A

PROJECT REPORT

ON

"FIX MY CODE"

Submitted to

Savitribai Phule Pune University

By

Mr. Shubham Sanjay Mali

Under the Guidance of

Ms. Kalpana Dhende

MES Institute of Management & Career Courses (IMCC), Pune

YEAR 2023-24

INDEX

1. INTRODUCTION

In the rapidly evolving landscape of software development, efficiency and collaboration are paramount. Developers often face challenges in maintaining clean, optimized, and error-free code. "FIX_MY_CODE" emerges as a solution, offering a smart code editor that not only facilitates code creation and editing but goes a step further by integrating artificial intelligence to assist developers in enhancing their code quality.

At its core, FIX_MY_CODE is a sophisticated web-based code editor that transcends traditional boundaries. It not only serves as a canvas for code creation and editing, but it transforms the coding experience by incorporating cutting-edge technologies.

Scope of the System:

1. **Code Editor:** Develop a web-based code editor using HTML, CSS, and JavaScript where users can paste their code.

2. **Code Operations**: Implement buttons that allow users to perform various code operations, such as debugging, fixing indentation, changing variable names to meaningful ones, and adding comments to the code or even optimizing their code.

3. **Code Transformation**: Utilize the OpenAI Codex API to perform code transformations based on the user's selections. The API will be responsible for making the actual code changes.

4. **User Authentication**: Provide two options for users: signing in and continuing without signing in. Implement user authentication using technologies like OAuth, Firebase, or a custom backend system.

5. **Code History**: If a user signs in, store the code history in a database (e.g., Firebase Realtime Database or MongoDB). Display this history in a "History" tab, allowing users to revisit their past code transformations.

2. Objectives:

1. To create a user-friendly web application that assists developers in optimizing and improving their code.

2. To integrate the OpenAI Codex API for code transformation.

3. To implement user authentication for a personalized experience.

4. To provide a code history feature for signed-in users to track their code improvements over time.

Proposed System:

The proposed system, envisioned within the FIX_MY_CODE project, represents a paradigm shift in the landscape of code editing and developer assistance. Crafted with precision and innovation, the proposed system aims to revolutionize the way developers interact with their code, offering a feature-rich, intelligent, and user-centric environment.

In summary, the proposed system within FIX_MY_CODE redefines the conventional code editing paradigm. It empowers developers with intelligent transformations, flexible authentication options, personalized experiences, and a rich technological stack, making it a cornerstone in the arsenal of tools for modern software development.

Study Of similar systems:

In our extensive study of similar systems within the domain of code editing, transformation tools, and developer assistance platforms, several key differences and challenges have been identified. These observations inform our approach in developing FIX_MY_CODE, as we aim to overcome specific limitations and introduce innovative solutions.

Observations:

- While conventional code editors excel in providing essential editing functionalities, they often lack advanced code transformation features.
- Some existing AI tools may lack transparency in their decision-making processes, making it difficult for developers to understand and trust the suggestions.
- Existing systems may offer user authentication but might not provide a flexible approach for users who prefer not to sign in.
- Personalization features, when present, may be limited in scope and customization.
- Existing smart code editors, particularly those offering advanced features, often adopt a monetization model that involves subscription fees or extensive advertisements. This can be disruptive to the user experience, leading to dissatisfaction among users seeking a streamlined and distraction-free coding environment.

Challenges Addressed by FIX_MY_CODE:

- Introducing an advanced code editor that combines the simplicity of lightweight editors with the intelligence of code transformation tools.
- Implementing a robust integration with the OpenAI Codex API to enhance the accuracy and intelligence of code transformations.
- Implementing a dual approach to user authentication, accommodating both signed-in and non-signed-in users for a tailored experience.
- Implementing a dual approach to user authentication, accommodating both signed-in and non-signed-in users for a tailored experience.
- Addressing the challenge of disruptive monetization models, FIX_MY_CODE aims to provide a clean and clutter-free user interface (UI). By eliminating excessive ads and subscription-based barriers, the platform ensures an optimal coding experience.

