BackDev

(Backend Development through Visual Programming)

By

NC Abdullah

NC Muhammad Uzair

NC Muhammad Umar bin Ejaz

NC Rabeea Aftab

Supervised by:

Dr. Tauseef Ahmad Rana

Submitted to the faculty of the Department of Computer Software Engineering,
Military College of Signals, National University of Sciences and Technology,
in partial fulfillment of the requirements of a B.E Degree in Software Engineering.

June 2022

In the name of Allah, the Most Beneficent, the Most Merciful

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report

"BackDev"

is carried out by

Muhammad Uzair, Muhammad Umar bin Ejaz, Rabeea Aftab and Abdullah

under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelor of Software Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad.

Approved by

Supervisor

Dr. Tauseef Rana

Date: _____

DECLARATION OF ORIGINALITY

We hereby declare that no portion of the work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains. We are thankful to Allah Almighty for giving us strength and courage for completion of this project despite all the challenges and troubles.

We are thankful to our parents and families. Their constant support and prayers made this work of enormous effort easy for us. Without their support this would not have been possible.

Our special thanks to our Supervisor Assistant Professor Dr. Tauseef Ahmad who supervised the whole process of development in a very guiding and encouraging manner. His support and guidance at times of challenges was always an asset for us.

We are grateful to our college faculty and our instructors who made us capable enough to develop this project and complete it in a timely manner. We are thankful to our course who helped and encouraged us in any way they could.

And of course, special acknowledgement for all the members of this group who tolerated each other so well throughout the period of project.

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for plagiarism. Turnitin report endorsed by Supervisor is attached.

_____

Muhammad Uzair

00000268978

_____

Abdullah

00000251175

_____

Muhammad Umar Bin Ejaz

00000261131

_____

Rabeea Aftab

00000261645

_____

Signature of Supervisor

ABSTRACT

Backend development has always been complex and involves a multitude of nuances that are hard for beginners to understand. Our project aims to simplify these complexities by providing a tool to automate many of the tedious, repetitive, and overly complicated problems in backend development. The project enables the user to develop the backend using a flow-based programming paradigm. The developer will design the process of his program on the canvas by dragging and dropping different operators. Each operator has certain functionality, e.g., in low-level operators all programming language constructs are available as operators. Also, there are high-level operators of most common functionalities, e.g., File manipulation, database interactions, requests, responses, etc. The intended users of this software are backend developers. This software is a prototype automation tool where developers can develop backend applications with less effort, i.e., more reusability is guaranteed, and the code of backend application will be generated by this IDE under the hood.

Table of Contents

List of Figures

List of Tables

Acronyms

NPM – Node Packet Manager

GUI – Graphical User Interface

JS – JavaScript

REGEX – Regular Expressions

1. Introduction

The Technology has revolutionized our way of living. So, with the advancement in technology, it is a need of time to get our lifestyle and needs updated with the technology requirements. The solution implemented will help to tackle the headache of developing the backend of application with hard coding.

With this increasing trend of automation, we have developed the tool that will generate code through visual programming. The code will be based on NodeJS. The developer or any programming who is familiar with NodeJS can use this tool in order to save their time and resources. The proposed solution will be helpful in future as anyone can integrate any other language instead of NodeJS, they can do it.

1.1. Overview

Backend development has always been a tedious and complicated task that is hard for beginners to get into. Especially when learning the nuances of backend development are constrained by the huge learning curve of tools required to develop the backend. Over time, many of the processes involved in backend development have been simplified however, there is still much left which discourages people from getting into this part of web development. Our project aims to simplify this process so that developers and especially beginners can get into the development of the backend without worrying about the tools required and the overall configuration of these tools. With BackDev the user can easily develop their program by using drag and drop. Also, with the provided operators of common functionalities the user can reduce redundancy in their program and reuse many components easily. The main idea behind BackDev is to help backend developers to do the development in a new and interactive way. The developer will design the processes of the application on the canvas by dragging and dropping operators instead of hard coding.

1.2. Problem Statement

The backend is a key component of every web application, whether big or small. However, it is still overly complicated and has much room for improvement. Some of the problems in backend development are as follows.

- No dedicated automation tool available for backend development.
- Overly complicated processes which can easily be simplified. Need to reduce the overall complexity of backend development.
- Repeated tasks and redundancy which can be reduced.

1.3. Proposed Solution

We propose that there should be an application which will make it easy to perform these tasks of backend development that are very complex and time consuming. The solution will be time saving and helpful for the backend development who knows the basics of NodeJS. The major goal of our proposed solution is to develop a desktop-based IDE which provides:

- Functionality to develop the backend using GUI (Drag and drop).
- Generates Node.js code.
- Built-in components for common functionalities in backend development.
- Ability to create new and reusable components.

1.4. Purpose

This thesis aims to provide enough detail about a system's design to enable software development to proceed with a clear understanding of what will be built and how it will be built. This document contains critical information about the software.

1.5. Scope

This project aims to reduce the complexity of and provide an easy approach to backend development. The backend developer will design the process on canvas by dragging and dropping operators. Each operator has certain functionality, e.g., low-level operators all programming language constructs will be available as operators. Also, there will be high-level operators of most common functionalities, e.g., File manipulation, database interactions, requests, responses, etc. The intended users of this software are backend developers.

1.6. Deliverables

Table 1.1: Deliverables

Sr	Tasks	Deliverables
1	Literature Review	Literature Survey and Feasibility Analysis
2	Requirements Specification	Software Requirements Specification document
3	Detailed Design	Software Design Specification document

1.7. Relevant Sustainable Development Goals

This project falls under the UN defined SDG's that are 'Industry, Innovation and Infrastructure' and 'Decent Work and Economic Growth' – this solution is intended to provide tool that is innovative and very helpful for the industry. Also, project contains decent work and will be helpful in the economic growth of the country.

2. Literature Review

Evolutions are made day by day and technology is getting towards easing the problems daily. The same thing applies to the long coding process and challenges faced in that area to develop a product. Coding especially for the backend is a long, hectic process that requires the ultimate effort of developers.

Therefore, to ease the process, this project is focused to help developers in new advanced ways. Leaving all that old-school coding methodologies apart, this innovation is focused on allowing users to generate their code easily using visual programming.

2.1. Problem Domain

Back-end coding is harder and trickier overall. Diving into it, it involves repetition of the same code again and again. It's challenging along with time taking for developers. It requires the effort of the person who is working on it.

2.2. Shortcomings/Issues

Coding and development at the backend is a long, time-consuming process that requires you to work on various frames along with expertise in knowledge of many domains. Not only this, but the work there usually reflects itself like a jumbled-up line of codes with a lot of repetition. Understanding the long code there is a challenging task.

2.3. Proposed Project

Looking at the issues discussed, this project proposes a new adaptive way for developers and coders that allows them to code easier. Now without repetition, they can present and work on their product and sites more easily. This project aims toward user convenience.

3. Software Requirements Specification

This document covers the software requirement specifications for our Final Year project BackDev. The idea is to create an Integrated Development Environment (IDE) for backend development of web applications. It is based on "flow-based programming" and "visual programming" concepts and provides an easy way of backend development. The development in this IDE will be done in such a way that developer will design the processes of application on the canvas by dragging and dropping operators. This software is a kind of automation tool where developers can develop backend applications with less effort.

3.1. Overall Description

3.1.1. Product Perspective

The main idea behind BackDev is to help backend developers to do the development in a new and interactive way. The developer will design the processes of the application on the canvas by dragging and dropping operators instead of hard coding. The sidebar will be opened for that respective component. The developer will set all the parameters of the component in a sidebar. The command prompt will be running in the background and while importing the NPM packages, they will be auto-installed in the background.

The goal of the BackDev is the development of a desktop-based prototype IDE that will provide the ability to develop web backends using a flow-based programming paradigm. Also, there is a tool available with the name of NoFlo which addresses the backend development like our software. But our software BackDev has its own set of requirements, working principles, and usability and will focus on a "developer first" approach to make it easy for developers to adapt to this environment.

3.1.2. Product Functions

- The user shall be able to drag a certain module and drop it onto the editor
- The system shall have a section where we can enter the different parameters in the selected module

- The system shall provide a help section where the information for the selected module will be displayed
- The system shall provide different functions of express JS library like routing, database connectivity, etc.
- The system will allow custom programming of NodeJS through low-level operators.
- The system will provide an interface to install NPM modules.

3.1.3. Operating Environment

BackDev is based on client-server architecture, The backend will be hosted on the web or local machines and the front end will be used on all major web browsers on any operating system

3.1.4. Design and Implementation Constraints

This software does come with its design and implementation constraints:

- The prototype will be used for some forms of special operators only.

3.1.5 User Documentation

The user will be able to use the following as guides for using the software:

- User Manual that contains textual and pictorial help for users in guiding them to use the software correctly and troubleshoot it.
- Project synopsis that will be used to understand the features and the constraints of the software prototype.
- A webpage in the website interface for the software that answers frequently asked questions and has a guide on using the software.

3.1.6. Assumptions and Dependencies

- BackDev resides in a physically controlled access facility that prevents unauthorized physical access, and each user has a subscription.
- There are one or more competent individuals assigned to manage BackDev.

- Authorized administrators who will manage BackDev are non-hostile and are appropriately trained to use, configure, maintain the software, and follow all the guidance.

3.2. External Interface Requirements

3.2.1. User Interfaces

The front-end user interfaces will have the following main screens available to the user:

- **Launch Screen —** Once the application is started, the first page that is displayed is the launch menu which gives a prompt to create a new project or to open any existing project that was done previously.
- **Main Screen —** This will be the main interface of our prototype where different components will be joined together to create the backend application.
- **Operator Interface —** On the left side, there will be an interface of different operators like programming constructs, database connectivity, routing, etc.
- **Parameter Interface —** On the top right side, there will be an interface of the parameter in which after clicking the operator, you will be able to set the different parameters of operators for routing you can set the name of the router, location of the router and for the database, you can set the name of the database, column names, data types, etc.
- **Help Interface —** On the bottom right corner, there will be an interface for the help portion. If the user does not know how to use any functionality or has any queries, the help option can be used. The help screen contains a text field to enter search terms. A list of search results for the query is displayed.
- **NPM Interface —** There will also be a portion for installation of npm packages and libraries from which users can install different packages according to their needs.

