SPRINT SIX

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*
Kole Davis- *QA Manager*
Craig Sharp-*Stakeholder*

Table of contents

Project Introduction

Project Overview

History of Project Approach

Summary: For the project approach we had some major adjustments after sprint three, so we chose to split the document into two sections. The first section describes our approach for the first three sprints and the second sections describes the approach for the last three sprints.

- Sprints 1-3
    - At the beginning of each sprint each member would review the requirements for that sprint
    - Within 25% of the time we would come up with 67% of the tasks required for the sprint and come up with the remaining tasks as we finished previous ones
    - We used a team Trello board to track assigned tasks for each team member
    - Each member determines tasks that fall under their specific role, and they are expected to complete those tasks to a quality standard requested by the shareholder
- Sprints 4-6
    - At the beginning of each sprint each member would review the requirements for that sprint
    - Within 25% of the time we would come up with 75% of the tasks required for the sprint
    - We used meetings more to determine what tasks needed to be done and relied more heavily on our team leader to keep track of task progress by team member
    - Each member determines tasks that fall under their specific role, and they are expected to complete those tasks to a quality standard requested by the shareholder
- Notes
    - When using tasks for measurements tasks vary greatly in size
    - When using time measurements we have to round to the nearest meeting log

Team Members

Summary: This section is a list of each team member and the subset of tasks that are specific to their individual roles (not every miscellaneous task that they completed).

Jaden Albrecht - Team Manager

- Keeping track of project progress
- Schedule team meetings
- Schedule Shareholder meetings
- Meeting Logs
- Team Performance

Cody Strange - Scribe and Information Manager

- Sprint Document
- Deployment Plan
- Metric Reports

Ethan Taylor - GUI Developer

- GUI Development
- High Level Design
- Low Level Design
- Database Integration

Tyler Deschamps - Chart and Milestone document builder

- Chart development
- Low Level Design
- Back-end development
- Backlog

Jordan Van Patten - V&V and Tester

- V&V
- Testing

Kole Davis - QA Manager

- Misc.

V&V Documents

Summary: This is the sprint six V&V documents

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-6
Application Name		**Application #**	
Project Manager		**Delivery Manager**	
Date Completed	4/15/2022		

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?	☒	☐	☐	

1. 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?	☒	☐	☐	

1. 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?	☒	☐	☐	
1. Has the presentation layer of the application been developed in accordance with prevailing industry standards?	☒	☐	☐	

1. 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?	☒	☐	☐	

1. 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
1. 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?	☐	☒	☐	
1. Does the system meet the standard password requirements?	☐	☒	☐	Password requirements will be added in at a later execution of our program

1. 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. 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
1. Are all vendor supplied default passwords revised prior to placing the application in a production environment?	☐	☒	☐	
1. 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

1. 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?	☐	☐	☒	
1. Based on the application's Information Security Classification, does the application meet the auditing functional control requirements?	☐	☐	☒	

1. 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?	☒	☐	☐	
1. Has code duplication been avoided?	☒	☐	☐	

1. 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?	☒	☐	☐	
1. Have all developers followed the same coding style and naming conventions?	☒	☐	☐	
1. Have all developers followed the coding best practices as set out by the organization which owns the technology?	☒	☐	☐	

1. 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?	☒	☐	☐	
1. Does each code unit have its own brief and accurate description?	☒	☐	☐	

1. 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?	☒	☐	☐	

1. Do the error messages contain minimal but meaningful information?	☒	☐	☐	
1. Does the application avoid displaying system information in error messages?	☒	☐	☐	
1. Are the error messages kept in a single location?	☐	☒	☐	Error messages are kept in separate locations in the code, and in the application it is shown below the effected area, I.E. if date is incorrect, it will display the error message below the date entry box

1. 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
1. Is logging being done on each server tier?	☐	☒	☐	
1. Are the logs kept in a single location / directory / database?	☒	☐	☐	We use an internet application where we report our bugs/errors
1. Are the logged errors specific enough to assist support personnel in troubleshooting production problems?	☒	☐	☐	
1. Is the code that logs the error messages written in a modular way?	☒	☐	☐	
1. Are the log files free of personally sensitive or identifiable information?	☒	☐	☐	

1. 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)?	☒	☐	☐	
1. Are field values being validated with regular expressions where possible (e.g. validating email addresses and dates for valid formats)?	☒	☐	☐	Date of birth is not currently being validated
1. Do the validations resulting in error messages prevent data from being written to persistent storage (databases, files, etc.)?	☒	☐	☐	
1. Are the validations being performed within the business logic, as well as on the presentation layer?	☒	☐	☐	
1. Have the validations been written so that users cannot bypass them?	☒	☐	☐	
1. Are all of the field lengths and types within the application consistent with the column lengths and types declared within the underlying database tables?	☒	☐	☐	
1. Are user inputs being sanitized (without exceptions) according to OWASP recommendations?	☒	☐	☐	

1. 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?	☐	☒	☐	Dates are automatically inputted by the code, not by the user, so there is no date validation. DOB is not yet validated and mandated to be in the month/day/year format but will be added in a future version
1. Do all relevant dates include a timestamp?	☐	☒	☐	

1. 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?	☒	☐	☐	
1. Do all hard-coded values reside exclusively within configuration and constant, centralized locations? (Central Locations that enable changes without recompiling source code)	☒	☐	☐	

1. 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
1. Did the application pass all negative tests?	☒	☐	
1. Have client testers completed the formal test plan in its entirety?	☐	☒	We have not made a formal test plan yet
1. Did the application pass all tests included in the formal test plan?	☐	☒	
1. Have all positive / negative test cases and test case results been documented?	☒	☐	

1. 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?	☒	☐	☐	
1. Have all previous tests been reconducted with the results compared against expected results?	☒	☐	☐	
1. 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?	☒	☐	☐	
1. 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?	☒	☐	☐	

1. 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)?	☐	☒	☐	
1. Has the application been tested with large numbers of concurrent transactions (i.e. a number of transactions that is representative of peak system usage)?	☐	☒	☐	
1. Did the system perform well with a large number of concurrent users?	☐	☐	☒	
1. Did the system perform well with a large number of concurrent transactions?	☐	☐	☒	
1. Are end-users satisfied with the application's performance and responsiveness during everyday use?	☐	☐	☒	

1. 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?	☐	☒	☐	
1. Have special software/certificate requirements been documented?	☐	☒	☐	
1. Does the documentation provide expiration dates and instructions for renewal?	☐	☒	☐	
1. Is the system / application free from trial versions of software?	☐	☐	☒	

1. 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
1. 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

1. 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
1. Has a test build been completed by Application Services using the code that has been handed over?	☐	☐	☒	
1. 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)	☒	☐	☐	

1. Source Code

Validation Questions	Yes	No	NA	Comments
1. 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)	☐	☐	☒	

1. 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?	☒	☐	☐	
1. Keys based on sequence numbers have unique sequences.	☒	☐	☐	
1. Are all keys and required fields set to 'not null' in all tables of the database?	☒	☐	☐	
1. Have triggers, stored procedures, sequences, and constraints been properly utilized?	☐	☐	☒	

1. 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?	☐	☒	☐	
1. Have the necessary firewall rules been added to allow Application Services support personnel to access the relevant servers (i.e. via the Jump Box)?	☐	☒	☐	
1. Have all server environments (development, test/staging, and production) been fully created?	☐	☒	☐	

1. 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. Are all of the server environments entirely consistent with each other?	☐	☒	☐	

Sprint 6 – Project Completion

Final Requirements Specifications

Final Requirement Specs

Summary: These are the final requirement specifications. Any requirements that have changed are colored blue, and any new requirements are colored in yellow.

- **Requirement Specifications**:
  - **Database Merging**: recognize when there are incomplete fields and making a flag pop up that says "this is an incomplete employee"
  - **Add employee**: Page(B) of the GUI, Admin access, button on page(A), input all required fields and add employee to database
  - **Edit employee**: Page(C) of the GUI, General access, button on page(A), list of employee information to edit(First name, Last name, Address, Office phone, Personal phone, Bank info, Office email, Personal email)
  - **Edit employee**: Page(C) of the GUI, Admin access, button on page(A), list of employee information to edit(SS#, D.O.B, Pay type, Title Dept., Permission level)
  - **Search employee by last name or ID**: Page(A) of the GUI, General access, button on page(A), if user inputs numbers will search based on ID, if user inputs letters will search based off of last name, pulls up list of names and numbers of every that matches the information input
  - **View employee**: Page(A) of GUI, Admin access, list of employees names and IDs, when user clicks on name/ID pulls up all information related to that employee, option to view/hide deactivated employees
  - **Deactivate employee**: Page(D) of GUI, Admin access, button on page(A), input ID of employee that the user wants to deactivate, confirmation message pops up along with employee information to make sure the user wants to deactivate this specific employee
  - **Win 10**: The program will be able to run on Windows 10
  - **Reports**: Page(E) of GUI, Admin access, button on page(A), options to produce various reports(pay, reimbursables, employees)
  - **Export Reports**: Page(E) of GUI, Admin access, button on page(A), option to export any report as a csv
  - **Secure records to only admin permissions**: Certain information/records will require the user to be flagged as an Admin to see.
  - **Intuitive GUI**: Mouse over input boxes to see what information is required, help buttons on each page
  - **User Manual**: Page(F) of GUI, General access, help button on page(A), information on what each page can do
  - **Readme.txt**: A short description of what data the code contains
  - **Simple just download installation that runs**: Download software and then the user is good to go

    - **Garbage proof entries**: Checks each field of data to make sure that all information coming in isn't junk
    - **Warn user of empty data fields**: When user adds/edits an employee, issue a warning if any data fields are left empty or incomplete
    - **Employee can view all personal fields**: Page(A) of GUI, when employee logs in with his/her ID that ID's corresponding information is pulled up
    - **Bug free**: Software will go through extensive testing to minimize/eliminate as many bugs as possible
    - **Requirements for employee information**: First name, Last name, Address(use separated fields), Office phone, Personal phone, Emp ID(Specific length, only numbers), Pay type(commission, hourly, salary), D.O.B, SS#(Specific length, only numbers), Start date, End date, Bank Info(if Direct Deposit), Permission level, Title Dept., Office email, Personal email
    - **Employee login**: Page(G) of GUI, requires employee ID and password in order to access the program.
- **Functional**
    - Add employee
    - Edit employee
    - Search employee
    - View employee
    - Deactivate employee
    - Reports
    - Export Reports
    - Secure records to only admin permissions
    - Warn user of empty data fields
    - Employee can view all personal fields
    - Requirements for employee information
    - Employee login
- **Non-functional**
    - Win 10
    - Bug free
    - Garbage proof entries
    - Simple just download installation that runs
    - Readme.txt
    - User Manual
    - Intuitive GUI
    - Database merging
- **Change orders**
    - Search employee using any attribute
    - Different visual themes based on user's permission level

Requirement Specification Evaluation

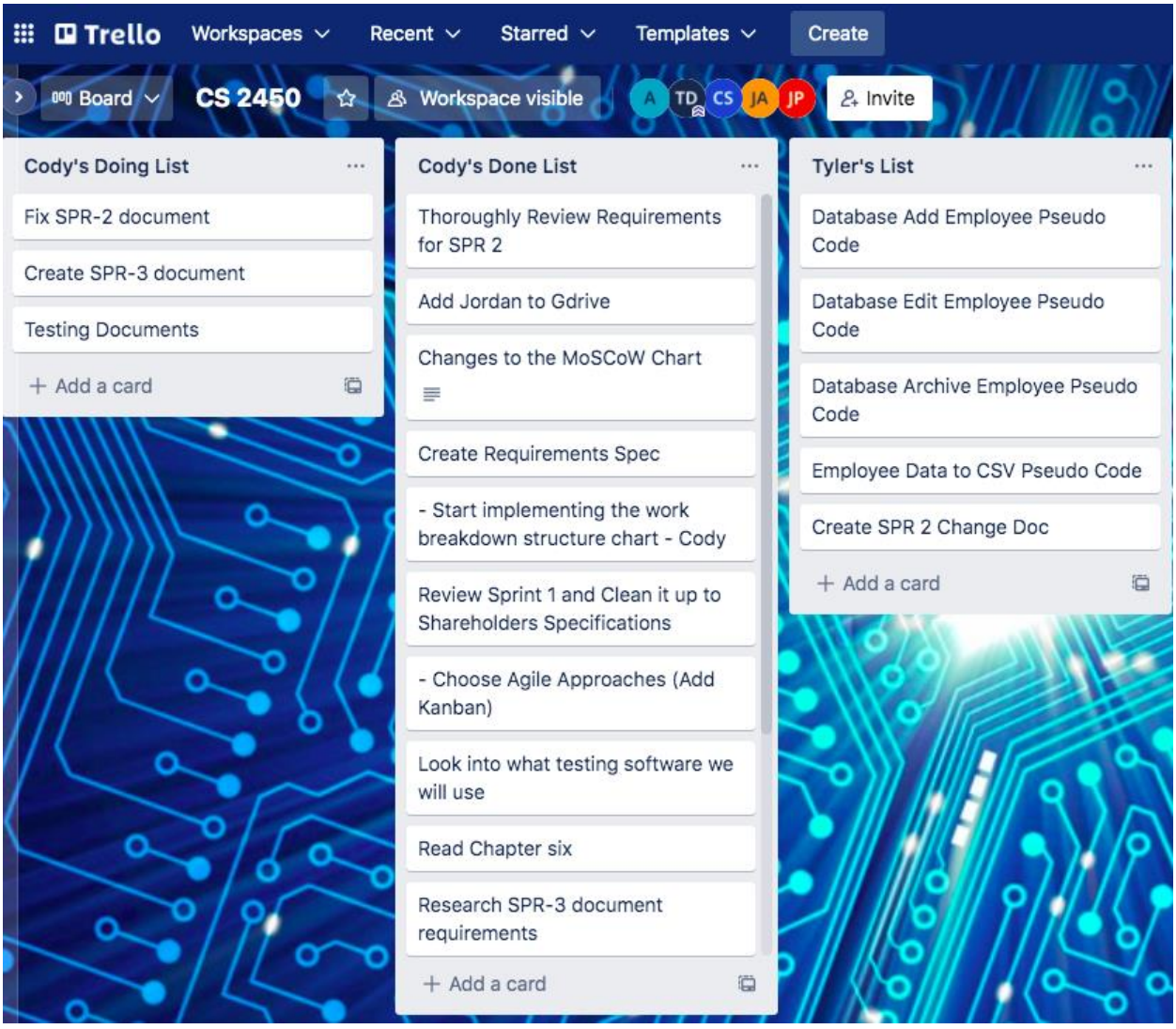
Evaluation:

- What we did well
    - We came up with the requirement specifications on time
    - We got the shareholder to sign off on our requirement specifications
    - We completed all of the requirements for the project
- What we will do better
    - Update the requirements more frequently as they change
    - Inform the shareholder whenever the requirement specifications need to be changed

Development Approach and Project Design

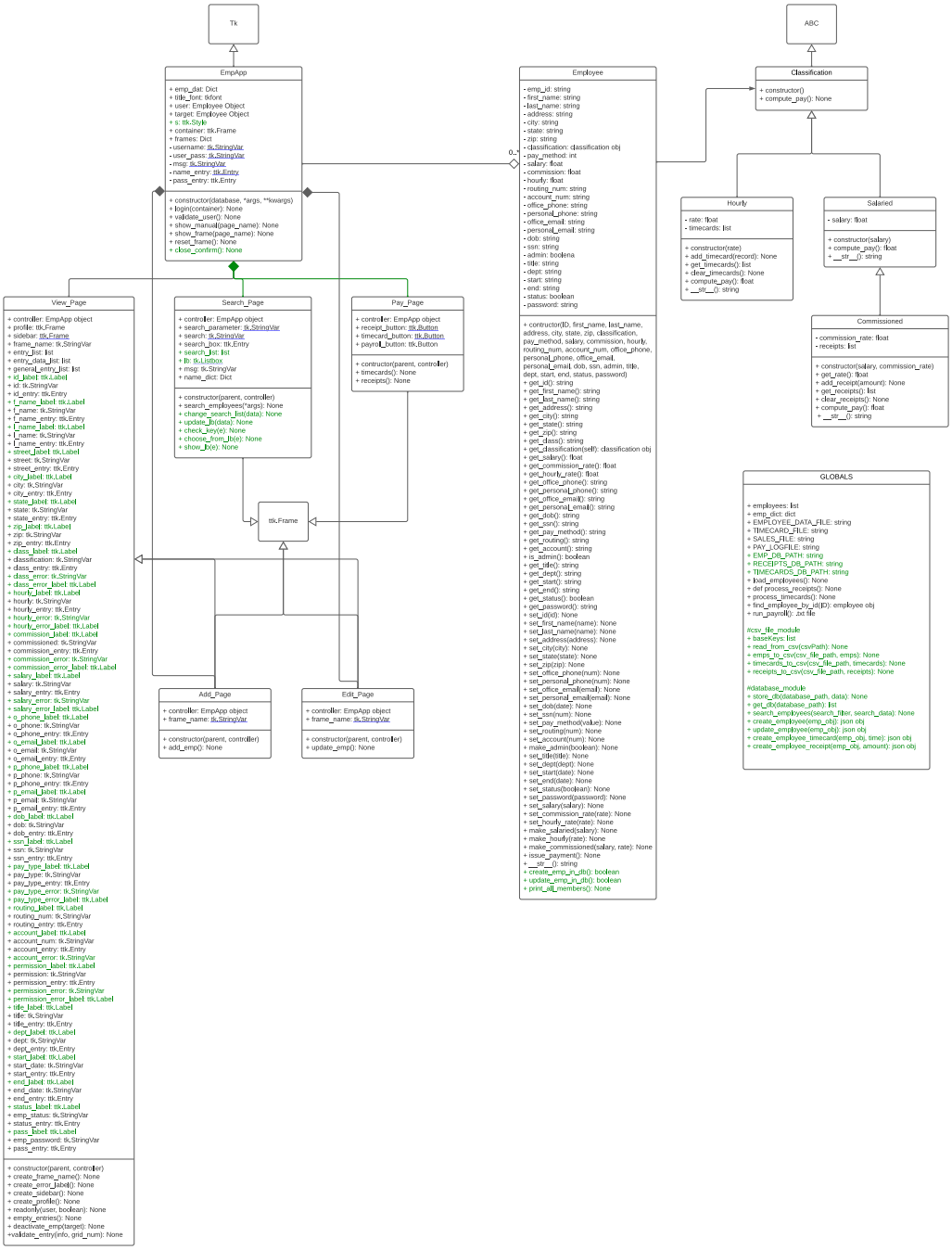
Project Development Approach

Summary: We tackled this project with an incremental agile developmental approach, focused on providing a usable product that could then be further refined with each sprint.
For example, the very first version only featured a working login screen. Subsequent versions then added a view page, add and edit pages, a search page, and lastly the payroll page. To aid us in organizing the tasks required for each sprint, we used Trello to implement a Kanban chart.



UML Diagrams

Summary: In our design phases, the team created a few different UML diagrams. The largest and most important of these was our class diagram, which was used to help plan and show the various parts of the program. As the program developed further, any changes we found necessary to add to the class diagram were added in green. The UML class diagram for the current version of the project can be seen below.


Tk
EmpApp
+ emp_dat: Dict
+ title_font: tkfont
+ user: Employee Object
+ target: Employee Object
+ s: ttk.Style
+ container: ttk.Frame
+ frames: Dict
- username: tk.StringVar
- user_pass: tk.StringVar
- msg: tk.StringVar
- name_entry: ttk.Entry
- pass_entry: ttk.Entry
+ constructor(database, *args, **kwargs)
+ login(container): None
+ validate_user(): None
+ show_manual(page_name): None
+ show_frame(page_name): None
+ reset_frame(): None
+ close_confirm(): None
0..*
Employee
- emp_id: string
- first_name: string
- last_name: string
- address: string
- city: string
- state: string
- zip: string
- classification: classification obj
- pay_method: int
- salary: float
- commission: float
- hourly: float
- routing_num: string
- account_num: string
- office_phone: string
- personal_phone: string
- office_email: string
- personal_email: string
- dob: string
- ssn: string
- admin: boolena
- title: string
- dept: string
- start: string
- end: string
- status: boolean
- password: string
+ contructor(ID, first_name, last_name, address, city, state, zip, classification, pay_method, salary, commission, hourly, routing_num, account_num, office_phone, personal_phone, office_email, personal_email, dob, ssn, admin, title, dept, start, end, status, password)
+ get_id(): string
+ get_first_name(): string
+ get_last_name(): string
+ get_address(): string
+ get_city(): string
+ get_state(): string
+ get_zip(): string
+ get_class(): string
+ get_classification(self): classification obj
+ get_salary(): float
+ get_commission_rate(): float
+ get_hourly_rate(): float
+ get_office_phone(): string
+ get_personal_phone(): string
+ get_office_email(): string
+ get_personal_email(): string
+ get_dob(): string
+ get_ssn(): string
+ get_pay_method(): string
+ get_routing(): string
+ get_account(): string
+ is_admin(): boolean
+ get_title(): string
+ get_dept(): string
+ get_start(): string
+ get_end(): string
+ get_status(): boolean
+ get_password(): string
+ set_id(id): None
+ set_first_name(name): None
+ set_last_name(name): None
+ set_address(address): None
+ set_city(city): None
+ set_state(state): None
+ set_zip(zip): None
+ set_office_phone(num): None
+ set_personal_phone(num): None
+ set_office_email(email): None
+ set_personal_email(email): None
+ set_dob(date): None
+ set_ssn(num): None
+ set_pay_method(value): None
+ set_routing(num): None
+ set_account(num): None
+ make_admin(boolean): None
+ set_title(title): None
+ set_dept(dept): None
+ set_start(date): None
+ set_end(date): None
+ set_status(boolean): None
+ set_password(password): None
+ set_salary(salary): None
+ set_commission_rate(rate): None
+ set_hourly_rate(rate): None
+ make_salaried(salary): None
+ make_hourly(rate): None
+ make_commissioned(salary, rate): None
+ issue_payment(): None
+ __str__(): string
+ create_emp_in_db(): boolean
+ update_emp_in_db(): boolean
+ print_all_members(): None
ABC
Classification
+ constructor()
+ compute_pay(): None
Hourly
- rate: float
- timecards: list
+ constructor(rate)
+ add_timecard(record): None
+ get_timecards(): list
+ clear_timecards(): None
+ compute_pay(): float
+ __str__(): string
Salaried
- salary: float
+ constructor(salary)
+ compute_pay(): float
+ __str__(): string
Commissioned
- commission_rate: float
- receipts: list
+ constructor(salary, commission_rate)
+ get_rate(): float
+ add_receipt(amount): None
+ get_receipts(): list
+ clear_receipts(): None
+ compute_pay(): float
+ __str__(): string
View_Page
+ controller: EmpApp object
+ profile: tk.Frame
+ sidebar: tk.Frame
+ frame_name: tk.StringVar
+ entry_list: list
+ entry_data_list: list
+ general_entry_list: list
+ id_label: ttk.Label
+ id: tk.StringVar
+ id_entry: ttk.Entry
+ f_name_label: ttk.Label
+ f_name: tk.StringVar
+ f_name_entry: ttk.Entry
+ l_name_label: ttk.Label
+ l_name: tk.StringVar
+ l_name_entry: ttk.Entry
+ street_label: ttk.Label
+ street: tk.StringVar
+ street_entry: ttk.Entry
+ city_label: ttk.Label
+ city: tk.StringVar
+ city_entry: ttk.Entry
+ state_label: ttk.Label
+ state: tk.StringVar
+ state_entry: ttk.Entry
+ zip_label: ttk.Label
+ zip: tk.StringVar
+ zip_entry: ttk.Entry
+ class_label: ttk.Label
+ classification: tk.StringVar
+ class_entry: ttk.Entry
+ class_error: tk.StringVar
+ class_error_label: ttk.Label
+ hourly_label: ttk.Label
+ hourly: tk.StringVar
+ hourly_entry: ttk.Entry
+ hourly_error: tk.StringVar
+ hourly_error_label: ttk.Label
+ commission_label: ttk.Label
+ commissioned: tk.StringVar
+ commission_entry: ttk.Entry
+ commission_error: tk.StringVar
+ commission_error_label: ttk.Label
+ salary_label: ttk.Label
+ salary: tk.StringVar
+ salary_entry: ttk.Entry
+ salary_error: tk.StringVar
+ salary_error_label: ttk.Label
+ o_phone_label: ttk.Label
+ o_phone: tk.StringVar
+ o_phone_entry: ttk.Entry
+ o_email_label: ttk.Label
+ o_email: tk.StringVar
+ o_email_entry: ttk.Entry
+ p_phone_label: ttk.Label
+ p_phone: tk.StringVar
+ p_phone_entry: ttk.Entry
+ p_email_label: ttk.Label
+ p_email: tk.StringVar
+ p_email_entry: ttk.Entry
+ dob_label: ttk.Label
+ dob: tk.StringVar
+ dob_entry: ttk.Entry
+ ssn_label: ttk.Label
+ ssn: tk.StringVar
+ ssn_entry: ttk.Entry
+ pay_type_label: ttk.Label
+ pay_type: tk.StringVar
+ pay_type_entry: ttk.Entry
+ pay_type_error: tk.StringVar
+ pay_type_error_label: ttk.Label
+ routing_label: ttk.Label
+ routing_num: tk.StringVar
+ routing_entry: ttk.Entry
+ account_label: ttk.Label
+ account_num: tk.StringVar
+ account_entry: ttk.Entry
+ account_error: tk.StringVar
+ permission_label: ttk.Label
+ permission: tk.StringVar
+ permission_entry: ttk.Entry
+ permission_error: tk.StringVar
+ permission_error_label: ttk.Label
+ title_label: ttk.Label
+ title: tk.StringVar
+ title_entry: ttk.Entry
+ dept_label: ttk.Label
+ dept: tk.StringVar
+ dept_entry: ttk.Entry
+ start_label: ttk.Label
+ start_date: tk.StringVar
+ start_entry: ttk.Entry
+ end_label: ttk.Label
+ end_date: tk.StringVar
+ end_entry: ttk.Entry
+ status_label: ttk.Label
+ emp_status: tk.StringVar
+ status_entry: ttk.Entry
+ pass_label: ttk.Label
+ emp_password: tk.StringVar
+ pass_entry: ttk.Entry
+ constructor(parent, controller)
+ create_frame_name(): None
+ create_error_label(): None
+ create_sidebar(): None
+ create_profile(): None
+ readonly(user, boolean): None
+ empty_entries(): None
+ deactivate_emp(target): None
+validate_entry(info, grid_num): None
Search_Page
+ controller: EmpApp object
+ search_parameter: tk.StringVar
+ search: tk.StringVar
+ search_box: ttk.Entry
+ search_list: list
+ lb: tk.Listbox
+ msg: tk.StringVar
+ name_dict: Dict
+ constructor(parent, controller)
+ search_employees(*args): None
+ change_search_list(data): None
+ update_lb(data): None
+ check_key(e): None
+ choose_from_lb(e): None
+ show_lb(e): None
Pay_Page
+ controller: EmpApp object
+ receipt_button: ttk.Button
+ timecard_button: ttk.Button
+ payroll_button: ttk.Button
+ contructor(parent, controller)
+ timecards(): None
+ receipts(): None
ttk.Frame
Add_Page
+ controller: EmpApp object
+ frame_name: tk.StringVar
+ constructor(parent, controller)
+ add_emp(): None
Edit_Page
+ controller: EmpApp object
+ frame_name: tk.StringVar
+ constructor(parent, controller)
+ update_emp(): None
GLOBALS
+ employees: list
+ emp_dict: dict
+ EMPLOYEE_DATA_FILE: string
+ TIMECARD_FILE: string
+ SALES_FILE: string
+ PAY_LOGFILE: string
+ EMP_DB_PATH: string
+ RECEIPTS_DB_PATH: string
+ TIMECARDS_DB_PATH: string
+ load_employees(): None
+ def process_receipts(): None
+ process_timecards(): None
+ find_employee_by_id(ID): employee obj
+ run_payroll(): .txt file
#csv_file_module
+ baseKeys: list
+ read_from_csv(csvPath): None
+ emps_to_csv(csv_file_path, emps): None
+ timecards_to_csv(csv_file_path, timecards): None
+ receipts_to_csv(csv_file_path, receipts): None
#database_module
+ store_db(database_path, data): None
+ get_db(database_path): list
+ search_employees(search_filter, search_data): None
+ create_employee(emp_obj): json obj
+ update_employee(emp_obj): json obj
+ create_employee_timecard(emp_obj, time): json obj
+ create_employee_receipt(emp_obj, amount): json obj

Requirements Approach

Summary: The very first step in our design process was requirements gathering. This was necessary in order for our team and the shareholder to have similar expectations regarding the functionality of the project. To accomplish this, in addition to using the materials provided by the shareholder, our team got together to brainstorm other possible functions/requirements that would be useful for the project, categorizing them under the headings “Must Have”, “Should Have”, “Could Have” and “Won’t Have”. The results of these brainstorming sessions were then filtered through a Delphi method down into a single chart which was shown to the shareholder and used in creating the final requirement specifications.

High Level Design Approach

Summary: The next step in our design process was creating a high-level design. Using our requirement specifications document as a guide, we were able to get a rough idea of the parts the program would have, and how they would interact with each other. Those parts would include things like what pages the GUI would need to show, what classes would needed to be coded, and how the database would be integrated with those pages and classes, as well as how admin and general privileges would differ. To help with high-level design, the team created several UML diagrams to show potential use-cases and classes as well as a rough template for what we wanted the GUI pages to look like. Using these diagrams and templates, we were able to create a plain English outline of what our code would look like, an example of which can be seen below.

Login Screen:

1. Import tkinter
2. Set up main window (can be inherited by other subsequent pages/windows)
3. Set title and labels for user inputs (username and password)
4. Get username and password from user inputs
5. Check for provided username in the database
6. If username not in database, report username is not valid employee
7. If username or password incorrect, display warning
8. Validate username and password, get user permission level, and display proper window
9. Load employees database into list of employee objects

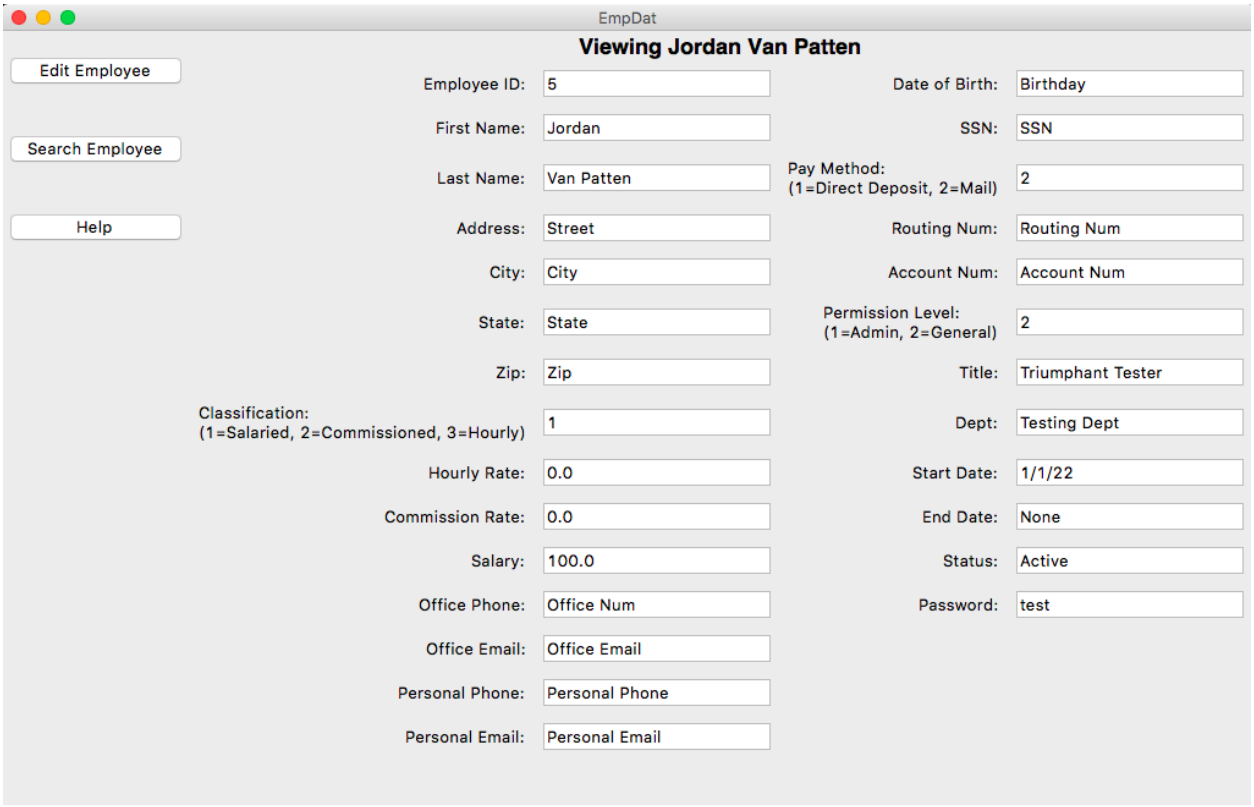
Low Level Design Approach

Summary: After getting a rough idea of what the project would entail in the high-level design phase, our team shifted toward low-level design. In the low-level design phase, we focused much more heavily on how the actual code for the project would function and interact. As none of our team was very familiar with Tkinter prior to starting this project, this phase of the design also required a lot of research into Tkinter. During this phase was also when our team decided to implement JSON as an intermediate step between the GUI and csv database, in order to smooth the transition between them. Using our Tkinter research as well as the plain English outline developed in the high-level design phase, we were able to create a more solid UML class diagram and pseudocode. Efforts were also made by the team to plan for tests the code would need once it was actually written. An example of our pseudocode for the Application class can be seen below.

```
# Application Class - Main Parent Class
  # Inherits from Tk
  # Constructor:
    # Uses parent constructor
    # Set title, geometry, and resizable values
    # Dictionary container for holding all of the different frame classes (Ex: self.frames = {})
      # Keys are names for the different pages (Ex: 'login_page')
      # Values are call to constructor for that page (Ex: self.frames['login_page'] =
Login_Page(--login constructor args--))
      # Included Frames:
        # Login_Page, View_Employee_Page, Edit_Employee_Page, Search_Employee_Page,
Add_Employee_Page
    # employees dictionary - holds data for all employees in database
      # Keys are employee ids
      # Values are employee objects with those ids
      # Initialize as empty list
    # timecards list - list of lists containing an employees ID and then all their
currently saved timecards
      # Initialize as empty list
    # receipts list - list of lists containing an employees ID and then all their currently saved
receipts
      # Initialize as empty list
  # change_frames function
    # Takes a frame name as an argument (Ex: 'login_page')
    # Uses this name to get the value from the self.frames dictionary, which is the
constructor
    # Calls reset() and tkraise() functions on that frame
```

Development Approach

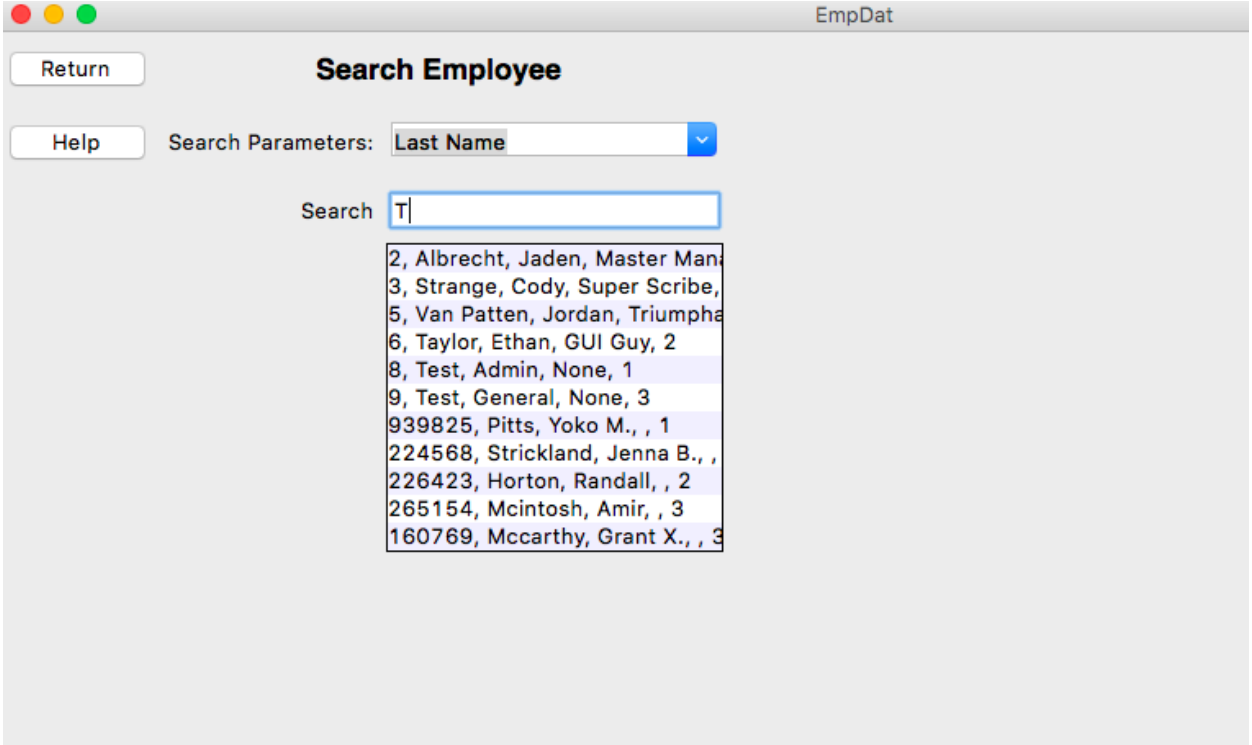
Summary: With our designs complete, we were ready to start coding. By this point, our pseudocode had developed enough that only a few changes were necessary to create usable code. Several changes to the earlier designs were required as the GUI started to take shape and small flaws in the initial plans were discovered. The GUI style we developed can be described as an "always moving forward" design. After logging in, instead of a home page or directory, the user is brought to the view page, viewing their own information. An example of the view page and the code that produces it is shown below.



main.py

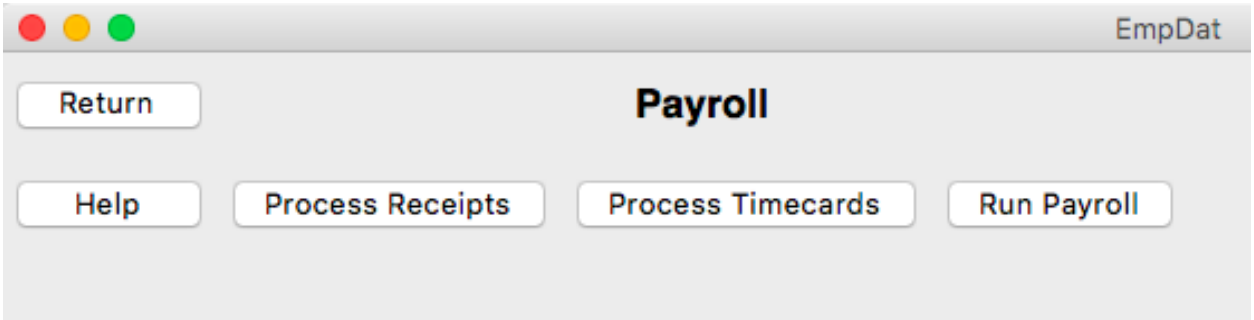
```
class View_Page(ttk.Frame):

    def __init__(self, parent, controller):
        ttk.Frame.__init__(self, parent)
        self.controller = controller

        #Set up frame name
        self.create_frame_name()

        #Set up page error message
        self.create_error_label()

        #Set up sidebar buttons
        self.create_sidebar(self.controller.user, self.controller.target)

        #Set up labels/entries based on target
        self.create_profile(self.controller.user, self.controller.target)

    def create_frame_name(self):
        '''Creates a frame for the name of the viewed employee'''
        self.frame_name = StringVar()
        emp_name = self.controller.target.get_first_name() + " " + self.controller.target.get_last_name()
        self.frame_name.set("Viewing " + emp_name)
        title = ttk.Label(self,
                          textvariable=self.frame_name,
                          font=self.controller.title_font)
        title.grid(column=1, row=0, sticky=(N,S))

    def create_sidebar(self, user, target):
        '''Creates the buttons in the sidemenu based on user permission'''
        self.sidebar = ttk.Frame(self)
        self.sidebar.grid(column=0, row=0, sticky=(N,W,E,S), rowspan=6)
        if user.is_admin() == True: #admin view
            edit = ttk.Button(self.sidebar, text="Edit Employee",
                              command=lambda: self.controller.show_frame("Edit_Page"))
            edit.grid(column=0, row=0, sticky=(N,W,E,S))
            add = ttk.Button(self.sidebar, text="Add Employee",
                             command=lambda: self.controller.show_frame("Add_Page"))
            add.grid(column=0, row=1, sticky=(N,W,E,S))
            fire = ttk.Button(self.sidebar, text="Deactivate Employee",
                              command=lambda: self.deactivate_emp(target))
            fire.grid(column=0, row=2, sticky=(N,W,E,S))
            if target.get_status() == False:
                fire.configure(state=DISABLED)
```

From the view page, the user can move to the search page to find another employee and be taken to their view page. From there, the user repeats the process, moving forward between the search page and the various view pages of the employees they have searched. There is no "Back" button or functionality, hence the "always moving forward" design description. To return to a previously visited employee, simply search for them on the search page.



Depending on the user's permission level, other options apart from going to the search page will be available from the view page. If the user is an admin, or is a general permission user viewing their own information, they will have the option to edit the information shown on the view page. Admin level users are also able to deactivate employees, add new employees, and access the payroll page.

On the payroll page, users are able to import any receipt or timecards, as well as issue payments for the current pay period. The pay report after issuing payments is output to a folder in the program directory called "output".



Quality Control Approach

Summary: To help maintain an acceptable program quality throughout development, we relied not only on numerous testing methods but also on reliable bug tracking and reporting. Any bugs found could then be fixed before the next iteration of the application. To help maintain a good quality of user-friendliness, members of the team who were not involved in the coding would review the GUI between iterations and give feedback. This was especially important since they were able to test on Windows systems, which is the operating system requested by the shareholder.

Testing Approach

Summary: A significant portion of our testing was actually performed by the coders themselves while writing the program. Because of this, many potential bugs were discovered and fixed before iterations of the application were even released. To catch bugs the coders may have missed, we relied on various other testing methods such as assert testing and user acceptance testing. Many tests focused on the login page in particular, since that page controls important aspects of application like establishing user privileges. For the other pages many of the tests focused on verifying inputs and outputs. An example of the testing code for the login page can be seen below.

```
76  #CODE FOR TESTING LOGIN (Validate user class only)
77  def test_correct_1():
78      #code that should pass username and password
79      x.username.set("8")
80      x.user_pass.set("test")
81      x.validate_user()
82      assert x.msg.get() != ("Invalid ID!") and ("Incorrect password or ID!") != x.msg.get()
83
84  def test_correct_2():
85      x.username.set("2")
86      x.user_pass.set("test")
87      x.validate_user()
88      assert x.msg.get() != ("Invalid ID!") and ("Incorrect password or ID!") != x.msg.get()
89
90  def test_correct_3():
91      x.username.set("3")
92      x.user_pass.set("test")
93      x.validate_user()
94      assert ("Invalid ID!") != x.msg.get() and ("Incorrect password or ID!") != x.msg.get()
95
```

Project Management plans *and* reports

Quality Control Plan and Results

Quality Control Plan

Summary: This is the quality control plan that was originally created in sprint three. The blue text shows the adjustments and changes that have been made throughout the previous sprints that have yet to be documented.

- **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**
  - Three meetings a week through MS Teams
  - Daily updates and through messaging app
- **Efficient code**
  - Code 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

Evaluation:

- What we did well
  - We followed the definition of done, everyone understood what it meant for something to be considered 'done' and we all agreed upon the final sprint documents before we submitted them.
  - We followed the consistency in standards, we would make sure that anything submitted in the final sprint document followed the same formatting standards as the rest of the document. We would also make sure that our code followed PEP-8 standards.

    - We followed the constant communication, most meetings we had at least 5/6 members in attendance, and we kept a consistent three meetings a week schedule. Along with updating our task progress through a messaging app.
- What we would do better
    - While our filenames followed the format of having underscores and versioning tags anything beyond that would be left to the member to decide what to name. While it is not a big deal with the size of project we were working on it could be inconvenient if we worked on a larger project without more detailed naming standards.
    - Our code could have gone through more thoroughly testing with more boundary tests.
    - Bugs could be documented as they were found for an easier and more accurate way to come up with metric reports.

Testing Plans, Metrics, and Evaluations

Assert/Unit Tests

Summary: This is the assert/unit testing from sprint five, the details of the test descriptions have been simplified for the shareholders convivence. Any additional tests added this sprint will be colored blue.

