

Voice Controlled Application For Finance (Finansis)

Course: Final Project

College: College of Computer and Artificial Intelligence

Class: Software Engineering

Full name: Mohamed Mohamud Mohamed Nur

Instructor: Wingo

College of Computer Science and Artificial Intelligence

November 2021

ABSTRACT

This project aims to develop a voice-controlled application for finance, a voice assistant for finance. this application is the same as Siri, (Siri is a virtual assistant that is part of Apple Inc.'s iOS) but Finansis is more focused on finance more specific the stock market. Finansis develop for traders or normal people that interested in the stock market. The system is a web application that can be accessed through the internet. This system can be used for knowing the stock market news (real-time news), knowing current a stock price (real-time price), knowing stock information, showing a stock chart, asking questions related to the stock market, the ability for finansis to 'learn' about the stock market, back-testing a trading strategy, and the most interesting is using Algorithmic Trading to trade stocks. The project has been planned to be having the view of cloud database. Using No-SQL database Mongo DB, Server and all the user interfaces have been designed using the MERN stack technologies (MongoDB, Express, React, Node) and python.

**Keywords:** Finance, Stock Market, Algorithmic Trading, python, MERN STACK; NO-SQL, Mongo database, scraping, automation.

TABLE OF CONTENTS

[LIST OF TABLES i](#_Toc9087)

[LIST OF FIGURES ii](#_Toc21107)

[1 INTRODUCTION 1](#_Toc14623)

[5.1.1 Future Goals For The System overview 1](#_Toc6374)

[5.1.2 Drawback of Existing system 1](#_Toc9918)

[2 SYSTEM PROBLEMS AND REQUIREMENTS ANALYSIS 2](#_Toc8991)

[2.1 Functional requirement 2](#_Toc29677)

[2.2 Non-Functional requirement [2] 5](#_Toc18593)

[2.3 Data flow graph: [3] 6](#_Toc27233)

[2.3.1 Data flow graph division tasks 6](#_Toc1528)

[2.3.2 The first data flow graph: 6](#_Toc20317)

[2.3.3 The second data flow graph: 7](#_Toc675)

[2.3.4 The third data flow graph: 7](#_Toc20125)

[3 SYSTEM DESIGN 9](#_Toc15552)

[3.1 System structure diagram 10](#_Toc12753)

[3.2 Module description: 11](#_Toc13085)

[3.2.1 Module description division tasks 11](#_Toc28909)

[3.3 Data Analysis 15](#_Toc2261)

[3.4 Database Design 15](#_Toc18886)

[3.4.1 Database Design division tasks 15](#_Toc14054)

[3.5 Algorithmic Design [5] 20](#_Toc16861)

[3.5.1 Algorithmic design division tasks 20](#_Toc17249)

[4 TEST AND ANAYLSIS REPORT 31](#_Toc19275)

[4.1 Test division tasks: 31](#_Toc29470)

[4.1.1 Applicant registration- Recruiter Registration - User login (Done By batis) 31](#_Toc19999)

[4.1.2 User add job – user update job – viewing applications 31](#_Toc21025)

[4.2 Applicant Registration Test [6] 31](#_Toc8632)

[4.3 Recruiter Registration Test 32](#_Toc4113)

[4.4 User login Test 33](#_Toc7644)

[4.5 user adding job 34](#_Toc22891)

[4.6 user update job info 35](#_Toc27036)

[4.7 viewing Applications 36](#_Toc23389)

[5 USER MANUAL 37](#_Toc7752)

[5.1 System Operating Environment Requirements. 37](#_Toc20335)

[5.2 System Development Environment Description 37](#_Toc18167)

[5.2.1 Programming language: 37](#_Toc5464)

[5.2.2 Frontend: 38](#_Toc18809)

[5.2.3 Backend 38](#_Toc21693)

[5.3 System functions (Recruiter module) 40](#_Toc27562)

[5.3.1 Add jobs 40](#_Toc11692)

[5.3.2 Jobs Applications management 40](#_Toc121)

[5.3.3 Employees management 42](#_Toc30215)

[5.3.4 Chatting 42](#_Toc25131)

[5.4 Group photo 43](#_Toc13124)

[5.5 Detailed description of division of labor 43](#_Toc19354)

[6 Conclusion 45](#_Toc15144)

[6.1 Need for new System 45](#_Toc28797)

[6.2 Summary of gains and losses 45](#_Toc31596)

[6.2.1 Gains 45](#_Toc29961)

[6.2.2 Loses 45](#_Toc10086)

[REFERENCES 46](#_Toc10080)

LIST OF TABLES

[Table 1 Register 11](#_Toc16217)

[Table 2 Log in 12](#_Toc13054)

[Table 3 log out 12](#_Toc22483)

[Table 4 update user info 12](#_Toc3586)

[Table 5 search job 12](#_Toc8345)

[Table 6 Apply for Job 13](#_Toc2216)

[Table 7 Message Recruiter 13](#_Toc10896)

[Table 8 Rate a job 13](#_Toc26262)

[Table 9 Job management 13](#_Toc24369)

[Table 10 View Applicant Request 14](#_Toc10929)

[Table 11 Message Applicant 14](#_Toc4911)

[Table 12 Change applicant 14](#_Toc24291)

[Table 13 Employee management 14](#_Toc11353)

[Table 14 User model 16](#_Toc2332)

[Table 15 applicant model 17](#_Toc24259)

[Table 16 Recruiter model 17](#_Toc29078)

[Table 17 Job model 19](#_Toc17696)

[Table 18 Rating model 20](#_Toc7774)

[Table 19 application model 20](#_Toc22643)

[Table 20 Applicant Registration 32](#_Toc10256)

[Table 21 Recruiter Registration 33](#_Toc2791)

[Table 22 User login 34](#_Toc19425)

[Table 23 adding job 35](#_Toc2554)

[Table 24 update job info 35](#_Toc1713)

[Table 25 viewing Applications 36](#_Toc25780)

[Table 26 System Operating Environment Requirements. 37](#_Toc11240)

[Table 27 Function1-division 44](#_Toc26459)

[Table 28 Function2-division 44](#_Toc19020)

[Table 29 Function3-division 44](#_Toc4209)

LIST OF FIGURES

[2.3.1.1 Figure 1 DFD-3-1 done by batis 6](#_Toc2685)

[2.3.1.2 Figure 2 DFD-3-2 done by benri 6](#_Toc28649)

[2.3.1.3 Figure 3 DFD-3-3 done by nur 6](#_Toc27511)

[Figure 4 DFD-1 6](#_Toc18072)

[Figure 5 DFD-2FD-2 7](#_Toc28095)

[Figure 6 DFD-3-1 7](#_Toc25449)

[Figure 7 DFD-3-2 8](#_Toc9393)

[Figure 8 DFD-3-3 8](#_Toc21307)

[Figure 9 System structure diagram 10](#_Toc22453)

[Figure 10 ER-DIAGRAM 15](#_Toc2382)

[Figure 11 Insert Job 21](#_Toc2498)

[Figure 12 Delete Job 23](#_Toc17136)

[Figure 13 Get all jobs 24](#_Toc6689)

[Figure 14 Register 25](#_Toc20058)

[Figure 15 Log in 26](#_Toc25204)

[Figure 16 Update user 27](#_Toc14549)

[Figure 17 Search Job 28](#_Toc30198)

[Figure 18 Apply for Job 28](#_Toc10273)

[Figure 19 Rating a job 30](#_Toc26116)

[Figure 20 add jobs 40](#_Toc25366)

[Figure 21 jobs applications 41](#_Toc12253)

[Figure 22 applications status 41](#_Toc19852)

[Figure 23 applications status 41](#_Toc14246)

[Figure 24 employees management 42](#_Toc23417)

[Figure 25 chatting 42](#_Toc29873)

# INTRODUCTION

The main problem that Finansis application tries to solve is to help people to have financial freedom, to create a passive income for them and lastly to make the developers of this application extremely rich. there are 8 main modules in this voice-controlled application User management, Stock Market News, Stock information, controlling the app with a voice, making Finansis response, asking questions about the stock market, back-testing stock trading strategy, and automating stock trading. there are 59 commands that Finansis understand plus stock related questions (for example 'what is market capitalization) and Finansis can ask more than 14 yes/no questions and then respond and do something based on a user's answer and lastly, Finansis can ask a user to choose stock from a list of stocks. Finansis has the ability to 'learn' about related Stock market questions and Stocks' symbol, actually Finansis just search in DuckduckGo (search engine like google) and then save the piece of information in the database. two mini-projects help Finansis news API, and Trading API, Trading API is a private project, news API is an API to save Scraped news in the database and let Finansis retrieve news from the database and news API is online (api URL:<https://news-api-lovat.vercel.app/api/v1/news).> there are four python scraping scripts running for 24 hours online, scraping news from yahoo finance, google finance, investing and seeking websites, and all that for free.

