AI for Agent Recruiting

Functional Specification

Name: Kerry Agabi.

Student Number: C00249804.

Supervisor: Greg Doyle.

Date: 17/04/2023

Introduction

This documentation will consist of the functional specification of this project. The application as a whole entity will be broken down and highlighted to what the actual product design is, its core and non-core pieces of functionalities, the metrics, and scaleable usability.
The aim of this project is to construct a functional predictive churn model that can predict employee attrition.
The first section of this functional specification document involves defining what the application is, the product, and finally the target users. The remaining sections of this document will highlight the use case of the project as a whole entity, which will consist of Unified modeling language diagrams. Similarities of other projects available will be discussed in detail and differences will be elaborated upon.

The Metric section of the report will attempt to measure how successful the project development is becoming. This section will also outline FURPS+ as a guide to measuring the success rate.

Model Architecture

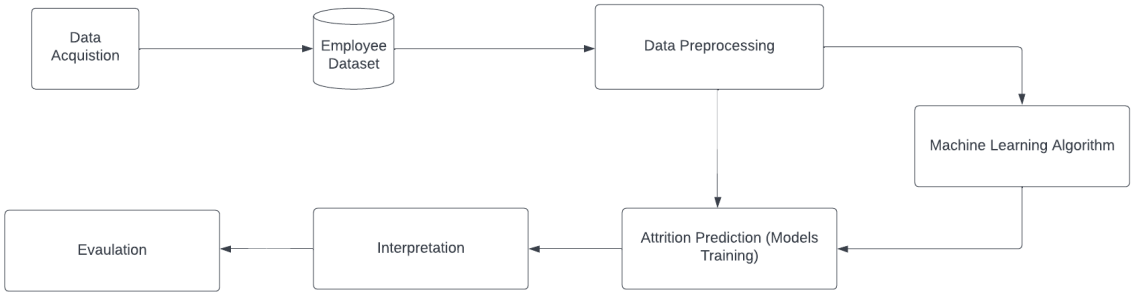

Figure 1: Architecture of the churn model.

Attrition Predictive Model:

Datasets are acquired from colonial life, pre-processed by the engineers, trained, and embedded into the model. Necessary libraries are imported, and Python functions are then ran to infuse the datasets into the machine-learning algorithms. For prediction accuracy, missing values are replaced with the mean of existing ones to prevent bias, The importance metric provides a score indicating how valuable each category was.

