SPRINT THREE

CS2450-002, Team 2
Cody Strange-*Scribe and Information Manager*
Ethan Taylor-*GUI Developer*
Jaden Albrecht-*Team Manager*
Tyler Deschamps-*Chart and Milestone document builder*
Jordan Van Patten-*V&V and Tester*
Craig Sharp-*Stakeholder*

Table of contents

Management

Procedures

Verification & Validation

summary: we have added a quality assurance checklist to the verification and validation process so that we can track what standards and requirements our project currently meets. Along with this all of our documents have gone through the V&V process.

APPLICATION QUALITY ASSURANCE CHECKLIST

Purpose: The Application Quality Assurance Checklist is intended to ensure “Custom-Built” applications adhere to development practices that promote quality solutions. It is recommended that the project team familiarize themselves with this checklist during the Design stage to ensure the developed application meets the quality standards when reviewed in the Execute stage. This deliverable should be listed as a requirement in the Transition Agreement.

Project Name	EmpDat	**Project #**	SPR-3
Application Name		**Application #**	
Project Manager		**Delivery Manager**	
Date Completed	3/12/2022		

IMPORTANT NOTES FOR COMPLETING THIS DOCUMENT

Each section of the Application Quality Assurance Checklist template must be completed in full. If a particular section is not applicable to this project, then you must write **Not Applicable** and provide a reason.

Important Note: No sections are to be deleted from this document. This template is not to be modified in any manner. Text contained within << >> provides information on how to complete that section and can be deleted once the section has been completed.

1. Development Framework

Validation Questions	Yes	No	NA	Comments
1. Has the application been developed with the most recent OCIO-sanctioned version of the framework for the chosen technology?	☒	☐	☐	

2. Development IDE

Applications should be developed using an OCIO-sanctioned Integrated Development Environment (IDE). This will allow Application Services resources to build and debug source code as needed.

Validation Questions	Yes	No	NA	Comments
1. Has the application been developed using an Integrated Development Environment that was approved by the OCIO?	☒	☐	☐	

3. Decoupling Business Logic From The Presentation Layer

Whenever possible, developers should avoid using business logic in the presentation layer. The presentation layer should mainly be used for navigation throughout the application and presenting data to the user. For example, the use of Java code directly within JSP pages (i.e. Scriptlets) should be avoided. The preferred approach would be to use Tag Libraries (JSTL/EL).

Also, the Presentation Layer of Web applications should be developed using prevailing industry standards (e.g. using Stylesheets to position and control presentation elements, using relative positioning instead of absolute positioning, etc.).

Validation Questions	Yes	No	NA	Comments
1. Is the presentation layer of the application free from business logic?	☒	☐	☐	
2. Has the presentation layer of the application been developed in accordance with prevailing industry standards?	☒	☐	☐	

4. Record Locking / Concurrent Users

Applications should be developed in such a way that users' changes do not clash with each other or create the potential for data loss/corruption.

Validation Questions	Yes	No	NA	Comments
1. Have precautions been taken to avoid data clashes?	☒	☐	☐	

5. Passwords

A password helps authenticate a user when accessing a software application. Adherence to appropriate password management will help maintain the confidentiality, integrity, availability of the data maintained by the software application and reduce the risk of inappropriate access and use.

Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Does the system have functionality to allow the user to revise their password and force user account expiry?	☐	☒	☐	The user is able to change their password, but does not force user account expiry, because user accounts would be deactivated or expired by an admin
2. Does the system support protected storage of passwords with privileged user access? The system ***should not*** support passwords in clear text embedded either in the application code, automated scripts, or the database?	☐	☒	☐	
3. Does the system meet the standard password requirements?	☐	☒	☐	Password requirements will be added in at a later execution of our program
4. Are the passwords in the production environment different than those in a non-production environment?	☒	☐	☐	For now, passwords in the production environment do not currently have any requirements, but that will change later
5. Are all vendor supplied default passwords revised prior to placing the application in a production environment?	☐	☒	☐	
6. Are passwords for privileged accounts different than passwords for non-privileged accounts?	☒	☐	☐	They are different passwords if the user chooses, but the user is the one that chooses passwords

6. Logging and Auditing

Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Based on the application's Information Security Classification, does the application meet the logging functional control requirements?	☐	☐	☐	
2. Based on the application's Information Security Classification, does the application meet the auditing functional control requirements?	☐	☐	☐	

7. Modularized Code With No Duplication

As much as possible, code should be organized into small, separate modules to avoid code duplication and to make future code changes easier to implement.

Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Is the application modularized?	☒	☐	☐	
2. Has code duplication been avoided?	☒	☐	☐	

8. Consistency of Code

Code sections with similar functionality should be written in a clear, predicable, and consistent way. Using different approaches to achieve the same basic purposes should be avoided. Project teams consisting of multiple developers should ensure that the developers follow the same coding style and naming conventions.

Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Is the code written in a consistent manner throughout the application?	☒	☐	☐	
2. Have all developers followed the same coding style and naming conventions?	☒	☐	☐	
3. Have all developers followed the coding best practices as set out by the organization which owns the technology?	☒	☐	☐	

9. Code Comments

Code sections should be well documented with comments. At a minimum, each section of code (code unit) should have an introductory brief and accurate description to explain the code functionality. Any potentially confusing / non-intuitive sections of code should be commented thoroughly.

Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Does all application code include sufficient comments for support personnel?	☒	☐	☐	
2. Does each code unit have its own brief and accurate description?	☒	☐	☐	

10. Error Handling – End User

Error messages presented to the end user should contain only that information which will allow the user to take corrective action (e.g. “Invalid date, please reenter in YYYY-MM-DD format”). In the case of unhandled exceptions, messages should be generic. Avoid displaying system information in error messages such as server names, versions, and patch information, as well as application variables, paths, and other configuration information. Avoid messages that could potentially lead to system exploitation (e.g. “Incorrect Login” is acceptable while the message “Incorrect Password” is not).

Error handling logic should be robust enough to gracefully and meaningfully handle all errors which could be reasonably expected to occur from user interactions with the system. The text for error messages should be contained in a single location within the application code or database to facilitate quick additions and modifications by support staff.

Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Does the application handle all the errors that could reasonably be expected to occur?	☐	☒	☐	Error handling will be added in a future version
2. Do the error messages contain minimal but meaningful information?	☐	☒	☐	No error messages are in the code yet
3. Does the application avoid displaying system information in error messages?	☐	☒	☐	No error messages yet
4. Are the error messages kept in a single location?	☐	☒	☐	No error messages yet

11. Error Logging

Application errors should be logged for support personnel in database tables that will be directly accessible to Application Services personnel. SQL can then be used to aid in searches for specific errors.

Log files for individual server tiers (i.e. Web and Application tiers) should be kept in a single directory on each server. Also, log files should be saved on a daily basis with a time-date stamp on each file.

The error messages that are logged should contain information that is useful for support personnel (absolutely no sensitive or personal data), such as which module of code encountered the error and what the specific error was. Meaningful and detailed error messages are particularly important when troubleshooting unknown/unexpected errors. These should definitely be captured and logged.

Logging is also required for applications as well as batch/scheduled jobs. Logging logic within applications should be written in a modular way to facilitate the easy addition of new error messages.

Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Are all errors for the application being logged?	☒	☐	☐	We are manually logging any errors we find
2. Is logging being done on each server tier?	☐	☒	☐	
3. Are the logs kept in a single location / directory / database?	☒	☐	☐	We have a file where we report our bugs/errors
4. Are the logged errors specific enough to assist support personnel in troubleshooting production problems?	☒	☐	☐	
5. Is the code that logs the error messages written in a modular way?	☐	☒	☐	
6. Are the log files free of personally sensitive or identifiable information?	☐	☒	☐	

12. Field Validations

Where possible, validations should be performed on both the presentation layer and the business layer. In Java, for example, validations may be done using JavaScript within JSP pages (presentation layer), but should also be done within Java classes on the business layer. Also, validations should be performed in such a way that they cannot be bypassed by end-users (e.g. by disabling JavaScript). Field lengths and types within an application should be consistent with the column lengths and types declared within the underlying database tables.

Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Are fields being checked for the correct type (e.g. date, integer, etc.) and the correct range of values (e.g. 1 – 12 for month)?	☐	☒	☐	Field validation will be added in the next version
2. Are field values being validated with regular expressions where possible (e.g. validating email addresses and dates for valid formats)?	☐	☒	☐	
3. Do the validations resulting in error messages prevent data from being written to persistent storage (databases, files, etc.)?	☐	☒	☐	
4. Are the validations being performed within the business logic, as well as on the presentation layer?	☐	☒	☐	
5. Have the validations been written so that users cannot bypass them?	☐	☒	☐	
6. Are all of the field lengths and types within the application consistent with the column lengths and types declared within the underlying database tables?	☐	☒	☐	
7. Are user inputs being sanitized (without exceptions) according to OWASP recommendations?	☐	☒	☐	

13. Dates

When testing functionality that is built around date checks, the testers should use date values that occur in the past, on the target date, and in the future. Dates should also be validated in the context of the established business rules of the application (e.g. given a person's birth date, is he/she eligible to vote?). When dates are recorded in a database or log, they should include a timestamp and not just the month, day, and year. Timestamps will not be required in specific situations (such as a birth date field) where a timestamp does not make sense.

Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Does the application validate dates in a way that is consistent with the system design specifications and business rules?	☐	☒	☐	No validation is set up for dates, but will be added in the next version
2. Do all relevant dates include a timestamp?	☐	☒	☐	

14. Hard-Coded Values

Hard-coding of server names, database names, domain names, IP addresses, etc. within application code should be avoided. These values should be contained in a single configuration file or database that is not part of the application build, so that it can be easily maintained for different server environments (development, testing/staging, and production) and will not need to be modified when new changes are built and deployed.

Fixed values that are repeatedly used throughout application code should be declared in a single location and referenced appropriately, as needed, within the application. As a practical guide, a change to one of these values should occur within a single reference point.

Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Does the application code avoid use of hard-coded values?	☐	☒	☐	At least for testing purposes, our code is using hard-coded values
2. Do all hard-coded values reside exclusively within configuration and constant, centralized locations? (Central Locations that enable changes without recompiling source code)	☐	☒	☐	

15. System Testing

System testing should consist of negative testing, as well as positive testing. During positive testing ("Testing to Pass"), the testers will ensure that a program behaves as it should (in terms of navigation, processing, reading and writing records, etc.). During negative testing ("Testing to Fail"), the testers will ensure that a program does not behave in a way that it shouldn't (e.g. allowing a past date to be entered into a future date field).

Validation Questions	**Yes**	**No**	**Comments**
1. Did the application pass all positive tests?	☐	☒	There was one positive test that failed, but the rest of them passed. The test that failed was editing a user's ID, and then searching for that user afterwards
2. Did the application pass all negative tests?	☒	☐	
3. Have client testers completed the formal test plan in its entirety?	☐	☒	We have not made a formal test plan yet
4. Did the application pass all tests included in the formal test plan?	☐	☒	
5. Have all positive / negative test cases and test case results been documented?	☒	☐	

16. Regression Testing

Regression Testing is any type of software testing that seeks to uncover new errors or regressions in existing functionality after changes have been made to the software, such as functional enhancements, patches or configuration changes. Regression testing ensures functionality that was working yesterday is still working today. New functionality should be added to a system without impairing existing functionality or introducing bugs.

Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. As new capability is introduced, is the new capability tested?	☒	☐	☐	
2. Have all previous tests been reconducted with the results compared against expected results?	☐	☐	☒	Only one round of testing has been done so far
3. Is every capability of the software supported with a test case and is the test case added to the test case library to support final and future system testing?	☐	☒	☐	Not every capability is currently being tested with test case coding tests
4. As bugs are detected and fixed, is the test that exposed the bug recorded and regularly re-tested after subsequent changes are applied to the application?	☐	☐	☒	They are not being tested after being fixed by our written test codes, because we have not found any bugs that our code needs to retest yet

17. Load Testing/Volume Testing

The load/volume testing that is performed on an application should be reflective of the demands that could reasonably be expected to occur when the application goes into production. The testing should try to anticipate future system growth, data growth, and an increase in the number of active users.

Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Has the application been tested with a large number of concurrent users (i.e. a number of users that is representative of peak system usage)?	☐	☒	☐	
2. Has the application been tested with large numbers of concurrent transactions (i.e. a number of transactions that is representative of peak system usage)?	☐	☒	☐	
3. Did the system perform well with a large number of concurrent users?	☐	☐	☒	
4. Did the system perform well with a large number of concurrent transactions?	☐	☐	☒	
5. Are end-users satisfied with the application's performance and responsiveness during everyday use?	☐	☐	☒	

18. Certificates / Environment Software

Any certificates or special software that needs to be installed on a server stack for an application to function (e.g. virus scanning software, SSL Certificates, etc.) should be documented in the Operations Procedure Manual. Documentation should include the relevant expiration dates and the processes that must be followed for renewal. Also, application deployments in production environments should not be comprised of any trial versions of software. All proprietary and copyrighted software should be properly licensed for Government use.

Validation Questions	Yes	No	NA	Comments
1. Has all proprietary and copyrighted software been properly licensed for government use?	☐	☒	☐	
2. Have special software/certificate requirements been documented?	☐	☒	☐	
3. Does the documentation provide expiration dates and instructions for renewal?	☐	☒	☐	
4. Is the system / application free from trial versions of software?	☐	☐	☒	

19. Business Requirements - Traceability

All of the business requirements that have been captured and agreed upon by the project stakeholders should be fully met in the final version of the application that is transitioned over to Application Services. All required system functionality should also be fully satisfied by this final version.

Validation Questions	Yes	No	NA	Comments
1. Have all of the business requirements been met by the finished application?	☐	☐	☒	The application is not yet finished, but we have met the requirements given for this current phase
2. Has all of the required functionality been met by the finished application?	☒	☐	☐	The current application's requirements are met by what we currently have for our application

20. Source Code

Validation Questions	Yes	No	NA	Comments
1. Has the final approved version of the Application Code been provided to Application Services for use and maintenance during the Transition Period?	☐	☐	☒	We do not have the final version of our code
2. Has a test build been completed by Application Services using the code that has been handed over?	☐	☐	☒	
3. Has a copy of the version of Open Source Code used by the application been provided to Application Services for retention? (Links are not recommended)	☒	☐	☐	

20. Source Code

Validation Questions	Yes	No	NA	Comments
4. Have the 3rd party developer code / plug-ins (e.g. Axis2, Eclipse) been identified and provided to Application Services for the continued maintenance of the application? (Links to the utility not satisfactory, 3rd party products need to be provided)	☐	☐	☒	

21. Database Design

Industry best practices should be followed in the design of databases for production applications: tables normalized, exceptions documented, constraints enforced, and required fields completed (nulls not permitted). Also, if table keys are based on sequence numbers, each table should have its own sequence.

Validation Questions	Yes	No	NA	Comments
1. Have the database tables been normalized?	☐	☒	☐	
2. Keys based on sequence numbers have unique sequences.	☐	☐	☒	
3. Are all keys and required fields set to 'not null' in all tables of the database?	☐	☒	☐	
4. Have triggers, stored procedures, sequences, and constraints been properly utilized?	☐	☐	☒	

22. Transition To Support Personnel

The necessary server environments to support an application (development, test/staging, and production) should be fully constructed prior to transition and should be entirely consistent with each other with respect to Operating Systems, software versions, database versions, environment hardening, configuration, etc.

The Application Services resources supporting an application should be granted access to development, test/staging, and production environments (as appropriate) prior to transition.

Validation Questions	Yes	No	NA	Comments
1. Have accounts been created on all servers for the appropriate support personnel?	☐	☒	☐	
2. Have the necessary firewall rules been added to allow Application Services support personnel to access the relevant servers (i.e. via the Jump Box)?	☐	☒	☐	
3. Have all server environments (development, test/staging, and production) been fully created?	☐	☒	☐	

22. Transition To Support Personnel

The necessary server environments to support an application (development, test/staging, and production) should be fully constructed prior to transition and should be entirely consistent with each other with respect to Operating Systems, software versions, database versions, environment hardening, configuration, etc.

The Application Services resources supporting an application should be granted access to development, test/staging, and production environments (as appropriate) prior to transition.

Validation Questions	**Yes**	**No**	**NA**	**Comments**
4. Are all of the server environments entirely consistent with each other?	☐	☒	☐	

23. Checklist Exceptions

If the answer was 'no' for any of the checklist items above, please explain why in this section.