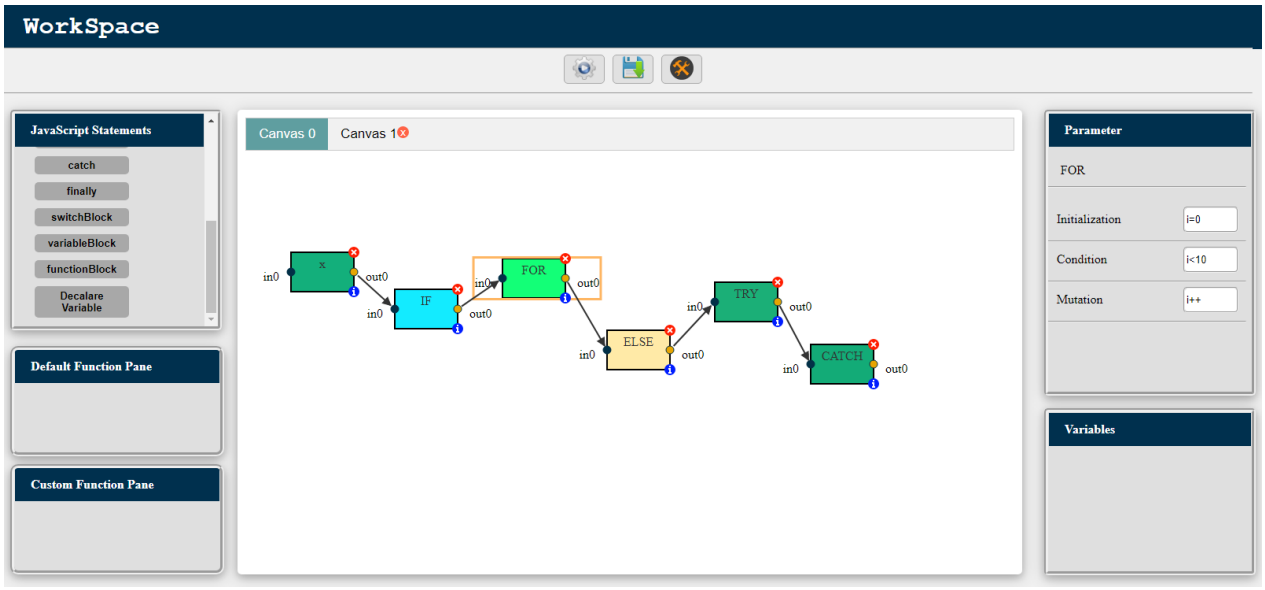

Figure 3.1: Workspace

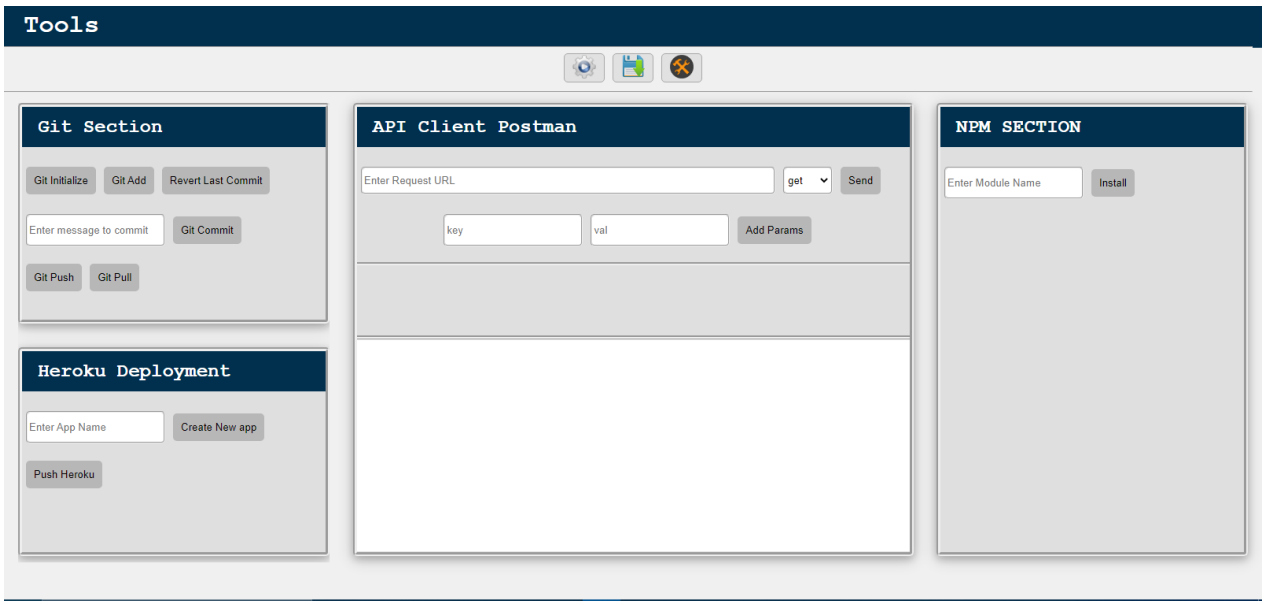

Figure 3.2: Tools

3.2.2. Hardware Interfaces

Since this application is not a resource-intensive application so it can be used on the hardware of average computation power.

3.2.3. Software Interfaces

- **Operating System —** The prototype IDE will run on every platform like Linux, Windows, Mac OS, etc.
- **Browser —** Any modern browser can run the application frontend.
- **NodeJS Environment —** NodeJS environment is needed to run the application backend.

3.2.4. Communications Interfaces

The communication between the front-end and back-end of the application will be based on HTTPS protocols.

3.3. System Features

This section describes in detail the system features of the system.

3.3.1. Visual programming Interface:

The system will provide a visual programming interface for JavaScript (NodeJS) enabling users to create backend applications by configuring functional modules using GUI based operators, allowing developers to design the processes of application on the canvas by dragging and dropping operators

3.3.2. Objects/Blocks:

The system shall provide graphical objects (blocks) of JavaScript programming constructs. The user shall drag the block into the canvas, the editor will then ask the user for parameters and generate

code. Every object/block shall have connectors that can be used to connect it according to the user requirement

3.3.3. Server Management:

The system shall provide an interface to graphically manage all the server configurations. This includes middleware management, database connection, socket management, etc.

3.3.4. Block Library:

The system shall provide a library of blocks, where frequently used functions such as database queries, encryption/decryption, authentication, etc. will all be provided as draggable objects which will generate code saving the hard labor.

3.3.5. Code Generation

The system will generate JavaScript code according to the logic defined by the user in a visual programming interface

3.3.6. Routes:

The system shall provide an interface to manage routing using the expressjs library.

3.3.7. Sockets:

The system shall provide inbuilt blocks for socket communication using the socket.io library.

3.3.8. Help:

On the fly, help would be available with full documentation of each block

3.3.9. Testing:

There will be an interface to test the generated code using JavaScript testing libraries like jest.

3.3.10. API client:

The system will provide an inbuilt rest API client for server testing like postman

3.3.11. Version Control:

The system will provide inbuilt git support to manage versions of a project

3.3.12. Importing Libraries:

The system will provide an interface to import NodeJS libraries from the npm server.

3.3.13. Deployment:

The system will provide an inbuilt deployment interface to deploy the project

3.3.14. Custom blocks/functions:

The user shall be able to extend the block library by creating custom blocks that can then be reused.

3.4. Other Non-functional Requirements

3.4.1. Performance Requirements

The system should be able to produce desired results in a reasonable amount of time. The system should not have any delays in performing normal operation

3.4.2. Safety Requirements

The system should not disclose any personal information of any user to other users. To prevent data loss in case of system failure, the code will be stored in a separate file in the form of JSON. The system should be able to recover itself from previous crashes and continue the drag-and-drop process. The system should be able to warn about malfunctioning of the system.

The use of the software product has no harm to the users; nor does it have any possibility of loss or damage that might be inflicted however during the use of the application. Users should not hold any eatable or anything that can result in damage to the hardware equipment.

3.4.3. Software Quality Attributes

3.4.3.1. Correctness:

The code generated by the system must be correct and syntax error free.

3.4.3.2. Reliability:

In the event of a failure, the data on the server must stay secure. The system shall be able to work in a normal way after restarting due to an error.

3.4.3.3. Portability:

The system should be able to run on different environments Linux/Windows/Mac. Windows and others which are compatible with the requirements defined for it.

3.4.3.4. Usability:

The interface of the application should be attractive and user-friendly, and it should be easy to use. Also, the system should be open source so that any developers anywhere can play with it to introduce all sorts of cool add-ons and such.

3.4.3.5. Scalability:

Initially, the system will only support JavaScript (NodeJS), but the system will be developed in such a way that additional programming languages can be integrated into the system

3.4.4. Business Rules

The product follows the rules set up by NUST and the SDGs goals of the United Nations

4. Design and Development

The introduction of the Software Design Specification (SDS) document provide overview of the entire SDS with purpose, scope, definitions, acronyms, abbreviations, references and overview of the SDS. The aim of this document is to present, in detail, the functional and non-functional aspects of the project BackDev which will be an Integrated Development Environment (IDE) for backend development of web applications. The detailed descriptions and visualizations of the BackDev are provided in this document.