Use Case Diagram

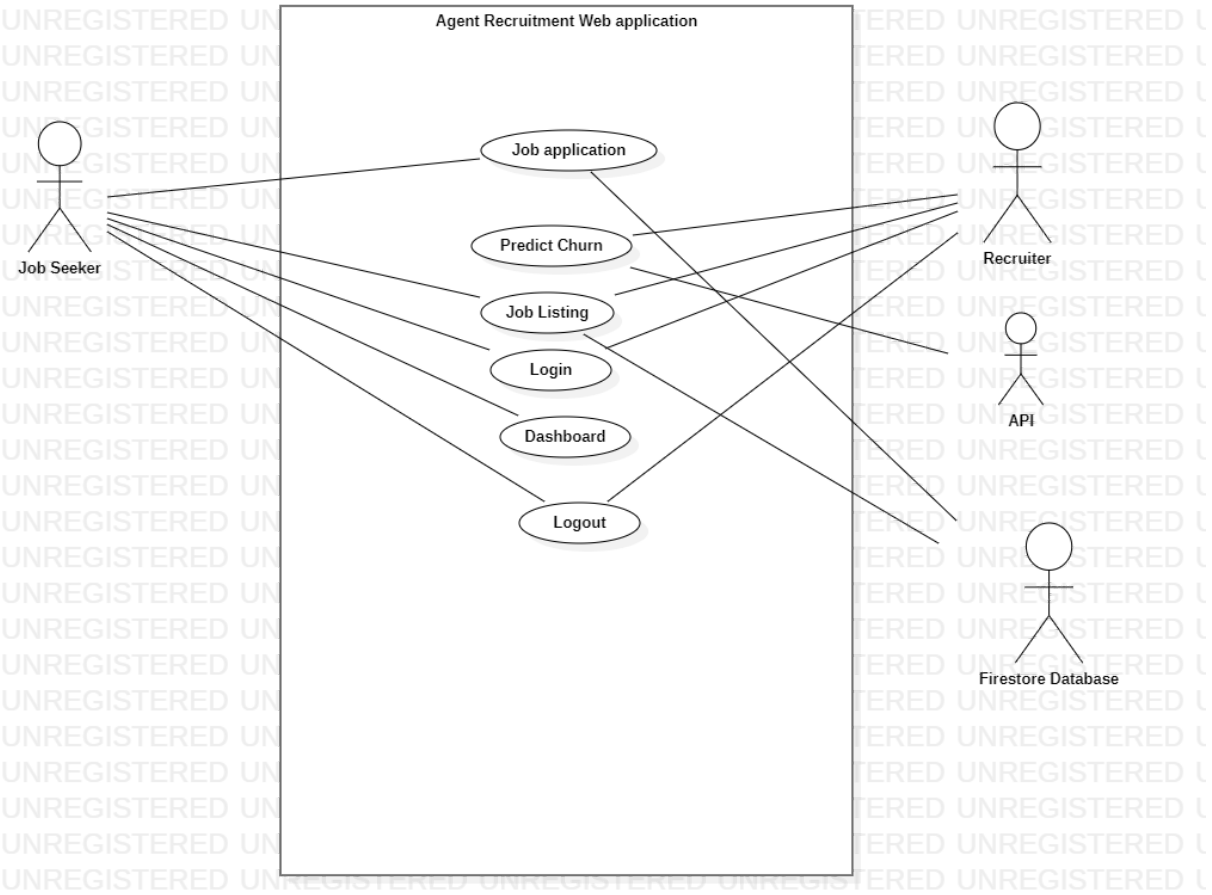

Figure 2: Use Case Diagram

Login Use Case

Use Case Name	Login
Actors	Recruiter, Job seeker
Brief Description	This use case describes the login process of a Recruiter or Job seeker.
Activity	1. The Recruiter or Job seeker loads the recruitment page. 2. The Recruiter or Job seeker clicks on the signup button in the Navbar. 3. The Recruiter or Job seeker clicks on login instead. 4. The Recruiter or Job seeker enters the required credentials e.g. email and password. 5. The Recruiter or Job seeker presses the login button. 6. Authentication and verification from the Firebase authenticator and the database are notified. 7. The Recruiter or Job seeker is redirected to the home page.
Alternatives	2a. The user enters the wrong credentials and is unauthorized 1. The user is prompted with a message saying “Invalid login credentials”.

Logout Use Case

Use Case Name	Log out
Actors	Recruiter, Job seeker
Brief Description	This use case describes the logging out process of a Recruiter or Job seeker.
Activity	1. The Recruiter or Job seeker presses the log-out button.

Dashboard Use Case (Upload Resume)

Use Case Name	Dashboard
Actors	Job seeker, Firestore Database
Brief Description	This use case describes the process of a Job seeker adding/updating a resume to the dashboard
Activity	1. The Job seeker goes to the dashboard page. 2. The Job seeker clicks on update dashboard. 3. The Job seeker clicks the choose file button in the upload CV card component. 4. The Job seeker is prompted with a file explorer library. 5. The Job seeker selects the file and is redirected to the upload page again with the name of the file that was chosen. 6. The Firestore database stores the uploaded file.
Alternatives	2a. The Job seeker picks a non-supported file format 1. The Job seeker is prompted with “File type not supported”. 2. Job seeker is redirected to step 2.

Predict Churn Use Case (Predictive Churn Model)

Use Case Name	Predict Churn
Actors	Recruiter, API
Brief Description	This use case describes the process of running the logistic regression model to predict employee attrition.
Activity	1. The Recruiter clicks on the recruiter button and is redirected to the recruiter menu page. 2. The Recruiter clicks predict churn. 3. The Recruiter fills in the required information. 4. The Recruiter clicks the predict churn button for the logistic regression model. 5. The API calls the flask server that and sends the information from the user. The necessary python script runs. 6. The API returns an output to the front end of the Application.
Alternatives	2a. Recruiter clicks on the user guide to learn how to use the employee attrition model.

