Horizon Electric Manpower Scheduling Board – Design Document

Introduction:

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."

Requirements:

Functional Requirements

- There must be at least one account with a username and password to ensure that no outside user connected to the network is able to access the system
- 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. Right-click the job that needs to be archived, choose the "archive" option, then the job and its related information will be archived.
    - 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.
- 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.

Non-functional Requirements

- The system must be constructed with the possibility of having multiple accounts in mind.
- The website will be accessible on the following desktop browsers: Google Chrome, Edge Firefox, Safari (?).
- 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 input fields must have both client-side and server-side validation.
- 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 will 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 will 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 10 active clients at one time.
- The system will 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
        - Actual employee counts will 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 [at what time?] on database
        - Can be used for visualisations
    - The history of user actions on the schedule

        - Saved on client
        - Used for undo
        - 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 a low-resolution projector as well as the colour of the backdrop and must be built in a way that all text will be readable on a 4k projector.
-

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.
- 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. Right-click the job that needs to be archived, choose the "archive" option, then the job and its related information will be archived.
    - 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
- 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

Diagrams and Mockups:

User Personas

Primary User

Gregory Thompson - Construction Manager

"Technology hates me."

Age: 50
Education: Bachelor's degree in Civil Engineering
Family: Married with two kids
Location: Kelowna, BC

Personality

Picky — Easy

Analytical — Creative

Introvert — Extrovert

Thinking — Feeling

Excellent Communicator | Problem Solver | Technologically-unsavvy | Worsening-Eyesight | Detail-Oriented | Resilient

Goals

- Complete all projects within a set timeline and budget, while maintaining quality.
- Be able to view at a glance what employees are working at which jobs, see what jobs need more or less employees and assign them accordingly.
- Eliminate tedious, manual updates of counts in their current schedule board, saving time and improving efficiency
- Create better-informed future decisions by being able to view charts and graphs based on the history of jobs and employees.
- Enhance project coordination by identifying critical path activities and potential scheduling risks in advance.

Tasks and Responsibilities

- Oversee projects and schedules to ensure they are completed on time, within budget, and meet all specifications.
- Manage employees so they are all working to the best of their ability while maintaining their safety.
- Collaborate with his fellow-construction managers within the office to assign employees to jobs in a way that satisfies the manpower and experience-level requirements for the job.
- Communicate with architects, engineers, contractors, and other stakeholders to ensure the projects are done to specifications.

Values

Ease-of-Use

Learnability

Memorability

Simplicity

Safety

Bio

Gregory is a hard-working, busy manager, who tries to take great care of his employees while also ensuring all projects are done on time. He typically tries to stay away from technology, as he finds himself frequently getting frustrated by overly-complex systems and their hard-to-navigate UI or inability to tolerate his mistakes. Additionally, it can take him some time for him to learn new things, and he finds himself only using a few features of most apps. When Gregory is not at work, he spends his time on the couch watching shows and spending time with his family.

Secondary User

Alex Mitchell - IT

Age: 27
Family: Married
Education: Bachelor's degree in Computer Science
Location: Kelowna, BC
Character: The IT guy

Bio

Alex spends most of his time carefully listening and helping clients fulfill their needs. He works independently as a contractor to help business with a variety of technical problems. He frequently finds himself frustrated when working on projects that have little documentation, as he finds it very hard to continue where people have left off. In his free time, he enjoys gardening and watching animals.

Technology

IT & Internet
Social Media
Financial Management
Social Networking

Motivation

Learnability
Safety
Ease of Use
Function
Simplicity

Major Responsibilities and Tasks

- Maintaining and troubleshooting the organization's IT infrastructure, including servers, networks, and software systems.
- Managing user accounts, permissions, and access controls.
- Ensuring data security and implementing backup and disaster recovery strategies.
- Providing technical support to end-users, addressing hardware and software issues.
- Researching and recommending new technologies and system upgrades to enhance efficiency and productivity.

Goals

- To have a functional way to access and restore databases and view diagnostic data.
- Upgrade the organization's network infrastructure to improve data transfer speeds and enhance overall system performance.
- To have accessible and readable documentation to understand how a system works so he can more efficiently and effectively perform maintenance.

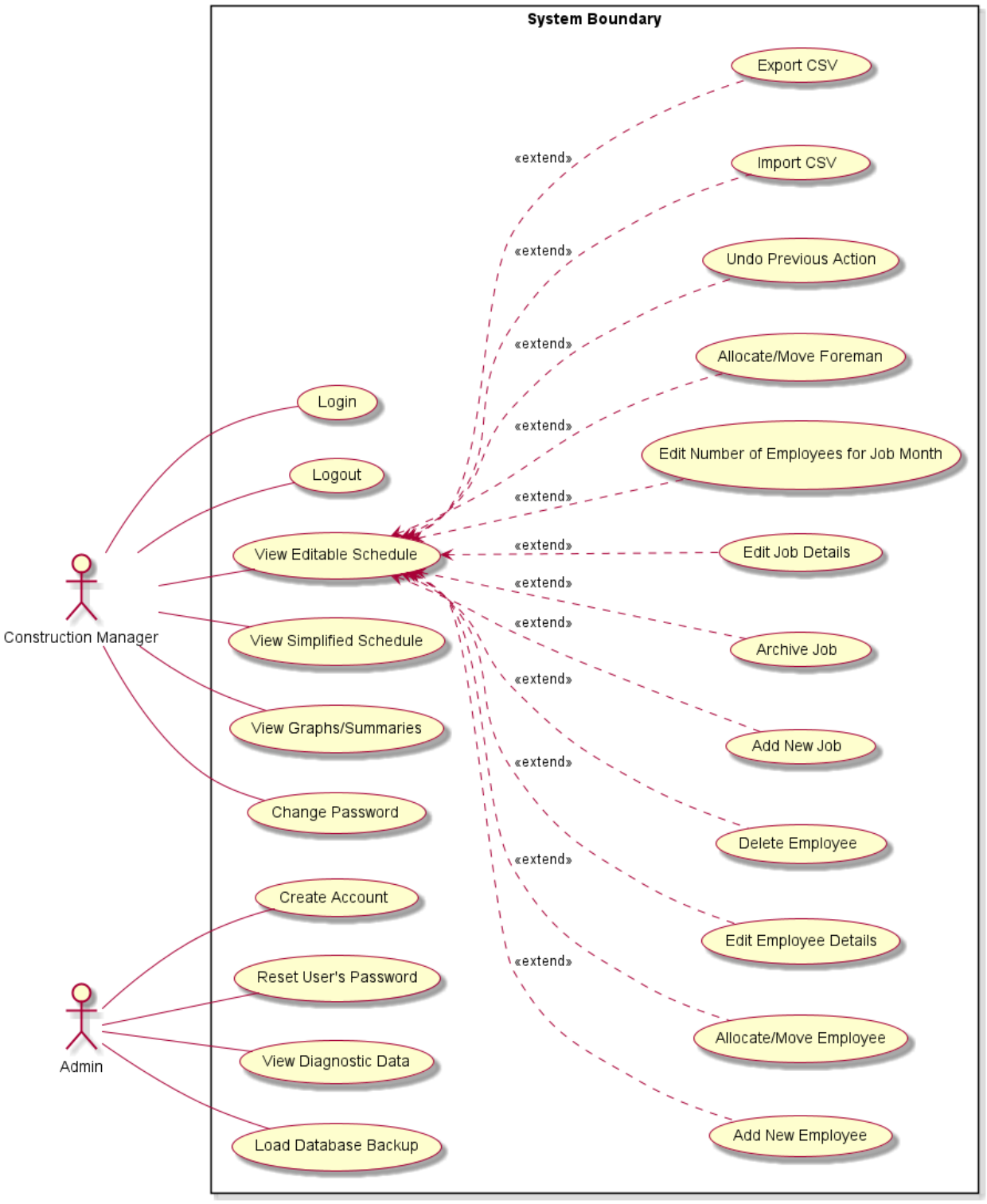
Personality

Introvert — Extrovert
Analytical — Creative
Loyal — Fickle
Passive — Active

Technical | Professional | Dependable

"I can fix that"

Use Case Diagram



Use Case 1: Add New Job

Primary actor: Construction Manager

Description: Allow the user to add a new job/project to the system. New jobs will require the following fields: Job name, start/end date, project manager (can be displayed as separate on the actual field). New jobs will show on the schedule and will display the job name, the project manager, space for employees, as well as displaying areas which contain the expected number of employees needed for the month (adjusted manually). The schedule will have indicators on where the expected start and end month is.

Pre-condition: The user must be logged in to the system (use case 5) and viewing the editable schedule (use case 7).

Post-condition: The new project is successfully added to the system and the schedule will be updated accordingly.

Main scenario:

1. The user selects the "Add New Job/Project" button from the user interface.
2. The system presents a form to prompts the user to enter the name, duration (start month, end month), and the project manager (optional field) of the new job/project.
3. The user fills in the required fields.
4. The user submits the form.
5. The system validates the entered information to ensure it meets any defined constraints.
6. The system adds the new job/project to the system.
7. The system stores the job/project details in the database.
8. The system provides feedback to the user indicating the successful addition of the new job/project, and the displayed schedule will be updated accordingly.

Extensions:

5.1. Validation errors

5.1.1. If the system encounters validation errors during the data entry process, such as missing required fields or conflicting data, it notifies the user and provides details about the encountered issues.

5.1.2. The system prompts the user to correct the errors before resubmitting the form.

7.1. Database update errors

7.1.1. If the system encounters errors while adding the new job/project to the database, such as database connectivity issues or conflicts with existing data, it notifies the user and suggests retrying the operation later.

7.1.2. The system ensures data consistency by rolling back any incomplete or failed updates, maintaining the integrity of the job database.

Use Case 2: Add New Employee

Primary actor: Construction Manager

Description: Allow the user to add a new employee to the system.

Pre-condition: The user must be logged in to the system.

Post-condition: The new employee is successfully added to the system.

Main scenario:

1. The user selects the "Add New Employee " button from the user interface.
2. The system presents a form to prompts the user to enter the details of the new employee.
3. The user fills in the required information for the new employee, such as name, address, journeyman level, and other relevant information.
4. The user submits the form.
5. The system validates the entered information to ensure it meets any defined constraints.
6. The system adds the new employee to the system.
7. The system stores the employee details in the database.
8. The system provides feedback to the user indicating the successful addition of the new employee.

Extensions:

5.1. Validation errors

5.1.1. If the system encounters validation errors during the data entry process, such as missing required fields or conflicting data, it notifies the user and provides details about the encountered issues.

5.1.2. The system prompts the user to correct the errors before resubmitting the form.

6.1. Database update errors

6.1.1. If the system encounters errors while adding the new job/project to the database, such as database connectivity issues or conflicts with existing data, it notifies the user and suggests retrying the operation later.

6.1.2. The system ensures data consistency by rolling back any incomplete or failed updates, maintaining the integrity of the job database.

Use Case 3: Allocate/Move Employee

Primary actor: Construction Manager

Description: Allow the user to easily and quickly move employees from jobs

Pre-condition: The user must be logged in to the system.

