|  |
| --- |
| **College of Electrical & Mechanical Engineering**  **Department of Software Engineering**    **Undergraduate project**  **Title: Talent Management system**      **Feb 06/20/2022** |

# Approval

**Group members:**

**No. Name ID Email**

1. **KIBROM ESAYAS**  **ETS 0607/11 kibromesayas4@gmail.com**
2. **KIRUBEL BERHANU ETS0624/11 kirubelberhanu2022@gmail.com**
3. **MARSHAL YORDANOS ETS 0678/11 marshalyordanosseifu@gmail.com**
4. **MOHAMMED HUSSEIN ETS 0771/11 hmuhammed18@gmail.com**

**Advisor Name: Mr. Tamiru Hailesilassie**

**Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date:** \_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_

# Acknowledgements

First of all, we would like to thank our almighty God who gives us love, patience, healthy, wisdom and ability to finish this semester project.

Our heartfelt appreciation goes to our adviser Mr. Tamiru Hailesilassie (MSc.) for his guidance, valuable suggestion, moral support, and constant encouragement in the documentation part and also for constructive opinion and willingness to participate in our documentation and his effective direction.

We would like to thank many people for their kindness and assistance during our project work. We truly appreciate the support of the people at the Afro art school and legacy modeling and acting school for their willingness of patience in answering our numerous questions.

Finally, we would like to express our love, thanks, appreciation, and respect for the ongoing support of all our members, friends and classmates who have one way or another contributed to the success of our project. And also, we would like to thank the teaching staffs who have contributed wholly to the success of this documentation.

**Table of Contents**

[Approval i](#_Toc106614200)

[Acknowledgements ii](#_Toc106614201)

[Lists of Tables vi](#_Toc106614202)

[List of Figures/Illustrations, Plates/Photographs vii](#_Toc106614203)

[List of Abbreviations, Symbols, Specialized Nomenclature viii](#_Toc106614204)

[Abstract x](#_Toc106614205)

[ረቂቅ x](#_Toc106614206)

[Chapter 1: Introduction 1](#_Toc106614207)

[1.1 Background of the organization 1](#_Toc106614208)

[1.2 Statement of the problem 1](#_Toc106614209)

[1.3 Existing system 2](#_Toc106614210)

[1.3.1 Major problem of existing system 2](#_Toc106614211)

[1.4 Proposed system 3](#_Toc106614212)

[1.4.1 Advantage of proposed system 3](#_Toc106614213)

[1.5 Motivation 4](#_Toc106614214)

[1.6 Scope and limitation of the project 4](#_Toc106614215)

[1.6.1 Scope of the project 4](#_Toc106614216)

[1.6.2 Limitation of the project 5](#_Toc106614217)

[1.7 Objective 5](#_Toc106614218)

[1.7.1 General objective 5](#_Toc106614219)

[1.7.2 Specific objective 5](#_Toc106614220)

[1.8 Methodology 6](#_Toc106614221)

[1.8.1 Data collection methodology 6](#_Toc106614222)

[1.8.2 System design and analysis tools 6](#_Toc106614223)

[1.8.3 System development tools 6](#_Toc106614224)

[Chapter Two: System Requirement Specification 7](#_Toc106614225)

[2.1 Background Overview 7](#_Toc106614226)

[2.2 Functional Requirement 8](#_Toc106614227)

[2.3 Non-Functional Requirement 9](#_Toc106614228)

[2.4 Feasibility study 10](#_Toc106614229)

[2.4.1 Technical feasibility 10](#_Toc106614230)

[2.4.2 Operational Feasibility 11](#_Toc106614231)

[2.4.3 Political Feasibility 11](#_Toc106614232)

[2.4.4 Schedule feasibility 11](#_Toc106614233)

[2.4.5 Behavioral feasibility 12](#_Toc106614234)

[2.4.6 Economic Feasibility 12](#_Toc106614235)

[Chapter Three: System Analysis and Modeling 13](#_Toc106614236)

[3.1 Overview 13](#_Toc106614237)

[3.2 Scenario Based Modeling 13](#_Toc106614238)

[3.2.1 Use case identification 13](#_Toc106614239)

[3.2.2 Actor identification 14](#_Toc106614240)

[3.2.3 Use case description 16](#_Toc106614241)

[3.2.4 Use case diagram 44](#_Toc106614242)

[3.2.5 Activity diagram 45](#_Toc106614243)

[3.3 Behavioral/Dynamic Modeling 54](#_Toc106614244)

[3.3.1 Sequence diagram 54](#_Toc106614245)

[3.3.2 State Diagram 64](#_Toc106614246)

[3.4 Class-Based Modeling 68](#_Toc106614247)

[3.4.1 Identifying classes 68](#_Toc106614248)

[3.4.2 Class Diagram 78](#_Toc106614249)

[Chapter Four: System Design 79](#_Toc106614250)

[4.1 Overview 79](#_Toc106614251)

[4.2 System Design 79](#_Toc106614252)

[4.2.1 Design Goal 79](#_Toc106614253)

[4.2.2 System Decomposition 80](#_Toc106614254)

[4.2.3 Module Description 81](#_Toc106614255)

[4.2.4 Purpose of the system design 86](#_Toc106614256)

[4.3 Architecture of the system 86](#_Toc106614257)

[4.3.1 Architectural style 86](#_Toc106614258)

[4.3.2 Architectural Pattern 87](#_Toc106614259)

[4.3.3 Component Diagram 88](#_Toc106614260)

[4.3.4 Deployment diagram 90](#_Toc106614261)

[4.5 Database design 91](#_Toc106614262)

[4.6 Hardware / Software Mapping 92](#_Toc106614263)

[4.7 Access control 92](#_Toc106614264)

[4.8 Single view system 93](#_Toc106614265)

[4.9 User Interface Design 93](#_Toc106614266)

[5. Chapter Five: Implementation and Testing 99](#_Toc106614267)

[5.1 Implementation 99](#_Toc106614268)

[5.1.1 Software tools 99](#_Toc106614269)

[5.1.2 Development languages 99](#_Toc106614270)

[5.1.3 Source Code 100](#_Toc106614271)

[5.2 Testing 120](#_Toc106614272)

[5.2.1 Unit test 120](#_Toc106614273)

[5.2.2 Integration Testing 121](#_Toc106614274)

[5.2.3 System testing 121](#_Toc106614275)

[6. Chapter Six: Conclusion and future work 126](#_Toc106614276)

[Appendix 127](#_Toc106614277)

[Questionnaire 127](#_Toc106614278)

[References 130](#_Toc106614279)

# Lists of Tables

[Table 1 use case to actor relation 14](#_Toc106614280)

[Table 2 use case login 16](#_Toc106614281)

[Table 3 Create new account 18](#_Toc106614282)

[Table 4 Post a blog 19](#_Toc106614283)

[Table 5 Give feedback 20](#_Toc106614284)

[Table 6 view jobs 20](#_Toc106614285)

[Table 7 view post 21](#_Toc106614286)

[Table 8 view event 22](#_Toc106614287)

[Table 9 view reward 23](#_Toc106614288)

[Table 10 View blog 23](#_Toc106614289)

[Table 11 Edit personal profile 24](#_Toc106614290)

[Table 12 Search for jobs 25](#_Toc106614291)

[Table 13 Search for posts 25](#_Toc106614292)

[Table 14 Post media 26](#_Toc106614293)

[Table 15 View profile 27](#_Toc106614294)

[Table 16 Generate report 28](#_Toc106614295)

[Table 17 Give approval 28](#_Toc106614296)

[Table 18 Respond to feedback 29](#_Toc106614297)

[Table 19 Post comment 30](#_Toc106614298)

[Table 20 Post job 30](#_Toc106614299)

[Table 21 Apply job 31](#_Toc106614300)

[Table 22 Prepare event 32](#_Toc106614301)

[Table 23 Edit blog 33](#_Toc106614302)

[Table 24 Logout 34](#_Toc106614303)

[Table 25 Delete blog 34](#_Toc106614304)

[Table 26 Delete post 35](#_Toc106614305)

[Table 27 Delete job 36](#_Toc106614306)

[Table 28 Delete reward 36](#_Toc106614307)

[Table 29 Send Message 37](#_Toc106614308)

[Table 30 Search Users 38](#_Toc106614309)

[Table 31 Delete user 38](#_Toc106614310)

[Table 32 Report post 39](#_Toc106614311)

[Table 33 Like a post 40](#_Toc106614312)

[Table 34 Dislike a post 40](#_Toc106614313)

[Table 35 Follow user 41](#_Toc106614314)

[Table 36 Unfollow user 42](#_Toc106614315)

[Table 37 Prepare reward 42](#_Toc106614316)

[Table 38 Claim reward 43](#_Toc106614317)

[*Table 39 User class* 68](#_Toc106614318)

[Table 40 reward 69](#_Toc106614319)

[Table 41 Talent 70](#_Toc106614320)

[Table 42 Event 71](#_Toc106614321)

[*Table 43 Company class* 72](#_Toc106614322)

[*Table 44 job class* 73](#_Toc106614323)

[*Table 45 talent profile* 74](#_Toc106614324)

[*Table 46 admin class* 75](#_Toc106614325)

[*Table 47 comment class* 76](#_Toc106614326)

[*Table 48 Blog class* 77](#_Toc106614327)

[Table 49 Access control 92](#_Toc106614328)

[Table 50 Use cases and Test Cases 121](#_Toc106614329)

[Table 51 Description and the Expected Results of each test case 123](#_Toc106614330)

# List of Figures/Illustrations, Plates/Photographs

[Figure 1 Gant chart 12](#_Toc106614331)

[Figure 2 Use case diagram 44](#_Toc106614332)

[Figure 3 Use case diagram 2 45](#_Toc106614333)

[Figure 4 Activity diagram - Login 46](#_Toc106614334)

[Figure 5 Activity diagram - Talent registration 47](#_Toc106614335)

[Figure 6 Company and art school registration 48](#_Toc106614336)

[Figure 7 Activity Diagram-Edit profile 49](#_Toc106614337)

[Figure 8 Activity Diagram-Post job 50](#_Toc106614338)

[Figure 9 Activity Diagram-Apply job 51](#_Toc106614339)

[Figure 10 Activity diagram-post media 52](#_Toc106614340)

[Figure 11 Activity Diagram-Search 53](#_Toc106614341)

[Figure 12 Activity Diagram-Event 54](#_Toc106614342)

[Figure 13 Sequence diagram-login 55](#_Toc106614343)

[Figure 14 Sequence diagram-register compant school 56](#_Toc106614344)

[Figure 15 Sequence Diagram-Search for jobs, posts 57](#_Toc106614345)

[Figure 16 Sequence diagram- Create account Talent 58](#_Toc106614346)

[Figure 17 Sequence Diagram-Edit profile 59](#_Toc106614347)

[Figure 18 Sequence Diagram-Give feedback 60](#_Toc106614348)

[Figure 19 Sequence diagram-view posts, jobs. 61](#_Toc106614349)

[Figure 20 Sequence diagram-post jobs, rewards, events 62](#_Toc106614350)

[Figure 21 Sequence diagram-apply for job (talent) 63](#_Toc106614351)

[Figure 22 State diagram of Talent and System 65](#_Toc106614352)

[Figure 23 State diagram of Company and System 66](#_Toc106614353)

[Figure 24 State diagram of Admin and System 67](#_Toc106614354)

[Figure 25 System class diagram 78](#_Toc106614355)

[Figure 26 system decomposition 80](#_Toc106614356)

[Figure 27 Three-tier Software Architecture 88](#_Toc106614357)

[Figure 28 Component diagram of the system 89](#_Toc106614358)

[Figure 29 package diagram 90](#_Toc106614359)

[Figure 30 deployment diagram 91](#_Toc106614360)

[Figure 31 E-R diagram 91](#_Toc106614361)

[Figure 32 Hardware/software mapping 92](#_Toc106614362)

[Figure 33 Login page 94](#_Toc106614363)

[Figure 34 Home page 94](#_Toc106614364)

[Figure 35 Registration page 95](#_Toc106614365)

[Figure 36 Profile page 95](#_Toc106614366)

[Figure 37 Main page (before login) 96](#_Toc106614367)

[Figure 38 Main page (after login) 96](#_Toc106614368)

[Figure 39 People page 97](#_Toc106614369)

[Figure 40 Video page 97](#_Toc106614370)

[Figure 41 Company page 98](#_Toc106614371)

[Figure 42 Chat page 98](#_Toc106614372)

List of Abbreviations, Symbols, Specialized Nomenclature

**Glossary**

AZURE DEVOPS –a Software as a service (SaaS) platform from Microsoft.

CHROME – A free web browser by google.

ENZYME – A software for quality and regulatory.

ESLINT – static code analysis tool for identifying problematic patterns.

EXPRESS – a minimal and flexible Node.js web application.

FIREFOX – A free web browser by Mozilla.

JUST – A software used for testing.

LINUX – A free operating system.

MONGODB – document-oriented NoSQL Database.

NODEJS – open-source JavaScript runtime environment.

POSTMAN – an API platform for building and using APIs.

REACTJS – an open-source JavaScript library.

REACT-REDUX – A library that can be used for managing and centralizing application state.

WINDOWS – A proprietary operating system

**Abbreviation and acronyms**

ADMIN – Administrators

BSON – Binary JavaScript Object Notation

CSS – Cascading Style Sheets

DB – Database

DOM – Document Object Model

ER – Entity Relation

HTML –HyperText Markup Language

IOS – Iphone Operating System

JS – Java Script1

JSON – JavaScript Object Notation

MERN – MongoDB Express React Node

MVC –Model View Controller

OS – Operating System

OS – operating System

SQL – Structured Query Language

TMS – Talent Management System

UML – Unified Modeling Language

UI – User Interface

Abstract

*Talent management is a global human resources strategy aimed at identifying, developing, deploying, and retaining high-potential workers in an organization. [1] This project primarily focuses on presenting creative individuals and organizations to the outside world or other companies who are interested in their ability to transform our country's artistic revolution. Using web-based talent management systems and current technologies to assist distinct individual talents in obtaining jobs and to assist firms in identifying the proper talent they require has numerous advantages over the current system. The middle man problem and obtaining work based on their talent is one of the key issues that talents encounter. This project's goal is to create a web-based talent management system for gifted persons. The method delivers data that assists talents in making an informed selection about which job they wish to pursue. This paper contains models and designs that will aid in the development of the system.*

ረቂቅ

*የተሰጥኦ አስተዳደር በአንድ ድርጅት ውስጥ ከፍተኛ አቅም ያላቸውን ሠራተኞች ለመለየት፣ ለማዳበር፣ ለማሰማራት እና ለማቆየት ያለመ ዓለም አቀፍ የሰው ኃይል ስትራቴጂ ነው። ይህ ፕሮጀክት በዋናነት የሚያተኩረው የሀገራችንን የጥበብ አብዮት ለመለወጥ ያላቸውን አቅም ለሚፈልጉ ግለሰቦች እና ድርጅቶች ለውጭው ዓለም ወይም ለሌሎች ኩባንያዎች በማቅረብ ላይ ነው። በድረ-ገጽ ላይ የተመሰረቱ የችሎታ አስተዳደር ስርዓቶችን እና ወቅታዊ ቴክኖሎጂዎችን በመጠቀም የተለያዩ የግለሰቦችን ተሰጥኦዎች ሥራ ለማግኘት እና ድርጅቶች የሚፈልጓቸውን ትክክለኛ ችሎታዎች እንዲለዩ ለመርዳት አሁን ካለው ስርዓት ብዙ ጥቅሞች አሉት። የመካከለኛው ሰው ችግር እና በችሎታቸው ላይ የተመሰረተ ስራ ማግኘት ችሎታዎች ከሚያጋጥሟቸው ቁልፍ ጉዳዮች ውስጥ አንዱ ነው. የዚህ ፕሮጀክት ግብ ተሰጥኦ ላላቸው ሰዎች በድር ላይ የተመሰረተ የችሎታ አስተዳደር ስርዓት መፍጠር ነው። ዘዴው የትኛውን ስራ ለመከታተል እንደሚፈልጉ በመረጃ ላይ የተመሰረተ ምርጫ ለማድረግ ችሎታዎችን የሚያግዝ መረጃ ያቀርባል. ይህ ወረቀት ለስርዓቱ እድገት የሚረዱ ሞዴሎችን እና ንድፎችን ይዟል.*

# Chapter 1: Introduction

## 1.1 Background of the organization

Talent management is a global human resources strategy aimed at identifying, developing, deploying, and retaining high-potential workers in an organization.[1] This project primarily focuses on presenting creative individuals and organizations to the outside world or other companies who are interested in their ability to transform our country's artistic revolution. However, as we attempted to conduct study on the current talent management system in Ethiopia's capital, Addis Ababa, we discovered a slew of potential issues. Companies are frequently ignorant of these talented individuals, and the conduit that directly connects these two entities is frequently manipulated or controlled by unimportant brokers who effectively disrupt communication and the interrelationships between these entities They selfishly use the budget allotted to talent by company x for their own personal benefit and offer the talent a small cut. Furthermore, they frequently recruit these skills in a favoritism or prejudiced manner, or to their family or close friends who are not even qualified for the position, rather than the best talent for the job. As a result, our team has devised solutions to the problem. We are going to create a large talent management website platform that will only be open to students of talent infrastructures who need to show their portfolios to the outside world along with some necessary information about themselves, and these companies will be able to directly enter the platform website and select the best talent for their project. We primarily refer to talent as actors/actresses, models, and artists, and businesses as advertising agencies, designers, film producers, and photographers. In the future, we will make an effort to incorporate automatic way to verify that talent came from said school.

## 1.2 Statement of the problem

There is no adequate platform for talents to display their full potential for talent searchers in the current system, which causes a challenge in hiring and recruiting the correct talent for a specific job.

The following are the primary issues that are now affecting talent acquisition and hiring:

* Embezzlement: Because the middlemen manage the budget and deal with the talent seekers, the talent has no means of knowing how much money is assigned to them.
* Favoritism: the middle men hire individuals who are close to them.
* Talents do not have a specialized platform to build their profile. Their profile could be any type of their previous work.
* Talents are discouraged to work since they fall in the middle men interest.
* Sometimes the talent and talent seeker are not aware of each other.
* The talent seekers find it difficult to work with the talent that they have not selected for the job.
* It’s hard to tell if a talent fits in for the job
* The talent’s background is off the record that no one knows or insists if the talent goes to any art school before.
* There is also no specified platform for talent seekers to look for the appropriate talent and select the talent that fits with their expectations.

## 1.3 Existing system

The current talent recruiting system in our country as far as we can tell from our observations of the current system, the methods used by the talent to expose themselves to the outside world are social media platforms such as telegram, Facebook, Instagram, and others, or they try to contact middlemen to locate work for them. because these middle men have direct contact with the talent seekers, they send some talent for them depending on their wishes, and without the talent seekers' knowledge, the talent seekers are frequently disappointed, and the work results are often deemed unsatisfactory. One reason for this is that there is no proof that the talent is certified or qualified for the position. Instead, the talent establishes a financial arrangement with the middlemen. This indicates that the talent is unaware of the funds allocated to them by the talent seekers. There are also an excessive number of people who apply as talents to middlemen, which has become fashionable. These middlemen don't even inquire about their qualifications. They've also been observed prioritizing a close relative.

### 1.3.1 Major problem of existing system

* Brokers Selfishly consume the budget allocated by the company x to talent
* The companies are often unaware of these talented individuals
* Disruption of the communication and interrelation among these entities
* The mechanisms the talent use to expose themselves for the outside world are some types of social media like telegram, Facebook, Instagram etc. or they try to contact with those middlemen to find a job for them. But this type of mechanism makes it hard for the system to filter out the real talent since almost many people are doing the same thing.
* From profession point of view there is no any legal entity who could supervise these talents to build their profile in a professional way. For instance, if a talent needs to post their picture it needs to be taken with a professional camera and no filters on.
* The current system is too discouraging for both the talent and talent seekers in a way that the talent doesn’t get a chance to expose its profile and talent seekers do not get any place to select for the appropriate talent based on their previous work.
* The talent is not aware of the cash that the talent seekers assigned for them since they receive their payment from the middle men.
* Those middle men are not even professional entities they are just hired because of their wide network with the talent seekers.
* Inexperienced individuals are seen to take part on the job. there is no way that the talent seekers know if the talent is well experienced in the industry.

## 1.4 Proposed system

The proposed system we are going to build is a talent management website/android platform which will let students of talent producing infrastructures who need to show his/her portfolio to introduce themselves to the outside world easier. The proposed system will let students register on the website/application build their profile, release different works based on their talent. they will be using jobs to build their portfolio which will be released by companies. these companies can directly enter in to the web page see the different available talents on the website with their works and choose the appropriate talent for their project when the talent applies for the job which will get rid of the broker in the middle allowing the talents to receive the appropriate compensation. We mainly try to refer the talent as an actor/actress or a model, an artist, musician and the companies as different advertising companies, designers, film producers, photographers.

### 1.4.1 Advantage of proposed system

Unlike the traditional talent management system which we have stated earlier, our system is has the following advantage which has a potential impact to the problem on this area.

* Time saving: - It saves time for the companies that they can directly get into our website and recruit talents that fit with their required task
* Quality of hire: - The talents have the opportunity to demonstrate their portfolio to the companies through our website therefore the companies can decide whom to recruit through their portfolio before making any decision
* Cost optimization: - the companies make the business deal directly with the talent and the talent is required to pay his/her payment to the manager so there will not be any embezzlement activities regarding the process
* Recruitment - it can direct the companies to the individual they are looking for by checking out their portfolio therefore both the employee and employer can be satisfied by finding what they need which in turn can Finalization of various competencies around which future development of an employee is to take place
* Career advancement- talented individuals can advance their career by working with different companies gaining different experiences from them which in turn leads to advancement in their career
* Transparency- the talent and talent seeker can deal on the job without any third-party interference which is very promising system for the stakeholders to stay in the system and increase the quality of the work they intended to do.

## 1.5 Motivation

The Primary motivation of designing this Talent management system is to submit this system as a capstone project, so that we complete our 4-year undergraduate study as software engineering students. Technology is now familiar to almost everyone on the planet. It is also preferable if all activities are carried out using new technology available in order to modernize human life. At this time, many organizations are looking for talent, and each organization has a preferred skill and work to meet their needs.

Attracting and developing individuals who fully meet the organization's criteria is an important function of talent management. To this end, a talent management system should be designed, allowing for the full realization of their potential.

The current system has numerous flaws, which leads to various technical and operational issues as a result, this project is desired to address these issues.

