IDE(integrated development environment)

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

MASTAR OF COMPUTER APPLICATION
(1'st Semester)

SUBMITTED BY:
Karan Kumar Mishra

SUBMITTED TO:
Ms. Nisha Chaudhary

(Assistant Professor)
Avviare Educational Hub, Noida (U.P)

AVVIARE EDUCATIONAL HUB, NOIDA
(GLOCAL UNIVERSITY, SAHARANPUR, UTTAR PRADESH)

ACKNOWLEDGEMENT

I/WE are highly grateful to the Ms. Kanika Singh, Director of Operations, Avviare Educational Hub, Noida for providing this opportunity to carry out the major project work.

The constant guidance and encouragement received from Mr. Ashutosh Rathore H.O.D. IT Department, Avviare Educational Hub, Noida has been of great help in carrying out the project work and is acknowledged with reverential thanks.

I/WE would like to express a deep sense of gratitude and thanks profusely to Ms. Nisha Choudhary, without her wise counsel and able guidance, it would have been impossible to complete the project in this manner.

I/WE express gratitude to other faculty members of IT department of Avviare Educational Hub, Noida for their intellectual support throughout the course of this work.

Finally, I/WE are indebted to all whosoever have contributed in this report work.

Names

Date

TABLE OF CONTENTS(SAMPLE)

Contents	Page No.

Note: The report of respective project should be as per prescribed format and in the same order though if some of the points are not applicable in regard with the concerned project, they might be omitted.

Format and Specification details:

Format for Major Project Report

Title page

- Abstract Acknowledgement
- List of Figures
- List of Tables
- Table of Contents

Chapter 1 Introduction

1. Introduction to Project
2. Project Category(Internet based, Application or System Development,
3. Research based ,Industry Automation, Network or System Administration)
4. Objectives
5. Problem Formulation
6. Identification/Reorganization of Need
7. Existing System
8. Proposed System
9. Unique Features of the System

Chapter 2. Requirement Analysis and System Specification

2.1 Feasibility study (Technical, Economical, Operational)
2.2 Software Requirement Specification Document which must include the following:
(Data Requirement, Functional Requirement, Performance Requirement ,Dependability Requirement, Maintainability requirement, Security requirement, Look and feel requirement)
2.3 Validation
2.4 Expected hurdles
2.5 SDLC model to be used

Chapter 3. System Design

3.1 Design Approach (Function oriented or Object oriented)
3.2 Detail Design
3.3 System Design using various Structured analysis and design tools such as : DFD's,Data Dictionary,Structured charts,Flowcharts or UML
3.4 User Interface Design
3.5 Database Design
3.5.1ER Diagrams

INTRODUCTION TO PROJECT

In the realm of software development, the command-line interface (CLI)serves as a fundamental tool for interacting with computer systems and executing tasks efficiently. The project at hand, titled "IDE" is a C++ based CLI application designed to [briefly describe the main purpose or functionality of your project]. This report aims to elucidate the development,features, and significance of the project, providing an in-depth exploration of its architecture, functionality, and the underlying principles that govern its operation.

Project Category(Internet based, Application or System Development

It's great to hear that I have created an IDE for code run and compile with lots of features in C++. An IDE (Integrated Development Environment) is a software application that provides comprehensive facilities to computer programmers for software development. This IDE operates in CLI mode, offering a highly user-friendly experience. Unlike traditional command-line interfaces such as Linux or CMD, users do not need to manually input commands for specific tasks. Instead, the IDE presents a menu of options, allowing users to easily choose the desired action, such as opening a file or compiling code.

Research based ,Industry Automation, Network or System Administration)

Objectives

The objective of this project is to introduce an
IDE in CLI mode that is user-friendly.
I have incorporated numerous features into this IDE.

Features:

1. Create a new file
2. Open existing files
3. Update files
4. Delete files
5. Open files in external software such as Notepad and VSCode
6. Run and compile code written in
various languages (e.g., C, C++, Java, Python, Node.js, etc.)
7. Customize text color, background color, and other display settings

PROBLEM FORMULATION

In the development of this project, we have encountered numerous challenges. Creating a CLI-mode Integrated Development Environment (IDE) for this type of software has proven to be quite difficult. To enhance user-friendliness, we have implemented several features.For instance, when a user opens a new file or accesses an existing one,a new tab is opened for that task. Additionally, for user convenience, we have incorporated a feature to seamlessly open and edit files in other software such as Notepad and VSCode.

Unique Features of the System

1.create a new file

2.open file

3.update file

4.delete file

5.open file in other software like notepad and VScode

6.run and compile the code of any lanuage like c,c++,java,python,nodejs etc.

7.change the text color and background color etc.

Chapter 2. Requirement Analysis and System Specification

In this chapter, we delve into the critical aspects of requirement analysis and system specifications for "IDE" To ensure smooth performance and accessibility on a wide range of Windows system, the following specifications have been identified:

1. **Windows OS Compatibility:** IDE is designed to run on Windows OS, with compatibility extending from older versions to the latest. It ensures that users with varying Windows system can enjoy the IDE.

2. **Minimum Hardware Requirements:**
   RAM: A minimum of 1GB RAM is required to ensure smooth IDEplay and responsiveness.
   Internal Memory: The IDE demands approximately 127kb of space in the internal memory of the Windows system for installation and operation.

These specifications are critical to accommodate a broad user base, making IDE accessible to a wide range of Windows system owners. The chapter delves deeper into the technical aspects, ensuring that the system meets the necessary requirements for optimal performance.

Feasibility study (Technical, Economical, Operational)

A comprehensive feasibility study was conducted to evaluate the development and implementation of IDE from three critical perspectives:
Technical, Economical, and Operational.

Technical Feasibility:

- *Project Complexity:* Evaluating the technical feasibility was essential, considering the complexities of incorporating developmenttechnology into
the IDE, ensuring compatibility with a wide range of Windows system.

Economical Feasibility:

- *Cost-Benefit Analysis:* The economical feasibility study focused on the cost-benefit aspect of the project.

Operational Feasibility:

- *Practical Implementation:* The operational feasibility study examined the practicality of implementing the system. This included the smooth transition between offline and online modes, ensuring data security and privacy, and delivering a seamless user experience.

These feasibility studies collectively provided a holistic view of the project, affirming its technical viability, economic feasibility, and practical operation. This guided the development of IDE and provided a solid
foundation for addressing challenges and ensuring the project's success.

1. Data Requirement
2. Functional Requirement,
3. Performance Requirement

Data Requirement:

In the context of the IDE project report, data requirements are crucial for the IDE's operation. The database for IDE is a critical data requirement, and data security and privacy .

Functional Requirement:

Functional requirements, as detailed in the IDE project report, describe the core IDE functions. Functional requirements specify how the IDE should respond to user interactions and input.

Performance Requirement:

Performance requirements, as outlined in the IDE project report, establish the criteria for the system's responsiveness, processing speed, and resource utilization. They also set benchmarks for system reliability, availability, and scalability to ensure a smooth devlopment experience for devloper.

1. Dependability Requirement
2. Maintainability requirement
3. Security requirement
4. Look and feel requirement

Dependability Requirement:

For IDE dependability requirements are essential to ensure the IDE's reliability. This means that the IDE operates consistently without crashes or disruptions. Measures are in place to prevent system failures and data loss, as reliability is paramount for user satisfaction. Dependability also extends to ensuring the IDE's functionality under varying conditions and maintaining a stable user experience.

Maintainability Requirement:

Maintainability requirements are crucial for IDE as they address the ease of updating, enhancing, and extending the IDE in the future. This includes providing clear documentation, employing a modular code structure, and establishing a user-friendly development environment to facilitate future modifications and expansions. Maintainability ensures that IDE remains adaptable to technological advancements and devloper preferences over time.

Security Requirement:

Security requirements in the context of IDE focus on protecting user data and ensuring the integrity of the IDE. This encompasses encryption for data transmission, secure storage of user information, and protection against hacking or unauthorized access. Security requirements are vital to safeguarding developer privacy and maintaining trust in the IDE.

Look and Feel Requirement:

Look and feel requirements address the user interface design and user experience aspects in IDE It specifies the visual appeal, layout, and interactivity to provide an engaging and enjoyable IDE environment. A consistent and appealing look and feel contribute to user satisfaction and retention, enhancing the overall development experience.

1. Validation
2. Expected hurdles

AI-Powered Solutions in Developing IDE IDE:

In the development of IDE AI develop a pivotal role in overcoming significant hurdles and enhancing the validation process. Here's how AI contributed:

- *developmentComplexity:* AI-driven database curation and algorithm optimization simplified the complex task of development of software, ensuring precise code identification.

- *Balancing Entertainment and Learning:* AI-powered insights and iterations helped strike the delicate balance between an engaging IDE and an effective educational tool.

- *Security Measures:* AI tools contributed to the robust implementation of security measures, safeguarding user information and IDE integrity.

- *Future Expansion Challenges:* AI-driven analytics provided valuable insights for planning future expansions, including multideveloper features and code database growth.

3. SDLC model to be used

*Software development life cycle for IDE

1. Planning: Initial concept and objectives.

2. Analysis:Feasibility studies and requirements.

3. Design: System architecture and user interface.

4. Development: Coding and language adoption.

5. Testing: Ensuring reliability and functionality.

6. Deployment: Release on Windows platforms.

7. Maintenance:Ongoing updates and adaptability.

These steps ensured a structured approach to the IDE's development.

CHAPTER 3
SYSTEM DESIGN

-Design Approach

In developing IDE for my MCA first sem college project, I employed an object-oriented design approach using c++.
This methodology offered several advantages:

- **Modularity and Reusability:** The object-oriented approach organized the code into manageable modules, enhancing reusability for future development.

- **Encapsulation:** c++ object-oriented design encapsulated data and functions within classes, ensuring data integrity and code reliability.

