



School of Computing

Functional Specification

Date: 4/12/2020
Project title: MERN Coding Test Platform
Course: CASE 4
Module code: CA400
Student 1: Janesh Sharma - 17473124
Student 2: Paul McNally - 17318221
Supervisor: Dr. Stephen Blott

0. Table of contents

0. Table of contents	1
1. Introduction	3
1.1 Overview	3
1.2 Business Context	3

1.3 Glossary	3
2. General Description	4
2.1 Product / System Functions	4
2.2 User Characteristics and Objectives	5
2.3 Operational Scenarios	5
2.3.1 Employer Operational Scenarios	5
2.3.2 Candidate Operational Scenarios	5
2.4 Constraints	6
2.4.1 Time Constraints	6
2.4.2 Hardware Constraints	6
2.4.3 Memory Constraints	6
2.4.4 Storage Constraints	6
2.4.5 API Constraints	6
2.4.6 Accessibility Constraints	7
3. Functional Requirements	7
3.1 Online Compiler API	7
3.2 Code Editor	7
3.3 Register and Login	7
3.4 Database	8
3.5 Generating Link And Sending Email	8
3.6 Coding Test Setup	8
3.7 Video Responses	8
3.8 Analytics	9
3.9 User Interface	9
4. System Architecture	9
4.1 System Architecture Diagram	9
4.2 CI/CD Pipeline	10
5. High-Level Design	11
5.1 Data Flow Diagram	11
External Entities	11
Processes	12
5.2 Sequence Diagram	13
6. Preliminary Schedule	14
7. Appendices	14

1. Introduction

1.1 Overview

The main goal of the project is to build a coding test platform for companies to use as a means to assess potential new hires. We will build a live coding environment allowing candidates to complete a coding test along with the option to complete video responses to questions proposed by an employer. In addition to this employers will have access to an analytics section where candidates are ranked based on performance(i.e. Time to completion, CPU time and memory used). This application will use the MERN stack and be hosted on Firebase with a full CI/CD workflow on GitLab. Jest will be used for testing.

1.2 Business Context

This System will be useful for any software related business that needs to hire and assess software developers' applications. Our system allows businesses to effectively assess an applicant's coding skills by providing a coding test platform, video responses to employer questions and analytics to rank the best candidates.

1.3 Glossary

- **React** - Open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies [\[2\]](#).
- **Node.js** - Open-source, cross-platform, back-end, JavaScript runtime environment that executes JavaScript code outside a web browser.
- **Express.js** - Back end web application framework for Node.js [\[3\]](#).
- **MERN** - Stands for MongoDB, Express, React, Node, after the four key technologies that make up the stack.
- **CI/CD** - Refers to the combined practices of continuous integration and either continuous delivery or continuous deployment. CI/CD bridges the gaps between development and operation activities and teams by enforcing automation in building, testing and deployment of applications.
- **Cloud Function** - Serverless execution environment for building and connecting cloud services. With Cloud Functions you write simple, single-purpose functions that are attached to events emitted from your cloud infrastructure and services. Your function is triggered when an event being watched is fired.
- **WCAG 2.1 Standards** - Web Content Accessibility Guidelines (WCAG) 2.1 defines how to make Web content more accessible to people with disabilities. Accessibility involves a wide range of disabilities, including visual, auditory, physical, speech, cognitive, language, learning, and neurological disabilities [\[1\]](#).
- **Firebase** - Platform developed by Google for creating mobile and web applications [\[5\]](#).

- **MongoDB** - Cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas [6].
- **MongoDB Cloud Atlas** - Global cloud database service for modern applications. Deploy fully managed MongoDB across AWS, Google Cloud, and Azure with best-in-class automation and proven practices that guarantee availability, scalability, and compliance with the most demanding data security and privacy standards [5].
- **Judge0** - Robust, scalable, and open-source online code execution system that can be used to build a wide range of applications that need online code execution features [7].
- **Jest** - Jest is a JavaScript testing framework maintained by Facebook, Inc. designed and built by Christoph Nakazawa with a focus on simplicity and support for large web applications [8].

