BSCS FINAL PROJECT
<Software Design Specification>

CodeFlow

Project Advisor

Mohsin Sami

Presented by:
Group ID: F23SE052

Student Reg#	Student Name
L1F20BSSE0191	Muhammad Haseeb Nawaz
L1F20BSSE0183	Muhammad Mujeeb
L1F19BSSE0073	Rohan Qamar

Faculty of Information Technology

University of Central Punjab

Software Design Specification

SDP Phase II

CodeFlow

Advisor: Mohsin Sami

Group F23SE052

Member Name	Primary Responsibility
Muhammad Haseeb Nawaz	Requirement Specification, Implementation and documentation
Muhammad Mujeeb	Requirement Specification, Implementation and documentation
Rohan Qamar	Requirement Specification, Implementation and documentation

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

Abstract

CodeFlow is an innovative web-based solution designed to address the challenges faced by novice programmers and students in understanding programming concepts. The project aims to bridge the gap between the abstract nature of code and the complexities of creating logical program flows. It provides an intuitive drag-and-drop interface for users to construct program code using flowchart elements. Additionally, CodeFlow offers a sophisticated algorithm for accurately converting flowcharts into textual code and existing code into flowchart, enabling learners to grasp program logic. To enhance the learning experience, the platform includes step by step flowchart execution with memory map visualization, allowing students to observe data storage and manipulation during program execution. By adopting CodeFlow, novice learners can expect a more accessible and approachable programming environment, boosting their confidence and problem-solving skills. The visual representation of code and the dynamic learning environment are expected to improve understanding and retention of programming principles. This project aims to foster creativity, engagement, and motivation among learners, offering them valuable insights into computer memory management. CodeFlow stands as a transformative solution to the challenges faced by those embarking on their programming journey, making programming concepts more comprehensible and accessible compared to existing solutions.

1. Introduction

1.1 Product

The field of programming education faces a significant challenge in providing an accessible and effective learning environment for novice programmers and students. The abstract nature of code, along with the complexities involved in creating logical program flows, poses substantial barriers to understanding fundamental programming concepts. Aspiring programmers often struggle to comprehend how programming works behind the scenes, how memory is allocated, and how to construct effective logical structures. The absence of user-friendly tools exacerbates these issues, hindering the learning progress and stifling enthusiasm for programming.
Existing solutions, such as Code and Flow and Raptor, have limitations in accurately translating textual code into visual representations and handling complex logical structures. While visual programming languages like Scratch offer a basic understanding of programming concepts, there is a lack of comprehensive platforms that seamlessly bridge the gap between visual and textual programming. There is a pressing need for an innovative, intuitive, and interactive solution that empowers learners to visualize and construct program logic effectively, fostering confidence and enhancing their programming skills.
CodeFlow addresses these challenges by providing a web-based platform with a user-friendly drag-and-drop interface. By allowing users to construct program code using flowchart elements, it simplifies the complexities of logical structure creation. The platform incorporates a sophisticated algorithm for accurate code-to-flowchart conversion, ensuring a seamless transition from textual code to visual representations. Additionally, CodeFlow integrates memory map visualization, enabling learners to observe data storage and manipulation during program execution. By offering a dynamic and interactive learning environment, CodeFlow aims to empower novice programmers, making programming concepts more accessible and comprehensible, thus fostering a new generation of Arial Italic confident and skilled programmers

1.2 Background

In today's digital age, programming literacy is no longer a niche skill but a fundamental requirement for various fields. As the demand for proficient programmers continues to rise, there is a growing need for effective and accessible programming education. Novice programmers and students often encounter challenges in understanding programming concepts due to the abstract nature of code and the complexities involved in creating logical program flows. Traditional methods of teaching programming rely heavily on textual explanations and lack interactive and intuitive tools for visualizing code. Existing programming education tools have limitations in providing a seamless and engaging learning experience for beginners. While visual programming languages like Scratch offer a graphical approach to coding, they often lack the depth required for transitioning to text-based languages. Flowchart-based solutions like Code and Flow and Raptor have shortcomings in accurately translating textual code into visual representations, hindering the understanding of complex programming logic.
Against this backdrop, the CodeFlow project was conceived. CodeFlow aims to revolutionize programming education by providing a comprehensive and user-friendly platform for novice learners. By combining an intuitive drag-and-drop interface with a sophisticated code-to-flowchart conversion algorithm, CodeFlow bridges the gap between abstract code and visual representation. This innovative approach simplifies the complexities of programming logic, making it accessible and understandable for learners at various levels of expertise.
Furthermore, CodeFlow integrates memory map visualization, a crucial aspect often overlooked in programming education. This feature allows learners to observe the inner workings of a program, understanding how data is stored and manipulated during execution. By addressing these fundamental aspects of programming comprehensively, CodeFlow aims to empower learners with a deep understanding of programming concepts, fostering confidence and proficiency in their coding journey.

In the following sections, the methodology, features, and expected outcomes of the CodeFlow project will be discussed, highlighting its potential to transform the landscape of programming education for aspiring programmers and students.

1.3 Objective(s)/Aim(s)/Target(s)

The CodeFlow project is driven by a set of clear objectives, aims, and targets, all geared toward transforming the programming learning experience for novice learners and students. These objectives define the project's purpose and the specific goals it aims to achieve.

1.4 Scope

The scope of the CodeFlow project encompasses the development and implementation of a comprehensive web-based platform aimed at enhancing the learning experience for novice programmers and students.

1.5 Business Goals

This project does not have formal business goals or revenue targets. Its primary focus is on facilitating Arial Italic student learning and fostering collaboration within the programming community.

