**Software Design and Engineering**

**Lab Document**

|  |  |
| --- | --- |
| **High Level Purpose Statement:** | I want to design and implement an experiment that tests how to use Gradle effectively for building, managing dependencies, and packaging a Java project into a deployable artifact. Additionally, I want to explore Gradle’s plugin system for automating tasks such as testing, code analysis, and documentation generation. |
| **Experimental Design:** | Select or create a Java project that includes at least two external libraries as dependencies. The project will incorporate functionality such as file manipulation and logging. Install Gradle if it is not already installed on my system. Initialize the project as a Gradle project using IntelliJ’s built-in tools for converting a standard Java project into a Gradle project. Configure the build.gradle file to include dependencies, such as log4j for logging and Apache Commons IO for file manipulation. Write unit tests for the project using JUnit and configure Gradle’s test task to automate test execution. Use Gradle commands such as gradle build, gradle test, and gradle assemble to build the project and verify its functionality. |
| **Resources Available:** | Official Maven Documentation. IntelliJ IDEA Documentation. Online tutorials and videos on Maven and its plugins. Maven community forums and Stack Overflow. |
| **Time Estimate:** | I estimate spending around 12 hours on this experiment: 3 hours setting up the Java project and integrating dependencies, 3 hours learning Gradle basics and setting up the build.gradle file, 3 hours experimenting with build automation, test execution, and packaging, and 3 hours exploring and documenting the use of Gradle plugins. |
| **Experiment Notes:** | Record key challenges encountered while setting up Gradle and configuring plugins. Document all commands, build.gradle configurations, and errors encountered, along with their resolutions. |
| **Results:** | Summarize how Gradle simplified (or complicated) the build and deployment process. Identify the most useful Gradle features and plugins based on this experiment. |
| **Consequences for the Future:** | Use Gradle as a standard build and dependency management tool for future Java projects. Incorporate Gradle’s plugin ecosystem to streamline workflows like testing, packaging, and documentation. Share insights with peers or incorporate them into a tutorial for others starting with Gradle. |