Project Phase #2:

Business Requirements and Preliminary Project Requirements Document

The world of business is a difficult one and many functions have to be completed to ensure everything runs efficiently. For smaller companies looking to scale it can often be difficult to manage employee growth using within new HR departments. We would love to propose for group 8 to create a simple system for internal operations in HR by creating an employee management system. This system will be used by HR and Management to track and manage employees at each part of a growing business. This would include having things such as location, occupation, where they worked prior, and contact information. HR and Management should be able to add, update, and remove any info from the system when necessary. There should also be a list sorted by different categories such as occupation, level, and location so that messages that should only go to certain groups be done easily. This system will increase the efficiency of any scaling business immensely if implemented.

To meet all the software requirements, the program should be able to complete these functions:

- This system should only be accessible to HR and Management.
- This system should allow to view all of the employees within the company.
- This system should allow to add new employees to the system after being hired.
- This system should allow to remove employees to the system after being fired.
- This system should allow to update the information of employees.
- This system should allow to sort a list of employees by Occupation, Location, and Level, Past Experience, and Time in position.
- This system should allow to search for employees by Occupation, Location, and Level, Past Experience, and Time in position.

User Experience

Our program is going to be used by HR & Management within this company. With this being said, it will have a readable UI that is easy to use. Users will navigate the menu by typing in a number to choose what task they want to do. When adding an employee, the user will be asked to add employees by Occupation, Location, and Level, Past Experience, and Time in position. For this part inputs will be strict to ensure consistency throughout the system. To remove items, the user will use the prompt from the menu to do so. To sort the employees, the menu will give the user options for the type of way it's sorted.

Mockup images of the UI

User Experience Test Plan

- **Test Case #1**: The HR management system should not be accessible to anyone who has not been authenticated.
    - If the user provides invalid login information the program should go to a different menu where they will be prompted to either retry login or quit the program entirely.
    - If the user has successfully authenticated, the program should take them to the proceeding menu
    - Both the password and login email must be valid in order for the authentication to be valid.

- **Test Case #2**: For each menu, the user must input a valid integer that has been shown in the menu.
    - If an invalid integer has been entered, an error message must be displayed and the same menu must be displayed again.
    - If a valid integer has been entered the program should proceed to an appropriate menu.
    - For a menu selection input to be valid, it must be a positive integer displayed on the screen.

- **Test Case #3**: For every menu, the back option should always take the user to the previous and no other menu.

- **Test Case #4**: All inputs should be validated.
    - For example: If an email is to be input, proper validation and message should be displayed.
    - For password input during account creation, the password must contain a numeric value, a symbol, and a capital letter, and must be at least 8 characters long.

Software

- **UI/UX:** At the most basic level, the UI should be built as a fully featured terminal application. In future iterations, it could be expanded to a web application or a mobile application.
- **Backend**: The backend will compromise a file that holds all information. Sensitive information MUST be encrypted. Future iterations can include a database (SQL, MongoDB, etc)
- **Menu**: The user must be able to go back from any menu to the previous menu. It can be implemented using the Stack data structure.
- **Categorization**: The application must be able to categorize employees according to their department.
- **OOP Philosophy**: The program should be written with OOP philosophy to ensure code readability, modularity, and portability of code. Each data entity must be modeled using a class for example <<Employee>>.

Software Test Plan

- Different inputs will need to be considered during testing such as handling invalid inputs correctly. With invalid inputs such as incorrect data type and out-of-bound values, the system will need to handle it by redirecting the user to try again. In addition, it should also handle any foreseeable error that a user should make by alerting the user.

Projected Usability

In today's world where individuals are moving from job to job, we project that usability should be at its maximum potential. Based on the five tests of usability - effectiveness, efficiency, error tolerance, how engaging it is, and ease of use; we expect to rank near the 80th percentile. According to the Pareto Principle (also known as the “80/20 rule”), we tend to use 20% of features 80% of the time. Translating this into our projections, we hope that our users can be adding/removing/viewing an employee, updating their information, and sorting/searching the database around 80% of the time, respectively. Alternatively, we hope the user can at least use two of the seven features eighty-percent of the time.