Post-condition: The employee is moved visually and in the database.

Main scenario:

1. User enters the edit page
2. User drags employee from job to other job
3. System visually keeps the change.
4. System updates the database with the change

Extensions:

5.1. Database error

5.1.1. Database fails to update with new data due to connection issues or a failed database constraint.

Use Case 4: Edit Number of Employees for the job month.

Primary actor: Construction Manager

Description: Allows the user to edit the number of employees assigned to a specific job for a particular month.

Pre-condition: User must be logged in to the system.

Post-condition: The number of employees assigned to the job for the specified month is successfully updated in the system.

Main scenario:

1. User selects the “Edit Mode” option from the bottom of the user interface.
2. User manually type the number of employees for a job month.
3. The system validates inputs and updates the number of employees allocated for the job month.

Extensions:

2.1. Invalid input format

2.1.1. The user uses a non-integer type as the input. The system will alert the user about the input’s formatting error.

Use Case 5: Log in

Primary actor: Construction Manager

Description: Allow users to log in to their own private account.

Pre-condition: User must have an account.

Post-condition: The user will be logged in to the system.

Main scenario:

1. User provides username and password.
2. User tells system to login.
3. System tells user is successfully logged in

Extensions:

1.1. No Username Given
  1.1.1. The system tells the user that the username must be filled
1.2. No password Given
  1.2.1. The system tells the user the password must be filled

2.

2.1. Pre-validate username & password fails
  2.1.1. Username/password is too long, empty, or has invalid characters and the user is informed of this
2.2. Server/database Timeout
  2.2.1. System tells user the server/database has timed out.
2.3. Incorrect password/username
  2.3.1. System informs user of incorrect username/password
2.4. Too many tries
  2.4.1. After 15 tries, the system will lock for 10 minutes and will inform the user of such

Use Case 6: Log out.

Primary actor: Construction Manager

Description: Allow users to log out from their private account.

Pre-condition: User must be logged in to the system.

Post-condition: The user will be logged out of the system.

Main scenario:

1. User selects the "Log Out" option from the portal.
2. The system prompts the user to confirm the log out.
3. User confirms.
4. The system terminates the user's session and logs the user out.

Extensions:

2.1. User cancels log out.

2.1.1. The user chooses not to log out and cancels the log out process.

2.1.2. The system returns the user to the previous state without logging them out.

Use Case 7: View Editable Schedule

Primary actor: Construction Manager

Description: Allow users to view an editable schedule portal.

Pre-condition: User must be logged in to the system.

Post-condition: The user can view the schedule and make edits if necessary.

Main scenario:

1. User selects the "Edit Mode" option from the bottom of the user interface.
2. The system retrieves the schedule data from the project database.
3. The system displays the schedule on the user interface, presenting tasks, employees, start and end dates, and other relevant details.
4. The user can edit details.
5. The user can save the changes made to the schedule if desired.
6. The system updates the project database with the modified schedule information.
7. The system provides feedback to the user indicating the successful saving of changes.

Extensions:

2.1. Schedule data not available

2.1.1. If there is no schedule data available in the project database, the system notifies the user and suggests creating a new schedule or contacting the appropriate personnel for assistance.

5.1. Invalid edits

5.1.1. If the user attempts to make invalid edits, such as 1 worker doing multiple projects, the system notifies the user and prevents the changes from being saved until the issues are resolved.

7.1. Database update errors

7.1.1. If the system encounters errors while updating the project database with the modified schedule information, such as database connectivity issues or conflicts with existing data, it notifies the user and suggests retrying the save operation later.

Use Case 8: Undo a previous change.

Primary actor: Construction Manager

Description: Allows the user to undo a previous action performed in the system.

Pre-condition: User must be logged in to the system.

Post-condition: The previous action is successfully undone, and the system reverts to the state before the action.

Main scenario:

1. User selects the option to undo a previous action.
2. System identifies the most recent action performed by the user.
3. System reverts the changes made by the identified action.

Extensions:

2.1. No previous changes found.

2.1.1. The system cannot identify any previous change performed by the user. The system turns the undo button grey.

Use Case 9: Allocate/move Foremen.

Primary actor: Construction Manager

Description: Allows the user to allocate or move a foreman to a specific job.

Pre-condition: User must be logged in to the system.

Post-condition: The foreman is successfully allocated/moved to the desired job.

Main scenario:

1. User selects the “Edit Mode” option from the bottom of the user interface.
2. User moves Foremen by dragging and dropping his name to the desired job.
3. The system updates the foremen’s change in position.

Use Case 10: View Simplified Schedule

Primary actor: Construction Manager

Description: Allow users to view a view-only schedule portal

Pre-condition: User must have a server to host the website locally.

Post-condition: The user can view the simplified schedule.

Main scenario:

1. User selects the "View Mode" option from the bottom of the user interface.
2. The system retrieves the schedule data from the project database.
3. The system displays the schedule on the user interface, presenting tasks, employees, start and end dates, and other relevant details.
5. The user can scroll through the schedule to view different time periods and different projects.

Extensions:

2.1. Schedule data not available

2.1.1. If there is no schedule data available in the project database, the system notifies the user and suggests creating a new schedule or contacting the appropriate personnel for assistance.

Use Case 11: Edit Employee Details

Primary actor: Construction Manager

Description: Allow the user to edit employee details in the system. The user will be able to change the employee's name, their experience level (5 (?) levels, journeyman, something, something, something), red seal, and whether or not they are active

Pre-condition: The user must be logged in to the system (use case 5) and viewing the editable schedule (use case 7).

Post-condition: The employee details are successfully updated in the system. If the employee is changed from active to inactive (or vise-versa) or if the employee's experience level is changed, the system displays as such.

Main scenario:

1. The user selects the employee whose details they want to edit.
2. The system displays the current details of the selected employee, such as name, address, whether or not they are active, experience level, red seal status, and other relevant information.
3. The user makes the necessary edits to the employee's details.
4. The user saves the changes.
5. The system validates the updated information to ensure it meets any defined constraints.
6. The system updates the employee's details in the system.
7. The system provides feedback to the user indicating the successful update of the employee's details.

Extensions:

5.1. Validation errors

5.1.1. If the system encounters validation errors during the update process, such as invalid information format or conflicting data, it notifies the user and provides details about the encountered issues.

6.1. Database update errors

6.1.1. If the system encounters errors while updating the employee's details in the database, such as database connectivity issues or conflicts with existing data, it notifies the user and suggests retrying the update later.

Use Case 12: Edit Job Details

Primary actor: Construction Manager

Description: Allow the user to edit job/project details in the system.

Pre-condition: The user must be logged in to the system.

Post-condition: The job/project details are successfully updated in the system.

Main scenario:

1. The user selects the job/project whose details they want to edit.
2. The system displays the current details of the selected job/project, such as name, start and end date, and other relevant information.
3. The user makes the necessary edits to the job's/project's details.
4. The user saves the changes.
5. The system validates the updated information to ensure it meets any defined constraints.
6. The system updates the job's/project's details in the system.
7. The system provides feedback to the user indicating the successful update of the job's/project's details.

Extensions:

5.1. Validation errors

5.1.1. If the system encounters validation errors during the update process, such as invalid information format or conflicting data, it notifies the user and provides details about the encountered issues.

6.1. Database update errors

6.1.1. If the system encounters errors while updating the employee's details in the database, such as database connectivity issues or conflicts with existing data, it notifies the user and suggests retrying the update later.

Use Case 13: View Graph/Summaries

Primary actor: Construction Manager

Description: Visual display of analytics related to jobs and previous employment numbers

Pre-condition: The user must be logged in to the system.

Post-condition: The user can view the analytics

Main scenario:

1. The user selects the analytics page of the software
2. User selects which graph or summary they wish to view
3. The system populates that summary/graph

Use Case 14: Archive Job

Primary actor: Construction Manager

Description: Allow the user to remove jobs from the active panel

Pre-condition: The user must be logged in to the system.

Post-condition: The job is removed from the main page and added to the archive page, employees assigned to the job are added to the unassigned tab

Main scenario:

1. User enters the edit page
2. User selects the archive option for the job
3. System prompts the user if they are sure
4. If the user selects yes the system visually removes the job from the page
5. Employees are visually moved to the unassigned tab
6. The system moves the job to the archived table

Extensions:

7.1. Database error

7.1.1. Database fails to update with new data due to connection issues or a failed database constraint.

Use Case 15: Delete Employee

Primary actor: Construction Manager

Description: Allow the user to remove employees permanently from the job page (terminated)

Pre-condition: The user must be logged in to the system.

Post-condition: The employee only shows in the archive not in the active job page.

Main scenario:

1. User enters the edit page
2. User selects the edit option for an employee
3. User selects the delete employee option
4. The System prompts the user if they are sure
5. Employee is removed from the page
6. The system sets the employee to terminated in the database

Extensions:

7.1. Database error

7.1.1. Database fails to update with new data due to connection issues or a failed database constraint.

Use Case 16: Import CSV file.

Primary actor: Construction Manager

Description: Allows the user to upload a CSV file containing relevant data.

Pre-condition: User must be logged in to the system.

Post-condition: The CSV file is successfully uploaded and processed by the system.

Main scenario:

1. User selects the option to upload a CSV file.
2. User chooses the desired CSV file from their local machine.
3. User confirms the file selection.
4. System validates the file format and content.
5. System processes the CSV file and updates the relevant data in the system.

Extensions:

2.1. Invalid file format

2.1.1. The user selects a file that is not in CSV format. The system notifies the user about the invalid file format.

2.2. Empty file

2.2.1. The user selects a CSV file that is empty. The system notifies the user that the file is empty.

2.3. Data validation failure

2.3.1. The system identifies errors or inconsistencies in the CSV file data. The system provides error messages or prompts the user to correct the data before proceeding.

Use Case 17: Export CSV file.

Primary actor: Construction Manager

Description: Allows the user to export a CSV file containing relevant job and/or employee data.

Pre-condition: User must be logged in to the system.

Post-condition: The CSV file is successfully exported by the system and downloaded to the client machine.

Main scenario:

1. User selects the option to export a CSV file from the job or employee screen.
2. User selects what data to export.
3. System processes database data and generates CSV file.
4. System sends generated file to client machine.
5. Client machine downloads file.

Extensions:

3.1. No data

3.1.1. There is no data to be exported on the current screen.

Use Case 18: Undo Action

Primary Actor: Construction Manager

Description: The Undo Action function allows the user to reverse the most recent action performed within the system.

Pre-condition: User must be logged in to the system.

Post-condition: The most recent action is successfully reversed, and the system returns to the state prior to the action.

Main Scenario:

1. User performs an action within the system.
2. User realizes that the action was incorrect or unintended and decides to undo it.
3. User navigates to the "Undo" option within the system's interface.
4. System identifies the most recent action performed by the user.
5. System reverts the effects of the action, restoring the data and state of the system to the state prior to the action.
6. System confirms the successful undo operation to the user.

Extensions:

4.1 No action to undo

4.1.1 The system determines that there are no previous actions to undo.

Use Case 19: Create Account

Primary actor: admin

Description: Allow admin to create a new account in the system.