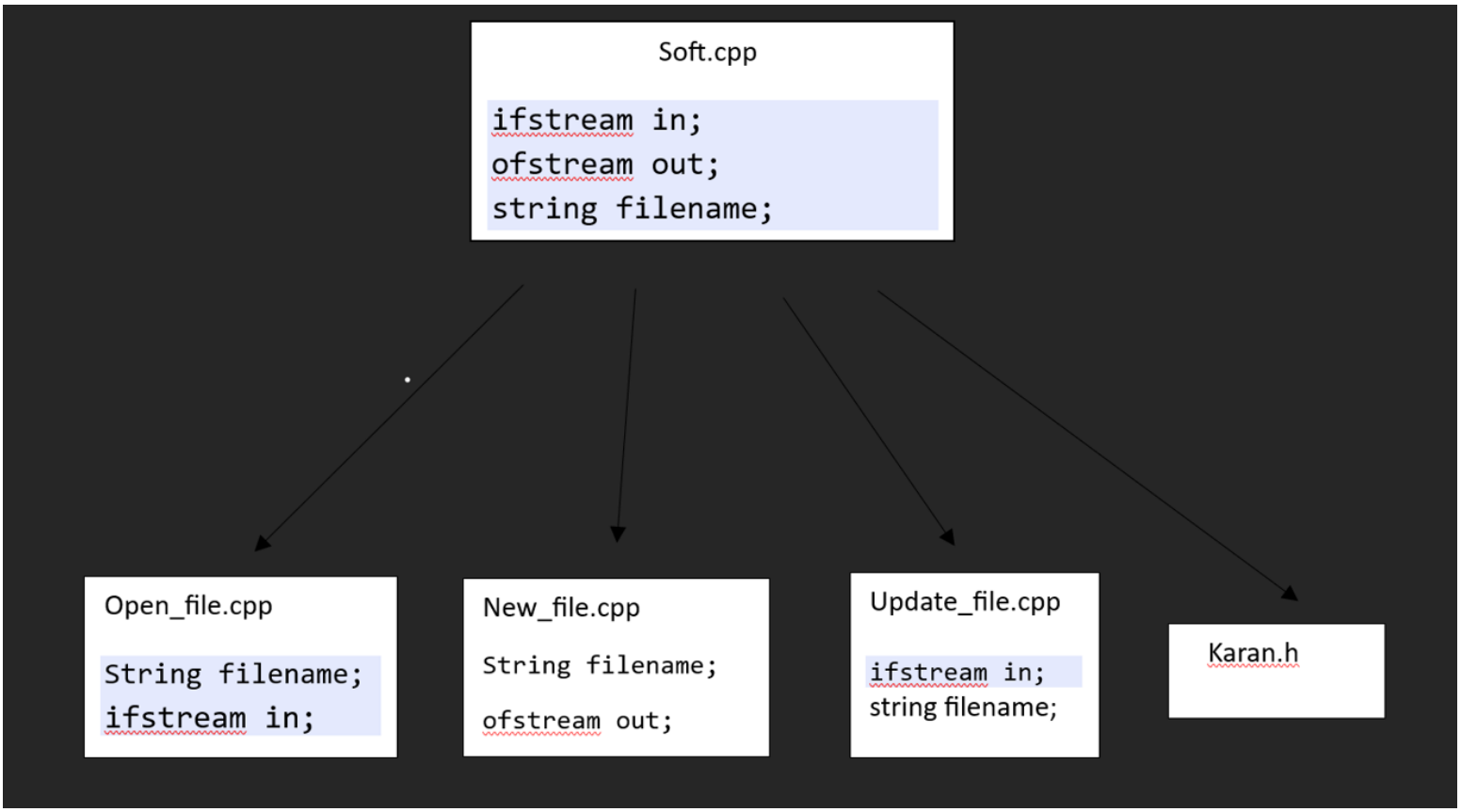
- **Inheritance:** Utilizing inheritance, I created a hierarchy of IDE elements, reducing redundancy in the code.

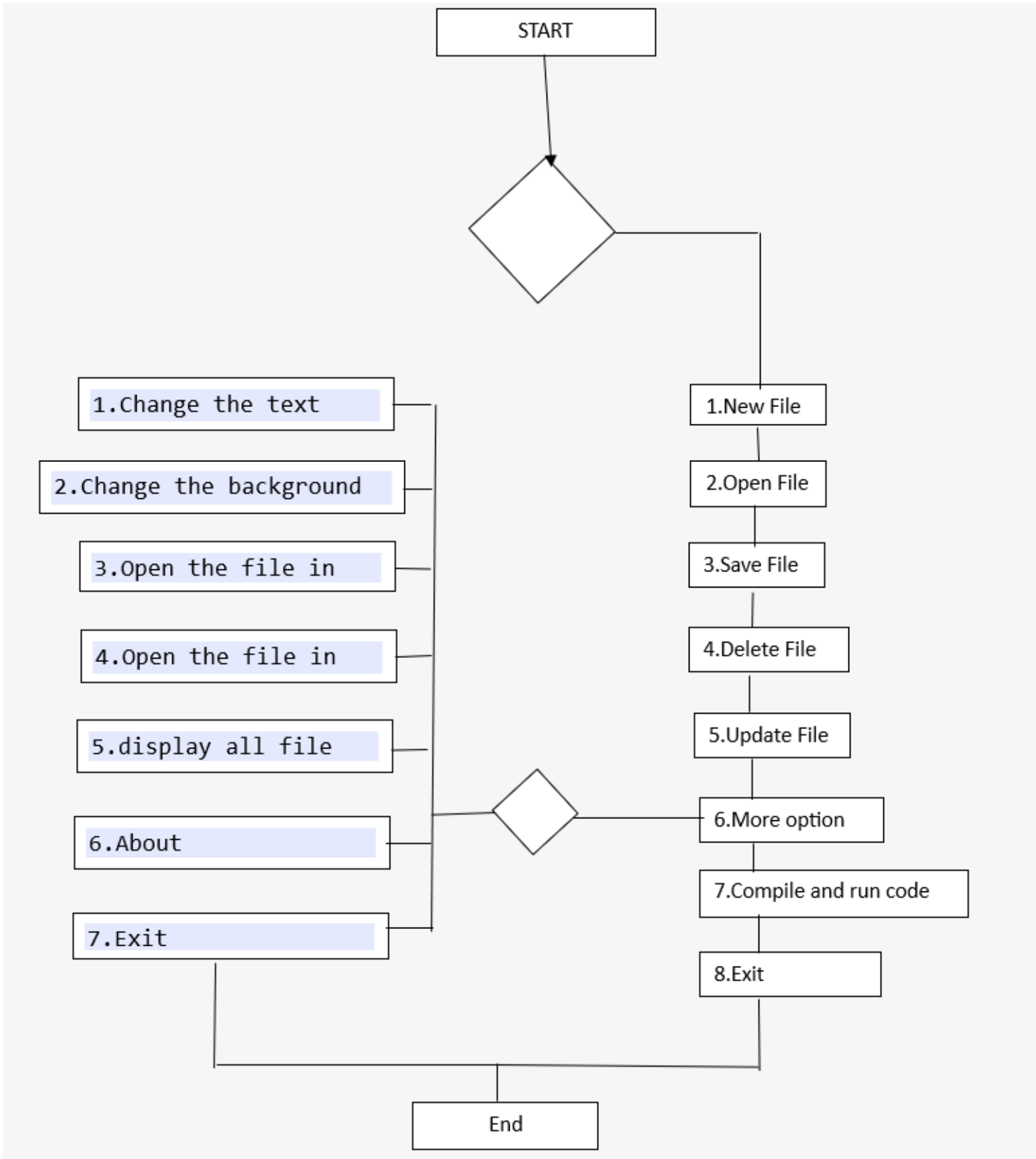
- **Polymorphism:** Polymorphism allowed for diverse code recognition challenges and IDEplay mechanics while maintaining code simplicity.

- **Abstraction:** Abstraction streamlined the creation of various IDE elements, maintaining a consistent structure.

This approach, combined with c++ provided a solid foundation for IDE ensuring code organization, reusability, and adaptability for future enhancements.

DATA FLOW DIAGRAM




START
1.Change the text
2.Change the background
3.Open the file in
4.Open the file in
5.display all file
6.About
7.Exit
1.New File
2.Open File
3.Save File
4.Delete File
5.Update File
6.More option
7.Compile and run code
8.Exit
End

DATA DICTIONARY

1. **developer Profile:**
   - *Attributes:* Username, developer ID, achievements, progress.
   - *Description:* Stores information about individual developers, including their in-IDE identity, achievements, and progression through the IDE.

2. **code Database:**
   - *Attributes:* code ID, brand name, image file, category.
   - *Description:* Containsdetailsaboutcodesused in theID E, including their identification, associated brand names, image files, and category classifications.

3. **IDE Progress:**
   - *Attributes:* developer ID, level, score, challenges completed.
   - *Description:* Tracks the progress of each developer, including the level they've reached, their score, and the number of challenges completed.

4. **User Settings:**
   - *Attributes:* Sound settings, notification preferences, interface preferences.
   - *Description:* Stores user-specific settings and preferences, such as compiler and interface settings.

6. **Hint System:**

   - *Attributes:* Hints available, hint usage.

   - *Description:* Records the availability of hints and how they are utilized the IDE

These data dictionary points help manage and organize the IDE's data, allowing for efficient data retrieval, storage, and utilization within the IDE IDE system.

USER INTERFACE DESIGN

The user interface (UI) of IDE has been meticulously crafted to be user-friendly and accessible to individuals of all age groups. The design focuses on simplicity and intuitiveness, ensuring a seamless development experience for everyone.

**1. Main Menu:

- The main menu provides a straightforward starting point for developers. It features large, easily recognizable option that lead to various IDE modes, settings, and achievements. The text is clear and legible, making navigation effortless.

**2. IDE Screen:

- During IDEplay, the IDE screen maintains a clean and uncluttered appearance. The code image is prominently disdevelop at the center, with space for developer inputs. Hints and scoring information are presented clearly, ensuring developers can focus on the challenge at hand.

The user interface of IDE embraces a design philosophy that puts user accessibility at the forefront. It has been crafted with simplicity and ease of use in mind, making it welcoming and engaging for developers of all ages.

DATABASE DESIGN AND NORMALISATION

In the development of IDE a normalized database design was employed to ensure data integrity and efficient management of information. The following aspects of normalization were applied:

1. **Main Menu Options
   - To avoid data redundancy and inconsistencies, the information related to main menu options was structured into separate tables, allowing for efficient storage and retrieval. This approach ensures that IDE modes, settings, and achievements are well-organized and coherent.