Activity	Tests	Test Inputs	Source	State
Employee has correct fields.	3	Integer	Jordan	
Searches for user in the database.	5	Int/Str	Jordan	
Fields are valid when valid results are input	1	Emp object	Jordan	
Fields are invalid when invalid results are input	1	Emp object	Jordan	
Info on general view page matches info in database	4	Integer	Jordan	
Info on admin view page matches info in database	2	Integer	Jordan	
Correct username/ID not trigger error message	7	Integer	Jordan	
Incorrect username does trigger an error message	8	Integer	Jordan	
Incorrect password does trigger an error message	11	Int/Str	Jordan	
Fields are valid when valid inputs are given	6	Integer	Jordan	
Test edit invalid variable	1	Emp object	Jordan	
Pay page does not trigger error msg with valid info	2	Emp object	Jordan	

Evaluation:

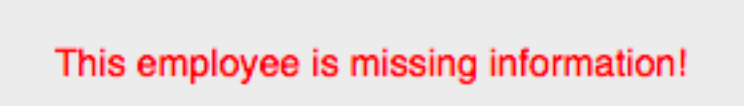
- What we did well
    - We tested every page runs properly under normal circumstances
    - We have heavily tested the login page
    - The program passes all of the test
- What we would do better
    - Add boundary testing
    - Increase the test count for the non-login pages
    - Separate the unit and assertion testing

User Acceptance

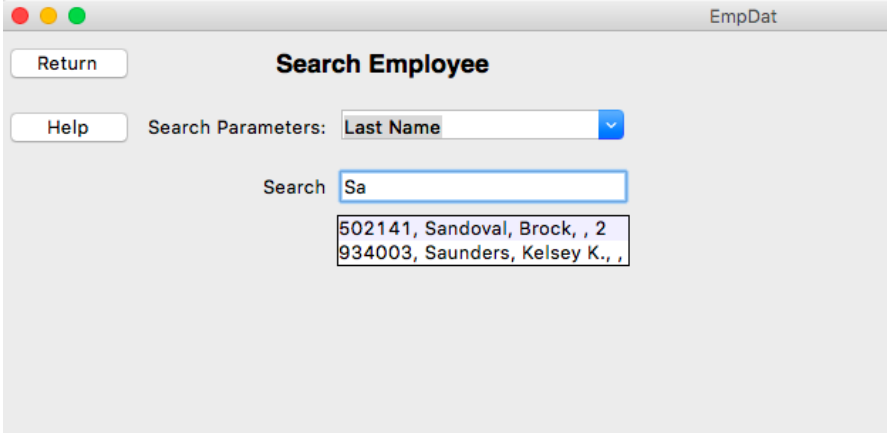
Summary: User acceptance testing was a critical part of improving the quality of the application's GUI. To get a reasonably unbiased opinion and perspective on the GUI, the members of the team that were not at all involved with the coding performed much of the user acceptance testing. An example of part of the results of the testing can be seen below.

- The employee classification field label would look a lot cleaner without the numerical classification options in the parentheses.
- Sometimes it can be difficult knowing if we have typed in all the correct employee information into each data field before attempting to change the employee's information and getting a message saying the information is incorrect.
- The program needs a theme.
- Certain fields like pay method, classification, and permission level would be more user friendly and easily validated with dropdowns.
- The alerts under each field that is required or needs attention after validation should be styled red.

When attempting to implement the suggested changes, we gave priority to those recommendations that were most common among all the different testers. Other results of the user acceptance testing that the team liked but did not deem critical we kept either as change orders or potential changes in future versions of the application. Some of the changes made to the application as a result of user acceptance testing include changing error messages to red for visibility and implementing a drop down search filter on the search page for ease of use. Examples of these changes can be seen below.



Colored Error Message



Search Page Filter

Functional/Non-functional testing

Summary: This is a table version of the Functional/Non-functional testing from sprint five.

- This is the most recent version of the Functional/Non-functional testing
- This table shows which requirement tests have passed or failed.
- This table was created for the convenience of the shareholder so that he could easily see the state of the Functional/Non-functional testing.
- A more detailed description of the Functional/Non-functional requirements and testing can be found in sprint five
- All new requirements to this sprint will be colored blue

Requirement	Functional/Non-Functional	Source	Passed/Failed
Database Merging	Functional	Jaden	
Add employee	Functional	Jaden	
Edit employee	Functional	Jaden	
Search employee	Functional	Jaden	
View Employee (General access)	Functional	Jaden	
View employee (Admin access)	Functional	Jaden	
Deactivate employee	Functional	Jaden	
Pay report	Functional	Jaden	
Garbage proof entries	Functional	Jaden	
Warn user of empty data fields	Functional	Jaden	
The program runs on Windows 10	Non-Functional	Jaden	
Intuitive GUI	Non-Functional	Jaden	
User Manual	Non-Functional	Jaden	
Readme.txt	Non-Functional	Jaden	
Simple installation that runs	Non-Functional	Jaden	
Employee can view all personal fields	Non-Functional	Jaden	
Bug free	Non-Functional	Jaden	
Requirements for employee information	Non-Functional	Jaden	
Employee login	Functional	Jaden	
Search employee using any attribute	Change order	Jaden	
Visual themes based on user's permissions	Change order	Jaden	

Evaluation:

- What we did well
  - We had one of our members go through the requirement specifications and test that the program followed everything in the specifications.
  - We tested the new requirement and our change orders as well to make sure our program had everything in it.
- What we would do better
  - We ran the test in sprint five when it was supposed to be done in sprint four. So making sure that we completed the test on time in the correct sprint.
  - The original functional/non-functional tests were NOT done in a table like the one above, the table above is in response to the shareholder's comments. So creating a more readable table the first time around.

Usability Testing

Summary: This is a table of the usability tests from sprint five, nothing was added or changed from the original document beyond the format.

Test	Result
#1	I was unable to get the program running, I downloaded the zip file and tried to run every file that was there, at best they did nothing I could see. At worst they through errors. I was unable to get it running and will need to discuss this with the team to find out why it is not working.
#2	I was able to get the program to work after adjusting one of the paths that are called in the code, I was brought to the login page and typed in the proper id and password and that worked fine. It instantly brought me to the Add Employee page, and only had a "Add Employee" button, "Cancel" button, and "Help" button. The help button through a message that said it was a work in progress. But when I tried to add an employee it would just say that I was missing fields even though I wasn't. Once I hit the cancel button something very odd happened. Most of the fields disappeared and the ones that remained could not be edited. The page was still called "Add Employee", the "cancel" button does not do anything. And I tried to exit out of the program but the "X" in the top right corner doesn't work.
#3	Cannot find out which fields that I am typing in are invalid, seems to require me to fill out an hourly wage, salary, and commission. Even though it asks me to decide which one to use. Also has me fill out the end date even though the employee is just starting.
#4	Using the Thonny IDE we were able to get the program running properly, there were some fields that didn't let the user know when they typed the incorrect information, but besides some minor inconveniences it was intuitive enough, besides that the payroll page should be properly explained in the user manual.

Evaluation:

- What we did well
    - The usability test was done on time and offered the opinion of someone who had never touched the code before
- What we would do better
    - Turns out that we mistook what the usability test was supposed to be and the tests above are closer to a user acceptance test than a proper usability test.

Bug Tracking and Resolution

Summary: This is a simplified table of the bug tracking from sprint five

- This is the most recent version of the bug tracking record all bugs new to this sprint are colored blue
- This table was made for the convenience of the shareholder so that he can easily see the state of the bugs found in the program
- Bug descriptions have been simplified for convenience
- Change descriptions have been removed for convenience
- More detailed description of the bug tests can be found in sprint five

ID#	Bug Description	Date Found	Date Fixed	Status
1	All new data is lost when program closes.	02/16/2022	03/02/2022	
2	Tkinter message boxes aren't being imported.	02/23/2022	02/23/2022	
3	Run payroll or run csv file btn non-functional.	03/15/2022	03/18/2022	
4	ID keyed off of # of employees in the database.	03/18/2022	03/19/2022	
5	No user information provided on new PCs	03/20/2022	03/20/2022	
6	File naming isn't explanatory.	03/12/2022	03/20/2022	
7	Issues with validation function error messages.	03/18/2022	03/18/2022	
8	The error messages were hardly noticeable.	03/27/2022	03/27/2022	
9	All fields were editable in the edit page	03/03/2022	03/03/2022	
10	Issues with setting "read only" in general view	03/01/2022	03/01/2022	
11	The ability to add timecards and receipts	03/25/2022	04/10/2022	
12	The search page was not user-friendly	03/27/2022	03/28/2022	
13	Issues moving from search page to view page	03/18/2022	03/18/2022	
14	GUI would not correctly update between pages	02/13/2022	02/13/2022	
15	Entry boxes of view page updating contents	02/13/2022	02/13/2022	

Evaluation:

- What we did well
    - We fixed most of the bugs the day that they were found
    - All of the bugs that have been found have been fixed
- What we would do better
    - Because we tried to fix every bug as early as possible, and we started coding relatively early we didn't document the bugs as they were found. This meant that in sprint five we had to go back and try to remember all of the bugs that appeared and when they were found and fixed. So documenting bugs as they were found.

Document Defect Tracking and Resolution

Summary: This is the document defects table from sprint five, no changes have been made beyond adding additional defects from the previous sprint which will be colored in blue

SPR#	Document Defect Description	Status
SPR-1	Incorrect link location	
SPR-1	Unspecified prototype candidate	
SPR-1	Document Versioning missing	
SPR-1	Coding standards missing	
SPR-1	Meeting schedule missing	
SPR-1	V&V document missing	
SPR-1	Requirements specifications missing	
SPR-2	V&V document missing	
SPR-2	Submitted incorrectly	
SPR-3	GUI images missing	
SPR-3	Backlogs missing	
SPR-3	Code doesn't run	
SPR-3	No readme.txt	
SPR-4	Missing backlogs	
SPR-4	Missing updated class diagram	
SPR-4	Missing User Acceptance Testing	
SPR-4	Missing Functional/Non-functional Testing	
SPR-5	Incorrect Submissions Name for Program Folder	

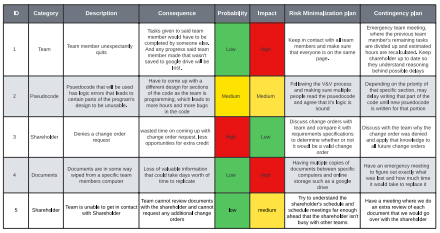
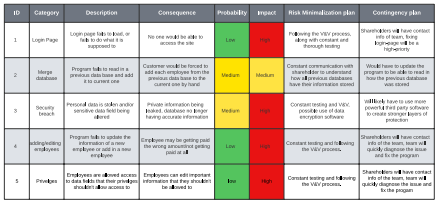
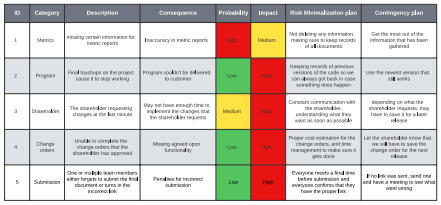
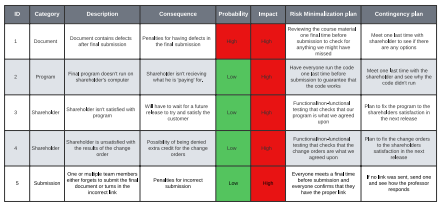
Evaluation:

- What we did well
    - Our defects were usually fixed the following sprint after they were found.
    - We fixed all of the defects that we have found in the document
- What we would do better
    - Have the shareholder review more of our documents before we turned them in so that we catch some of the defects before we submitted the sprint document.
    - Review the rubric as a team before we submit the sprint document to do a last-minute check for defects. (We were doing this by sprint five, but we should've been doing it much earlier)

Risk Management

Summary: This is a table of the risk management plans of each sprint starting from sprint three and their evaluations.

- It contains an individual evaluation of each risk management plan.
- First plan was created in sprint three
- After the table is a comprehensive evaluation of all the risk management plans
- For more detailed information on each plan and larger images view their corresponding sprints

SPR#	Risk Management Plan	Evaluation
3		The only risk that we ran into was the one that was labeled 'High'. This was the shareholder denying multiple of our change order requests. And we followed the contingency plan and in our next meeting we discussed why they were denied.
4		We didn't run into any of the risks that we created in this sprint, however some of these risks could still occur once we give the program to the shareholder.
5		We ran into two risks this sprint, we ran into problems with our metrics lacking certain information that would have been beneficial. The larger problem was the problem we had with the submission of the project. We followed the plan and met with the professor to fix the problem that arose.
6		Because this is the final sprint we won't find out if we ran into any of the risk until it is to late to fix anything. The most likely problem is that we end up missing something from the rubric or other course material as we have had at least one defect per sprint.

Evaluation:

- What we did well
  - We created risk management plans for each sprint that we were supposed to
  - We ran into a couple of the risks that were labeled, usually the ones that were labeled with a 'high' probability and we responded to the risks as labeled in the plan
- What we would do better
  - We ran into many problems that were not inside the risk management plan. This means we could've had a more thought-out plan that covered a wider scope of problems that could have occurred.

Project Performance, Scheduling, and Management Tracking

Team Performance Evaluation

Summary: This is a simplified table version of the team performance document from sprint five

- This is the most recent version of the team performance all new items are colored blue
- This table was made for the convenience of the shareholder so that he can easily see what each member accomplished in each sprint

Sprint	Member	Time	Meetings attended	Tasks Completed
SPR-1	Jaden Albrecht	9hrs	3/3	• Personal MoSCoW chart for Delphi method • Thoroughly read and reviewed SPR-1 document requirements
SPR-2	Jaden Albrecht	35min	4/4	• Distribution of MoSCoW charts for Delphi method • Review SPR-2 document requirements • Review GUI template and use-cases for HLD • Review plain-English description of HLD • Establish connections between project requirements and tasks • Develop minute-tracking method • Keep meeting logs up to date
SPR-3	Jaden Albrecht	19hrs 40min	6/6	• Review all SPR-3 document requirements (100% • Keep meeting logs up to date throughout SPR-3 • SPR-3 change order request form • Develop version history folders in google drive • Build UML class diagrams • Review prototype code) • UML research • Tkinter research • Pytest research
SPR-4	Jaden Albrecht	5hrs	3/3	• Review all SPR-4 document requirements • Keep meeting logs up to date throughout SPR-4 • User manual
SPR-5	Jaden Albrecht	30hrs	11/11	• Review all SPR-5 document requirements • Functional/non-functional requirements testing • Personal user acceptance test

				• Add search parameters and improvements to search page • Help with function-point analysis • SPR-5 meeting logs
SPR-6	Jaden Albrecht	5hrs	5/5	• Team performance evaluations • Final SPR-6 document introduction • Help with final SPR-6 document
SPR-1	Cody Strange	9hrs	3/3	• Personal MoSCoW chart for Delphi method • Thoroughly read and reviewed SPR-1 document requirements • SPR-1 document
SPR-2	Cody Strange	9hrs 20min	4/4	• Fix SPR-1 document • Work breakdown structure for SPR-2 • Construct requirements spec • Synchronize requirements spec with final MoSCoW chart • Choose SPR-1 prototype candidate • SPR-2 document
SPR-3	Cody Strange	17hrs 40mins	6/6	• Review all SPR-3 document requirements • Work breakdown structure for SPR-3 • Risk management plan • Testing documents • Field validation code for login page • Review Prototype code • Fix SPR-2 document • Reorganize google drive folders • Change request board • SPR-3 document
SPR-4	Cody Strange	14hrs	3/3	• Review all SPR-4 document requirements • Work break-down structure for SPR-4 • Usability tests • Risk management plan for SPR-4 • Maintenance plans • Deployment plan • SPR-4 document
SPR-5	Cody Strange	19hrs 40min	11/11	• Review all SPR-5 document requirements • Help with function-point analysis • SPR-5 document • Metrics reports • Personal user acceptance test • Defects list • Risk management plan for SPR-5
SPR-6	Cody Strange	16hrs 20	5/5	• SPR-6 document • Project management and reports
SPR-1	Ethan Taylor	9hrs	3/3	• Personal MoSCoW chart for Delphi method

				• Thoroughly read and reviewed all SPR-1 document requirements
SPR-2	Ethan Taylor	6hrs	4/4	• GUI template • Rough draft for HLD • Review SPR-2 document requirements • Develop GUI classes • Plain-English pseudocode • Develop use-case scenarios based on requirements • Docstring/comment outlining
SPR-3	Ethan Taylor	20hrs 45mins	6/6	• Review all SPR-3 documents • Review prototype code • Pseudocode for Login GUI page • Pseudocode for search page • Pseudocode for view/add/edit page • Login GUI • Search GUI • View/add/edit GUI • Deactivate employee GUI • Payroll GUI • Tkinter research
SPR-4	Ethan Taylor	9hrs 40mins	3/3	• Review all SPR-4 document requirements • Implement add/edit validation code into GUI • Integrate database with payroll page updates • Update payroll page • Tkinter research
SPR-5	Ethan Taylor	25hrs 30min	9/11	• Review All SPR-5 document requirements • Defects analysis • Update UML diagrams • Help button/user manual integration • Integrate old database • Integrate CSV database
SPR-6	Ethan Taylor	5hrs	5/5	• Quality control analysis • Design analysis
SPR-1	Tyler Deschamps	9hrs	3/3	• Personal MoSCoW chart for Delphi method • Thoroughly read and reviewed SPR-1 document requirements
SPR-2	Tyler Deschamps	11hrs	4/4	• Develop Pert/Gantt/burn-down charts for SPR-2 • Review SPR-2 document requirements • Logging for Pert/Gantt charts throughout SPR-2
SPR-3	Tyler Deschamps	15hrs	6/6	• Review all SPR-3 document requirements

				• Update Pert/Gantt/burn-down charts throughout SPR-3 • Review prototype code • Search employee database pseudocode • Database pseudocode • CSV pseudocode • Database tables to CSV • JSON research
SPR-4	Tyler Deschamps	15hrs 30min	2/3	• Review all SPR-4 document requirements • Update Pert/Gantt/burn-down charts throughout SPR-4 • Report receipts • Implement database logic into GUI • Merge database with Emp_Dat_V3 • Payroll and CSV integration
SPR-5	Tyler Deschamps	12hrs	11/11	• Review all SPR-5 document requirements • Defects analysis • Backlogs for SPR-5 • Personal user acceptance test • SPR-5 charts development
SPR-6	Tyler Deschamps	16hrs	5/5	• Decomposed Gantt chart Report • Backlog changes report • Pert chart report • Final burn-down chart report • Maintenance plan • Backlogs for previous sprints
SPR-1	Jordan Van Patten	n/a	n/a	n/a
SPR-2	Jordan Van Patten	1hr 30min	3/4	• Help with GUI class development • Review SPR-2 document requirements • V&V covering requirements and documentation • Review use-cases for HLD
SPR-3	Jordan Van Patten	6hrs 50min	3/6	• Review all SPR-3 document requirements • Review prototype code • Team file/folder naming conventions document • Quality assurance plan document • V&V documents for SPR-3 • Pytest research • Black research • Debugging software research
SPR-4	Jordan Van Patten	6hrs 30min	3/3	• Review all SPR-4 document requirements • V&V documents for SPR-4 • Test cases for add/edit/view and search GUI pages

				• Update quality assurance plan • Implement validation code for add/edit/view GUIs
SPR-5	Jordan Van Patten	3hrs 10min	9/11	• Review all SPR-5 document requirements • V&V documents for SPR-5
SPR-6	Jordan Van Patten	2hrs	2/5	• Bug testing reports
SPR-1	Kole Davis	n/a	n/a	n/a
SPR-2	Kole Davis	n/a	n/a	n/a
SPR-3	Kole Davis	n/a	n/a	n/a
SPR-4	Kole Davis	n/a	n/a	n/a
SPR-5	Kole Davis	2hrs 30min	9/11	• Review all SPR-5 document requirements • Ishikawa diagrams • User acceptance test report
SPR-6	Kole Davis	1hr	5/5	• V&V documents

Evaluation of Sprint One

Team Member	What was expected	What could be improved
Jaden Albrecht	• Read and review SPR-1 requirements • Finish personal MoSCoW chart for Delphi method • Establish a role	• Nothing to address
Cody Strange	• Read through requirements for sprint • Research agile methodologies • Create sprint document	• Started research before our team was even created
Tyler Deschamps	• Development of Time Management Chart Templates. • Develop WBS template. • Develop Moscow Chart Template. • Read assigned chapters. • Research tKinter. • Attend all meetings including those with Shareholder.	• Could have designed a color-coded theme that was consistent with WBS, and the time management charts.
Ethan Taylor	• Read through all the introduction material	• I should've taken notes while reading, so I wouldn't have to keep referring back Canvas when I had questions.
Jordan Van Patten	n/a	n/a
Kole Davis	n/a	n/a

Evaluation of Sprint Two

Team Member	What was expected	What could be improved
Jaden Albrecht	• Read and review all SPR-2 document requirements • Review GUI template and plain-English code • Distribute MoSCoW charts for Delphi method • Establish a minute-tracking method • Update meeting logs throughout the sprint	• I should have updated the meeting logs after each meeting rather than waiting until the sprint due date to do them all

Cody Strange	• Work breakdown structure • Sprint document • MoSCoW chart	• Read through the requirements more thoroughly • Update the Trello board more often
Tyler Deschamps	• Develop a Requirements doc to do Delphi method on with Moscow Chart. • Manage tasks on Trello. • Compile time logs to a Gantt chart. • Make time estimations and create sprint Pert Chart. • Make a burndown chart. • Attend all meetings including those with Shareholder.	• Should have taken the initiative of checking on people's tasks to make sure they were done in Trello instead of expecting team leader to.
Ethan Taylor	• MoSCoW chart • High-level Design	I should have finished these task early enough to allow the shareholder to review them before submission.
Jordan Van Patten	• Review Use case for high level design • review spr-2 doc requirements • V&V covering requirements and documentation	I should have completed all tasks sooner than i had
Kole Davis	n/a	n/a

Evaluation of Sprint Three

Team Member	What was expected	What could be improved
Jaden Albrecht	• Read and review all SPR-3 document requirements • Update meeting logs throughout the sprint • Create change order request form • Review prototype code • Build UML class diagrams • Research Tkinter • Research Pytest • Research UML	• Could have done more research on how to construct a change order request form • Could have established a better method of building the UML diagrams
Cody Strange	• Sprint document • Change control board • Change order form • Quality control plan • Risk management plan Work breakdown structure	• Figure out what all of the requirements for the sprint are at the start of the sprint and get a better time estimate for the tasks
Tyler Deschamps	• Begin to develop pseudocode, code, and database for the back end of the project. • Compile time logs to a sprint Gantt chart. • Make time estimations and create sprint Pert Chart. • Make a burndown chart. • Attend all meetings.	• Clearer communication between the myself and the GUI developer to avoid reworking each other's code.
Ethan Taylor	• Low-level Design	• Pseudocode, began implementing GUI
Jordan Van Patten	• Review all spr-3 document requirements • review prototype code • file/folder naming convention document, • V&V document • pytest and black research • debugging software research	• I should have written better notes while doing the research, and uploaded the finished documents to our google drive as soon as they were finished
Kole Davis	n/a	n/a

Evaluation of Sprint Four

Team Member	What was expected	What could be improved

Jaden Albrecht	• Review all SPR-4 document requirements • Update meeting logs throughout the sprint • Build user manual	• Could have made more adjustments to the user manual after changes to the code were made
Cody Strange	• Sprint document • Deployment plan • Work breakdown structure • Risk management plan	• Research better what a usability test is
Tyler Deschamps	• Further Develop back-end logic in the program. • Compile time logs to a sprint Gantt chart. • Make time estimations and create sprint Pert Chart. • Make a burndown chart. • Attend all meetings.	• We were all confused on what was expected on the payroll. Should have clarified very early on in the project with the professor what was expected there.
Ethan Taylor	• Create fully functioning GUI	• Reviewing the features with the shareholder would've been a good idea, since there were some he didn't like.
Jordan Van Patten	• Review SPR-4 document requirements, • V&V document for SPR-4 • test cases for add/view/edit/search GUI pages • update quality assurance plan, pytest documents for payroll page • implement validation code for add/edit/view GUIs	• i should have started working on and finished the test cases/validation code earlier
Kole Davis	n/a	n/a

Evaluation of Sprint Five

Team Member	What was expected	What could be improved
Jaden Albrecht	• Review all SPR-5 document requirements • Update meeting logs throughout the sprint • Functional/non-functional requirements testing • Team performance review • Add search parameters/improvements to search page (change order development) • Personal user acceptance test • Function-point analysis	• Could have spent less time working on the change order and more time on function-point analysis, functional/non-functional requirement tests, and team performance review • Could have updated the meeting logs more frequently, considering how many meetings there were for this sprint)
Cody Strange	• Sprint document • Metric Reports • Work breakdown structure • Risk management plan • Function-point analysis	• Pay better attention to what attributes would actually give useful information for a metric report
Tyler Deschamps	• Compile time logs to a sprint Gantt chart. • Make time estimations and create sprint Pert Chart. • Make a burndown chart. • Attend all meetings	• During V&V I should have been more thorough through the code files so we could have avoided the code folder being named wrong.
Ethan Taylor	• Integrate database with GUI • update UML class diagram • integrate user manual with GUI	• More thorough testing while I was integrating the database would've saved me some trying to debug it, since I didn't catch the bugs until late in the coding process.
Jordan Van Patten	Review all SPR-5 document requirements, V&V for SPR-5	• I should have spent some time during this sprint in updating our pytest test cases to get ahead and prepare for our last sprint

Kole Davis	Ishikawa Diagram, Bug Report	• I could have made myself available to help others

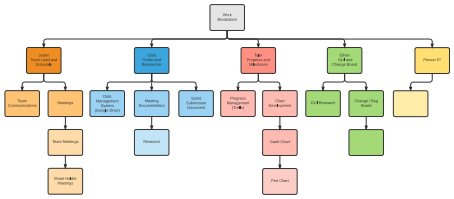
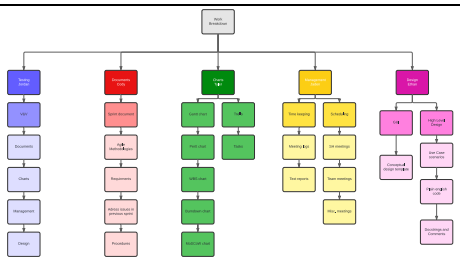
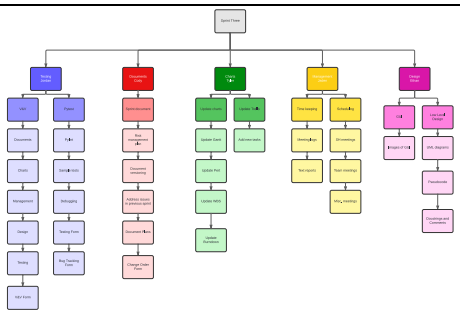
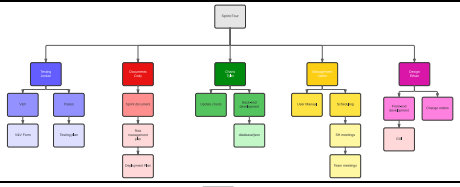
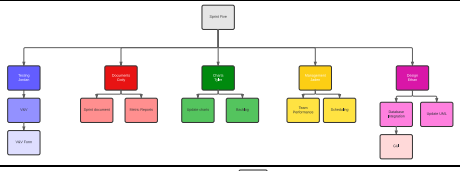
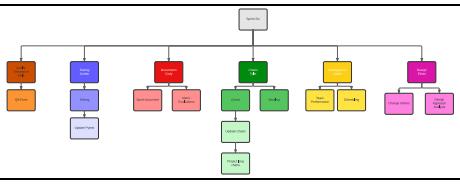
Evaluation of Sprint Six

Team Member	What was expected	What could be improved
Jaden Albrecht	• Review all SPR-6 document requirements • Update meeting logs throughout the sprint • Team performance evaluation • SPR-6 document introduction • Help put SPR-6 document together	• Could have given myself more to do for this sprint
Cody Strange	• Sprint document • Report evaluations • Appendix	Time management, start sooner and don't push work onto the last minute
Tyler Deschamps	• Create a maintenance plan. • Create a full project Gantt chart and analysis with current sprint Gantt chart. • Create a full project Pert chart and analysis as with a current sprint Pert chart. • Create a project backlog report with backlogs for previous sprints. • Attend all meetings.	• We all could have started on our tasks earlier on in the sprint. • No time was really logged, except for meetings on the first week of this sprint. • Should have read requirements the day after we turned in the previous sprint so we could be ready to assign tasks at the first meeting.
Ethan Taylor	• User acceptance analysis • project design and approach	• It would've been nice if I had finished these quickly enough for the shareholder to have time to review them.
Jordan Van Patten	• update test cases to make sure they work with our final code	• I should have updated the test cases earlier in this sprint to give more time in case I had noticed any bugs while testing that needed to be fixed.
Kole Davis	V&V	• Could have made myself more available.

Project Long WBS Evaluations

Summary: This is a table of the work breakdown structures of each sprint and their evaluations.

- This table is so the shareholder can quickly assess what the WBS looked like and what our teams evaluations on said WBS
- It contains an individual evaluation of each work breakdown structure
- After the table is a comprehensive evaluation of all the work breakdown structures

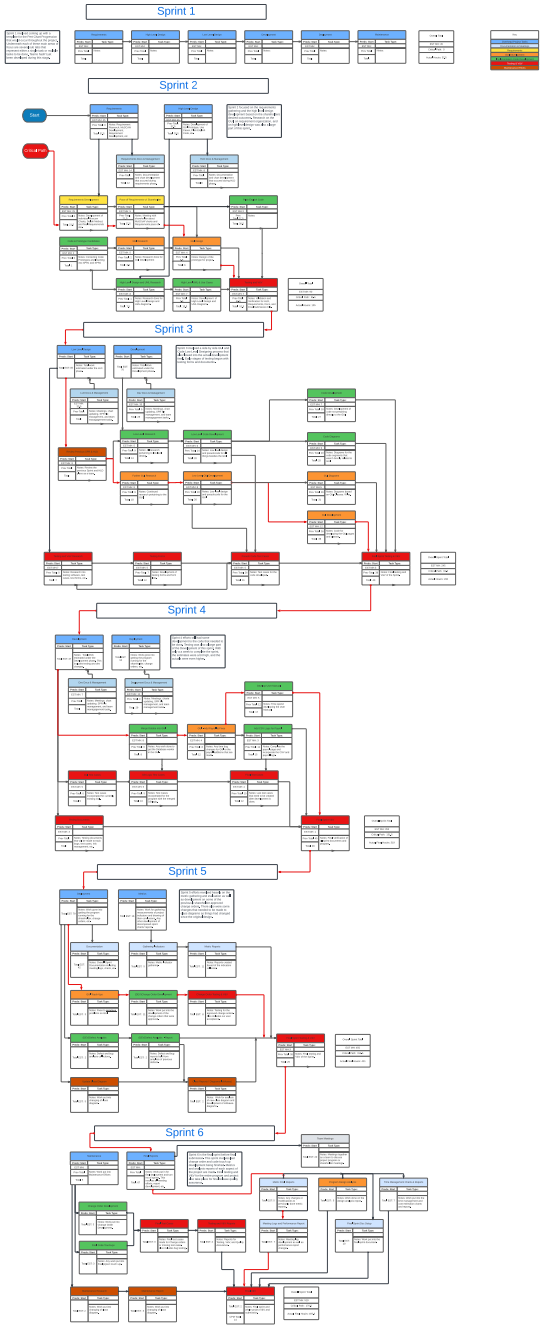
SPR#	Work Breakdown Structure	Evaluation
1		Original WBS, colors used for members/roles are different from all of the other WBS documents. Went over everyone's roles and what they were responsible for.
2		Colors for roles were changed and roles themselves were simplified. WBS went over the tasks that each member was responsible in overseeing.
3		As the sprints got larger so did the WBS because it was done based off of each task that each member was responsible for
4		We decided to change the structure of the WBS to just show the major tasks that each member was responsible for. This is why the WBS has significantly shrunk in size
5		Once again we just put on the major tasks that each member was responsible for. However we did forget to put our newest member on the WBS.
6		We added in our final member and listed everyone's major tasks that they are responsible for.

Evaluation:

- What we did well
    - We created an accurate WBS for each sprint that went over the responsibilities of each team member
- What we would do better
    - The WBS was usually created later than it should've been. For next time we could create it in the first meeting that we come up with the tasks for that sprint.

Project Long Pert Chart Evaluations

Summary: This is a project long pert chart that is a combination of each of the individual pert charts.



Evaluation:

- What we did well
    - We create an accurate pert chart for each sprint and then compiled them into
- What we would do better
    - We would keep them as is

Project Long Gantt Chart Evaluations

- Summary: This is decomposed project long gantt chart.

Task Type 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

Sprint 1				
Task Type	Task Name/Description	Actual Hours	Est. Hours	Days Worked On
DOC	Set up Code and Document Sharing	1	3	1/17
MEET	Team Meetings	13	10	1/17, 1/19, 1/21/, 1/24
DOC	Agile Methodologies Research	5	10	1/18, 1/20
DOC	Gantt, Pert, Moscow, and other time chart templates	4	5	1/18, 1/21
GUI	GUI research	7	5	1/17, 1/18/ 1/20, 1/22
DOC	Sprint 1 Submission document	5.5	10	1/19, 1/21, 1/24
SBMT	Sprint 1 Submission	2	2	1/24
	Sprint Totals	37.5	45	
	Overall Totals	37.5	45	

Sprint 2				
MEET	Team Meetings and SPR 2 Planning	10	10	1/24, 1/28, 2/3
REQ	Requirements Research	7.5	9	1/28, 1/29
REQ	MoSCoW Chart Development	8	10	1/27, 1/29, 1,31
GUI	GUI Research	2.5	5	1/25, 1/27, 2/3
REQ	SPR2 & MoSCoW Discussion	4	5	1/31, 2/4
REQ	Meeting w/ shareholder	5	5	1/31, 2/7
REQ	MoSCoW to Requirements	2	3	2/2, 2/3
GUI	GUI Framework Development	2.5	4	2/2, 2/4
DEV	High Level Research	7	8	1/28, 1/29, 2/3, 2/4
DEV	Plain English Code Dev	2	3	2/3
DEV	High Level UML	2	3	2/5, 2/7
DEV	Use Cases	2	4	2/5
DOC	SPR1 Review & Update	2.5	2	1/28, 2/4
DOC	Gantt, Pert, & Burndown Charts	6.5	8	1/31, 2/2, 2/3, 2/4, 2/5
DOC	Work Breakdown Structure	1	2	2/5
TEST	GUI Represents Requirements V&V	1	3	2/5
TEST	Plain English Code & UML V&V	1	2	2/5
TEST	SPR2 Submission V&V	1.5	2	2/7
SBMT	SPR2 Submission	2.5	2	2/7
	Sprint Totals	70.5	90	
	Overall Totals	108	135	

Sprint 3				
SBMT	Turn in SPR 2	2.5	3	2/7
REV	Review SPR 2	2.5	5	2/7
MEET	(1) Team Collaberation	13.5	10	2/11, 2/14, 2/18
REQ	Review Requirments & Docs for SPR 3	3.5	3	2/9, 2/10
DOC	Create Coding Standard Doc	0.5	3	2/15
DEV	Research Low Level Design	3	5	2/12, 2/16
GUI	Tkinter GUI Research	3	5	2/12, 2/14
GUI	(1) GUI Development & Pseudocode	2	3	2/14, 2/16
DEV	(1) Dev Pseudo Code	3	3	2/14, 2/16
DOC	(1) Progress Documentation	4	3	2/12, 2/15, 2/16
DOC	(1) Chart and Time Management	2	2	2/16
DEV	Development Research	3.5	3	2/12, 2/15, 2/16, 2/17
MEET	Review SPR 2 with SH	5	2	2/16, 2/17
TEST	(1) Testing Research	2.5	5	2/16
DOC	SPR 3 Doc Development	6	5	2/16, 2/17, 2/24
DEV	(2) Dev Pseudo Code	4.5	5	2/18, 2/21
REV	Research V&V	1	3	2/25
GUI	(2)Tkinter GUI Research	3.5	4	2/18
GUI	(2) GUI Development & Pseudocode	11	15	2/18, 2/19, 2/21, 2/23, 2/26
MEET	(2) Team Collaberation	7	5	2/21, 2/23
DOC	(2) Progress Documentation	7.5	5	2/23, 2/24
DOC	(2) Chart and Time Management	1.5	3	2/23
TEST	Test Cases	3	5	2/26
TEST	Test Forms	2.5	5	2/25
MEET	(3) Team Collaberation	9	12	2/28, 3/2, 3/4, 3/6
GUI	(3)Tkinter GUI Research	0.5	0	3/2
GUI	(3) GUI Development & Pseudocode	1	2	3/1
DEV	(3) Dev Pseudo Code	2.5	5	3/1, 3/4
DOC	(2) Progress Documentation	1.5	5	3/2, 3/3
DOC	(2) Chart and Time Management	2.5	3	3/4, 3/9
DEV	UML Development	11.5	10	3/3, 3/5, 3/7, 3/8
TEST	(2) Testing Research	2.5	2	3/8
MEET	(4) Team Collaberation	3	3	3/10
TEST	Final SPR 3 V&V	3	3	3/11
	Sprint Totals	135	155	
	Overall Totals	243	290	

Sprint 4				
SBMT	Submit SPR 3	0	-	3/1
TEST	Testing Plans & Documents	4.5	3	3/11, 3/14
REV	Review Progress of SPR 3	2	2	3/14
MEET	SPR 3 Planning In Team Meetings	4.5	5	3/11 3/14
DEV	Meet to Go Over Program Database	3	2	3/15
DEV	Integrate Employee JSON Database	6	6	3/15, 3/16
DOC	(1) Sprint 4 Document Dev	4	4	3/15
TEST	GUI Test Cases	6	5	3/16, 3/17
GUI	GUI Changes & Bug Fixes	2.5	2	3/15
DEV	Merge GUI Changes with Program	1	1	3/17
MEET	Team Meeting	9.5	7.5	3/16, 3/18, 3/19
TEST	Program Test Cases & Documents	6.5	5	3/18, 3/19, 3/21
GUI	Payroll CSV Changes	3.5	2	3/16, 3/18
DEV	User Manual	5.5	5	3/18, 3/19
DEV	Merge CSV Logic into GUI	6	3	3/19
MEET	Meet with Shareholder	3	2	3/21
MEET	Team Meeting	3	2.5	3/21
TEST	Final Sprint V&V	1	1	3/21
DOC	(2) Sprint 4 Document Dev	2.5	5	3/19, 3/21
DOC	Chart / Time Management	2	2	3/21
SBMT	Submit SPR 4	0	-	3/21
	Sprint Totals	76	65	
	Overall Totals	319	355	

Sprint 5				
SBMT	Submit SPR4	0	-	3/22
MEET	(1)Team Meetings	9.5	8	3/23, 3/25, 3/28
GUI	GUI Touch-ups	2	3	3/23
DOC	(1)Backlogs / Task list	2	3	3/30, 4/1
DOC	Collect Metric Indicators	1.5	5	4/1
DOC	(1)Create Metric Reports	1	3	4/1
DOC	(1)Meeting Log / SPR Doc	4	5	3/28, 4/4
DEV	Code Defect Analysis	2.5	2	3/31
DEV	(1)Change Order Development	6.5	3	3/28, 3/30, 4/1
MEET	(2)Team Meetings	8	8	3/30, 4/1, 4/4, 4/6
DOC	(2)Create Metric Reports	2.5	3	4/12
DEV	UML Update / Program Fixes	7.5	7	4/4, 4/6, 4/8, 4/11
TEST	User Acceptance	3	3	4/12, 4/14, 4/15
DEV	(2)Change Order Development	9.5	3	4/4, 4/7, 4/8, 4/11
DEV	User Manual / Help	8.5	5	4/8, 4/11, 4/13, 4/15
DOC	(2)Backlogs / Task list	0.5	1	4/13
DOC	Team Plans	1.5	2	4/13, 4/15
DOC	Defect Analysis Report	2	2	4/9, 4/12
MEET	(3)Team Meetings	9.5	8	4/11, 4/13, 4/15, 4/16
DOC	(1)Sprint Document Charts / Diagrams	5	5	4/8, 4/13, 4/15, 4/16
TEST	Testing of Implemented Change Orders	2	5	4/16
DOC	Function Point Analysis	3	2	4/16
DOC	(2)Meeting Log / SPR Doc	5	5	4/15, 4/16, 4/18
DOC	(3)Create Metric Reports	2.5	1	4/14
TEST	Final Sprint V&V	3	3	4/18
SBMT	Submit SPR5	0	-	4/18
	Sprint Totals	102	95	
	Overall Totals	421	450	

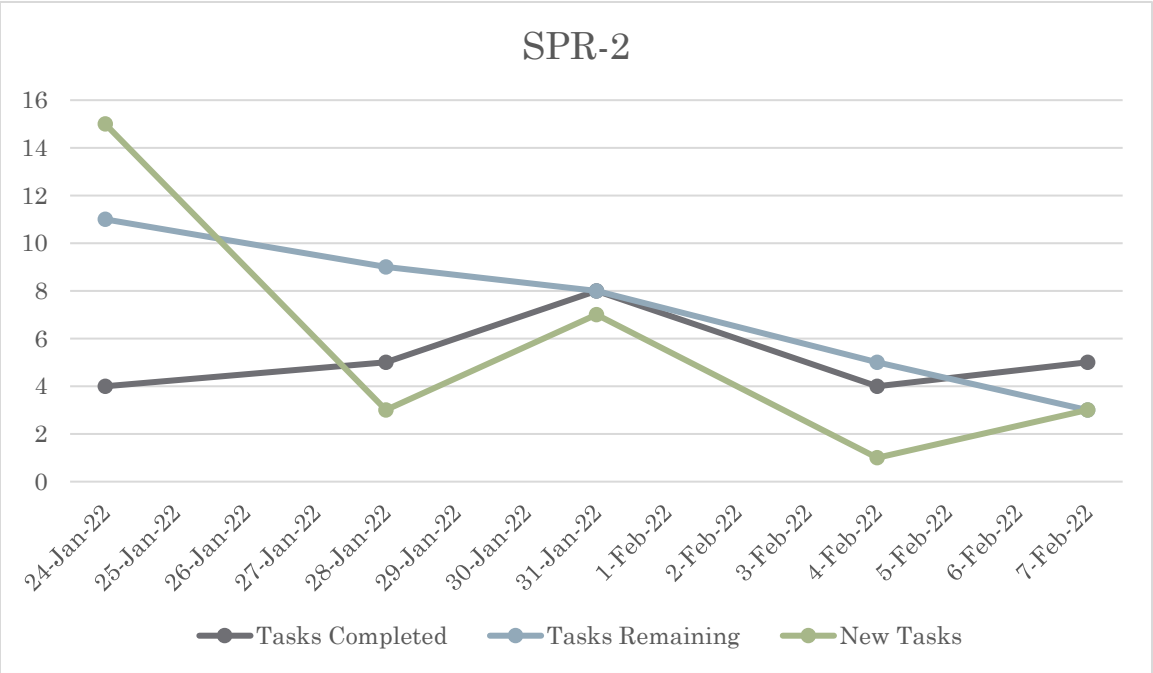
Sprint 6				
SBMT	Submit SPR5	0	-	Times Not Yet Finished
MEET	(1)Team Meetings	10	10	4/20, 4/22, 4/25
GUI	Design/Approach Analysis	3	2	4/23, 4/27
MTNC	Maintenance Research & Plan	3	5	4/25
DEV	Change Order Development	0	3	
DEV	Last Program Touch-ups	0	3	
TEST	Quality Control / V&V Doc	1	3	4/27, 4/29
TEST	User Acceptance Analysis	1	2	4/29
DOC	Meeting Logs	2	7	4/28, 4/30
DOC	Cost Estimation (Gantt and Pert Charts)	7	5	4/29, 4/30, 5/2
DOC	Sprint 6 Final Project Doc	11	10	4/26, 4/28, 4/29
DOC	Back Logs and Report	3.5	2	4/28, 4/29, 4/30
DOC	Other Metric Reports	5	5	4/27, 4/28
DOC	Team Performance	2	1	4/30, 5/2
TEST	Finish Bug Tests & Reports	2	2	4/30
MEET	(2)Team Meetings	20.5	10	4/26, 4/27, 4/29, 5/2
TEST	Final Sprint V&V	2	2	5/2
SBMT	Submit Final Sprint	0	-	5/2
	Sprint Totals	77	72	
	Overall Totals	497.5	519	

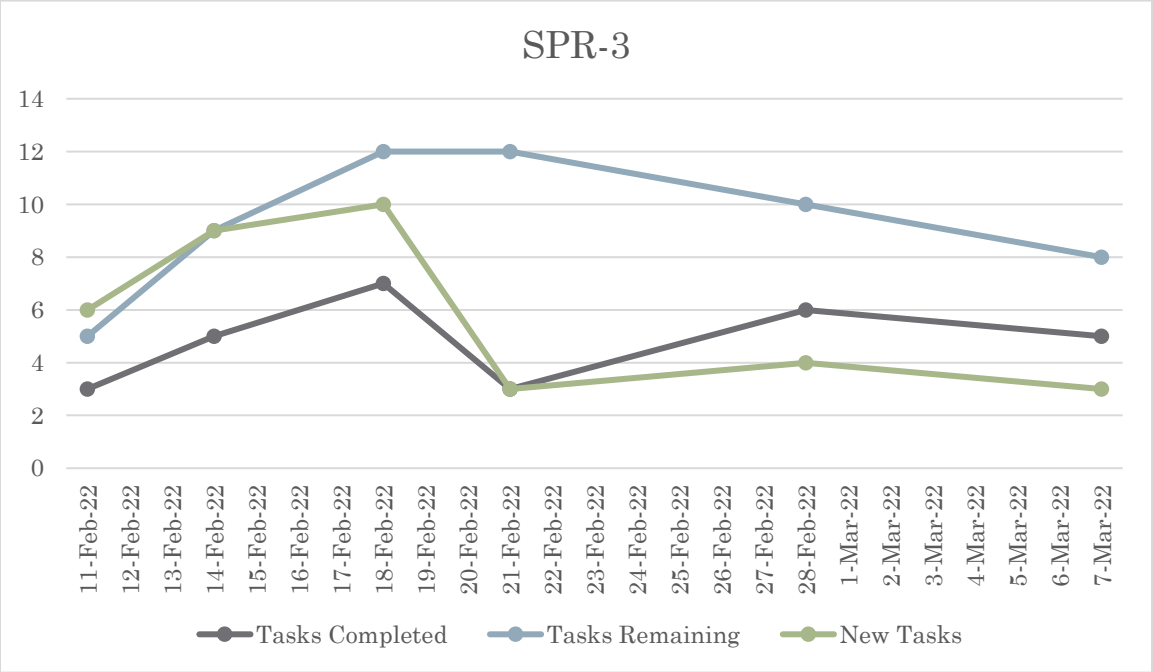
Evaluation:

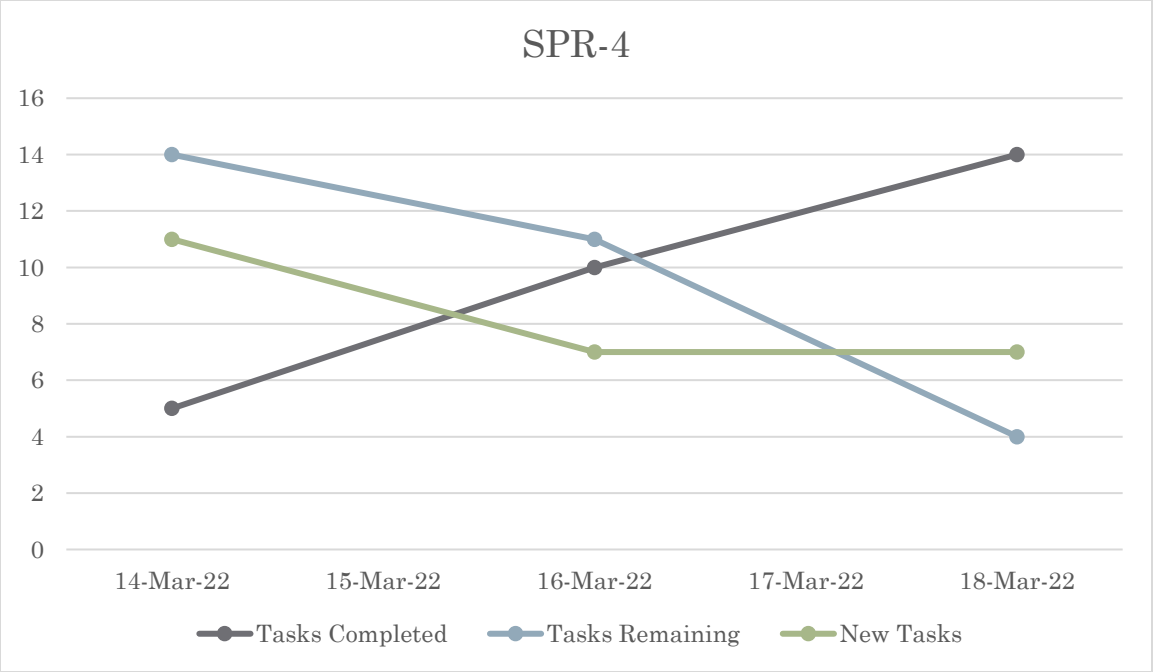
* Important Note - Estimated hours were not always based on 9-hour weeks. Some sprints time estimates were gauged from individual task time estimates. * It's extremely difficult to estimate the time required to accomplish tasks and sprints in a project. We were a bit inaccurate with our estimations in the beginning of the project. However, as time progressed, each sprint tended to show a more accurate estimation. The percent error between the estimated times and the actual times for each sprint grew lower and lower. This illustrates that the team improved in areas such as working well together as a team working as well as areas of workload competency.

