International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

AI-Driven Autonomous Software Engineering System (Jellyfish AI)

Jay Pradip Deshmukh, Siddhi Kadam, Prajakta Badhe, Professor Snehal Awatade, Professor Bhagyashri Ramchandra Gunjal
Department of Artificial Intelligence & Data Science Savitribai Phule Pune University, Pune, India

Abstract- — The use of Generative AI has started changing how software is developed today. Many tools are now available that help developers write code faster or fix errors, but most of them only support specific tasks. They do not handle the complete software development process from In this paper, we propose a system called JellyfishAI ,which aims to automate the full software development lifecycle. The idea is simple — a user can describe their requirements in normal language, and the system will generate a complete, ready-to-use application. It combines different technologies like natural language processing, code generation models, testing, and deployment into one system. Technology is advancing very fast, and modern applications need to be scalable, reliable, and developed quickly. Because of this, traditional development methods are under a lot of pressure. Usually, software development includes many steps like requirement analysis, design, coding, testing, and deployment. These steps often require manual work and skilled developers, which makes the process slow and costly. It can also lead to mistakes and inconsistent results. With the introduction of Generative AI, things have started to improve. These systems can understand user input and generate code, documentation, and even design ideas. They help developers save time by assisting in coding and debugging tasks. However, these tools still have limitations. Most of them only focus on one part of development, like code suggestions or bug fixing, and do not provide a complete solution. Another issue is that developers still need to use multiple tools separately and connect them manually. This makes the workflow complicated and less efficient. Also, the code generated by AI tools often needs to be checked by humans to ensure it is correct, secure, and follows proper standards. Even though CI/CD pipelines help automate deployment, they mostly follow fixed rules and do not have intelligent decision-making capabilities. Similarly, testing tools can find errors but cannot automatically improve the quality of the code. Because of this, there is a need for a system that can intelligently manage all stages of development together. Today's software systems are also becoming more complex, and there is a growing need to build applications quickly. Startups and companies want to turn their ideas into working products as fast as possible. However, depending on skilled developers and manual work often slows things down. This becomes even more difficult when dealing with large systems that require proper coordination and integration. To solve these problems, we introduce JellyfishAI, which is designed to automate the entire development process. Unlike existing tools, this system brings everything together in one place. It can understand user requirements, generate code, test it, check for errors, and deploy the application automatically. The system is built using multiple layers that include language processing, AI-based code generation, validation, and deployment. This ensures that the generated code is not only functional but also tested and ready to use. The system also improves over time using feedback and learning mechanisms. JellyfishAI also focuses on important factors like security, scalability, and maintainability. It includes features to detect vulnerabilities, manage dependencies, and improve performance, so that the final application meets industry standards. This system can change how software development is done. By automating repetitive tasks, developers can focus more on designing and solving problems instead of doing routine work. It also makes it possible for people without strong technical skills to build applications. In conclusion, AI has the potential to improve software development by making it faster, cheaper, and more efficient. However, to achieve full automation, we need systems that combine all stages of development in one place. JellyfishAI is an attempt to do that by providing a complete, end-to-end solution for building software automatically.

Keywords— Generative AI, Software Engineering, Code Generation, Automation, CI/CD, Machine Learning

I. INTRODUCTION

Software development is not a simple task; it is a complete process that involves multiple stages such as requirement analysis, system design, coding, testing, deployment, and maintenance. Each of these stages plays an important role in building a successful application. In traditional software development, most of these tasks are done manually by developers, which requires strong technical knowledge, experience, and a lot of time.

Because of this, developing software becomes a slow and complex process, especially when the project size increases. Developers need to spend a lot of time understanding requirements, writing code from scratch, fixing errors, and testing the application again and again. This not only increases development time but also increases the chances of human errors. Sometimes, due to tight deadlines or lack of proper coordination, the quality of the software also gets affected.