4.1. Work Breakdown Structure

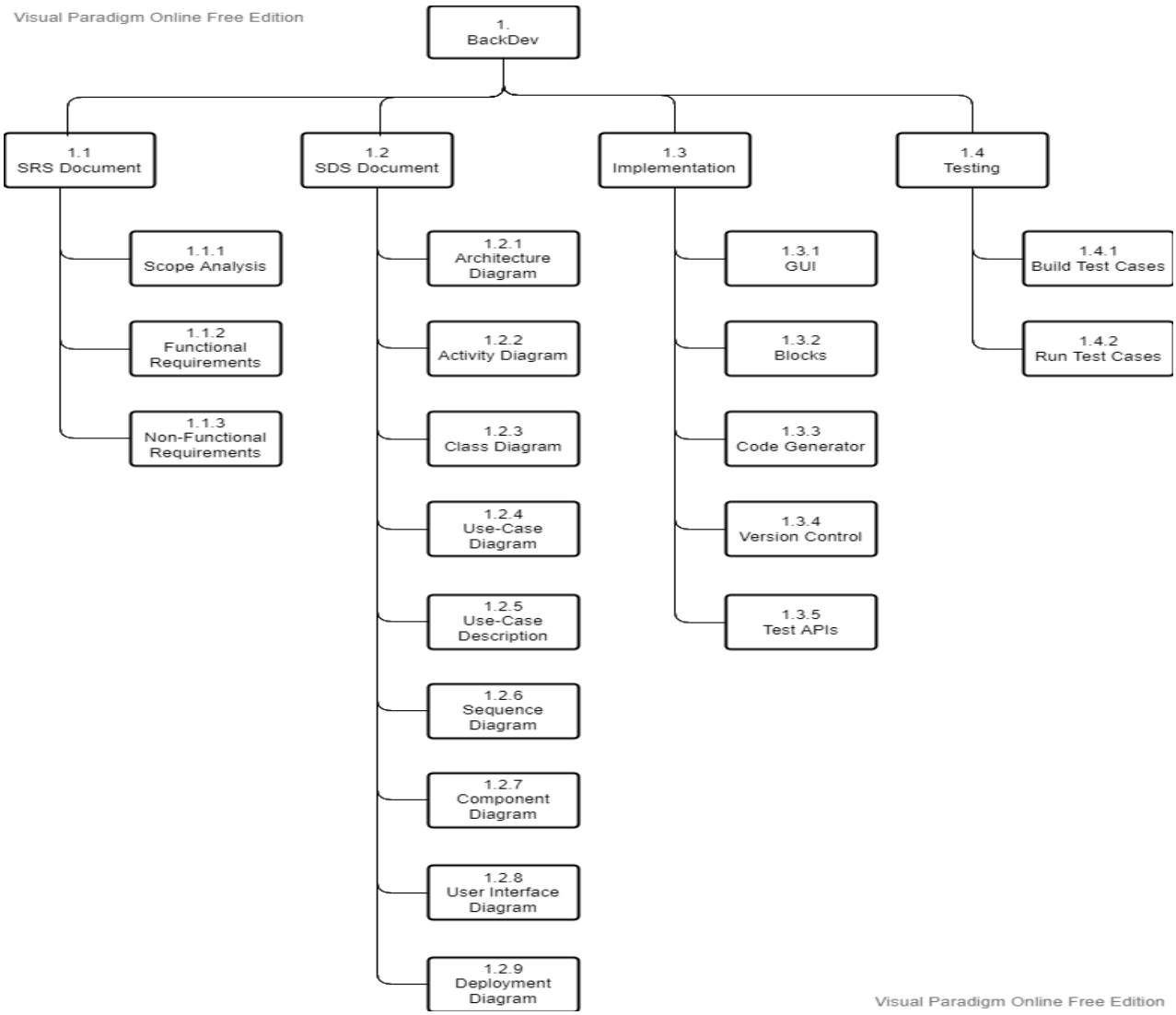

Figure 4.1: Work Breakdown Structure

4.2. System Overview

The application will allow users to use drag-and-drop action to speed up their backend development process. The developer will design the processes of the application on the canvas by dragging and dropping operators instead of hard coding. A sidebar will be opened for that respective component. The developer will set all the parameters of a component in the sidebar. The command prompt will be running in the background and while importing the NPM packages, they will be auto-installed in the background.

The developer can use different blocks that are available. Also, the developer can create their custom blocks and save them for later use. The application will be built for NodeJS, the user shall be able to drag a certain module and drop it onto the editor which will generate the code for the user.

The user can set different parameters for a certain block and create program logic. After the creation of program logic, the intermediate code in the form of JSON will be stored temporarily in the local file and after pressing the run button that code will be converted to NodeJS code in a single file.

4.3. System Architecture

4.3.1. Architectural Design

The overall architecture of this application is based on pipe and filter architecture. The system is divided into fully independent and replaceable modules termed filters in this architecture.

In this system, all modules are located on the same machine, some run on a browser, and some on a desktop environment.

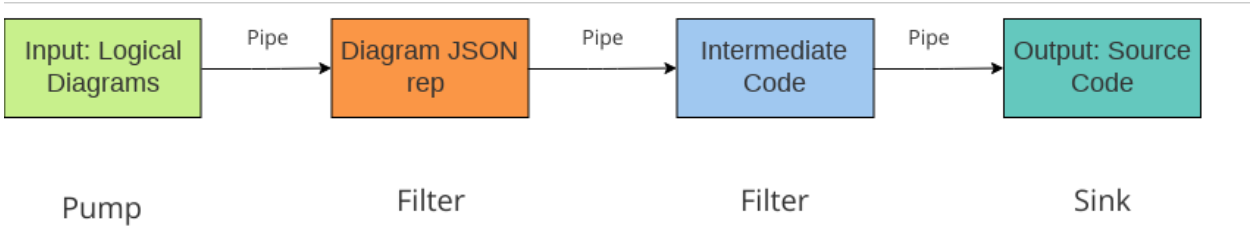

Figure 4.2: Architectural Design

- **Logical diagrams**

The user drags and drops blocks onto the canvas and draws the program logic using connectors and other blocks. This is an abstract view of the overall view.

- **Diagram JSON representation**

The diagram that is drawn will be represented in the form of a JSON object.

- **Intermediate code**

The blocks will be converted into intermediate code which will be sent to the code generator

- **Source code**

The code for the program logic built in the diagram will be generated using the code generator.

4.3.2. Decomposition Description

4.3.2.1. Module Decomposition

It consists of different modules. The one component will be desktop application that will call other components like Browser for front end and Node App for Back end. There will be other components within those components that will perform here functionalities. Each component is related to other component