2. General Description

2.1 Product / System Functions

The functions of this system are to have two types of users, employers and job applicants/candidates. Only the employers will have a login to access the system and the job candidates will access the system and their coding test by using a link sent via email.

The employers will use this web application to access applicants for jobs. Employers will be able to use the application for several functions:

- Set coding challenges for the applicants, the employer will provide a description of the challenge as well as the test cases for each challenge to test the applicants code when the applicants run their code.
- The employer will also have the option to set interview questions for each applicant to record a video response too. i.e. Explain their solution to one of the coding challenges.
- After the applicant(s) have completed the application process, the employer will be able to view the analytics from the applicant(s) i.e. Time to completion, CPU time and memory used.

Applicants will use this web application to complete the coding challenges and provide video responses to interview questions set by the employing company.

- Applicants will access the system and their coding test by using a link sent via email
- When completing the coding tests, the applicants will be able to choose from a selection, which programming language they would like to use for the coding challenges.
- Applicants will be able to run their code for each challenge against the test cases before submitting their final code.

2.2 User Characteristics and Objectives

Our intended users for this project are expected to have advanced computer knowledge. The job applicants are expected to have strong computer programming experience as they are required to complete coding challenges and have the competence to use an IDE.

For the employer, the recruiting staff for a company is not expected to have computer programming skills but should be able to understand how to set up the coding challenges under clear guidance and instruction. They should also be able to set the interview questions and be able to interpret what the analytics after the coding test shown to them mean.

2.3 Operational Scenarios

2.3.1 Employer Operational Scenarios

- **Employer Registration** - An employer can register using their google account or create an account using their email and created password, if they choose to register with their google account, it is authenticated through firebase. After a successful registration, the user details are stored in MongoDB
- **Employer Login and Logout** - The user will login using their credentials from when they registered or login using their google account etc, which is authenticated through firebase. The user can logout by clicking on the logout button or if they close the browser tab, they will automatically be logged out.
- **Employers set coding challenges** - They must provide a description of exactly what the program should do(i.e. fibonacci sequence), what the input will be and what the output of each program will be.
- **Employer sets test cases** - When an applicant taking the coding test wants to run their code to ensure it's correct, it'll be run using test cases. As the employers are the ones setting the coding challenges, they must provide at least one test case for each coding challenge.
- **Employer sets interview questions** - The employer will set one or more interview questions that the job candidate will be required to record a video response too.
- **Employer selects applicants** - Employers must select which of the applicants have been selected to continue the application process and participate in a coding test. The successful candidates will be notified via an email containing a link to the coding test.
- **Employer views analytics** - When the candidates have completed their coding test the employer will be able to view analytics to rank the best candidates. The analytics will display information such as time to completion, CPU time and memory used etc.

2.3.2 Candidate Operational Scenarios

- **Applicants access the application via link** - If applicants are successful in being selected to participate in the coding test, they will receive an email containing a link to the coding test.

- **Applicants develop within IDE** - Candidates will be able to write code in an online IDE environment.
- **Applicants select programming language** - Candidates will be able to choose from a list which programming language they would like to complete the coding test in.
- **Applicants run their code** - Candidates can run their code whenever they want. When the code is run, it'll use the test cases provided by the employer.
- **Applicants submit code** - When applicants have completed each coding challenge to their satisfaction they hit submit, then their submission is processed and their code and analytics is stored in the database(MongoDB).
- **Applicants answer interview questions** - Applicants record their responses to the set interview questions. They must grant access to their camera and microphone.
- **Finish** - When all parts of the coding test is complete the code, analytics and video responses are stored in the database(MongoDB).
- **Applicant disconnects** - If an applicant disconnects or quits before completing the coding test, their progress up until that point is recorded and when they resume by re-clicking their link to the test, they can continue from the last point that data was stored.

2.4 Constraints

2.4.1 Time Constraints

When the user clicks the “run code” button the code must compile and return the output within a reasonable amount of time.