2. **code Database
   - The code Database was normalized by categorizing codes and their attributes into distinct tables. This approach reduces data duplication and improves data integrity. It enables efficient management of code information, promoting IDE variety and accessibility.

By implementing normalization, IDE optimizes data storage and retrieval, ensuring that the IDE's database remains organized and efficient. This normalization approach aligns with the IDE's design philosophy of simplicity and accessibility.

DATABASE MANIPULATION

the IDE assets and database are enclosed within the IDE itself, and there is no external online database for codes. As a result, database manipulation is confined to what is stored within the IDE files and assets. Here's how it operates:

**1. Local Database Integrity:
- The database of codes, including attributes like code ID, brand names, image files, and categories, is integrated into the IDE's local assets. These assets are packaged with the IDE and reside on the developer's device.

**2. IDE File Security:
- To maintain the integrity of the IDE and prevent unauthorized manipulation, the IDE assets are protected. IDE files are secured to minimize tampering with the database or any other critical IDE components.

**3. Database Updates:
- If any changes or updates to the database are required, they need to be implemented as part of the IDE's official updates or releases. These updates are managed by the IDE's development team and distributed through app stores or official channels.

**4. IDE Assets Consistency:
- Ensuring the consistency of IDE assets is essential to maintain the IDE's reliability and fairness. Any attempt to manipulate the local database or assets could result in disruptions or unintended consequences during development.

In IDE the IDE's database and assets are self-contained within the IDE files. This approach provides a high level of security and control to maintain the IDE's integrity and fairness. Any modifications or updates to the database are managed and implemented by the IDE's developers to ensure a consistent and enjoyable development experience.

USER INTERFACE DESIGN

The user interface (UI) of IDE has been meticulously crafted to be user-friendly and accessible to individuals of all age groups. The design focuses on simplicity and intuitiveness, ensuring a seamless development experience for everyone.

**1. Main Menu:

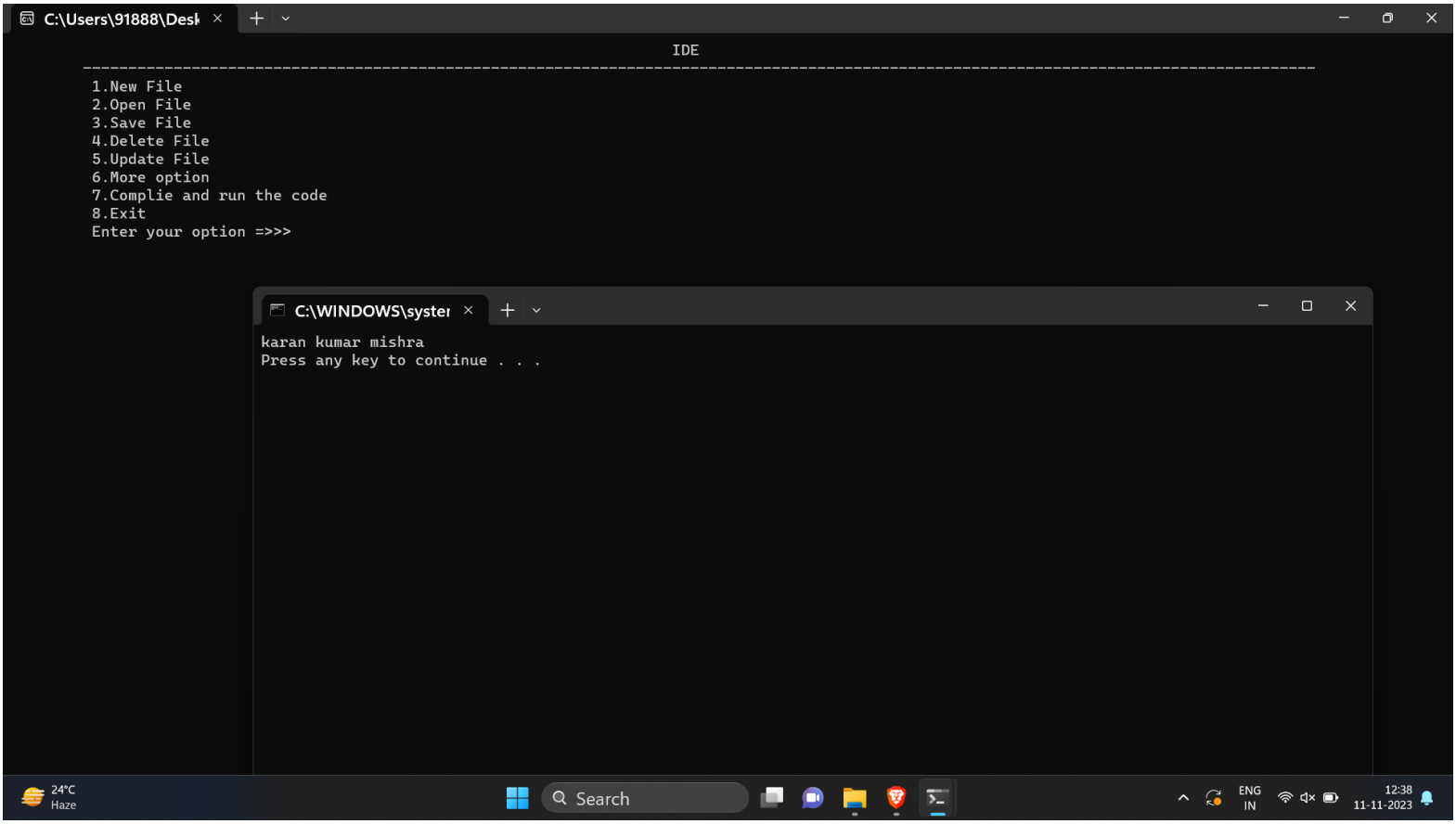
- The main menu provides a straightforward starting point for developers. It features large, easily recognizable icons that lead to various IDE modes, settings, and achievements. The text is clear and legible, making navigation effortless.

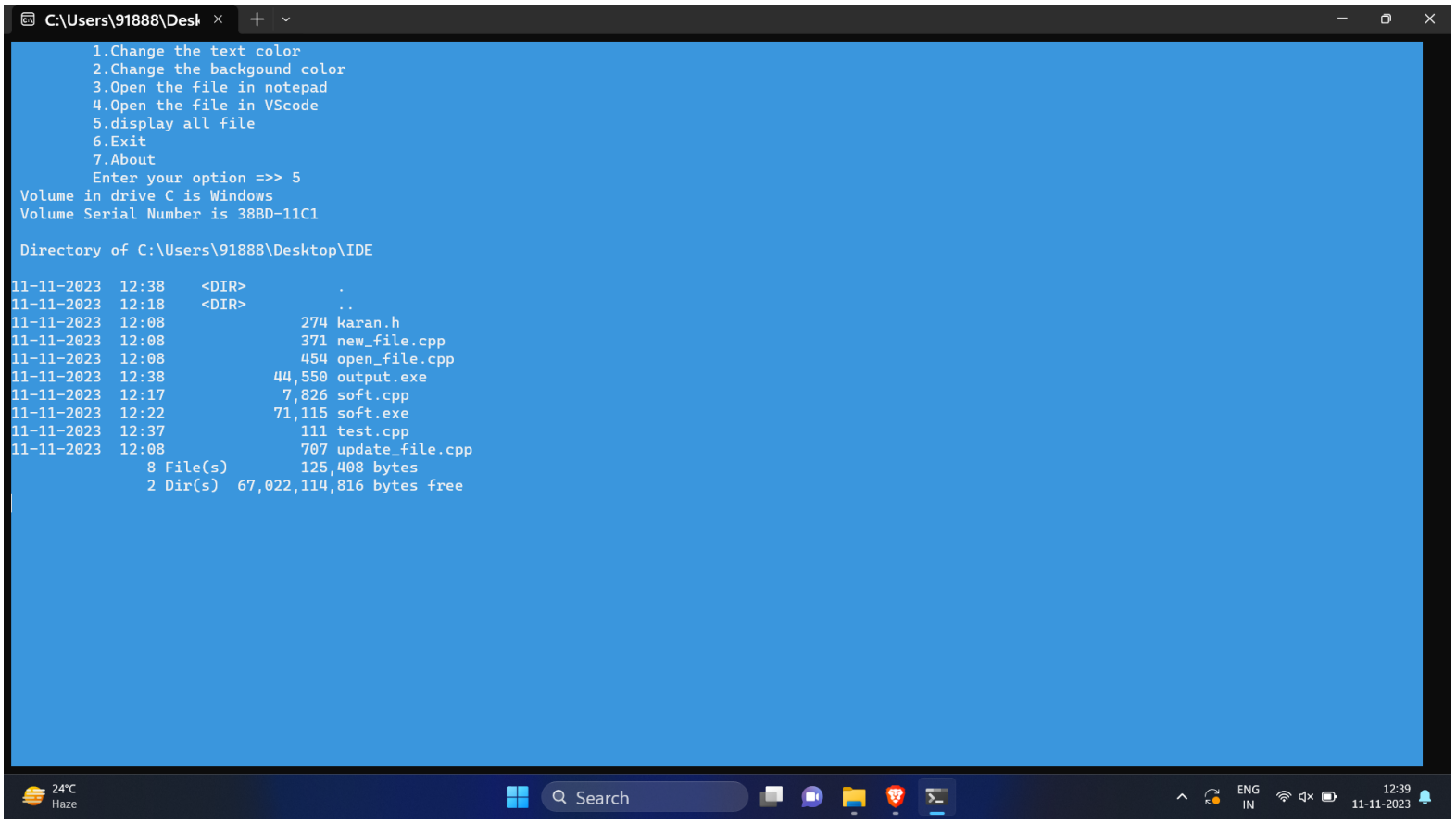
**2. IDE Screen:

- During IDE's development, the IDE screen maintains a clean and uncluttered appearance. The code image is prominently disdevelop at the center, with space for developer inputs. Hints and scoring information are presented clearly, ensuring developers can focus on the challenge at hand.

