COSC 499 - Capstone

Manpower Scheduling Board Project Charter

Horizon Electric Inc

Adam Fipke
Jordan Roberts
Eddy Zhang
Edwin Zhou
Yuan Zhu

Project Purpose or Justification

The purpose of this project is to develop a manageable chart for Horizon Electric Inc., a local Electrical Contractor employing 100+ employees, to maintain and provide outlook for future manpower requirements. Detailed manpower scheduling is essential in the construction industry to maintain productivity on all projects. This project aims to improve Horizon Electric Inc.'s manpower scheduling and planning processes, ultimately increasing their productivity and efficiency.

Assumptions and Constraints

Assumptions

- The developers must be able to program in HTML, CSS, JS, PHP.
- The developers must allocate time to learn necessary technologies that they are not familiar with, such as Docker.
- The developers must follow the plan established in this and other planning documents.
- The developers must be able to deliver key milestone and complete the project within the planned time.
- Stakeholders must be available within the scheduled period.
- The developers must have functional computers to work with.
- The client must possess a way to host the final product.
- None of the developers must drop the course.
- The client must have a server to host the website locally.
- It is assumed that each developer must put in 16-20 hours per week.
- It is assumed that developers must regularly stay in contact with one another and must mention if they will be unavailable

Constraints

- The developers' proficiency in technology.
- Time constraints. The project must be done and handed over to the client by the week of August 13th, 2023.
- ~~The projector view will have to take in consideration of a low-resolution projector as well as the colour of the backdrop.~~ The client will be using 16:9 T.V. with a resolution of 3840*2160 for the presenter view
- The managers will be using chrome or edge browsers on windows computers

High-level Project Description and Boundaries

- Digitize Horizon Electric's existing employee "whiteboard" and make it easy to reschedule employees via dragging and dropping name "bubbles."
- For each job, have a manually editable future month employee count forecast.
- Project must be able to run constantly on a docker container on a locally hosted server.

High-level Risks

- Unexpected loss of team members (dropping out of class, sickness, switching teams, etc.)
- Poor communication within the team.
- Poor communication with the client.
- Learning curve of Docker, CI/CD pipeline or other new technologies.
- Poor estimation of the complexity of tasks.
- Technical errors that result in the loss of data.
- Not doing the project will result in the failure of the course and an unhappy client, as well as possibly damaged university reputation.
- If the system were to be error-prone, it might result in the unintentional mismanagement of employees.

Summary Milestone Schedule

May 30	Development start

June 25	50% completion of the project
July 20	At least 80% complete (Start review, refactoring, and polishing software)
Aug 6	Aim to have 100% complete and deliver product to client.

Stakeholder List

- Horizon Electric Inc.: The client and primary stakeholder for this project.
    - Dave Clark: The contact person for Horizon Electric Inc.
- Project team: The team responsible for developing the manageable chart for Horizon Electric Inc.
- Horizon Employees: The employees of Horizon Electric Inc. will be affected by how well and effectively the system performs. Errors from the software could cause employees to be allocated to a job that has too many employees or has too few.
- University of British Columbia Okanagan: Responsible for supplying the contract between Horizon Electric and the project team. Reputation may be damaged if project done poorly.
- Clients of Horizon Electric: These clients rely on the proper management of Horizon Electric's time and employees. Mismanaged employees might cause the job to be done inefficiently and ineffectively.

Approvals

- Project Sponsor: Dave Clark, Horizon Electric Inc., [Signature]
- UBC Representative: Scott Fazackerley, The University of British Columbia, [Signature]
- Project Developer: Edwin Zhou, Development Team, [Signature]
- Project Developer: Adam Fipke, Development Team, [Signature]
- Project Developer: Eddy Zhang, Development Team, [Signature]
- Project Developer: Yuan Zhu, Development Team, [Signature]
- Project Developer: Jordan Roberts, Development Team, [Signature]

Scope

Description of Software

A standalone, locally deployed web-based manpower scheduler that quickly allows users to quickly view and allocate employees to jobs. Users will be able to view the project manager(s), the employees, and the history of number of employees a current or upcoming job has. "The goal is to produce a manageable chart to maintain and provide outlook for future manpower requirements."

Updated Requirements

Disclaimer: The strongness of the word "will" is equal to the strongness of the word "must." Treat them as synonymous.

