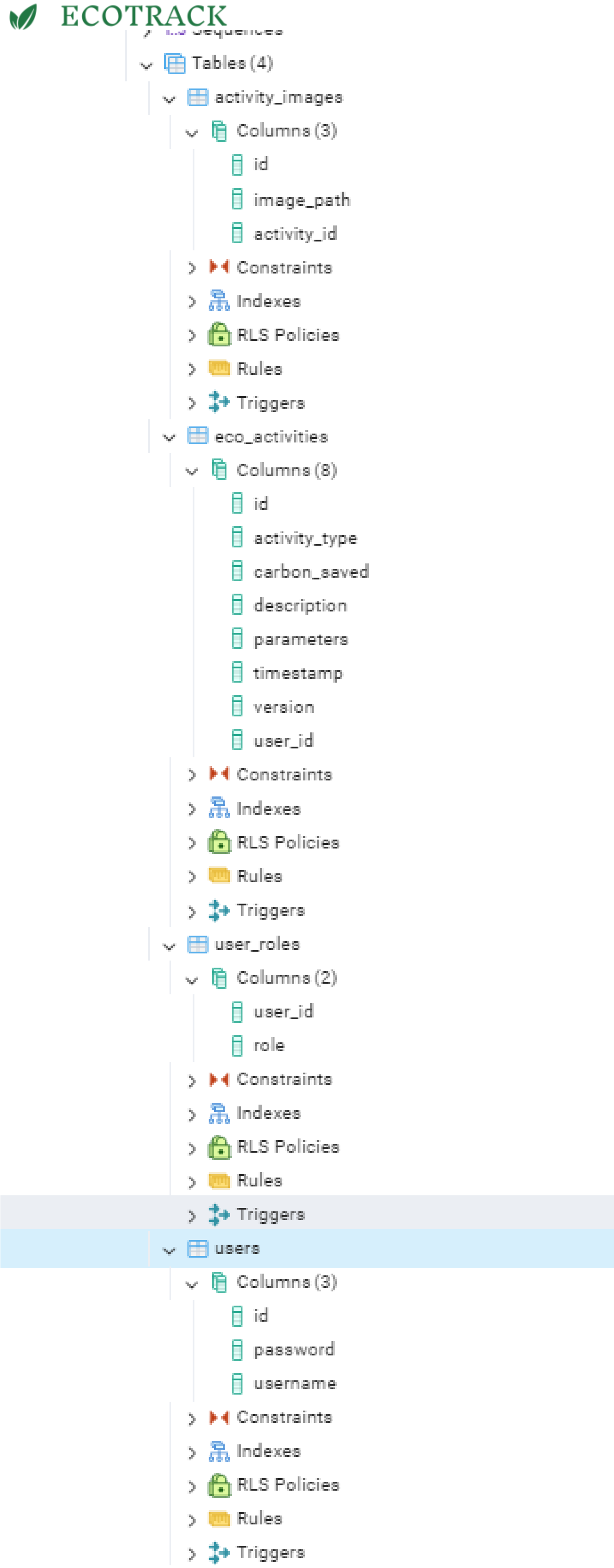
- `images`: Stores image URLs associated with activities.



*Name : Akoujan Ali*
*Contact : akoujanali37@gmail.com*

# Documentation for GreenPioneers Hackathon project:

ECOTRACK

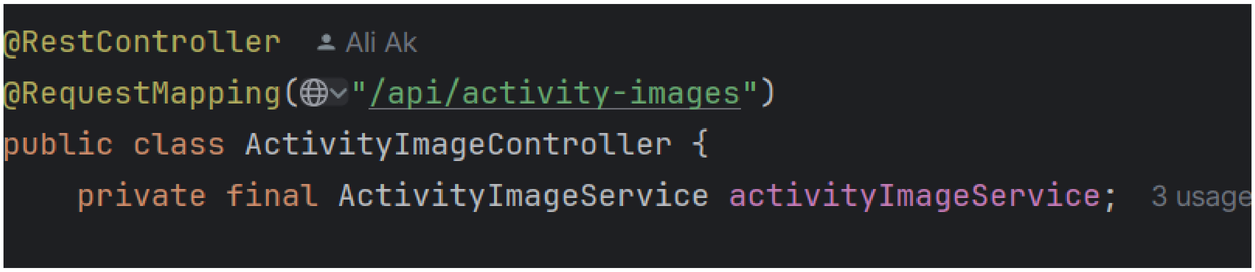


```
> Sequences
v Tables (4)
  v activity_images
    v Columns (3)
        id
        image_path
        activity_id
    > Constraints
    > Indexes
    > RLS Policies
    > Rules
    > Triggers
  v eco_activities
    v Columns (8)
        id
        activity_type
        carbon_saved
        description
        parameters
        timestamp
        version
        user_id
    > Constraints
    > Indexes
    > RLS Policies
    > Rules
    > Triggers
  v user_roles
    v Columns (2)
        user_id
        role
    > Constraints
    > Indexes
    > RLS Policies
    > Rules
    > Triggers
  v users
    v Columns (3)
        id
        password
        username
    > Constraints
    > Indexes
    > RLS Policies
    > Rules
    > Triggers
```