3. Analysis and Design:

3.1. Requirements Gathering:

The foundation of FIX_MY_CODE lies in a thorough understanding of user needs and industry standards. Through surveys, interviews, and user feedback, we've identified key requirements for the system:

• **Code Editing Features :** Users require a sophisticated code editor capable of handling multiple languages, with features such as syntax highlighting, auto-indentation, and intelligent code completion.

• **Advanced Code Transformations:** The desire for advanced code transformations using the OpenAI Codex API has been a recurring theme. Users seek functionalities such as code optimization, variable renaming, and intelligent bug fixing.

• **User Authentication :** A flexible user authentication system, allowing both signed-in and non-signed-in users, has been identified. OAuth, Firebase, or custom backend systems are preferred options.

• **Code History:** A desire to track and revisit code transformations over time for signed-in users has led to the incorporation of a Code History feature. This includes storing transformation details in databases like Firebase Realtime Database or MongoDB.

3.2. User Personas:

Developing detailed user personas has been crucial in shaping the user experience:

- Novice Developers: Require a user-friendly interface with guided code transformations to enhance learning and efficiency.
- Experienced Developers: Seek advanced code editing features and the ability to leverage AI for intricate code transformations.
- Occasional Coders: Prefer the option to use the platform without authentication for quick edits and transformations.

3.3. Risk Analysis:

Identified potential risks and devised mitigation strategies:

• External Service Dependency: Potential disruptions due to changes in external APIs are mitigated by monitoring API updates and maintaining fallback mechanisms.

• Security Risks: Regular security audits and continuous monitoring will be implemented to identify and address potential vulnerabilities.

Design:

1. System Architecture Design:

The architecture design reflects a clear separation of concerns:

• Frontend: Developed using React for dynamic and interactive user interfaces.

• Backend: Utilizes JavaScript for server-side logic and API integration.

• Database: Chooses Firebase Realtime Database for its real-time capabilities and ease of integration.

2. User Interface Design:

Clean UI: Prioritizes a clutter-free interface to enhance the user experience, removing distractions such as excessive ads.

• Responsive Design: Ensures a seamless experience across various devices and screen sizes.

3. Authentication System Design:

Specifies the user authentication flow for a secure and seamless experience:

• OAuth Integration: For users preferring a signed-in experience.