Another challenge in traditional development is that different stages of development are handled using different tools. For example, one tool is used for coding, another for testing, and another for deployment. Developers need to switch between these tools and manually connect them, which makes the workflow complicated and less efficient. This lack of proper integration creates delays and confusion during development.
In recent years, Generative Artificial Intelligence has started changing the way software is developed. Many AI-based tools are now available that help developers in writing code, suggesting improvements, and fixing bugs. These tools are useful because they reduce repetitive work and help developers complete tasks faster. For example, instead of writing long code manually, developers can generate small parts of code using AI tools.

However, even though these tools are helpful, they are still limited in many ways. Most of them can only generate small code snippets or help in debugging. They cannot build a complete application on their own. Developers still need to guide the process, combine different tools, and manage the full development lifecycle manually.

Studies show that these AI tools can reduce development effort by around 15–20%, but they are not capable of handling everything automatically. Human supervision is still required to check whether the generated code is correct, secure, and working properly. Also, these tools do not fully understand the complete requirements in all cases, which can lead to incorrect or incomplete outputs.

Another important issue is that there is no proper system that connects all stages of the Software Development Life Cycle (SDLC). Requirement analysis, coding, testing, and deployment are still treated as separate tasks. Because of this, developers need to spend extra time managing these stages instead of focusing on actual problem-solving and innovation.
Also, modern applications are becoming more complex and require faster development. Startups and companies want to build and launch products quickly to stay competitive. But depending completely on manual work and skilled developers often becomes a bottleneck.

This problem becomes even bigger in large-scale systems where coordination between different teams is required.

To overcome all these problems, there is a need for a system that can automate the entire development process in a smooth and efficient way. Such a system should be able to understand user requirements, generate code, test it, fix errors, and deploy the application without requiring too much manual effort.

In this paper, we introduce JellyfishAI ,which is designed to solve these challenges. The main idea of this system is to allow users to describe their requirements in simple language, and the system will automatically convert those requirements into a complete software application. It combines different technologies like natural language processing, code generation, testing, and deployment into a single platform.

JellyfishAI reduces the need for manual work and helps in saving time. It also improves the quality of the software by automatically checking for errors and ensuring that the application works properly before deployment. The system is designed in such a way that it can be used not only by experienced developers but also by beginners who have limited technical knowledge.

The main goal of this system is to make software development faster, easier, and more efficient. By automating repetitive and time-consuming tasks, developers can focus more on designing better solutions and solving real-world problems. In the future, such systems can play an important role in changing how software is built and used.

Objectives

- To design a system that can understand user requirements written in simple, everyday language and convert them into a structured format for development.
- To reduce the dependency on manual coding by automatically generating code for different parts of an application.
- To combine all stages of software development such as requirement analysis, coding, testing, and deployment into one single platform.
- To minimize human errors by including automatic testing and debugging in the development process.
- To improve the overall speed of software development so that applications can be built in less time.
- To create a system that not only generates code but also checks its quality, performance, and security.

- To make software development easier for beginners or non-technical users who may not have strong programming knowledge.
- To provide a smooth workflow where developers do not need to switch between multiple tools for different tasks.
- To build a system that can adapt and improve over time based on feedback and usage.
- To support the development of scalable and reliable applications that can be used in real-world scenarios.

III. LITERATURE REVIEW

1. Generative AI in Software Engineering
In the last few years, Generative AI has become a useful support tool in software development. It can assist in writing code, creating basic documentation, and suggestingsimple system structures from user input. Most existing work in this area mainly focuses on helping developers generate code faster.

However, these solutions are not complete systems. They usually handle only specific tasks and are not widely used in real-world development environments. Also, different studies use different ways to measure performance, which makes comparison difficult. Because of this, there is still a gap between what is developed in research and what is actually used in practice.

2. AI in Requirements Engineering
Understanding user requirements is a critical step in building any software system. AI is now being used to simplify this process by analyzing user input and converting it into structured information.

Some key observations are:
Use of language-based models
These models help in understanding user input even when it is written in simple or non-technical language.

Lack of clarity in output
Many systems do not clearly explain how they arrive at a particular result.

Inconsistent behavior
The same input may produce different outputs at different times.