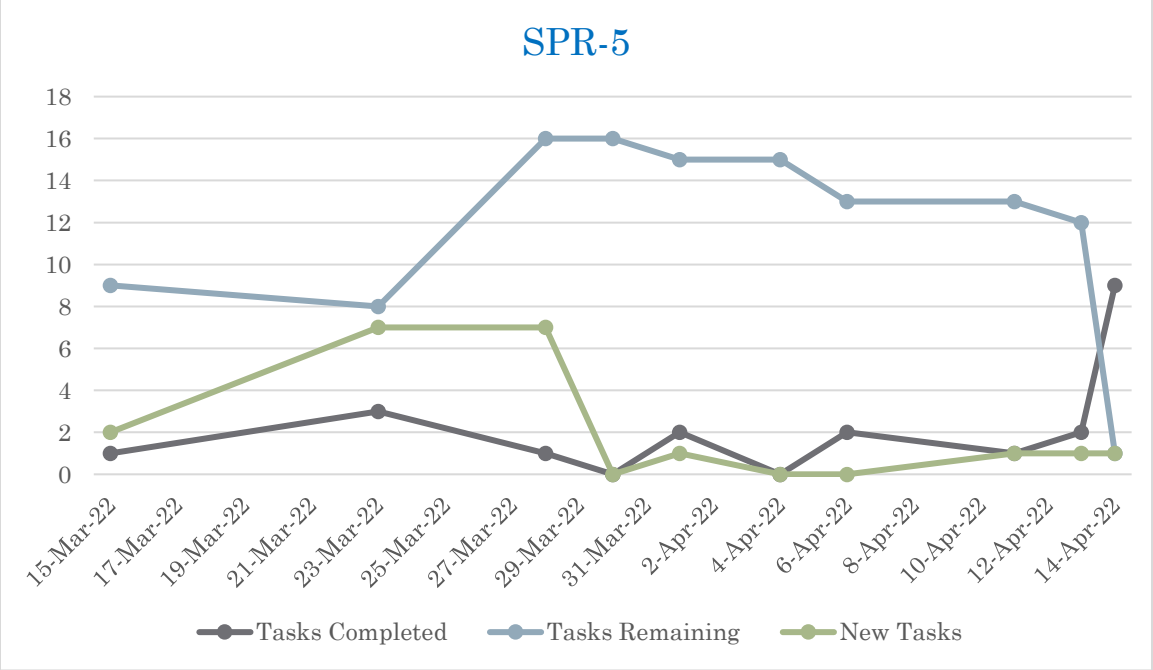
Tasks Completed to Tasks Remaining by Sprint Evaluation

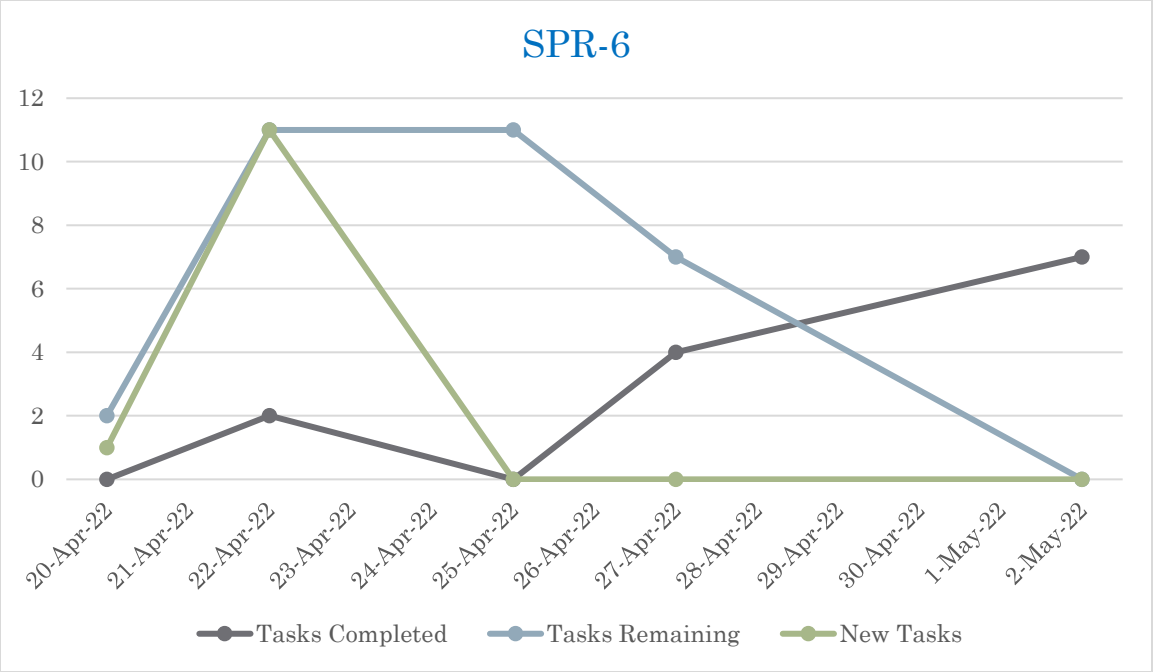
Summary: These charts are designed to show how many tasks were completed by each meeting where they could be reported to the overall tasks remaining. It also shows the amount of new tasks that were being added which would affect the overall tasks remaining.






SPR-4
16
14
12
10
8
6
4
2
0
14-Mar-22
15-Mar-22
16-Mar-22
17-Mar-22
18-Mar-22
Tasks Completed
Tasks Remaining
New Tasks


SPR-5
18
16
14
12
10
8
6
4
2
0
15-Mar-22
17-Mar-22
19-Mar-22
21-Mar-22
23-Mar-22
25-Mar-22
27-Mar-22
29-Mar-22
31-Mar-22
2-Apr-22
4-Apr-22
6-Apr-22
8-Apr-22
10-Apr-22
12-Apr-22
14-Apr-22
Tasks Completed
Tasks Remaining
New Tasks


SPR-6
12
10
8
6
4
2
0
20-Apr-22
21-Apr-22
22-Apr-22
23-Apr-22
24-Apr-22
25-Apr-22
26-Apr-22
27-Apr-22
28-Apr-22
29-Apr-22
30-Apr-22
1-May-22
2-May-22
Tasks Completed
Tasks Remaining
New Tasks

Project Long Meeting Minutes Evaluations

Summary: This is the team meetings table from sprint five metrics. The information from it will be used to help evaluate each sprint and any new data added to the table will be colored in blue. The evaluation of each sprint will be done in a table below this one. And after the evaluation of each individual sprint's meeting minutes there will be a comprehensive evaluation of all the sprints combined.

Meeting	Sprint	Date	Attendees	Length of meeting	Tasks completed	Tasks remaining	New tasks added
1	SPR-1	January 20, 2022	4/4	1hr 12min	n/a	n/a	n/a
2	SPR-1	January 21, 2022	4/4	41min	n/a	n/a	n/a
3	SPR-2	January 24, 2022	4/4	55min	4/15	11	15
4	SPR-2	January 28, 2022	5/5	1hr 5min	5/14	9	3
5	SPR-2	January 31, 2022	5/5	1hr 15min	8/16	8	7
6	SPR-2	February 4, 2022	4/5	55min	4/9	5	1
7	SPR-2	February 7, 2022	5/5	40min	5/8	3	3
8	SPR-3	February 11, 2022	4/5	1hr 4min	3/9	5	6
9	SPR-3	February 14, 2022	5/5	1hr 3min	5/14	9	9
10	SPR-3	February 18, 2022	4/5	1hr 8min	7/19	12	10
11	SPR-3	February 21, 2022	4/5	1hr 6min	3/15	12	3
12	SPR-3	February 28, 2022	5/5	1hr 4min	6/16	10	4
13	SPR-3	March 7, 2022	6/6	20min	5/13	8	3
14	SPR-4	March 14, 2022	5/6	1hr 4min	5/19	14	11
15	SPR-4	March 16, 2022	5/6	44min	10/21	11	7
16	SPR-4	March 18, 2022	6/6	45min	14/18	4	7
17	SPR-5	March 23, 2022	5/6	15min	3/11	8	7
18	SPR-5	March 25, 2022	6/6	41min	1/10	9	2
19	SPR-5	March 28, 2022	5/6	1hr	1/17	16	7

20	SPR-5	March 30, 2022	5/6	15min	0/16	16	0
21	SPR-5	April 1, 2022	5/6	30min	2/17	15	1
22	SPR-5	April 4, 2022	5/6	15min	0/15	15	0
23	SPR-5	April 6, 2022	5/6	15min	2/15	13	0
24	SPR-5	April 11, 2022	5/6	20min	1/14	13	1
25	SPR-5	April 13, 2022	6/6	15min	2/14	12	1
26	SPR-5	April 15, 2022	5/6	23min	3/13	10	1
27	SPR-5	April 18, 2022	5/6	4hr 30min	9/10	1	0
28	SPR-6	April 20, 2022	5/6	30min	0/2	2	1
29	SPR-6	April 22, 2022	5/6	47min	2/13	11	11
30	SPR-6	April 25, 2022	5/6	1hr 9min	0/11	11	0
31	SPR-6	April 27, 2022	6/6	49min	4/11	7	0
32	SPR-6	May 2, 2022	6/6	1hr	7/7	0	0

Evaluation:

- What we did well
    - We met often and nearly everyone would show up to each meeting.
- What we would do better
    - Try to reduce the length of certain meetings so that we could work more efficiently

Individual Sprint Evaluations

Summary: These tables are designed to give some information to assist in evaluating sprint one specifically. These are the attributes that we have determined to best show how well the sprint left. We chose to leave out tasks completed because of how the size of tasks tend to differ it would be a misleading indicator. Each table comes with some important notes to explain some important details of the sprint that might skew the evaluation and an Evaluation of the sprint.

Sprint One

Sprint length (days)	14
Meetings	2
Defects	7
Predicted hours	45
Actual hours	37.5
Grade	102.64%

Important Notes:

- Sprint one grade was inflated because we only had 4 members on our team compared to the 6 members that some other teams had.
- Though the sprint was technically 14 days long we really didn't know to get started and our team wasn't created until around day 7-10

Evaluation: It was our very first sprint and we had a team of four. We struggled in the beginning trying to figure out what needed to be done but we had multiple meetings in a row so that we could figure it out so that we could have a workable document completed by the deadline

Sprint Two

Sprint length (days)	14
Meetings	5
Defects	2
Predicted hours	90
Actual hours	70.5
Grade	85%

Important Notes:

- We had fixed defects from sprint one but didn't submit that we did to the professor, so we didn't get any extra credit for it, hence the low grade
- Just got our fifth member

Evaluation: We had a better understanding of how to go about doing the project and we got our fifth member. We fixed up the problems from the first sprint but failed to tell the professor about it, so we didn't get credit for it. We spent a lot of the sprint figuring out the requirements for the sprint and we begun the high level design along with a GUI template. Along with getting our charts and time management started. This was where most of our record keeping started.

Sprint Three

Sprint length (days)	33
Meetings	6
Defects	4
Predicted hours	155
Actual hours	135
Grade	92.66%

Important Notes:

- This was the sprint that had spring break at the end of it hence the few meetings over a very long period of time

Evaluation: We finished the low-level design and began the implementation in this sprint. There was spring break during this sprint and that lead to a couple of issues. We finished up most of the sprint before the sprint but underestimated the amount of work that was left so we were left rushing to get everything finished in time.

Sprint Four

Sprint length (days)	10
Meetings	4
Defects	4
Predicted hours	65
Actual hours	76
Grade	88.6%

Important Notes:

- Sprint three was extended into sprint four's time period so sprint four was only a week long

Evaluation: This sprint was significantly shorter but because we finished most of the implementation in sprint three it wasn't that difficult of a sprint. We learned from the previous sprint and figured out everything that needed to be done by the first meeting and got on top of it.

Sprint Five

Sprint length (days)	27
Meetings	11
Defects	1
Predicted hours	95
Actual hours	102
Grade	116.66%

Important Notes:

- Got our sixth member

Evaluation: This sprint was a rather long amount of time, so we spent most of the first week or so figuring out what exactly needed to be done for the sprint so that when the time came we were able to efficiently knock out the tasks as fast as possible. We ran into some issues because

we submitted the code with the wrong name but was able to contact the professor and get it fixed.

Sprint Six

Sprint length (days)	14
Meetings	6
Defects	n/a
Predicted hours	72
Actual hours	77
Grade	n/a

Important Notes:

- Grade and defects are unknown

Evaluation: This was our final sprint and we had two weeks to finish it up. We spent the first week figuring out all of the requirements and clearing up some problems with the previous sprint with the professor. The second week was spent gathering information from the previous sprints and updating/evaluating it. Then combining everything from all the sprints into one final document.

Overall Project

Project length (days)	112
Meetings	33
Defects	15
Predicted hours	522
Actual hours	497.50
Grade	n/a

Important Notes:

- Grade can change based results of sprint six

Evaluation: It was a very educational experience if not very difficult and stressful at times. We split of the tasks into roles for everyone while still learning how to work together as a team. We learned a lot through each sprint and got better with each progressive sprint. After the third sprint we were confident in how to go about completing each sprint and gathering the requirements and splitting up the tasks as soon as possible so each member could complete their responsibilities for that sprint.

Maintenance

Maintenance Plan

Overview

Summary: Our efforts throughout all planning processes and through development are always focused on providing a quality product that would last longer than the shareholder would expect it to. In our planning and development efforts, we designed the code structure in such a way so that it can grow, be modified, added to, or easily navigated to make repairs. The plan for our handling of those future events follows four main areas of focus. They are Perfective, Adaptive, Corrective, and Preventative efforts.

Maintenance Plan Breakdown

Perfective (Cost Estimation: 50%)

- We know that at best, there can still be additions to the software after release. Even with the released product being up to working requirements, there are improvements that can be made, and new features that can be added. Because of this, our time and cost estimation will largely be funded towards the efforts of project addition change orders. We plan for this to be handled by change orders requests that are submitted to give proper notice and detail. Our team will then use those requests and reports to further the product quality and effectiveness through development, testing, and verification/validation.

Adaptive (Cost Estimation: 25%)

- For businesses, their products, services, or processes can and often change. There will likely be needs for changes to be made for the software that we developed. We decided (as it is recommended) to allot 25% of our cost estimation and time for adapting to the current needs of the shareholder. We plan for this to be accomplished through current project functionality change orders that will allow our team to tailor the software to the shareholders new specific needs and requirements.

Corrective (Cost Estimation:20%)

- Since there will always be small issues or problems that will arise over time in the software, some of our cost estimation and time will be put towards the patching of bug and problem fixes. Corrective efforts are important as they keep the program in working order, but with the level of quality that the software is released at, the estimation of time and cost is substantially lower than that of the perfective efforts. We plan and expect for users to promptly send reports of bugs or problems with the software to our team so that a solution can be quickly developed and released.

Preventative (Cost Estimation: 5%)

- With much of our budget being focused on good corrective, adaptive, and perfective methods, very little of the cost estimation budget and time will be focused on preventative (restructuring of code for maintainability) efforts. A large focus of our pre-release development involved thorough testing by multiple developers. Because of this, we put extra effort previously into trying to keep things friendly upon revisitation of developers. As time goes on, we plan to spend time where necessary to allow for cleaner representation of the shareholders requirements in our organization of our code.

User Manual

This manual is designed to help you get started using the EmpDat program and will describe how to use some of its most basic features.

In this manual, you will learn how to:

- Install Python (required to run application!)
- Login using your employee ID and password
- Navigate the user interface
- Search employees in the database
- Edit employee information
- Add/deactivate employees
- Issue payroll information

Note: Both adding/deactivating employees and issuing payroll information requires administrator privileges.

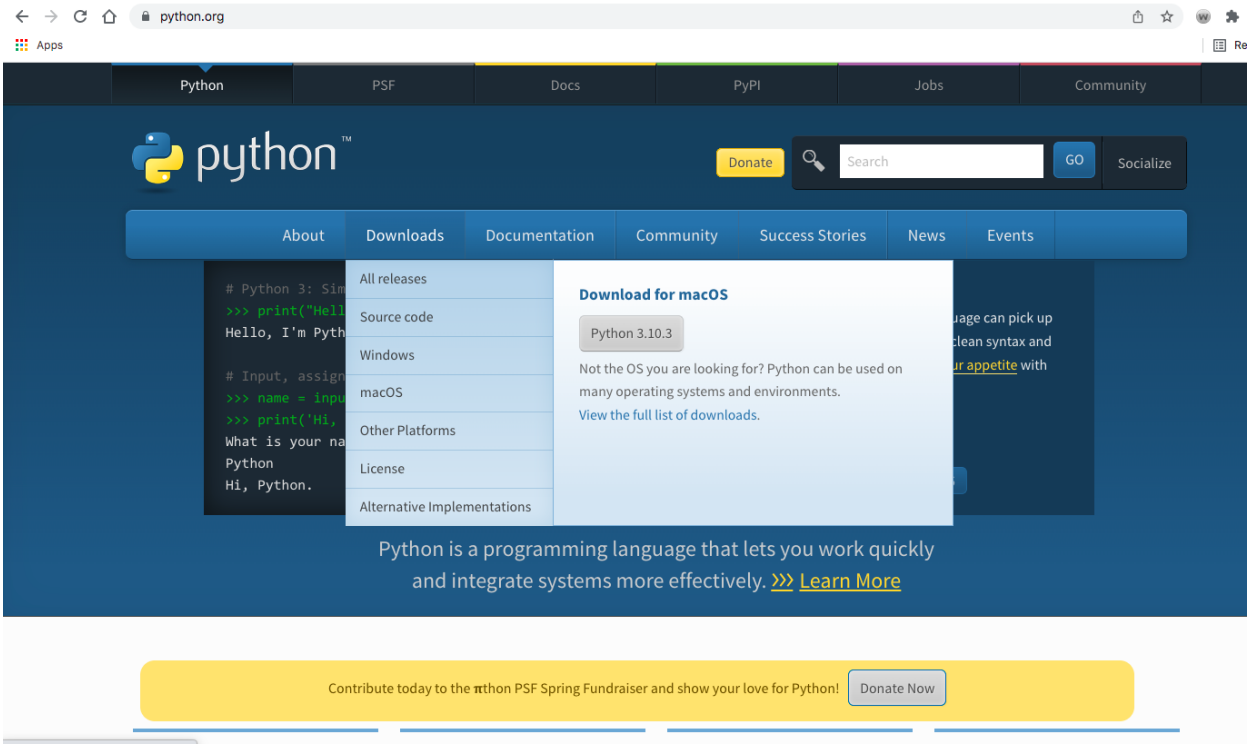
Relevant sections of this manual can be accessed by selecting the '**Help**' button within the GUI.

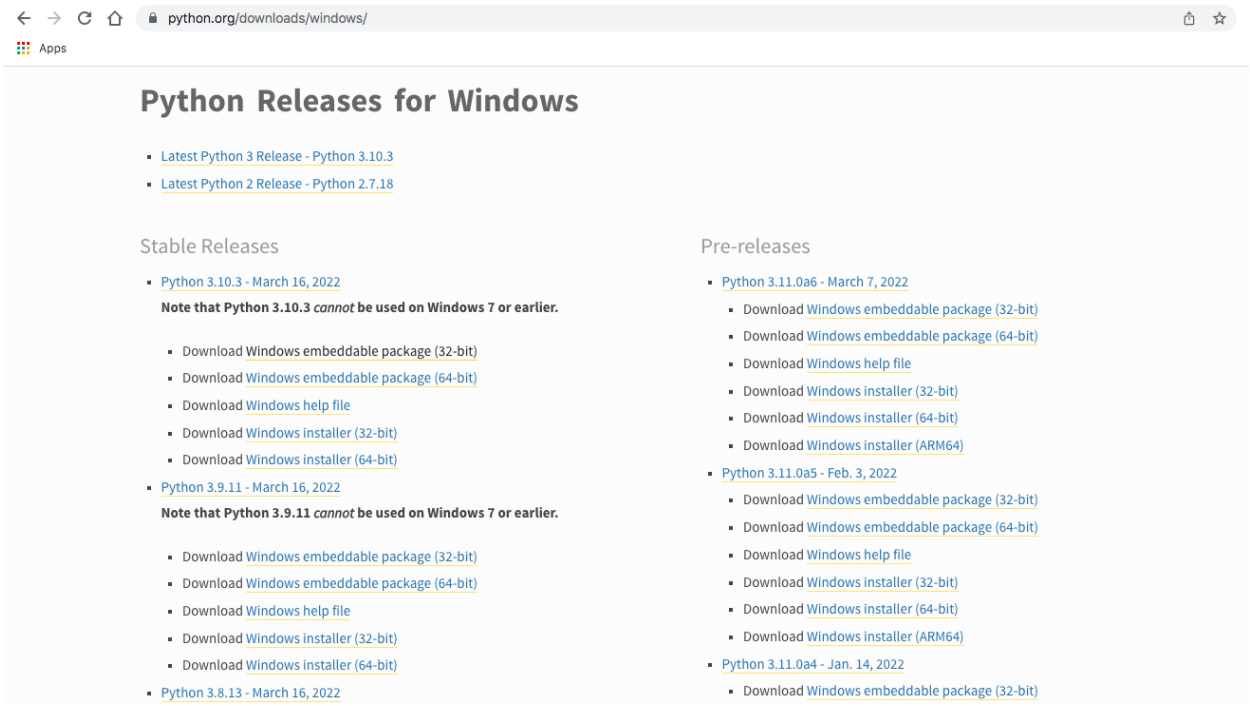
1: Installing Python

The first step is to install Python onto your computer. If you already have Python installed, then you may skip this step. To install Python, go to the following website:

https://www.python.org/.

Once you are on this page, click on the "Downloads" link and you will be taken to a page containing all the different Python versions to install. Download the latest version of Python and select the Windows 10 option.

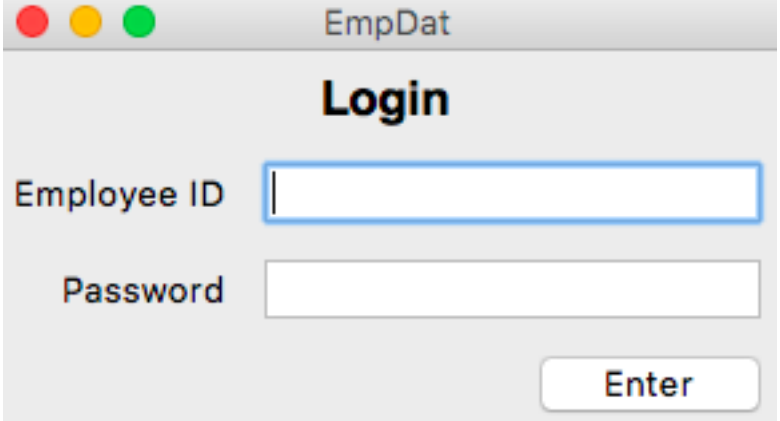




2: User Login

Once you have successfully installed Python and have started the application, you will be taken to the user login page. Here, you will enter your employee ID number and password into the provided fields. Next, click on the '**Enter**' button, or press the enter key while tabbed

into one of the fields and the application will search for your information. If either the ID number or password is incorrect, a red error message will appear to notify you that your information is invalid. The Id and Password fields will clear themselves so you may try and enter your information again.

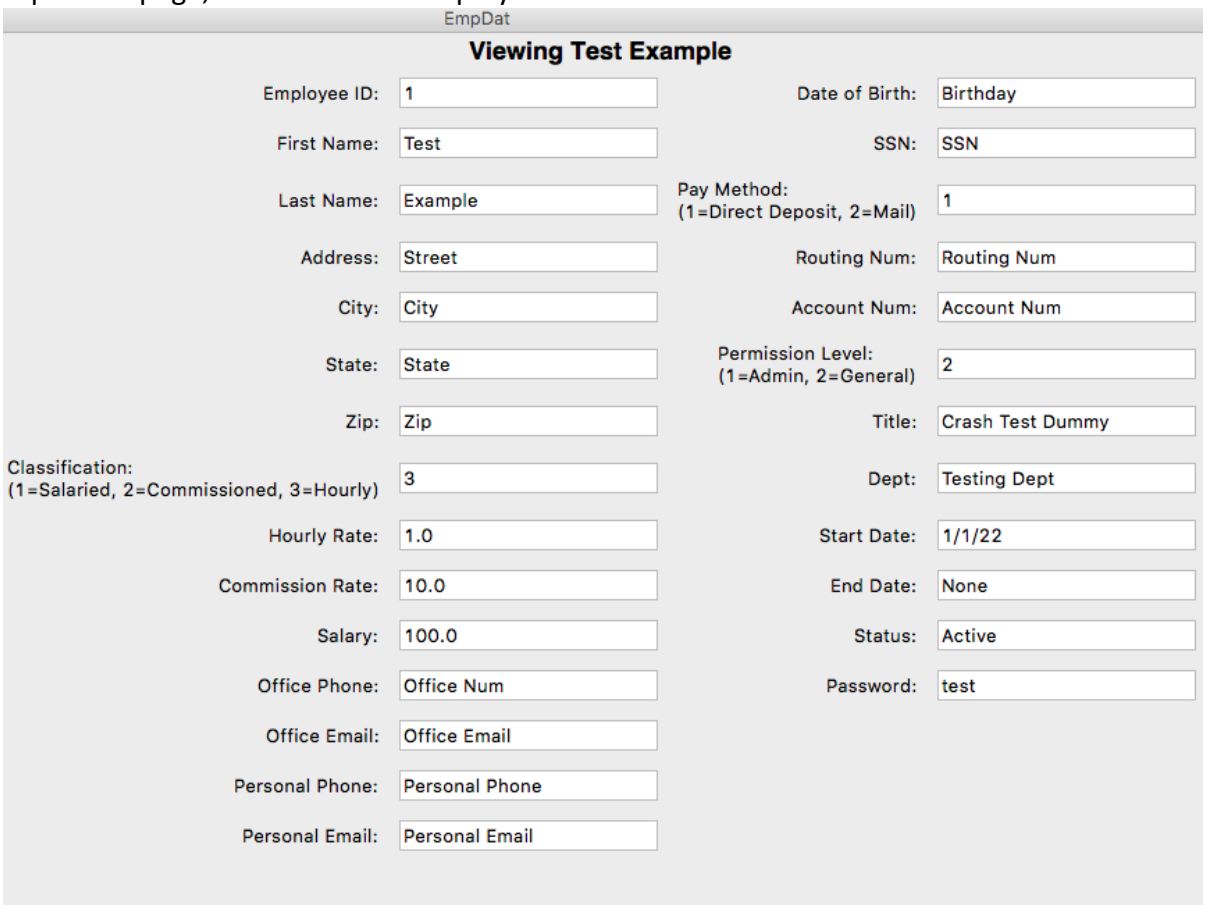


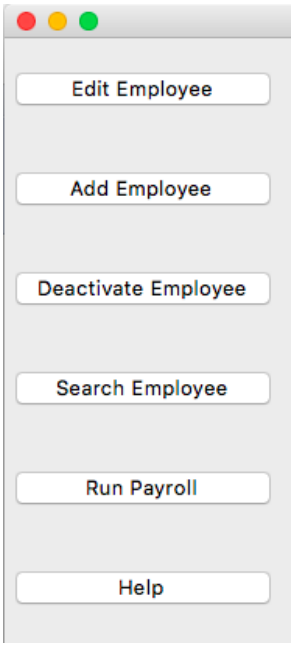
3: Navigating The User Interface

3.1 Admin View Page

Once you have successfully logged in, you will be brought to the view page, viewing your own information. The view page is the center of this application. Here, with admin privileges, you will be able to view all the information of any specified employee, as well as view all of the actions available to you as an admin-privileged user via the menu of buttons on the left side of the page.

On the view page, the specified employee you are viewing is shown in **bold font** at the top of the page, with all of that employee's information visible below.





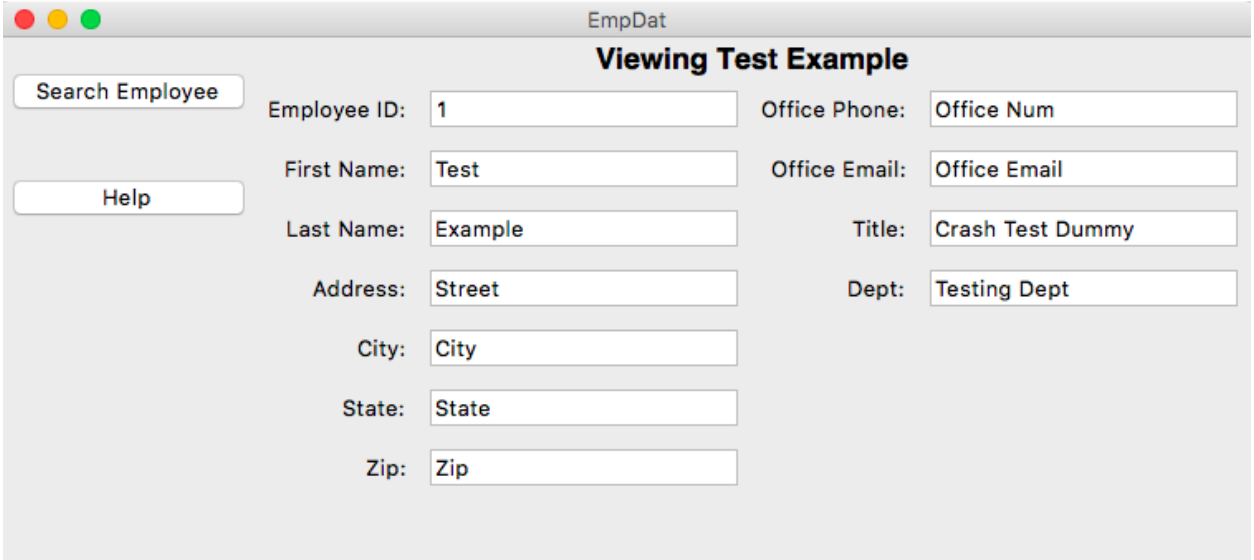
Please note that if any information is missing for the employee you are viewing, a red error message will be shown at the bottom of the page. For this application, some fields can be marked as 'None', which does not count as missing information, but rather as a placeholder for information you may wish to add later.

On the left side of the page, you will find a menu of buttons detailing all available actions. The '**Edit Employee**' button will allow you to edit the currently viewed employee. The '**Add Employee**' button will allow you to fill out the information for and create a new employee. The '**Deactivate Employee**' button will deactivate the currently viewed employee. A popup message will ask for confirmation of this action to prevent you from accidentally firing someone! The '**Search Employee**' button will take you to the search page where you can search all the employees in the database and is how you navigate from profile to profile. The '**Run Payroll**' button will take you to the payroll page where

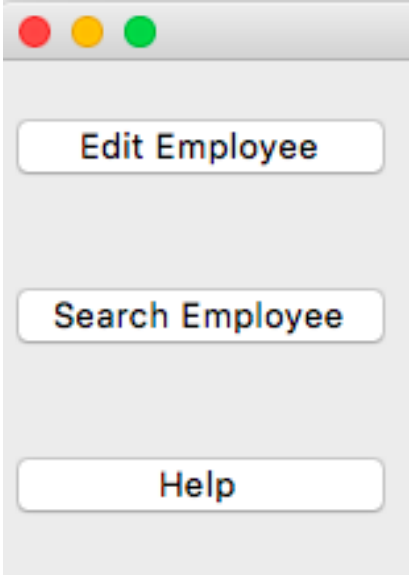
you may import your receipt and timecard files as well as output the pay report. Lastly, the '**Help**' button is what brought you here!

3.2 General View Page

Once you have successfully logged in, you will be brought to the view page, viewing your own information. The view page is the center of this application. Here, with general privileges, you will be able to view all of your own information if viewing yourself, and the business-relevant information of other specified employees, as well as all the actions available to you as a general-privileged user via the menu of buttons on the left side of the page. On the view page, the specified employee you are viewing is shown in **bold font** at the top of the page, with all of that employee's relevant information visible below.



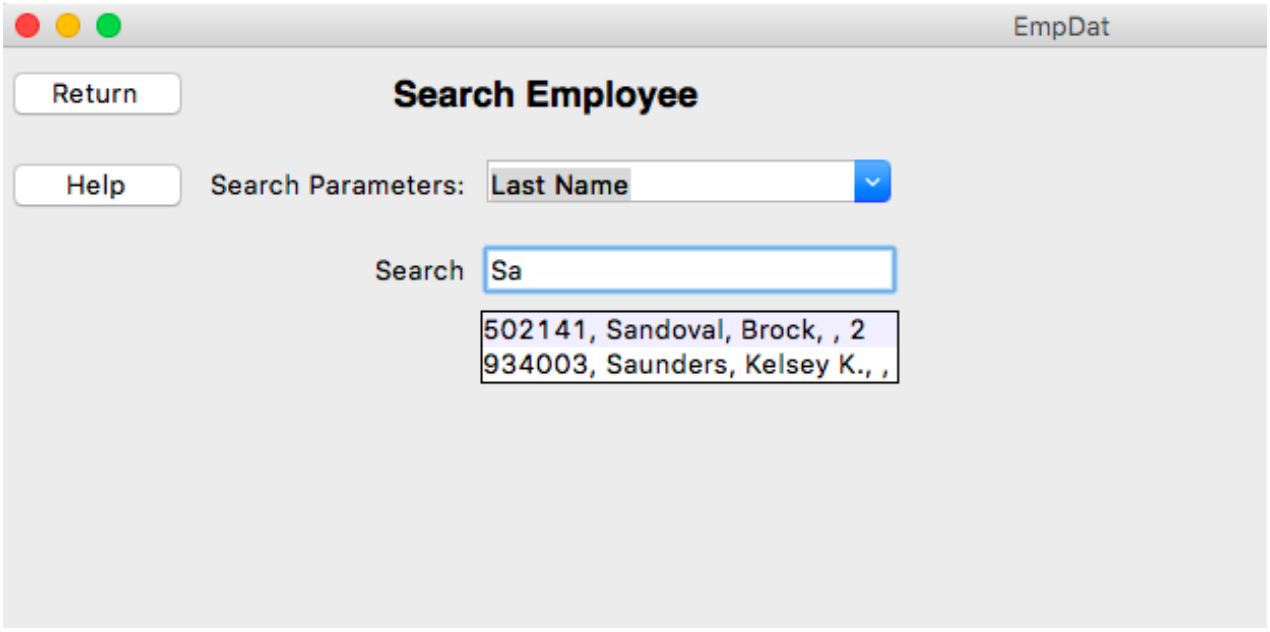
Please note that with general privileges, you will only be able to view all the available information when viewing yourself, otherwise you are restricted to viewing only business-related information.



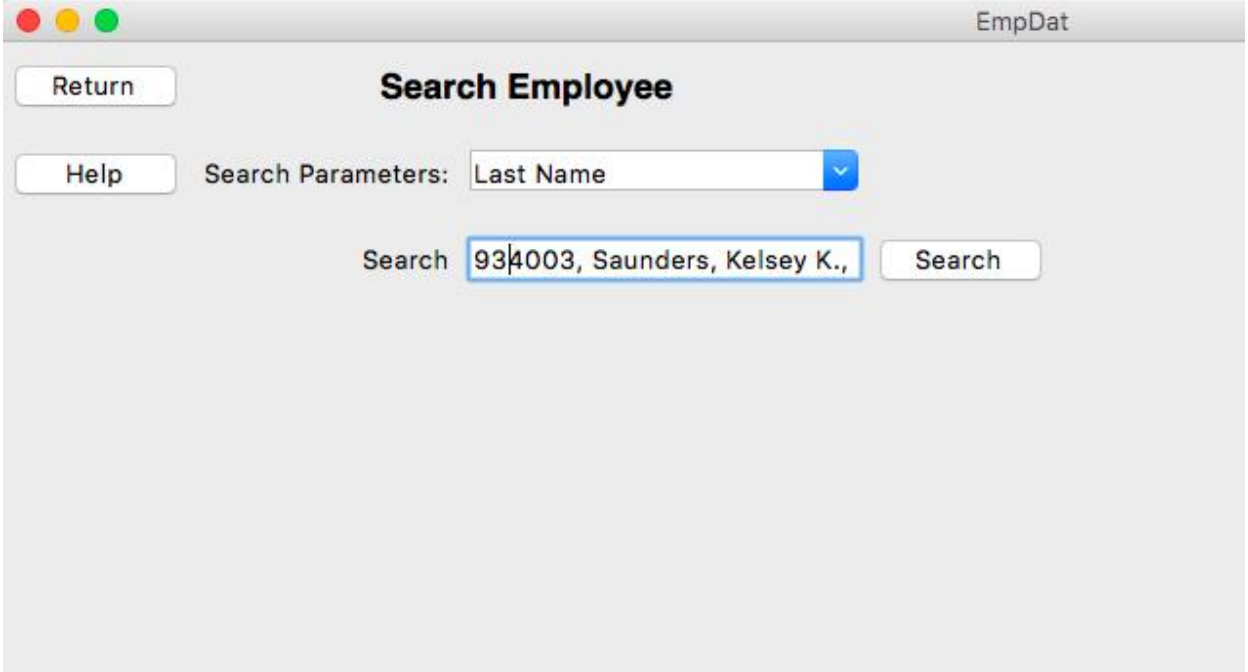
On the left side of the page, you will find a menu of buttons detailing all available actions. The '**Edit Employee**' button will only be shown if you are viewing yourself, and will allow you to edit your information. The '**Search Employee**' button will take you to the search page where you can search all the employees in the database, and is how you navigate from profile to profile. Lastly, the '**Help**' button is what brought you here!

4: Searching Employees

If you wish to view another employee's information, you can select the '**Search Employee**' button on the main view page and a new page will appear. In the main section of this new page, you will see one bar label '**Search**' and another bar with a dropdown list labeled '**Search Parameters**'. You *must* select one of these search parameters before attempting to search the database, as noted by a red message below the search bar. The currently available search parameters are *Last Name*, *Id*, *Classification*, and *Title*. After selecting a search parameter, when you first begin typing in the search bar a list will appear beneath it filled with all the employees in the database that match what you have typed. As you continue to type, the list below will automatically filter out employees that no longer match.



You may notice that there is no search button to go along with the search bar. For this application, a search button *will not* appear until you select an employee from the list below the search bar. This prevents any pointless searches for employees not in the database. If you do not see the employee you are looking for in the list, then they are not in the database or you have typed something wrong. Otherwise, once you see the employee you are searching for, click their name in the list and their information will automatically fill the search bar and the '**Search**' button will appear. Then, you may select the '**Search**' button or simply hit the enter key while tabbed in the search bar to be taken to the view page for the employee you searched. Please note that in order for the list below the search bar to filter properly, you must enter values appropriate for the search parameter you have chosen. If you have chosen to search by Id or Classification for example, you must enter numbers, otherwise you must enter characters.

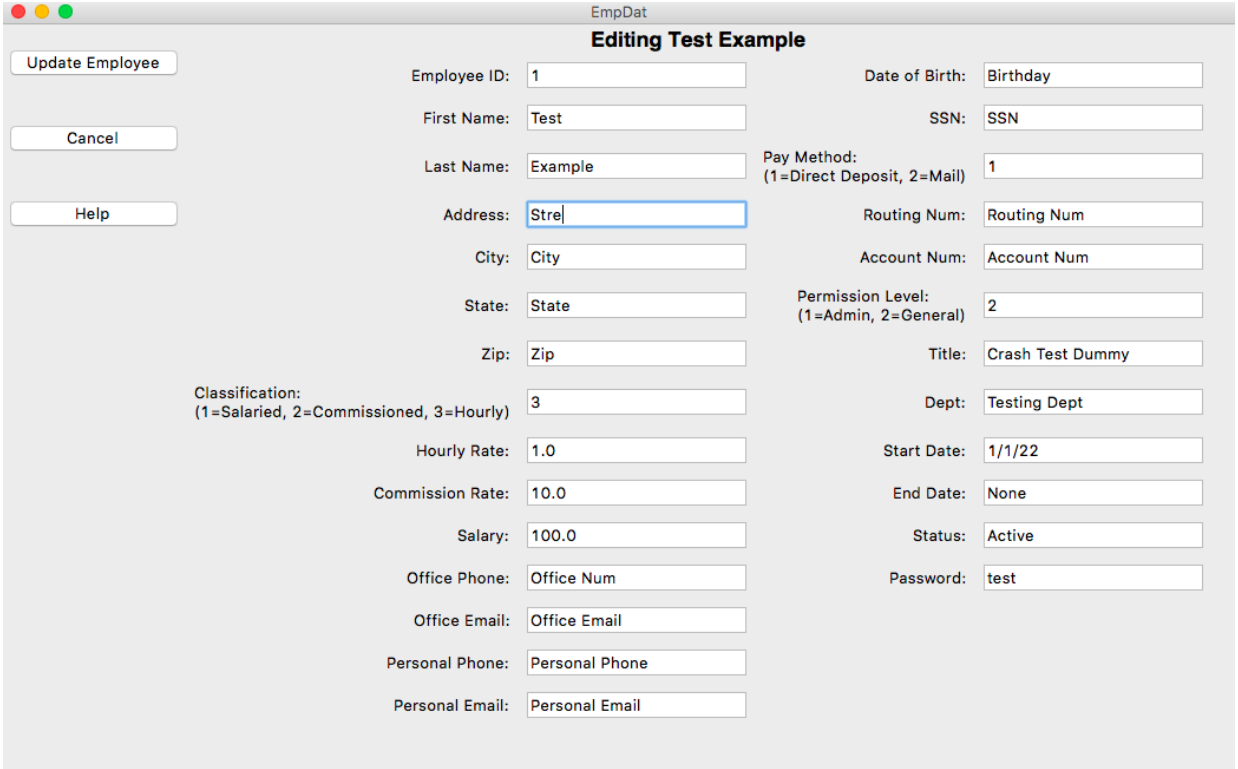


If you wish to leave the search page, you may click the '**Return**' button on the left-hand side of the page to be taken back to the view page you were at prior to visiting the search page.