High Level Requirements

- Knowledge of the Linux, Apache, MySQL, PHP (LAMP) web programming stack. ✓
- Experience in designing graphical user interfaces to ensure that the system's operation is smooth and reliable. ✓
- Experience with version control systems such as Git. ✓

Non-functional Requirements

- The system must be constructed with the possibility of having multiple accounts in mind. ✓
- The website must be accessible on the following desktop browsers: Google Chrome, Edge ✓
- The website must be visually similar and function the same across browsers. ✓
- For usability goals in the website's UI, it must prioritise learnability and memorability, meaning that it will not take a large effort to re-learn. We can test this by having the construction managers try to navigate the UI for the first time. ✓
- The websites UI must use appropriate use of colours to identify and indicate important areas. ✓
- All frequent input fields must have both client-side and server-side validation, where applicable. ✓
- ~~Data sent from the client to the server will use HTTPS for encryption.~~
- The software/website must initially load within 5 seconds. ✓
- The system must correctly sum and display the counts of the employees. ✓
- Developers must regularly communicate with each other via Discord and messages must be viewed at least once a day. ✓
- The web server must have a "time-last-changed" variable that the clients must query before pulling all of the information from the web server, reducing web traffic✓
- All open views of the website must be periodically updated every X seconds, displaying any recent changes ✓
- The system must be able to function and run efficiently (I.e., with at most a 0.5s added delay) with up to 3 active clients at one time. ✓
- The system must use a database to hold jobs, their employees, and their history. ✓
    - This must run periodic, daily backups, and hold these backups for up to 30 days. ✓
- The following histories must be saved:
    - What jobs an employee has worked and for how many days ✓
        - Saved on database
        - Incremented at midnight
        - Can be used for visualisations
    - ~~The expected employee counts for each job and the actual employee counts~~ ~ not needed, we calculated it from our *assignments* table which keeps track of who was assigned where and for how long
        - ~~Actual employee counts must be saved over time. Since employees are not always assigned to one job for the entire month, it must be saved as a fraction~~
        - ~~Saved daily at midnight on database~~
        - Can be used for visualisations

- ~~The history of user actions on the schedule~~ **X**
  - ~~Saved on client~~
  - ~~Used for undo~~
    - This was not implemented due to the exponential amount of complexity that it would introduce into the project. In the interest of delivering a complete program to the client, the opportunity cost was too great.
    - Basic text undo will still be available through the default browser undo functionality.
  - ~~Updated after every action~~
- Employees are able to work multiple jobs ~~(not more than one at a time),~~ and this must be taken into account when building a history of employees. ✓
- ~~The projector view will have to take in consideration of the colour of the backdrop and must be built in a way that all text will be readable on a 4k projector~~ The presenter view be visually presentable on the client's 16:9 T.V. with a resolution of 3840*2160 ✓

New Non-Functional Requirements

- The system must be responsive in that it must notify the user when changes have been successful or errors ✓
- The system must give the user emphasised warnings for permanent or not easily reversible actions ✓
- The system's server must use an SQL account with minimal privileges (INSERT, UPDATE, DELETE, SELECT, any other required for the system to function) wherever possible ✓

Functional Requirements

- Managers will be able to view a history of previous jobs and when each employee was assigned to the job. ✓
- The system must ensure that recent changes are reflected in all open views of the webpage ✓
- Managers must be able to:
  - Easily allocate and move employees from job to job. Allocate and move by 1-click mouse move. Click the employee who the manager wants to move, and release the mouse at where the employee should be assigned to. This will auto-update the sum of employees for that job, and show a total of all allocated employees at the bottom of the screen. ✓
    - Foremen can be allocated to multiple jobs but must only be counted once in the employee counts. ✓
    - 12 months must be displayed at one time, starting at the current month. ✓
      - Upon the current real month changing, the entire calendar will shift and will stop displaying the last month and will display a new month (rolling calendar). The old month will be archived ✓
  - Edit employee information. Click on the employee's name, then a window pop up to allow manager to edit the details of the employee, click save before submitting the new details. ✓
  - Create new jobs. Click create new jobs button, then a window pops up to allow manager to enter the details of the new job, click save before submitting. ✓
  - Archive Jobs. The user must be able to indicate to archive a job, then the job and its related information will be archived. This essentially performs a "soft delete", where the job is made hidden from most pages but the data of who worked on the job is still kept. ✓
  - Edit job details. Right-click the job that needs to be edited, choose the "edit" option, a window pops up allowing the enter new/update old information. Click save before updating. ✓
  - Edit the number of required employees for a job in a given month, which the system must show a sum of at the bottom of the page ✓