Limited control for users
Developers may find it difficult to guide or adjust the system properly.

These issues show that while AI is useful in requirement analysis, it still needs improvement in terms of consistency and transparency. Important aspects like data security and ethical concerns are also not fully handled.

3. Machine Learning in Software Development
Machine Learning techniques are used in different stages of software development to improve efficiency and quality.
Some common uses include:

- Detecting bugs and predicting errors
- Evaluating code quality
- Supporting software maintenance
- Generating basic documentation

Although these applications are helpful, they usually exist as separate tools. There is no single system that brings all these features together into one smooth workflow.

4. AI-Assisted Development and Productivity
AI tools have improved developer productivity, especially in coding and testing tasks. Developers can complete repetitive work faster with the help of these tools.

- Coding, testing, and documentation benefit the most
- Early stages like planning and requirement analysis are still less supported

This shows that current tools focus on individual tasks rather than the entire development process.

5. Explainability and Trust in AI Systems
A major concern with AI systems is that their decision-making process is not always clear. Users often cannot understand how the system produces its output.

- Developers prefer systems that provide clear explanations
- This is important in applications where accuracy and security are critical
- There is a need for systems that can explain their outputs in a simple way

Because of this limitation, developers still need to manually verify the results generated by AI.

6. AI in Learning and Developer Support
AI tools are also widely used by students and developers for learning and assistance. Advantages include:

- Faster understanding of concepts
- Immediate help while coding
- Learning through examples However, there are some drawbacks:
- Over-reliance on AI tools

- Reduced problem-solving ability
- Weak understanding of basic concepts

This shows that AI is helpful, but it should be used carefully.

7. Challenges in Existing Systems
From the study of existing systems, some common challenges are identified:

Disconnected tools
Different stages of development use separate tools that are not properly linked.

Incomplete Automation
Most systems handle only specific parts of the development process.

Errors in generated code
The output may contain bugs or security issues.

Security and ethical concerns
Issues like data privacy and bias are not fully addressed.

Limited practical usage
Many systems are still not widely used in real-world applications.

Research Gap
Based on the above discussion, it is clear that:
- Existing solutions focus only on parts of the development process
- There is no fully integrated system that covers all stages
- Important factors like security, reliability, and clarity need more attention

Motivation for Proposed Work
To overcome these challenges, there is a need for a system that can manage the entire software development process in one place.

Such a system should:
- Combine all stages of development
- Reduce manual effort
- Improve code quality and security
- Provide clear and understandable outputs

The proposed JellyfishAI system is designed to meet these requirements by offering a unified and easy-to-use platform for software development.

III. PROBLEM STATEMENT

In modern software development, even though many advanced tools are available, several challenges still exist. The overall development process is not fully streamlined, and developers often face difficulties in managing different stages efficiently.

One of the major issues is that the workflow is fragmented. Developers need to use different tools for coding, testing, and deployment, which are not properly connected. This makes the process complex and time-consuming. Another important problem is that debugging and testing are still mostly done manually, requiring repeated effort to identify and fix errors.

There is also a lack of proper integration between different stages of the Software

Development Life Cycle (SDLC). Outputs from one stage are not smoothly transferred to the next, which creates delays and reduces efficiency. Additionally, with the use of AI tools for code generation, new problems have arisen. The generated code may contain logical errors, security issues, or may not fully match the user requirements, requiring manual verification.

Because of these limitations, current systems are not able to provide a complete and efficient solution for software development. Developers still need to depend heavily on manual work and multiple tools, which increases effort, time, and chances of errors.

Key Problems Identified
Fragmented Development Workflow
Different tools are used for different stages, and they are not properly connected.

Manual Debugging and Testing
Developers have to manually find and fix errors, which takes time and effort.

Lack of Integration
Poor connection between development stages leads to delays and inefficiency.

Security Issues in AI-Generated Code
Automatically generated code may contain bugs or vulnerabilities.

High Dependency on Human Effort
Continuous human supervision is required throughout the process.

Increased Development Time and Cost
Due to manual work and inefficient workflows.

