AI powered code review system.

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology

In

Computer Science and Engineering School of

Engineering and Sciences

Submitted by

Sai Varshith Popuri (AP21110011034)

Omkar Kaushik Gadde(AP21110011019)

Under the Guidance of (Mr.Hema Kumar Yarnagula)

SRM University–AP Neerukonda, Mangalagiri, Guntur,

AndhraPradesh – 522 240 | May, 2024

Certificate

Date: 15-May-24

This is to certify that the work present in this Project entitled "**AI powered code review system.**" has been carried out by [**Varshith , Omkar]** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

Supervisor
(Signature)

Prof./ Dr. [Mr.Hema Kumar Yarnagula]

Designation

Affiliation.

Co-supervisor

(Signature)

Dr. [Name]

Designation

Affiliation

Acknowledgements

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success.

I am extremely grateful and express my profound gratitude and indebtedness to my project guide, Mr Hema Kumar Yarnagula, Department of Computer Science & Engineering, SRM University, Andhra Pradesh, for her kind help and for giving me the necessary guidance and valuable suggestions in completing this project work.

SOFTWARE REQUIREMENTS

SPECIFICATION FOR

AI powered code review system.

Group Members :

- Omkar Kaushik Gadde(AP21110011019)
- Sai Varshith Popuri(AP21110011034)

Table of Contents

Abstract :

The "AI Powered Code Review System" is an innovative web-based platform designed to revolutionize the code review process by harnessing the capabilities of artificial intelligence (AI) technology. In today's fast-paced software development landscape, efficient code review is essential for ensuring code quality, identifying bugs, and improving overall software reliability. However, traditional manual code review processes can be time-consuming, error-prone, and resource-intensive.

The AI Powered Code Review System addresses these challenges by automating and augmenting key aspects of the code review process. Through an intuitive web interface, users can submit their code for review, leveraging the system's AI-powered analysis capabilities. Behind the scenes, sophisticated machine learning algorithms analyze the submitted code, identifying potential issues such as bugs, security vulnerabilities, performance bottlenecks, and adherence to coding standards.

One of the system's core functionalities is the generation of comprehensive feedback reports based on the analysis results. These reports provide detailed insights into the identified issues, including their severity, impact, and recommended remediation steps. By surfacing actionable feedback in a clear and concise manner, the system

empowers developers to address code quality issues more effectively and efficiently.

Furthermore, the AI Powered Code Review System incorporates features for tracking the history of code reviews, allowing users to monitor the progress of their projects over time. This historical tracking capability enables teams to identify trends, track improvements, and maintain a record of past code review activities.

In addition to its technical capabilities, the system also prioritizes user experience and accessibility. With a user-friendly interface and customizable settings, developers of all skill levels can easily navigate the platform, submit code for review, and interpret analysis results.

Overall, the AI Powered Code Review System represents a significant advancement in the field of software development, offering a scalable, intelligent, and efficient solution for code review that accelerates development cycles, enhances code quality, and drives innovation in software engineering practices.

1. Introduction

The AI Powered Code Review System represents a cutting-edge approach to software development, integrating artificial intelligence (AI) and machine learning (ML) technologies into the code review process. By automating analysis and providing intelligent recommendations, this system aims to enhance code quality, streamline review workflows, and improve developer productivity.

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to outline the requirements and specifications for the development of the AI Powered Code Review System. This document serves as a comprehensive guide for stakeholders, including developers, testers, and project managers, to understand the objectives, features, and functionalities of the proposed system.

1.2 Scope

The AI Powered Code Review System streamlines code evaluation, leveraging AI and machine learning for automated analysis. Its scope includes seamless integration with Git and support for multiple programming languages. By automating routine tasks, developers can focus on innovation, boosting productivity. The system evolves iteratively through user

feedback, ensuring accuracy over time. It scales effortlessly from startups to enterprises, offering flexibility in deployment options. Security features prioritize code protection and compliance with industry standards. In essence, it revolutionizes code review, enhancing efficiency, insight, and collaboration within software development teams.

1.3 References

The development of the AI Powered Code Review System is informed by the following references:

IEEE Standard for Software Requirements Specifications (IEEE Std 830-1998)

Python Documentation

Django Documentation

JavaScript Documentation

HTML5 and CSS3 Documentation

The references for the above software are as follows:-

https://ai.google.dev/gemini-api

https://chatgpt.com/c/420def24-b174-42f5-94fc-cfe6bb1b6f3f

1.4 Overview

Section 1.0 discusses the purpose and scope of the software.
Section 2.0 describes the overall functionalities and constraints of the software and user characteristics.
Section 3.0 details all the requirements needed to design the software.

2. The Overall Description

2.1 Product Perspective

The AI Powered Code Review System serves as a comprehensive web application that operates independently to facilitate code reviews efficiently. While it functions as a standalone system, it can seamlessly integrate with version control platforms like Git to enhance code management and review processes. This integration allows users to fetch code repositories directly into the system for analysis and feedback generation, streamlining the overall workflow of code review and collaboration.

2.2 Product Functions

- **Automated code analysis:** The system employs sophisticated machine learning algorithms to automatically analyze submitted code snippets. It identifies errors, inefficiencies, and deviations from coding standards, providing detailed insights into code quality and potential improvements.

- **Feedback generation:** Based on the code analysis, the system generates comprehensive feedback reports for users. These reports highlight specific issues within the code, suggest corrective actions, and offer best practices to enhance code quality and maintainability.

- **User management:** The system offers robust user management capabilities, allowing users to create accounts, securely log in, and manage their profiles. Users can customize their preferences, access their review history, and track the progress of ongoing code reviews.

- **Integration with version control:** By integrating with version control platforms such as Git, the system offers enhanced code management capabilities. Users can seamlessly fetch code repositories into the system, initiate code reviews directly from version control, and synchronize feedback with code repositories, ensuring seamless collaboration and version control.

2.3 User Characteristics

The system caters to a diverse user base, including developers, software engineers, and coding enthusiasts. Users of the system typically possess a foundational understanding of programming concepts and are comfortable navigating web-based applications. While the system accommodates users with varying levels of coding expertise, it is designed to provide valuable insights and

actionable feedback to users at all skill levels.

2.4 Constraints

- Performance: The system's performance may be influenced by factors such as server resources, network latency, and the complexity of code being analyzed. Optimizing performance to ensure timely and accurate code analysis is a key consideration.

- Scalability: As the user base and volume of code submissions grow, ensuring scalability and responsiveness becomes increasingly important. The system must be able to handle increased traffic and workload efficiently without compromising performance or user experience.

- Security: Protecting user data, code submissions, and feedback reports from unauthorized access or tampering is paramount. Implementing robust security measures, such as encryption, access controls, and secure authentication mechanisms, is essential to safeguarding sensitive information.

2.5 Assumptions and Dependencies

The requirements stated in the SRS could be affected by the following factors:

- Availability of training data: The effectiveness of the machine learning models used for code analysis depends on the availability of sufficient and relevant training data. Access to diverse and high-quality training datasets is essential to ensure accurate and reliable code analysis results.
- Stable internet connection: Users are assumed to have a stable internet connection to access the web application and submit code for review. Uninterrupted connectivity is necessary to facilitate smooth interactions with the system and ensure timely feedback delivery.
- Browser compatibility: The system assumes compatibility with modern web browsers to provide a seamless user experience across different devices and platforms. Compatibility testing with popular browsers ensures that users can access and utilize the system without encountering compatibility issues.
- Third-party dependencies: The system may rely on third-party libraries, frameworks, or APIs to support various functionalities, such as code analysis, user authentication, and version control integration. Managing dependencies and ensuring their availability and reliability are essential to maintain system functionality and performance.