(scripts url: https://replit.com/@mHmdnwr/Chromium-Advanced-5ad#main.py)

## 5.1.1 Future Goals For The System overview

* Improving UI design.
* Trading stocks with real money.
* Making the System online.
* Creating our own voice Recognition AI.
* Adding more commands to control the System.
* Adding Smart Contracts

## 5.1.2 Drawback of Existing system

* Existing system is failing in providing high security
* The website design is not good
* Trading stock with fake money.
* Finansis is offline (only run in developer localhost)
* Can only work in Chrome Browser

# SYSTEM PROBLEMS AND REQUIREMENTS ANALYSIS

## Functional requirement

The modules involved are:

* + - User Management
    - Stock Market News
    - Stock Information
    - controlling the app with a voice
    - making Finansis response
    - asking questions about the stock market
    - back-testing stock trading strategy
    - automating stock trading

**The User management** module contains all functionality related to users. a user can register to the app, can log in, can update his/her information, can update his/her password, can add and remove stocks from his/her watch list, and if a user forgets his/her password a user can create a new password by receiving a reset token in his/her email. the main function responsible for this module is useUserCommandsHandler in the source code

**The Stock Market News** module contains all functionality related to the news. getting news by source, keyword in the headline, latest or top stories. Get more news. scraping news from yahoo finance, google finance, investing, and seeking websites. saving scraped news in the database. letting Finansis start reading news headlines from the first article, start reading them from specific article's number, or stop reading them. opening article's details page with help of Finansis or without, if a window opened with Finansis’s help, Finansis can scroll the window. the main function is useNewsCommandsHandler.

**Stock information** module contains all functionality related to stock information. opening stock chart using yahoo finance's charts with Finansis's help or without, if the chart is opened with Finansis's help, Finansis can change the chart's typed daily, 1 minute, or whatever valid type, and Finansis can also zoom the chart in or out. opening multiple charts at once only without Finansis's help. Get the stock's current price (real-time price) for free using the following API (URL: https://yahoo-finance-api.vercel.app/AAPL). helping finansis to learn about companies. showing users the most active, gainers, or losers stocks by opening yahoo finance stocks screener. showing users a stock statistics by opening yahoo finance stock statistics section. showing users sold stocks by opening yahoo finance stock chart and then changing the date range to equal date of buying and selling of the sold stock.

**Controlling the app with a voice** module is to control the app by telling Finansis commands. navigating between pages by telling finansis 'go forward', 'go back' or 'go to {news|info|...} page' ({news|info|..} is dynamic values, for example: 'go to news page'). zooming in or out charts, scrolling news page, scrolling details news page. closing any popup window.

**Making Finansis response** module is to make Finansis speak out loud. responding to users, asking users yes/no questions, or asking users to choose a stock from a stock list. stopping Finansis from recognizing when Finansis start speaking out loud.

**Back-testing stock trading strategy** module is to let users test stock trading strategy of this application. users will input start date, end date, initial cash, and account risk per trade. helping developers of this app to come up with a better trading strategy. I will talk about back-testing later in this thesis.

**Asking questions about the stock market** module is to let users ask questions about the stock market and then Finansis answers the question, if the answer of the question not exist in the database, then Finansis will search in the DuckduckGo (search engine like Google) and after that save the answer in the database and lastly answer the question so if someone asked the same question again Finansis just answer it.

**Automating stock trading (Trading Bot)** module is to let Finansis buy, sell and set stop loss for stocks using Algorithmic Trading. the trading flow first set stop loss for bought stocks (I will talk about stop loss in the Algorithmic Trading section), second find sell signals for bought stocks and then sell them if a signal has been found, third scrap S&P 500 stocks (I will also talk about S&P 500 stocks in this thesis), and then save them in python array, fourth loop through all S&P 500 stocks to find buy signals then save them in the database if a signal has been found, lastly, get found buy signal stocks from the database and then buy them. in this project for current version 1.0 all buy and sell stocks in a simulator for the stock market I used Investopedia's simulator (URL: https://www.investopedia.com/simulator).

## major problems occurred in this project [2]

* **Let Finansis recognizes English words:** solved by using react-speech-recognition npm package which uses Web Speech API.
* **Let Finansis speaks out loud:** solved by copying react-speech-kit npm package's 'useSpeechSynthesis' function and then adding more logic to the function. useSpeechSynthesis function uses Web Speech API too.
* **Run scraping news script for 24 hours**: solved by using python's webbot and Beautiful Soup packages to write scraping news scripts and Replit online IDE to run the scrips for 24 hours and also using python's request package to call news API's 'save articles' endpoint.
* **Let Finansis recognizes unknown commands**: solved by creating some logic to check for the user's command against existing commands, you can find the logic in the source code appUtils file under 'code for handling unknown commands' comment.
* **Deploy new's api**: solved by using verce.
* **Open yahoo finance website with different stocks (with dynamic value)**: solved by finding a pattern in yahoo finance' URL which is passing the stock symbol in ‘${dynamic value}’ https://finance.yahoo.com/quote/${dynamic value}?p=${dynamic value}.
* **Get real-time stock prices**: solved by finding free API to provide real-time stock prices (API URL: https://yahoo-finance-api.vercel.app/${symbol}), (youtube https://www.youtube.com/watch?v=JVZXz6awRj4&t=3573s).
* **Let Finansis calls puppeteer scripts (automation scrips)**: solved by creating auto API using node js express which is each endpoint will call specific automation script, and then using Axios to let Finansis calls the API. note: I tried to call puppeteer scripts in react which is Client-side rendering but didn't work because puppeteer scripts only run in Server-side rendering.
* **Let Finansis control the popup window that Finansis opened**: solved by using puppeteer automation functionality and then doing the same as the above problem's solution.
* **Let Finansis calls python scripts (Algorithmic Trading scripts)**: solved by creating API using python Flask each endpoint will call specific python script, and then using Axios to let Finansis calls Flask API.
* **Let the user enter info field by field and finansis keep interacting with the user**: solved by creating 'InputModal' modal which will open when the time to enter information and after a user finishes inputting then entering 'Enter' and then close the modal and save entered data in redux, lastly let finansis response when a data in redux changes.
* **Send emails to users**: solved by creating endpoint in auto API which will call puppeteer automation script to send emails to a user. note: I tried to use nodemailer but it didn't work without a VPN
* **Create back-testing interface**: solved by cloning Investopedia simulator home page and then customizing it.
* **Create back-testing functionality**: solved by creating python Flask buy, sell, get sp500 Data, update stocks price and check is market open or not endpoints and then using them in React.
* **Manage state in big application**: solved by using redux which is react state management
* **get the real stocks in the s&p 500 for a specific year in backTesting functionality**:this problem is has been not solved yet.
* **Let finansis learn about stock market questions:** solved by searching for an answer in DuckduckGo search engine using puppeteer to type the question, select the first website, scrap the answer and then save the answer in the database. actually, if the question was 'what is market capitalization' finansis will search about the question and add 'investopedia' word at the end of the question for example 'what is market capitalization investopedia' so then the first result will be from investopedia website and that because investopedia is a good website for finding stock market answers
* **Manage state in big application**: solved by using redux which is react state managemen
* **Manage state in big application**: solved by using redux which is react state managemen

