

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/370074438>

ONLINE COMPILER, IDE, INTERPRETER, AND REPL FOR MULTIPLE PROGRAMMING LANGUAGES

Article · September 2021

CITATIONS

0

READS

128

1 author:



Eyitayo Ogunbiyi

University of Lagos

1 PUBLICATION 0 CITATIONS

SEE PROFILE

**ONLINE COMPILER, IDE, INTERPRETER, AND
REPL FOR MULTIPLE PROGRAMMING
LANGUAGES**

BY

OGUNBIYI EYITAYO

150408005

September 2021.

PROJECT REPORT
ON
ONLINE COMPILER, IDE, INTERPRETER, AND REPL FOR
MULTIPLE PROGRAMMING LANGUAGES

SUBMITTED TO THE UNIVERSITY OF LAGOS, AKOKA.



IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR
THE AWARD OF THE DEGREE OF BACHELOR OF
SCIENCE(B.Sc. Hons) IN COMPUTER ENGINEERING,
UNIVERSITY OF LAGOS

BY:
OGUNBIYI Eyitayo Oluwadurotimi
MATRIC NO: 150408005
PROJECT SUPERVISOR: DR. A.O BALOGUN

CERTIFICATION

This is to certify that this project, **ONLINE COMPILER, IDE, INTERPRETER AND REPL FOR MULTIPLE PROGRAMMING LANGUAGES**, was carried out by **OGUNBIYI EYITAYO OLUWADUROTIMI (150408005)** and submitted in partial fulfillment of the requirements for the award of Bachelor of Science (B.Sc.) degree in Computer Engineering from the University of Lagos, Akoka.

Signature

Ogunbiyi Eyitayo Oluwadurotimi

(Author)

Date

Signature

Dr. A.O Balogun

(Supervisor)

Date

Signature

Dr. (MRS.) A.O. Gbenga-Ilori

(Project Coordinator)

Date

Signature

Dr. S.O. Adetona

(Head Of Department)

Date

DEDICATION

This report is dedicated to my parents and siblings whose support and advice remained invaluable and priceless throughout my university education.

ACKNOWLEDGMENT

Firstly, I would like to acknowledge my parents, Mr & Mrs. Ogunbiyi for the provision of both financial and emotional support throughout this journey. My roommates also, for enabling a conducive environment for work.

To the entire python & go lang open source community.

I also want to thank my project supervisor, Dr. Balogun for his constructive criticism, suggestions, and advice.

ABSTRACT

In today's age, computer software has made life considerably simpler for us. From seemingly trivial software such as social networks to more critical systems such as airplane software or financial systems, nearly every aspect of our lives depends on software. The coronavirus pandemic-induced lockdown highlighted how crucial it is for engineers, scientists, and students to be able to remotely collaborate on these crucial software systems that our lives depend on. The purpose of this project is to develop an online compiler/development environment that enables the collaborative creation and maintenance of software systems.

CERTIFICATION	3
DEDICATION	4
ACKNOWLEDGMENT	5
ABSTRACT	6
LIST OF FIGURES	9
CHAPTER ONE	1
INTRODUCTION	1
1.1 Background Of Study	1
1.2 Statement Of The Problem	2
1.3 Aim Of Study	2
1.4 Objectives Of Study	3
1.5 Significance Of Study	3
CHAPTER TWO	4
LITERATURE REVIEW	4
2.1 Introduction	4
2.2 Overview Of Cloud-Based Development Environments	4
2.3 Theoretical Framework	8
2.3.1 Cyclomatic Complexity	8
2.3.2 Compilers	10
2.3.3 Interpreters	10
2.3.4 Virtualization	10
2.3.5 Integrated Development Environment (IDE)	11
2.3.6 Web Sockets	12
2.4 Summary	13
CHAPTER THREE	14
METHODOLOGY	14
3.1 Introduction	14
3.2 Methodology	14
3.3 System Analysis	14
3.4 Software Architecture	15
3.5 Programming Languages Supported	17
3.6 Client-Side	17
3.7 Server-Side	20
3.8 Security	21
3.8.1 Use Of HTTPS	21
3.8.2 Password Hashing	21
3.9 Software Tools Used	22
3.10 Code Compilation & Execution Flow	23
	7

CHAPTER FOUR	24
IMPLEMENTATION	24
4.1 Introduction	24
4.2 Registration	24
4.3 Login	25
4.4 Forgot Password	26
4.4.1 Requesting a password reset	26
4.4.2 Complete the password reset	27
4.5 Profile	28
4.6 Application Dashboard	28
4.6 Code Editor	29
4.6.1 - File Creation	30
4.6.2 - Code Execution Result	31
4.7 Code Editor - Collaboration Mode	31
4.8 Database Tables	33
4.8.1 - Users Table	34
4.8.2 - Files Table	35
4.8.3 - Code Submissions Table	36
4.8.5 - File Collaborators Table	36
4.8.6 - Password Reset Token Table	37
4.9 Performance Graphs	37
4.10 Comparison To Related Work	38
CHAPTER 5	40
CONCLUSION	40
5.1 Introduction	40
5.2 Concluding Remarks	40
5.3 Challenges Faced	41
5.4 Suggestions For Improvement	42
REFERENCES	43

LIST OF FIGURES

- Figure 2.1 - Internals of type 1 and type 2 hypervisors
- Figure 2.2 - Sample program showing WebSocket initialization
- Figure 3.1 - Layered architecture pattern
- Figure 3.2 - RestAPI communication flow
- Figure 3.3 - Working schematic of online compiler/ide
- Figure 4.1 - Sign up page
- Figure 4.2 - Login page
- Figure 4.3 - Initiate password reset page
- Figure 4.4 - Email received upon password reset
- Figure 4.5 - Complete password reset page
- Figure 4.6 - User profile page
- Figure 4.7 - Application dashboard
- Figure 4.8 - Code editor page
- Figure 4.9 - File Creation
- Figure 4.10 - Code execution result page
- Figure 4.11a - Starting a collaboration session
- Figure 4.11b - Collaboration invite email
- Figure 4.11c - Adding a collaboration ID
- Figure 4.12 - Database ER Diagram
- Figure 4.13 - User table
- Figure 4.14 - Code files table
- Figure 4.15 - Code submissions table
- Figure 4.16 - File collaborators table
- Figure 4.17 - Password reset token table
- Figure 4.18 - CPU, Bandwidth and Disk I/O Usage Graphs

CHAPTER ONE

INTRODUCTION

1.1 Background Of Study

In the field of computing, a compiler is a system software program that processes computer programs written in a source language (usually a high-level language) into a different, usually lower-level language. Usually, the source language is a high-level programming language (e.g C++, Matlab), and the destination language, a lower-level language (e.g machine code, assembly language). To translate the source statements, the compiler has to go through a variety of stages including lexical analysis, syntax analysis, code generation. Due to how fundamental compilers are to building software, they must be as efficient as possible and compiler correctness is an entire branch of computing that ensures compilers behave as expected. Correctness is ensured using techniques such as formal verification. Interpreters are similar to compilers, however, they execute statements of the source language without a requirement of conversion to a low-level language firstly. The tradeoff of using an interpreter as compared to a compiler is increased execution time of interpreted programming languages. Compilation and interpretation are not completely independent from each other as some interpreters have some sort of conversion tasks.

A compiler usually transforms the entire source code to its destination format before any execution begins. However, interpreters perform the behavior specified in the source file on the fly, as each statement is invoked. The distinction between interpreters and compilers has become thinner in modern times, as hybrid techniques such as just-in-time compilation (JIT). JIT compilation involves the compilation of a program as it is being executed, instead of compiling the program before any execution. The optimization in this kind of compiler comes from analysis of portions of the code which the speed gained from compiling the code would outweigh the time spent in compiling the code. Several modern languages now employ the use of JIT compilation such as Matlab, Javascript [[JIT compilation](#)]

Creating ways to access various compilers or interpreters in the cloud is timely now, as the importance of remote collaboration tools has been very well emphasized by the lockdown for the better part of 2020. With the improvement of web technology, many traditional desktop applications have become accessible through cloud-based services and this made engineers and researchers show more interest in developing tools to enable programming in the cloud. The advantages of an online development environment include managing source code in seemingly infinite storage, huge computing power, multi-device access. The disadvantage of using this online environment includes the requirement of internet connectivity.

Some web-based development environments supporting various languages include [Repl.it](#), [AWS Cloud 9](#), [CodeTasty](#). Repl.it supports up to 50 programming languages and allows collaborative editing among several users. AWS Cloud 9 has tight integration with AWS services which already makes it a good choice for engineers whose companies already use AWS. However, none support core scientific programming languages such as Octave, SciLab, which are very important in engineering.

This project looks into building a cloud-based development environment, with the usual features and the addition of support for scientific languages like Octave.

1.2 Statement Of The Problem

During the year 2020, the global lockdown highlighted how important remote collaboration has become. Engineers and scientists need to be able to remotely collaborate on crucial software and hardware systems that our lives are deeply integrated with. Students need to be able to collaborate without being physically present at hostels or classrooms.

1.3 Aim Of Study

This study aims to come up with a solution that enables multiple persons to remotely collaborate on the same programming project.

1.4 Objectives Of Study

The major objectives of undertaking this project include:

1. Review of current online-based programming tools.
2. Develop a web application that enables online collaborative programming between multiple users.
3. Using WebSockets to enable the real-time collaboration process.
4. Evaluation of the developed application.

1.5 Significance Of Study

Developing an online-based integrated development environment focused on enabling collaboration between multiple users improves the process of working on programming-related projects among geographically distributed teams. It also removes the overhead involved in having to set up multiple compilers and development environments when first using a programming language. Expenses that were previously incurred in having to transport workers in a different location, to the location where a programming project is being worked on, can be saved. Students can also collaborate on projects regardless of their physical location, facilitating inter-university collaboration.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter reviews various related texts that deal with building online development environments with the goal of understanding software engineering concepts that they employed, as well as other related concepts.

2.2 Overview Of Cloud-Based Development Environments

Literature information from various published studies on the development of cloud-based development environments are presented below

Yi Wang's [1] (2017) research involved looking into understanding how users of cloud-based IDEs interacted with the emerging cloud-based IDEs. The author highlighted the rise in cloud-based development environments and realized that observing how users behave while using these tools can help us in improving them. The author believes if we can discover the user's pain points with current solutions, then we can provide improved designs and features in the future. Modeling user behavior can also serve as training data for building futuristic recommendation systems that can assist users working on programming projects. The author believes that studying the programmer's behavior while they work on projects is going to be fundamental in improving cloud-based IDEs and give us the ability to provide development environments for software developers.

L'uboš C. and Martin K [2] (2017) developed an educational web-based software application that enabled end-users to develop basic block schemes within their browsers. The block schemes created are encoded using extensible markup language (XML), making it fully integrable with MATLAB/Simulink development environment. They tested schemes created within the browser by making an HTTP request to a MATLAB web server. The HTTP server responds with an output file in a format that the client can take in and use for further processing.

Budi Y. et Al [3] (2017) built Harmonik, to convert novice programmers' algorithms (which are very high level) into real programming languages. Their tool aims to guide young programmers as they gradually get proficient in building software with a professional IDE. Harmonik is a block-code software aimed at converting beginners into proficient programmers. The authors highlighted that several obstacles exist when beginner programmers are attempting to ramp up their skills. Removing the obstacles they might face when first trying to get proficient. They used Rational Unified Process (RUP) as their software development model.