<< Please explain any exceptions using specific reference numbers from above. >>

PREPARED BY	
PROJECT MANAGER	
	Print name Signature Date (YYYY-MM-DD)
REVIEWED BY	
APPLICATION SERVICES TEAM LEAD	
	Print name Signature Date (YYYY-MM-DD)

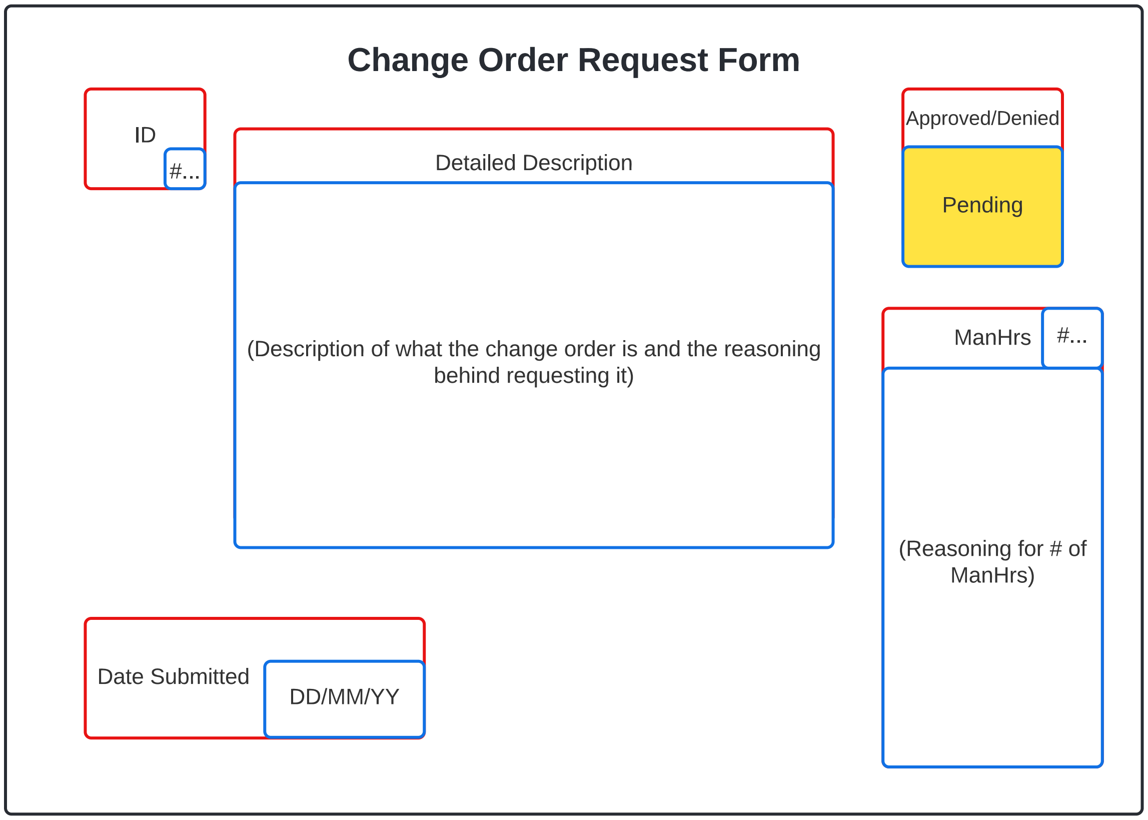
Change Control Board

Summary: The change control board keeps a record of all the change orders that have been brought to the shareholder to review. Each change order has an ID, the change that's being requested, a brief description of what the change entails, the reason we think the change order should be accepted, the status of the change order, and any comments the shareholder made on the change order.

ID	Request	Description	Reason	Status	Notes
1	Separate tabs for office info, personal info, and pay info	When viewing an employee, separate the information into multiple tabs	For security. In case someone is peeking on an admin's screen they won't see all of an employee's information at once	Under Review	Needs further explanation
2	Allow employees to see their current pay due for the pay period	A tab on the view page when viewing oneself that would allow one to see their pay accured during the current pay period	It's something an employee may want to see without having to calculate it themselves	Denied	Not our problem
3	Allow password changing/recover from the login screen	The ability to change one's password from the login screen, in case the user has forgotten their password	If an employee has forgotten their password, this feature would save them the hassle of contacting an admin to remind them of their password	Denied	Not our problem
4	Search other parameters(aside from the employee's last name and ID number)	The ability to search the employee database using any employee attribute, not just ID and last name	It would make searching more flexible	Accepted	Prepare change order
5	Logout button that returns user to the login screen	A button on every page that logs out the current user and returns the program to the login screen	In case an employee wants to exit the program without closing it	On Hold	Delay for future design/end of sprint six
6	Separate themes based on permission level	Depending on the user's permission level, a different visual theme will be shown in the GUI	Another visual indicator of permission level	Accepted	Prepare change order Use Tkinter themes to help user interface

Change Order Form

Summary: If a change order was accepted by the shareholder on the change control board then a change order form is filled out to give the shareholder more detailed information on what the change order entails, and the amount of man hours that the team will be requesting for said change order. The rest of the information on the form is for documenting purposes.



Change Order Request Form

ID	#...
Detailed Description	(Description of what the change order is and the reasoning behind requesting it)
Approved/Denied	Pending
ManHrs	#...
	(Reasoning for # of ManHrs)
Date Submitted	DD/MM/YY

Plans

Quality Control Plan

Summary: Quality is a foundation of any project that wishes to satisfy a shareholders requirements and expectations. Below is a list of practices that we follow to ensure that we are creating a quality product that matches or exceeds what the shareholder would expect.

- Definition of done
    - Has gone through the V&V process
    - Has been implemented into the current sprint document
    - Every member agrees that the final sprint document has been completed and is satisfied with the end result
- Consistency in standards
    - Everyone’s code, file names, document designs and styles all follow the same set of standards and guidelines that have been agreed upon.

    - Font styles and spacing is consistent throughout all documents to ensure readability
- Constant communication
    - Two formal meetings a week
    - One informal meetup a week
    - Constant updates and light communication through messaging app
- Efficient code
    - Pseudocode follows standards and guidelines to ensure readability
    - Has gone through the V&V process
    - Has been thoroughly tested
    - Bugs are properly documented to ensure that they can be efficiently dealt with

SPR-3 Risk Management Plan

Summary: The Risk Management Plan covers the major risks that we found in completing sprint three. It determines what risks we might run into while completing the sprint the likely hood of that risk occurring and the severity of the risk. Because the scope of the project/sprint is rather small we went with a simple plan that only has low, medium, and high for probability and impact. The higher the probability the more precautions need to be taken and the higher the impact the higher the priority of fixing said risk if multiple were to occur at the same time. If two risks of the same impact level occur then the team would need to decide which one is more pressing at the time and take care of that one first.

ID	Category	Description	Consequence	Probability	Impact	Risk Minimalization plan	Contingency plan
1	Team	Team member unexpectantly quits	Tasks given to said team member would have to be completed by someone else. And any progress said team member made that wasn't saved to google drive will be lost.	Low	High	Keep in contact with all team members and make sure that everyone is on the same page.	Emergency team meeting, where the previous team member's remaining tasks are divided up and estimated hours are recalculated. Keep shareholder up to date so they understand reasoning behind possible delays
2	Pseudocode	Psuedocode that will be used has logic errors that leads to certain parts of the program's design to be unusable.	Have to come up with a different design for sections of the code as the team is programming, which leads to more hours and more bugs in the code	Medium	Medium	Following the V&V process and making sure multiple people read the psuedocode and agree that it's logic is sound	Depending on the priority of that specific section, may delay writing that part of the code until new psuedocode is written for that portion
3	Shareholder	Denies a change order request	wasted time on coming up with change order request, less opportunities for extra credit	High	Low	Discuss change orders with team and compare it with requirements specifications to determine whether or not it woudl be a valid change order	Discuss with the team why the change order was denied and apply that knowledge to all future change orders
4	Documents	Documents are in some way wiped from a specific team members computer	Loss of valuable information that could take days worth of time to replicate	Low	High	Having multiple copies of documents between specific computers and online storage such as a google drive	Have an emergency meeting to figure out exactly what was lost and how much time it would take to replace it
5	Shareholder	Team is unable to get in contact with Shareholder	Team cannot review documents with the shareholder and cannot request any additional change orders	low	medium	Try to understand the shareholder's schedule and schedule meetings far enough ahead that the shareholder isn't busy with other teams	Have a meeting where we do an extra review of each document that we would go over with the shareholder