2.4.2 Hardware Constraints

The system needs to be hosted on capable hardware sufficient enough to handle multiple user load.

2.4.3 Memory Constraints

The application will be hosted on Firebase which provides us with a maximum constraint of 2048MB of RAM.

2.4.4 Storage Constraints

- In regards to the video response to questions functionality, the videos will need to be stored to be viewed later by an employer, thus we will need to source additional storage from Firebase which provides 5GB of free storage and 1GB of download per day.
- Our Database will be hosted on MongoDB Cloud Atlas which provides a maximum 512MB of free storage space.

2.4.5 API Constraints

- Firebase Cloud function - 125k invocations and 40k cpu seconds a month, which is 11.11 hours of application runtime per month.

- The online Compiler called Judge0 hosted by RapidAPI provides a monthly quota of 10k submissions per month and 100 requests per minute.

2.4.6 Accessibility Constraints

- The user interface needs to be compliant with the minimum guidelines set out in the WCAG 2.1 web standards [\[1\]](#).

3. Functional Requirements

3.1 Online Compiler API

- **Description** - The code needs to be compiled/interpreted upon submission, for this functionality we will use the Judge0 API which is an online compiler.
- **Criticality** - Code compilation is the most critical component of the system as it's output is used to evaluate the code against test cases for candidate evaluation and ranking.
- **Technical issues** - Handling errors appropriately such as code that takes more than e.g. 10 seconds to run as it will be halted by the API.
- **Dependencies with other requirements** - This is dependent on the Online Code editor as all code must be written either online or offline before submission.

3.2 Code Editor

- **Description** - The System must allow a user to write code online as they would offline in an IDE with syntax highlighting.
- **Criticality** - This is critical as a user must be able to write and submit code online using our system.
- **Technical issues** - Handling input and output to provide a correctly formatted object to the Code Compiler API. Additionally, saving user code continuously to avoid data loss upon potential disconnection from the application.
- **Dependencies with other requirements** - This component is heavily dependent on the Code Compiler as all code needs to be compiled to evaluate the user and the Coding Test component as the coding test needs to be correctly set for a candidate to solve the coding test.

3.3 Register and Login

- **Description** - The user will have to register first and will then be securely logged in using authentication. Only the employer is required to login as they are required to set up their custom coding test.
- **Criticality** - This is critical because an employer needs to be able to register/login in order to set up the coding test, sending emails to applicants and to view analytics.
- **Technical issues** - Setting up authentication correctly such that users will be able to login/register with their email, google account etc. Each user's details need to be

stored correctly in MongoDB so they can set up a coding test and have the correct applicants do the coding test.

- **Dependencies with other requirements** - This component is dependent on the Database component as the user details need to be stored/retrieved upon registration/login.

3.4 Database

- **Description** - Saving employer details, candidate code and analytics.
- **Criticality** - This component is critical for the system to have a working system without data loss.
- **Technical issues** - Setting up a database and designing schemas for NoSQL Data models.
- **Dependencies with other requirements** - The database is dependent on receiving/requesting data from all components in the system.

3.5 Generating Link And Sending Email

- **Description** - The application should be able to generate a custom link for each selected applicant and send it to them via email.
- **Criticality** - This is critical because if the link does not work and/or is not sent to the applicant, they cannot attempt the coding test.
- **Technical issues** - Making sure that the application generates unique links that can be used to access the coding test more than once.
- **Dependencies with other requirements** - This component is not dependent on any component within the system, but is required to provide access for the applicant users of the system.

3.6 Coding Test Setup

- **Description** - The employer must be able to set up a coding question along with tests cases and a time limit to allow the system to evaluate candidates.
- **Criticality** - This component is critical as it allows the system to evaluate a candidate and apply ranking.
- **Technical issues** - Provide a friendly and simple user interface so the coding test can be set up with ease by a non technical person.
- **Dependencies with other requirements** - This component is not dependent on any component within the system, but is required to provide a coding test to candidates.

3.7 Video Responses

- **Description** - Candidates should be able to record a video response to questions proposed by the employer.