1.6 Document Conventions

Font: Times
Body
Body size: 12
Line spacing: 1.0
Heading
Heading1 size: 18
Heading2 size: 14
Sub-Heading: 12
Line spacing: 1.5

2. Overall Description

2.1 Product Features

Drag-and-Drop Interface
Intuitive interface allowing users to construct flowchart by dragging and dropping elements.
Flowchart-to-code Conversion
Accurate algorithm translating visual flowcharts into textual code
Code-to-flowchart Conversion
Accurate algorithm translating textual code into visual flowcharts for improved comprehension.
Program Execution
Step-by-step execution of created programs for visualizing logic and identifying errors.
Memory Map Visualization
Real-time representation of data storage and manipulation during program execution.
Export Flowchart and Json
Export flowchart as png and custom json file for any future editing

2.2 Functional Description

CodeFlow is a web-based platform its key functionality includes an interactive flowchart creation feature, enabling users to effortlessly build and modify programming flowcharts using an intuitive drag-and-drop interface. Additionally, the system offers flowchart-to-code conversion capabilities, converting user-created flowcharts into executable C++ code, facilitating the comprehension of how visual logic translates into textual programming. Conversely, it can transform existing C++ code into visual flowcharts with its code-to-flowchart conversion function, aiding users in grasping the structure and logic of written code. CodeFlow also provide step-by-step code execution, allowing users to run the generated C++ code incrementally, gaining valuable insights into the execution process and aiding in debugging. Furthermore, CodeFlow offers real-time learning and feedback, delivering immediate insights into flowchart logic and code as users interact with the system, thereby enhancing the overall learning experience.

2.3 User Classes and Characteristics

Novice Programmers: Limited programming experience, basic understanding of logic, seeking simplified learning tools.
Students: Enrolled in programming courses, varying levels of expertise, diverse educational backgrounds, in need of practical learning aid.
Educators: Experienced programmers, teaching programming, integrating technology into education, requiring effective teaching tools.
Self-Learners and Enthusiasts: Curious about programming, learning independently, exploring coding for hobbies or personal projects.

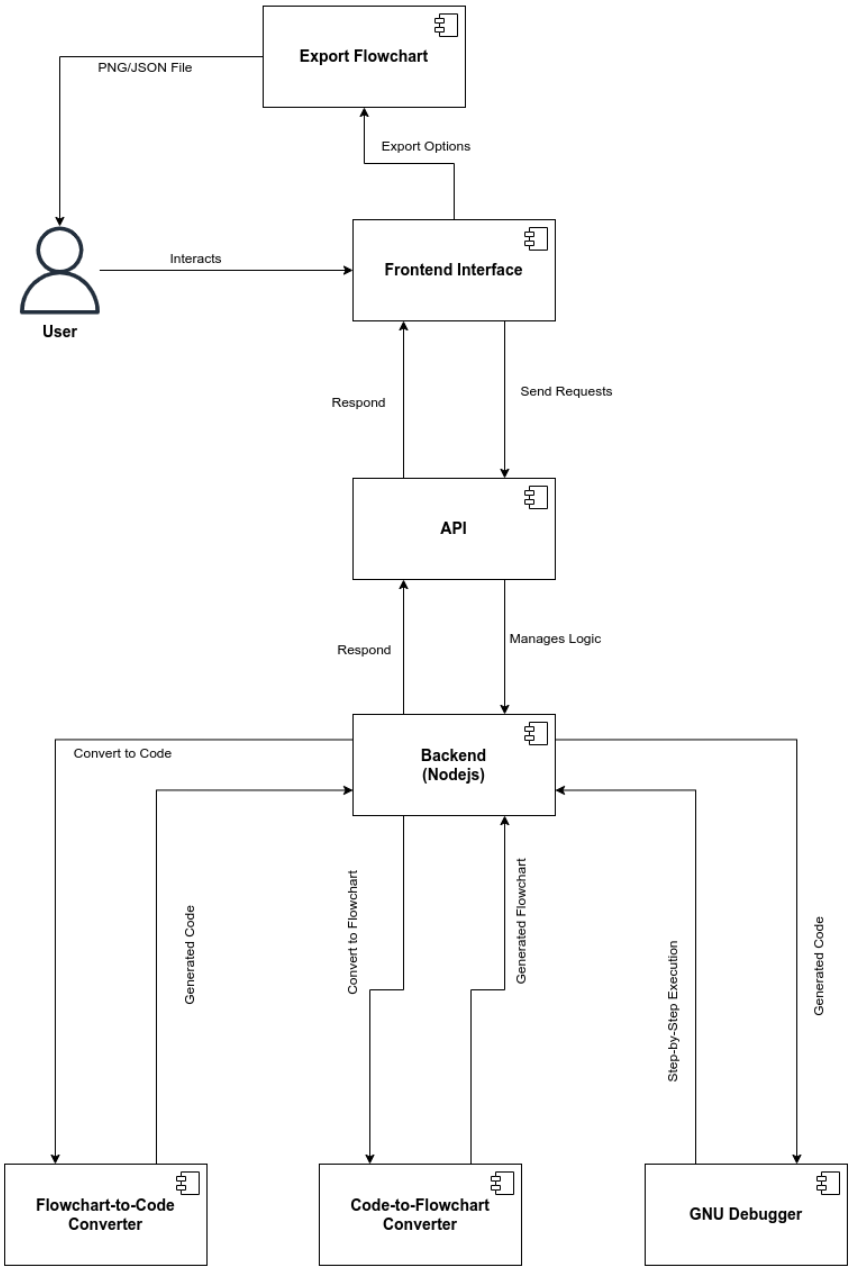
2.4 Design and Implementation Constraints

Browser Compatibility:
Constraint: CodeFlow's functionality is dependent on web browser capabilities
Mitigation: Prioritize compatibility with widely used browsers, and provide clear recommendations to users regarding optimal browser choices.