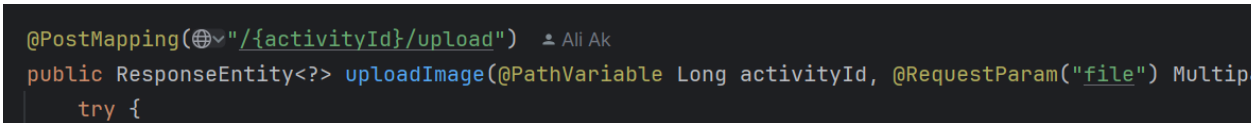
*Caption:* *"PostgreSQL schema for user and activity management."*

## Endpoints

The following RESTful API endpoints are implemented:

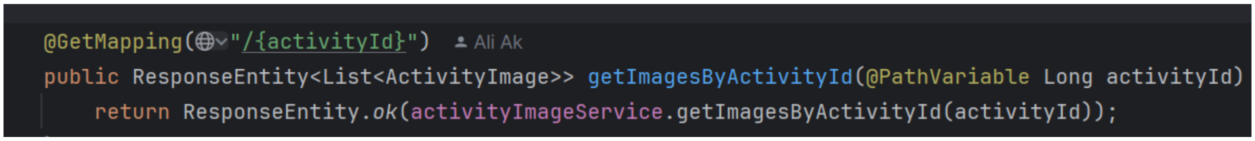


```
@RestController  Ali Ak
@RequestMapping("/api/activity-images")
public class ActivityImageController {
    private final ActivityImageService activityImageService;  3 usage
```



```
@PostMapping("/{activityId}/upload")  Ali Ak
public ResponseEntity<?> uploadImage(@PathVariable Long activityId, @RequestParam("file") Multip
    try {
```