## 1.6 Scope and limitation of the project

### 1.6.1 Scope of the project

The TMS (Talent management system) is an integrated software solution that covers the full scope of talent management namely:

* Talent recruitment for a specified job and searching for talents who fit the specified requirements raised by the companies.
* Processes to attract, onboard, develop, motivate, and retain high-performing talents.[2]
* This software solution only acts as a bridge between the talent and the company and manages the talents and directs them to the specified companies but doesn’t give any consulting advise
* It doesn’t interfere in the communication between the two entities.
* It is not responsible for compensating the talent once he or she has completed a job.
* The scope of the system usage covers only the region of Ethiopia-Addis Ababa specifically in regions where there is internet connectivity.

### 1.6.2 Limitation of the project

There may be some possible limitations in this project, such as: -

* Lack of prior system, we can’t find a local system that works on the same objective that solve the same problems. Many individuals with talents in our country uses nonautomated or manual way of finding a company for a job.

## 1.7 Objective

### 1.7.1 General objective

The main objective of this project is to develop a Talent Management System (TMS).

### 1.7.2 Specific objective

The specific objective of this talent management system is defined as follows: -

* Requirement analysis

Observing the process and existing system, questionnaires, and solitary thinking are some of the ways used to collect requirements.

* Design

For designing the diagrams, we will be using lucid charts, moon modeler and visual paradigm.

* Implementation

For implementation Vscode for editing, Reactjs, react-Redux, material-UI and react-router, for front end, Nodejs, express and back end respectively, and MongoDB for the database.

* Testing

For testing the project Eslint, Enzyme, just, and postman will be used.

* Deployment

Azure DevOps will be used.

## 1.8 Methodology

This project started with identifying and analyzing the flaws in the existing talent management system, then collecting all relevant data for analysis and design purposes of the new suggested system. Because it aids in completely modeling a system before moving to the development stage, the basic approach we employed to complete this project can be categorized as object-oriented modelling and design.

### 1.8.1 Data collection methodology

**Information gathering**

The methods and techniques that we have used to analyze the existing system to design the new web-based/mobile system include questionnaire and practical observation. These methods helped us to gather the required data to analyze our project. We have used observation to know how the existing system works. we tried to collect as much relevant information as possible and Data for developing the new system obtained from different sources which helped us to know the feedback of the talent and the companies.

**Observation:** Our team tried to have a careful and concise look at the situations related to talents.

**Questionnaires:** our project team prepared questions for talents and companies on how the whole process works.

**Secondary data collection methods**

We reviewed different websites on what the current system looks like and on how we should design our proposed project.

### 1.8.2 System design and analysis tools

During this phase, we have used object-oriented design approach to analyze the output from requirement gathering to obtain a complete list of functional and non-functional requirements that can be used to design and implement the system model as well as system design using a UML design are easier to understand by technicians and ordinary people.

### 1.8.3 System development tools

We will use a Waterfall model and follow the system development life cycle, which is comprised of a number of clearly defined and distinct work phases, to plan, design, build, test and deliver the talent management system.

In order to display the prototype, we used Adobe XD. The proposed system will be web-based/android. Hence, we plan to use web development tools and android development tools. For the android app we are going to use Flutter. For the website UI we are going to use react and material UI for front end, for state management React-redux, and NodeJS and express for backend and MongoDB for database.

We use different platforms to accomplish our tasks since we work on multiple computers and at different places and not everybody has the same preferences. But mostly we use visual studio code, MongoDB compass, postman and google chrome.

# Chapter Two: System Requirement Specification

## 2.1 Background Overview

Due to the lack of a specific purpose website platform in our country for a talent to reveal or reflect its talent for the outside world, we were able to understand the main difficulties that are encountered while using the existing system, in which the hiring authority is completely controlled by these middlemen. We were able to acquire some background information on the existing talent management system by conducting questionnaires with talent. They felt that models and actors, like professional athletes and other entertainers, could be handled as talent and that their fame could be exploited into strong personal brands. As a result, most talents in Addis Ababa are hired or picked for jobs in such a way that they have a close or seamless relationship with the middlemen. Occasionally, a person without any qualifications who has attended a talent school is noticed taking over the job. These people have made little effort in the field, which discourages many talented people. Furthermore, because the talent and the talent searchers do not make a straight contract about payment, some middlemen are found to postpone the payment that the talents deserve and even remove an unacceptable amount from the talents' payment for their own personal purposes. There are no procedures in place for talent seekers to pick the talent they require. As a result, we've proposed a web-based application that accomplishes several tasks in order to establish a direct line of communication between the talent and the talent seeker. Talent's name, talent's school name, talent's age, talent's experience (if any), talent's size information (such height, shoe size), talent's photo and video for application, talent's educational background, and company's (talent seeker's) name are some of the important inputs of the proposed system. By providing various features and interfaces, the suggested system will address potential issues in this area. This part contains a description of system functionalities as well as the necessary conditions for our system to function. In addition, we conducted a thorough evaluation of our system's viability. Functional requirement, nonfunctional requirement and feasibility study of the project is discussed here.

## 2.2 Functional Requirement

The developed system is expected to provide the following functionalities: -

**FR1** Login to the system

**FR2** Create account

**FR3** Post media

**FR4** Edit personal profile

**FR5** Register companies

**FR6** Post a Blog

**FR7** View jobs

**FR8** View events

**FR9** View posts

**FR10** View rewards

**FR11** Give feedback

**FR12** Respondtofeedback

**FR13** Search for jobs

**FR14** Search for posts

**FR15** Edit company profile

**FR16** Edit a Blog

**FR17** Edit a post

**FR16** Delete a Blog

**FR17** Delete a post

**FR18** Delete a user

**FR19** Delete Reward

**FR20** Give approval

**FR21** Get Rewards

**FR22** Post comment

**FR23** Post job

**FR24** Apply for job

**FR25** Generate report

**FR26** Prepare event

**FR28** Delete reward

**FR29** Logout

**FR30** Prepare reward

**FR31** Claim reward

**FR32** Send message

## 2.3 Non-Functional Requirement

The following list of nonfunctional requirements is expected from the system:

**Performance Requirement**

We will be using react to enhance the performance of the system. The virtual DOM is a virtual representation of the Document Object Model, where the UI is kept separately from the actual DOM and stored in the local memory. React synchronizes it with the actual DOM via reconciliation. Therefore, the virtual DOM doesn’t regenerate the entire page Instead of rolling out and processing new pages for each interaction, you can implement small changes there by making the performance faster,

**Interoperability** **Requirement**

* The System will be platform independent since it is web-based it will work for anyone who has any device that has a standard web browser such as Chrome, Firefox, and an operating system such as Windows, Linux, android, IOS.

**Reliability Requirement**

* The system will give only valid result, if no data is found with the specified criteria the system. The system will not crash or give invalid response and to achieve this we will use Nodejs which uses async/await and try-catch statements, or. catch () errors.

**Security Requirement**

* The system should allow login to only authorized users i.e., users that have username and password. The password is going to be hashed before being stored
* The system will ensure system level accounts have limited privileges by giving users. This helps to avoid attackers escalate user’s accounts to access administrator’s features.
* The system will use MERN which offers top-notch security for app development.one problem faced is SQL injection but since mongoDB is a NoSQL database MongoDB uses the Binary JSON (BSON) data format and comes with a secure BSON query assembly tool. Queries are also represented as BSON objects, so direct string injection is not possible.

**Availability Requirement**

* The system is web-based so it will be available 24/7 anywhere with an internet connection.
* The system will be available with a reasonable uptime. The uptime is calculated as the percentage of time during the year in which the software system was available to the public.

**Maintainability Requirement**

* The software system will be simple to maintain and cost-effective since it’s going to be built using an open-source technology MERN stack.

**Usability Requirement**

* The system will be user-friendly in order to make the system user interactive and easy to use it will be easy to navigate since we will be using components the navbar will be consistent and the features available will be conveniently accessible.
* The system will be easy to learn with a user-friendly user interface which requires little to no time of training. we will use Error messages which will give the user an understanding of what is wrong with the system to easily learn of the problem and possibly fix it.
* It includes many icons that are common and can be easily identified by users.
* It will provide specific guidelines to a user for using this system
* There will be a user manual provided to be distributed for every user of this system

## 2.4 Feasibility study

The feasibility issues listed below determine the project viability or the discipline of planning, organizing, and managing resources for bringing successful completion of the project’s objective.

### 2.4.1 Technical feasibility

Our project is technically possible because the system will provide users with quick access and they will be able to utilize it from their phone or any computer. Furthermore, all of the technology required for the project is readily available and accessible. We utilize React as a front end and combine it with NodeJS, Express, and MongoDB to create a database to hold the data, making this system availability and flexibility. This system can provide assurances of correctness, use, and dependability.

### 2.4.2 Operational Feasibility

The proposed system will provide accurate, secured service and decrease the labor workload. Also, the system will be easily operated and it satisfies the requirement of the project; it does not affect the existing organizational structure. So, the system will be operationally feasible.

### 2.4.3 Political Feasibility

The proposed system does not conflict with any governmental rules and regulations rather it gives services effectively and efficiently by providing the user with terms and conditions to agree to for using the system. So, the system is politically feasible.

### 2.4.4 Schedule feasibility

Within the time duration, the activities of the project in order to accomplish the project objective within the schedule requirement have been identified.

All the team members expect that the project will be completed within the timeframe stated so that the system will be feasible.

The steps involve in our project are

1. Planning
2. Requirement Gathering and analysis
3. Design
4. Testing
5. Implementation
6. Deployment
7. Final review

Our Gantt chart is as below

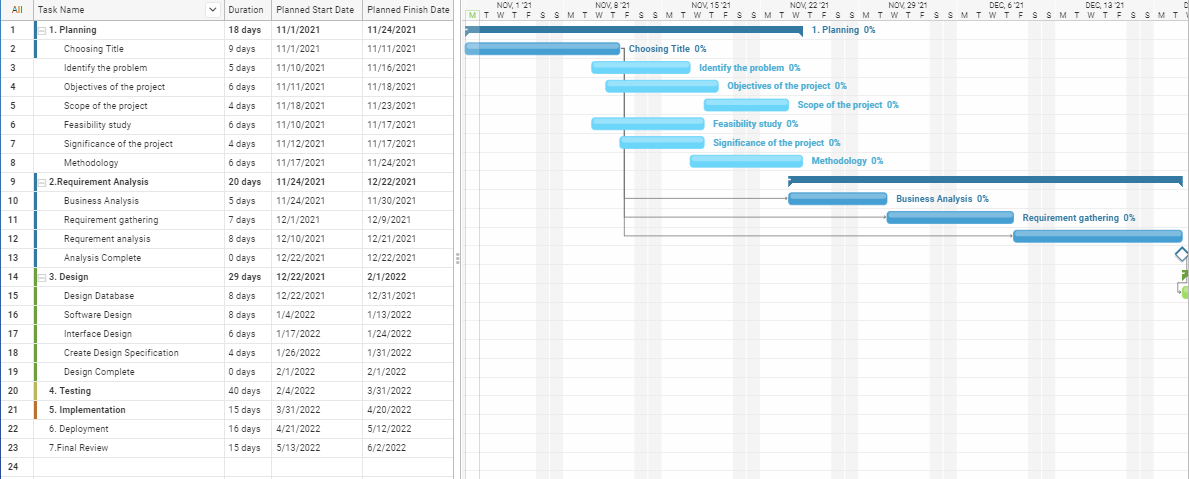


Figure 1 Gant chart

### 2.4.5 Behavioral feasibility

The project would be beneficial because it satisfies the objectives when the project is developed and deployed. All behavioral aspects are considered carefully and we can conclude that the project is behaviorally feasible.

### 2.4.6 Economic Feasibility

The proposed system is economically feasible since it enables the user to easily access the system with minimum cost and takes less capital since its built using open-source technology as compared to the existing system.

# Chapter Three: System Analysis and Modeling

## 3.1 Overview

In chapter 2, we have seen the different steps, approaches and the main processes involved in the talent recruitment and management system. In this chapter, the functional and non-functional requirements of the proposed system will be described and modeled using UML models.

## 3.2 Scenario Based Modeling

### 3.2.1 Use case identification

The following use cases have been identified from the system specification:

UC\_1 Login

UC\_2 Register companies

UC\_3 Create new account

UC\_4 Post a blog

UC\_5 Give feedback

UC\_6 View jobs

UC\_7 View posts

UC\_8 View event

UC\_9 View reward

UC\_10 View blog

UC\_11 Edit personal profile

UC\_12 Search for jobs

UC\_13 Search for posts

UC\_14 Post media

UC\_15 View profile

UC\_16 Generate report

UC\_17 Give approval

UC\_18 Respond to feedback

UC\_19 Post comment

UC\_20 Post job

UC\_21 Apply job

UC\_22 Prepare event

UC\_ 23 Edit blog

UC\_24 Logout

UC\_25 Delete blog

UC\_26 Delete post

UC\_27 Delete job

UC\_28 Delete reward

UC\_29 Send message

UC\_30 Search users

UC\_31 Delete user

UC\_32 Report post

UC\_33 Like post

UC\_34 Dislike post

UC\_35 Follow user

UC\_36 Unfollow user

UC\_37 Prepare reward

UC\_38 Claim reward

### 3.2.2 Actor identification

The identified actors that will be participating in the system are:

* Admin
* Company representative
* Talent
* Non-professional user

Table 1 use case to actor relation

|  |  |
| --- | --- |
| Use case | Actors |
| UC\_1 Login | Admin, Company representative, Talent, Non-professional user |
| UC\_2 Register companies | Admin |
| UC\_3 Create new account | Talent, Non-professional user |
| UC\_4 Post a blog | Company representative |
| UC\_5 Give feedback | Company representative, Talent, Non-professional user |
| UC\_6 View jobs | Talent |
| UC\_7 View posts | Admin, Company representative, Talent, Non-professional user |
| UC\_8 View event | Talent |
| UC\_9 View reward | Talent |
| UC\_10 View Blog | Talent, Company representative, Admin, Non-professional user |
| UC\_11 Edit personal profile | Talent, Admin, Company representative |
| UC\_12 Search for jobs | Talent |
| UC\_13 Search for posts | Company representative, Talent, Non-professional user |
| UC\_14 Post media | Talent |
| UC\_15 View profile | Admin, Company representative, Talent |
| UC\_16 Generate report | Admin |
| UC\_17 Give approval | Company representative |
| UC\_18 Respond to feedback | Admin |
| UC\_19 Post comment | Company representative, Talent, Non-professional user |
| UC\_20 Post job | Company representative |
| UC\_21 Apply job | Talent |
| UC\_22 Prepare event | Company representative |
| UC\_ 23 Edit blog | Company representative |
| UC\_24 Logout | Admin, Company representative, Talent, Non-professional user |
| UC\_25 Delete blog | Company representative, Admin |
| UC\_26 Delete post | Talent, Admin |
| UC\_27 Delete job | Company representative, Admin |
| UC\_28 Delete reward | Talent |
| UC\_29 Send message | Admin, Company representative, Talent |
| UC\_30 Search users | Admin, Company representative, Talent, Non-professional user |
| UC\_31 Delete user | Admin, Talent |
| UC\_32 Report post | Company representative, Talent, |
| UC\_33 Like post | Talent, Company representative |
| UC\_34 Dislike post | Talent, Company representative |
| UC\_35 Follow user | Talent |
| UC\_36 Unfollow user | Talent |
| UC\_37 Prepare reward | Admin |
| UC\_38 Claim reward | Talent |

### 3.2.3 Use case description

The descriptions of all the use cases are described in the following tables;

Table 2 use case login

|  |  |
| --- | --- |
| UC\_ID | UC\_1 |
| Use case Name | Login |
| Actors | Administrator, Nonprofessional users, company representative, Talent |
| Description | This use case allows the user to login into the system to access the relevant functions according to the user’s role. |
| Pre-condition | User must have valid password and username |
| Post-condition | The system displays the relevant page |
| Main Flows: | 1. The user submits username and password  2. The system validate the username and password  3. The system verifies the username and password  4. The system displays landing page  5. The use case end |
| Alternative Flows: | Missing username and/or password  2.1 The system prompts for username and password  Invalid password and /or username  2.2 The system displays invalid password and/or username  2.3 The system prompts for username and password |

Table 3 Register companies

|  |  |
| --- | --- |
| UC\_ID | UC\_2 |
| Use case Name | Register companies |
| Actors | Administrator |
| Description | This use case allows the administrator to assign username and password for company officials |
| Pre-condition | The admin must login to the system |
| Post-condition | Companies can access the system |
| Main Flows: | 1. The admin login to the system  2. The admin selects register new company  3. The admin submit the company detail  4. The system validate the input  5. The system creates password and username  6. The System display successful message  7. Use case ends |
| Alternative Flows: | Invalid input   * 1. The system displays error message   2. The system allows to refill the form |

Table 3 Create new account

|  |  |
| --- | --- |
| UC\_ID | UC\_3 |
| Use case Name | Create new account |
| Actors | Talent, Non-professional users |
| Description | This use case allows talent to create new account to access the system |
| Pre-condition | Open the website |
| Post-condition | The user can access the system |
| Main Flows: | 1. the user selects register  1. The user enters the new account details and chooses the type of user they are  2. The user submit the form  3. The system validates the form  4.The system displays successful message  5. Use case Ends |
| Alternative Flows: | If there is unfilled space   * 1. The system displays error message   2. The system allows to refill the form   if there is an invalid input   * 1. The system displays error message   2. The system prompt to attach the document |

Table 4 Post a blog

|  |  |
| --- | --- |
| UC\_ID | UC\_4 |
| Use case Name | Post a blog |
| Actors | Company representative |
| Description | This use case allows the company representative to post a blog |
| Pre-condition | The Company representative must login to the system |
| Post-condition | The company representative can post a blog |
| Main Flows: | 1. The user logs in to the system  2. The user submits the blog to be posted  3. The system save the media  4. Use Case End |
| Alternative Flows: | Invalid input   * 1. The system displays error message   2.2 The system allows to refill the form |

Table 5 Give feedback

|  |  |
| --- | --- |
| UC\_ID | UC\_5 |
| Use case Name | Give feedback |
| Actors | company representative, Non-professional users, Talent |
| Description | This use case allows users to give feedback and comments about the system. |
| Pre-condition | User must have email address |
| Main Flows: | 1. The user selects feedback from contact us option from home page  2. The system displays feedback page  3. The user inserts name, email address and give feedback on the text area provided  4. The user submits the feedback  5. The system saves the feedback  6. Use case ends |
| Post-condition | The user gives feedback |

Table 6 view jobs

|  |  |
| --- | --- |
| UC\_ID | UC\_6 |
| Use case Name | View jobs |
| Actors | Talent |
| Description | This use case allows the user to see the posted and available jobs on the website |
| Pre-condition | Open the website |
| Main Flows: | 1.The user gets access to the system  2.The user selects jobs from the main menus  2.The system displays posted jobs  3.The user view posted jobs  4. Use case ends |
| Post-condition | Talent view posted job |

Table 7 view post

|  |  |
| --- | --- |
| UC\_ID | UC\_7 |
| Use case Name | View posts |
| Actors | Talent, Non-professional users, company representative, admin |
| Description | This use case allows the user to see the posted media on the website |
| Pre-condition | Open the website |
| Main Flows: | 1.The user gets access to the system  2.The user is directed to the landing page  2.The system displays posted media  3.The user view posted media  4. Use case ends |
| Post-condition | User view posted media |

Table 8 view event

|  |  |
| --- | --- |
| UC\_ID | UC\_8 |
| Use case Name | View event |
| Actors | Talent, Non-professional users |
| Description | This use case allows the user to see the posted events on the website |
| Pre-condition | Open the website |
| Main Flows: | 1.The user gets access to the system  2.The user selects events from the main menus  2.The system displays posted events  3.The user view posted events  4. Use case ends |
| Post-condition | User view posted event |

Table 9 view reward

|  |  |
| --- | --- |
| UC\_ID | UC\_9 |
| Use case Name | View reward |
| Actors | Talent |
| Description | This use case allows the user to see the posted rewards on the website |
| Pre-condition | Open the website |
| Main Flows: | 1.The user gets access to the system  2.The user selects rewards from the main menus  2.The system displays posted rewards  3.The user view posted rewards  4. Use case ends |
| Post-condition | User view posted rewards |

Table 10 View blog

|  |  |
| --- | --- |
| UC\_ID | UC\_10 |
| Use case Name | View blog |
| Actors | Talent |
| Description | This use case allows the user to see the posted blogs on the website |
| Pre-condition | Open the website |
| Main Flows: | 1.The user gets access to the system  2.The user selects blog from the main menus  2.The system displays posted blogs  3.The user view posted blogs  4. Use case ends |
| Post-condition | User view posted rewards |

Table 11 Edit personal profile

|  |  |
| --- | --- |
| UC\_ID | UC\_11 |
| Use case Name | Edit personal profile |
| Actors | Talent, Non-professional users |
| Description | This use case allows the user to Edit their profile after registering |
| Pre-condition | The user must login |
| Main Flows: | 1. The user login to the system  2. The user goes to their profile  3. The user enters new information  4. The user submits the entered information  5. The system displays successful message  6.Use case Ends |
| Post-condition | The new user profile page is created |

Table 12 Search for jobs

|  |  |
| --- | --- |
| UC\_ID | UC\_12 |
| Use case Name | Search for jobs |
| Actors | Talent |
| Description | This use case allows users to search for specific jobs by using the search button on the page. |
| Pre-condition | Open the website |
| Post-condition | Results of searched information is displayed |
| Main Flows: | 1.The user enters the text to be searched  2. The user submit the information to be searched  3. The system retrieves the required information  4. The system displays the required information  5. Use case ends |
| Alternative Flows: | No such types of information   * 1. The system displays “no such file is found” message |

Table 13 Search for posts

|  |  |
| --- | --- |
| UC\_ID | UC\_13 |
| Use case Name | Search for posts |
| Actors | Talent, company representative, Non-professional users |
| Description | This use case allows users to search for specific posts by using the search button on the page. |
| Pre-condition | Open the website |
| Post-condition | Results of searched information is displayed |
| Main Flows: | 1.The user enters the text to be searched  2. The user submit the information to be searched  3. The system retrieves the required information  4. The system displays the required information  5. Use case ends |
| Alternative Flows: | No such types of information   * 1. The system displays “no such file is found” message |

Table 14 Post media

|  |  |
| --- | --- |
| UC\_ID | UC\_14 |
| Use case name | Post media |
| Actors | Talent |
| Description | This use case allows the user to post media on the website |
| Pre-condition | The Talent should login |
| Flow of Events | 1. The user logs in to the system  2. The user submits the media to be posted  3. The system save the media  4. Use Case End |
| Post-condition | New media will be provided for users |

Table 15 View profile

|  |  |
| --- | --- |
| UC\_ID | UC\_15 |
| Use case Name | View profile |
| Actors | Talent, Admin, company representatives |
| Description | This use case allows the view a talents profile |
| Pre-condition | The user must login |
| Post-condition | The profile is viewed |
| Main flows: | 1. The user login to the system  2. The user searches for the talent  3. The user selects view profile  4. The system displays profile page  5. Use case ends |