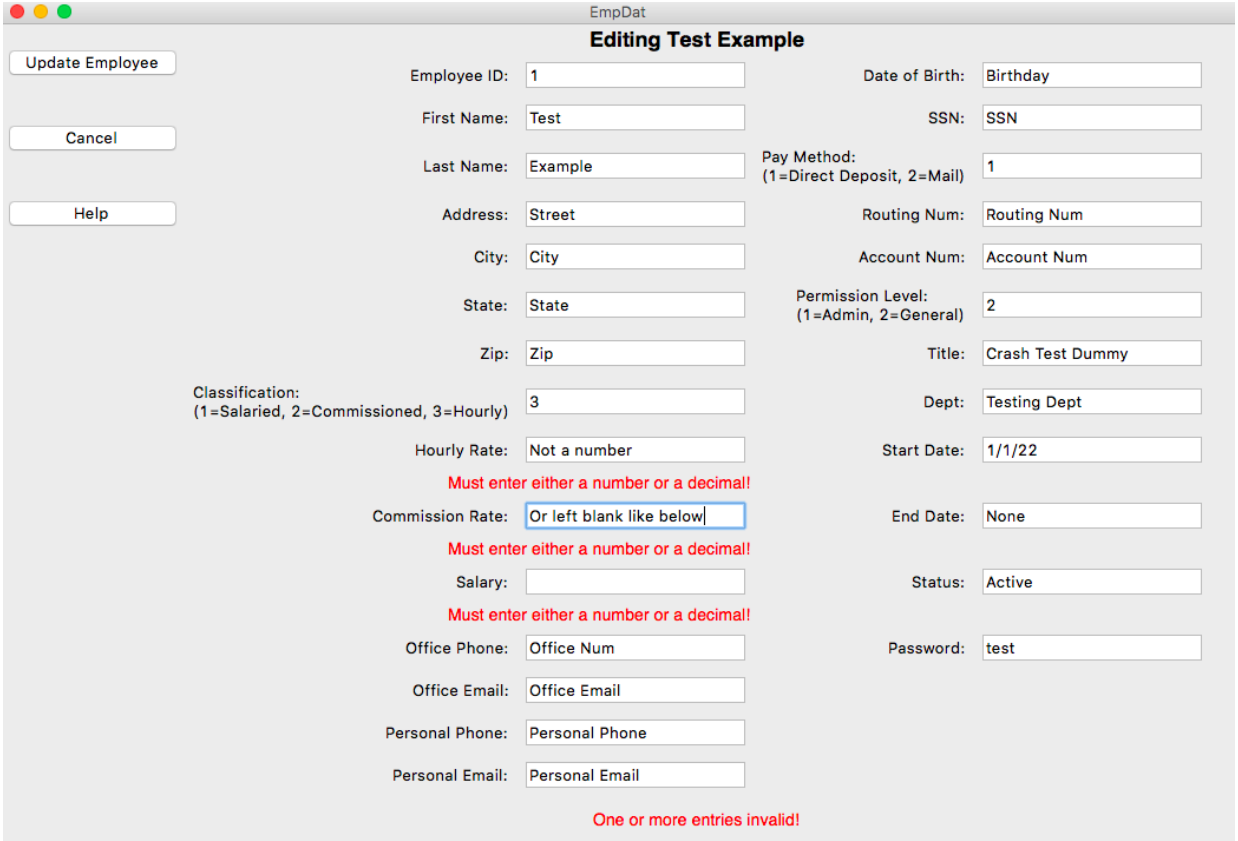
5: Edit An Employee

If you wish to change your employee data, you can select the '**Edit Employee'** button located on the view page to be brought to the edit page. The edit page looks very similar to the view page, though now the title will declare that you are editing rather than viewing the specified employee. On the edit page, you are able to edit the specified employee's data fields by selecting the field you wish to change. Take note however, that you will not be able to change every field. As a general-privileged user editing your own information, *you will only be able to edit personal information*, and not anything related to the company. As an admin-

privileged user, you will have more capability but are still not able to change the **Employee Id**, **Start Date**, **End Date**, or **Status** fields, as those are controlled directly by the application.



Once you have made your desired changes, you can select the '**Update Employee**' button located on the left-hand side of the page, or simply press the enter key while tabbed in to one the data fields and you will be returned to the main view page for the specified employee where you may view the changes you have made. However, if there are any potentially problematic changes you have made like creating an empty field or filling a field with incompatible data, the page will not change and a red error message will appear informing you of what is wrong so that you may correct it and try again.

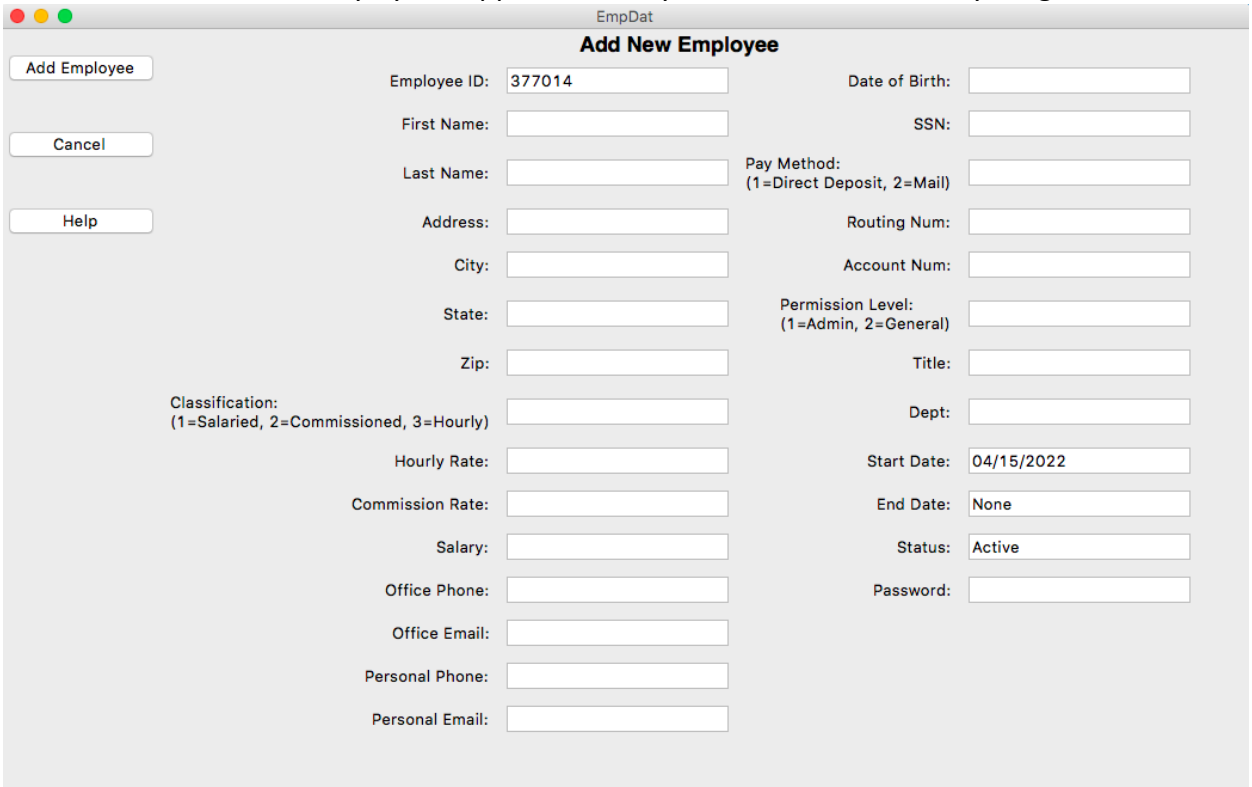


If you decide to cancel your changes, you may select '**Cancel**', which is located below the '**Update Employee**' button, and you will be returned to the main view page for the specified employee.

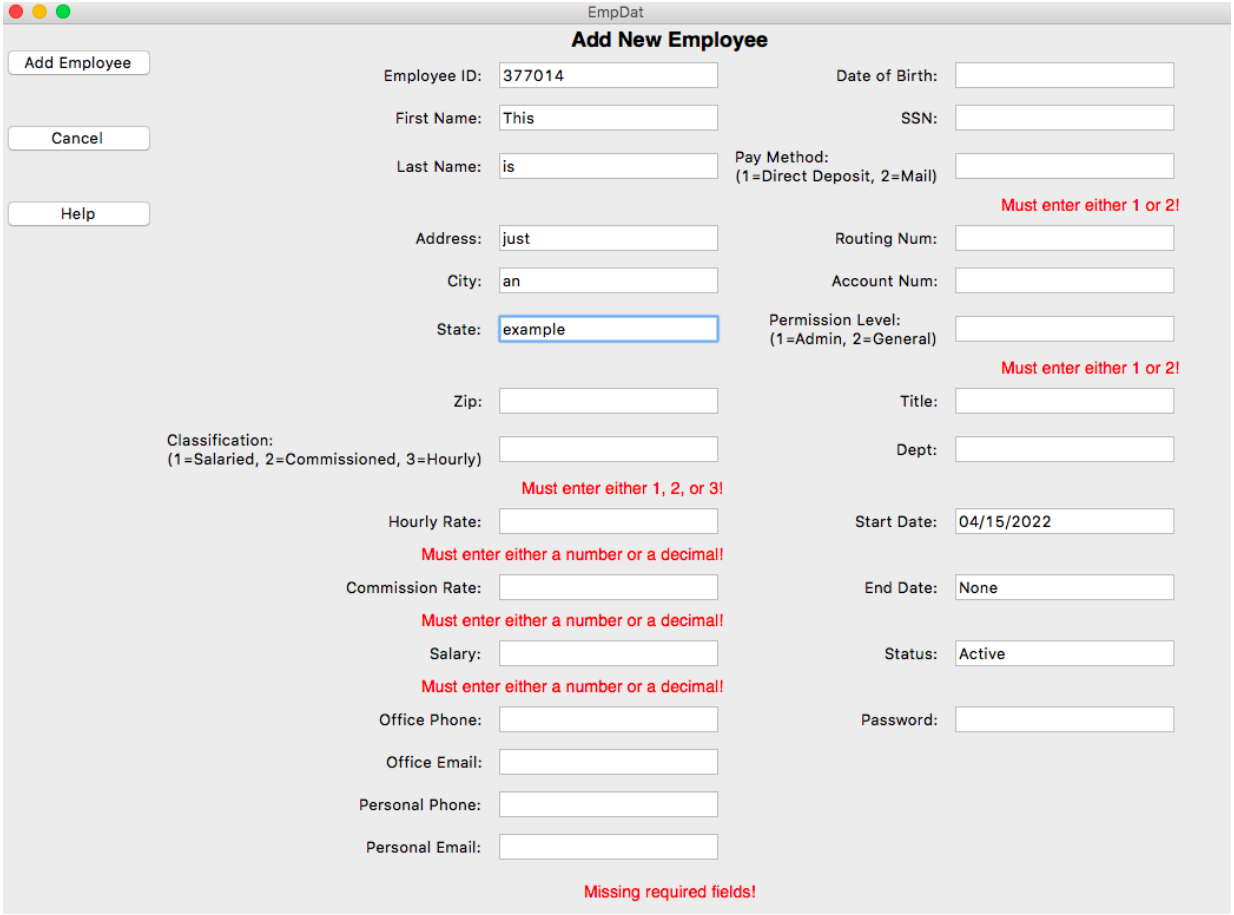
6: Add An Employee

If you wish to add a new employee to the database, you can select the '**Add Employee'** button located on the view page to be brought to the add page. The edit page looks very similar to the view page, though now the title will declare that you are adding an employee and nearly all the employee data fields will be empty. On the add page, you are able to enter a new employee's data by selecting the field you wish to fill out. Take note however, that some fields are already filled, and unable to be changed. The **Employee Id**, **Start Date**, **End Date**, and **Status**

fields are controlled directly by the application, so you don't need to do anything with them.



Once you have filled out all the data fields, you can select the '**Add Employee**' button located on the left-hand side of the page, or simply press the enter key while tabbed in to one of the data fields, and you will be brought to the main view page viewing the employee you just added to the database. However, if there are any issues with the data you entered, the page will not change and a red error message will appear informing you of what is wrong so that you may correct it and try again.

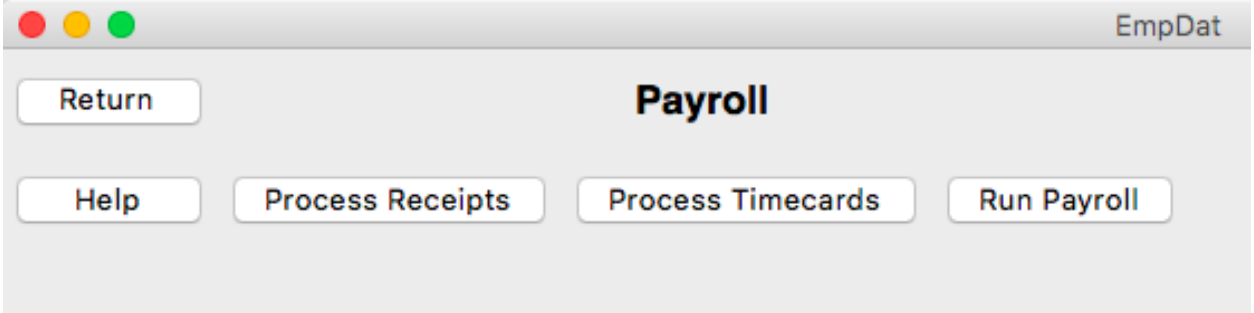


Please note that this application requires you to input something for *every* data field before a new employee can be created. If you do not have all the necessary information to fill out every field, simply enter '**None**'. In the case of fields that require specific inputs like **Classification** (the inputs for which will be declared in the label next to the data field), you may simply pick a random input from those declared and change it later via the edit page if needed. For the **Hourly Rate**, **Salary**, and **Commission Rate** fields, simply enter a **0** if you do not have data for any of those fields.

If you decided to cancel your changes, you may select '**Cancel**', which is located below the '**Add Employee**' button, and you will be returned to the main view page of the employee you were viewing when you first selected the '**Add Employee**' button.

7: Running Payroll

To access the payroll page, click the '**Run Payroll**' button on the view page. On the payroll page, you are presented with three options:



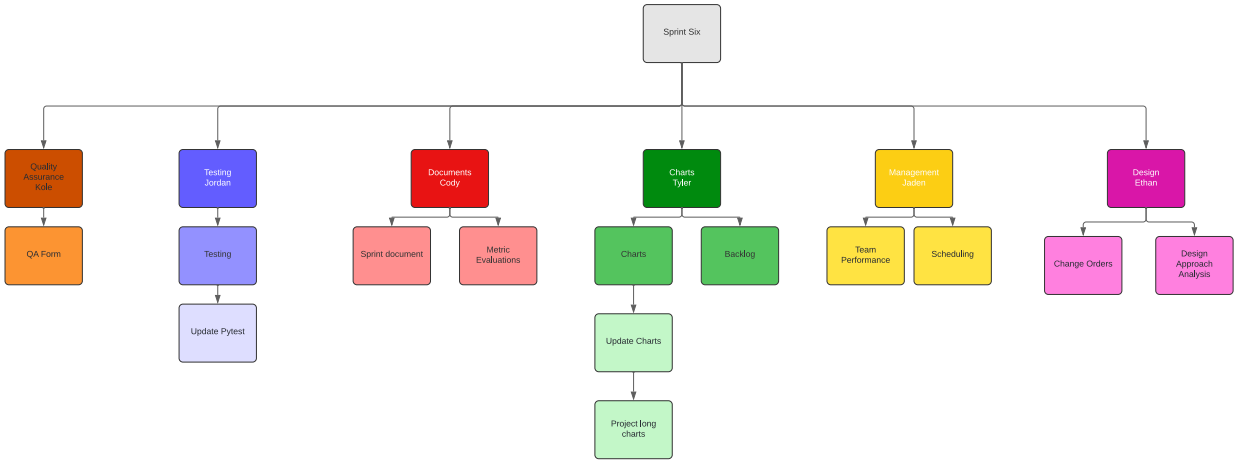
The '**Process Receipts**' button will import the *receipts.csv* file from the *csv* folder in the application and will add the receipt data contained within to the appropriate employees. The '**Process Timecards**' does something similar, importing the *timecards.csv* file and adding the timecard data to the appropriate employees. If either of these is unable to be accessed by the application (whether they don't exist or were simply put into the wrong file), a red error message will appear informing you that the specified file was not found. The '**Run Payroll**' will take all the payment information currently available and output a pay report, which can be located in the *output* folder of the application. This will also reset all the employee's payment information so they are ready for the next pay period. Upon completion of their function, each of these buttons will create a small popup notice confirming their success.

If you are finished with the payroll page, you may click the '**Return**' button on the left-hand side of the page to be returned to the view page you were at prior to visiting the payroll page.

Charts/Forms

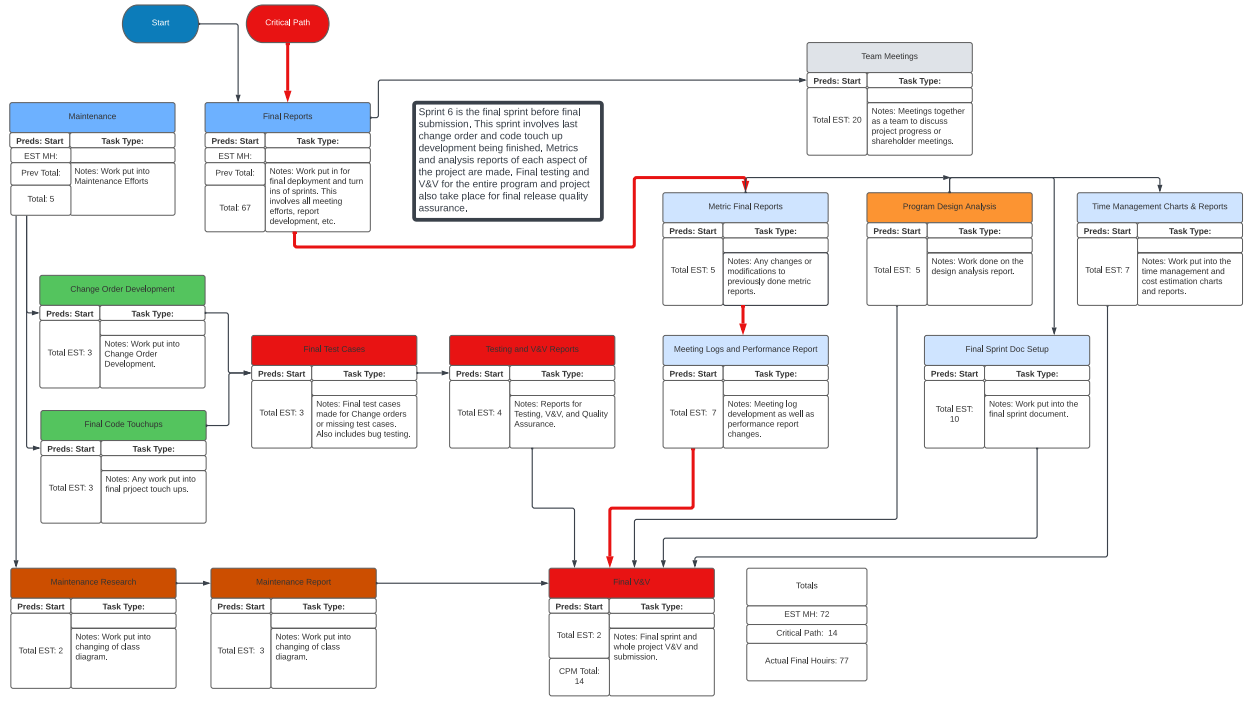
SPR-6 Work Breakdown Structure

Chart



SPR-6 Pert Chart

Chart



SPR-6 Gantt Chart

Chart

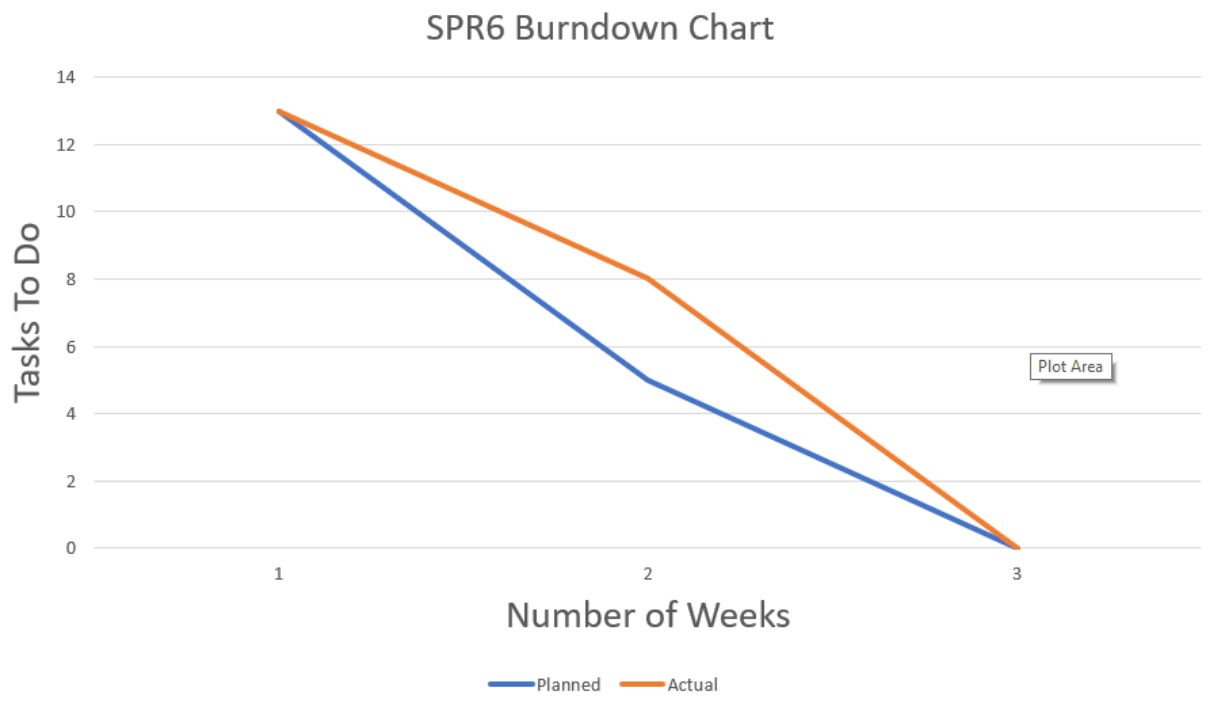
Spr6 - Gantt Chart

Category	Task	ACT	EST	Week 14						Week 15						Week 16					
				4/18	4/19	4/20	4/21	4/22	4/23	4/25	4/26	4/27	4/28	4/29	4/30	5/2	5/3	5/4	5/5	5/6	5/7
SBMT	Submit SPR5	0	-	SBMT																	
MEET	(1)Team Meetings	10	10			2.5		3.5		4											
GUI	Design/Approach Analysis	3	2						2			1									
MTNC	Maintenance Research & Plan	3	5						1	2											
DEV	Change Order Development	0	3																		
DEV	Last Program Touch-ups	0	3																		
TEST	Quality Control / V&V Doc	1	3									0.5		0.5							
TEST	User Acceptance Analysis	1	2											1							
DOC	Meeting Logs	5	7										2		3						
DOC	Cost Estimation (Gantt and Pert Charts)	7	5											3	3	1					
DOC	Sprint 6 Final Project Doc	11	10								1.5		1.5	8							
DOC	Back Logs and Report	3.5	2										1.5	1	1						
DOC	Other Metric Reports	5	5									4	1								
DOC	Team Performance	3	1												1	2					
TEST	Finish Bug Tests & Reports	2	2												2						
MEET	(2)Team Meetings	20.5	10								7.5	5		3		5					
TEST	Final Sprint V&V	2	2													2					
SBMT	Submit Final Sprint	0	-													SBMT					
		0	0																		
		0	0																		
		0	0																		
		0	0																		
		0	0																		
		0	0																		
	Total Allotted	77	72	0	0	2.5	0	3.5	3	6	9	10.5	6	16.5	10	10	0	0	0	0	0
									9						58						10
				0	0	2.5	2.5	6	9	15	24	34.5	40.5	57	67	77	77	77	77	77	77

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-6 Burndown Chart

Chart



Meeting Logs

Meeting Log#26

Meeting Information

- Team #: T2-002
- Meeting log #: 26
- Current Sprint: SPR6
- Date: April 20, 2022
- Time: 3:45pm – 4:15pm (MT)
- Location: In-class meeting
- Attendees: Ethan Taylor, Jaden Albrecht, Tyler Deschamps, Cody Strange, Kole Davis
- Next team meeting scheduled for: April 22, 2022, 7:00pm (MT)

Progress From Previous Meeting

- No progress

Topics Discussed

- SPR-6 requirements
- Next team meeting

Obstacles Encountered

- No obstacles

Finished Items

- No finished items

Unfinished Items

- SPR-6 document
- Test cases for payroll page

Tasks Until Next Meeting

- Read chapter 11

Notes

- We didn't discuss much during this meeting. We each needed more time to go over the sprint requirements before assigning tasks for this sprint.

Meeting Log#27

Meeting Information

- Team #: T2-002
- Meeting log #: 27
- Current Sprint: SPR6
- Date: April 22, 2022
- Time: 7:00pm – 7:47pm (MT)
- Location: MS Teams (watch video here)
- Attendees: Ethan Taylor, Jaden Albrecht, Tyler Deschamps, Cody Strange, Kole Davis
- Next team meeting scheduled for: April 25, 2022, 7:00pm (MT)

Progress From Previous Meeting

- Each team member has read and reviewed all SPR-6 requirements
- Each team member has read chapter 11

Topics Discussed

- Task assignment
- Maintenance analysis
- Backlog analysis
- Team performance analysis
- Design analysis
- Project V&V
- Bug test analysis
- QA analysis
- Final Pert/Gantt/Burn-down charts
- Appendix for SPR-6 document

Obstacles Encountered

- No obstacles

Finished Items

- Read chapter 11
- Read and review all SPR-6 requirements

Unfinished Items

- SPR-6 document
- Maintenance analysis
- Backlog analysis
- Team performance analysis
- Design analysis
- Project V&V
- Bug test analysis
- QA analysis
- Final Pert/Gantt/Burn-down charts
- Appendix for SPR-6 document

- Test cases for payroll page

Tasks Until Next Meeting

- Maintenance analysis
- Backlog analysis
- Team performance analysis
- Design analysis
- Project V&V
- Bug test analysis
- QA analysis
- Final Pert/Gantt/Burn-down charts
- Appendix for SPR-6 document

Notes

- No additional notes

Meeting Log#28

Meeting Information

- Team #: T2-002
- Meeting log #: 28
- Current Sprint: SPR6
- Date: April 25, 2022
- Time: 7:00pm – 8:09pm (MT)
- Location: MS Teams (watch video here)
- Attendees: Ethan Taylor, Jaden Albrecht, Tyler Deschamps, Cody Strange, Kole Davis
- Next team meeting scheduled for: April 27, 2022, 3:45pm (MT)

Progress From Previous Meeting

- Ethan Taylor:
    - Design analysis (80%)

Topics Discussed

- Scheduled meeting with professor to discuss SPR-5 final grade

Obstacles Encountered

- No obstacles

Finished Items

- No finished items

Unfinished Items

- SPR-6 document
- Maintenance analysis
- Backlog analysis
- Team performance analysis
- Design analysis

- Project V&V
- Bug test analysis
- QA analysis
- Final Pert/Gantt/Burn-down charts
- Appendix for SPR-6 document
- Test cases for payroll page

Tasks Until Next Meeting

- Maintenance analysis
- Backlog analysis
- Team performance analysis
- Design analysis
- Project V&V
- Bug test analysis
- QA analysis
- Final Pert/Gantt/Burn-down charts
- Appendix for SPR-6 document

Notes

- We spent this meeting talking mostly about our meeting with our professor to go over our final grades for SPR-5.

Meeting Log#29

Meeting Information

- Team #: T2-002
- Meeting log #: 29
- Current Sprint: SPR6
- Date: April 27, 2022
- Time: 3:45pm – 4:34pm (MT)
- Location: MS Teams (watch video here)
- Attendees: Ethan Taylor, Jaden Albrecht, Tyler Deschamps, Cody Strange, Jordan Van Patten, Kole Davis
- Next team meeting scheduled for: April 29, 2022, 7:00pm (MT)

Progress From Previous Meeting

- Ethan Taylor:
    - Design analysis (100%)
- Cody Strange:
    - SPR-6 document (80%)
    - Project management and reports (85%)
- Tyler Deschamps:
    - Maintenance plan (100%)

    - Backlogs for previous sprints (100%)

Topics Discussed

- SPR-6 V&V/Quality control analysis
- User acceptance evaluation
- Update bug tests
- Team performance

Obstacles Encountered

- No obstacles

Finished Items

- Design analysis
- Maintenance plan
- Backlogs from previous sprints
- Scheduled meeting with shareholder to go over SPR-5 final grade

Unfinished Items

- SPR-6 document
- Team performance analysis
- Project V&V
- Bug test analysis
- QA analysis
- Final Pert/Gantt/Burn-down charts
- Appendix for SPR-6 document

Tasks Until Next Meeting

- Team performance analysis
- Project V&V
- Bug test analysis
- QA analysis
- Final Pert/Gantt/Burn-down charts
- Appendix for SPR-6 document

Notes

- No additional notes

Meeting Log#30

Meeting Information

- Team #: T2-002
- Meeting log #: 30
- Current Sprint: SPR6
- Date: May 2, 2022
- Time: 12:00pm – 12:30pm (MT)
- Location: MS Teams (watch video)

- Attendees: Ethan Taylor, Jaden Albrecht, Tyler Deschamps, Cody Strange, Jordan Van Patten, Kole Davis
- Next team meeting scheduled for: No meetings after this one

Progress From Previous Meeting

- Ethan Taylor:
    - Quality control analysis (100%)
- Jaden Albrecht:
    - Team performance evaluations (100%)
    - Final SPR-6 document introduction (100%)
    - Help with final SPR-6 document (100%)
- Cody Strange:
    - SPR-6 document (100%)
    - Project management and reports (100%
- Tyler Deschamps:
    - Decomposed Gantt chart Report (100%)
    - Backlog changes report (100%)
    - Pert chart report (100%)
    - Final burn-down chart report (100%)
- Jordan Van Patten:
    - Bug testing reports (100%)
- Kole Davis:
    - V&V documents (100%

Topics Discussed

- Final SPR-6 document revisions
- Team performance evaluations

Obstacles Encountered

- No obstacles

Finished Items

- Final SPR-6 document
- Bug test reports
- Final Pert/Gantt/burn-down charts and reports
- Quality control analysis
- Team performance evaluations

Unfinished Items

- None

Tasks Until Next Meeting

- None

Notes

- No additional notes

SPRINT FIVE

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*
Kole Davis- *QA Manager*
Craig Sharp-*Stakeholder*

Table of contents

Management

Procedures

Verification & Validation

Summary: V&V documents for sprint five specifically.

APPLICATION QUALITY ASSURANCE CHECKLIST

Project Name	EmpDat	**Project #**	SPR-5
Application Name		**Application #**	
Project Manager		**Delivery Manager**	
Date Completed	4/15/2022		

IMPORTANT NOTES FOR COMPLETING THIS DOCUMENT

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?	☒	☐	☐	

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

1. Decoupling Business Logic From The Presentation Layer				
Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Is the presentation layer of the application free from business logic?	☒	☐	☐	
1. Has the presentation layer of the application been developed in accordance with prevailing industry standards?	☒	☐	☐	

1. Record Locking / Concurrent Users				
Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Have precautions been taken to avoid data clashes?	☒	☐	☐	

1. Passwords				
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
1. 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?	☐	☒	☐	
1. Does the system meet the standard password requirements?	☐	☒	☐	Password requirements will be added in at a later execution of our program
1. 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
1. Are all vendor supplied default passwords revised prior to placing the application in a production environment?	☐	☒	☐	
1. 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

1. **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?	☐	☐	☒	
1. Based on the application's Information Security Classification, does the application meet the auditing functional control requirements?	☐	☐	☒	

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

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

1. Code Comments				
Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Does all application code include sufficient comments for support personnel?	☒	☐	☐	

1. Code Comments				
Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Does each code unit have its own brief and accurate description?	☒	☐	☐	

1. Error Handling – End User				
Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Does the application handle all the errors that could reasonably be expected to occur?	☒	☐	☐	
1. Do the error messages contain minimal but meaningful information?	☒	☐	☐	
1. Does the application avoid displaying system information in error messages?	☒	☐	☐	
1. Are the error messages kept in a single location?	☐	☒	☐	Error messages are kept in separate locations in the code, and in the application it is shown below the effected area, I.E. if date is incorrect, it will display the error message below the date entry box

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

1. Field Validations				
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)?	☒	☐	☐	
1. Are field values being validated with regular expressions where possible (e.g. validating email addresses and dates for valid formats)?	☒	☐	☐	Date of birth is not currently being validated
1. Do the validations resulting in error messages prevent data from being written to persistent storage (databases, files, etc.)?	☒	☐	☐	
1. Are the validations being performed within the business logic, as well as on the presentation layer?	☒	☐	☐	
1. Have the validations been written so that users cannot bypass them?	☒	☐	☐	
1. Are all of the field lengths and types within the application consistent with the column lengths and types declared within the underlying database tables?	☒	☐	☐	

1. Field Validations				
Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Are user inputs being sanitized (without exceptions) according to OWASP recommendations?	☒	☐	☐	

1. Dates				
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?	☐	☒	☐	Dates are automatically inputted by the code, not by the user, so there is no date validation. DOB is not yet validated and mandated to be in the month/day/year format but will be added in a future version
1. Do all relevant dates include a timestamp?	☐	☒	☐	

1. Hard-Coded Values				
Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Does the application code avoid use of hard-coded values?	☒	☐	☐	
1. Do all hard-coded values reside exclusively within configuration and constant, centralized locations? (Central Locations that enable changes without recompiling source code)	☒	☐	☐	

1. System Testing			
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
1. Did the application pass all negative tests?	☒	☐	
1. Have client testers completed the formal test plan in its entirety?	☐	☒	We have not made a formal test plan yet
1. Did the application pass all tests included in the formal test plan?	☐	☒	
1. Have all positive / negative test cases and test case results been documented?	☒	☐	

1. Regression Testing				
Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. As new capability is introduced, is the new capability tested?	☒	☐	☐	
1. Have all previous tests been reconducted with the results compared against expected results?	☒	☐	☐	
1. 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?	☒	☐	☐	
1. 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?	☒	☐	☐	

1. Load Testing/Volume Testing				
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)?	☐	☒	☐	
1. Has the application been tested with large numbers of concurrent transactions (i.e. a number of transactions that is representative of peak system usage)?	☐	☒	☐	
1. Did the system perform well with a large number of concurrent users?	☐	☐	☒	
1. Did the system perform well with a large number of concurrent transactions?	☐	☐	☒	
1. Are end-users satisfied with the application's performance and responsiveness during everyday use?	☐	☐	☒	

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

1. Business Requirements - Traceability				
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
1. 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

1. 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
1. Has a test build been completed by Application Services using the code that has been handed over?	☐	☐	☒	
1. 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)	☒	☐	☐	
1. 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)	☐	☐	☒	

1. Database Design				
Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Have the database tables been normalized?	☒	☐	☐	
1. Keys based on sequence numbers have unique sequences.	☒	☐	☐	

1. Database Design				
Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Are all keys and required fields set to 'not null' in all tables of the database?	☒	☐	☐	
1. Have triggers, stored procedures, sequences, and constraints been properly utilized?	☐	☐	☒	

1. Transition To Support Personnel				
Validation Questions	**Yes**	**No**	**NA**	**Comments**
1. Have accounts been created on all servers for the appropriate support personnel?	☐	☒	☐	
1. Have the necessary firewall rules been added to allow Application Services support personnel to access the relevant servers (i.e. via the Jump Box)?	☐	☒	☐	
1. Have all server environments (development, test/staging, and production) been fully created?	☐	☒	☐	
1. Are all of the server environments entirely consistent with each other?	☐	☒	☐	

Backlog

Summary: The sprint five backlog covers the tasks in the sprint and the team member that is assigned to the task, along with a due date attached to that task. It also has the tasks from the previous sprint that were not completed in that sprint.

~ SPR5 Task List		
Task	**Assigned To**	**Due Date**
Create Backlogs with Timeboxes - Includes Task Assignment	Tyler	04/13/2022
Collect Indicators	Cody	04/06/2022
Create Metric Report(s)	Cody	04/15/2022
Current Sprint Meeting Logs	Jaden	04/18/2022
Change Order Development	Jaden	04/15/2022
Gather Defects Data	Tyler / Ethan	03/30/2022
Defects Report	Kole	04/08/2022
Last touch-ups on program	Jaden	04/15/2022
Current Sprint Work Breakdown Structure	Cody	03/30/2022
Current Sprint Pert, Gantt, and Burndown charts	Tyler	04/18/2022
Gather and Format decomposed Charts	Tyler	04/15/2022
Performance Plan	Cody	04/18/2022
Function Points Chart	Cody	04/15/2022
Ishikawa diagram(s) of a defect (not the easiest nor the hardest to address)	Kole	04/15/2022
User Acceptance	All Team Members	04/06/2022
Testing of Implemented Change Orders	Jordan	04/06/2022
Help sections	Ethan	04/13/2022
Sprint PDF Document	Jaden	04/18/2022

Team Members
Jaden Albrecht
Cody Strange
Ethan Taylor
Jordan Van Patten
Kole Davis
Tyler Deschamps

~ Previous Sprint Backlog		
Task	**Assigned To**	**Due Date**
Finish Test Cases	Jordan	04/01/2022
Updated Class Diagram	Ethan	04/06/2022
Correct CSV Database to Text Database	Ethan	04/15/2022

Team Performance

Summary: This review tracks each team member and the total amount of hours they logged for each sprint. It also covers the amount of meetings they attended per sprint and the tasks that they completed during the sprint.

T2-002 PERFORMANCE REVIEW

SPR-1

Ethan Taylor:

- Total Hours: 9 hours
- Total Meetings Attended: (3/3)
- Tasks Completed:
    - Personal MoSCoW chart for Delphi method (100%)
    - Thoroughly read and reviewed all SPR-1 document requirements (100%)

Jaden Albrecht:

- Total Hours: 9 hours
- Total Meetings Attended: (3/3)
- Tasks Completed:
    - Personal MoSCoW chart for Delphi method (100%)
    - Thoroughly read and reviewed SPR-1 document requirements (100%)

Cody Strange:

- Total Hours: 9 hours
- Total Meetings Attended: (3/3)
- Tasks Completed:
    - Personal MoSCoW chart for Delphi method (100%)
    - Thoroughly read and reviewed SPR-1 document requirements (100%)
    - SPR-1 document (100%)

Tyler Deschamps:

- Total Hours: 9 hours
- Total Meetings Attended: (3/3)
- Tasks Completed:
    - Personal MoSCoW chart for Delphi method (100%)
    - Thoroughly read and reviewed SPR-1 document requirements (100%)

SPR-2

Ethan Taylor:

- Total Hours: 6 hours 45 minutes
- Total Meetings Attended: (4/4)
- Tasks Completed:
    - GUI template (100%)
    - Rough draft for HLD (100%)
    - Review SPR-2 document requirements (100%)
    - Develop GUI classes (100%)

    - Plain-English pseudocode (100%)
    - Develop use-case scenarios based on requirements (100%)
    - Docstring/comment outlining (100%)

Jaden Albrecht:

- Total Hours: 35 minutes
- Total Meetings Attended: (4/4)
- Tasks Completed:
    - Distribution of MoSCoW charts for Delphi method (100%)
    - Review SPR-2 document requirements (100%)
    - Review GUI template and use-cases for HLD (100%)
    - Review plain-English description of HLD (100%)
    - Establish connections between project requirements and tasks (100%)
    - Develop minute-tracking method (100%)
    - Keep meeting logs up to date (100%)

Cody Strange:

- Total Hours: 9 hours 20 minutes
- Total Meetings Attended: (4/4)
- Tasks Completed:
    - Fix SPR-1 document (100%)
    - Work breakdown structure for SPR-2 (100%)
    - Construct requirements spec (100%)
    - Synchronize requirements spec with final MoSCoW chart (100%)
    - Choose SPR-1 prototype candidate (100%)
    - SPR-2 document (100%)

Tyler Deschamps:

- Total Hours: 45 minutes
- Total Meetings Attended: (4/4)
- Tasks Completed:
    - Develop Pert/Gantt/burn-down charts for SPR-2 (100%)
    - Review SPR-2 document requirements (100%)
    - Logging for Pert/Gantt charts throughout SPR-2 (100%)

Jordan Van Patten*:

- Total Hours: 1 hour, 30 minutes
- Total Meetings Attended: (3/4)
- Tasks Completed:
    - Help with GUI class development (100%)
    - Review SPR-2 document requirements (100%)
    - V&V covering requirements and documentation (100%)
    - Review use-cases for HLD (100%)

SPR-3

Ethan Taylor:

- Total Hours: 20 hours, 45 minutes
- Total Meetings Attended: (6/6)
- Tasks Completed:
    - Review all SPR-3 documents (100%)
    - Review prototype code (100%)
    - Pseudocode for Login GUI page (100%)
    - Pseudocode for search page (100%)
    - Pseudocode for view/add/edit page (100%)
    - Login GUI (100%)
    - Search GUI (100%)
    - View/add/edit GUI (100%)
    - Deactivate employee GUI (100%)
    - Payroll GUI (100%)
    - Tkinter research

Jaden Albrecht:

- Total Hours: 19 hours, 40 minutes
- Total Meetings Attended: (6/6)
- Tasks Completed:
    - Review all SPR-3 document requirements (100%
    - Keep meeting logs up to date throughout SPR-3 (100%)
    - SPR-3 change order request form (100%)
    - Develop version history folders in google drive (100%)
    - Build UML class diagrams (100%)
    - Review prototype code (100%))
    - UML research
    - Tkinter research
    - Pytest research

Cody Strange:

- Total Hours: 17 hours, 40 minutes
- Total Meetings Attended: (6/6)
- Tasks Completed:
    - Review all SPR-3 document requirements (100%)
    - Work breakdown structure for SPR-3 (100%)
    - Risk management plan (100%)
    - Testing documents (100%)
    - Field validation code for login page (100%)
    - Review Prototype code (100%)
    - Fix SPR-2 document (100%)
    - Reorganize google drive folders (100%)

        - Change request board (100%)
        - SPR-3 document (100%)

Tyler Deschamps:

- Total Hours: 15 hours, 30 minutes
- Total Meetings Attended: (6/6)
- Tasks Completed:
    - Review all SPR-3 document requirements (100%)
    - Update Pert/Gantt/burn-down charts throughout SPR-3 (100%)
    - Review prototype code
    - Search employee database pseudocode (100%)
    - Database pseudocode (100%)
    - CSV pseudocode (100%)
    - Database tables to CSV (100%)
    - JSON research

Jordan Van Patten:

- Total Hours: 6 hours, 50 minutes
- Total Meetings Attended: (3/6)
- Tasks Completed:
    - Review all SPR-3 document requirements (100%)
    - Review prototype code (100%)
    - Team file/folder naming conventions document (100%)
    - Quality assurance plan document (100%)
    - V&V documents for SPR-3 (100%)
    - Pytest research
    - Black research
    - Debugging software research

SPR-4

Ethan Taylor:

- Total hours: 9 hours, 40 minutes
- Total Meetings Attended: (3/3)
- Tasks Completed:
    - Review all SPR-4 document requirements (100%)
    - Implement add/edit validation code into GUI (100%)
    - Integrate database with payroll page updates (100%)
    - Update payroll page (100%)
    - Tkinter research

Jaden Albrecht:

- Total hours: 5 hours
- Total Meetings Attended: (3/3)
- Tasks Completed:

        - Review all SPR-4 document requirements (100%)
        - Keep meeting logs up to date throughout SPR-4 (100%)
        - User manual (100%)

Cody Strange:

- Total hours: 14 hours
- Total Meetings Attended: (3/3)
- Tasks Completed:
    - Review all SPR-4 document requirements (100%)
    - Work break-down structure for SPR-4 (100%)
    - Usability tests (100%)
    - Risk management plan for SPR-4 (100%)
    - Maintenance plans (100%)
    - Deployment plan (100%)
    - SPR-4 document (100%)

Tyler Deschamps:

- Total hours: 15 hours, 30 minutes
- Total Meetings Attended: (2/3)
- Tasks Completed:
    - Review all SPR-4 document requirements (100%)
    - Update Pert/Gantt/burn-down charts throughout SPR-4 (100%)
    - Report receipts (100%)
    - Implement database logic into GUI (100%)
    - Merge database with Emp_Dat_V3 (100%)
    - Payroll and CSV integration (100%)

Jordan Van Patten:

- Total hours: 6 hours, 30 minutes
- Total Meetings Attended: (3/3)
- Tasks Completed:
    - Review all SPR-4 document requirements (100%)
    - V&V documents for SPR-4 (100%)
    - Test cases for add/edit/view and search GUI pages (100%)
    - Update quality assurance plan (100%)
    - Pytest documents for payroll page (100%)
    - Implement validation code for add/edit/view GUIs (100%)

SPR-5

Ethan Taylor:

- Total Hours: 25 hours 30min
- Total Meetings Attended: (9/11)
- Tasks Completed:
    - Review All SPR-5 document requirements (100%)

    - Defects analysis (100%)
    - Update UML diagrams (100%)
    - Help button/user manual integration (100%)
    - Integrate old database (100%)
    - Integrate CSV database (100%)

Jaden Albrecht:

- Total Hours: 30 hours, 15 minutes
- Total Meetings Attended: (11/11)
- Tasks Completed:
    - Review all SPR-5 document requirements (100%)
    - Functional/non-functional requirements testing (100%)
    - Personal user acceptance test (100%)
    - Add search parameters and improvements to search page (100%)
    - Help with function-point analysis (100%)
    - SPR-5 meeting logs (100%)

Cody Strange:

- Total Hours: 19 hours, 40 minutes
- Total Meetings Attended: (11/11)
- Tasks Completed:
    - Review all SPR-5 document requirements (100%)
    - Help with function-point analysis (100%)
    - SPR-5 document (100%)
    - Metrics reports (100%)
    - Personal user acceptance test (100%)
    - Defects list (100%)
    - Risk management plan for SPR-5 (100%)

Tyler Deschamps:

- Total Hours: 12 hours
- Total Meetings Attended: (11/11)
- Tasks Completed:
    - Review all SPR-5 document requirements (100%)
    - Defects analysis (100%)
    - Backlogs for SPR-5 (100%)
    - Personal user acceptance test (100%)
    - SPR-5 charts development (100%)

Jordan Van Patten:

- Total Hours: 3 hour, 10 minutes
- Total Meetings Attended: (9/11)
- Tasks Completed:
    - Review all SPR-5 document requirements (100%)
    - V&V documents for SPR-5 (100%)

Kole Davis*:

- Total Hours: 2hrs 30 minutes
- Total Meetings Attended: (9/11)
- Tasks Completed:
    - Review all SPR-5 document requirements (100%)
    - Ishikawa diagrams (100%)
    - User acceptance test report (100%)

*Jordan and Kole joined the team in sprint 2 and 5 respectively

Plans

SPR-5 Risk Management Plan

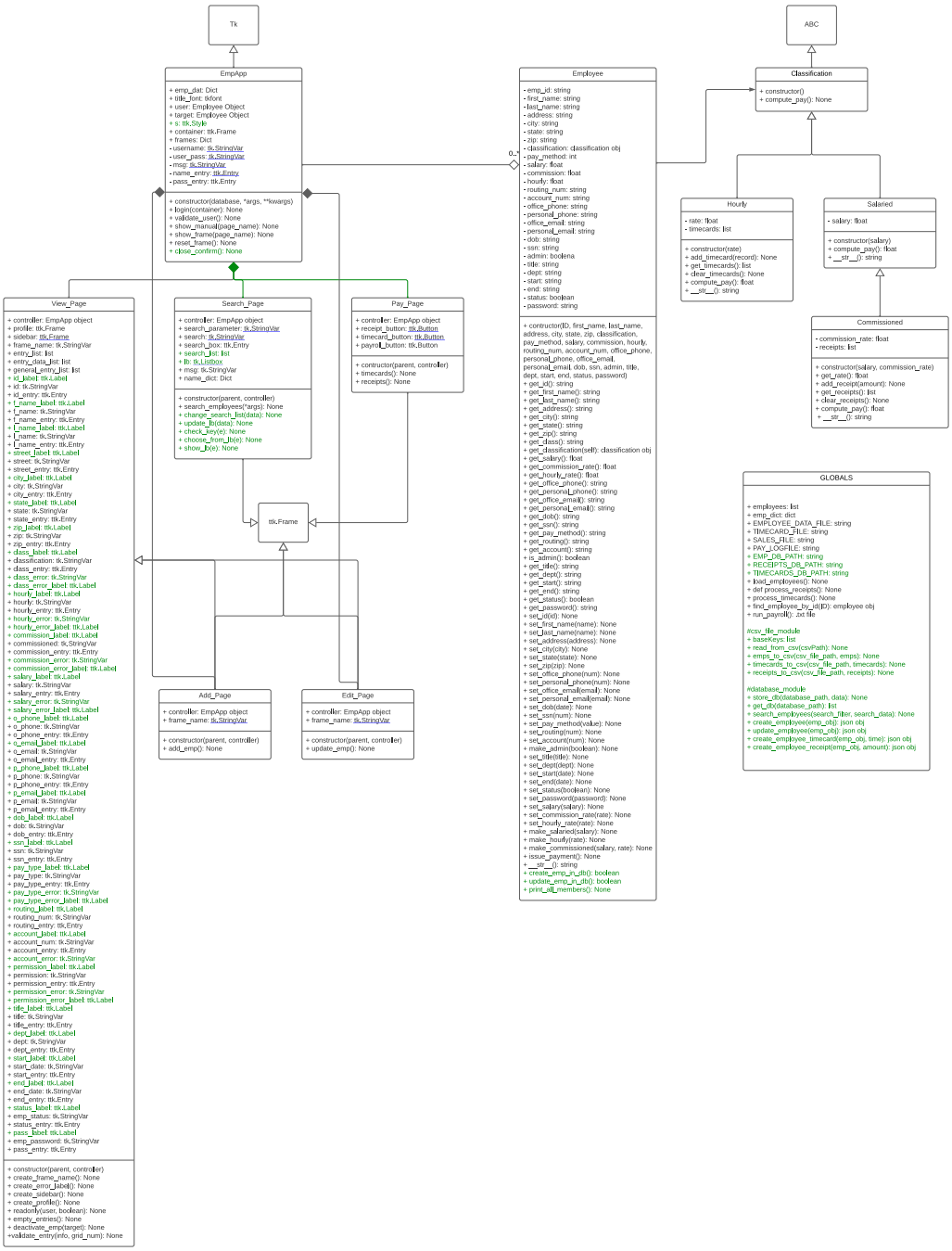
Summary: This is the sprint five risk management plan.