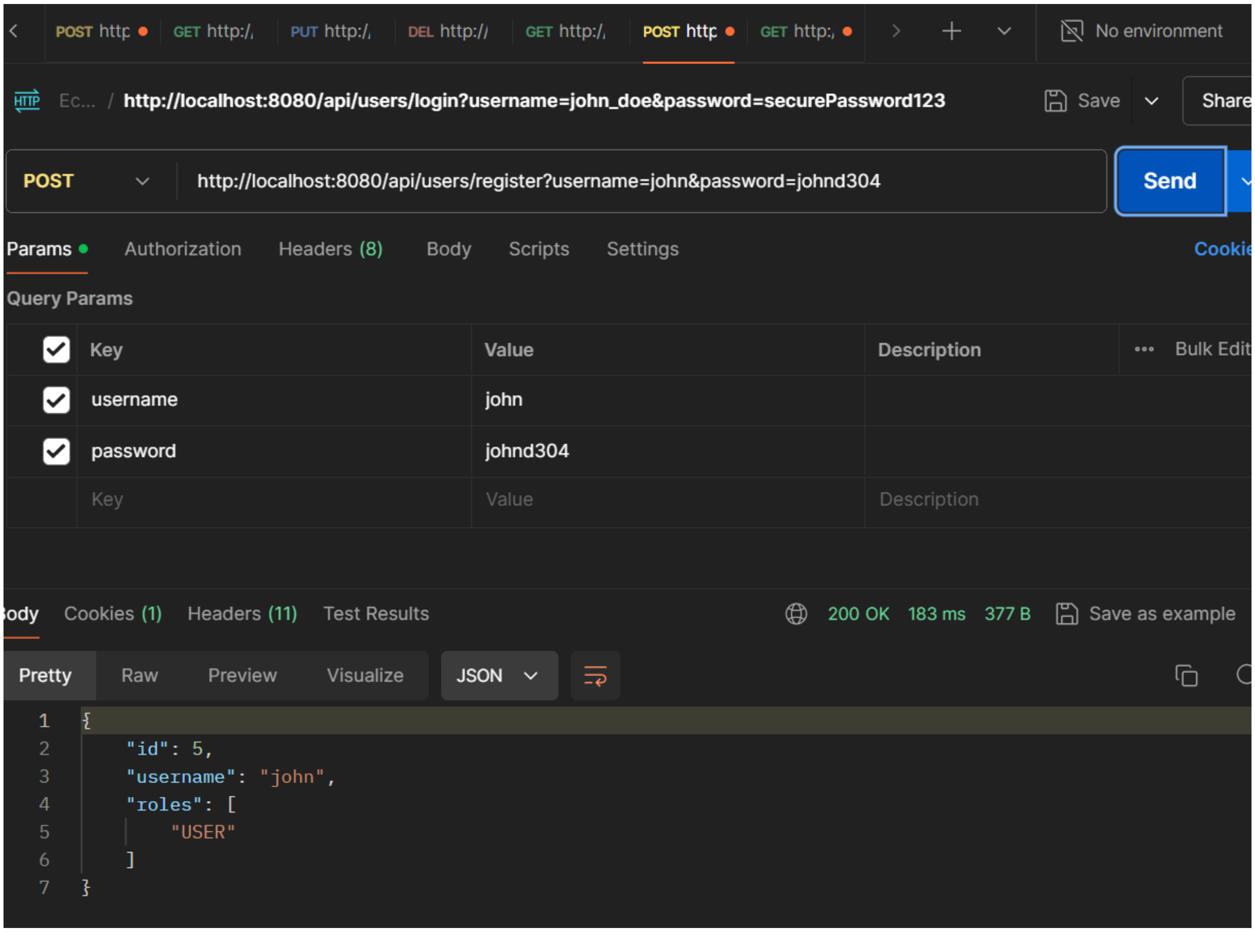
\



```
@GetMapping("/{activityId}")  Ali Ak
public ResponseEntity<List<ActivityImage>> getImagesByActivityId(@PathVariable Long activityId)
    return ResponseEntity.ok(activityImageService.getImagesByActivityId(activityId));
```

# Documentation for GreenPioneers Hackathon project:

ECOTRACK



http://localhost:8080/api/users/login?username=john_doe&password=securePassword123