Pre-condition: admin must already log in to create an account

Post-condition: a new user's account is successfully created in the system.

Main scenario:

1. Admin logs into the main panel.
2. Press the "register" button.
3. Provide the necessary details for creating an account, such as name, username, password in the popup window.
4. The system validates the entered information to ensure it meets any defined constraints.
5. The system checks for username availability and uniqueness.
6. The system creates a new account
7. The system securely stores the account information.
8. The system provides feedback to the user indicating the successful creation of the account

Extensions:

4.1 Validation errors

4.1.1 If the system encounters validation errors during the registration process, such as missing required fields or conflicting data, it notifies the user and provides details about the encountered issues.

4.1.2 The system prompts the user to correct the errors before resubmitting the form.

5.1 Username availability

4.1.3 If the chosen username is already taken or fails to meet the availability criteria, the system notifies the user and prompts them to choose a different username.

Use Case 20: View Diagnostic Data (Admin)

Primary Actor: Admin

Description: The View Diagnostic Data function allows the admin to access and analyze diagnostic information related to the system's performance and health.

Pre-condition: Admin must be logged in to the system.

Post-condition: The admin successfully views the diagnostic data and gains insights into the system's performance and health.

Main Scenario:

1. Admin navigates to the "Diagnostic Data" section
2. Admin selects the desired diagnostic data category or specific metrics to view
3. System retrieves the requested diagnostic data from the relevant sources or logs.
4. System presents the diagnostic data in a readable format, such as tables, graphs, or reports.
5. Admin completes the diagnostic data analysis and exits the diagnostic data view.

Use Case 21: Load Database Backup

Primary Actor: Construction Manager

Description: The Load Database Backup function allows the user to load a backup of the database from a CSV file.

Pre-condition: User must be logged in to the system and they must have an existing csv backup file

Post-condition: The database is successfully restored from the backup CSV file.

Main Scenario:

1. User decides to load a database backup.
2. User navigates to the “Settings” tab within the system’s interface and selects the “Load Database Backup” option.
3. System prompts the user to select the backup CSV file.
4. User selects the backup CSV file.
5. System loads the data from the CSV file and restores the database to its state at the time of the backup.
6. System confirms the successful database restoration to the user.

Extensions:

4.1 Invalid CSV file

4.1.1 The system determines that the selected CSV file is invalid and prompts the user to select a valid backup CSV file.

Use Case 22: Change Password

Primary Actor: Construction Manager

Description: The Change Password function allows the user to change their password within the system.

Pre-condition: User must be logged in to the system.

Post-condition: The user's password is successfully changed, and the system updates its records accordingly.

Main Scenario:

1. User decides to change their password.
2. User navigates to the "Change Password" option within the system's interface.
3. System prompts the user to enter their current password and new password.
4. User enters their current password and new password.
5. System verifies that the current password is correct and updates the user's password to the new one.
6. System confirms the successful password change to the user.

Extensions:

5.1 Incorrect current password

5.1.1 The system determines that the entered current password is incorrect and prompts the user to try again.

5.2 Password does not meet requirements

5.2.1 The system detects that the password does not meet complexity requirements and prompts the user to choose a different password

Use Case 23: Reset User Password

Primary Actor: Admin

Description: The Reset User Password function allows the admin to reset the password of a user within the system.

Pre-condition: Admin must be logged in to the system.

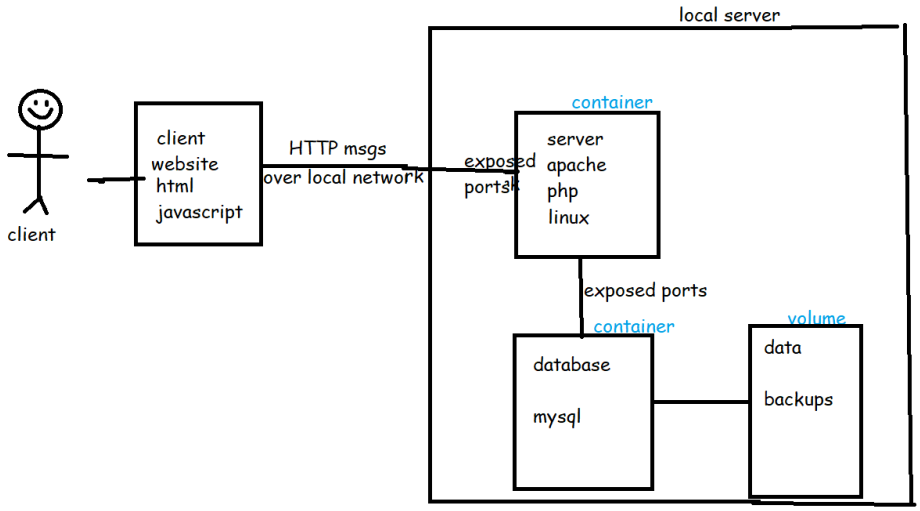
Post-condition: The selected user's password is successfully reset, and the system updates its records accordingly.

Main Scenario:

1. Admin decides to reset a user's password.
2. Admin navigates to the "Settings" tab within the system's interface and selects a user.
3. Admin clicks the "Reset" button next to the selected user's profile.
4. System resets the selected user's password to a default value.
5. System confirms the successful password reset to the admin.

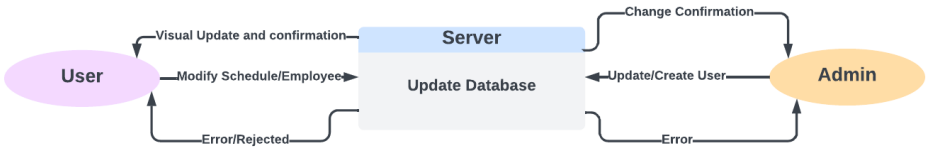
System Architecture Diagrams:

Architecture Diagram

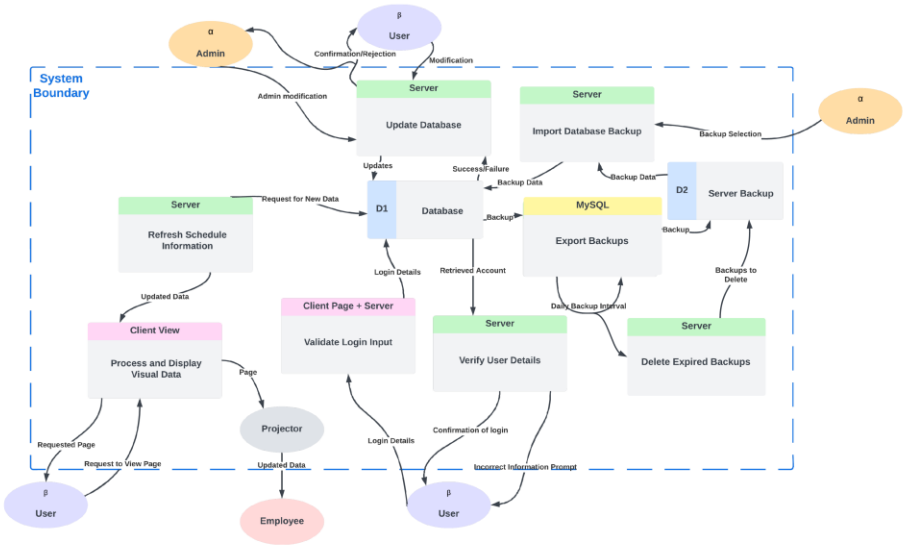


Data Flow Diagrams

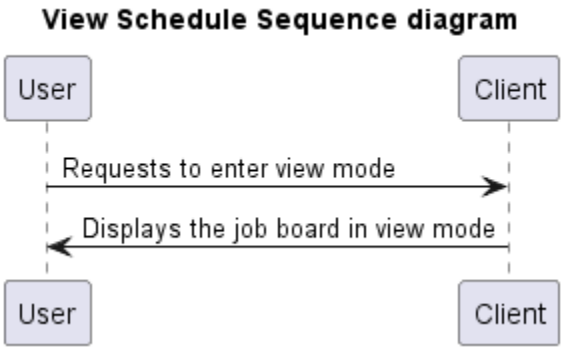
Level 0:

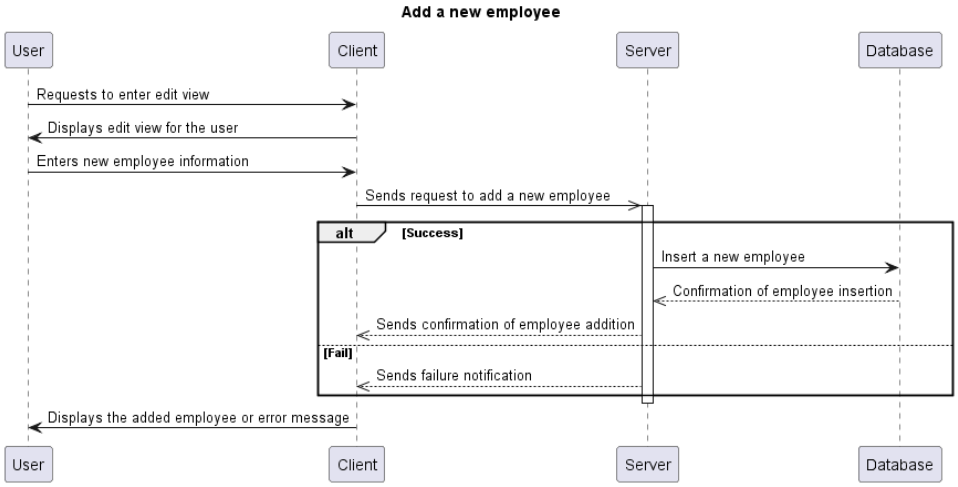


Level 1:

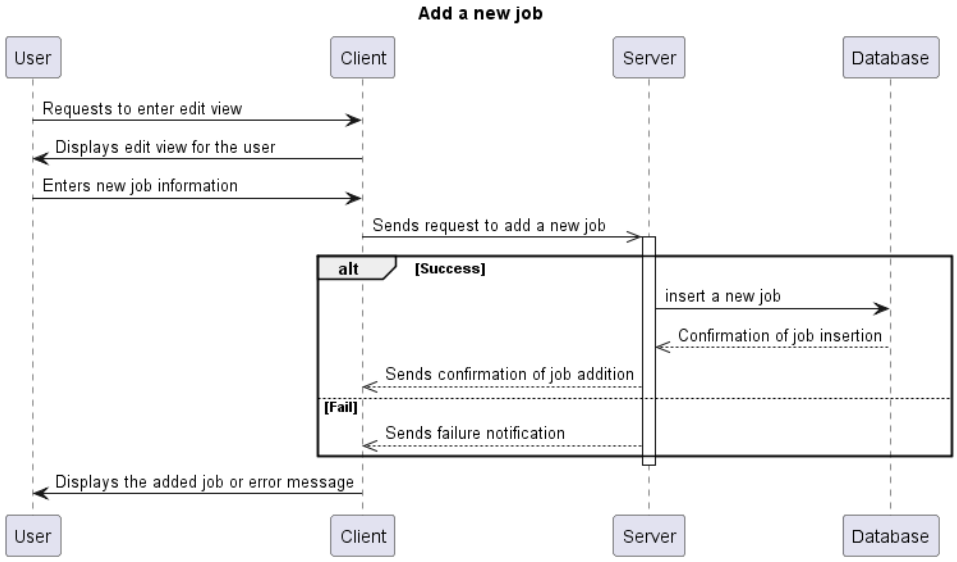


Sequence Diagrams

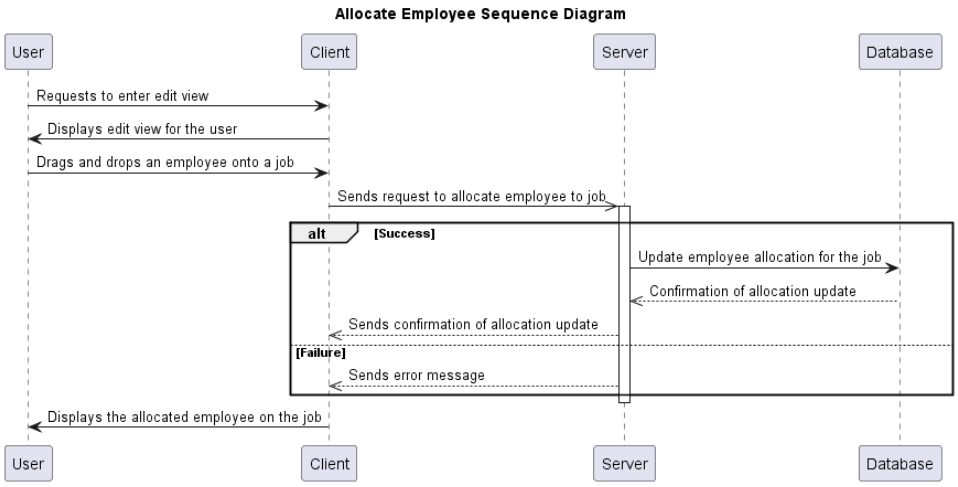




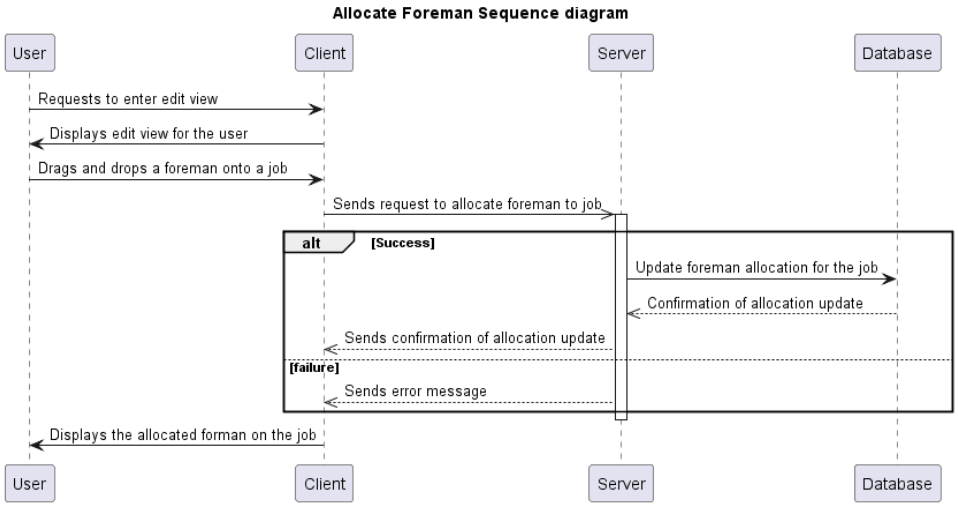
- User requests to enter the edit view to add a new employee.
- Client displays the edit view for the user to input the new employee information.
- User enters the new employee details through the client interface.
- Client sends a request to the server, containing the new employee information.
- Server processes the request, interacting with the database to insert the new employee. If successful, the server sends a confirmation message to the client; otherwise, a failure notification is sent. The client displays the appropriate response to the user.



- The user requests to enter the edit view in order to add a new job.
- The client interface displays the edit view, enabling the user to input the details of the new job.
- The user enters the new job information through the client interface.
- The client sends a request to the server containing the new job details.
- The server processes the request and interacts with the database to insert the new job. If successful, a confirmation message is sent to the client; otherwise, a failure notification is sent. The client interface then displays the appropriate response to the user, showing either the added job's details or an error message.

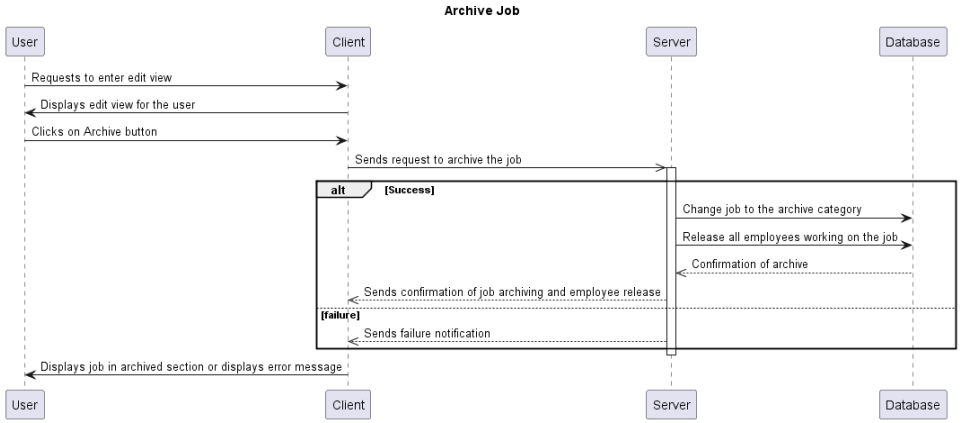


- The user requests to enter the edit view to allocate an employee to a job.
- The client interface displays the edit view, allowing the user to interact with the allocation feature.
- The user performs a drag-and-drop action to allocate a specific employee onto a job.
- The client sends a request to the server, containing the employee and job details for allocation.
- The server processes the request, updating the employee allocation in the database. If successful, a confirmation message is sent to the client; otherwise, an error message is sent. The client interface then displays the allocated employee on the job to the user.

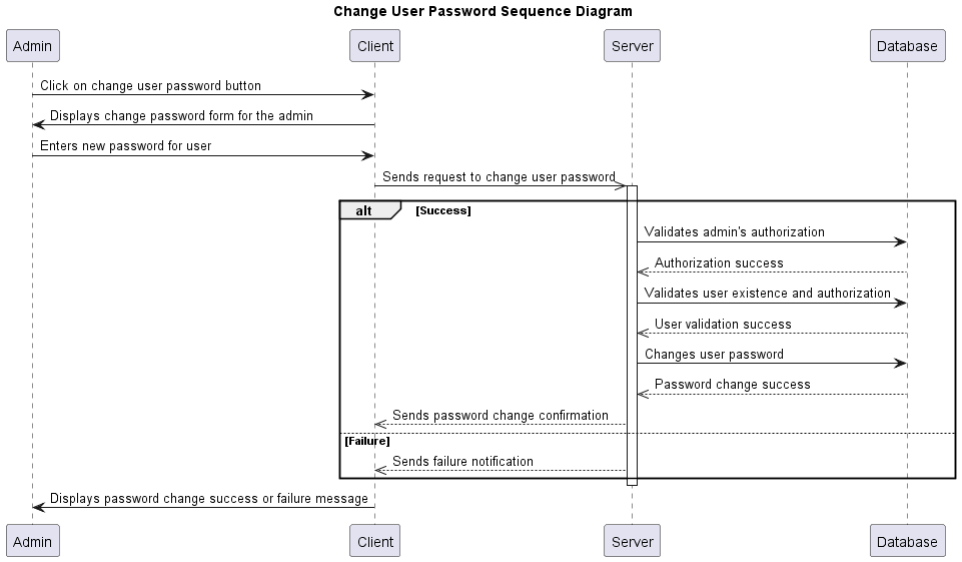


- User requests to enter the edit view to allocate a foreman to a job.
- Client displays the edit view for the user to interact with.
- User performs a drag-and-drop action to allocate a foreman to a specific job.
- Client sends a request to the server, including the foreman and job details for allocation.

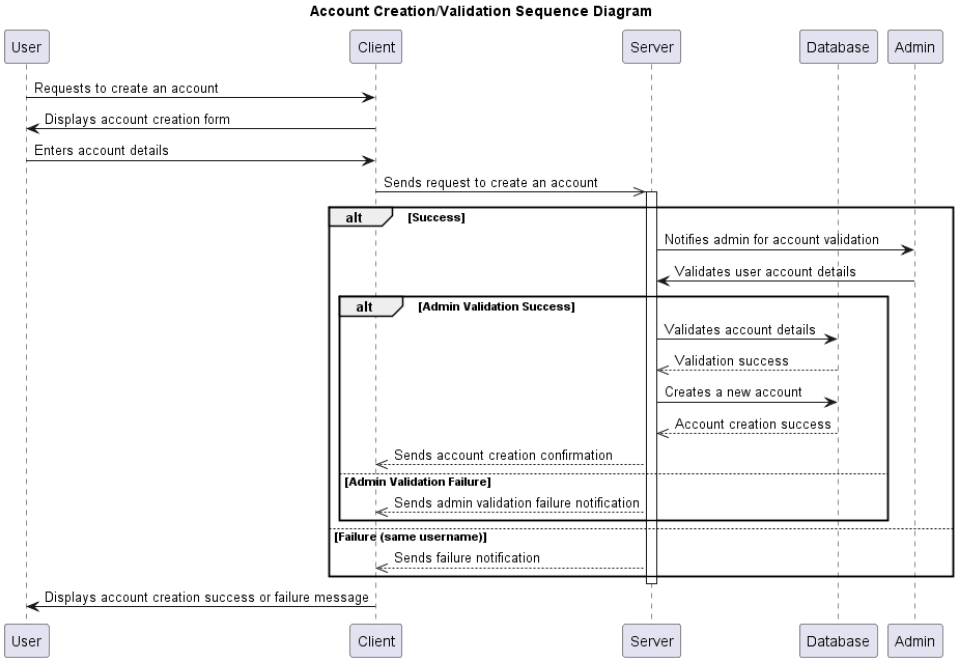
- Server processes the request, updating the foreman allocation in the database. If successful, a confirmation message is sent to the client; otherwise, an error message is sent. The client interface then displays the allocated foreman on the job to the user.