3. External Interface Requirements

3.1.1 User Interface Requirements

The user interface of the AI Powered Code Review System is designed to be intuitive, user-friendly, and responsive, providing an efficient and engaging experience for users interacting with the system. Key user

interface requirements include:

- Dashboard: The system presents users with a dashboard upon login, displaying relevant information such as recent code submissions, analysis results, and notifications. The dashboard serves as the central hub for accessing various functionalities and monitoring the status of code reviews.
- Submission Form: Users can submit code for review through a submission form accessible from the dashboard or navigation menu. The submission form includes fields for uploading code files, entering project details, and specifying review preferences such as analysis depth and focus areas.
- Analysis Results: Upon submitting code for review, users receive comprehensive analysis results presented in a clear and structured manner. The results include insights into code quality, potential issues, performance optimizations, and best practices. Visualizations, charts, and graphs may accompany textual feedback to enhance understanding.
- Feedback Viewer: Users can view detailed feedback generated by the system for each code submission, including recommendations for improvements, code snippets, and explanations. The feedback viewer provides interactive features such as code highlighting, inline comments, and navigation options for easy exploration and understanding.
- Settings Panel: Users have access to a settings panel where they can customize preferences, configure notification settings, and manage account details. The settings panel offers options for adjusting analysis parameters, integrating with version control systems, and enabling collaboration features.
- Navigation Menu: The system features a navigation menu with intuitive navigation options, including links to the dashboard, submission form, analysis results, settings panel, and user profile. The navigation menu ensures easy access to key functionalities and promotes efficient workflow navigation.
- Responsive Design: The user interface is designed to be responsive and adaptable to different screen sizes and devices, ensuring optimal usability across desktop computers, laptops, tablets, and smartphones. Responsive design principles are applied to optimize layout, typography, and interactive elements for each device type.

By fulfilling these user interface requirements, the AI Powered Code Review System delivers an engaging, efficient, and accessible user experience, empowering developers and teams to improve code quality, collaboration, and productivity effectively.

3.1.2 Hardware Interface Requirements

The AI Powered Code Review System primarily operates as a web-based application and does not have specific hardware requirements beyond those typical for accessing web applications. Users can interact with the system using standard computing devices such as desktop

computers, laptops, tablets, and smartphones connected to the internet. The system is platform-independent and compatible with various operating systems, including Windows, macOS, Linux, iOS, and Android.

While the system itself does not have hardware dependencies, it may benefit from hardware acceleration capabilities for processing large code repositories and executing resource-intensive machine learning algorithms. However, these hardware enhancements are optional and not essential for basic system functionality.

3.1.3 Software Interface Requirements

- The AI Powered Code Review System relies on a combination of software components and external services to deliver its functionality seamlessly. Key software interface requirements include:
- Web Browser: Users access the system through a standard web browser such as Google Chrome, Mozilla Firefox, Apple Safari, or Microsoft Edge. The system is designed to be compatible with modern web browsers, ensuring a consistent user experience across different platforms.
- Version Control Systems: The system integrates with popular version control platforms such as Git to fetch code repositories for analysis and feedback generation. It communicates with version control systems using standard protocols such as HTTP/HTTPS and SSH, allowing users to authenticate and authorize access to their repositories securely.
- Machine Learning Frameworks: The system utilizes machine learning frameworks and libraries to implement code analysis algorithms and models. Examples include TensorFlow, PyTorch, Scikit-learn, and NLTK (Natural Language Toolkit). These frameworks provide essential functionalities for training, evaluating, and deploying machine learning models within the system.
- Database Management System (DBMS): The system relies on a DBMS to store and manage user data, code repositories, analysis results, and feedback reports. Commonly used database systems include MySQL, PostgreSQL, SQLite, MongoDB, or cloud-based solutions such as Amazon RDS or Google Cloud SQL. The choice of DBMS depends on factors such as scalability, performance, and data storage requirements.
- RESTful APIs: The system exposes RESTful APIs (Representational State Transfer) to enable integration with third-party applications, services, and development tools. These APIs allow external systems to interact with the code review system programmatically, accessing functionalities such as submitting code for review, fetching analysis results, and managing user accounts.
- By adhering to these software interface requirements, the AI Powered Code Review System ensures interoperability, flexibility, and ease of integration with existing software ecosystems, enabling seamless collaboration and productivity for developers and software

development teams.

4. System Features

1. Real Time Language Translation Application

Description

1. **Code Submission:** Users can submit their code for review through a user-friendly submission form. They provide essential details about the project, such as title, description, programming language, and specific areas of concern or focus for the review. This feature ensures that users can easily share their code with the system for comprehensive analysis.
2. **AI Analysis:** The system employs advanced machine learning algorithms to conduct a thorough analysis of the submitted code. This analysis encompasses various aspects, including syntax checking, style enforcement, code structure, complexity, performance, and potential issues like bugs, inefficiencies, or security vulnerabilities. By leveraging AI capabilities, the system can provide insightful and actionable feedback to users, helping them improve the quality, efficiency, and reliability of their code.
3. **Real-time Feedback:** Upon code submission, users receive real-time feedback from the system. This feedback includes detailed reports, visualizations, and recommendations highlighting both strengths and weaknesses of the code. Users can immediately identify areas for improvement, understand the rationale behind the suggestions, and take proactive steps to enhance their code quality. This real-time feedback mechanism enables agile development practices, fostering continuous improvement and iteration.
4. **Customization Options:** The system offers users a range of customization options to tailor the analysis process to their specific needs and preferences. Users can adjust parameters such as analysis depth, focus areas, severity thresholds, and preferred coding standards. This flexibility allows users to align the analysis with their project requirements, coding style, and quality standards, ensuring that the feedback provided by the system is relevant and actionable.
5. **Collaboration Tools:** To facilitate effective collaboration among team members, the system integrates collaboration features directly into the code review process. Users can provide inline comments, suggestions, or annotations on specific code segments. Additionally, the system supports code highlighting, version control integration, and notifications, enabling seamless communication and coordination among team members. These collaboration tools enhance productivity, knowledge sharing, and collective problem-solving within development teams.
6. **Performance Monitoring:** The system includes performance monitoring capabilities to track the progress and impact of code submissions over time. Users can monitor key metrics such as code quality scores, defect densities, and improvement trends through intuitive dashboards and reports. By analyzing historical data and trends, users can identify patterns, prioritize areas for improvement, and make informed decisions to optimize their development

processes and outcomes.

Validity Checks

- Code submissions are sequenced based on various factors such as submission time, priority settings, and workload distribution. The system prioritizes urgent or critical review requests to ensure timely attention and resolution of pressing issues. By efficiently sequencing analysis tasks, the system optimizes resource allocation and processing efficiency, minimizing wait times and maximizing user satisfaction.

Error Handling/ Response to Abnormal Situations

- The system implements robust error handling mechanisms to effectively address abnormal situations, including server errors, network disruptions, and unexpected system behavior. In the event of an error or failure, the system provides informative error messages, prompts users for appropriate actions or interventions, and initiates automatic recovery procedures whenever possible. This proactive approach to error management helps minimize downtime, mitigate risks, and maintain uninterrupted service availability for users.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- **Response Time:** The system should respond to user interactions, such as code submission and feedback retrieval, within an acceptable timeframe, typically less than 5 seconds, to ensure a smooth and responsive user experience.
- **Scalability:** The system should be capable of handling increasing user loads and concurrent requests without significant degradation in performance. It should scale horizontally by adding additional resources or nodes to the infrastructure to accommodate growing demand effectively.
- **Resource Utilization:** The system should efficiently utilize computing resources, including CPU, memory, and storage, to optimize performance and minimize operational costs. It should implement efficient algorithms, caching mechanisms, and resource management techniques to avoid resource bottlenecks and maximize throughput.

5.2 Security Requirements

- **Data Encryption:** All sensitive user data, including code submissions, user credentials, and feedback reports, should be encrypted both in transit and at rest using industry-standard encryption algorithms and protocols to prevent unauthorized access or tampering.
- **Access Control:** The system should enforce strict access control policies to regulate user access based on roles, permissions, and authentication levels. It should authenticate users securely using strong authentication mechanisms such as multi-factor authentication (MFA) and implement role-based access control (RBAC) to restrict access to sensitive functionalities and data.
- **Data Privacy:** The system should comply with data privacy regulations and standards, such as GDPR, HIPAA, and CCPA, by implementing robust data privacy practices, including data anonymization, consent management, and data retention policies. It should prioritize user privacy and ensure that user data is handled confidentially and transparently.
- **Audit Logging:** The system should maintain comprehensive audit logs of user activities, system events, and security-related incidents to facilitate forensic analysis, compliance auditing, and incident response. It should record detailed information, including timestamps, user identities, and actions performed, to enable accountability and traceability of system behavior.
- **Vulnerability Management:** The system should undergo regular security assessments, vulnerability scans, and penetration testing to identify and remediate security vulnerabilities proactively. It should follow secure coding practices, perform code reviews, and apply patches and updates promptly to mitigate security risks and protect against common attack vectors such as SQL injection, cross-site scripting (XSS), and session hijacking.

