

Software Design Document (SWDD) Template

Background

Software design is a process by which the software requirements are translated into a representation of software components, interfaces, and data necessary for the implementation phase. The SWDD shows how the software system will be structured to satisfy the requirements. It is the primary reference for code development and, therefore, it must contain all the information required by a programmer to write code. The SWDD is performed in two stages. The first is a preliminary design in which the overall system architecture and data architecture is defined. In the second stage—i.e., the detailed design stage—more detailed data structures are defined and algorithms are developed for the defined architecture.

This template is an annotated outline for a software design document adapted from the *IEEE Recommended Practice for Software Design Descriptions*. The *IEEE Recommended Practice for Software Design Descriptions* have been reduced in order to simplify this assignment while still retaining the main components and providing a general idea of a project definition report. For your own information, please refer to IEEE Std 1016¹ for the full *IEEE Recommended Practice for Software Design Descriptions*.

¹ Available to LANL users at <http://ieeexplore.ieee.org/browse/standards/collection/ieee>

SWDD- [obtain number from Conduct of Engineering
Document Numbering [SharePoint site](#)]

Thoth

Software Design Document

Name (s): Dana Alkhouri - Bemen Girgis - Ziad Hassan - Layla Hussein - Shaza Ali - Omar Bahgat
Section: 1

Date: (9/12/2023)

TABLE OF CONTENTS

Software Design Page	1
1.0 INTRODUCTION	4
1.1 Purpose	4
1.2 Scope	4
1.3 Overview	4
1.4 Reference Material	
2.0 SYSTEM OVERVIEW	
3.0 SYSTEM ARCHITECTURE	
3.1 Architecture Design	
3.2 Decomposition Description	
4.0 DATA DESIGN	
4.1 Data Description	
5.0 COMPONENT DESIGN	
6.0 HUMAN INTERFACE DESIGN	
6.1 Overview of User Interface	
6.2 Screen Images	

1.0 INTRODUCTION

1.1 Purpose

This software design document delineates the functionality of our C++ language whose primary objective is to facilitate the comprehension of the language for everyone.

1.2 Scope

This software aims to create an engaging and customized learning environment for students, emphasizing visual interactions with a variety of C++ algorithms. The program is equipped with the ability to track student performance through assessments designed to evaluate their understanding and progress in the learning journey. Additionally, it provides lessons and YouTube videos to enhance comprehension and clarity for users.

1.3 Overview

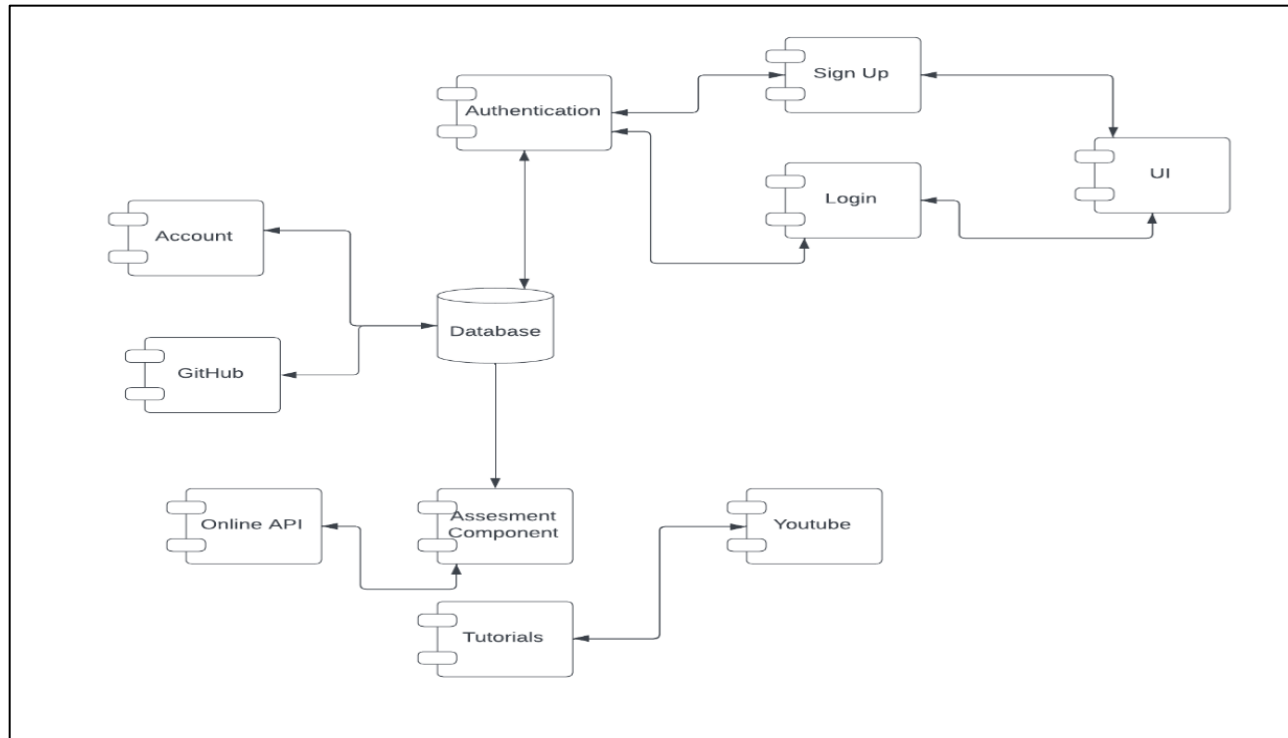
Thoth is a specialized C++ learning tutorial program that employs assessments, lessons, and videos to create an engaging and tailored learning environment. Focused on software engineering, Thoth provides features a robust assessment mechanism through purpose-built quizzes, enhancing the overall learning experience by systematically evaluating student performance. Thoth seamlessly integrates these elements to offer a concise and effective pathway for mastering C++.