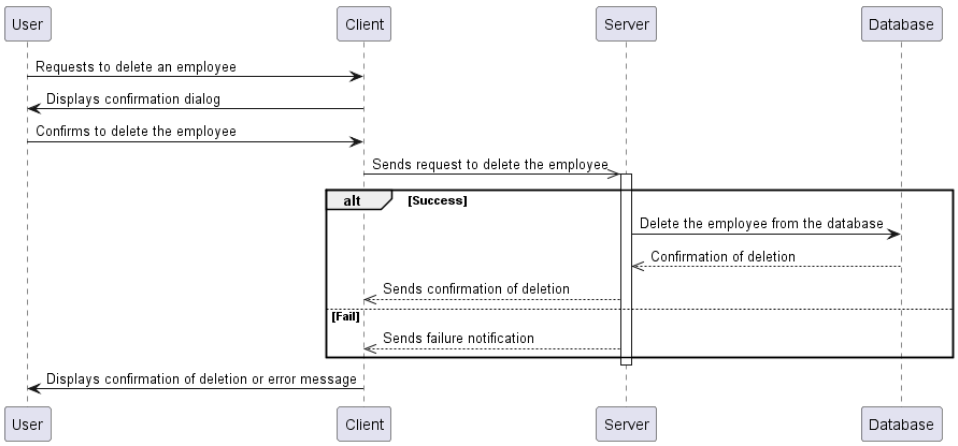
- User requests to enter the edit view to archive a job.
- Client interface displays the edit view for the user to interact with.
- User clicks on the Archive button to initiate the job archiving process.
- Client sends a request to the server, instructing to archive the job.
- Server processes the request, changing the job's category to "archive" in the database and releasing all employees associated with the job. If successful, a confirmation message is sent to the client; otherwise, a failure notification is sent. The client interface then displays the job in the archived section or shows an error message to the user.



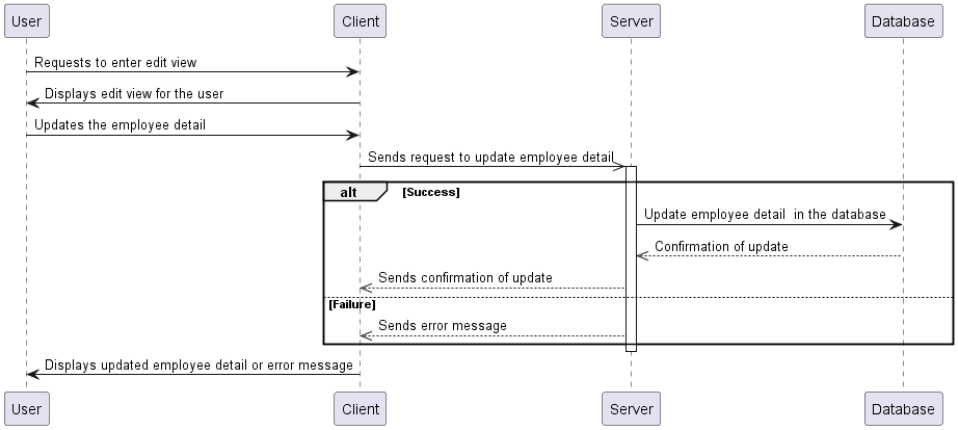
- Admin initiates the process by clicking on the "change user password" button.
- Client interface displays a password change form for the admin.
- Admin enters a new password for the user through the client interface.
- Client sends a request to the server, aiming to change the user's password.
- Server validates the admin's authorization, then checks the existence and authorization of the user. If successful, the server changes the user's password in the database and sends a confirmation message to the client. If there are any failures during the process, the server sends a failure notification. The client interface displays the appropriate password change success or failure message to the admin.



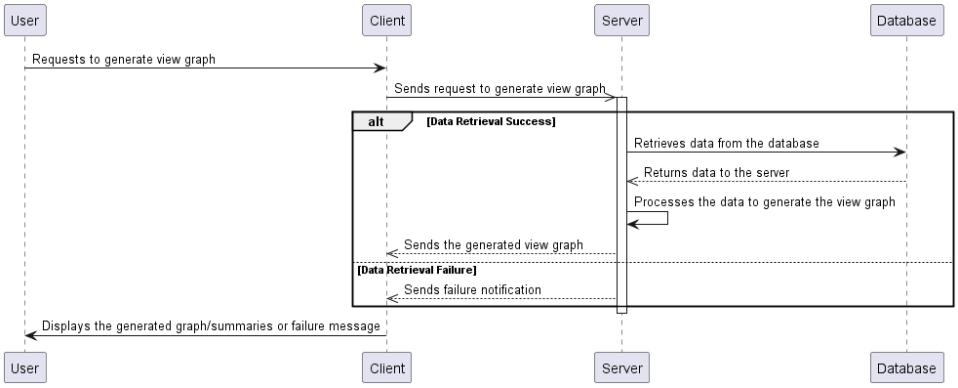
Delete Employee Sequence Diagram

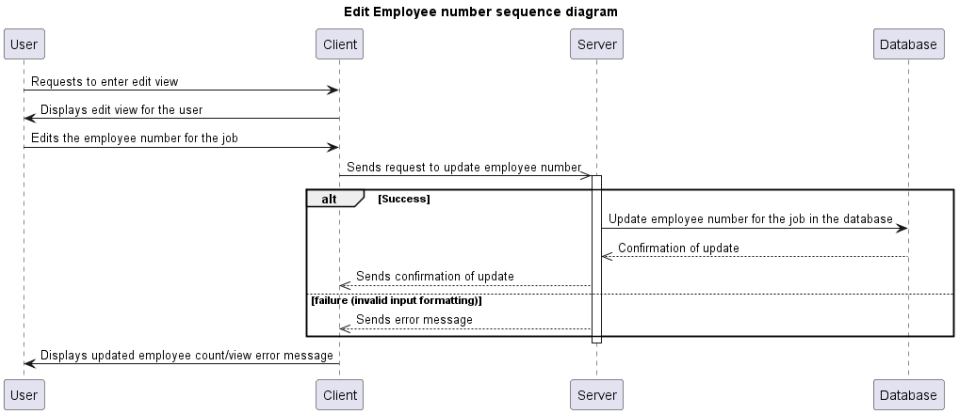


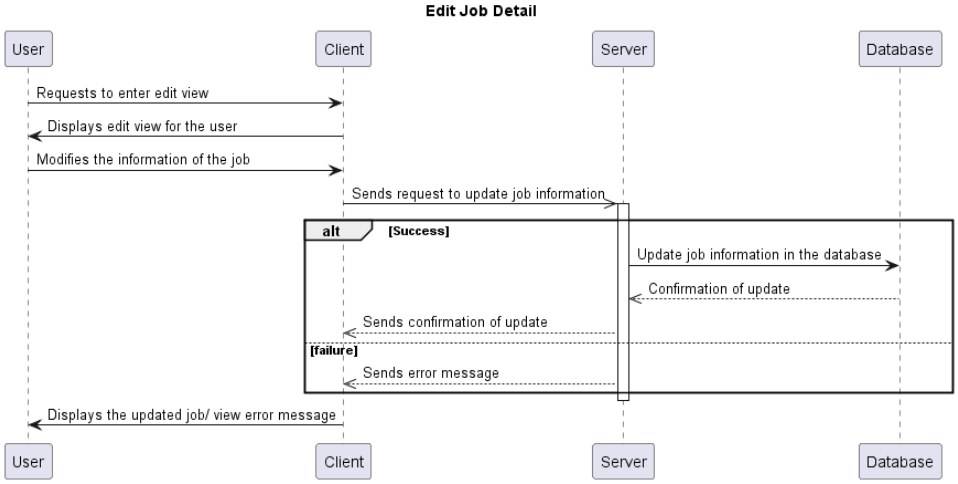
Edit Employee Detail Sequence Diagram

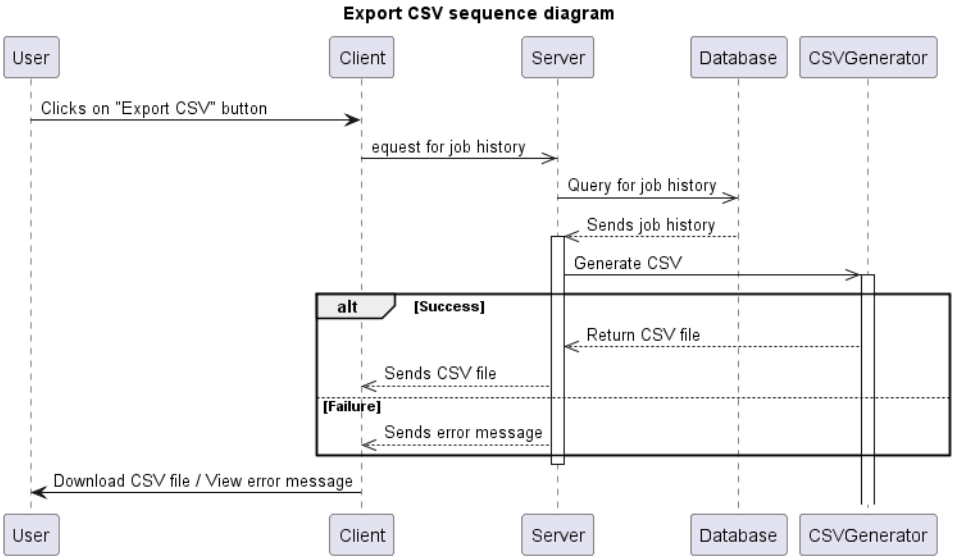


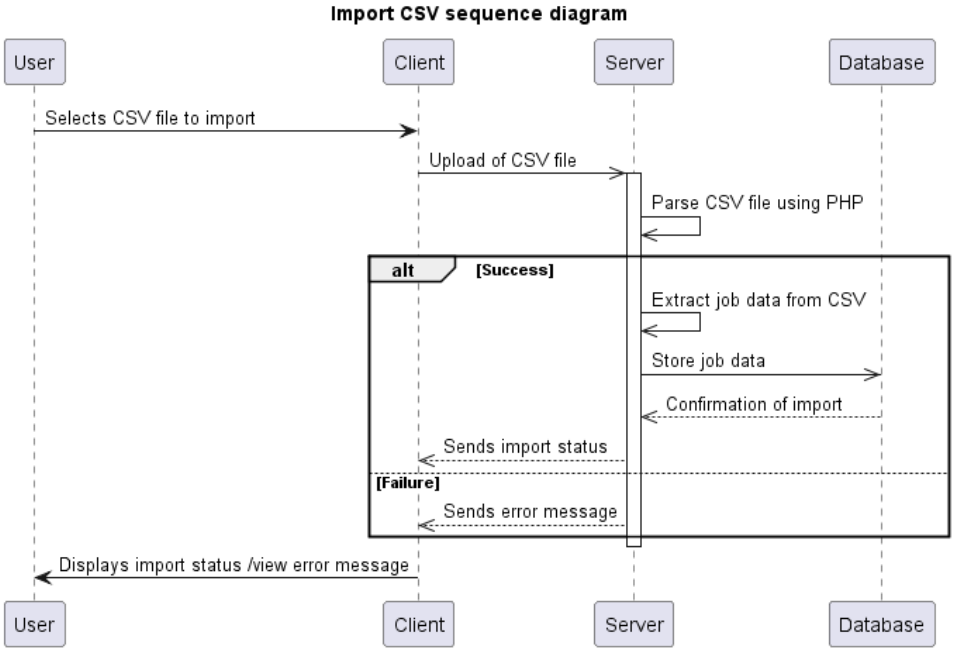
Generate View Graph/Summaries Sequence Diagram