6. Other Requirements

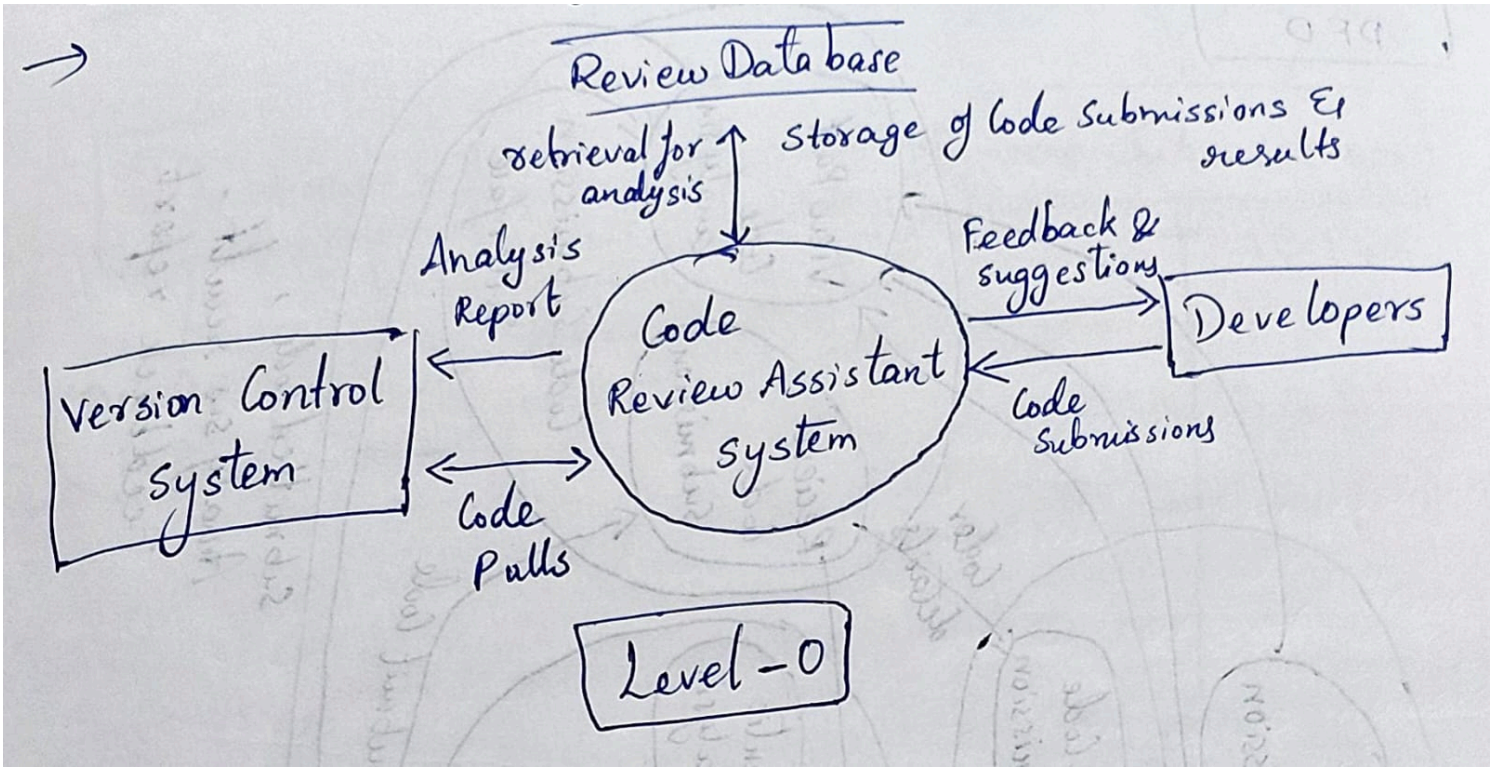
Appendix A: Glossary

- AI: Artificial Intelligence
- API: Application Programming Interface
- RBAC: Role-Based Access Control
- GDPR: General Data Protection Regulation
- HIPAA: Health Insurance Portability and Accountability Act
- CCPA: California Consumer Privacy Act
- SQL: Structured Query Language
- XSS: Cross-Site Scripting

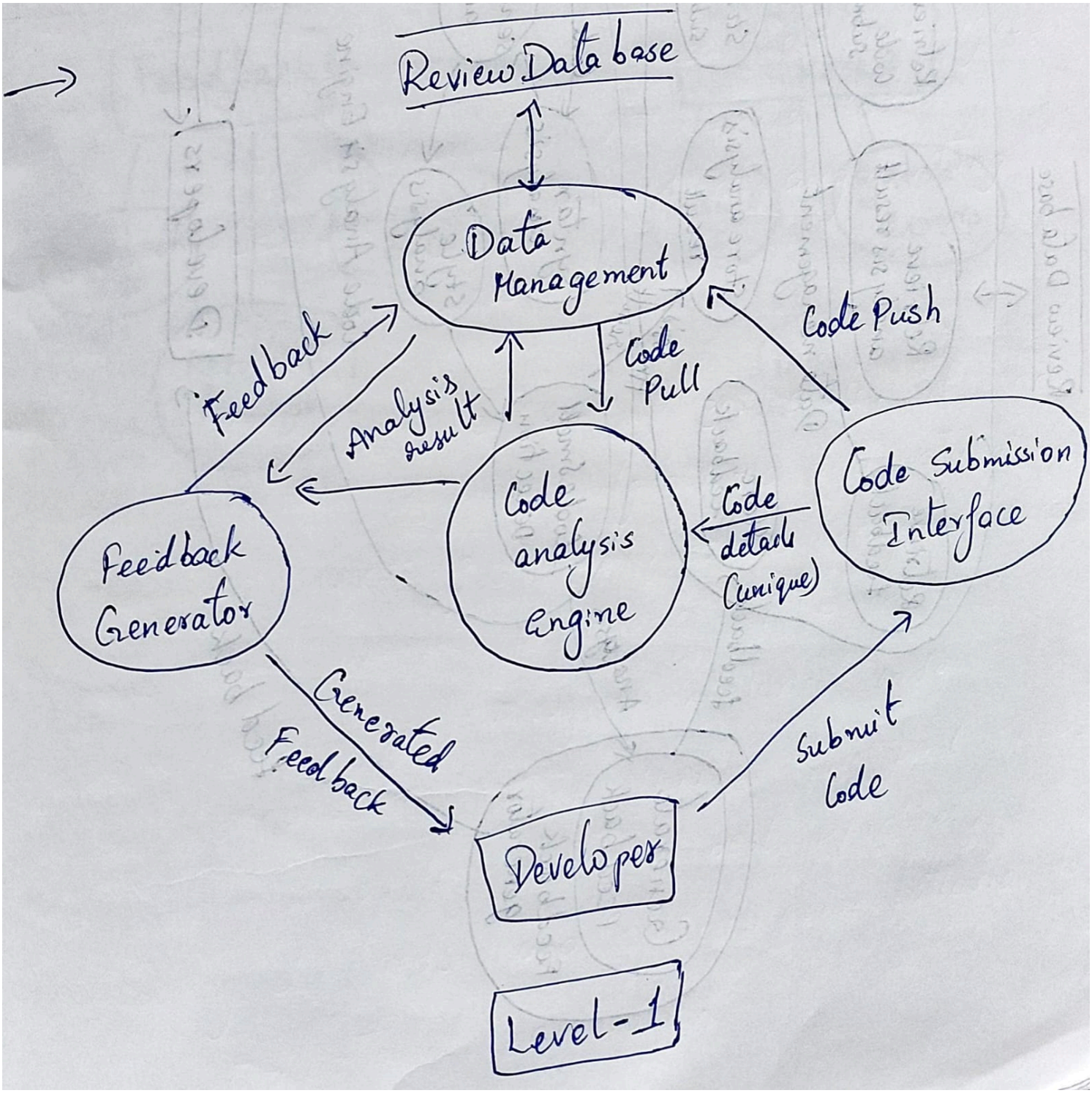
Appendix S: Analysis Models:

- Class Diagram: A visual representation of the system's classes, their attributes, methods, and relationships.
- Object Diagram: A snapshot of the system's objects and their relationships at a specific point in time.
- Sequence Diagram: A graphical depiction of the interactions between system components or objects over time, illustrating the flow of messages or method calls.