ID	Category	Description	Consequence	Probability	Impact	Risk Minimalization plan	Contingency plan
1	Metrics	missing certain information for metric reports	Inaccuracy in metric reports	High	Medium	Not deleting any information, making sure to keep records of all documents	Get the most out of the information that has been gathered
2	Program	Final touchups on the project cause it to stop working	Program couldn't be delivered to customer	Low	High	Keeping records of previous versions of the code so we can always got back in case something does happen	Use the newest version that still works
3	Shareholder	The shareholder requesting changes at the last minute	May not have enough time to implement the changes that the shareholder requests	Medium	High	Constant communication with the shareholder, understanding what they want as soon as possible	depending on what the shareholder requests, may have to save it for a later release
4	Change orders	Unable to complete the change orders that the shareholder has approved	Missing agreed upon functionality	Low	High	Proper cost estimation for the change orders, and time management to make sure it gets done	Let the shareholder know that we will have to save the change order for the next release
5	Submission	One or multiple team members either forgets to submit the final document or turns in the incorrect link	Penalties for incorrect submission	Low	High	Everyone meets a final time before submission and everyone confirms that they have the proper link	If no link was sent, send one and have a meeting to see what went wrong

Previous Sprint Items

Updated Class UML

Summary: The UML class diagram has been updated after we have made changes to the original plan for the project. The changes that have been made are marked in green.



Functional/Non-functional testing

Summary: For this test we went over each of the function and non-functional requirements and made sure that the program met each of them.

We have verified the contents of this document using our requirements specification documentation

Functional Requirements:

- Database Merging: Recognizes empty data fields based on both JSON and CSV databases and notifies the user of the missing data.
- Add employee (Admin access): inputs all required fields and constructs a new employee object to be added to an internal database.
- Edit employee (General & admin access): Lists selected employee's editable information fields (First name, Last name, Address, Office phone, Personal phone, Bank info, Office email, Personal email, etc.)
- Search employee by last name, ID, Title, or Classification (General & admin access): When user selects search parameter from search combo box, they are then able to start searching for employees. If a user enters numbers, the program will either display employees based on classification or ID number. Otherwise, the program will display employees based on job title or last name.
- View employee (General access): Displays selected employee's information in read-only mode and allows user to search for other employees' read-only information
- View employee (Admin access): Behaves similar to general access with the addition to deactivate an employee and run payroll.
- Deactivate employee (Admin access): Allows admin to input ID of employee they wish to deactivate. Once this occurs, a confirmation message appears, along with the selected employee's information to confirm the admin wishes to deactivate the selected employee
- Pay report (Admin access): Program Generates pay report including delivery information for each active employee.
- Garbage proof entries (General & admin access): Program ensures each data field contains valid information.
- Warn user of empty data fields (Admin access): When admin adds/edits an employee, issue a warning if any data fields are left empty or incomplete.

Non-Functional Requirements:

- The Program runs on Windows 10.
- Intuitive GUI: Mouse over input boxes to see what information is required, help buttons on each page.
- User Manual: Provides information on how to use and navigate the current page in the program (can be accessed by clicking the "Help" button).

- Readme.txt: A short description of what data the code contains.
- Simple "just download" installation that runs: Download software and then the user is good to go
- Employee can view all personal fields: When employee logs in with his/her ID, that ID's corresponding information is pulled up
- Bug free: Software will go through extensive testing to minimize/eliminate as many bugs as possible using Pytest and or Black, or any undiscovered software
- Requirements for employee information: First name, Last name, Address(use separated fields), Office phone, Personal phone, Emp ID(Specific length, only numbers), Pay type(commission, hourly, salary), D.O.B, SS#(Specific length, only numbers), Start date, End date, Bank Info(if Direct Deposit), Permission level, Title Dept., Office email, Personal email

Metrics

Defect Analysis

Document Defects

Summary: This analysis goes over each of the defects that have appeared in our sprint documents and explain how it could have been avoided, how we could have caught it earlier, and how we could have caught it before the shareholder.

SPR-1

- Incorrect link location
    - Could have been avoided by directly checking with the shareholder prior to submission.
    - We could have caught this earlier and beat the shareholder to the discovery by finishing the document quicker so we could double-check the link.
- Unspecified prototype candidate
    - Could have been avoided by being more specific about which prototype we were leaning toward.
    - We could have caught this earlier and beat the shareholder to the discovery by more thoroughly reviewing the requirements and rubric with him.
- Document Versioning missing
    - Could have been avoided by more thoroughly reviewing the sprint requirements.

    - Could have been caught earlier and beat the shareholder to the discovery by finishing the document early enough to review it with the shareholder.
- Coding standards missing
    - Could have been avoided by more thoroughly reviewing the sprint requirements.
    - Could have been caught earlier and beat the shareholder to the discovery by finishing the document early enough to review it with the shareholder.
- Meeting schedule missing
    - Could have been avoided by more thoroughly reviewing the sprint requirements.
    - Could have been caught earlier and beat the shareholder to the discovery by finishing the document early enough to review it with the shareholder.
- V&V document missing
    - Could have been avoided by more thoroughly reviewing the sprint requirements.
    - Could have been caught earlier and beat the shareholder to the discovery by finishing the document early enough to review it with the shareholder.
- Requirements specifications missing
    - Could have been avoided by more thoroughly reviewing the sprint requirements.
    - Could have been caught earlier and beat the shareholder to the discovery by finishing the document early enough to review it with the shareholder.

SPR-2

V&V document missing

- We could have avoided this by checking with our V&V teammate and ensuring they had completed all their assigned tasks.
- We could have caught this earlier by checking in on our V&V teammate more frequently.
- Submitted incorrectly
- Could have been avoided by directly checking with the shareholder prior to submission.
- We could have caught this earlier and beat the shareholder to the discovery by finishing the document quicker so we could double-check the link.

SPR-3

- GUI images missing
    - We could have avoided this by remembering to include the images we had in our user manual in the sprint document itself.
    - We could have caught this earlier by checking our sprint document more frequently for missing required information.
- Backlogs missing
    - Could have been avoided by keeping better track of the tasks we did and did not complete in previous sprints.
    - We could have caught this defect earlier and beat the shareholder to the discovery by more thoroughly reviewing the sprint requirements to know we needed to be keeping track of backlogs.
- Code doesn't run

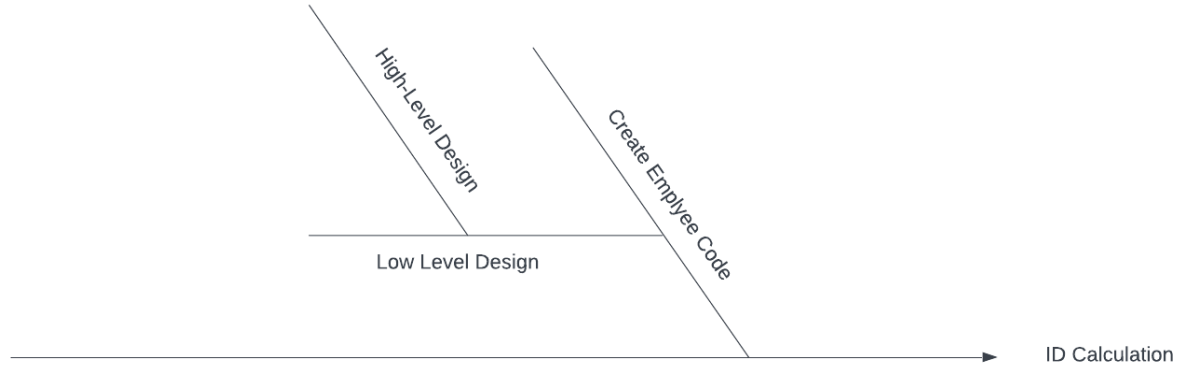
    - This defect could have been avoided by more thoroughly testing the code on a virgin computer.
    - We could have caught this defect earlier and beat the shareholder to the discovery by making sure to test the code on the platform designated in the requirements specification.
- Insufficient info in readme.txt
    - If we would have assigned the task at the beginning of the sprint, this file could have been produced incrementally through the sprint.
    - We included the info in the readme.txt very late. Finishing the development portion of this sprint earlier would have allowed us to add more to the file.
    - To summarize, early assignment and earlier completion of the development stage would have allowed more info to be in the readme.txt file to satisfy the shareholder.

SPR-4

- Missing backlogs
    - This defect could have been avoided if we had kept track of early project tasks as we progressed through sprints.
    - We hadn't left behind too many undone tasks so a backlog still could have existed, it just would have been empty. Perhaps having more team members would have allowed us to catch this defect.
    - Again, more people on the project would have allowed us to have more attention on this area of the project.
- Missing updated class diagram
    - This could have been avoided from V&V of the requirements and current contents of our program and sprint documents.
    - It could have been found earlier if our V&V had taken place earlier in the sprint.
    - Progressive V&V of progress and requirement comparison would have brought this to our attention before the shareholder discovered it.
- Missing User Acceptance Testing
    - This could have been avoided from V&V of the requirements and current contents of our program and sprint documents.
    - It could have been found earlier if our V&V had taken place earlier in the sprint.
    - Progressive V&V of progress and requirement comparison would have brought this to our attention before the shareholder discovered it.
- Missing Functional/Non-functional Testing
    - Our understanding of the requirements of this sprint and of the Functional/Non-functional Testing was not the same as the expectation from the shareholder. Further review with the Shareholder would have brought this to our attention so it could have been included.
    - Meeting early with the shareholder could have increased our likelihood of understanding our shareholders expectations.

- Early detection would have kept the shareholder from seeing that this report was missing.

Ishikawa Diagram



Software Metrics

Team Meetings

Summary: This table is designed to gather all of the information that has been gathered from the Team meetings, such as how many meetings we had, how often each member attended said meetings, and how many tasks were completed.

Meeting #	Sprint #	Date	# of attendees	Length of meeting	tasks completed	Tasks remaining	New tasks added
1	SPR-1	January 20, 2022	4/4	1hr 12min	n/a	n/a	n/a
2	SPR-1	January 21, 2022	4/4	41min	n/a	n/a	n/a
3	SPR-2	January 24, 2022	4/4	55min	4/15	11	15
4	SPR-2	January 28, 2022	5/5	1hr 5min	5/14	9	3
5	SPR-2	January 31, 2022	5/5	1hr 15min	8/16	8	7
6	SPR-2	February 4, 2022	4/5	55min	4/9	5	1
7	SPR-2	February 7, 2022	5/5	40min	5/8	3	3
8	SPR-3	February 11, 2022	4/5	1hr 4min	3/9	5	6
9	SPR-3	February 14, 2022	5/5	1hr 3min	5/14	9	9
10	SPR-3	February 18, 2022	4/5	1hr 8min	7/19	12	10

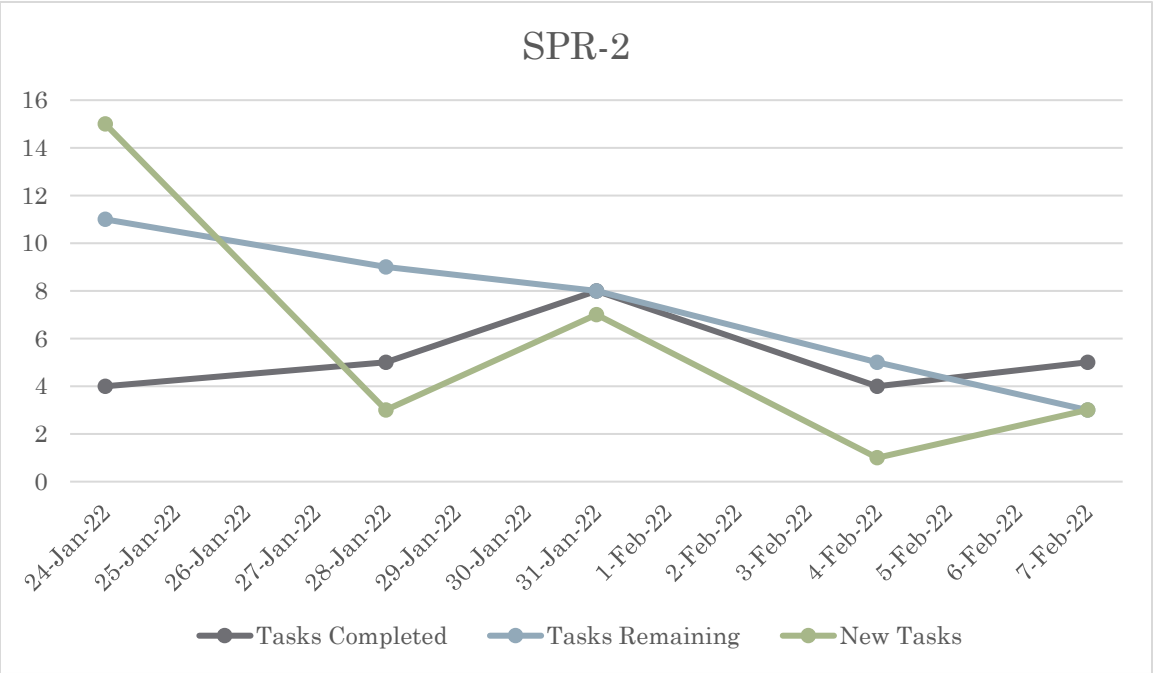
11	SPR-3	February 21, 2022	4/5	1hr 6min	3/15	12	3
12	SPR-3	February 28, 2022	5/5	1hr 4min	6/16	10	4
13	SPR-3	March 7, 2022	6/6	20min	5/13	8	3
14	SPR-4	March 14, 2022	5/6	1hr 4min	5/19	14	11
15	SPR-4	March 16, 2022	5/6	44min	10/21	11	7
16	SPR-4	March 18, 2022	6/6	45min	14/18	4	7

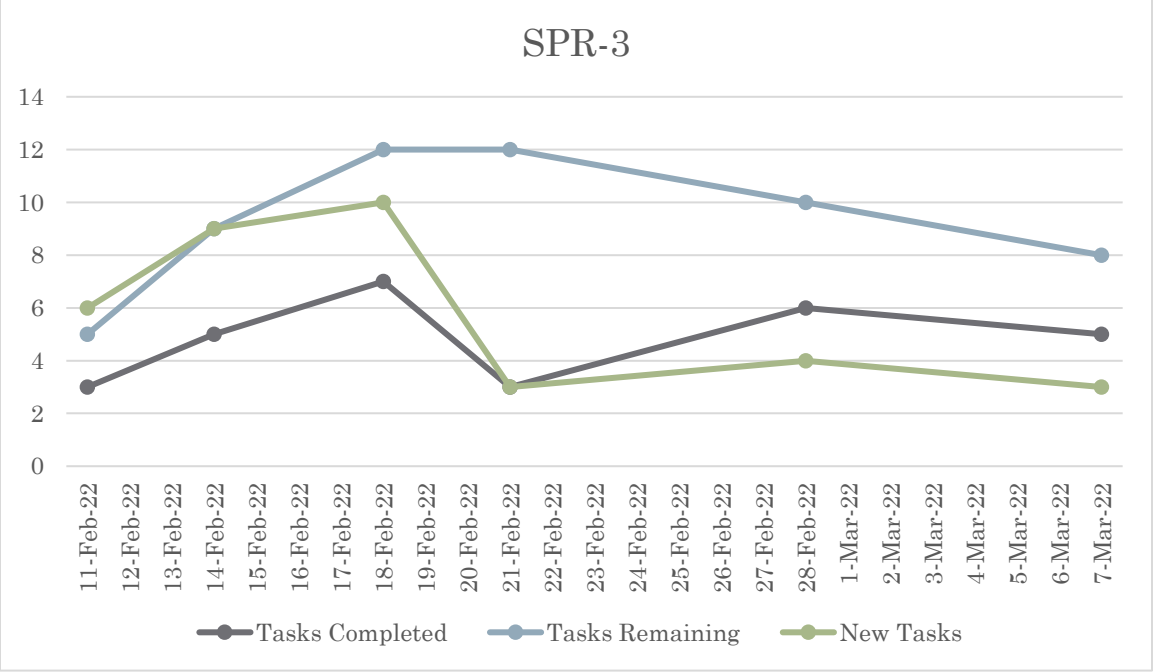
Report Analysis:

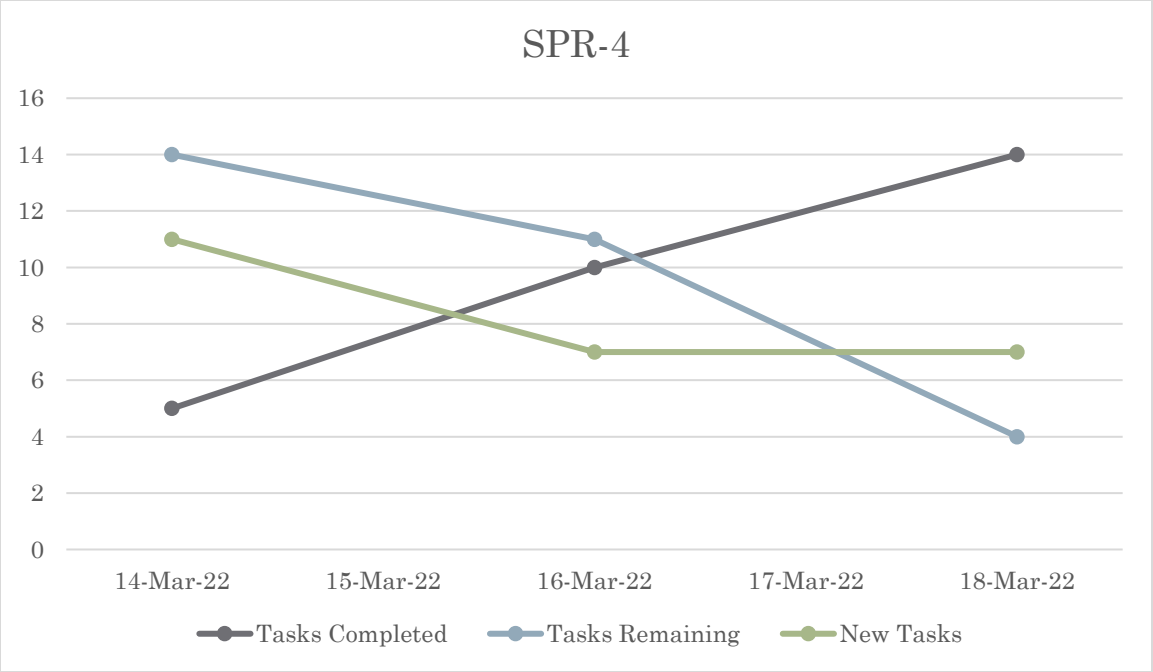
- What we did
    - We had two meetings a week on Monday and Friday for roughly one hour per meeting. We eventually added in a short meeting on most Wednesdays, we also didn't have many meetings/brief meetings over holidays and spring break.
    - Meetings tasks were not being tracked for the first sprint
- What we keep
    - Three meetings a week seems to have worked pretty well, and for projects of similar size we would keep the schedule we currently have. An hour length has been enough time for everyone to say everything that they need to, so we would keep that as well.
- What we would do better
    - Meeting more often over long holidays
    - Keep track of the short meetings as well as the normal ones
    - Keep track of tasks starting with the first sprint

Tasks Completed to Tasks Remaining by Sprint

Summary: These charts are designed to show how many tasks were completed by each meeting where they could be reported to the overall tasks remaining. It also shows the amount of new tasks that were being added which would affect the overall tasks remaining.







Report Analysis

- What we did
    - Each sprint we would figure out each of us would need to do and then we would set out to do it
    - As new things were discovered that needed to be done, we would bring it up in a one of our meetings and decide who got to take care of it
    - We would report when we finished it through the group chat and how long it took us to do it
- What we would keep
    - The group chat worked out really well and we would likely keep it
    - Everyone figuring out what they needed to get done based on their role
- What we would do better
    - We have already changed this, keeping better tabs on what everyone is doing and how far along they are on their tasks
    - Try to figure out all of the tasks for the sprint sooner

Assert Testing

Summary: This table is designed to look at groups of assert tests and states what they test for, how many tests they have total, what different inputs they are testing for, who ran the tests, and whether or not they all passed.

Activity	Metric Measure	number of tests	Test Inputs	Source	Current State
Tests that employee has the correct ID, classification and pay	Assert	3	ID numbers: 2, 3, 7	Jordan	Passed
searches for a user that should be in the database	Assert	5	ID numbers: 5, 1, 100 Last name: johnasdk, LastNameChanged	Jordan	Passed
Tests that fields are registered as valid when valid results are input	Assert	1	Employee object: With all valid field	Jordan	Passed
Tests that fields are registered as invalid when invalid results are input	Assert	1	Employee object: Missing one field	Jordan	Passed
Tests that the information shown on the general users viewpage matches their informatin in the database	Assert	4	ID numbers: 8, 2, 3, 4	Jordan	Passed
Tests that the information shown on the admin users viewpage matches their information in the database	Assert	2	ID numbers: 8, 6	Jordan	Passed
Tests that using the correct username and ID does NOT trigger the "Incorrect password or ID" message	Assert	7	ID numbers: 8, 2, 3, 4, 5, 6, 7	Jordan	Passed
Tests that using the incorrect username but correct password does trigger the "Incorrect password or ID" message	Assert	8	ID numbers: Jim, tony, 123, 57, gfad, wrong, almost, so close	Jordan	Passed
tests that using the incorrect password but correct username does	Assert	11	Passwords: tests, test!, tesT, Test, nope, tESt, password, letMeIn,	Jordan	Passed

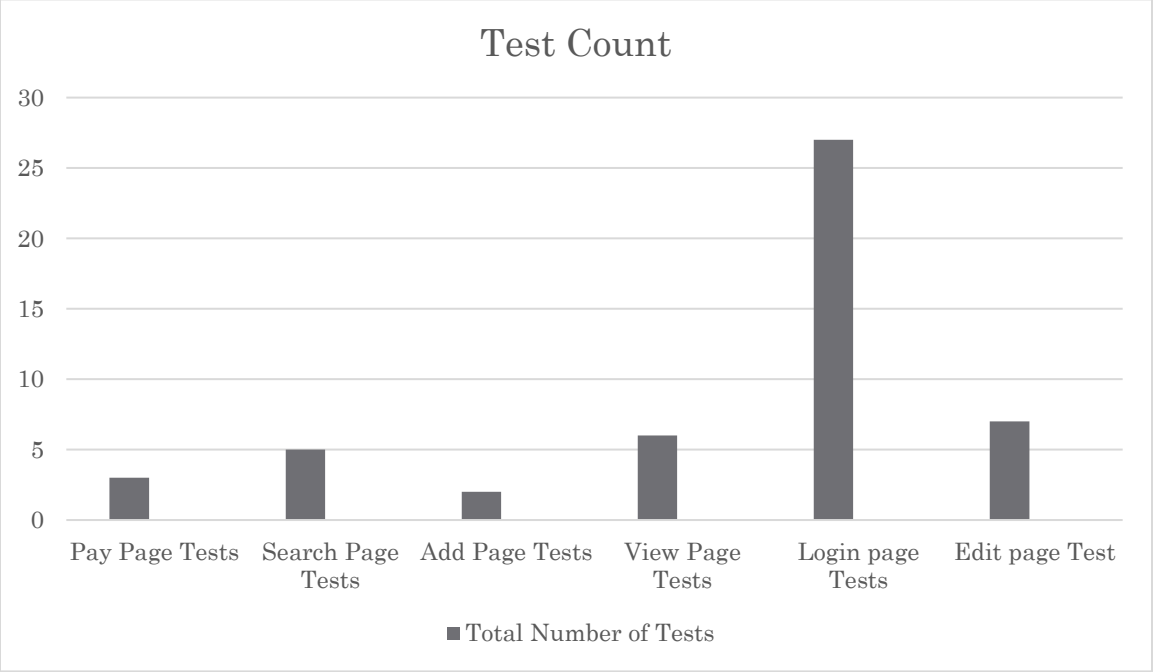
trigger the "Incorrect password or ID" message			please, prettyplease, 123456		
Tests that all fields are valid when valid inputs are given	Assert	6	ID numbers: 8, 2, 3, 4, 6	Jordan	Passed
test_edit_invalid_variable	Assert	1	Employee object: Invalid fields	Jordan	Passed

Report Analysis

- What we did
    - Extensive testing was we programmed so that our code was more likely to pass all of the assertion tests
    - Write assertion tests for each page
    - Test for a variety of inputs
- What we would keep
    - Testing the code as you write it, that way you can catch and fix a bug as soon as possible
    - Using pytest to do assertion testing
- What we would do better
    - Write tests for more unique scenarios, such as negative numbers, special characters like @, and decimal numbers

Test Count by GUI Page

Summary: This chart shows the number of tests that were written for each page. We increased login tests to make sure that the information would be correct. For the add page each individual test involved a lot of fields being tested at once. The other pages didn't need as many fields to be checked and had less that could be incorrect for each test, so not as many tests were required.



Report Analysis

- What we did
    - Wrote tests to check for what a user might be inputting and make sure that it would run as we the developers would expect it to run.
- What we would keep
    - We would keep all of the tests that we have
- What we would do better
    - Add boundary tests

Document Defects

Summary: This table is designed to show the document defects that arose throughout the sprints in the project and the current status of said defects. Document defects are items that are missing from the sprint documents that are required for the project to match the requirement specifications.

SPR#	Document Defect Description	Status
SPR-1	Incorrect link location	
SPR-1	Unspecified prototype candidate	
SPR-1	Document Versioning missing	
SPR-1	Coding standards missing	
SPR-1	Meeting schedule missing	
SPR-1	V&V document missing	
SPR-1	Requirements specifications missing	
SPR-2	V&V document missing	
SPR-2	Submitted incorrectly	
SPR-3	GUI images missing	
SPR-3	Backlogs missing	
SPR-3	Code doesn't run	
SPR-3	No readme.txt	
SPR-4	Missing backlogs	
SPR-4	Missing updated class diagram	
SPR-4	Missing User Acceptance Testing	
SPR-4	Missing Functional/Non-functional Testing	

Report Analysis

- What we did
    - We each researched on our own what was required for each of our roles for the sprint
    - We came up with what we believed were the requirements for the sprint and tasks that need to be completed to meet those requirements in team meetings
- What we would keep
    - Individual research
    - Team discussions
- What would we do better
    - Contact the shareholder more often to help understand everything that is required to avoid creating defects

Program Bugs

Summary: This table is designed to show the bugs that have appeared in the code, what the bug caused, what was changed to fix the bug, when the bug was found and fixed, and whether the bug is currently fixed or not.

ID#	Bug Description	Change Description	Date Found	Date Fixed	Status
1	-The GUI currently functions, but the data being operated on (employee objects) is currently hard coded into the program. All new data, changes, etc. is lost when program closes.	-Add JSON text files as databases so we can add, edit, and retrieve records for employees, timecards, and receipts. -Add modules for the employee class that will add, edit, and retrieve records from the employees JSON file.	02/16/2022	03/02/2022	
2	-Tkinter message boxes aren't being imported when the program loads. (Only for development in windows)	-Include the specific library that applies to Tkinter libraries.	02/23/2022	02/23/2022	
3	-The GUI page for payroll did not have the run payroll or run csv file buttons functional.	-Add a CSV and handling module for the imported csv's that need to calculate the final pay for hourly, and compensated employees. -Add JSON text file for a database as a backup and add accessibility functions for this database in a module.	03/15/2022	03/18/2022	
4	-The ID was being keyed off of the number of employees in the database. With testing, and perhaps future functionality, deleting employees would cause conflicts with newly created employee ID's.	-Whenever a new employee is added, their Id is one plus the last employee in the database ID.	03/18/2022	03/19/2022	
5	-The user would have been unable to access the program as there was no user information provided when the program was loaded onto a new computer.	-Included a readme.txt file that included information regarding basic login for admin. From there, the user would be able to look up other employee information and create / edit employees as necessary.	03/20/22	03/20/2022	
6	-There was trouble for the shareholder originally with starting up the GUI. They were trying to launch the program from an	-There were some irrelevant files that existed in the next version of the program that could have caused more confusion. These were removed, and the file that the program needs to be launched from	03/12/2022	03/20/2022	

	incorrect file. File naming isn't explanatory.	was renamed to main.py for clarity. This was then mentioned in the readme.txt file.			
7	-There were issues caused by the validation function when it tried to update the field-specific error messages.	-The issues were caused by a logic error in the validation function attempting to alter label widgets that had not been created. The issue was solved by including a try/except block that checks for widget presence before alteration.	03/18/2022	03/18/2022	
8	-The error messages on all GUI pages were hardly noticeable.	-Solved by implementing a specific style with larger, more colorful text for error messages.	03/27/2022	03/27/2022	
9	-All fields were editable in the edit page, including things that realistically should not be changed, even by an admin (e.g. employee id, start date, etc)	-Solved by enabling read only mode on select entry boxes. -Admins can change everything except employee id, start date, end date, and employee status, while general permission employees editing themselves can change only personal information and not anything related to the company like their pay and job title.	03/03/2022	03/03/2022	
10	-There were issues attempting to set the entry boxes of the view page to "readonly" when in general view	-Issue was caused by a logic error in the method meant to set all entry boxes to "readonly". It would change the widgets created by the admin view, which included many not used for the general view. Issue was solved by creating a second list to be passed to the readonly method when not in admin view.	03/01/2022	03/01/2022	
11	-The shareholder did not approve of the ability to add timecards and receipts individually	-Solved by reverting the pay page of the GUI back to its initial state with only the ability to import the timecard and receipt csv files.	03/25/2022	04/10/2022	
12	-The search page on the GUI was not very user-friendly, requiring prior knowledge of the database to operate easily.	-Implemented a listbox widget below the search bar that shows all the employees in the database that match the current contents of the search bar.	03/27/2022	03/28/2022	
13	-There were issues moving from the search page to the view page of the searched employee when	-Issues were caused by the program attempting to alter widgets that had not been created (they are created only in admin view or when viewing oneself).	03/18/2022	03/18/2022	

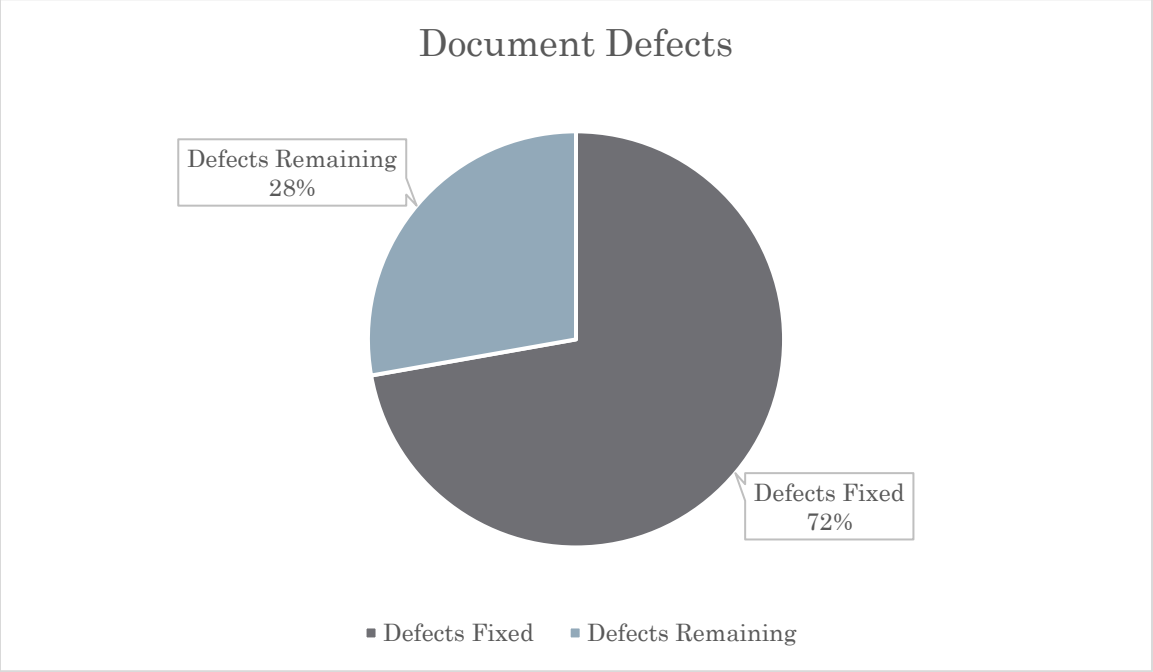
	logged in under general permission.	Solved the issues by placing the code that altered those widgets within an if statement that checks whether they exist or not first.			
14	-The pages of the GUI would not correctly refresh/update when switching between pages (e.g. old data on the add page would remain instead of being cleared).	-Created a new method for the Application class to be called with every page change that destroys and then recreates the pages to allow them to update properly.	02/13/2022	02/13/2022	
15	-There was trouble getting the entry boxes of the view page to correctly update their contents/configurations for inheritance with the add and edit pages.	Moved the code for creating the entry boxes out of the constructor for the View_Page class and into a method that can then be called by that class or any of its children as necessary.	02/13/2022	02/13/2022	

Report Analysis

- What we did
    - We fixed the bugs as we wrote the code, so the dates are not guaranteed to be 100% accurate but all of the bugs have been fixed because they were dealt with as they were found
- What we would keep
    - Keep fixing bugs as we code
- What we would do better
    - Document bugs as soon as they are found

Document Defects Remaining vs Defects Fixed

Summary: This chart shows the percent of defects that have been found in the document and have yet to be fixed.

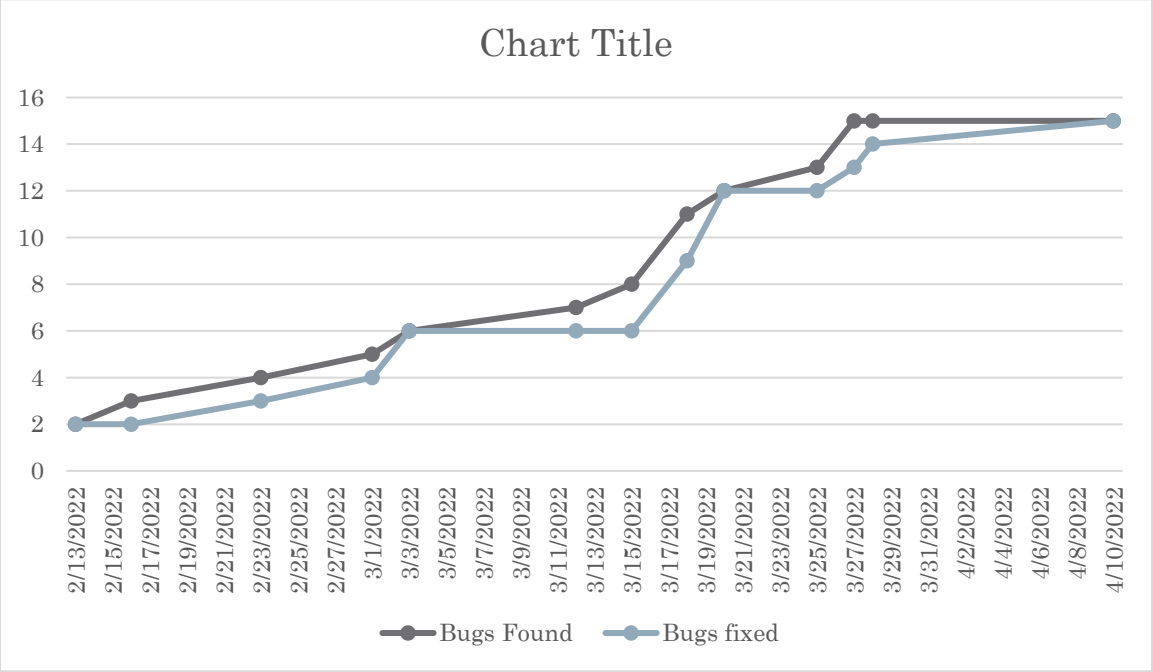


Report Analysis

- What we did
  - Fix defects from the previous sprint in the current sprint
  - Go through the comments left on sprint submission to see what defects are in the document
- What we would keep
  - Keep fixing defects in the sprint when they are found
  - Document defects found
- What we would do better
  - document dates of when defects are found and fixed

Bugs Found to Bugs Fixed

Summary: This chart shows the total number of bugs that were found in the program and when they were found, along with the total number of bugs fixed in the program and when they were fixed.



Report Analysis

- What we did
    - We fixed bugs as they were found which meant we had to go back and figure out when they were found and when they were fixed.
- What we would keep
    - Fixing bugs as we found them
- What we would change
    - Document bugs as they are found

Function Point Analysis

Summary: The function point analysis calculates a complexity value for each of the items that are being counted and uses them to get a sense of the overall complexity.

Category	Number		Low	Medium	High		Result
Inputs	50	x	3	4	6	=	150
Outputs	26	x	4	5	7	=	104
Inquires	9	x	3	4	6	=	27
Internal Files	8	x	7	10	15	=	56
External Files	0	x	5	7	10	=	0

Total = 337

Importance	0
Irrelevant	1
Minor	2
Moderate	3
Average	4
Significant	5
Essential	6

Factor	Rating
Data communication	2
Distributed data processing	4
Heavily used configuration	1
Transaction rate	1
Online data entry	0
End user efficiency	0
Online update	0
Complex processing	0
Reusability	5
Installation ease	5
Operational ease	5
Multiple sites	0
Facilitate change	3
Total (CAV)	26

Calculate Adjusted FP

FP = (337) * (0.65 + 0.01 * 26) = 307

Alpha/User Acceptance Test

Bugs

Summary: We combined all of our individual bug tests into one larger report. This is the Alpha portion.

- When we try to search for an employee, the page header shifts to the left once the search box gains keyboard focus and then shifts back to its original location once it is no longer in focus.

- When we hover our mouse over the search parameter combo box, the edit cursor appears, which makes no sense if I'm selecting an item from the list rather than typing something into the entry (My suggestion would be to make it so the mouse doesn't change when hovering over the search parameter combo box in order for users to not be caught off guard by the edit cursor when they're supposed to be selecting a list item).

- When we begin searching for an employee and wish to go back to change my search parameter, both the search list box and the search parameter combo box are stacked on top of one another, and this doesn't look very clean.

- Deactivated employees are still included in the payroll, and get paid out. A deactivated employee shouldn't be paid out at all.

- Can pay someone negative amount. This should be an easy fix with a simple entry validation.

- Employee ID 7 isn't on the payroll

- After trying to login with an incorrect ID and then immediately logging in with a proper ID, the "Incorrect ID" flashes whenever I change tabs. Same if I type in the incorrect password.

- Many fields can be left blank. From my testing, most of the fields that are still required on the add new employee don't necessarily need to be filled out just yet. An employee may not have an office phone set up yet, not have bank info yet, not have been given a title, etc.

- If we deactivate an employee, there should be a way to reactivate them as well.
- We need to develop a read me txt file for the program.

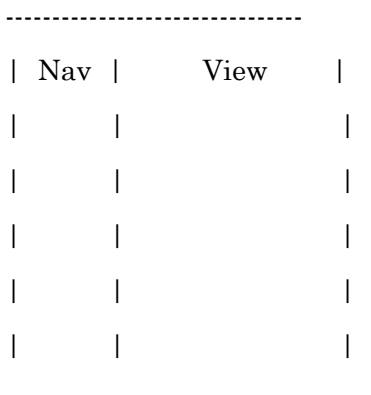
Functional or Design Improvements

Summary: This is the user acceptance portion. Here we focused on more functional and design improvements.

- The employee classification field label would look a lot cleaner without the numerical classification options in the parentheses (My suggestion would be to make this entry field a combo box, which would prevent users from entering wrong information into this field).

- When we select an employee from the search drop-down list, the search box is filled with the employee's information, but the box itself is too small to fit all the information.