2.0 SYSTEM OVERVIEW

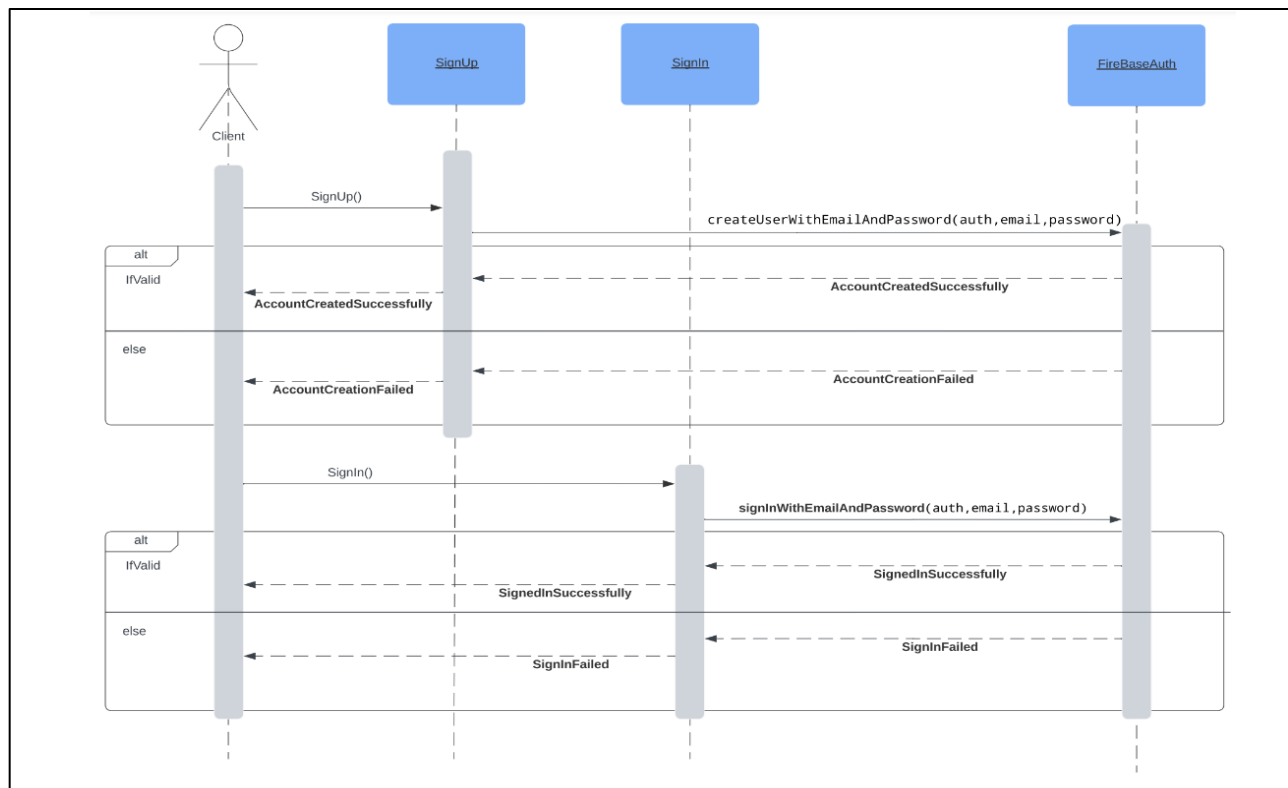
Designed as a web application using React, HTML, and CSS, the user interface (UI) comprises page activities that act as orchestrators for various essential backed features in our system. This design places the UI as a pivotal component responsible for communicating with the majority of other system components. It is structured as a road map, incorporating several quizzes, videos, and exercises that participants must complete before reaching the tutorial's conclusion.