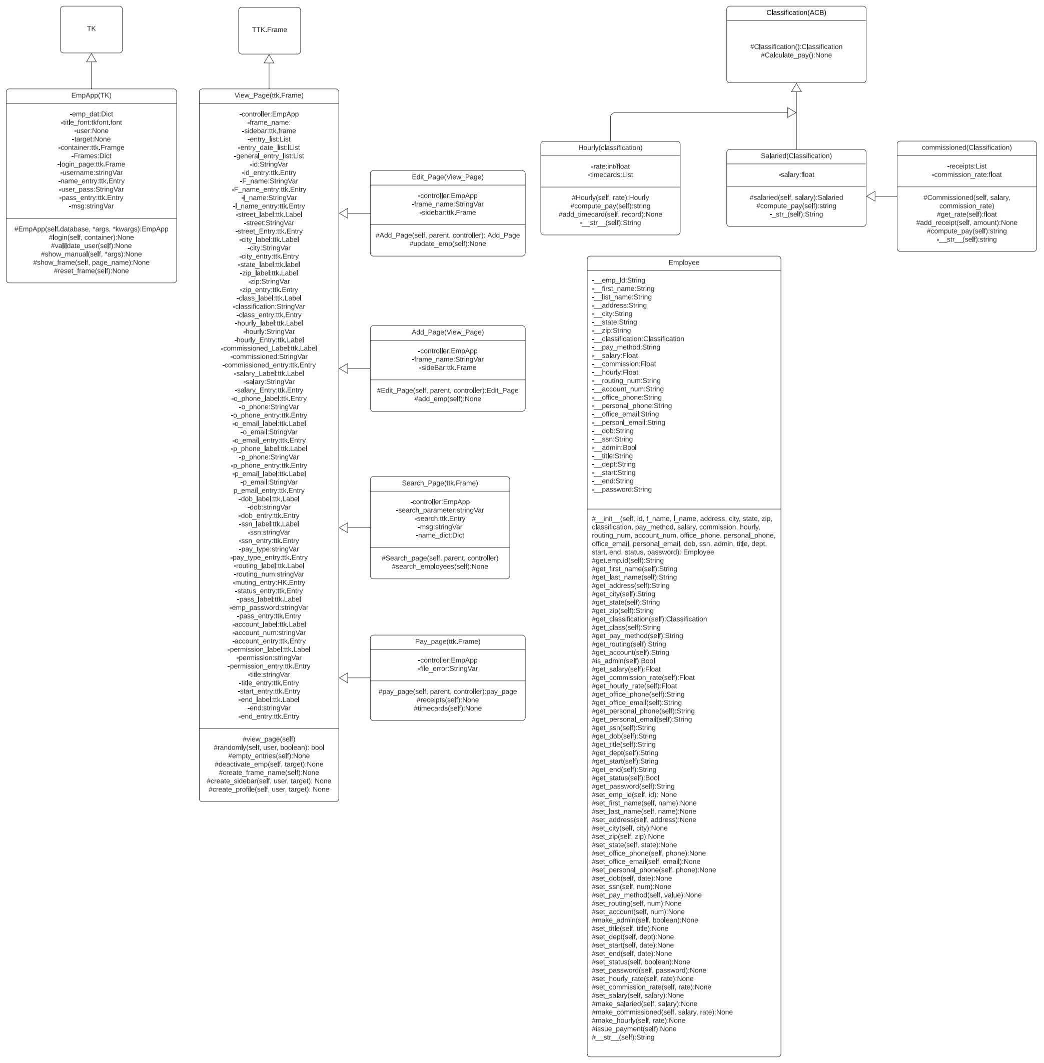
Maintenance Plan

Summary: In this sprint maintenance mainly involves updating previous charts and documents to have relevance in sprint three from sprint two if needed. Along with maintaining the google drive creating new folders as old folders get crowded and files can be grouped together.

Programming

Low-Level Design

UML diagrams



Pseudocode

# CLASS AND METHOD PSEUDOCODE:

1. Application Class - Main Parent Class
2. Inherits from Tk
3. Constructor:

4. Uses parent constructor
5. Set title, geometry, and resizable values
6. Dictionary container for holding all of the different frame classes (Ex: self.frames = {}) Keys are names for the different pages (Ex: 'login_page')
7. Values are call to constructor for that page (Ex: self.frames['login_page'] = Login_Page(--login constructor args--))
8. Included Frames:
9. Login_Page, View_Employee_Page, Edit_Employee_Page, Search_Employee_Page, Add_Employee_Page
10. employees dictionary - holds data for all employees in database
11. Keys are employee ids
12. Values are employee objects with those ids
13. Initialize as empty list
14. timecards list - list of lists containing an employees ID and then all their currently saved timecards
15. Initialize as empty list
16. receipts list - list of lists containing an employees ID and then all their currently saved receipts
17. Initialize as empty list
18. change_frames function
19. Takes a frame name as an argument (Ex: 'login_page')
20. Uses this name to get the value from the self.frames dictionary, which is the constructor
21. Calls reset() and tkraise() functions on that frame
22. load_employees function - for filling the employees dictionary with the info in the employees.csv file
23. Opens the employees.csv file
24. Traverses through the file, creating Employee objects based on data therein
25. Adds Employee object to dictionary with Employee ID as key
26. Login Page Frame Class
27. Inherits from Application Class
28. Constructor:
29. Create label with text 'Login' at grid column=0 row=0 columnspan=3
30. Create label space at the bottom of the page dedicated to error messages, initially blank, at grid column=0 row=4 columnspan=2
31. Create 2 Labels for Employee ID, and Password - column=0, rows=2 and=3
32. Create 2 Entry boxes next to matching labels - column=1, rows=2 and=3
33. On password entry, show='*'
34. Create button with text 'Login' at grid column=1 row=4 sticky=W
35. Button calls validate_login method, passing entry box values as arguments
36. Bind <Enter> key to validate_login method
37. validate_login:

38. Takes an employee ID and password string as arguments
39. If employee ID not in employee dictionary keys, or if employee object with said ID is deactivated:
40. Set error message to 'Invalid ID!
41. Set entry box values to empty
42. Set focus back to ID entry box
43. Elif password != employee object password:
44. Set error message to 'Incorrect ID or Password!'
45. Set entry box values to empty
46. Set focus back to ID entry box
47. Else:
48. Call the parent class (application class) change_frame function with a value of 'view_employee_page' as the argument
49. View Employee Frame Class
50. Takes user object and employee object as arguments
51. Inherits from Application Class
52. Constructor:
53. Create frame for buttons in grid column=0 row=1
54. Create frame for labels and entry boxes in grid column=1 row=1
55. Create Label with text 'Viewing {Employee first and last name}' in grid column=0 row=0 columnspan=2
56. Create frame for Add and Update buttons, used for Add employee and Edit employee functions, in grid column=0 row=2
57. If employee argument == user object or None, user's information will be displayed
58. create_menu method:
59. Takes user object, and employee object as arguments
60. If user permission = admin:
61. Create button with text 'Edit Employee' and command 'edit_employee'
62. Create button with text 'Add Employee' and command 'add_employee'
63. Create button with text 'Deactivate Employee' and command 'deactivate_employee'
64. Create button with text 'Search Employee' and command 'search_employee'
65. Create button with text 'Payroll' and command 'payroll'
66. Create button with text 'Help' and command 'Help'
67. Else:
68. If user == employee (general permission viewing themselves):
69. Create button with text 'Edit Employee' and command 'edit_employee'
70. Create button with text 'Search Employee' and command 'search_employee'
71. Create button with text 'Help' and command 'Help'
72. create_profile method:
73. Takes user object and employee object as arguments
74. If user = employee (someone viewing themselves) or user permission = admin:

75. Create labels and entries for every Employee Object attribute, with labels in the left column and entries in the right
76. If user != employee object, fill out entry boxes with employee object info, else use user info
77. Set entry boxes to 'readonly'
78. add_employee method:
79. Set all entry boxes to empty and remove 'readonly' state
80. Disable all side menu buttons except 'help'
81. Create button with text 'Add New Employee' and command 'update_database'
82. edit_employee method:
83. Takes user object as an argument
84. If user permission != admin, remove 'readonly' state on all entry boxes except employee ID, Office Phone, Office Email, Classification, Hourly, Salary, and Commission
85. Disable all side menu buttons except 'help'
86. Create button with text 'Update Employee' and command 'update_database'
87. deactivate_employee method:
88. Pops up a confirmation window
89. On confirmation, automatically update end_date and status employee attributes
90. search_employee method:
91. calls parent class (Application class) change_page function with 'search_page' as argument
92. payroll method:
93. outputs payroll.csv, erases old timecard and receipt data
94. help method:
95. Takes a page as an argument
96. Pops up a text window with the user manual, on the section pertaining to specified page
97. update_database method:
98. Pop-up a confirmation window
99. On confirmation, update database with new employee
100. If employee already in database, replace with updated version
101. Search Page Class:
102. Takes user object as argument
103. Inherits from Application Class
104. Constructor:
105. Create labels for search parameters and search bar
106. Create radio buttons for search parameters (Last name and employee id)
107. Create entry box for search bar
108. Apply filter to search bar to supply drop-down list of matching employees
109. On selection of an employee, call parent class (Application class) change_page function with 'view_employee' as argument