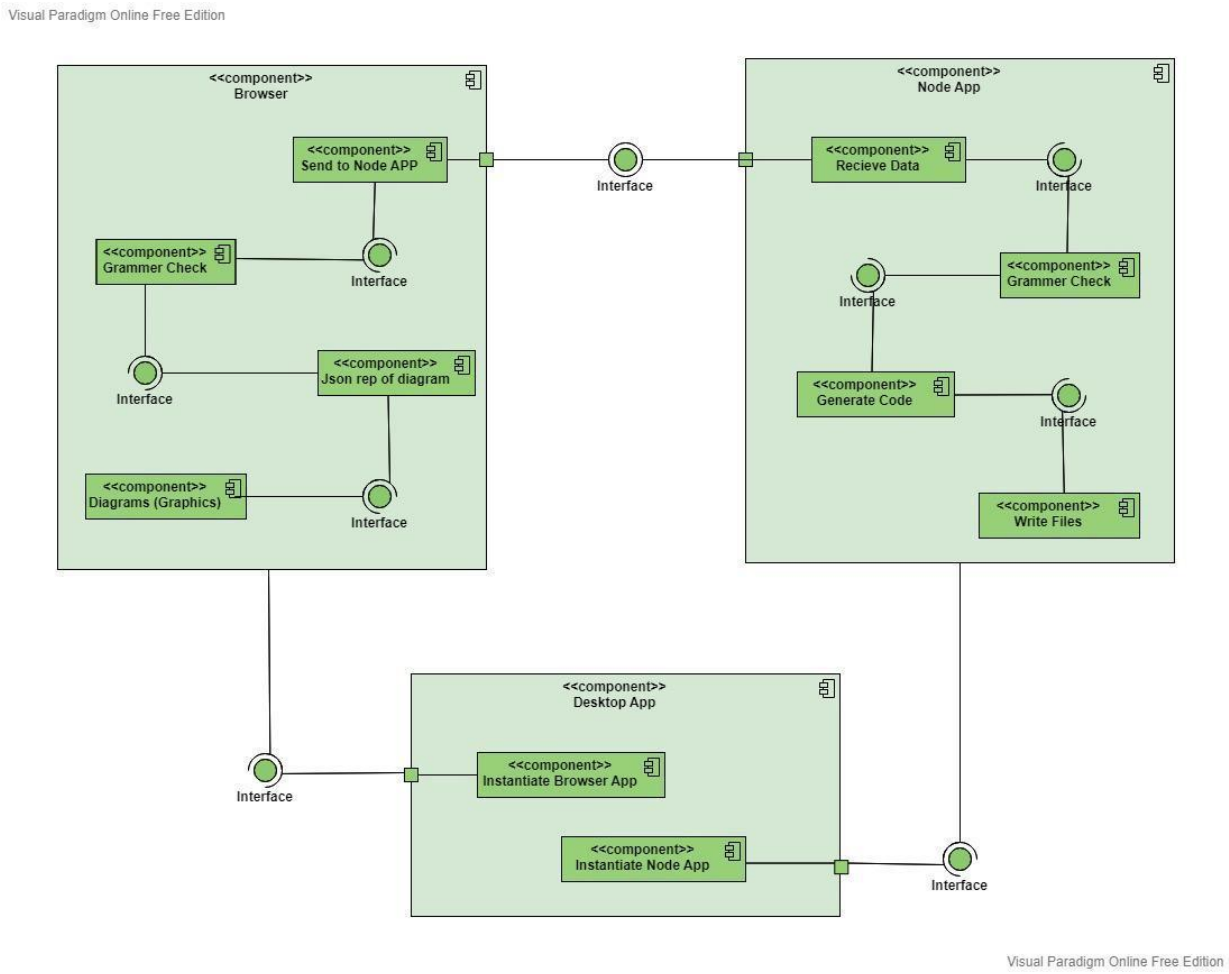

Figure 4.3: Module Decomposition

Class Diagram

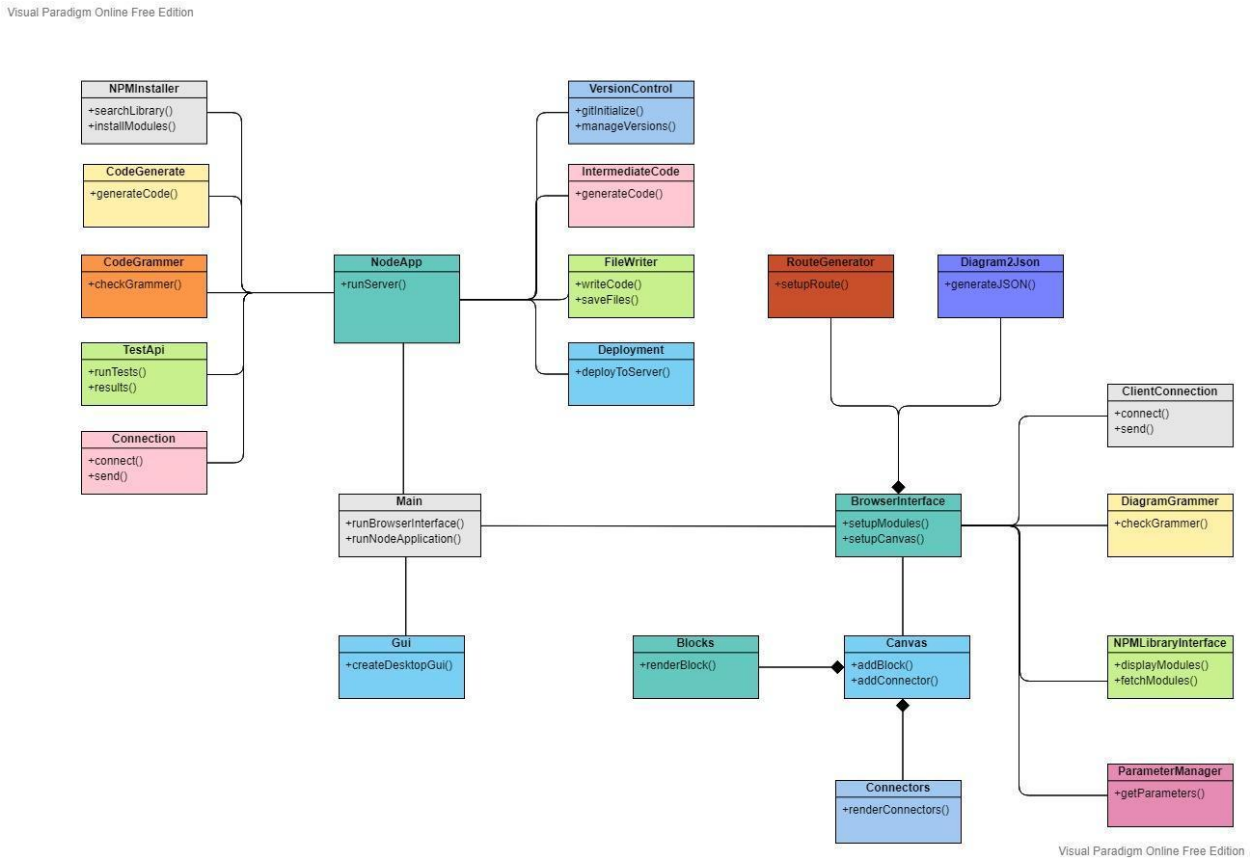

Figure 4.4: Class Diagram

There are 21 different classes/modules in our application.

- **Main** class will be the class that is connected with the **BrowserInterface** on the client side and also with **NodeApp** on the server side.
- **NodeApp** is a server-side class that manages the different server configurations like version control, file writer, deployment, npm installer, connection, etc.
- **Browser Interface** will be the client-side class that manages different client-side configurations like client connection, grammar, canvas connectors, blocks, etc.
- **NPM Installer** will install different NPM packages that the user will want.

- **Code Generate** will generate the code for nodeJS.
- **Code Grammar** will check the code grammar.
- **TestApi** will test the code and give the results.
- **Connection** will be established with the client side.
- **Version Control** will manage different versions of code and allow to work in a team
- **Intermediate Code** is a class that will generate intermediate code in between the processes when the user gives different parameters to the block.
- **File Writer** will write code and save that code in a file.
- **Deployment** will deploy the backend application to the server.
- **Route Generator** will set up the route according to the user's need.
- **Diagram2JSON** will convert diagram logic to JSON format for the understanding of the computer.
- **Client connection** will establish a connection with the server.
- **NPM Library interface** in which user will select different NPM packages to install.
- **Block** which can be dragged and dropped at a certain place.
- **Connectors** will be used to join blocks which will help in the creation of program logic.
- **Canvas** in which the user will create program logic by connecting different blocks.

4.3.2.2. Process Decomposition

The process decomposition is explained through a sequence diagram and use case diagram which decomposes the system into well-defined and cohesive processes. The use cases explain the set of actions that a user undertakes while using BackDev.

Use Case Diagram

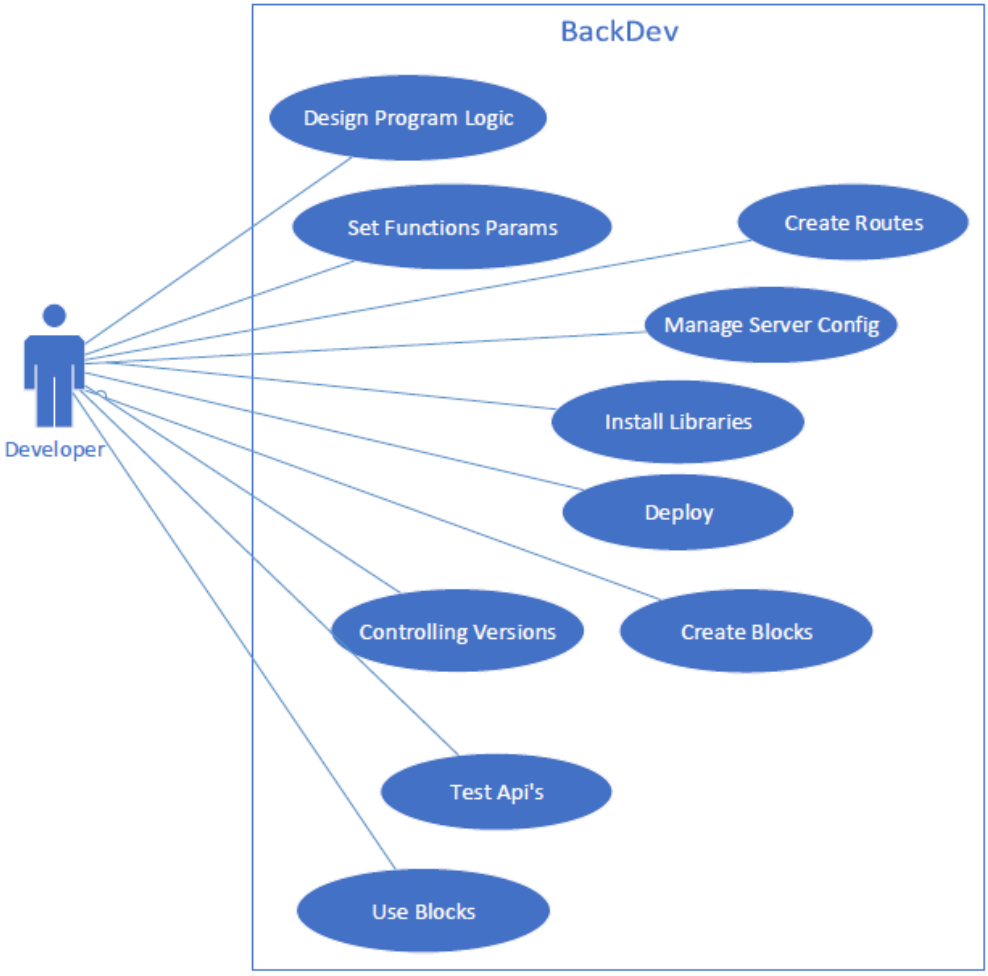

Figure 4.5: Use Case Diagram

The primary actor, the developer in the use case diagram can perform all the functionalities listed in the use case starting from Design Program Logic, Set function Params, Test APIs to Version Control.