3.0 SYSTEM ARCHITECTURE

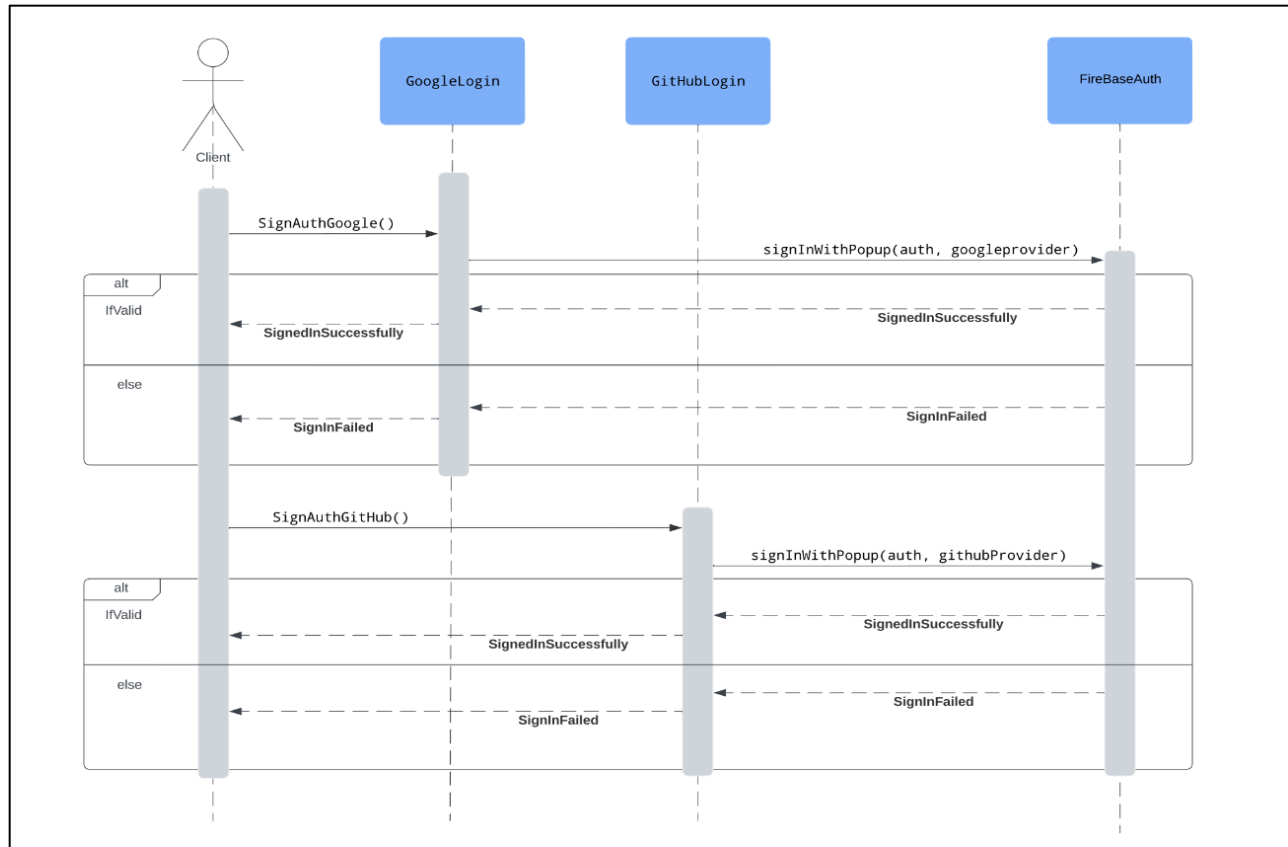
3.1 Architectural Design