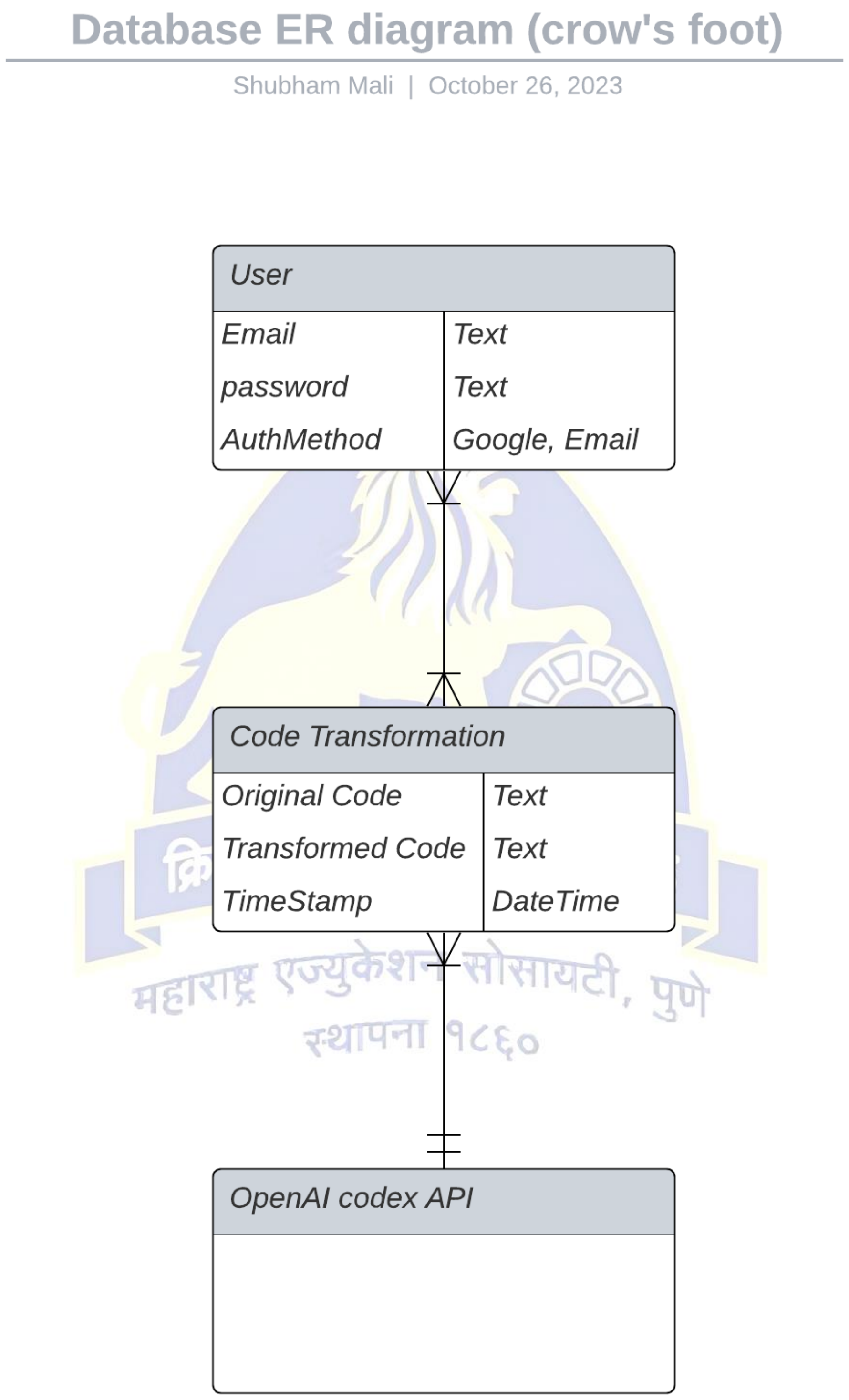
• Anonymous Authentication: Allows non-signed-in users to access basic functionalities.

