A Minor Project Final Report On

**Bug Finder System**

Submitted in Partial Fulfilment of the Requirements for the degree of

Bachelor of Engineering in Information Technology

Under Pokhara University

Submitted by:

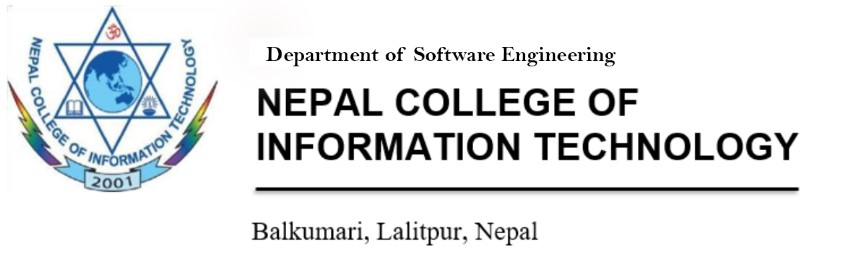
**APSARA ARYAL, 201743**

**­­­BIKASH BANJADE, 201705**

**DHIRAJ YADAV, 201708**

**SANDESH ADHIKARI, 201730**

Date:

21 August 2024

## **Abstract**

The “Bugs Finder system” is a web application designed to assist users in identifying and resolving bugs in their code. With the increasing complexity of software development, debugging remains a significant challenge for developers. This platform aims to empower users by providing a user-friendly interface to upload their code, identify errors, and receive guidance on debugging techniques. The ultimate goal is to enhance the efficiency and effectiveness of code debugging processes, benefiting both individual developers and the software development community as a whole.

For the project’s development, we are using CSS and mainly React.JS for the frontend and Backend DJANGO framework for handling user requests, managing data flow, and integrating with the debugging model. Database MYSQL for storing user code submissions and debugging reports.

**Keywords**

*Debugging, React.JS, CSS, MYSQL, Code Analysis, Web application*

## **Acknowledgement**

The Completion of our project would not have been possible without the support and guidance of our colleagues and teachers. We would especially like to express our utmost gratitude to our project guide, **Mr. Roshan Chitrakar,** for giving us this opportunity. His proper guidelines have been extremely helpful throughout the making of this project, and we are wholeheartedly thankful to him.

We also express our gratitude to our project supervisor, **Mr. Subash Manandhar,** for lending us his assistance throughout the project. His monitoring, support, and suggestions were invaluable during this short period of time.

Finally, we extend our appreciation to all those who have provided us with their support and encouragement throughout the development of this project.

**Table of Contents**

[**Abstract I**](#_Toc166509300)

[**Acknowledgement II**](#_Toc166509300)

[**1 Introduction 1**](#_Toc166509301)

[**2 Problem Statement 2**](#_Toc166509302)

[**3 Project Objectives 3**](#_Toc166509303)

[**4 Significance of the study 4**](#_Toc166509304)

[**5 Scope and Limitations 5**](#_Toc166509305)

[5.1 Scope 5](#_Toc166509306)

[5.2 Limitations 5](#_Toc166509307)

[**6 Literature Review 6**](#_Toc166509308)

[**7 Proposed Methodology**](#_Toc166509310) **8**

[7.1 Software Process Model](#_Toc166509311) 8

[**8 Tools and Technology**](#_Toc166509312) **11**

[8.1 Technologys](#_Toc166509311) 11

[8.2 Tools](#_Toc166509311) 11

[**9 List of figures 1**](#_Toc166509312)**2**

[9.1 Use Case Diagram 1](#_Toc166509313)2

[9.2 Sequence Diagram 1](#_Toc166509314)3

[9.3 Activity Diagram 14](#_Toc166509315)

[9.4 ER-Diagram 15](#_Toc166509315)

[9.5 Class Diagram 16](#_Toc166509315)

[9.6 Development Model](#_Toc166509315) 17

[**10 Project Task and Time Schedule 1**](#_Toc166509312)**8**

[**11 Testing 1**](#_Toc166509312)**9**

[**12 Result and conclusion**](#_Toc166509312) **20**

[**References**](#_Toc166509316) **21**

**Appendix****22**

# **Introduction**

In the rapidly evolving landscape of software development, managing code quality and debugging efficiently are critical challenges that can significantly handle project timelines and overall software performance. Tradition debugging methods can be time-consuming and often depend heavily on the developer’s expertise and experience, which can lead to inconsistent results and overlooked errors. To address these challenges, we introduce the concept of the “Bugs Finder” a sophisticated online platform that leverages the power of Large Language Models (LLMs) for software debugging.

Our project, “Bugs Finder”, is designed to be an accessible tool on the World Wide Web, allowing developers to upload their code and receive automated insights into potential bugs and their fixes. This system is intended for a diverse range of users, From novice programmers who require guidance to seasoned developers who seek to optimize their debugging process. By integrating generative AI models, specifically LLMs, our system analyses the uploaded code, identifies errors, and suggests actionable solution efficiently.

The primary issues that our project aims to solve include the reduction of debugging time, providing a learning platform for less experienced developers to improve their coding skills, and offering a robust tool for complex software projects that require rigorous testing and maintenance. With the Bugs Finder, users can track their code’s performance, view detailed reports of issues, and receive notifications about potential fixes, making the entire process more manageable and efficient.

This project stands to revolutionize the way developers interact with their code, fostering a culture of precision and efficiency in software development that aligns with modern technological advancements and the demands of current software engineering landscapes.

# **Problem Statement**

The modern software development, debugging is a critical process that can be time-consuming and often complex, particularly for newer developers or these working on sophisticated systems. The “Bugs finder” aims to streamline this process by utilizing a web-based platform where developers can upload their code to detect and rectify errors with the help of LLMS model. Despite the advances in development tools several key challenges continue in the debugging domain:

1. Limited automation in bug detection to a time-consuming debugging process.
2. Enhanced Error Resolution.
3. Accessible Debugging.

# **Project Objectives**

The primary goal of the “Bugs finder” project is to develop a comprehensive and user-friendly platform that assists developers in identifying, understanding and resolving coding errors effectively through automated and intelligent diagnostics. The specific objectives are as follows:

1. To provide users with a user-friendly interface to upload their code, run it, and view the output.
2. To include a comprehensive cheat sheet for users to learn Python, C++, C, JavaScript, and Java.
3. To present users with actionable suggestions for fixing identified bugs, facilitating debugging processes.

# **Significance of the study**

The “Bugs Finder” represents a pivotal advancement in the field of software development, addressing critical challenges associated with the debugging process. As software systems become increasingly complex and integral to all aspects of modern life, the efficiency of developing, testing, and maintain software solutions becomes crucial.

The Bugs Finder significantly improves the quality and reliability of software applications, By enabling developers to swiftly identify and correct bugs, the system ensures that software products are more stable and performant upon release. This reduces the occurrence of costly failures and enhances user satisfaction.

Particularly beneficial for new developers, the system provides an education platform that highlights common and uncommon programming errors, offering solutions and best practices.

By reducing the time spent on debugging, the Debug Finder system contributes to more sustainable software development practices. It allows companies to allocate their resources more efficiently and reduce the environmental impact associated with extended development cycles, such as energy consumption and electronic waste.

# **Scope and Limitations**

The “Bugs finder” project is designed to revolutionize how developers handle the debugging phase of software development by automating the identification and resolution of code errors. The project leverages the power of Large Language Models (LLMs) to analyse code submitted by users, pinpoint issues, and provide actionable solutions.