Weimin D. et al [4] (2016) did a study that centered around how virtualization and cloud computing can be integrated and improve software development in general. Due to the high-efficiency resource offered by various cloud providers, there is a need for improvement in the way today's applications are engineered. The paper looked into building a more solid theoretical background, looking more intently into the gaps currently existing, and eventually improving the process of virtualization. The paper concluded by encouraging analysis of various virtualization approaches, making suggested improvements on each piece.

G. Fylaktopoulos et Al [5] (2016) performed a study that reviewed modern technologies for cloud-based software development environments. They looked into some specific characteristics which they felt were important to a cloud-based tool such as productivity enhancements, programming languages, and databases, integrated version control. Their result shows that most existing tools focus on the programming stages and post-development stages (e.g application deployment) aren't yet fully developed. The authors feel that debugging and application auditing should be paramount features in future implementations of these kinds of systems.

Ratnadip K. et al [6] (2016) developed software capable of compiling and executing different programming languages source code. They wanted engineers to be able to write different programs and debug them online, without any difficulty. Another problem they tackled was disk space management and portability. Using their web application, the overhead incurred when setting up development environments- on each of your local computers is avoided. You can pick from a wide range of available compilers without having to manually install any software on your local device.

Aditya K. et al [7] (2015) researched and built an application called CodeR. Their tool enabled real-time code editing using web-socket technology to enable programmers to collaborate while working on a project. Their application allows them to write and display the results of their programs through a terminal, as well as chat, real-time collaboration with other programmers. Their application supports C, C++, and Java programming languages. They want programmers to use collaborative technologies to work together to fix bugs and discuss the program they're working on, in the same single environment but different geographical areas.

Surya C. et Al [8] (2015) researched building a platform that enabled the compilation and execution of programs that can be securely deployed on a private cloud. Their reasons for a private cloud solution involved ease of execution of programs, as well as security reasons as a private cloud solution is dedicated solely to one organization. Their research project uses SOAP as its messaging protocol & the solution was able to save up to 50% monthly costs for customers who deployed their cloud online compiler, also increased flexibility and ease of use.

Stephan K. and Bernd B. [9], (2014) did research on how early feedback makes it easier to build software applications that best suit the end user's needs. They also discussed how to raise the usability and user experience of software applications using 2 software models: Tornado and Rugby models. They also emphasized the importance of conducting usability testing, as usability and the experience of the users are very important for software applications today.

Arjun D. and Arnab K. [10] (2014) worked on online compilers as a cloud service. They wrote a paper focused on tackling the challenges centered around managing disk storage space effectively and also, the potential of compilers to be easily ported from one system to another. They implement a software-defined controller which decides the compiler server against which a user-submitted program is executed. The criteria for selecting a compiler server are based on the load which a certain server has. The distribution is also verified by calculating the response time of programs that are allocated to the compiler. Their solution involves 3 tiers - user interface tier, controller tier & compilation tier.

Mutiara A.B et al [11] (2014) research exploited the advantages of cloud computing by building a secure cloud base IDE capable of developing applications which are commercial-scale. The languages which their research result analyzes were C, C++, and Java. The decision of which languages to support was a result of popularity. They tested the compiled code on basic functions such as IO, string and number operations, and iterations. Their central objective was to build an IDE that could compile code on the backend and return the result to the web browser client. Testing was emphasized in their research and they conducted several kinds of tests e.g stress test, load test, performance test, and latency test. Their result was able to drive down the time spent on manually installing multiple setup programs. It also drives down the disk space required as the user most likely already has a web browser installed on their local computer.

Mayank Patel [12] (2013) worked on an Online Java Compiler, using a cloud computing approach. They used cloud computing due to how quickly resources can be provisioned to scale up the online service. They also established the fact that using a cloud computing provider leads to the loss of direct control of the user data, but the several advantages outweigh the disadvantages. Their tool removes the need for separate installations of various compilers which a programmer would like to use for a project. Also, due to the seemingly infinite size of the cloud, the user never has to worry about running out of disk space. The user simply has to create an online account on their tool and be able to create and execute Java programs and the compilation occurs on a central server.

Timo A. et Al [13] (2012) worked on Arvue, a browser-based development environment. They highlighted the fact that the web is becoming more popular as an application platform due to the ease of maintenance, as well as how easy it is to update the application if it is a web application. They also mentioned how having a cloud-based IDE has a lot of benefits such as the developer not needing to worry about installations of software, or updating the versions of the compiler used. The project built during the paper was developed using Java, specifically the servlets feature of java which is a powerful tool for building Java web applications

Ling W. et al [14] (2011) built an IDE called CEclipse, It is designed for programming in a cloud environment. They harnessed the benefits of cloud computing and built this tool intending to

improve the experience developers have during the development process. They focused on 3 main problems faced by existing tools - Security guarantee, function implementation, and advanced utilization. CEclipse solved these problems using their custom approaches - such as analyzing the way the program behaves and using this for some intelligence.

Max Goldman et al [15] (2011) developed Collabode, a web-based integrated development specifically designed to support collaborative and synchronous development between multiple programmers. They also describe an algorithm for error-mediated integration of user's programs. Their algorithm was evaluated using data from previous experiments and also small-size studies with both professional programmers and students. They observed current approaches for developers to collaborate which were either - Working together on a single code copy or working in parallel using version control software. They concluded that neither was effective enough for truly synchronous collaboration between the users.

2.3 Theoretical Framework

2.3.1 Cyclomatic Complexity

Cyclomatic complexity is a software metric that is used to estimate the complexity of a software program. It measures the amount of linear and non-dependent paths through the source code of the given program.

To estimate the cyclomatic complexity, the control-flow graph of the program is used where the following labelling is used:

1. Nodes - indivisible groups of commands of a program.
2. Directed edge - connects two nodes together if the nodes occur sequentially

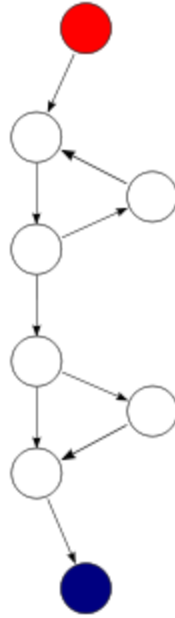


Figure 2.1 - Control-flow graph for a program

We can define the cyclomatic complexity with reference to the control flow graph, which is a directed graph. Mathematically, the complexity **M** is defined as:

$$M = E - N + 2P$$

where

E = number of graph edges

N = number of nodes in the graph

P = the number of connected components

We can extend the formula to programs with multiple exit points,

$$\pi = s + 2$$

Where π = number of decision points in the program, s = number of exit points.

2.3.2 Compilers

Compilers are computer programs that translate computer programs written in a source programming language into another language. As software programs have grown in size, and programming languages have become more user-friendly and thus, more high level, the importance of efficient compilers becomes increasingly important. Recent compiler improvements have added new strategies for compilation such as just-in-time compilation, ahead of time compilation, and trans-compilation. These strategies were formulated to improve the programmer's experience. Hardware compilers are also used to generate integrated circuits from a VHDL or Verilog program. The logic gates corresponding to each defined behavior in the VHDL script are placed on the target IC.

2.3.3 Interpreters

Interpreters are similar to compilers but instead of converting the source language into an output language, they directly execute the instructions which are written in the source language. The language used to implement the interpreter is called the implementation language. The major difference between compilers and interpreters is that a compiler generates machine language and the interpreters execute the instructions without any intermediate code stage [16]. When an error is encountered by an interpreter, it can stop at the error line and report the error details to the programmer.

2.3.4 Virtualization

Virtualization is the process of creating a virtual instance of a computing resource. Most commonly virtualized resources are operating systems, storage devices, network resources. Virtualized resources usually appear transparent to applications using them. There are different levels of virtualizations from full virtualization, where the entire computer hardware is simulated, to paravirtualization which lets the applications see a software-defined interface that allows them to interact with the operating system directly.

In virtualization, a hypervisor is a tool that controls the lifecycle of virtual machines - from creation to running. There are 2 types of hypervisors shown in the figure below.

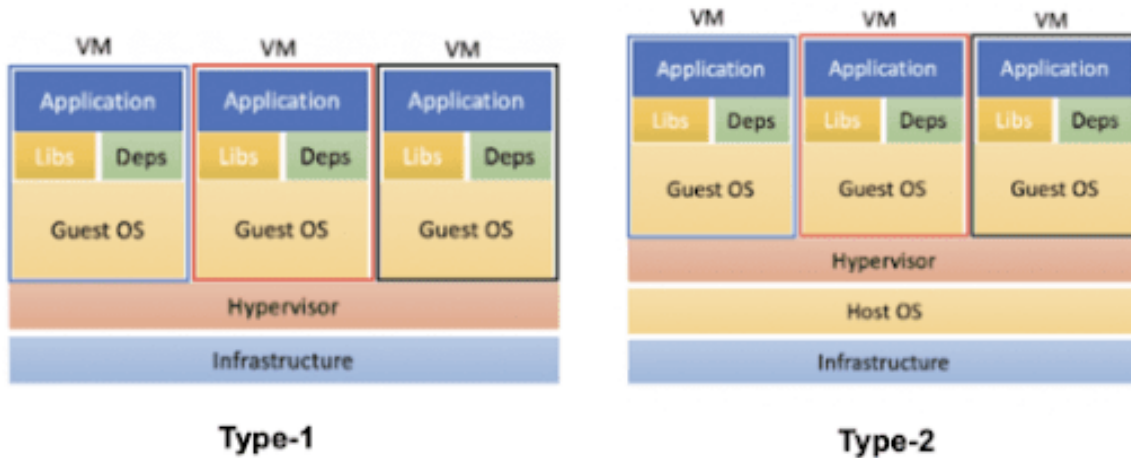


Figure 2.2 - Internals of type 1 and type 2 hypervisors.

Type-1 hypervisors are also known as bare-metal hypervisors and they run directly on the system's hardware. Type-2 hypervisors are software-defined and run inside the operating systems. They exist as a way to manage the virtual machines created.

2.3.5 Integrated Development Environment (IDE)

An Integrated development environment (IDE) is application software that is provided to programmers to help in building software. An IDE has at least, the following components:

1. Source code editor - This acts as an editing area where the software programmer types in code instructions. Modern editors have advanced features such as inbuilt syntax highlighting, autocomplete functionality, and code lint features.
2. Local build automation - Functionality that allows programmers to do frequently performed tasks with ease. Some of these include compiling the code, running unit, and integration tests.
3. Debugger - This is a computer program written with the purpose of testing and discovering bugs in other programmers. It allows the programmer to execute a program under specific controlled conditions. A debugger usually allows the programmer to halt a program at specific points, inspect variables and memory usage at several points, and change variable values in a bid to expose a software bug

The main goal of working with an IDE is to reduce the amount of manual setup and configuration needed when trying to build complex software, requiring several development tools. It provides a set of features as a single, modular unit.

2.3.6 Web Sockets

Web sockets are a computer communication protocol that allows full-duplex communication with a remote host. It enabled real-time communication between multiple clients. Before the introduction of web sockets, technologies such as polling, long polling, and HTTP streaming were used.