Conclusion of Problem
Due to these issues, there is a strong need for a system that can automate and integrate all stages of software development. Such a system should reduce manual effort, improve efficiency, and ensure better quality and security in the final application.

IV. PROPOSED SYSTEM

The proposed system, JellyfishAI, is designed as a complete and integrated solution for software development. Instead of using separate tools for different tasks, this system combines all stages of development into one platform. The main idea is to make the process easier, faster, and more efficient by reducing manual work.

The system works in multiple steps, where each step is connected to the next, forming a smooth workflow from requirement input to final deployment.

Main Functions of JellyfishAI
Natural Language Requirement Analysis
In this step, the system takes input from the user in simple language. The user does not need to write technical specifications; they can just describe what they want to build.

The system then:
- Understands the meaning of the input
- Identifies key features and functionalities
- Converts it into a structured format

This makes the system easy to use even for beginners.

AI-Based Code Generation
After understanding the requirements, the system automatically generates code. It can:
- Create frontend and backend code
- Generate APIs and database structure
- Follow proper coding practices

This reduces the need for manual coding and saves a lot of time.

Automated Debugging and Testing
Once the code is generated, the system checks whether it is working correctly. In this step:
- Errors and bugs are detected automatically
- The system tries to fix common issues
- Different test cases are applied

This improves the reliability of the application and reduces human effort.

Security Validation
Security is an important part of software development. The system checks the generated code for possible vulnerabilities. It ensures:
- Safe coding practices
- Detection of common security risks
- Better protection of the application

This helps in building secure and trustworthy software.
Continuous Deployment
After testing and validation, the system automatically deploys the application. This includes:
- Preparing the application for hosting
- Configuring necessary settings
- Making the application live

This removes the need for manual deployment steps and makes the process faster.

V. SYSTEM ARCHITECTURE

The system architecture consists of the following layers:
- User Interface Layer
- NLP Processing Layer
- Code Generation Engine
- Validation Engine
- Deployment Engine

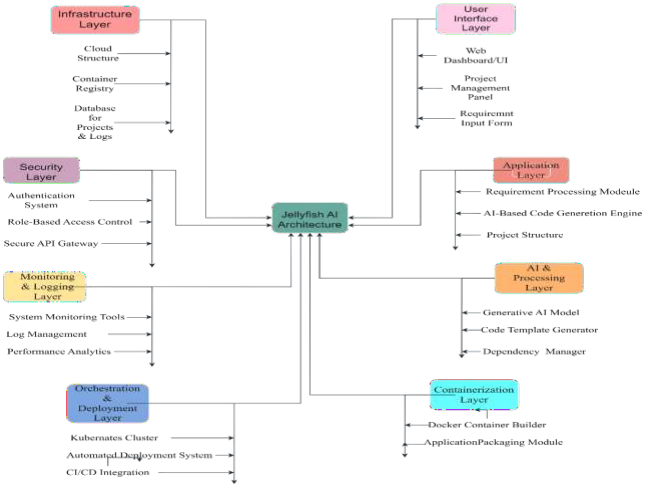

Fig. 1. System Architecture

VI. METHODOLOGY

The development of JellyfishAI is done step by step in a structured way so that the system works properly and efficiently. Each phase has a specific purpose, and all phases are connected with each other.

1. Requirement Analysis
This is the first and most important step. In this phase, we understand what the user actually wants to build. The user gives input in simple language, like describing an app idea or functionality.

The system analyzes this input using language processing techniques and tries to understand the meaning, intent, and features required. It identifies things like:

- What type of application is needed
- What features should be included
- What kind of output is expected

This step is important because if the requirements are not clear, the final output will also not be correct.

2. System Design
After understanding the requirements, the next step is to plan how the system will be built. In this phase, the overall structure of the application is decided.

This includes:

- Deciding the architecture (frontend, backend, database)
- Planning how different parts of the system will interact
- Selecting suitable technologies

Basically, this phase acts like a blueprint. It helps in organizing the development process and avoids confusion later.