## **Scope**

* Real-time Feedback and Learning.
* User-friendly interface supporting multiple programming languages.
* Automated Bug Detection and Resolution.

## **Limitations**

* Difficult in handle complex code.
* Accuracy dependent on the capabilities of the Google Gemini API.
* Limited language support and potential performance constraints.

# **Literature Review**

We have studied and researched local as well as global places for attending meetings problem and solutions.

## **Challenges in Current Bug Finding System**

**Manual and Inefficient Debugging**

Traditional debugging methods often involve manual code inspection and error finding, which are time consuming and prone to human error. Zhao et al.(2022) discuss how these manual approaches are inadequate for modern, complex software environments, emphasizing the need for more automated solution to enhance debugging efficiency.(Zhao et al.,2022)

**Limited Analysis and Reporting Capabilities**

Many existing debugging tools offer limited analytical insights and basic reporting functionalities. Chen et al. (2018) highlight that comprehensive analysis and detailed reports are critical for effective debugging. The “Bugs Finder” system addresses these limitations by integrating the Google Gemini API, which provides advanced analysis and actionable suggestions, thereby improving the debugging process (Chen et al, 2018).

**Scalability Issues**

Managing and tracking bugs becomes increasingly complex as software projects grow. Fehlmann (2020) underlines the necessity for system utilizes Django and MySQL to ensure robust and scabble management of code submissions and bug reports, addressing this challenge effectively (Fahlman, 2020).

## **Recent Trend and Advances**

**Integration of Advance APIs**

Recent trends show a shift towards integrating advanced APIs for enhanced debugging capabilities. The Google Gemini API exemplifies this trend by leveraging AI to provide accurate and efficient bug detection (Elmishali et al., 2018).This integration aligns with the “Bugs Finder” system’s objective to deliver intelligent and automated diagnostics.

**Education and Support Features**

There is growing trend of incorporating educational resources within debugging tools. The inclusion of cheat sheets for programming languages like Python, C++, JavaScript, and java, c in the “Bug Finder” system reflects this trend, offering developers quick learning aids and resources to support their debugging efforts (Budhiraja et al., 2018).

**Enhanced Reporting and Documentation**

Detailed reporting and documentation have become crucial components of modern debugging systems. The ability of the “Bug Finder” system generate and save comprehensive reports addresses the need for better debugging and management of debugging activities, as highlighted by Harman(2012)(Harman, 2012).

The ”Bugs Finder” system represents a notable advancement in addressing prevalent debugging challenges. By automating bug detection, enhancing analysis, and integrating educational features, it aligns with recent trends and meets the evolving needs of software developers. This system not only addresses the limitations of existing tools but also contributes to a more efficient and scalable debugging process.

# **Proposed Methodology**

The “Bug Finder” system, we will employ the following methodologies to effectively apply knowledge, skill, techniques across various activities to meet project requirements:

## **Software Development Life Cycle**

The development of the “Bug Finder” system website follows the incremental model of the Software Development Life Cycle (SDLC). This approach supports iterative development, allowing the system to evolve through successive increments, each introducing new features and improvements.

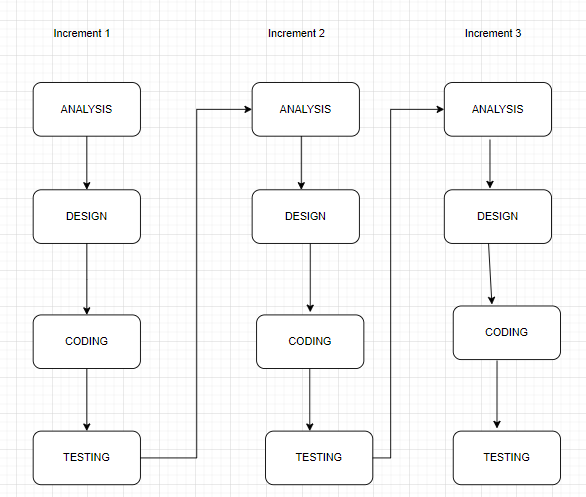


Figure1: Incremental model of software development life cycle

### **Development Model**

The incremental model is characterized by gradual development and continuous enhancement. Each increment introduces additional functionality and improvements, with the system being tested and refined in each iteration. The system is considered complete when all specified requirements are satisfied.

### **Analysis Phase**

In the analysis phase, comprehensive requirements for the “Bugs Finder” system are gathered and documented. This results in a System Requirements Specification (SRS), detailing both functional and non- functional requirements essential for the system’s development.

### **Design Phase**

In the design phase, the SRS is translated into a detailed system design. Key design artifacts include:

* **Entity-Relationship Diagram(ER-Diagram):** Define data entities and their relationships.
* **Use Case Diagram:** Describe the system’s functionalities and user interactions.
* **Class Diagram:** Detail the system’s object-oriented design, including classes and relationships.

### **Coding Phase**

The coding phase involves translating design specifications into a functional website. The development focuses on implementing the following features incrementally:

* **User Signup and Signin:** Create authentication pages for user registration and login.
* **Code upload and Executing:** Enable users to upload code, which is executed using an API, with output displayed.
* **Code Analysis:** Integrate Google Gemini API for analysing uploaded code and providing diagnostics.
* **Cheat Sheets:** Add educational content for programming languages such as Python, C++, JavaScript, and Java.
* **Report Generation and saving:** Implement functionality for generating and saving reports based on code analysis.

### **Testing Phase**

The testing phase involves multiple levels of testing to ensure system quality:

* **Unit Testing:** Testing individual components such as React components and Django views.
* **Integration Testing:** Testing interactions between frontend and backend, ensuring smooth data flow.
* **System Testing:** End-to-end testing of the entire application, covering all use cases.

**Testing Tools:**

* Jest for unit react components.
* PyTest for testing Django views and models
* Selenium for automated integration and system testing.

By following this methodology, the “Bug Finder” website will be developed incrementally, allowing foe continuous improvement and adaptation based on ongoing testing and feedback.

# **Tools and Technology Used**

## **Technologies**

**Frontend**

* ReactJS: Builds dynamic user interfaces.
* CSS: Styles the website.

**Backend**

* Django: Manages server-side logic and data.

**Database**

* MySQL: Store and manages data.

**Debug Engine (Model)**

* Gemini-1.5-flash :

Gemini-1.5-flash is a code-specialized version of Gemini Pro that was created by further training Gemini Pro on its code-specific datasets, sampling more data from that same dataset for longer. Essentially, Gemini-1.5-flash features enhanced coding capabilities. It can generate code, and natural language about code, from both code and natural language prompts.

## **Tools**

**Visual Studio Code:** IDE for frontend and backend development.

**E Draw:** For creating diagrams and design schematics.

**Postman:** For testing APIs and backend services.

**XAMPP:** Local server environment for testing and development.

# **List of figures**

## **Use Case Diagram**

A use case diagram is a way to summarize details of a system and the users within that system. It is generally a graphical representation of interaction among different elements in a system. The bugs finder system two actor are Users and Admin.