Job Listing Use Case (Advertise)

Use Case Name	Job Lisitng
Actors	Recruiter, Firestore database
Brief Description	This use case describes how a Recruiter can advertise a job listing
Activity	1. The Recruiter clicks on the recruiter button and is redirected to the recruiter menu page. 2. The Recruiter clicks on advertise a Job. 3. The Recruiter fills in the form with the job listing information. 4. On submit the Recruiter is then prompted with a successful alert. 5. The firestore database populates the new job listing .
Alternatives	2a. The Recruiter leaves a field blank which triggers the required attribute.

Job Listing Use Case (Update)

Use Case Name	Job Lisitng
Actors	Recruiter, FirestoreDatabase
Brief Description	This use case describes how a Recruiter can update a job listing
Activity	1. The Recruiter clicks on the recruiter button and is redirected to the recruiter menu page. . 2. The Recruiter clicks on Update a Job listing. 3. The Recruiter clicks on Update Job button. 4. The Recruiter fills in the form with the updated job listing information. 5. The firestore database populates the updated information.
Alternatives	3a. The Recruiter does not wish to update a job and exits the page

Job Listing Use Case (Delete):

Use Case Name	Job Lisitng
Actors	Recruiter, Database
Brief Description	This use case describes how a Recruiter can delete a job listing
Activity	1. The Recruiter clicks on the recruiter button and is redirected to the recruiter menu page. 2. The Recruiter clicks on Delete a Job listing. 3. The Recruiter clicks on Delete Job button. 4. The Recruiter is prompted with an alert to ask are they sure they want to delete the job lisitng. 5. The Recruiter confirms the deletion and the job listing is deleted from the firestore database.
Alternatives	3a. The Recruiter does not wish to delete a job and exits the page

Job Listing Use Case (View):

Use Case Name	Job Listing
Actors	Recruiter, Job seeker
Brief Description	This use case describes how users can view a job listing
Activity	1. The Recruiter or Job seeker clicks on the jobs button in the Nav bar and is routed to the jobs card page. 2. The Recruiter or Job seeker clicks on view details button to view the particular job listing description.
Alternatives	2a. The Recruiter or Job seeker toggles to the hide details button to hide the job description for the particular listing and exits the page.

Job Application Use Case:

Use Case Name	Job Application
Actors	Recruiter, Firestore Database
Brief Description	This use case describes how a Recruiter can view a job application
Activity	1. The Recruiter clicks on the recruiter button and is redirected to the recruiter menu page. 2. The Recruiter clicks on the view job applications button. 3. The Recruiter clicks on view applications button to view all the applicants for the particular job lisitng.
Alternatives	3a. The Recruiter toggles the view details button to view the job description for the particular listing and exits the page

Job Application Use Case(View):

Use Case Name	Job Application
Actors	Job seeker, Database
Brief Description	This use case describes how a Job seeker, can view a job application
Activity	1. The Job seeker, clicks on the Job Seeker button and is redirected to the Job seeker's menu page. 2. The Job seeker clicks on the view your job applications button. 3. The Job seeker's applications are then generated from the firestore database back to the page..
Alternatives	2a. The Job seeker toggles to the view details button to view the job description for the particular listing and exits the page

Job Listing Use Case(View Matching):

Use Case Name	Job Listing
Actors	Job seeker, Firestore Database
Brief Description	This use case describes how a Job seeker, can view a job application.
Activity	1. The Job seeker, clicks on the Job Seeker button and is redirected to the Job seeker's menu page. 2. The Job seeker, clicks on the view matching job listing button. 3. The Job seeker's information is fetched from the firestore database which is inputed into a job matching algorithm. 4. The best matching jobs for the Job seeker are then filtered and displayed to the Job seeker.
Alternatives	4a. The Job seeker does not have the sufficient information stored in the firestore database and the job cards are filtered as normal.

Dashboard Use Case (Upload Profile Photo)

Use Case Name	Dashboard
Actors	Job seeker, Firestore Database
Brief Description	This use case describes the process of a Job seeker adding/updating a profile Photo to the dashboard
Activity	1. The Job seeker goes to the dashboard page. 2. The Job seeker clicks on update dashboard. 3. The Job seeker clicks the choose file button in the upload profile photo card component. 4. The Job seeker is prompted with a file explorer library. 5. The Job seeker selects the file and is redirected to the upload page again with the name of the file that was chosen. 6. The Firestore database stores the uploaded file.
Alternatives	2a. The Job seeker picks a non-supported file format 1. The Job seeker is prompted with “File type not supported”. 2. Job seeker is redirected to step 2.