#CSV FILE FROM EMPLOYEE LIST PSEUDOCODE

```
1. Import Csv library
2. Import the DB_Access_Module
3. Define read_from_csv function with a csv_path parameter.
4. Declare an empty emps_from_csv list.
5. Declare an empty reader_list list.
6. Open csv file with csv_path in reading mode with loop as emps.
7. Declare a variable named reader and set it equal to csv reader of emps.
8. For each row in the reader, append them to reader_list.
9. Declare csv_keys as the first element in reader_list.
10. Loop through each reader_list element except for the first one.
11. Declare and define temp_dict as a dict that zips up the current reader_list element
12. with the csv_keys.
13. Append temp_dict to emps_from_csv.
14. Return emps_from_csv.
15. Define emps_to_csv function with csv_file_name as a parameter.
16. Declare emp_database and set it to current path to employees database json file.
17. Declare emps and set it as the list that's returned from get_json() function that
18. uses emp_database as an argument.
19. Declare file and assign it to an open file with csv_file_name as name, and write as open type.
20. Declare write and assign it to csv write for the file variable.
21. Write a row in write of the first element of employees which will just assign keys.
22. Loop through each employee as emp in emps.
23. Declare row and assign to empty list.
24. Loop through each key value pair in emp as key and value.
25. Append a the value of the given pair as a string to row.
26. Write a row in write of the row list.
27. Close the file.
28. Define timecards_to_csv function with csv_file_name as parameter.
29. Declare timecards_datatbase and set it to current path to timecards database json file.
30. Declare timecards and set it as the list that's returned from get_json() function that
31. uses timecards_database as an argument.
32. Declare timecard_rows and set it to an empty list.
33. Loop through each of the timecards as tc.
34. Declare matched_row as a loop boolean indicator of finding a match and set it to false.
35. If length of timecard_rows is greater than one,
36. then loop through each timecard_rows as row.
37. If the row id is equal to the tc's employee id,
38. then append the current tc's time to the row vals list.
```

39. Set matched_row equal to true.
40. Break out of the lower loop.
41. If there was no matched row(matched_row is false),
42. then append a new dictionary to timecard_rows with 'id' as the current tc's emp_id and vals as a list with the first element as the current tc's time.
43. After loop is done, declare file and set it to an open file with with csv_file_name as name, and write as open type.
44. Declare write and set it to csv write for the file variable.
45. Loop through each timecard_rows as row.
46. Declare new_row and set it to a list with the current row's id casted into a string.
47. Loop through each value in the current rows vals.
48. Append each val casted into a string to the new_row list.
49. Write the new_row list with the write variable.
50. After loop is done, close the file.

51. Define receipts_to_csv function with csv_file_name as parameter.
52. Declare receipts_datatbase and set it to current path to receipts database json file.
53. Declare receipts and set it as the list that's returned from get_json() function that uses receipts_database as an argument.
54. Declare receipt_rows and set it to an empty list.
55. Loop through each of the receipts as rc.
56. Declare matched_row as a loop boolean indicator of finding a match and set it to false.
57. If length of receipt_rows is greater than one,
58. then loop through each receipt_rows as row.
59. If the row id is equal to the rc's employee id,
60. then append the current rc's receipt val to the row vals list.
61. Set matched_row equal to true.
62. Break out of the lower loop.
63. If there was no matched row(matched_row is false),
64. then append a new dictionary to receipt_rows with 'id' as the current rc's emp_id and vals as a list with the first element as the current rc's receipt val.
65. After loop is done, declare file and set it to an open file with with csv_file_name as name, and write as open type.
66. Declare write and set it to csv write for the file variable.
67. Loop through each receipt_rows as row.
68. Declare new_row and set it to a list with the current row's id casted into a string.
69. Loop through each value in the current rows vals.
70. Append each val casted into a string to the new_row list.
71. Write the new_row list with the write variable.
72. After loop is done, close the file.

#USER INTERFACE AND DATABASE INTERACTION PSEUDOCODE

```
1. Import Json library
2.
3. Declare database_path and initialize it appropriately.
4. Declare filter_options as empty dictionary.
5. Define store_json function with a list parameter named data.
6. Declare json_obj variable as a "json.dumps" method with data as first parameter and indent set to four as second parameter.
7. Declare file variable as an "open" method with database_path and a "w" for write as parameters.
8. Write to file with the json_obj as parameter.
9. Close file.
10. Define get_json function.
11. Declare an empty emp_table list.
12. Declare an empty emps list.
13. Open Database file with database_path as read.
14. Set emp_table to file read.
15. Parse Json emp_table into emps.
16. Close file.
17. Return emps.
18. Define search_database with args as parameter.
19. Declare emps and set it result of get_json.
20. Declare column and set it to search_filter radio button widget value.
21. Declare search_data and initialize to search_value input widget value.
22. Loop through each of the emps as emp and compare each emp's specified column for the search value given.
23. If the value given matches the value in the column of the current employee being observed,
24. create a new Employee class object from the data in the the employee dictionary.
25. Return the new Employee object.
26. Else, continue the loop through employees.
27. If employee is never found, return that the employee was not found.
28. Define create_employee with emp_object from Employee class as parameter.
29. Declare emps and set it to result of get_json function.
30. Declare new_emp dictionary with
31. "id" as the length of emps plus one,
32. "first_name" as emp_object's first name data member,
33. "last_name" as emp_object's last name data member,
34. "street_address" as emp_object's street address data member,
35. "city" as emp_object's city data member,
```

36. "state" as emp_object's state data member,
37. "zip" as emp_object's zip data member,
38. "office_phone" as emp_object's office phone data member,
39. "pay_type" as emp_object's pay type data member,
40. "date_of_birth" as emp_object's date of birth data member,
41. "social_security_number" as emp_object's social security number data member,
42. "start_date" as emp_object's start date data member,
43. "bank_routing_num" as emp_object's bank routing number data member,
44. "bank_account_num" as emp_object's bank account number data member,
45. "permission_id" as emp_object's permission id data member,
46. "title" as emp_object's title data member,
47. "department" as emp_object's department data member,
48. "office_email" as emp_object's office email data member,
49. "personal_email" as emp_object's personal email data member.
50. Substitute any empty fields with empty strings.
51. Append new_emp to emps list.
52. Call store_json function and pass in emps as argument.
53. Define update_employee with emp_object from Employee class as parameter.
54. Declare emps and set it to result of get_json function.
55. Loop through each of the emps as emp.
56. If the id of the emp matches the id of the emp_object then
57. set emp's "first_name" as emp_object's first name data member,
58. set emp's "last_name" as emp_object's last name data member,
59. set emp's "street_address" as emp_object's street address data member,
60. set emp's "city" as emp_object's city data member,
61. set emp's "state" as emp_object's state data member,
62. set emp's "zip" as emp_object's zip data member,
63. set emp's "office_phone" as emp_object's office phone data member,
64. set emp's "pay_type" as emp_object's pay type data member,
65. set emp's "date_of_birth" as emp_object's date of birth data member,
66. set emp's "social_security_number" as emp_object's social security number data member,
67. set emp's "start_date" as emp_object's start date data member,
68. set emp's "bank_routing_num" as emp_object's bank routing number data member,
69. set emp's "bank_account_num" as emp_object's bank account number data member,
70. set emp's "permission_id" as emp_object's permission id data member,
71. set emp's "title" as emp_object's title data member,
72. set emp's "department" as emp_object's department data member,
73. set emp's "office_email" as emp_object's office email data member,
74. set emp's"personal_email" as emp_object's personal email data member.
75. Break loop.
76. Call store_json function and pass in emps as argument.

Class Tests

Summary: We will be running tests by page, and currently we have two pages tests written out in pytest. The document lists the name of each test category, the description of what that category is testing for, the amount of individual tests in that category, and the percent of those tests past. The pass/fail box will be labeled 'red' for not fully passed, and 'green' for completely passed.

Login Page				
Name	Description	Count	%Passed	P/F
test_correct	tests that using the correct username and ID does NOT trigger the "Incorrect password or ID" message	8	0%	🟥
test_incorrect	tests that using the incorrect username but correct password does trigger the "Incorrect password or ID" message	8	0%	🟥
test_incorrectP	tests that using the incorrect password but correct username does trigger the "Incorrect password or ID" message	11	0%	🟥

View Page				
Name	Description	Count	%Passed	P/F
test_nonAdminPass	tests that the information shown on the general users viewpage matches their informatin in the database	4	0%	🟥
test_AdminPass	tests that the information shown on the admin users viewpage matches their information in the database	4	0%	🟥

Bug Tracking Software

summary: We will be using backlog as a way to keep track of bugs that are found in the program

Bug Tracking Report Template

summary: This is a template that will be filled out for each bug that is found in the program so that it can be well documented and dealt with.