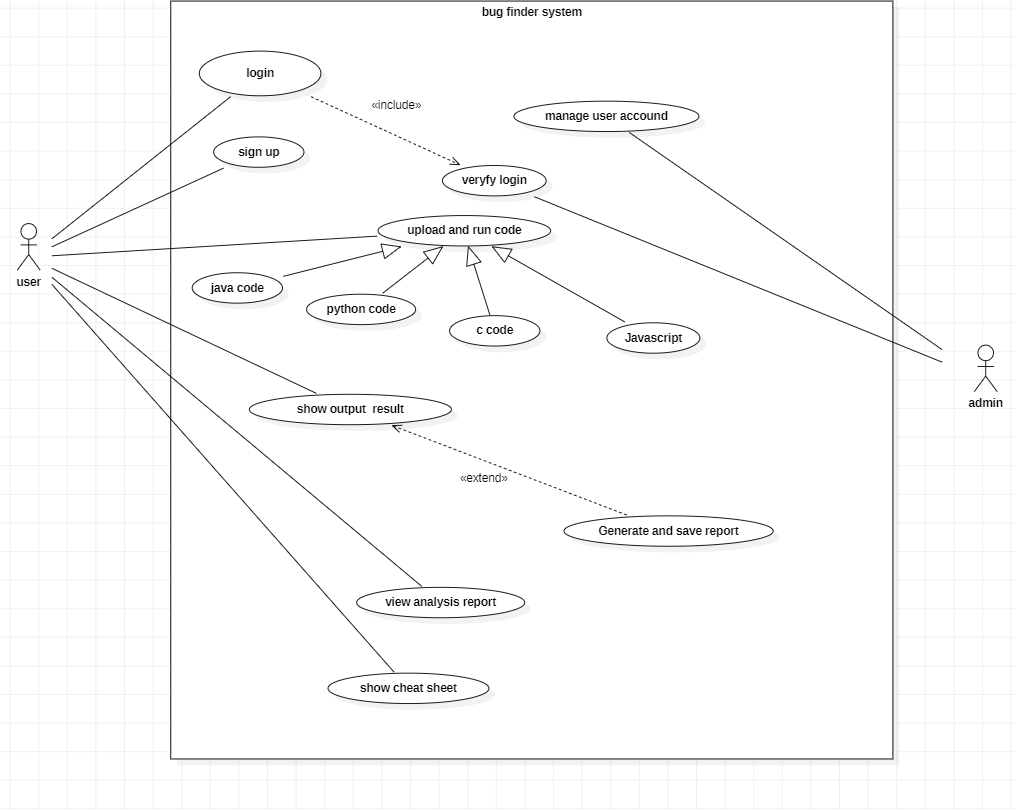


Figure 2: Use case Diagram

## **Sequence Diagram**

Sequence diagrams are a popular dynamic modelling solution in UML because they specifically focus on lifelines or the processes and objects that live simultaneously, and the messages exchanged between them to perform a function before the lifeline ends.

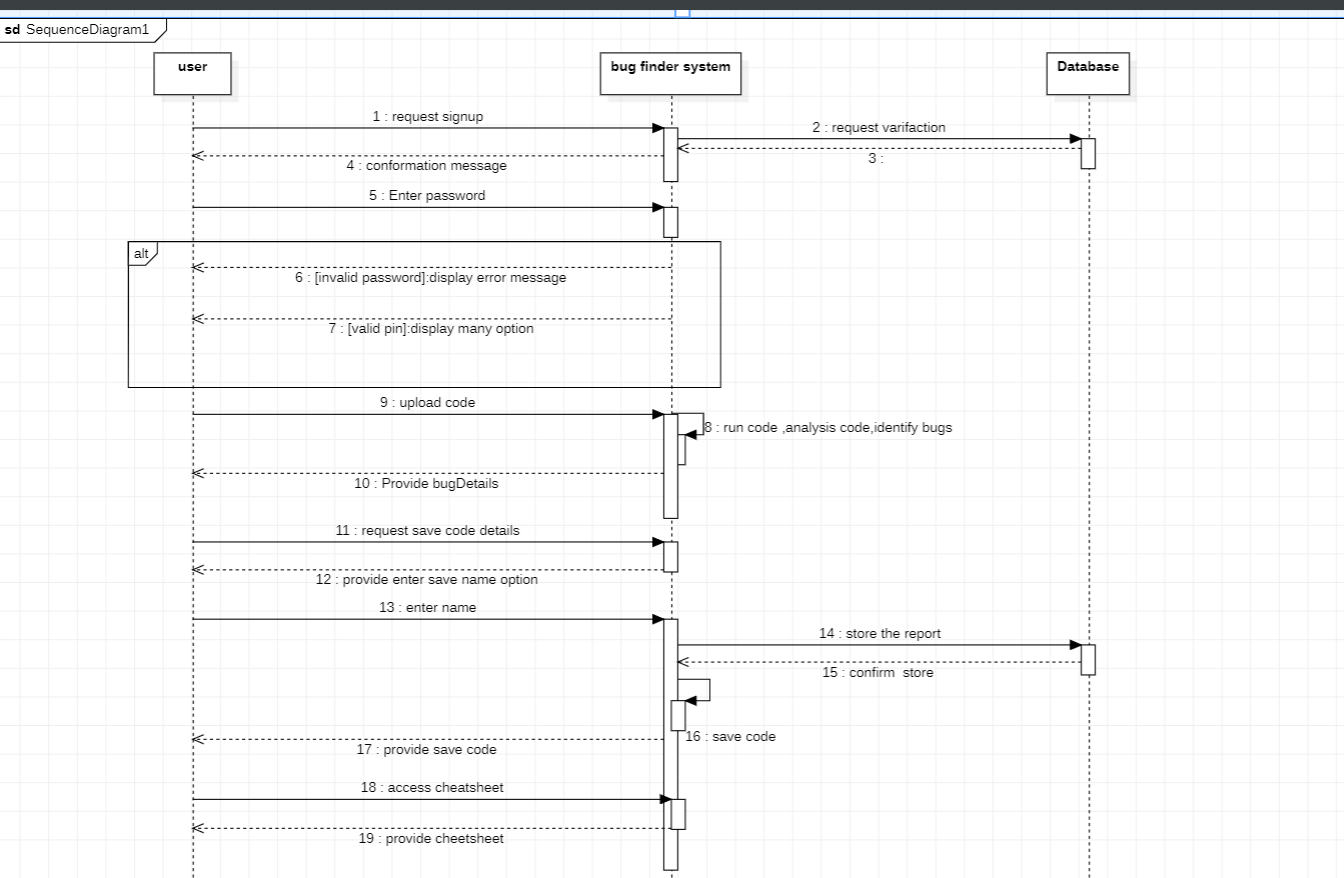


Figure 3: Sequence Diagram

Three object user, bugs finder system, database are shown in the above diagram.

## **Activity diagram**

The Active diagram for the bug finder system outlines the flow of activities and processes involved in using the system.