4. Deployment Strategy:

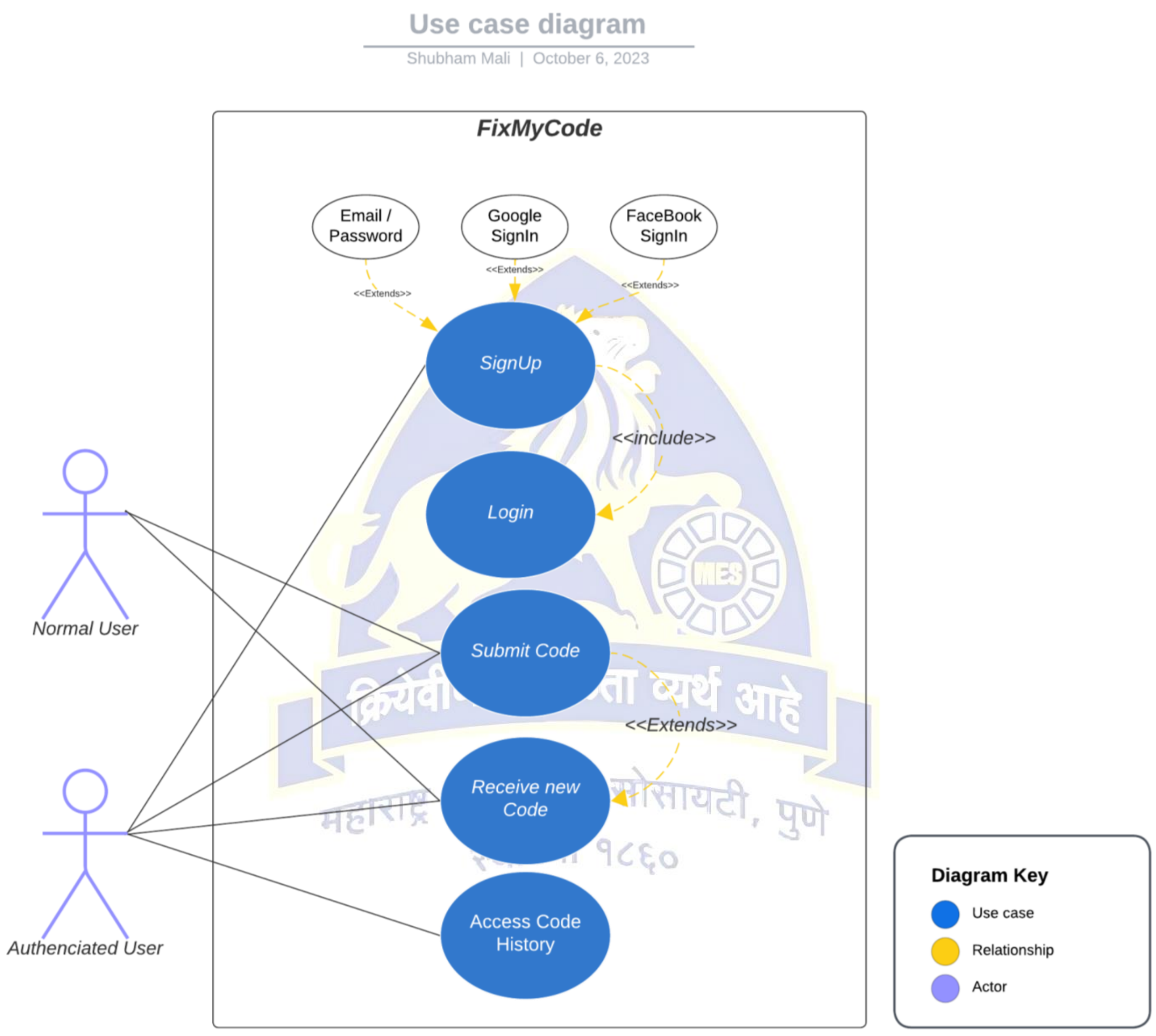
Outlines the plan for deploying FIX_MY_CODE:

• Hosting Platform: Considers options like Netlify, Vercel, or Heroku for their ease of use and scalability.

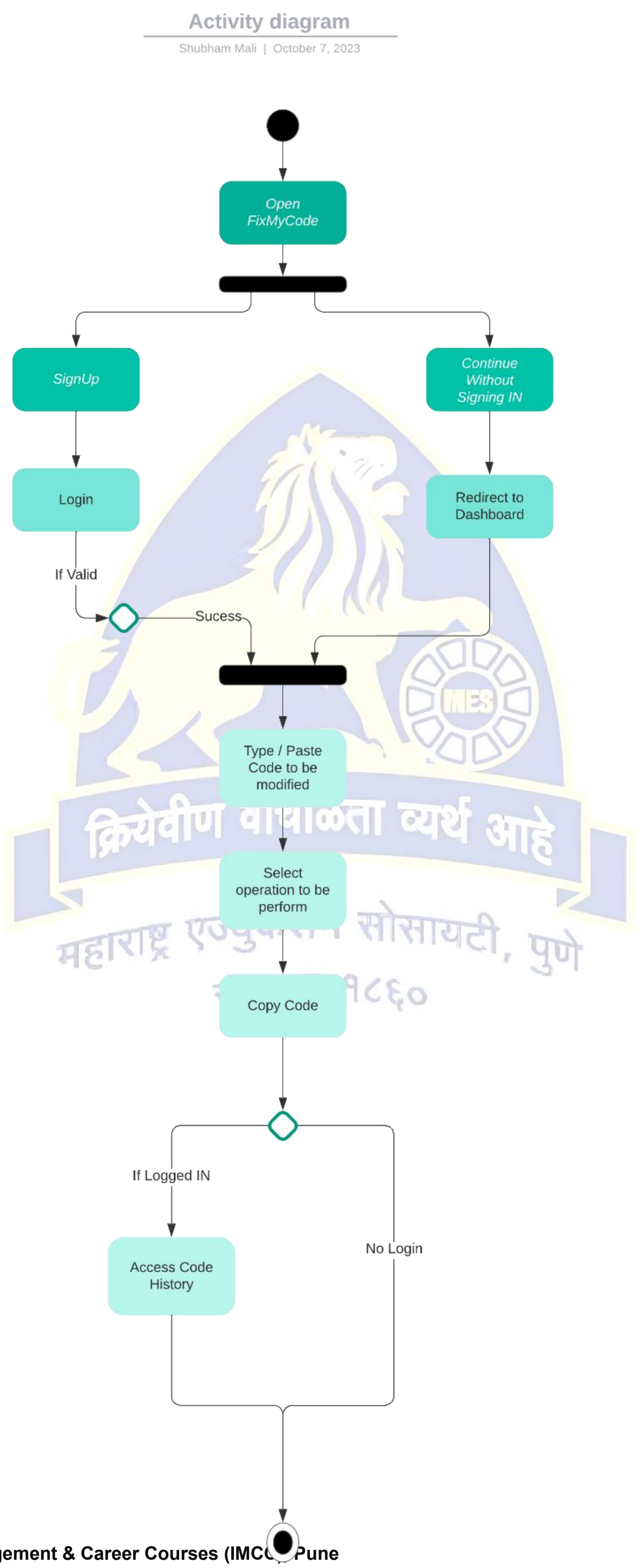
ERD:


Database ER diagram (crow's foot)
Shubham Mali | October 26, 2023
User
Email
Text
password
Text
AuthMethod
Google, Email
Code Transformation
Original Code
Text
Transformed Code
Text
TimeStamp
DateTime
OpenAI codex API

Use Case Diagram:

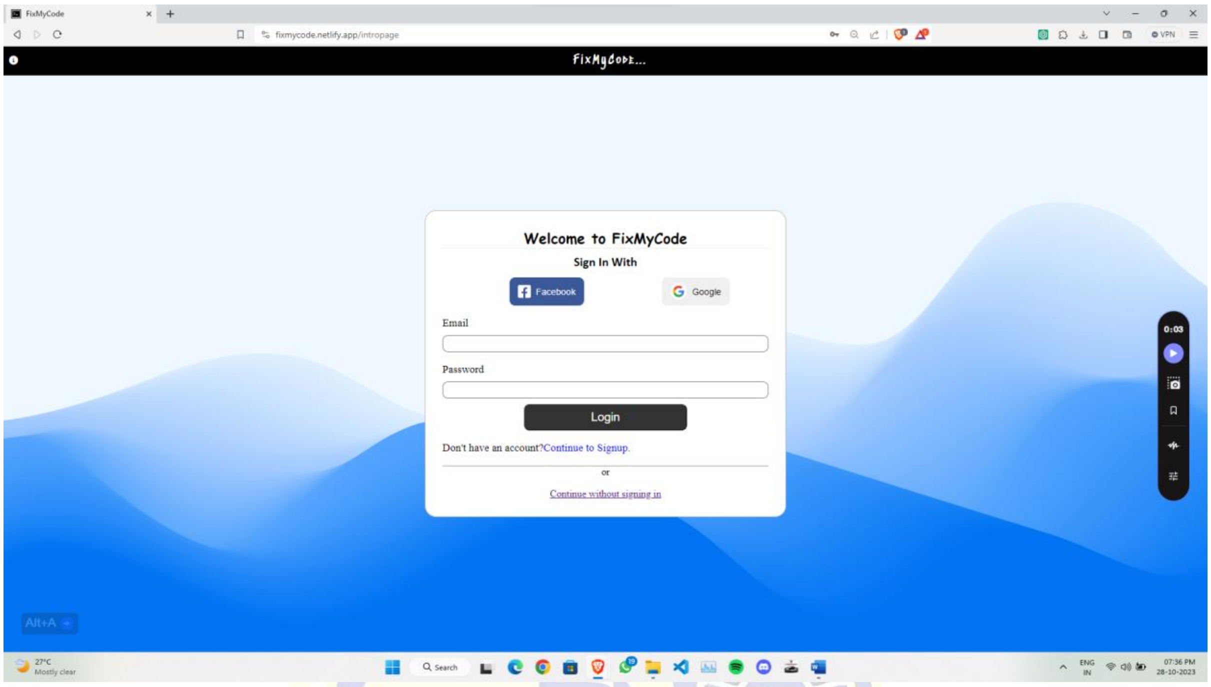


Activity Diagram:

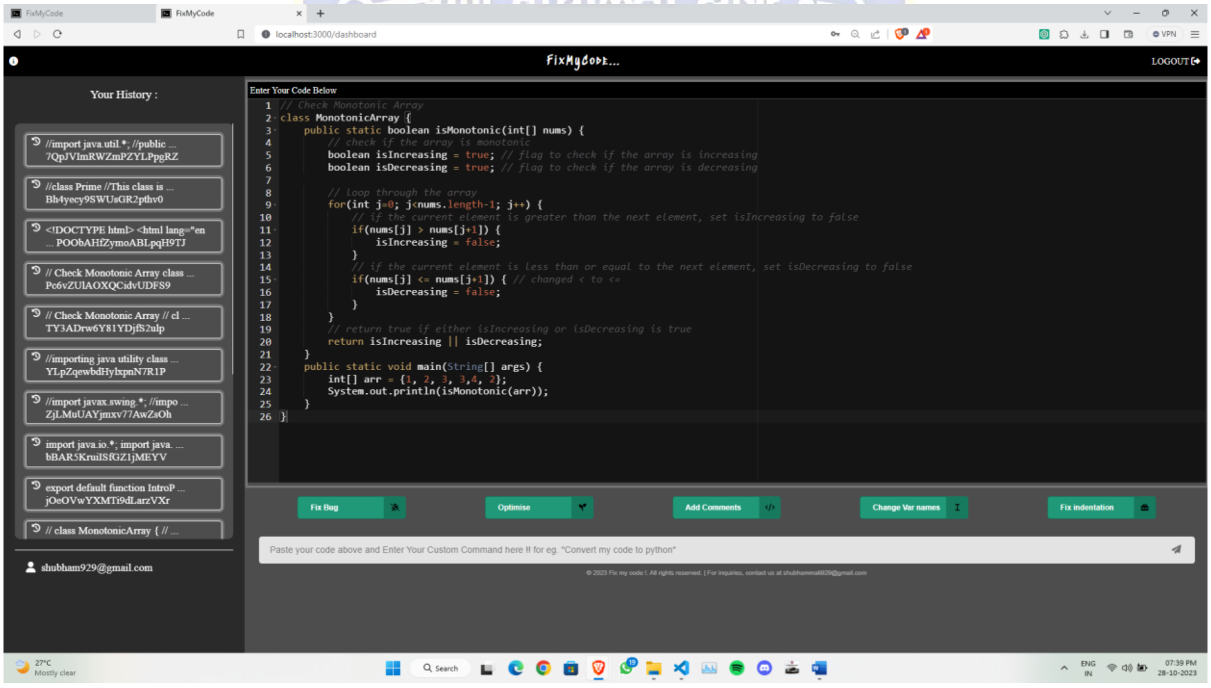


4. UI Screens and Database Design/Table Structure

Login/Signup page:


FixMyCode
fixmycode.netlify.app/intropage
FixMyCode...
Welcome to FixMyCode
Sign In With
Facebook
Google
Email
Password
Login
Don't have an account?Continue to Signup.
or
Continue without signing in

Dashboard



//import java.util.*; //public ... 7QpJVImRWZmPZYLPpgRZ
//class Prime //This class is ... Bh4yecy9SWUsGR2pthv0
<!DOCTYPE html> <html lang="en ... POObAHfZymoABLpqH9TJ
// Check Monotonic Array class ... Pc6vZUlAOXQCidvUDFS9
// Check Monotonic Array // cl ... TY3ADrw6Y81YDjfS2ulp
//importing java utility class ... YLpZqewbdHylxpnN7R1P
//import javax.swing.*; //impo ... ZjLMuUAYjmxv77AwZsOh
import java.io.*; import java. ... bBAR5KruilSfGZ1jMEYV
export default function IntroP ... jOeOVwYXMTi9dLarzVXr
// class MonotonicArray { // ...