Sequence Diagram

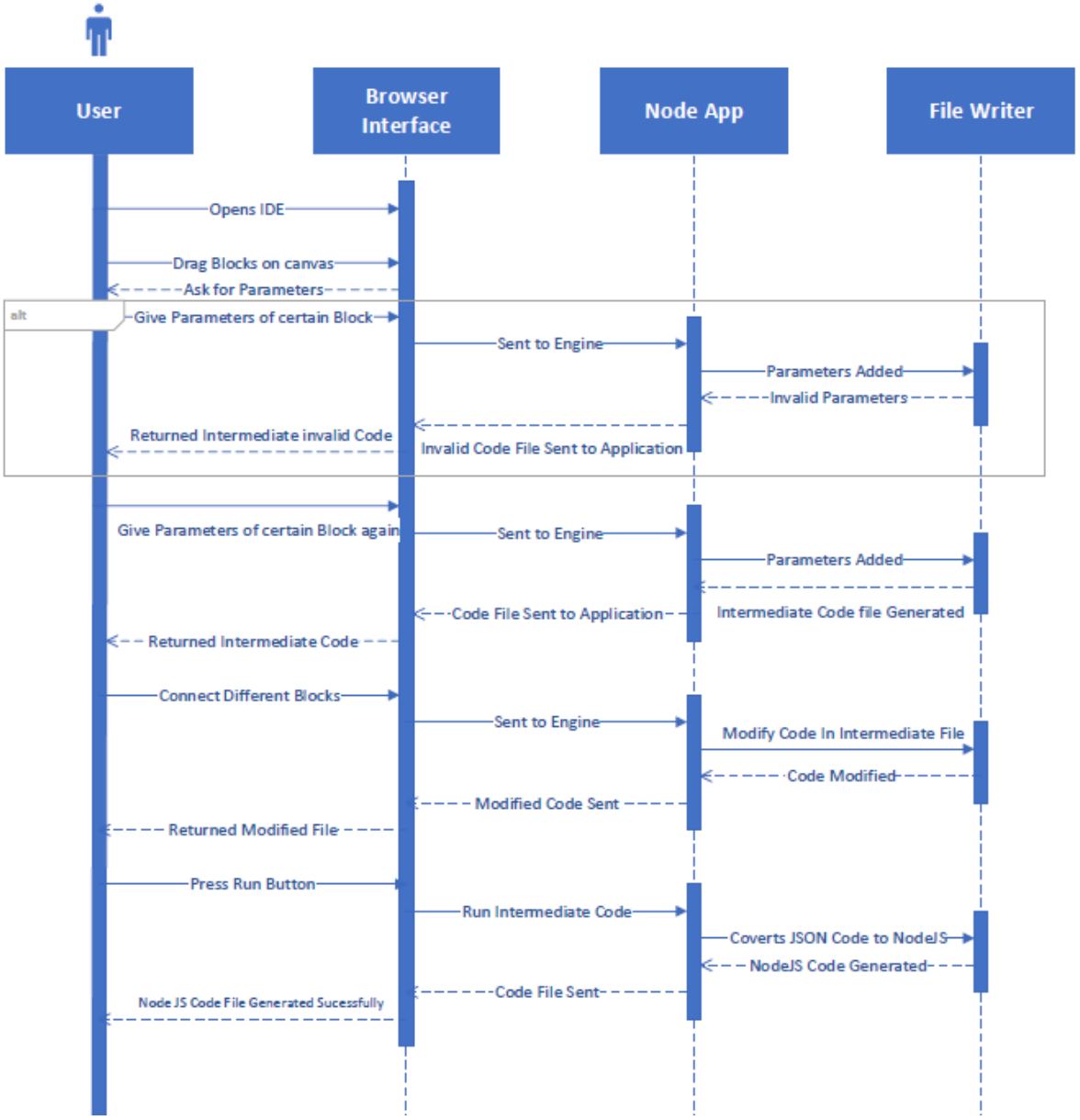

Figure 4.6: Sequence Diagram

The sequence diagram shows the sequence of events and object interactions arranged in a time sequence. The user opens the Main Window to interact with the system, which further displays Canvas to the user. The user will select certain blocks and drag them on the canvas and select

different parameters and then the parameters will send to the code generation engine. After adding the parameters, the intermediate code will be generated in a file.

Then the user will connect different blocks with connectors and the code will be modified in the file. After the creation of program logic, the user will press the run button and the intermediate code will be converted to nodeJS code and the user will get that code in a file.

Deployment Diagram:

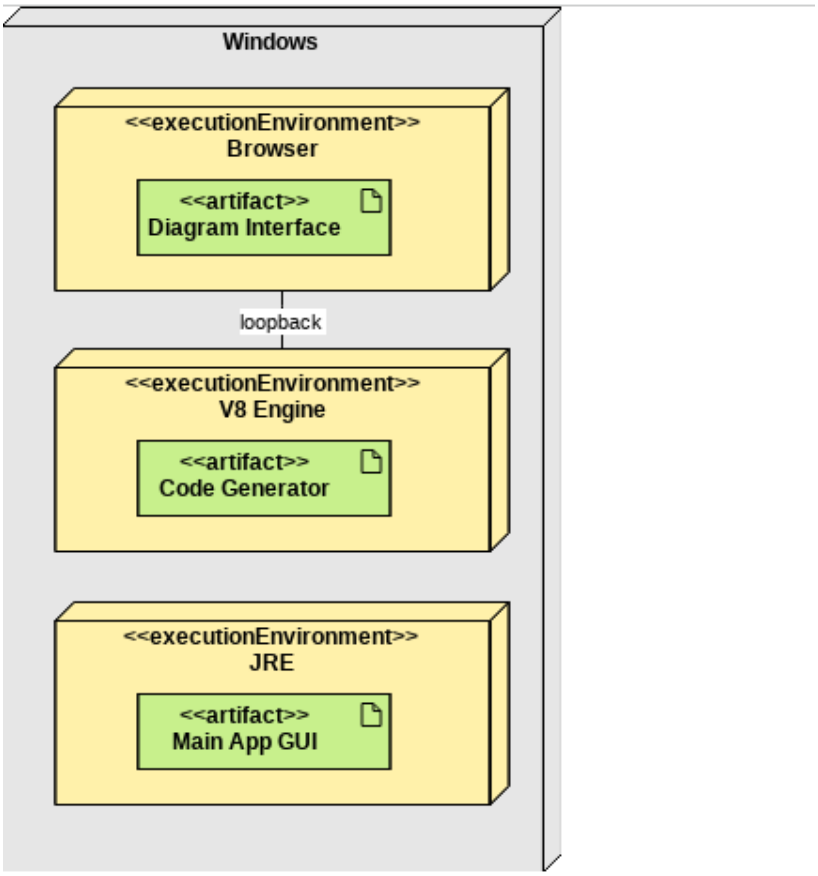

Figure 4.7: Deployment Diagram

4.3.3.Design Rationale

The architectural style of this application is a pipe and filter. Each filter is responsible for certain functions and the overall product functionality can be broken down into simpler filters in the architecture. The output of one filter is fed as input to the next filter which will generate its output.

The pipe and filter architecture provides an effective way to implement an IDE with added complexities of Visual Programming.

Consideration was given to the Model View Controller Architecture; however, the complexity of the application necessitated the use of a less generalized architecture style.

4.4. Data Design

4.4.1. Data Description

Our application uses a particular file to store JSON data of the blocks, diagrams, and user data. The multiple JSON files will then be converted into a single NodeJS file after pressing the run button. A repository will also be used for version control.

4.4.2. Data Dictionary

Table 4.1: Data Dictionary

Objects	**Attributes**	**Methods**	**Parameters**
NodeAPP	Name: String	runServer()	
NPMInstaller	Name: String	SearchLibrary() installModules()	Input: String

Code Generate		generateCode()	Output: String
TestAPI		runTest() results	
Connection	Flag: bool	connect() send()	
Version Control		gitInitialize() manageVersions()	Output: File
Intermediate Code		generateCode()	Output: File
File Writer		writeCode() saveFiles()	Output: File
Deployment		deployToServer()	Output: File
Main		runBrowserInterface() runNodeApp()	
GUI		createDesktopGUI()	
Browser Interface		setUpModules() setUpCanvas()	
Diagram2JSON		generateJSON()	
Client Connection		connect() send()	
Blocks	Name: String	renderBlock()	
Connectors		renderConnectors()	
Canvas		addBlock()	

		addConnector()	
NPM Library		displayModule() fetchModule()	
Route Generator		setUpRoute()	
Parameter Manager		getParmeters()	

4.5. Component Design

In this section of component design, we will take a closer look at each component of BackDev in a more systematic way. Each Component will have a functional description and closer detail. The main components of this application are the client and server which have been studied below through the activity diagram.

Client Side

- User Opens IDE
- Application displays the main screen.
- User will choose a certain block and drop it on canvas.
- User will set different parameters of certain blocks.
- User will connect different blocks.
- user can view intermediate code in the form of JSON.
- User presses the run button.
- NodeJS Code will be displayed to the user.

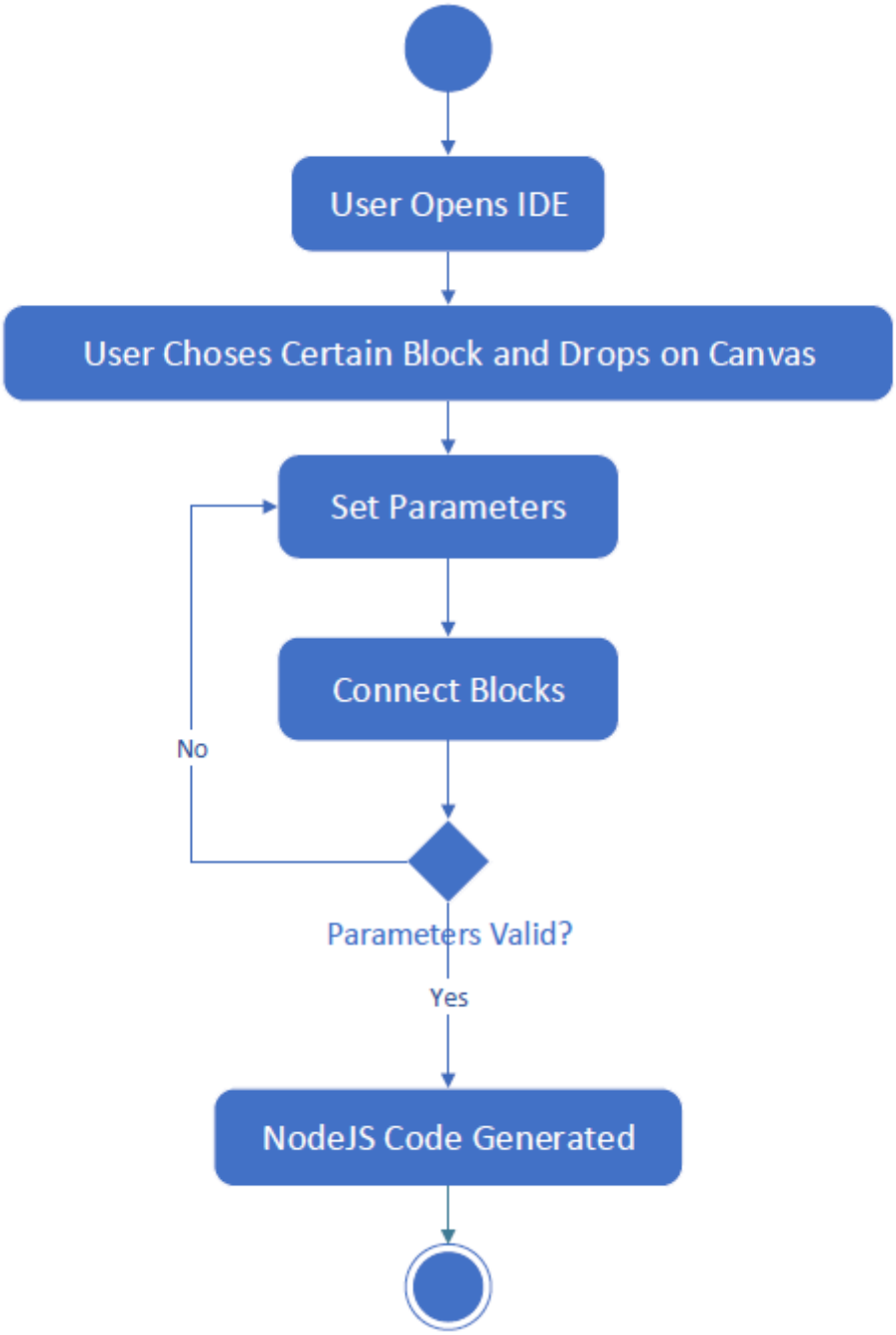

Figure 4.8: Activity Diagram (Client)

Server Side

- Server will start
- Get parameters that were entered by the user.
- Generate intermediate code in the form of JSON if parameters are valid otherwise generate an error message.
- Code Updates after joining different blocks.
- If the updated code is valid, then generate a NodeJS file else go back to step 4.
- Server will stop.

