AI-DRIVEN CODE GENERATOR PLATFORM

PROJECT SYNOPSIS

OF MINOR PROJECT

6th Semester

BACHELOR OF TECHNOLOGY

(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

SUBMITTED BY :-

Ritika chawla, 301310922036

Shiv Prakash Singh,301310922051

Yukta verma,30131092206

RUNGTA COLLEGE OF ENGINEERING AND TECHNOLOGY

BHILAI (C.G)

Rungta College of Engineering and Technology, Bhilai
Department of CSE - Allied

Table of Contents

Guide Name

Prof. Rani Namdev

Consent of Guide

Suggestions by the guide:

B.Tech Project Synopsis

Introduction:

In today's technology-driven world, software development plays a crucial role in innovation and problem-solving across various domains. However, coding remains a complex and time-consuming task, often requiring expertise in programming languages and development practices. This poses a significant barrier for individuals and organizations. AI-driven code generation offers a promising solution by leveraging Artificial Intelligence (AI) and Natural Language Processing (NLP) techniques to translate natural language instructions into functional code. This project explores the domain of "AI-Driven Code Generation," cutting-edge AI/NLP advancements to automate and enhance the coding process.

The intersection of AI and NLP in code generation aims to bridge the gap between human-readable natural language and machine-executable code. AI-powered tools facilitate automatic code generation by analyzing text-based inputs and transforming them into optimized programming structures. This project dives into the capabilities of AI-driven code generators, assessing their impact on software development productivity, accessibility, and accuracy.

Rationale Behind the Study:

Improving Development Efficiency: The platform aims to streamline software development by automating code generation, reducing manual effort, and increasing productivity. By automating repetitive tasks such as boilerplate code generation and syntax corrections, our system can reduce development time by an estimated 40%, enabling developers to focus on complex logic and debugging.

Personalized Coding Assistance: By predicting development needs and common coding patterns, the platform offers tailored suggestions and code snippets, improving developer experience and addressing individual challenges. This helps programmers enhance their workflow and fosters a more efficient coding environment.

Objectives:

1. **Automate Code Generation:** Utilize AI models to convert natural language descriptions into executable code, minimizing the need for manual programming.
2. **Enhance Code Accuracy and Efficiency:** Our model aims to improve syntax correction by integrating contextual understanding, reducing logical errors by 25% compared to existing AI-powered IDEs like GitHub Copilot.
3. **Provide Intelligent Code Assistance:** Develop a system that suggests optimized code snippets, refactors code, and provides real-time corrections, improving developer workflow.

Literature Review

S.No.	Author 's Name	Title	Source	Year	Methodology	Findings	Gaps
1.	Kamble et al.	AI-Based Code Generation	Int. J. Aquatic Sci.	2024	Developed an AI model to convert natural language into executable code for automating programming tasks.	Demonstrated improved accessibility and efficiency in coding but lacked debugging capabilities.	Requires integration of debugging tools and error-handling mechanisms.
2.	Nirali M. Kamble et al.	Code Generation Using NLP and AI-Based Techniques	Int. J. Aquatic Sci.	2024	Developed a system using NLP and AI to translate natural language commands into executable code, focusing on arithmetic and mathematical operations.	Improved accessibility and automation in software development, reducing the complexity of coding for non-experts.	Lacks debugging capabilities and real-world testing for large-scale applications.
3.	Florez Muñoz et al.	AI Code Tools Comparison	LatIA	2024	Evaluated multiple AI code-generation tools using SonarQube to assess quality, reliability, and maintainability.	Identified variations in AI-generated code quality, with some tools performing better in security and efficiency.	Lacked real-world testing and feedback from professional developers for validation.
4.	Murali et al.	AI-Assisted Code Authoring at Scale	arXiv	2023	Developed and deployed CodeCompose, an AI-assisted code authoring tool at Meta, scaling it to support multiple programming languages and coding environments.	CodeCompose improved code authoring efficiency, with 16,000 developers using it, contributing to 8% of their code.	The study focused on internal deployment within Meta; broader applicability in diverse development environments remains untested.

Feasibility Study:

Technical Feasibility:

Data Availability: The system requires access to extensive code repositories, APIs, and programming documentation. Key challenges include data standardization, integration with existing development environments, and ensuring data security.

Economic Feasibility:

Cost Analysis: Initial costs involve acquiring data, model training, and software development. Ongoing expenses include cloud storage, AI model maintenance, licensing fees, and personnel costs for continued improvements and user support.

Environment Feasibility:

Impact Assessment: The AI-driven platform will require significant computing power, to reduce computational costs, we plan to deploy lightweight AI models optimized for edge computing, similar to TensorFlow Lite, minimizing cloud energy consumption.