        - The schedule will have indicators on where the expected start and end month is, but must allow users to change the employee count for months after the end month in case projects can go longer than expected ✓
    - ~~The system must allow users to undo previous actions.~~ X Not worth the opportunity cost
- For a job's employee requirements for a month, the system must highlight when there an inadequate, an adequate, and when there is a surplus amount of employees by displaying different colors in the portal. ✓

New Functional Requirements

- The system must be able to archive and unarchive employees, effectively performing a "soft delete" where the employee is no longer visible on most screens (still needing somewhere to see them so they can be unarchived) except the history of what jobs the employee has worked on in the past is retained. ✓
- All visible and changeable information in the old system (the whiteboard) must be similarly visible and changeable in the new system including: jobs and their basic info (titles, project manager, address), employees and their basic info (names, roles, notes), where employees are assigned (a specific job, inactive, in school, or just unassigned), the next 3 months of projections and the sum of these projections for each month (sum does not need to be changeable), and employee sums for each job and a total across all jobs (does not need to be changeable) ✓

List of Technical Requirements

- The website will be developed using HTML, CSS, JS, PHP, and MySQL as the primary programming languages. ✓
- The developers will validate user login with frontend and backend validation and encrypt passwords via hash and salt before storing them in a database. ~ system hashes but does not salt the passwords.
- The system must be able to translate and upload the employees in their current excel sheet (exported to a csv) into the database ✓
- The website is hosted on a local server that does not have access to the internet, therefore no web-based APIs on the server may be used. ✓

List of User Requirements

- Managers will be able to view a history of previous jobs and when each employee was assigned to the job. ✓
- The system must ensure that recent changes are reflected in all open views of the webpage ✓
- Managers must be able to:
    - Easily allocate and move employees from job to job. Allocate and move by 1-click mouse move. Click the employee who the manager wants to move, and release the mouse at where the employee should be assigned to. This will auto-update the sum of employees for that job, and show a total of all allocated employees at the bottom of the screen. ✓
    - Foremen can be allocated to multiple jobs but must only be counted once in the employee counts. ✓
    - 12 months must be displayed at one time, starting at the current month. ✓
    - Upon the current real month changing, the entire calendar will shift and will stop displaying the last month and will display a new month (rolling calendar). The old month will be archived ✓
    - Edit employee information. Click on the employee's name, then a window pop up to allow manager to edit the details of the employee, click save before submitting the new details. ✓
    - Create new jobs. Click create new jobs button, then a window pops up to allow manager to enter the details of the new job, click save before submitting. ✓

        - Archive Jobs. The user must be able to indicate to archive a job, then the job and its related information will be archived. This essentially performs a “soft delete”, where the job is made hidden from most pages but the data of who worked on the job is still kept. ✓
    - Edit job details. Right-click the job that needs to be edited, choose the “edit” option, a window pops up allowing the enter new/update old information. Click save before updating. ✓
    - Edit the number of required employees for a job in a given month, which the system must show a sum of at the bottom of the page ✓
    - The schedule will have indicators on where the expected start and end month is, but must allow users to change the employee count for months after the end month in case projects can go longer than expected ✓
        - ~~The system must allow users to undo previous actions.~~ X Not worth the opportunity cost
- For a job’s employee requirements for a month, the system must highlight when there an inadequate, an adequate, and when there is a surplus amount of employees by displaying different colors in the portal ✓
- Any employee should be able to be duplicated ✓
    - Duplicate Employees should appear separate from their regular counterparts (grey-ed out) ✓
    - Duplicate employees will not be counted in the employee counts and totals ✓
    - Assigning a duplicate employee will still count as the employee being assigned ✓

New User Requirements

- The job view must implement a method for the user to custom sort the list of jobs (even in non-standard orders) ✓
- The system must be able to archive and unarchive employees, effectively performing a “soft delete” where the employee is no longer visible on most screens (still needing somewhere to see them so they can be unarchived) except the history of what jobs the employee has worked on in the past is retained. ✓
- All visible and changeable information in the old system (the whiteboard) must be similarly visible and changeable in the new system including: jobs and their basic info (titles, project manager, address), employees and their basic info (names, roles, notes), where employees are assigned (a specific job, inactive, in school, or just unassigned), the next 3 months of projections and the sum of these projections for each month (sum does not need to be changeable), and employee sums for each job and a total across all jobs (does not need to be changeable) ✓

