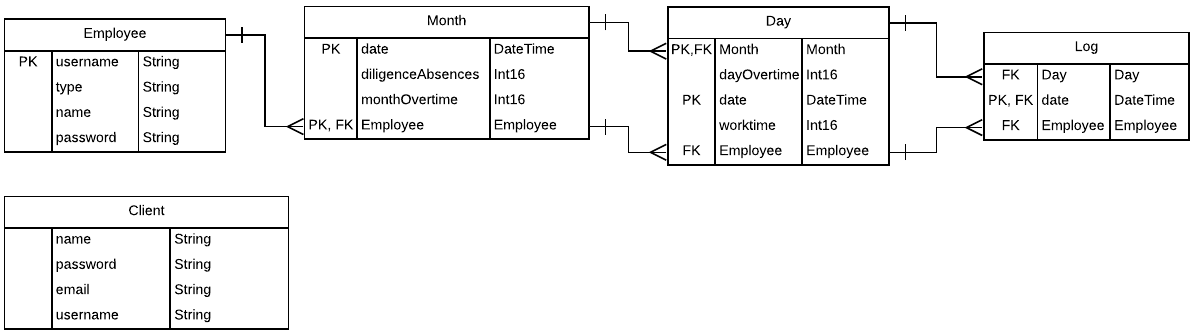
Criterion B: Design Overview

- The client wishes that the program will store the following extra wage types: normal pay, diligence pay, fixed pay, canceled bonus, and no extra pay (*refer to Appendix B– Summary of Client Interview*).
- The client wishes for the program only the most basic information (username, password, name) for the client and employee profile.
- The client is adamant on a JSON file of *only* type, overtime, and diligence absence so that he can manipulate the data with a JSON app he uses (*refer to Appendix B*).

Entity Relationship Diagram (ERD)



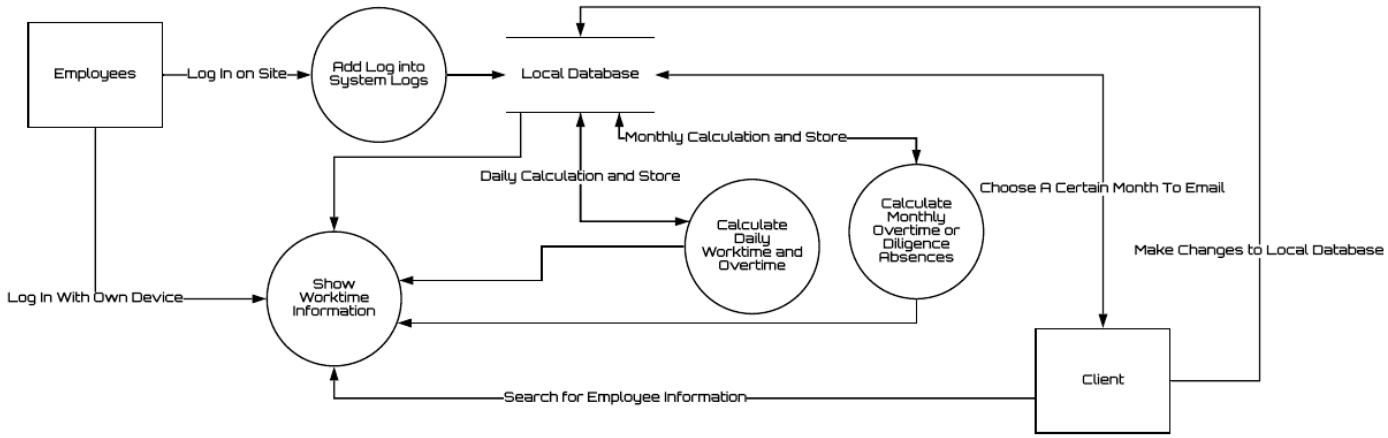
TECHNICAL

- Each employee will be uniquely identified by their username.
- There is only one client, so there is only one row, for the “Client” table, and no primary key is needed. This client account will be hard-coded and can be changed when the client needs to.
- Data rows added to the “Log” table are done through a user login, so only the employee account and the date of the login are recorded. Both attributes are necessary to connect logs to “Day” and “Month” entities.Thus, there are two foreign keys from “Log” all the way to “Month”.
- Only monthly overtime for normal pay employees and diligence absences for diligent pay employees are important monthly information. Other employee types have either a fixed, canceled, or zero extra pay.
- “Month”, “Day”, and “Logs” should be related to each other, but all entities should relate back to an “Employee”. The delete rule should be cascading– e.g. deleting an “Employee” deletes all “Month”, “Day”, “Log” relating to that employee.

USER EXPERIENCE

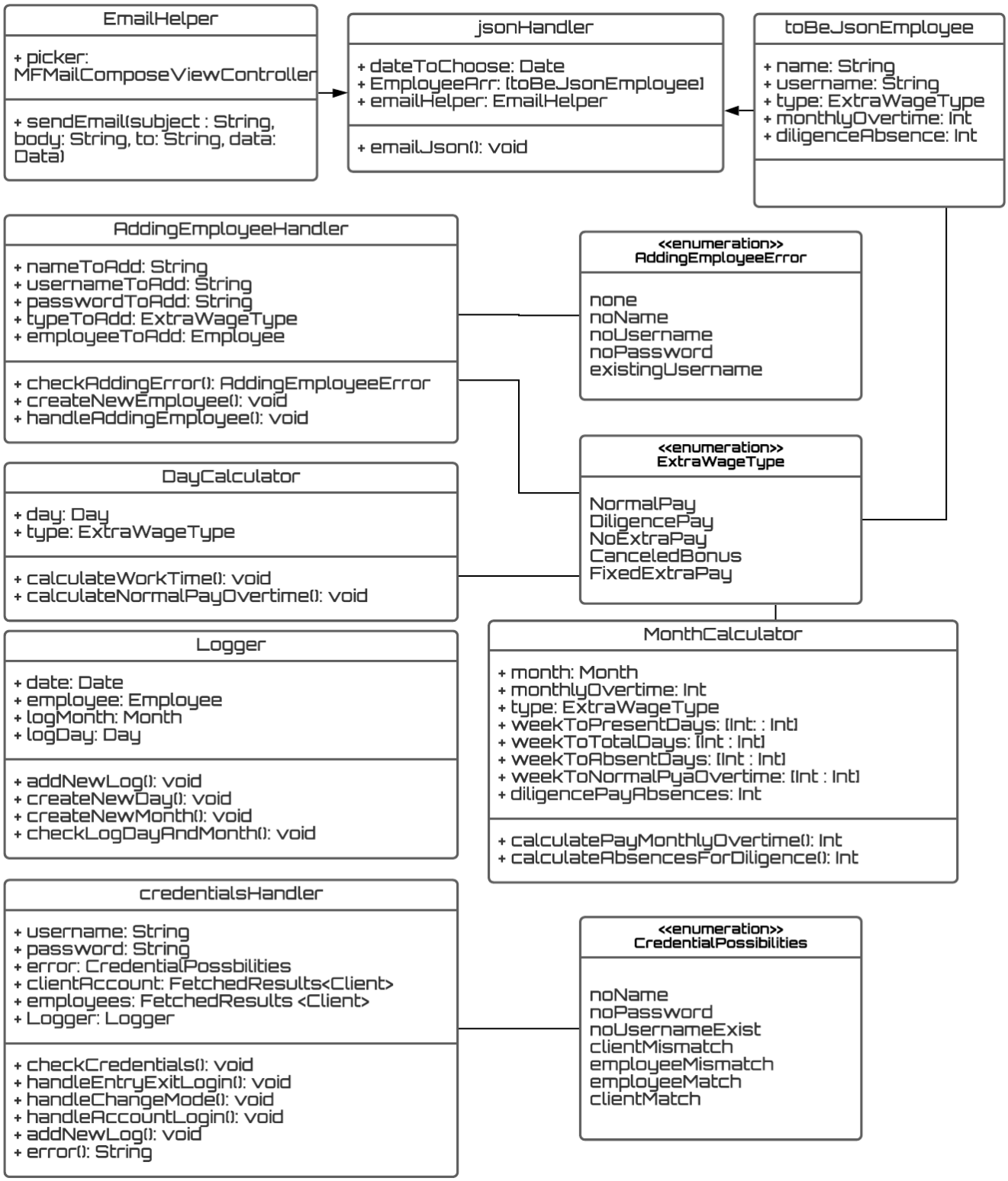
- The user does not explicitly see these entities. The “Employee” and “Client” entities hold credentials that allow for log in.
- When employees look for their information on a certain day/month, they will see information from a hybrid of these entities.

Data Flow Diagram (DFD)



- In the vicinity of the workplace, employee check-in/outs will be recorded in the system database.
- Otherwise, employees can view their own information and logs, and clients can navigate and view all employees and information.
- The program stores all the logs for recording purposes, and, if applicable, processes the diligence pay absences or monthly overtimes and daily worktimes.
- The client can choose to email this information to himself as a JSON file for him to view with his preferred JSON app viewer.
- If needed, the client can change any logs or work time in the local database.