Internet Dependency:
Constraint: CodeFlow requires a stable internet connection for working as it is a web based project.
Mitigation: Making the project open sourced so anyone can self host the project.

Device Limitations:
Constraint: Limited screen size and processing power on certain devices (especially tablets and smartphones) might impact the user experience.
Mitigation: Optimize the user interface for smaller screens, ensuring essential features are accessible and functional on various devices.

3. Technical Architecture



Is the system custom-built? COTS?

- CodeFlow is a custom-built system, incorporating the React Flow library and GDB as integral components of its unique architecture.

What type of processing is the current system responsible for?

- The CodeFlow system primarily handles online processing with a focus on transaction processing. It is designed to interactively process user inputs such as flowchart creation and code generation in real-time, providing immediate feedback and results.

What are the major application components?

- **Frontend Interface (React):** Manages user interactions, flowchart creation, and display. It provides the drag-and-drop interface for building and editing flowcharts.
- **Backend (Nodejs):** Handles the logic for converting flowcharts to code and vice versa, managing GDB sessions for step-by-step code execution, and processing API requests from the frontend.
- **GDB Integration:** Facilitates the step-by-step execution of the generated C++ code, allowing users to observe the flow of execution in correspondence with the flowchart.
- **API :** Serves as the communication bridge between the frontend and backend, handling data transfer and command execution requests.
- **Local Storage Management:** Manages saving and retrieving user data, flowcharts, and code snippets locally, as there is no database integration.

What data does the current system collect and manage?

- The system does not manage persistent user data or profiles as it does not integrate a database.

What is the basic application architecture (layered, client/server, etc.)?

- The basic application architecture of CodeFlow is a client-server model.

What programming language is the current system built in?

- **JavaScript (with React):** Used for developing the frontend interface, including the integration of the React Flow library for flowchart functionality.
- **Node.js:** Likely used for the backend server development, especially considering the integration with a JavaScript-based frontend.

  C++: Utilized in the context of GDB for the generation and step-by-step execution of C++ code from flowcharts.

What is the hardware platform that supports the current system?

- The CodeFlow system is web-based, which means it primarily relies on the user's device with a modern web browser to access and interact with the platform. There are no specific hardware requirements for the user's device beyond having internet access and a compatible web browser.

What database platform supports the current system?

- CodeFlow system does not rely on a traditional database platform for data storage. Instead, it utilizes JSON files for saving and exporting project data. Therefore, there is no specific database platform supporting the current system, as it operates without a database backend.

Does the system have an end-user interface? If so, what type of user interface? (e.g., browser based, thick client)?

- Yes, the CodeFlow system has an end-user interface. The user interface is browser-based, which means it is accessed and interacted with through a web browser on the user's device.

What is the basic network architecture (e.g., available on LAN, WAN, Internet)?

- The basic network architecture for CodeFlow is designed to be accessible over the Internet. It is a web-based platform that users can access from any location with an internet connection.

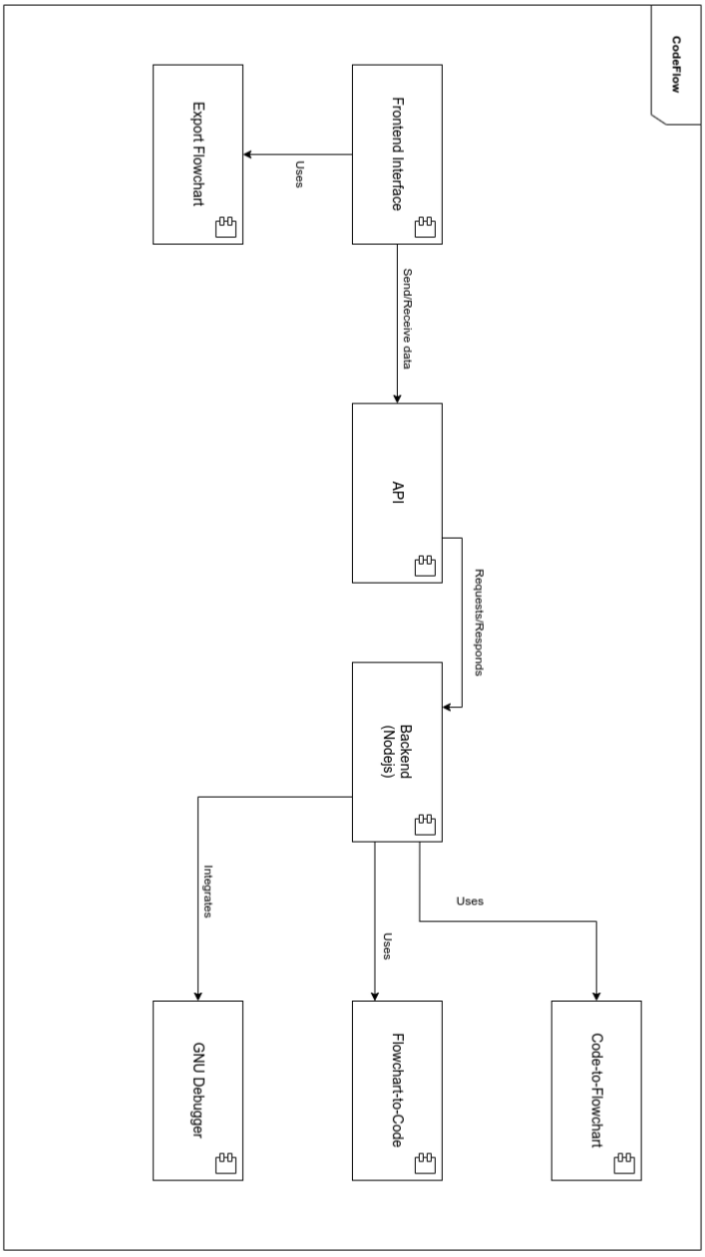
Where is the system hosted (e.g., Enterprise Data Center, Other CMS Data Center, External Data Center)?

