CS 319 Object-Oriented Software Engineering

Instructor: Eray Tüzün, Teaching Assistant: Yahya Elnouby

AdayBilgi
Deliverable 5 - Final Report

Group 1

Emine Noor

Eray İşçi

Hatice Kübra Çağlar

İbrahim Çaycı

İrem Damla Karagöz

Yiğit Özhan

Table of Contents

1. User's Guide

Prerequisites

1. **Development Environment**
    - Install **Java JDK 17+** for the backend.
    - Install **Node.js (v16 or higher)** and **npm** for the frontend.
    - Install a database tool like **MySQL Workbench** to connect to the database.
    - Install **Maven** (optional) for building the backend if not using the built-in wrapper.
    - Install an **IDE** (e.g., IntelliJ IDEA for backend, VS Code for frontend).
2. **Database Configuration**
    - Set up a MySQL instance (local or remote).
    - Configure your database connection string in the backend's application.properties file.
3. **Environment Variables**
    - Ensure you have configured any sensitive environment variables like JWT keys, SMTP credentials, or database passwords.

Step 1: Clone the Repository

```
git clone https://github.com/damlakragoz/AdayBilgi_CS319.git
cd AdayBilgi_CS319

# Checkout the branches
git checkout backend_final  # For backend
```

Step 2: Backend Setup (backend_final)

1. **Navigate to the Backend Directory**

```
cd BTO_Application  # The backend project folder
```

2. Configure application.properties

→ Locate the application.properties file under src/main/resources. Make sure it is like as follows:

```
# JDBC URL for your MySQL database

spring.datasource.url=jdbc:mysql://bto-database.cvgoaqoq62o0.us-east-1.rds.amazonaws.com/bto_database

# Username and password for your MySQL instance

spring.datasource.username=admin

spring.datasource.password=bto_database

# Hibernate settings

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.show-sql=false

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl

# Optional: Set the server port if you want a specific port for the Spring Boot application

server.port=8081

# ================================================

# JWT and Security Configuration (Optional - Example)

# ================================================

# Secret key for JWT token (use a strong, secure key)

# jwt.secret=your_secure_secret_key_here

# Expiration time for JWT token in milliseconds (e.g., 3600000 = 1 hour)

jwt.secret = KzNDLS0L3lc9Y/tPM9Zrt0Qpk2+H0RzcW3ISRkE7mgg=

jwt.expirationMs = 3600000

# ================================================

# Email Configuration (Add Mail Service Config)

# ================================================
```

SMTP server configuration (for example, Gmail SMTP)

spring.mail.host=smtp.gmail.com

spring.mail.port=587

spring.mail.username=btomailservice@gmail.com

spring.mail.password=addbkjjkaptkanoc

spring.mail.properties.mail.smtp.auth=true

spring.mail.properties.mail.smtp.starttls.enable=true

Suppress Hibernate SQL logs

logging.level.org.hibernate.SQL=OFF

logging.level.org.hibernate.type.descriptor.sql.BasicBinder=OFF

3. Build and Run the Backend

   → Using the Maven Wrapper

```
./mvnw clean install  # For Linux/Mac
mvnw.cmd clean install # For Windows
./mvnw spring-boot:run
```

   → Or directly in IntelliJ IDEA

   - Open the backend project in IntelliJ.
   - Navigate to BtoApplication.java (main class) and run it.

4. Verify Backend

   → The backend will run on http://localhost:8081.

Step 3: Frontend Setup (frontend_final)

1. Clone the github to another directory
2. Checkout the frontend_final branch by running:

   ```
   git checkout frontend_final # For frontend
   ```
3. Navigate to the BTO_Application folder: ls BTO_Application
4. Navigate to 319_Frontend folder: ls 319_Frontend
5. Install Dependencies

   npm install
6. Build the Frontend for Production
   - Run the following command to create a production build:

   npm run build

   - The built files will be available in the build folder.
7. Install the missing dependencies:

   ```
   npm install react-chartjs-2 chart
   ```

   If there are any other missing dependencies, you can install them as well.
8. Serve the Application Locally

   npm start

   - The frontend will run on http://localhost:3000.
   - Make sure it runs on http://localhost:3000. Sometimes npm suggests it to be runned on a different port. If you want to change the port you must modify BTO_Application/src/main/java/com/CS319/BTO_Application/Config/WebConfig.java file in the backend_final branch from `.allowedOrigins("http://localhost:3000")` to your desired port.

Step 4: Running the Full Application

1. Start the **Backend**
   - Ensure the backend is running on http://localhost:8081.
2. Start the **Frontend**
   - Serve the React application on http://localhost:3000.
3. Open the frontend in your browser

    - Navigate to http://localhost:3000.
4. When logging into user accounts, each user's password is edited to be “{FirstName}123” for testing. You can check the email accounts on the User table.

2. Work Allocation

Note: Code contributions can be seen on our repository on different branches.

2.1 Emine Noor

- Developed Feedback, Profile and Payment functionalities on the backend.
- Designed mockup for Coordinator and updated them during the second iteration.
- Worked on frontend pages, including Homepage, Log In, Counselor Sign Up, Profile Upload, Settings, Individual Application, Counselor, Header, and Sidebar. Also added several pop-ups.
- Contributed to Sequence, Class, Detailed Class Diagrams (including edits), and worked on Design Patterns for Deliverable 4.
- Set up the RDS Database.

2.2 Eray İşçi

- Developed the Tour, Priority Management, Tour Application, and User Authentication functionalities. Worked especially on the priority algorithm for tour allocations.
- Designed mock-ups for Counselor and updated them during the second iteration.
- Worked on many frontend pages.
- Coordinated the team for GitHub usage and handled backend/frontend merges.
- Contributed to Activity, Class, Detailed Class, and State Diagrams.
- Worked on Design Patterns for Deliverable 4.

2.3 Hatice Kübra Çağlar

- Implemented Fair and Fair Invitation functionalities and Executive operations in the backend.
- Contributed to Activity, State, Sequence, Use Case, Class, and Detailed Class Diagrams. Designed mockup for Executive.
- Worked on Design Patterns for Deliverable 4 and edited Deliverable 3.
- Added several pop-ups and edited frontend pages, including headers and sidebars.

2.4 İbrahim Çaycı

- Worked on Use Case, Sequence, Class, and Detailed Class Diagrams, also the Use Case Textual Descriptions. Designed UI for Advisor.
- Implemented functionalities for Statistics, Notifications, and Mail Service on the backend.
- Added the Statistics, Notifications View, Change Password and Forgot Password Page.
- Added Contract comments to the backend.

2.5 İrem Damla Karagöz

- Developed functionalities for Tour, Tour Application, User Management (users, their methods etc.), and Admin.
- Worked on many frontend pages, and arranged layouts for different user types.
- Set up the database and connected most frontend pages to the backend.
- Designed the coordinator ui mockups.
- Worked on Sequence, Detailed Class, and Class Diagrams.
- Worked on Design Patterns for Deliverable 3.
- Edited the Demo video.

2.6 Yiğit Özhan

- Developed functionalities for Puantage, Withdraw Acceptance/Rejection, and Activity Entrance.
- Designed mock-up for Tour Guide and updated it in the second iteration.
- Worked on State, Activity and Use Case diagrams.
- Added All Available Tours and General Puantage Table pages. Made some small changes on navigation menus.