**Criterion B: Record of Tasks**

| Planned Action | Planned Outcome | Time Estimated | Target Completion Date | Criterion |
| --- | --- | --- | --- | --- |
| Reaching out to the client and interviewing | To come out with a signed contract and a transcript of the interview. Also, to get an understanding of the problem as well as the expected functionality of the program. | 2 days | Thursday,  28 May | A |
| Sharing the Proposed Solution with the client | To get approval that the design and functionality fit the client’s needs and to get feedback on what to add to the program. | 1 day | Friday,  29 May | A |
| Researching about the swift programming language online | To understand how the benefits of swift can help in creating a fitting proposed solution. | Within 1 day | Friday,  29 May | A |
| Researching about core data in the swift programming language | Understand the basic idea of how to use core data as a database to allow for persistent data storage. | Within 1 day | Friday,  30 May | A |
| Designing what the app could look like | Consolidating sketches of the expected product with appropriate labels. | 1 day | Saturday,  30 May | B |
| Identifying classes/entities and examining the relationships between these classes/entities. | Finished UML diagrams that outline attributes, relationships, and methods. Finished entity-relationship diagrams. | 1 day | Saturday,  31 May | B |
| Roughly outline the logic behind the program’s functionalities. | Finished flowcharts with descriptions of each module of functionality. | 1 day | Saturday,  1 June | B |
| Learn how to convert data into JSON files, and how to send emails and attach files through code. | Understand how to use JSONEncoder and MFMailComposeViewController (MessageUI framework). | 5 days | Wednesday, 10 June | B |
| Learn how to use Core Data | Understand how to make Core Data entities, and to read/write entities in code. | 13 days | Thursday, 23 June | C |
| Setting up the database, and abstracting attribute types. | Create core data entities with their automatic entity classes/extensions. Wrap attribute types to local Swift data types. | 2 days | Monday, 30 November | C |
| Making a login page. | Account logins will be functional and will lead to respective account’s pages.  Entry/exit login will be functional and will record a timestamp for the employee in the database.  Errors will be displayed and handled. | 1 day | Thursday, 3 December | C |
| Testing month and day entity creations during entry/exit login. | When an entry/exit login is the first log of the day, a corresponding day entity is created in the database.  When the log is the first log of the month, a corresponding month entity is created in the database. | 3 days | Monday, 7 December | D |
| Making a list page on a tab view with the client’s information. | A list for the logged in client will display all the employees in alphabetical order. There is a search tab that allows the searching for employee names. Tapping on a name will lead to the selected employee’s information.  There is another tab that leads to the client’s information that can be changed. | 2 days | Wednesday, 9 December | C |
| Testing the deleting employees feature. | Employees can be deleted only by pressing an edit button and activating the edit mode. Deleting in the list will delete the employee in the database. | 1 day | Friday, 11 December | D |
| Making a page to add employees. | A page with username, password, name, and type as editable fields.  A button to add a new employee with the filled out information and error messages when the information is invalid. | 1 day | Saturday, 12 December | C |
| Testing adding employees | The add employee page can be brought up as a sheet from the employee list.  Adding an employee with an username that already exists is invalid and prompts an error message.  After adding an employee, the sheet is dismissed and the new employee is displayed in the list and in the database. | 2 days | Monday, 14 December | D |
| Making an employee information page. | The employee’s name, username, password, and extra wage type is displayed. Users can select a date for corresponding work time information.  The page can be viewed by employees and clients; clients can edit all fields, employees can only edit passwords. | 2 days | Thursday, 17 December | C |
| Making a page that displays absent and odd days in the month. | A list of two things. First, the days in the month when the employee is absent, including future days of the month. Second, the days in the month when the employee logged in/out an odd number of times. | Within 1 day | Friday, 18 December | C |
| Making a logs page. | A list of logs for a selected day is displayed.  For clients, logs can be added, changed, and deleted, and reflected in the database. Any changes will also refresh the monthly and daily overtimes and worktimes and diligence pay absences in the database, if appropriate. | Within 1 day | Saturday, 19 December | C |
| Coding algorithms for calculations and all cases for normal pay employees. | Create classes that calculate work time, daily overtime, weekly overtime, and monthly overtime. Have the class account for absences. | 4 days | Friday, 23 December | C |
| Testing that all cases are accurate. | After populating manual logs, the day and month work time and overtimes are shown in the database with the expected result. | 6 days | Thursday, 31 December | D |
| Coding algorithms for diligence pay employees. | Create a separate class for diligence pay employees that calculate for the number of diligence pay absences. Diligence absences are capped at 3. | Within 1 day | Saturday, 2 January | C |
| Testing and perfecting algorithms for the email and JSON feature of the program. | Create a class for emailing and JSON conversions. Email successfully sends and JSON data is accurate and readable. | 3 days | Tuesday, 5 January | D |
| Polish the application by using Forms and NavigationViews. | GUI is friendly and intuitive. GUI transitions are seamless. | 1 day | Wednesday, 6 January | D |
| Looking through the program with the client. | Each functionality is working and the client is satisfied. | 1 days | Monday, 18 January | E |
| Going through the application with the supervisor. | The supervisor and I will test all the functionalities in the program together. Fix any unnoticed errors/bugs. | 1 day | Wednesday, 20 January | E |
| Make final edits to the program. | All functionality is working as expected. | 7 days | Friday, 29 January | C |
| Meet with the client to discuss future recommendations. | The client and I will look through the program together again and discuss what is still missing/can be improved.  Discuss the implementation of the product. Train the client to get familiar with the program. | 1 day | Saturday, 30 January | E |
| Training employees on how to use the program. | Receive employee feedback on the program.  Gathering some employees and getting them familiar with how the program works, hopefully to move towards implementation. | 5 days | Monday, 22 Feb | E |