The user interface of IDE embraces a design philosophy that puts user accessibility at the forefront. It has been crafted with simplicity and ease of use in mind, making it welcoming and engaging for all developers .

USER INTERFACE DESIGN





CHAPTER 4
IMPLEMENTATIONS

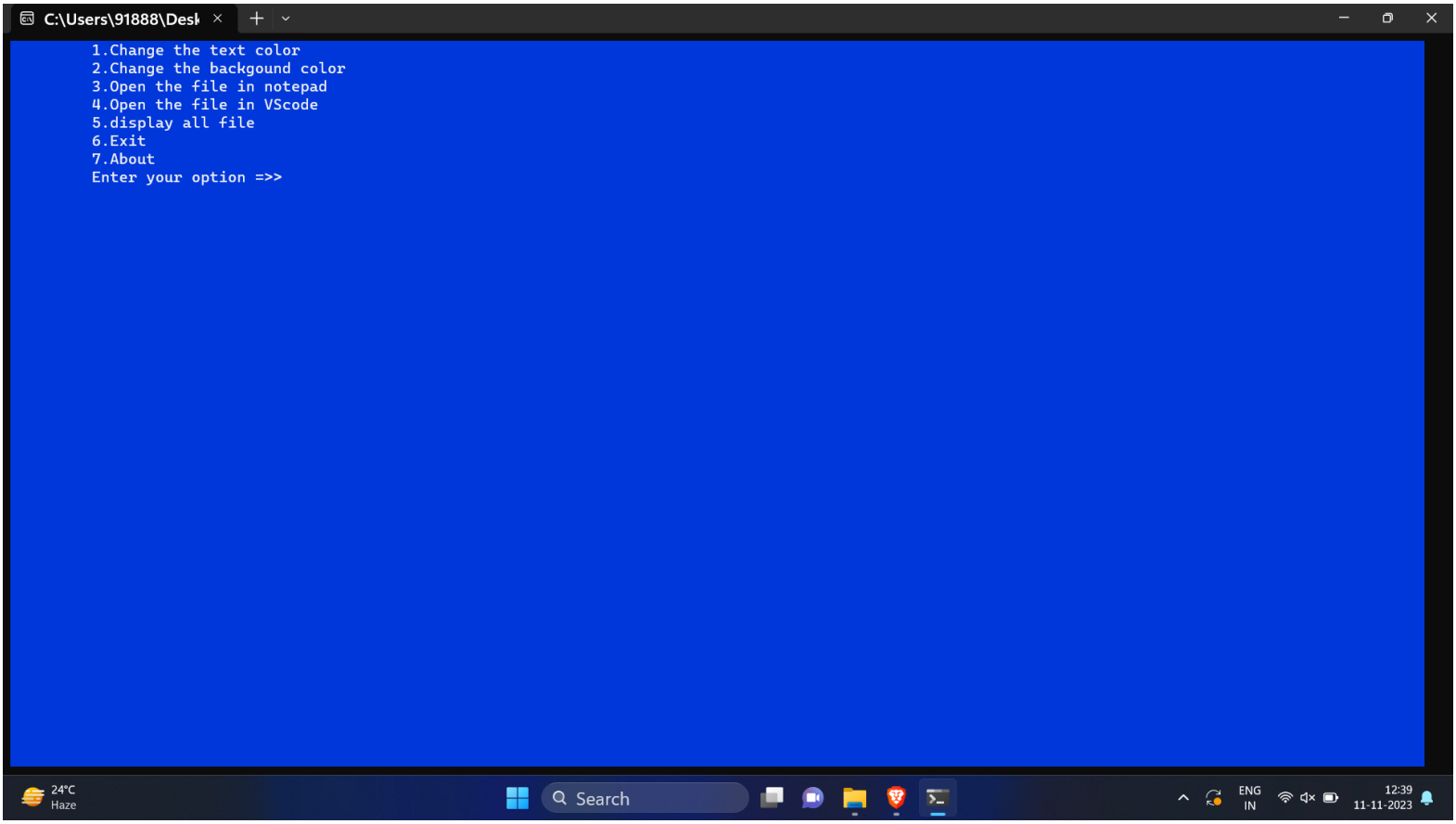
In the development of IDE we employed a systematic approach that included implementation, testing, and plans for future maintenance.

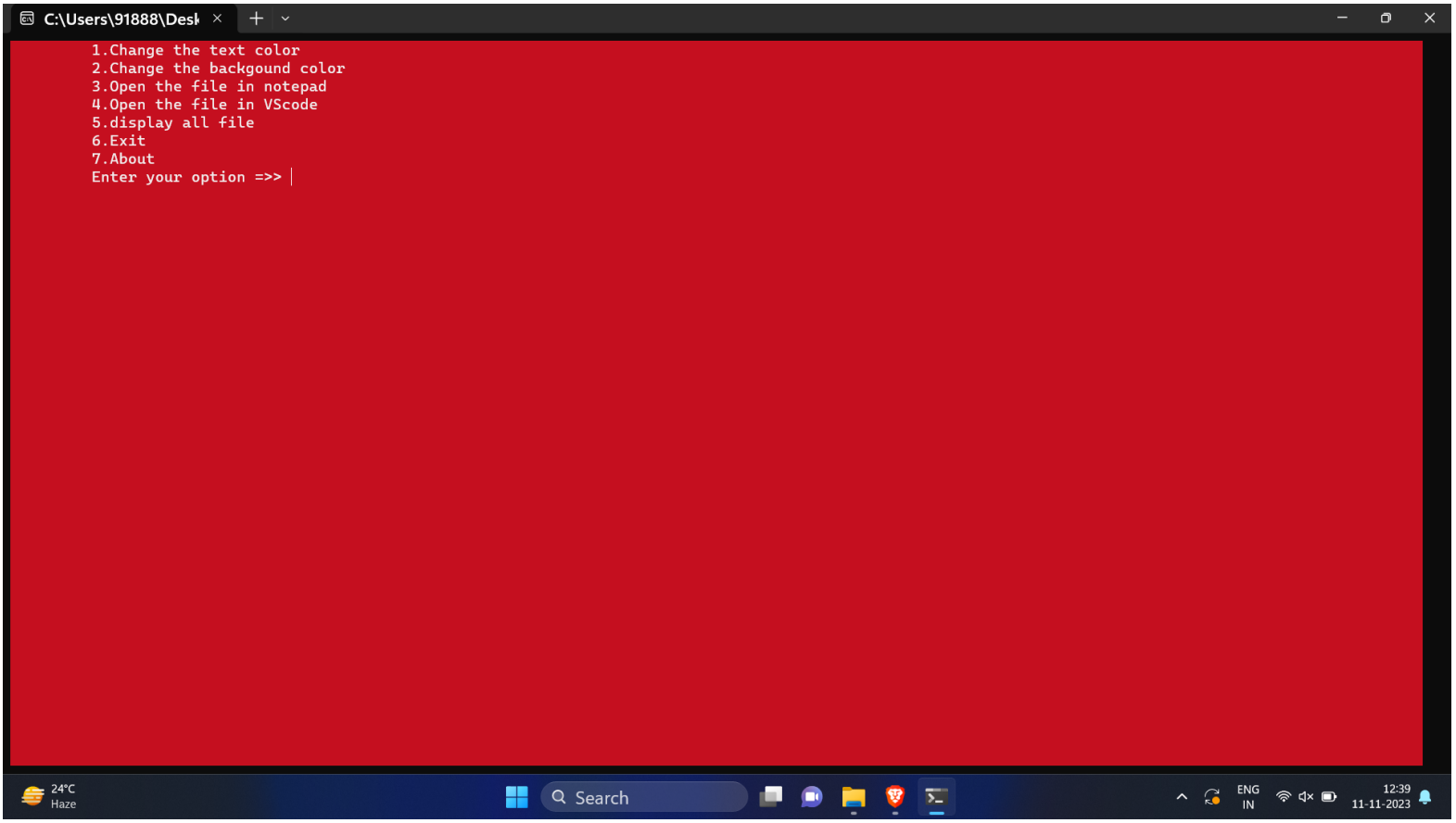
**Implementation:
- Windows was our primary development environment for creating IDE This powerful IDE provided the tools and resources required for Windows IDE development.
- We meticulously implemented the IDE's features, from the code recognition algorithm to the user interface design. The object-oriented approach in Kotlin ensured efficient code development, and the database design facilitated data management.

**Testing:
- For testing, we leveraged the Windows plugin, which allowed us to create a virtual Windows environment on our PCs. This virtual setup mimicked the Windows operating system, enabling comprehensive testing without the need for physical devices.
- Rigorous testing phases, including unit testing and integration testing, were executed to ensure the IDE's reliability, functionality, and user experience. User testing provided valuable insights for improvements.

**Maintenance:
- The concept of maintenance in IDE is not only about fixing issues but also about re-engineering the software for further enhancements.
- Over time, as the IDE evolves and developer demands change, we anticipate making improvements, including adding new code categories, challenges, and features.
- Maintenance also includes updates to accommodate changes in Windows OS versions and to enhance the IDE's performance and security.