Dashboard Use Case (Personal Details)

Use Case Name	Dashboard
Actors	Job seeker, Firestore Database
Brief Description	This use case describes the process of a Job seeker adding/updating their personal details to the dashboard
Activity	1. The Job seeker goes to the dashboard page. 2. The Job seeker clicks on update dashboard. 3. The Job seeker fills out the forms in personal details card component and clicks save. 4. The Job seeker is prompted with a re-authentication alert. 5. The Job seeker fills out the re-authentication form and the change in data is reflected back to the database.
Alternatives	4a. The Job seeker puts in the wrong credentials in the authentication form and the personal detail fail to save.

Dashboard Use Case (Career Information)

Use Case Name	Dashboard
Actors	Job seeker, Firestore Database
Brief Description	This use case describes the process of a job seeker adding/updating their career information to the dashboard
Activity	1. The Job seeker goes to the dashboard page. 2. The Job seeker clicks on update dashboard. 3. The Job seeker fills out the forms in career information card component .

	4. The Job seeker's change in data is reflected back to the database.
Alternatives	3a. The Job seeker ignores a field in the card component and the HTML required attribute is triggered. F

Dashboard Use Case (Work Experience)

Use Case Name	Dashboard
Actors	Job seeker, Firestore Database
Brief Description	This use case describes the process of a job seeker adding/updating their work experience information to the dashboard
Activity	1. The Job seeker goes to the dashboard page. 2. The Job seeker clicks on the update dashboard button. 3. The Job seeker fills out the forms in the work experience card component. 4. The Job seeker's change in data is reflected back to the database.
Alternatives	3a. The Job seeker ignores a field in the card component and the HTML required attribute is triggered.

Dashboard Use Case (Education)

Use Case Name	Dashboard
Actors	Job seeker, Firestore Database
Brief Description	This use case describes the process of a job seeker adding/updating their education information to the dashboard
Activity	1. The Job seeker goes to the dashboard page. 2. The Job seeker clicks on the update dashboard button. 3. The Job seeker fills out the forms in the education card component. 4. The Job seeker's change in data is reflected back to the database.
Alternatives	3a. The Job seeker ignores a field in the card component and the HTML required attribute is triggered.

Application

This Application is designed to streamline the hiring process for companies by automating tasks such as job listing, resume screening, and predicting potential employee turnover. The design of the job recruitment web application would also include a user-friendly interface that allows easy navigation and access to the various features of the application. This application will be built using React JS and python backend frameworks such as Flask.

Functionality in order of importance includes

1.1 Predictive Employee Churn Model

This model can be used to improve the recruitment process, by predicting which candidates are likely to stay with the company for a long period of time this model will use logistic regression model to predict the probability of employees leaving the organization (attrition) based on various features, such as demographics, job characteristics, and satisfaction levels. The functionality of this model can streamline the hiring and recruiting process. The functionality importance is high due to the fact that it will be a separation from the current competition of job recruitment web applications and it will be extremely beneficial to recruiters.

Target Users

The target user for this application are recruitment managers and job seekers whose goal is to find the best fit for their individual roles and predict the high level of churn while also recruiting the most efficient candidates.

Metrics

To gauge the success of this project, There will be an accuracy standard in terms of the algorithmic prediction which will be evaluated. For the success of this project, user must be able to complete all the use cases listed above in figure 1 and get an output that correlates with the dataset.

The metric can also be measured by the amount of time it takes to fill a job opening, from the time the job listing is created to the time a candidate is hired. A lower time-to-hire indicates that the recruitment process is efficient and streamlined.

In terms of the implementation of a predictive employee churn model, the metric can be later on measured by analyzing the percentages of new hires that stay with registered recruiters for a certain period of time. A high retention rate indicates that the recruitment process is effective at hiring candidates that are a good fit for the company and that the onboarding process is successful.

To further establish the success rate of this project, the FURPS+ model will be used as a guideline. “FURPS is a technique to validate the prioritized requirements after an understanding of the client's needs and necessities. The acronym FURPS is Functionality, Usability, Reliability, Performance, and Supportability, the “+” of the FURPS+ acronym allows us to specify constraints, including design, implementation, interface, and physical constraints.