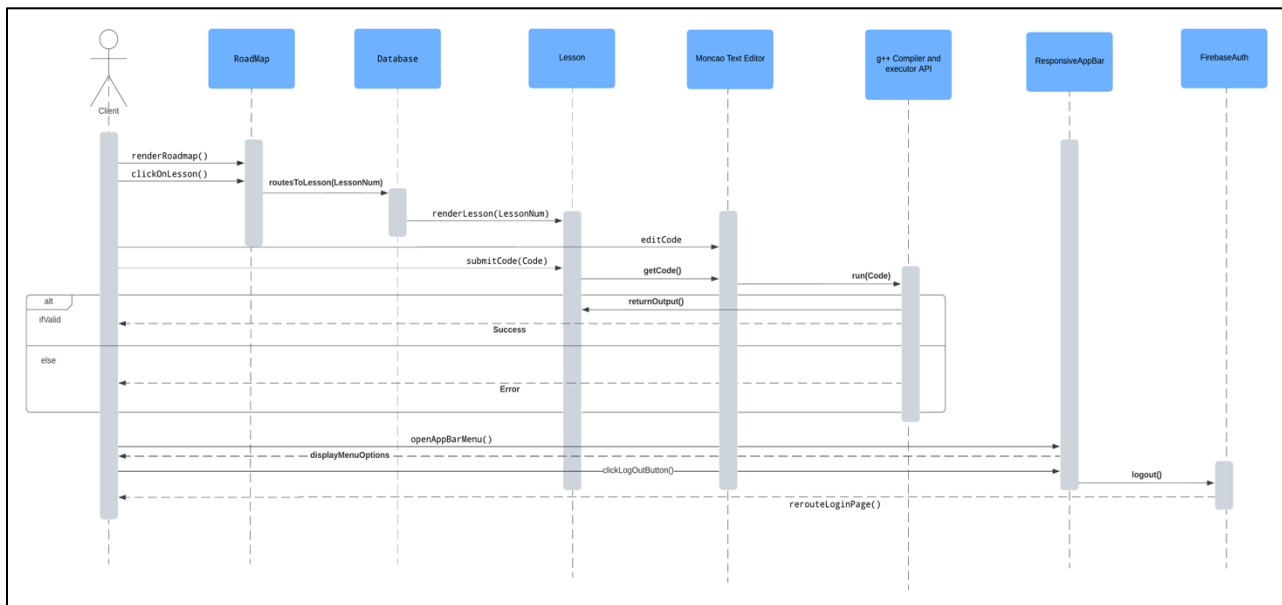
3.2 Decomposition Description



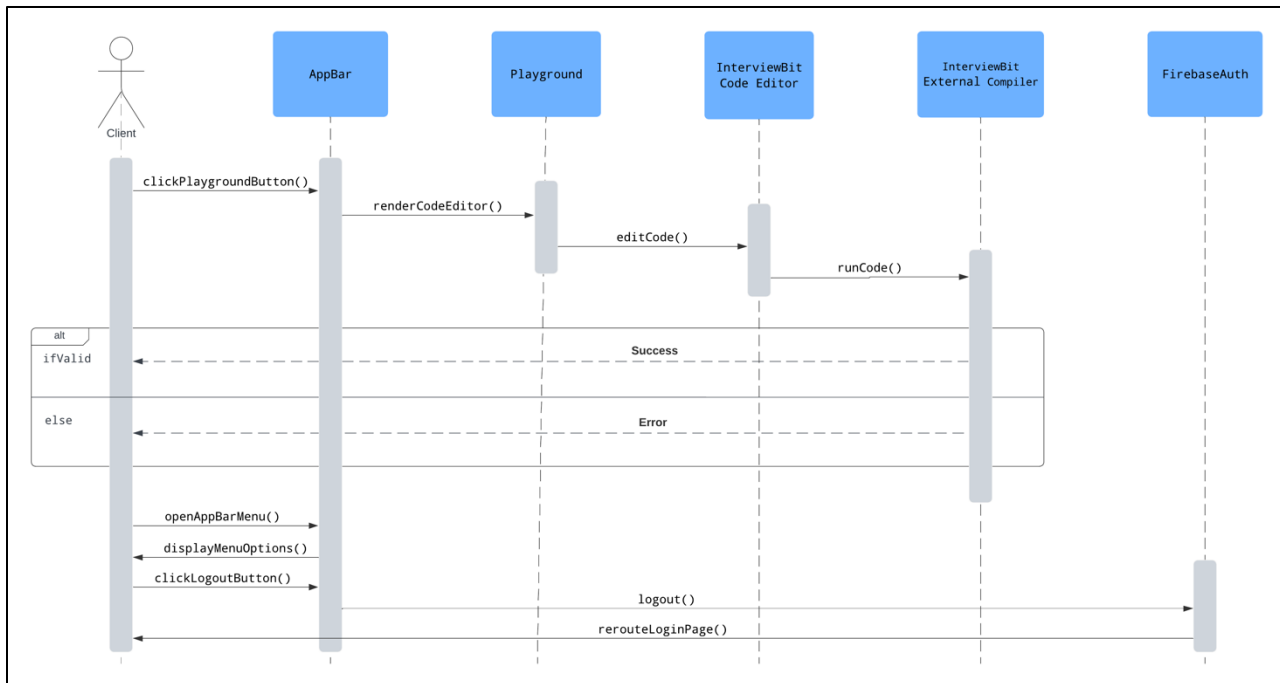