POST http://localhost:8080/api/users/register?username=john&password=johnd304

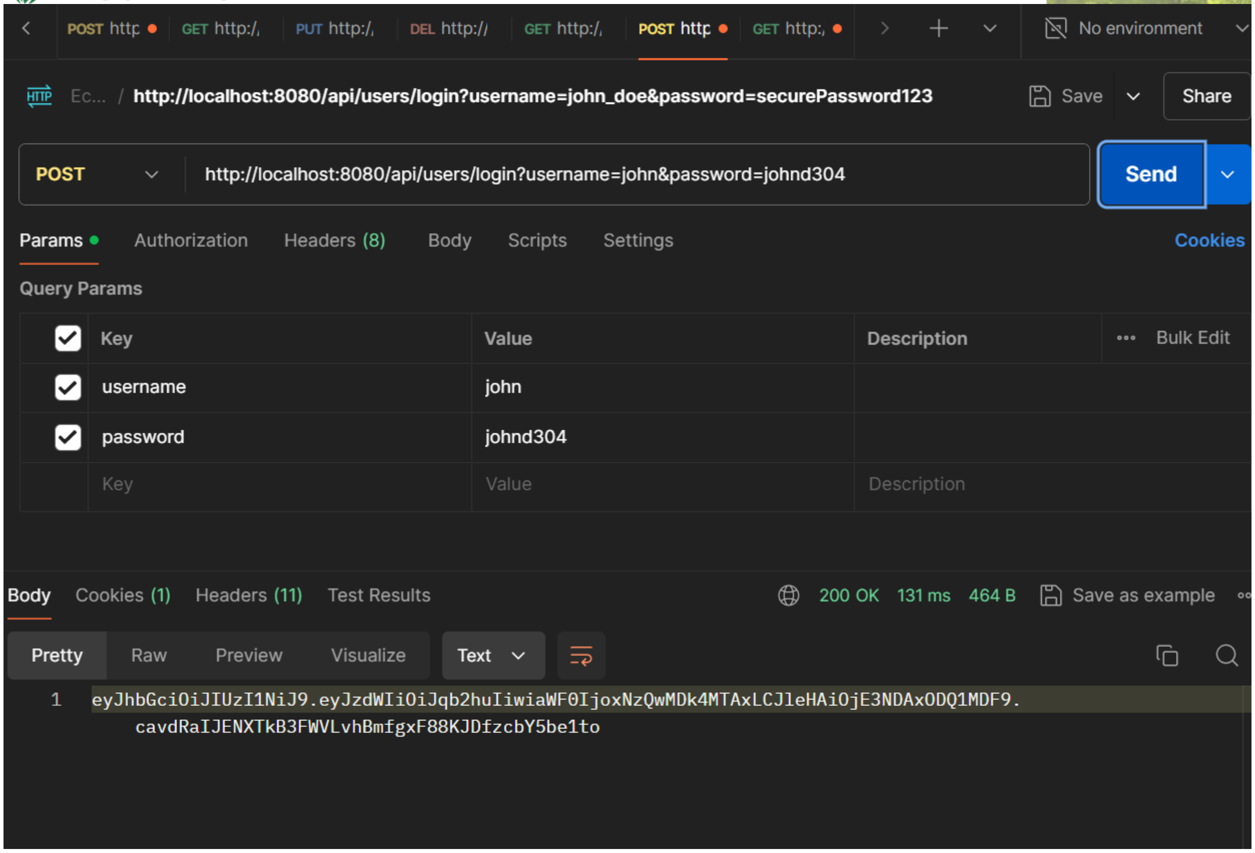
Query Params

| | Key | Value | Description |
|---|---|---|---|
| ☑ | username | john | |
| ☑ | password | johnd304 | |
| | Key | Value | Description |

200 OK 183 ms 377 B

```
{
    "id": 5,
    "username": "john",
    "roles": [
        "USER"
    ]
}
```

*Name : Akoujan Ali*
*Contact : akoujanali37@gmail.com*



POST http://localhost:8080/api/users/login?username=john&password=johnd304

Query Params

| ☑ Key | Value | Description |
|---|---|---|
| ☑ username | john | |
| ☑ password | johnd304 | |
| Key | Value | Description |

200 OK 131 ms 464 B

```
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJqb2huIiwiaWF0IjoxNzQwMDk4MTAxLCJleHAiOjE3NDAxODQ1MDF9.
    cavdRaIJENXTkB3FWVLvhBmfgxF88KJDfzcbY5be1to
```