shubham929@gmail.com

```java
// Check Monotonic Array
class MonotonicArray {
    public static boolean isMonotonic(int[] nums) {
        // check if the array is monotonic
        boolean isIncreasing = true; // flag to check if the array is increasing
        boolean isDecreasing = true; // flag to check if the array is decreasing

        // loop through the array
        for(int j=0; j<nums.length-1; j++) {
            // if the current element is greater than the next element, set isIncreasing to false
            if(nums[j] > nums[j+1]) {
                isIncreasing = false;
            }
            // if the current element is less than or equal to the next element, set isDecreasing to false
            if(nums[j] <= nums[j+1]) { // changed < to <=
                isDecreasing = false;
            }
        }
        // return true if either isIncreasing or isDecreasing is true
        return isIncreasing || isDecreasing;
    }
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 3,4, 2};
        System.out.println(isMonotonic(arr));
    }
}
```

Paste your code above and Enter Your Custom Command here !! for eg. "Convert my code to python"

© 2023 Fix my code ! All rights reserved. | For inquiries, contact us at shubhammai929@gmail.com

Database Design :

Collections

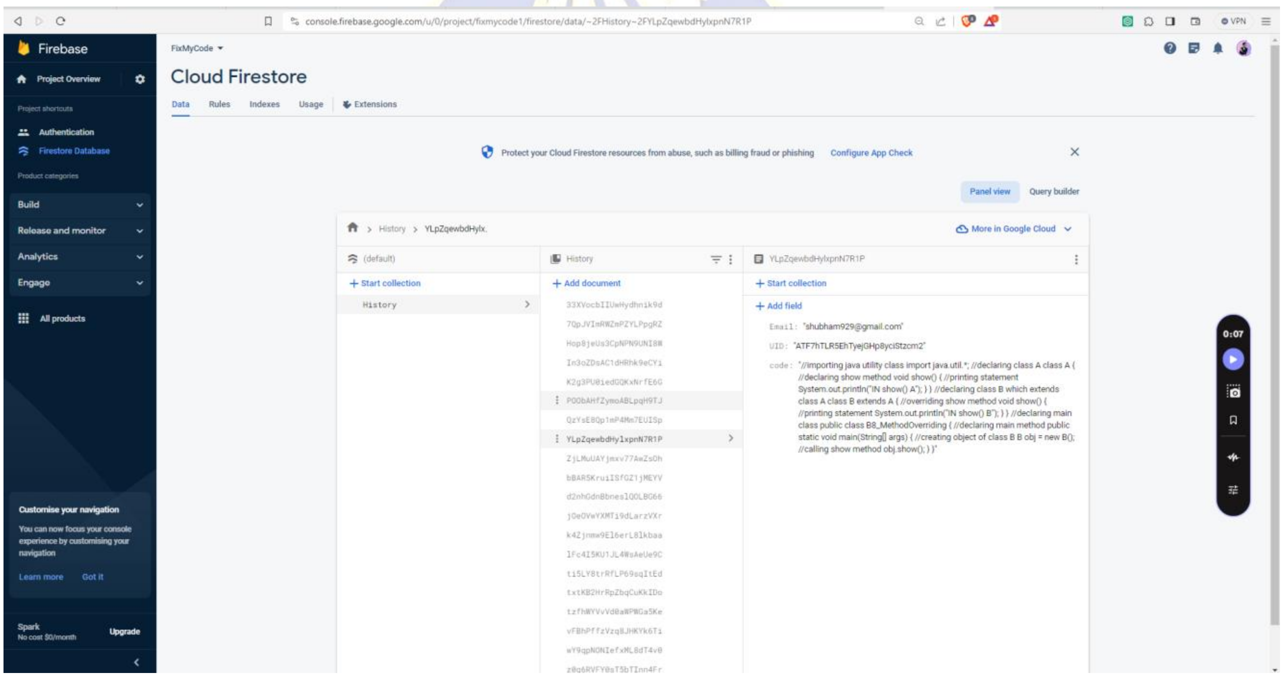
History :

UID (string): User ID (foreign key referencing Users collection)

Email (string): User's email (denormalized for quick retrieval)

code (string): The code snippet submitted by the user.

timestamp (timestamp): Timestamp of when the code was submitted.



