University of Central Punjab

BSCS FINAL PROJECT
Requirements 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 Requirements Specification

Version <Version #>

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

Table of Contents i
Revision History ii
Abstract iii
1. Introduction and Background 1
1.1 Product (Problem Statement) 1
1.2 Background 2
1.3 Scope 2
1.4 Objective(s)/Aim(s)/Target(s) 2
1.5 Challenges 3
1.6 Learning Outcomes 3
1.7 Nature of End Product 4
1.8 Completeness Criteria 4
1.9 Business Goals 5
1.10 Related Work/ Literature Survey/ Literature Review 5
1.11 Document Conventions 5
2. Overall Description 6
2.1 Product Features 6
2.2 User Classes and Characteristics 6
2.3 Operating Environment 6
2.4 Design and Implementation Constraints 6
3. Functional Requirements 8
3.1 Construct Flowchart 9
3.2 Flowchart-to-Code Conversion 9
3.3 Code-to-Flowchart Conversion 10
3.4 Program Execution 10
3.5 Memory Map Visualization 11
3.6 Export Flowchart 11
3.7 Requirements Analysis and Modeling 12
4. Nonfunctional Requirements 15
4.1 Performance Requirements 15
4.2 Safety Requirements 15
4.3 Additional Software Quality Attributes 16
5. Other Requirements 16
6. Revised Project Plan 16
7. References 18
Appendix A: Glossary 19
Appendix B: IV & V Report 20

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 and Background

In the rapidly advancing digital era, programming skills have become essential for individuals aspiring to participate meaningfully in the technology-driven world. However, the process of learning programming, especially for novice programmers and students, presents significant challenges. The abstract nature of code and the complexities involved in creating logical program flows often act as daunting barriers, hindering the progress of learners. This project, CodeFlow, emerges as a response to these challenges, offering an innovative and user-friendly approach to programming education. The field of programming education has witnessed various attempts to simplify the learning process, but there remains a gap in creating an intuitive and effective learning environment, especially for beginners. Existing solutions, such as Raptor have limitations like they don't translate into code. Additionally, while tools like Scratch provide a graphical interface for basic programming concepts, there is a need for a more comprehensive platform that seamlessly bridges the transition from visual programming to textual languages.

Recognizing these challenges, CodeFlow has been conceptualized. This project is grounded in the understanding that a hands-on, visual approach can significantly enhance the learning experience for novice programmers. By combining a drag-and-drop interface with a robust algorithm for code-to-flowchart conversion, CodeFlow provides a unique platform where learners can construct program logic visually. Furthermore, the integration of memory map visualization adds a crucial dimension, allowing learners to observe how data is stored and manipulated during program execution.

1.1Product (Problem Statement)

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 confident and skilled programmers

1.2Background

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.

1.3Scope

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.

1.4Objective(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.

1.5Challenges

Algorithmic Complexity: Designing an accurate and efficient code-to-flowchart conversion algorithm that can handle a wide range of programming languages and complex logical structures poses a significant challenge. Ensuring the algorithm's reliability and performance is vital for providing users with a seamless experience.

User Interface Design: Creating an intuitive and user-friendly drag-and-drop interface that caters to both beginners and intermediate learners requires careful consideration of user experience design. Balancing simplicity with functionality is essential to prevent overwhelming users, especially those new to programming.

Memory Visualization: Implementing a memory map visualization feature that accurately represents data storage and manipulation during program execution can be intricate. Ensuring real-time and accurate visualization of memory usage poses technical challenges that need to be addressed for a comprehensive learning experience.

Cross-Platform Compatibility: Ensuring CodeFlow's compatibility with various web browsers, operating systems, and devices adds complexity to the development process. The platform must function seamlessly across different platforms to reach a broad user base.

Scalability and Performance: As the user base grows, ensuring the scalability and performance of the platform under heavy user load is critical. CodeFlow must be able to handle a large number of concurrent users without compromising performance or responsiveness.