## Data flow graph: [3]

### Data flow graph division tasks

#### Figure 1 DFD-3-1 done by batis

#### Figure 2 DFD-3-2 done by benri

#### Figure 3 DFD-3-3 done by nur

### The first data flow graph:

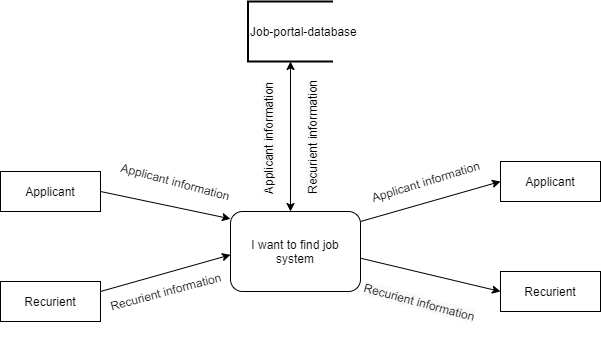


Figure 4 DFD-1

### The second data flow graph:

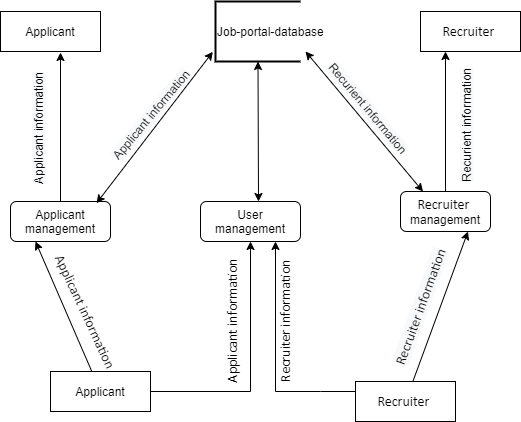


Figure 5 DFD-2FD-2

### The third data flow graph:



Figure 6 DFD-3-1

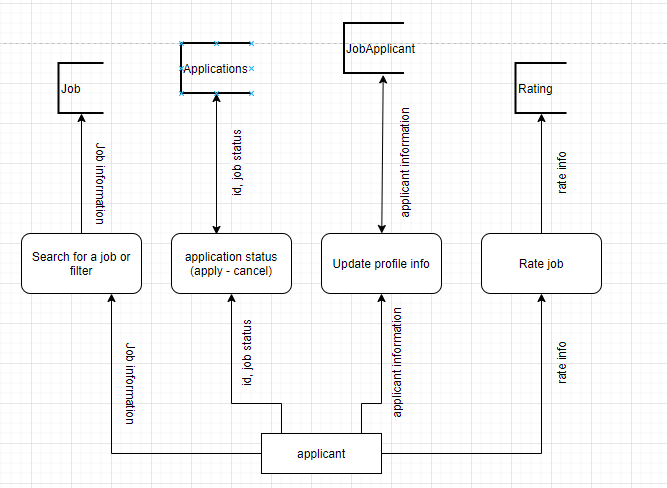


Figure 7 DFD-3-2

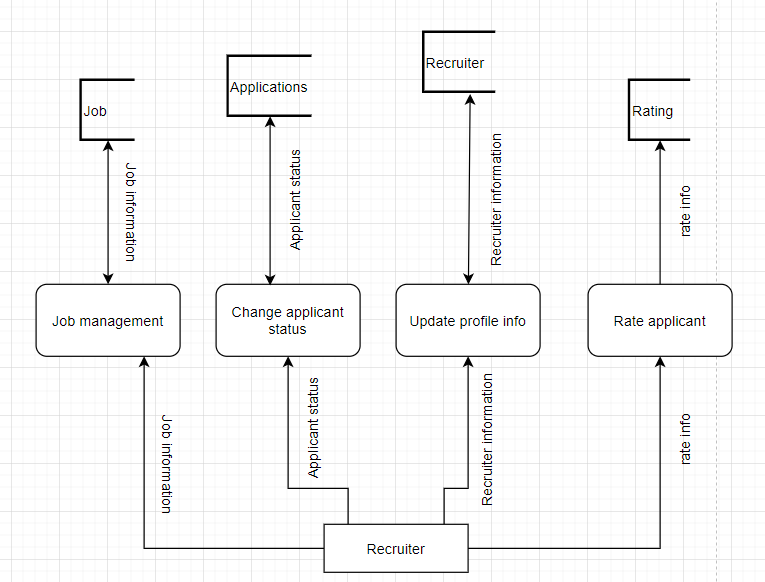


Figure 8 DFD-3-3

# SYSTEM DESIGN

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement has been specified and analyzed, system design is the first of the three technical activities - design, code and test that is required to build and verify software. [4]

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

## System structure diagram

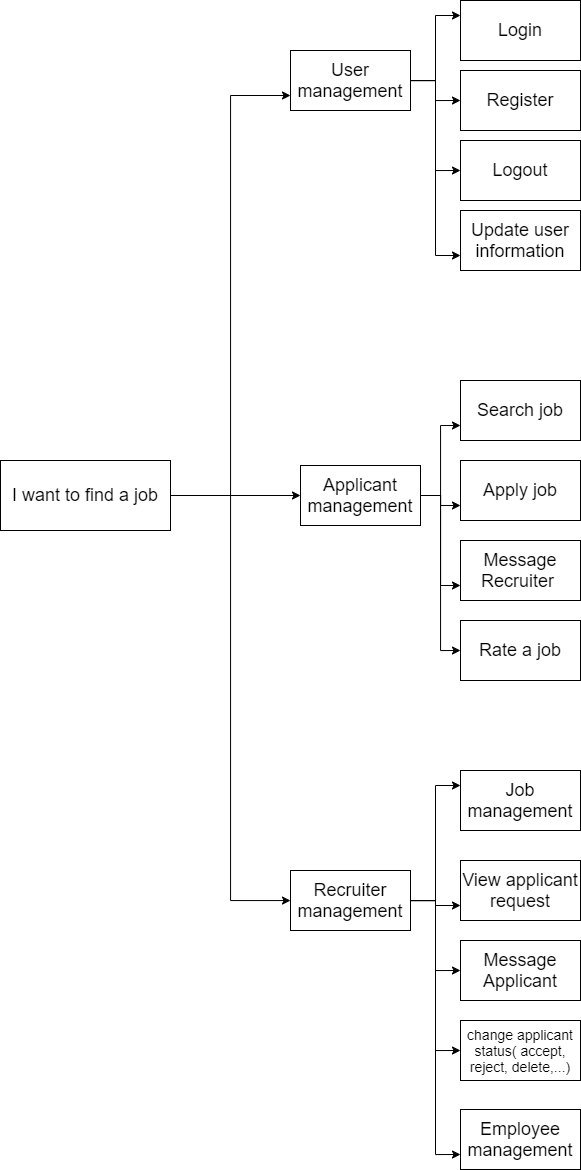


Figure 9 System structure diagram

## Module description:

### Module description division tasks

#### User management: Register, login, logout, update user information

#### Recruiter management: Job management, View applicant request, Message Applicant, change applicant status (accept, reject, delete), Employee management.

#### Applicant management: search Job, apply for job, Message Recruiter

* User management: Register, login, logout, update user information

|  |  |
| --- | --- |
| Register |  |
| input | first the user will select whether he is an applicant or recruiter.  if the user is an applicant he has to input: username, password, email, education, skills, resume and profile picture.  If the user is recruiter he has to input: username, password, email, contact umber and bio. |
| process | The system will check all the required information and will give notice if he does not fill some info.  The system will check the information has been filled in the correct way. |
| output | If the information all have been filled correctly, the information will be inserted in the (user and applicant) or (user and recruiter) tables. Otherwise, the system will return the error message. |

Table 1 Register

|  |  |
| --- | --- |
| Login |  |
| input | The user will input the information to log in to the website: username or email, password. |
| process | The system will check whether the user is already registered or not. |
| output | if the user has been already registered then he will go directly to the home page, if not or the information is not correct the system will return the error message. |

Table 2 Log in

|  |  |
| --- | --- |
| Logout |  |
| input | the user has to click to the logout button |
| process | Close the page and open another page |
| output | After the user click on the logout button the user will go directly to the login page. |

Table 3 log out

|  |  |
| --- | --- |
| Update user info |  |
| input | The user has the ability to update his information but it depends of the type of the users.  if the user is applicant he can edit: username, password, email, education, skills, resume, profile picture.  If the user is recruiter he can edit: username, password, email, contact umber, bio. |
| process | The system will check all the information that has been edited. |
| output | the information that has been edited will edit the information in the user and applicant table or the user and recruiter table. Otherwise, the system will return the error messages. |

Table 4 update user info

* Applicant management: search Job, apply for job, Message Recruiter

|  |  |
| --- | --- |
| Search Job |  |
| input | Typing the job title, and you can add some filters to it. |
| process | The system will check the input query and then if there are jobs that satisfy the query it will return the jobs. |
| output | array of jobs that satisfy the query |

Table 5 search job

|  |  |
| --- | --- |
| Apply for Job |  |
| Input | The user will input will click on specific job’s apply button and send short message with the application |
| Process | The system will add the application to the database |
| Output | Message for the user “applied successfully” |

Table 6 Apply for Job

|  |  |
| --- | --- |
| Message Recruiter |  |
| Input | Text to the recruiter |
| process | Sending the message to Chat Engine |
| output | Sent message |

Table 7 Message Recruiter

|  |  |
| --- | --- |
| Rate a job |  |
| Input | Number from 0 to 5 by select on the number of starts |
| Process | The system will check how many starts the user selected |
| output | New average ring for the job. |

Table 8 Rate a job

* Recruiter management: Job management, View applicant request, Message Applicant, change applicant status (accept, reject, delete), Employee management.

|  |  |
| --- | --- |
| Job management |  |
| input | The user can add job to the website. |
| process | The system will update the information of the job. |
| output | If the information all have been filled correctly, the information of the job will be displayed on the job list. |

Table 9 Job management

|  |  |
| --- | --- |
| View applicant request |  |
| input | The user will check the applicant request about the job. |
| process | The system will generate the list of current applied applicants. |
| output | The system will return the list of applicants. |