Polling involved repeatedly making requests at intervals to fetch the new data and making the request at sufficiently short intervals allowed the system to appear to be real-time. The challenge with polling involved the inability to predict the exact interval which would guarantee getting updates in a timely fashion. HTTP Streaming has the major difference of keeping the TCP connection open and the server can use this to send messages to the client at any time [17].

The communication is usually done over TCP port 443. The protocol URI is defined as `ws` (i.e for unencrypted WebSocket connection) and `wss` (i.e unencrypted WebSocket connection).

Below is a sample code snippet, written in javascript, showing how to implement a web socket client using few lines of code.

```
const sock = new WebSocket('ws://example.com:110010/realtime-changes');
socket.onopen = function () {
  setInterval(function() {
    if (socket.bufferedAmount === 0)
      socket.send(getUpdateData());
  }, 100);
};
```

Figure 2.3 - Sample program showing WebSocket initialization

2.4 Summary

In this chapter, the theoretical foundational concepts including cyclomatic complexity, as well as other tools powering a cloud-based IDE such as compilers, web sockets, virtualization were discussed.

Also, multiple journals based on online compilers, integrated development environments, and collaboration-focused development environments were discussed. After this review, it is realized that there is still room for improvement, especially around the technologies used to facilitate collaborative software development. Several research materials have been studied to gain an understanding of the current state of the field of cloud/online-based IDEs and this project hopes to make advancements to what has currently been done in this area.

CHAPTER THREE

METHODOLOGY

3.1 Introduction

In this chapter, I will focus on the processes taken to develop the online development environment. I also present the limitations of each component of my methodology and also how these limitations are not substantial enough to degrade the validity of the methodology.

3.2 Methodology

The process of developing the online compiler and development environments was carried out in various phases, similar to how traditional software development life cycles work [18]. The phases include:

1. Requirement gathering
2. System analysis
3. Software design
4. Implementation
5. Testing
6. Maintenance

3.3 System Analysis

For the project, after defining the problem statement and gathering requirements, the system analysis stage involved studying the system to be built and identifying the core goals of the system. Some of the core goals of the system to be built are:

1. **Ease of use** - One of the main reasons for working on this project is to create an easy way for collaboration between programmers. Hence, we want the process of collaborating to be as frictionless as possible.
2. **Multiple programming language support** - Usually, programming languages have tasks which they are best suited for. Programmers may want to use multiple programming languages on the same piece of software and as such, my project must be robust enough to support multiple programming languages. This way, users of the software do not get stranded when they need to work with multiple programming languages.
3. **Security** - Programmers treat their programs as confidential as they may contain company trade secrets. Thus, our application needs to handle data with utmost security. This is more important as our application is a collaborative application and we want only parties that have been granted access to collaborate, to be able to collaborate on a given programming project.

A cloud-based IDE requires a high level of system analysis especially on the security and user experience side of things. It is very important that the protection of user files, as well as other data, is given utmost priority during the entire development and maintenance life cycle.

3.4 Software Architecture

The application developed uses the popular architectural pattern known as the layered architecture pattern. The pattern consists of multiple layers, where each layer has a dedicated responsibility within the entire system.

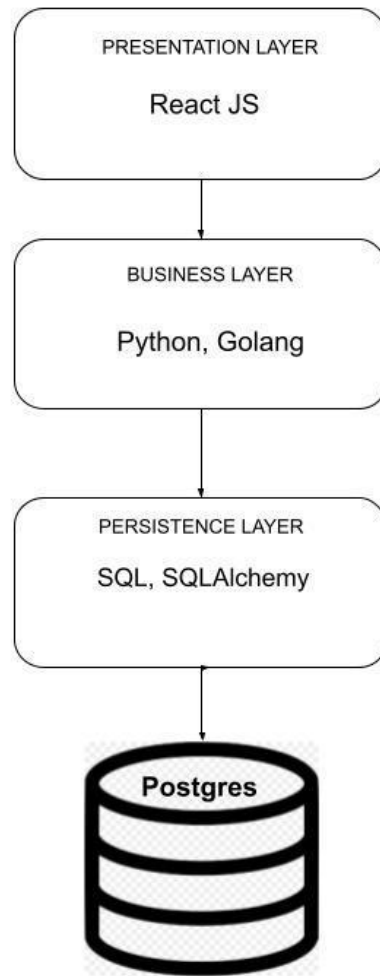


Figure 3.1 - Layered architecture pattern

A very strong advantage of using this pattern is the separation of various concerns among the various pieces in the system. The presentation layer doesn't need to know how the business layer or other layers beneath it operate. Its only job is to present the data to the end-user. This separation of concerns lets us make changes to components in different layers, without affecting components in other layers.

3.5 Programming Languages Supported

For the online compilers and interpreters supported, we would be focusing mainly on the most popular programming languages, as well as some programming languages which are commonly used in the engineering community. These include

1. Python
2. Octave (an open-source equivalent of MATLAB)
3. C++
4. Golang
5. JavaScript

These programming languages which are supported are popular in most day-to-day tasks which programmers perform. However, other programming languages can easily be added and will be added as the project evolves.

3.6 Client-Side

The client-side of this project is built using ReactJS, which is a JavaScript library for building user interfaces. Multiple blocks make up the client-side including

1. Components
2. Services
3. Constants
4. Tests
5. State Management

Components

The components are independent and reusable blocks of code that are used to build up the user interface. The idea of using components is fundamentally based on the branch of software engineering known as Component-based software engineering, which is an approach centered around putting together reusable, loosely coupled independent components to make a system. Some of the components defined in my project are

1. **Dashboard Component**, which serves as a presentation component for many of the smaller components. According to ReactJS terminology, it is a functional component, which means it is implemented using a function as opposed to a class.
2. **Editor Component** implements a full-fledged editor that enables end-users to write the source code of their programs. It is the most complex component and leverages on an open-source editor library known as ***CodeMirror*** to make implementation easier
3. **Login Component** is a component that renders a simple form that enables users to log in to the application using their credentials. It communicates with the backend to ensure that this username and password match each other

Other components include: Loader, Registration

Services

Services refer to a set of software functionality that different clients can use for different purposes. In this project, the clients of the service are the components and they use the services to accomplish basic tasks that are not component-specific.

Two major services within the application will be discussed, namely Authentication & API Services

1. **Authentication** - This is a small service that helps clients to manage the access token of users of the application. Some common functions in the authentication service are *saveAccessToken*, *removeAccessToken*, *buildAuthenticationHeader*.
2. **API Service** - This is a set of services that allows components to be able to talk to the server-side. It provides a clearly defined layer between the components and the server-side. It uses the concept of inheritance to share code across various sub-services which are enclosed within this single service. The core service, *APICore*, serves as a base for other services such as a *UserAPI* and *CodeExecutionAPI*. This service also manages & intercepts the responses before the responses are finally dispatched. This is to ensure responses are consistent and avoid any variation

Constants

This is used to manage values that never change within the application. It consists of asset URLs, error messages, and success messages. A major advantage of using a dedicated constants folder/file is that constants can be easily reused across various parts of the application.

Tests

Testing is an integral component of the client side of this project. We are testing to ensure if the output on the client-side matches the expected requirements. The project makes use of a testing library known as Jest and consists of both Unit & Integration Tests. Manual testing is also done frequently during the development of this project.

State Management

This refers to how we synchronize the state of the application within all components of the application. This project uses MobX to aid with state management at the UI level.

3.7 Server-Side

The server side of this project is implemented using multiple technologies including Python and Go programming language. There are multiple components of the server-side, including

1. REST API
2. Code Execution Engine
3. Database

REST API

This project provides a REST API, built using a python web framework called FastAPI. FastAPI enables the client side to communicate easily with the server side of the application. Communication is over the protocol known as HTTP. The client-side makes an HTTP request to a specified endpoint, using the appropriate HTTP method and the client receives a response in JSON format

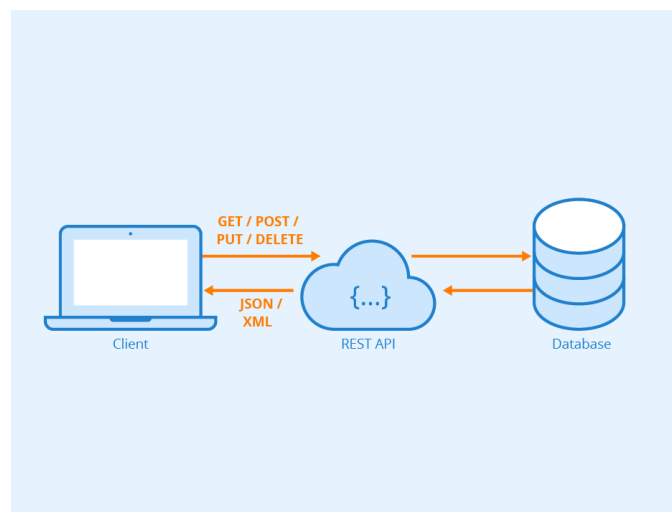


Figure 3.2 - RestAPI communication flow

This project's REST API communicates with the DB directly and provides a clean way for the client-side to retrieve database data.

Code Execution Engine

The code execution engine is a system component that is responsible for the compilation and capturing of the output of any user's program which is submitted. It uses docker heavily to enable OS-level virtualization. OS-level virtualization is important in this project to guarantee that users do not submit malicious code which ends up harming our servers [19]. The code execution engine is implemented using the Go programming language

Database

A single database, PostgreSQL, is used to store all the user information which we want to retain. PostgreSQL is an open-source database with a high level of performance similar to what we want for a highly interactive application like an online compiler / integrated development environment.

3.8 Security

Security is paramount to any software application and must be a high priority. To ensure security of the application, several techniques are employed. These are

3.8.1 Use Of HTTPS

HTTPS is an upgrade to the regular HTTP, which ensures security. It ensures the data transmitted between the sender and the receiver in an application is encrypted. This is particularly important when the data transmitted is sensitive and must not be accessed by a third-party. HTTPS makes use of an asymmetric public key cryptographic system to encrypt the communication channel between the two communicating parties.

3.8.2 Password Hashing

Password hashing is the process of transforming user's passwords into undecipherable strings that are not convertible back into the original plaintext password. This means that hashing is one-way. The reason why password hashing is used in this project is to ensure that in the event

that a bad actor gains access to the database, they are unable to view the passwords of the application's users.

Some commonly used password hashing algorithms are MD5, bcrypt, SHA-256. This project makes use of the bcrypt hashing algorithm

3.9 Software Tools Used

For this project, I will be making use of the following software tools

1. Python programming language
2. FastAPI - a high-performance web framework for building APIs with Python
3. Go programming language
4. Gin - HTTP web framework written in Go
5. Docker - a product that uses OS-level virtualization to deliver software in packages called containers.
6. SQLAlchemy - open-source object relation mapper for python.
7. JavaScript - programming language
8. React - open-source, front end, JavaScript library for building user interfaces
9. MobX - a simple, scalable state management tool
10. PostgreSQL - open-source relational database management system

3.10 Code Compilation & Execution Flow

The client-side application submits the source code and the programming language to the server through the API. The API proceeds to request the code execution engine to create an independent virtualized environment for the current execution context (using docker) and executes the program submitted using an appropriate compiler or interpreter. The program runs inside this isolated environment with constrained memory and CPU time. The program also has an upper bound on the time it is allowed to execute. Upon execution completion or timeout, the result is sent back to the client-side app. The container used is terminated and all the created resources are permanently deleted. No two programmers can view the other's data server-side.