UML Diagrams

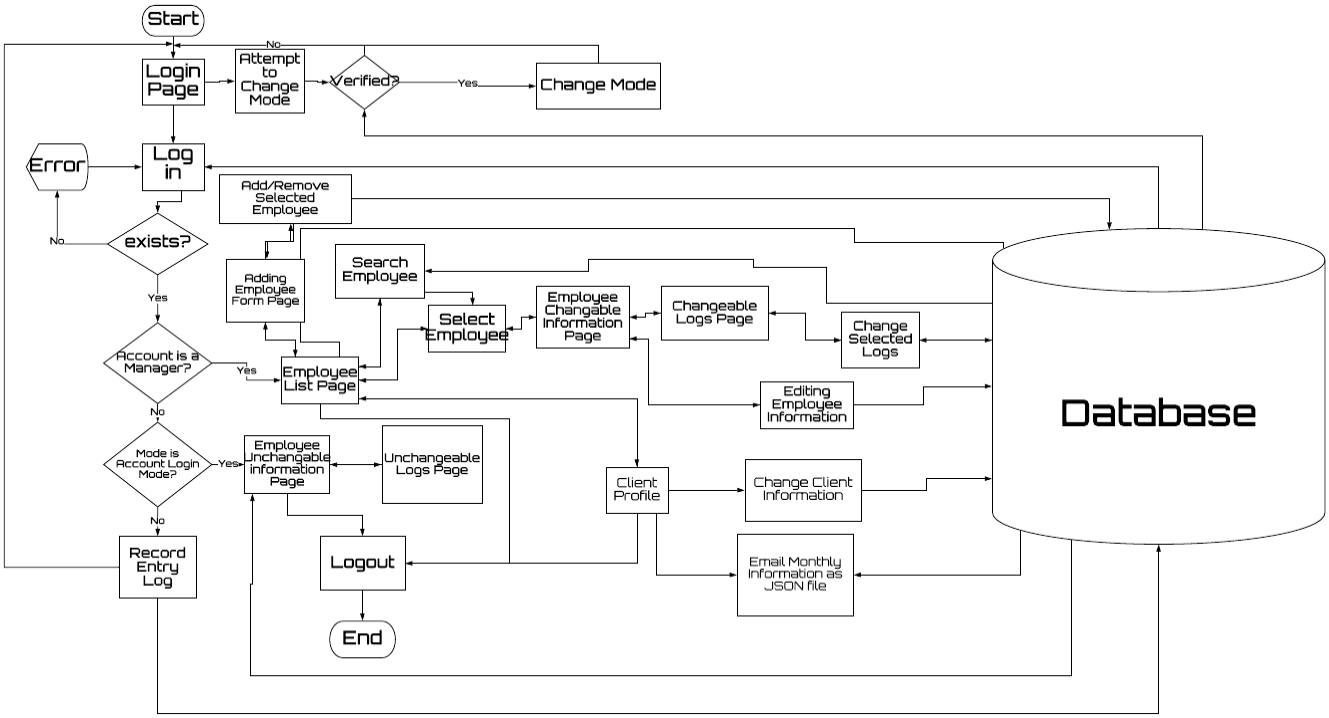


A line connecting an enumerator and a class means that the enumerator is used in the class.[1]

- The "JsonHandler" HAS AN "EmailHelper", and HAS MANY "ToBeJsonEmployee(s)". These help send the JSON file to the client's email.
- The "credentialsHandler" class handles checking for entered user credentials for check-in/out and user account authentication.
- The "Logger" class handles adding check-in/out logs, or client-changes to logs.
- "DayCalculator" class calculates daily work time and, if applicable, overtime of every day. The "MonthCalculator" class calculates the monthly overtimes or diligence pay absences, where applicable, of every month.
- Classes are abstract and separate from the UI. Users will not directly see or interact with classes.

Flowcharts

System Flowchart: Overview of the App



- The design starts with a login page that leads to a user's account page.
- By authenticating the client account, the program's login mode can be changed to a mode for check-in/out or vice versa.
- Employee's can check information like their status, work time, and logs for the day, but cannot change them.
- A client can add, delete, search, view, and change the employee's basic profile information, including logs of any employee.

---

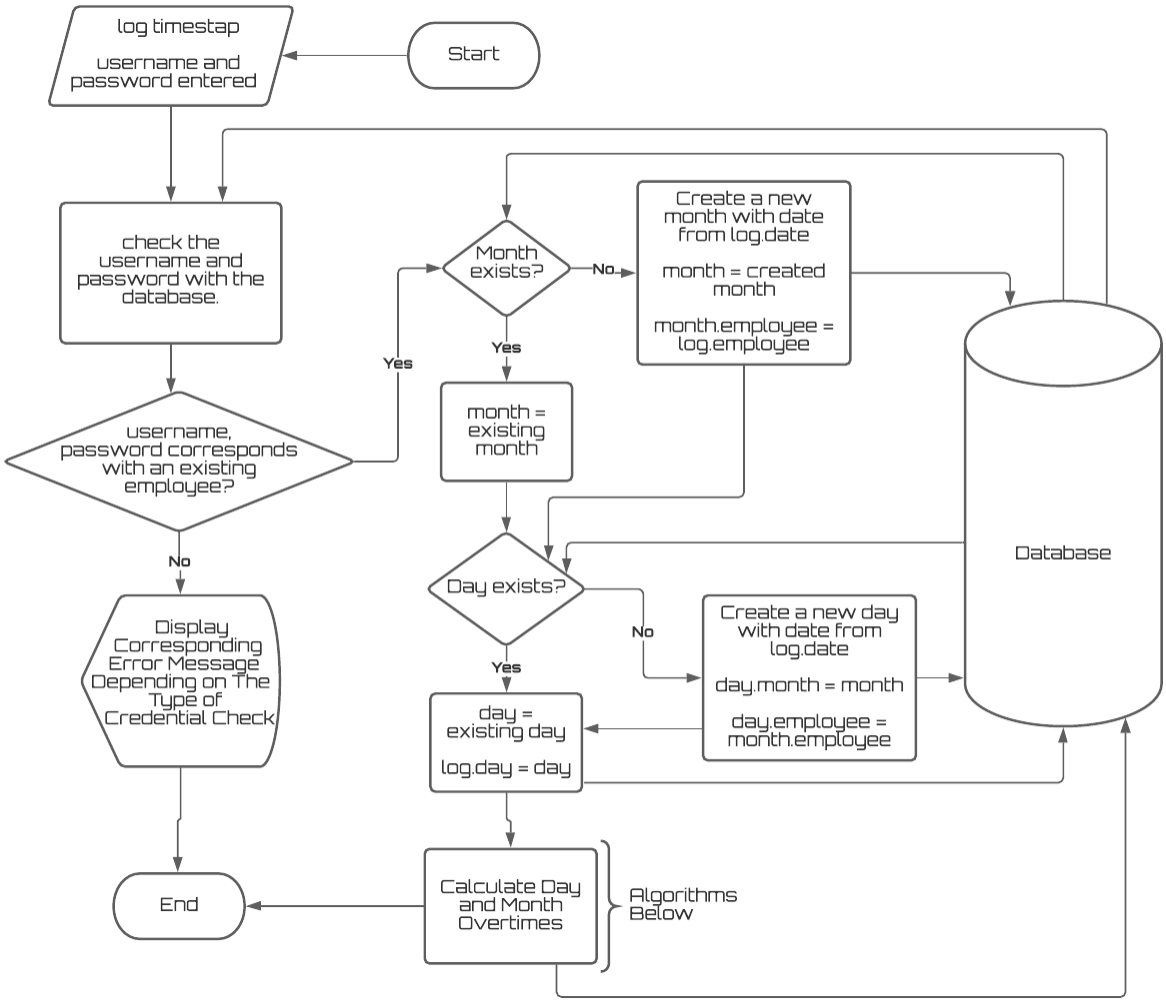
[1] ("UML Class Diagram Enumeration - Software Ideas Modeler")

- A client can also view and change his profile information, and choose a month to send JSON information of his employees to his email.
- All changes are reflected in the local database.

---

*Algorithm Flowcharts (refer to Criterion A– Success Criteria **# 8, 9, 10**)*

Checking-in or out



- A specific error message should display for each case of whether the credentials were empty, an password mismatch, or a client-attempted check-in/out.

