2024 Geotechnical Testing Guidance Handover Document

Prerequisites

To access the files for our project use this command in a terminal, use the command git clone https://github.com/spe-uob/2024-GeotechnicalTestingGuidance.git (you may need to create a github account first at https://github.com/). This will download all files onto your computer so that the website can be viewed locally. We also have a website at “PUT WEBSITE HERE” this will only be up for a few months due to restrictions on our course and Amazon Web Services. To get the website running on a local machine you will need to download the following:

- Docker Desktop
- An IDE (Integrated Development Environment) of any kind

Docker can be downloaded at https://www.docker.com/products/docker-desktop/, you will also need to create an account for this.

For an IDE we recommend VS Code, downloadable at https://code.visualstudio.com/download.

When you start up VSC, all open the project files, you may be prompted to download extensions for all the languages in the files, this should be done otherwise the code cannot properly run. From here you should be able to run the website locally.

1. Project Overview

- Purpose: Web-based system for managing geotechnical test data following international standards
- Key Features:
  - Test data entry/management
  - User role management (Admin/Engineer/Viewer)
  - Rock & soil sample tracking
  - Compliance reporting
- Tech Stack:
  - Frontend: React.js + CSS Modules

    - Backend: Spring Boot (Java 17)
    - Database: MySQL 9.0
    - Infrastructure: Docker + GitHub Actions CI/CD

2. System Setup

To start the site on your computer, you should open up a terminal in your IDE and enter into the project files using "cd 2024-GeotechnicalTestingGuidance". To initialise the database, run these commands.

```
cd backend/database
mvn clean package
```

Once this process has finished, run cd ../.. to return to the correct folder. From here run "docker compose up –build" to start the site. If this returns an error, make sure docker desktop is open and that you are logged into docker by running the command "docker login". This should start the website, which can be accessed at http://localhost:3100/ on any internet browser.

Once you are on the site, you will be asked to login. There is currently only one account, with the username "admin" and password "adminpass", however once logged in there is the option to create others with either user or admin privileges. As a user, you will be able to view all the tests and their parameters, but a person with an admin role can further edit these tests, and also add and delete others.

3. Architecture Overview

The system follows a 3-tier modular architecture:

1. Presentation Layer: React-based frontend handling UI/UX with role-based views
2. Application Layer: Spring Boot backend exposing REST APIs with:
    - JWT authentication flow
    - Role-based access control
    - Data validation against ASTM/DIN standards
3. Data Layer: MySQL database with:
    - Normalized test data schema
    - Audit tables for compliance tracking
    - Batch processing capabilities

The architecture implements a microservices-ready design using Docker containerization, with clear separation between:

- Frontend services (Node.js runtime)
- Backend services (Java 17/JVM)

- Database layer (MySQL 9.0 cluster)

Key Components

Component	Location	Description
API Gateway	`backend/database/src/main/java/com/example/accessingdatamysql/WebConfig.java`	CORS configuration
Security Layer	`backend/database/src/main/java/com/example/accessingdatamysql/config/SecurityConfig.java`	JWT authentication
Data Models	`backend/database/src/main/java/com/example/accessingdatamysql/GeotechnicalEntry.java`	Core entity definitions

4. API Documentation

Core Endpoints

```
// User Management (UserController.java)
POST /api/users/add - Create new user
GET /api/users/all - List all users (Admin only)

// Test Data (RocksUserController.java)
POST /api/tests/add - Submit new test results
GET /api/tests/{id} - Get test details
```

5. Testing & Validation

5.1 Automated Testing

```
# Run all backend tests
mvn test

# Frontend validation
npm run test:coverage

# Standards compliance check
npm run validate-standards
```

5.2 Manual Testing

1. Postman Collection: `/tests/GeotechnicalAPI.postman_collection.json`
   - Contains 42 test scenarios covering:
   - Authentication flows
   - Data validation edge cases
   - Role-based access control
2. Performance Testing

```
# Run load test (requires k6)
k6 run tests/load-test.js
```

5.3 Audit Trails

- All test results stored in `/tests/results/`
- Compliance reports generated after each deployment

6. Deployment Checklist

1. Update `docker-compose.yml` environment variables

Run database migration:
```
./migrate.sh production
```

7. Maintenance Contacts

Role	Contact	Availability
Lead Developer (Backend)	Millar Scandrett	ju22534@bristol.ac.uk
Backend Development	Dillan Hankinson	eu23444@bristol.ac.uk
Full-stack Integration	Kecen Li	yo22285@bristol.ac.uk
Frontend Development	Christopher Chen	ak23423@bristol.ac.uk
Frontend Development	Zhenghang Shao	qo23397@bristol.ac.uk