Security and Privacy: Safeguarding user data, ensuring secure communication, and protecting against potential security threats are paramount. Implementing robust security measures to protect user information and maintain user privacy is a challenge that requires constant vigilance.

Community Engagement: Building and maintaining an active user community around CodeFlow is essential for its long-term success. Encouraging user engagement, gathering feedback, and implementing user-driven improvements pose challenges in community building and management.

Addressing these challenges requires a collaborative effort, involving software developers, educators, and learners. By overcoming these obstacles, CodeFlow can achieve its objectives, providing an enriching and accessible programming learning experience for users**.**

1.6Learning Outcomes

The implementation and utilization of CodeFlow are expected to yield a range of positive learning outcomes for its users, particularly novice programmers and students. These outcomes are designed to enhance understanding, foster creativity, and empower learners with practical programming skills. The learning outcomes of CodeFlow include

Enhanced Understanding of Programming Logic: will gain a deep understanding of programming logic through visual representations, enabling them to grasp complex concepts and logical structures more effectively.

Improved Problem-Solving Skills: CodeFlow's interactive environment encourages experimentation and problem-solving. We can explore different algorithms and logical flows, honing their problem-solving skills by tackling programming challenges in a creative and supportive platform.

Increased Confidence in Coding Abilities: By providing a user-friendly interface and step-by-step visualization of program execution, CodeFlow boosts users' confidence in their coding abilities. Learners can witness their code in action, reinforcing their understanding and encouraging them to tackle more complex programming tasks.

Deeper Insight into Memory Management: The memory map visualization feature allows users to observe how data is stored and manipulated during program execution. This insight into memory management enhances learners' knowledge of variables, data storage, and memory allocation, essential for writing efficient programs.

Facilitated Transition to Textual Coding: For users transitioning from visual programming languages to textual languages, CodeFlow acts as a stepping stone. The platform's visual representations provide a smooth transition, helping users understand how flowchart elements correspond to code structures in text-based languages.

Encouragement of Creativity and Exploration: CodeFlow's intuitive interface and real-time feedback foster a creative learning environment. Users can experiment with different algorithms and program structures, encouraging innovation and exploration in problem-solving approaches.

By achieving these learning outcomes, CodeFlow aims to empower learners with the knowledge, skills, and confidence necessary to excel in programming, preparing them for future educational and professional endeavors in the field of computer science and software development.

1.7Nature of End Product

The end product, CodeFlow, is a sophisticated web-based learning platform tailored for novice programmers and students aspiring to master programming concepts. It embodies several distinctive characteristics that define its nature and value within the realm of programming education.

1.8Completeness Criteria

S.No.	Criteria	Weightage %
1	Web Application	**10**
2	Drag-&-Drop Flowchart feature	**10**
3	Flowchart to code Algorithm	**20**
4	Code to flowchart Algorithm	**20**
5	Step-by-step visual execution	**20**
6	Memory map visualization	**15**
7	Communication	**5**

1.9Business Goals

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

1.10Related Work/ Literature Survey/ Literature Review

Raptor is a flowchart-based programming software designed to simplify coding for beginners. It shares similarities with your project in terms of providing a visual representation of code logic. Raptor's primary approach is based on creating flowcharts, which is beneficial for visualizing program logic. However, this approach may not directly translate to traditional programming languages that predominantly use textual code.

Scratch is a visual programming language developed by the MIT Media Lab. It provides a graphical interface where users can drag and drop code blocks to create interactive stories, games, and animations. Scratch is entirely block-based, making it highly accessible for beginners. CodeFlow, on the other hand, incorporates both textual and visual elements, allowing users to gradually transition to text-based programming languages. Scratch is often used for creating interactive animations and simple games. CodeFlow's focus extends to more comprehensive programming concepts, including memory management, to support a deeper understanding of programming principles.