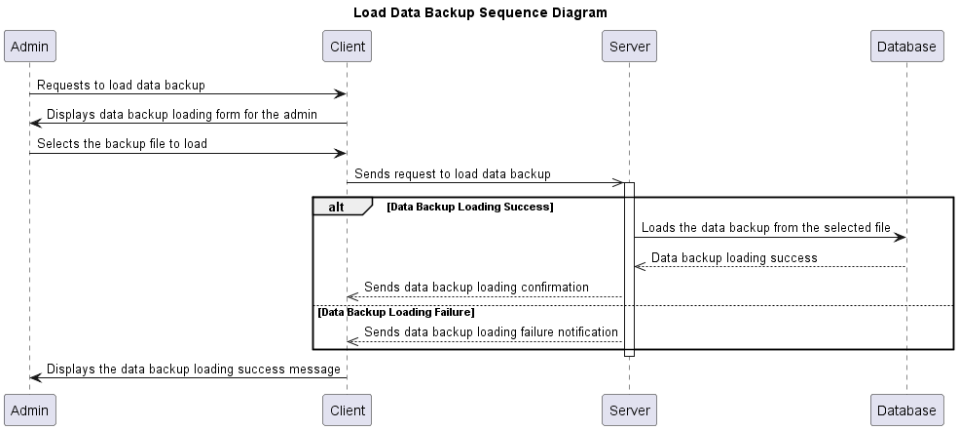


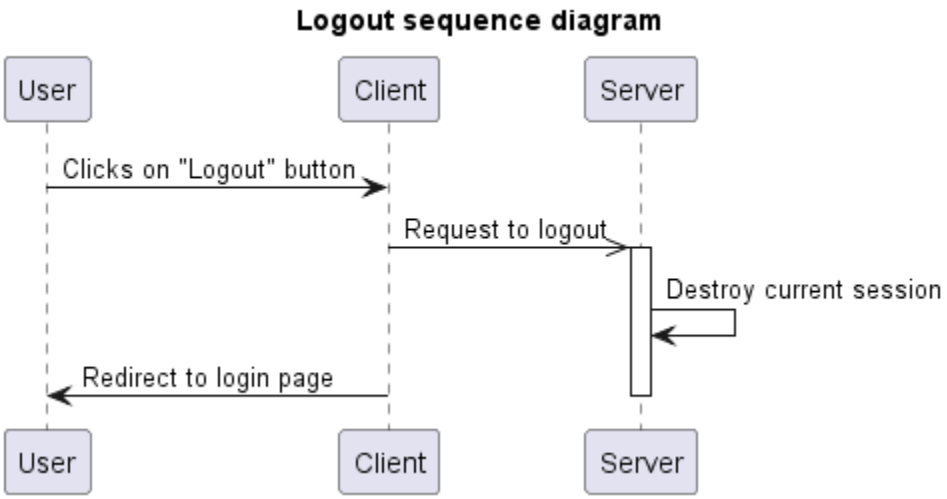

Edit Employee number sequence diagram
User
Client
Server
Database
Requests to enter edit view
Displays edit view for the user
Edits the employee number for the job
Sends request to update employee number
alt
[Success]
Update employee number for the job in the database
Confirmation of update
Sends confirmation of update
[failure (invalid input formatting)]
Sends error message
Displays updated employee count/view error message
User
Client
Server
Database

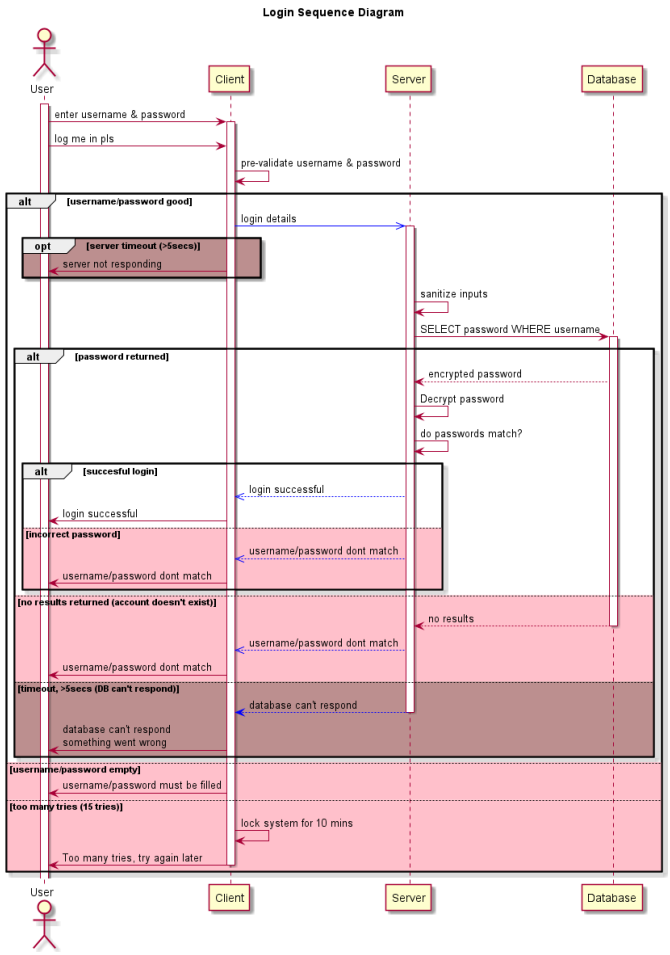

Edit Job Detail
User
Client
Server
Database
Requests to enter edit view
Displays edit view for the user
Modifies the information of the job
Sends request to update job information
alt
[Success]
Update job information in the database
Confirmation of update
Sends confirmation of update
[failure]
Sends error message
Displays the updated job/ view error message
User
Client
Server
Database

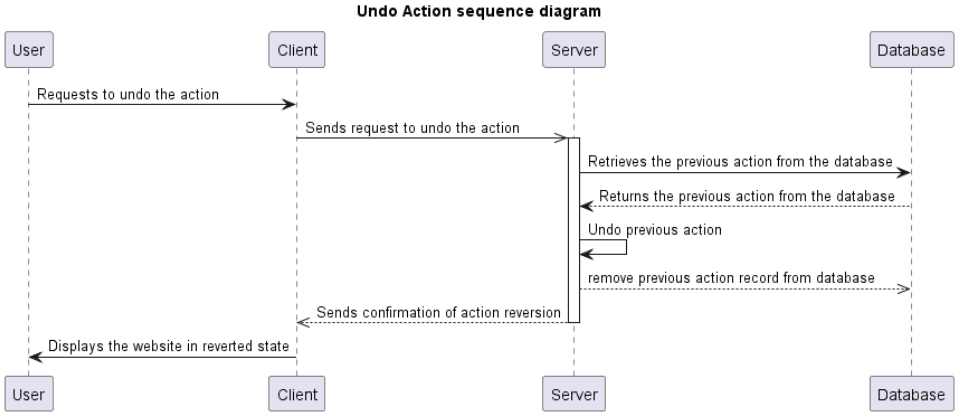

Export CSV sequence diagram
User
Client
Server
Database
CSVGenerator
Clicks on "Export CSV" button
equest for job history
Query for job history
Sends job history
Generate CSV
alt
[Success]
Return CSV file
Sends CSV file
[Failure]
Sends error message
Download CSV file / View error message
User
Client
Server
Database
CSVGenerator

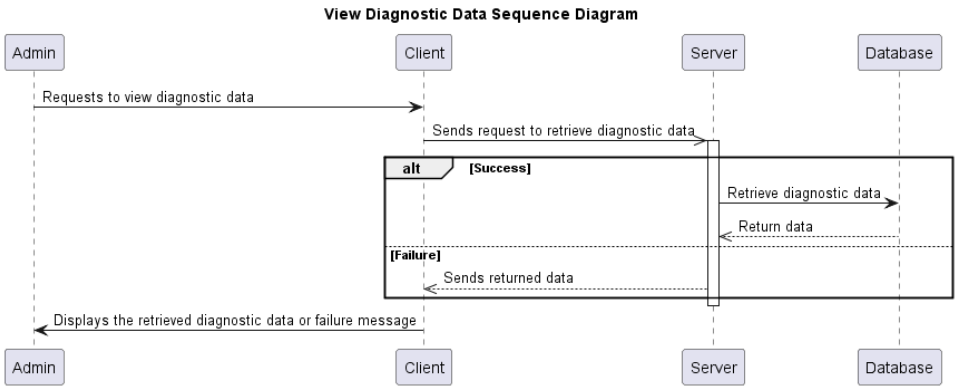

Import CSV sequence diagram
User
Client
Server
Database
Selects CSV file to import
Upload of CSV file
Parse CSV file using PHP
alt
[Success]
Extract job data from CSV
Store job data
Confirmation of import
Sends import status
[Failure]
Sends error message
Displays import status /view error message
User
Client
Server
Database


Load Data Backup Sequence Diagram
Admin
Client
Server
Database
Requests to load data backup
Displays data backup loading form for the admin
Selects the backup file to load
Sends request to load data backup
alt
[Data Backup Loading Success]
Loads the data backup from the selected file
Data backup loading success
Sends data backup loading confirmation
[Data Backup Loading Failure]
Sends data backup loading failure notification
Displays the data backup loading success message
Admin
Client
Server
Database


Logout sequence diagram
User
Client
Server
Clicks on "Logout" button
Request to logout
Destroy current session
Redirect to login page
User
Client
Server


Login Sequence Diagram
User
Client
Server
Database
enter username & password
log me in pls
pre-validate username & password
alt
[username/password good]
login details
opt
[server timeout (>5secs)]
server not responding
sanitize inputs
SELECT password WHERE username
alt
[password returned]
encrypted password
Decrypt password
do passwords match?
alt
[succesful login]
login successful
login successful
[incorrect password]
username/password dont match
username/password dont match
[no results returned (account doesn't exist)]
no results
username/password dont match
username/password dont match
[timeout, >5secs (DB can't respond)]
database can't respond
database can't respond
something went wrong
[username/password empty]
username/password must be filled
[too many tries (15 tries)]
lock system for 10 mins
Too many tries, try again later
User
Client
Server
Database