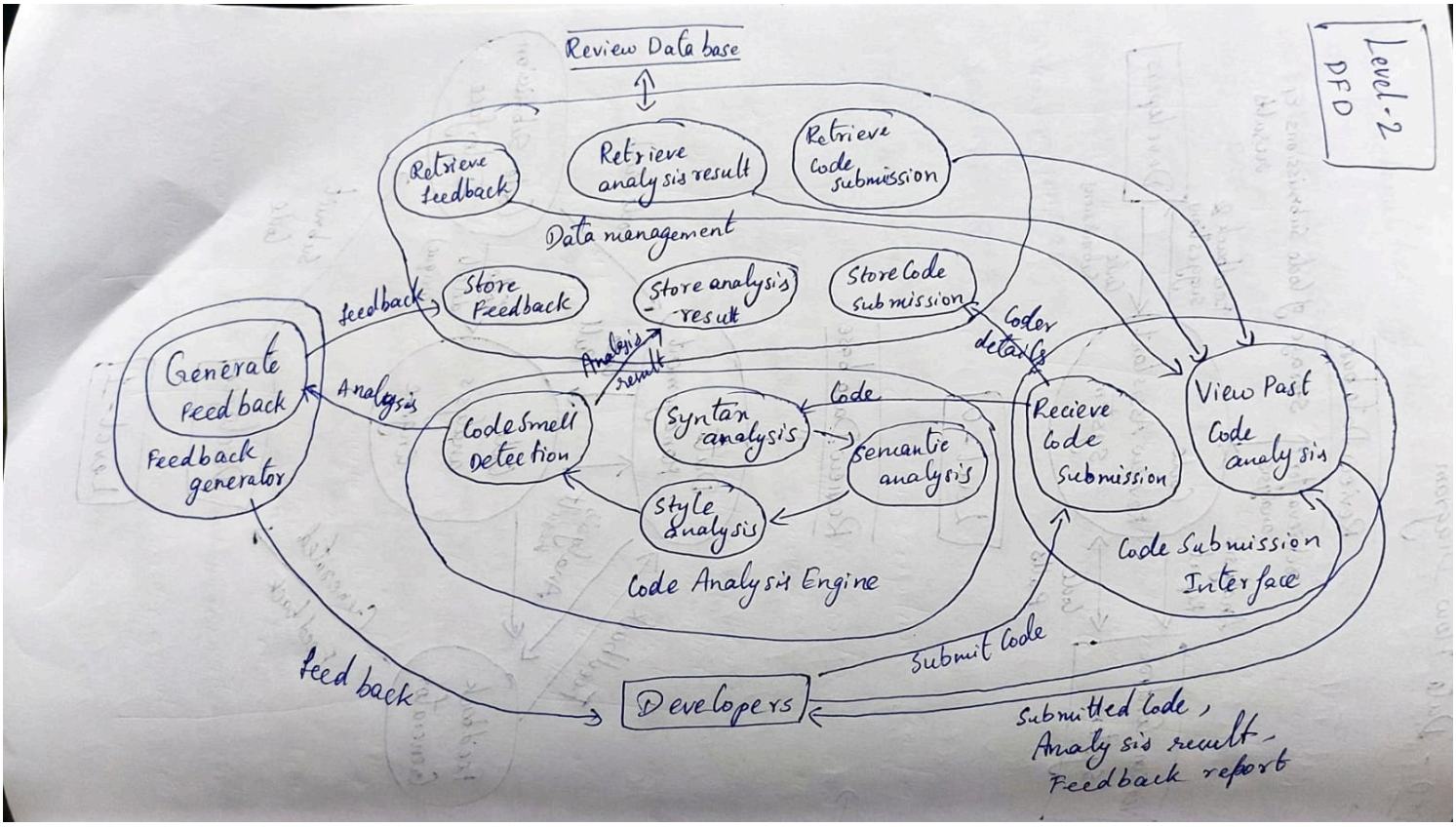
DFD



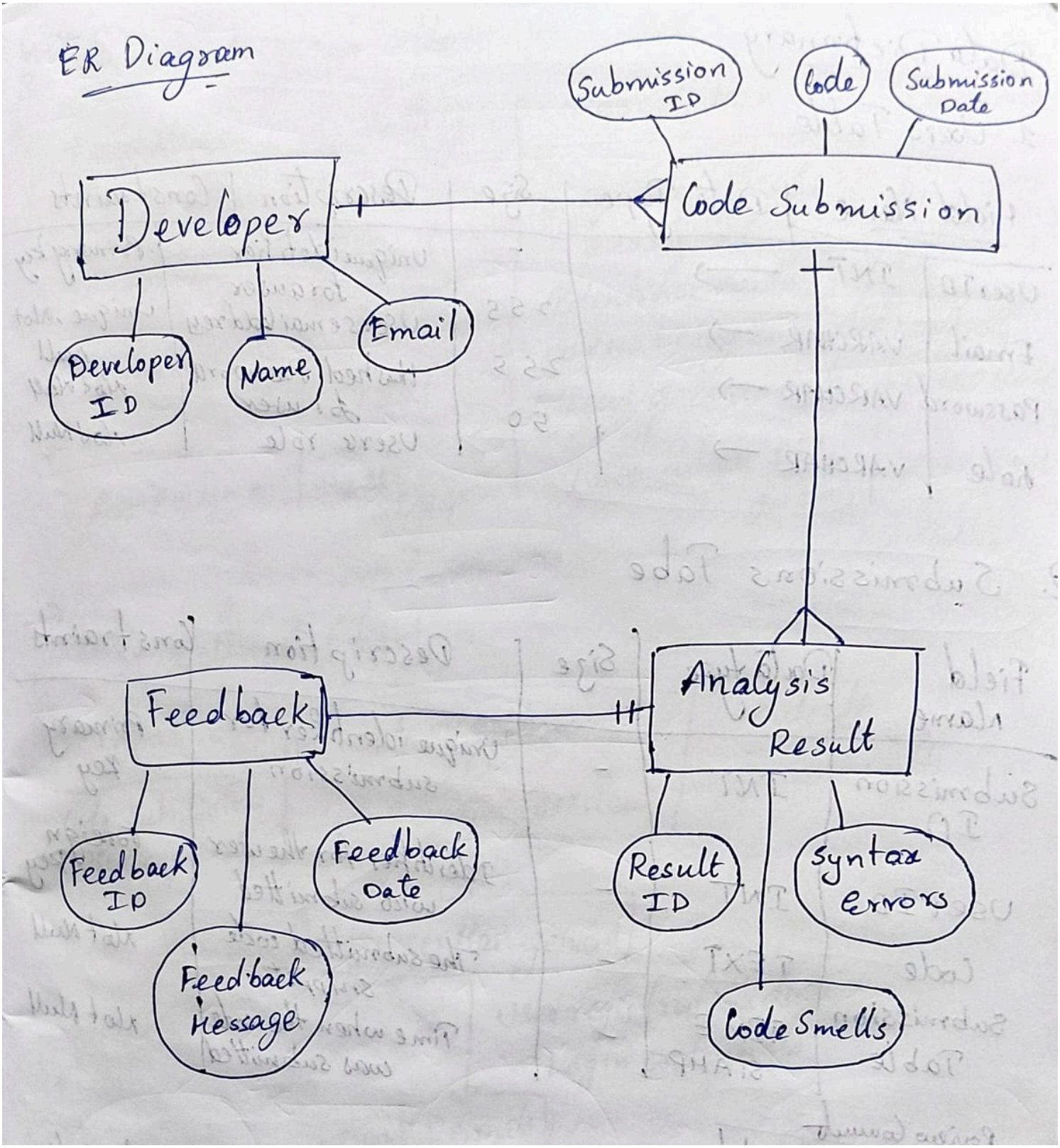
Level 0 DFD


Review Database
Data Management
Feedback
Analysis result
Code Pull
Code Push
Code analysis Engine
Feedback Generator
Code detail (unique)
Code Submission Interface
Generated Feedback
Submit Code
Developer
Level-1

Level 1 DFD


Level-2 DFD
Review Database
Retrieve feedback
Retrieve analysis result
Retrieve code submission
Data management
Store Feedback
Store analysis result
Store Code submission
feedback
Coder details
Analysis result
Generate feed back
Analysis
Code Smell Detection
Syntax analysis
Code
Semantic analysis
style analysis
Code Analysis Engine
Feedback generator
Recieve code submission
View Past Code analysis
Code Submission Interface
feed back
Developers
Submit Code
Submitted code, Analysis result, Feedback report