1.11Document Conventions

Font: Times New Roman

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

2.1Product 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

2.2User 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

2.3Operating Environment

As CodeFlow is a web application by ensuring compatibility with major web browsers we aims to provide a consistent and reliable user experience regardless of the user's operating system or platform preference.

- Google Chrome: Latest version and previous versions.
- Mozilla Firefox: Latest version and previous versions.
- Safari: Latest version for macOS users.
- Microsoft Edge: Latest version for Windows users.

2.4Design 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.

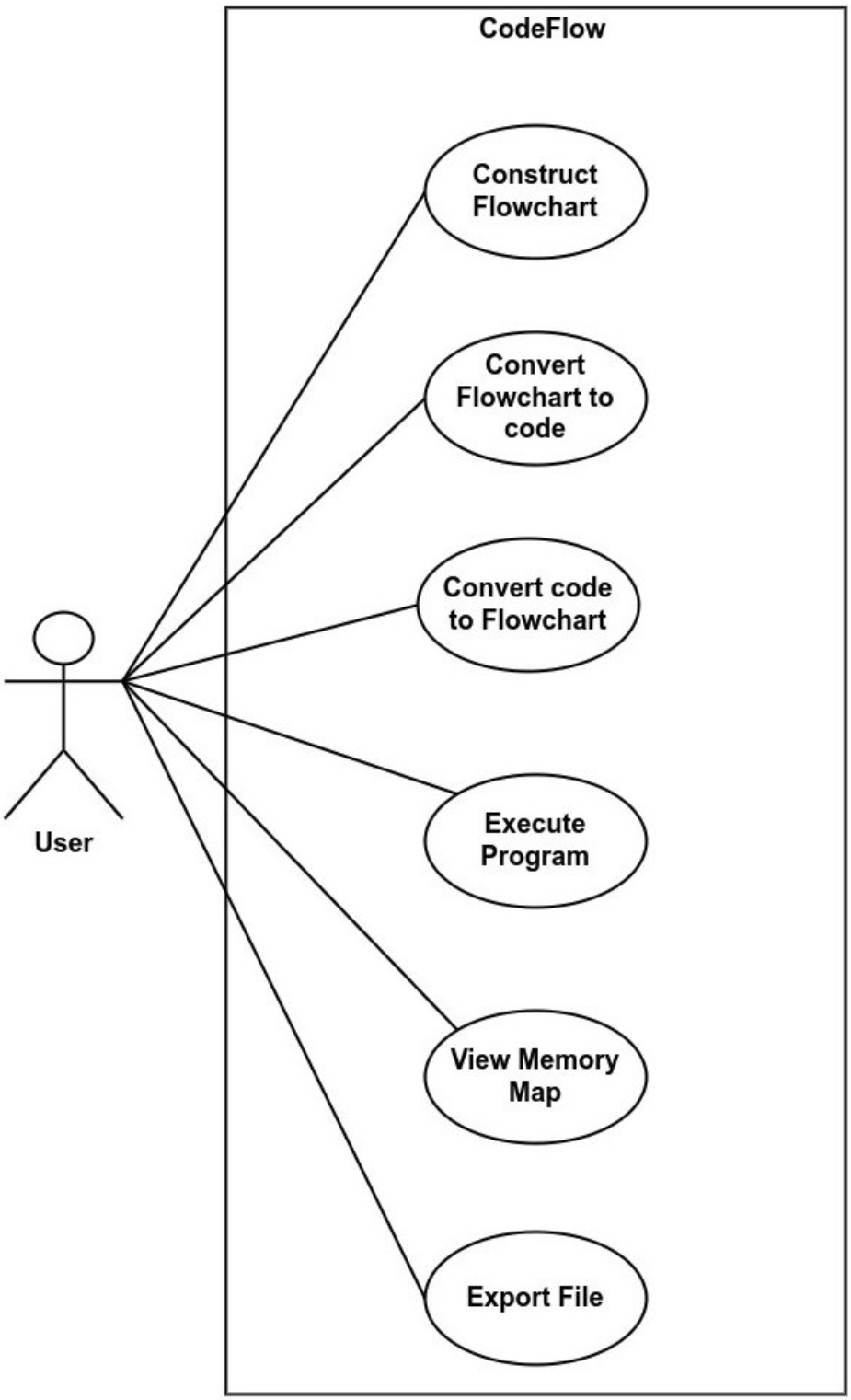
Internet Dependency:

Constraint: CodeFlow requires a stable internet connection for working as it is a web based project.
Mitigation: Making the project opensourced 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.Functional Requirements



3.1Construct Flowchart

Identifier	UC-1	
Purpose	Enable users to visually construct a program's logic using flowchart elements.	
Priority	High	
Pre-conditions	User has accessed the CodeFlow platform and initiated a flowchart creation session by selecting the appropriate option or workspace.	
Post-conditions	A new flowchart is created and available for manipulation and can be saved, exported, or used for further features like conversion to code.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	User begins a new flowchart session	System provides a canvas with a palette of flowchart elements.
2	User selects and drags flowchart elements onto the canvas	System places elements on the canvas and allows configuration.
3	User connects flowchart elements to construct logical flow	System updates the visual representation in real-time and validates logical connections.
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	User tries to connect incompatible elements	System shows an error message or does not allow the connection.

3.2 Flowchart-to-Code Conversion

Identifier	UC-2	
Purpose	To convert a user-created flowchart into executable code.	
Priority	High	
Pre-conditions	User has created a complete flowchart on the platform All flowchart elements are correctly connected.	
Post-conditions	The system generates the corresponding textual code of the flowchart. The user can view, edit, or execute the generated code.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	User selects 'Convert to Code' option	System verifies flowchart completeness
2	System validates flowchart logic	System generates corresponding code
3	User views the generated code	System provides options to edit or export the code

Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	Flowchart is incomplete or has errors	System notifies and highlights errors

3.3 Code-to-Flowchart Conversion

Identifier	UC-3	
Purpose	To translate textual code into a visual flowchart for enhanced understanding.	
Priority	High	
Pre-conditions	User has textual code ready for conversion. The code is correct	
Post-conditions	A flowchart representing the textual code is generated User can interact with the visual representation.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	User selects 'Code input' option	System display a code editor
2	User inputs code into the system	
3	User selects 'Convert to Flowchart' option	System parses the code for conversion
4	System processes the code and generates a flowchart	System displays the flowchart
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	Code contains errors	System highlights errors and prompts for correction

3.4 Program Execution

Identifier	UC-4	
Purpose	To offer a step-by-step execution of programs to trace logic and identify errors.	
Priority	Medium	
Pre-conditions	User has a flowchart ready for execution.	
Post-conditions	Execution flow is visualized for the user.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	User initiates program execution	System check for flowchart completion Generate code if not generated
2	System start highlighting each step	
3		
Alternate Course of Action		
S#	**Actor Action**	**System Response**