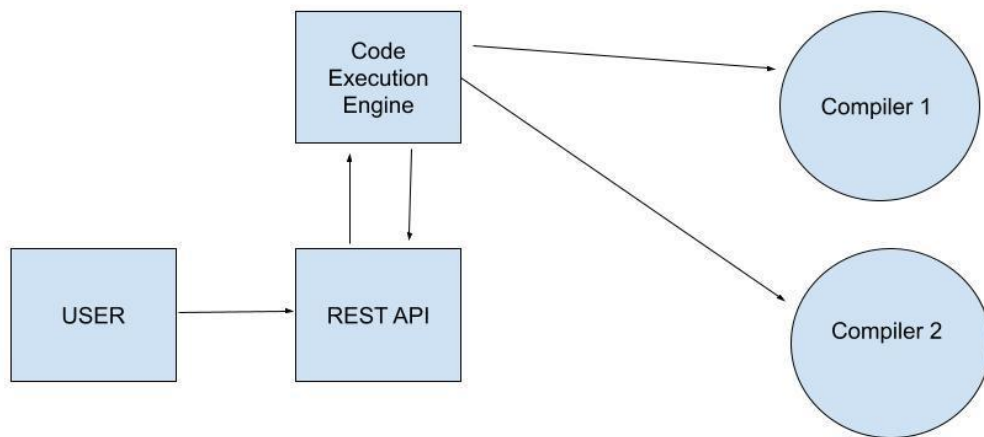


Figure 3.3 - Working schematic of online compiler/ide

CHAPTER FOUR

IMPLEMENTATION

4.1 Introduction

In this chapter, we will discuss the implementations of the final software for this project. The chapter will touch on all the features that were developed for this software, how they were implemented, with screenshots or images where appropriate. The application can be accessed using the URL - <https://compiley.netlify.app>

4.2 Registration

The purpose of this page is to enable the prospective user to create an account that will be used with the application. During implementation, the design philosophy was to collect as little information as possible, and thus we collect 3 things:

1. Email address - for sending relevant information to the user
2. First Name & Last Name - To have something to identify the user by
3. Password - To verify the user's identity during authentication

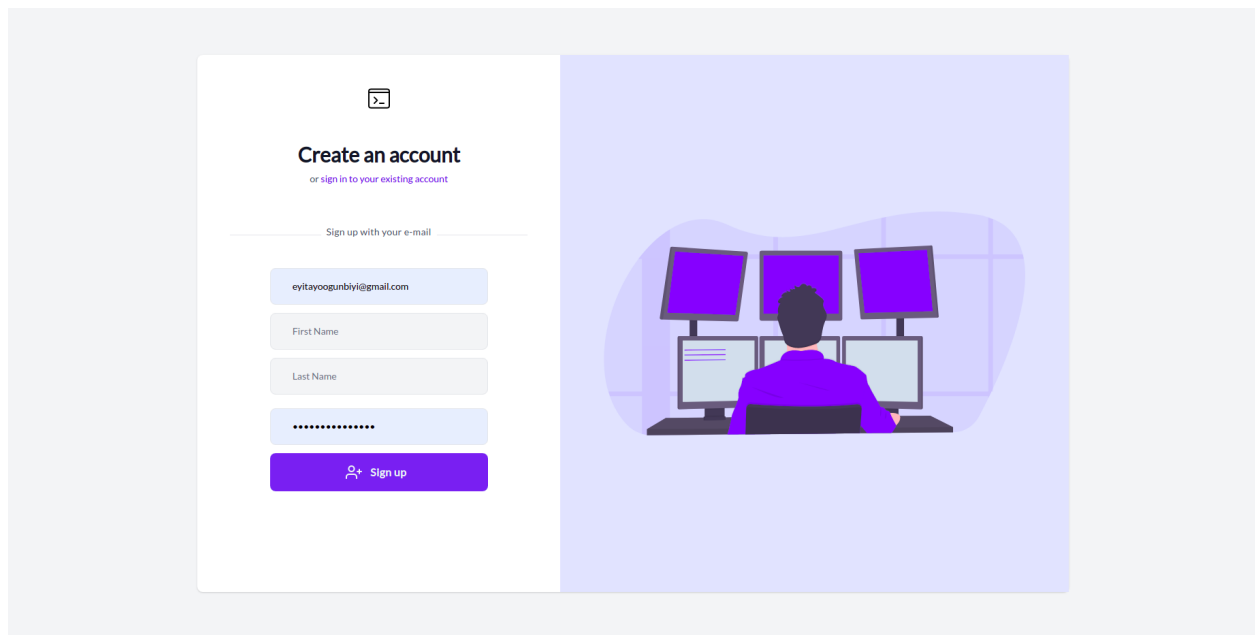
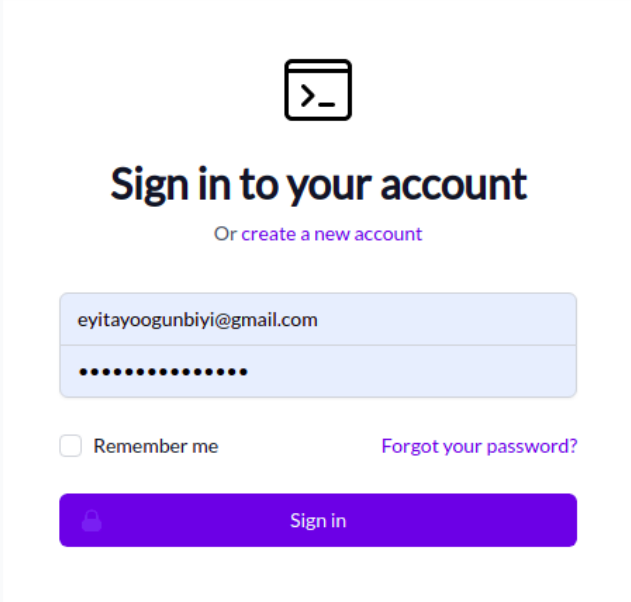


Figure 4.1 - Sign Up Page (<https://compiley.netlify.app>)

4.3 Login

This page is used to authenticate the user, to grant them access to other features of our application. The user has to provide their email address and password and they are admitted into the application if the supplied credentials are correct.

The image shows a login page with a white background centered on a light gray surface. At the top is a terminal icon (a square with a greater-than sign and an underscore). Below it is the heading "Sign in to your account" in bold black text, followed by the link "Or create a new account" in purple. There are two input fields: the first contains the email "eyitayoogunbiyi@gmail.com" and the second contains masked characters (dots). Below the fields is a checkbox labeled "Remember me" and a link "Forgot your password?" in purple. At the bottom is a purple button with a lock icon and the text "Sign in".

Sign in to your account

Or [create a new account](#)

eyitayoogunbiyi@gmail.com

.....

☐ Remember me [Forgot your password?](#)


 Sign in

Figure 4.2 - Login Page (<https://compiley.netlify.app/login>)

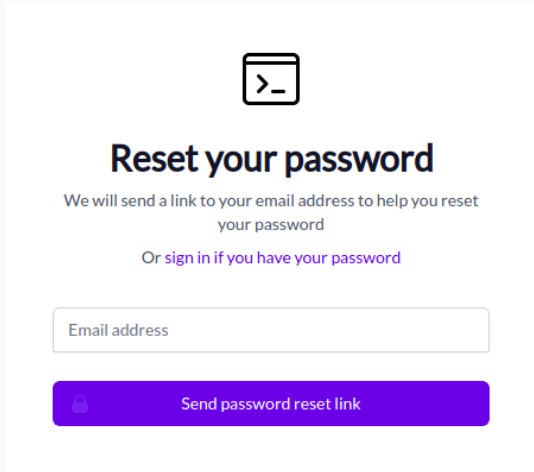
4.4 Forgot Password

Sometimes, users forget their passwords. This can be due to changing their password recently, or not visiting the application for an extended duration of time. The application provides the user with a way to reset their password.

This process consists of two stages:

4.4.1 Requesting a password reset

This involves the user supplying their email address. A message is sent to the supplied mailbox containing a URL which the user can then use to reset their password.



The screenshot shows a web form for resetting a password. At the top is a terminal icon. Below it is the heading "Reset your password". Under the heading, there is a message: "We will send a link to your email address to help you reset your password". Below this message is a link: "Or sign in if you have your password". There is an input field labeled "Email address". Below the input field is a purple button with a lock icon and the text "Send password reset link".

Figure 4.3 - Password reset page (<https://compiley.netlify.app/login>)

4.4.2 Complete the password reset

This is the second and final stage which can be accessed by clicking the URL sent in the supplied email address

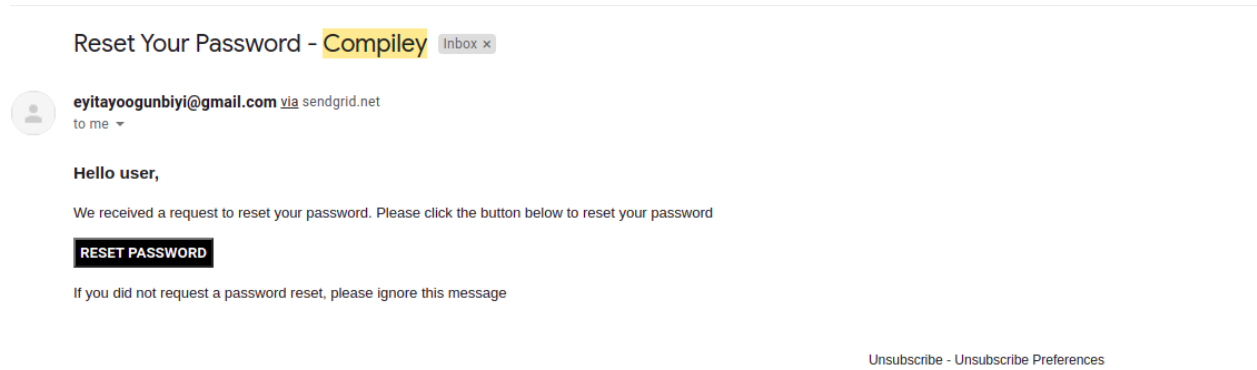


Figure 4.4 -Email received upon password reset

When the link is clicked, the user is presented with a field to supply their new password