Table 10 View Applicant Request

|  |  |
| --- | --- |
| Message Applicant |  |
| input | the user has can message the applicant regarding the job. |
| process | Open messages or close messages. |
| output | The applicant receives the messages from the user. |

Table 11 Message Applicant

|  |  |
| --- | --- |
| Change applicant status (accept, reject, delete) |  |
| input | The user has the ability to update the applicant information. User can accept, reject or delete the information of the applicant. |
| process | The system will check all the information that has been edited. |
| output | the information that has been edited will update the status on the applicant table. Otherwise, the system will return the error messages. |

Table 12 Change applicant

|  |  |
| --- | --- |
| Employee management |  |
| input | the user can filter the employee according to the new applicants or old applicants. |
| process | Filter the employee list. |
| output | After the user filters the settings. System will show the list of the applicants according to the filter. |

Table 13 Employee management

## Data Analysis

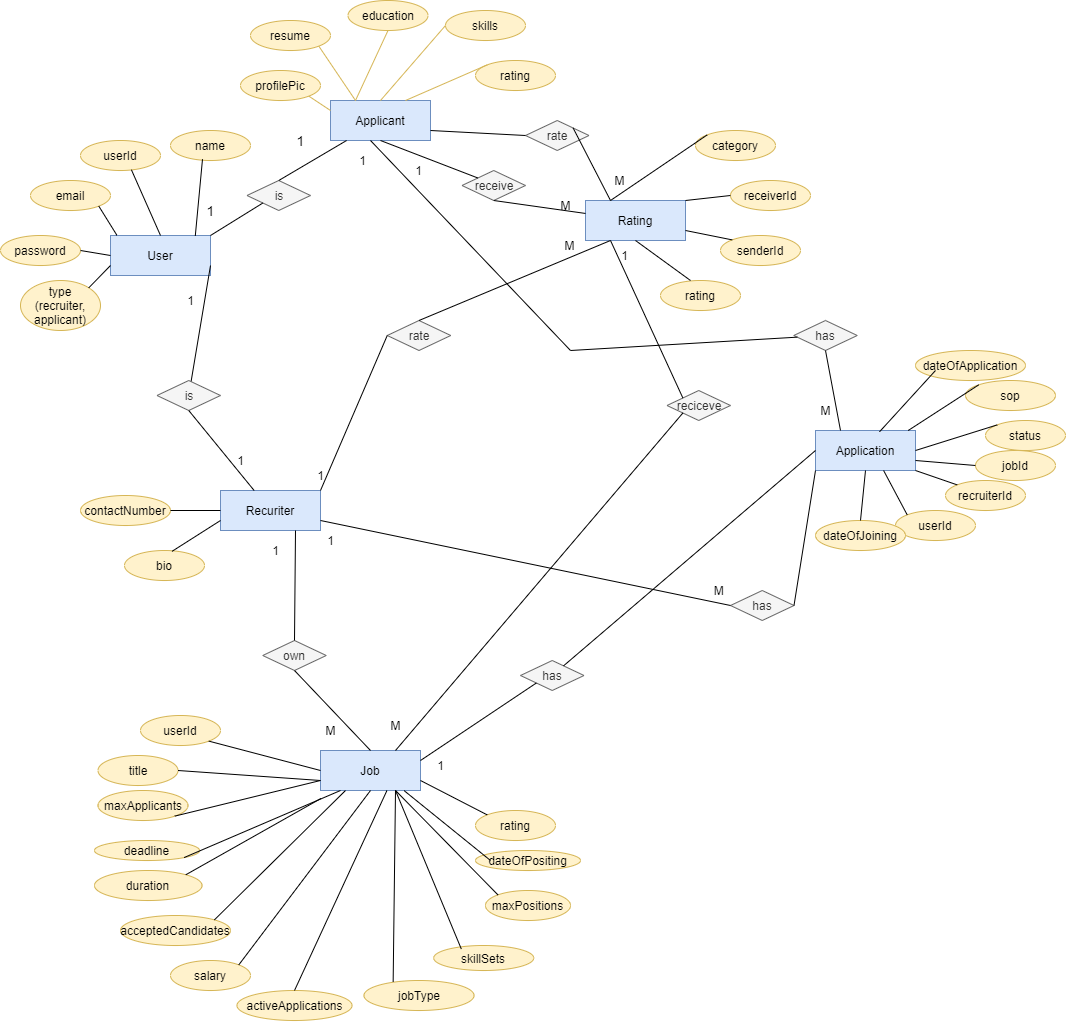


Figure 10 ER-DIAGRAM

## Database Design

### Database Design division tasks

#### User management (user, applicant, recruiter) done by batis

#### Job – rating (done by nur)

#### Application (done by benri)

* User (userId, name, email, password, type)

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Field description | Type | Remarks |
| userId | The id of the user | mongoose.SchemaTypes.ObjectId | required: true, |
| Name | The name of the user | String | required: true |
| email | Email of the user | mongoose.SchemaTypes.Email | unique: true,  lowercase: true,  required: true, |
| password | Password of the user | String | required: true,  select: false, |
| type | Type of the user whether he is an applicant or recruiter | String | enum:["recruiter", "applicant"],  required: true, |

Table 14 User model

* Applicant (userID, education, skills, rating, usernameChatEngine, resume, profilePic)

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Field description | Type | Remarks |
| Education | Applicant’s education and it contains:  institutionName  startYear  endYear | Array(string,Number,Number) | Required: true, |
| skills | The skills that the applicant has | [String] | max: 5.0,  default: -1.0, |
| Rating | The applicant can give rating | Number |  |
| usernameChatEngine | usesnameChatEngine of the applicant | String |  |
| Resume | The resume of the applicant | String |  |
| profilePic | The profile picture of the applicant | String |  |

Table 15 applicant model

Part of the simulation data:

{

(121 ,{‘wenzhou’,2018,2022}, [‘React’,’Node.js’] , ‘batis121’,’C:\Download’,’ C:\Download’)

(122, {‘wenzhou’,2018,2022},[‘React’,’Node.js’], nur122, ‘C:\Download’, ‘C:\Download’))

(123, {‘wenzhou’,2018,2022},[‘React’,’Node.js’],,‘benri123’,’C:\Download’, ‘C:\Download’))

}

* Recruiter (userID, contactNumber, bio)

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Field description | Type | Remarks |
| ContactNumner | The contact number of the Recruiter | String | Validate whether the number is Valid or not. |
| bio | The bio of the Recruiter | String |  |

Table 16 Recruiter model

Part of the simulation data:

{

(121,’16212988212’, ’ this is my bio ’)

(122,’12231313412’,’ this is my bio’)

(123,’12131312412’,’ this is my bio’))

}

* job (userId, title, jobType, salary, skillSets, maxPositions, dateOfPosition, maxApplicants, activeApplications, duration, acceptedCandidates, deadline, rating) （Benri）

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Field description | Type | Remarks |
| userId | Id of the recruiter | mongoose.Schema.Types.ObjectId, | required: true, |
| title | Title of the job | String | required: true, |
| jobType | Type of job | String | required: true, |
| salary | Salary of the job | Number | msg: "Salary should be positive", |
| skillSets | Skills required for the job | String |  |
| maxPositions | Number of Position allocated for the particular job | Number | msg: "maxPositions should be an integer",  msg: "maxPositions should greater than 0", |
| dateOfPosition | Starting date of the applicant for the job | Date |  |
| maxApplicants | Maximum applicants applicable for the job | Number | msg: "maxApplicants should be an integer",  msg: "maxApplicants should greater than 0", |
| activeApplicants | Active applicants applied for the job | Number | msg: "activeApplications should be an integer",  msg: "activeApplications should greater than equal to 0", |
| Duration | Duration of the job | Number | msg: "Duration should be an integer", |
| AcceptedCandidates | Applicants accepted for the job | Number | msg: "acceptedCandidates should be an integer",  msg: "acceptedCandidates should greater than equal to 0", |
| Deadline | Deadline for applying the job | Date | msg: "deadline should be greater than dateOfPosting", |
| Rating | Applicants give rating to job after getting accepted for the job | number |  |

Table 17 Job model

* Rating: (Nur)

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Field description | Type | Remarks |
| Category | Category of rating (Job or Application) | String | enum:  ["job","applicant"],  required: true, |
| reciverId | Who receive the rating a ID (job or Application) | mongoose.Schema.Types.ObjectId | required: true, |
| senderId | Who send the rating A id (applicant or recuriter) | mongoose.Schema.Types.ObjectId | required: true, |
| rating | Give a rating to job or applicant | Number | max: 5.0,  default: -1.0,  min: -1.0 |

