B. Deployment Guide

This section describes how to build and deploy the Airport Departure Planner backend using Apache Tomcat. It also lists the required Java version, server version, and third-party libraries used in the project. These steps are mainly intended for local deployment or redeployment when testing or debugging.

Deployment Environment Requirements

Java

- JDK / JRE Version: Java 17
- Java must be installed and available on the system `PATH`

To verify:

```
java -version
```

Application Server

- Apache Tomcat Version: 9.x

The backend is built as a WAR file and deployed directly to Tomcat.

Build Tool

- Apache Maven
- Used to compile the backend and package it into a WAR file

To verify:

```
mvn -version
```

Third-Party Libraries and Frameworks

The backend relies on the following libraries and APIs:

Java / Servlet Libraries

- Java Servlet API (`HttpServlet`, `HttpServletRequest`, `HttpServletResponse`)
    - Provided by Apache Tomcat
    - Used to handle HTTP requests and responses

JSON Handling

- Gson
    - Used to parse incoming JSON requests
    - Used to construct JSON responses sent back to the frontend

Google APIs (Backend Only)

The backend integrates with several Google APIs using HTTP requests:

- Google Places API
- Google Geocoding API
- Google Routes API
- Google Weather API

These APIs are accessed using:

- `HttpURLConnection`
- Secure API key stored as an environment variable

Required Environment Configuration

Before deploying the backend, a Google Maps API key must be configured as an environment variable.

Example:

```
GOOGLE_MAPS_API_KEY=your_api_key_here
```

This key must have access to:

- Places API
- Geocoding API
- Routes API
- Weather API

Billing must also be enabled in the Google Cloud project.

Backend Build and Deployment Steps

Step 1: Stop Tomcat

Before deploying a new version, stop the Tomcat server.

Windows:

```
shutdown.bat
```

Mac / Linux:

```
./shutdown.sh
```

Step 2: Remove Old Deployment (Recommended)

To avoid cached or stale files, remove the existing backend deployment from Tomcat.

Delete:

- `webapps/backend`
- `webapps/backend.war`
- `work/Catalina/localhost/backend`

This step helps prevent issues such as old code running or unexpected 404 errors.

Step 3: Build the Backend WAR File

Navigate to the backend project directory and run:

```
mvn clean package
```

Maven will compile the project and generate a WAR file in the `target` folder.

Step 4: Deploy the WAR to Tomcat

Copy the generated WAR file into Tomcat's `webapps` directory.

Example:

```
copy target/backend.war TOMCAT_HOME/webapps/backend.war
```

Tomcat will automatically deploy the application when it starts.

Step 5: Start Tomcat

Restart the Tomcat server.

Windows:

```
startup.bat
```

Mac / Linux:

```
./startup.sh
```

Once Tomcat starts, the backend application will be available locally.