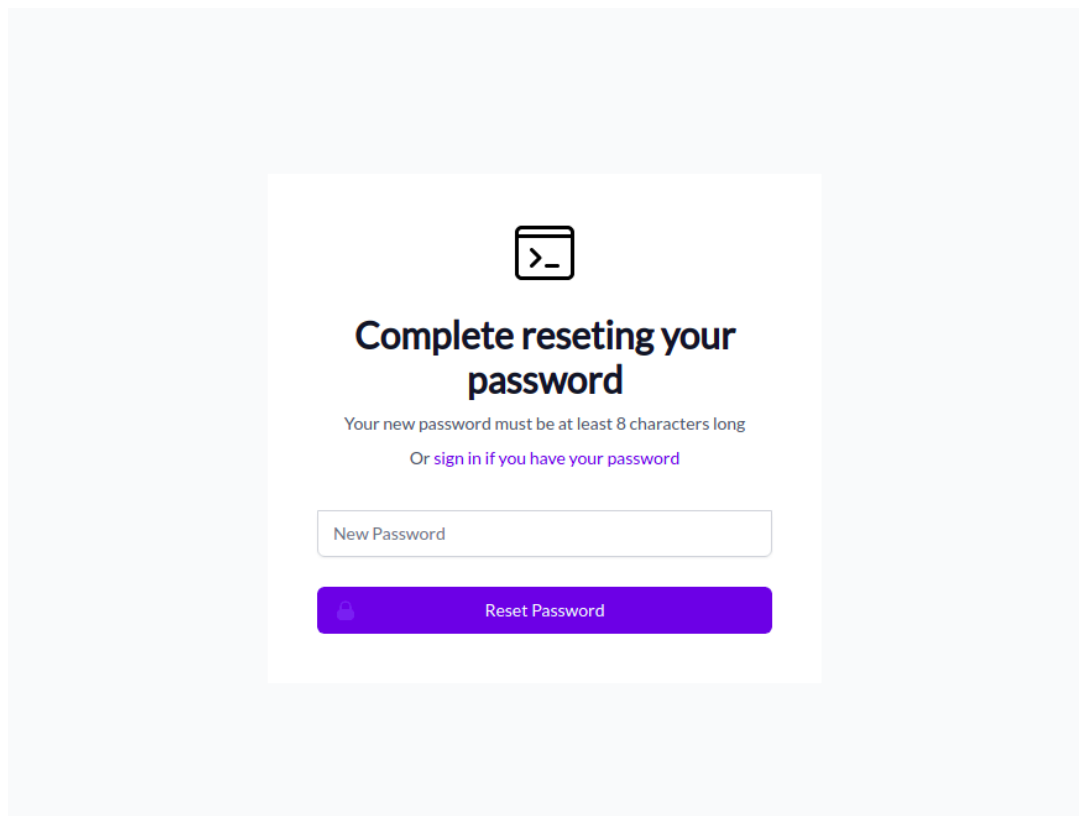
The image shows a web page for completing a password reset. At the top is a terminal icon. Below it is the heading "Complete resetting your password". A note states "Your new password must be at least 8 characters long" and there is a link "Or sign in if you have your password". Below this is a text input field labeled "New Password". At the bottom is a purple button with a lock icon and the text "Reset Password".

Figure 4.5 - Password reset page (<https://compiley.netlify.app/reset-password>)

4.5 Profile

This page shows the user basic information about their account. It allows them to view basic details such as their name, email, the recent code files they have worked with.

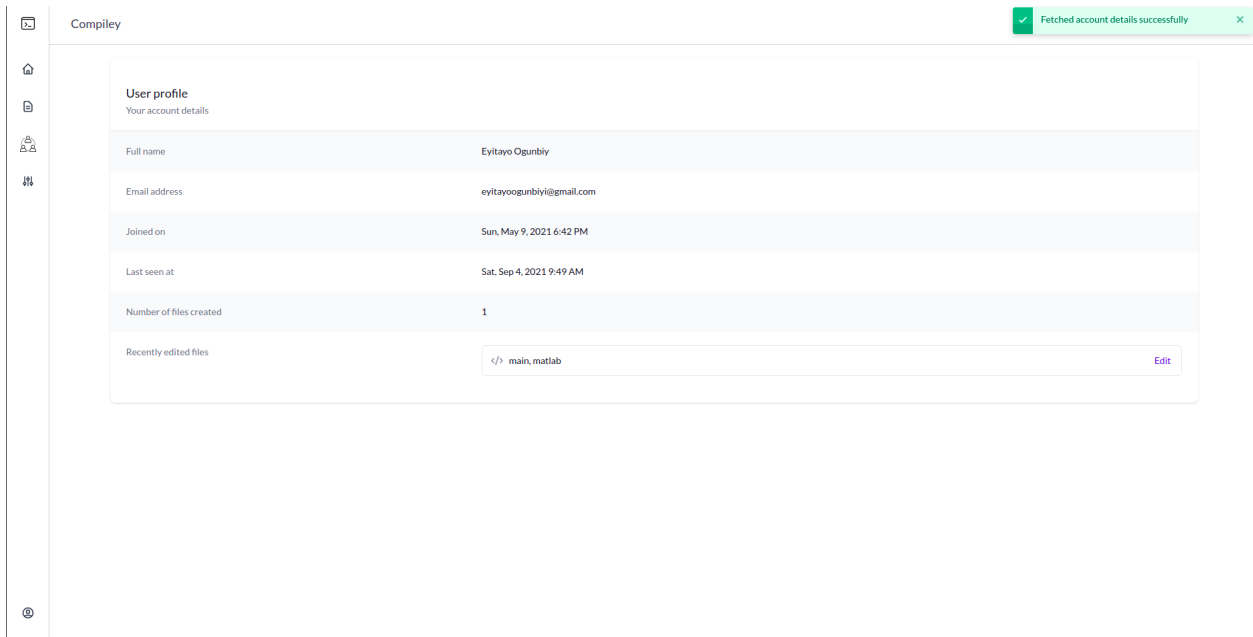


Figure 4.6 - Password reset page (<https://compiley.netlify.app/app/profile>)

4.6 Application Dashboard

When a user logs in to the application, there are several actions they can perform. The application dashboard presents all the application's core features to the users in an easy-to-understand way.

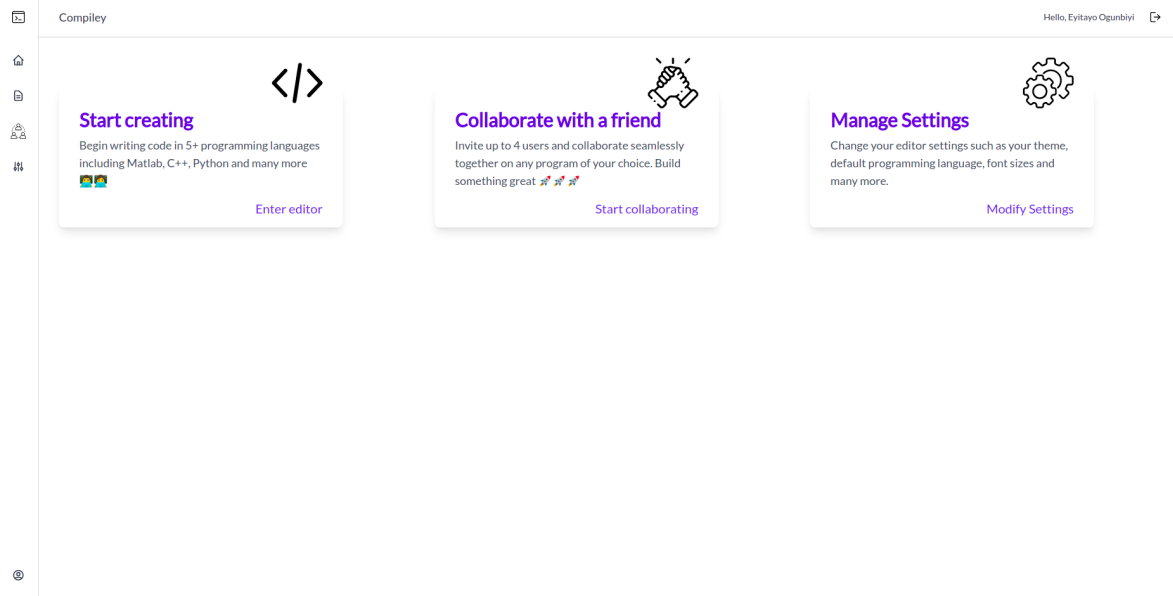


Figure 4.7 - Application dashboard (<https://compilify.netlify.app/app>)

It also allows users to log out of the application at the top right and many other actions.

4.6 Code Editor

The code editor is a core feature of the application and the application can not exist without it. It allows the user to type in programs of their code, execute the written code and debug if any errors occur. It supports up to six programming languages (Python, C++, Golang, JavaScript, Octave, and Matlab)

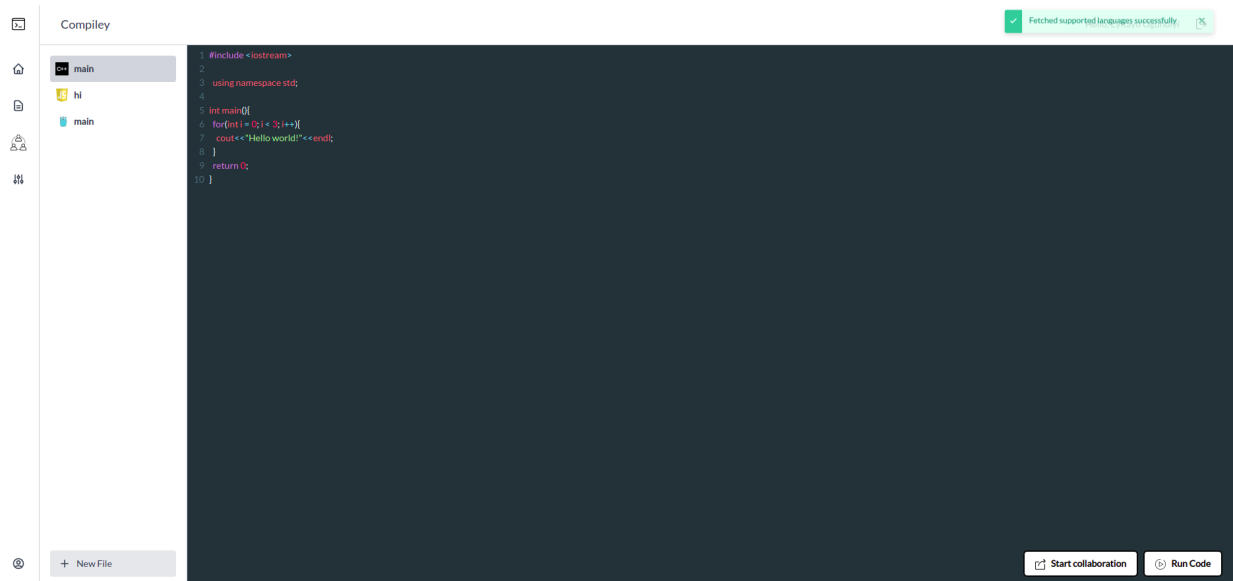


Figure 4.8 - Code Editor (<https://compiley.netlify.app/app>)

4.6.1 - File Creation

Creating a new file can be triggered using the editor. On the bottom left, there's a button to create a new file. Upon clicking, a modal is opened asking for the details of the file we wish to create.