- **Criticality** - This component is not critical but aids in the employers evaluation of potential candidates by allowing them to view a candidates' thinking, communication and problem-solving abilities.
- **Technical issues** - The video recording must be saved to a storage bucket on Firebase so it can be reviewed by an employer in the future.
- **Dependencies with other requirements** - This component is not dependent on any component within the system, but aids in the employers understanding of the type of individual the candidate is.

3.8 Analytics

- **Description** - The system must be able to assess an applicants' code based on predefined test cases set by the employer and metrics such as cpu time and memory used so as to allow the ranking of candidates.
- **Criticality** - This critical for the user to be able to see the analytics of the coding tests to see which were the best performing applicants.
- **Technical issues** - It must make sure that it is displaying the correct data for each applicant user. Ensuring metrics such as cpu time and memory used are accurate and replicable for the same piece of code.
- **Dependencies with other requirements** - This is dependent on the database as it needs to pull the analytics data from the database.

3.9 User Interface

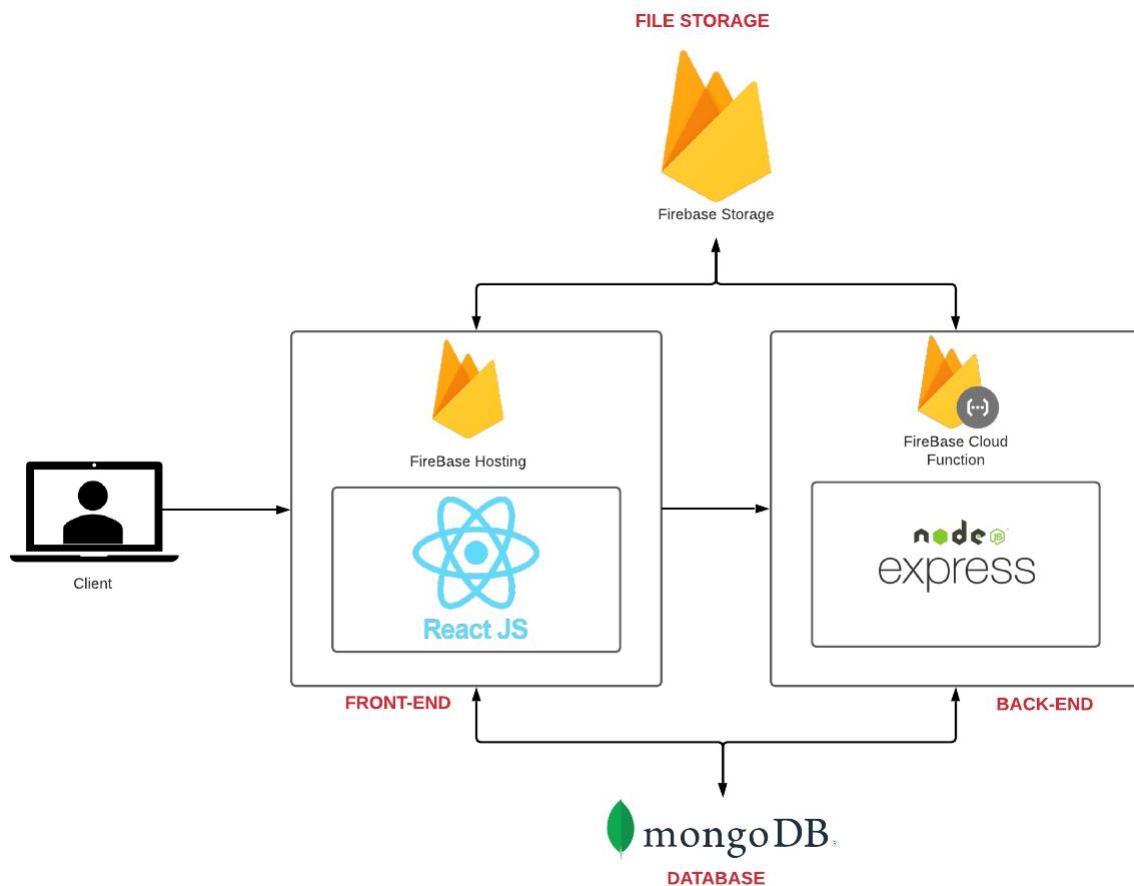
- **Description** - The UI should be designed with simplicity and accessibility in mind so as to allow non technical users to operate the employer section and allow various individuals with disabilities to operate the site easily.
- **Criticality** - This is critical as we need a simplistic interface to allow users to interact with the system effectively.
- **Technical issues** - Designing a website that at least meets the basic guidelines set out in the WCAG 2.1 web standards [\[1\]](#).
- **Dependencies with other requirements** - This component is not dependent on any component within the system, but is vital to delivering a usable application to the end user.