1	Error encountered during execution	System pauses and highlights error
2		

3.5 Memory Map Visualization

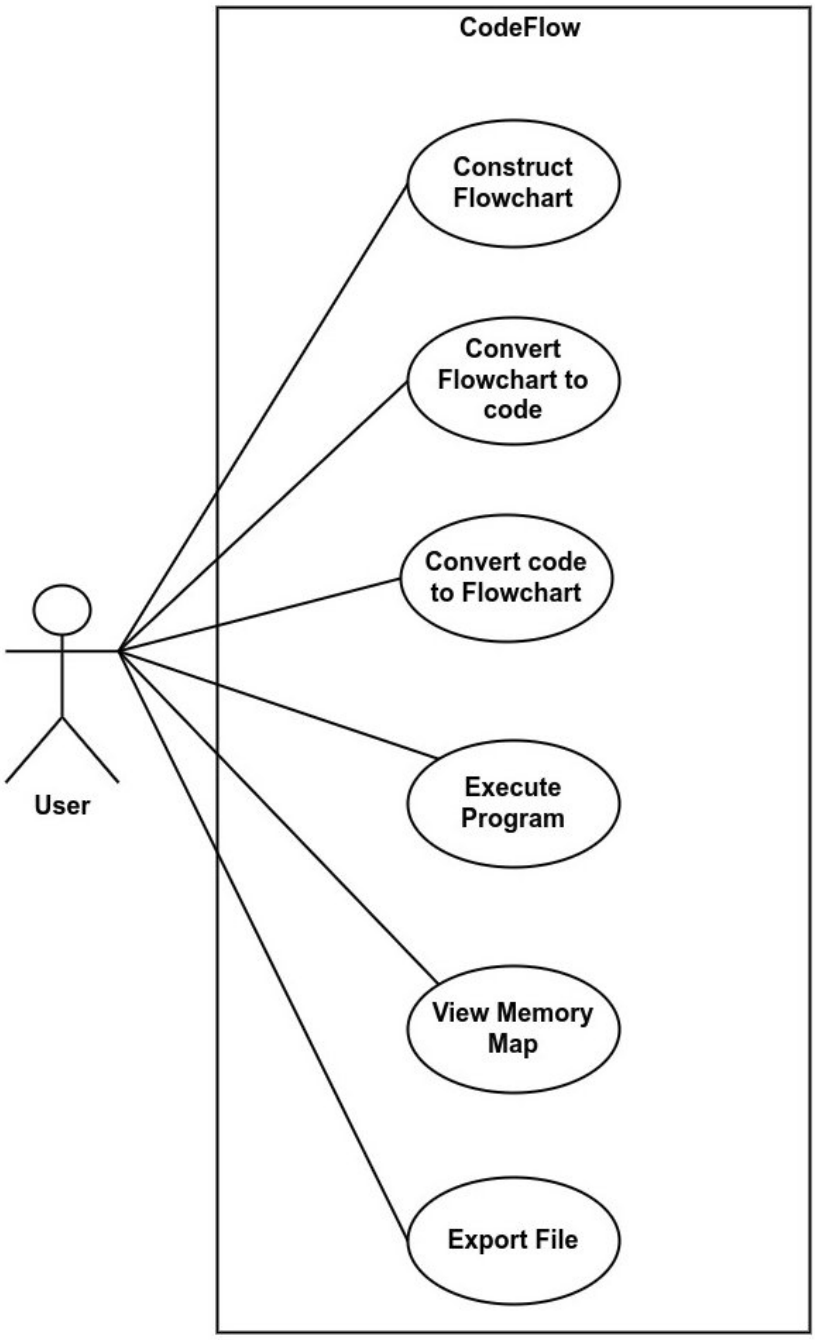
Identifier	UC-5	
Purpose	To visually demonstrate data storage and manipulation during program execution	
Priority	Medium	
Pre-conditions	Program is executing.	
Post-conditions	Memory allocation and de-allocation are visualized.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	User initiates program execution	System check for flowchart completion Generate code if not generated
2	Data storage or manipulation occurs	System updates memory map accordingly
3	User explores memory map	System displays detailed information
…		
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1		

3.6 Export Flowchart

Identifier	UC-6	
Purpose	To enable users to export their flowcharts for future editing.	
Priority	Medium	
Pre-conditions	User has created a flowchart on the platform	
Post-conditions	Files are exported for user access.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	User selects the export option	System prompts for format and destination
2	User confirms export	System exports the file and confirms the action
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	Error during export	System notifies user