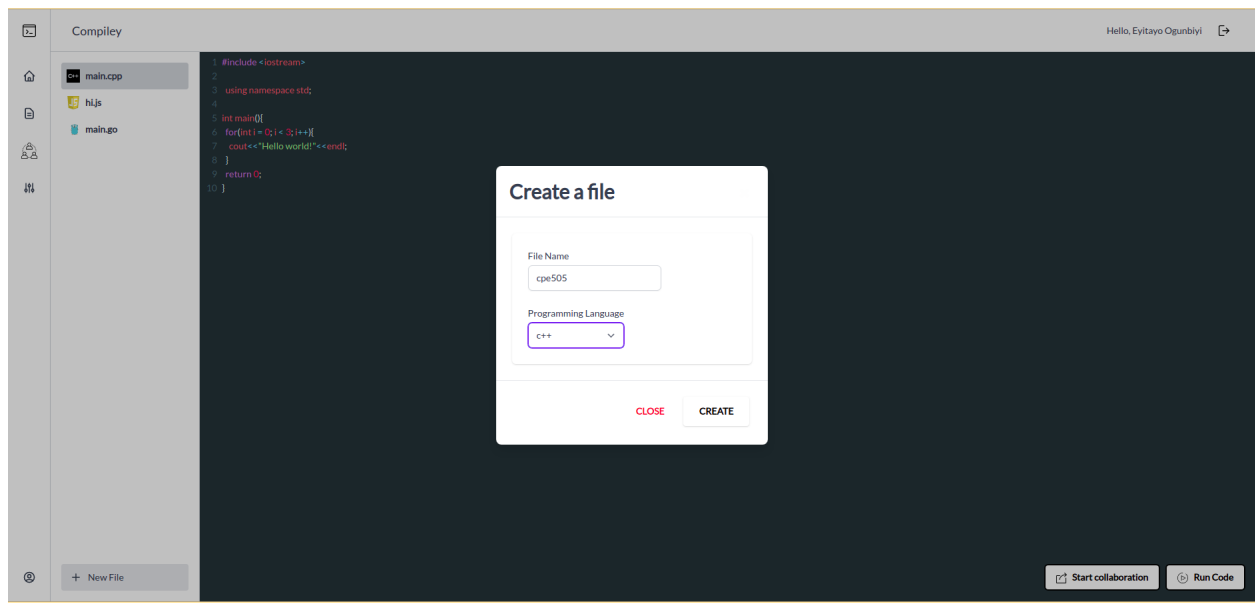


Figure 4.9 - File Creation (<https://compiley.netlify.app/app>)

4.6.2 - Code Execution Result

Upon writing the program of our choice, to execute it, we can click the “Run Code” button in the bottom right of the code editor.

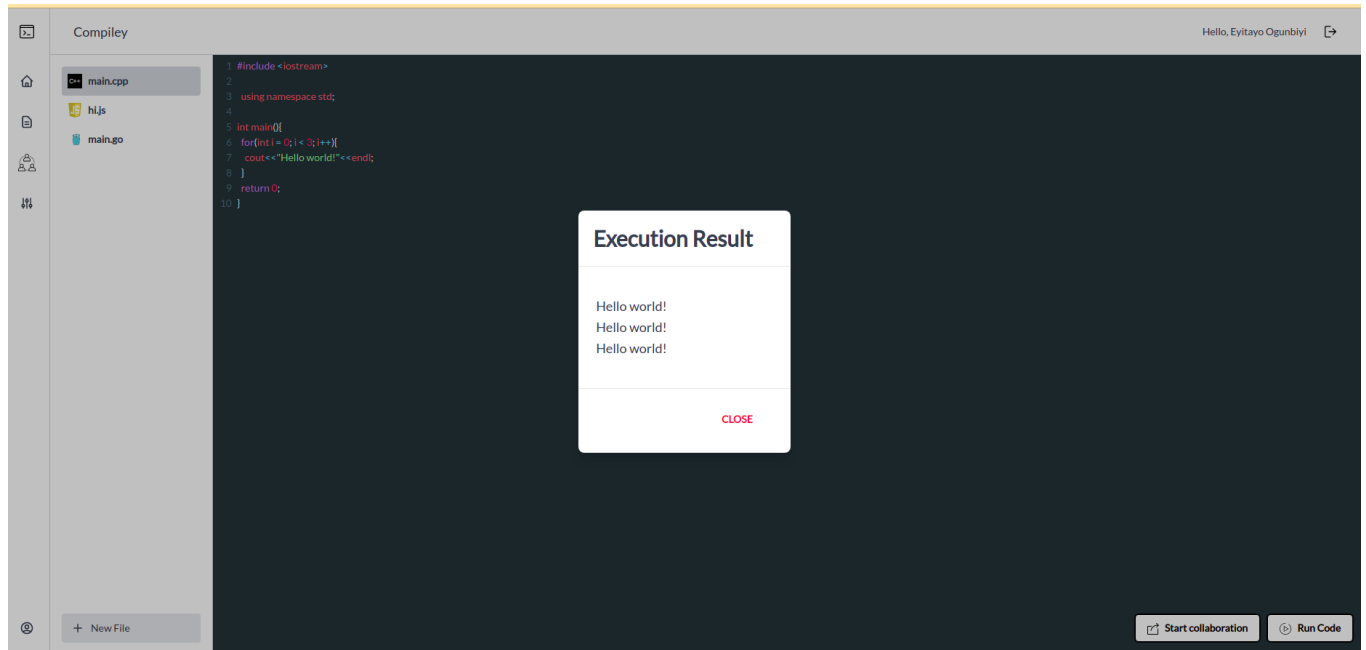


Figure 4.10 - Code Execution Result (<https://compiley.netlify.app/app>)

4.7 Code Editor - Collaboration Mode

The code editor developed allows multiple users to collaborate on a single file by using a single URL. A user can click the button that says “Start Collaboration” in the bottom right of the code editor to generate a collaboration URL.

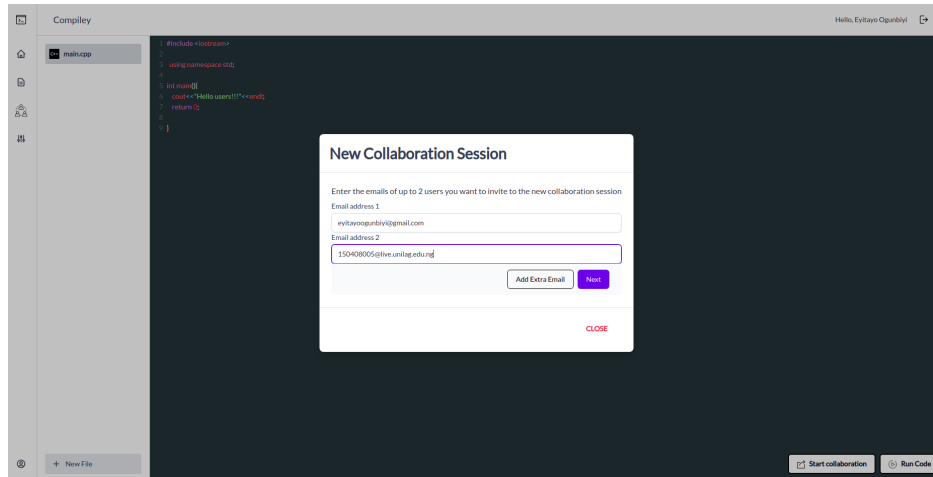


Figure 4.11a - Starting Collaboration (<https://compiley.netlify.app/app>)

Upon entering the emails of the users you intend to share the file with, the users get an email inviting them to join the collaboration.

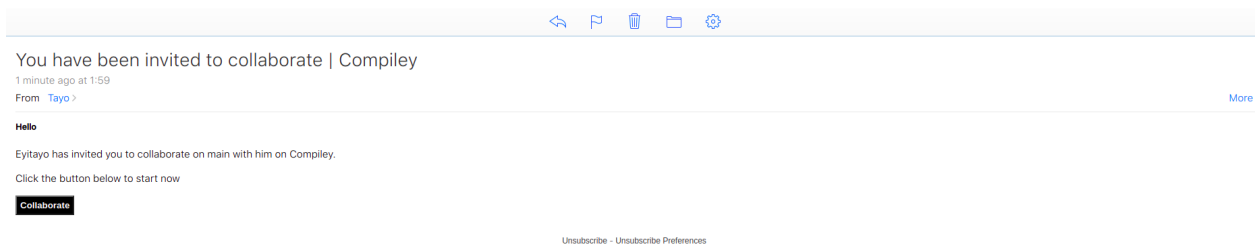


Figure 4.11b - Collaboration Email Invite (<https://compiley.netlify.app/app>)

When they follow the link, they can now work on the same file with you, the sharer, in real-time.

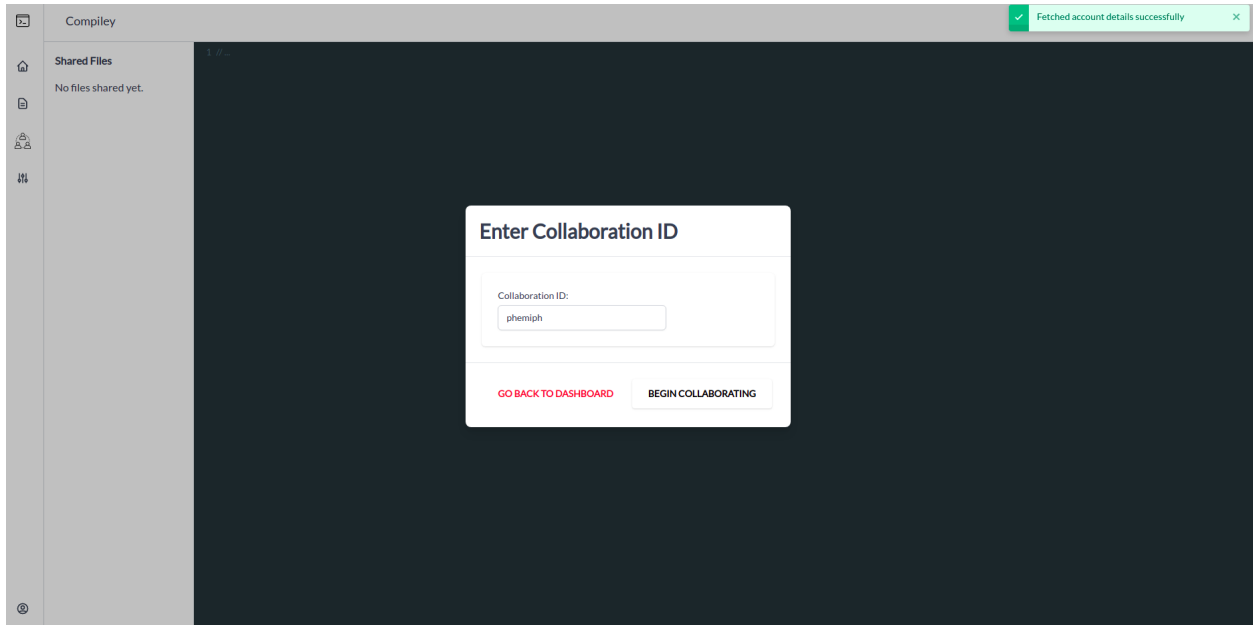


Figure 4.11c - Enter collaboration ID (<https://compiley.netlify.app/app>)

4.8 Database Tables

The application built consists of several database tables. Each of the database tables exists within a relational database (PostgreSQL) and relationships exist within the different tables. The database also makes use of indexes and primary keys to improve the speed of queries.