Level 2 DFD


ER Diagram
Submission ID
Code
Submission Date
Developer
Code Submission
Developer ID
Name
Email
Feedback
Analysis Result
Feedback ID
Feedback Message
Feedback Date
Result ID
Syntax Errors
Code Smells

ER DIAGRAM

Data Dictionary

1. Users Table

Field Name	Data Type	Size	Description	Constraints
UserID	INT	-	Unique identifier for a user	Primary key
Email	VARCHAR	255	User's email address	Unique, Not Null
Password	VARCHAR	255	Hashed Password for user	Not Null
Role	VARCHAR	50	User's role	Not Null

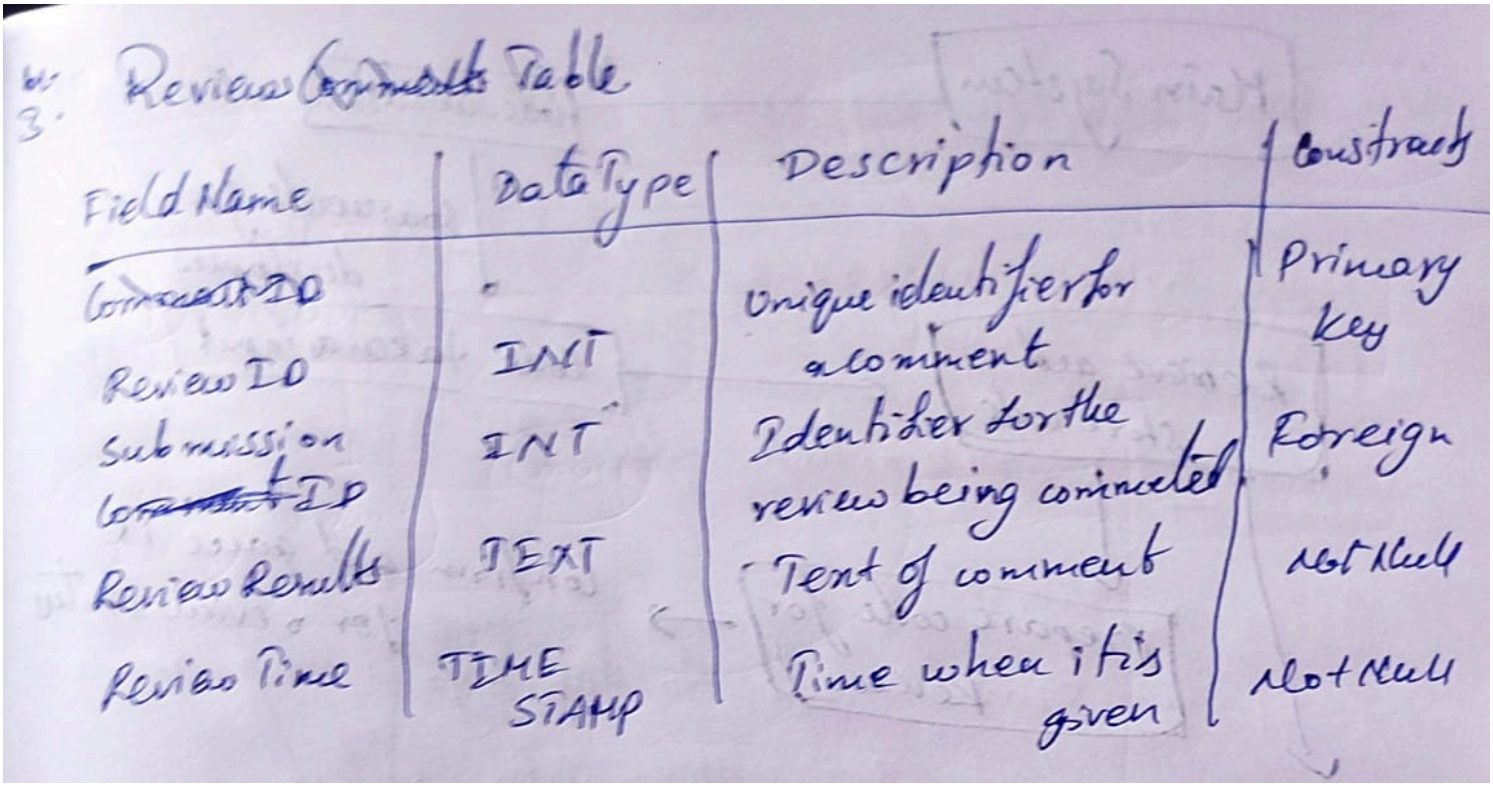
2. Submissions Tabe

Field Name	Data type	Size	Description	Constraints
Submission ID	INT	-	Unique identifier for submission	Primary Key
User ID	INT	-	Identifier for the user who submitted	Foreign Key
Code Submission	TEXT	-	The submitted code snippet	Not Null
Table	TIME STAMP	-	Time when the code was submitted	Not Null

Review Comment

3. ~~Reviews~~ Table

Field Name	Data type	Size	Description	Constraints
Comment ID	INT		Unique identifier for comment	Primary key
Review ID	INT		Identifier for review	Foreign key