3.7Requirements Analysis and Modeling

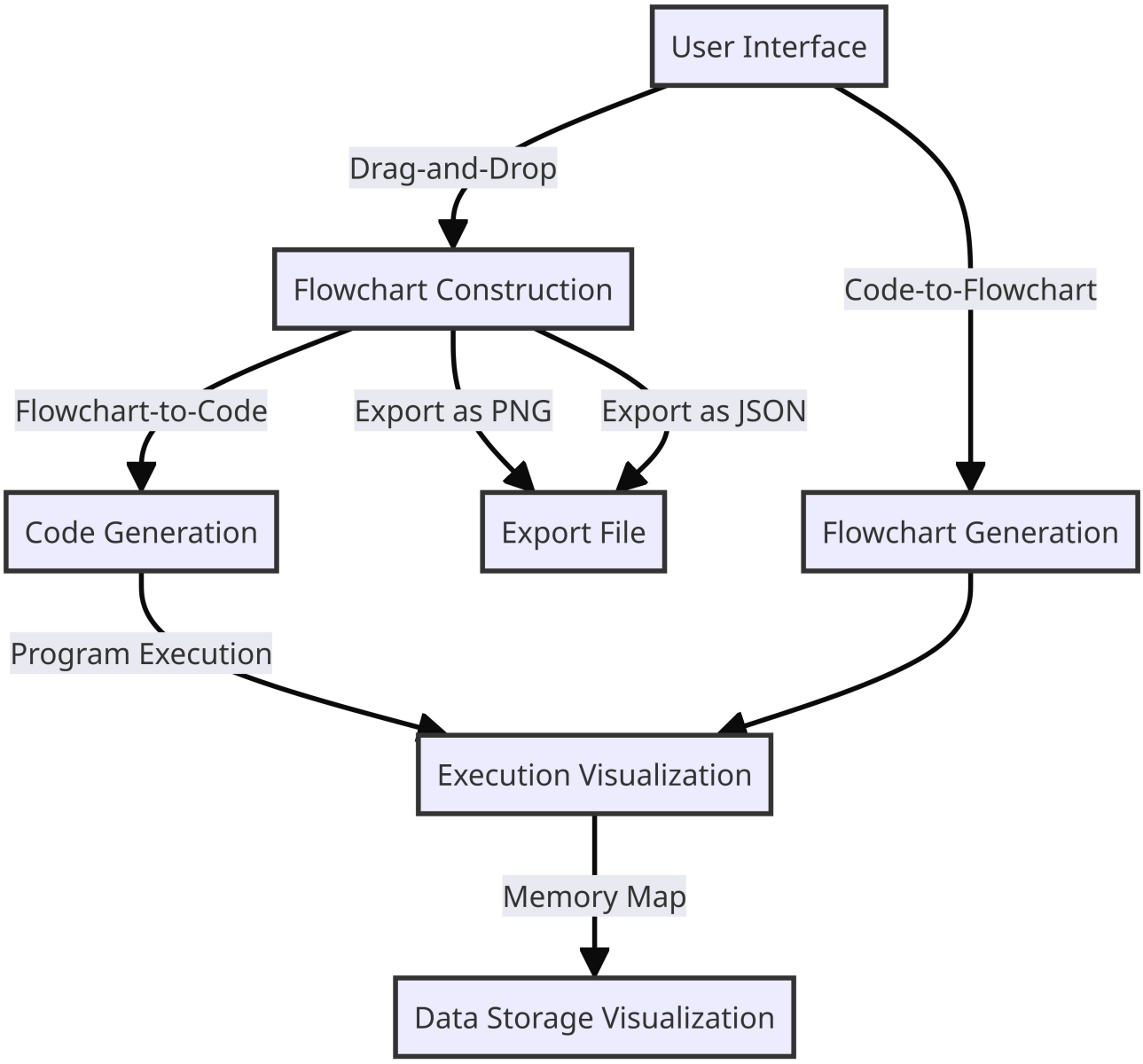
Use Case Diagram



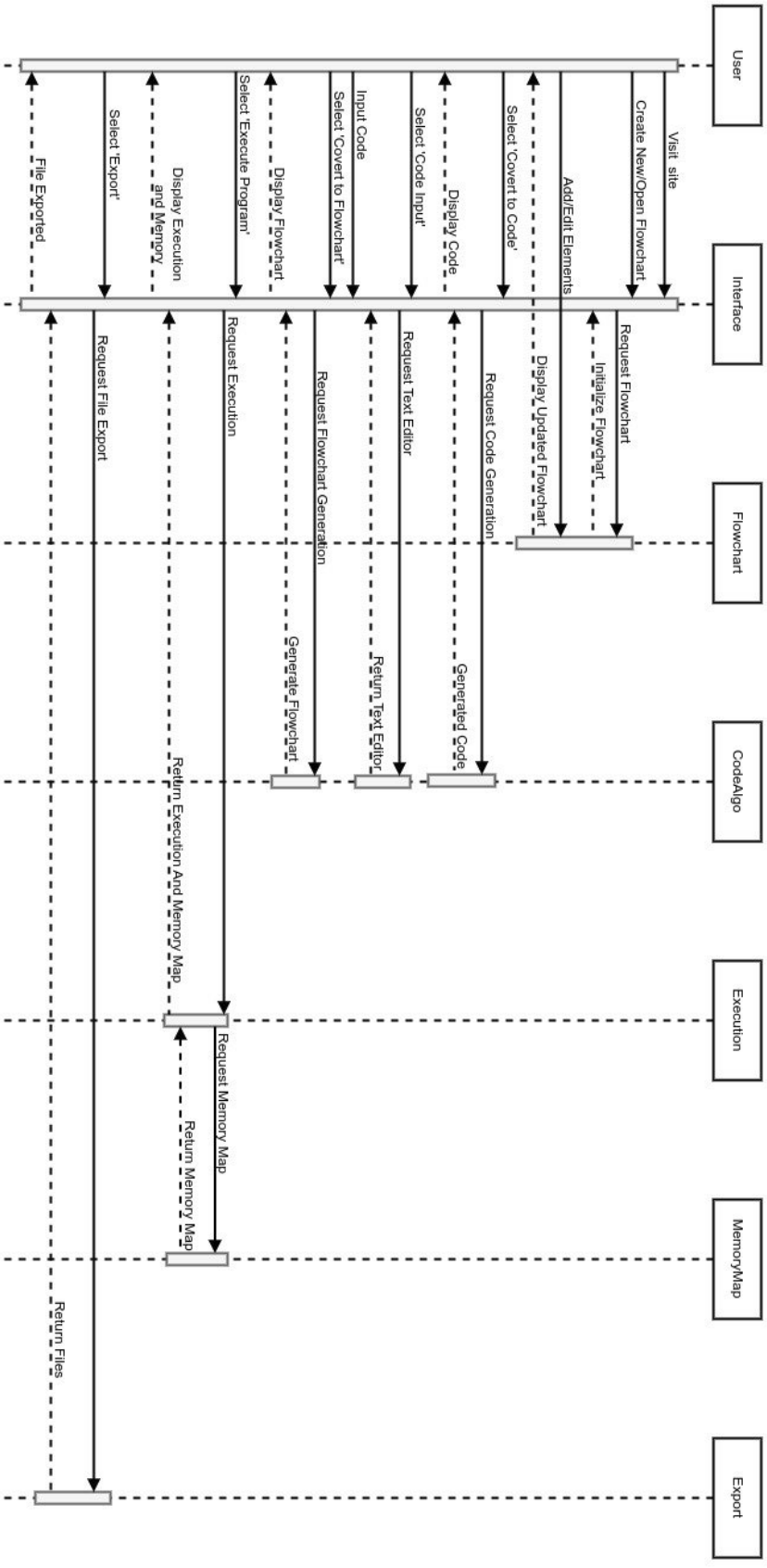
Class Diagram
Not Applicable

ER Diagram
Not Applicable

Data Flow Diagram



Sequence Diagram



4.Nonfunctional Requirements

4.1Performance Requirements

Responsiveness:
CodeFlow must respond to user interactions (e.g., dragging elements, executing code) within milliseconds to ensure a smooth and interactive user experience.