- The CodeFlow system is not hosted in a specific location. Instead, it is an open-source project that can be self-hosted by users. This means that users have the flexibility to deploy and run the application on their own computers according to their preferences and needs. The CodeFlow codebase and resources will be available for users to set up and host independently.

3.1 Application and Data Architecture

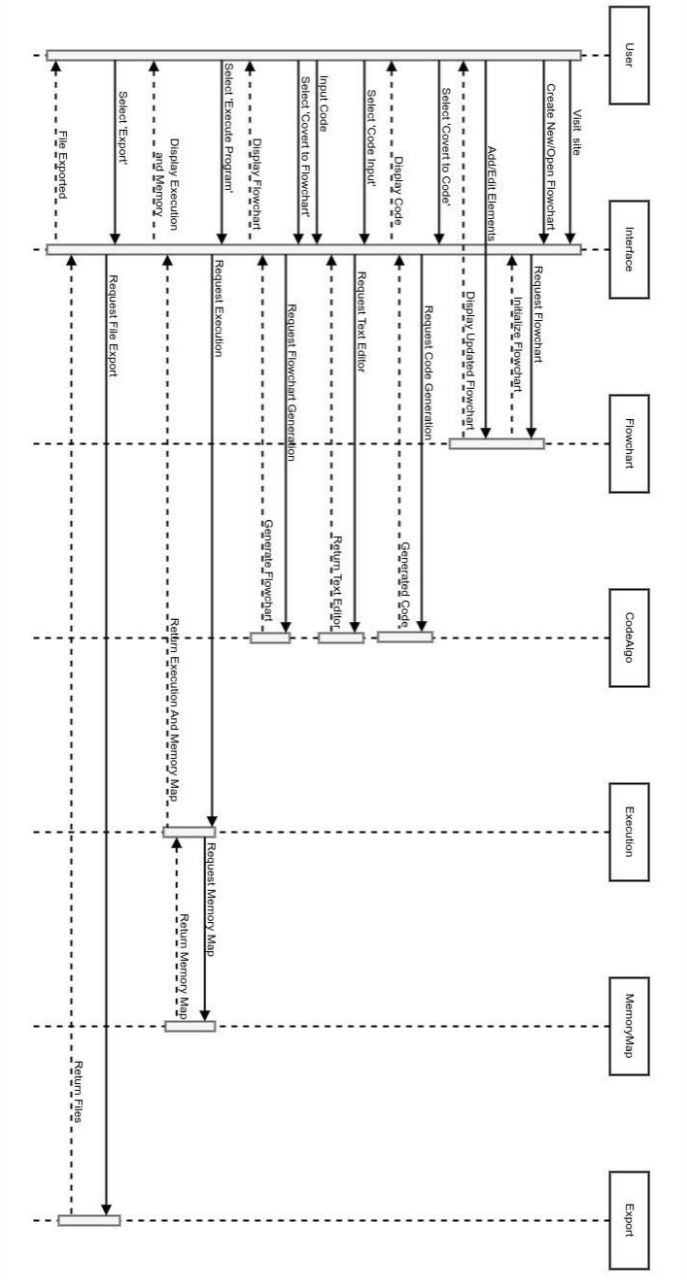
User: The primary actor who interacts with the system, utilizing the various features provided by the application.
Frontend Interface: This is where the user interaction takes place. The user can create, edit, and interact with flowcharts using a drag-and-drop interface.
Export Flowchart: A feature that allows the user to export the created flowchart as a PNG image or a JSON file for future use or editing.
API: Acts as the intermediary that facilitates communication between the frontend interface and the backend server. It sends user requests to the backend and delivers responses back to the frontend.
Backend (Node.js): The server-side component that manages the core logic of the application, including data processing and the execution of various functionalities such as conversion algorithms and debugging processes.
Flowchart-to-Code Converter: A component that converts the user-created flowchart into executable code, likely C++ in this context.
Code-to-Flowchart Converter: This component takes existing code and converts it into a visual flowchart representation, aiding users in understanding the code structure.
GNU Debugger: Responsible for running the generated code step by step and providing insights into the memory usage of the program. It allows users to inspect the state of the application during execution, which includes observing variable values, the call stack, and the memory consumption of the program at various execution points.