Windows SNAPSHOT



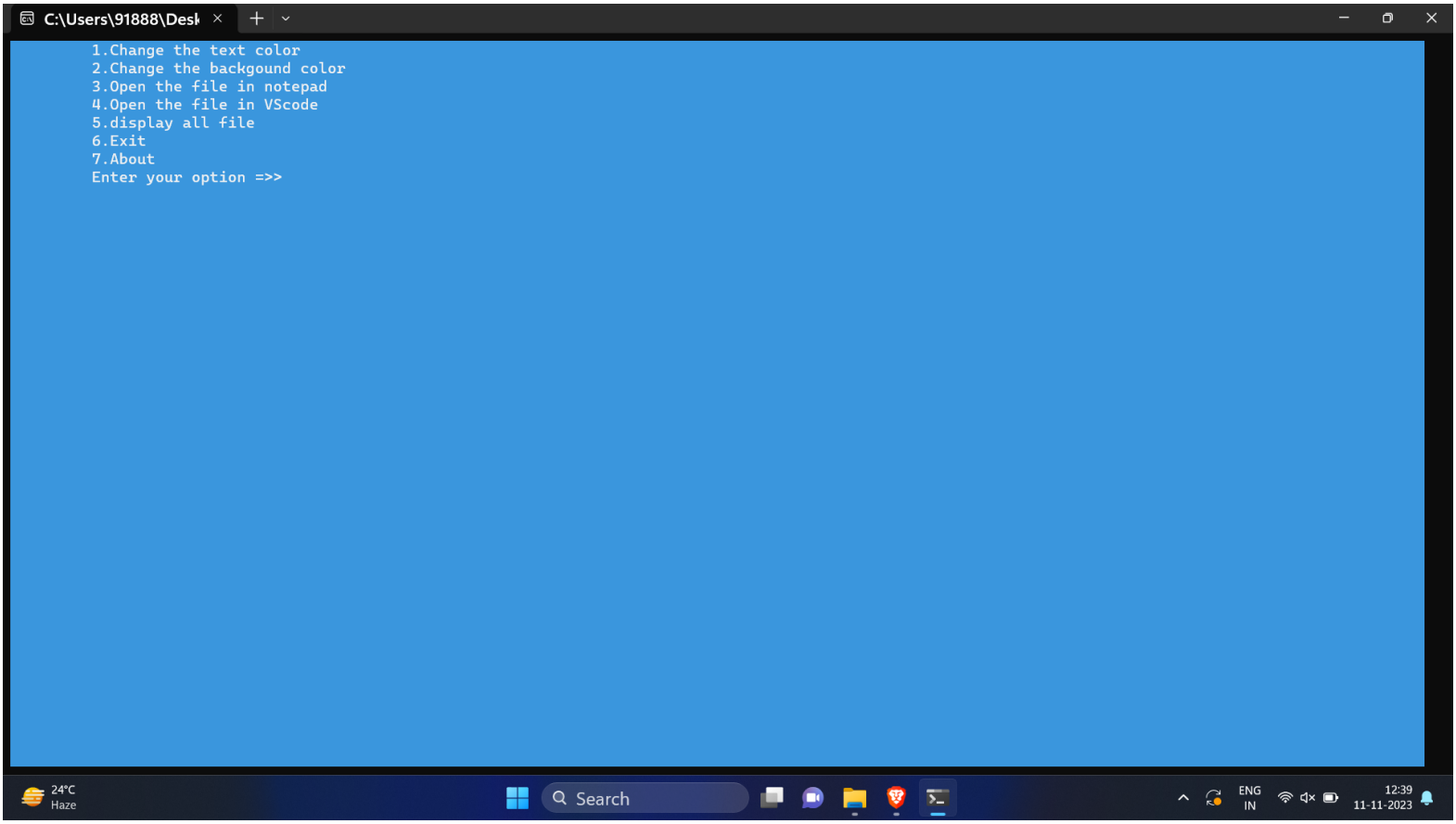


INTRODUCTION TO LANGUAGES

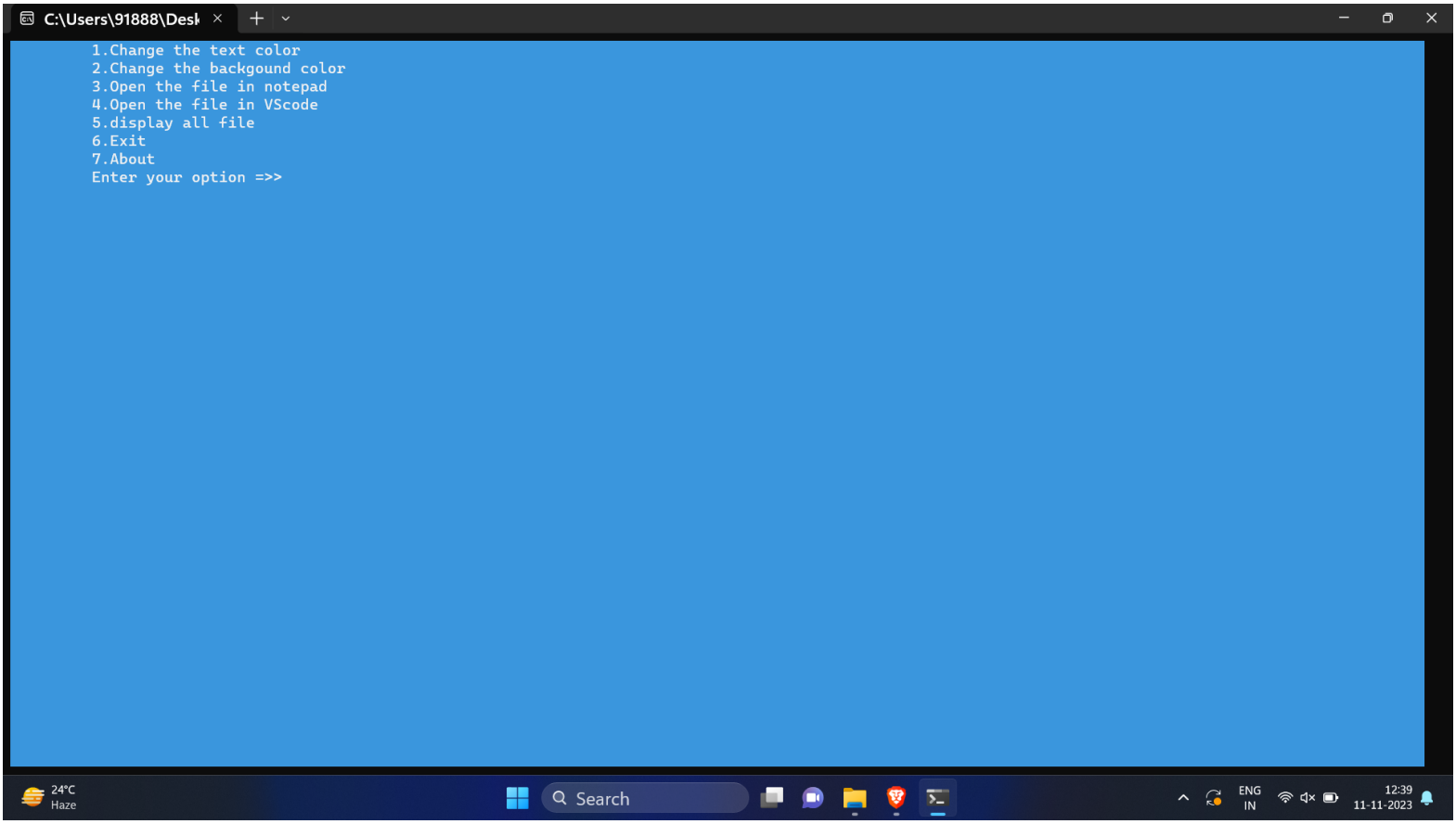
The development of IDE was made possible through a carefully chosen technology stack, which encompassed programming languages, design tools, version control systems, and integrated development environments (IDEs). This chapter provides an overview of the technologies employed in the creation of the IDE.

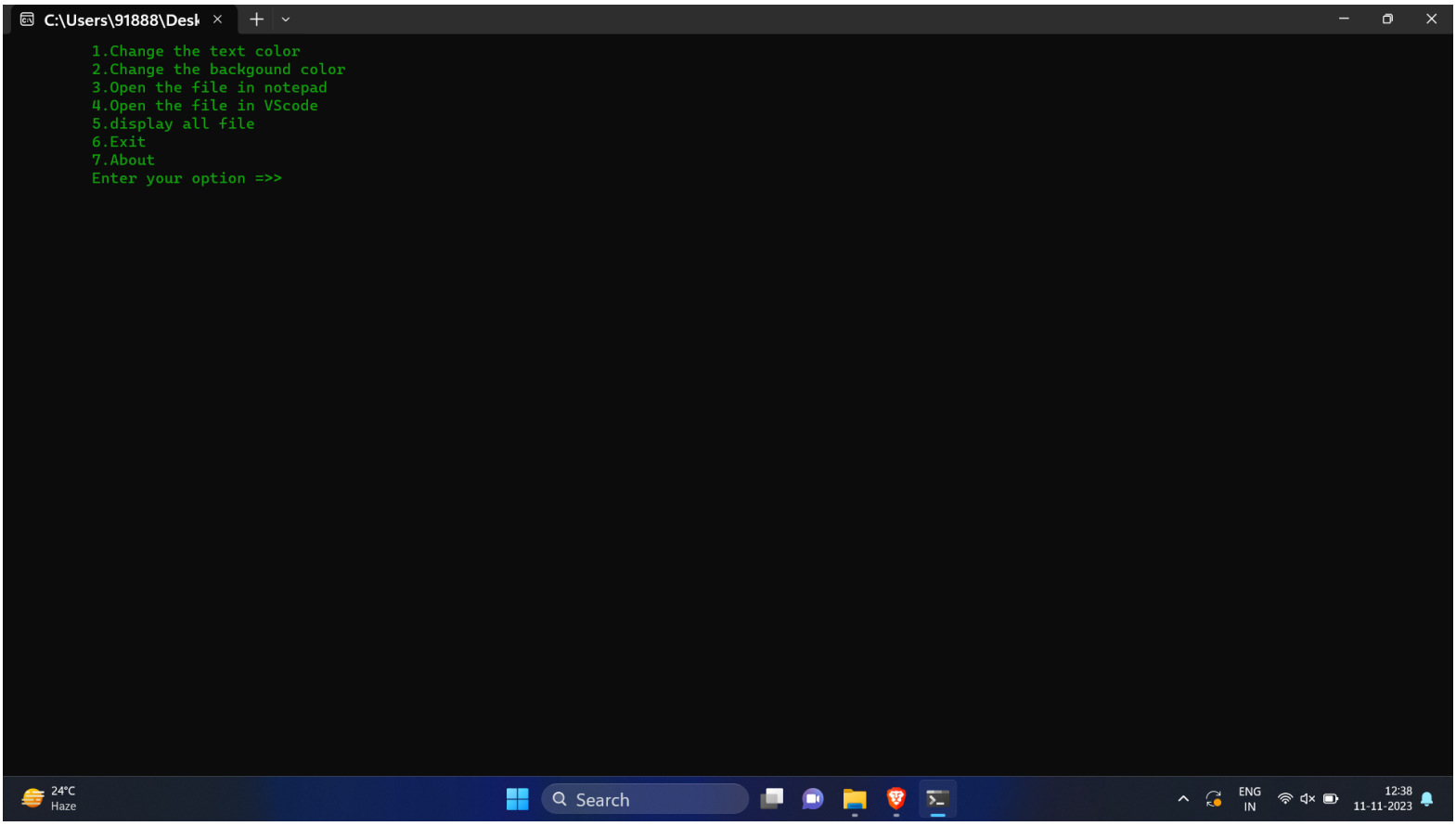
Programming Languages:

1. **c++: The primary programming language used for IDE development, known for its conciseness and compatibility with windows.