3. AI Model Integration
In this phase, AI models are connected to the system so that it can generate code and perform intelligent tasks.

The system uses models that can:

- Understand user input
- Generate code based on requirements
- Suggest improvements

These models are integrated in such a way that they can take input from the previous phase and produce meaningful output.

This is the core part of JellyfishAI because it handles automation.

4. Development
In this phase, the actual code is generated and the application starts taking shape. The system:

- Converts requirements into code
- Creates frontend and backend components
- Connects with the database

Instead of writing everything manually, most of the code is generated automatically. This saves time and reduces effort.

5. Testing and Debugging
Once the code is generated, it is important to check whether everything is working correctly or not.
In this phase:

- The system checks for errors in the code
- Tests different functionalities
- Fixes bugs automatically wherever possible

Both static checking (code analysis) and dynamic testing (running the code) are done. This helps in improving code quality and reducing mistakes.

6. Deployment
This is the final phase where the application is made ready for use. In this phase:

- The application is deployed on a server or platform
- Necessary configurations are done
- The system ensures that the application runs smoothly

After deployment, the user can access and use the application. This step completes the full development cycle.

Algorithms and Models
Code Generation Model
In this system, transformer-based models are used to generate code from user input. These models are capable of understanding natural language and converting it into meaningful code.

When a user provides a requirement in simple language, the model analyzes the input and generates the required code structure. It can create different parts of the application such as frontend, backend, and database-related code. This helps in reducing manual coding effort and speeds up the development process.

Error Detection
After generating the code, it is important to check whether the code is correct and working properly. For this, two types of analysis are used:

Static Code Analysis
In this method, the code is checked without actually running it. The system scans the code to find syntax errors, incorrect logic, or bad coding practices. This helps in identifying issues at an early stage.

Dynamic Execution Analysis
In this method, the code is executed and tested in a real environment. The system checks how the application behaves during runtime and identifies errors that occur while execution. This ensures that the application works correctly in practical scenarios.

Optimization
Once the code is generated and tested, it is further improved using optimization techniques. Reinforcement learning is used to enhance the performance of the code.

The system learns from previous outputs and feedback to make better decisions in future code generation. This helps in improving efficiency, reducing unnecessary complexity, and generating better-quality code over time.

VII. SYSTEM REQUIREMENTS

Hardware Requirements
To run the JellyfishAI system smoothly, certain basic hardware is required:

Intel Core i5 processor or higher
A good processor is needed to handle processing tasks efficiently.

Minimum 8GB RAM
Sufficient memory is required to run multiple processes and handle AI models.

256GB SSD storage
Faster storage helps in quick data access and improves overall system performance.

Software Requirements
The system uses a combination of different software technologies:

Node.js
Used for backend development and handling server-side operations.

Python
Used for implementing AI models and processing data.

React.js
Used for building the user interface of the application.

Docker
Helps in containerization, making deployment easier and more consistent.

MongoDB / PostgreSQL
Used as databases to store and manage application data.