Table 18 Rating model

* application (

)

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Field description | Type | Remarks |
| dateOfJoining | Date of joining (data of accepting) | date | dateOfJoining should be greater than dateOfApplication |
| sob | A short Message to the recruiter | String | Statement of purpose should not be greater than 250 words |
| status | Status accepted, rejected… | Array(string,Number,Number) | Required: true, |
| jobid | Job id | Mongoose.objectID | required: true, |
| Recryiterid | recruiterid | Mongoose.objectID | required: true, |
| userId | User id to | Mongoose.objectID | required: true, |
| DateOfApplication | Date of applying for the job | date | default: date of applying to the job |

Table 19 application model

## Algorithmic Design [5]

### Algorithmic design division tasks

#### Insert job – delete job – get all jobs (done by batis)

#### Search job –apply for Job – rating Jobs (done by Nur)

#### Register – log in – update user (done by benri)

* Insert Job algorithm

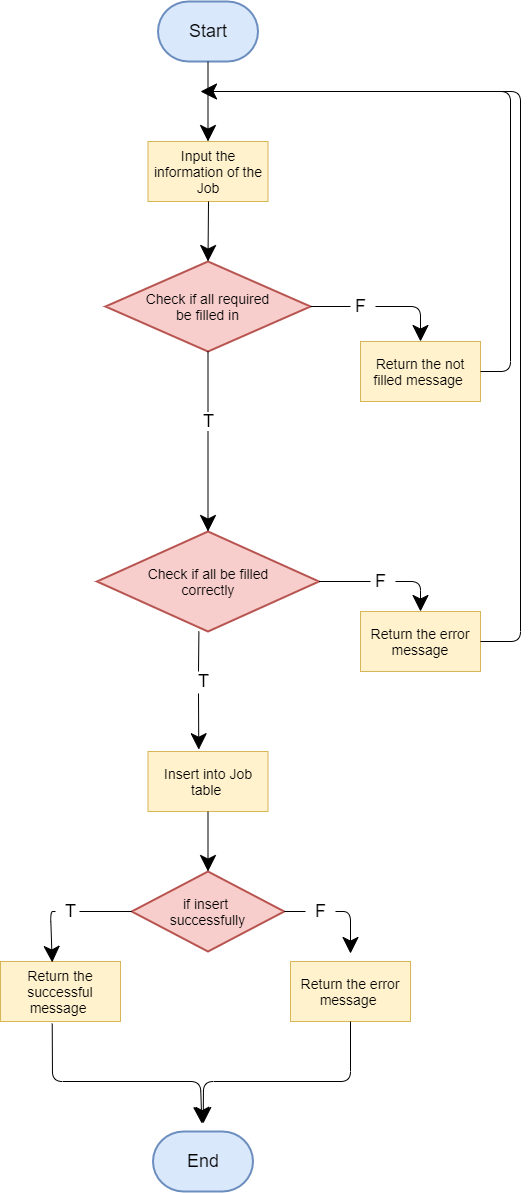


Figure 11 Insert Job

* Delete Job algorithm

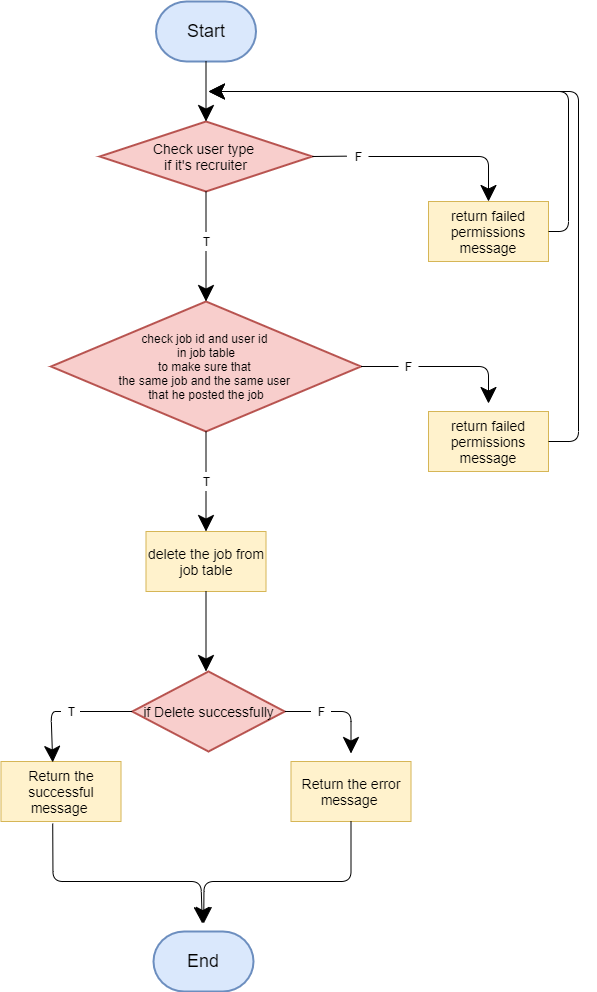


Figure 12 Delete Job

* Get all Jobs

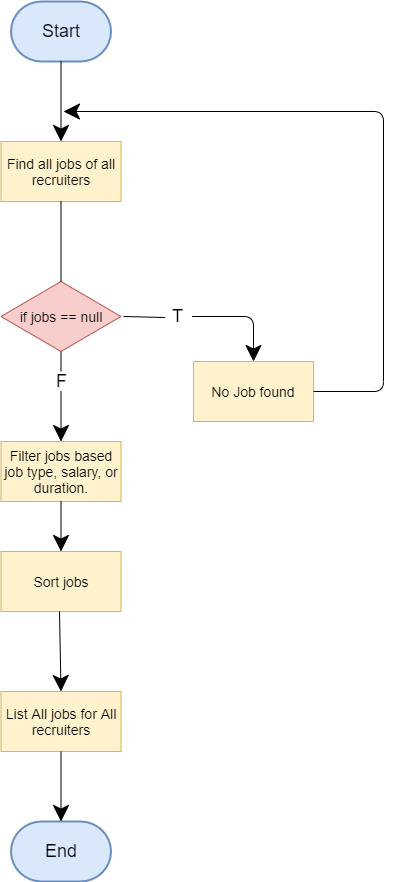


Figure 13 Get all jobs

* REGISTER NEW USER algorithm

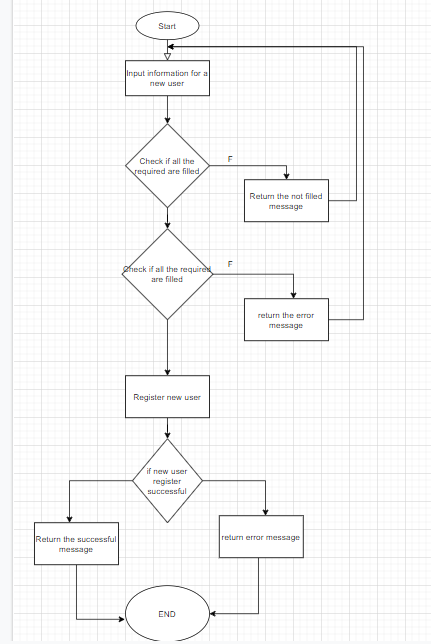


Figure 14 Register

* Login User Algorithmic Design

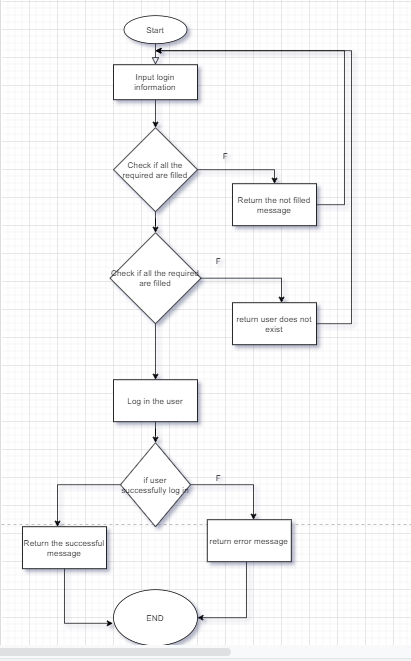


Figure 15 Log in

* Update User Algorithmic Design

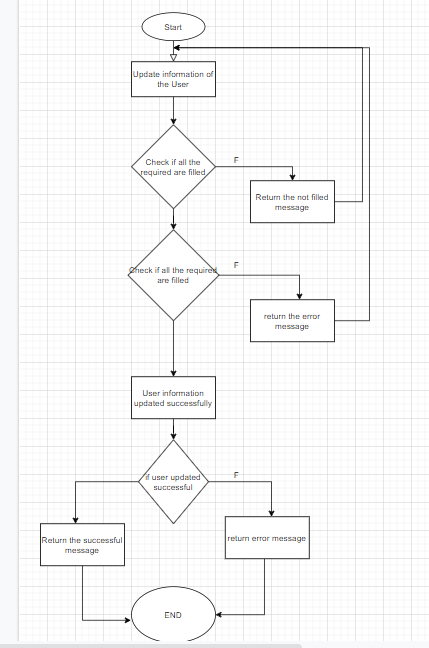


Figure 16 Update user

* search Job algorithm

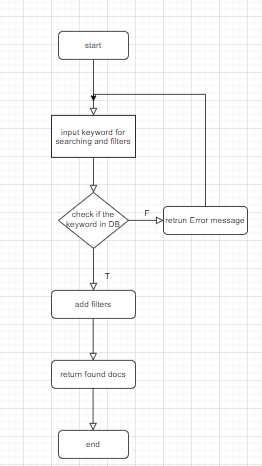


Figure 17 Search Job

* apply for Job algorithm

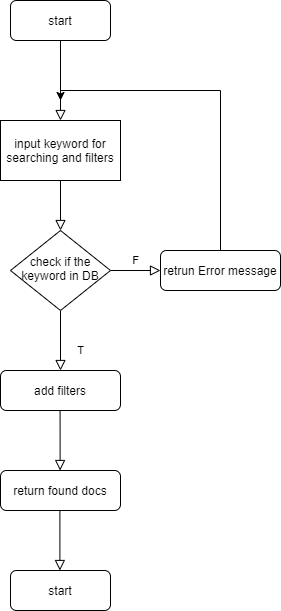


Figure 18 Apply for Job

* rating Jobs

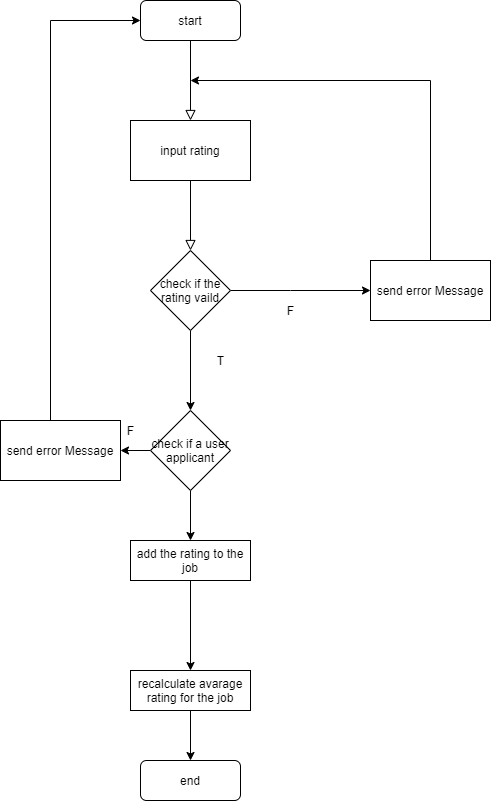


Figure 19 Rating a job

# TEST AND ANAYLSIS REPORT

## Test division tasks:

### Applicant registration- Recruiter Registration - User login (Done By batis)

### User add job – user update job – viewing applications

## Applicant Registration Test [6]

* Function name Applicant Registration

Testing purposes: To test whether users can’t register with invalid input such as name, email, password, education start year date, education start year date, skills, profile photo, resume or blank field.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Design Reasons** | **Test Case Data** | **Expected Result** | **Actual Result** | **Conclusion** |
| 1 | Invalid-Equivalence Partitioning | Mohammed Omer Salem batis. | Username must be less than 20 |  | Fail |
| 2 | Error Guessing | batis@batis.com | Email is already existed. |  | pass |
| 3 | Boundary Analysis | Verify if blank spaces are passed in required fields. | Name, Password email, is required |  | Pass |
| 4 | Invalid-Equivalence Partitioning | 1234567892548225 | password must be less than 20 |  | fail |
| 5 | Boundary Analysis | Education start year 1929 | Start Year` (1929) is less than minimum allowed value (1930) |  | pass |
| 6 | Boundary Analysis | Education end year greater than start year  start year: 2000  end year: 1999 | End year should be greater than or equal to Start year |  | pass |
| 7 | Error Guessing | Start year or end year float. | Start year and end year must be integer. |  | pass |
| 8 | Error Guessing | Profile pic format: .jpg/.png and resume format .pdf | Select only these formats from file system. |  | pass |

Table 20 Applicant Registration

Conclusion: By the test, I found that most of test cases passed, except the password and email length I haven’t realized them well, now I have to do a specific length for username and password.

## Recruiter Registration Test

* Function name Recruiter Registration
* Testing purposes: to test whether users can’t register with invalid input such as name, email, password, phone number.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Design Reasons** | **Test Case Data** | **Expected Result** | **Actual Result** | **Conclusion** |
| 1 | Boundary Analysis | blank spaces are passed in required fields (name, email, password). | Name is required  Email is required  Password is required. |  | pass |