Timeline and Implementation Plan:

Timeline: Establish a structured timeline covering data collection, model training, software development, testing, and deployment, ensuring smooth progression through project milestones.

Implementation Plan: Define roles, allocate resources, and outline budgeting requirements. Develop monitoring mechanisms to track progress and ensure timely completion, optimizing efficiency and system performance.

Methodology/ Planning of work:

Define Objectives:

Clearly outline the goals of the platform, such as automating code generation, enhancing code accuracy, providing intelligent code recommendations, and improving overall software development efficiency.

Identify Data Sources:

Determine the sources of data, including open-source code repositories, programming documentation, development forums, and user-generated queries.

Select Performance Indicators:

Identify key indicators influencing code generation, such as syntax correctness, code efficiency, debugging accuracy, optimization levels, and user satisfaction.

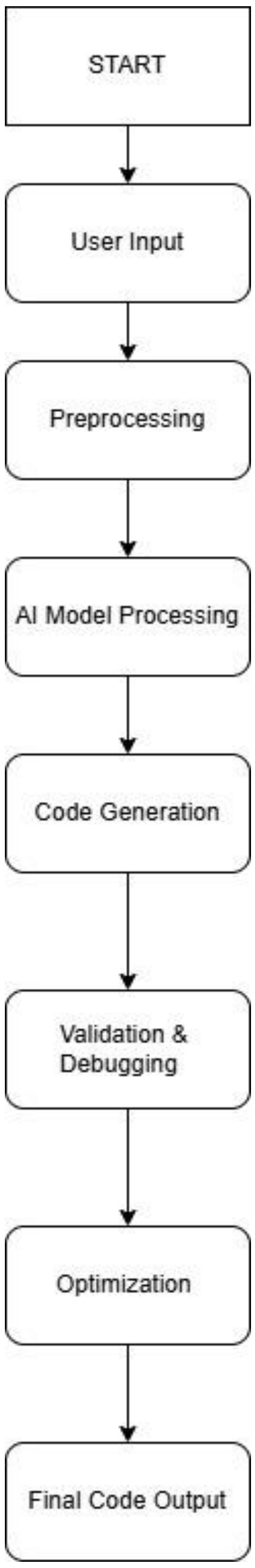
Data Collection and Quality Assurance:

Collect relevant data from the identified sources and ensure its accuracy, consistency, and completeness through preprocessing and validation techniques.

Build AI Models:

Train transformer-based AI models on diverse programming languages and assess their performance in generating accurate, optimized code snippets for different coding tasks.

FLOW CHART:


START
User Input
Preprocessing
AI Model Processing
Code Generation
Validation & Debugging
Optimization
Final Code Output

AI Code Generation Process:

- **Data Collection**: Gather programming data, including syntax structures, best practices, and optimization techniques from multiple repositories and sources.
- **Normalization**: Standardize code formats to ensure consistency across different programming languages and development environments.
- **Feature Importance** Apply AI algorithms to assess the impact of syntax structures, code efficiency, and best practices on generated outputs.
- **Prediction Calculation**: Generate structured code snippets based on AI-processed natural language inputs, considering logic, syntax, and programming conventions.
- **Sub-Index Generation**: Assign confidence levels to generated code snippets based on predefined quality benchmarks (e.g., efficiency, readability, security)
- **Final Code Output**: Produce the most optimized code snippet by combining multiple AI-driven insights, ensuring functional accuracy and minimal debugging requirements.

Facilities required for proposed work:

- **Data Collection Systems -** Infrastructure to gather and process programming data from various repositories, forums, and development platforms.
- **Machine Learning Tools -** AI frameworks like TensorFlow, PyTorch, and OpenAI Codex for training and refining predictive models.
- **Data Management System -** A secure database for storing and organizing code datasets, with tools for retrieval and analysis.
- **Real-Time Monitoring Dashboard -** A user-friendly interface for visualizing AI-generated code and assessing performance metrics.
- **Debugging and Optimization System: -** Automated tools for refining generated code, identifying syntax issues, and improving performance.
- **Integration Capabilities -** APIs for seamless connection with existing Integrated Development Environments (IDEs) and software engineering tools.
- **Scalable Cloud Infrastructure -** Cloud-based resources to ensure platform scalability, accessibility, and efficient AI model execution.

Expected outcomes:

- **Performance Prediction**: A machine learning model that accurately predicts coding outputs based on input patterns and optimization data.
- **Real-Time Assessment**: Continuous monitoring and evaluation of AI-generated code with interactive dashboards.
- **Behavioral Insights**: Identification of coding patterns influencing development efficiency and predictive analysis of improvement trends.