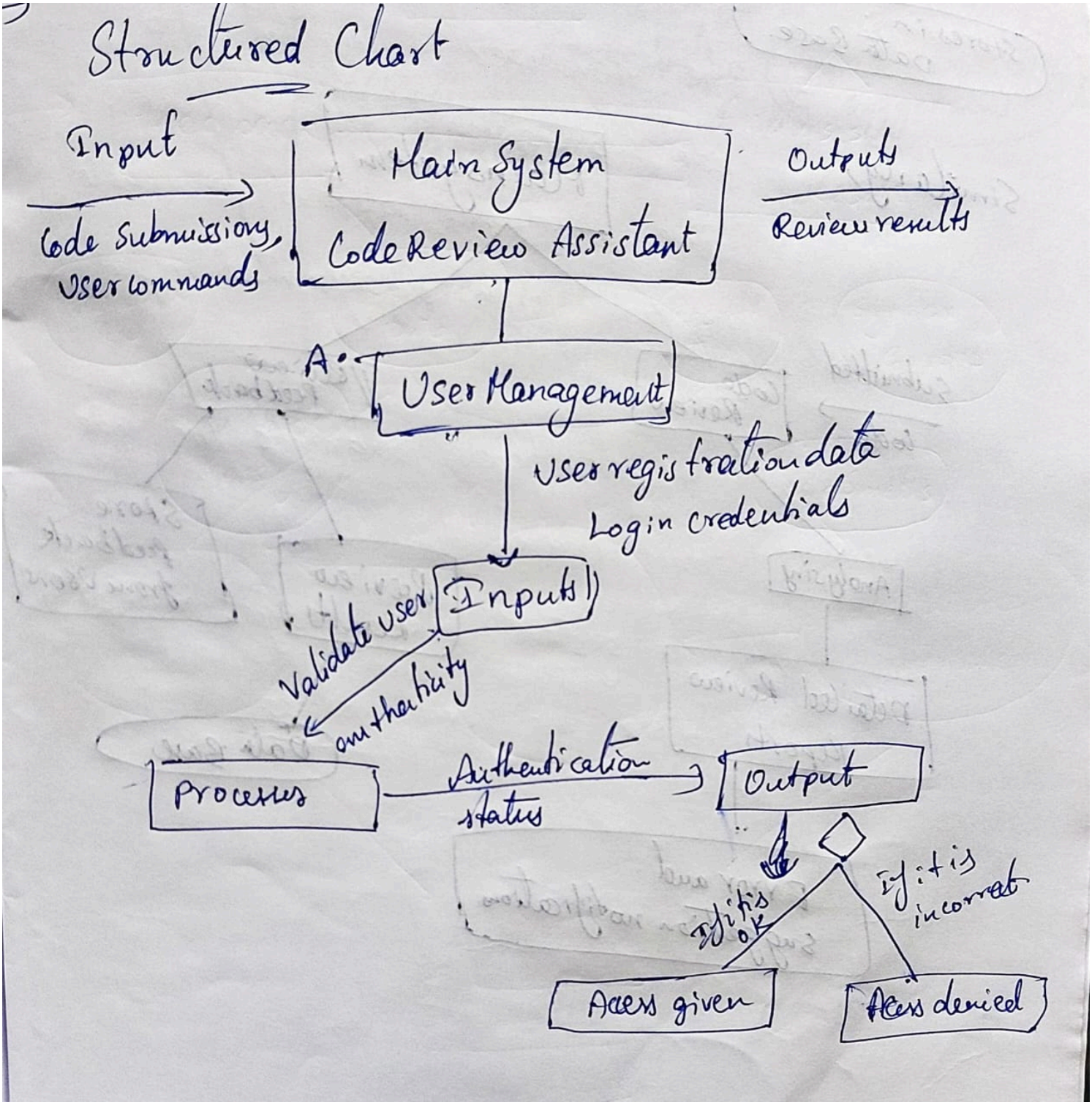


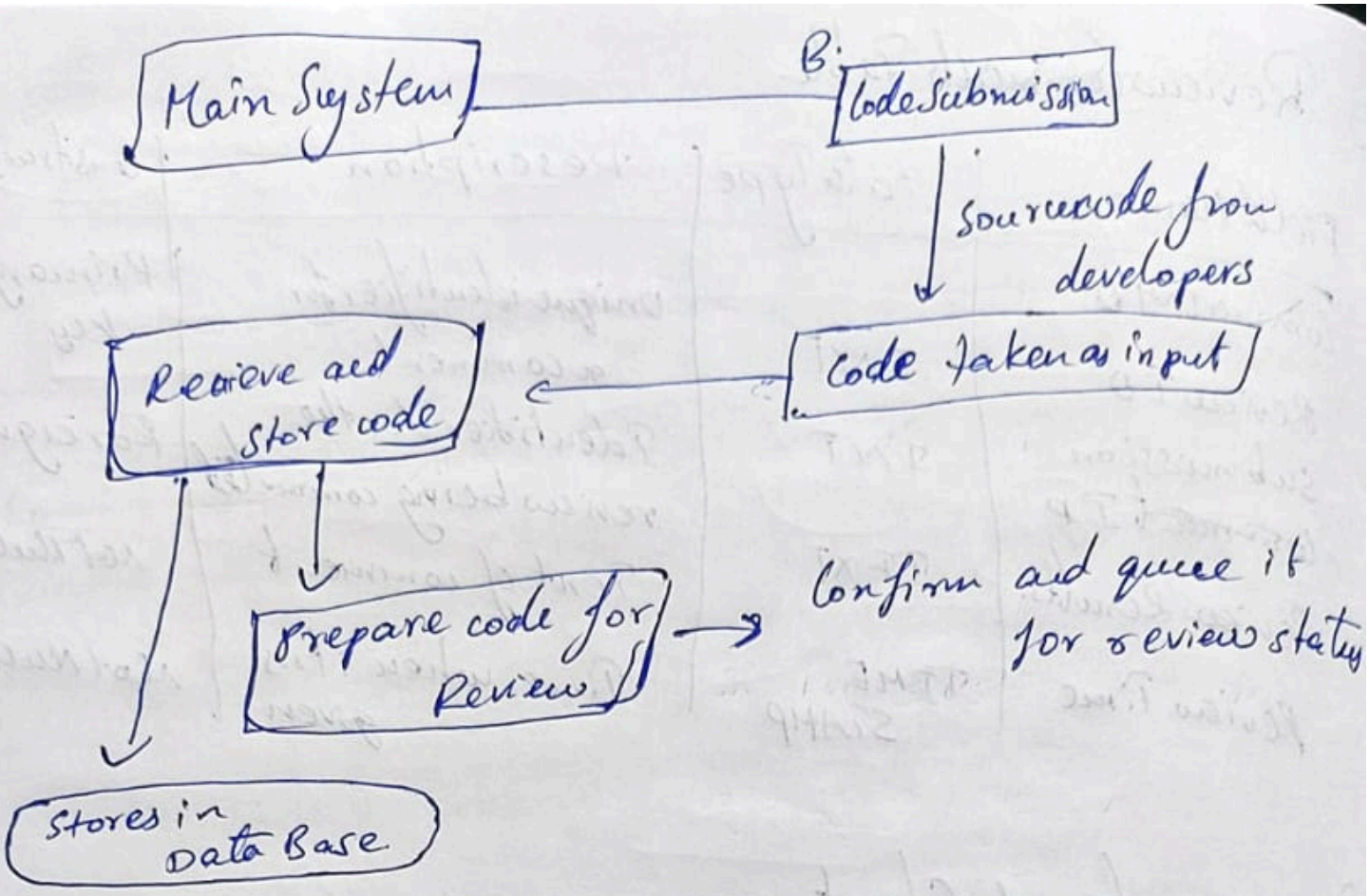
3. Reviews ~~Comments~~ Table

Field Name	Data Type	Description	Constraints
~~Comment ID~~ Review ID	INT	Unique identifier for a comment	Primary Key
Submission ~~Comment~~ ID	INT	Identifier for the review being commented	Foreign
Review Results	TEXT	Text of comment	Not Null
Review Time	TIME STAMP	Time when it is given	Not Null