Sign-Up/In Sequence Diagram



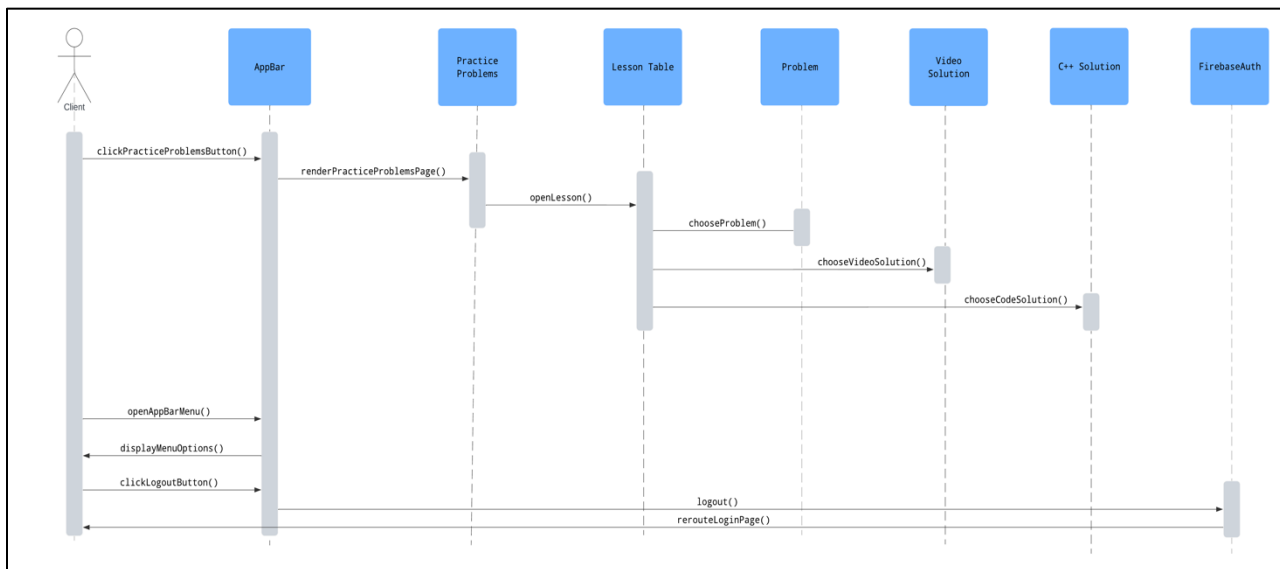
Authentication Provider Sequence Diagram