THESE ARE SOME IDE'S EXAMPLES





CODING STANDARDS OF LANGUAGE USED

In the development of IDE adherence to coding standards was a fundamental practice to maintain code quality, readability, and consistency. The following coding standards were applied to the languages used in the development process:

1. c++:

- In c++ we followed naming conventions that reflect the nature and purpose of variables, functions, and classes. This promotes code readability and understanding.
- We utilized consistent indentation and spacing to ensure code formatting consistency, making the codebase more accessible for developers.

The application of these coding standards served to maintain the codebase's quality and readability. By following best practices, we aimed to facilitate collaboration among developers, streamline debugging, and enhance the overall stability and maintainability of IDE

PROJECT SCHEDULING AND MONITORING

The development and launch of IDE involved a carefully planned project schedule, along with monitoring mechanisms to track key performance indicators. Here's an overview of our project scheduling and monitoring approach:

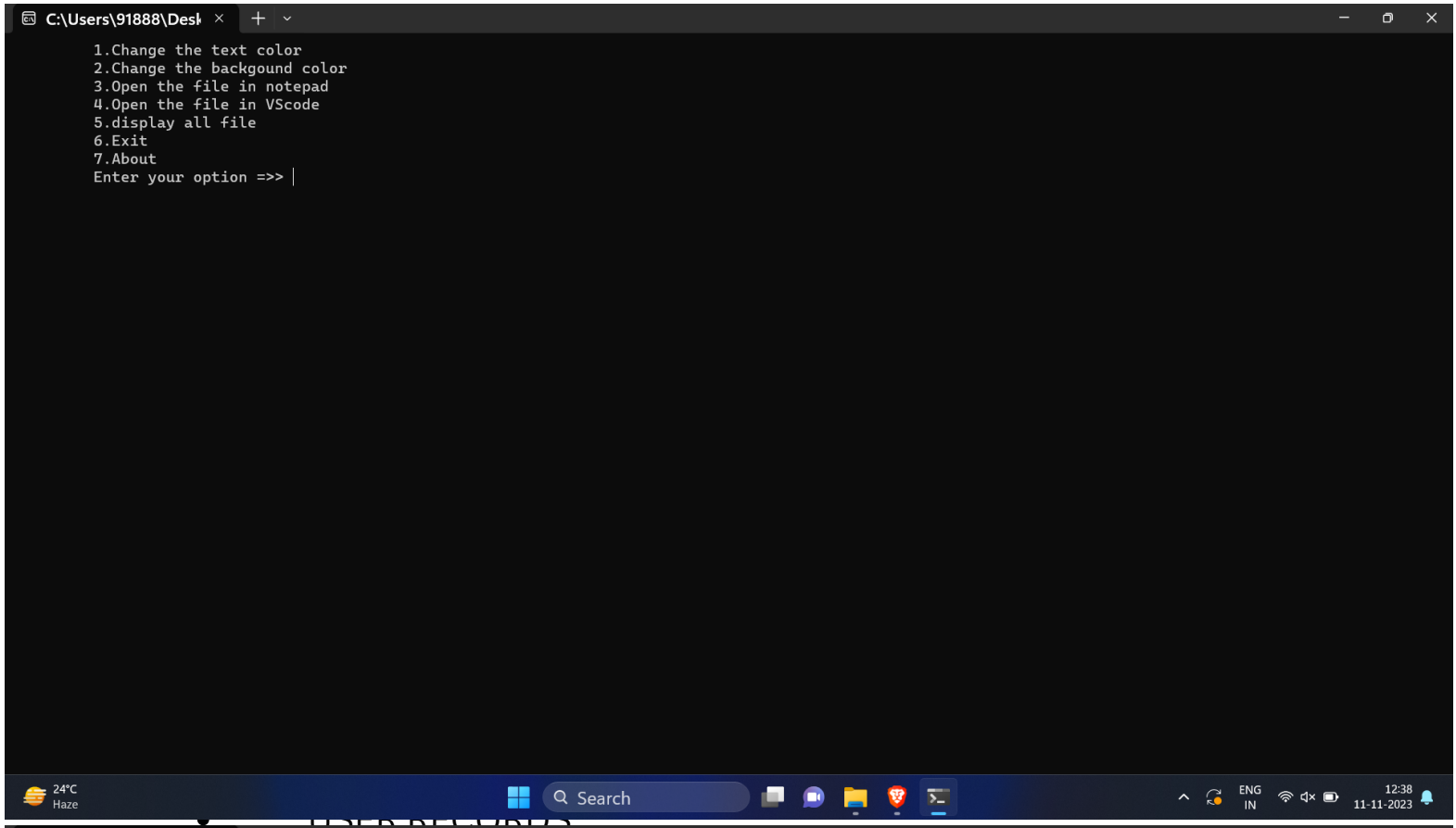
Project Scheduling:

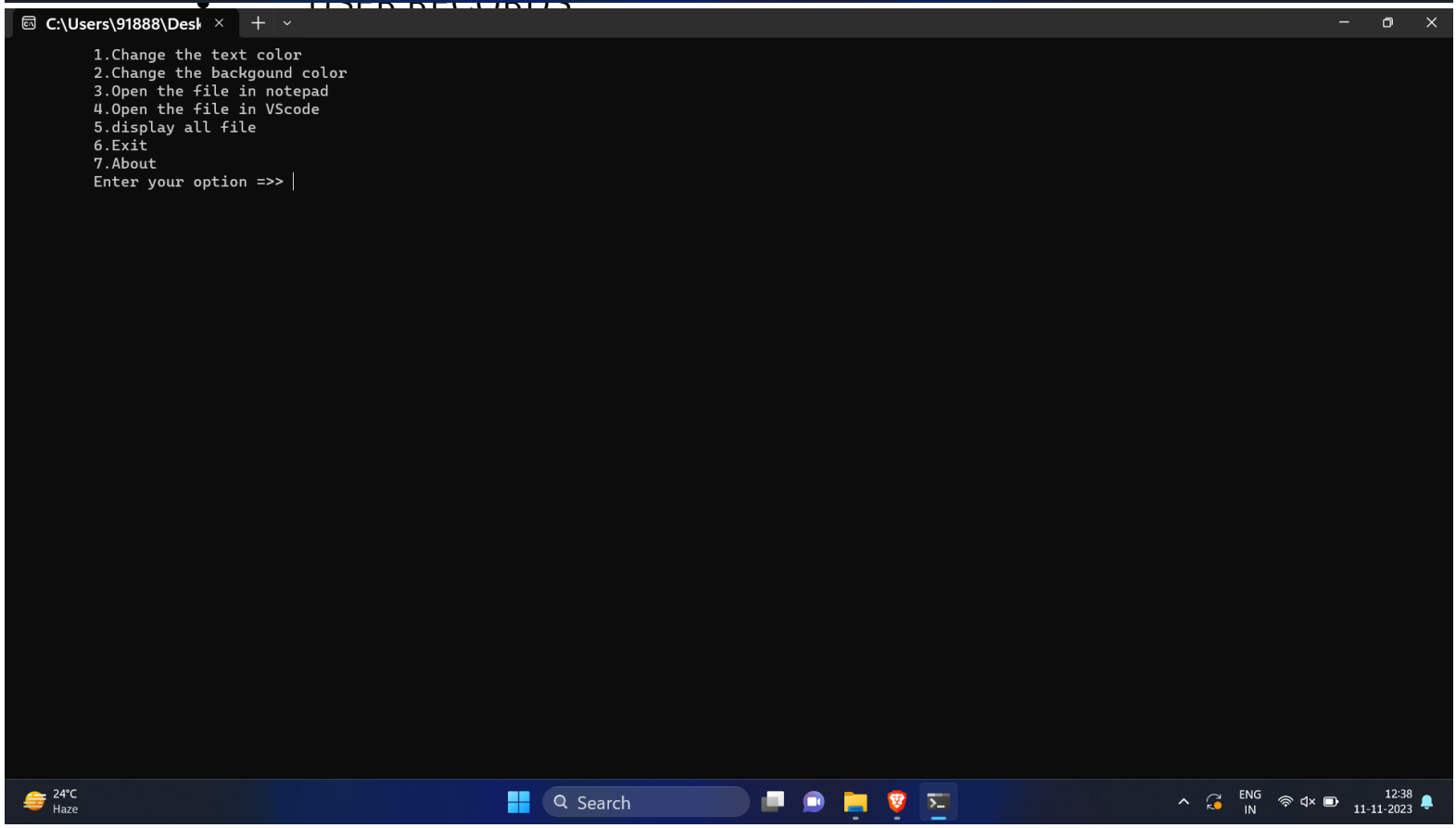
- **Project Phases:** The development was divided into distinct phases, including planning, design, coding, testing, and deployment. Each phase had its own timeline and milestones.
- **Task Dependencies:** Task dependencies were established to ensure a smooth transition between phases. For instance, design elements had to be finalized before coding began.
- **Agile Methodology:** An agile development approach was adopted, allowing for flexibility in accommodating changes and improvements as the project progressed.

Monitoring and Analytics:

- **Google Analytics:** To monitor the IDE's performance, Google Analytics was integrated into IDE This tool provided insights into various aspects of user engagement, such as downloads, active users, and in-IDE behavior.
- **Download Metrics:** Google Analytics tracked the number of downloads, including the source of downloads (e.g., Github.com). This data helped gauge the IDE's initial success.
- **Active Users:** The number of active IDE's users was continuously monitored, enabling us to assess developer retention and engagement levels.
- **developer Behavior:** Analytics also provided data on in-IDE actions, such as the completion of challenges, use of hints, and the progression of developers through various levels.