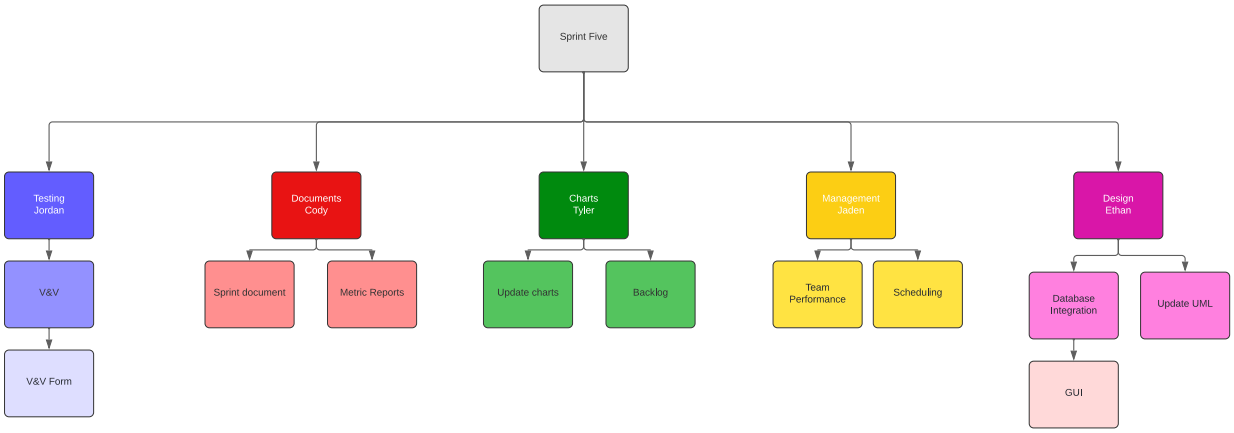
- Sometimes it can be difficult knowing if we have typed in all the correct employee information into each data field before attempting to change the employee's information and getting a message saying the information is incorrect (My suggestion is instant validation).
- The program needs a theme.
- Certain fields like pay method, classification, and permission level would be more user friendly and easily validated with dropdowns.
- The alerts under each field that is required or needs attention after validation should be styled red.
- Pop-up at bottom of screen that says not all fields are valid probably would be more visible at the top of the screen.
- Have a filter for the pay type classification so that you cannot enter the other pay type fields to avoid messing up payroll.
- Create notifications of some sort to let the user know that they had successfully processes timecards, receipts, and printed out payroll.
- Add a search list that can be dynamically filtered by the search keys entered into the search bar.
- We may want to section off our view window from our nav bar on the left. Help to be more user friendly and guide the eyes of the user better. Ex.



Charts/Forms

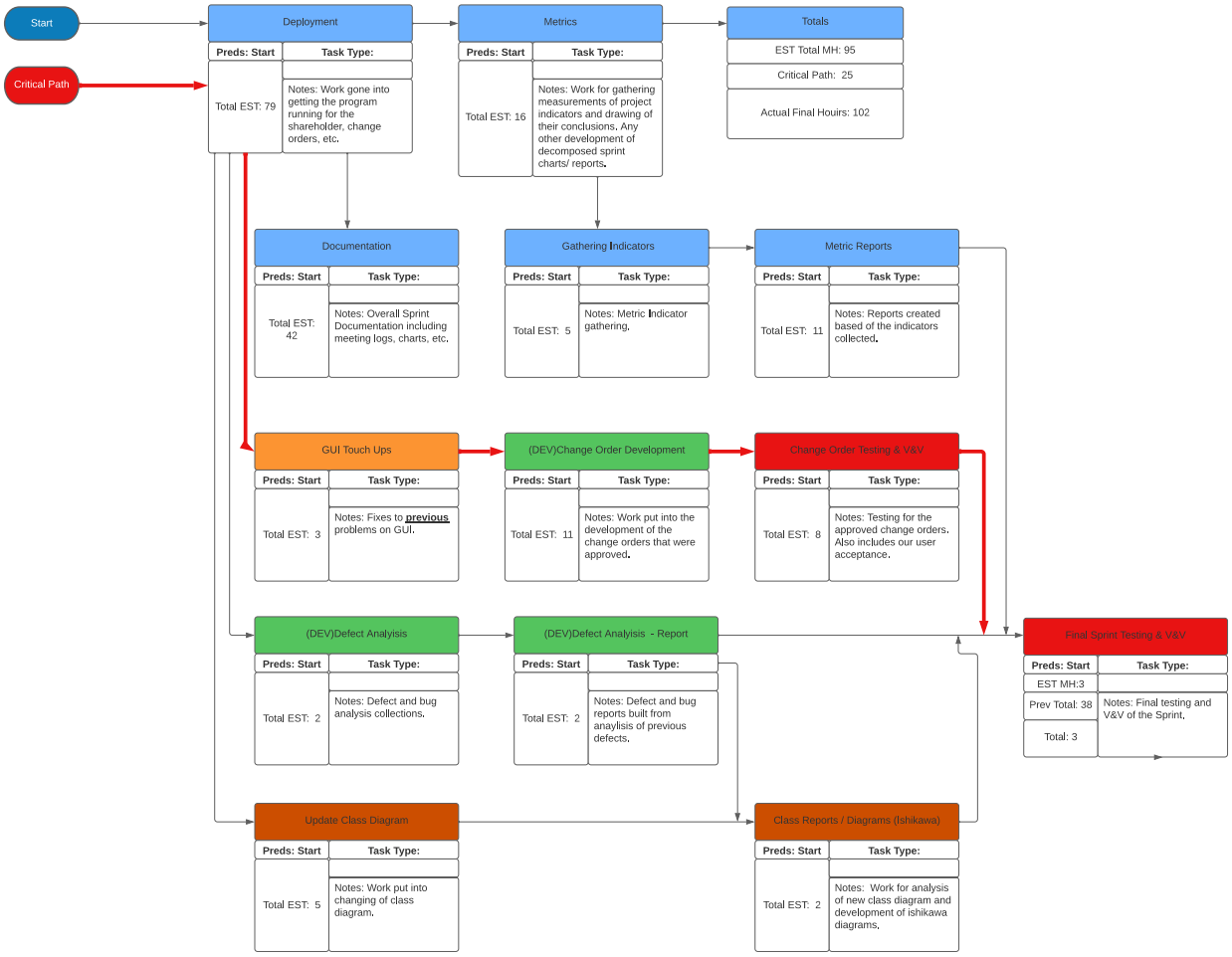
SPR-5 Work Breakdown Structure

Chart



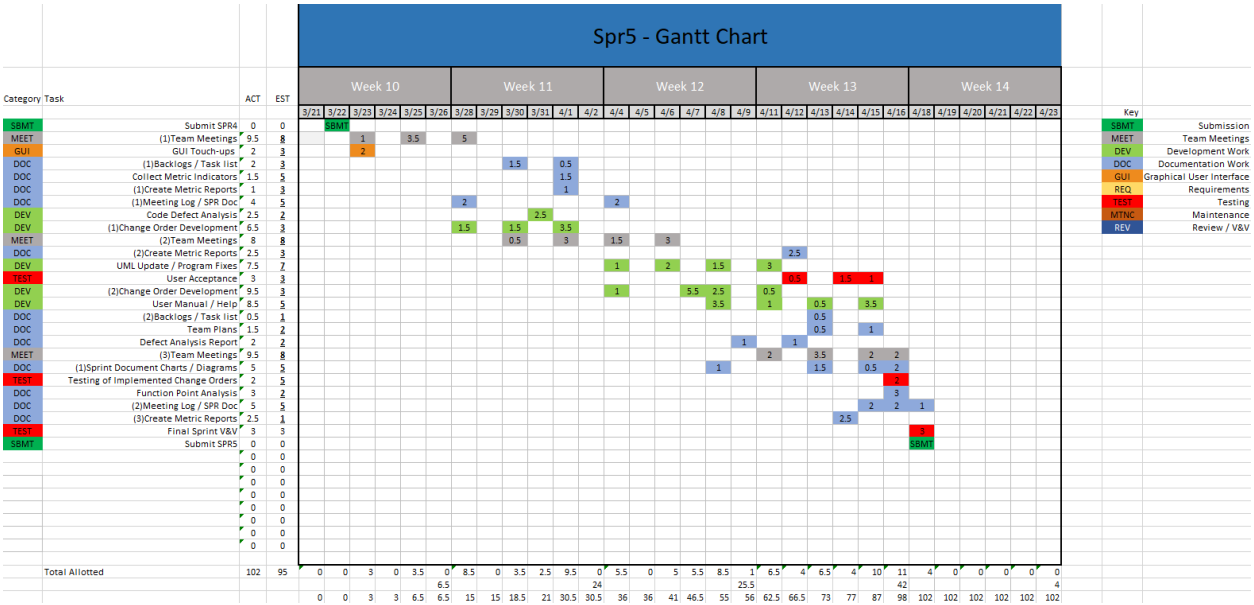
SPR-5 Pert Chart

Chart



SPR-5 Gantt Chart

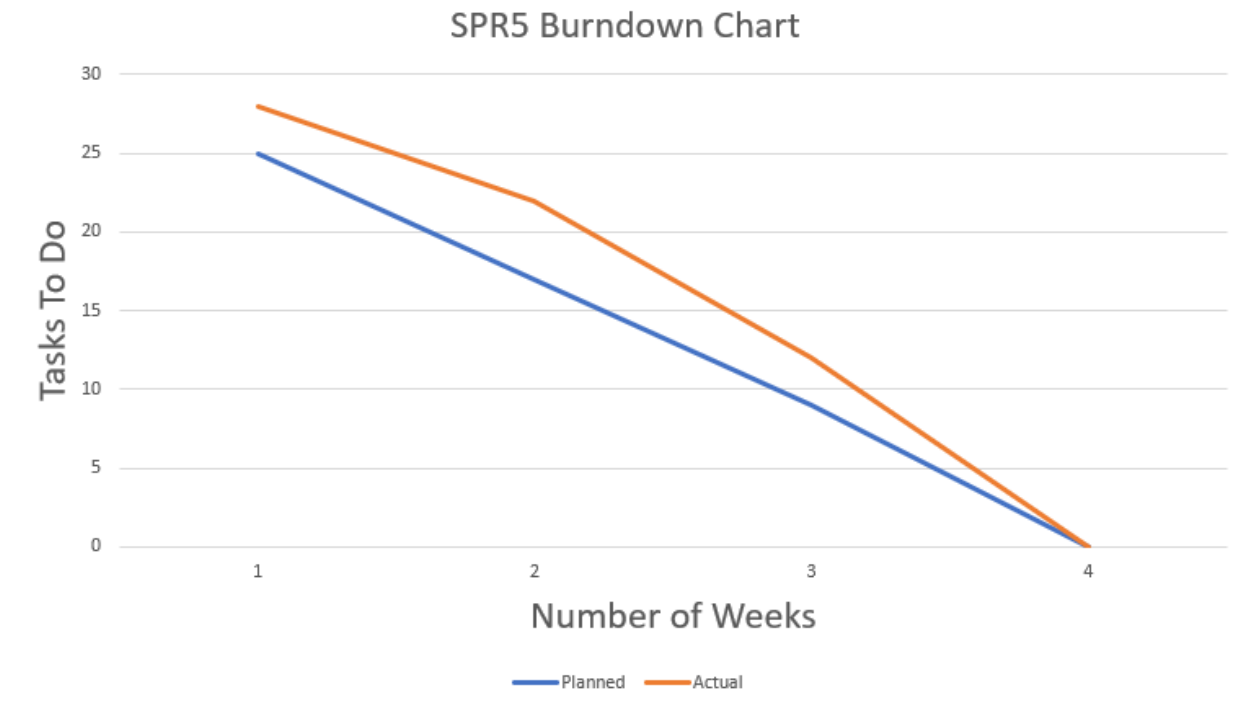
Chart



Category	Task	ACT	EST
SBMT	Submit SPR4	0	0
MEET	(1)Team Meetings	9.5	8
GUI	GUI Touch-ups	2	3
DOC	(1)Backlogs / Task list	2	3
DOC	Collect Metric Indicators	1.5	5
DOC	(1)Create Metric Reports	1	3
DOC	(1)Meeting Log / SPR Doc	4	5
DEV	Code Defect Analysis	2.5	2
DEV	(1)Change Order Development	6.5	3
MEET	(2)Team Meetings	8	8
DOC	(2)Create Metric Reports	2.5	3
DEV	UML Update / Program Fixes	7.5	7
TEST	User Acceptance	3	3
DEV	(2)Change Order Development	9.5	3
DEV	User Manual / Help	8.5	5
DOC	(2)Backlogs / Task list	0.5	1
DOC	Team Plans	1.5	2
DOC	Defect Analysis Report	2	2
MEET	(3)Team Meetings	9.5	8
DOC	(1)Sprint Document Charts / Diagrams	5	5
TEST	Testing of Implemented Change Orders	2	5
DOC	Function Point Analysis	3	2
DOC	(2)Meeting Log / SPR Doc	5	5
DOC	(3)Create Metric Reports	2.5	1
TEST	Final Sprint V&V	3	3
SBMT	Submit SPR5	0	0
		0	0
		0	0
		0	0
		0	0
		0	0
		0	0
		0	0
	Total Allotted	102	95

SPR-5 Burndown Chart

Chart



Meeting Logs

Meeting Log#17

Meeting Information

- Team #: T2-002
- Meeting log #: 17
- Current Sprint: SPR5
- Date: March 28, 2022
- Time: 7:00pm – 8:00pm (MT)
- Location: MS Teams (Watch video here)
- Attendees: Jaden Albrecht, Tyler Deschamps, Jordan Van Patten, Cody Strange, Kole Davis
- Next team meeting scheduled for: March 30, 2022, 3:45pm (MT)

Progress From Previous Meeting

- Ethan Taylor:
  - Defects analysis (100%)
- Jaden Albrecht:
  - Personal user acceptance test (100%)
- Tyler Deschamps:
  - Defects analysis (100%)

Topics Discussed

- Ishikawa diagrams
- User acceptance testing
- Functional/non-functional requirements testing
- Function-point analysis
- Backlogging for SPR5
- Team performance review

Obstacles Encountered

- No obstacles

Finished Items

- Ethan and Tyler have gone through and put together a list of defects found during the implementation of our final project

Unfinished Items

- SPR-5 document
- Metric reports
- Team performance review
- V&V documents for SPR-5
- Defect analysis/report
- Functional/nonfunctional requirements test
- Updated UML diagram
- WBS for SPR-5

- Backlogs for SPR-5
- Ishikawa diagram
- Pert/Gantt/burn-down charts for SPR-5
- Implementation of approved change order requests from previous sprints
- Usability test
- User acceptance test report

Tasks Until Next Meeting

- User acceptance test report
- Ishikawa diagram(s)
- WBS for SPR5
- Pert/Gantt/burn-down chart development for SPR5
- Functional/non-functional tests

Notes

- Ethan was sick and couldn't attend this meeting

Meeting Log#18

Meeting Information

- Team #: T2-002
- Meeting log #: 18
- Current Sprint: SPR5
- Date: March 30, 2022
- Time: 3:45pm – 4:00pm (MT)
- Location: In-class meeting
- Attendees: Jaden Albrecht, Tyler Deschamps, Jordan Van Patten, Cody Strange, Kole Davis
- Next team meeting scheduled for: April 1, 2022, 7:00pm (MT)

Progress From Previous Meeting

- Jaden Albrecht:
  - Functional/non-functional requirements testing (20%)
- Tyler Deschamps:
  - Defects research

Topics Discussed

- Ishikawa diagrams
- User acceptance testing
- Functional/non-functional requirements testing
- Function-point analysis
- Backlogging for SPR5
- Team performance review

Obstacles Encountered

- No obstacles

Finished Items

- No items finished

Unfinished Items

- SPR-5 document
- Metric reports
- Team performance review
- V&V documents for SPR-5

- Defect analysis/report
- Functional/nonfunctional requirements test
- Updated UML diagram
- WBS for SPR-5
- Backlogs for SPR-5
- Ishikawa diagram
- Pert/Gantt/burn-down charts for SPR-5
- Implementation of approved change order requests from previous sprints
- Usability test
- User acceptance test report

Tasks Until Next Meeting

- User acceptance test report
- Ishikawa diagram(s)
- WBS for SPR5
- Pert/Gantt/burn-down chart development for SPR5
- Functional/non-functional tests

Notes

- No additional notes

Meeting Log#19

Meeting Information

- Team #: T2-002
- Meeting log #: 19
- Current Sprint: SPR5
- Date: April 1, 2022
- Time: 7:00pm – 7:30pm (MT)
- Location: MS Teams (watch video here)
- Attendees: Ethan Taylor, Jaden Albrecht, Tyler Deschamps, Cody Strange, Kole Davis
- Next team meeting scheduled for: April 4, 2022, 7:00pm (MT)

Progress From Previous Meeting

Cody Strange:

- Assert test report (100%)
- Defect list (100%)

Topics Discussed

- Ishikawa diagrams
- User acceptance testing
- Functional/non-functional requirements testing
- Function-point analysis
- Backlogging for SPR5
- Team performance review

Obstacles Encountered

- No obstacles

Finished Items

- Defects report
- Assertion test report

Unfinished Items

- SPR-5 document
- Metric reports
- Team performance review
- V&V documents for SPR-5
- Functional/nonfunctional requirements test
- Updated UML diagram
- WBS for SPR-5
- Backlogs for SPR-5
- Ishikawa diagram
- Pert/Gantt/burn-down charts for SPR-5
- Implementation of approved change order requests from previous sprints
- Usability test
- User acceptance test report

Tasks Until Next Meeting

- User acceptance test report
- Ishikawa diagram(s)
- WBS for SPR5
- Pert/Gantt/burn-down chart development for SPR5
- Functional/non-functional tests
- Update UML class diagram

Notes

- No additional notes

Meeting Log#20

Meeting Information

- Team #: T2-002
- Meeting log #: 20
- Current Sprint: SPR5
- Date: April 4, 2022
- Time: 7:00pm – 7:15pm (MT)
- Location: MS Teams (Watch video here)
- Attendees: Jaden Albrecht, Tyler Deschamps, Jordan Van Patten, Cody Strange, Kole Davis
- Next team meeting scheduled for: April 6, 2022, 3:45pm (MT)

Progress From Previous Meeting

- No tasks completed

Topics Discussed

- Ishikawa diagrams
- User acceptance testing
- Functional/non-functional requirements testing
- Function-point analysis
- Backlogging for SPR5
- Team performance review
- Updating UML diagrams

Obstacles Encountered

- No obstacles

Finished Items

- No new tasks have been completed from previous meeting

Unfinished Items

- SPR-5 document
- Metric reports
- Team performance review
- V&V documents for SPR-5
- Defect analysis/report
- Functional/nonfunctional requirements test
- Updated UML diagram
- WBS for SPR-5
- Backlogs for SPR-5
- Ishikawa diagram
- Pert/Gantt/burn-down charts for SPR-5
- Implementation of approved change order requests from previous sprints
- Usability test
- User acceptance test report

Tasks Until Next Meeting

- User acceptance test report
- Ishikawa diagram(s)
- WBS for SPR5
- Pert/Gantt/burn-down chart development for SPR5
- Functional/non-functional tests

Notes

- Ethan was sick and couldn't attend this meeting

Meeting Log#21

Meeting Information

Team #: T2-002

- Meeting log #: 21
- Current Sprint: SPR5
- Date: April 6, 2022
- Time: 3:45pm – 4:00pm (MT)
- Location: In-class meeting
- Attendees: Jaden Albrecht, Tyler Deschamps, Jordan Van Patten, Cody Strange, Kole Davis
- Next team meeting scheduled for: April 11, 2022, 7:00pm (MT)

Progress From Previous Meeting

- Ethan Taylor:
  - Updated UML diagrams (100%)
  - Integrate old database (100%)
- Jaden Albrecht:
  - Functional/non-functional requirements testing (20%)
- Cody Strange:
  - SPR-5 document (90%)
- Tyler Deschamps:
  - Backlogs for SPR-5 (40%)
- Kole Davis:
  - Ishikawa diagram (50%)

Topics Discussed

- User acceptance testing
- Functional/non-functional requirements testing
- Function-point analysis
- Backlogging for SPR5
- Team performance review

Obstacles Encountered

- No obstacles

Finished Items

- Updated UML diagrams
- Ishikawa diagrams

Unfinished Items

- SPR-5 document
- Metric reports
- Team performance review
- V&V documents for SPR-5
- Functional/nonfunctional requirements test

- WBS for SPR-5
- Backlogs for SPR-5
- Pert/Gantt/burn-down charts for SPR-5
- Implementation of approved change order requests from previous sprints
- Usability test
- User acceptance test report

Tasks Until Next Meeting

- User acceptance test report
- Ishikawa diagram(s)
- WBS for SPR5
- Pert/Gantt/burn-down chart development for SPR5
- Functional/non-functional tests
- Implementation of previous change order requests

Notes

- No additional notes

Meeting Log#22

Meeting Information

- Team #: T2-002
- Meeting log #: 22
- Current Sprint: SPR5
- Date: April 11, 2022
- Time: 7:00pm – 7:20pm (MT)
- Location: MS Teams (watch video here)
- Attendees: Ethan Taylor, Jaden Albrecht, Tyler Deschamps, Jordan Van Patten, Cody Strange
- Next team meeting scheduled for: April13, 2022, 3:45pm (MT)

Progress From Previous Meeting

- Ethan Taylor:
    - Integrate CSV database (100%)
- Jaden Albrecht:
    - Functional/non-functional requirements testing (90%)
    - Search parameters for search page (90%
- Tyler Deschamps:
    - Backlogs for SPR-5 (80%)

Topics Discussed

- User acceptance testing
- Functional/non-functional requirements testing
- Function-point analysis

- Backlogging for SPR5
- Team performance review

Obstacles Encountered

- No obstacles

Finished Items

- CSV database integration

Unfinished Items

- SPR-5 document
- Metric reports
- Team performance review
- V&V documents for SPR-5
- Functional/nonfunctional requirements test
- WBS for SPR-5
- Backlogs for SPR-5
- Pert/Gantt/burn-down charts for SPR-5
- Implementation of approved change order requests from previous sprints
- Usability test
- User acceptance test report

Tasks Until Next Meeting

- User acceptance test report
- WBS for SPR5
- Pert/Gantt/burn-down chart development for SPR5
- Functional/non-functional tests
- Implementation of previous change order requests

Notes

- No additional notes

Meeting Log#23

Meeting Information

- Team #: T2-002
- Meeting log #: 23
- Current Sprint: SPR5
- Date: April 13, 2022
- Time: 3:45pm – 4:00pm (MT)
- Location: In-class meeting
- Attendees: Ethan Taylor, Jaden Albrecht, Tyler Deschamps, Jordan Van Patten, Cody Strange, Kole Davis
- Next team meeting scheduled for: April 15, 2022, 7:00pm (MT)

Progress From Previous Meeting

- Ethan Taylor:
    - Help button/user manual integration (50%)
- Jaden Albrecht:
    - Personal user acceptance (100%)
    - Add search parameters to search page (100%)
- Cody Strange:
    - SPR-5 risk management plan (100%)
- Tyler Deschamps:
    - Backlogs for SPR-5 (100%)
    - SPR-5 chart development (100%)

Topics Discussed

- Functional/non-functional requirements testing
- Function-point analysis
- Team performance review
- User manual integration

Obstacles Encountered

- No obstacles

Finished Items

- Risk management plan for SPR-5
- Pert/Gantt/burn-down charts for SPR-5

Unfinished Items

- SPR-5 document
- Metric reports
- Team performance review
- V&V documents for SPR-5
- Functional/nonfunctional requirements test
- WBS for SPR-5
- Backlogs for SPR-5

- Implementation of approved change order requests from previous sprints
- Usability test
- User acceptance test report

Tasks Until Next Meeting

- User acceptance test report
- WBS for SPR5
- Functional/non-functional tests
- Implementation of previous change order requests

Notes

- No additional notes

Meeting Log#24

Meeting Information

- Team #: T2-002
- Meeting log #: 24
- Current Sprint: SPR5
- Date: April 15, 2022
- Time: 7:00pm – 7:23pm (MT)
- Location: MS Teams (watch video here)
- Attendees: Ethan Taylor, Jaden Albrecht, Tyler Deschamps, Cody Strange, Kole Davis
- Next team meeting scheduled for: April 18, 2022, 7:00pm (MT)

Progress From Previous Meeting

- Ethan Taylor:
    - Help button/user manual integration (100%)
- Cody Strange:
    - All metrics reports (100%)
    - Personal user acceptance test report (100%)

Topics Discussed

- Functional/non-functional requirements testing
- Function-point analysis
- Team performance review
- Metrics reports

Obstacles Encountered

- No obstacles

Finished Items

- Metrics reports
- Help button integration

Unfinished Items

- SPR-5 document

- Team performance review
- V&V documents for SPR-5
- Functional/nonfunctional requirements test
- WBS for SPR-5
- Backlogs for SPR-5
- User acceptance test report

Tasks Until Next Meeting

- User acceptance test report
- WBS for SPR5
- Functional/non-functional tests
- Implementation of previous change order requests
- Usability test
- User acceptance report

Notes

- No additional notes

Meeting Log#25

Meeting Information

- Team #: T2-002
- Meeting log #: 25
- Current Sprint: SPR5
- Date: April 18, 2022
- Time: 7:00pm – 11:30pm (MT)
- Location: MS Teams (watch video)
- Attendees: Ethan Taylor, Jaden Albrecht, Tyler Deschamps, Cody Strange, Kole Davis
- Next team meeting scheduled for: April 20, 2022, 7:00pm (MT)

Progress From Previous Meeting

- Ethan Taylor:
    - Fix search page change order (100%)
- Jaden Albrecht:
    - Team Performance Review (100%)
    - Functional/non-functional requirements document ((100%))
    - SPR-5 meeting logs (100%)
- Cody Strange:
    - All metrics reports (100%)
    - Personal user acceptance test report (100%)
- Tyler Deschamps:
    - SPR-5 chart development (100%)

- Jordan Van Patten:
    - Calculate total hours per sprint for all team members (100%)
    - V&V documents for SPR-5 (100%)
    - Quality assurance checklist (100%)
- Kole Davis:
    - User acceptance report (100%)

Topics Discussed

- Functional/non-functional requirements testing
- Team performance review
- Metrics reports

Obstacles Encountered

- No obstacles

Finished Items

- Search page change order has been fully implemented
- SPR-5 document
- Team performance review
- V&V documents for SPR-5
- Functional/nonfunctional requirements test
- WBS for SPR-5
- Backlogs for SPR-5
- User acceptance test report

Unfinished Items

- No unfinished items

Tasks Until Next Meeting

- No tasks until next meeting

Notes

- No additional notes

SPRINT FOUR

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: V&V documents for sprint four specifically

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-4
Application Name		**Application #**	
Project Manager		**Delivery Manager**	
Date Completed	3/21/2022		

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

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
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?	☒	☐	☐	

2. Do the error messages contain minimal but meaningful information?	☒	☐	☐	
3. Does the application avoid displaying system information in error messages?	☒	☐	☐	
4. Are the error messages kept in a single location?	☐	☒	☐	Error messages are kept in separate locations in the code, and in the application it is shown below the effected area, I.E. if date is incorrect, it will display the error message below the date entry box

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 use an internet applicatin 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)?	☒	☐	☐	
2. Are field values being validated with regular expressions where possible (e.g. validating email addresses and dates for valid formats)?	☒	☐	☐	Date of birth is not currently being validated
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?	☐	☒	☐	Dates are automatically inputted by the code, not by the user, so there is no date validation. DOB is not yet validated and mandated to be in the month/day/year format but will be added in a future 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?	☒	☐	☐	
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?	☒	☐	☐	
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?	☒	☐	☐	
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?	☒	☐	☐	

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

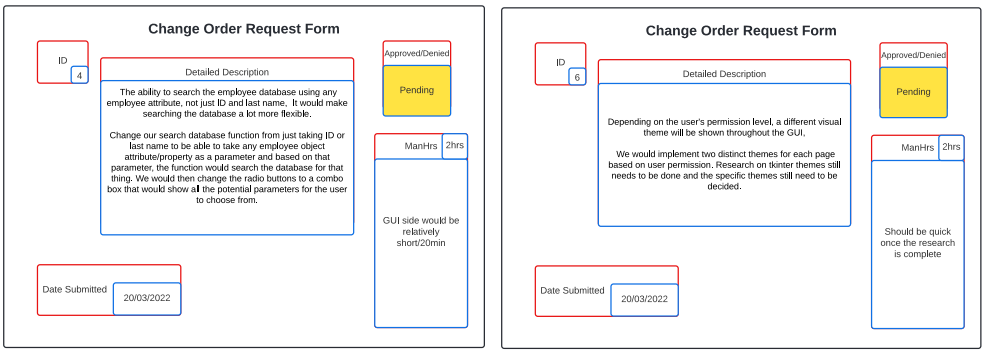
If the answer was 'no' for any of the checklist items above, please explain why in this section.

<< 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)

SPR-4 Change Order Forms

Summary: These are all of the change order forms that have been filled out for this sprint



User Manual

Summary: This manual is designed to help you get started using the EmpDat program and will describe how to use some of its most basic features.

In this manual, you will learn how to:

- Install Python (required to run application!)
- Login using your employee ID and password
- Navigate the user interface
- Search employees in the database
- Edit employee information
- Add/deactivate employees
- Issue payroll information

Note: Both adding/deactivating employees and issuing payroll information requires administrator privileges.

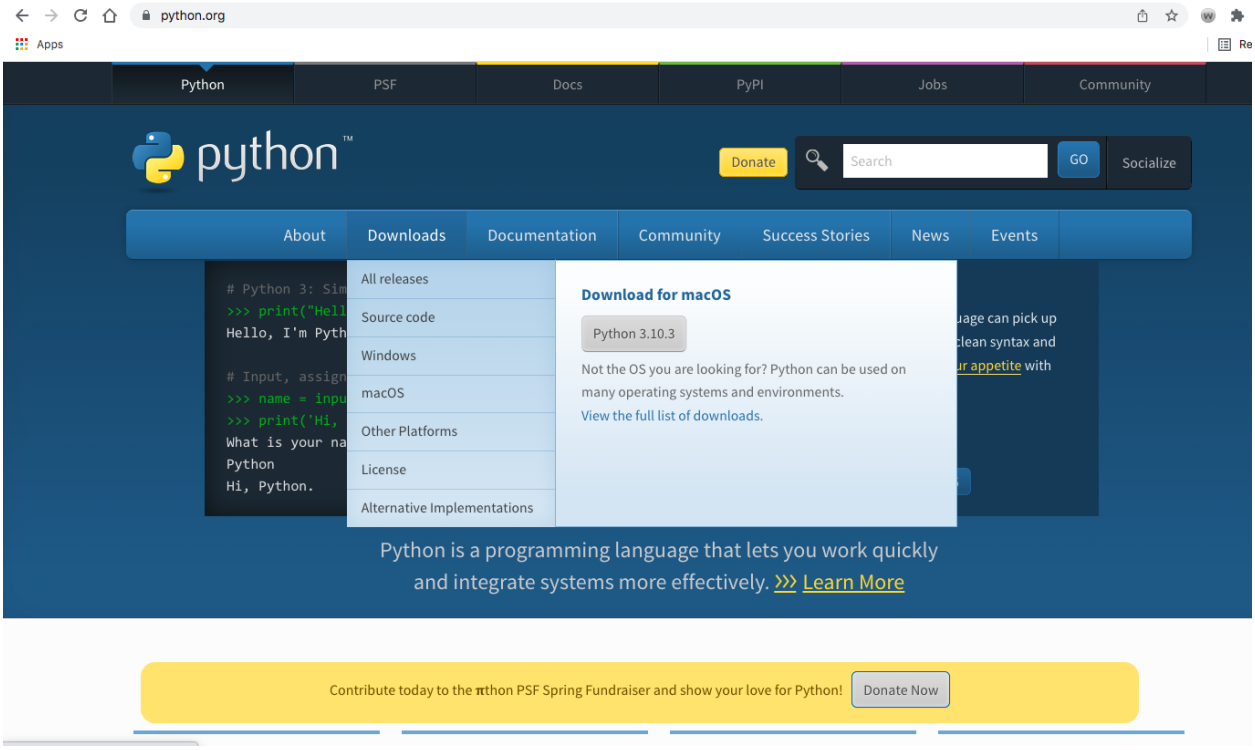
You may refer to this user manual at any time by clicking the “Help” button on the main view page.

1: Installing Python

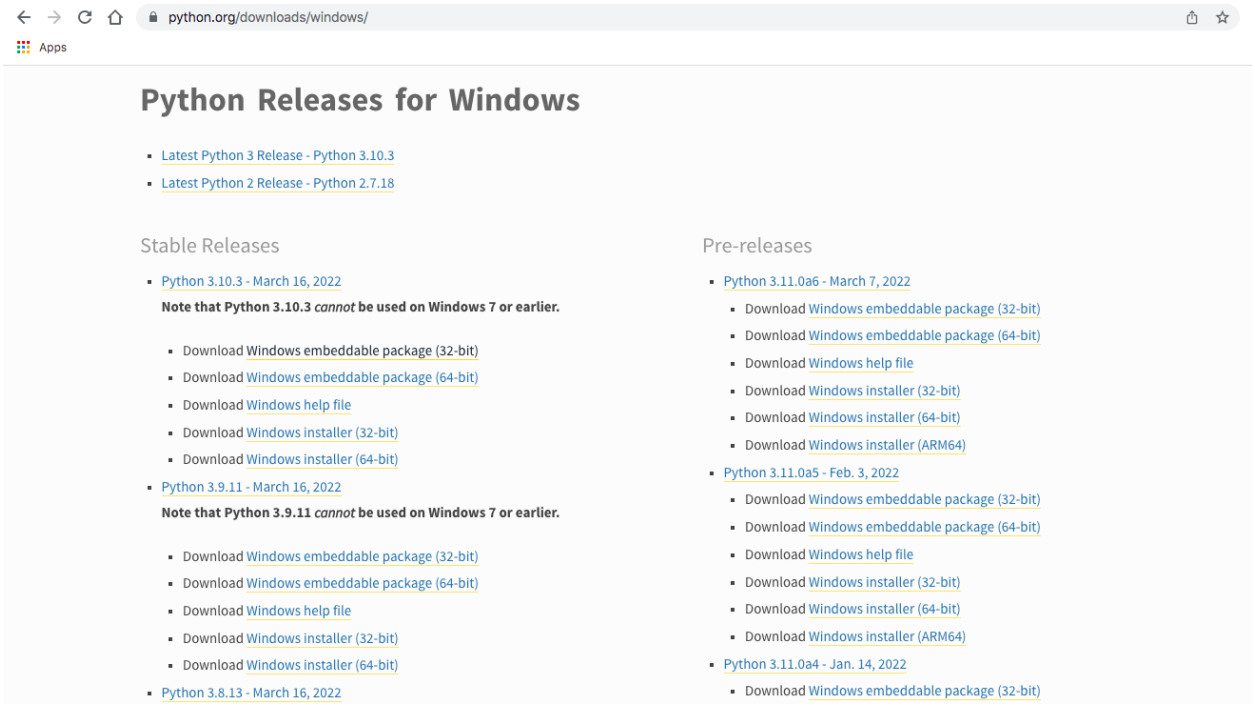
The first step is to install Python onto your computer. If you already have Python installed, then you may skip this step. To install Python, go to the following website:

https://www.python.org/.

Once you are on this page, click on the “Downloads” link and you will be taken to a page which contains all the different Python versions to install. Download the latest version of Python and select the Windows 10 option.



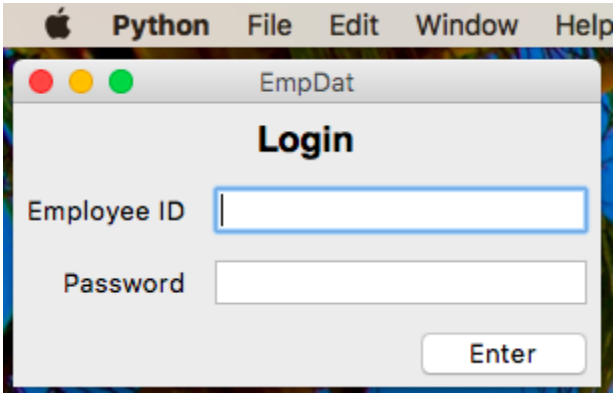
(Image of python website homepage)



(Image of python downloads page)

2: User Login

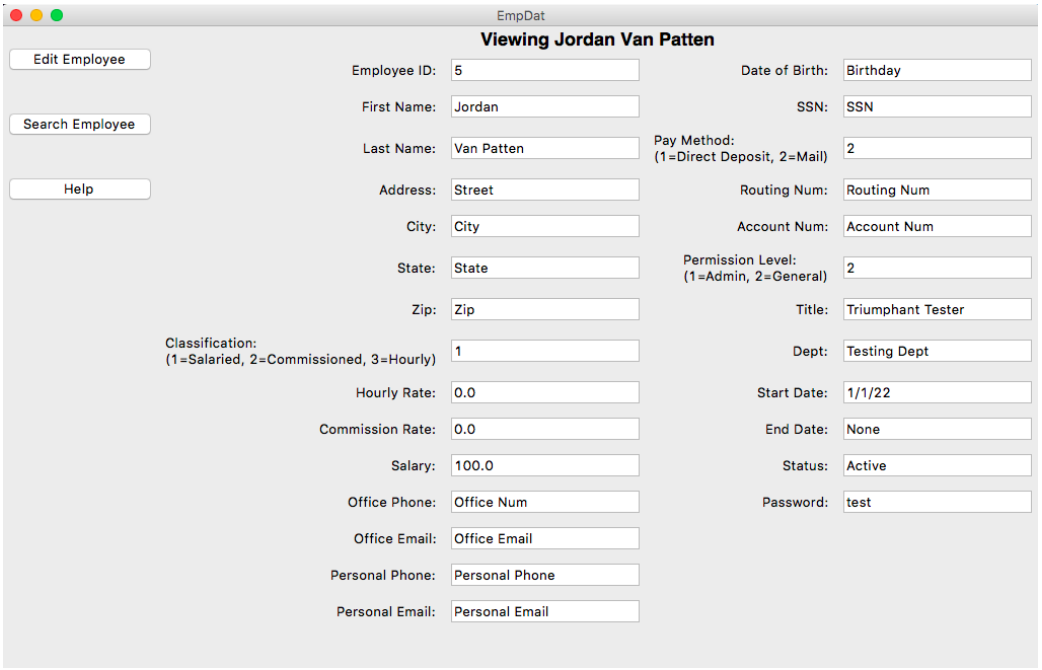
Once you have successfully installed Python and have started the application, you will be taken to the user login page. Here, you will enter your employee ID number and password into the provided fields. Next, click on the "Enter" button and the application will search for your information. If either the ID number or password are incorrect, an error message will appear to notify you that your information is invalid.



(Image of Login screen)

3: Navigating User Interface

Once you have successfully logged in, you will be brought to the main homepage where all your public employee information is displayed (It is worth mentioning that both you and your peers can view the information on your view page). On the left-hand side of the view page, you will see three buttons: "Edit Employee", "Search Employee", and "Help". You can select the "Help" button at any time to view this user manual. The other two buttons are covered in the following sections.



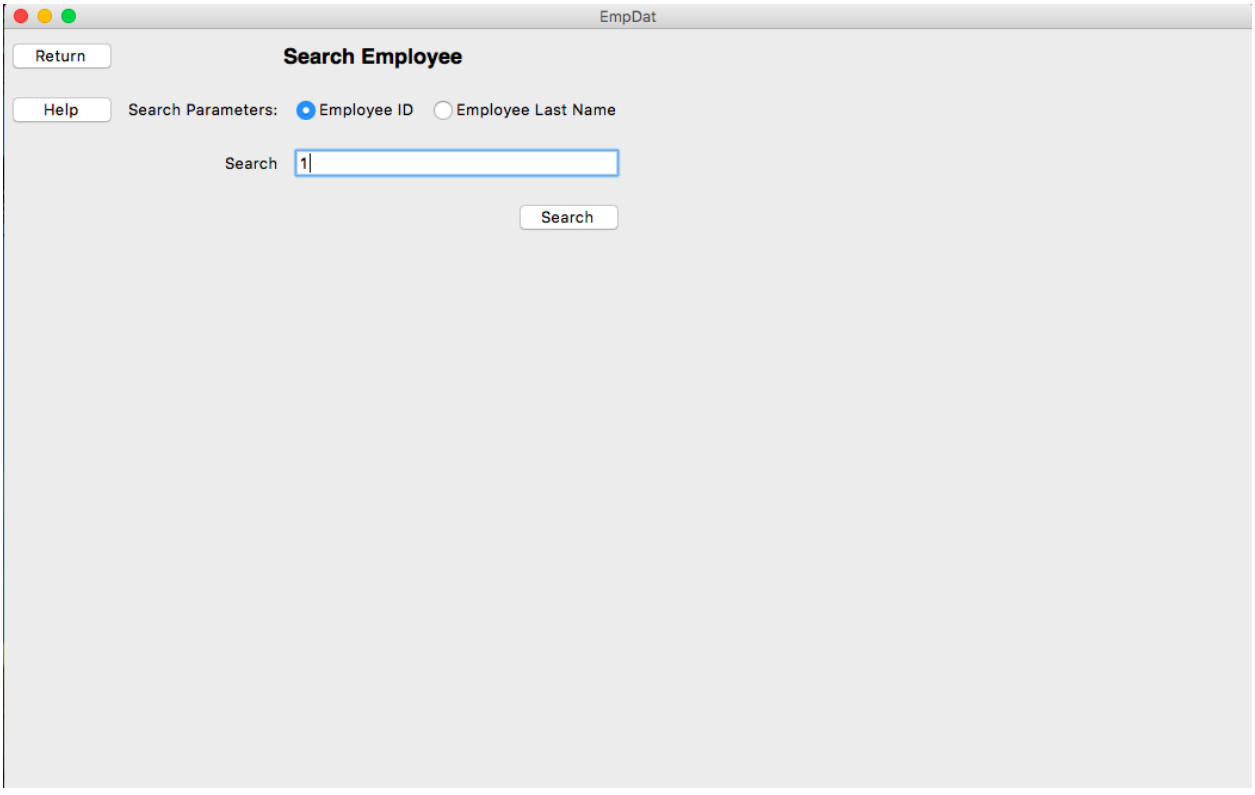
(Image of general view page)

4: Searching Employees

If you wish to view another employee's public information, you can select the "Search Employee" button on the main view page and a new page will appear. In the main section of this new page, you will see a search bar, an "Enter" button, and two search parameter radio buttons. You can search for an employee by typing a search parameter into the search bar and clicking the "Enter" button. Once an employee's information is matched with the search parameters, that employee's view page will appear and allow you to view their public information. If you wish to leave this page, you may click the "Return" button on the left-hand side and you will be returned to the main view page.

4.1: Search Employee By ID

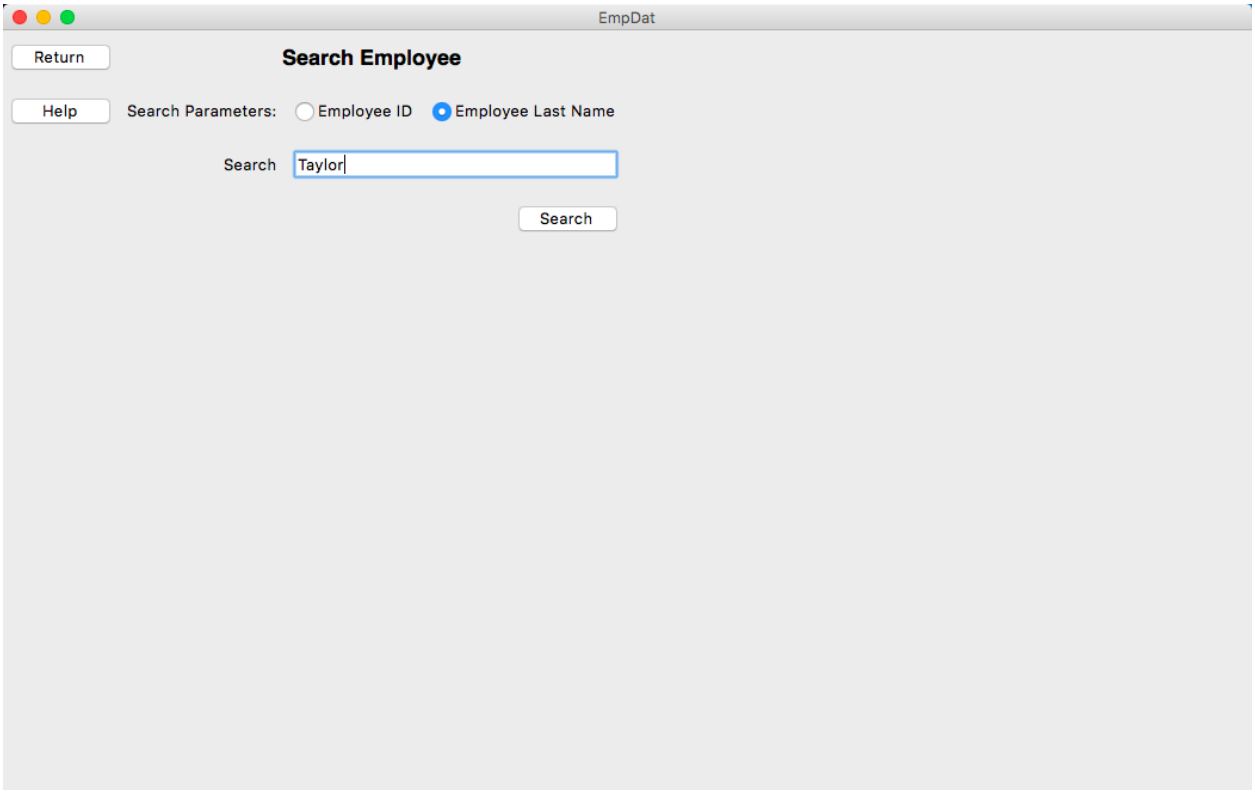
There are two methods for searching employees. The first method is to search for an employee's ID number. First, you must select the radio button labeled "Employee ID" in order for the search to work properly. Now you can type the ID number of the employee you are searching for into the search bar and hit the "Enter" button to begin searching. If the ID number is invalid, the application will notify you and will not provide any results until a valid employee ID is entered.