Component Diagram:

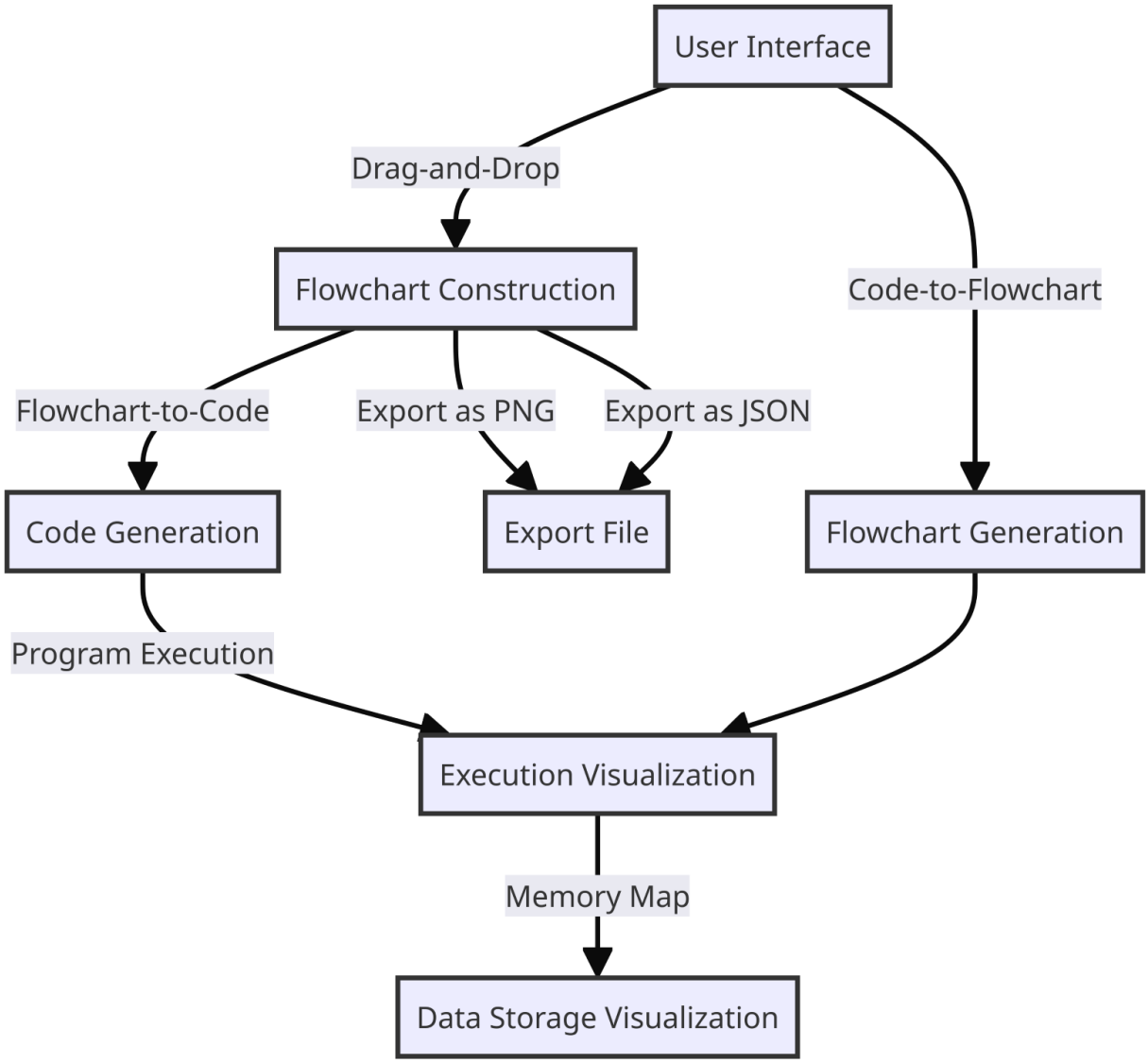


3.2 Component Interactions and Collaborations

Sequence Diagram:



Dataflow Diagram:



3.3 Design Reuse and Design Patterns

ReactFlow Library: The utilization of this third-party library for building the flowchart interface is a significant example of reuse. It provides some basic components and functionalities that save development time and ensure a reliable user experience.

GDB Integration: Using the GNU Debugger for step-by-step code execution and memory map is an example of tool reuse, as it leverages an existing, robust solution for debugging.

3.4 Technology Architecture

The anticipated infrastructure to support the CodeFlow application would involve a cloud-based, serverless architecture that leverages managed services to minimize the need for infrastructure maintenance.