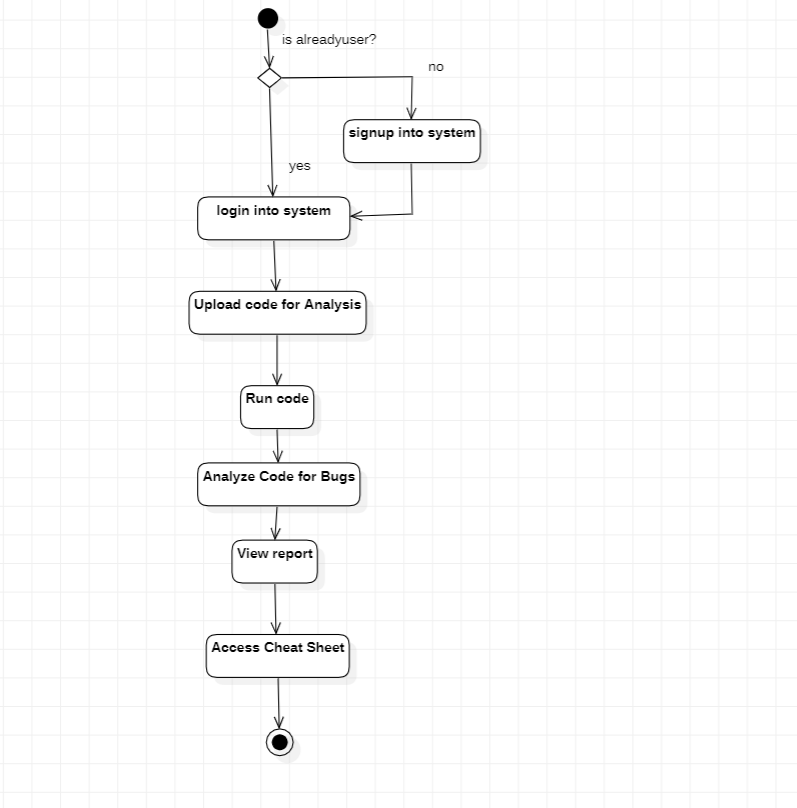


Figure 4: Activity diagram

## **ER-diagram**

An Entity-Relationship (ER) diagram is a visual representation of the data and the relationships between entities within a system.

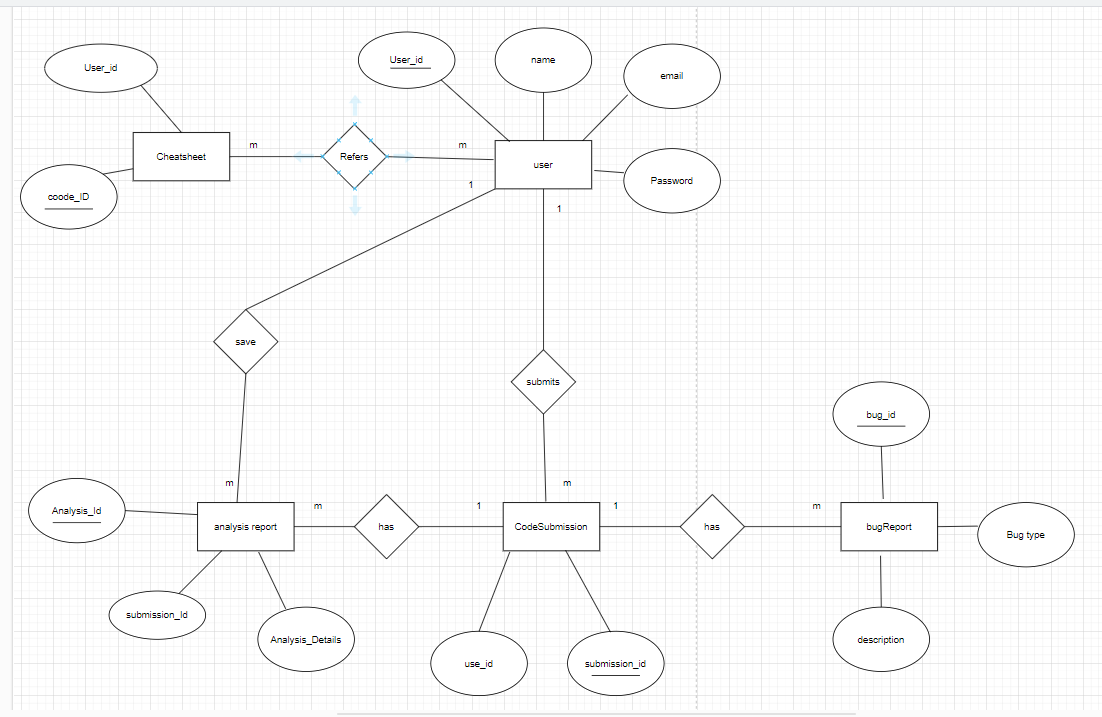
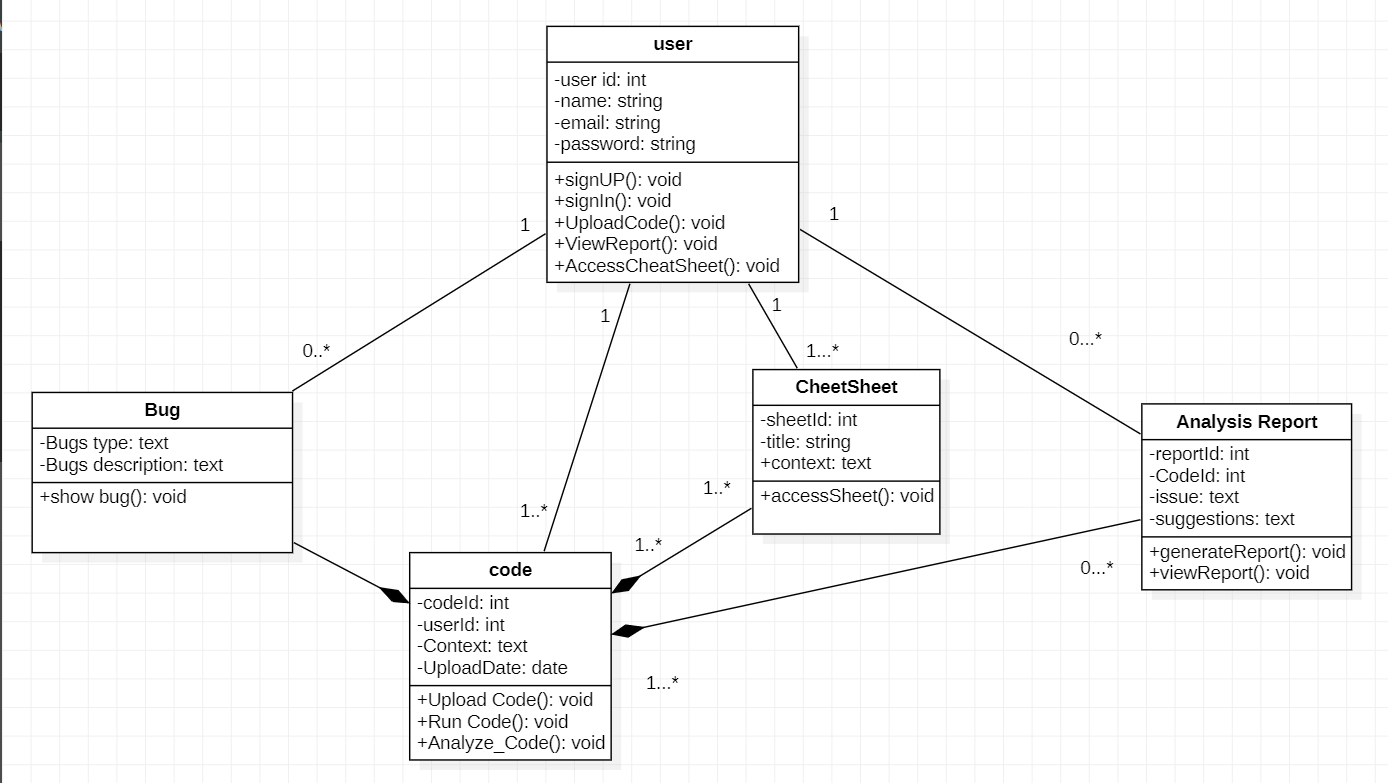