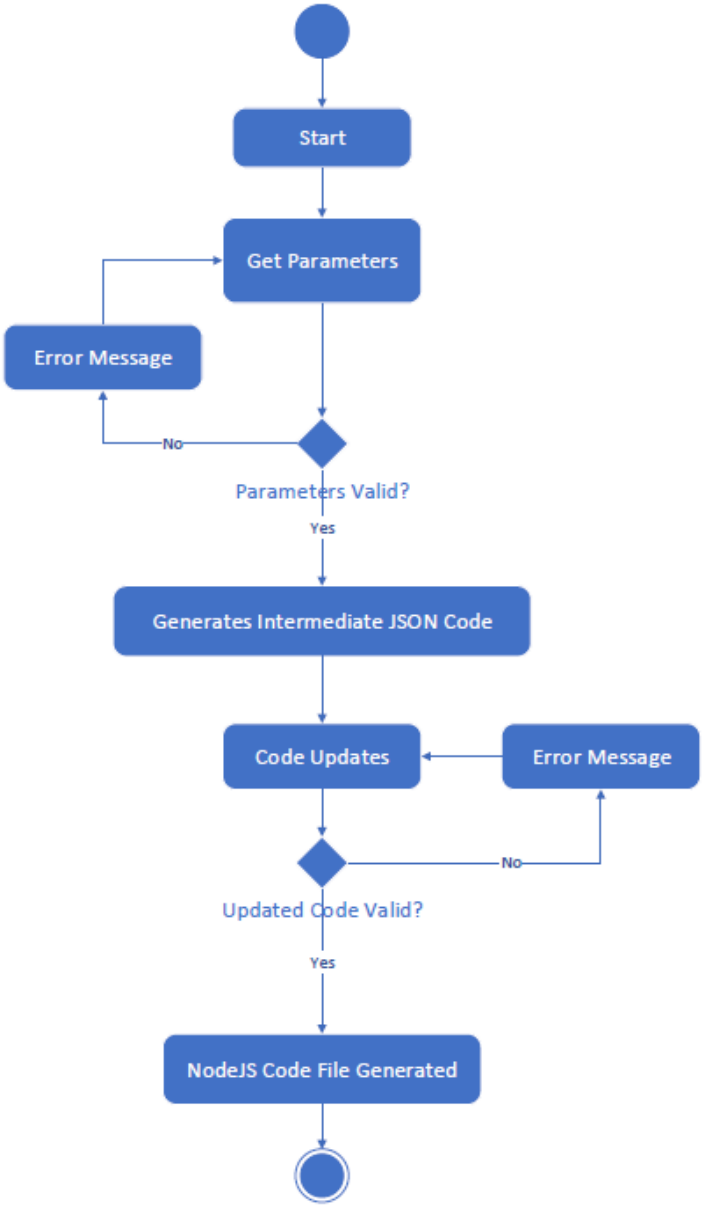

Figure 4.9: Activity Diagram (Server)

4.6. Human Interface Design

4.6.1.Overview of Human Interface

BackDev is a Desktop Application that will display the main screen when opened first by the user. From the user's perspective, when the main screen is displayed, he can select different blocks from the block library and drops them into the canvas. Then the user will set the different parameters for

the specified block and connect different blocks. Users can also create custom blocks for the specified functionality and they will be saved in the block library for future use. Users can import different npm packages and libraries. A command prompt will be running in the background which will install different npm packages for the user. After creating the program logic completely, the user will enter the run button and the nodeJS file will be created. The user can also view the NodeJS code file.

4.6.2.Screen Images

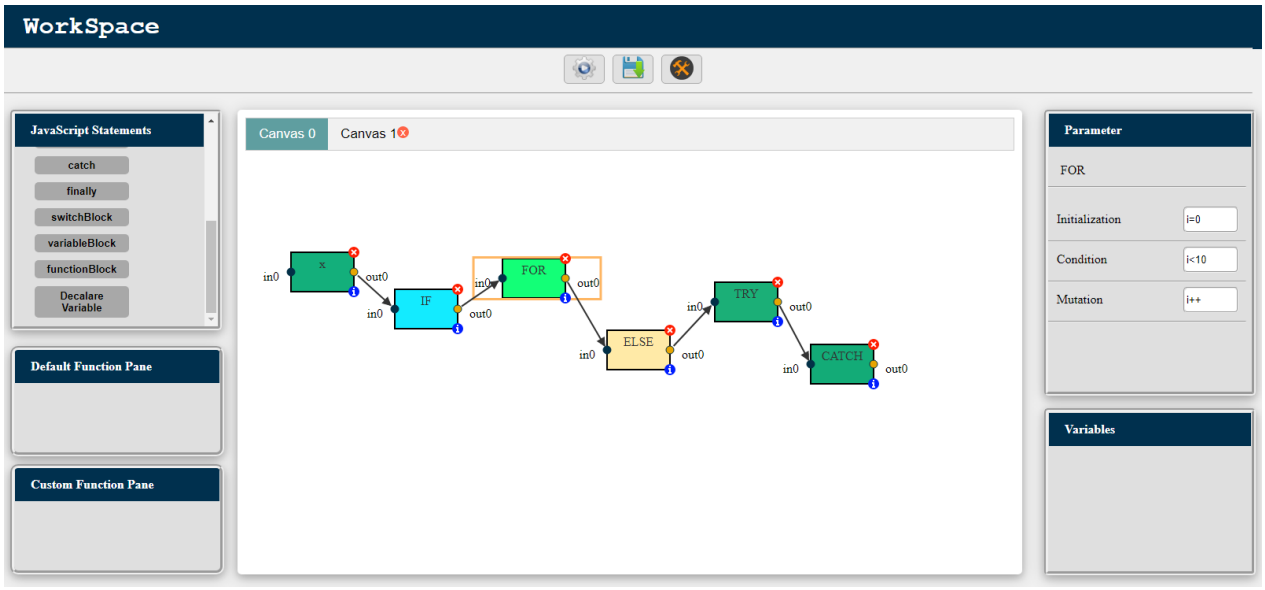

Figure 4.10: Workspace

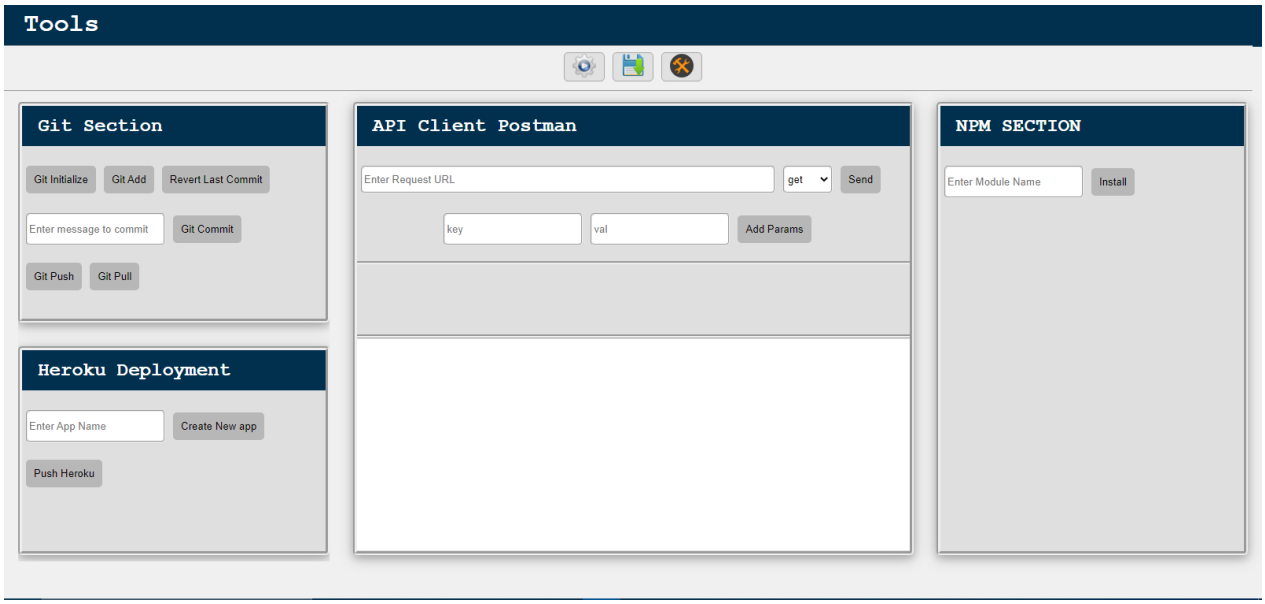

Figure 4.11: Tools

4.6.3.Screen Objects and Actions

- **Canvas**

In a canvas, the user will be able to drag different blocks and create program logic by connecting different blocks with a connector. This will act as a drawing page for the development of backend applications.

- **Parameter Section:**

This section will have a list of all the parameters that the user can add for the specific block. Users can adjust parameters according to the requirement.

- **Block**

There will be a unique block for the specific kind of functionality, by which a parameter section will be displayed concerning that block. Users will set different parameters for that specific block.