Description of Proposed Workflow and Methodology

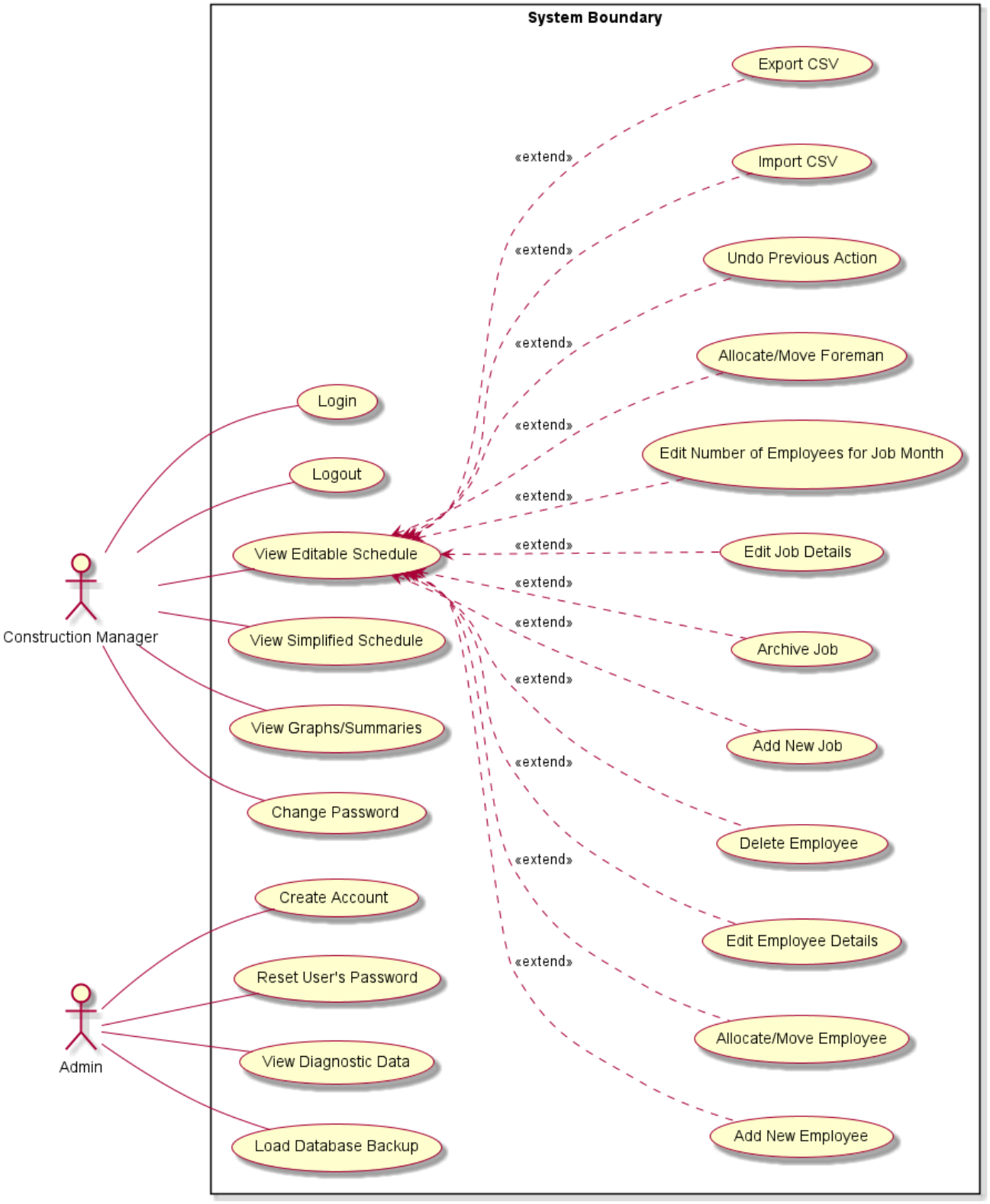
Test driven/Kanban approach.

- Write unit tests first, then write codes to pass tests, then refactor the code.
- Weekly team meeting on Wednesday and Friday at regular scheduled class time (8:30 AM – 11:50 PM).
- Use of a Kanban Board via GitHub projects to track ongoing tasks by team members. Team members self-assign tasks from the backlog but may only pick up to 3 at a time.
- Biweekly client meeting with clients on Wednesday using Microsoft teams.
- The developers will write detailed code comments that help in future maintenance and integrations.
- Team members will be expected to use clockify (or some other means) to record their hours spent on the project. This will be used to update the WBS.
- Team members will be expected to provide a log of everything they had done and worked on, which will be included in our weekly reports.
- If a team member wants to create a feature branch off of another feature branch, their new branch will not be reviewed & merged until the original branch is merged (to avoid re-reviewing the same code). Once the original branch is reviewed and ready to merge, it is the team member’s responsibility to merge any changes from the original branch into their new feature branch (to avoid new changes in the original branch getting overwritten).