| 2 | Boundary Analysis | +8615658563280 | Phone number length just 12 digits with the country key number. |  | pass |
| 3 | Invalid-Equivalence Partitioning | Batis111111111111111111 | Username length is less than 20. |  | fail |
| 4 | Invalid-Equivalence Partitioning | 5846657752366574221444 | Password must be less than 12 |  | fail |
| 5 | Error Guessing | Batis2@batis2.com | Email is already existed. |  | pass |
| 6 | Invalid-Equivalence Partitioning | this is a new company t……… | Bio is more than 250 characters |  | fail |

Table 21 Recruiter Registration

* Conclusion: Same as applicant register user name and password should have a specific length also recruiter bio length exceeds the limit.

## User login Test

* Function name User login
* Testing purposes: to check When passing a correct or not correct email and password

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Design Reasons** | **Test Case Data** | **Expected Result** | **Actual Result** | **Conclusion** |
| 1 | Boundary Analysis (Verify the login page for email and password, when the field is blank and Submit button is clicked. | Blank field for both email and password | Wrong email and password. |  | pass |
| 2 | Verify if a user will be able to login with a valid email and valid password. | [Batis12@gmail.co](mailto:Batis12@gmail.co)m  12341234 | Logged successfully |  | pass |
| 3 | Verify if a user cannot login with a valid email and an invalid password. | [Batis12@gmail.co](mailto:Batis12@gmail.co)m  123 | Wrong password or email |  | pass |
| 4 | Verify if a user cannot login with a invalid email and an valid password. | [Bati@gmail.co](mailto:Bati@gmail.co)m  12341234 | Wrong password or email |  | pass |
| 5 | Verify if the data in password field is either visible as asterisk or bullet signs. | 12341234 | visible as asterisk |  | pass |
| 6 | Verify if a user is able to login with a new password only after he/she has changed the password. | [Batis12@gmail.com](mailto:Batis12@gmail.com)  23456789 | User can only not edit password. |  | fail |

Table 22 User login

* Conclusion: By the test, I found that most of test cases passed, except the password, the user

## user adding job

* Testing purposes: to test whether users can add job with valid input to the database

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Design Reasons** | **Test Case Data** | **Expected Result** | **Actual Result** | **Conclusion** |
| 1 | Equivalence Partitioning | Application Deadline:  13/29/2021 | Reset the month to 12 |  | Pass |
| 2 | Error Guessing | Maximum Number Of Applicants: 5  Positions Available:30 | Tell the user maximum number of applicants greater then positions available |  | Pass |
| 3 | Error Guessing | Salary: - 1999 | Salary should be positive |  | fall |
| 4 | Error Guessing | Title: 123456789 | Title should be text not number |  | pass |
| 5 | Error Guessing | Title: {empty} | Title is required |  | fall |

Table 23 adding job

* Conclusion: by the test, I found that: maximum number of applicants greater then positions available validations not working, but the other validations are working

## user update job info

* Testing purposes: to test whether users can update job info with valid input

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Design Reasons** | **Test Case Data** | **Expected Result** | **Actual Result** | **Conclusion** |
| 6 | Error Guessing | Title:web1 | title should not contain a number |  | fall |
| 7 | Error Guessing | Salary: -1999 | Salary should be positive |  | fall |
| 7 | Equivalence Partitioning | Application Deadline:  11/18/2021 | deadline should be greater than dateOfPosting |  | fall |
| 8 | Error Guessing | Maximum Number Of Applicants: 20  Positions Available:30 | Tell the user maximum number of applicants greater then positions available |  | pass |
|  |  |  |  |  |  |

Table 24 update job info

Conclusion: by the test, I found that: maximum number of applicants greater then positions available validations not working, but the other validations are working.