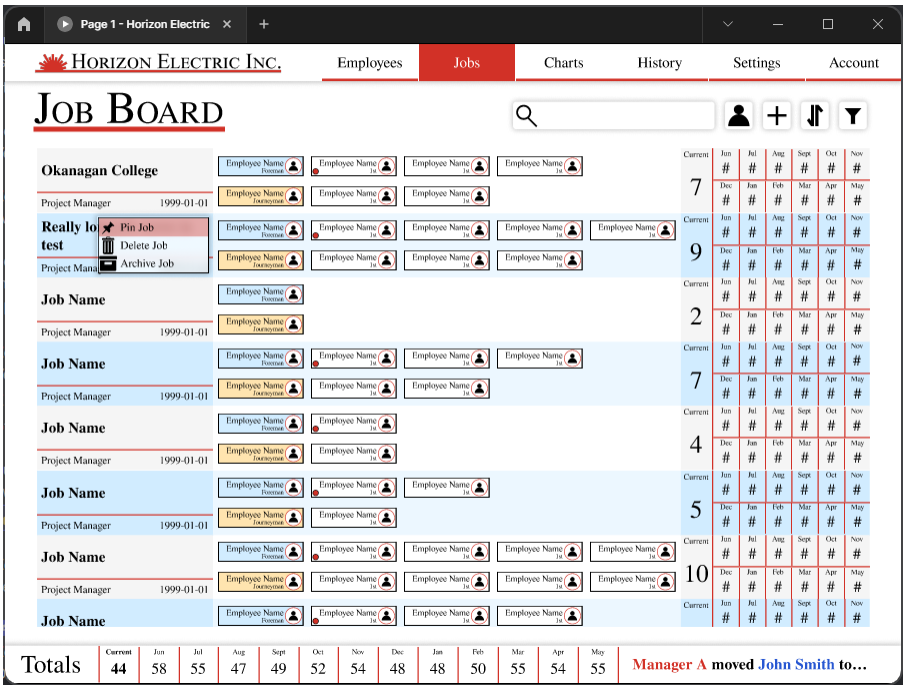

Undo Action sequence diagram
User
Client
Server
Database
Requests to undo the action
Sends request to undo the action
Retrieves the previous action from the database
Returns the previous action from the database
Undo previous action
remove previous action record from database
Sends confirmation of action reversion
Displays the website in reverted state
User
Client
Server
Database



System Specific Documentation

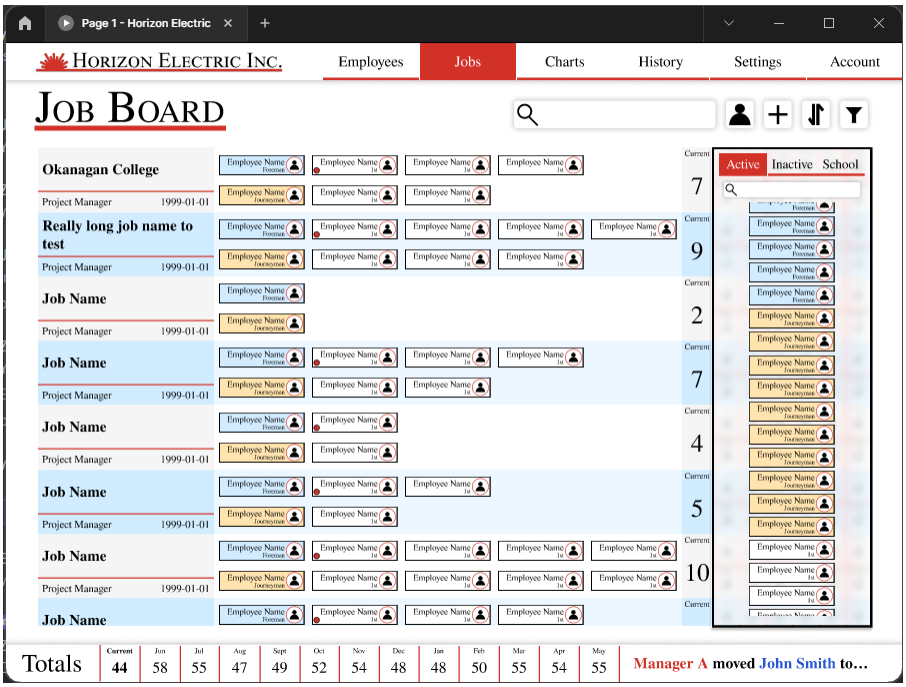
This system will be hosted within the Client's local network using Docker or similar container-based software. The front end will utilize HTML, CSS, JavaScript as well as frameworks such as jQuery. The backend will utilize PHP, alongside MySQL for database integration.

UI mockups

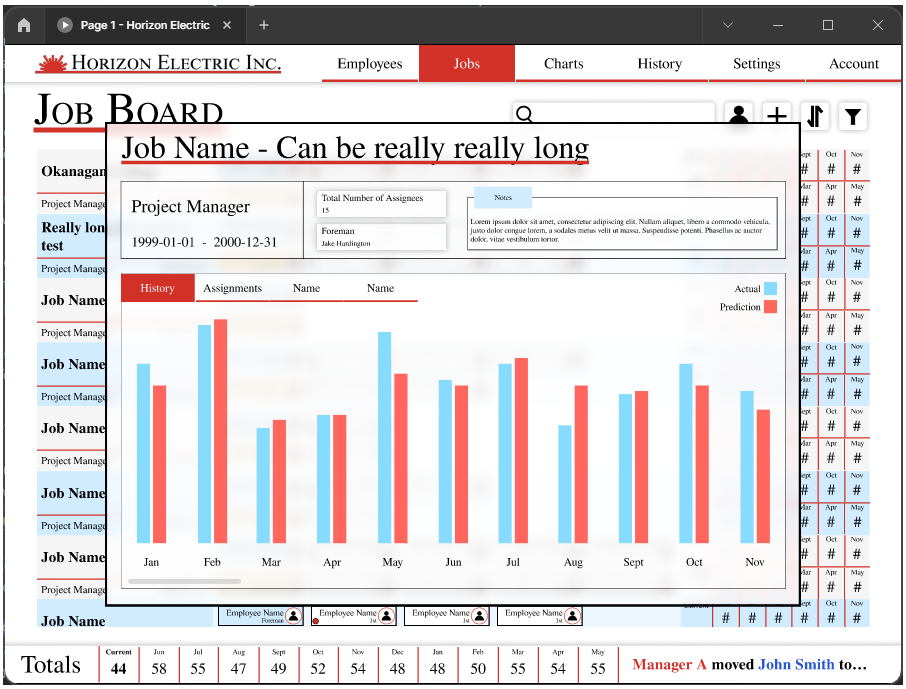


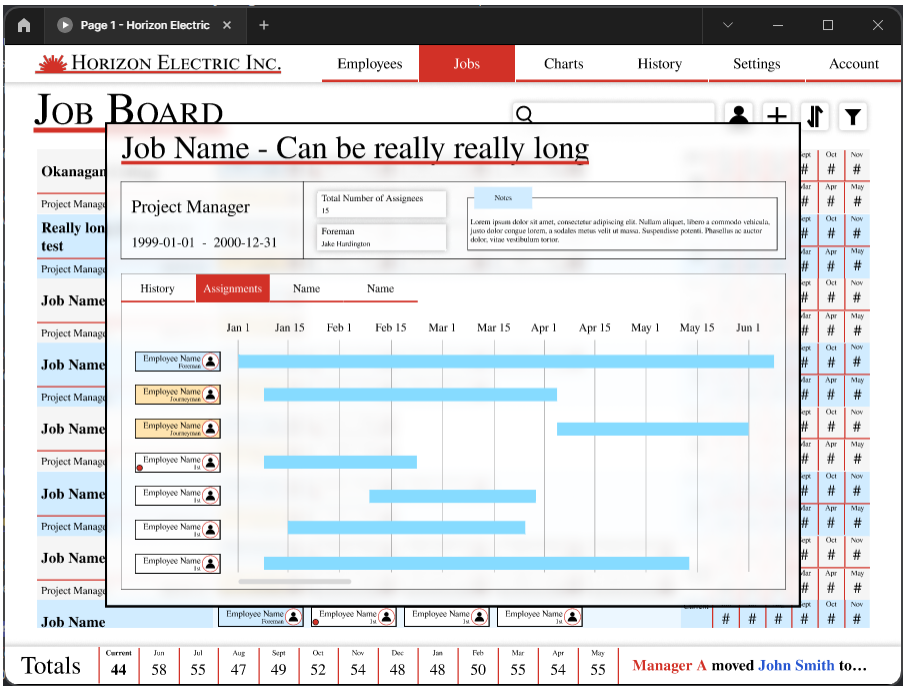
- Edit view - job screen.
- Top and bottom bars are fixed. Middle section scrollable depending on length of job list.
- Current number of employees assigned to a job automatically calculated.
- Future predictions of employee numbers viewable on right side of job listing and editable with a click.

- Current employee names listed in middle section of each job listing. These employee boxes can be dragged and dropped to other jobs.
- Right clicking a job listing allows for the job to be pinned to top of screen for easy access. Jobs can also be deleted or archived this way.
- Total number of current and predicted future employees automatically calculated at bottom of screen.

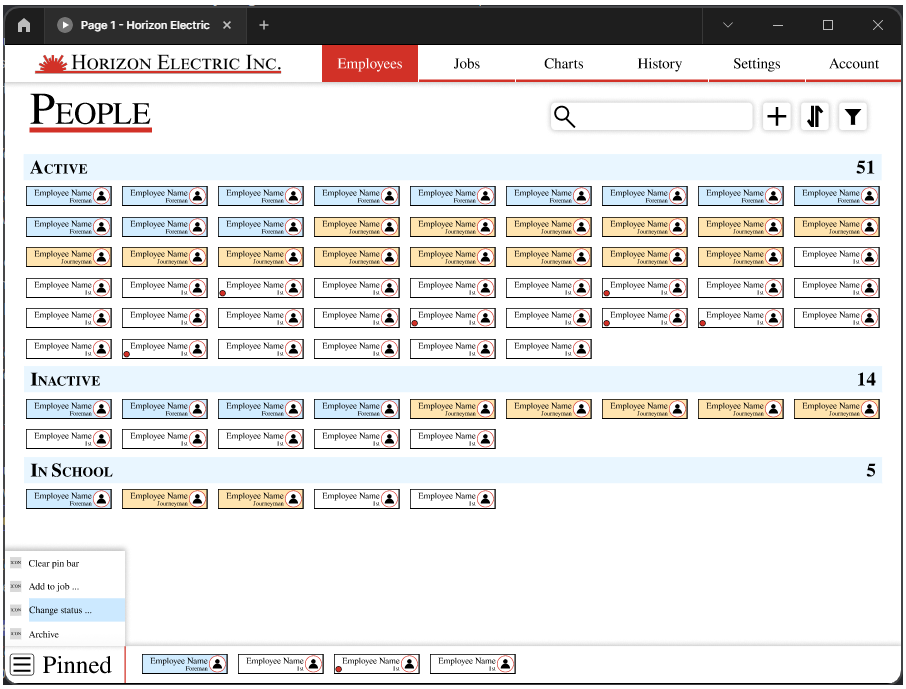


- Clicking person icon on job screen opens up list of employees. The list will not disappear until person icon is clicked again.
- List divided into categories which can be switched by clicking on tabs.
- Employees can be dragged and dropped from this list to different jobs.