4. System Architecture

4.1 System Architecture Diagram

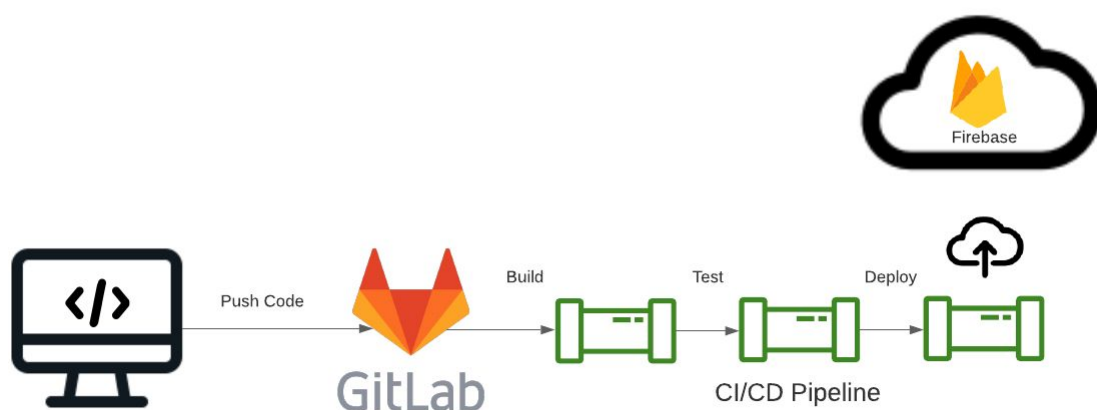
From looking at the diagram below you can see the overall architectural design of the system. We will be using a MERN stack architecture with Firebase for hosting and storage, where we will have the React static frontend application hosted on Firebase hosting and the

backend dynamic Node.js application hosted on a Firebase cloud function in a sort of microservices design pattern. Additionally the database will be a NoSQL database provided by MongoDB Cloud Atlas.