Table 16 Generate report

|  |  |
| --- | --- |
| UC\_ID | UC\_16 |
| Use case Name | Generate report |
| Actors | Administrator |
| Description | This use case allows the admin to generate a report on the required data |
| Pre-condition | The admin must login to the system |
| Main Flows: | 1. The admin selects generate report  2. The system generate reports  3. The system displays the successful message  4. Use case Ends |
| Post-condition | The report is generated on the required data |

Table 17 Give approval

|  |  |
| --- | --- |
| UC\_ID | UC\_17 |
| Use case Name | View report |
| Actors | Company representatives |
| Description | This use case allows users to review those who applied for the job |
| Pre-condition | The user must login to the system |
| Main Flows: | 1. The user login to the system  2. The user chooses view the list of talents who applied for the job  3. The system responds to the requested action  4. The system displays the list of talent  5. The user decide which talent he wants based on the job  6. The user clicks on approve or disapprove  7. the system displays successful message  8. Use case ends |
| Post-condition | The user view report |

Table 18 Respond to feedback

|  |  |
| --- | --- |
| UC\_ID | UC\_18 |
| Use case Name | Respond to feedback |
| Actors | Administrator |
| Description | This use case allows admin to respond to feedbacks from the users |
| Pre-condition | The user must login to the system |
| Main Flows: | 1. The user login to the system  2. The user selects respond to feedback  3. The system responds to the requested action  4. The system displays the list of user feedbacks  5. The user decide which feedback he wants to respond to  6. The user responds to selected feedback  7. the system displays successful message  8. Use case ends |
| Post-condition | The user responds to feedback |

Table 19 Post comment

|  |  |
| --- | --- |
| UC\_ID | UC\_19 |
| Use case Name | Post comment |
| Actors | company representative, Non-professional users, Talent |
| Description | This use case allows users to give comments about the post |
| Pre-condition | User must have a valid account |
| Main Flows: | 1. The user selects comment from the post  2. The system displays comment form  3. The user inserts name comment on the text area provided  4. The user submits the comment  5. The system saves the comment  6. Use case ends |
| Post-condition | The user gives comment |

Table 20 Post job

|  |  |
| --- | --- |
| UC\_ID | UC\_20 |
| Use case Name | Post job |
| Actors | Company representative |
| Description | This use case allows the user to post jobs on the website |
| Pre-condition | The user must login |
| Post-condition | The job is posted |
| Main flows: | 1. The user login to the system  2. The user enters the information’s about the job  3. The user selects submit  4. The system validate the input  5. The system display the success message  6. Use case ends |
| Alternative Flows: | Invalid input   * 1. The system displays error message   2. The system allows to refill the form |

Table 21 Apply job

|  |  |
| --- | --- |
| UC\_ID | UC\_21 |
| Use case Name | Apply job |
| Actors | Talent |
| Description | This use case allows the user to apply for jobs on the website |
| Pre-condition | The user must login |
| Post-condition | The job is applied |
| Main flows: | 1. The user login to the system  2. The user enters the information’s about the job he wants  3. The user submit  4. the user selects the type of job he wants  5. the user applies for the job  6. The system validate the input  7. The system display the success message  8. Use case ends |
| Alternative Flows: | Invalid input   1. The system displays error message 2. The system allows to reapply for a job |

Table 22 Prepare event

|  |  |
| --- | --- |
| UC\_ID | UC\_22 |
| Use case Name | Prepare event |
| Actors | Company representative |
| Description | This use case allows the user to prepare an event on the website |
| Pre-condition | The user must login |
| Post-condition | The event is posted |
| Main flows: | 1. The user login to the system  2. The user enters the information’s about the event  3. The user selects submit  4. The system validate the input  5. The system display the success message  6. Use case ends |
| Alternative Flows: | Invalid input   * 1. The system displays error message   2. The system allows to refill the form |

Table 23 Edit blog

|  |  |
| --- | --- |
| UC\_ID | UC\_23 |
| Use case Name | Edit blog |
| Actors | Company representative |
| Description | This use case allows the company representative to edit the blog |
| Pre-condition | Company representative must login |
| Post-condition | The blog is edited |
| Main Flows: | 1. The Company representative login to the system  2. The Company representative select edit form  3. The Company representative edit the form  4. The Company representative submit the edited form  5.The system validate the edited form  6. The system displays successful message  7. Use Case Ends |
| Alternative Flows: | If something is missing  5.1 The system displays error message  5.2 The system allows to re-edit the form |

Table 24 Logout

|  |  |
| --- | --- |
| UC\_ID | UC\_24 |
| Use case Name | Logout |
| Actors | Administrator, Non-professional users, companies’ representative, Talent |
| Description | This use case allows the user to exit or log out from the page |
| Pre-condition | User finish their work |
| Flow of Events | 1. The user submits logout  2. The system responds to the requested action.  3. The system returns the users to home page.  4. Use case Ends |

Table 25 Delete blog

|  |  |
| --- | --- |
| UC\_ID | UC\_25 |
| Use case Name | Delete blog |
| Actors | Administrator, companies’ representative |
| Description | This use case allows the user to delete a blog |
| Pre-condition | The actor is a registered user |
| Flow of Events | 1. The user logs into the system  2. The system authenticates the actor and starts a session  3. The actor chooses to delete a blog  4. The system prompts confirmation for deletion  5.The system acknowledges that blog deleted  6. Use case ends |

Table 26 Delete post

|  |  |
| --- | --- |
| UC\_ID | UC\_26 |
| Use case Name | Delete post |
| Actors | Administrator, Talent |
| Description | This use case allows the user to delete a post |
| Pre-condition | The actor is a registered user |
| Flow of Events | 1. The user logs into the system  2. The system authenticates the actor and starts a session  3. The actor chooses to delete a post  4. The system prompts confirmation for deletion  5.The system acknowledges that post deleted  6. Use case ends |

Table 27 Delete job

|  |  |
| --- | --- |
| UC\_ID | UC\_27 |
| Use case Name | Delete job |
| Actors | Administrator, companies’ representative |
| Description | This use case allows the user to delete a job |
| Pre-condition | The actor is a registered user |
| Flow of Events | 1. The user logs into the system  2. The system authenticates the actor and starts a session  3. The actor chooses to delete a job  4. The system prompts confirmation for deletion  5.The system acknowledges that job deleted  6. Use case ends |

Table 28 Delete reward

|  |  |
| --- | --- |
| UC\_ID | UC\_28 |
| Use case Name | Delete reward |
| Actors | Administrator |
| Description | This use case allows the user to delete a reward |
| Pre-condition | The actor is a registered user |
| Flow of Events | 1. The user logs into the system  2. The system authenticates the actor and starts a session  3. The actor chooses to delete a reward he has searched for  4. The system prompts confirmation for deletion  5.The system acknowledges that reward deleted  6. Use case ends |

Table 29 Send Message

|  |  |
| --- | --- |
| UC\_ID | UC\_29 |
| Use case Name | Send Message |
| Actors | Talent, Company representative, Admin |
| Description | This use case allows the user to send messages to other users |
| Pre-condition | The actor is a registered user |
| Flow of Events | 1. The user logs into the system  2. The system authenticates the actor and starts a session  3. The actor chooses Chat from the main menu  4. The user sends a message to the selected user he wants to contact  5.The system starts a chat session for the users  6. Use case ends |

Table 30 Search Users

|  |  |
| --- | --- |
| UC\_ID | UC\_30 |
| Use case Name | Search Users |
| Actors | Talent, Admin, Company representative |
| Description | This use case allows the user to search for users |
| Pre-condition | The actor is a registered user |
| Flow of Events | 1.The user enters the text to be searched  2. The user submit the information to be searched  3. The system retrieves the required information  4. The system displays the required information  5. Use case ends |
| Alternative Flows: | If the user doesn’t exist  2.1 The system displays no user found |

Table 31 Delete user

|  |  |
| --- | --- |
| UC\_ID | UC\_31 |
| Use case Name | Delete user |
| Actors | Talent, Admin |
| Description | This use case allows the user to delete a user account |
| Pre-condition | The actor is a registered user and authorized for deleting a user |
| Flow of Events | 1. The user logs into the system  2. The system authenticates the actor and starts a session  3. The actor chooses to delete a user  4. The system checks if the user is authorized to delete a user  5.The user searches for the user he/she wants to delete  6. The system prompts confirmation for deletion  7.The system acknowledges that user deleted  8. Use case ends |
| Alternative Flows: | If the user doesn’t exist  4.1 The system displays user not found message |

Table 32 Report post

|  |  |
| --- | --- |
| UC\_ID | UC\_32 |
| Use case Name | Report post |
| Actors | Talent, Company representative |
| Description | This use case allows the user to report a post |
| Pre-condition | The actor is a registered user |
| Flow of Events | 1. The user logs into the system  2. The system authenticates the actor and starts a session  3. The actor chooses to report a post  4.The system acknowledges the report  5. Use case ends |

Table 33 Like a post

|  |  |
| --- | --- |
| UC\_ID | UC\_33 |
| Use case Name | Like a post |
| Actors | Talent, Company representative |
| Description | This use case allows the user to like a post |
| Pre-condition | The actor is a registered user |
| Flow of Events | 1. The user logs into the system  2. The system authenticates the actor and starts a session  3. The actor chooses to like a post  4. The system acknowledges the like  5. Use case ends |

Table 34 Dislike a post

|  |  |
| --- | --- |
| UC\_ID | UC\_34 |
| Use case Name | Dislike a post |
| Actors | Talent, Company representative |
| Description | This use case allows the user to dislike a post |
| Pre-condition | The actor is a registered user |
| Flow of Events | 1. The user logs into the system  2. The system authenticates the actor and starts a session  3. The actor chooses to dislike a post  4. The system acknowledges the dislike  5. Use case ends |

Table 35 Follow user

|  |  |
| --- | --- |
| UC\_ID | UC\_35 |
| Use case Name | Follow user |
| Actors | Talent |
| Description | This use case allows the user to follow another user |
| Pre-condition | The actor is a registered user |
| Flow of Events | 1. The user logs into the system  2. The system authenticates the actor and starts a session  3. The actor chooses to follow the user  4. The system acknowledges the follow  5. Use case ends |

Table 36 Unfollow user

|  |  |
| --- | --- |
| UC\_ID | UC\_36 |
| Use case Name | Unfollow user |
| Actors | Talent |
| Description | This use case allows the user to unfollow another user |
| Pre-condition | The actor is a registered user |
| Flow of Events | 1. The user logs into the system  2. The system authenticates the actor and starts a session  3. The actor chooses to unfollow the user  4. The system acknowledges the unfollow  5. Use case ends |

Table 37 Prepare reward

|  |  |
| --- | --- |
| UC\_ID | UC\_37 |
| Use case Name | Prepare reward |
| Actors | Admin |
| Description | This use case allows the admin to prepare reward for talents |
| Pre-condition | The user must login |
| Post-condition | The reward is posted |
| Main flows: | 1. The user login to the system  2. The user enters the information’s about the reward  3. The user selects submit  4. The system validate the input  5. The system display the success message  6. Use case ends |
| Alternative Flows: | Invalid input   * 1. The system displays error message   2. The system allows to refill the form |

Table 38 Claim reward

|  |  |
| --- | --- |
| UC\_ID | UC\_38 |
| Use case Name | Claim reward |
| Actors | Talent |
| Description | This use case allows the user to claim a reward |
| Pre-condition | The actor is a registered user |
| Flow of Events | 1. The user logs into the system  2. The system authenticates the actor and starts a session  3. The actor chooses to claim a reward  4. The system checks if the user has enough points to claim said reward  5.The system acknowledges that reward as claimed  6. Use case ends |
| Alternative Flows: | If the user doesn’t have enough points  4.1 The system displays error message  4.2 The system allows to reclaim a reward |

### 3.2.4 Use case diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well.

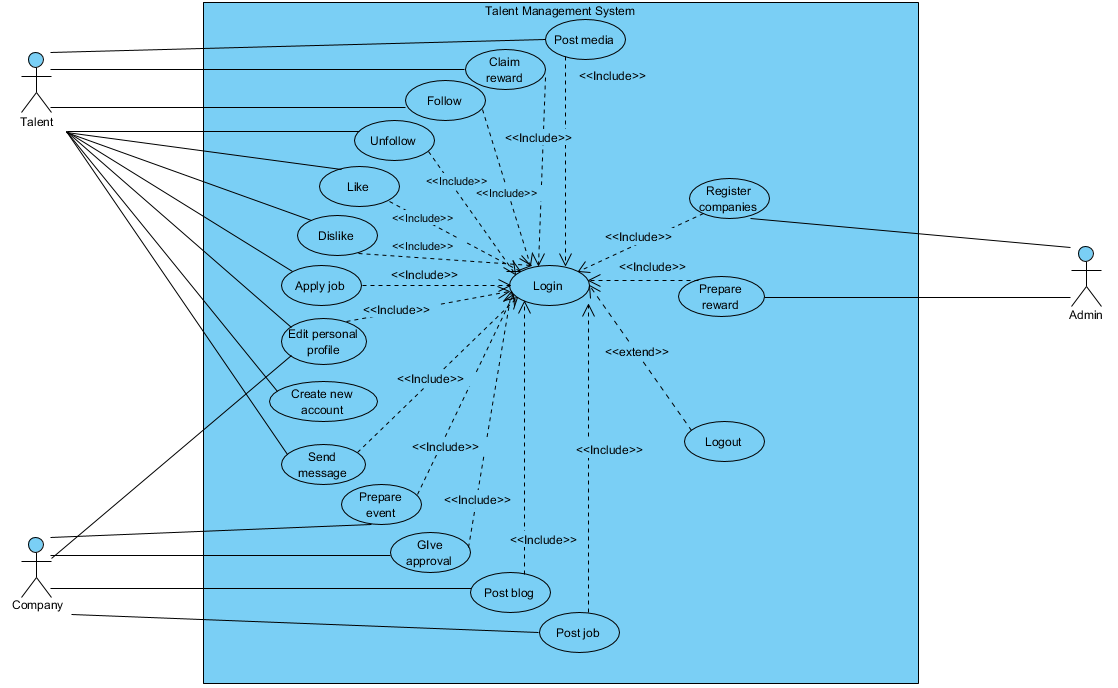


Figure 2 Use case diagram

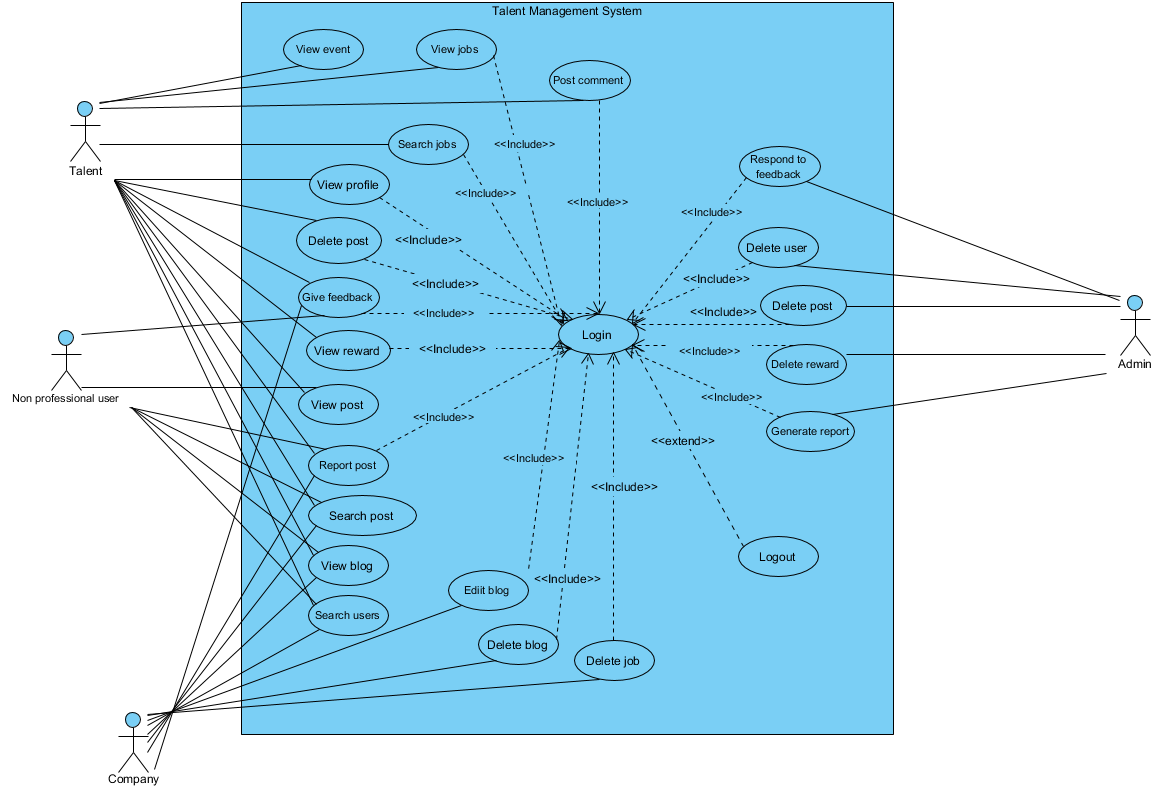


Figure 3 Use case diagram 2

### 3.2.5 Activity diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency that show how activities are carried out on our talent management. Activity diagram is basically a flowchart to represent the flow from one activity to another activity.[3] The activity can be described as an operation of the system. We have tried to show how the system activities flow using the following activity diagram.

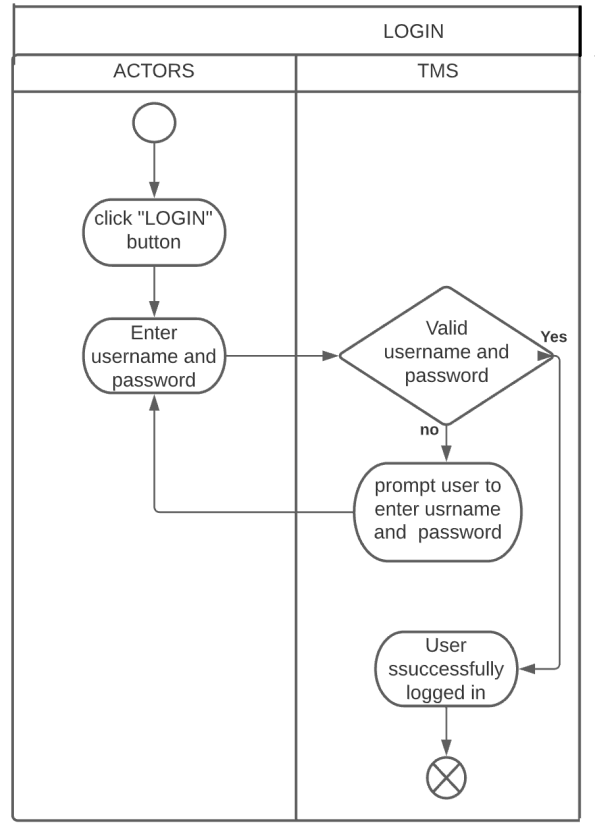


Figure 4 Activity diagram - Login



Figure 5 Activity diagram - Talent registration

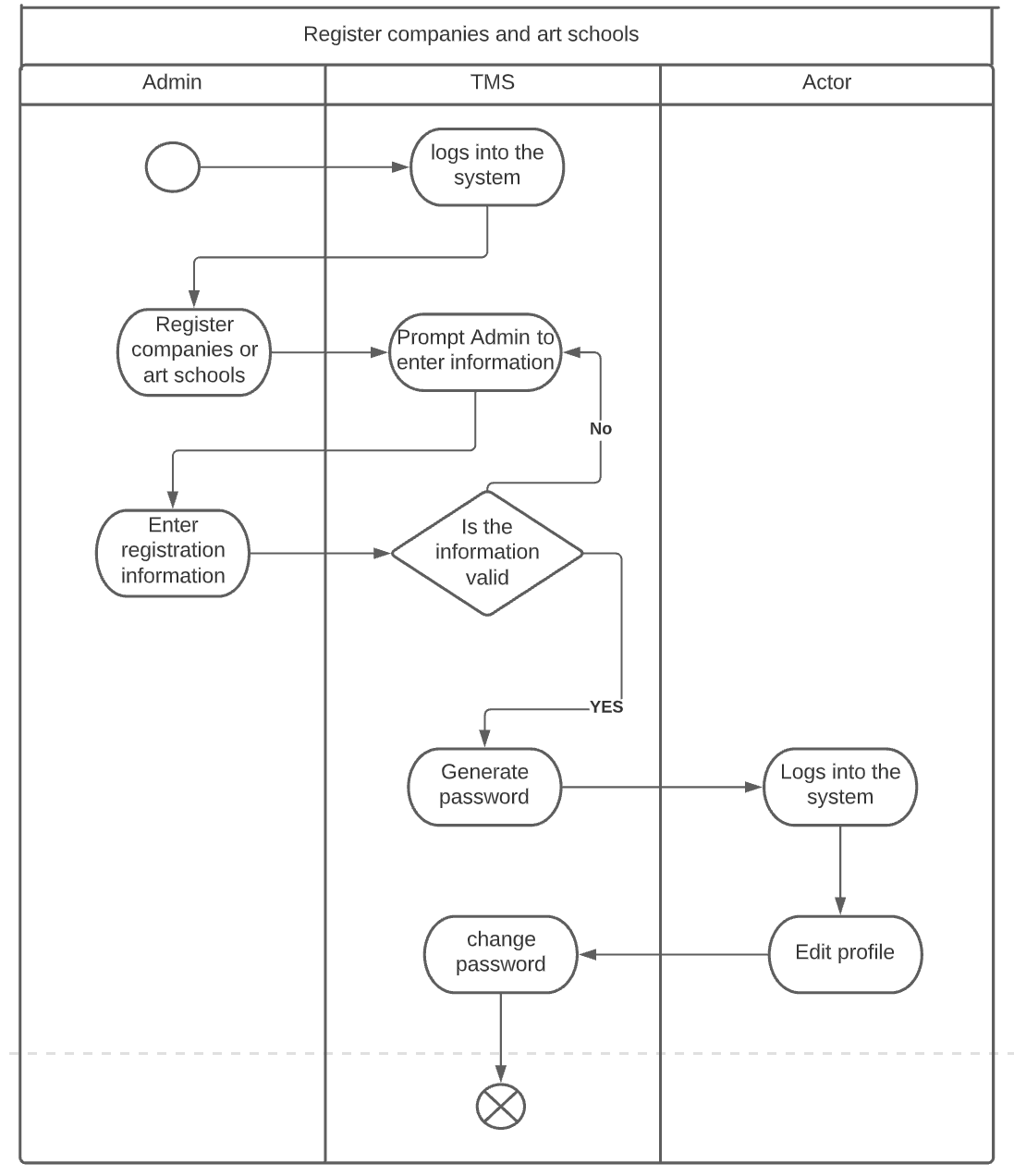
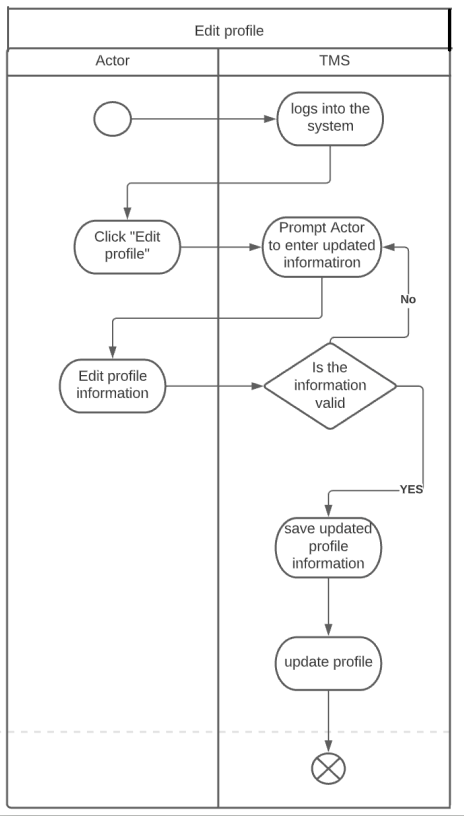


