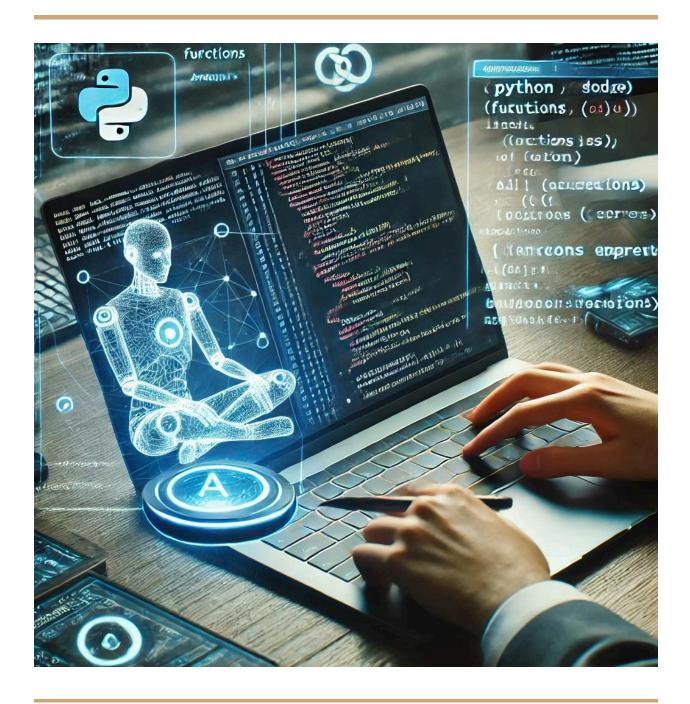
Gen AI for Software Development:

REPORT TITLE

AI-Assisted Python Development for Software

Engineering



1. Introduction:

The AI-Assisted Python Development for Software Engineering project explores the integration of Generative AI (Google Bard) in the software development process. This project focuses on leveraging AI to generate, debug, and optimize Python code, enhancing efficiency and problem-solving capabilities.

With the rise of **Generative AI in software development**, tools like **Google Bard** are transforming how developers write, debug, and optimize code. This project explores **AI-assisted Python development**, demonstrating how AI can help in **code generation**, **debugging**, **and logic structuring**.

The project consists of two key implementations:

- 1. **A banking system withdrawal feature** Handling user input, applying conditions, and ensuring validation.
- 2. **A guessing game** Implementing loops, conditional logic, and randomization to create an interactive program.

By utilizing **Google Bard** as a coding assistant, this project enhances programming efficiency, improves problem-solving, and demonstrates best practices in **AI-assisted software engineering**.

2.Objective:

- 1. Leverage **AI tools** (Google Bard) to assist in Python development.
- 2. Improve code efficiency and debugging using AI-generated suggestions.
- 3. Implement **real-world programming concepts** such as loops, conditionals, and user input handling.
- 4. Explore AI's role in optimizing development workflows.
- 5. Develop a **banking withdrawal feature** and an **interactive guessing game** using Python.

3. Specifications:

3.1 Tools & Technologies

1. Programming Language: Python

2. Development Environment: VS Code

3. AI Tool for Assistance: Google Bard

4. Python Modules Used: random, input handling

4. Project Workflow & Implementation:

4.1 Setting Up the Development Environment

- Installed VS Code and configured it for Python development.
- Integrated Google Bard for AI-assisted coding and debugging.

4.2 Defining Variables & Handling User Input

- Created variables for a banking system withdrawal feature.
- Implemented input() function to take user input and validate responses.

4.3 Implementing Control Structures

- Loops (for, while): Used for handling user interactions.
- Conditional Statements (if-elif-else): Applied decision-making logic for banking transactions and guessing game rules.

4.4 Implementing Randomization

• Used the random module to select a random word for the guessing game.

4.5 Developing the Game Logic

- Designed if-elif-else conditions to handle different game scenarios.
- Implemented loops to continue gameplay until a correct guess is made.

4.6 Creating the Banking Withdrawal Feature

- Focused on validating withdrawal amounts.
- Ensured the system only allows withdrawals if the account balance is sufficient.
- Displayed error messages when the withdrawal is not allowed.

4.7 Debugging & Optimization Using AI

- Used Google Bard to identify and fix syntax/logical errors.
- Optimized code structure for readability and efficiency.

5. Challenges & Solutions:

Challenges	Solutions
Generating accurate AI responses	Improved prompt engineering with better context & examples
Debugging complex logic	Used AI-assisted debugging to analyze errors
Structuring the game loop effectively	Structuring the game loop effectively
Handling incorrect user inputs	Implemented error handling using try-except and validation conditions

6. Key Learnings:

- AI-Driven Development: Learned how to effectively use Google Bard for coding assistance. Python Fundamentals: Strengthened knowledge of variables, loops, conditionals, and functions.
- Debugging & Optimization: Improved code quality using AI-powered suggestions.

- Modular Programming: Structured the project into smaller, reusable components.
- Error Handling: Implemented input validation and exception handling for robustness.

7. Future Enhancements:

- Expand AI Integration: Explore other AI coding tools like GitHub Copilot.
- Database Connectivity: Store user transaction data in a database for banking system improvements.
- GUI Interface: Develop a simple Tkinter-based UI for user interaction.
- Enhance Game Mechanics: Add scoring systems and multiple difficulty levels.

8. Conclusion

This project demonstrated the power of AI-assisted software development by leveraging Google Bard for Python programming, debugging, and logic optimization. The successful implementation of a banking withdrawal system and a guessing game highlights the efficiency, accuracy, and problem-solving capabilities AI brings to software engineering. By integrating AI into coding workflows, developers can

streamline processes, enhance productivity, and write better-structured, optimized code for real-world applications.

9. References

- Python Documentation: https://docs.python.org/
- Google BardAI: https://bard.google.com/
- VS Code Setup Guide: https://code.visualstudio.com/docs