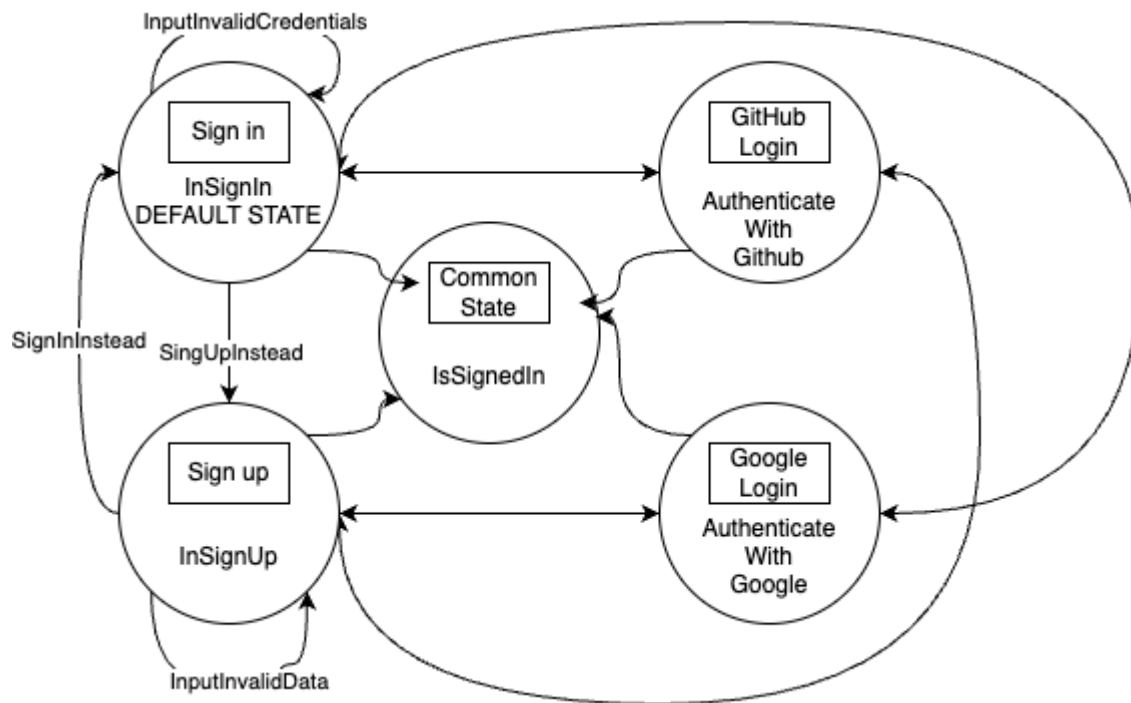
User Interaction Sequence Diagram



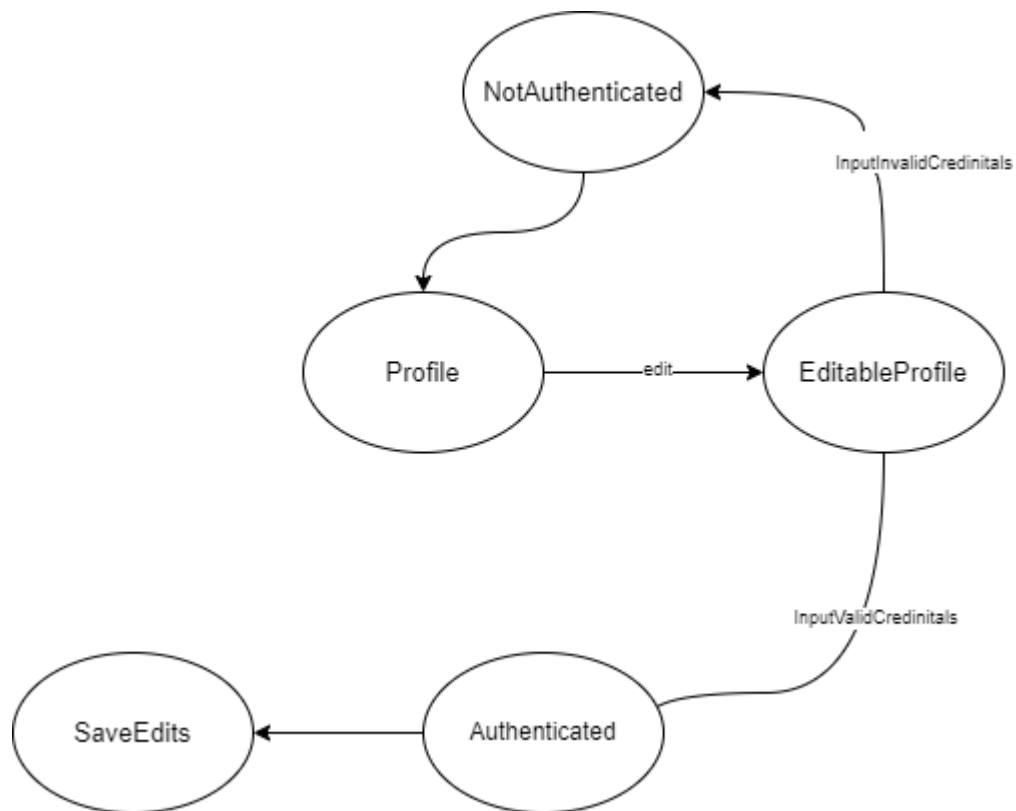
Playground Sequence Diagram



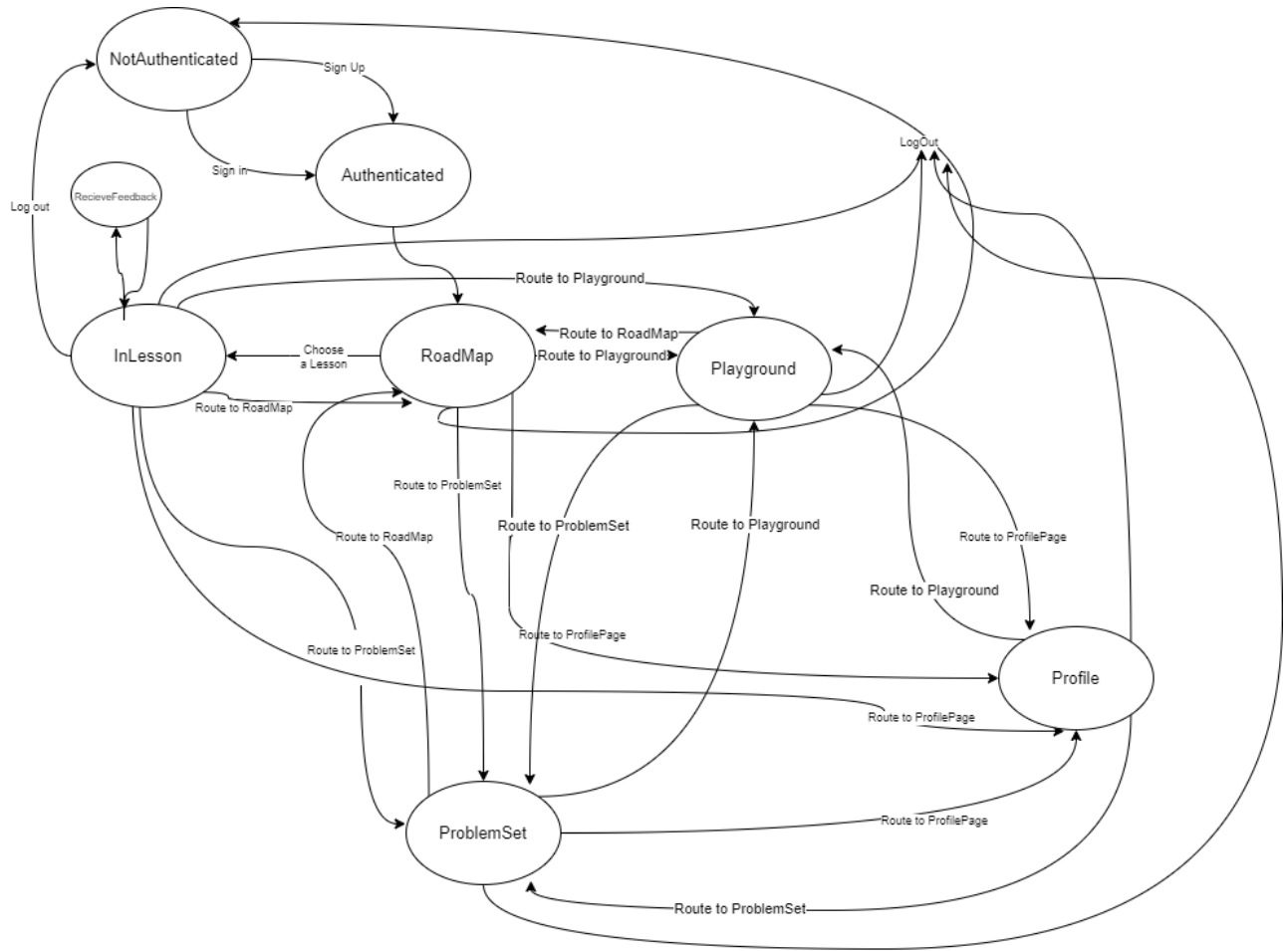
Practice Problems Sequence Diagram



Log In FSM



Profile Page FSM



Learning FSM

4.0 DATA DESIGN

4.1 Data Description

Information about the user are collected in one of two situations. First of all, when the user signs up, username, email and password are stored in firebase authentication database. In addition, firebase live database was used to store user related information that were collected after signing up. The user can access his profile and add these additional information. Firebase live database was also used to store the content of each of the existing lessons.

5.0 COMPONENT DESIGN

Components are designed to have states that changes based on the user interactions with the page.

6.0 HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

The website has a user friendly interface that starts with showing the sign in or sign up options. In addition, the user interface integrates a responsive app bar that is existing all time as long as the user is authenticated and logged in. The App bar can be used to navigate and access any of the pages in the website directly except for lessons which is accessed indirectly through the road map which is accessed directly from the App bar menu. The road map is interactive and it gives the user a better experience as it highlights the currently lesson on which the user hovers and make it interacts live with the user. In addition the user can submit code and interact with the two code and text editors provided. The UI returns feedback to the user live based on his code as well.

6.2 Screen Images

Sign In!

Email Address *

Enter your email address

Password *

SIGN IN



SIGN UP INSTEAD!