VIII. RESULTS AND ANALYSIS

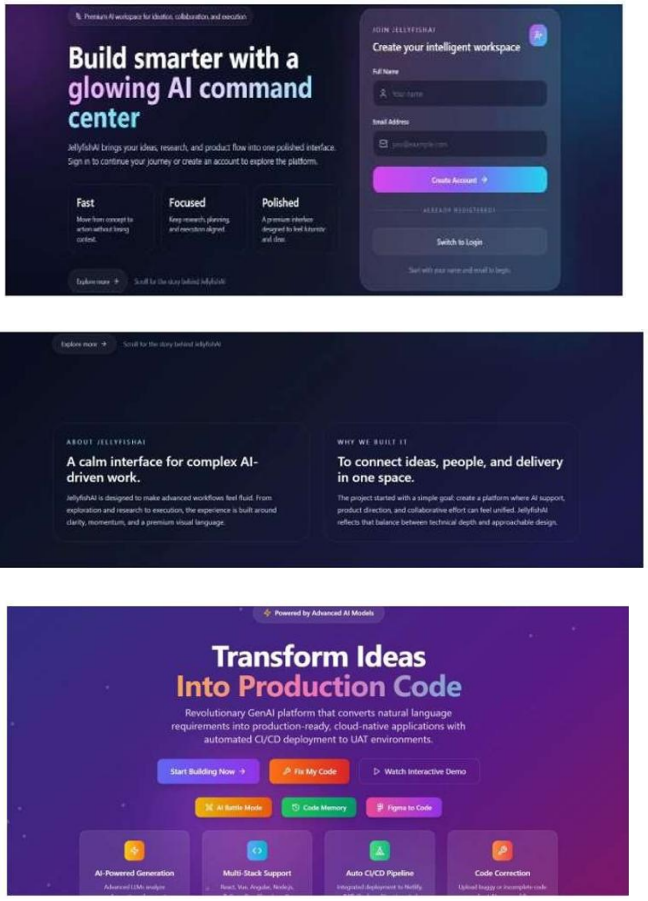

Fig 2: Snaphot of homapge and Inteface displaying AI-powered features

Table 2: Performance Comparison

Metric	Existing System	Proposed System
Development Time	High	Reduced
Error Rate	High	Low
Deployment Speed	Manual	Automated

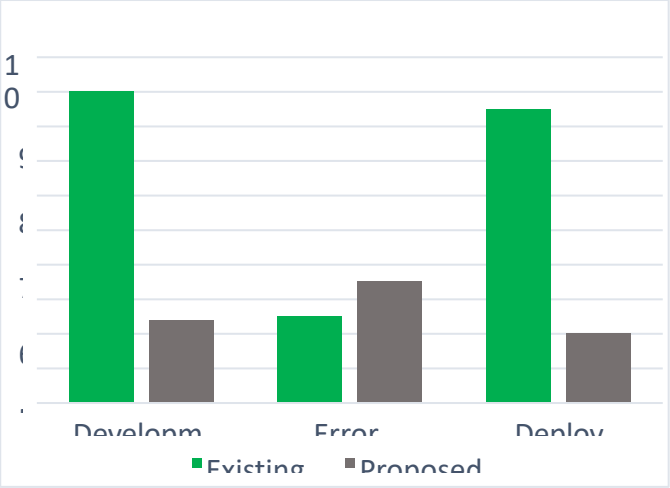

Fig. 2. Performance Graph

Applications

- Automated Software Development
- Educational Platforms
- Startup MVP Development
- Enterprise System Development
- AI-based Coding Assistants

IX. CONCLUSION

The proposed JellyfishAI system helps in improving the software development process by bringing all stages together into one platform. It automates tasks like understanding requirements, generating code, testing, and deployment, which reduces the need for manual work.

Because of this, development becomes faster and more efficient. The system also helps in improving the quality of the code by automatically checking for errors and ensuring better performance. Overall, JellyfishAI makes software development easier and more organized.

This system shows how automation can simplify complex development processes and can be a step towards building fully automated software systems in the future.

Future Work

The current system can be further improved in many ways to make it more powerful and useful.

Multi-agent AI collaboration systems

In the future, multiple AI agents can work together, where each agent handles a specific task like coding, testing, or deployment. This can make the system faster and more efficient.

Self-learning optimization mechanisms

The system can be enhanced to learn from its past outputs and user feedback. Over time, it can improve its performance, generate better code, and reduce errors automatically.

Integration with emerging technologies such as IoT and Blockchain

JellyfishAI can be extended to support modern technologies. For example, it can help in building IoT-based applications or secure systems using blockchain, making it more versatile and useful in real-world scenarios.

REFERENCES

1. R. K. Mallidi et al., “Story Point Estimate Model: Project Development using Generative AI Tools,” International Journal of Computer Applications, vol. 186, no. 54, 2024.
2. OpenAI, “GPT-4 Technical Report,” 2023.
3. S. Gupta et al., “AI-Powered Code Generation and Automation,” International Journal of AI, 2024.
4. IEEE Software Engineering Conference Papers, 2023.

.