*Caption:* *"Postman test for the `/api/users/login` endpoint."*

---

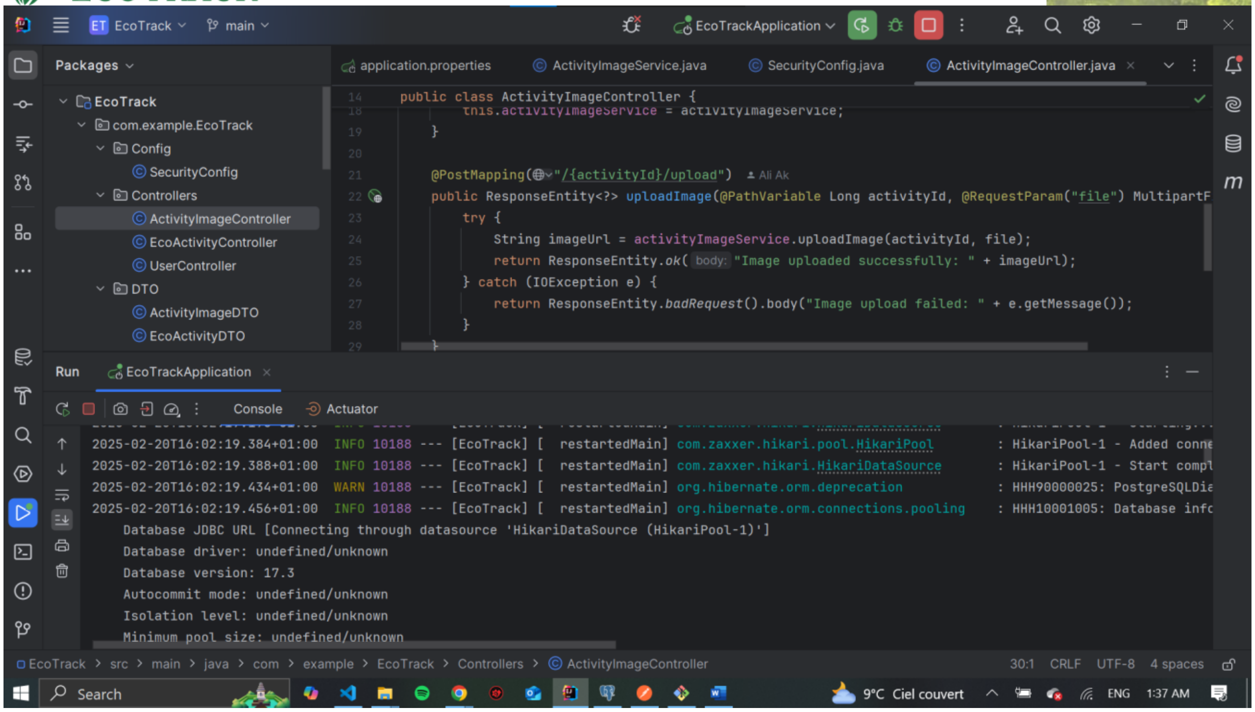
# V. Screenshots

## Backend

### 1.Code Organization:

# Documentation for GreenPioneers Hackathon project:

ECOTRACK



```java
public class ActivityImageController {
        this.activityImageService = activityImageService;
    }

    @PostMapping("/{activityId}/upload")
    public ResponseEntity<?> uploadImage(@PathVariable Long activityId, @RequestParam("file") MultipartF
        try {
            String imageUrl = activityImageService.uploadImage(activityId, file);
            return ResponseEntity.ok( body: "Image uploaded successfully: " + imageUrl);
        } catch (IOException e) {
            return ResponseEntity.badRequest().body("Image upload failed: " + e.getMessage());
        }
}
```

```
2025-02-20T16:02:19.384+01:00  INFO 10188 --- [EcoTrack] [  restartedMain] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added conne
2025-02-20T16:02:19.388+01:00  INFO 10188 --- [EcoTrack] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start compl
2025-02-20T16:02:19.434+01:00  WARN 10188 --- [EcoTrack] [  restartedMain] org.hibernate.orm.deprecation            : HHH90000025: PostgreSQLDia
2025-02-20T16:02:19.456+01:00  INFO 10188 --- [EcoTrack] [  restartedMain] org.hibernate.orm.connections.pooling    : HHH10001005: Database infc
	Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
	Database driver: undefined/unknown
	Database version: 17.3
	Autocommit mode: undefined/unknown
	Isolation level: undefined/unknown
	Minimum pool size: undefined/unknown
```

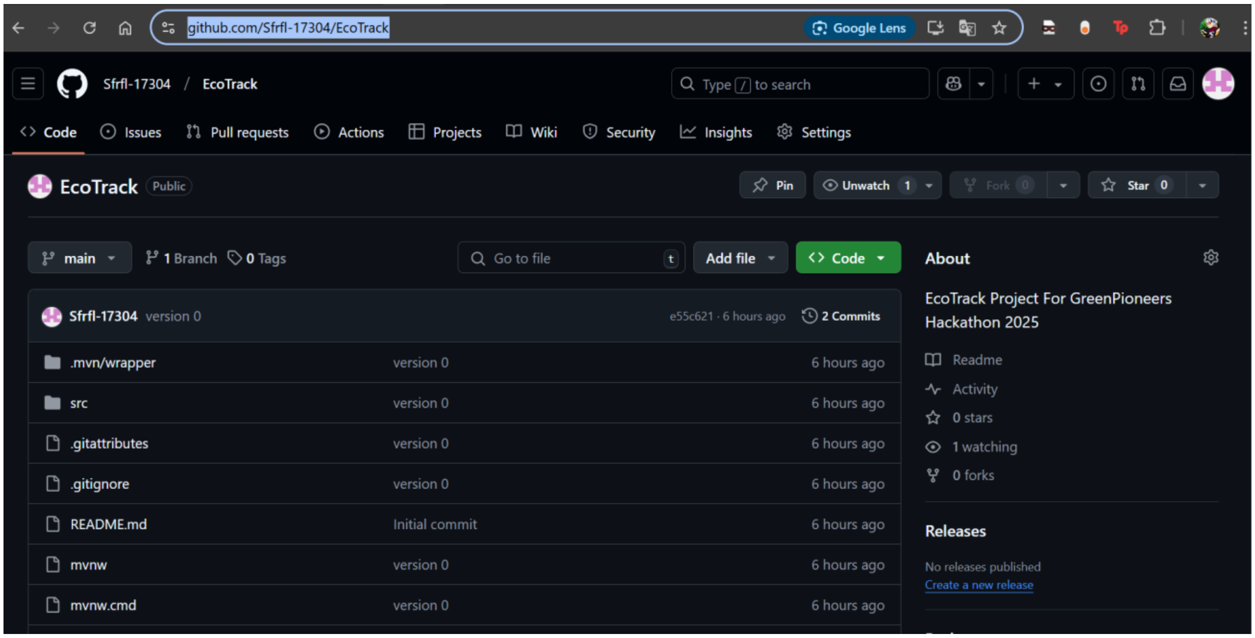
*Caption:* *"Spring Boot project structure in IntelliJ IDEA."*