Scalability:
CodeFlow should handle a large number of concurrent users, ensuring consistent performance even during peak usage times.

Load Time:
The platform should load within a maximum of 3 seconds to prevent user frustration and encourage seamless engagement.

Code-to-Flowchart Conversion Speed:
The code-to-flowchart conversion algorithm must process code snippets promptly, generating visual representations within a few seconds.

Error Handling:
Error detection and debugging tools must operate swiftly, allowing users to identify errors in their code efficiently.

Data Security:
Secure data encryption and transmission protocols should be implemented to safeguard user data, ensuring optimal performance while maintaining robust security.

4.2Safety Requirements

Secure File Uploads:
If the platform allows file uploads, implement strict validation and scanning mechanisms to prevent the upload of malicious files.

Data Encryption (if implemented):
User data, including login credentials and personal information, must be encrypted during transmission and storage to prevent unauthorized access.

Secure APIs:
If APIs are used, ensure they are secure, validating user input, and implementing proper authentication and authorization mechanisms..

Error Handling:
Implement robust error handling mechanisms to prevent the exposure of sensitive system information in case of errors or failures.

User Privacy:
CodeFlow should adhere to strict privacy policies, ensuring that user data is not shared with third parties without explicit consent.

4.3Additional Software Quality Attributes

Flexibility:
The platform should be flexible, allowing users to work with various programming languages enabling a wide range of learning experiences.

Collaboration Support:
Implement features supporting collaborative learning, enabling users to work together on projects, share knowledge, and engage in meaningful discussions.

Feedback Mechanisms:
Provide feedback mechanisms for users to report issues, suggest improvements, and provide comments, fostering a sense of community and continuous improvement.

Customizability:
Allow users to customize their learning experience, including interface preferences, color schemes, and personalized Tool bar, enhancing user engagement.

Integration Capabilities:
Integrate with external tools and APIs, enabling seamless data exchange and enhancing the platform's functionality by connecting with relevant third-party services.

5.Other Requirements

Compliance:
CodeFlow should comply with relevant industry standards, programming best practices and integrity in all aspects of its operation.

Documentation:
Provide detailed documentation for developers, educators, and users, covering installation, usage, troubleshooting, and API references, ensuring comprehensive understanding and support resources..

User Support:
Offer responsive user support through multiple channels (email, chat, forums), addressing user queries, issues, and providing timely assistance to enhance user satisfaction.

Comprehensive Testing:
Conduct extensive testing, including unit testing, integration testing, and user acceptance testing, to identify and rectify bugs, ensuring the platform's stability and functionality.

6.Revised Project Plan

Green	**Completed**
Yellow	**In progress**
Blue	**Not Started**


ID
Name
Oct, 23
Nov, 23
Dec, 23
Jan, 24
Feb, 24
Mar, 24
Apr, 24
May, 24
Jun, 24
24 CodeFlow
25 Learning & Requirement Analysis
26 Learn React
27 Learn Node.js (Next.js/Express.js)
28 Requirement Gathering
29 Project Division and Assignment Planning
30 Designing
31 Figma designs
32 Prototyping
33 Database designing
34 Algorithm Designing
35 Implementation
36 Implement User interface
37 Implement Drag & Drop Elements
38 Flowchart-to-code Algorithm
39 Code-to-Flowchart Algorithm
40 Implement Visual Execution
41 Memory map visualization
42 Testing and Completion
43 Algorithm Testing
44 Module Testing
45 Unit Testing
46 System Testing

7.References

Website: https://scratch.mit.edu/

MIT OpenCourseware - Scratch Programming: https://ocw.mit.edu/courses/6-00-introduction-to-computer-science-and-programming-fall-2008/

Website: 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 3: List of non-trivial defects