Figure 6 Company and art school registration



22

Figure 7 Activity Diagram-Edit profile

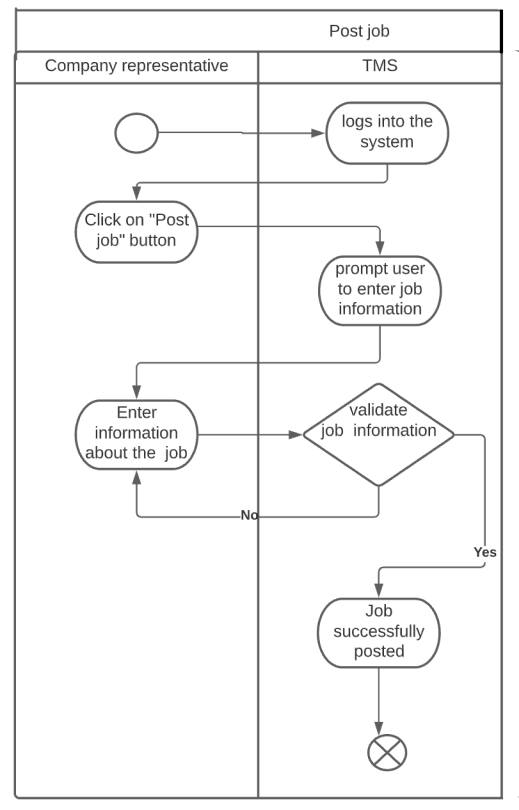


Figure 8 Activity Diagram-Post job

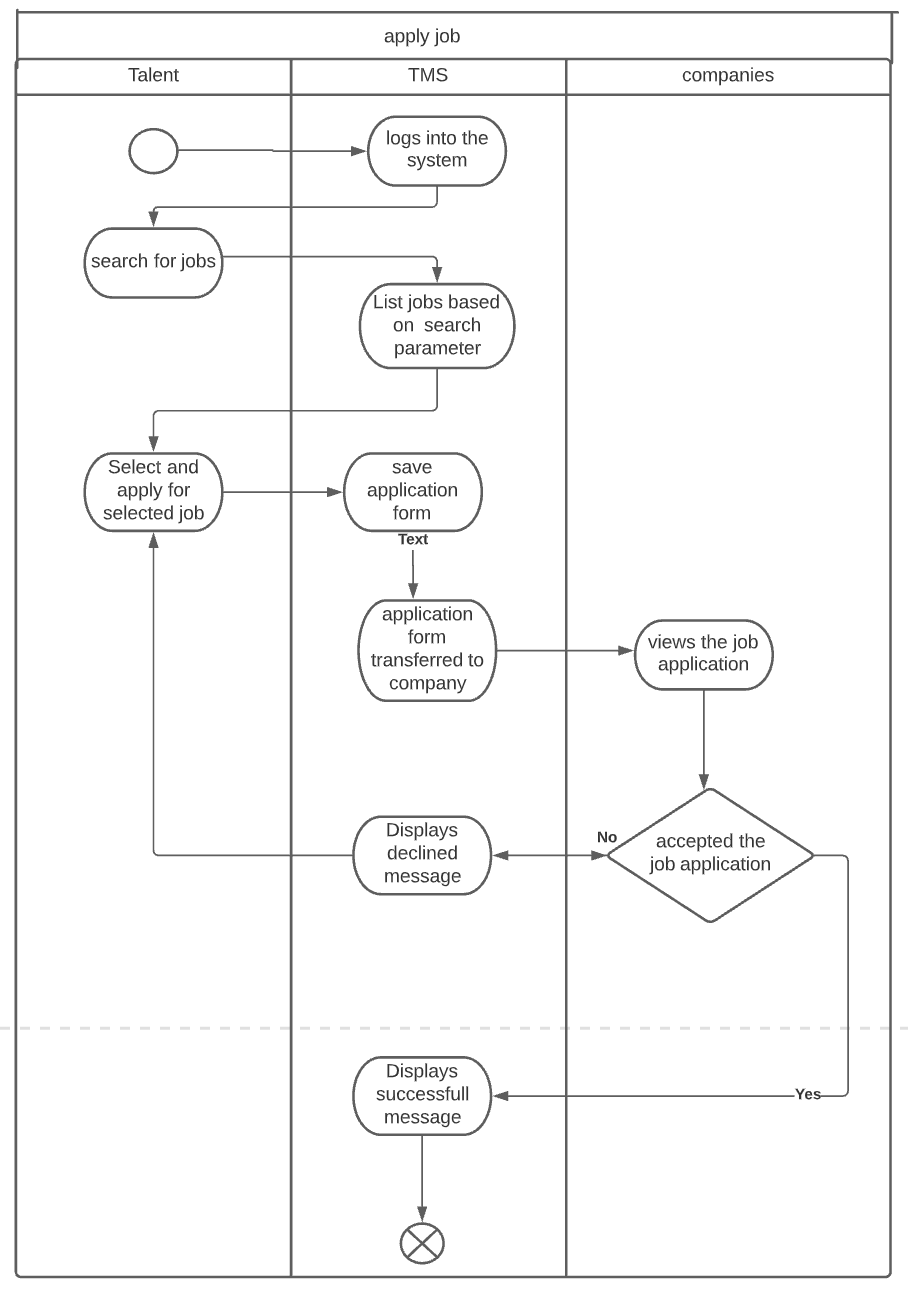


Figure 9 Activity Diagram-Apply job

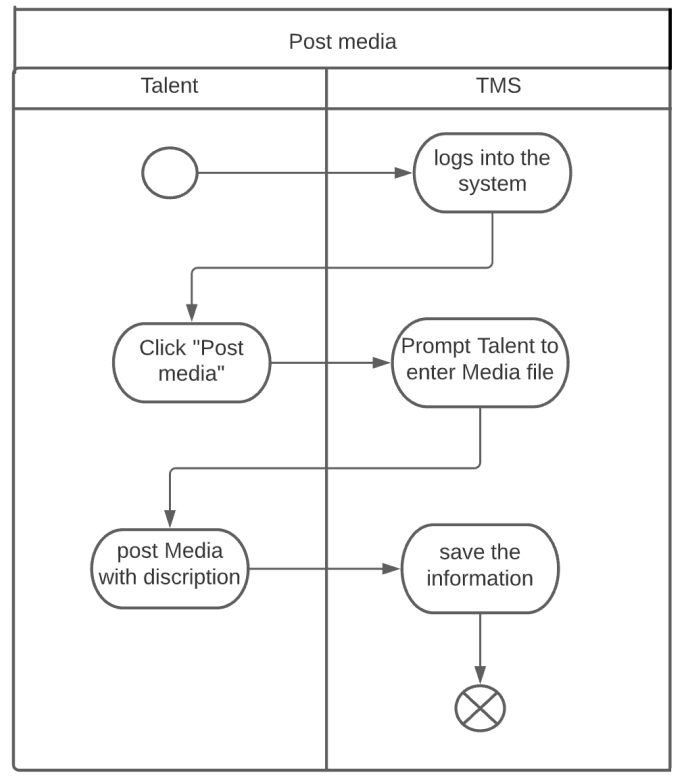


Figure 10 Activity diagram-post media

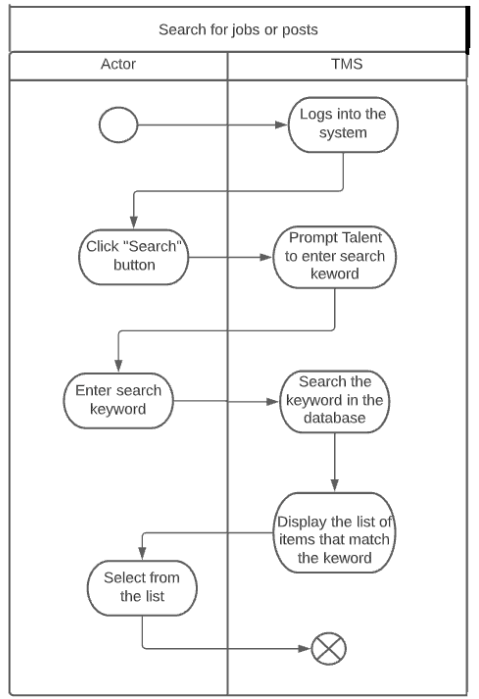


Figure 11 Activity Diagram-Search

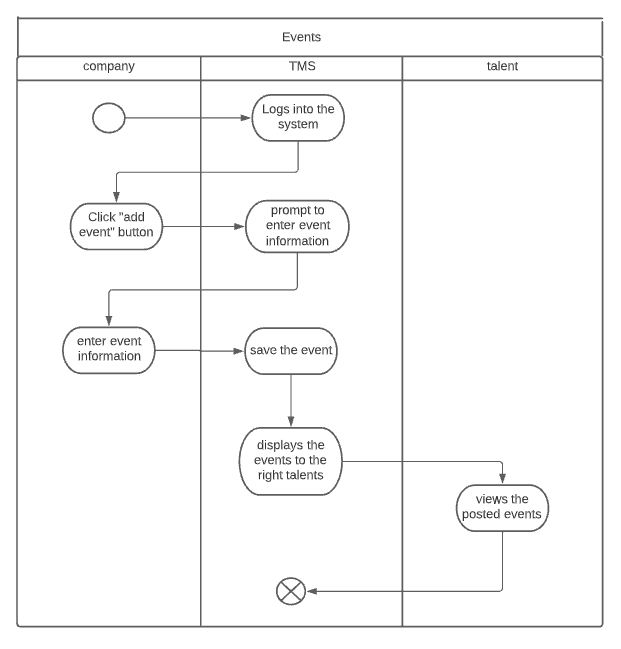


Figure 12 Activity Diagram-Event

## 3.3 Behavioral/Dynamic Modeling

### 3.3.1 Sequence diagram

Sequence Diagrams describe patterns of communication among a set of interacting objects. An object interacts with another object by sending messages. The reception of a message by an object triggers the execution of an operation, which in turn may send messages to other objects. The main sequence diagrams identified in our project are presented as follows.

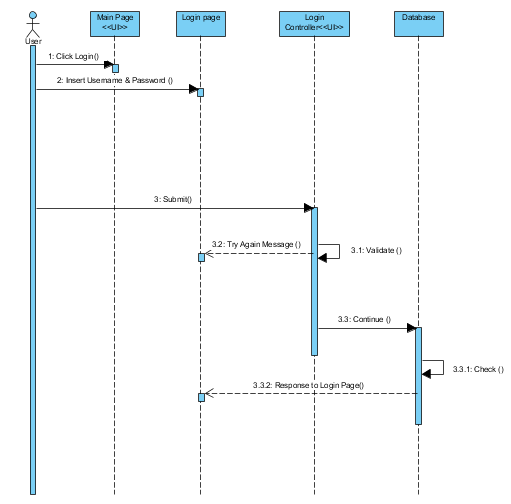


Figure 13 Sequence diagram-login

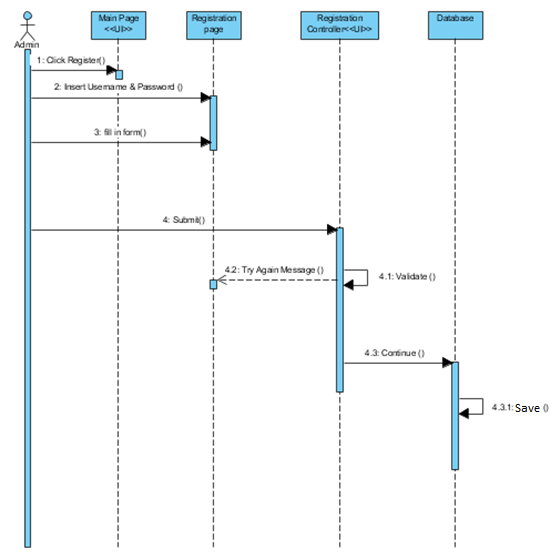


Figure 14 Sequence diagram-register compant school

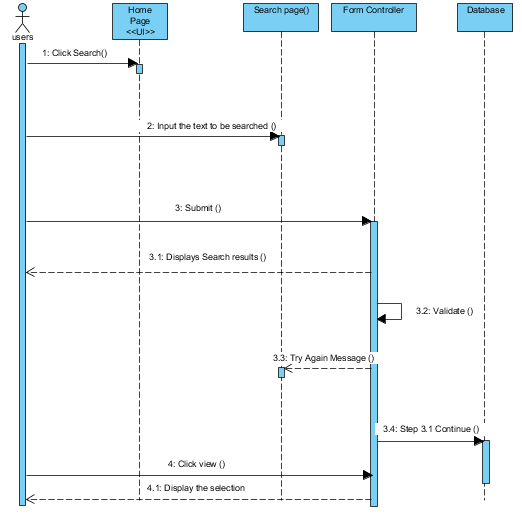
****

Figure 15 Sequence Diagram-Search for jobs, posts

****

Figure 16 Sequence diagram- Create account Talent

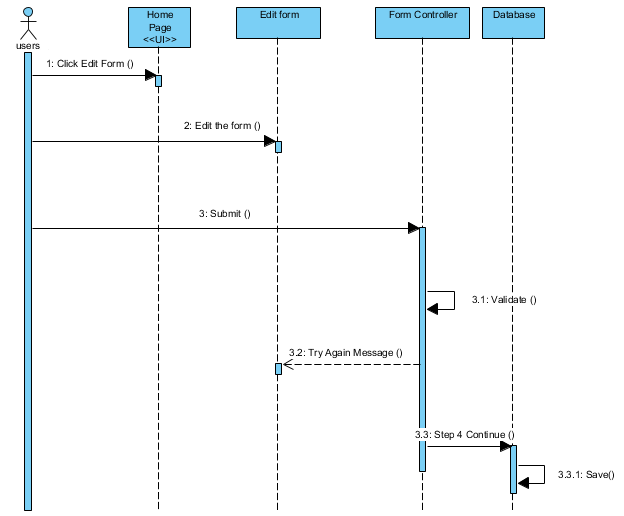


Figure 17 Sequence Diagram-Edit profile

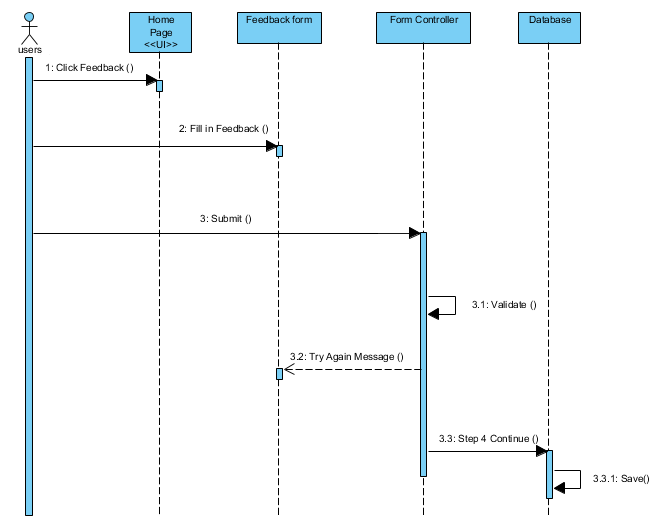
****

Figure 18 Sequence Diagram-Give feedback

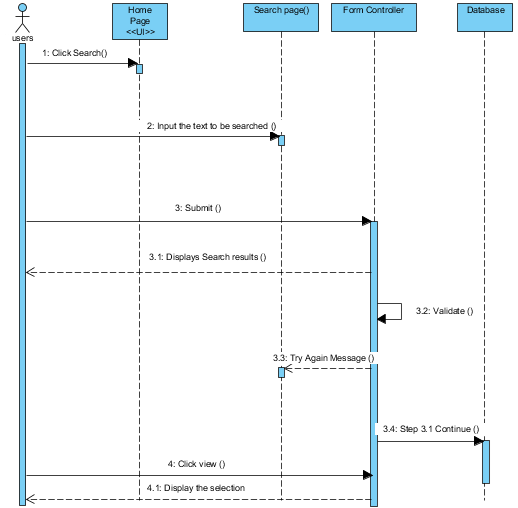
****

Figure 19 Sequence diagram-view posts, jobs.

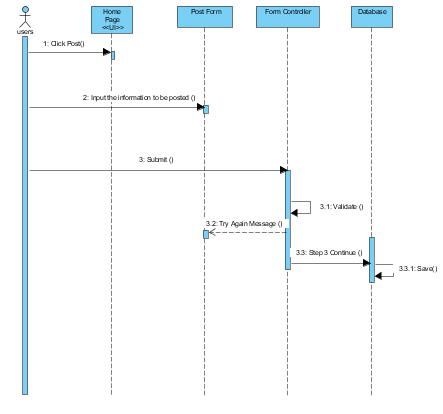


Figure 20 Sequence diagram-post jobs, rewards, events

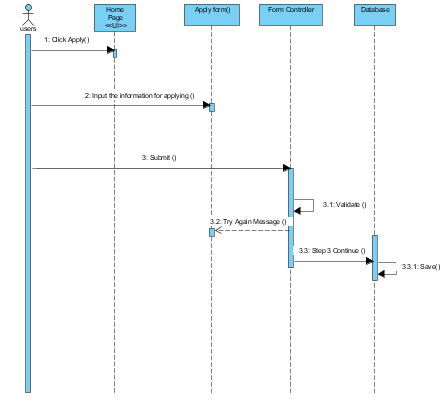
****

Figure 21 Sequence diagram-apply for job (talent)

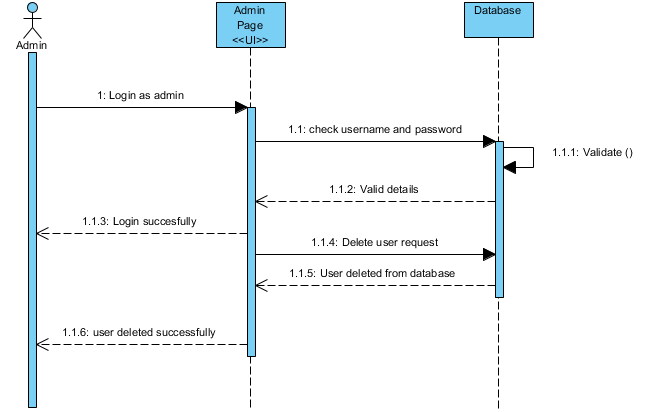
****

Figure 22 Sequence diagram delete user for admin

### 3.3.2 State Diagram

State diagrams or state chart diagrams describe the flow of control from one state to another state. These states are conditions in which objects exist and it changes when some event is triggered.