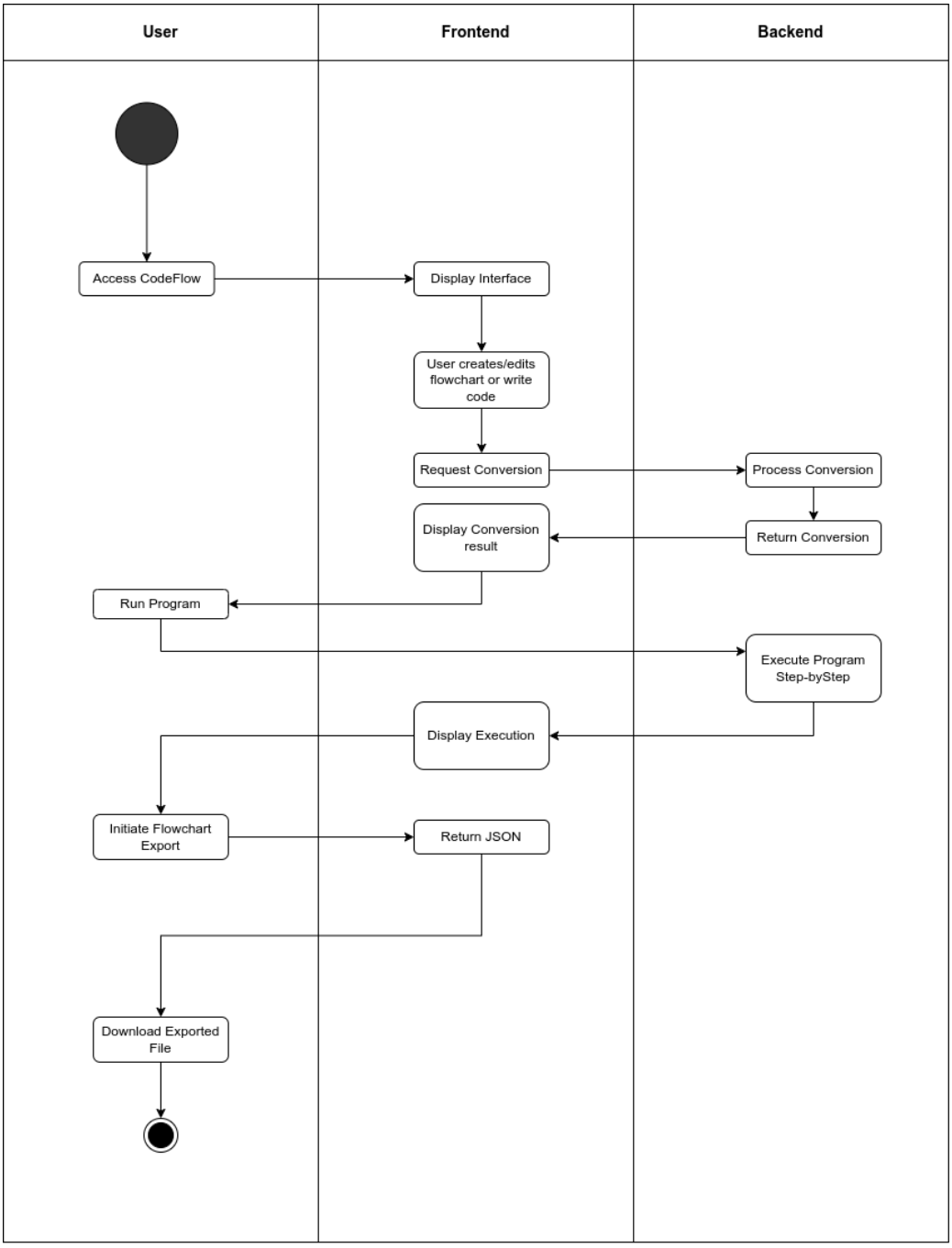
Frontend Hosting
Content Delivery Network (CDN): Use a CDN to serve the static frontend assets of the Next.js application for high availability and low latency.
Managed Hosting Service: Platforms like Vercel or Netlify, which are optimized for hosting Next.js applications and provide features like server-side rendering and static generation.

Backend Services
Function as a Service (FaaS): Utilize serverless functions (e.g., AWS Lambda, Google Cloud Functions) for backend logic. This includes flowchart-to-code conversion, code execution with GDB, and other computational tasks.
API Gateway: Deploy an API gateway to manage, authenticate, and route API calls to the appropriate serverless functions.

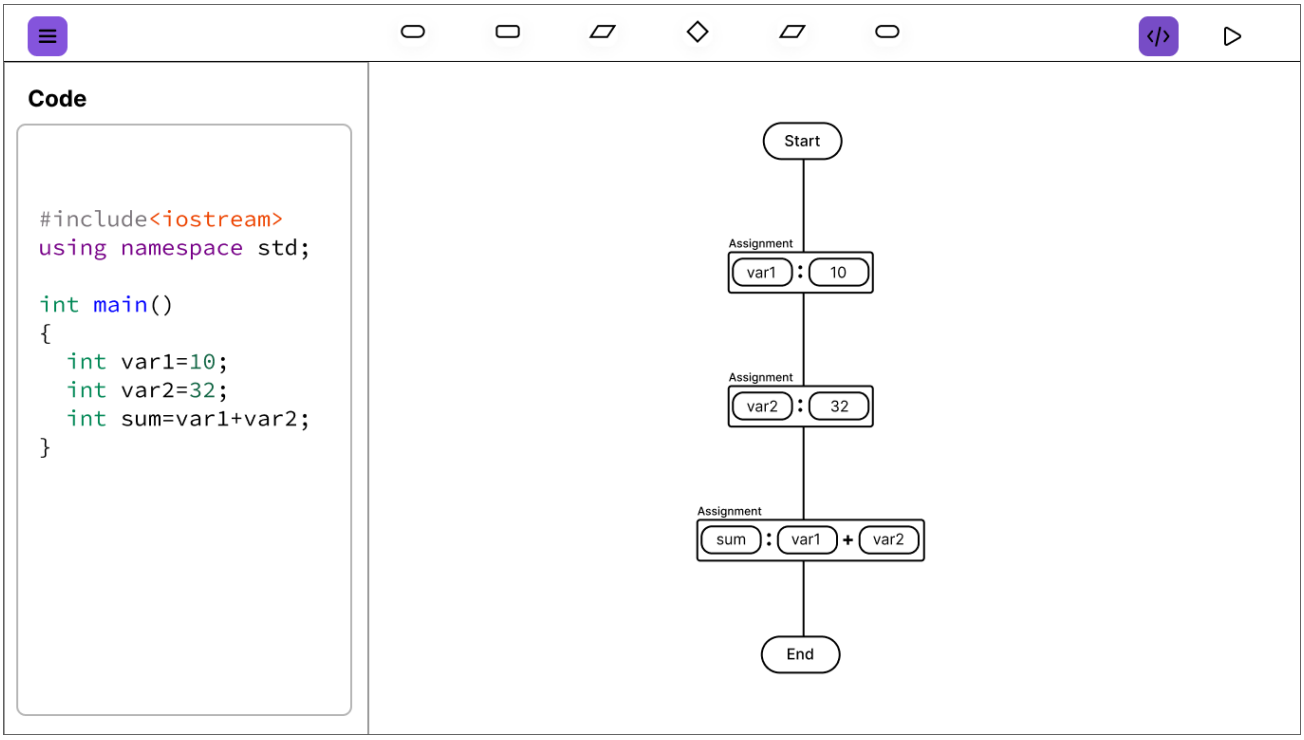
Execution Environment
Containerized Microservices: Use container services (e.g., AWS , Google Cloud Run) to run stateless containers for tasks requiring persistent computation or execution environments, such as GDB sessions.