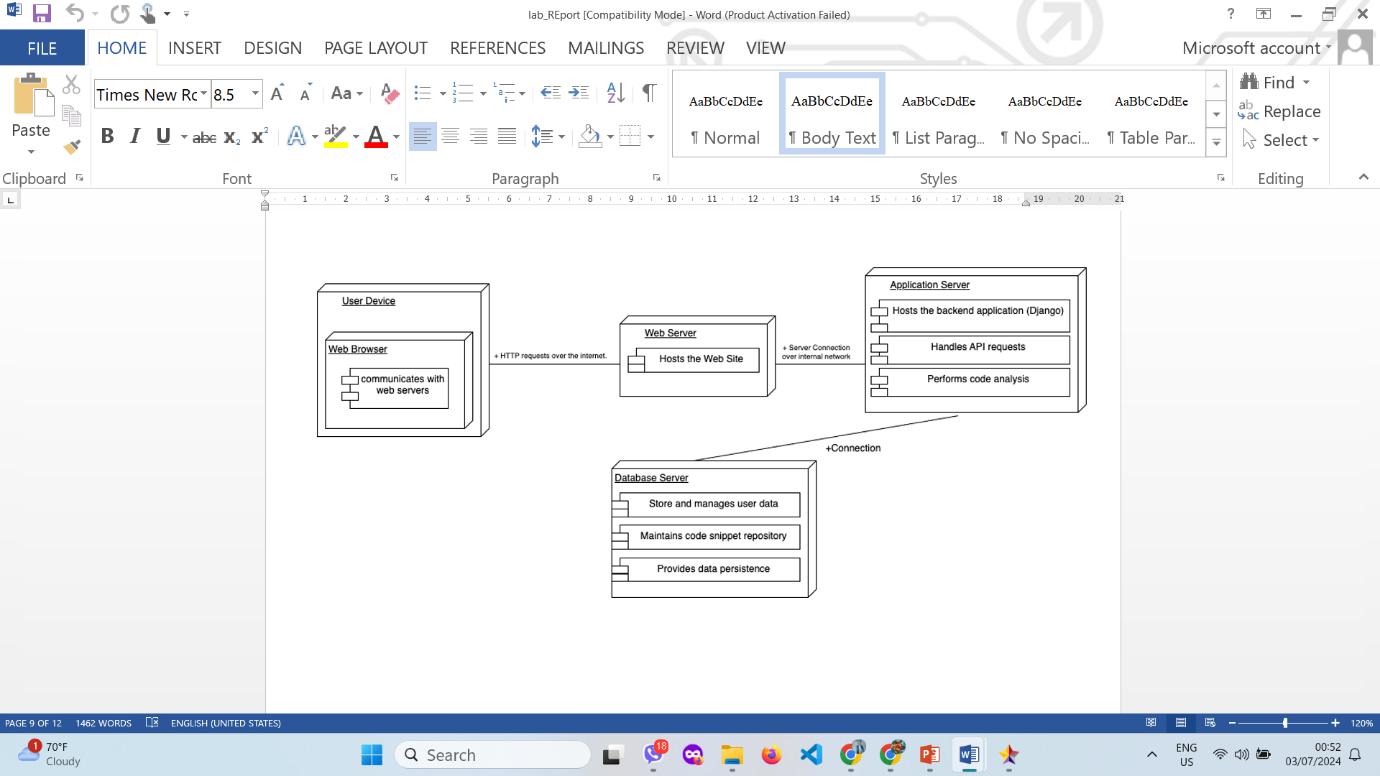
Figure 5: ER-Diagram

## **Class-diagram**

A class diagram is a type of static structure diagram in Unified Modelling Language (UML) that describes the structure of a system by showing its classes, attributes, methods, and the relationships among objects.

 Figure 6: Class diagram

## **Development model**

****

# **Project Task and Time Schedule**

|  |  |  |
| --- | --- | --- |
| Team Member | Role | Responsibilities |
| BIKASH BANJADE | Frontend Developer | Prepare the User Interface(UI) of Bug Finder Website and Documentation. |
| APSARA ARYAL | Frontend Developer | Prepare the User Interface (UI) of Bugs Finder system Website and documentation. |
| DHIRAJ YADAV | Backend Developer | Integrate the Gemini-1.5-flash model, Handle API endpoints requests from model and send responses to the Website. |
| SANDESH ADHIKARI | Backend Developer | Manage the database and handle project. |

# **Testing**

To make sure all the elements of our “Bug finder system developed function properly, we created test cases for our work, where validation, reliability and user acceptance were tested. The following testing table shows all the tests.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Unit Test** | **Expected Result** | **Outcome** |
| 1 | Layout | Overall layouts of all page function correctly. | success |
| 2 | User Signup and Signin | User can successfully register and login. | success |
| 3 | Code Upload and Execution | User can upload code and see execution results. | success |
| 4 | Code Analysis | Code is analysed and diagnostics are provided. | success |
| 5 | Cheat Sheets | Users can access and view cheat sheets for programming language. | success |
| 6 | Report Generation and Saving | User can generate and save reports based on code analysis. | success |

# **Results and Conclusion**

The our “Bugs Finder system” successfully provides a comprehensive platform for debugging code efficiently. It allow users to securely upload and execute their code, receive detailed bug reports using the Google Gemini API, and access educational resources like cheat sheets for various programming languages. The system’s user-friendly interface, combined with robust search functionality and scalable architecture, ensures that users can effectively manage and resolve coding errors. By automating bug detection and enhancing analysis capabilities, the Bugs finder system meets the evolving needs of software developers, making the debugging process more efficient and user- friendly.

