**Automating PL SQL deployment and testing**

**Use Case Overview:**

 Automate the lifecycle from understanding a BRD to generating, validating, deploying, and testing PL/SQL code using an LLM agent. This solution leverages a Large Language Model (LLM) to intelligently analyse the user's Business Requirement Document (BRD) and understand the product specifications across all layers of the system architecture. The LLM is designed to interpret existing PL/SQL packages, identify the most appropriate code development practices, and determine the optimal insertion points for implementing new requirements. Also, test the deployed code in the target environment.

**The core responsibilities of this LLM include:**

* **Understanding Requirements:** Parsing the BRD to extract functional needs and translate them into implementation logic.
* **Code Modification:** Locating the relevant procedure or function in existing PL/SQL packages and generating syntactically correct and context-aware code changes.
* **Deployment Automation:** Managing code deployment into the target environment with user-controlled approval.
* **Automated Testing:** Generating and executing appropriate test scenarios to validate the deployed changes within the target environment.

**To streamline this workflow, the LLM solution is organized into three core modules:**

1. **Code Generation & Validation** – Creates or modifies PL/SQL code based on user input and validates its correctness.

2. **Code Deployment** – Automates the deployment of updated packages into version control and database environments with necessary confirmations.

3. **Code Testing** – Runs synthetic or real test cases to ensure the deployed changes function as expected.

**Effort & Time Comparison:** Classical vs AI Agentic SDLC

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Traditional Effort** | **LLM-Based Effort** | **Efforts saved** |
| BRD to impact analysis | 6–8 hours | ~30 minutes | 80–90% |
| Code generation + review | 2–3 days | 1–2 hours | 85–90% |
| Deployment process | 1 day | ~15 minutes | 95% |
| Test planning & execution | 2–3 days | ~1–2 hours | 90% |

Overall reduction: Approx. 80–90% reduction in overall SDLC turnaround time for repetitive development-deploy-test cycles.

**Summary: Why Use AI Agentic SDLC Here?**

* **Speed:** Converts BRD to tested, deployed code in hours vs days.
* **Consistency:** Adheres to coding and architectural standards.
* **Autonomy:** Automates low-level repetitive tasks.
* **Precision:** Identifies exactly where and how to change legacy code.

**Status of implementation:**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Module** | **Status** |
| 1 | [PL/SQL creation and code validation](http://1.pl/SQL) | **Success** |
| 2 | Code deployment into GitHub and Oracle Database | **Failed** |
| 3 | Code validation in oracle database and testing the updated code using synthetic data | **Inprogress** |