## viewing Applications

* Testing purposes: : to test whether recruiters can view Applications and applicants info.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Design Reasons** | **Test Case Data** | **Expected Result** | **Actual Result** | **Conclusion** |
| 9 | Error Guessing | Clicking on view applications btn | List of applications if there are applications |  | pass |
| 10 | Error Guessing | Clicking on view applications btn | Show No Applications Found fi there are no applications |  | pass |
| 11 | Error Guessing | Clicking on view resume btn | Show the recruiter applicant resume if applicant has resume |  | pass |
| 12 | Error Guessing | Clicking on reject btn | Reject applicant and let applicant know |  | pass |
| 13 | Error Guessing | Click on chat with applicant | Start chat with applicant |  | pass |

Table 25 viewing Applications

Conclusion: everything I test in viewing Applications function is working.

# USER MANUAL

## System Operating Environment Requirements.

System development environment involves all the requirements that are needed to build the project, these requirements are the key to the project development with which the is made up of. The environments are classified into different aspects which are the frontend, backend, browser, OS, and programming language Below is a table shown with the list of development requirements used for job website development.

|  |
| --- |
| Programming language；JavaScript |
| Visual Studio Code |
| Windows 10 |
| Google Chrome |
| Backend (NODE.JS, EXPRESS.JS, NoSQL ：Mongo DB) |
| Frontend (html, CSS, react.js, styled components) |

Table 26 System Operating Environment Requirements.

## System Development Environment Description

### Programming language:

JavaScript is a scripting language which is versatile and robust that is popularly used for web frontend and backend development as well as deep learning, it is also used in development of games and mobile applications that works for both android and IOS mobile operating system. JavaScript has a lot of frameworks that are popularly used in a lot of high-tech companies. The most popularly used frameworks for both mobile and web frontend developments are shown below:

* + React (web)
  + React Native (mobile)
  + Angular (web)
  + Vue (web)

In web backend development Express framework and node are mostly and widely used for server-side rendering, testing. This project is JavaScript based full stack web application, the frontend of this project uses react.js, style component and Marital Ui while the backend include node and express, authentication and testing etc.

### Frontend:

The frontend consists of the system visible interactions with visitor or user, the frontend has specific languages used for its development personally, those used in this project are listed as follows below;

* React.js
* Martial UI
* **React.js:** is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. It's used for handling the view layer for web and mobile apps. React also allows us to create reusable UI components
* **Material Ui:** Material-UI is simply a library that allows us to import and use different components to create a user interface in our React applications. This saves a significant amount of time since the developers do not need to write everything from scratch.
* **Style Component:** Styled-components is a library built for React and React Native developers. It allows you to use component-level styles in your applications. Styled-components leverage a mixture of JavaScript and CSS using a technique called CSS-in-JS.

### Backend

The backend consists of the server side rendering and database connectivity of the system which communicates with the system frontend and send back requests or data, the backend building blocks are shown as follows;

**• Mongo Database**

**• Express framework**

**• Node**

**MongoDB**: MongoDB is a document-oriented No-SQL cross platform online database. MongoDB uses json like format to document data with optional schemas. mongo DB can store json data directly as practiced in this project.

**Express**.js: Express.js is a backend web application framework for Node.js, It is designed for web application development, APIs, web-scraping and deep learning. Nodejs: Node.js is a cross-platform, backend JavaScript runtime environment that allows JavaScript execution without browser.

**node.js: i**s uses the middleware for webpage server-side rendering.

## System functions (Recruiter module)

### Add jobs

In this function recruiter has to add job information as you see in Figure 1 this information includes Title, Skills, Job type (full time, part time, flexible …), duration, salary, application deadline, max number of applications and position available.

When the recruiter clicks to the create job button, the job will be listed in the homepage and applicants can view it.

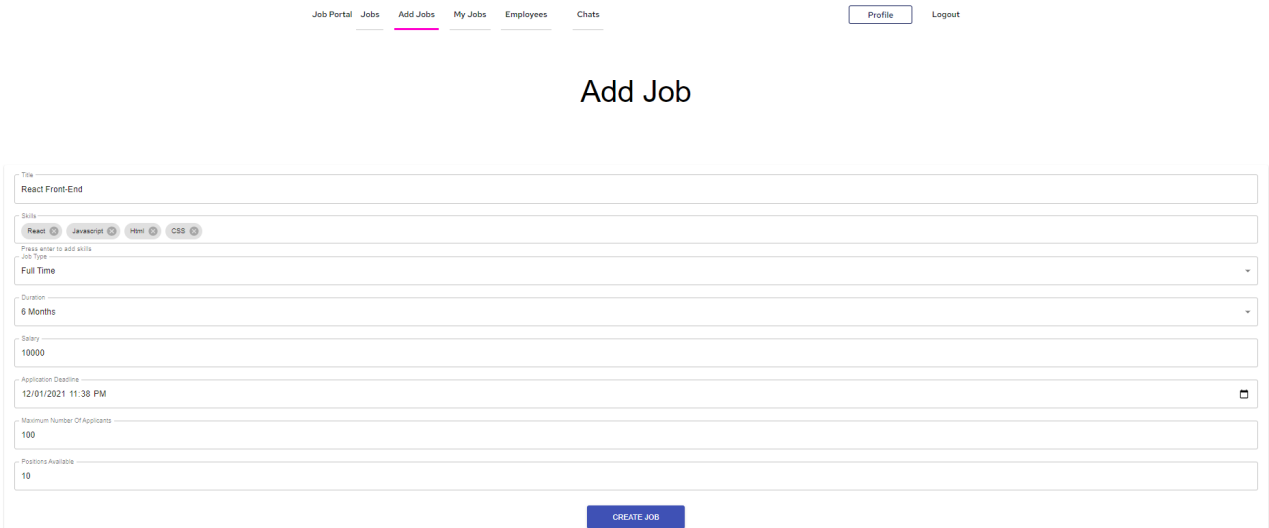


Figure 20 add jobs

### Jobs Applications management

When the applicants apply for any job, in this page recruiter will receive the applications for each job, as you see below in this Figure 2, recruiter can update the details for these jobs for example extend the max number as well as the available positions for this job, also he can delete the job, so there is no applicant will apply for it anymore.



Figure 21 jobs applications

If the recruiter views the applications, he will see all the applicants that has been applied for this job, as you see in this Figure 3, recruiter can download the resume of the applicant and view his picture, chat with him, also he can reject him so he will not be there anymore. When the recruiter clicks to shortlist that means the applicant has the change to be accepted there, but he has to wait for an interview or any kind of information that the recruiter required him.



Figure 22 applications status

When the recruiter clicked to shortlist, the Figure 4 below will appear, so the applicant has the chance to be accepted, if the recruiter clicks on the Accept button.

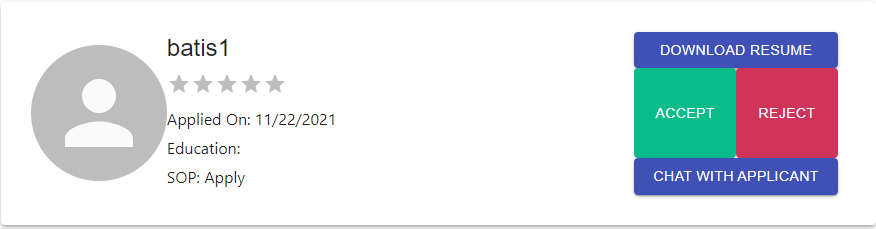


Figure 23 applications status

### Employees management

If the applicant accepted in the job. The applicant now will be an employee in this company, so the recruiter can rate him as well as end this job then he will not be there anymore.