Bug ID	Bug Found Date	Description	Severity	Start Date	Note	End date
1	DD/MM/YY	(Description of the problem that the bug creates)	low/med/high	DD/MM/YY	(Any information that would prove useful to someone who will be attempting to fix the bug in the future)	DD/MM/YY

Confirmation

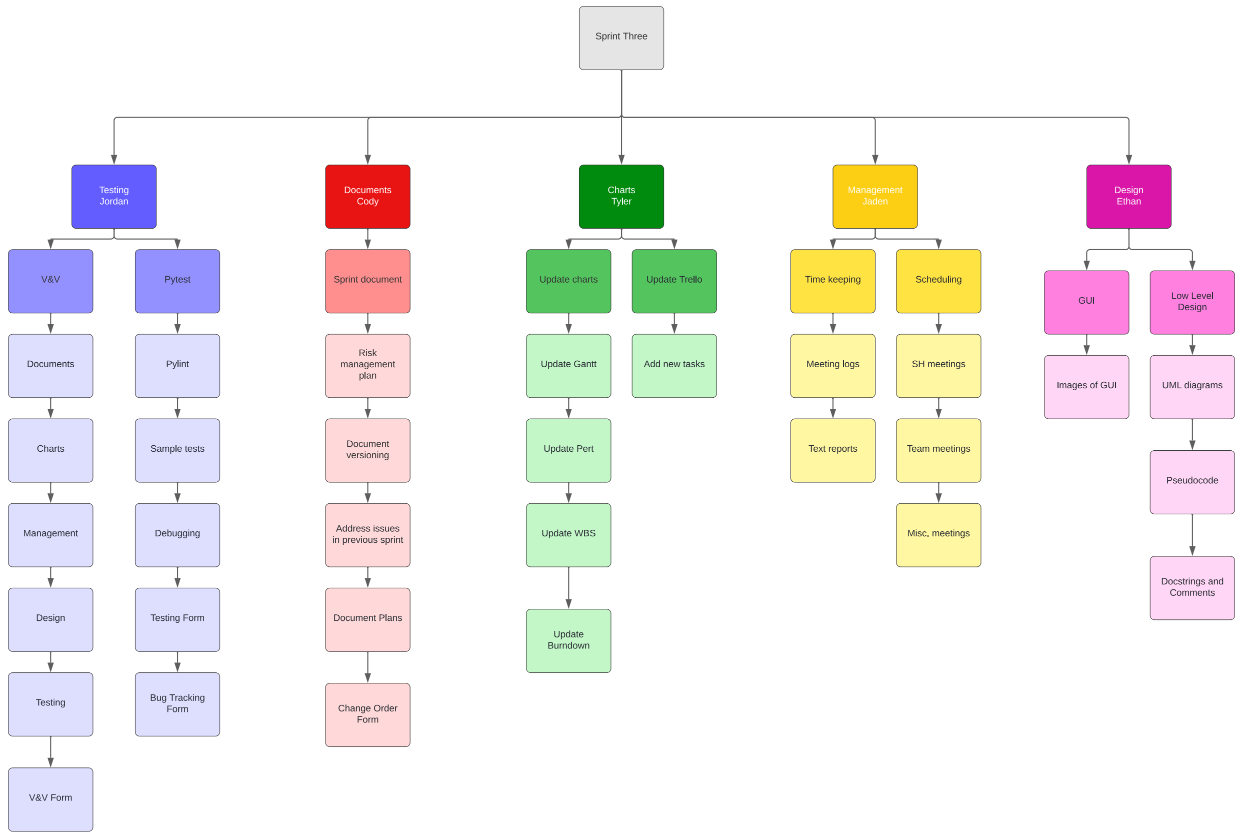
Summary: We have confirmed that the LLD design will function with the desired Hardware, Network configuration, Development environment, and Source Code Control

Charts/Forms

SPR-3 Work Breakdown Structure

Chart

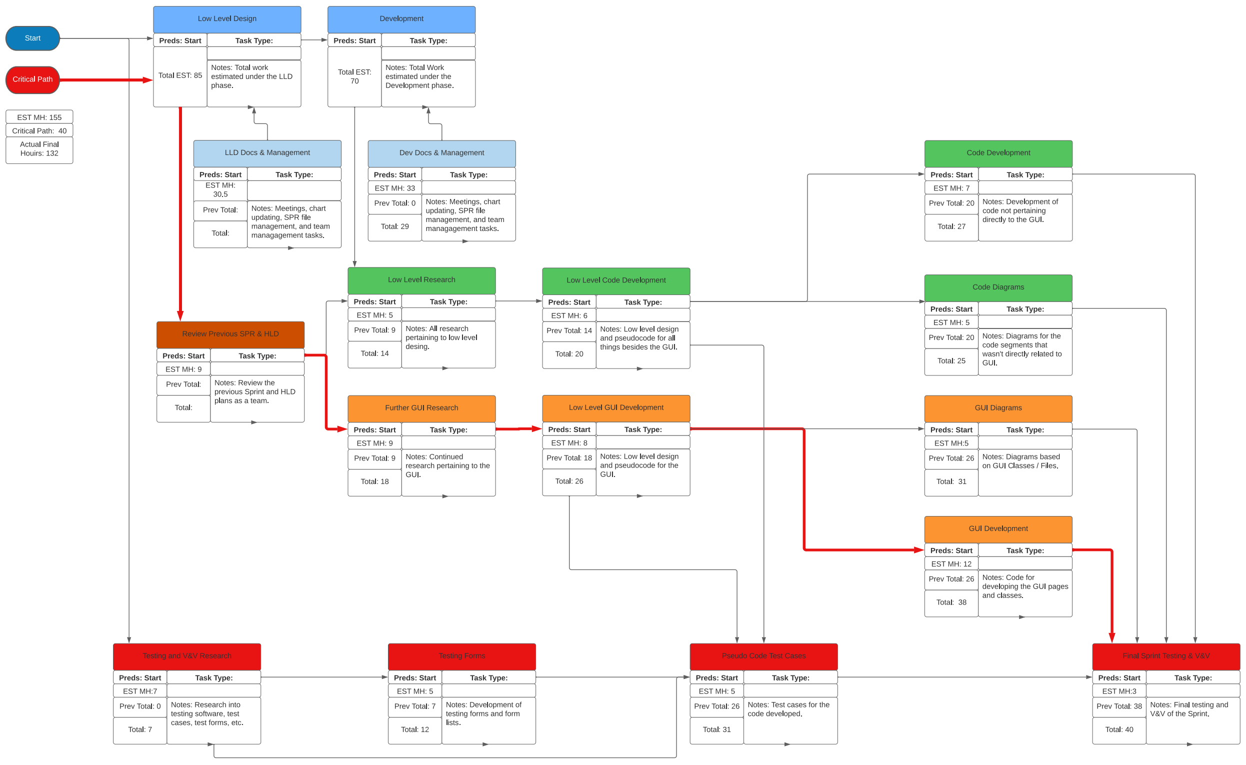
Summary: This is a Work Breakdown Structure for Sprint three. It details each major tasks that need to be completed and who is in charge of managing said tasks, making sure that they are completed. All tasks listed are relevant to SPR3 and should be completed by the end of the sprint. It is important to not that the listed tasks do NOT represent which tasks each team member will be expected to complete.



SPR-3 Pert Chart

Chart

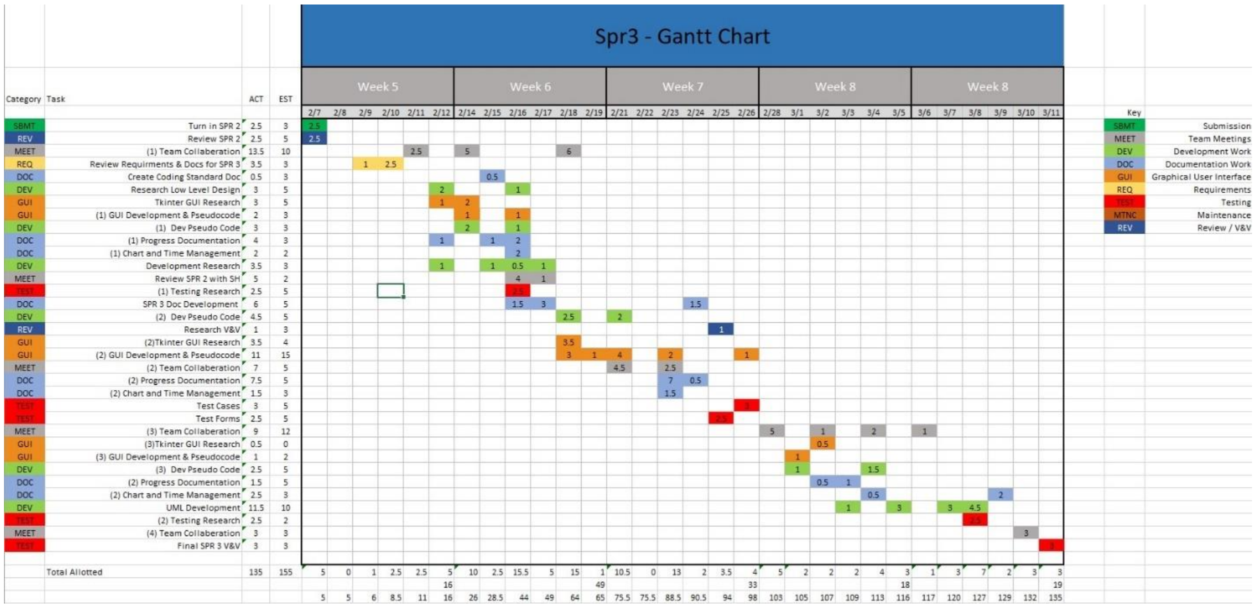
Summary: This is the Sprint Three Pert Chart; it details the time estimates for the tasks in sprint three and the prerequisites for specific tasks. Along with the critical path that lists the longest chain of tasks.