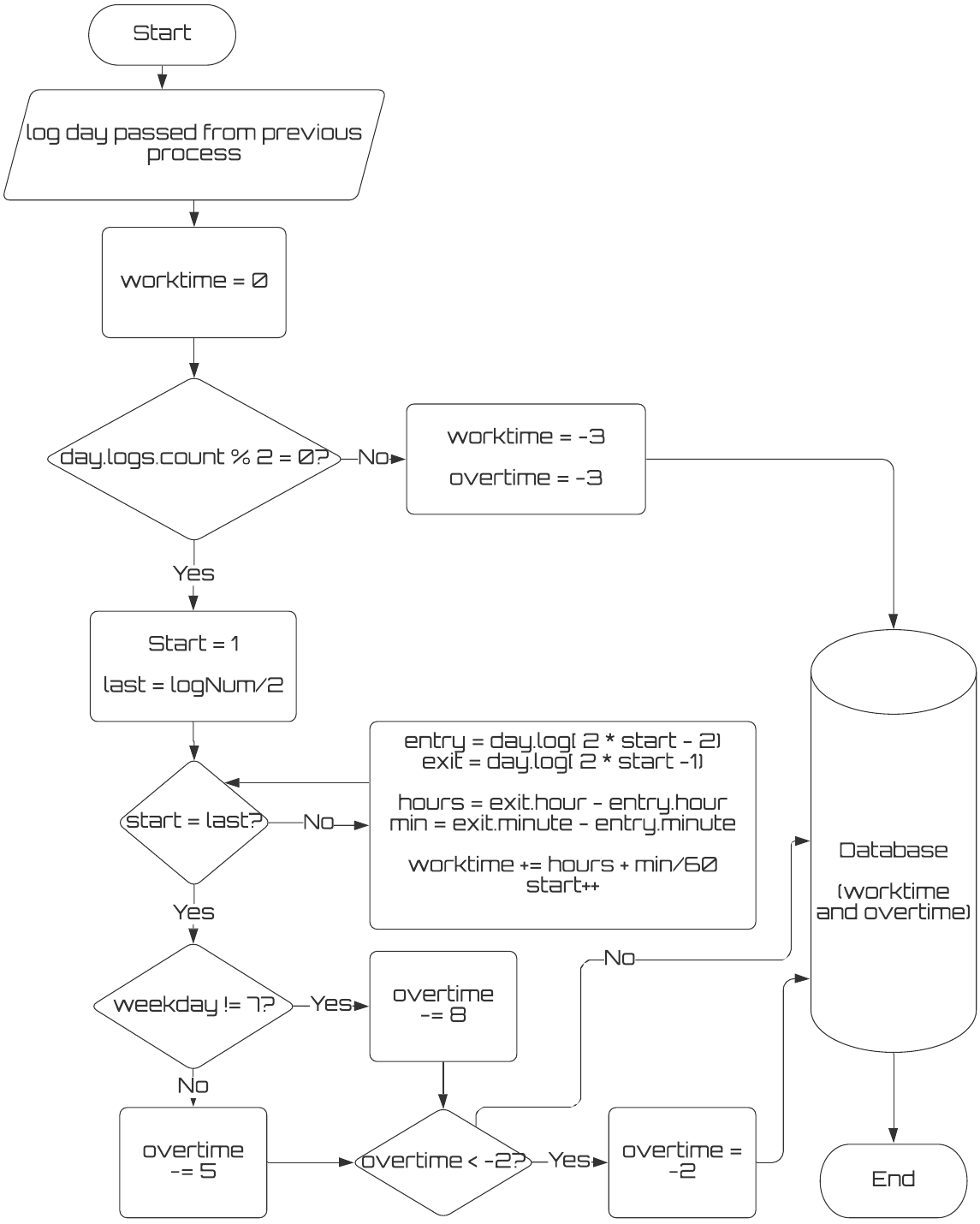
- The algorithm checks credentials before recording the timestamp.

- The timestamp used is the “Date” fundamental data type offered in Swift, and stored as a dateTime in the sqlite database.

- The algorithm ensures that every log will have a month and day entity corresponding to it, and that worktimes and overtimes, if applicable, will be *automatically* updated.

- “Month” and “Day” in flowchart refers to Day and Month entities that are compositely identified by an employee it belongs to and the date.
-

Calculating For Day Work Time And Overtime

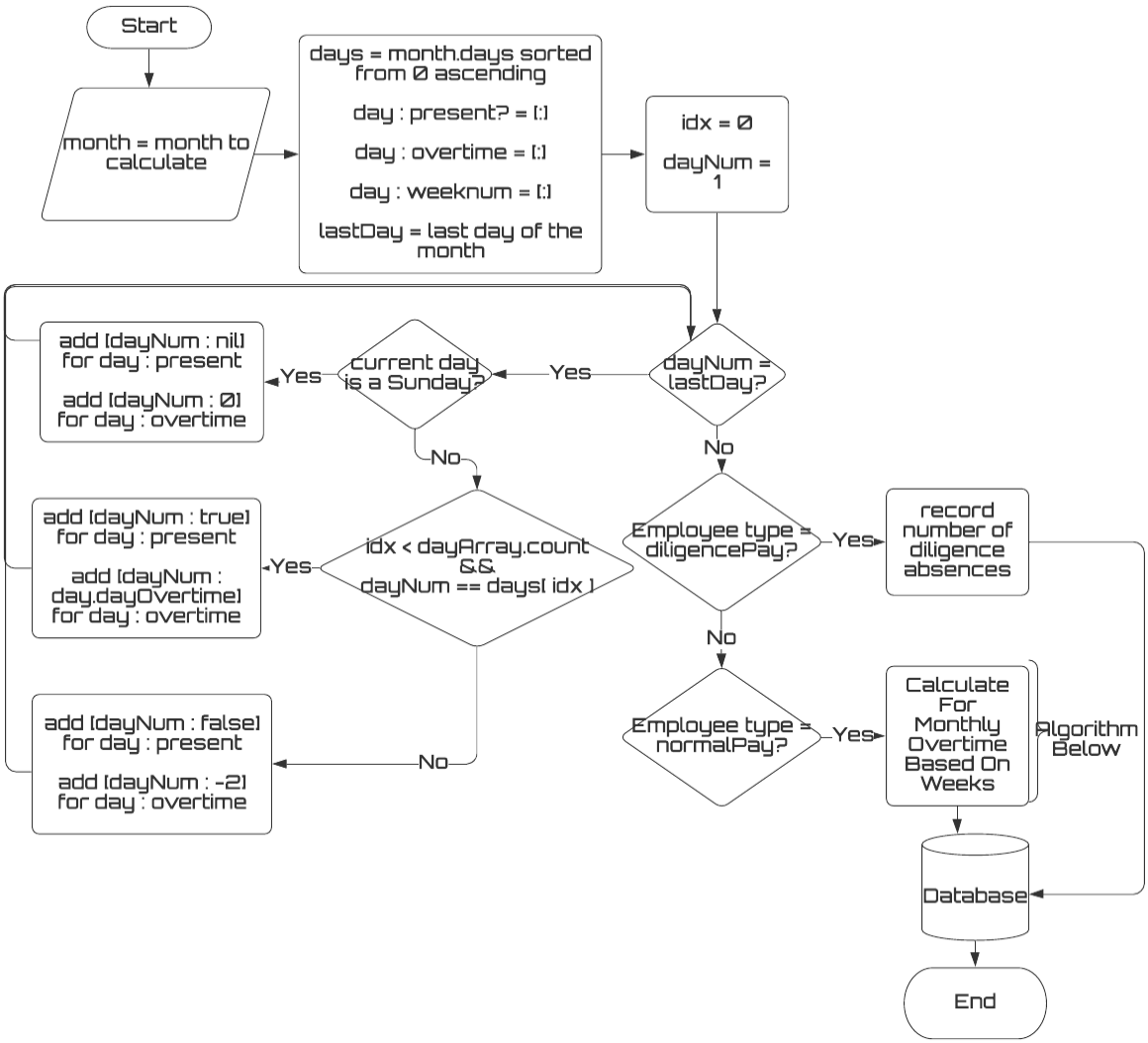


- *Note*: -3 as a work time and overtime will never be reached if the employee has an even number of logs and represents a day with an odd number of logs.

- The algorithm accounts for intermediate exits of an employee by calculating work time in pairs of logins/logouts.

- For normal pay employees, any more than 5 hours on a Saturday and 8 hours on a weekday is overtime.