## 2. GitHub Repository:

*Name : Akoujan Ali*
*Contact : akoujanali37@gmail.com*

# Documentation for GreenPioneers Hackathon project:

ECOTRACK



Sfrfl-17304 version 0 e55c621 · 6 hours ago 2 Commits

| | | |
|---|---|---|
| .mvn/wrapper | version 0 | 6 hours ago |
| src | version 0 | 6 hours ago |
| .gitattributes | version 0 | 6 hours ago |
| .gitignore | version 0 | 6 hours ago |
| README.md | Initial commit | 6 hours ago |
| mvnw | version 0 | 6 hours ago |
| mvnw.cmd | version 0 | 6 hours ago |

About

EcoTrack Project For GreenPioneers Hackathon 2025

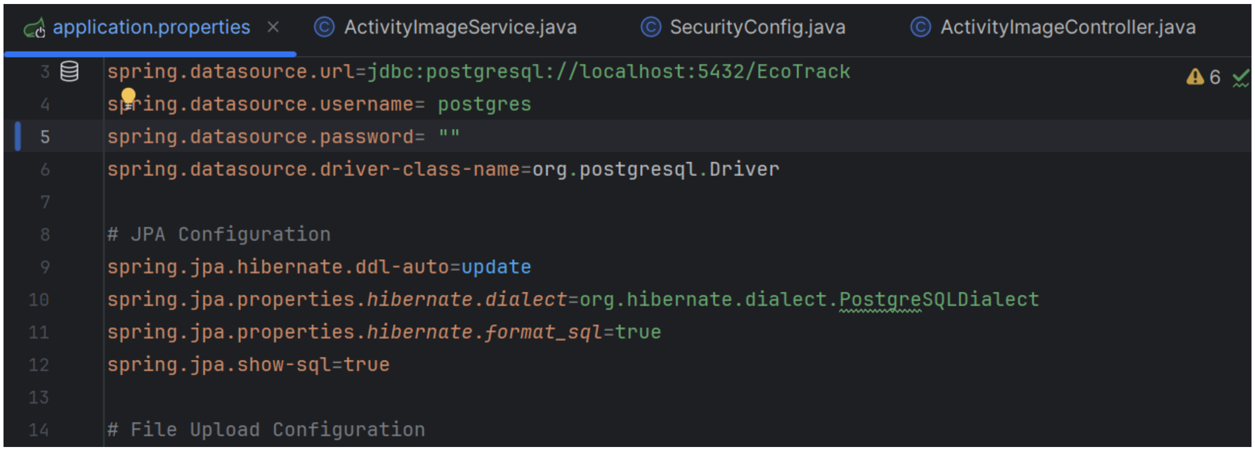
Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

*Caption:* "GitHub repository for EcoTrack."

## 3. PostgreSQL Setup:



```
spring.datasource.url=jdbc:postgresql://localhost:5432/EcoTrack
spring.datasource.username= postgres
spring.datasource.password= ""
spring.datasource.driver-class-name=org.postgresql.Driver

# JPA Configuration
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.show-sql=true

# File Upload Configuration
```

*Caption:* "PostgreSQL configuration in `application.properties`."