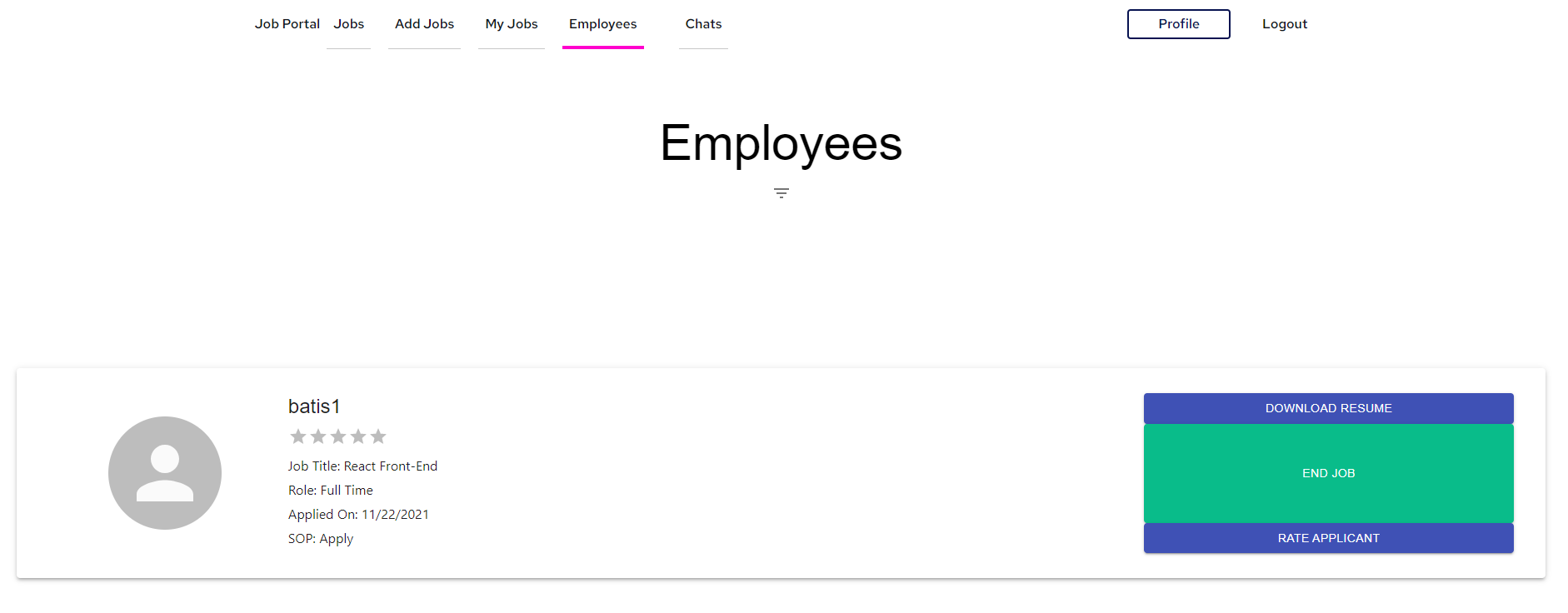


Figure 24 employees management

### Chatting

If the recruiter wants to arrange an interview with applicant, he can chat with him and arrange it, also he can get any information if he wants.

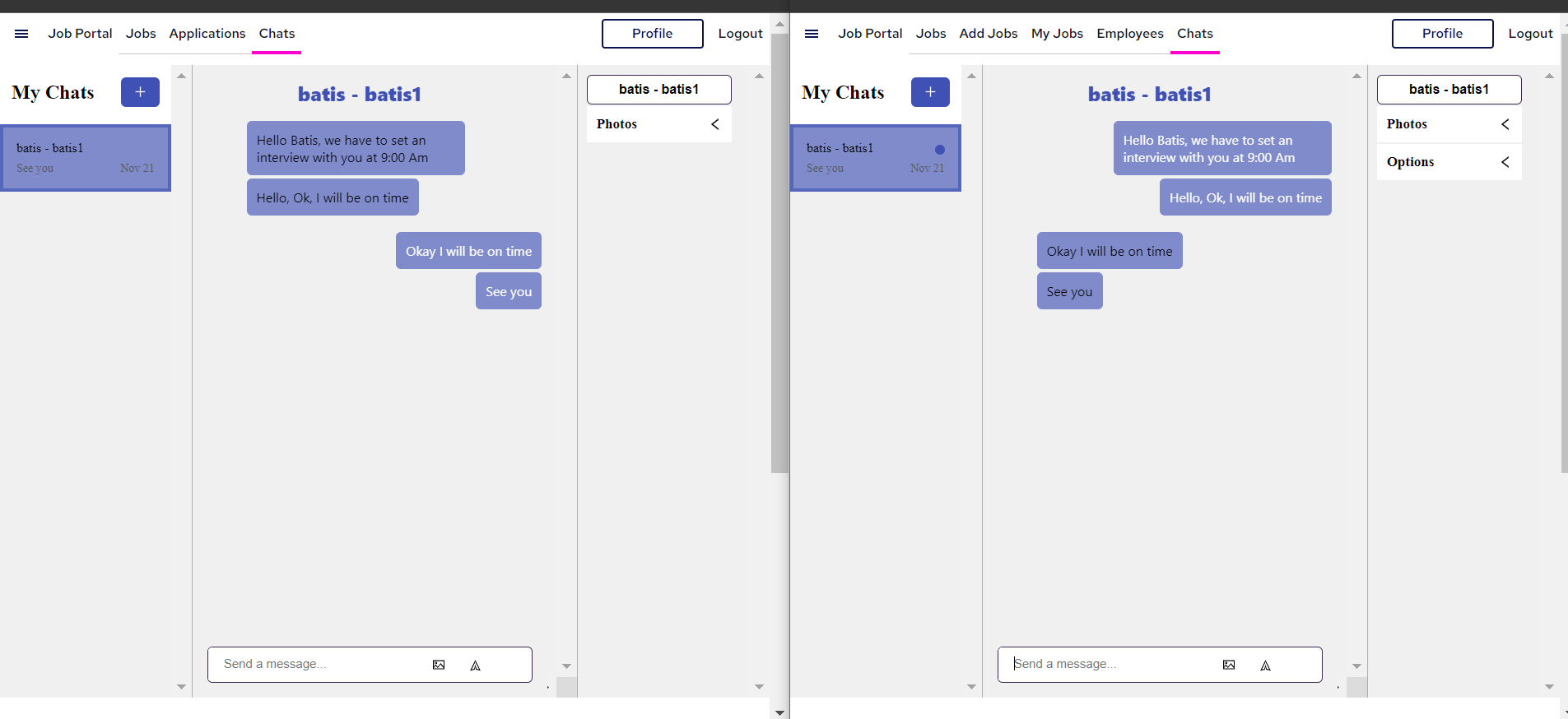


Figure 25 chatting

## Group photo



## Detailed description of division of labor

* The system consists of three main functions:
* Function 1: User management
* Function 2: Applicant management
* Function 3: Recruiter management
* 5.3.2 The group consists of three students:
* Batis A
* Nur B
* Benri C
* Function 1:

|  |  |
| --- | --- |
| **duty** | **student** |
| 1. Requirement Analysis +Summary Design | batis |
| 1. Detailed Design+ Coding | Benri |
| 1. Test | Batis |

Table 27 Function1-division

* Function 2:

|  |  |
| --- | --- |
| **duty** | **student** |
| 1. Requirement Analysis +Summary Design | Benri |
| 1. Detailed Design+ Coding | Nur |
| 1. Test | Benri |

Table 28 Function2-division

* Function 3:

|  |  |
| --- | --- |
| **duty** | **student** |
| 1. Requirement Analysis +Summary Design | Nur |
| 1. Detailed Design+ Coding | batis |
| 1. Test | Nur |

Table 29 Function3-division

# Conclusion

By the end of this course, we would like to thank the teacher for giving us this opportunity to design the whole system. During this project I developed problem solving skill, improved in algorithm writing and developed fast research skills, also following Software engineering method to develop any system.

## Need for new System

* Proposed Job Portal system consists of 3 modules: Job Seeker, Employer and Administrator.
* Online Job Portal will provide the fast operation and low-cost expense than old system.
* Easy job search, which is a job seeker need
* Location search which is must needed

## Summary of gains and losses

### Gains

* Working collaboratively, instead of individually, helps improve productivity。
* follow software engineering cycle to develop a whole system
* Good time management
* learn more about some new technology’s frameworks.
* Learn how to work as group remotely using GitHub, and use cloud database for all of us.

### Loses

* Daly some tasks on time
* We didn’t make the system design consistency on every pages
* working with a team group, sometimes it’s quite challenging, since the differ on the level and the responsibility for the individual.
* Using new frameworks made us a bit slowly, since not everyone of us has been used it before.

REFERENCES

|  |  |
| --- | --- |
| [1] | K. Tucker, " Definition of Job Portals," [Online]. Available: https://careertrend.com/facts-6883857-definition-job-portals.html. |
| [2] | B. M. Martin, "Non-functional requirment," [Online]. Available: https://www.guru99.com/non-functional-requirement-type-example.html. |
| [3] | W. b. C. Chi, "A Beginner's Guide to Data Flow Diagrams," [Online]. Available: https://blog.hubspot.com/marketing/data-flow-diagram. |
| [4] | E. Team, "9 Steps to Write a System Design Document," , [Online]. Available: https://www.projectpractical.com/9-steps-to-write-a-system-design-document-free-template/. |
| [5] | V. s. g. website, "Using flow charts to design algorithms," [Online]. Available: https://www.education.vic.gov.au/school/teachers/teachingresources/discipline/english/literacy/Pages/using-flow-charts-to-design-algorithms.aspx. |
| [6] | B. T. Hamilton, "What is BLACK Box Testing? Techniques, Example & Types," [Online]. Available: https://www.guru99.com/black-box-testing.html. |