Processing for month information



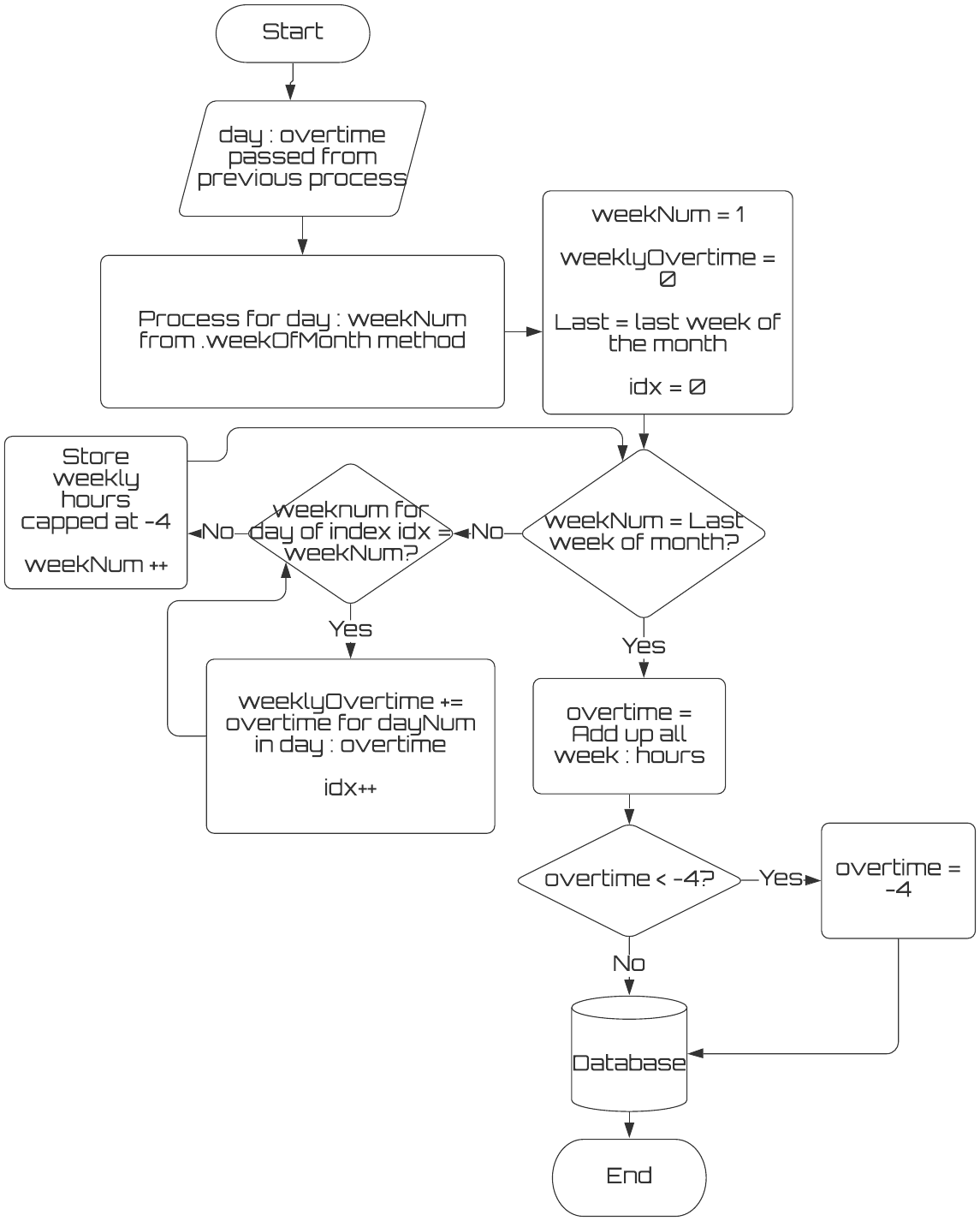
- This algorithm creates dictionaries that hold specific values for each possible day of the month. The use of dictionaries are convenient because the different dictionaries can be related by the day of the month as a key.

- Accounting for every day of the month is necessary to handle absences.

- Only absences are important for diligence pay employees. Canceled bonus, no extra pay, and fixed pay employees do not need the monthly overtimes either.

- The algorithm ensures that (for normal pay employees) an absence on Sunday would not have any overtime effect, while an absence on any other day would be a deduction of 2 overtime hours.
-

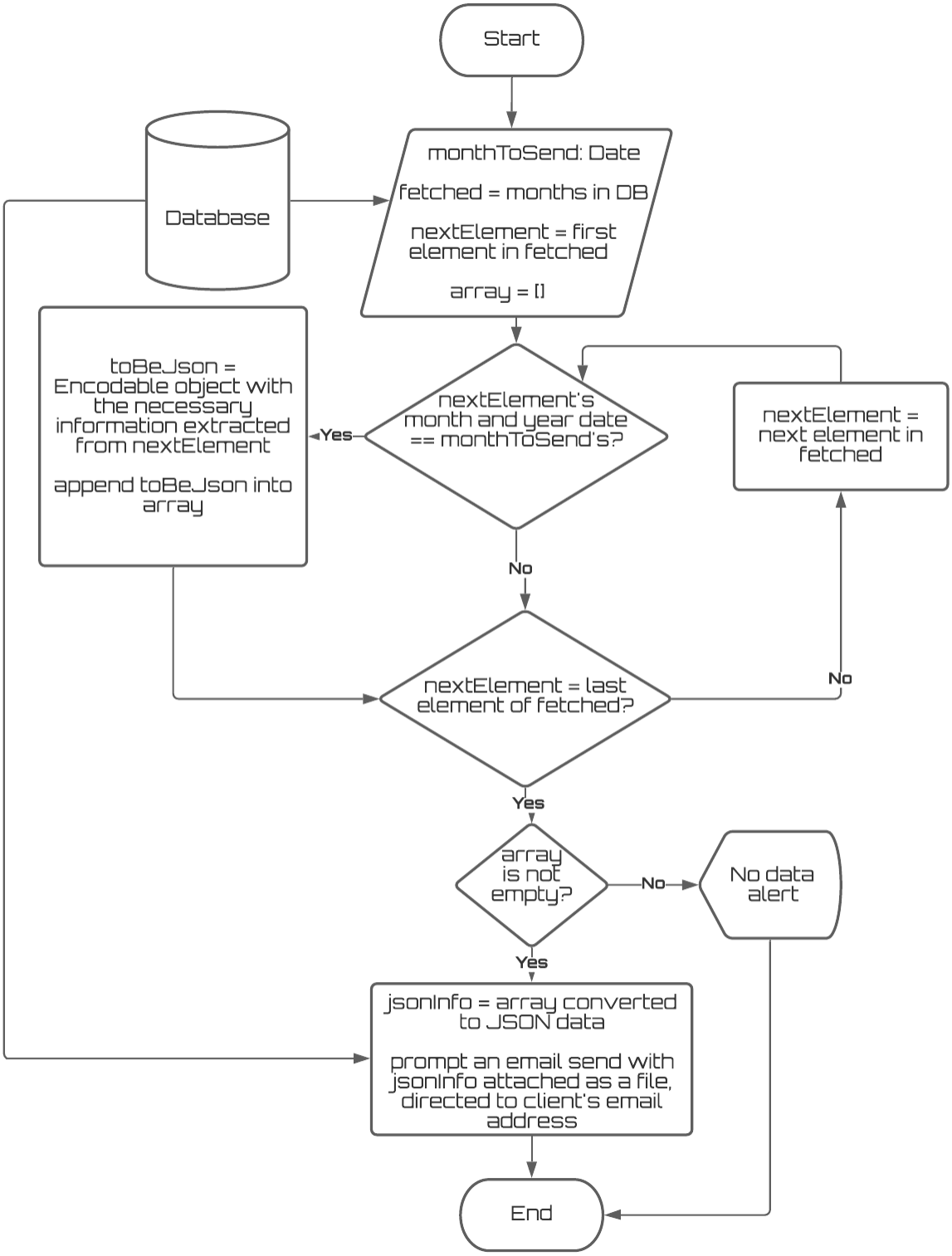
Monthly Overtime



- This algorithm applies only to normal pay employees.
- The algorithm handles for the regulation: weekly overtimes cannot be below -4 hours. This requires looping through each week and every day of the month.

- The overtime hours per week is then added to get normal pay overtime hours.

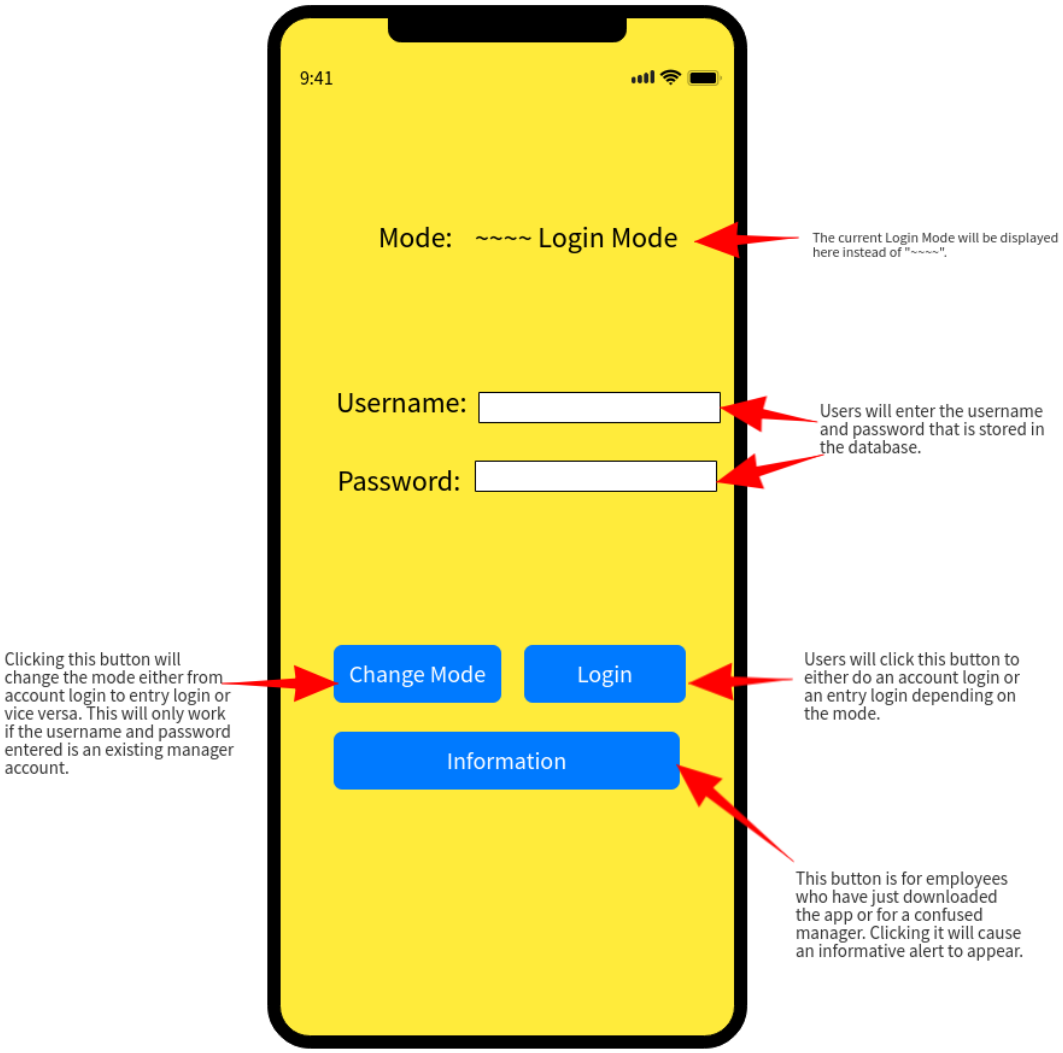
Sending Json Information as an Email To Client