SPR-3 Gantt Chart

Chart

Summary: This is the Sprint Three Gantt Chart; it will list each of the tasks that we will be doing in sprint three, with the estimated time and actual time for each task.



Spr3 - Gantt Chart

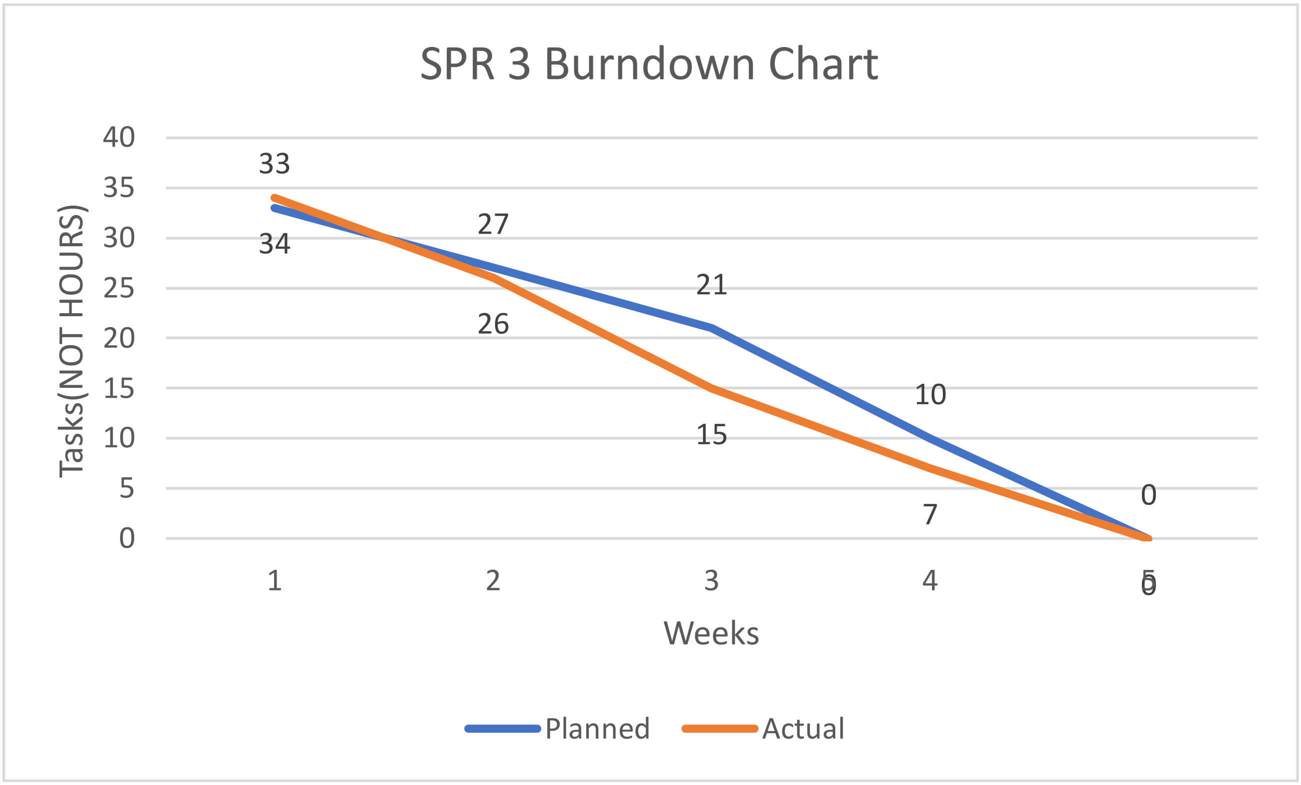
Category	Task	ACT	EST	Week 5						Week 6						Week 7						Week 8						Week 8					
				2/7	2/8	2/9	2/10	2/11	2/12	2/14	2/15	2/16	2/17	2/18	2/19	2/21	2/22	2/23	2/24	2/25	2/26	2/28	3/1	3/2	3/3	3/4	3/5	3/6	3/7	3/8	3/9	3/10	3/11
SBMT	Turn in SPR 2	2.5	3	2.5																													
REV	Review SPR 2	2.5	5	2.5																													
MEET	(1) Team Collaberation	13.5	10					2.5		5				6																			
REQ	Review Requirments & Docs for SPR 3	3.5	3			1	2.5																										
DOC	Create Coding Standard Doc	0.5	3								0.5																						
DEV	Research Low Level Design	3	5						2			1																					
GUI	Tkinter GUI Research	3	5						1	2																							
GUI	(1) GUI Development & Pseudocode	2	3							1		1																					
DEV	(1) Dev Pseudo Code	3	3							2		1																					
DOC	(1) Progress Documentation	4	3						1		1	2																					
DOC	(1) Chart and Time Management	2	2									2																					
DEV	Development Research	3.5	3						1		1	0.5	1																				
MEET	Review SPR 2 with SH	5	2									4	1																				
TEST	(1) Testing Research	2.5	5									2.5																					
DOC	SPR 3 Doc Development	6	5									1.5	3						1.5														
DEV	(2) Dev Pseudo Code	4.5	5											2.5		2																	
REV	Research V&V	1	3																	1													
GUI	(2)Tkinter GUI Research	3.5	4											3.5																			
GUI	(2) GUI Development & Pseudocode	11	15											3	1	4		2			1												
MEET	(2) Team Collaberation	7	5													4.5		2.5															
DOC	(2) Progress Documentation	7.5	5															7	0.5														
DOC	(2) Chart and Time Management	1.5	3															1.5															
TEST	Test Cases	3	5																		3												
TEST	Test Forms	2.5	5																	2.5													
MEET	(3) Team Collaberation	9	12																			5		1		2		1					
GUI	(3)Tkinter GUI Research	0.5	0																					0.5									
GUI	(3) GUI Development & Pseudocode	1	2																				1										
DEV	(3) Dev Pseudo Code	2.5	5																				1			1.5							
DOC	(2) Progress Documentation	1.5	5																					0.5	1								
DOC	(2) Chart and Time Management	2.5	3																							0.5					2		
DEV	UML Development	11.5	10																						1		3		3	4.5			
TEST	(2) Testing Research	2.5	2																											2.5			
MEET	(4) Team Collaberation	3	3																													3	
TEST	Final SPR 3 V&V	3	3																														3
	Total Allotted	135	155	5	0	1	2.5	2.5	5	10	2.5	15.5	5	15	1	10.5	0	13	2	3.5	4	5	2	2	2	4	3	1	3	7	2	3	3
									16						49						33						18						19
				5	5	6	8.5	11	16	26	28.5	44	49	64	65	75.5	75.5	88.5	90.5	94	98	103	105	107	109	113	116	117	120	127	129	132	135

Key	
SBMT	Submission
MEET	Team Meetings
DEV	Development Work
DOC	Documentation Work
GUI	Graphical User Interface
REQ	Requirements
TEST	Testing
MTNC	Maintenance
REV	Review / V&V

SPR-3 Burndown Chart

Template

Summary: The burndown chart shows the amount of time expected to be spent on tasks in the sprint against the actual amount of time spent for sprint three.