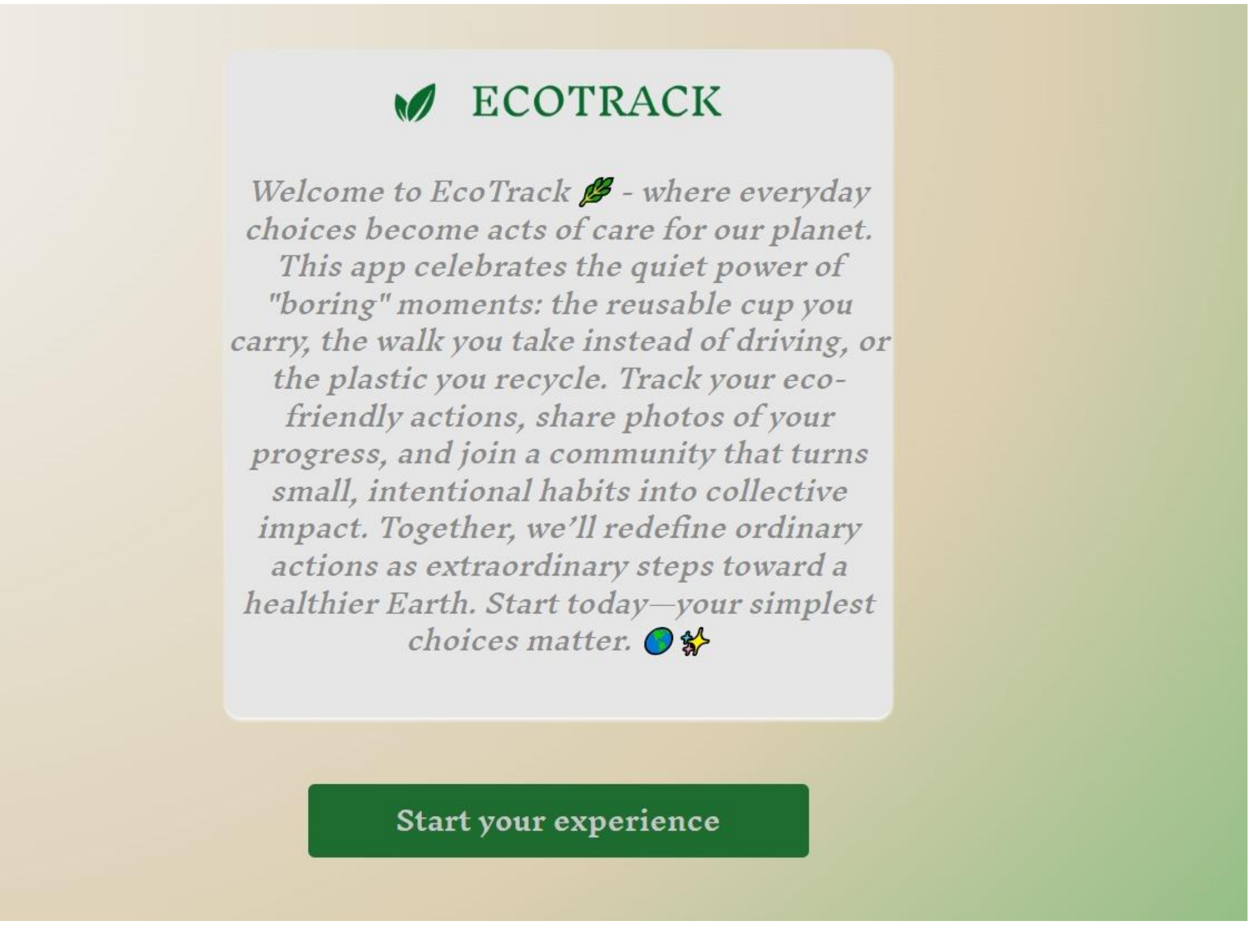
*Name : Akoujan Ali*
*Contact : akoujanali37@gmail.com*