# **References**

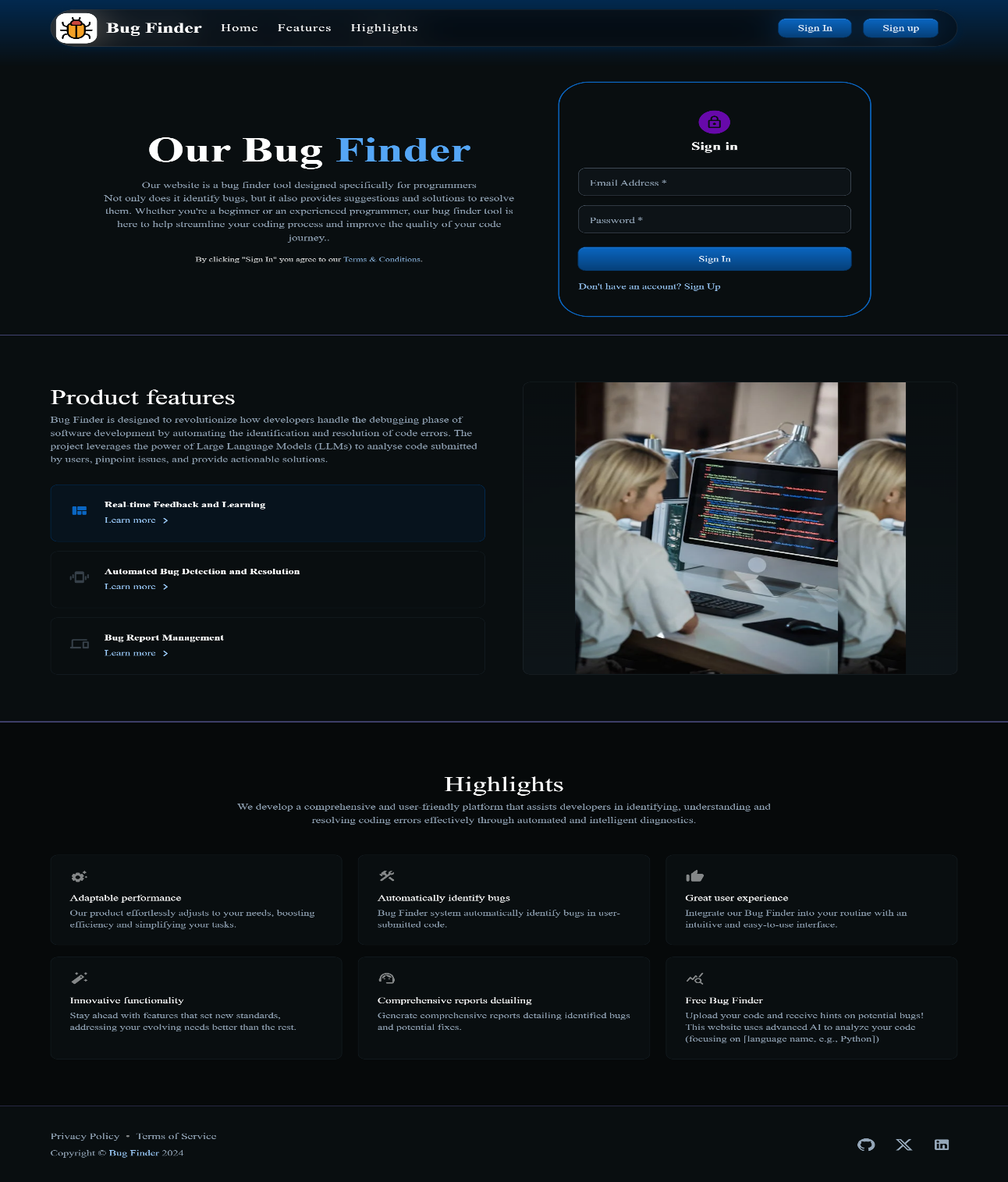
1. Nam, J., Wang, S., Xi, Y., & Tan, L. (2019). A bug finder refined by a large set of open-source projects. *Information and Software Technology*, *112*, 164-175.
2. Williams, C. C., & Hollingsworth, J. K. (2004, May). Bug Driven Bug Finders. In *MSR* (pp. 70-74).
3. Xufeng Yao, Haoyang Li, Tsz Ho Chan, “HDLdebugger: Streamlining HDL debugging with Large Language Model” vol-4, p. 1-8, August 25–29, 2024.
4. Shapiro, D. G. (1980). A Proposal for Sniffer: a System that Understands Bugs.
5. Roy, S., Pandey, A., Dolan-Gavitt, B., & Hu, Y. (2018, October). Bug synthesis: Challenging bug-finding tools with deep faults. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 224-234).
6. Hovemeyer, D., & Pugh, W. (2004). Finding bugs is easy. *Acm sigplan notices*, *39*(12), 92-106.
7. [Aarnphm](https://github.com/aarnphm) [Aaron Pham](https://github.com/aarnphm) , “*OpenLLM documentations*” [Online],

Available : <https://github.com/bentoml/OpenLLM?tab=readme-ov-file>.

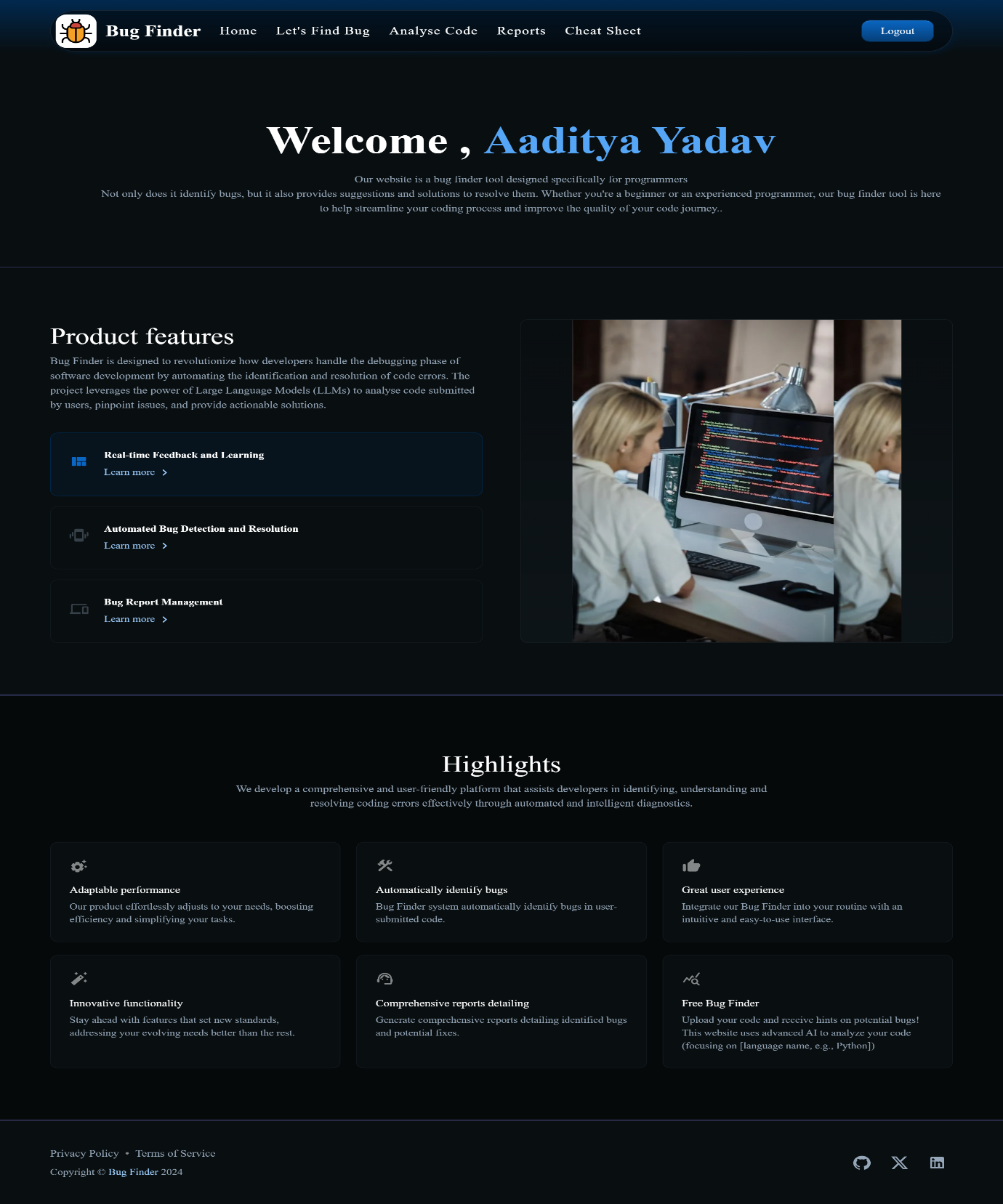
# **APPENDIX – I**

This appendix lists the screenshots of various screenshot’s of UI. Please note that the UI could be quite different on user devices because the layouts depend on screen size, pixel quantity, etc. of the target device.