4.2 CI/CD Pipeline

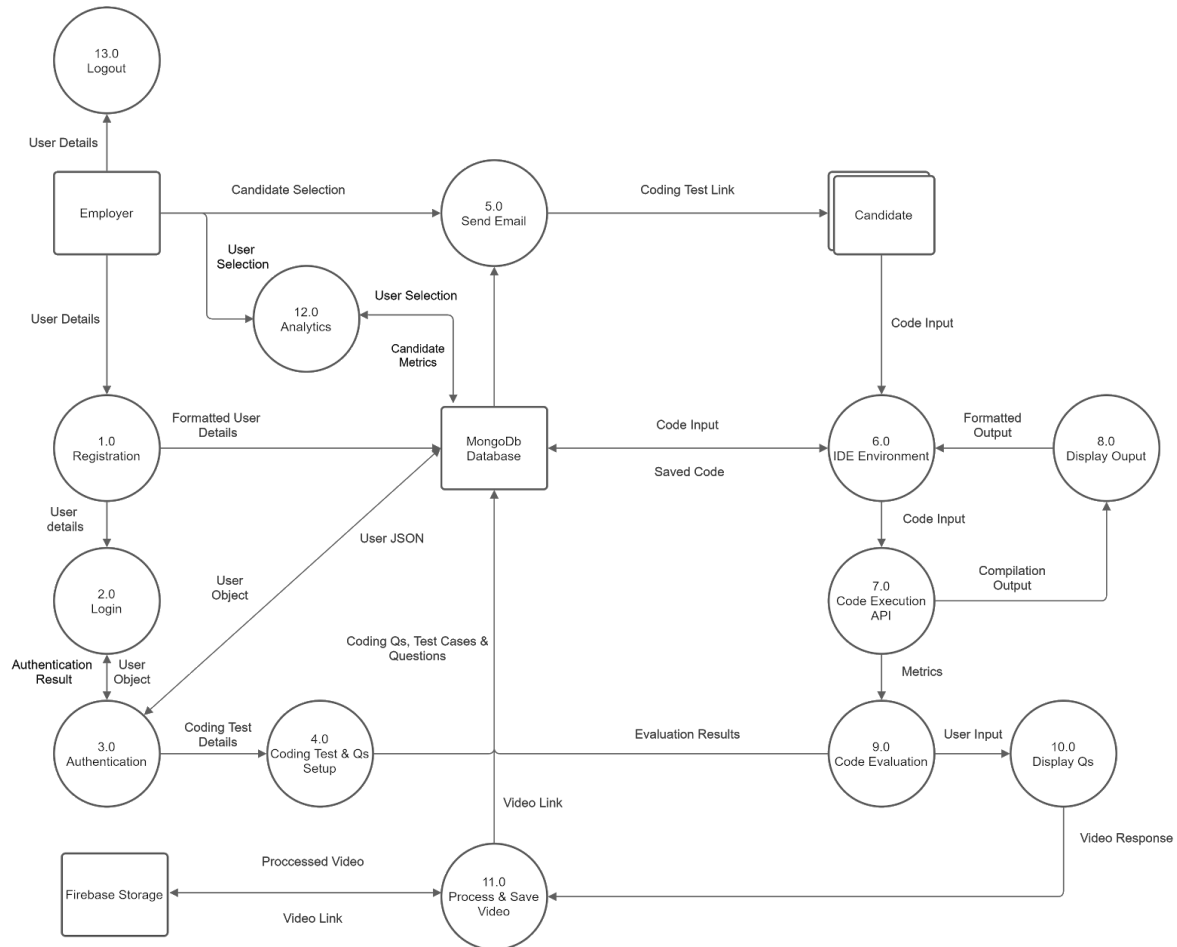
From looking at the diagram below you can see we have a full CI/CD pipeline, with build, test stages and finally deployment to production in Firebase.



5. High-Level Design

5.1 Data Flow Diagram

A data flow diagram or DFD represents the flow of data of a process or system. The DFD provides information about the outputs and inputs of each entity and the process itself.