# Documentation for GreenPioneers Hackathon project:

ECOTRACK

Frontend

**1.Welcome Page:**



ECOTRACK

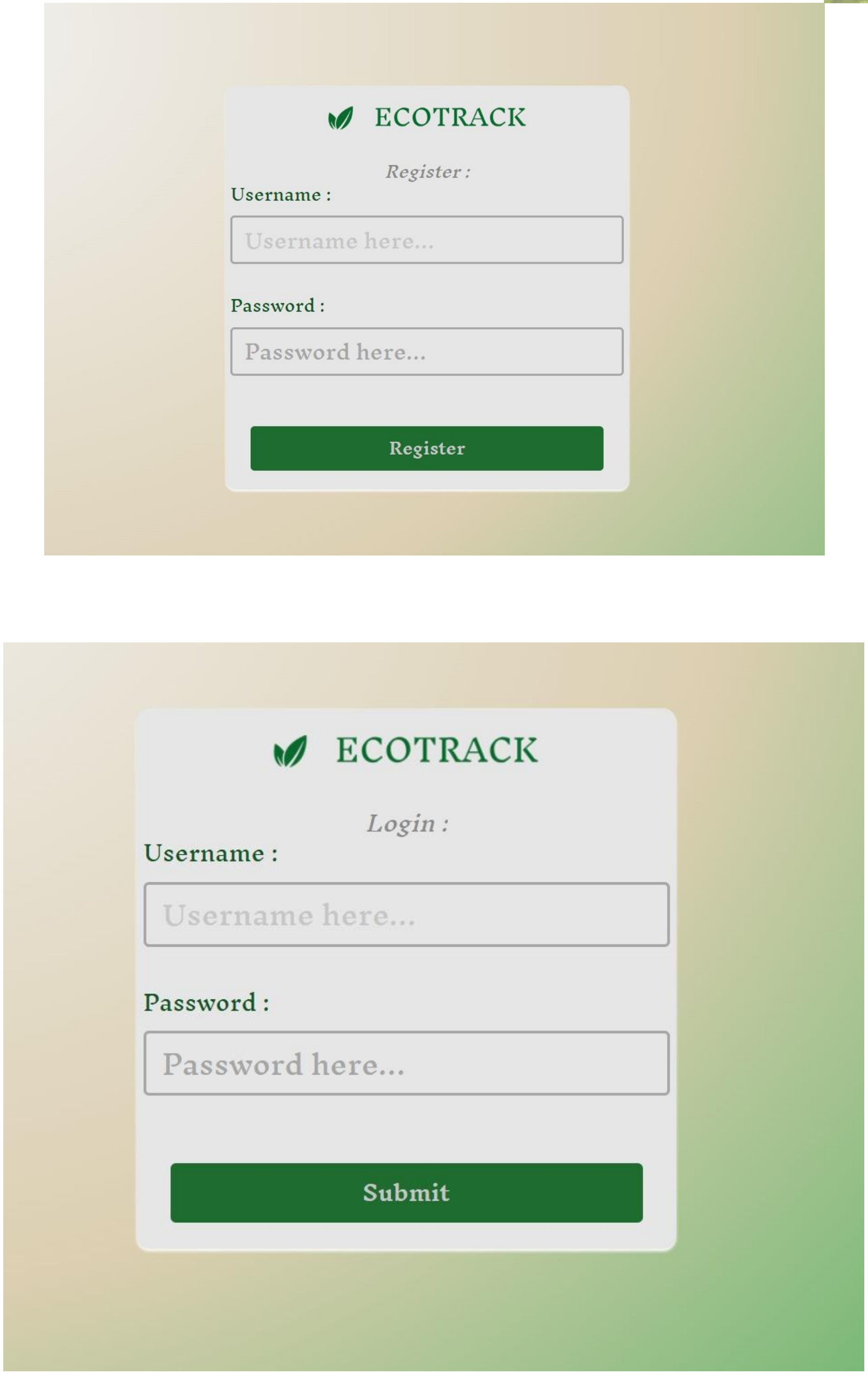
*Welcome to EcoTrack 🌿 - where everyday choices become acts of care for our planet. This app celebrates the quiet power of "boring" moments: the reusable cup you carry, the walk you take instead of driving, or the plastic you recycle. Track your eco-friendly actions, share photos of your progress, and join a community that turns small, intentional habits into collective impact. Together, we'll redefine ordinary actions as extraordinary steps toward a healthier Earth. Start today—your simplest choices matter. 🌍✨*

*Caption*: "EcoTrack welcome page."

**2.Login/Register Pages:**

# Documentation for GreenPioneers Hackathon project:

ECOTRACK



*Caption*: "Login and register pages for EcoTrack."

*Name : Akoujan Ali*
*Contact : akoujanali37@gmail.com*

# VI. Future Work

- Frontend Completion: Integrate with React/Angular for a polished UI.

- Gamification: Add badges/leaderboards for user engagement.

- Image Recognition: Use AI to verify eco-friendly activities (e.g., recycling bins in images).

- Mobile App: Develop a cross-platform mobile app using Flutter or React Native.

# VII. References

- GitHub Repository: https://github.com/Sfrfl-17304/EcoTrack

# VIII. Conclusion

EcoTrack demonstrates how technology can empower individuals to contribute meaningfully to sustainability efforts. By combining a robust backend with an intuitive frontend (in progress), the app encourages users to adopt eco-friendly habits and share their progress with a community. This project highlights the potential of small, everyday actions to drive significant environmental change.

*Name : Akoujan Ali*
*Contact : akoujanali37@gmail.com*

# Documentation for GreenPioneers Hackathon project:

ECOTRACK

*Name : Akoujan Ali*
*Contact : akoujanali37@gmail.com*