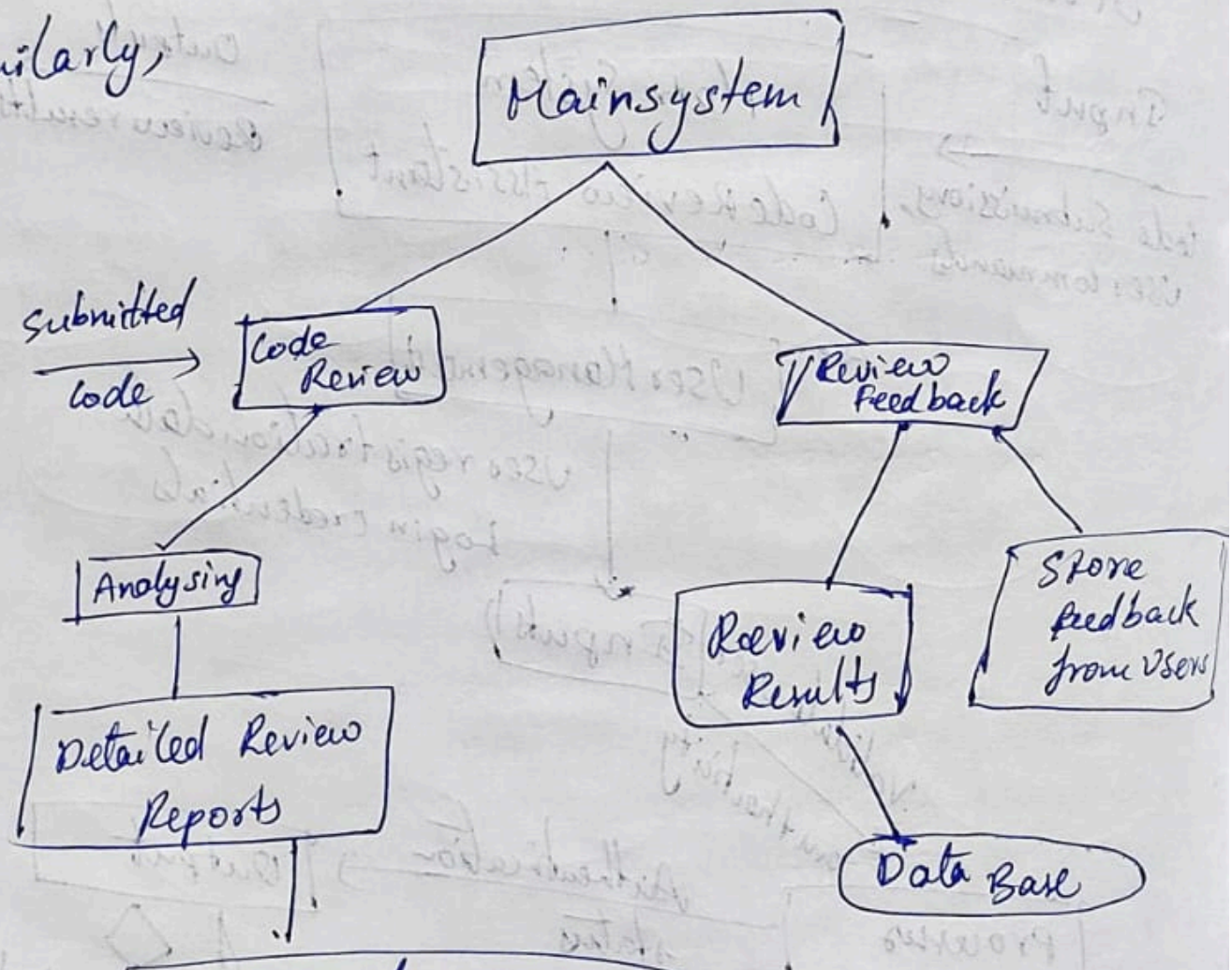
Data Dictionary

Structured Chart

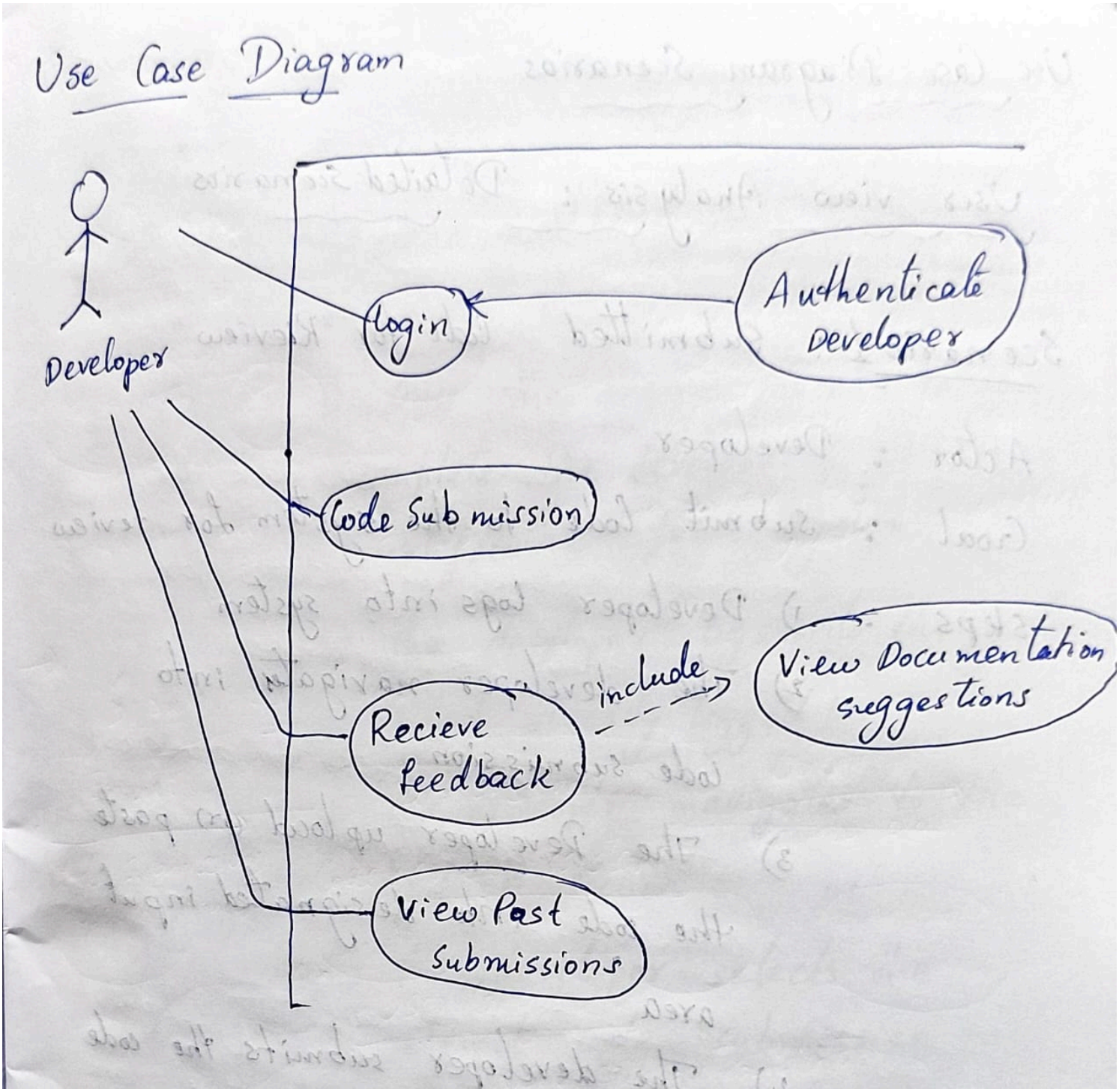



Main System
B.
Code Submission
Source code from developers
Code taken as input
Receive and store code
Prepare code for Review
Confirm and queue it for review status
Stores in Data Base

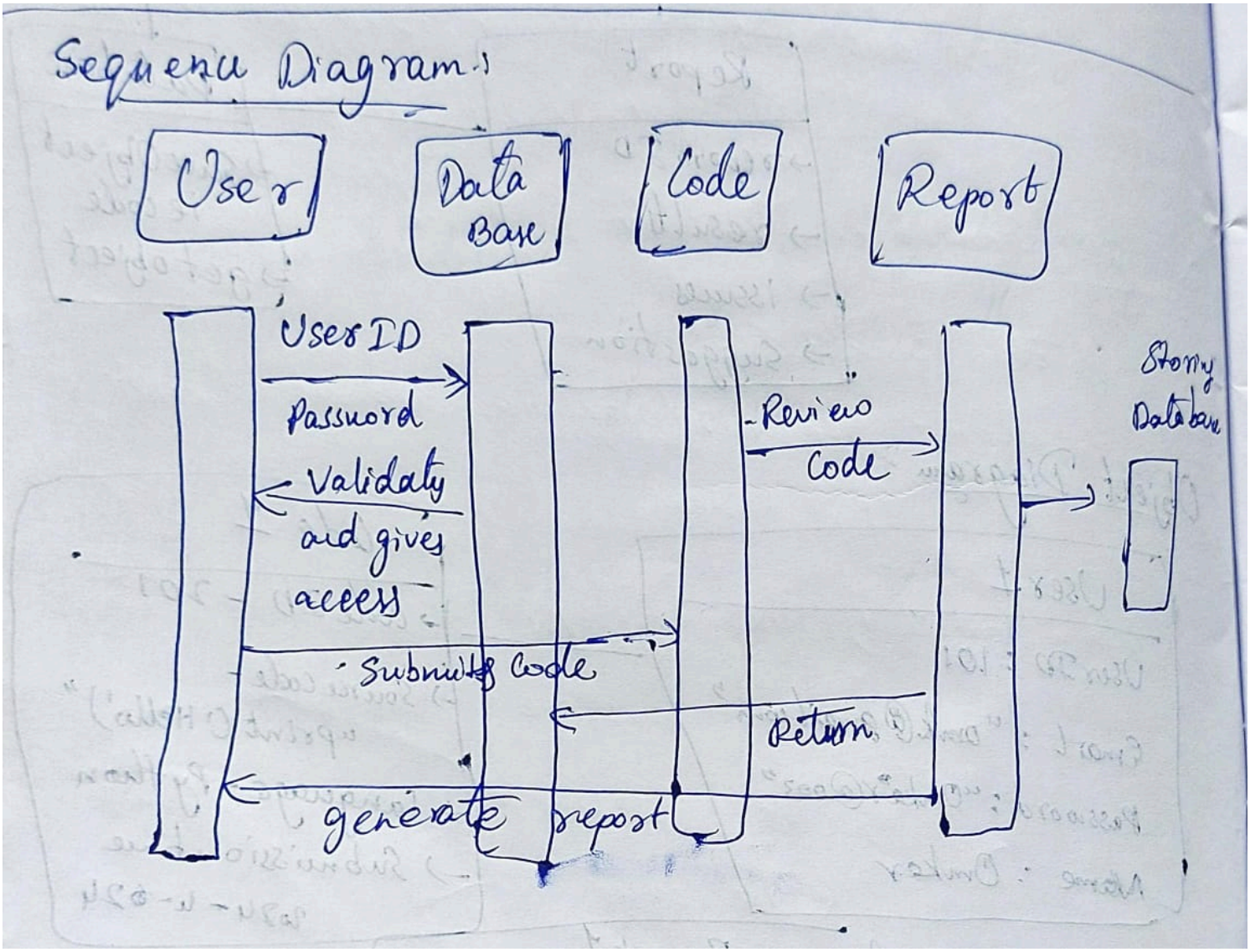
Similarly,


Mainsystem
Submitted code
Code Review
Review Feedback
Analysing
Review Results
Store feedback from users
Detailed Review Reports
Data Base

Structured chart



USE CASE DIAGRAM



Sequence diagram

USE CASE DIAGRAM SCENARIOS

Use Case Diagram Scenarios
User view Analysis: Detailed scenarios
scenario 1: Submitted code for Review.
Actor: Developer
Goal : submit Code to the system for review
steps: I Developer logs into system
3) The developer navigate into
Code submission.
3) The Developer upload or paste
the code into designated input
area
4) The developer submits the code

for Review
5) The system acknowledges reciepts
begin the review process.