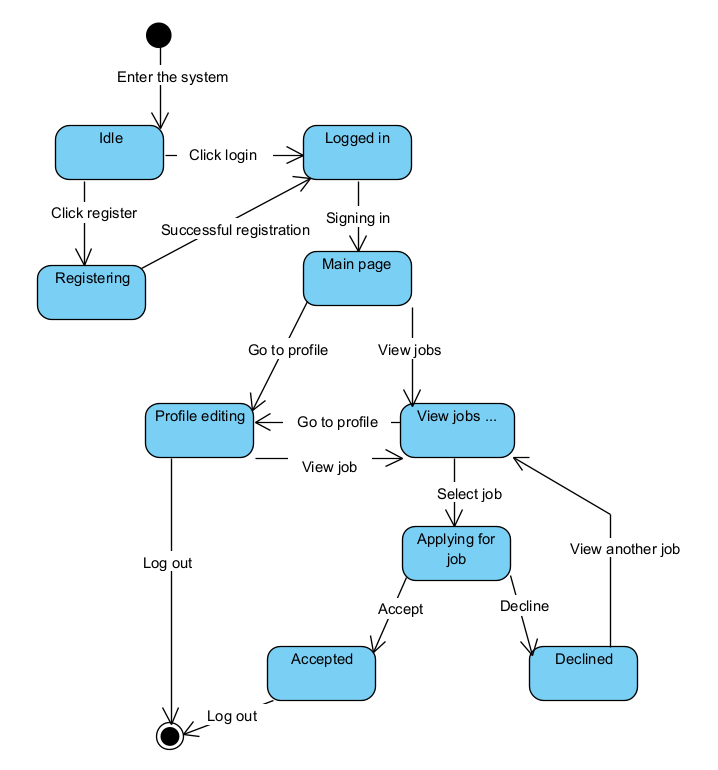


Figure 22 State diagram of Talent and System

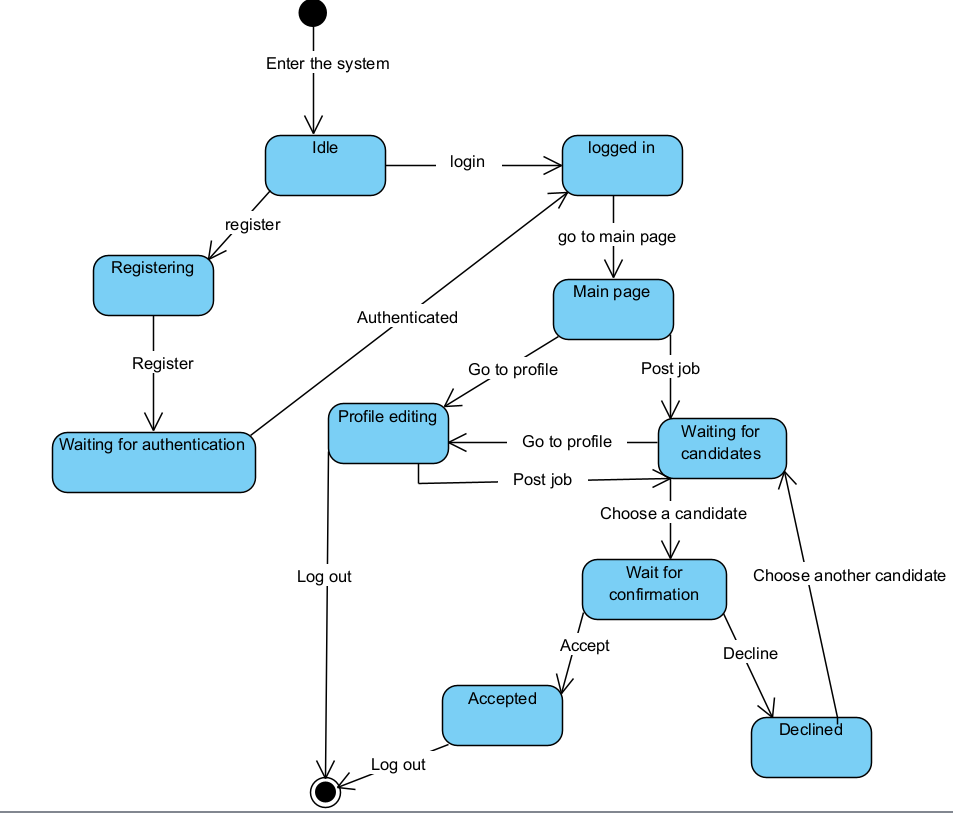


Figure 23 State diagram of Company and System



Figure 24 State diagram of Admin and System

## 3.4 Class-Based Modeling

### 3.4.1 Identifying classes

The following tables show the classes of the project with their attribute and attribute values:

*Table 39 User class*

|  |  |
| --- | --- |
| User | |
| Attributes/methods | Data type/function |
| User Name | String |
| Password | String |
| Role | String |
| Gender | String |
| Email | String |
| Phone number | String |
| adress | String |
| createdDate | Date |
| getName() | Get the name of the system user |
| getPassword() | Get the password of the system user |
| makeComment() | Makes comment |
| deleteComment | Deletes the comment |
| getAddress() | Get the address of the system user |
| Login() | Make user sign in to the system |
| updateProfile() | Can update their profile |
| GetRole() | User gets role as talent, company or adminstrator |

Table 40 reward

|  |  |
| --- | --- |
| Reward | |
| Attributes/methods | Data type/function |
| RewardType | String |
| Amountofpoints | String |
| image | Sring |
| SetRewardtype() | Sets the type of reward |

Table 41 Talent

|  |  |
| --- | --- |
| Talent | |
| Attributes/methods | Data type/function |
| Likes | int |
| age | int |
| height | Double |
| weight | Double |
| Applyjob() | Talent applies for the job |
| getStatus() | View the status |
| postTalent() | Posts their talent |
| editTalentpost() | Edit their talentpost |
| DeletetalentPost() | delete their talentpost |
| Register() | Talent registers to the system |
| getComment() | Gets the comment |
| collectReward() | Collects reward |
| GetRewardtype() | Gets the type of reward |
| ParticipateEvent() | Participate in event |

Table 42 Event

|  |  |
| --- | --- |
| Event | |
| Attributes/methods | Data type/function |
| EventType | String |
| date | date |
| EventDescription | String |
| duration | String |
| Setduration() | Duration of the event |
| SeteventDate() | When it will be held |

*Table 43 Company class*

|  |  |
| --- | --- |
| Company | |
| Attributes/methods | Data type/function |
| CompanyName | String |
| AreaOfSpecialization | Array |
| CompanyAdress | Array |
| getCompanyName() | Get the name of the company |
| getCompanyId() | Get the ID of the company |
| getCompanyAddress() | Get the address of the company |
| PostJob() | Company will post a job |
| UpdateJob | Update or edit the posted job |
| deletejob | Delete the posted job |
| giveApproval() | Company will select from the talent who applied |
| updateProfile() | Company will update their profile |
| Register() | Company registers to the system |
| createEvent() | Creats an event |

*Table 44 job class*

|  |  |
| --- | --- |
| Job | |
| Attributes/methods | Data type/function |
| Jobtitle | String |
| user | Object id |
| Applied user | Object id |
| Approved user | Object id |
| responsibilities | String |
| Requirement | String |
| JobType | String |
| Salary | String |
| Location | String |
| Description | String |
| postedDate | Date |
| expiredDate | Date |
| getJobName() | Get the name of the job |
| getJobId() | Get the ID of the job |

*Table 45 talent profile*

|  |  |
| --- | --- |
| Talent profile | |
| Attributes/methods | Data type/function |
| followers | Object id |
| following | Object id |
| Rating | String |
| Rating average | String |
| magicalRating | String |
| totallikes | int |
| Year of experience | int |
| language | Array |
| viewprofile() | Can see talent’s profile |

*Table 46 admin class*

|  |  |
| --- | --- |
| Admin | |
| Attributes/methods | Data type/function |
| getAdminName() | Get the name of the admin |
| Backup () | Store a backup of the system and database |
| registerCompanies () | Register all the companies |
| Deletetalentpost () | able to delete comments of users and companies |
| Deleteuser() | Can delete or remove user |
| Deletecommet() | Can delete unnecessary comments |
| Preparereward() | Prepare reward for talent |
| Login () | Admin must login to use the admin site |

*Table 47 comment class*

|  |  |
| --- | --- |
| Comments | |
| Attributes/methods | Data type/function |
| message | String |
| user | Object |
| date | date |
| post | Object id |
| Display () | Display comments |

*Table 48 Blog class*

|  |  |
| --- | --- |
| Blog | |
| Attributes/methods | Data type/function |
| TItle | String |
| Description | Array |
| date | date |
| catagory | String |
| user | Object id |
| coverphoto | String |
| Display () | Display blog |

### 3.4.2 Class Diagram

The following diagram shows the class diagram that shows the interactions between different classes of the system or the way how classes are interacted each other to accomplish their task.

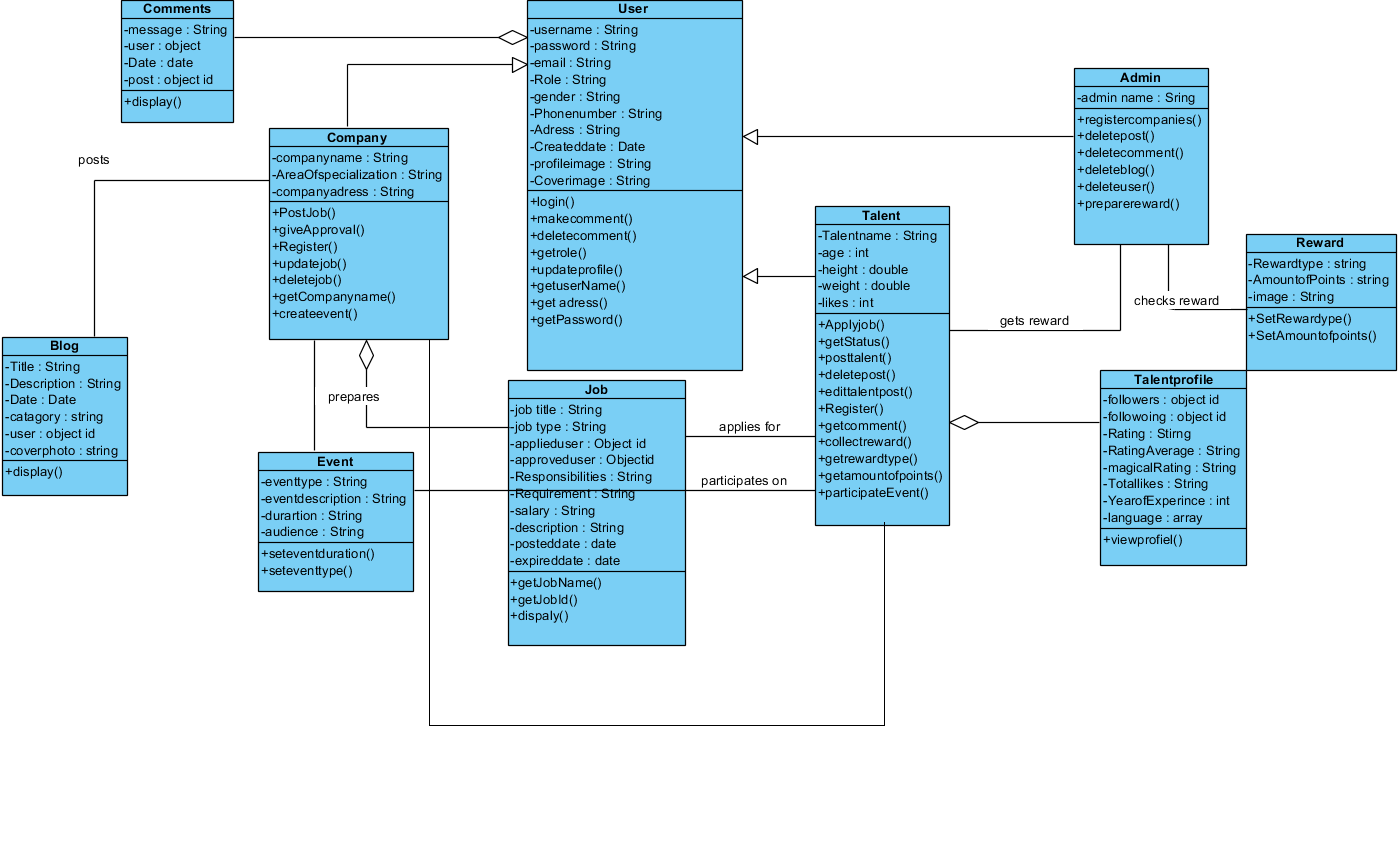


Figure 25 System class diagram

# Chapter Four: System Design

## 4.1 Overview

This chapter mainly concerned with the design part of Web based talent management system in Ethiopia/Addis Ababa. Up to now the project was focusing on the problem domain. Since system design is the first part to get into the solution domain in a software development this part will focus on transforming the analysis model into the design model that takes into account the non-functional requirements and constraints described in the problem statement and requirement analysis sections discussed earlier.

In this phase we illustrate design goal of the project, software architecture, component diagram, Database design, and user interface of the proposed system.

## 4.2 System Design

### 4.2.1 Design Goal

The overall system design goal is to provide an optimal, modular architecture that reduces system complexity, allows for change, and is simple to apply. This will be accomplished by creating a system that is loosely coupled but highly cohesive.[4] Furthermore, this document will provide interface design models that are consistent, user-friendly, and allow simple transitions between system functions.

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere which we will use to make our project implementation come to life.

The following are among the design goals that we have planned to achieve.

* Efficiency: the term efficiency in context to system characteristics refers to the ability of a system to run all its operations within a short period of time enabling the end user to process system jobs with very few commands. For our system to accomplish this we will segment our system into smaller modules. Each module will have a specific task.
* Performance: performance of a system is measured by its latency and throughput. The latency should be minimized while the throughput should be maximized working with the available bandwidth. For our system to accomplish this we will use a lightweight graphic interface so it will load fast, and we will choose a fast and dependable server.
* Scalability: Scalability is the property of a system to handle a growing amount of work by adding resources to the system. The fact that our system is composed of different services that can be scaled independently, and new components can be added without requiring downtime and redeployment of the entire system. Services can also be deployed in multiple servers which reduces the performance impact of more demanding components, which wouldn’t have been possible had we followed a monolithic architecture.
* Resiliency: Implementing micro service-based architecture adds ease to the process of identifying and resolving the root cause of performance issues. The improved fault isolation offered by individual modules means larger applications remain unaffected by a single failure. Consequently, the risk of downtime is reduced since developers can roll back an update or make changes to a module without redeploying the entire application.
* Reliability: ·The reliability of a system is usually defined as the probability that the system will operate without failure for a specified number of uses or a specified period. We will achieve this goal by using redundant hardware that will take on the services of the failed ones.
* Security: We have implemented security features on different levels, namely, in password hashing, using frameworks

### 4.2.2 System Decomposition

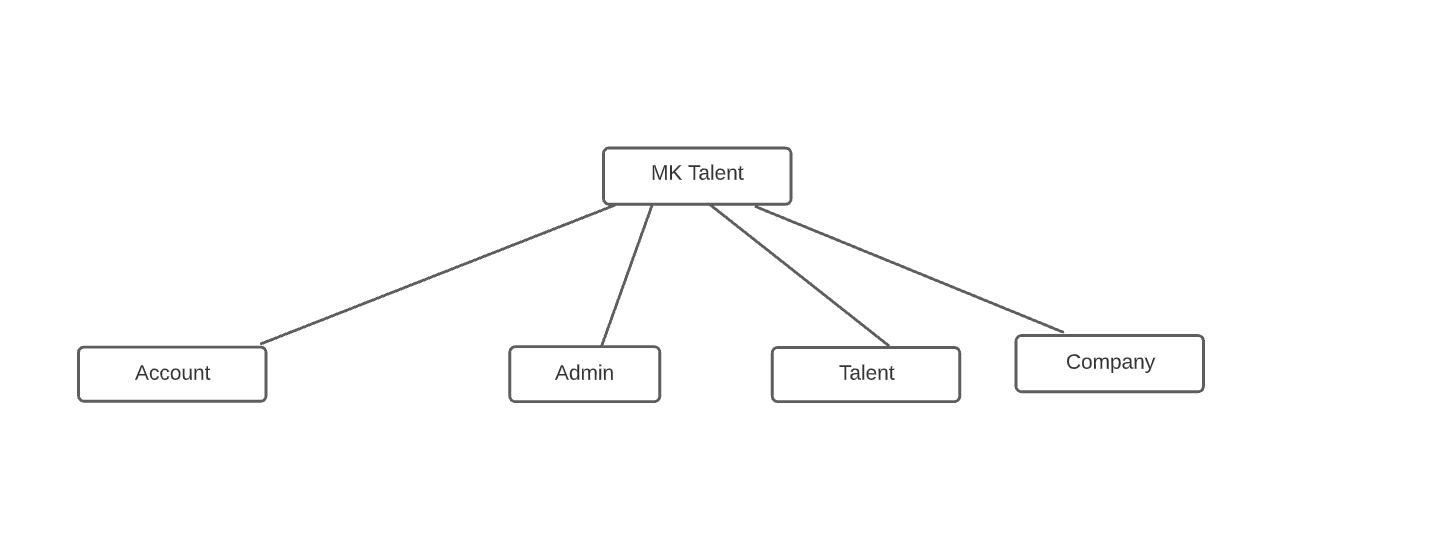


Figure system decomposition

### 4.2.3 Module Description

1. **ACCOUNT: -**

This module provides a way to create, delete, update, control, and manage the different accounts within the system.

* Provided Interface - differs per user
* Required Interface – each user's respective home page.

**Processing:**

* **Creating an account** Talent and Non-professional user can create their own account. Admin can create other admin accounts.
  + Talent fills in all the necessary information.
    - full name
    - username
    - birth date
    - sex
    - phone
    - image
    - document
    - age
    - address
    - email
    - password
  + Non-professional user fill in all the necessary information
    - full name
    - username
    - birth date
    - sex
    - phone
    - image
    - age
    - address
    - email
    - password
  + After filling in the information the user presses the ‘Register’ button and the account will be created ready to be used. Talent registration has to pass through the verification process by the admin.
  + A login page for the user based on the credentials will be displayed
* **Activating/deactivating and deleting an account**

Only the admin can activate, deactivate and delete other accounts. Talent’s account can also be deactivated by the admin

* Admin goes to the “Talent” or “user” tab.
* A List of all users will be displayed.
* By selecting any user, the admin can view detailed information about that user.
* The admin clicks on the “Activate”, “Delete”, or” Deactivate” button which will update/delete the user account.
* **Updating/ Viewing profile**

Each user can see/update their profile

* The user goes to the “profile” tab
* List of all information regarding the user will be listed
* By clicking “edit” the user can edit his/her information.
* Once the user updates the information, the user can click on the “save” button which will update the profile of that user.
* **Access Information**
  + In order to login and use the system the following access information is required;
* Email
* Password.
* Each user will be given a role that determines how they are going to access the system
  + **Talent**

Talent can create, view, and update their own information. If a Talent account is deactivated it can no longer access the system.

* **User**

User can create, view, and update their own information. If a Talent account is deactivated it can no longer access the system.

* **Admin**

Admins can update and view their own account, delete, activate and deactivate other accounts within the system. Can also create other admin accounts.

1. **Talent**

This module provides a way to create, delete, update, control, and manage the different Talent within the system.

* Provided interface: list of all previous Talent.
* Required Interface: Talent page.
* **Create/Post Talent**

Talent is the one who can post talent.

* Talent goes to the “Post” tab.
* Fill in all the required information.
  + Talent ID
  + Image
  + Caption
* Click on the “post” button which posts the Talent and make it available to all the users.
* **Edit/Delete Talent**

the one who posts the Talent is the one who can edit and delete that Talent.

* Doctor goes to the “profile” tab.
* List of all Talent will be listed.
* By clicking on the talent, the doctors can update the information or delete the Talent.
* Click on the “save” button, which updates the Talent.
* **Comment /Share/ View  Talent**

All user can share, view or comment on Talent

* The user goes to the “Home” tab.
* List of all Talent will be listed.
* The user can comment or share the talent by clicking on it and choosing the appropriate button for each purpose.

1. **Company**

This module provides all the required interface and processing logic for managing and applying for a Company

* Provided interface: defer based on the user
* Required Interface: Company page.
* **Registering by admin**
* **Create/update/delete Job**

Company will create jobs, update jobs, delete jobs.

1. The Company goes to “Add job” tab
   1. Fill in all the required information.
      1. Job name
      2. Post date
      3. Job type
      4. Caption

* Click on the “post” button which posts the Job and make it available to all the Talent.
* **Applying to get a job**

Talent can apply to get a job based on their choice.

* The Talent goes to the “job” tab.
* Talent can browse and search from the list.
* The Talent can click on the “details” button to see the Job information.
* The Talent clicks the “apply” button.
* **Accepting/Receiving talent request**

Company will receive requests from Talent which they can either accept or decline.

1. The Company will get notification
2. The Company goes to “Company” tab
3. The Company can see all Talent that request his Job
4. The Company can see the Talent and request detailed information by clicking on the “details” button.
5. The Company can click either accept or decline button which sends the result back to the Talent.

* The Talent clicks the “apply” button which will send the request to the Company
* **Make event**

Company can prepare

1. The Company goes to “event” tab
   1. Fill in all the required information.
      1. Event name
      2. Event type
      3. Image
      4. Created time
      5. Has reward
      6. Caption

* Click on the “post” button which posts the event and make it available to all the Talent.

1. **Admin**

This module provides all the required interface and processing logic for managing and applying for a Admin

* Provided interface: Admin page
* Required Interface: reward page, register page.
* **Register art school, company and other Admin**

Admin can register Art school, company and other admin

* Admin goes to the “Admin” tab and click add other admin/Art school/company.
  + Fill in all the required information based on the form.
* The admin clicks on the “registration”
* **Add reward**

1. Admin will create a reward
   1. Fill in all the required information.
      1. Reward name
      2. Expire date
      3. Created date
      4. Image
      5. Caption

* Click on the “post” button which posts the reward and make it available to all the Talent.

### 4.2.4 Purpose of the system design

The purpose of the system design document of talent management system is to find out how the system is to be constructed and to get the information needed, to control the actual implementation of the system. Here are the specific purposes of the system design document:

* To breakdown the system in to sub systems that are easy to be handled or controlled without much difficulty
* To find out the hardware/software stages on which the system will operate.
* To determine access control of each class operation.
* To develop a single view system

## 4.3 Architecture of the system

### 4.3.1 Architectural style

The architectural style is a very specific solution to particular software that typically focuses on organizing the code created for the software.

We chose the Layered (n-tier) architecture and event driven architecture for our website.

**Layered (n-tier) architecture**

This approach is probably the most common because it is usually built around the database, and many applications in business naturally lend themselves to storing information in tables.[5]

Since We are going to use Express it was built with this structure in mind, so chose this layered architecture style.

The code is arranged so the data enters the top layer and works its way down each layer until it reaches the bottom, which is usually a database. Along the way, each layer has a specific task, like checking the data for consistency or reformatting the values to keep them consistent. It’s common for different programmers to work independently on different layers.

**Event driven architecture**

Many programs spend most of their time waiting for something to happen. This is especially true for computers that work directly with humans, but it’s also common in areas like networks. Sometimes there’s data that needs processing, and other times there isn’t.

The event-driven architecture helps manage this by building a central unit that accepts all data and then delegates it to the separate modules that handle the particular type. This handoff is said to generate an “event,” and it is delegated to the code assigned to that type.[5]

Since this architectural style is best suited for Asynchronous systems with asynchronous data flow and nodejs has an asynchronous event-driven architecture we chose this event driven architectural style [6]

### 4.**3.2 Architectural Pattern**

An architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture within a given context. The user interfaces of our system change often and the same information is presented in different ways, but the core business logic and data is stable. Due to this reason, we will be using the MVC architectural pattern

The proposed system has a three-layer system architecture design. Using this architecture, the system is divided into 3 different layers:

Client layer

* These are views(webpages) which is visible to the client.
* Client layer will be developed using React which combinedly uses Java script, HTML & CSS.
* On this layer, end users try to access the feature of our application.

Application layer

* This is the business logic layer which is an Express application on top of Node.js platform.
* This application server acts as the bridge of communication between client & database layer.
* This layer accepts client's requests and serves them by fetching appropriate data from database layer.

Database layer

* Database layer runs the mongoDB server.
* All application data is stored in this mongoDB database.

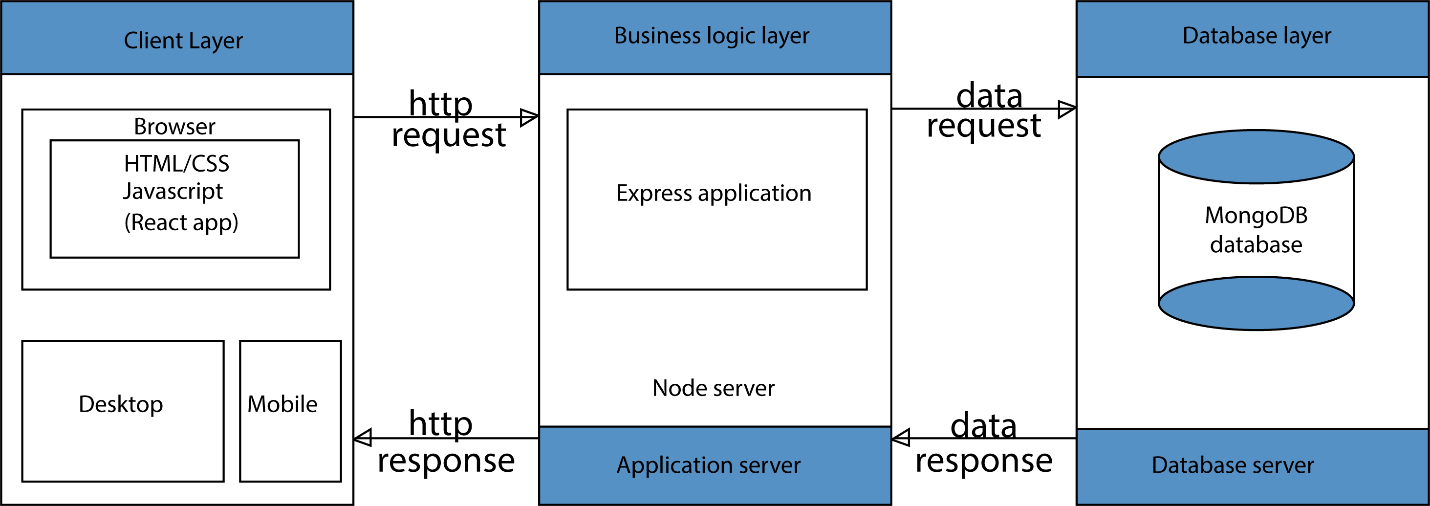


Figure 27 Three-tier Software Architecture

### 4.3.3 Component Diagram

The following component diagrams show how the proposed system physical components are organized. In this diagram components of the system will be wired showing that there is relation among components, management of the system, database and operations performed on databases and security.