PROJECT SCHEDULING (MONITORING)





TESTING TECHNIQUES

The quality and reliability of IDE were of paramount importance, and thus, comprehensive testing techniques were employed to ensure a robust IDE. The following testing techniques and test plan were integral to the development process:

Testing Techniques:

1. **Unit Testing:**
   - Individual IDE components were tested in isolation to verify that they functioned as expected. This ensured that each module, class, or method performed its intended purpose accurately.

2. **Integration Testing:**
   - Integration tests were conducted to assess how various IDE components interacted with one another. This ensured that different parts of the IDE worked seamlessly when combined.

3. **User Testing:**
   - Real users were involved in testing to gain valuable insights into the IDE's usability and overall user experience. This feedback develop a crucial role in improving IDEplay and addressing user concerns.

4. **Performance Testing:**
   - Performance tests were conducted to assess the IDE's responsiveness, processing speed, and resource utilization. This ensured smooth IDEplay for developers across different devices and scenarios.

5. **Security Testing:**
   - Security tests were implemented to identify and rectify vulnerabilities, protect user data, and safeguard the IDE against hacking or unauthorized access.

TEST PLAN

The Test plan for IDE was structured to cover every aspect of the IDE:

1. **Objective:** The primary objective was to verify the IDE's functionality, performance, and security to ensure a high-quality user experience.

2. **Testing Scope:** This included testing various IDE features such as code recognition, user interface, IDE mechanics, database functionality, and offline modes.

3. **Testing Environment:** A controlled testing environment was set up, including real Windows devices, emulators, and varying Windows OS versions.

4. **User Scenarios:** Real-world user scenarios were simulated to test the IDE under conditions that developers would encounter.

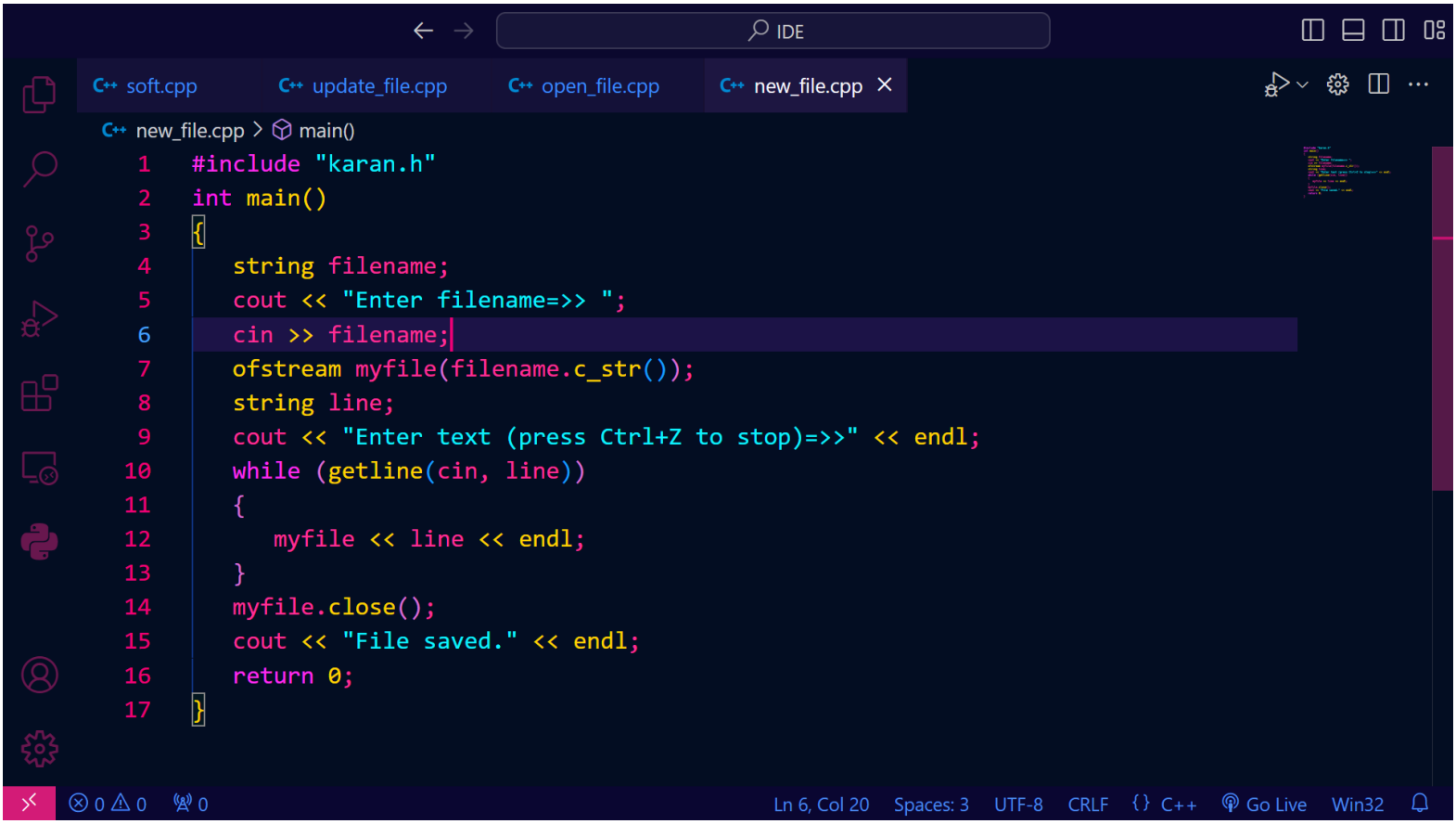
5. **Testing Timeline:** A timeline was established to ensure that testing phases were conducted at appropriate project stages, from unit testing during development to user testing and security testing before release.

6. **Bug Tracking:** A system for reporting and tracking bugs was put in place to identify and rectify issues during testing.

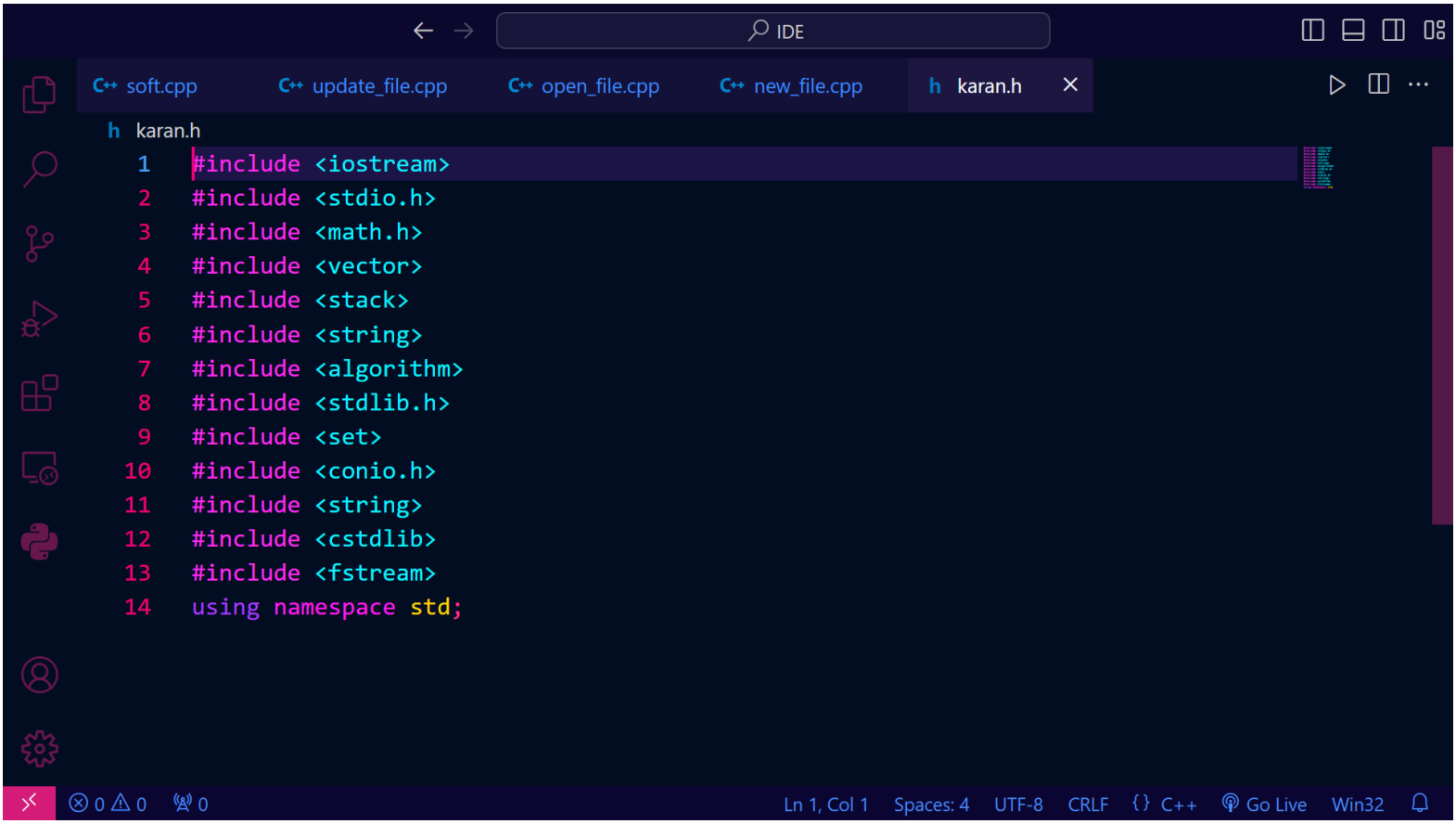
7. **Iterative Testing:** Testing was an ongoing process, with continuous iterations and updates to address issues and make improvements based on user feedback.