(Searching employee by their ID)

4.2: Search Employee By Last Name

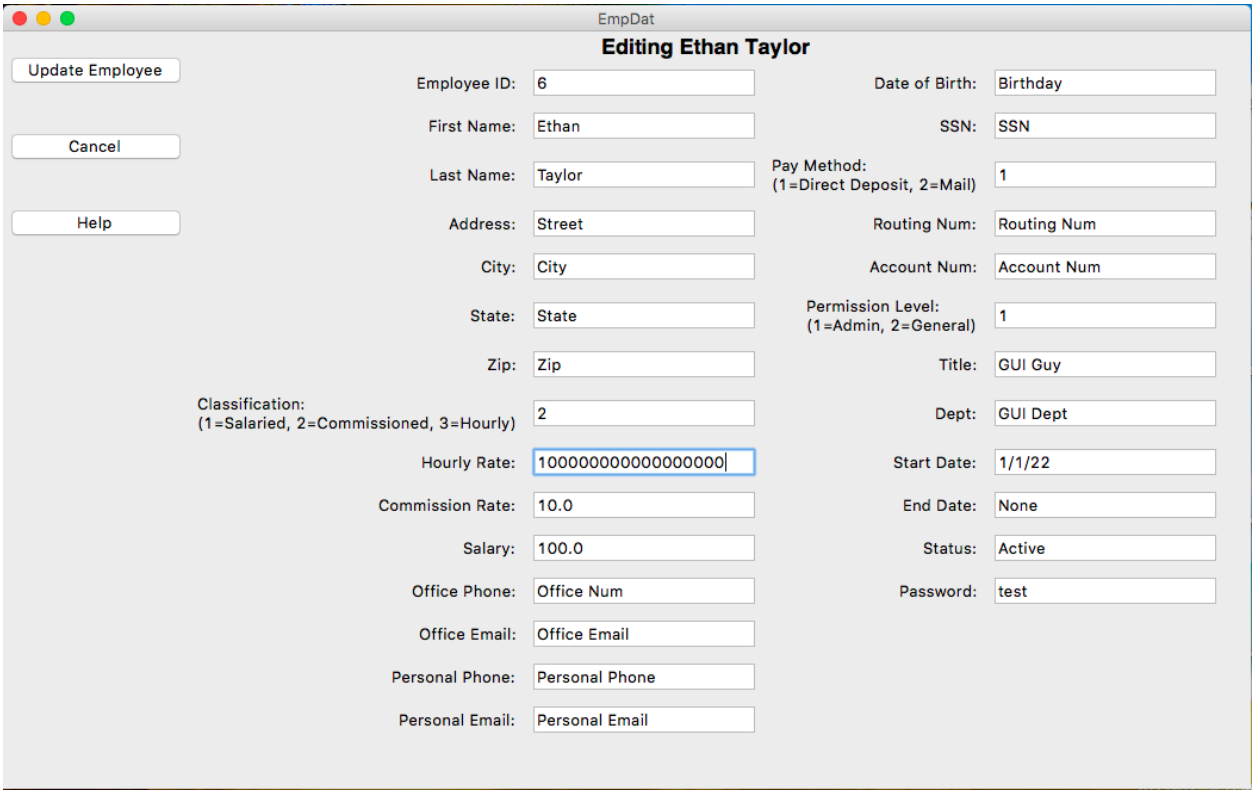
The other method for searching an employee is to search by their last name. Similar to how you search by employee ID numbers, the "Employee Last Name" radio button must be selected before beginning the search. Once this radio button is selected, you may then begin entering the employee's last name into the search bar and hit the "Enter" button to begin the search. If there is no such employee with the last name you entered, the program will notify you and no results will appear.



(Search an employee by their last name)

5: Edit An Employee

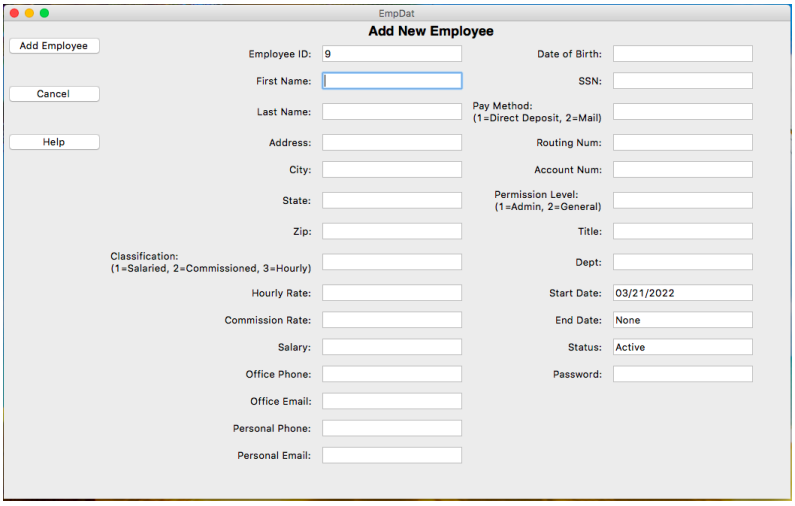
If you wish to change your employee data, you can select the "Edit Employees" button located on the main view page and the page will refresh. Once this has occurred, you are now able to edit any of your own employee data fields. Once you have made your desired changes, you can select the "Update Employee" button located on the left-hand side of the page and you will be returned to the main view page. If you decide to cancel your changes, you may select "Cancel", which is located below the "Update Employee" button, and you will be returned to the main view page. Your employee data has now been saved and the results are displayed.



(Edit employee page)

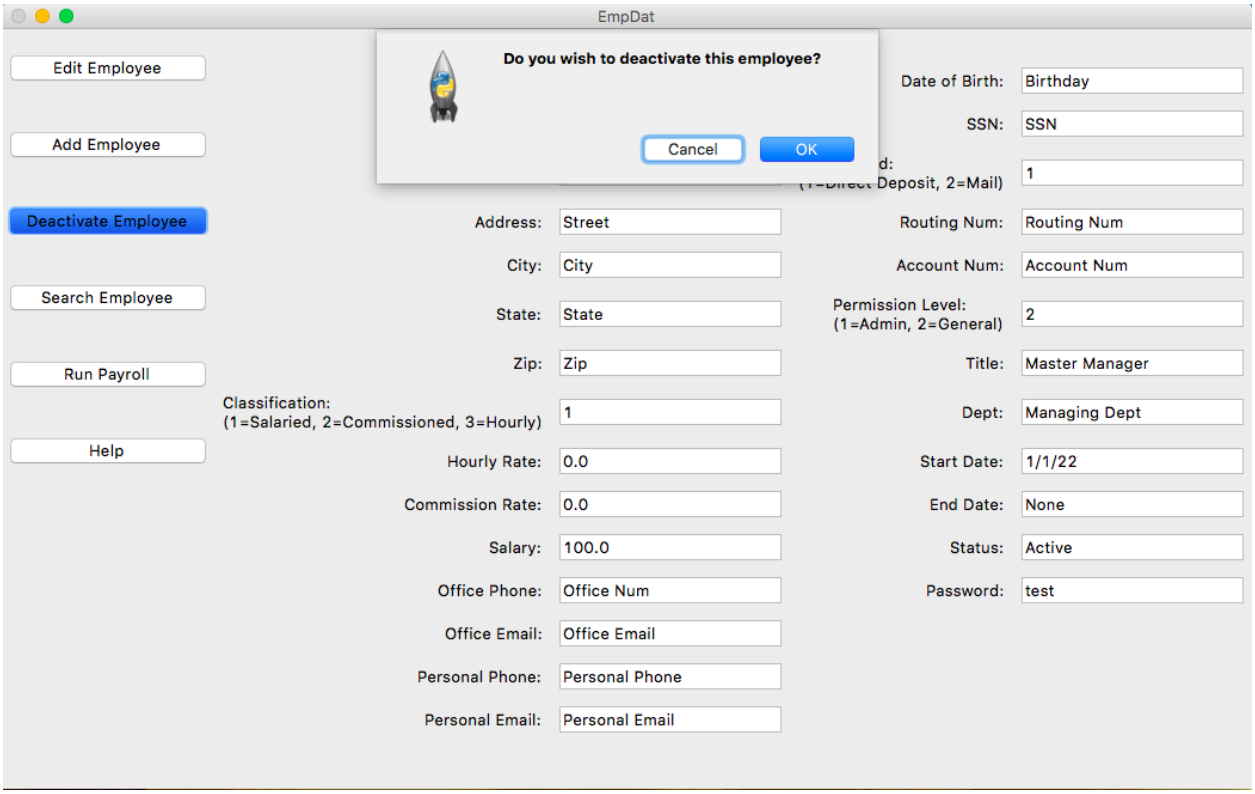
6: Add/Deactivate Employee

If you are an administrator, you have access to additional functions which general employees are not authorized to use. The first of these is adding/dropping employees. You can add an employee by clicking on the "Add Employee" button on the left-hand side of the main view page (if you are given administrator access), and you will be taken to a page similar to the "Edit Employee" page where most data fields are left empty for you to enter a new employee's data. The exception to these data fields is the starting date for the new employee.



[Add an employee (requires administrative privileges)]

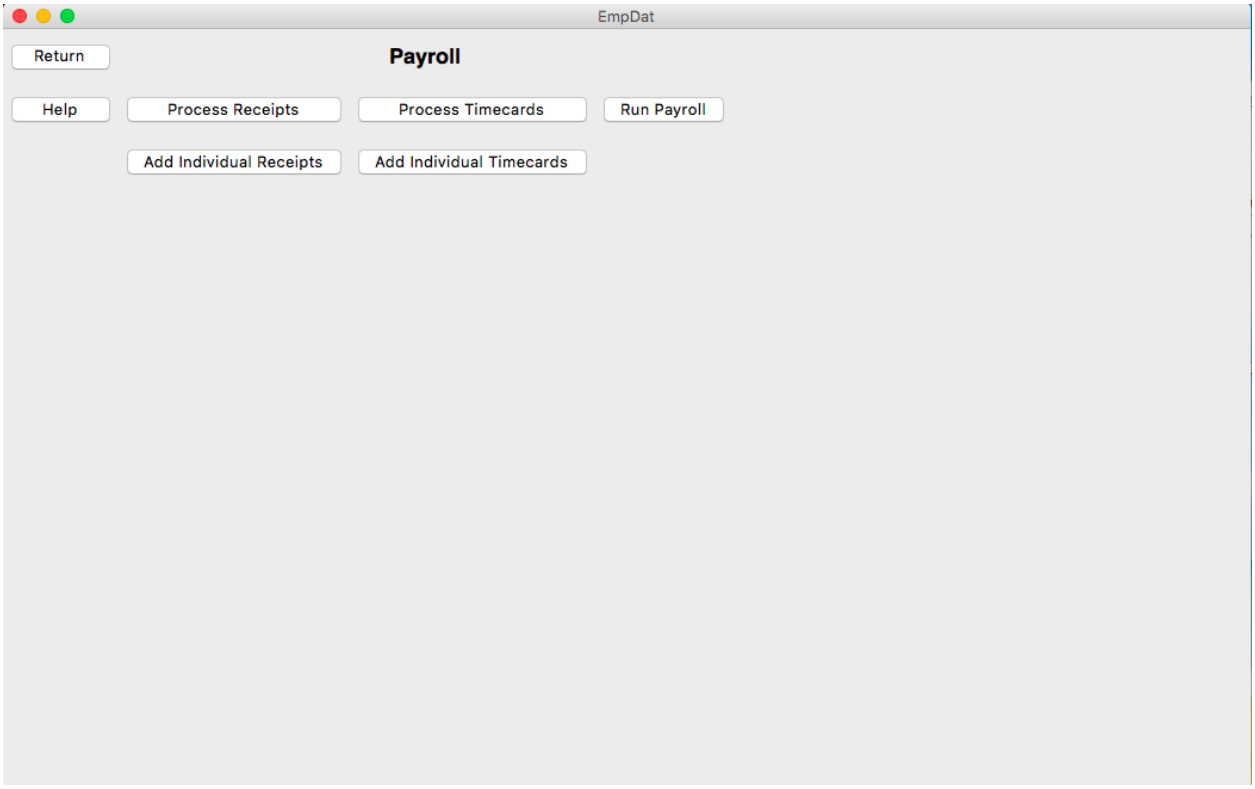
You can also deactivate the employee by clicking the “Deactivate Employee” button and that employee will be flagged as an inactive employee.



[Deactivate an employee (requires administrative privileges)]

7: Running Payroll

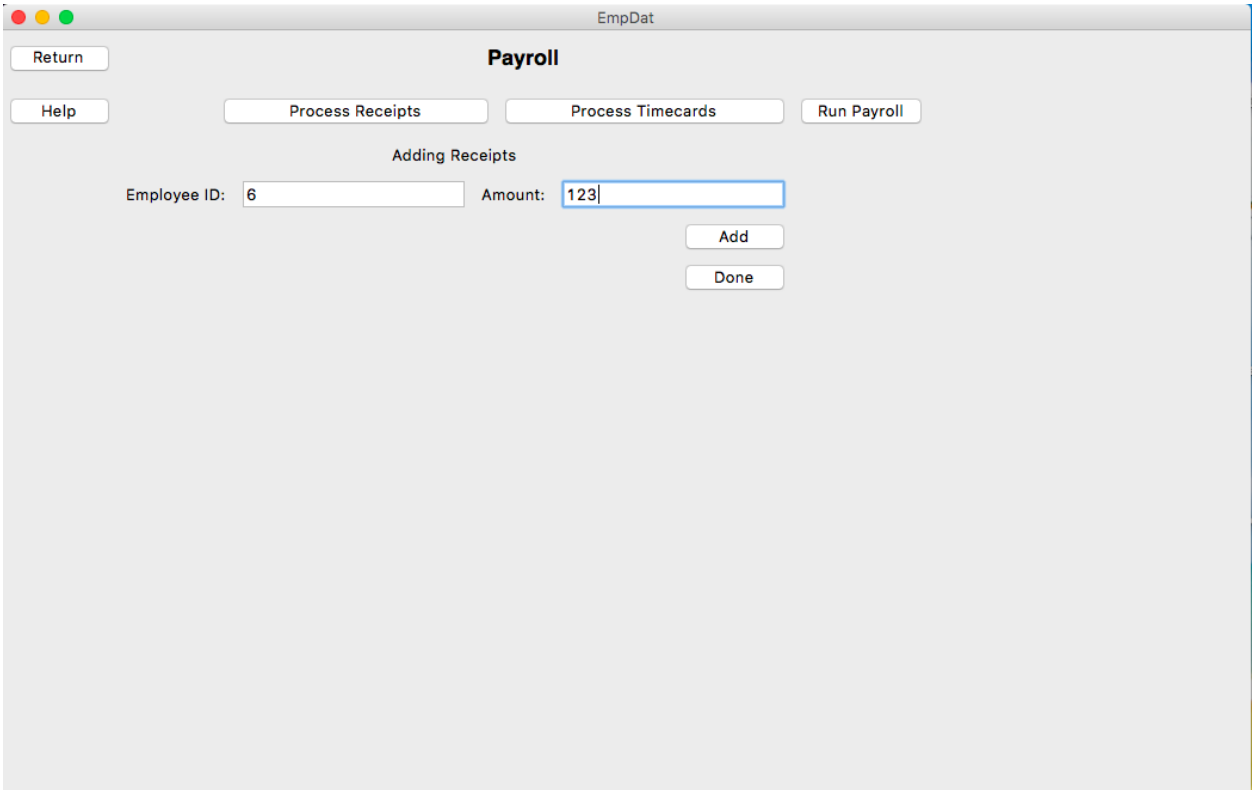
Another administrator-specific function is the ability to generate a payroll report of each employee in the system. The run payroll function generates a pay report using data stored in each employee's profile and exporting the information into a CSV file for your viewing.



(Payroll page)

7.1: Add Receipts

You can also add receipts to a commissioned employee by clicking on the "Add Receipt" button and entering in the amount to store in that receipt. This function adds the receipt to the employee's record and is displayed when that employee's receipts are processed in reports.



(Add receipts to a commissioned employee)

7.2: Add Timecards

You, the administrator, are also given permissions to add timecards to any hourly employees that are currently active in the system. To do this, click on the "Add Timecards" button and enter the time you wish to store in that particular time card. Once the "Enter" button is pressed, the time card is added to the specified employee's record and is displayed when that employee's timecards are processed in payroll reports.

Remember that this user's manual can be accessed at any time when using this application by pressing on the "Help" button" in the left-hand side of the main view page!

<Insert image of edit page>

Plans

SPR-4 Risk Management Plan

Summary: This is the risk management plan that focuses on potential risks on sprint four. It is oriented to risks that we face when writing the code and the finished product.

ID	Category	Description	Consequence	Probability	Impact	Risk Minimalization plan	Contingency plan
1	Login Page	Login page fails to load, or fails to do what it is supposed to	No one would be able to access the site	Low	High	Following the V&V process, along with constant and thorough testing	Shareholders will have contact info of team, fixing login-page will be a high-priorty
2	Merge database	Program fails to read in a previous data base and add it to current one	Customer would be forced to add each employee from the previous data base to the current one by hand	Medium	Medium	Constant communication with shareholder to understand how all previous databases have their information stored	Would have to update the program to be able to read in how the previous database was stored
3	Security breach	Personal data is stolen and/or sensitive data field being altered	Private information being leaked, database no longer having accurate information	Medium	High	Constant testing and V&V, possible use of data encryption software	Will likely have to use more powerful third party software to create stronger layers of protection
4	adding/editing employees	Program fails to update the information of a new employee or add in a new employee	Employee may be getting paid the wrong amount/not getting paid at all	Low	High	Constant testing and following the V&V process.	Shareholders will have contact info of the team, team will quickly diagnose the issue and fix the program
5	Privelges	Employees are allowed access to data fields that their privelges shouldn't allow access to	Employees can edit important information that they shouldn't be allowed to	low	High	Constant testing and following the V&V process.	Shareholders will have contact info of the team, team will quickly diagnose the issue and fix the progam

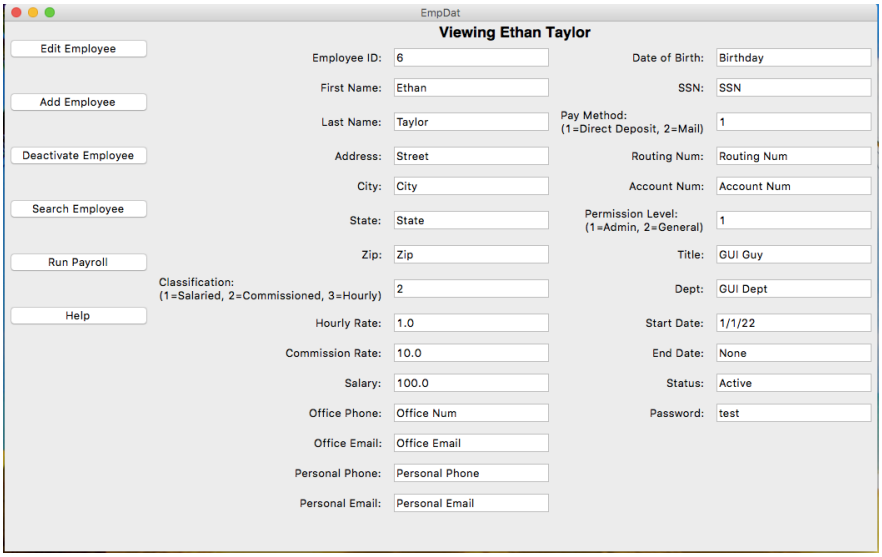
Maintenance Plan

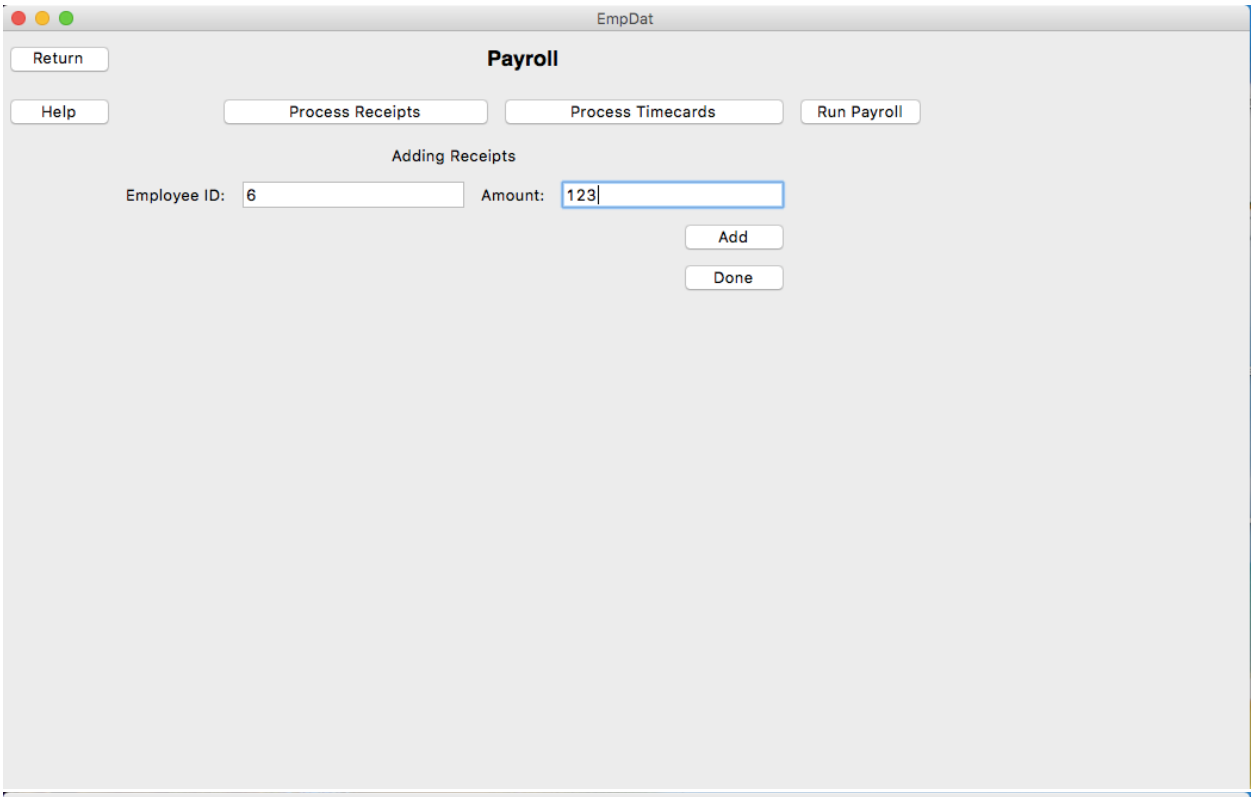
Summary: The maintenance for sprint four will be focused on maintaining code through proper testing and V&V as well as reporting bugs and fixing them quickly based on priority.

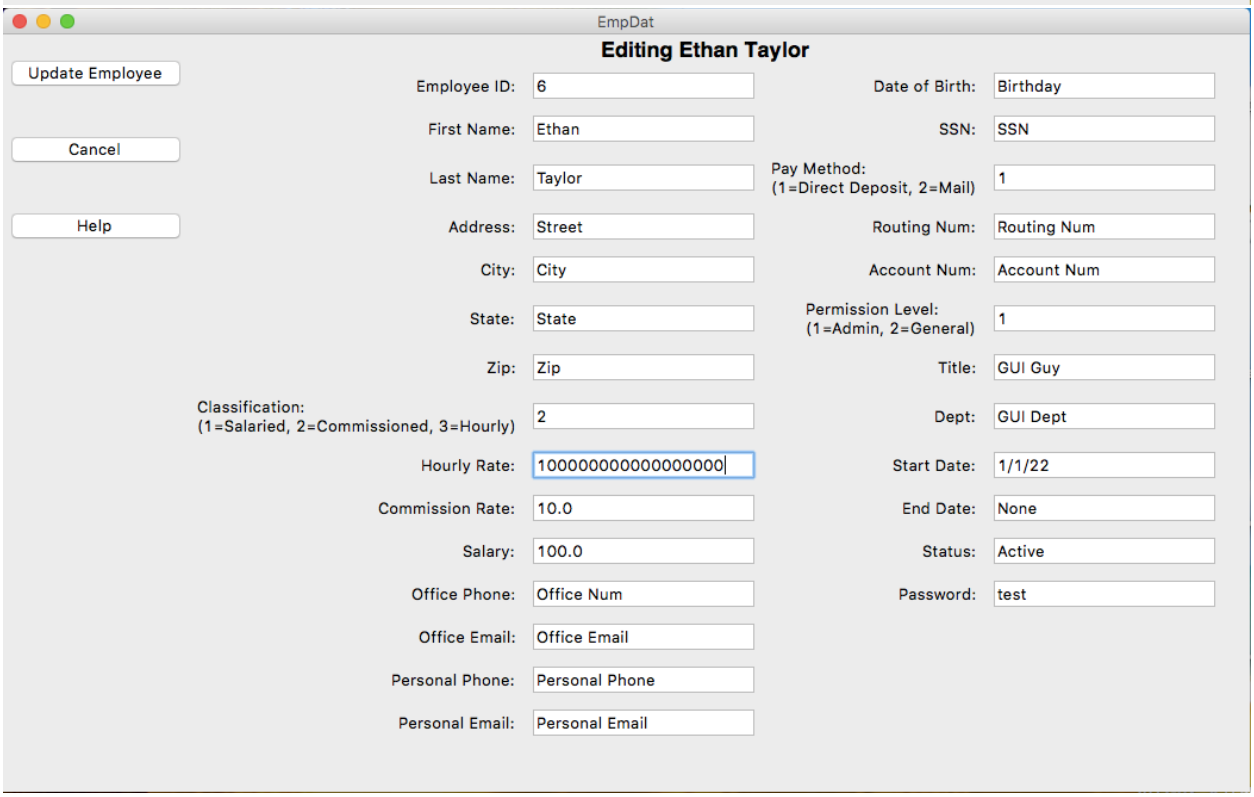
Previous Sprint Items

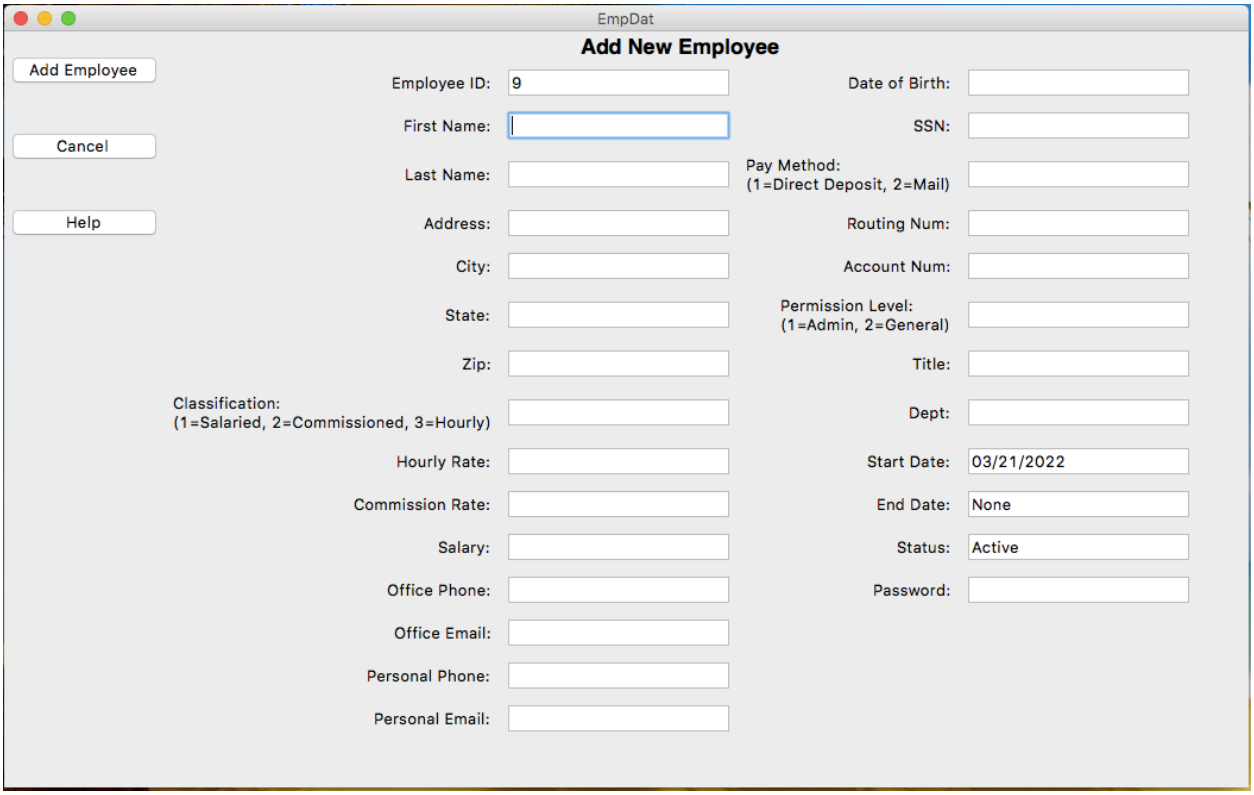
GUI Images

Summary: These are images of the GUI that we were missing in the last sprint




EmpDat
Return
Payroll
Help
Process Receipts
Process Timecards
Run Payroll
Adding Receipts
Employee ID:
6
Amount:
123
Add
Done


EmpDat
Editing Ethan Taylor
Update Employee
Cancel
Help
Employee ID:
6
First Name:
Ethan
Last Name:
Taylor
Address:
Street
City:
City
State:
State
Zip:
Zip
Classification:
(1=Salaried, 2=Commissioned, 3=Hourly)
2
Hourly Rate:
100000000000000000
Commission Rate:
10.0
Salary:
100.0
Office Phone:
Office Num
Office Email:
Office Email
Personal Phone:
Personal Phone
Personal Email:
Personal Email
Date of Birth:
Birthday
SSN:
SSN
Pay Method:
(1=Direct Deposit, 2=Mail)
1
Routing Num:
Routing Num
Account Num:
Account Num
Permission Level:
(1=Admin, 2=General)
1
Title:
GUI Guy
Dept:
GUI Dept
Start Date:
1/1/22
End Date:
None
Status:
Active
Password:
test


EmpDat
Add New Employee
Add Employee
Cancel
Help
Employee ID:
9
First Name:
Last Name:
Address:
City:
State:
Zip:
Classification:
(1=Salaried, 2=Commissioned, 3=Hourly)
Hourly Rate:
Commission Rate:
Salary:
Office Phone:
Office Email:
Personal Phone:
Personal Email:
Date of Birth:
SSN:
Pay Method:
(1=Direct Deposit, 2=Mail)
Routing Num:
Account Num:
Permission Level:
(1=Admin, 2=General)
Title:
Dept:
Start Date:
03/21/2022
End Date:
None
Status:
Active
Password:

Deployment

Scope

Project Justification

Summary: The customer needs a program that can store a database of employees and their relevant information. Along with the ability to add and edit employees in the database and let employees view and edit some of their own information.

Product Scope

Summary: We need to deliver a program that uses Tkinter to create the GUI so that the customer has the ability to interact with the database. The GUI needs to have an edit page, view page, add page, and show different information based on whether the employee is and admin or a general user.

Acceptance Criteria

Summary: We agree that this is delivered we can run the program through the command line and see the main deliverables. There should be no defects that prevent the main functionality of the product. The program should work as described in the requirements specifications. Customer should provide a sign-off on the final results

Deliverables

Summary: list of all of the deliverables that the product will produce

- user manual
- login page
- add employee page
- edit employee page
- view employee page
- pay employee page
- search employee page

Assumptions

Summary: uncertainties that haven't been 100% clarified

- the customer has access to the correct version of python
- the customer will be using windows

Cutover

Staged Development

Summary: We plan on reducing the chance of errors in deployment by using staged deployment. We have a brand-new account created on a surface pro 7 that we will be able to download everything from scratch to see if it works on a device that hasn't been used for programming before and the exact steps that it'll take to make it work.

Gradual Cutover

Summary: We do NOT plan on applying gradual cutover as our product is small enough that it should be possible to thoroughly test it and safely move every user over to it from the old product.

Incremental Deployment

Summary: Incremental deployment would not work with how monolithic application. We would release the entire product at once.

Parallel Testing

Summary: It would be relatively easy to apply parallel testing, we would copy the database from the old product and read it into the new one. We would have both systems running for a couple of days, so that we can guarantee that the program works as it should before we remove the old product. It does create additional work for a couple of days for the employees, but it is a safer way to go about deploying the product.

Deployment Plan

Deployment Tasks

Summary: Tasks required to perform a successful deployment

- Hardware – computers with Windows 10 installed
- Documentation – user manual
- Training – explaining to the users how the program works, and making sure they are confident in its use
- Database – ensuring the shareholder's database is compatible with the new software
- Software – the product that we are creating for the shareholder

Rollback Plan

Summary: The plan if our product shows to have glaring functional issues and requires us to reverse our deployment plan, we will simply remove the current version of the product from the user's computers and have them continue using the old software until fix our software

Point of No Return

Summary: After sprint six, we would have reached the point of no return for any major functional errors that would require us to virtually restart development. At that time we have to either decide to push forward with what we have or to scrap the project completely.

Testing

Testing

Test Plan

Summary: Our tests were created through pytest using assertion tests, they tested each page to make sure that they were functional and followed the requirements.

```
import EmpDat_v5 #code file to test
from employee_v3 import * #code for employee
import pytest    #testing framework
from tkinter import ttk

# emp_dict = {}
# emp1 = Employee("1", "Jaden", "Albrecht", "Street", "City", "State", "Zip", "1", "1",
#             100.00, 10.00, 1.00, "Routing Num", "Account Num", "Office Num",
#             "Personal Phone", "Office Email", "Personal Email", "Birthday",
#             "SSN", False, "Master Manager", "Managing Dept", "1/1/22", None, True,
#             "test")
# emp2 = Employee("2", "Cody", "Strange", "Street", "City", "State", "Zip", "2", "2",
#             100.00, 10.00, 1.00, "Routing Num", "Account Num", "Office Num",
#             "Personal Phone", "Office Email", "Personal Email", "Birthday",
#             "SSN", False, "Super Scribe", "Writing Dept", "1/1/22", None, True,
#             "test")
# emp3 = Employee("3", "Tyler", "Deschamp", "Street", "City", "State", "Zip", "3", "1",
#             100.00, 10.00, 1.00, "Routing Num", "Account Num", "Office Num",
#             "Personal Phone", "Office Email", "Personal Email", "Birthday",
#             "SSN", False, "Chart Champion", "Documenting Dept", "1/1/22", None, True,
#             "test")
# emp4 = Employee("4", "Jordan", "Van Patten", "Street", "City", "State", "Zip", "1", "2",
#             100.00, 10.00, 1.00, "Routing Num", "Account Num", "Office Num",
#             "Personal Phone", "Office Email", "Personal Email", "Birthday",
#             "SSN", False, "Triumphant Tester", "Testing Dept", "1/1/22", None, True,
#             "test")
# emp5 = Employee("5", "Ethan", "Taylor", "Street", "City", "State", "Zip", "2", "1",
#             100.00, 10.00, 1.00, "Routing Num", "Account Num", "Office Num",
#             "Personal Phone", "Office Email", "Personal Email", "Birthday",
#             "SSN", True, "GUI Guy", "GUI Dept", "1/1/22", None, True,
```

```
#           "test")
# emp6 = Employee("6", "Etha", "Taylor", "Street", "City", "State", "Zip", "2", "1",
#           100.00, 10.00, 1.00, "Routing Num", "Account Num", "Office Num",
#           "Personal Phone", "Office Email", "Personal Email", "Birthday",
#           "SSN", True, "GUI Guy", "GUI Dept", "1/1/22", None, True,
#           "test")
# emp7 = Employee("7", "Eth", "Taylor", "Street", "City", "State", "Zip", "2", "1",
#           100.00, 10.00, 1.00, "Routing Num", "Account Num", "Office Num",
#           "Personal Phone", "Office Email", "Personal Email", "Birthday",
#           "SSN", True, "GUI Guy", "GUI Dept", "1/1/22", None, True,
#           "test")
# emp8 = Employee("8", "Et", "Taylor", "Street", "City", "State", "Zip", "2", "1",
#           100.00, 10.00, 1.00, "Routing Num", "Account Num", "Office Num",
#           "Personal Phone", "Office Email", "Personal Email", "Birthday",
#           "SSN", True, "GUI Guy", "GUI Dept", "1/1/22", None, True,
#           "test")

# emp_dict[emp1.get_id()] = emp1
# emp_dict[emp2.get_id()] = emp2
# emp_dict[emp3.get_id()] = emp3
# emp_dict[emp4.get_id()] = emp4
# emp_dict[emp5.get_id()] = emp5
# emp_dict[emp6.get_id()] = emp6
# emp_dict[emp7.get_id()] = emp7
# emp_dict[emp8.get_id()] = emp8
# x = EmpDat_v5.EmpApp(emp_dict)

emps = get_db("database/employees.json")
emp_dict = {}

for emp in emps:
    emp_dict[str(emp["id"])] = Employee(str(emp["id"]), emp["first_name"], emp["last_name"],
emp["address"], emp["city"],
                                        emp["state"] ,emp["zip"], emp["classification"], emp["pay_method"],
emp["salary"],
                                        emp["commission"], emp["hourly"], emp["routing_num"],
emp["account_num"],
                                        emp["office_phone"], emp["personal_phone"], emp["office_email"],
emp["personal_email"],
                                        emp["dob"], emp["ssn"], emp["admin"], emp["title"], emp["dept"],
emp["start"]
```

Test Report

Summary: All of the tests have been ran and have passed with 100% completion.

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	100%	
test_incorrect	tests that using the incorrect username but correct password does trigger the "Incorrect password or ID" message	8	100%	
test_incorrectP	tests that using the incorrect password but correct username does trigger the "Incorrect password or ID" message	11	100%	

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	100%	
test_AdminPass	tests that the information shown on the admin users viewpage matches their information in the database	4	100%	

Edit Page				
Name	Description	Count	%Passed	P/F
test_edit	Tests that all fields are valid, when valid inputs are given	6	100%	
test_edit_invalid_variable	Tests that if all fields are invalid, that the error will pop up and variables will not have changed, it then tries to change to all valid variables	1	100%	

Add Page				
Name	Description	Count	%Passed	P/F
test_add	Tests that fields are registered as valid when valid results are input	1	100%	
test_add_incorrect	Tests that fields are registered as invalid when invalid results are input	1	100%	

Search Page				
Name	Description	Count	%Passed	P/F
test_search	searches for a user that should be in the datbase	5	100%	

Pay Page				
Name	Description	Count	%Passed	P/F
test_pay	Tests that employee has the correct ID, classificaton and pay	3	100%	

Usability Test

Summary: Cody Strange ran the usability test because he has the least amount of knowledge in dealing with the code.

- Test#1: I was unable to get the program running, I downloaded the zip file and tried to run every file that was there, at best they did nothing I could see. At worst they through errors. I was unable to get it running and will need to discuss this with the team to find out why it is not working.
- Test#2: I was able to get the program to work after adjusting one of the paths that are called in the code, I was brought to the login page and typed in the proper id and password and that worked fine. It instantly brought me to the Add Employee page, and only had a "Add Employee" button, "Cancel" button, and "Help" button. The help button through a message that said it was a work in progress. But when I tried to add an employee it would just say that I was missing fields even though I wasn't. Once I hit the cancel button something very odd happened. Most of the fields disappeared and the ones that remained could not be edited. The page was still called "Add Employee", the "cancel" button does not do anything. And I tried to exit out of the program but the "X" in the top right corner doesn't work.
- Test#3: Cannot find out which fields that I am typing in are invalid, seems to require me to fill out an hourly wage, salary, and commission. Even though it asks me to decide which one to use. Also has me fill out the end date even though the employee is just starting.
- Test#4: Using the Thonny IDE we were able to get the program running properly, there were some fields that didn't let the user know when they typed the incorrect information, but besides some minor inconveniences it was intuitive enough, besides that the payroll page should be properly explained in the user manual.

Bug Tracking

Bug Tracking Plan

Summary: We are using backlog to keep track of bugs that appear in the program. When a new bug is discovered, the programmer/tester creates and 'issue' on backlog where they give the bug a brief name and detailed description of what the bug causes. It is then assigned to someone to fix and a due date to fix by and labels it so that it appears in the 'Open' section of the board. The assignee then moves it to the 'In Progress' section while he is fixing it. Once he fixes the bug it is then moved to the 'Resolved' section and the person who fixed it leaves a comment giving a brief description on how they fixed it.

Bug Report

Summary: The bug report has been slightly adjusted from last sprint, with the "Severity" category being renamed to "Priority.