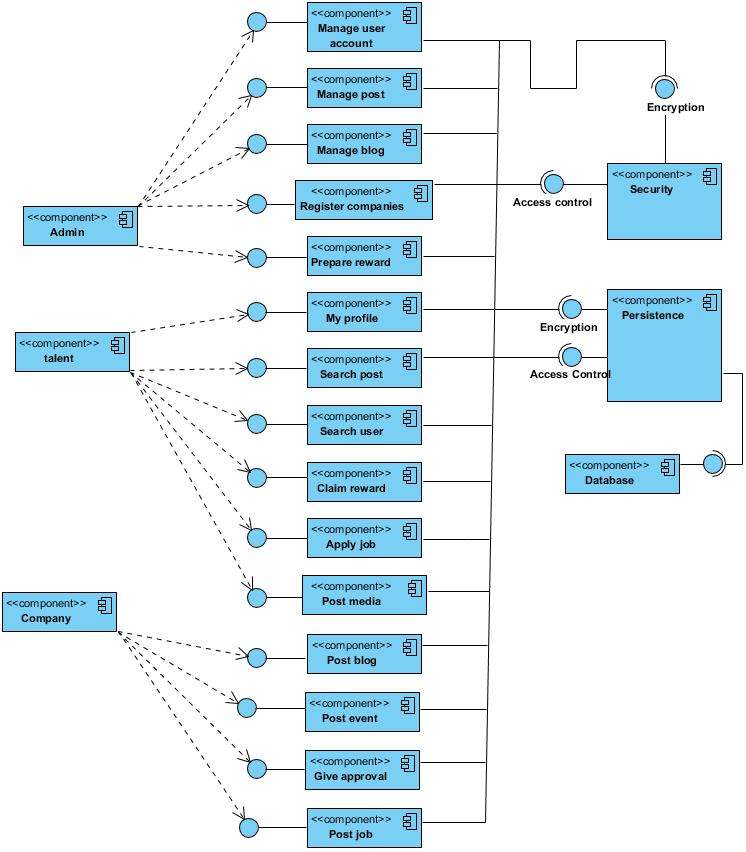


Figure 28 Component diagram of the system

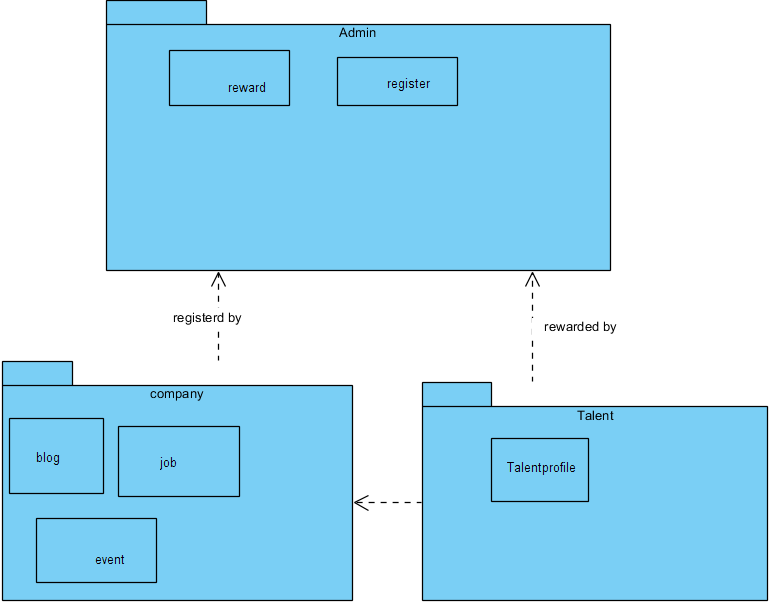


Figure 29 package diagram

### 4.3.4 Deployment diagram

A UML Deployment diagram models the execution architecture of a system, including nodes such as hardware, software, execution environments, and the middleware connecting them. They are typically used to visualize the physical hardware and software of a system. They describe the hardware components, where the software components are deployed.

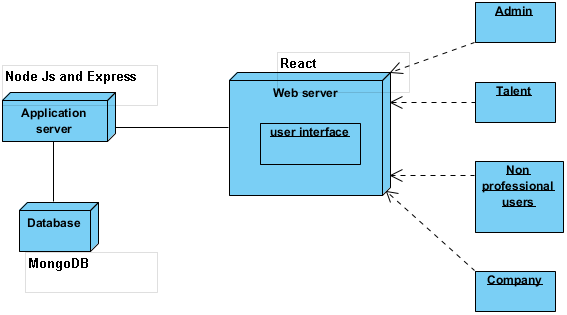


Figure deployment diagram

## 4.5 Database design

The database subsystem subsumes the Data base management subsystem. This sub system provides management of data for the talent management system.

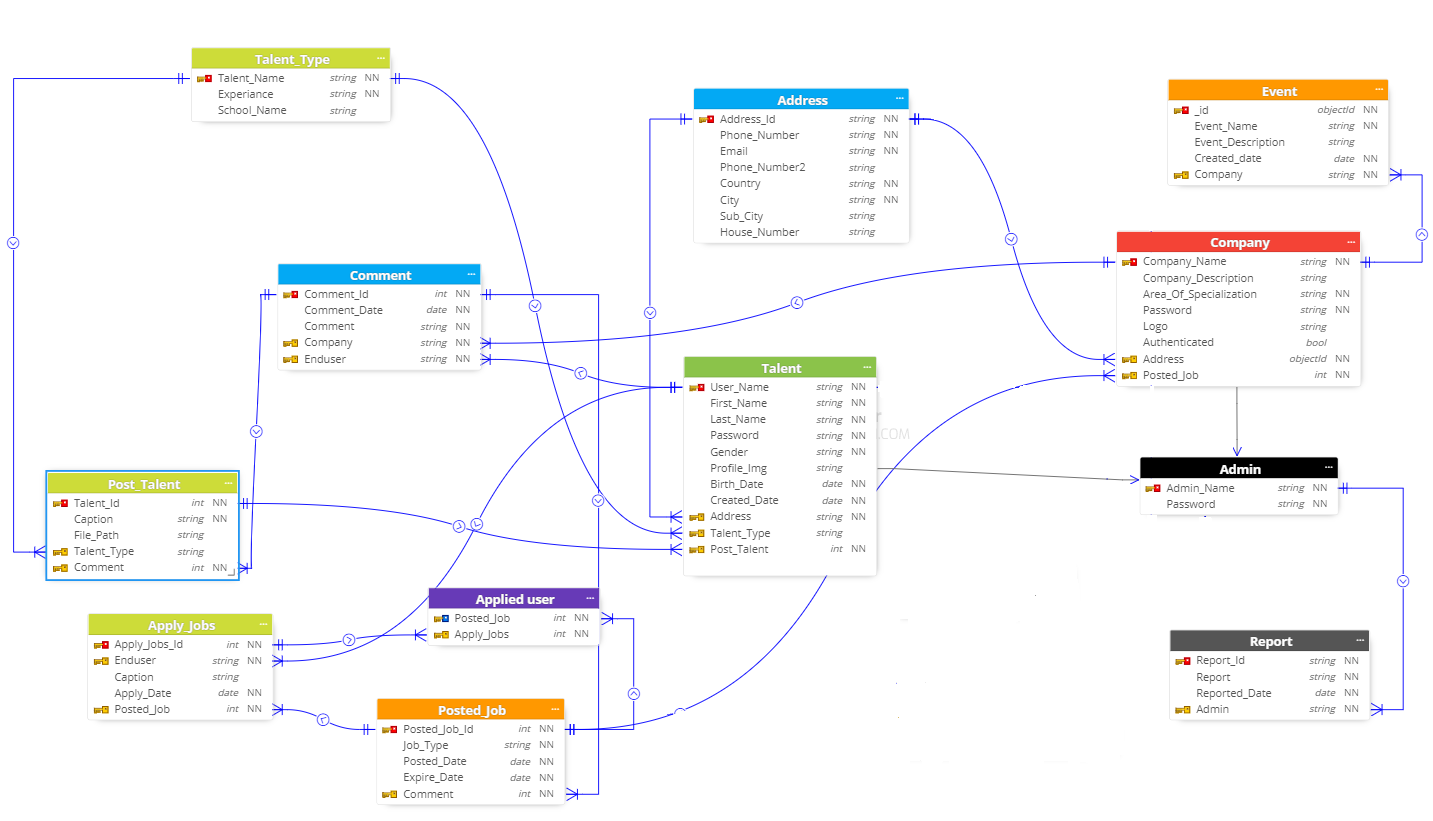


Figure 31 E-R diagram

## 4.6 Hardware / Software Mapping

This section provides the information needed for a system development team to build and integrate the hardware components, code and integrates the software modules, and interconnect the hardware and software segments into a functional product. The Talent management system includes three major components namely client, backend-server and database component. Clients communicate with the server and the server then communicates with the database.

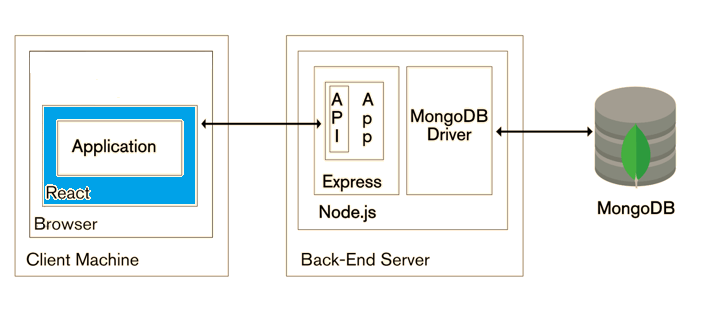


Figure 32 Hardware/software mapping

## 4.7 Access control

Access control is a security technique that regulates who or what can view or use resources in a computing environment. In other terms, access control is a process by which users are granted access and certain privileges to systems, resources or information.

Table 49 Access control

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Actor  Action | End User (talent) | Companies Representative | Administrator | Guest user |
| 1. | Login | Access | Access | Access | No access |
| 2. | Register | Access | Access | No access | No access |
| 3. | View talents | Access | Access | Access | Access |
| 4. | Rate talents | Access | No access | No access | Access |
| 5. | Apply jobs | Access | No access | No access | No access |
| 6. | Post jobs | No access | Access | No access | No access |
| 7. | Update Information | Access | Access | No access | No access |
| 8. | Give feedback | Access | Access | No access | Access |
| 9. | Generate and view Reports | No access | No access | Access | No access |
| 10. | Authenticate company | No access | No access | Access | No access |

## 4.8 Single view system

A sound data management strategy can help us unlock unlimited opportunities from our data by unifying data across many platforms. But developing a single view of data across all platforms is not easy and many have failed in this pursuit.

MongoDB can easily help us achieve that single view. As a next generation database offering a flexible data model, MongoDB allows for wide data integration, all for less money than traditional database technologies.

How MongoDB makes single view possible:

* Manages any type of data- Be able to build powerful applications that incorporate any type of data from any source.
* Dynamic schema- Be able to iterate more quickly thanks to flexible, dynamic schema. We will be productive as we spend less time prepping the data and more time moving the project forward.
* Less cost- Efficiency gains greatly impact the bottom-line while being able to use commodity hardware means our projects can be a fraction of the cost associated with a relational database.