- **Connectors**

Connectors will be used to link different blocks which will be merged as a whole, and program logic will be completed.

- **Block Library**

It will contain all the blocks which will be used in the canvas for the creation of program logic. It will be updated by a user-customized block.

4.7. Requirement Matrix

Table 4.2: Requirement Matrix

Req No.	Requirement	Component
R-4.1	The system will provide a visual programming interface for JavaScript (NodeJS) enabling users to create backend applications by configuring functional modules using GUI-based operators.	Diagrams (Graphics)
R-4.2	The system shall provide graphical objects (blocks) of JavaScript programming constructs. The user shall drag the block into the canvas, the editor will then ask the user for parameters and generate code.	Diagrams (Graphics)
R-4.3	The system shall provide an interface to graphically manage all the server configurations. This includes middleware management, database connection, socket management, etc.	Generate Code
R-4.4	The system shall provide a library of blocks, where frequently used functions such as database queries, encryption/decryption, authentication, etc.	Diagrams (Graphics)
R-4.5	The system will generate JavaScript code according to the logic defined by the user in a visual programming interface	Write Files

R-4.6	The system shall provide an interface to manage routing using the ExpressJS library.	Diagrams (Graphics)
R-4.7	The system shall provide inbuilt blocks for socket communication using the socket.io library.	Diagrams (Graphics)
R-4.8	On the fly, help would be available with full documentation of each block	Diagrams (Graphics)
R-4.9	There will be an interface to test the generated code using JavaScript testing libraries like jest.	Grammar Check
R-4.10	The system will provide an inbuilt rest API client for server testing like postman	Grammar Check
R-4.11	The system will provide inbuilt git support to manage versions of a project	Write Files
R-4.12	The system will provide an interface to import NodeJS libraries from the npm server.	Receive Data
R-4.13	The system will provide an inbuilt deployment interface to deploy the project	Generate Code
R-4.14	The user shall be able to extend the block library by creating custom blocks that can then be reused	Diagrams (Graphics)

5. System Implementation

5.1. Tools

Following tools were used in the development of the prototype application

- **Visual Studio Code**

  Visual Studio Code was used as the code editor for most of the development.

- **Node.js**

  Node was used as the platform for our javascript application.

- **React**

  React javascript library was used in developing the front end of the application in combination with HTML/CSS.

- **JointJS**

  JointJS library was used to achieve certain functionalities of drag and drop in the canvas.

5.2. UI/Interface

Screenshots of the prototype app that was developed are attached below:

5.2.1. Splash/Loading Screen

This is the loading screen that appears whenever we start the application

Figure 5.1: Splash Screen

5.2.2. Workspace

This is the workspace of the application where the user has access to various things including the canvas, functions, statements, etc.

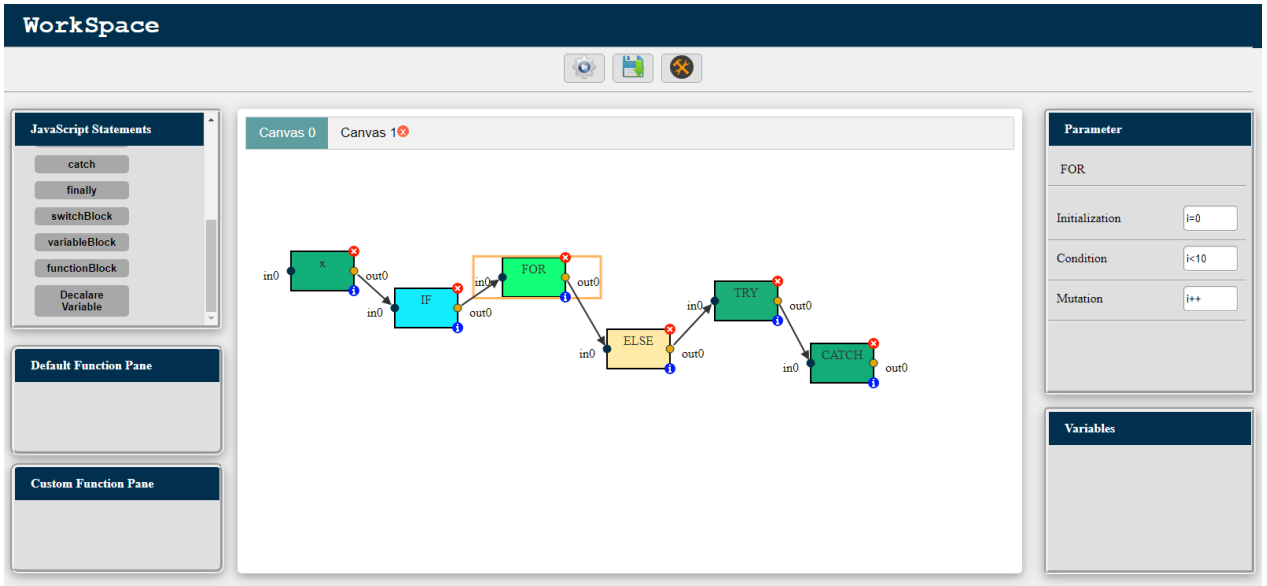

Figure 5.2: Workspace

5.2.3. Canvas

The canvas is where the user draws the program logic. It is developed using HTML Canvas and the JointJS library.

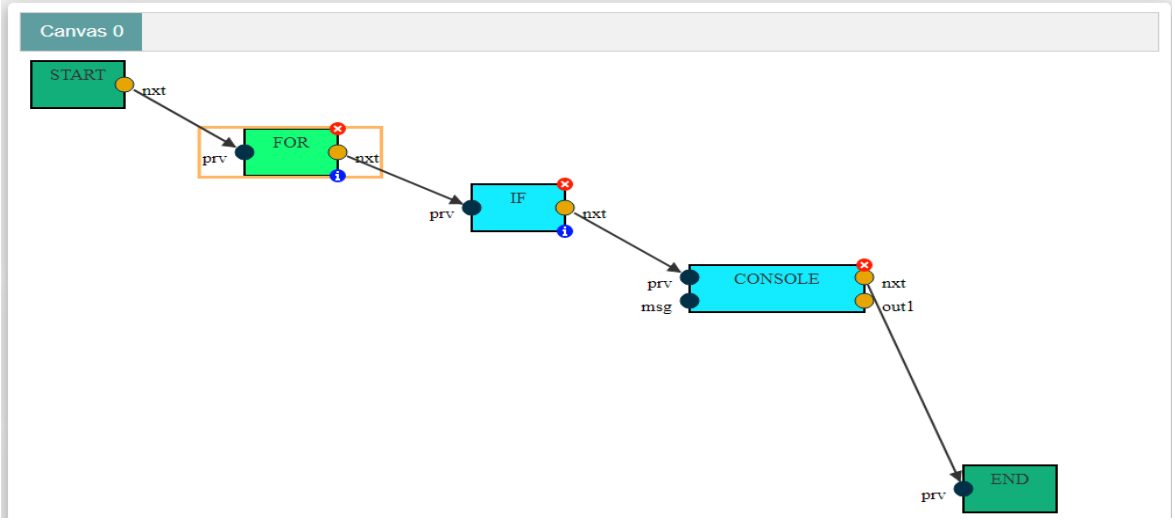

Figure 5.3: Canvas

5.2.4. JavaScript Statements

The user has access to different JavaScript statements in this section of the workspace. When the user clicks a certain item, it appears on the canvas.

Figure 5.4: JavaScript Statements

5.2.5. Parameters

Here the user can set different parameters for a selected block from the canvas.

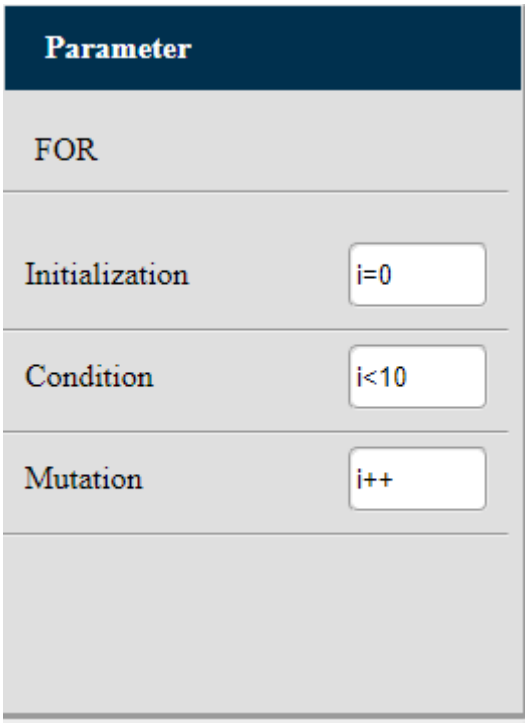

Figure 5.5: Parameter

5.2.6. Default Functions

In this section, the user has access to many different functions that javascript provides by default. Of course, the user can also make some custom functions and add them to the list of custom functions for later use.

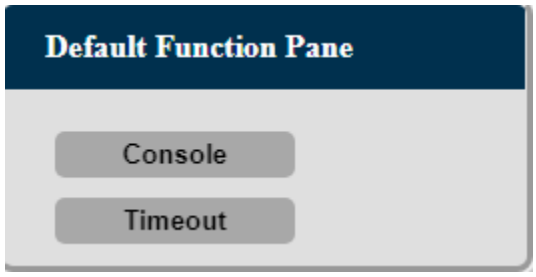

Figure 5.6: Default Function

5.2.7. Tools Pane:

This is the tools pane where the user has access to many different tools for his application.