- Algorithm sends a JSON email of the information of an employee on a month.
- The client chooses a month and year, which identifies the month.
- “Fetched” refers to a fetchresults of Month entities– which have an employee related to it.
- Through these Month entities, *username, extra wage type, name, monthly overtime, and diligence absences* are extracted.

Mockups

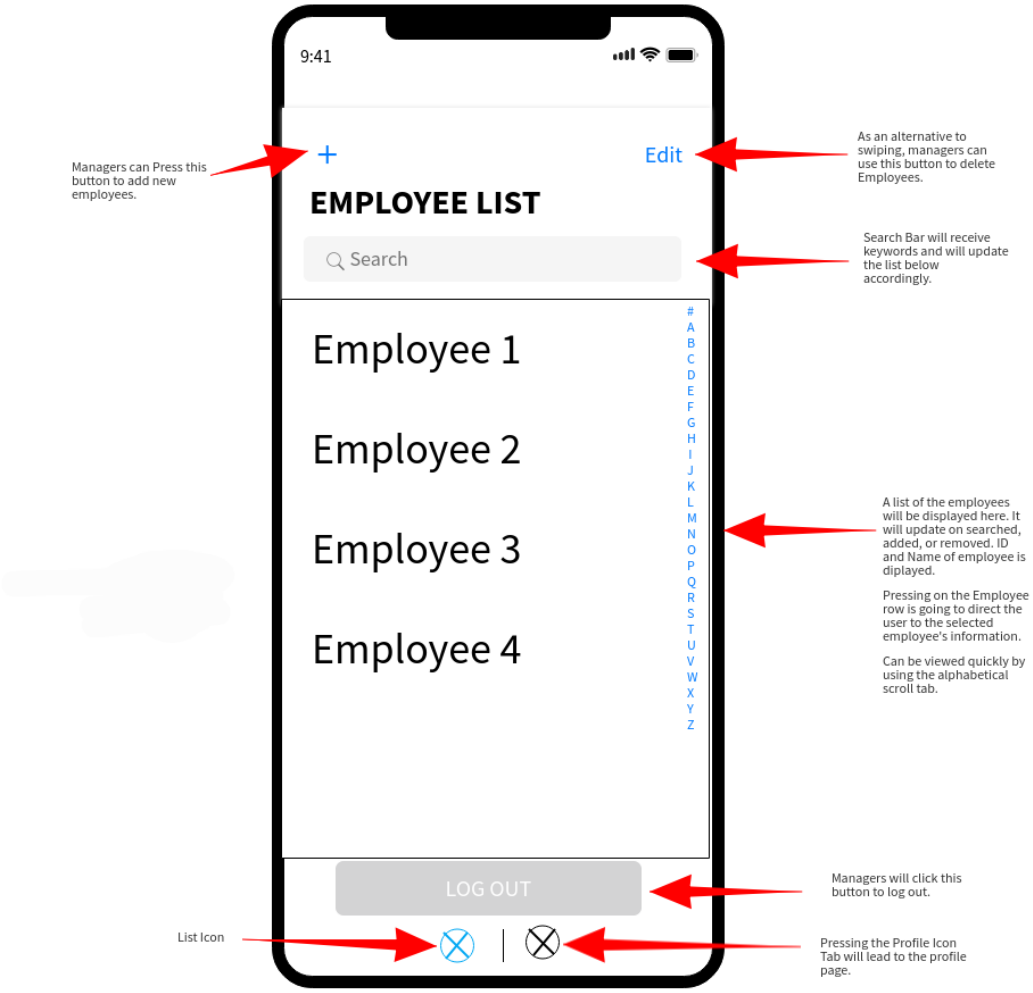
*Made With Mockflow.com (MockFlow)
On the iPad, the program should look the same as the iPhone mockups, only bigger.
Log In Page



- The only manager is the client.

- The two modes available are ‘check-in | check-out’ and ‘account login’ mode. (*refer to Appendix C– proposed product outline*)
- Only the client can change modes and the default is the ‘account login’ mode.
- check-in/out logins will record timestamps as part of an employee’s work time.
- check-in/out is intended logins on a device in the work vicinity.
- To change modes, the client will have to fill out his username and password.
- The informative alert can be dismissed, showing this page again.
- Login credentials are validated by local database information.

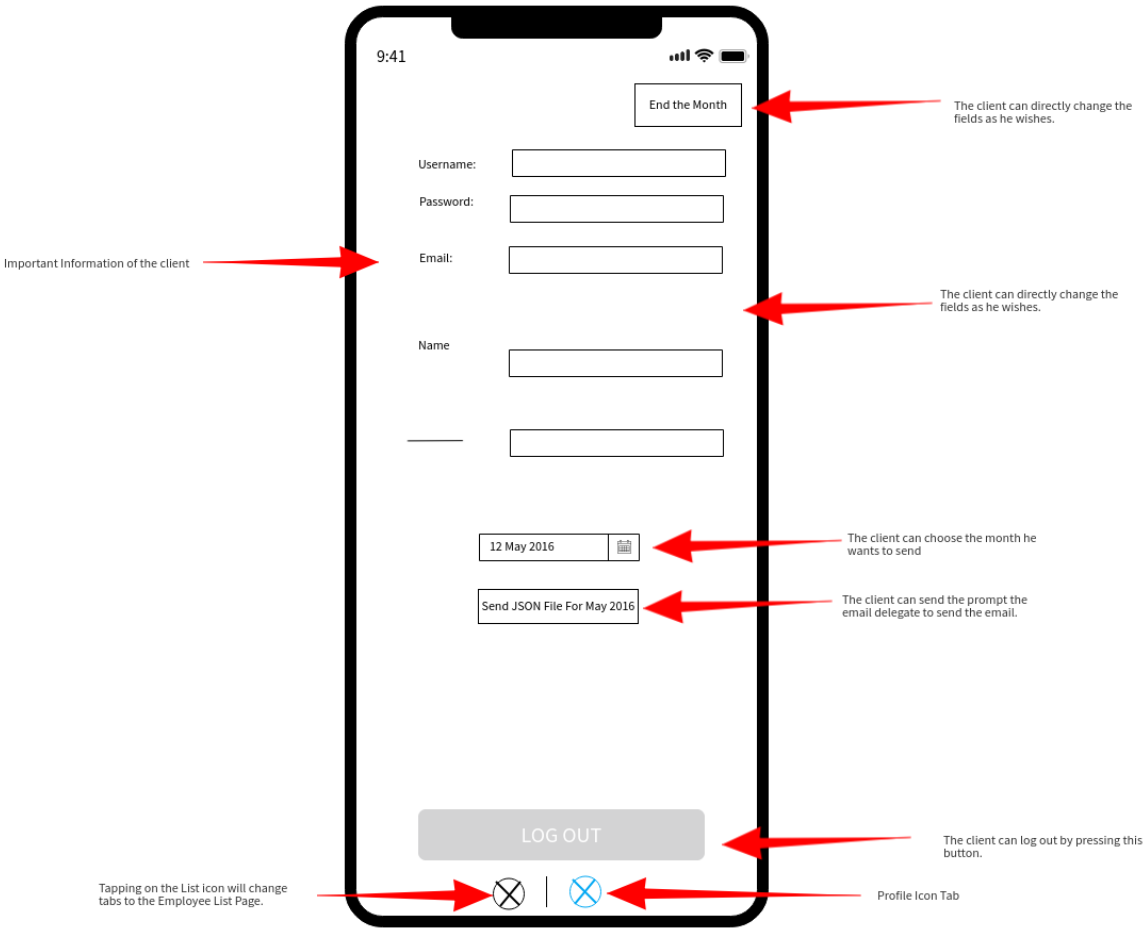
Employee List Page



- Pressing on the name of an employee will open the Employee Changeable Information Page of that employee.
- Deleting an employee will also delete the employee in the local database.