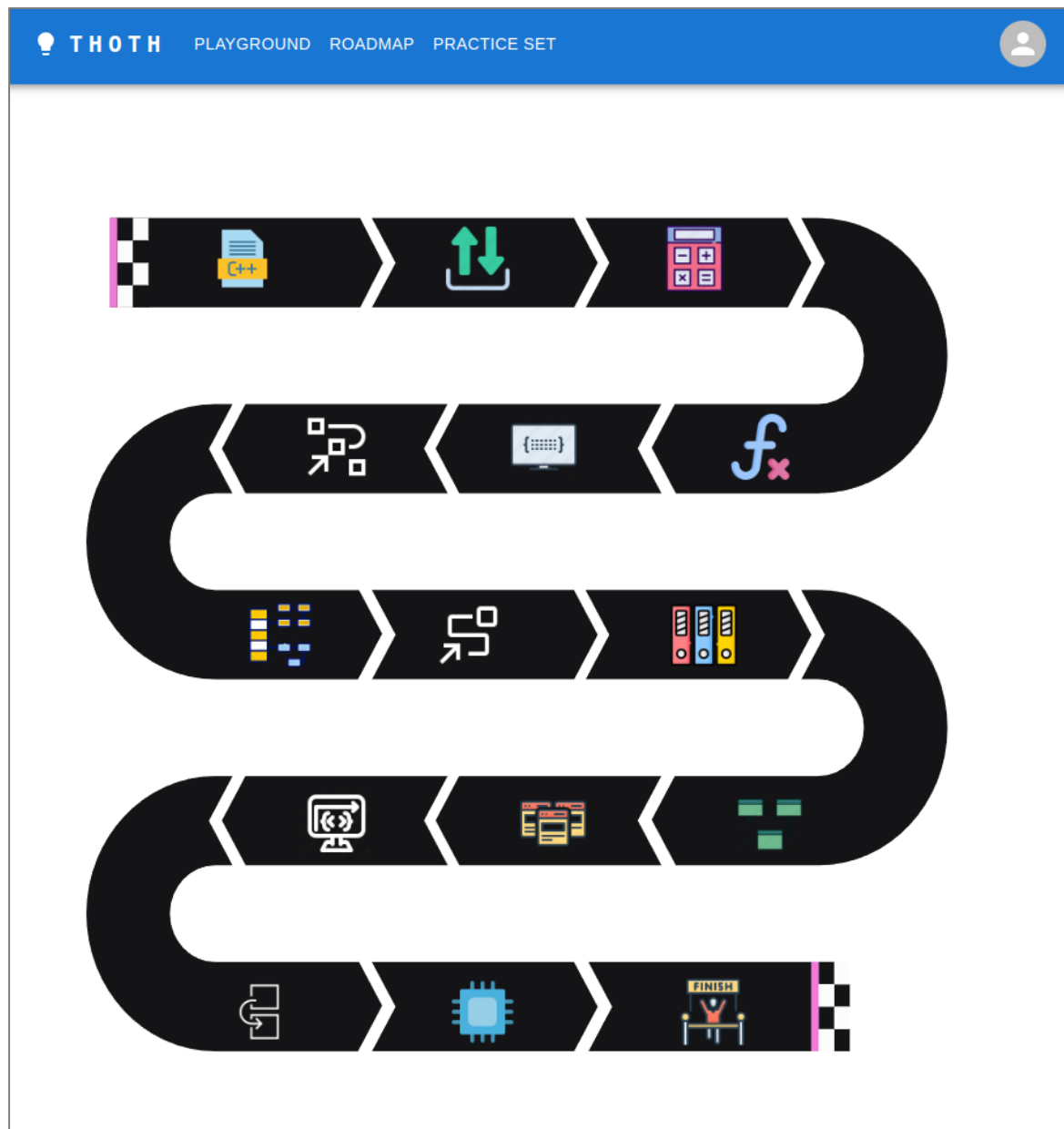
FORGOT PASSWORD?

G

Sign Up!

[SIGN IN INSTEAD!](#)





THOTH

PLAYGROUND

ROADMAP

PRACTICE SET

Z

Email Address

ziadmhassan@aucegypt.edu

Username

ziadmhassan

Password



Type Your New Password

City

No City

EDIT

SAVE


 **THOTH** [PLAYGROUND](#) [ROADMAP](#) [PRACTICE SET](#) 


Practice Problems

Lesson 1

Problem 1


Easy






Problem 2


Medium






Problem 3

Hard







Lesson 2

Problem 1


Easy






Problem 2


Medium







Problem 3

Hard








 **THOTH** [PLAYGROUND](#) [ROADMAP](#) [PRACTICE SET](#) 2

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      // Your code goes here;
6
7      return 0;
8  }
9
```

Output

Powered by  InterviewBit  [Run](#) 

THOTHPLAYGROUNDROADMAPPRACTICE SETZ

Introduction to C++: Printing "Hello, World!"

Welcome to the world of C++ programming! In this introductory lesson, we'll start with a simple yet classic example: printing "Hello, World!" to the console.

Writing Your First C++ Program

In C++, the standard way to create a simple program is by using the `main()` function. This function is the entry point of every C++ program. Let's create our first program:

```
#include <iostream>

int main() {
    // Your code goes here
    return 0;
}
```

In this example, we have a basic structure for a C++ program. The `#include <iostream>` line is necessary for input and output operations. The `main()` function is where the execution of our program begins and ends. The `return 0;` statement indicates a successful execution.

Printing "Hello, World!"

Now, let's modify our program to print the classic message to the console. We'll use the `std::cout` statement for output:

```
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

The `std::cout` statement is used to display output. In this case, we are printing the string "Hello, World!" followed by a newline (`std::endl`) to ensure a clean console output.

Copy the code, compile, and run your program. You should see:

```
Hello, World!
```

Congratulations! You've just written and executed your first C++ program. This simple example sets the stage for exploring the vast world of C++ programming.

```
1 #include <iostream>
2
3 int main() {
4     // your code goes here
5     return 0;
6 }
7
```