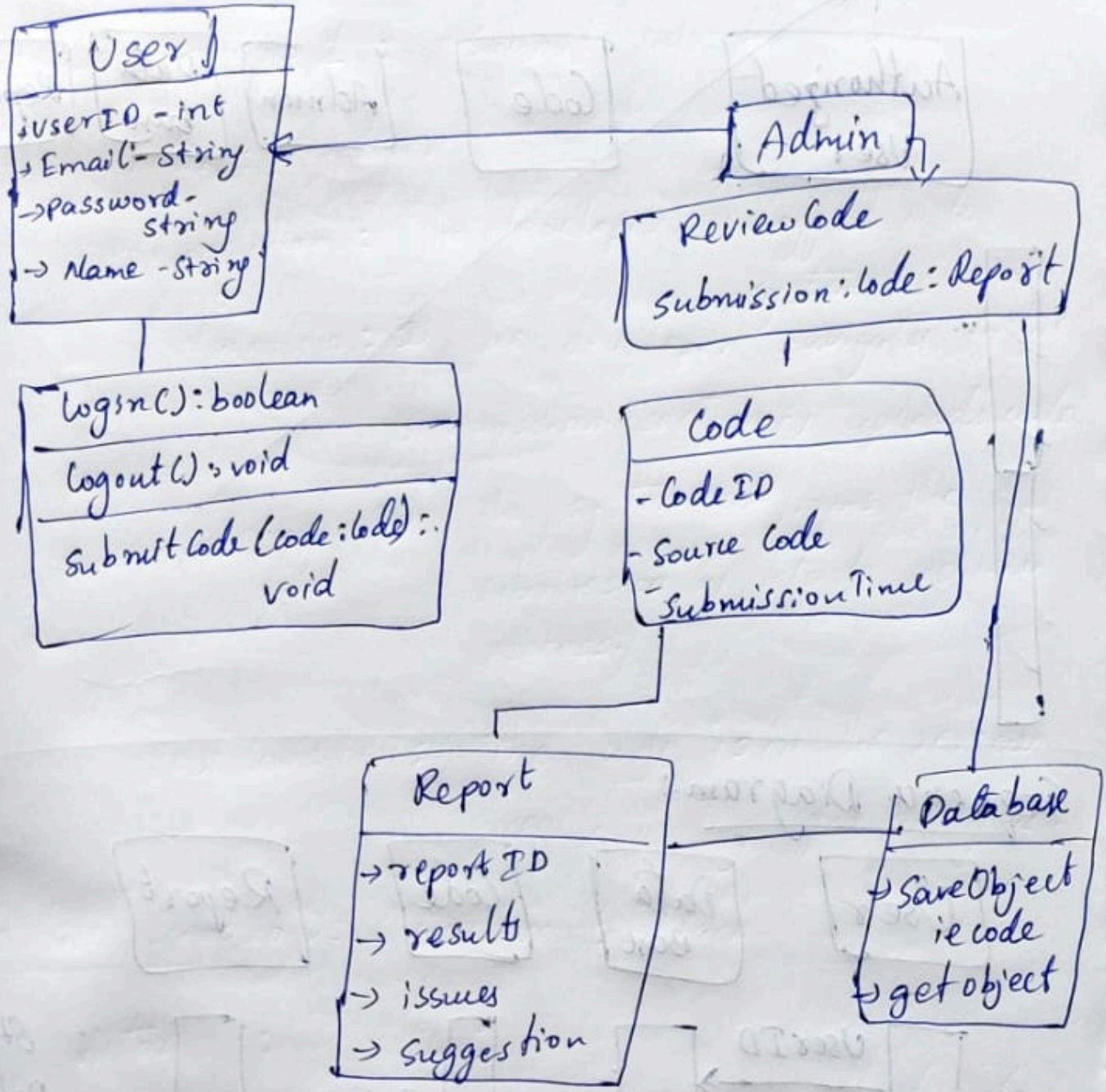
Scenario 2 : Recieving and Viewing feedback
Actor: Developer
Goal: Recieve feed back from system
based on the submitted code reviews.
steps: 1) Once the review process is
complete the developer - recieves a
notification from system
2) The developer logs into the system
if not already logged In.
3) The developer navigates to the
feedback section
4) The developer selects the
relevant code submission to
view detailed feedback
5) The feedback includes comments
on code quality. suggestions for
improvement and identified issues.

Scenario 3: viewing Documentation suggestions
Actor: Developer
Goal :view suggestions for improving or
adding to code documentation.
steps : 1)After recieving feedback, the
developer notices suggestions for
documentation improvements
2). The developer clicks on a link or
button to view detailed documentation
suggestions.
3) The system displays suggestions,
including areas of code that lack
documentation.
4) The developer can then apply them
suggestions to their code

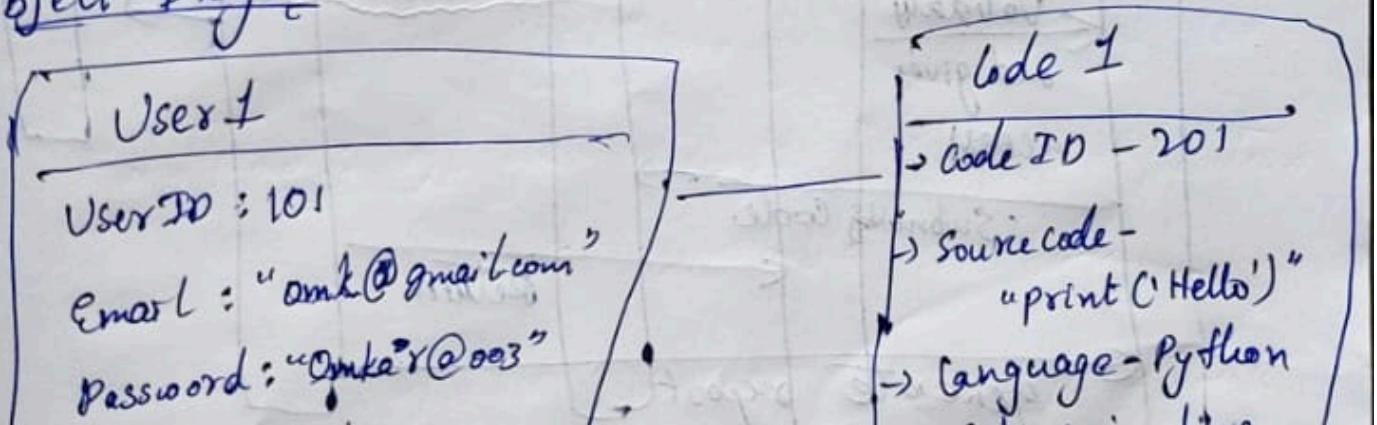
Use Case Diagram Scenarios
Use case scenario 1: Submit code for reviews
Actor: Developer
Precondition: The developer is logged into
the system main flow: The developer navigates to the
submission section and uploads code

2) The system confirms the
submission of start review
process.
Post Condition: The code is in the review
queue, waiting for analysis.
Use Case Scenario 2: Recieve feedback
Actor: Developer
Precondition :The developer has submitted code
for Reviews
Main flow: 1) The developer recieves notification
about available feedback
2) The developer views feed back
in the system.
Post Condition : The developer is informed about
the quality of submitted code of
possible improvements.
Use Case Scenario 3:
Actor : Developer.
Precondition : The developer has recieved
feedback that includes documentation suggestions
Main flow : 1) The developer accesses the
section for documentation suggesting
within the feed back.
2) The system displays specific suggestions for improving
documentation.
Post Condition: The development knows
how to improve the code
documentation.

Class Diagram



Object Diagram :



Class and object diagram

The code for our project can be viewed by

http://github.com/varshith2003/CodeReviewAssistant