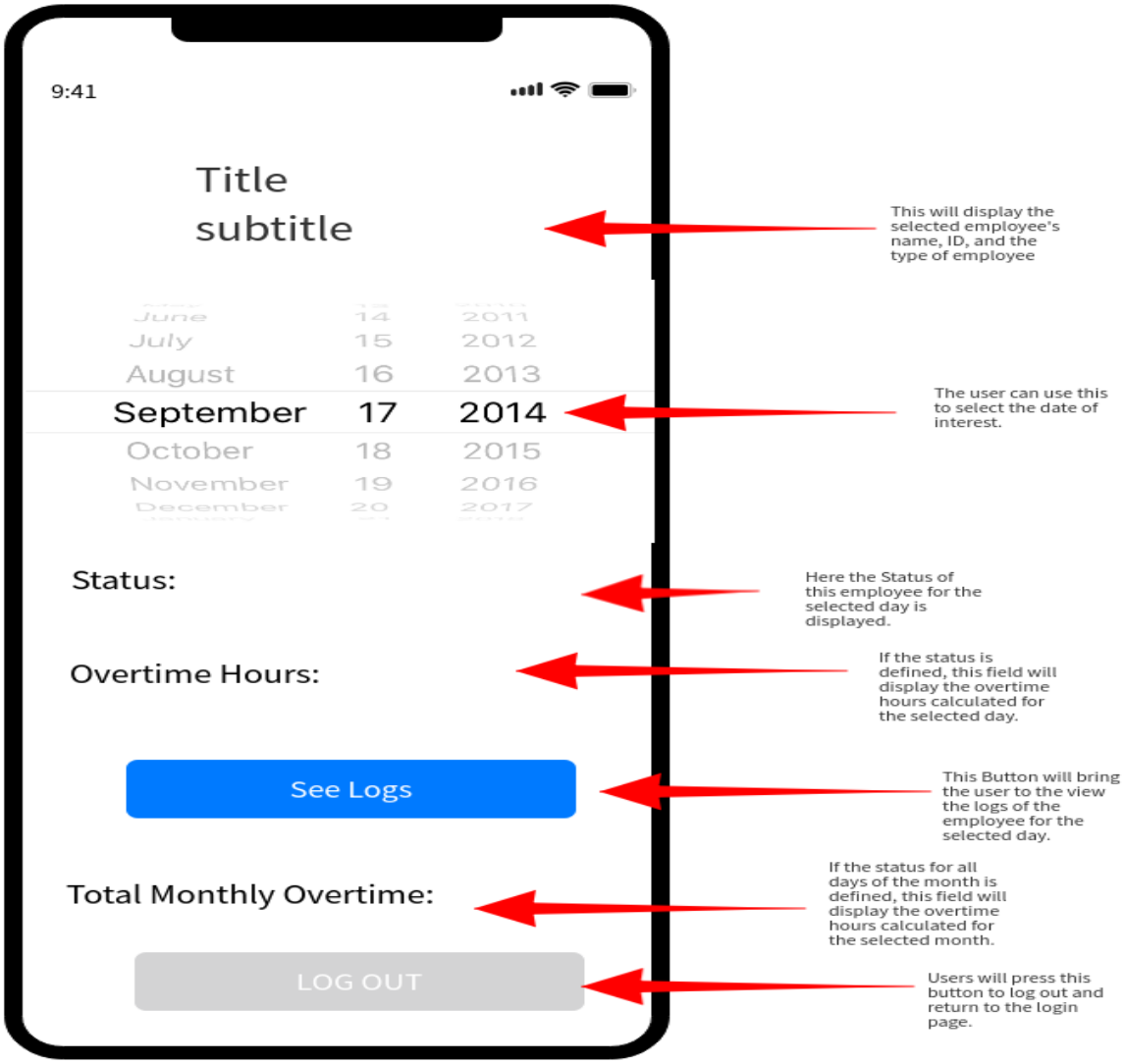
- The list of employees is scrollable, and is sorted alphabetically.
- Search system will show employees with last or first names containing the keyword.
- Allow users to log out.
- Allow adding new employees manually.
- An option to switch to the Profile tabs is provided.

Client Profile Page



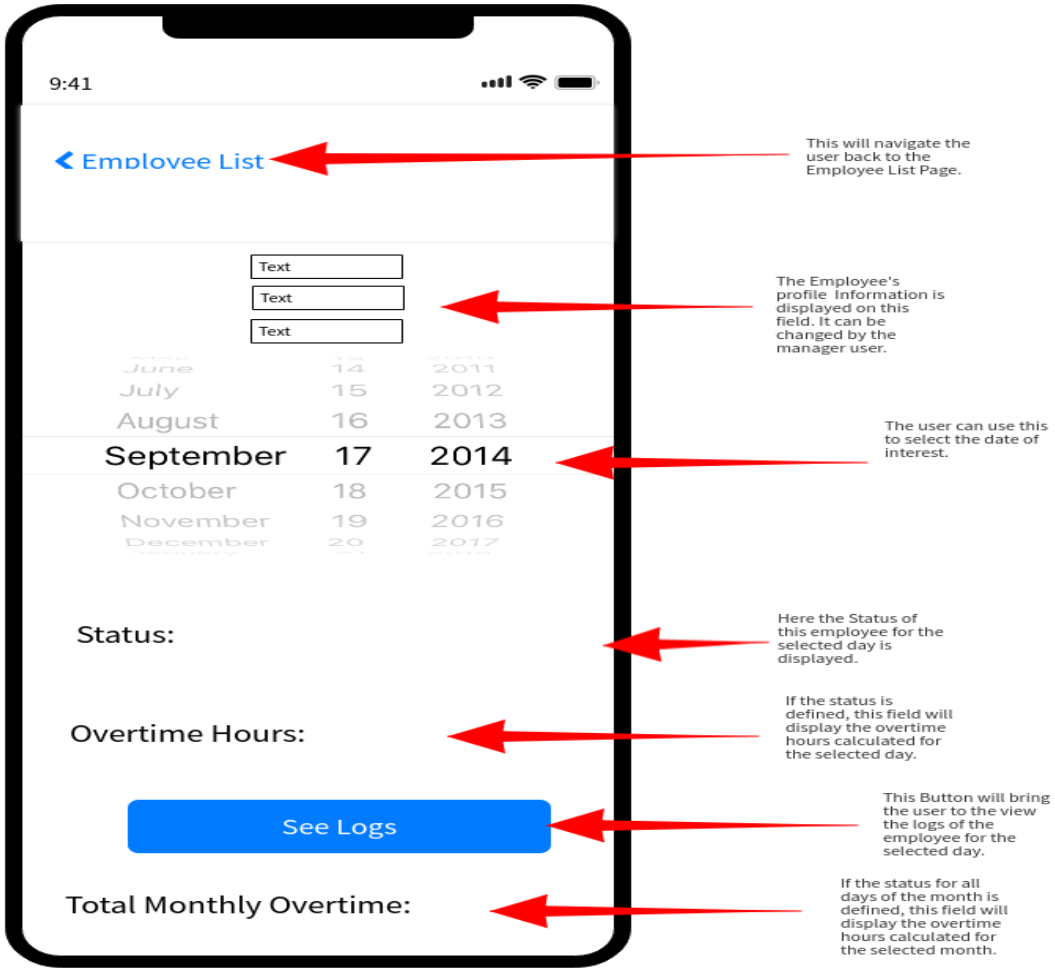
- An option to switch to the Profile tabs is provided.
- Direct changes to the client's profile will be reflected in the database.
- Choosing a month (and year) with the datepicker will allow the client to send an email of all necessary information (*refer to Criteria A*) as a JSON.