# **Home-Page**

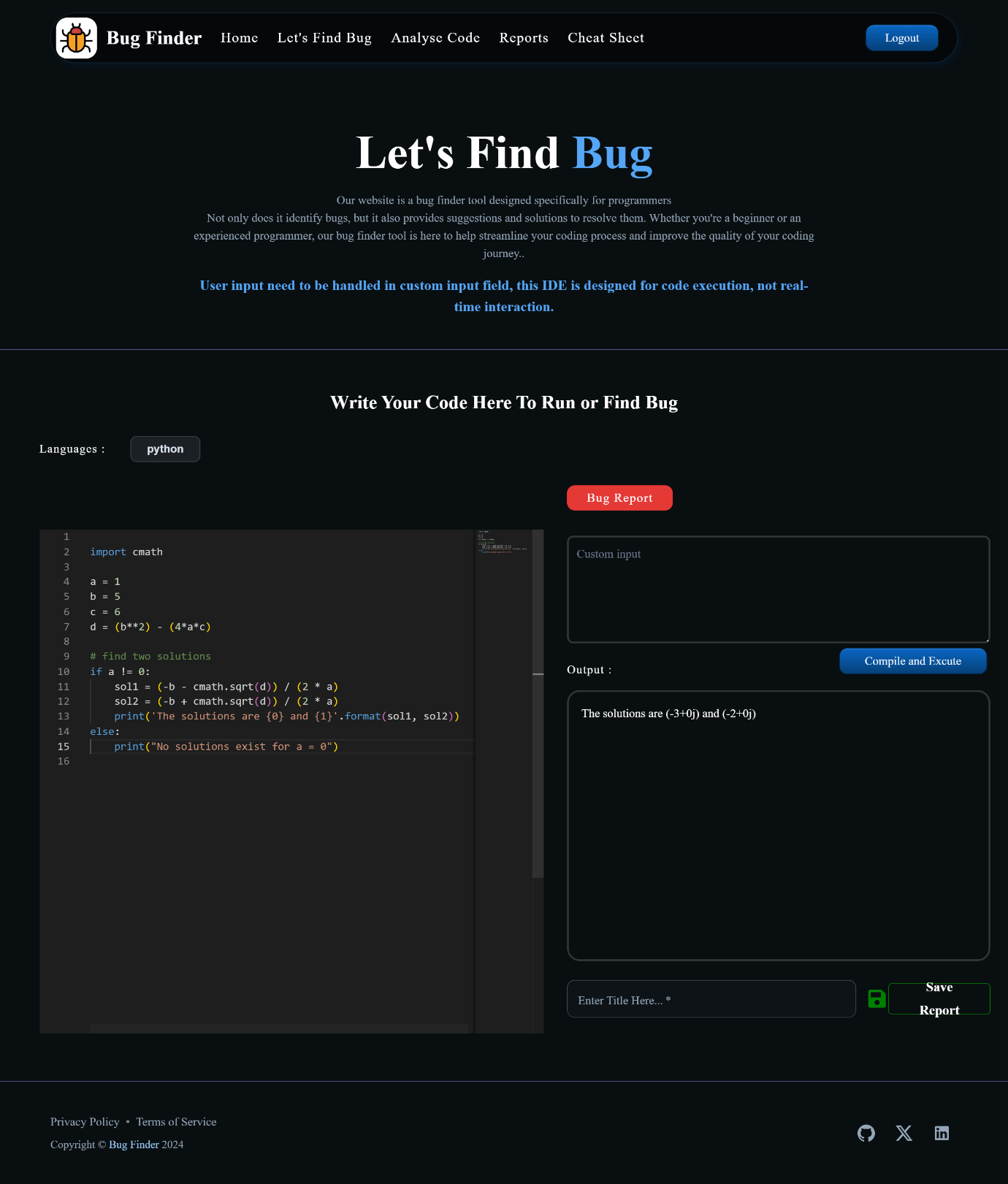


# **Landing-Page**



# 

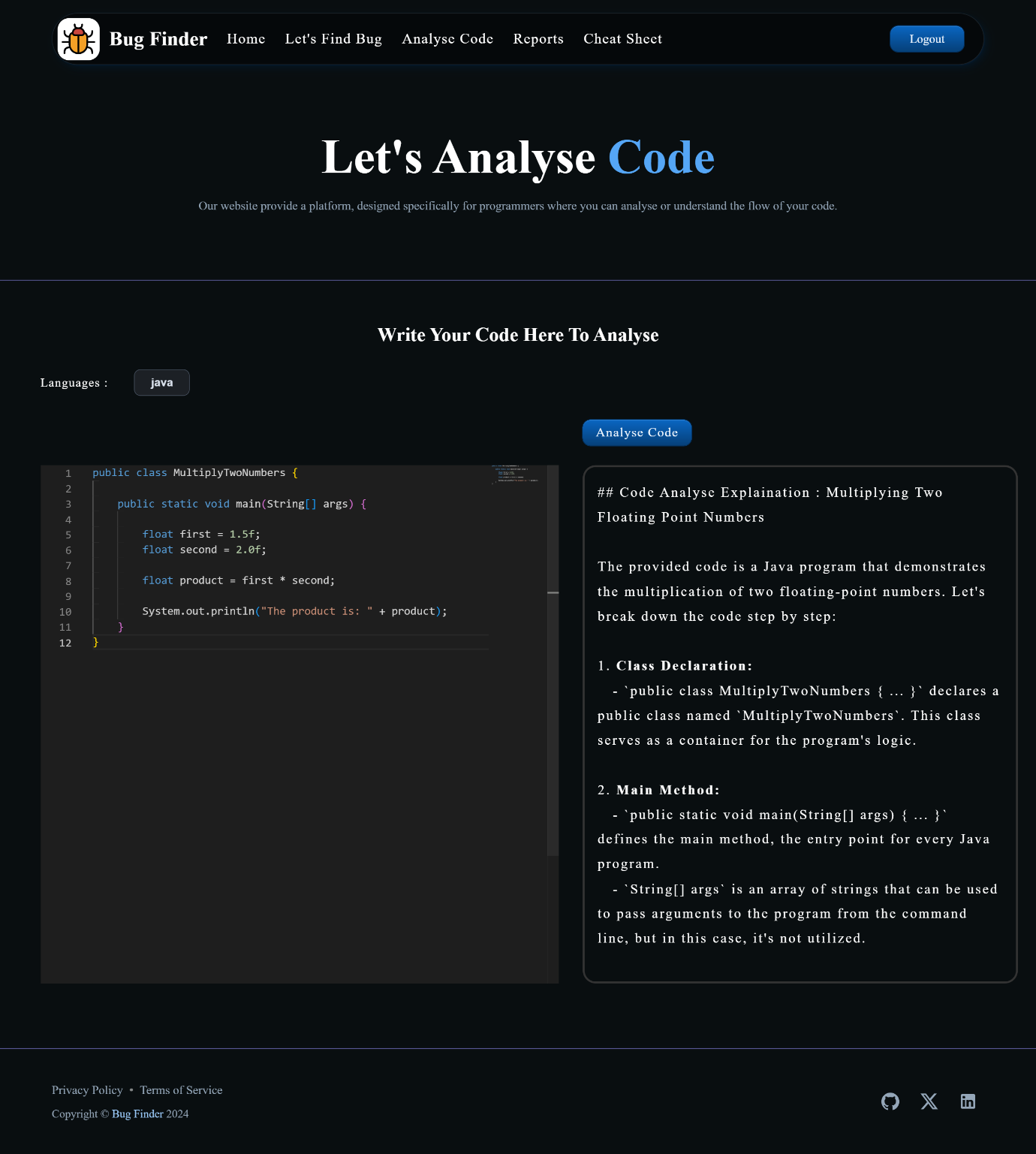
# **Run-Code**



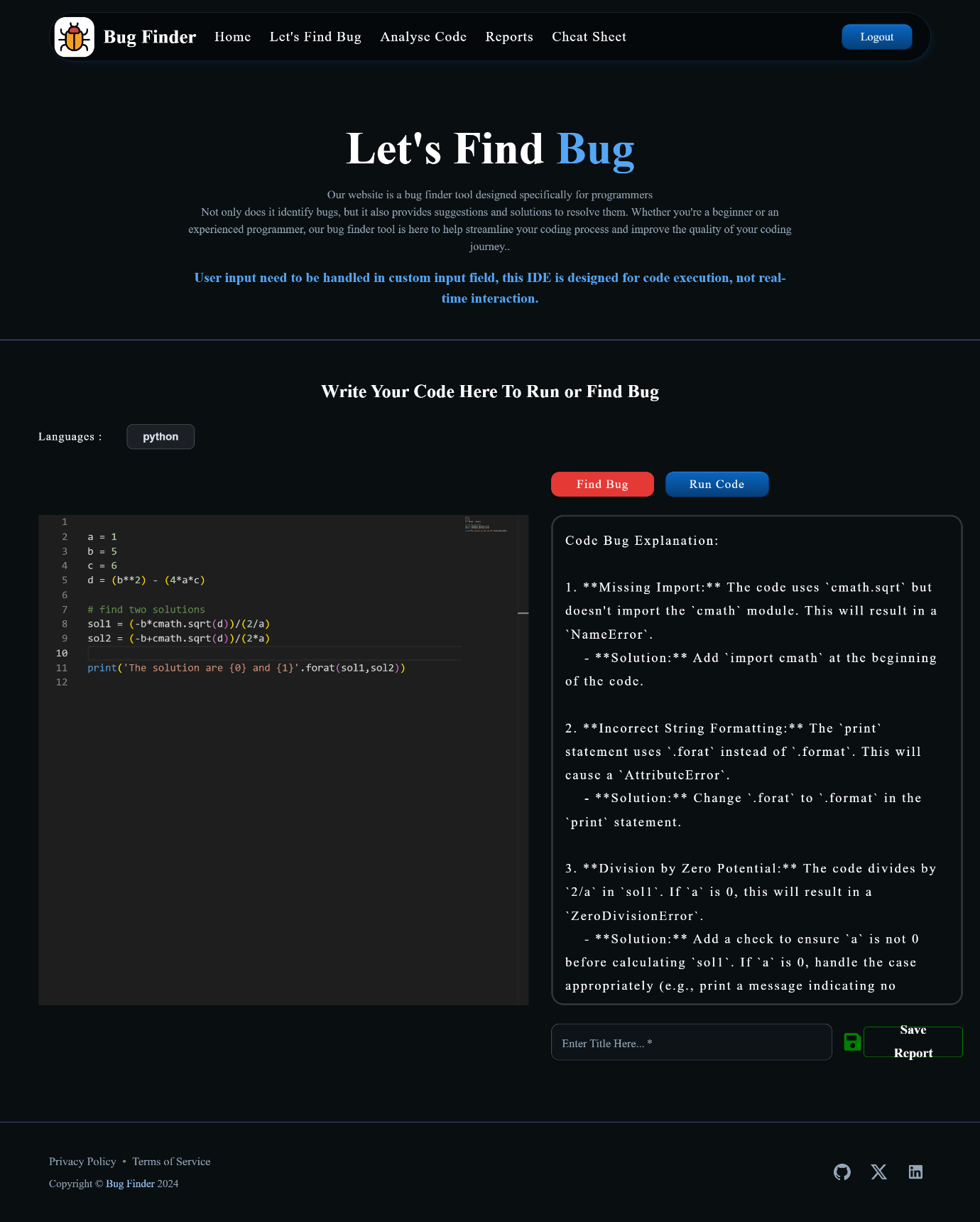