External Entities

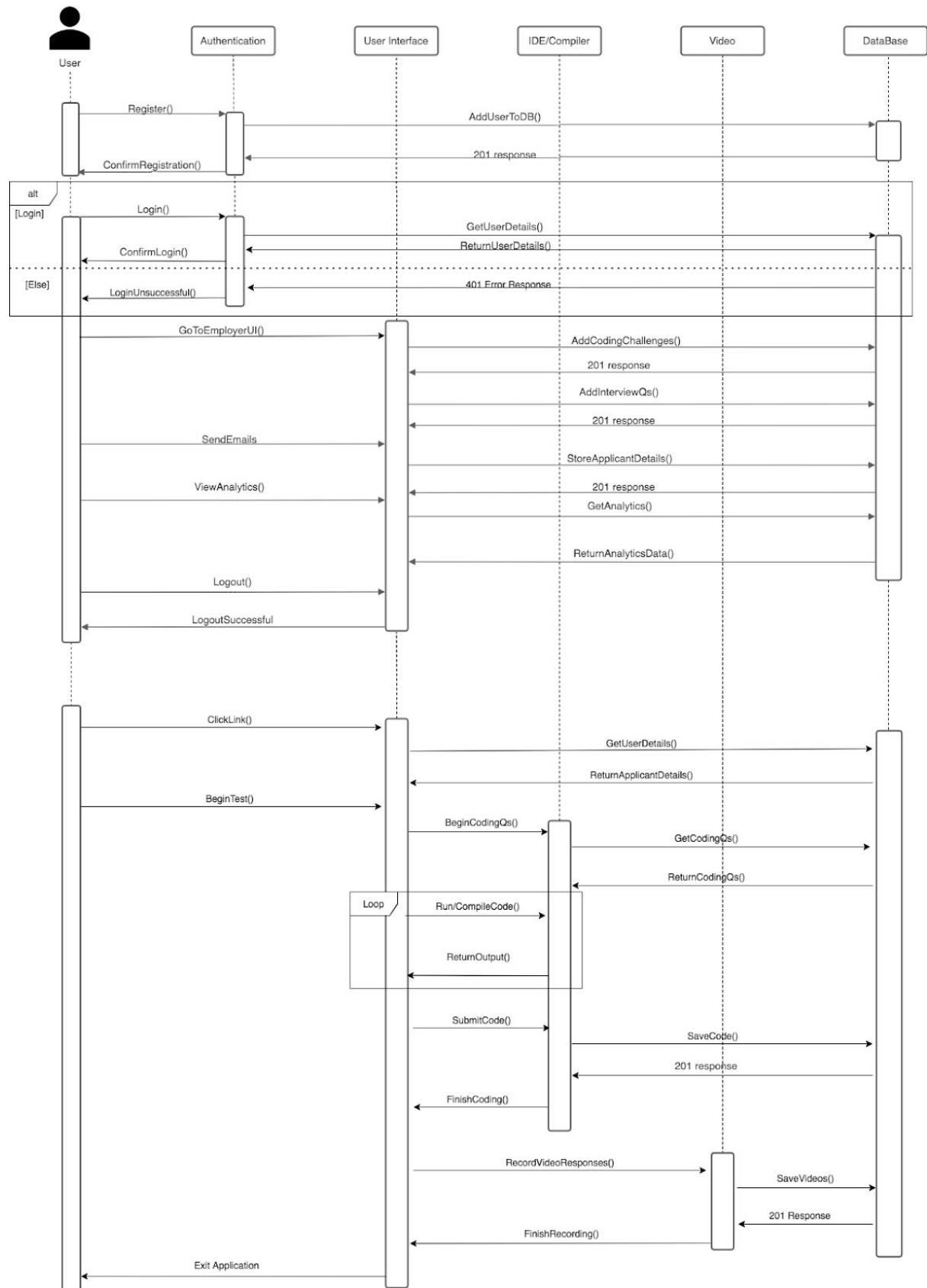
From looking at the diagram above you can see there are four external entities, namely:

1. **Employer** - This represents a person from the company reviewing candidates.
2. **Candidate** - This represents a person chosen to perform a coding test for the Employer.
3. **MongoDB Database** - This represents the MongoDB database which will be used to store various details such as user data, analytics etc.
4. **Firebase Storage** - This represents the Firebase file storage bucket within which candidate video responses will be stored for future viewing by the Employer.

Processes

1. **Registration** - The user enters details which are then stored in the database.
2. **Login** - The user enters user details and sends the data for authentication.
3. **Authentication** - The user details are authenticated against the details stored in the database and the user is authenticated depending on the result.
4. **Coding Test & Qs Setup** - The user inputs the coding question in text format along with test cases and any questions for the candidate.
5. **Send Email** - The user selects chosen candidates and provides them with a link to the coding assessment in an email.
6. **IDE Environment** - The candidate is given a link which brings them to the coding test IDE with syntax highlighting.
7. **Code Execution Environment** - This process compiles code and returns the output, and metrics such as cpu time and memory used.
8. **Display Output** - The code compilation output is displayed for the candidate to show the output of the code they executed.
9. **Code Evaluation** - The code is evaluated against all the test cases and the results are stored in the MongoDB database.
10. **Display Qs** - The questions set by the employer are displayed to the candidate who must provide a video response to each question.
11. **Process & Save Video** - The video is pre-processed to reduce size and required storage space and sent to Firebase Storage to be saved, then a link is stored in MongoDB to reference it at a later time.
12. **Analytics** - The candidates for a particular coding test will be displayed and ranked according to performance metrics such as cpu time and memory used which tells us the overall efficiency of a solution.
13. **Logout** - The employer will be logged out of their account.