An ER diagram showing the relationship between the tables in the database is shown below:

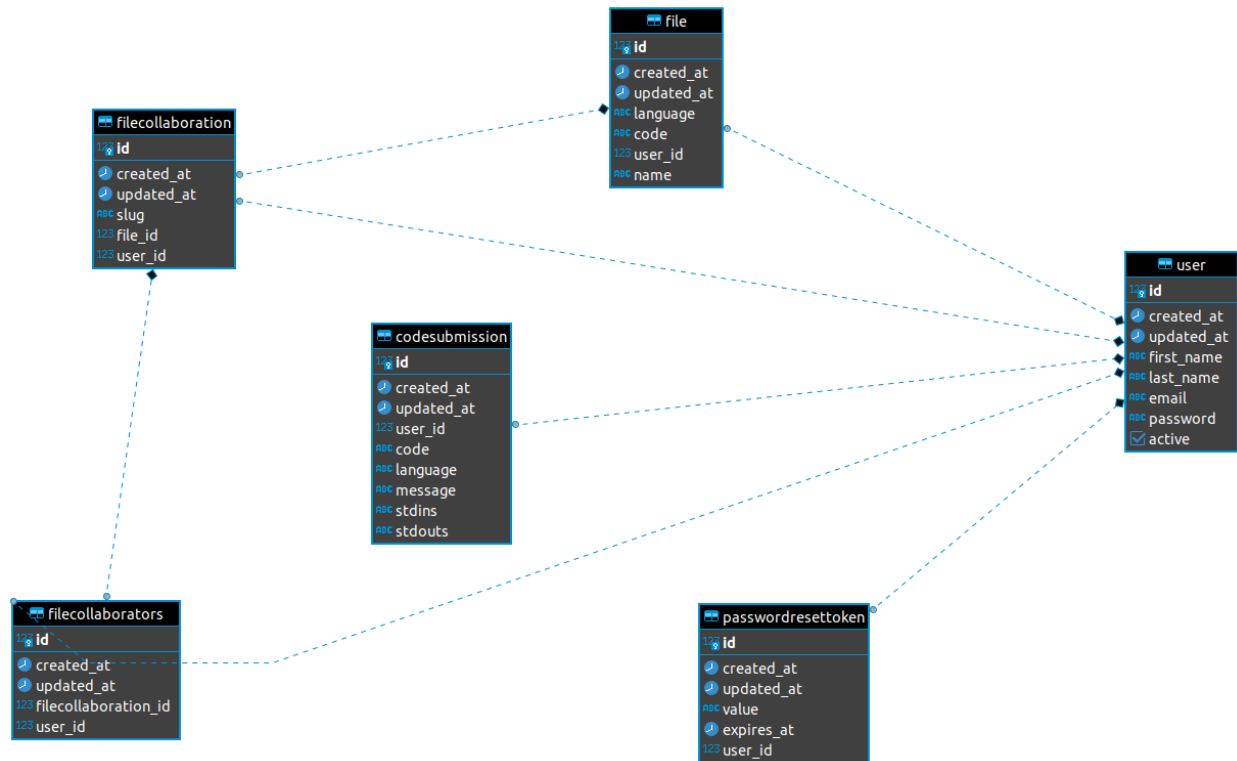


Figure 4.12- Database ER Diagram

4.8.1 - Users Table

This table holds the information of all the application users. It includes the id as the primary key, along with other fields like the email address, first name, last name, and a hashed version of the password

The screenshot shows a database application interface with the 'user' table selected. The table has the following columns: created_at, updated_at, id, first_name, last_name, email, password, and active. Two rows of data are visible.

created_at	updated_at	id	first_name	last_name	email	password	active
2021-09-01 06:56:34	2021-09-01 09:16:55	1	Eyitayo	Ogunbiyi	eyitayoogunbiyi@gmail.com	\$2b\$12\$ekjCH0k5.R3JP52AwMwOnowWkC	[v]
2021-09-04 08:22:08	2021-09-04 08:22:08	2	Eyitayo	Ogunbiyi	eyitayoogunbiyi@icloud.com	\$2b\$12\$5SLgjh7VJUn1xhwQ/vaiXuLJPQHov	[v]

Figure 4.13- User Table

4.8.2 - Files Table

This table holds information on all the source code files which the user has created.

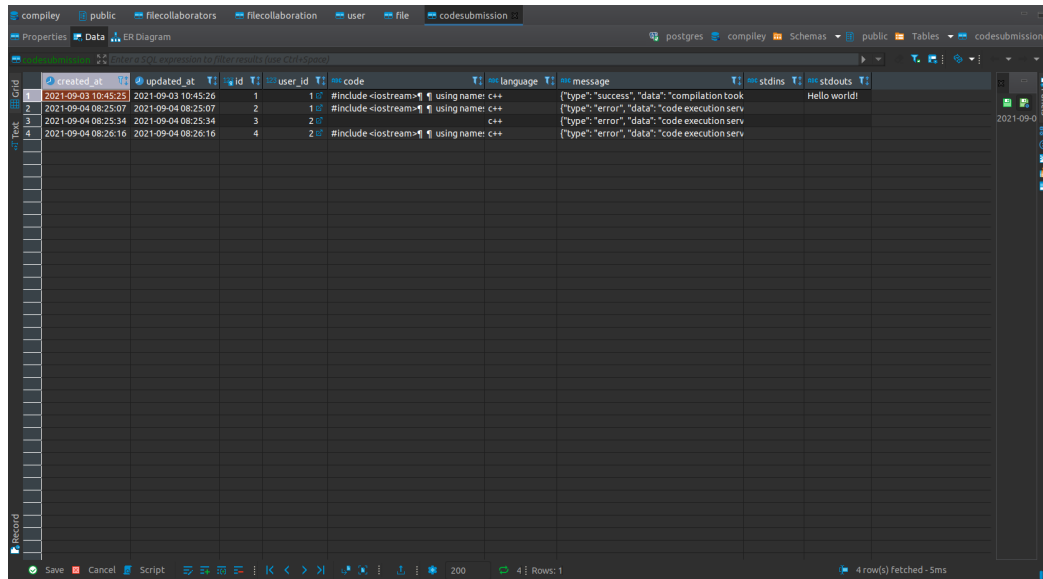
The screenshot shows a database application interface with the 'file' table selected. The table has the following columns: created_at, updated_at, id, language, code, user_id, and name. One row of data is visible.

created_at	updated_at	id	language	code	user_id	name
2021-09-03 10:44:58	2021-09-04 08:26:16	1	C++	#include <iostream> using namespace std; int main() { return 0; }	1	main

Figure 4.14- Files Table

4.8.3 - Code Submissions Table

This table shows all the attempted submissions the application users have made. When a user attempts to submit a code snippet, it stores the information in this table.

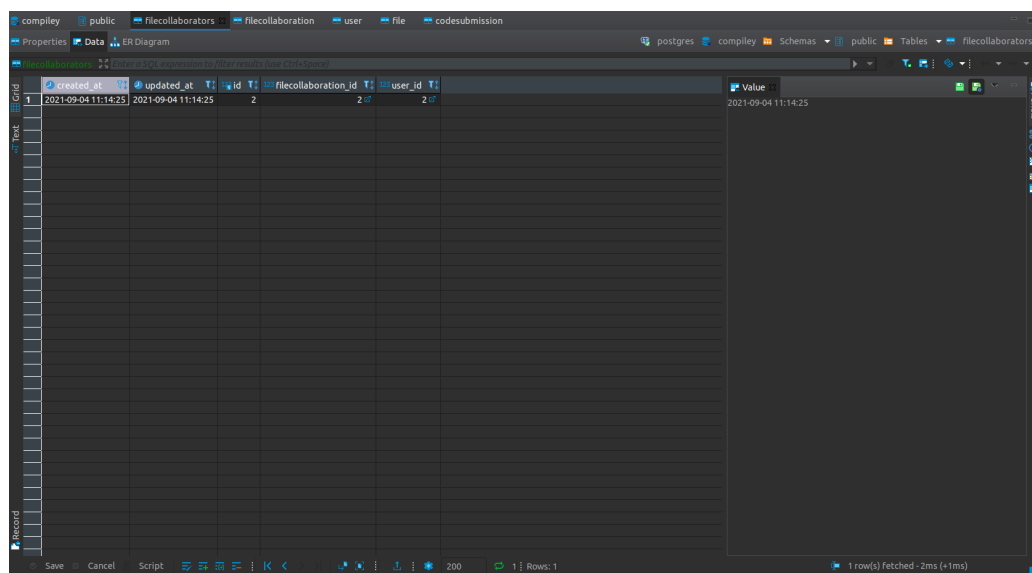


	created_at	updated_at	id	user_id	code	language	message	stdins	stdouts
1	2021-09-03 10:45:25	2021-09-03 10:45:26	1	1	#include <iostream> using namespace std;	C++	["type": "success", "data": "compilation took"]		Hello world!
2	2021-09-04 08:25:07	2021-09-04 08:25:07	2	1	#include <iostream> using namespace std;	C++	["type": "error", "data": "code execution serv"]		
3	2021-09-04 08:25:34	2021-09-04 08:25:34	3	2		C++	["type": "error", "data": "code execution serv"]		
4	2021-09-04 08:26:16	2021-09-04 08:26:16	4	2	#include <iostream> using namespace std;	C++	["type": "error", "data": "code execution serv"]		

Figure 4.15- Code submission Table

4.8.5 - File Collaborators Table

This table stores a list of all the users invited to collaborate on a given file.

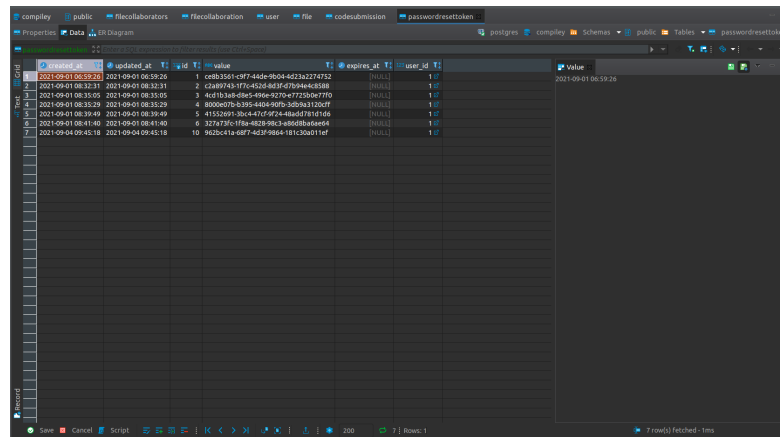


	created_at	updated_at	id	filecollaboration_id	user_id
1	2021-09-04 11:14:25	2021-09-04 11:14:25	2	2	2

Figure 4.16- File Collaborators Table

4.8.6 - Password Reset Token Table

This table is what enables users to reset their passwords. When a password reset request is made, a token is generated and stored in this table. This token is then used to validate the password reset request.



	id	updated_at	expires_at	value	user_id
1	1	2021-09-01 06:59:26	2021-09-01 06:59:26	ce8b3561-c9f7-445e-80d4-4d23a2274732	10
2	2	2021-09-01 08:32:21	2021-09-01 08:32:21	6489f45b-1f76-4506-803f-47b9a4c008e	10
3	3	2021-09-01 08:35:05	2021-09-01 08:35:05	84c1b3a4-d6e5-496e-9270-4772b0ae77f0	10
4	4	2021-09-01 08:35:29	2021-09-01 08:35:29	8000e07b-3395-40d4-90fb-3d9bc312dcff	10
5	5	2021-09-01 08:39:49	2021-09-01 08:39:49	41532d93-35a4-47d7-9f24-4dad5f91c346	10
6	6	2021-09-01 08:41:40	2021-09-01 08:41:40	327a73fc-1f8a-4028-98c3-48d8d8a6a9d4	10
7	7	2021-09-04 09:43:18	2021-09-04 09:43:18	9d2bc41a-68f7-4d3f-9864-181c30ae31af	10

Figure 4.17- Password reset token table

4.9 Performance Graphs

The following graphs show the performance of the servers on which the application was deployed.