Employee Unchangeable Information Page



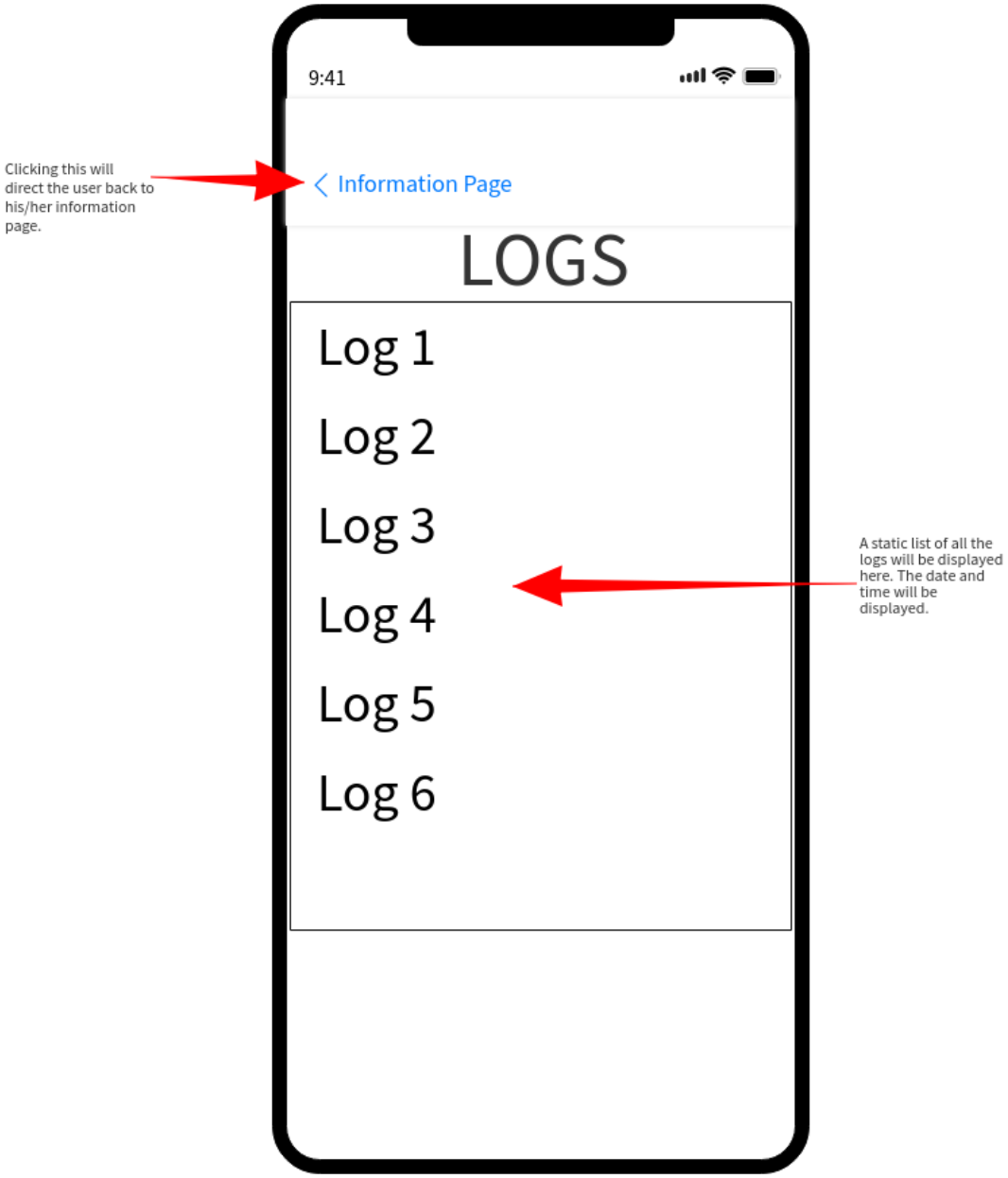
- This page is for employees who do an 'account' mode login.
- Employees are given read-only access; They cannot change their logs manually.
- Users can log out.

Employee Changeable Information Page



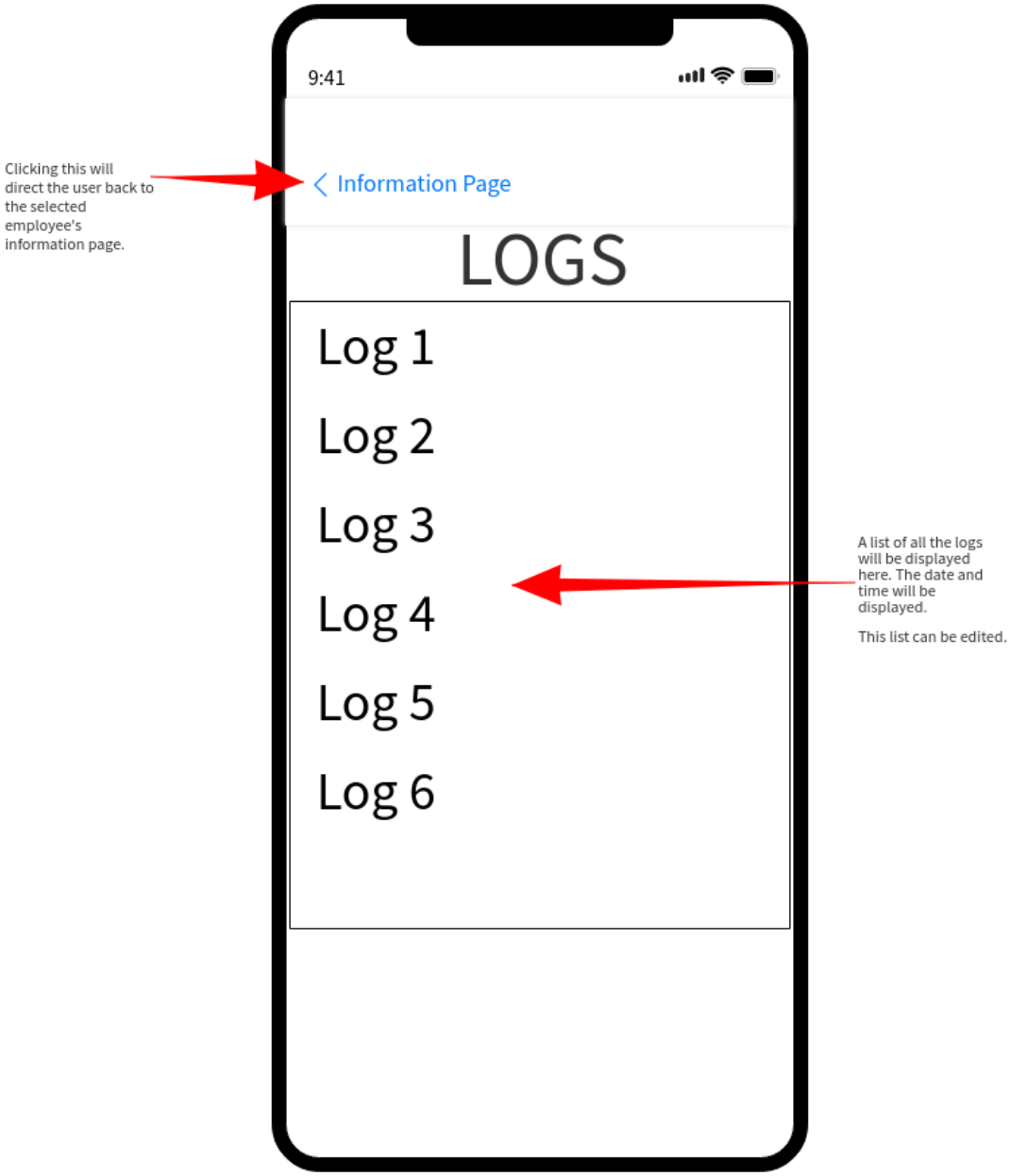
- The client can see all the information of employee work time.
- The client can change any information, so as to deal with special circumstances.
- The client can smoothly move back and forth between employee information and the list of employees.
- In addition to the information above, the work time hours for the day will also be calculated and shown.
- Overtimes will be shown for Normal Pay Employees, and diligence absences will be shown for diligence pay employees.