5.2 Sequence Diagram



The sequence diagram shown above shows the various operations that can be carried out on our application. It captures the interaction between objects in the context of collaboration.


- When a user (an employer) registers, their login is authenticated through firebase and their details are stored in the database(MongoDB).
- When an employer logs in, they are again authenticated through firebase and their user details are retrieved from MongoDB(if they are not registered or have entered incorrect details, the login will fail), they then proceed to set up the coding test by adding the coding challenges and interview questions and adding them to the database. Users will send emails containing links to the coding test to the applicants and the applicants data (i.e. email) is stored in MongoDB. They also can view the analytics of the coding tests results, with the data again being retrieved from MongoDB.
- The user (an applicant) accesses the application by clicking their unique link, they then begin the coding test. First they attempt the coding challenges, the user can run and compile their code as much as they like. After completing the coding challenges there code and analytics are stored in MongoDB. Finally they record themselves answering the interview questions, the videos are then stored in firebase cloud storage.

6. Preliminary Schedule

	Start Date	End Date	Timeline	Status
Final Year Project	9-11-2020	28-05-2021		Active
Get project approved by approval panel	9-11-2020	09-11-2020		Complete
Functional Specification	10-11-2020	05-12-2020		Active
Setup Database, Firebase and CI/CD	7-12-2020	20-12-2020		Upcoming
MongoDB NoSQL Data Modelling	14-12-2020	20-12-2020		Upcoming
Setup Jest Testing (unit, integration, e2e)	14-12-2020	20-12-2020		Upcoming
Build Register/Login/Logout Authentication	11-1-2021	17-01-2021		Upcoming
Setup Simple Code Execution with IDE	11-1-2021	31-1-2021		Upcoming
Develop Employer Coding Test Setup UI	18-1-2021	28-2-2021		Upcoming
Email Candidates Link To Coding Test	1-3-2021	7-3-2021		Upcoming
Fully Integrate Code Execution system with IDE and sync code incase of disconnect	8-3-2021	4-4-2021		Upcoming
Analytics - Evaluate Candidate Solutions	5-4-2021	11-4-2021		Upcoming
Candidate Video Responses - Save to Firebase	12-4-2021	18-4-2021		Upcoming
Refactoring and Finish Code	19-4-2021	7-5-2021		Upcoming
Final Documentation and video walkthrough	19-4-2021	7-5-2021		Upcoming
Submission	7-5-2021	07-05-2021		Upcoming
Project Expo	17-5-2021	28-5-2021		Upcoming
Project Demonstration	17-5-2021	28-5-2021		Upcoming
			Overall Progress	

7. Appendices

1. (WAI), W3C Web Accessibility Initiative. "WCAG 2.1 at a Glance." Web Accessibility Initiative (WAI), www.w3.org/WAI/standards-guidelines/wcag/glance/. Accessed 25th Nov. 2020.
2. "React – A JavaScript Library for Building User Interfaces." React, reactjs.org. Accessed 4 Dec. 2020.
3. "Express - Node.Js Web Application Framework." Express, expressjs.com. Accessed 4 Dec. 2020.
4. MongoDB. "The Most Popular Database for Modern Apps." MongoDB, www.mongodb.com. Accessed 4 Dec. 2020.
5. Firebase, Google, firebase.google.com/. Accessed 4 Dec. 2020.

6. "Managed MongoDB Hosting | Database-as-a-Service." MongoDB, www.mongodb.com/cloud/atlas. Accessed 4 Dec. 2020.
7. "Judge0." GitHub, github.com/judge0/judge0. Accessed 4 Dec. 2020.
8. "Jest ·  Delightful JavaScript Testing." Jest, jestjs.io. Accessed 4 Dec. 2020.