The data on the plot obtained is obtained during active use of the application and this usage can be thought of as peak usage for the purposes of this project

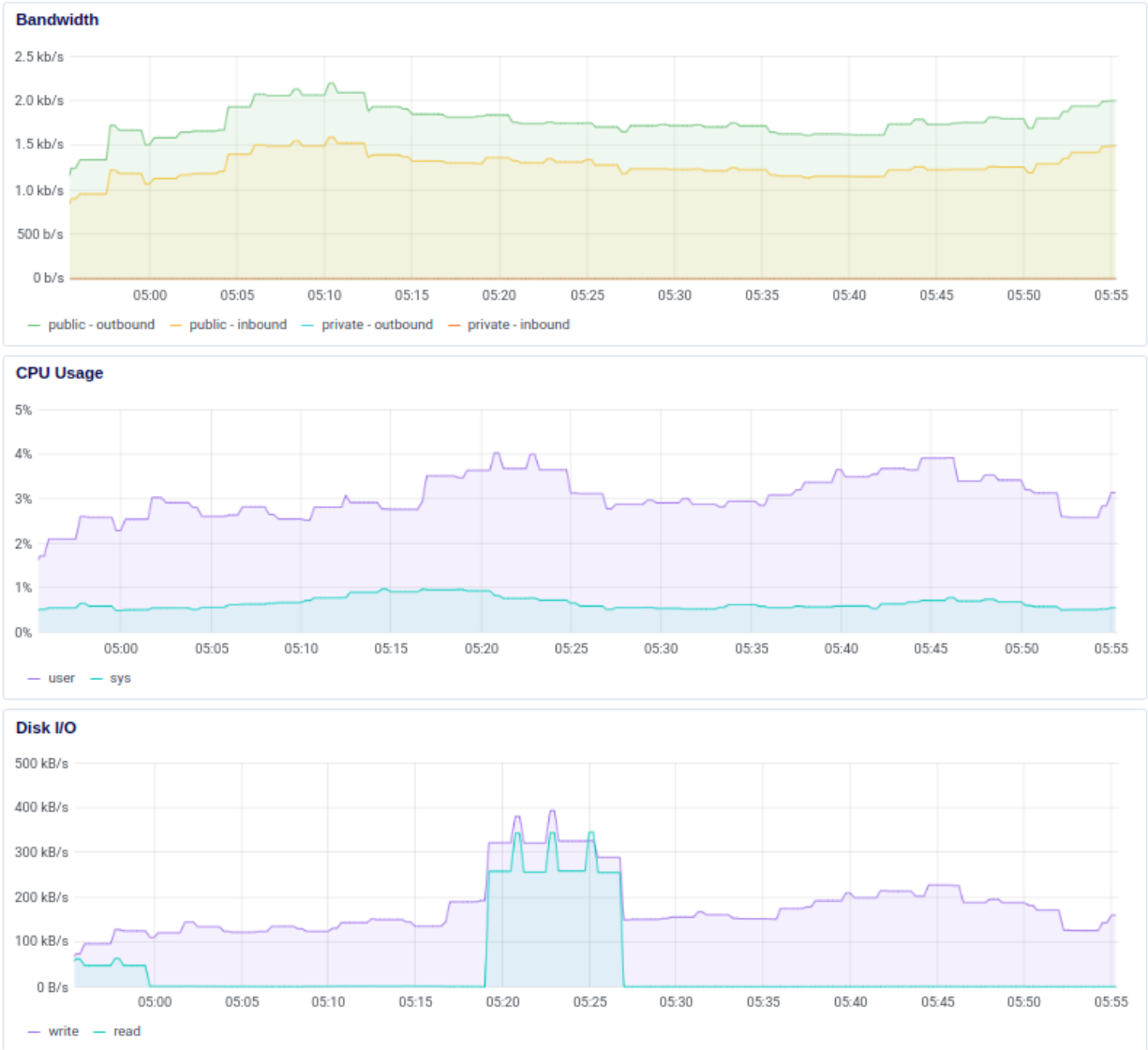


Figure 4.18- CPU, Bandwidth and Disk I/O Usage Graphs

4.10 Comparison To Related Work

In comparison with related work, just like Wiemman D. et al [4] this project exploited cloud computing and virtualization. Most notably, this project was able to show how powerful virtualization technologies such as Docker can be used to create truly isolated software environments.

Also, upon observing Mayank Patel [12], who worked on an Online Java Compiler, this project provides support for several more programming languages, most notably Octave/Matlab.

As room for improvement, this project can employ the techniques used by L'uboš C. and Martin K [2] to improve our software to support the development of more advanced block schemes within the browser.

CHAPTER 5

CONCLUSION

5.1 Introduction

In this chapter, some concluding remarks regarding the project will be discussed. Challenges faced, as well as suggestions for improvements will also be discussed.

5.2 Concluding Remarks

At the start of this project, the objectives of the study were fully listed out and research was carried out in the direction of the listed objectives. Examining previous literature by authors who had worked on related projects proved to be of great assistance. Some of this previous work had techniques and principles that broadly applied to this project and as such, applying the core ideas in them significantly improved the quality of my research, as well as led my implementation direction.

The web application was developed using a wide range of technologies, ranging from using React, a JavaScript user interface library, for the frontend of the application, to using Python for the backend of the application. Execution of user-submitted code was done using the go programming language. The database used is PostgreSQL, a powerful and free relational database. Deployment of the application was enabled by Heroku and DigitalOcean. The code editor (shown in chapter four) built into the application is an enhanced version of CodeMirror, optimized for collaborative editing.

In conclusion, this project aimed to come up with a software solution that enables multiple persons to collaborate on a single programming project, in near real-time and we can confidently say this aim has been achieved.

5.3 Challenges Faced

5.3.1 - Integrating code editor for collaboration

Integrating the code editor for collaboration turned out to be challenging as codemirror was created for regular javascript integration and not for more complex integrations like building collaborative editing tools.

After research, an open-source binding library that enables more powerful functionalities on Codemirror was discovered. It allows the project to have synchronized code mirror editors across multiple users, shared cursors, and recovery upon document divergence.

5.3.2 - Reducing Collaboration Latency

When multiple engineers are working together on a project, they want to be able to see the changes made by their work partner as quickly as possible and thus, a major challenge faced was reducing the latency between when a user makes a change to the code file and when the other collaborators can see this change.

Ideally, we want this time to be less than a second and to enable this, the project uses a standalone server solely for handling the WebSocket connections between multiple users. This server can then be scaled independently to meet the needs of the users.

5.4 Suggestions For Improvement

5.4.1 - Source Code Storage

Currently, the source code submitted by users is stored within the database. The problems with this are:

1. The difficulty of searching the source code
2. Source code files can grow large, and get too large to be stored in a single database column

A much better way of handling the source code files would be with object storage. The file can be saved in an object storage system like Amazon S3 and can be retrieved, indexed, and searched with ease.

5.4.2 UX Improvements (User Experience)

Some users had challenges in figuring out how to use the software on their own. Manually instructing them on how to use the software can not scale as users increase. Hence, performing more user research on the best way to arrange content on the software would improve the user experience as well as increase the marketability of the software.

REFERENCES

- [1] Yi Wang, “Characterizing Developer Behavior in Cloud-Based IDEs”, *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2017, pp. 48-51.
- [2] L’uboš C. and Martin K, “A Web-based Tool for Design of Simulink Models”, *21st International Conference on Process Control (PC)*, 2017. pp. 92-94
- [3] Budi Y. et Al, “Harmonik = ++(Web IDE)”, *2nd International Conference on Computer Science and Computational Intelligence ICCSCI*. 2017. pp 222-225
- [4] Weimin D. et al, “Research on the Virtualization Technology in Cloud Computing Environment”, *International Journal of Engineering Research in Africa*, 2016.
- [5] G. Fylaktopoulos et Al, “An overview of platforms for cloud-based development”, SpringerPlus, Open Journal, 2016. DOI: [10.1186/s40064-016-1688-5](https://doi.org/10.1186/s40064-016-1688-5)
- [6] Ratnadip K. et al, “Online Editor for Compiling and Executing Different Languages Source Code”, *International Journal of Advanced Research in Computer Science and Software Engineering*, 2016
- [7] Aditya K. et al, “CodeR: Real-time Code Editor Application for Collaborative Programming”, *International Conference on Computer Science and Computational Intelligence (ICCSCI 2015)* 2015. pp. 510-511
- [8] Surya C. et Al, “ONLINE C, C++ & JAVA COMPILERS USING CLOUD COMPUTING”, *IJCSMC, Vol. 4, Issue. 8, August* 2015. pp. 348-355

- [9] Stephan K. and Bernd B, “User Feedback in Mobile Development”, 2014
- [10] Arjun D. and Arnab K, “Online Compiler as a Cloud Service”,*2014 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*. 2014
- [11] Mutiara A.B et al, “DEVELOPING A SAAS-CLOUD INTEGRATED DEVELOPMENT ENVIRONMENT (IDE) FOR C, C++, AND JAVA ”, 2014
- [12] Mayank Patel, “Online Java Compiler using Cloud Computing”, *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 2013
- [13] Timo A. et al, “Designing IDE as a Service”, 2012
- [14] Ling W. et al, “CEclipse: An Online IDE for Programing in the Cloud ”, *2011 IEEE World Congress on Services*, 2011. pp. 45-46.
- [15] Max Goldman et al, “Real-Time Collaborative Coding in a Web IDE”, *UIST’11, October 16–19, 2011, Santa Barbara, CA, 2011*. pp 798-801.
- [16] Fan Wu et al, “Design and Implementation of an Interpreter Using Software Engineering Concepts”, *International Journal of Advanced Computer Science and Applications*, 2014.
- [17] Qigang Liu, Xiangyang Sun, “Research of Web Real-Time Communication Based on Web Socket”, *Int. J. Communications, Network and System Sciences*, 2012, 5, 797-801, 2012
- [18] Navita, “A Study on Software Development Life Cycle & its Model”, *International Journal of Engineering Research in Computer Science and Engineering (IJERCSE) Vol 4, Issue 9*, 2017

[19] František Špaček, Radomír Sohlich, Tomáš Dulík, “Docker as Platform for Assignments Evaluation”, *25th DAAAM International Symposium on Intelligent Manufacturing and Automation, DAAAM, 2014, pp. 1.*

[20] Figure 2.1, *Internals of type 1 and type 2 hypervisors*. Accessed 03/05/2021.

<http://www.techplayon.com/what-is-hypervisor/>

[21] Figure 2.2, *Sample program showing WebSocket initialization*. Accessed 03/05/2021.

<https://en.wikipedia.org/wiki/WebSocket>

[22] Figure 3.2, *Rest API Flow*. Accessed 03/05/2021.

https://www.seobility.net/en/wiki/REST_API