Functionality

This application should have all the features and functionalities that are needed to effectively manage the recruitment process, such as job listing, resume parsing, and candidate matching.

- The authorized Job seeker or recruiter must be able to log in.
- The authorized Job seeker or recruiter must be able to log out.
- The system should be able to return the login validation.
- The system should allow the recruiter to advertise a job.
- The system should allow the recruiter to delete a job.
- The system should allow the recruiter to update a job.
- The system should allow the recruiter and Job seeker to view the job listings
- The system should allow the Job seeker to update their dashboard.
- The system should allow the Job seeker to view jobs that are outputted from the matching algorithm.
- The system should allow the Job seeker to apply for a job.
- The system should be able to give the recruiter a prediction on the specified employee.
- The system should allow the recruiter to view job applications based on the matching score of the matching algorithm.
- The system should allow the Jobseeker to view their job applications.

Usability

The application should be easy to navigate and use, with a user-friendly interface that makes it simple for recruiters, hiring managers, and job seekers to access the features they need.

- The application should be responsive across different browsers. E.g. Chrome, Firefox, Edge.
- The application should be user-friendly.
- Job seekers should be able to import resumes, profile photos, and other dashboard entities effortlessly.
- Error messages or prompts should be visible, informative, and straight to the point.

Reliability

The application should be reliable and available at all times, with minimal downtime and errors. It should be able to handle high volumes of data and traffic. Although due to the

implementation of a machine learning backend would have an impact on the response times, the downtime should be kept at a very high standard.

- The application must have a 99% uptime.
- The application must be able to handle errors and debug issues without crashing completely.

Performance

The application should have good performance and response times, with minimal delays and lag when accessing and processing data.

- There are machine learning models, which means the application should be able to handle a massive load of user data from the front-end of the application JSON response stored in the database should be instantaneous.
- Data stored to the firestore or real-time database should be instantaneous

Supportability

The application should have good documentation and support resources available, making it easy for users to get help when they need it. It should also be easy to update and maintain, with minimal downtime.

- This application should allow the addition of more training to the models.
- This application must be supported across the three major browsers. E.g. Chrome, Firefox, Edge.

+

This allows us to specify constraints, including design, implementation, interface, and physical constraints.

- The application must use the latest Hypertext Transfer Protocol Security(HTTPS) especially when being integrated from a Flask server to a React front-end.

- To ensure secure access to the web application, Firebase Authentication was implemented. This service provided a range of authentication methods, including email and password, social media logins, and Single Sign-On (SSO) options. Additionally, Firebase's built-in security features protected the application's data, ensuring compliance with data privacy regulations.

Similar Application:

International Business Machines(IBM) currently have an employee attrition human resource analytic model. This model takes in factors such as:

- Salary
- Satisfactory level
- Growth opportunities
- Facilities
- Policies and procedures.

This model uses Decision Tree Modeling to sample the data based on the proportion of the current attrition in the company. Validation techniques are used to assess how the results of the model will generalize independent test datasets.

The main difference between this predictive churn model and IBM's employee attrition analytic model is the simple fact that this model is completely based on specific information from resumes provided, unlike IBM's analytic model which is based on information provided at exit interviews.

Recruitment Agency websites such as Jobs.ie are similar applications.

Conclusion

This functional specification document highlights the key functionality required for the success of this application. The use case is discussed in detail by the creation of a detailed use case table for every use case in the UML diagram. The metrics and target users are defined with the requirements and roles they contribute to the success of this application. In conclusion, a job recruitment web application built with React JS can provide an efficient, user-friendly, and reliable solution for streamlining the hiring process. By incorporating features such as job listing, applicant tracking, and candidate matching. The application can automate many of the tasks associated with recruitment and help to improve the overall efficiency of the process. The use of React JS in the development of the application allows for the easy creation of reusable components, efficient rendering, and fast performance, which are crucial in building a responsive web application that can handle large amounts of data and traffic. The application should also be able to meet the FURPS requirements (Functionality, Usability, Reliability, Performance, and Supportability) to ensure that it is quality software that meets the needs of all the users the application is intended for.