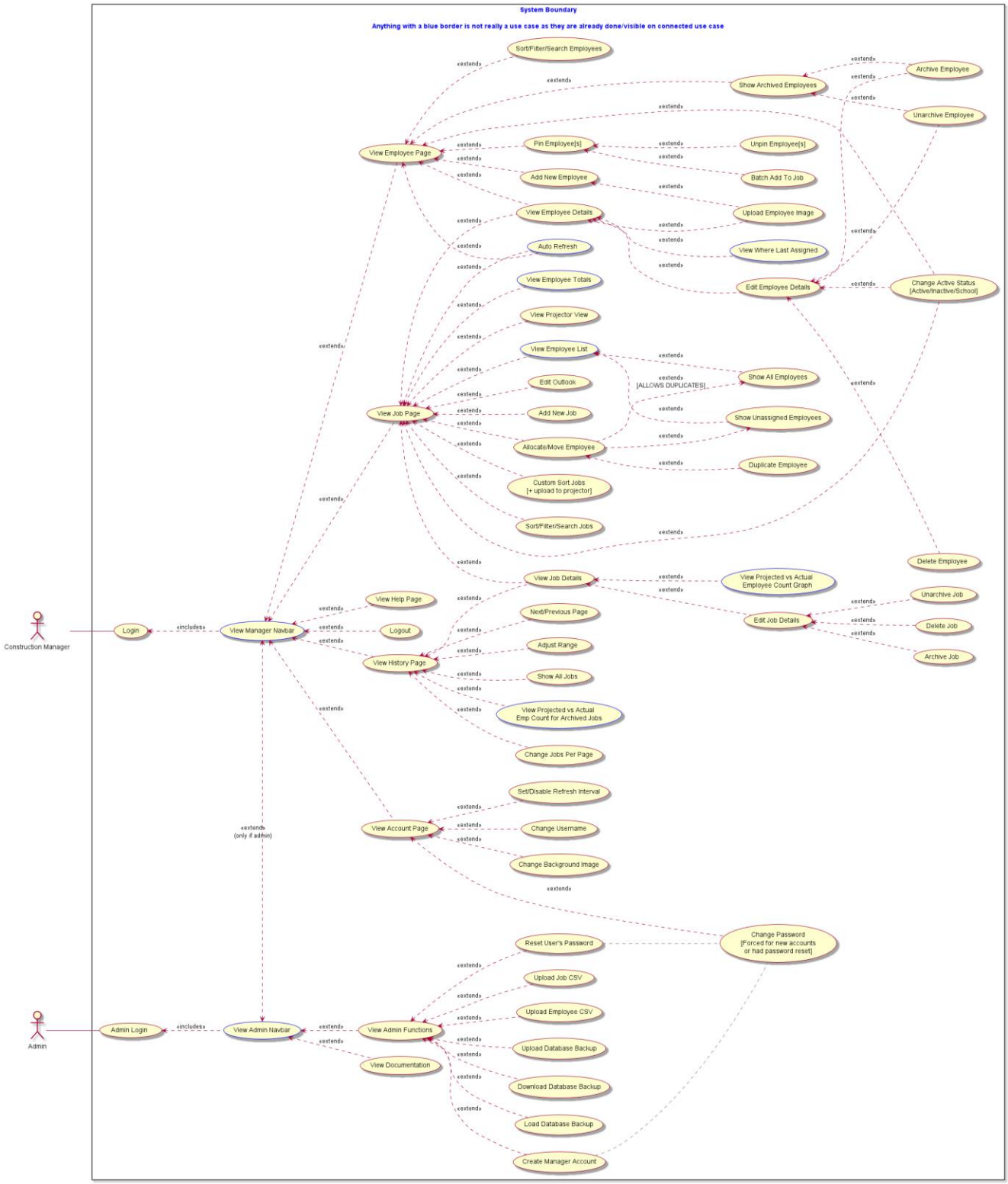
Workflow

- The developers will use git as a version control tool and implement one feature in one branch.
- Once someone has finished passing the tests for their code, they will create a pull request on GitHub. This will automatically run the regressions tests via GitHub actions. Once the tests pass, at least one other team member must thoroughly review the code before merging to the main branch.

User Groups and UML Case Diagrams



Updated use case diagram



Use Case Descriptions on Design Document

Simplified Work / Feature Breakdown Structure

- Document in separate Excel spreadsheet