Research

Sources

- Beginning Software Engineering(chapter 3)
- CS2450 Lecture #6(2/2/2022)

Meeting Logs

Meeting Log#9

Meeting Information
Team #: T2-002
Meeting log #: 9
Current Sprint: SPR3
Date: February 14, 2022
Time: 6:55pm – 7:58pm (MT)
Location: MS Teams (watch video here)
Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Tyler Deschamps, Jordan Van Patten
Next team meeting scheduled for: February 18, 2022, 7:00pm (MT)

Progress From Previous Meeting
Ethan Taylor:
Thoroughly read chapter 7 (100%)
Tkinter research (100%)
Login GUI page pseudocode (100%)

Jaden Albrecht:
Thoroughly read chapter 7 (100%)
Keep meeting logs up to date (100%)
UML research
Created version history folders in google drive (100%)
Tkinter research

Cody Strange
Thoroughly read chapter 7 (100%)
Review all SPR3 documents (100%)

Tyler Deschamps:
Thoroughly read chapter 7 (100%)
Search employee database pseudocode (60%
Tkinter research

Jordan Van Patten:
Thoroughly read chapter 7 (100%)

Topics Discussed
File naming conventions for classes
Change board request document
UML diagrams
Low-level design for user login page (verification of employee data), and other GUI pages (classes)
Methods for searching and sorting employee data

Options for testing software

Obstacles Encountered
No obstacles

Finished Items
All members have thoroughly read chapter 7
Developed a method for organizing and searching employee data
Pseudocode for user login page
pseudocode for search employee database page

Unfinished Items
Schedule next shareholder meeting
Create SPR3 document
Low-level design of view/add/edit page
UML diagrams
Login GUI
View/add/edit GUI
Search GUI
Research PyTest
Tested all components using PyTest
Research debugging software
Develop more effective algorithm for searching and sorting employee data (sorting methods will appear in change order request)

Tasks Until Next Meeting
Research tkinter
Research debugging software
Research PyTest
Work on low-level design for view/add/edit page
Login GUI
View/add/edit GUI
Search GUI
V&V documents
Risk management plan
Team folder naming conventions document

Notes
Tyler has suggested we use JSON module for handling employee objects and data, which will help with extracting data from both employees.csv files provided in CS1410 and CS2450

Meeting Log#10

Meeting Information
Team #: T2-002
Meeting log #: 10
Current Sprint: SPR3
Date: February 18, 2022
Time: 6:58pm – 8:06pm (MT)
Location: MS Teams (watch video here)
Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Tyler Deschamps
Next team meeting scheduled for: February 21, 2022, 7:00pm (MT)

Progress From Previous Meeting
Ethan Taylor:
Login GUI (100%)
Search GUI (100%)
GUI research and experimentation
Tkinter research
Low-level design for view/add/edit page

Jaden Albrecht:
Keep meeting logs up to date (100%)
UML research
Tkinter research
PyTest research

Cody Strange
Review all SPR3 documents
PyTest research
Watch previous team meeting recording in 2x speed (100%)
Fix SPR2 document requirements
Reorganize google drive folders (100%)
Create SPR3 document (100%)

Tyler Deschamps:
Update PERT chart (100%)
Update Gantt chart (100%)
Update burn-down chart (100%)
Low-level design for search employee database page (100%)

Jordan Van Patten:
Team folder naming conventions document (100%)
Research PyTest
Research debugging software

Topics Discussed
Change board request document
UML diagrams
Low-level design for user login page (verification of employee data), and other GUI pages (classes)

How to use PyTest

Obstacles Encountered
Ethan reported a bug in the view/add/edit GUI regarding loaded employee data fields. The preloaded data values were printed to the console but were not displayed in text entry fields. This issue is being looked at and will hopefully be resolved before the next meeting on February 21, 2022.

Finished Items
Login GUI
Search GUI
Created SPR3 document
All charts have been updated for SPR3
Meeting logs are up to date
Established class folder naming conventions
Low-level design for all pages (GUIs)

Unfinished Items
Schedule next shareholder meeting
UML diagrams
Testing pseudocode
Wired GUIs with database
Write test code files using PyTest
Test all components using PyTest
Research debugging software
change order request)
Risk management plan
QA plan
V&V documents

Tasks Until Next Meeting
Research debugging software
UML diagrams
Research PyTest
QA plan
V&V documents
Risk management plan
Wire all GUIs to database
Add theme to GUIs

Notes
Our main focus going forward is writing test code to ensure the program meets all specified requirements authorized by the shareholder. We also plan to have a full UML diagram for each GUI class to provide an understanding of class hierarchies.

Meeting Log#11

Meeting Information
Team #: T2-002
Meeting log #: 11
Current Sprint: SPR3
Date: February 21, 2022
Time: 6:40pm – 7:46pm (MT)
Location: MS Teams (watch video here)
Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Tyler Deschamps
Next team meeting scheduled for: February 25, 2022, 7:00pm (MT)

Progress From Previous Meeting
Ethan Taylor:
Search GUI (100%)
View GUI (100%)
Low-level design for view/add/edit page (100%)

Jaden Albrecht:
Keep meeting logs up to date (100%)
UML research
PyTest research

Cody Strange
Review all SPR3 documents (100%)
PyTest research
SPR3 document (70%)

Tyler Deschamps:
Employee database pseudocode (70%)
CSV pseudocode (50%)
JSON research

Jordan Van Patten:
Research PyTest
Research debugging software

Topics Discussed
Change board request document
UML diagrams
Test code files using PyTest
Final changes to GUIs (such as adding a theme)

Obstacles Encountered
No obstacles

Finished Items
Search GUI
Meeting logs are up to date
Low-level design for all pages (GUIs) and database

Unfinished Items
Schedule next shareholder meeting
UML diagrams
Testing pseudocode
Wired GUIs with database
Write test code files using PyTest
Tested all components using PyTest
Research debugging sofware
change order request)
Risk management plan
QA plan
V&V documents

Tasks Until Next Meeting
Research debugging software
UML diagrams
Research PyTest
QA plan
V&V documents
Risk management plan
Wire all GUIs to database
Add theme to GUIs

Notes
No additional notes

Meeting Log#12

Meeting Information
Team #: T2-002
Meeting log #: 12
Current Sprint: SPR3
Date: February 28, 2022
Time: 6:57pm – 8:01pm (MT)
Location: MS Teams (watch video here)
Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Tyler Deschamps, Jordan Van Patten
Next team meeting scheduled for: March 7, 2022, 7:00pm (MT)

Progress From Previous Meeting
Ethan Taylor:
Add Employee GUI (100%)
Edit Employee GUI (100%)
Deactivate Employee GUI (100%)

Jaden Albrecht:
Keep meeting logs up to date (100%)
Build UML Diagrams (40%)
UML research
PyTest research

Cody Strange
SPR3 document (90%)
WBS (100%)
Risk Management Plan (100%)
Tests Document (100%)

Tyler Deschamps:
Employee database pseudocode (100%)
CSV pseudocode (100%)
JSON research

Jordan Van Patten:
Research PyTest
Research debugging software

Topics Discussed
Change board request document
UML diagrams
Test code files using PyTest
Final changes to GUIs (such as adding a theme)

Obstacles Encountered
No obstacles

Finished Items
Add/Edit/Deactivate GUI pages
Meeting logs are up to date

WBS for SPR3
Risk management document
Database pseudocode
CSV pseudocode

Unfinished Items
Schedule next shareholder meeting
UML diagrams
Testing pseudocode
Wired GUIs with database
Write test code files using PyTest
Tested all components using PyTest
Research debugging sofware
change order request
QA plan
V&V documents

Tasks Until Next Meeting
Research debugging software
UML diagrams
Research PyTest
QA plan
V&V documents
Risk management plan
Wire all GUIs to database
Add theme to GUIs

Notes
No additional notes

Meeting Log#13

Meeting Information
Team #: T2-002
Meeting log #: 13
Current Sprint: SPR3
Date: March 7, 2022
Time: 7:00pm – 7:20pm (MT)
Location: Text group chat (no video)
Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Tyler Deschamps, Jordan Van Patten
Next team meeting scheduled for: March 14, 2022, 7:00pm (MT)

Progress From Previous Meeting
Ethan Taylor:
Potential change orders (100%)
Payroll GUI (100%)

Jaden Albrecht:
Keep meeting logs up to date (100%)
Build UML Diagrams (100%)
Change order request form (70%)
UML research

Cody Strange
Field validation code for login page (50%)

Tyler Deschamps:
Database tables to CSV (100%)
Update pert/Gantt/burn-down charts (100%)

Jordan Van Patten:
Research PyTest
Research debugging software
QA plan (10%)

Topics Discussed
Change board request document
UML diagrams
Final changes to GUIs (such as adding a theme)

Obstacles Encountered
No obstacles

Finished Items
Potential change orders
Meeting logs are up to date
UML diagrams
Database tables to CSV
Schedule next shareholder meeting

Unfinished Items

Testing pseudocode
Wired GUIs with database
Write test code files using PyTest
Tested all components using PyTest
change order request
QA plan
V&V documents

Tasks Until Next Meeting
QA plan
V&V documents
Wire all GUIs to database
Add theme to GUIs

Notes
No additional notes