## 4.9 User Interface Design

The user interface of the proposed system looks like the following

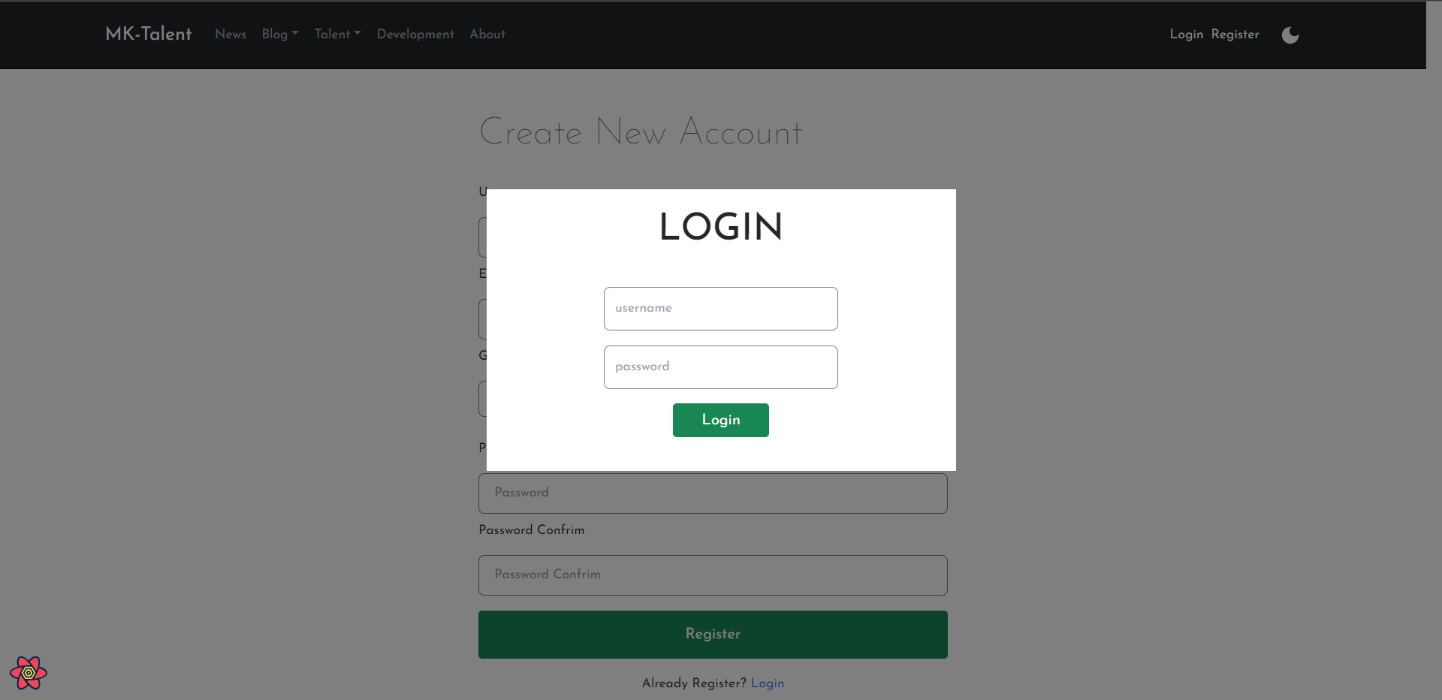


Figure 33 Login page

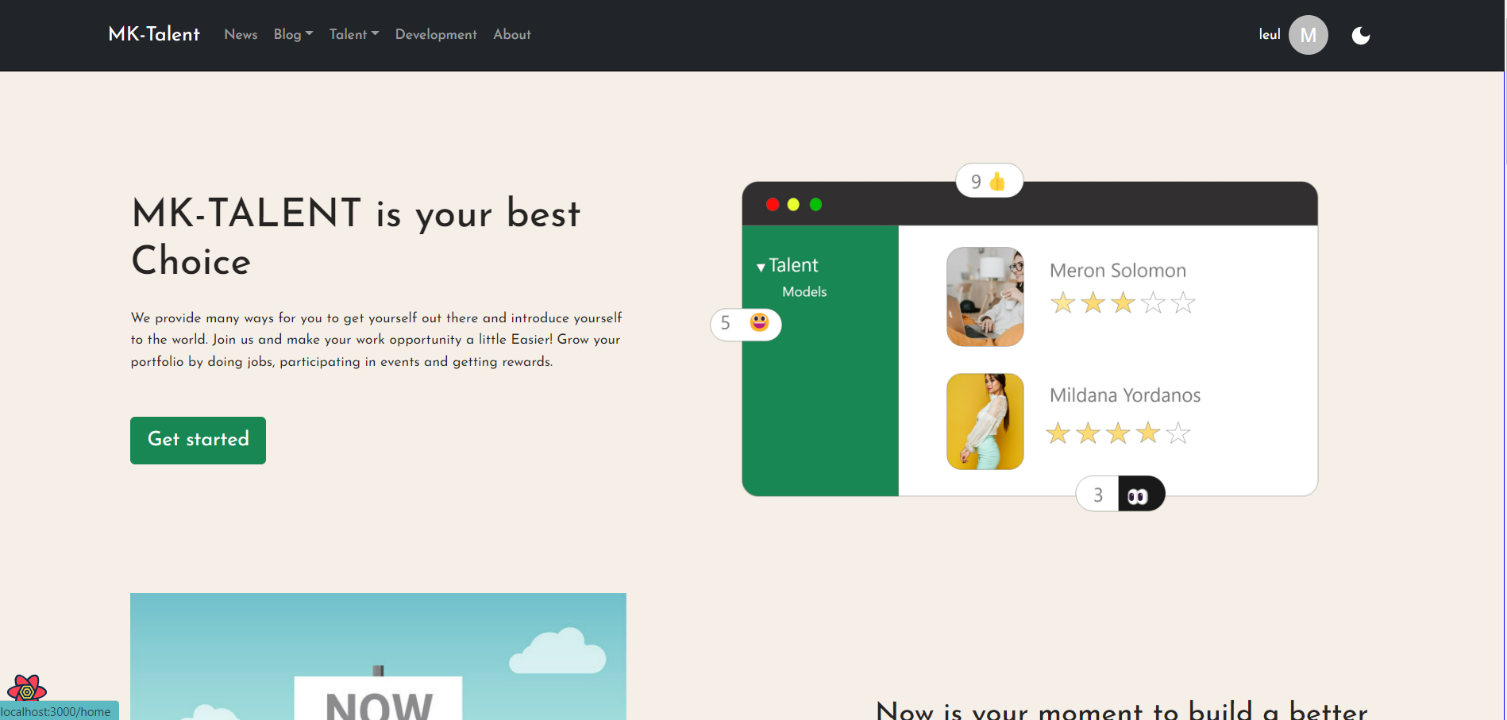


Figure 34 Home page

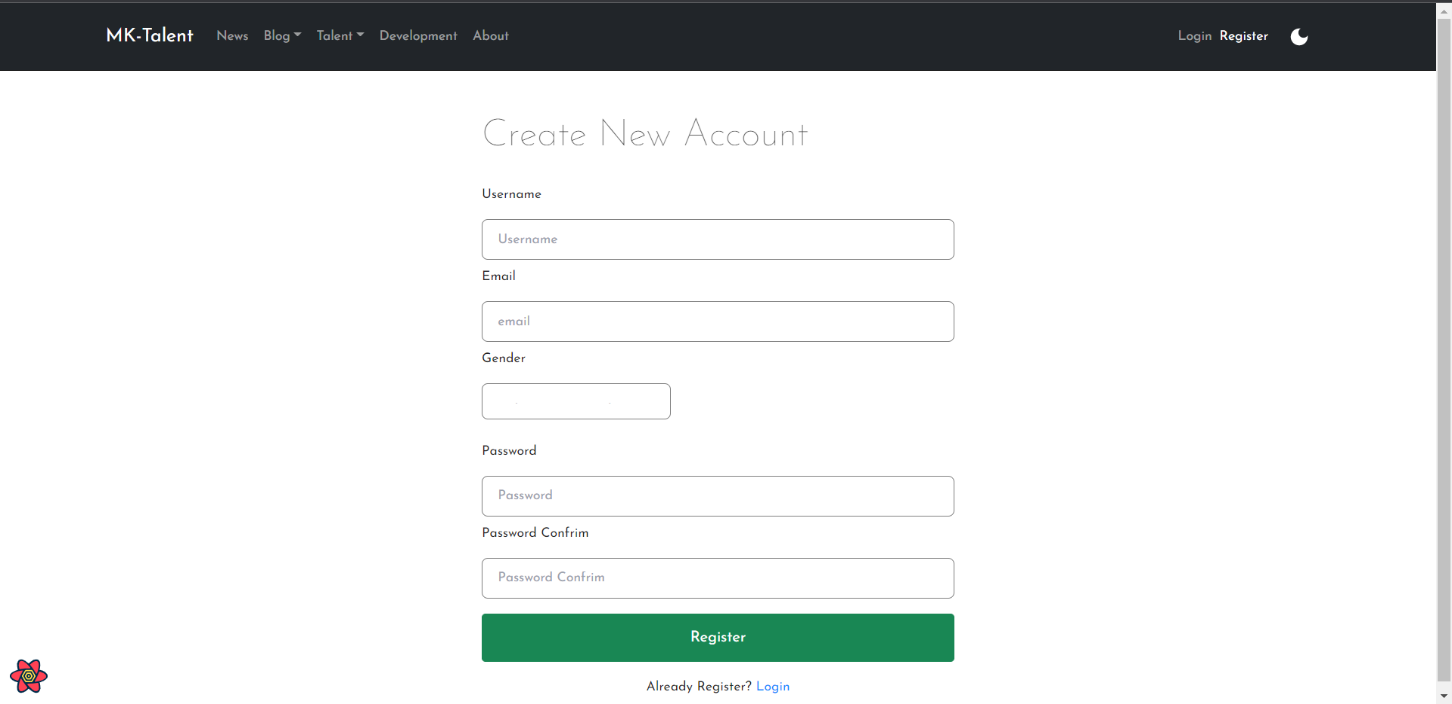


Figure 35 Registration page

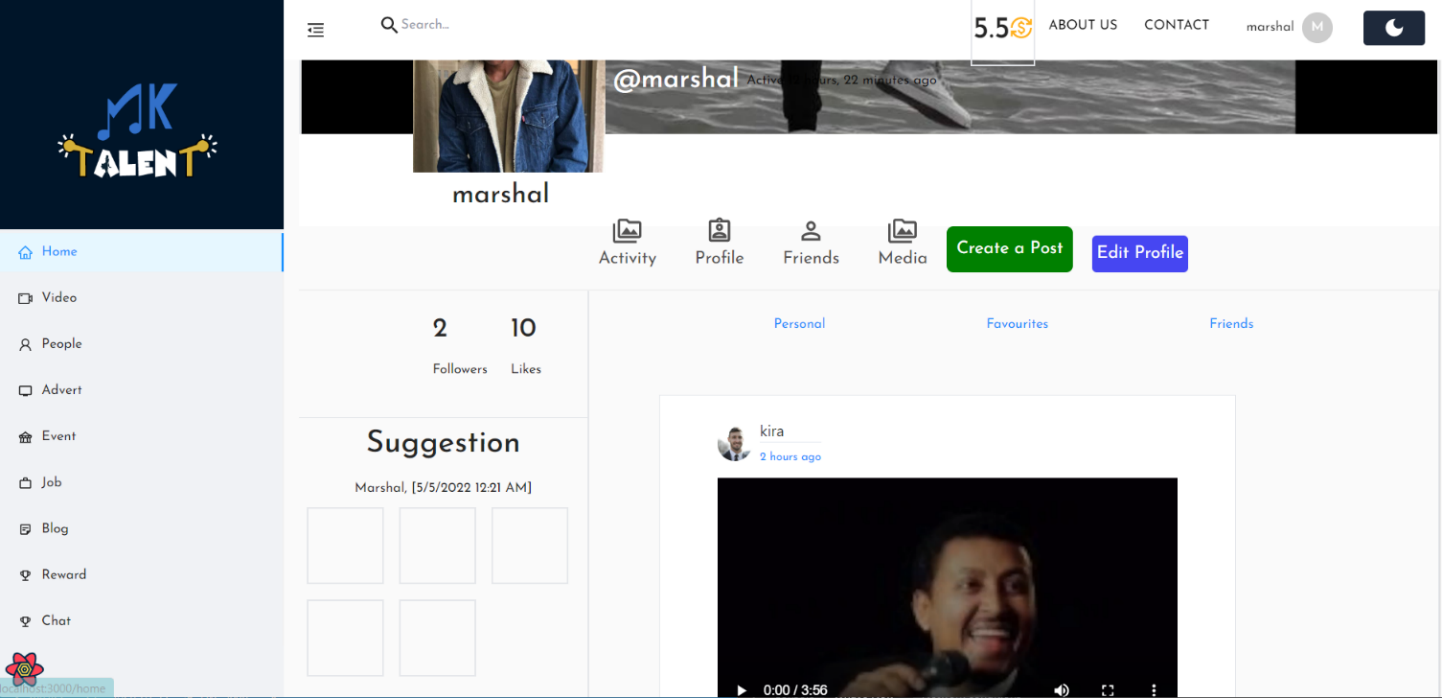


Figure 36 Profile page

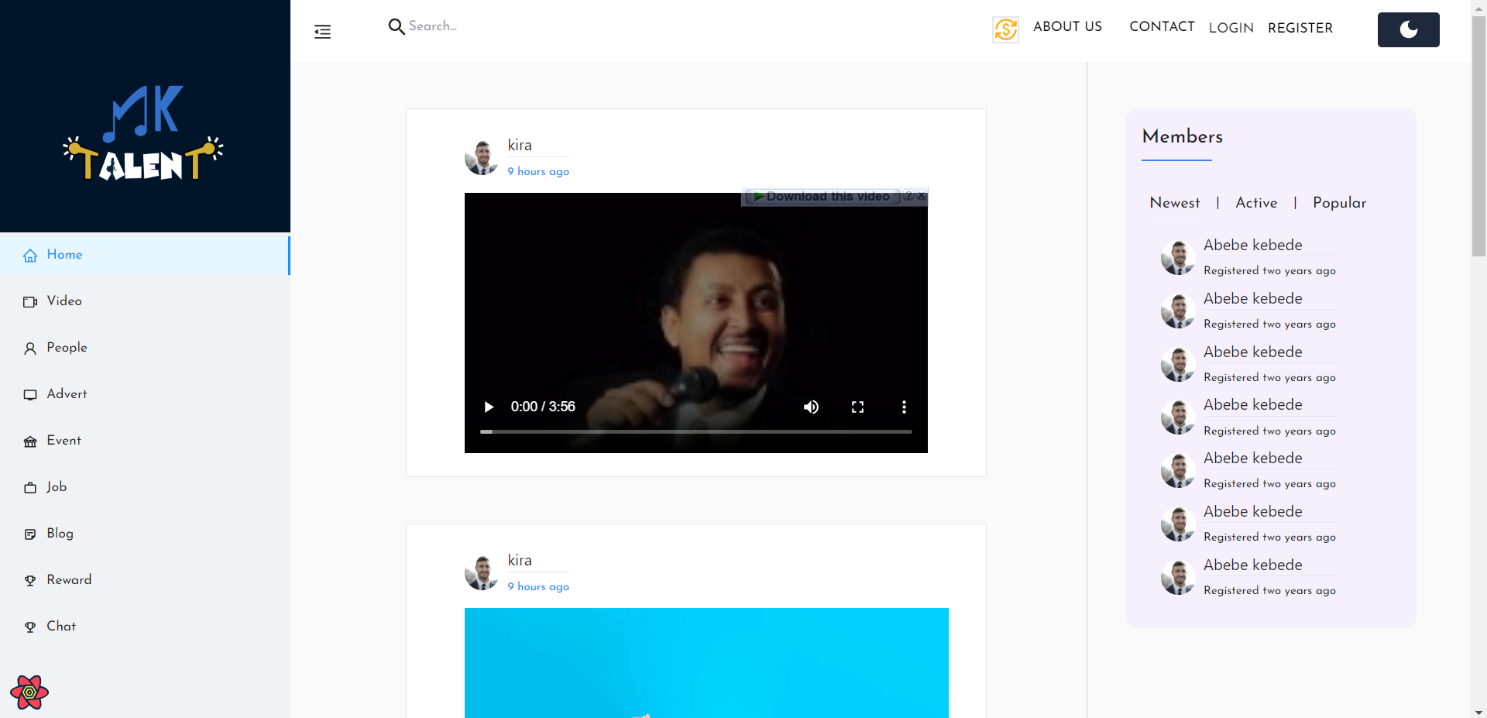


Figure 37 Main page (before login)

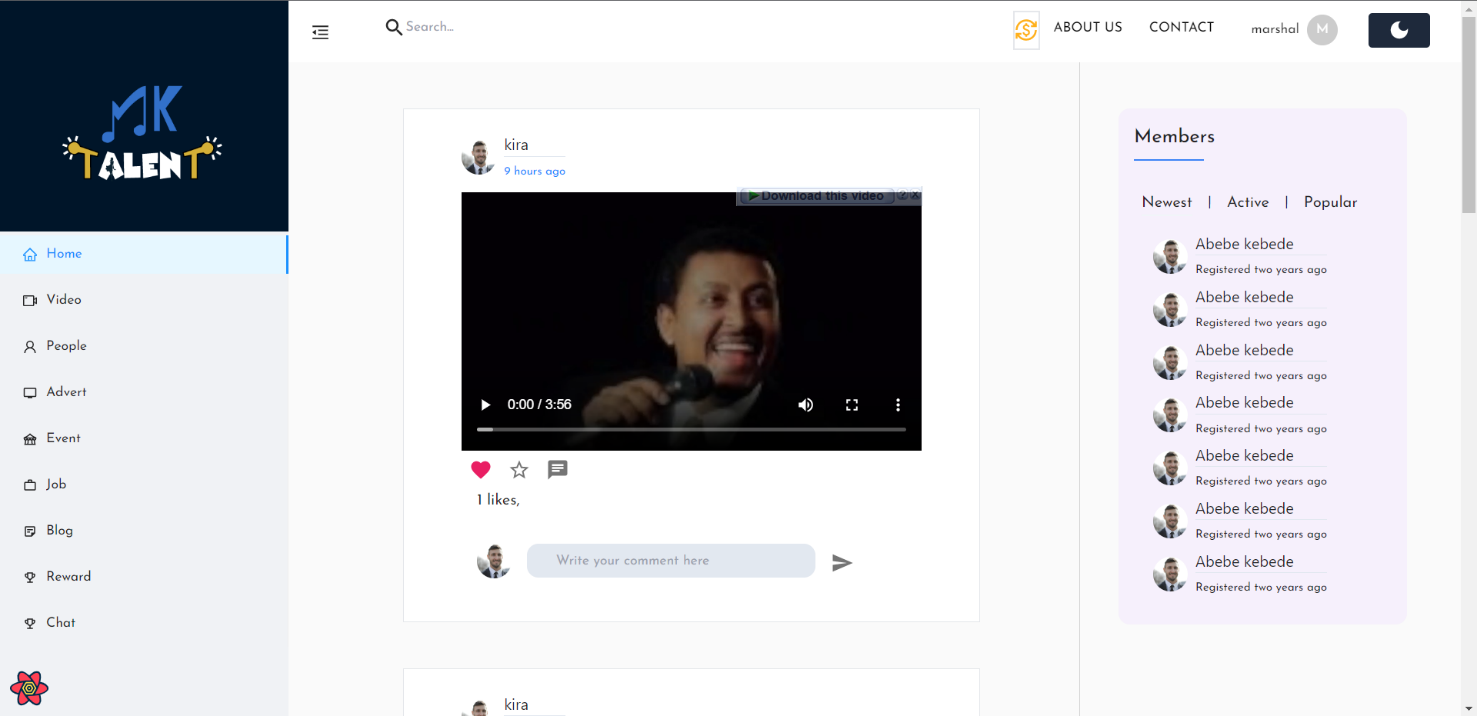


Figure 38 Main page (after login)

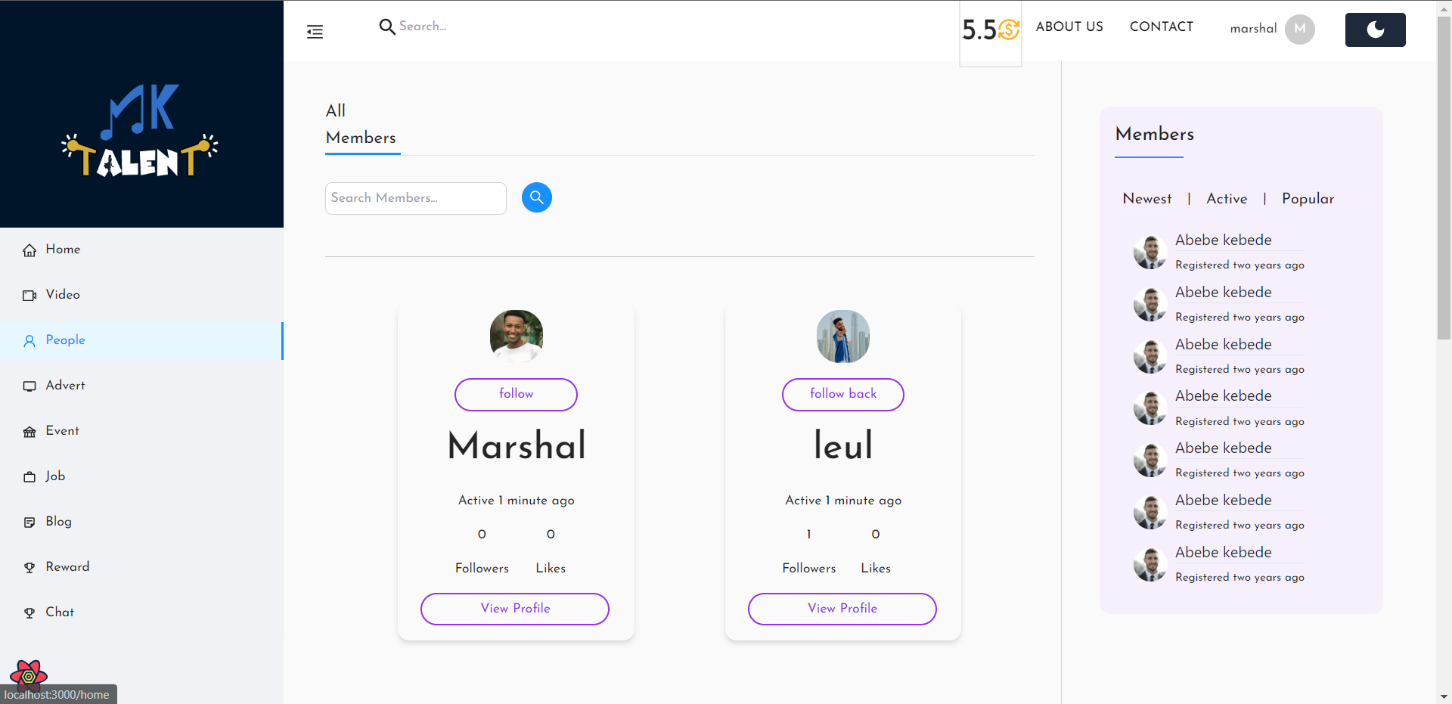


Figure 39 People page

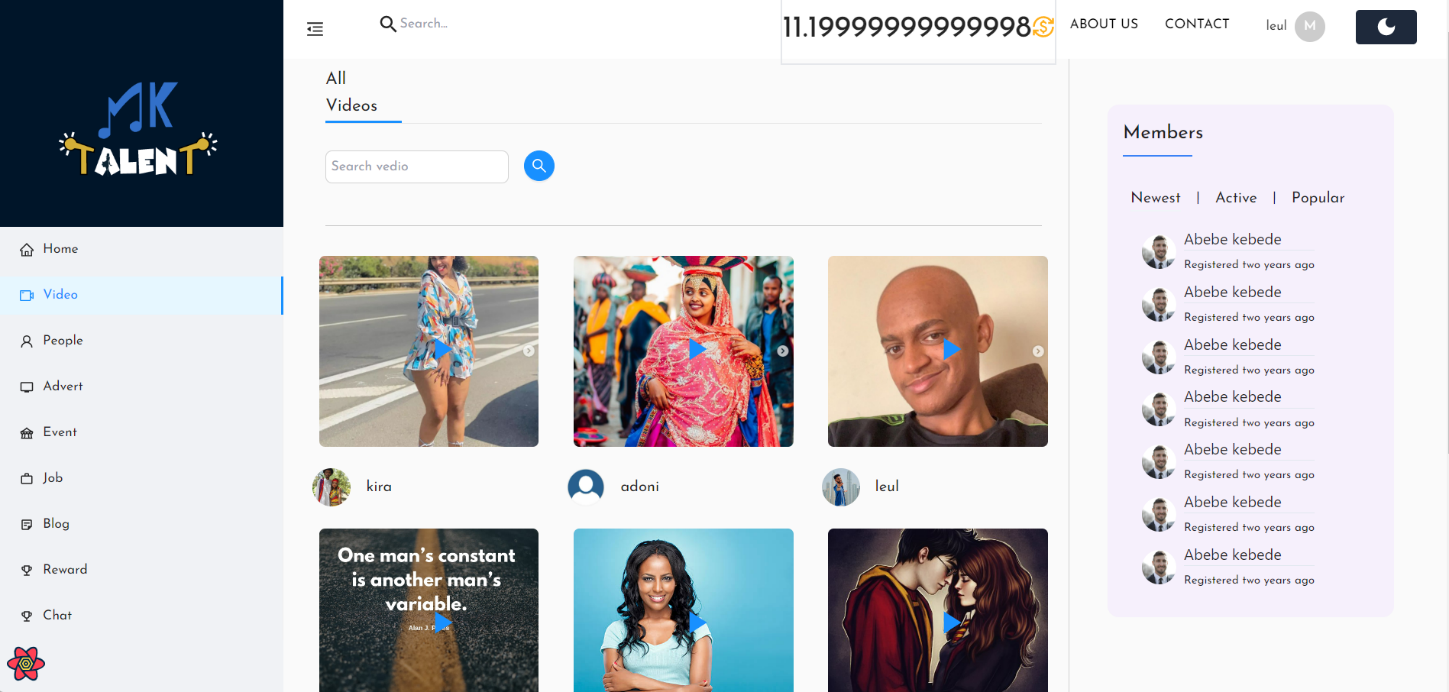


Figure 40 Video page

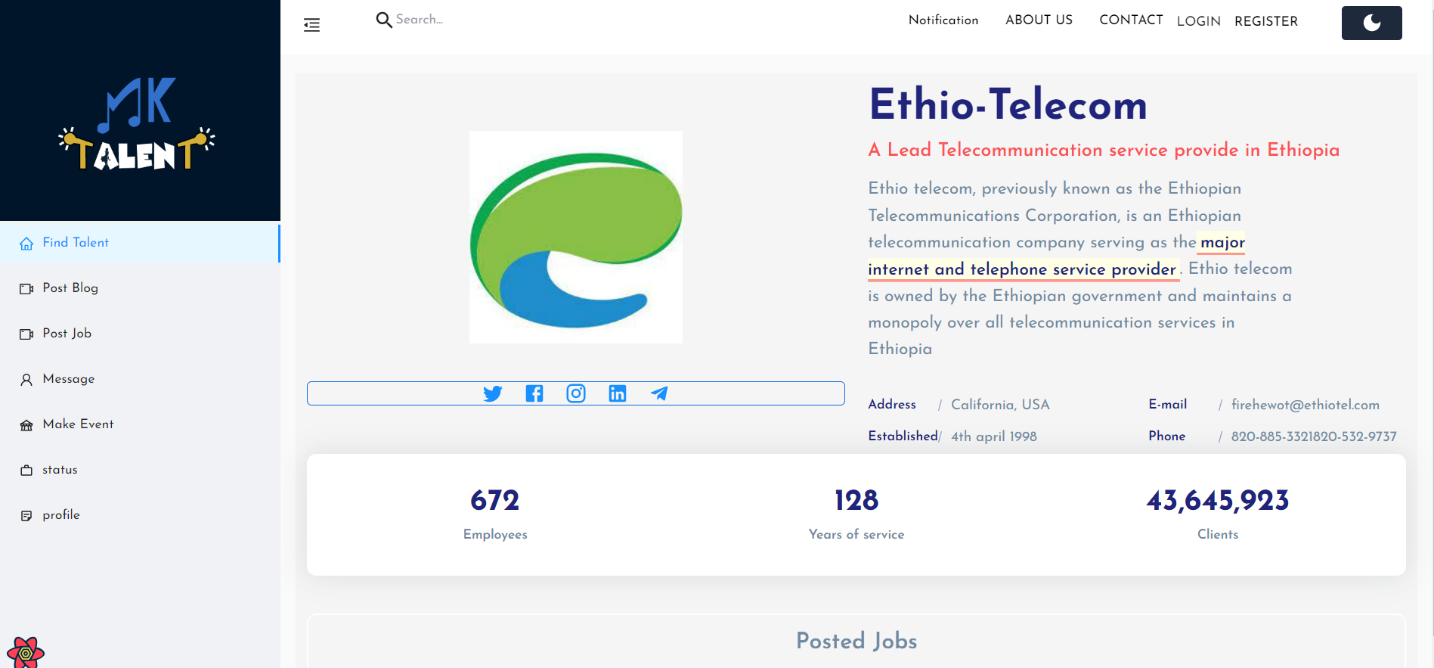


Figure Company page

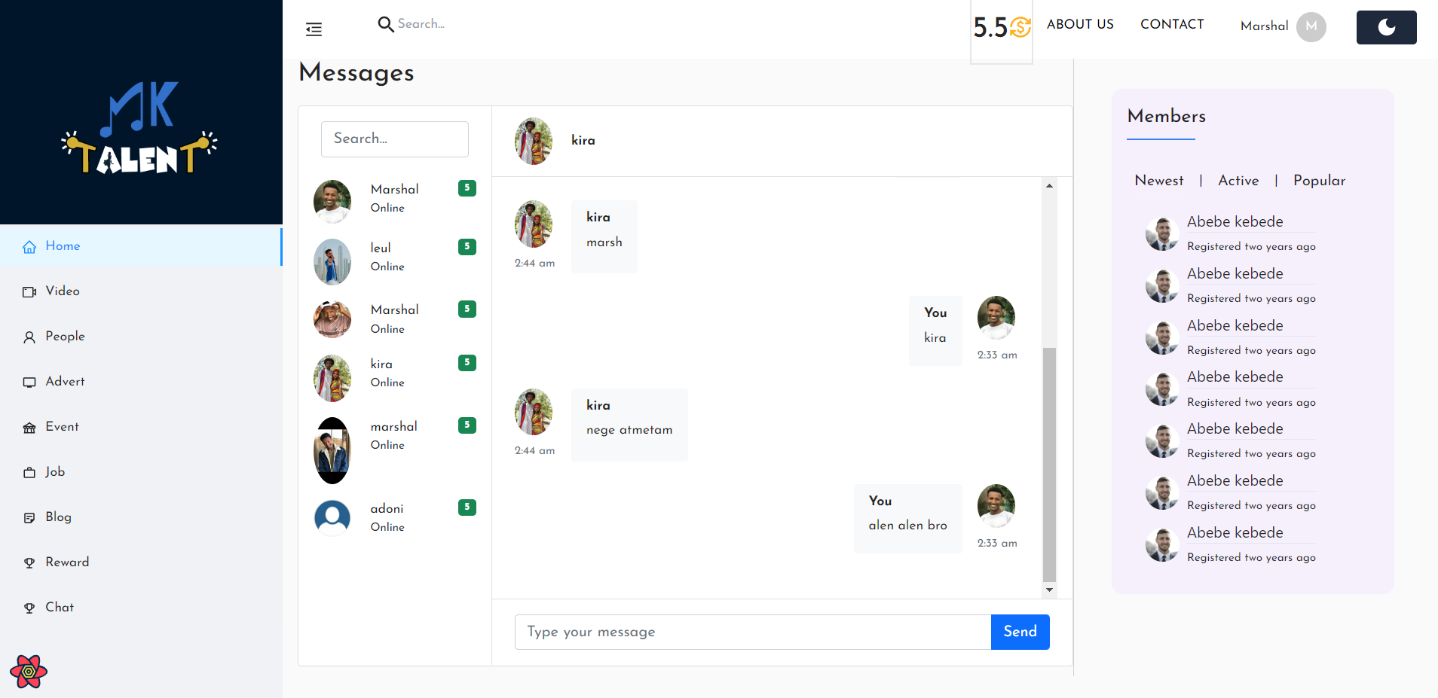


Figure Chat page

# 5. Chapter Five: Implementation and Testing

## 5.1 Implementation

This section will give a brief description about the different software development languages used

to implement the system. The section will justify the use of the tools and languages, and also the

source code of the main functions of the system.

### 5.1.1 Software tools

**VS Code** – free source code editor made for windows, mac and Linux systems.

**MongoDB Compass** – analyze the document and displays rich structures within your

collections through a GUI.

