Interactive Transcript Application

By Victor Boateng

A person sitting at a desk with a computer

Description automatically generated

TABLE OF CONTENTS

1. Introduction
2. Background
3. Software Design
   1. Use Case Diagram
   2. Software Requirements (FR & NFR)
      1. Use Case 1
      2. Use Case 2
   3. Software Architecture Design
      1. Entity-Relationship Diagram
      2. Class Diagram
      3. Sequence Diagram
   4. Features Mapping
4. Implementation

# **Software Requirements**

The project has two use cases of which one is the initial use case, and the second being an extended version of the application. Only, once the initial use case is developed is the second use case considered.

## **3.2.1. Use Case 1**

Use case 1, also known as the initial phase of the project is designed simply for students.

**Functional Requirements:**

1. The system shall provide users with an option to **register** for an account if not already registered.
2. The system shall only completely register a user once his/her email address is verified.
3. The system shall store **users’ registration details** in a **database**.
4. When login into the system, the system shall compare details to user details store in the database.
   1. the system shall redirect to the home page, if details are entered correctly.
   2. the system shall display an error message informing the user details entered are incorrect, If details entered are not a match with the details stored in the database,
5. The system shall enable users to enter their university **module** and **contents** of each module.
6. The system shall display all entered modules/topics to corresponding user.
7. They system shall calculate the overall grade from all the entered module **grades** and content and then display the current **classification** & **percentage**.
8. They system shall also display the necessary average required for the user to attain the following: first class (1:1), upper-second class (2:1), lower-second class (2:2) & third class (3:3).
9. The system shall enable students to reveal their grades to employers anonymously.

**Non-Functional Requirements**

## **3.2.2. Use Case 2**

The second use case is the extended use case which extends the system from useful to only students but also to recruiters/employers. Inspired by LinkedIn, the application will be extended to support recruiters in the opportunity to headhunt early talent based off parameters that might be asset skills to their employers. It also provides students the opportunity to be headhunted based off their grades, effectively encouraging them to do well in sought after modules.

**Functional Requirements:**

1. The system shall enable recruiters to register for an account; of which there are two. A plus account & a basic account.
2. The system shall only allow recruiters/employers to login once their registration is verified via their email address.
3. The system shall enable employers/recruiters to view non-anonymous & anonymous registered students’ grades.
   1. The system shall enable employers/recruiters to filter students based of a range of parameters.
4. For plus members only, the system shall enable employers/recruiters to message students via the app.

**Non-Functional Requirements:**

# **Software Architecture Design**

A diagram of a service layer

Description automatically generated

Fig.3, displays the software architecture of Transcript Application with numbers attached to each of the components.

Let’s now explain each of the following components from Fig.3. The project is split into three layers: Interface, Service and Data layers where the interface layer is designed in Reacts, the service layer is designed in java spring boot & the data layer that stores all the tables required for the project (mentioned in chapter 3.3.1 ER diagram).

First the Interface Layer:

1. The login component will make a post request, passing the **user login details** entered to the rest Controller. The rest controller will then pass those details to the service layer for it to compare the details to the detail stored in the database via the interface. Once that information is processed the rest controller will return confirmation of whether the user can login with the entered details or not.
2. The registration component will pass **registration details** to the rest controller and the rest controller will pass those details to the service layer to log those details into the correct database.
3. The Transcript component will pass details such as module & content to the rest controller and that will then store those details into their respective tables. On top of that it will make a call to the service layer to calculate overall grade.

Second the Service Layer:

1. Rest Controller: This layer exposes endpoints for the front end to communicate with the back end.
2. Authentication and Authorization Module: This module manages user authentication and authorization. It ensures that only verified users can access the system.
3. User Registration: This module handles user registration, verification, and stores registration details.
4. Transcript Management: This module manages user transcripts, module, and content information.
5. User Grades Calculation: This module calculates the overall grade and provides classification & percentage.
6. Employer Interaction: This module allows students to reveal their grades to employers anonymously.
7. User Profile: This module manages user profiles, including personal information, module selections, and preferences.

Finally, the data Layer:

11,12,13 & 14 - This just contains all the tables and is already explained in Chapter 3.3.1 ER diagram.