# **Reports-Page**



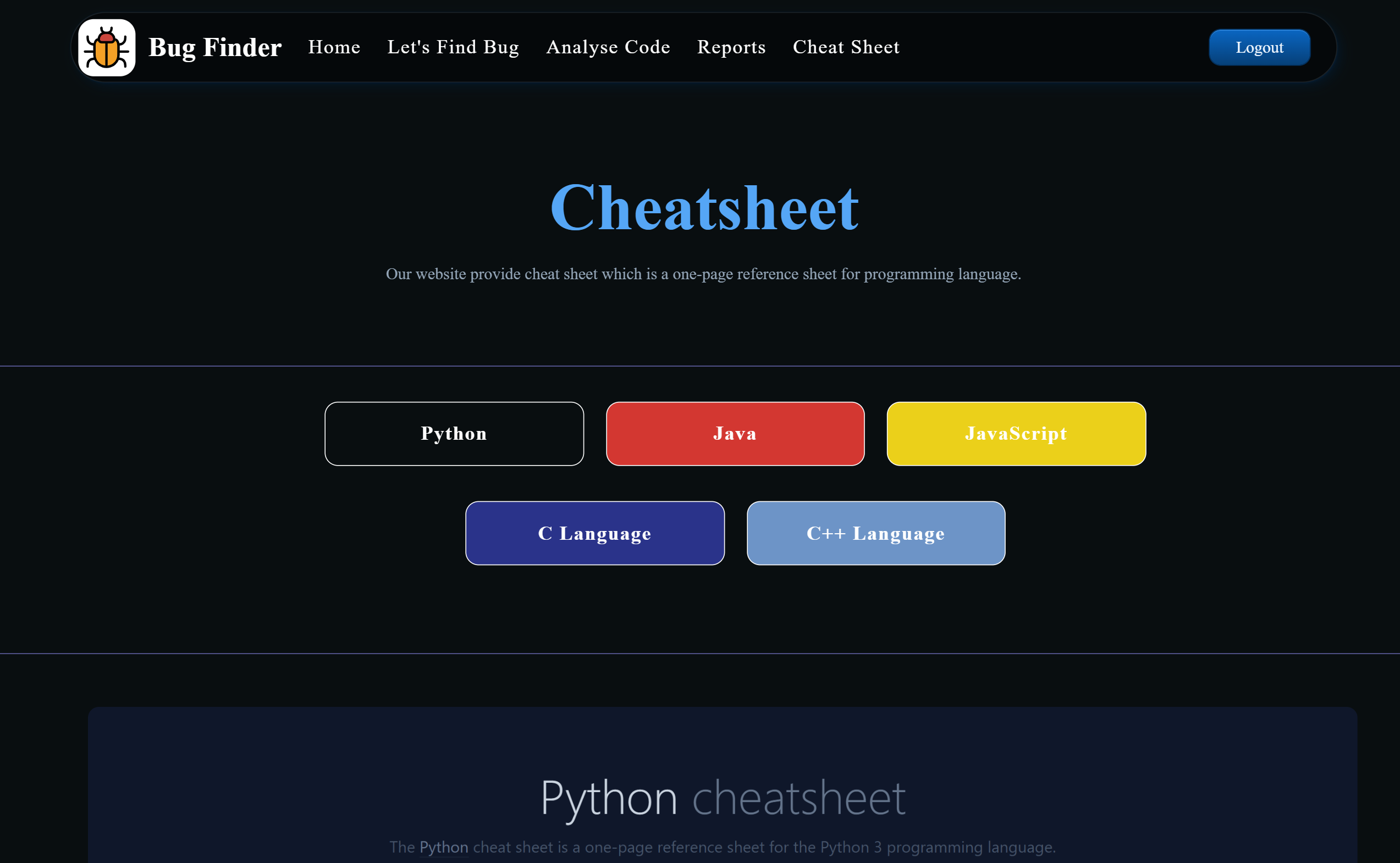
# **Analyse-Code-Page**



# **Find-Bug Page**



# **Cheat-Sheet Page**



# **Project Progress Log Sheet**