History

```
33XVocbIIUwHydhnik9d
7QpJVImRWZmPZYLPpgRZ
Hop8jeUs3CpNPN9UNI8W
In3oZDsAC1dHRhk9eCYi
K2g3PU0iedGQKxNrfE6G
POObAHfZymoABLpqH9TJ
QzYsE8Qp1mP4Mm7EUISp
YLpZqewbdHy1xpnN7R1P
ZjLMuUAYjmxv77AwZsOh
bBAR5KruiISfGZ1jMEYV
d2nhGdnBbnes1QOLBG66
jOeOVwYXMTi9dLarzVXr
k4Zjnmw9El6erL8lkbaa
lFc4I5KU1JL4WsAeUe9C
ti5LY8trRfLP69sqItEd
txtKB2HrRpZbqCuKkIDo
tzfhWYVvVd0aWPWGa5Ke
vFBhPffzVzq8JHKYk6Ti
wY9qpNONIefxML8dT4v0
z0q6RVFY0sT5bTInn4Fr
```

YLpZqewbdHylxpnN7R1P

```
Email: "shubham929@gmail.com"
UID: "ATF7hTLR5EhTyejGHp8yciStzcm2"
code: "//importing java utility class import java.util.*; //declaring class A class A { //declaring show method void show() { //printing statement System.out.println("IN show() A"); } } //declaring class B which extends class A class B extends A { //overriding show method void show() { //printing statement System.out.println("IN show() B"); } } //declaring main class public class B8_MethodOverriding { //declaring main method public static void main(String[] args) { //creating object of class B B obj = new B(); //calling show method obj.show(); } }"
```

5. Test Design/Test Cases

User Authentication:

Test Case: User Registration

Verify that a user can successfully register with a valid email and password.

Ensure the user is redirected to the correct page after registration.

Test Case: User Login

Confirm that a registered user can log in with valid credentials.

Ensure the user is redirected to the correct page after login.

Test Case: Invalid Login

Attempt to log in with an invalid email or password.

Verify that the system shows an appropriate error message.

Test Case: User Logout

Confirm that a logged-in user can successfully log out.

Ensure the user is redirected to the correct page after logout.

Code Submission and Processing:

Test Case: Code Submission

Enter a code snippet in the editor and submit it.

Confirm that the code is successfully submitted and processed.

Test Case: API Response Handling

Simulate an API response with a mock response.

Verify that the application correctly handles and displays the response.

History and User Data:

Test Case: View User History

Log in as a user with a known history.

Verify that the user's code history is correctly displayed.

Security and Permissions:

Test Case: Unauthorized Access

Attempt to access a user's data without proper authentication.

Ensure the system denies access and displays an error message.

Usability and UI:

Test Case: Responsive Design

Test the application on various devices and screen sizes.

Confirm that the user interface is responsive and displays correctly.

Test Case: Code Editor Functionality

Verify that the code editor allows users to write, edit, and format code correctly.

Error Handling:

Test Case: Error Scenarios

Intentionally induce errors (e.g., network failure, API unavailability) and verify that the application provides appropriate error messages.

Performance:

Test Case: Code Submission Performance

Test the application's performance when handling large or complex code submissions.

Cross-Browser Testing:

Test Case: Browser Compatibility

Test the application on different browsers (e.g., Chrome, Firefox, Safari) to ensure compatibility.

Accessibility:

Test Case: Accessibility

Use accessibility testing tools to ensure the application is accessible to users with disabilities.