Unchangeable Logs Page



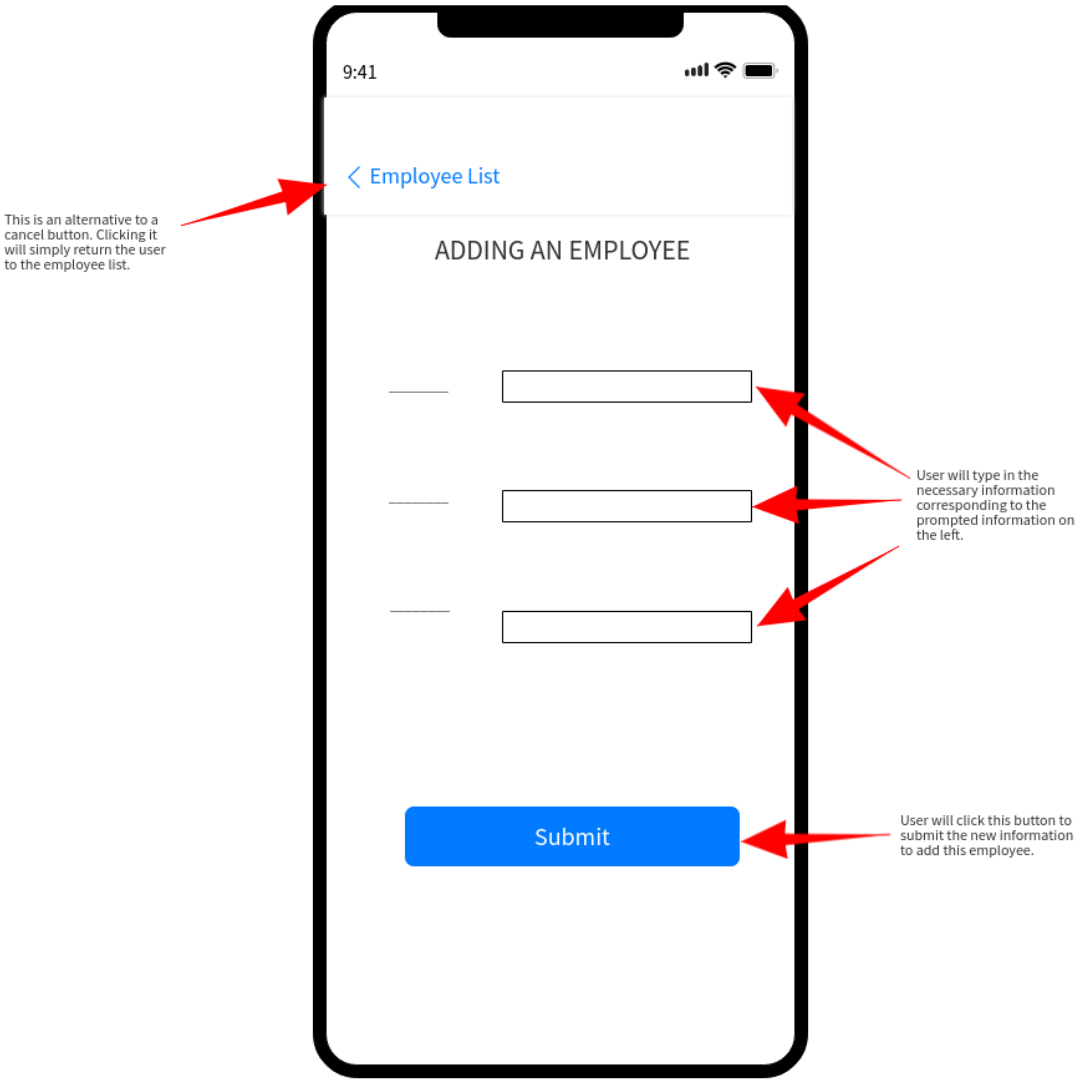
- This list is for employees.
- Read-only access is provided. Any issues can be directed to the client.

Changeable Logs Page



- Client has read-write access to employee logs. Changing logs will update the worktime calculations.

Adding Employee Form Page



- The client can add new employees this way.
- One of the fields is the password of the employee, so adding an employee will also register their account.

Software Development Framework

- The intended framework for this application is the Waterfall model.
- The Waterfall model breaks down the intended product into many different tasks. Its highlight is to create a system by performing these tasks in a sequential manner that will depend on the deliverables of the previous task.
- This framework is most fitting because the client prefers to be involved in a minimal number of interviews, and does not want to be actively involved in the production process of the application.
- Thus, it is sensible to gather all the wants and needs of the client beforehand, and then work with that on to the next task.
- This makes it easy to have the end goal in mind from the start, which will be helpful for creating a relevant product.
- Breaking down the program into smaller parts also allows for a clear and defined structure which makes the program more easy/efficient to approach and create.

Testing Plan

Action Test	Method of Testing	Desired Result
Making sure the account login/logout system is secure and functional. This feature satisfies part of the 1rst success criteria.	In Account login mode: - Login with a non-existent username and both an existent and non-existent password. - Login with a client username but a non-matching password. - Login with an employee username but a non-matching password. - Login with a matching employee username and password. - Login with a matching client username and password. - Logging out from both the employee and client account page.	 - Access denied. Displays a noUsername error. - Access denied. Displays a clientMismatch error. - Access denied. Displays an employeeMismatch error. - The login page will be directed to the corresponding employee information page. - The login page will be directed to the corresponding employee list page. - The page will be directed to the login page with username and password fields cleared.

Making sure changing mode is only allowed for clients. This satisfies the client verification feature outlined in success criteria 6.	From both Check-in/out to Account AND from Account to Check-in/out: - Attempt to change mode by entering an employee account. - Attempt to change mode by entering a non-matching and non-existent account. - Attempt to change mode by entering a client account.	 - Denied. Displays employee error. - Denied. Displays the same errors as corresponding to No. 1. - Mode will be changed
Check-in/out for employees. This satisfies the timestamp recording feature outlined in success criteria 6. *This feature is not intended for clients.*	- Attempt to do check-in/out for a client account. - Attempt to do an check-in/out by entering a non-matching and non-existent account. - Doing a check-in/out for an employee account.	- Error will be displayed for the client. - Displays the same errors as No. 1. - Logs will be recorded in the database with the timestamp and employee. If the log day and/or month does not exist, it will be created. And then, connections between the log and the corresponding day will be made.
Searching for an employee and viewing their information by the client. This satisfies the 3rd success criteria.	In the client's account mode: - Attempt to search for names of the clients on search bar. - Pressing on the name of the employee.	 - As the name is searched, the list of employees will automatically update to matched employees with names that start with the searched letters. -This will redirect the client to the selected employee's editable information page.
Edits and read-write access to client information, employee information, and logs. Editing access of employee and log information largely by clients will satisfy the 5th and 7th success criteria.	- Attempt by an account logged in employee to change information. - Attempt by client to change employee name, username, and password. - Attempt by client to change client information. - Attempt by client to change employee type.	- All fields are disabled except for the password. Changes to password are saved in the database. - Username text field is disabled (username is unique). Changes to employee name and password will be saved to the database. - All information except username can be changed. - Changes are saved to the database. Daily work time and overtime and monthly overtime and diligence absences will update correspondingly.

Editing the client profile information satisfies the 2nd success criteria.	- Attempt to change, add, and delete logs of a selected employee.	- Changes are saved in the database. Daily work time and overtime and monthly overtime and diligence absences will update correspondingly.
An adding employee feature, but only for clients, satisfying the 4th success criteria.	Attempting to add employees by: - Logging in as an employee. - Trying all cases: no name entered, no username entered, no password entered, and trying an existing username. - Submitting a valid name, password, username, and type.	- There is no way to add an employee. - Corresponding error message displayed. - The added employee will be saved to the database with the corresponding added information.
Method of calculating daily work time and daily overtime. Accurate processing and capping of these values satisfy the 8th and 9th success criteria.	On all types of employees: - Populating a weekday with 2 logs with a workday between 6 and >6 hours, and below 6 hours. - Same for a weekend: Populating a weekend with a workday between 5 and >5 hours, and below 5 hours. - Populating a weekday with 4 logs. - Populating a weekday with 3 logs.	- Work Time for all types of workers is calculated by (check-out log - check-in log). For Normal Pay Workers: The first daily overtime will yield (workday - 7) hours. The last case will yield a cap of -2 hours. - Work Time is calculated the same way. For Normal Pay Workers: The first two cases will just yield a daily overtime of the (workday - 5) hours. The last case will yield a cap of -2 hours. - The work time should yield a value of (fourth log time - third log time) + (second log time - first log time) - The daily work time and overtime (for normal pay workers) should yield a -3. The monthly overtime will also be a -3.
Handling absences and capping monthly overtime for	For a *NormalPay* employee (without populating any logs on Sundays):	

NormalPay employees. This also satisfies the 8th and 9th success criteria.	- Populate logs for a whole month except Sundays, with a fixed daily overtime above 0 for each day. - Populate logs for a whole month except Sundays with some absences in the same week of the month. - Populate logs for a whole month except Sundays with a negative daily overtime for each day.	- The monthly overtime is the total of the daily overtimes added. - Each day's absence will count as a -2 to the monthly overtime. The weekly overtime will be capped at -4. - The monthly overtime will be capped at the minimum of 0 hours.
Monthly processing depending on the type of the employee. This also satisfies the 8th and 9th success criteria. (mainly for non-normal type employees: *DiligencePay, NoExtraPay, CanceledBonus, FixedPay* Unless indicated otherwise.)	- Populating a *DiligencePay* employee with logs that indicate 1, 2, 3 and 4 absences in a month and calculate monthly extra. - Inputting random logs for *NoExtraPay, CanceledBonus, and FixedPay* employees.	- 1, 2 and 3 diligence absences will be stored in the database as 1, 2 and 3 respectively. 4 absences will be stored as 3, but when viewing the absent days, there will be 4 absences. - no changes to attributes. Their types speak for their extra wages.
Making sure that the email feature works and that the JSON information and file is accurate. This satisfies the 10th success criteria.	In a client account: - Choose different months to send (month and year specified) as JSON.	- A mail composer delegate should show up with pretty printed data for username, name, extraWageType, monthlyOvertime, and diligenceAbsences of all employees that logged in the month. The information should be printed as a JSON and attached as a JSON file. When sent, email is received by destination.

BIBLIOGRAPHY

Works Cited

MockFlow. “MockFlow - Wireframe Tools, Prototyping Tools, UI Mockups, UX Suite, Remote Designing.” *Mockflow.com*, mockflow.com. Accessed 4 Feb. 2021.

“UML Class Diagram Enumeration - Software Ideas Modeler.” *Www.softwareideas.net*, www.softwareideas.net/class-diagram-enum. Accessed 4 Feb. 2021.