**Google Chrome** – freeware web browser that facilitates the software development process

by rapid debugging and on the fly style change.

**Postman-** is an application used for API testing. It is an HTTP client that tests HTTP requests, utilizing a graphical user interface, through which we obtain different types of responses that need to be subsequently validated.

### 5.1.2 Development languages

**Node JS** - is a JavaScript runtime environment built on the chrome’s V8 JavaScript engine.

**React JS** – is a free and open-source front-end JavaScript library for building user interfaces based on UI components.

**Redux** - an open-source JavaScript library for managing and centralizing application state.

**Mongo DB** – is a cross platform document-oriented database program. It uses a JSON like

documents with optional schemas.

**Express-** is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.[7]

### 5.1.3 Source Code

**<<Login>>**

*const getToken = (id) => {*

*//check your code aeound here you may have forgotten id:id and said only id*

*return jwt.sign({ id: id }, process.env.JWT\_SECRETE, {*

*expiresIn: "30d",*

*});*

*};*

exports.login = catchAsync(async (req, res, next) => {

const { email, password } = req.body;

// check the email and the password is exist

if (!email || !password) {

return next(new AppErorr("email and password must be provided!", 401));

}

// check if user is exist and oasword is coreact

const user = await User.findOne({ email: email }).select("+password");

// const correct = await user.correctPassword(password,user.password)

// console.log(user);

if (!user || !(await user.correctPassword(password, user.password))) {

return next(new AppErorr("Incorrect email or password", 401));

}

console.log(user);

//get the token

const token = getToken(user.\_id);

res.status(200).json({

status: "succes",

token: token,

data: {

role: user.role,

\_id: user.\_id,

username: user.username,

gender: user.gender,

createdAt: user.createdAt,

updatedAt: user.updatedAt,

},

});

});

**<<Company>>**

*//CREATE COMPANY*

exports.createCompany = catchAsync(async (req, res, next) => {

req.body.user = req.user.id;

const newCompany = await Company.create(req.body);

res.status(200).json({

success: "success",

data: newCompany

});

});

*exports.updateCompany = catchAsync(async (req, res) => {*

*const Company = await Company.findByIdAndUpdate({ \_id: req.params.id }, req.body, {*

*new: true,*

*runValidators: true,*

*});*

*if (!Company) {*

*return next(new AppErorr("There is no Company with this ID", 404));*

*}*

*res.status(200).json({*

*status: "success",*

*data: Company,*

*});*

*});*

*//DELETE Company*

*exports.deleteCompany = catchAsync(async (req, res) => {*

*const Company = await Company.findById({ \_id: req.params.id });*

*if (!(req.user.id == Company.user)) {*

*return next(new AppErorr("this is not your Company post", 404));*

*}*

*const CompanyDelete = await Company.findByIdAndDelete({ \_id: req.params.id });*

*if (!Company) {*

*return next(new AppErorr("There is no Company with this ID", 404));*

*}*

*res.status(200).json({*

*status: "success",*

*data: CompanyDelete,*

*});*

*});*

**<<Talent Posts>>**

//CREATE POST

exports.createPost = catchAsync(async (req, res, next) => {

req.body.user = req.user.id;

console.log(

"::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::"

);

console.log(req.files, req.body);

if (req.files.images) {

req.body.images = [];

req.files.images.map((file) => req.body.images.push(file.filename));

req.body.dataType = "image";

}

if (req.files.video) {

req.body.video = req.files.video[0].filename;

req.body.dataType = "video";

req.body.videoImage = req.files.videoImage[0].filename;

}

if (req.files.audio) {

req.body.audio = req.files.audio[0].filename;

req.body.dataType = "audio";

}

const post = await Post.create(req.body);

res.status(200).json({

status: "success",

data: post,

});

});

//GET ALL POSTS

exports.getAllPost = catchAsync(async (req, res, next) => {

// let query = Post.find();

// console.log(req.query);

// const page = req.query.page \* 1 || 1;

// const limit = +req.query.limit || 100;

// const skip = (page - 1) \* limit;

// query.skip(skip).limit(limit);

// query.sort("-createdAt");

const featur = new APIFeature(Post.find(), req.query)

.filter()

.sort()

.fields()

.paging();

const post = await featur.query

.populate("comments")

.populate("user")

.populate("likes");

const posts = await Post.find();

res.status(200).json({

totalLength: posts.length,

length: post.length,

status: "success",

data: post,

});

});

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* update post \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

exports.updatePost = catchAsync(async (req, res, next) => {

const post = await Post.findByIdAndUpdate({ \_id: req.params.id }, req.body, {

new: true,

runValidators: true,

});

if (!post) {

return next(new AppErorr("There is not post in this ID", 404));

}

const oneProfile = await Profile.findOne({ user: post.user });

const point = oneProfile.point + 0.1;

console.log(

point,

oneProfile

);

const profile = await Profile.findByIdAndUpdate(

{ \_id: oneProfile.\_id },

{ point: point },

{

new: true,

runValidators: true,

}

);

console.log("999999999999999999999999999999999999999999", profile);

console.log("999999999999999999999999999999999999999999");

next();

});

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* delete post \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

exports.deletePost = catchAsync(async (req, res, next) => {

const post = await Post.findById({ \_id: req.params.id });

if (!(req.user.id == post.user)) {

return next(new AppErorr("this is not your post", 404));

}

const postDelete = await Post.findByIdAndDelete({ \_id: req.params.id });

if (!post) {

return next(new AppErorr("There is not post in this ID", 404));

}

const comment = await Comment.deleteMany({ post: req.params.id });

if (post.images) {

const path = post.images;

try {

path.map((p) => fs.unlinkSync(`../public/assets/img/post/${p}`));

//file removed

} catch (err) {

console.error(err);

}

}

if (post.video) {

const path = post.vedio;

try {

fs.unlinkSync(`../public/assets/video/${path}`);

//file removed

} catch (err) {

console.error(err);

}

}

if (post.audio) {

const path = post.audio;

try {

fs.unlinkSync(`../public/assets/audio/${path}`);

//file removed

} catch (err) {

console.error(err);

}

}

res.status(200).json({

status: "success",

data: post,

});

});

<<**Register**>>

import React, { useState } from "react";

import { Button, Navbar } from "react-bootstrap";

import { useSelector, useDispatch } from "react-redux";

import { Link } from "react-router-dom";

import { ToastContainer, toast } from "react-toastify";

import "react-toastify/dist/ReactToastify.css";

import { register } from "../../../redux/authReducer";

import { useNavigate } from "react-router-dom";

// import ModelRegistrationCon from '../RegisterStyle'

// import './ModelRegistration.css'

import styled from "styled-components";

import api from "../../../api/api";

const ModelRegistration = (props) => {

const navigate = useNavigate();

const dispatch = useDispatch();

const userData = useSelector((state) => state.auth);

const light = useSelector((state) => state.mode.light);

const roleType = useSelector((state) => state.mode.role);

const [value, setValue] = useState({

username: "",

email: "",

gender: "",

password: "",

passwordConfirm: "",

});

const toastOption = {

position: "bottom-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

theme: light ? "light" : "dark",

// progress: undefined

};

const onChangeHandler = (e) => {

setValue({ ...value, [e.target.name]: e.target.value });

};

const handleValidation = async (e) => {

e.preventDefault();

const { email, username, gender, password, passwordConfirm } = value;

if (!username) {

toast.error("Please provide your username", toastOption);

} else if (!email) {

toast.error("Please provide your emal", toastOption);

} else if (!gender) {

toast.error("Please provide your gender", toastOption);

} else if (!password) {

toast.error("Please provide your password", toastOption);

} else if (password !== passwordConfirm) {

toast.error(

"Your password and confirm password is not match",

toastOption

);

} else {

try {

dispatch(register({ loading: true }));

const { data } = await api.post("/users/signup", value);

dispatch(register({ data: data, loading: false }));

console.log("pppppppp", data);

if (data.data.role == "talent") {

navigate("/main");

} else if (data.data.role == "admin") {

navigate("/admin");

}

} catch (err) {

dispatch(

register({

error: err.response && err.response.data.message,

loading: false,

})

);

}

}

};

return (

<>

{props.children}

<ModelRegistrationCon

style={{ backgroundColor: light ? "white" : "#3b3b3b" }}

>

{/\* <div className='image'>

<img src={SignupImg} alt="" />

</div> \*/}

<div className="step1">

<form className={light ? "" : "dark"}>

<h1>Create New Account</h1>

<label for="lname">Username</label>

<input

name="username"

onChange={(e) => onChangeHandler(e)}

value={value.username}

id="username"

type="text"

placeholder="Username"

/>

<label for="email">Email</label>

<input

name="email"

onChange={(e) => onChangeHandler(e)}

value={value.email}

id="email"

type="emial"

placeholder="email"

/>

<div

style={{

display: "flex",

justifyContent: "space-between",

width: "500px",

}}

>

<div className="form-group">

<label for="gender">Gender</label>

<select

className="form-control"

onChange={(e) => onChangeHandler(e)}

name="gender"

id="gender"

>

<option hidden>Select your gender</option>

<option value="male">Male</option>

<option value="female">Female</option>

</select>

</div>

</div>

<label for="password">Password</label>

<input

name="password"

onChange={(e) => onChangeHandler(e)}

value={value.password}

id="password"

type="Password"

placeholder="Password"

/>

<label for="passwordConfirm">Password Confrim</label>

<input

name="passwordConfirm"

onChange={(e) => onChangeHandler(e)}

value={value.passwordConfirm}

id="passwordConfirm"

type="Password"

placeholder="Password Confrim"

/>

<Button type="submit" onClick={handleValidation} variant="success">

Register

</Button>

</form>

<span style={{ textAlign: "center" }}>

Already Register? <Link to="/login">Login</Link>

</span>

</div>

</ModelRegistrationCon>

<ToastContainer />

</>

);

};

const ModelRegistrationCon = styled.div`

/\* background-color: black; \*/

width: 100%;

height: 100%;

display: flex;

justify-content: center;

/\* color: white; \*/

.image {

img {

z-index: 0;

position: fixed;

right: 4rem;

bottom: 2rem;

}

}

.step1 {

border-radius: 10px;

z-index: 0;

display: flex;

max-width: 700px;

flex-direction: column;

align-items: center;

/\* padding: 4rem; \*/

margin-top: 30px;

padding: 1rem;

.dark {

color: white;

input {

background-color: #3b3b3b;

color: white;

}

button {

background-color: #3b3b3b;

color: white;

}

}

.dark select {

background-color: #3b3b3b;

color: white;

}

form {

display: flex;

flex-direction: column;

align-items: flex-start;

justify-content: center;

h1 {

font-weight: 100;

}

label {

padding: 7px 0;

}

.profileImg {

margin: 15px 0;

width: 500px;

padding: 1rem;

border: 1px solid gray;

border-radius: 0.4rem;

}

input {

border: 1px solid gray;

padding: 0.6rem 1rem;

border-radius: 0.4rem;

outline: none;

width: 500px;

}

select {

border: 1px solid gray;

padding: 1rem 1rem;

border-radius: 0.4rem;

outline: none;

width: 200px;

}

div.row {

width: 500px;

display: flex;

flex-direction: row;

justify-content: space-between;

div {

display: flex;

flex-direction: column;

align-items: flex-start;

width: 210px;

}

input {

width: 210px;

}

}

button {

margin: 1rem 0;

width: 500px;

padding: 0.8rem;

}

}

}

`;

export default ModelRegistration;

## 5.2 Testing

Testing is the process of evaluating the software item to detect differences between given input and expected output. The testing assesses the quality of the software product.

### 5.2.1 Unit test

It is the testing of an individual unit or a group of related units of functionality. It is often done by the programmer to test whether the unit he/she has implemented is producing the expected output against the given input. The objective of unit testing is to:

* + - * Check whether the return type of the functions is correct.
      * Check the input data that we write on the GUI must be submitted to the database.
      * Check the GUI can access the privileged data from the database.
      * Check if the correct output is produced for different inputs.

As we planned to test this project, an individual program module or component has been tested to ensure that it works according to its predetermined specifications, operates within acceptable parameters, and meets the appropriate standards. We figured out how the functions were tested during the implementation as sample below in the test cases table.

### 5.2.2 Integration Testing

This system contains different individually tested modules and can work briefly after a combination of modules as one unit.

### 5.2.3 System testing

It is a way that looks at how the various subsystems work together to accomplish the intended result. Its objectives are to assess whether a subsystem's functionality is correct after it has been combined with other subsystems and to check whether the overall functionality of the system meets the user's requirements. System testing ensures that software works in a variety of situations while the operating system remains functional.

Table Use cases and Test Cases

|  |  |  |
| --- | --- | --- |
| Use Case ID | Description | Test Case ID |
| UC\_1 | Login | TC\_1 |
| UC\_2 | Register companies | TC\_2 |
| UC\_3 | Create new account | TC\_3 |
| UC\_4 | Post a blog | TC\_4 |
| UC\_5 | Give feedback | TC\_5 |
| UC\_6 | View jobs | TC\_6 |
| UC\_7 | View posts | TC\_7 |
| UC\_8 | View event | TC\_8 |
| UC\_9 | View reward | TC\_9 |
| UC\_10 | View Blog | TC\_10 |
| UC\_11 | Edit personal profile | TC\_11 |
| UC\_12 | Search for jobs | TC\_12 |
| UC\_13 | Search for posts | TC\_13 |
| UC\_14 | Post media | TC\_14 |
| UC\_15 | View profile | TC\_15 |
| UC\_16 | Generate report | TC\_16 |
| UC\_17 | Give approval | TC\_17 |
| UC\_18 | Respond to feedback | TC\_18 |
| UC\_ 19 | Post comment | TC\_ 19 |
| UC\_20 | Post job | TC\_20 |
| UC\_21 | Apply job | TC\_21 |
| UC\_22 | Prepare event | TC\_22 |
| UC\_23 | Edit blog | TC\_23 |
| UC\_24 | Logout | TC\_24 |
| UC\_25 | Delete blog | TC\_25 |
| UC\_26 | Delete post | TC\_26 |
| UC\_27 | Delete job | TC\_27 |
| UC\_28 | Claim reward | TC\_28 |
| UC\_29 | Send message | TC\_ 29 |
| UC\_30 | Search users | TC\_30 |
| UC\_31 | Delete user | TC\_31 |
| UC\_32 | Report post | TC\_32 |
| UC\_33 | Like post | TC\_33 |
| UC\_34 | Dislike post | TC\_34 |
| UC\_35 | Follow user | TC\_35 |
| UC\_36 | Unfollow user | TC\_36 |
| UC\_37 | Prepare reward | TC\_37 |

Table Description and the Expected Results of each test case

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE ID | TEST CASE ID | DESCRIPTION | EXPECTED RESULTS |
| UC-1 | TC-1 | User enters a valid email and password. | User is granted the access to the system. |
| UC-2 | TC-2 | The admin approves the company registration. | The company is verified and can post jobs. |
| UC-3 | TC-3 | The User create new account (registration). | Confirmation message appear verifying the registration. |
| UC-4 | TC-4 | The company posts blogs. | The posted blogs will be shown. |
| UC-5 | TC-5 | The user give feedback about the system to the admin. | The admin will receive the feedbacks. |
| UC-6 | TC-6 | The user sees the posted job and available information. | posted jobs will be shown. |
| UC-7 | TC-7 | The user sees the posted posts. | Posted posts will be shown. |
| UC-8 | TC-8 | The user sees the posted event. | Posted events will be shown. |
| UC-9 | TC-9 | The user sees how much reward they have. | Total amount of rewards will be shown. |
| UC-10 | TC-10 | The user sees the posted blog. | Posted blog will be shown. |
| UC-11 | TC-11 | The user modifies their personal profiles. | The edited profile will be updated and stored in the database. |
| UC-12 | TC-12 | The user searches for jobs. | Results will be filtered based on the search parameter. |
| UC-13 | TC-13 | The user searches for posts. | Results will be filtered on the search parameter. |
| UC-14 | TC-14 | The user post a media file on their timeline. | The posted media will be shown. |
| UC-15 | TC-15 | The user views other users profile. | Other users profile information will be shown. |
| UC-16 | TC-16 | The admin asks the system to generate a report about the activities. | Report will be generated. |
| UC-17 | TC-17 | The company approves the users application on the job. | Approval message will be sent to the right user. |
| UC-18 | TC-18 | The admin responds to the feedback sent by the user | Response will be sent to the user. |
| UC-19 | TC-19 | The user will comment on the posts. | The comment will be shown under the post. |
| UC-20 | TC-20 | The company will post a job for the users to apply. | The job will be available for users. |
| UC-21 | TC-21 | The users will apply for the job. | The job application will be sent to the company. |
| UC-22 | TC-22 | The company prepares an event post. | The event will be shown in the event category. |
| UC-23 | TC-23 | The company edit information on the blog. | The updated information of the blog will be shown |
| UC-24 | TC-24 | The user logs out of the system | The user safely exits the system. |
| UC-25 | TC-25 | The company deletes the entire blog. | The blog will no longer appear on the blog page. |
| UC-26 | TC-26 | The user deletes a post | The post won’t be available on the user’s timeline |
| UC-27 | TC-27 | The company removes a posted job. | The job will be removed from the jobs page. |
| UC-28 | TC-28 | The user claims reward based on their total points achieved. | The user receives a message to prove their identity to collect the reward. |
| UC-29 | TC-29 | The user sends and receives a message between other users | The message will be sent to the right user. |
| UC-30 | TC-30 | The user search for a specific user based on their username. | The user with the correct username will be shown. |
| UC-31 | TC-31 | The user deletes their account and all their information in the system. | The account and all its information will be removed from the system |
| UC-32 | TC-32 | The user reports inappropriate post to the admin. | The admin will receive the report. |
| UC-33 | TC-33 | The user likes a post posted by another user. | The total number post like will increase by one. |
| UC-34 | TC-34 | The user dislikes a post posted by another user. | The total number post like will decrease by one. |
| UC-35 | TC-35 | The user follows another user. | The followed user will be shown in followed users list |
| UC-36 | TC-36 | The user unfollows other user. | The followed user will be removed from followed users list. |
| UC-37 | TC-37 | The admin prepares rewards and assign a point to them. | The prepared rewards will be available in the reward section. |

# 6. Chapter Six: Conclusion and future work

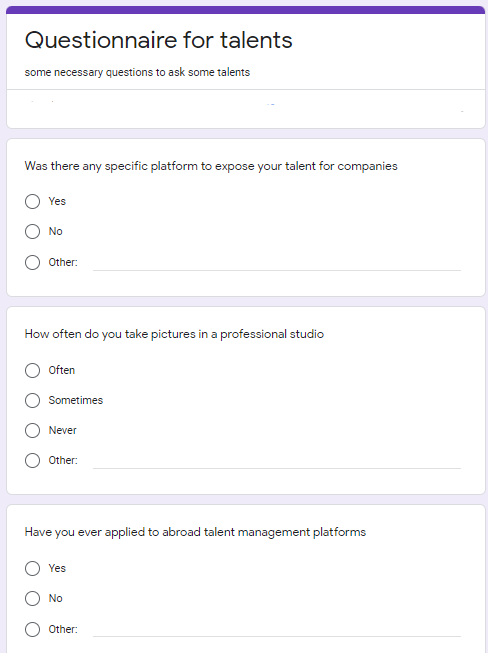
In summary, we strongly believe that this system will help the Model sector of Ethiopia facilitate its activities. There will also be so many people and related actors that will be benefited from this system.

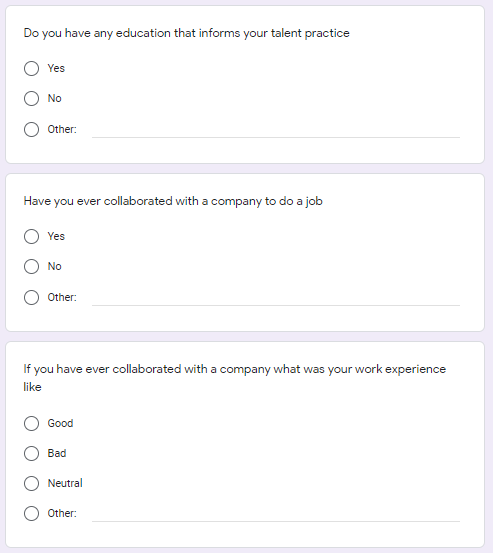
We also believe that it might be a good motivation for those who want to introduce themselves to the world, get more job opportunities with the talent they possess and facilitate a more comfortable workplace. It has been quite pleasurable building it.

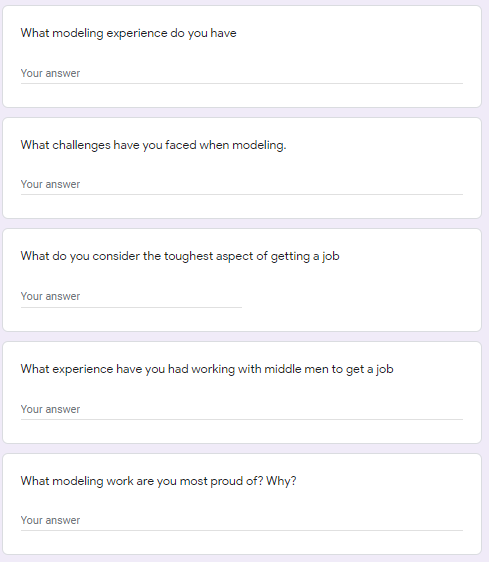
In the future, we are planning to extend the project in which more specialized actors can interact with it. We are also planning to fix all the bugs, during use if any. We will make it more efficient by modifying some of its parts, by changing the technologies we used, by updating the versions of the technologies already used and so on when deemed necessary.

### Appendix

## Questionnaire







# References

[1] “SAP INSIGHTS,” [Online]. Available: [https://insights.sap.com](https://insights.sap.com/). [Accessed November 5 2021]

[2] Erik van Vulpen, “Academy to innovate HR”, Feb 12,2019. [online]. Available: <https://WWW.aihr.com>. [Accessed 7 November 2021].

[3] “Activity\_Diagram,” [Online]. Available: <https://en.wikipedia.org/wiki/Activity_diagram>. [Accessed feburary 2 2022]

[4] “Science direct, Strategic talent managent” [Online]. Available : [https://www.sciencedirect.com](https://www.sciencedirect.com/) [Accessed 7 Feburary 2022 ]

[5] Peter wayner,” Architectural patterns”. Available: <https://techbeacon.com/app-dev-testing/top-5-software-architecture-patterns-how-make-right-choice> [Accessed 10 Febuy 2022 ]

[6] Miller juma,” Node js””. Available: <https://www.section.io/engineering-education/guide-to-nodejs-event-emitter/#:~:text=Node.js-,Node.,an%20asynchronous%20event%2Ddriven%20architecture>. [Accessed 10 January 2022]

[7] “Express” Available: <https://expressjs.com/> . [Accessed 29 May 2022]