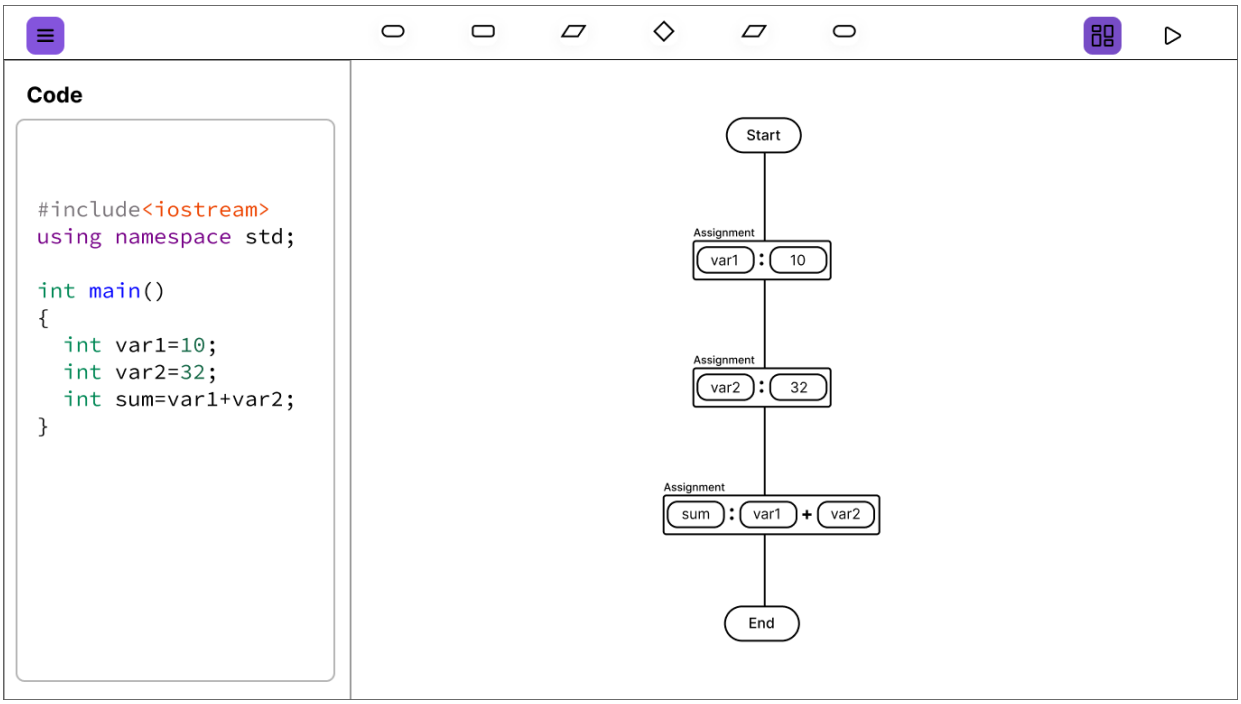
4. Screenshots/Prototype

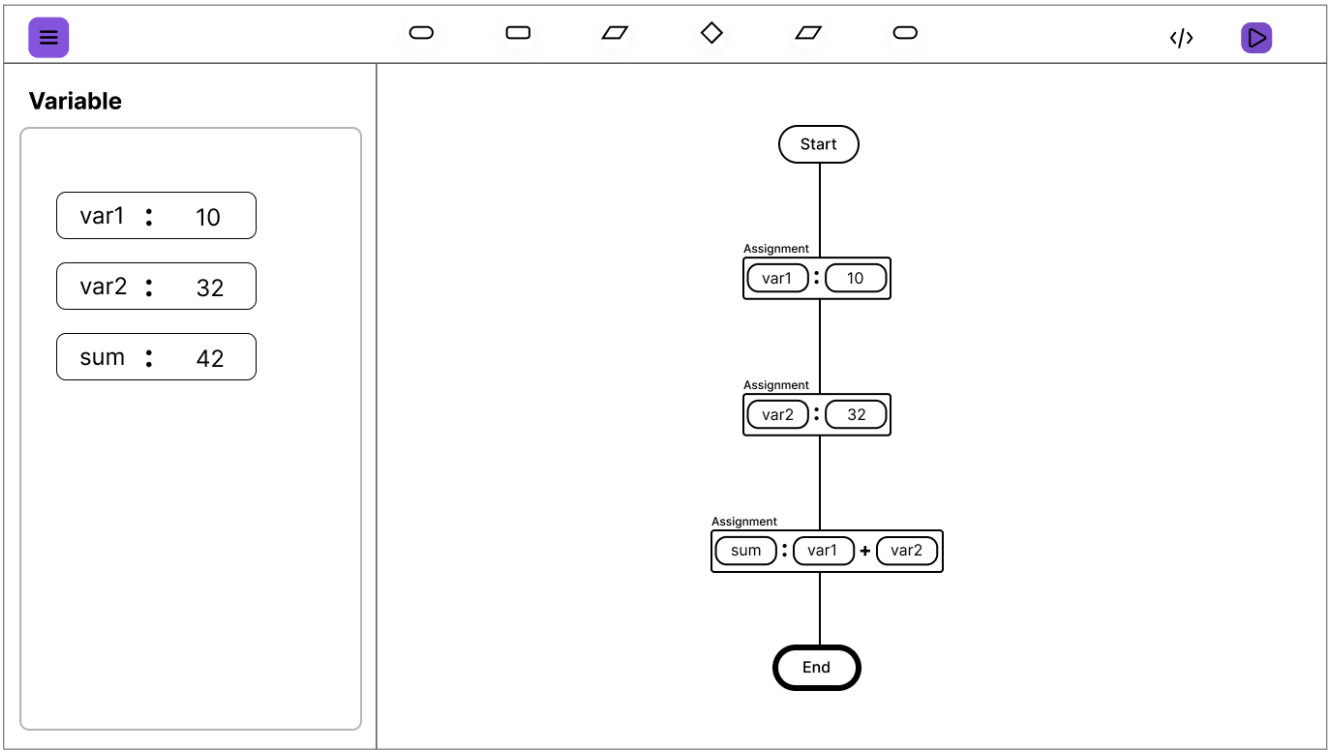
4.1 Workflow



4.2 Screens

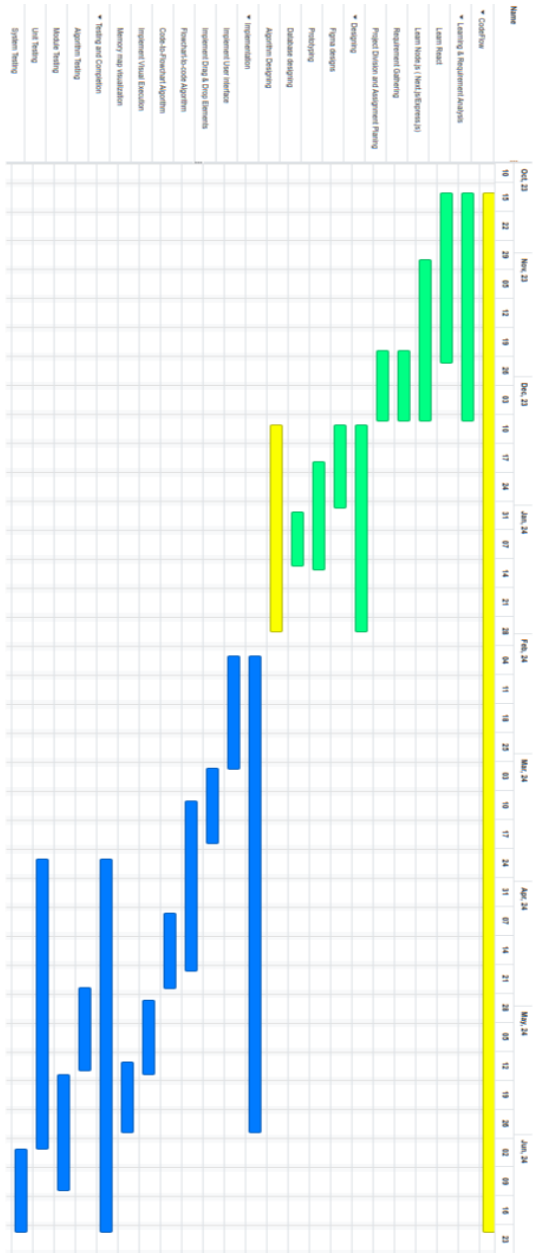

Code
#include<iostream>
using namespace std;
int main()
{
int var1=10;
int var2=32;
int sum=var1+var2;
}
Start
Assignment
var1 : 10
Assignment
var2 : 32
Assignment
sum : var1 + var2
End


Code
#include<iostream>
using namespace std;
int main()
{
int var1=10;
int var2=32;
int sum=var1+var2;
}
Start
Assignment
var1 : 10
Assignment
var2 : 32
Assignment
sum : var1 + var2
End


Variable
var1 : 10
var2 : 32
sum : 42
Start
Assignment
var1 : 10
Assignment
var2 : 32
Assignment
sum : var1 + var2
End

5. Revised Project Plan

Green **Completed**
Yellow **Inprogress**
Blue **Not started**



6. References

Scratch, "Scratch - Imagine, Program, Share," Accessed on: Month Day, Year. [Online]. Available: https://scratch.mit.edu/

MIT OpenCourseWare, "6.00 Introduction to Computer Science and Programming, Fall 2008," [Online]. Available: https://ocw.mit.edu/courses/6-00-introduction-to-computer-science-and-programming-fall-2008/

Raptor, "Raptor - Flowchart Interpreter," [Online]. Available: https://raptor.martincarlisle.com/

Appendix A: Glossary

Not Applicable

Appendix B: IV & V Report

(Independent verification & validation)
IV & V Resource

Name Signature

S#	Defect Description	Origin Stage	Status	Fix Time	
				Hours	Minutes
1					
2					
3					
…					

Table 1: List of non-trivial defects

This document has been adapted from the following:

1. Previous project templates at UCP
2. High-level Technical Design, Centers for Medicare & Medicaid Services. (www.cms.gov)