**Technical Report: AI-Language-Tutor Backend Development**

**1. Project Overview**

**AI-Language-Tutor** is an application designed to assist users in learning languages through AI-powered interactions. The backend is responsible for handling API requests, managing data, and interacting with external services. It is developed in **Kotlin** using **Gradle** as the build automation tool and **Retrofit** for API communication. The project is managed using **Git**, developed in **IntelliJ IDEA**, and integrates with **Android Studio** for frontend development.

**2. Technologies & Tools Used**

**2.1 IntelliJ IDEA**

* **What it is**: IntelliJ IDEA is a powerful IDE for Java, Kotlin, and other languages.
* **Purpose**: Used for backend development, code editing, debugging, and managing Gradle.
* **How it's used**:
  + Writing backend logic in Kotlin.
  + Managing project dependencies using Gradle.
  + Running and testing the application.
  + Debugging and inspecting API responses.

**2.2 Android Studio**

* **What it is**: Android Studio is the official IDE for Android development.
* **Purpose**: It will be used later for designing the **UI using Jetpack Compose** and connecting it to the backend.
* **How it’s used**:
  + Writing frontend UI.
  + Testing the Android app on an emulator.
  + Managing Gradle dependencies for mobile development.

**2.3 Git**

* **What it is**: A version control system to track changes and collaborate on code.
* **Purpose**:
  + Managing code history and branches.
  + Pushing and pulling updates from remote repositories (e.g., GitHub).
* **How it’s used**:
  + git init → Initializes a Git repository.
  + git add . → Adds files to staging.
  + git commit -m "message" → Saves changes.
  + git push origin main → Pushes to a remote repository.

**2.4 Kotlin**

* **What it is**: A modern programming language designed for JVM and Android development.
* **Purpose**: Used for writing backend logic due to its concise syntax and strong type safety.
* **How it’s used in this project**:
  + Writing API request handlers.
  + Processing data from Retrofit responses.
  + Implementing business logic.

**2.5 Gradle**

* **What it is**: A build automation tool that compiles code, manages dependencies, and executes tasks.
* **Purpose**:
  + Handles dependencies like Retrofit and Gson.
  + Automates the build process.
* **How it’s used**:
  + gradle init → Initializes the project.
  + ./gradlew build → Compiles the project.
  + settings.gradle.kts → Configures project settings.
  + build.gradle.kts → Manages dependencies.

**2.6 Retrofit**

* **What it is**: A library for making API calls in Kotlin.
* **Purpose**:
  + Fetching data from external APIs.
  + Sending user requests to a backend server.
* **How it’s used**:
  + Define a **data model** for the API response.
  + Create a **Retrofit interface** for making requests.
  + Call the API asynchronously using enqueue().

**3. Project Structure & Explanation of Every File**

**Project Root Files**

| **File/Folder** | **Description** |
| --- | --- |
| .gradle/ | Stores Gradle’s cache files and build information. Should not be modified. |
| .idea/ | IntelliJ-specific settings and project configurations. Can be ignored. |
| .git/ | Stores version history if the project is under Git. Hidden by default. |
| .gitignore | Specifies files and folders that Git should ignore (e.g., .gradle/, .idea/). |
| build.gradle.kts | The main Gradle configuration file. Defines dependencies and build settings. |
| gradle.properties | Configuration properties for Gradle, like JVM options. |
| gradlew | The Gradle wrapper script (used to execute Gradle commands). |
| gradlew.bat | Windows-specific Gradle wrapper. |
| settings.gradle.kts | Defines the root project name and includes submodules (e.g., app). |

**App Module (app/)**

| **File/Folder** | **Description** |
| --- | --- |
| src/ | Contains the actual source code and tests. |
| src/main/kotlin | Stores Kotlin source files. |
| src/main/resources | Stores additional resources (e.g., JSON files). |
| src/test/kotlin | Stores unit test files. |

**4. How These Components Work Together**

1. **Gradle Manages Dependencies**
   * You define libraries (e.g., Retrofit, Gson) in build.gradle.kts.
   * Gradle downloads and includes them in your project.
2. **Kotlin Backend Code**
   * You write API handlers and data processing logic in src/main/kotlin.
3. **Retrofit Handles API Requests**
   * You create an **interface** with Retrofit annotations (@GET, @POST).
   * Retrofit converts API responses into Kotlin objects.
4. **Testing with JUnit**
   * You write tests in src/test/kotlin to verify code logic.

**5. How to Set Up & Build the Project**

1. **Clone the Project (If Using GitHub)**

git clone https://github.com/yourusername/AI-Language-Tutor.git

cd AI-Language-Tutor

1. **Run Gradle Build**

./gradlew build

1. **Sync Project in IntelliJ**
   * Open the project.
   * Go to File > Sync Gradle.
   * Ensure all dependencies are installed.

**Step 6: Complete Setup and Configuration of the Project (Start to Finish)**

1. **Installing IntelliJ IDEA**

If IntelliJ IDEA is not already installed, it can be downloaded from the official JetBrains website. After installation, open IntelliJ IDEA and check whether the Kotlin plugin is installed. This can be done by going to File → Settings → Plugins, then searching for "Kotlin" and installing it if necessary.

1. **Installing Gradle**

Gradle does not need to be installed separately if the project is using the Gradle Wrapper, which allows Gradle to be run without a global installation. If a global installation is required, it can be downloaded from the official Gradle website, and after installation, the following command can be used to check if Gradle is properly installed:

gradle -v

1. **Creating a New Project in IntelliJ IDEA**

To create a new project, follow these steps:

1. Open IntelliJ IDEA.
2. Click on New Project.
3. Choose Gradle as the build system.
4. Select Kotlin as the programming language.
5. Ensure that the Gradle DSL is set to Kotlin (.kts).
6. Set the Java Virtual Machine (JVM) version to 17.
7. If version control is needed, check the box for initializing a Git repository.
8. Click Create to generate the project.
9. **Configuring Gradle Settings**

Inside the settings.gradle.kts file, ensure that the project name is correctly defined. The file should contain:

rootProject.name = "AI-Language-Tutor"

1. **Configuring the Gradle Build Script**

The build.gradle.kts file should include all necessary plugins, repositories, and dependencies.

1. S**etting Up the Source Code Directory**

The src/main/kotlin folder should contain the main application file. Inside this folder, create a file called App.kt.

1. **Running the Initial Build**

To ensure that the project is correctly configured, a Gradle build should be run. This can be done in IntelliJ IDEA by:

1. Opening the Gradle tool window.
2. Clicking on the "Refresh" button to sync the project.
3. Running the "Build" task to compile and verify dependencies.

Alternatively, the following command can be run from the terminal inside the project root directory:

./gradlew build

1. **Initializing Git and Connecting to a Remote Repository**

If Git has not been initialized yet, navigate to the project directory in the terminal and run:

git init

git add .

git commit -m "Initial commit"

To connect to a remote repository, such as GitHub, use the following command, replacing <repository-url> with the actual repository link:

git remote add origin <repository-url>

git push -u origin main

1. **Running the Application**

Once everything is set up, the application can be executed using one of the following methods:

1. Click the green Run button in IntelliJ IDEA.
2. Use the Gradle tool window to run the application task.
3. Run the following command in the terminal:

./gradlew run

**7. Summary**

This project is structured to allow efficient backend development in **Kotlin** using **Gradle**, **Retrofit**, and other modern tools. The backend handles API requests, processes data, and communicates with a frontend (Android app). The project is managed using **Git** and developed in **IntelliJ IDEA**.