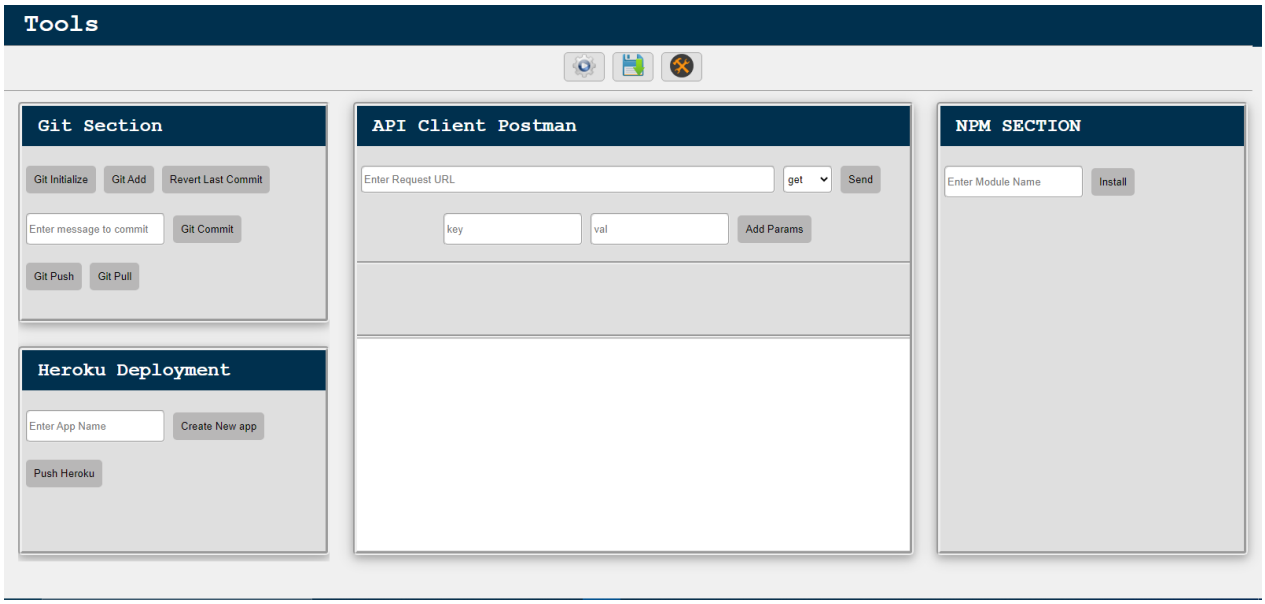

Figure 5.7: Tools

5.2.8. Git Section

The git section allows the user to use git commands directly using a GUI instead of having to use the terminal/command prompt.

Figure 5.8: Git

5.2.9. Deployment

This section allows the user to deploy their app to Heroku.

Figure 5.9: Deployment

5.2.10. API Testing

This section allows the user to test their APIs using Postman

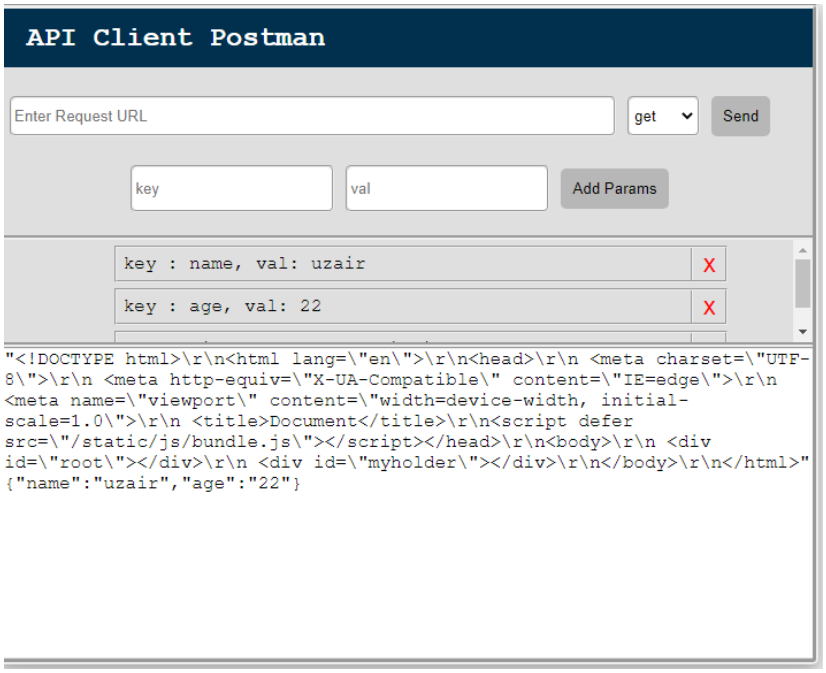

Figure 5.10: API

5.2.11. NPM Packages

Here the user can search for different node packages and install them directly from the app instead of using the terminal/command prompt.

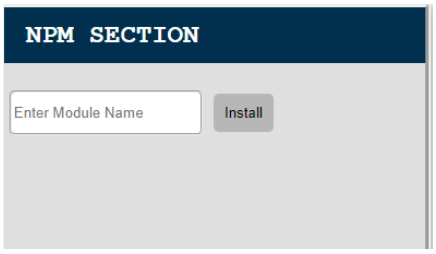

Figure 5.11: NPM

5.2.12. Code Generated

Here is some code that we have generated after creating some program logic.

```
if(x > 0){
  console.log("Greater Than zero");
}
else{
  console.log("Less or Equal to zero")
}
```

Figure 5.12: Generate

5.3. Code Analysis

5.3.1. Front End

Following is the basis of our front-end code base which runs our web-based GUI.

```
<h1 className="heading">Welcome to BackDev</h1>
<div className="big">
  <div onClick={canvas} className="btn">
    <div className="in">
      <img src={blank} alt="blank" className="blank" />
    </div>
    <div className="in">
      <h3>Create Project</h3>
      <p>Start a new process from scratch</p>
    </div>
  </div>
  <div className="btn" onClick={Open}>
    <div className="in">
      <img src={blank} alt="blank" className="blank" />
    </div>
    <div className="in">
      <h3>Open Existing Project</h3>
      <p>Open previously Existing Project</p>
    </div>
  </div>

  <div className="btn">
    <div className="in">
      <img src={blank} alt="blank" className="blank" />
    </div>
    <div className="in">
      <h3>User Guide</h3>
```

Figure 5.13: Frontend

ReactJS

Here is some Code for ReactJS that we have used it as our Front End.

```
function App() {
  const [blockState, setBlockState] = useState([]);
  const [connections, setconnections] = useState([]);
  const [variables, setVariables] = useState([]);
  const [highlight, setHighlight] = useState(null);
  const [varCounter, setVarCounter] = useState(0);

  const [loading, setLoading] = useState(false);
  const [first, setFirst] = useState(true);
  const [newProject, setNewProject] = useState(false);
  const [prevProject, setPrevProject] = useState(false);
  const [prevProjectData, setPrevProjectData] = useState([]);
  const [canv, setCanv] = useState(false);

  let graph = new dia.Graph({}, { cellNamespace: shapes })
  graph.on('remove', function (cell) {

    setCanvases(state => {
      let temp = [...state]

      temp = temp.map((v) => {
        if (v.hide == false) {
          let a = [...v.variables]
          a = a.filter(v => {
            return v.id != cell.id
          })
          return {
```

Figure 5.14: React

JointJS

Here is some code for the Joint JS that we have used in our project. We have integrate in with ReactJS.

```
var model = new shapes.standard.Rectangle({
    position: { x: 275, y: 50 },
    size: { width: w, height: h },
    attrs: {
        body: {
            fill: color,
        },
        label: {
            text: name,
            fontSize: 16,
            y: -10
        }
    },
    ports: {
        groups: {
            'in': portsIn,
            'out': portsOut
        }
    }
});
const portArray = []
for (let i = 0; i < input; i++) {
    if(inputs) {
        portArray.push({
            group: 'in',
            attrs: { label: { text: i == 0?'prv':inputs[i - 1] }}
```

Figure 5.15: Joint

5.3.2. Back End

As for the Backend, we have used NodeJS. Here is some code of NodeJS.

```
app.get("/openPro", function (req, res) {
  //console.log(req);
  console.log(process.cwd());
  fs.readdir("../SavedProjects", (err, items) => {
    console.log(items);
    res.send(items);
  });
});

app.post("/gitinit", function (req, res) {
  //console.log(req);
  console.log("in git init");

  if(checkGit(process.cwd())) {
    console.log('git repository already exist');
  } else {

    console.log('Git repository not exist..... Now initializing');

    spawn("cmd", ["/c", "git init"], {
      //<----
      shell: true,
      stdio: "inherit",
    })
```

Figure 5.16: Backend

6. Conclusion and Future Direction

6.1. Future Direction

Working with Backend Development and specifically, working with a NodeJS is a very huge research field. With current project as a proof of concept, it has opened path for multiple research areas to follow.

People interested in developing backend for applications through visual programming can contribute to the project via developing modules to make this application better. Further, they can contribute via developing module for the system to update the application as they can add any other language to create backend instead of NodeJS.

Fellows from software engineering may contribute via designing and developing a more sophisticated design to better accommodate and replicate backend techniques with more accuracy and efficacy.

In this section, we will discuss the different possibilities of things that we can integrate into our tool to improve it in the future.

- More components
- More functionality
- Improvement of the overall system
- Integrate other backend languages like PHP, JAVA etc.

6.2. Conclusion

We are developing a prototype visual programming tool that allows users to use drag-and-drop based features to develop their programming logic. Due to limited time and resources, we couldn't fully explore the scope of this project. If implemented completely then it would surely be a very helpful tool to not only students but also developers.

7. References and Citations

- Ambler, S. W. (2005). *The elements of umltm* 2.0 style. Cambridge University Press.

- "IEEE Recommended Practice for Software Requirements Specifications," in *IEEE Std 830-1998* , vol., no., pp.1-40, 20 Oct. 1998, doi: 10.1109/IEEESTD.1998.88286.

- "IEEE Standard for Information Technology--Systems Design--Software Design Descriptions," in *IEEE STD 1016-2009* , vol., no., pp.1-35, 20 July 2009, doi: 10.1109/IEEESTD.2009.5167255.

- Young, A., Meck, B., Cantelon, M., Oxley, T., Harter, M., Holowaychuk, T. J., & Cantelon, M. (2017). *Node.js in action.* Manning.

- Mardanov, A. (2013). *Express.js Guide: The comprehensive book on express.js*. Lean Publishing.