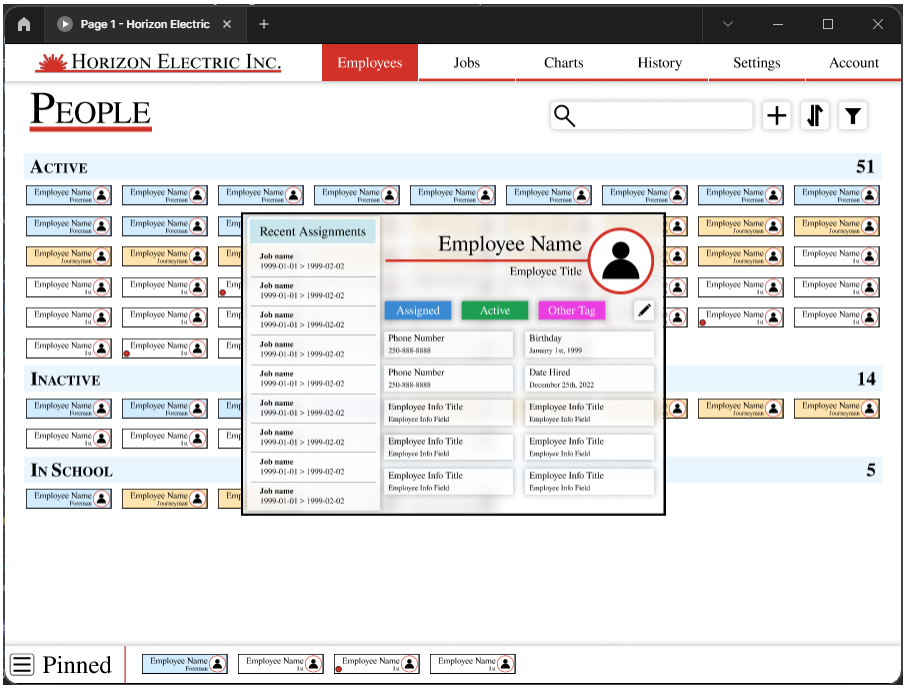




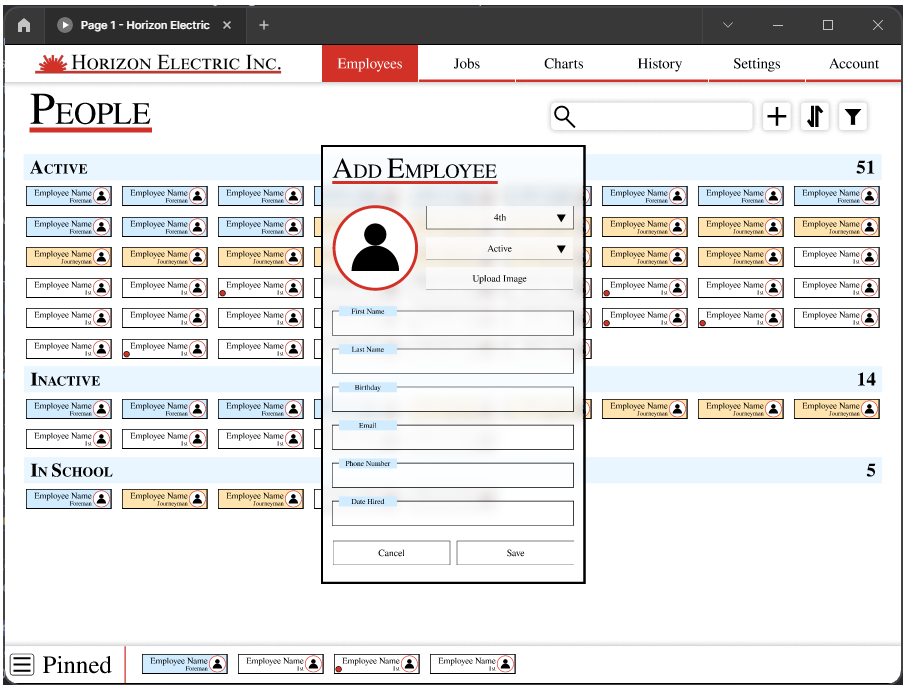
- Clicking a job brings up the job details screen.
- Job notes, project manager, and start and end dates can be edited by clicking on them.
- History tab shows history of predicted number of employees vs. actual number of employees.
- Assignments tab shows list of assigned employees and which dates they were assigned.

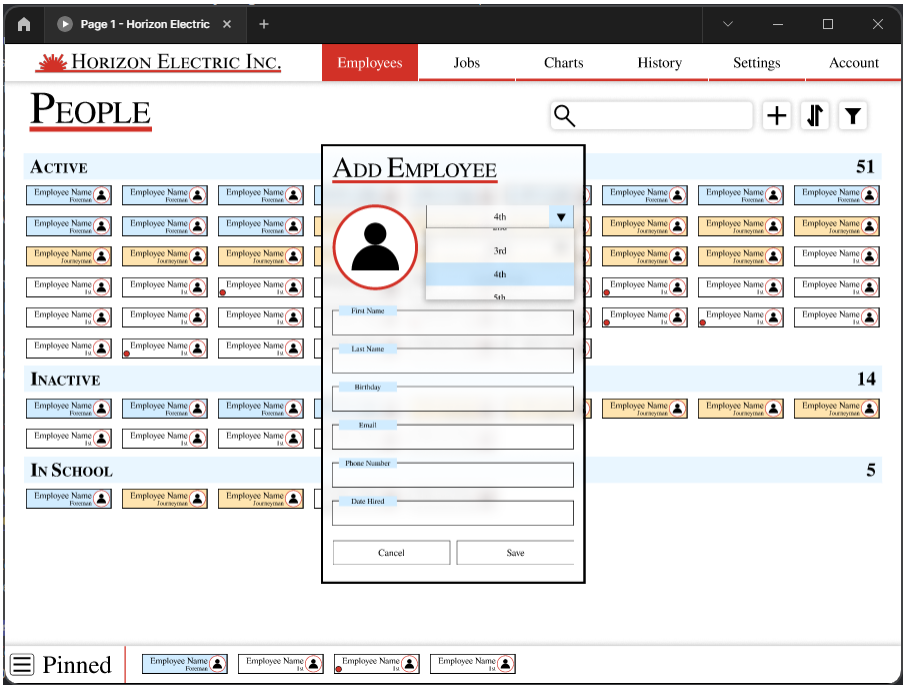


- Employee screen contains list of employees divided into groups based on current status.
- Dragging employee boxes into bottom “Pin bar” allows for easy access to a group of selected employees and also batch actions.



- Clicking on an employee box will bring up that employee details.
- Employee information such as phone number and recent assignments are viewable.
- Employee information is editable by clicking on the edit icon which will make the boxes editable.



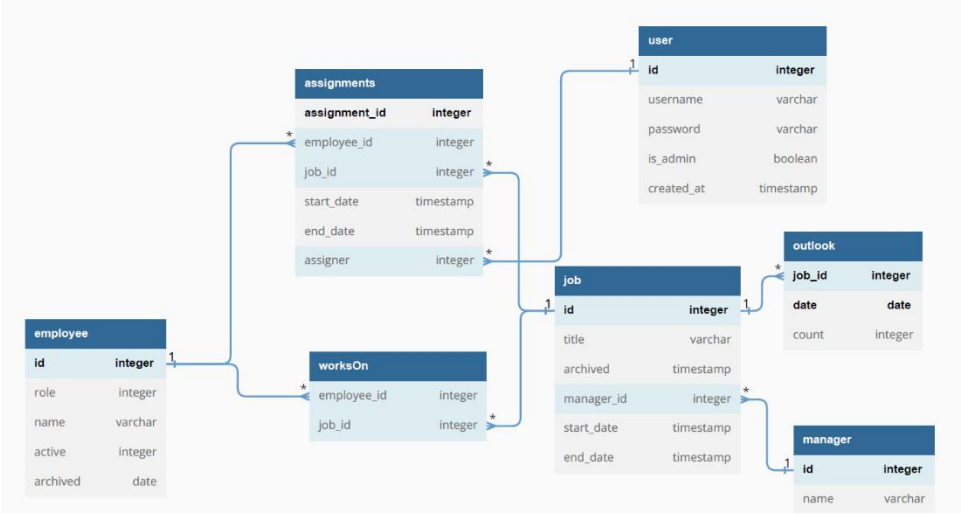


- Clicking the + button brings up the add employee screen which can be used to add new employees.
- Dropdowns are available to change certain fields

UI Mockup Client Feedback

- Client is happy with current design. There were only small feedback such as the suggestion to remove the giant page title and the addition of a history page.

Database Diagram



Testing

1. Unit testing

Unit tests will be used throughout the development process to ensure that the project meets all the functional requirements of the clients. All major functional components must pass their respective unit tests. Unit tests should be written prior to the implementation of functional components.

Example: Unit testing for user login function

Verify the validity of the user login function by comparing the login credentials entered by the user with the login credentials in the database. The login function should return a Boolean value of true if a matching record is found in the database, or false if no such record is in the database.

Sample code:

```
public function testLogin()
{
    // dummy data
    $username = 'test';
    $password = 'test123';

    // Call the function or method that performs the user login
    $isLoggedIn = logIn($username, $password);

    // Assert that the user login is successful
    $this->assertTrue($isLoggedIn);
}
```

The login() function will establish a connection to MySQL database, take $username and $password as

2. Integration testing

The purpose of integration testing is to verify the proper integration and interaction between functional components, and make sure that an existing component's functionality won't change due to the integration of a new component. The functional components must pass unit testing before integration testing can take place. A continuous integration solution such as GitHub Actions or Travis CI will help automate the integration testing process.

Example: Archiving an existing job

When a job is archived, it should be removed from the current ongoing jobs list and be moved to the archived jobs section. As a result, other metrics on the ongoing jobs list might change, such as the total number of employees currently working. As these systems depend on each other, both the archiving job and the total number of employees should pass all unit tests before they can be merged.

3. User testing

The purpose of user testing is to ensure that the project fits all the non-functional requirements of the client. In this process the developers will briefly introduce the functions of the website to the clients and let clients explore these functions themselves. Developers will gather feedback from the clients to make further adjustments to the project. User testing should take place after we have a minimum viable product.

Example: Test employee creation, employee allocation

Briefly instruct the client to create an employee, let the client test features involving employee allocation, such as drag an employee to a different job, add a new employee to a job, or delete an employee from a job. Gather feedback on the intuitiveness of the employee creation and allocation process, along with any identifiable issues.

4. Regression testing

The purpose of regression testing is to ensure that the existing functionality of the system has not been affected by new changes or updates. Regression testing is performed after a change is made in the software.

Example: Employee Creation and Allocation

Test Steps After adding 'create new employee' feature:

- Run tests for allocating an employee to a different job, adding a new employee to a job, and deleting an employee from a job (as well as all other created tests).
- Identify any failing tests or unexpected behavior.
- Verify the intuitiveness of the employee creation and allocation process.

Expected Result: All tests must pass, and the employee creation and allocation process should be intuitive, and all related features should function correctly without any issues.

Type of testing for each component

Function/component	Type of testing
User Login	1,3,4
User Logout	1,3,4
Upload CSV file	1,3,4
Export CSV file	1,3,4
Undo Previous action	1,2,3,4
Allocate/move employee	1,2,3,4
Create employee	1,3,4
Delete employee	1,2,3,4
Edit employee	1,2,3,4
Create Job	1,2,3,4
Edit Job	1,2,3,4
Archive Job	1,2,3,4
Create Account	1,3,4
Reset User Password	1,3,4
Load Database Backup	1,2,3,4
Change from view/edit mode	1,3,4
Graph display	1,2,3,4