The combination of these testing techniques and a comprehensive test plan develop a pivotal role in delivering a high-quality, reliable, and enjoyable development experience with IDE

CHAPTER 5
RESULTS AND DISCUSSIONS

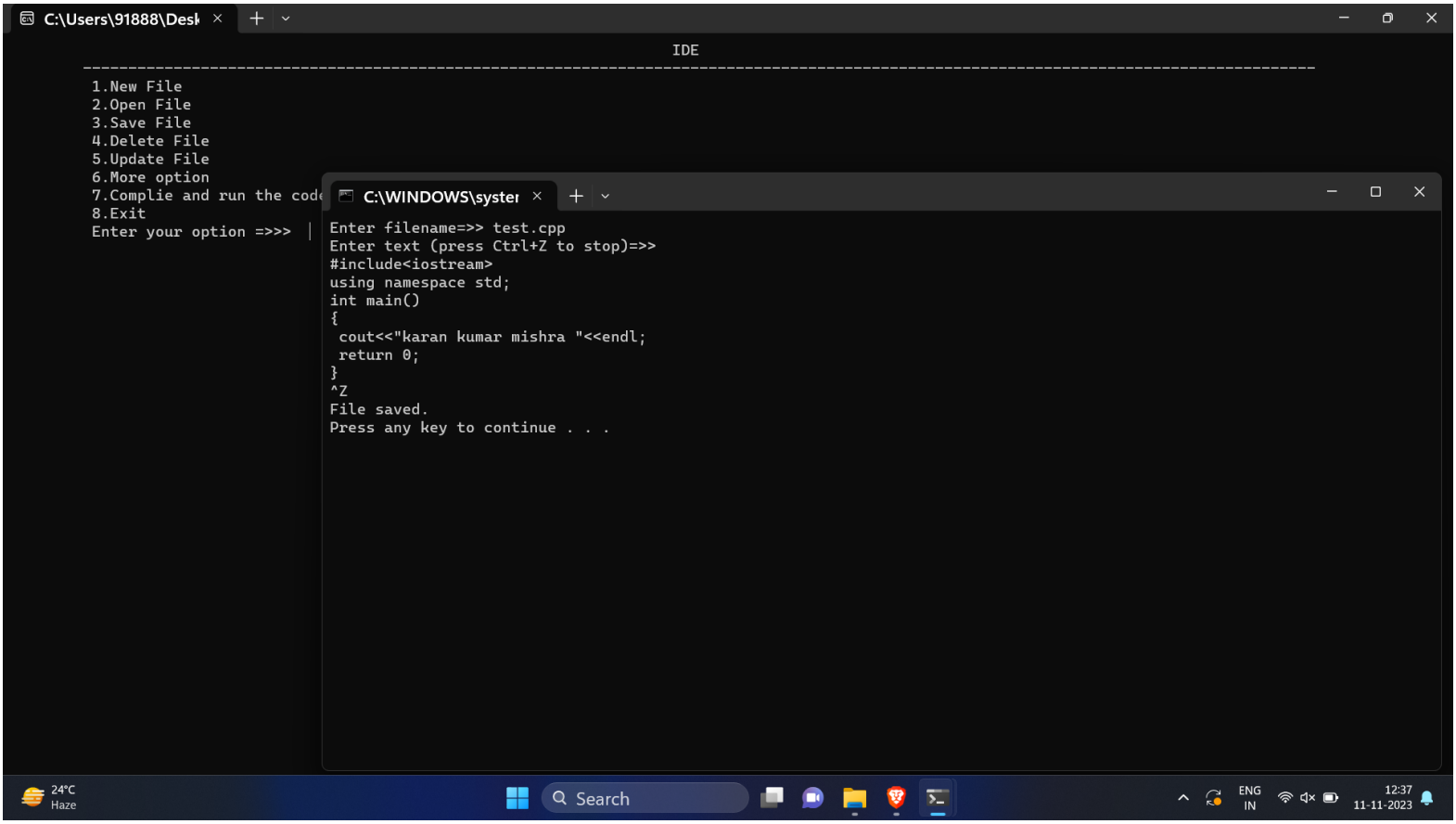


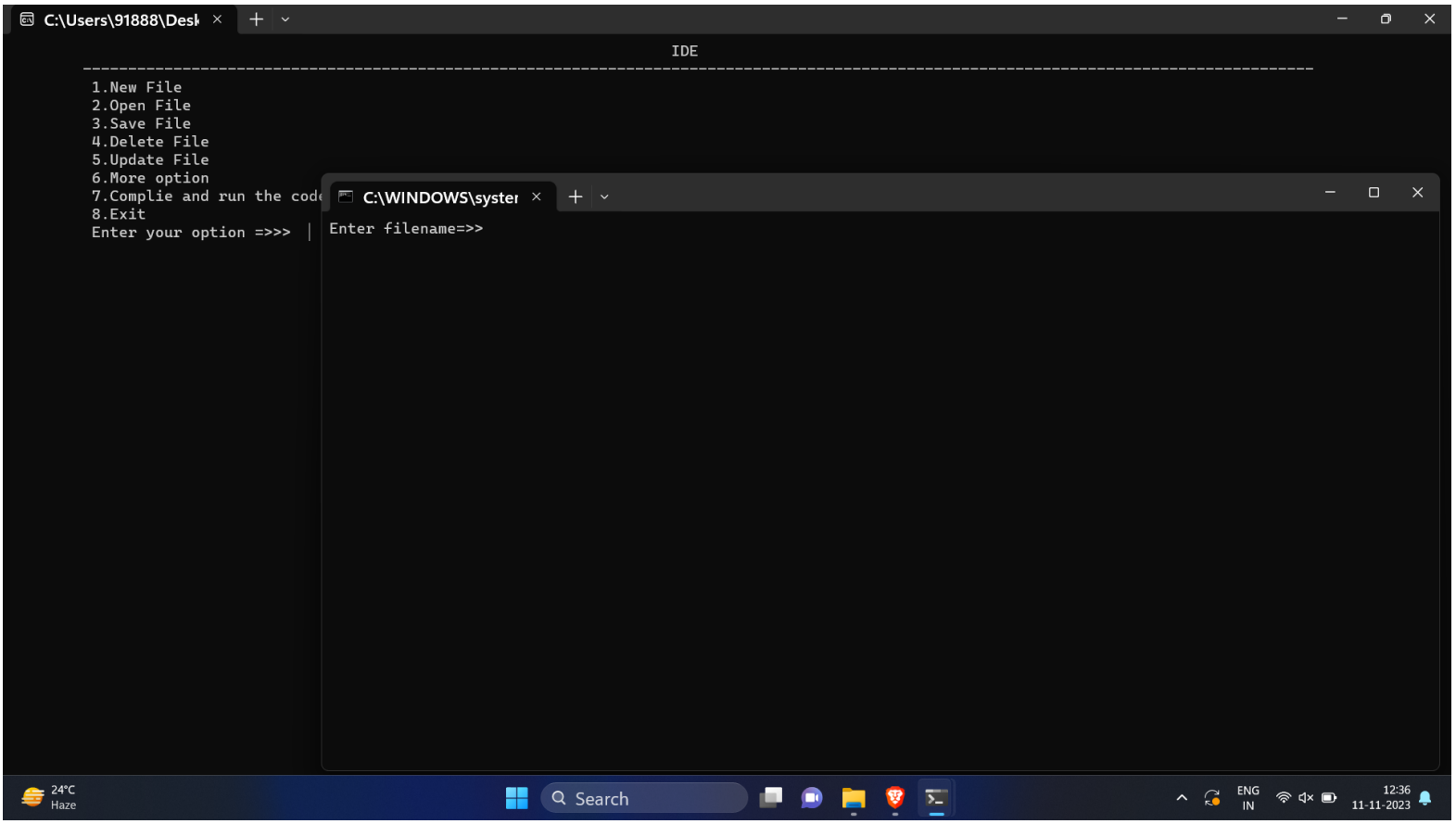
SCREENSHOT


IDE
soft.cpp
update_file.cpp
open_file.cpp
new_file.cpp
karan.h
karan.h
#include <iostream>
#include <stdio.h>
#include <math.h>
#include <vector>
#include <stack>
#include <string>
#include <algorithm>
#include <stdlib.h>
#include <set>
#include <conio.h>
#include <string>
#include <cstdlib>
#include <fstream>
using namespace std;
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Win32


IDE
soft.cpp
update_file.cpp
update_file.cpp > main()
#include "karan.h"
int main()
{
string filename;
cout << "Enter the file name=>" << endl;
cin >> filename;
ifstream in;
in.open(filename);
if (!in.good())
{
cout << "File is not found try again " << endl;
getch();
return 0;
}
cout << "Enter text (press Ctrl+Z to stop)=>>" << endl;
char ch;
in >> ch;
while (!in.eof())
{
Ln 15, Col 5
Spaces: 4
UTF-8
CRLF
C++
Go Live
Win32

USER INTERFACE REPRESENTATION





BRIEF DESCRIPTION OF MODELS OF THE SYSTEM

In the development of IDE various system models were utilized to guide and structure different aspects of the IDE. These models provided a conceptual framework for understanding, designing, and implementing the IDE. Here's a brief description of the key system models employed:

1. Waterfall Model:

- The Waterfall model was used for initial project planning and requirements analysis. It provided a structured approach for defining project goals, objectives, and core requirements. This model set the project's direction and scope.

2. Agile Model:

- The Agile model was pivotal for development. It facilitated adaptability and responsiveness to changing project needs. Agile principles allowed for iterative development, incorporating user feedback and refining the IDE incrementally.

3. Object-Oriented Model:

- The Object-Oriented model, particularly using c++ the development of IDE It organized code into modular, reusable objects, making it more maintainable and efficient.

4. User-Centered Design Model:

- The User-Centered Design model guided the creation of the IDE's user interface and overall user experience. It prioritized the needs and preferences of developers, resulting in a user-friendly and engaging design.

5. Iterative Model:

- The Iterative model was essential for ongoing updates and enhancements post-launch. It allowed for continuous improvement, addressing user feedback and refining the IDE over time.

Each of these system models develop a distinct role in the development of IDE collectively ensuring the IDE's successful creation, user-friendliness, and adaptability to evolving developer preferences.

BACKEND REPRESENTATION (SNAPSHOT)

BACKEND REPRESENTATION (SNAPSHOT)