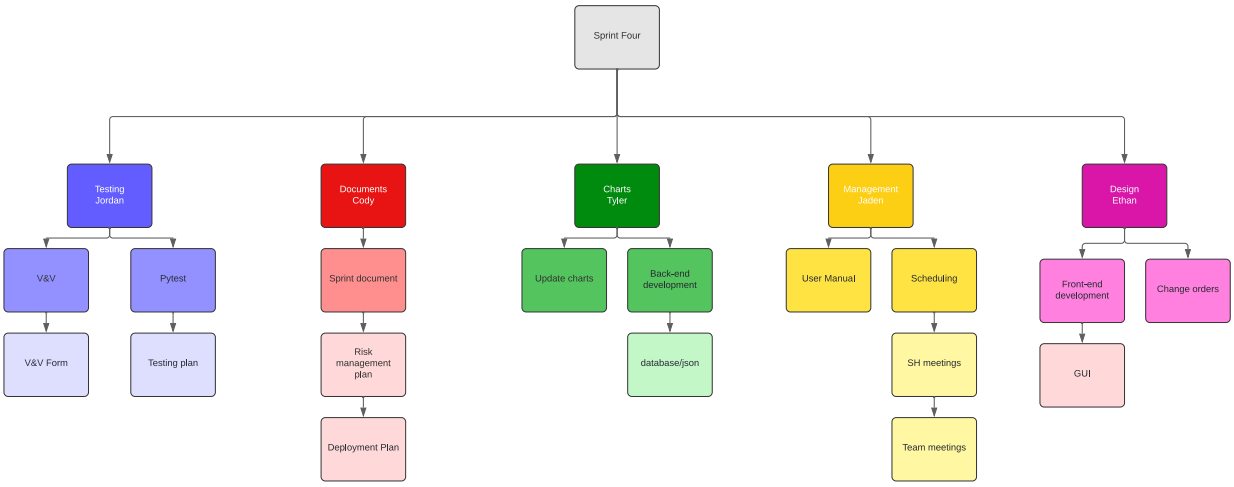
Bug ID	Bug Found Date	Description	Priority	Start Date	Note	End date
1	17/03/2022	When a non-admin employee tries to search for another employee, the program crashes. The error says: "AttributeError: 'Ed@it_Page' object has no attribute 'status_entry'".	high	18/03/2022	FIXED: Bug was caused by tkinter attempting to change a widget that hadn't yet been created (it would be created in admin mode). Resolved by placing that area of the Edit page inside a 'if user.is_admin() == True' branch	18/03/2022

Charts/Forms

SPR-4 Work Breakdown Structure

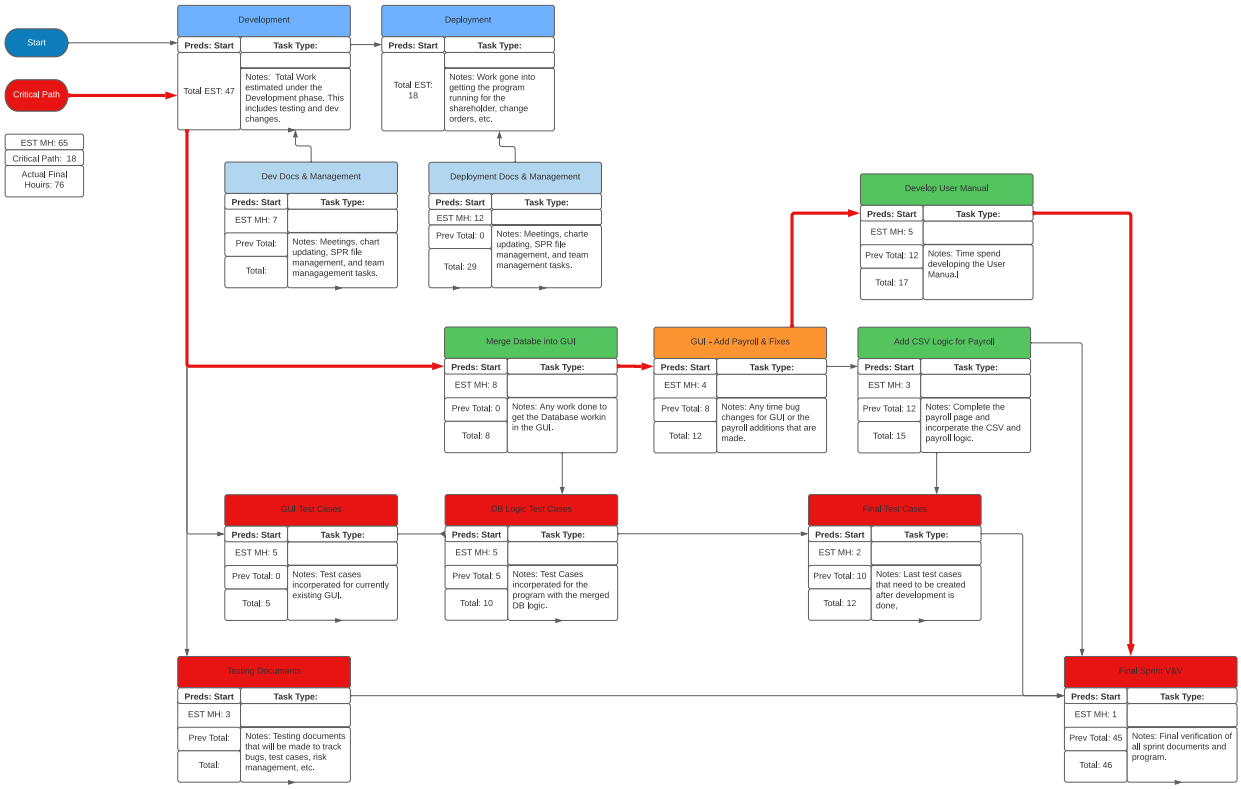
Chart

Summary: This WBS chart shows the general responsibilities that everyone in the sprint were in charge of



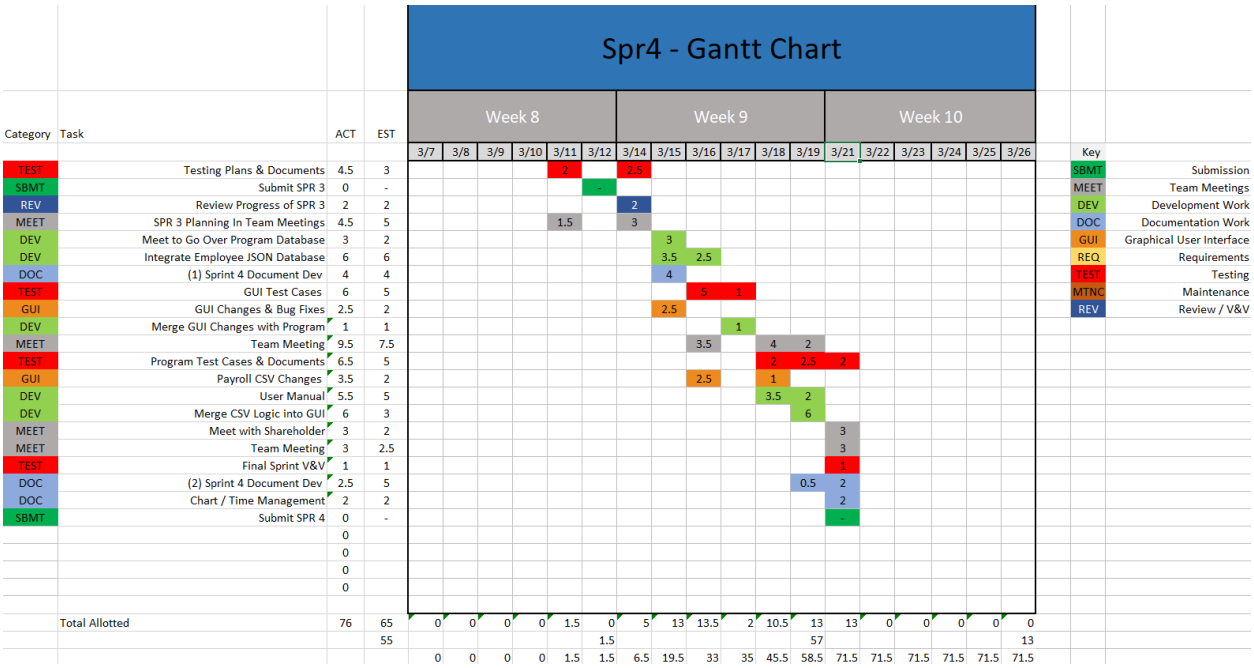
SPR-4 Pert Chart

Chart



SPR-4 Gantt Chart

Chart

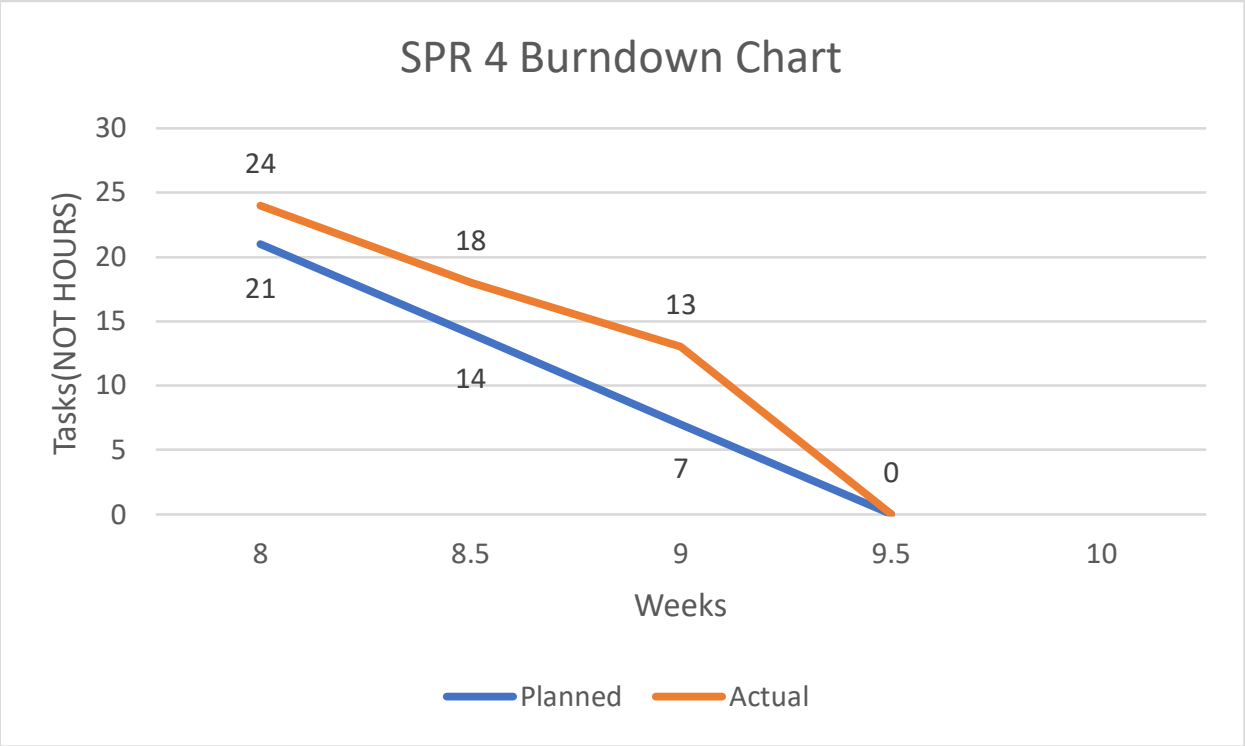


Category	Task	ACT	EST	3/7	3/8	3/9	3/10	3/11	3/12	3/14	3/15	3/16	3/17	3/18	3/19	3/21	3/22	3/23	3/24	3/25	3/26
TEST	Testing Plans & Documents	4.5	3					2		2.5											
SBMT	Submit SPR 3	0	-						-												
REV	Review Progress of SPR 3	2	2							2											
MEET	SPR 3 Planning In Team Meetings	4.5	5					1.5		3											
DEV	Meet to Go Over Program Database	3	2								3										
DEV	Integrate Employee JSON Database	6	6								3.5	2.5									
DOC	(1) Sprint 4 Document Dev	4	4								4										
TEST	GUI Test Cases	6	5									5	1								
GUI	GUI Changes & Bug Fixes	2.5	2								2.5										
DEV	Merge GUI Changes with Program	1	1										1								
MEET	Team Meeting	9.5	7.5									3.5		4	2						
TEST	Program Test Cases & Documents	6.5	5											2	2.5	2					
GUI	Payroll CSV Changes	3.5	2									2.5		1							
DEV	User Manual	5.5	5											3.5	2						
DEV	Merge CSV Logic into GUI	6	3												6						
MEET	Meet with Shareholder	3	2													3					
MEET	Team Meeting	3	2.5													3					
TEST	Final Sprint V&V	1	1													1					
DOC	(2) Sprint 4 Document Dev	2.5	5												0.5	2					
DOC	Chart / Time Management	2	2													2					
SBMT	Submit SPR 4	0	-													-					
		0																			
		0																			
		0																			
		0																			
	Total Allotted	76	65	0	0	0	0	1.5	0	5	13	13.5	2	10.5	13	13	0	0	0	0	0
			55						1.5						57						13
				0	0	0	0	1.5	1.5	6.5	19.5	33	35	45.5	58.5	71.5	71.5	71.5	71.5	71.5	71.5

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

Chart



Meeting Logs

Meeting Log#14

Meeting Information

- Team #: T2-002
- Meeting log #: 14
- Current Sprint: SPR4
- Date: March 14, 2022
- Time: 7:00pm – 8:04pm (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 16, 2022, 3:50pm (MT)

Progress From Previous Meeting

- Ethan Taylor:
  - COMPLETED ALL TASKS FOR SPR3
- Jaden Albrecht:
  - Keep meeting logs up to date (100%)
  - Change order request form (100%)
- Cody Strange
  - SPR3 document (100%)
  - Testing documents (100%)
  - Change request board (100%)
  - Risk management plan (100%)
- Tyler Deschamps:
  - Update pert/Gantt/burn-down charts (100%)
- Jordan Van Patten:
  - Research PyTest
  - Research Black/Pytest
  - Read Chapter 7 (100%)
  - V&V documents (100%)
  - Test cases (75%))
  - QA plan (100%)

Topics Discussed

- Changes to final SPR3 document
- User manual
- Testing plans
- Maintenance plans
- Usability test cases
- Bug tracking method
- Assign role for new team member in SPR5
- Refining database functionality with GUI
- Next shareholder meeting

Obstacles Encountered

- No obstacles

Finished Items

- All charts are up to date
- Meeting logs are up to date
- SPR3 document
- Testing pseudocode
- Assigned new team member as a coder/tester

Unfinished Items

- Wired GUIs with database
- Usability test cases
- Tested all components using PyTest
- Testing documents
- V&V documents for SPR4
- Risk management plan for SPR4
- Maintenance plan
- WBS for SPR4
- User manual
- Schedule next shareholder meeting

Tasks Until Next Meeting

- V&V documents for SPR4
- Wire all GUIs to database
- Implement field validation code into GUI
- Risk management plan
- Testing documents
- Usability test cases
- Maintenance plan

Notes

- At the time of this meeting, we only had worked on tasks assigned from SPR3.

Meeting Log#15

Meeting Information

- Team #: T2-002
- Meeting log #: 15
- Current Sprint: SPR4
- Date: March 16, 2022
- Time: 3:50pm – 4:34pm (MT)
- Location: MS Teams (Watch video here)
- Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Jordan Van Patten
- Next team meeting scheduled for: March 18, 2022, 7:00pm (MT)

Progress From Previous Meeting

- Ethan Taylor:
    - Implement add/edit validation code into GUI (100%)
    - Update payroll page (100%)
    - Tkinter research

- Jaden Albrecht:
    - Keep meeting logs up to date (100%)
    - Chapter 8 (100%)

- Cody Strange
    - WBS for SPR4 (100%)
    - Research requirements for SPR4 (100%)

- Tyler Deschamps:
    - Update pert/Gantt/burn-down charts (100%)
    - Report receipts (100%)
    - Implement GUI and database logic (100%)
    - Merge database with Emp_Dat_V3 (100%)

- Jordan Van Patten:
    - Implement validation code to GUI (100%)
    - Test cases for search page (100%)

Topics Discussed
- User manual
- Testing plans
- Maintenance plans
- Usability test cases
- Refining database functionality with GUI

Obstacles Encountered
- No obstacles

Finished Items
- All charts are up to date
- Meeting logs are up to date
- Implementation of add/edit validation code
- Updated payroll page to work with JSON database
- WBS for SPR4
- Merge database with Emp_Dat_V3
- GUI and database logic implementation

Report receipts
- Test cases for search page

Unfinished Items
- Test cases for payroll page
- Testing documents
- V&V documents for SPR4
- Risk management plan for SPR4
- Maintenance plan
- User manual
- Schedule next shareholder meeting

Tasks Until Next Meeting

- V&V documents for SPR4
- Risk management plan
- Testing documents
- Usability test cases
- Maintenance plan

Notes
- No additional notes

Meeting Log#16

Meeting Information
- Team #: T2-002
- Meeting log #: 16
- Current Sprint: SPR4
- Date: March 18, 2022
- Time: 7:00pm – 7:45pm (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 21, 2022, 7:00pm (MT)

Progress From Previous Meeting
- Ethan Taylor:
    - Integrate database with pay page updates (100%)

- Jaden Albrecht:
    - Keep meeting logs up to date (100%)
    - User manual (80%)
    - Scheduled next shareholder meeting (100%)

- Cody Strange
    - Usability tests (30%)
    - Risk management for SPR4 (100%)
    - Maintenance plans (100%)
    - Deployment plan (100%)
    - SPR4 document (70%)

- Tyler Deschamps:
    - Update pert/Gantt/burn-down charts (100%)
    - Merge database with Emp_Dat_V3 (100%)
    - Payroll and CSV integration (100%)

- Jordan Van Patten:
    - Make Pytest documents for payroll page (60%)

Topics Discussed
- User manual
- Testing plans
- Maintenance plans

- Usability test cases
- Next shareholder meeting

Obstacles Encountered
- No obstacles

Finished Items
- All charts are up to date
- Meeting logs are up to date
- Integrate database with pay page updates
- Risk management plan for SPR4
- Maintenance plans
- Deployment plan
- Test plans
- Merge database with Emp_Dat_V3
- Payroll and CSV integration

Unfinished Items
- Test cases for payroll page
- Testing documents
- V&V documents for SPR4
- User manual

Tasks Until Next Meeting
- V&V documents for SPR4
- Testing documents
- Usability test cases

Notes
- No additional notes

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)

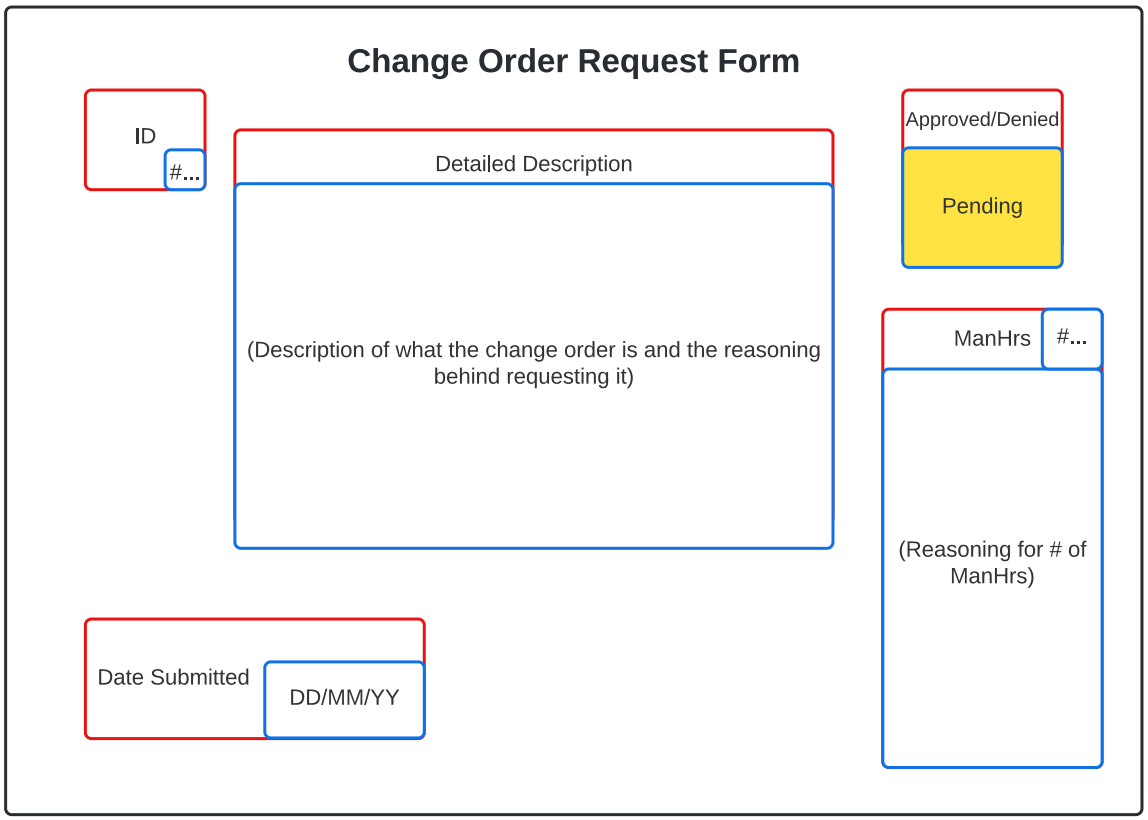
Change Control Board

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.



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

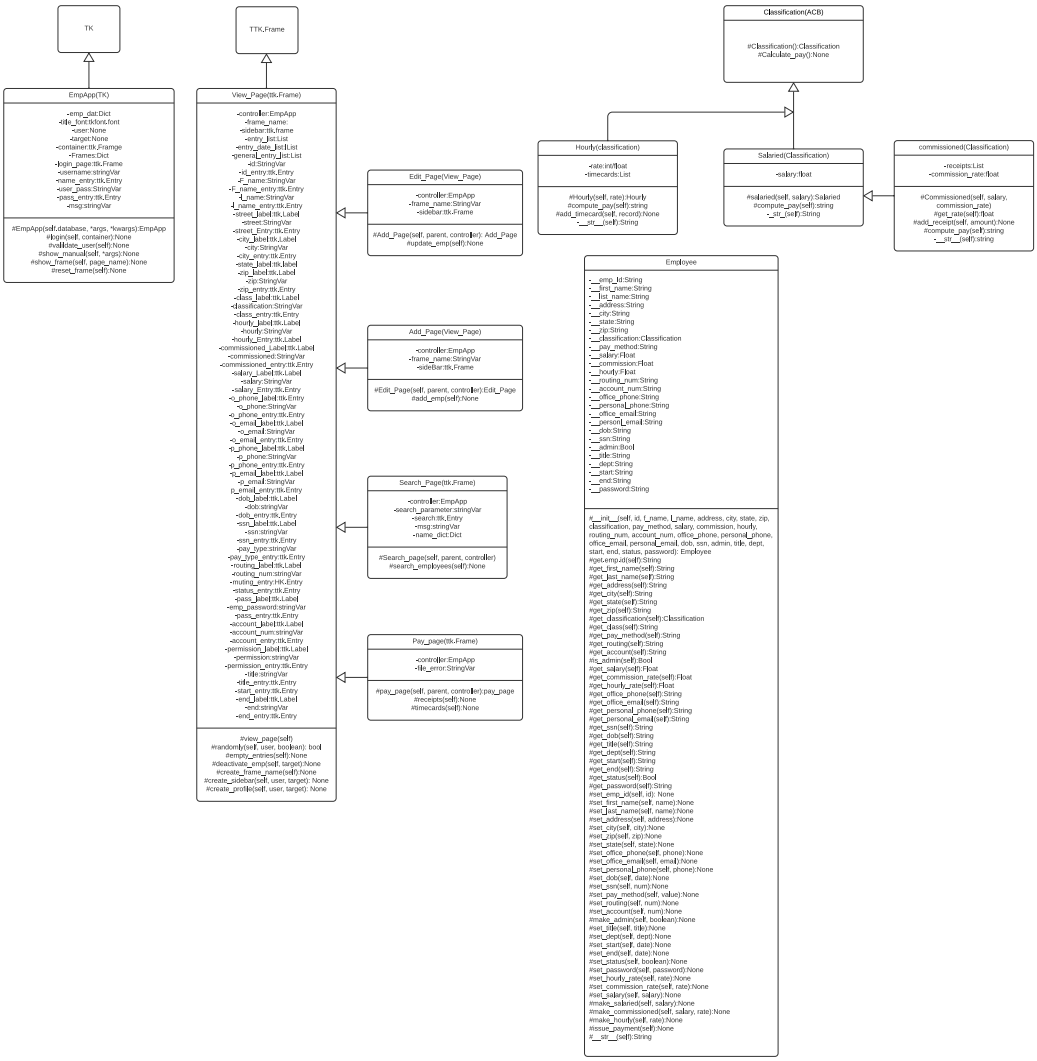
Maintenance Plan

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

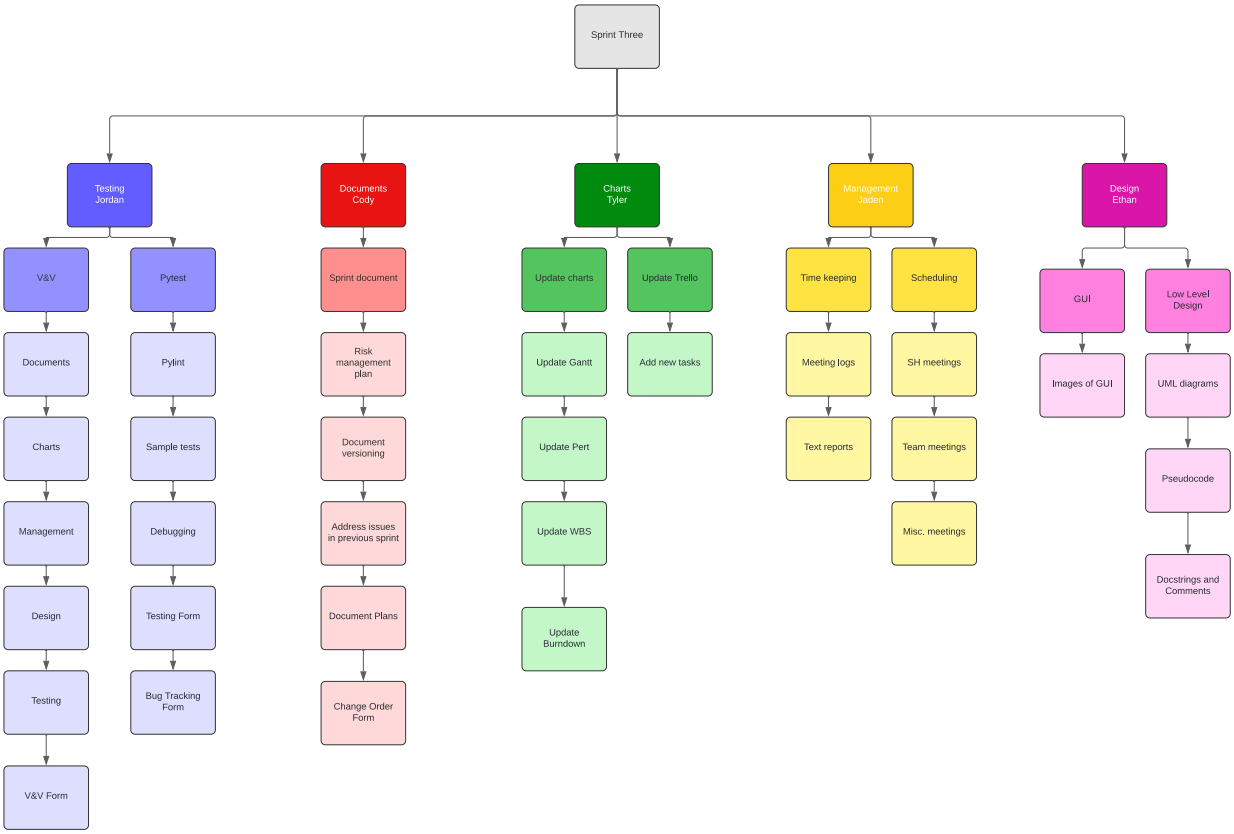
Summary: We have confirmed that the LLD design will function with the desired Hardware, Network configuration, Development environment, and Source Code Control

Charts/Forms

SPR-3 Work Breakdown Structure

Chart

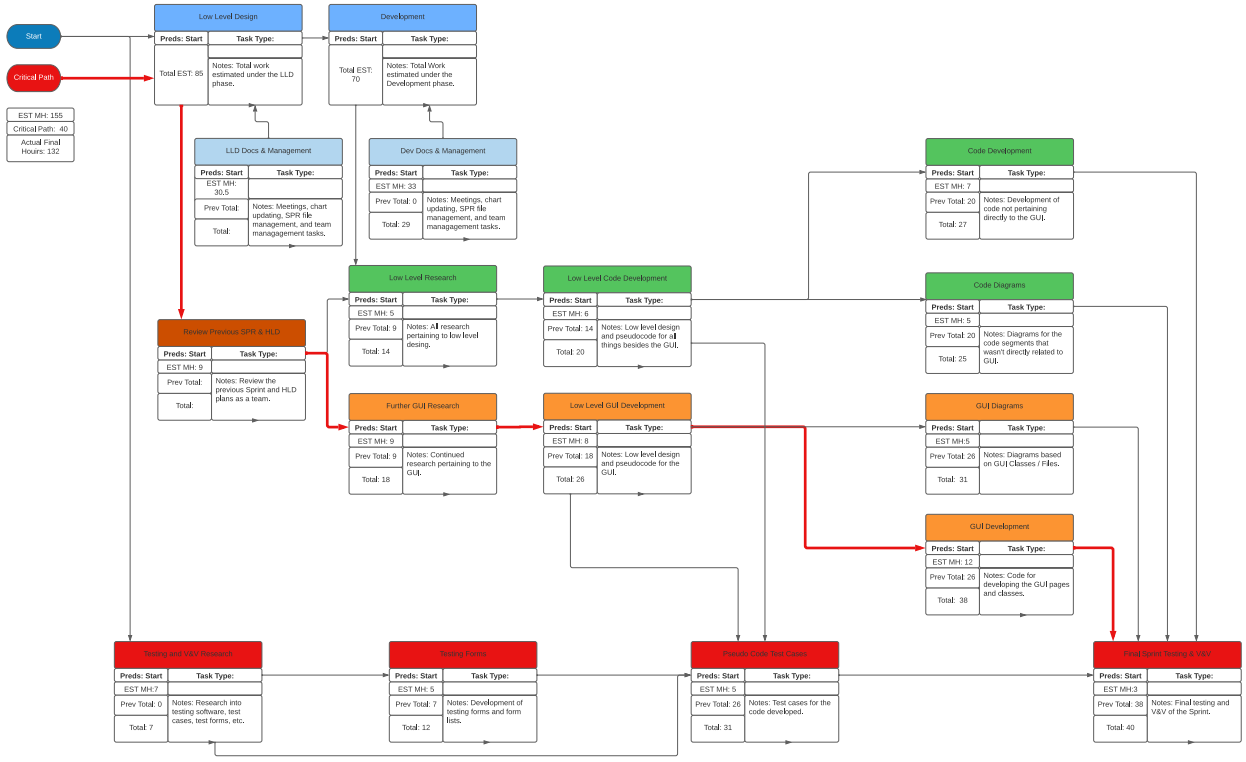
Summary: This is a Work Breakdown Structure for Sprint three. It details each major tasks that need to be completed and who is in charge of managing said tasks, making sure that they are completed. All tasks listed are relevant to SPR3 and should be completed by the end of the sprint. It is important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



SPR-3 Pert Chart

Chart

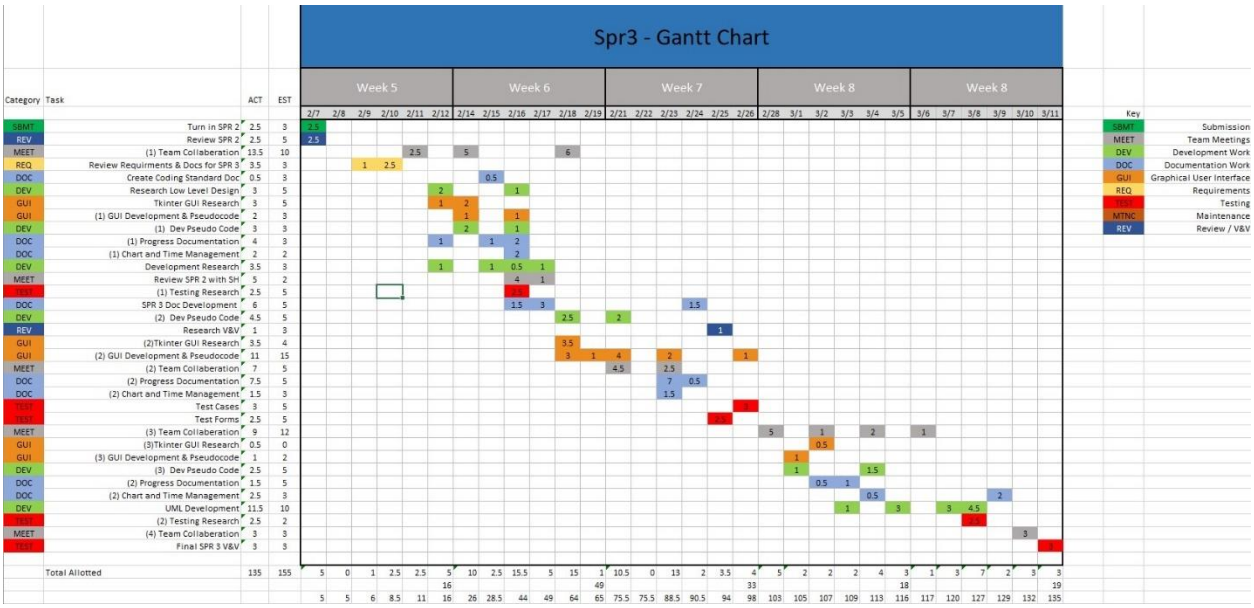
Summary: This is the Sprint Three Pert Chart; it details the time estimates for the tasks in sprint three and the prerequisites for specific tasks. Along with the critical path that lists the longest chain of tasks.



SPR-3 Gantt Chart

Chart

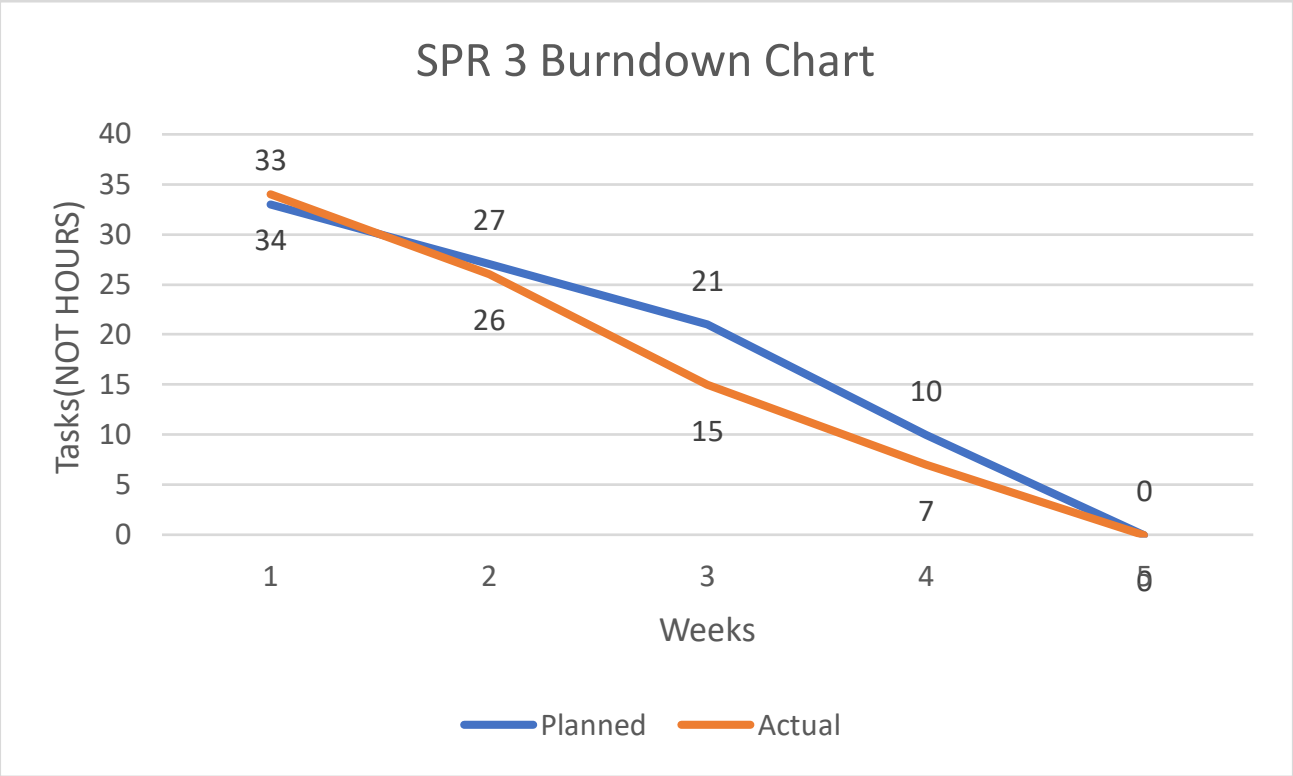
Summary: This is the Sprint Three Gantt Chart; it will list each of the tasks that we will be doing in sprint three, with the estimated time and actual time for each task.



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

SPRINT TWO

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

Introduction

Requirements Gathering

Requirements spec.

Summary: Requirements gathered from meeting with customer, creating MoSCoW charts, research. Requirements have been split between functional and non-functional. There are also the must-have requirements and the change orders that have been requested.

- Requirement Specifications:
    - Database Merging: recognize when there are incomplete fields and making a flag pop up that says “this is an incomplete employee”
    - Add employee: Page(B) of the GUI, Admin access, button on page(A), input all required fields and add employee to database
    - Edit employee: Page(C) of the GUI, General access, button on page(A), list of employee information to edit(First name, Last name, Address, Office phone, Personal phone, Bank info, Office email, Personal email)
    - Edit employee: Page(C) of the GUI, Admin access, button on page(A), list of employee information to edit(SS#, D.O.B, Pay type, Title Dept., Permission level)
    - Search employee by last name or ID: Page(A) of the GUI, General access, button on page(A), if user inputs numbers will search based on ID, if user inputs letters will search based off of last name, pulls up list of names and numbers of every that matches the information input
    - View employee: Page(A) of GUI, Admin access, list of employees names and IDs, when user clicks on name/ID pulls up all information related to that employee, option to view/hide deactivated employees
    - Deactivate employee: Page(D) of GUI, Admin access, button on page(A), input ID of employee that the user wants to deactivate, confirmation message pops up along with employee information to make sure the user wants to deactivate this specific employee
    - Win 10: The program will be able to run on Windows 10
    - Reports: Page(E) of GUI, Admin access, button on page(A), options to produce various reports(pay, reimbursables, employees)
    - Export Reports: Page(E) of GUI, Admin access, button on page(A), option to export any report as a csv
    - Secure records to only admin permissions: Certain information/records will require the user to be flagged as an Admin to see.
    - Intuitive GUI: Mouse over input boxes to see what information is required, help buttons on each page
    - User Manual: Page(F) of GUI, General access, help button on page(A), information on what each page can do
    - Readme.txt: A short description of what data the code contains

    - Simple just download installation that runs: Download software and then the user is good to go
    - Garbage proof entries: Checks each field of data to make sure that all information coming in isn't junk
    - Warn user of empty data fields: When user adds/edits an employee, issue a warning if any data fields are left empty or incomplete
    - Employee can view all personal fields: Page(A) of GUI, when employee logs in with his/her ID that ID's corresponding information is pulled up
    - Bug free: Software will go through extensive testing to minimize/eliminate as many bugs as possible
    - Requirements for employee information: First name, Last name, Address(use separated fields), Office phone, Personal phone, Emp ID(Specific length, only numbers), Pay type(commission, hourly, salary), D.O.B, SS#(Specific length, only numbers), Start date, End date, Bank Info(if Direct Deposit), Permission level, Title Dept., Office email, Personal email
- Functional
    - Add employee
    - Edit employee
    - Search employee
    - View employee
    - Deactivate employee
    - Reports
    - Export Reports
    - Secure records to only admin permissions
    - Warn user of empty data fields
    - Employee can view all personal fields
    - Requirements for employee information
- Non-functional
    - Win 10
    - Bug free
    - Garbage proof entries
    - Simple just download installation that runs
    - Readme.txt
    - User Manual
    - Intuitive GUI
    - Database merging
- Change orders
    - N/A

Prototype candidate

Jaden Albrecht

Summary: The program can read in a csv file of employee information, each employee is required to have an Id, first name, last name, Address, City, State, zip code, and pay classification. The program holds every employee's information in a list that can be accessed to look up an individual's information. The program also requires timecard, receipt, and pay log files in order to calculate each employee's pay. This is the prototype that we chose to use, because its code is more flexible out of the two prototypes we had. It is code from a teammate that is already in Python.

Current Features

- Calculate an employee's pay by their total salary, commission, or hourly wage
- Add employees to the database
- Find employees based on their ID number
- Reads in an csv file and out puts a list of employee objects
- Return specific employee information such as ID, name, address(city, state, zip code), classification(hourly pay, commissioned pay, salary pay)
- Alter an employee's classification

Needed Features

- Recognize when there are incomplete fields and make a flag pop up that says "this is an incomplete employee"
- Add an employee's:
    - First name
    - Last name
    - Office phone
    - Personal phone
    - Bank info
    - Office email
    - Personal email
    - SS#
    - D.O.B
    - Title Dept.
    - Permission level
- Search employees by last name
- Deactivate employees
- Export reports as a csv
- Secure records to only admin permissions
- GUI

Resources

Communication

Summary: We primarily use text messaging group to stay in contact with each other outside of planned meetings. We use it to ask for quick updates on tasks that may affect what we are currently doing. MS teams is what we use for our team meetings, we are currently planning to meet twice a week Monday and Friday at 7:00pm. There is also the project email and google drive that is being used to send documents to each other and to store files for the project.

- MS teams
- Text group chat
- Project email and Google drive

Software

Summary: We are using Lucid Charts as the software to create the Pert, Gantt, MoSCow, and Work Breakdown charts. Microsoft word is what we are using for documentation for our meeting logs, notes, and sprint documents. We are using Trello to help us organize a to-do list that is similar to Kanban. Python is the programming language that we will be working in, and we are still not sure whether we will be using Thonny or Visual Studio Code as the IDE. Tkinter will be used for the GUI. We will be using a group Gmail to send information to the group and Google Drive to save all of our documents so that the entire group and the shareholder will have access to them.

- Lucid Charts
- Microsoft word
- Trello
- Python
- Thonny/VS code
- Tkinter
- Debugging Software*
- Messages(Time keeping)
- Gmail
- Google Drive

Research

Sources

- Beginning Software Engineering(chapter 4)
- CS2450 Lecture #6(2/2/2022)
- Customer Meeting #2(2/1/2022)
- CS2450 Lecture #5(1/31/2022)

Management

Procedures

Document versioning

Summary: We currently have an archive folder in our google drive that we put any old documents into when we get a new one. Also in the titles of documents we put _v(x) for whatever version of that document is passed the initial version of the document.

Scheduling

Summary: We have the given deadlines from the customer for scheduling when our sprints need to be completed by. Jaden Albrecht is our team manager and schedules our meetings with the stakeholder. We have two meetings scheduled each week at 7pm on Monday and Friday.

Verification & Validation

Summary: We have processes for both verification and validation. Both are in early stages of implementation and are likely to change as we see what works and what doesn't as the sprints progress.

- Verification: We meet the Friday after the sprint begins and discuss our initial thoughts on the requirements of this sprint. We then have the weekend to mull over our ideas and figure out what we missed. Then we meet back on Monday and confirm what we agree needs to be done. We then schedule a meeting with the stakeholder so that we can get a final confirmation on what all needs to be done.
- Validation: We have a V&V folder that when we finish a document we submit it into that folder. Jordan then validates that the documents meet the proper requirements and are ready to be reviewed by the professor. Jordan then either sends them back to us to revise or to the review folder for the professor.

Timekeeping

Summary: We have the Gantt chart and the Burndown chart that shows the estimated time and the total time that is spent on the sprint, and tasks in the sprint. We also have a text message group that each day we send all the tasks we worked on and the amount of time we have spent on them.

Brainstorming

Summary: We have our team meetings on Friday and Monday at 7pm. This allows us to come together on Friday to begin brainstorming and then we have the weekend to think over the ideas that were presented and meet back on Monday to finish the brainstorming.

Plans

Quality Control

Summary: When someone finishes a section of code for the project another teammate will test and verify the code to guarantee that the code runs properly and follows the proper coding standards.

Risk Management

Summary: Risk management is accomplished through using the V&V processes and using debugging software and proper coding standards.

Testing

Summary:

Agile Methodologies

Scrum

Overview

Summary: As the project sprints are designed around the Scrum methodology, we will be using Scrum in our project management. We will be completing our project in incremental iterations so that at the end of each sprint we will have a fully tested and approved piece of software. We still need to decide exactly what parts of Scrum will and won't work for our team.

- Scrum focuses on breaking large problems into smaller tasks and completing tasks with speed and efficiency.
- Scrum focuses on creating a small workable product soon and building new and more complex versions of that product every sprint.
- Scrum is flexible, allowing constant changes throughout development to meet customers everchanging requirements.

Crystal Clear

Overview

Summary: Along with Scrum will be incorporating part of the Crystal Clear development methodology. Crystal Clear like scrum focuses on frequent releases that get additional features as time goes on. Crystal Clear is very heavy on team communication, it recognizes that in small teams it is valuable to have everyone pitch in on ideas and that rigid structures are not necessary.

- Much more informal methodology
- Expected to deliver production every 2-3 weeks, possibly shorter with non-released development iterations
- Take into consideration what the program will be used for when deciding how strict certain requirements such as testing need to be
- Emphasis on communication and in person collaboration whenever possible

Kanban

Overview

Summary: The last methodology that we will be using is Kanban, and we will be using Trello to implement this. We use the Kanban board to visualize what needs to be done, what is currently being worked on, and what has been completed.

- Kanban enables us to visualize the work that we are doing today in the context of other tasks
- Kanban board

Research

Sources

- Beginning Software Engineering(chapter 14)

Programming

Coding Standards

PEP-8

Summary: We will be following the PEP style guide for python code, as well as using either Pytest or Black to ensure that coding standards are followed.

High-Level Design

Plain English Code

Summary: The Plain English Code gives a description of our solution to the software we plan on creating in plain English that should be easy to read and understand. It goes over each of the screens that we plan on creating for the GUI and what needs to be done on each of them. It uses the Requirement Specifications as a guideline for the functionality planned for each window. The Plain English Code does not state specifics on how each GUI window will be created/implemented only what needs to be done on each one.

Login Screen:

- Import Tkinter
- Set up main window (can be inherited by other subsequent pages/windows)
- Set title and labels for user inputs (username and password)
- Set up grid manager to control widget size and placement
- Get username and password from user inputs
- Check for provided username in the database
- If username not in database, report username is not valid employee
- If username or password incorrect, display warning
- Validate username and password, get user permission level, and display proper window based on permission level

View Employee Screen (Admin):

- Set up view screen window (can be inherited by non-admin permission screen, as well as Add Employee and Edit Employee since they use the same labels)
- Set title and labels with accompanying entries for employee attributes (first name, last name, address, city, state, zip, office phone, personal phone, Emp ID, pay type/classification, D.O.B, SS#, start date, bank info(if Direct Deposit), permission level, title, dept., office email, personal email, employment status)
- Set up buttons in a sidebar for edit employee(opens edit employee screen), add employee(opens add employee screen), deactivate employee(opens deactivate employee screen), search database(opens search screen), print payroll(outputs payroll info as a csv file), and help(opens user manual)

- Set up grid manager to control widget size and placement
- Get employee object from database using provided username and password
- Use attributes of employee object to set the various fields so that by default the user is looking at their own info
- Set entry fields to read-only
- If employee is missing any required fields, display a warning

View Employee Screen (Non-admin):
- Inherit window and grid properties from admin version of screen
- Restrict employee attribute labels (first name, last name, address, city, state, zip, office phone, office email, title, dept, employment status)
- Restrict sidebar buttons to just edit employee, search database, and help
- Get employee object from database using provided username and password
- Use attributes of employee object to set the various fields so that by default the user is looking at their own info
- Set entry fields to read-only
- If employee username does not match name in employee object, disable edit button (so a non-admin employee can only edit themselves)
- If employee is missing any required fields, display a warning

Edit Employee Screen:
- Inherit window and grid properties from admin version of screen
- Set screen title and labels with accompanying entries for all employee attributes (first name, last name, address, city, state, zip, office phone, personal phone, Emp ID, pay type/classification, D.O.B, SS#, start date, bank info(if Direct Deposit), permission level, title, dept., office email, personal email, employment status)
- Set up a "save-changes" button that will return the user to the view screen and help button
- Get employee object from database
- Use attributes from employee object to fill out entry fields
- If employee is missing any required fields, display a warning
- Validate entry fields with specific requirements (Emp ID(specific length, only numbers), SS#(specific length, only numbers))
- Upon click of "save-changes" button, update database and return to view screen

Add Employee Screen:
- Inherit window and grid properties from admin version of screen
- Set screen title and labels with accompanying entries for all employee attributes (first name, last name, address, city, state, zip, office phone, personal phone, Emp ID, pay type/classification, D.O.B, SS#, start date, bank info(if Direct Deposit), permission level, title, dept., office email, personal email, employment status)
- Set up button for add employee and help button
- Validate entry fields with specific requirements (Emp ID(specific length, only numbers), SS#(specific length, only numbers))
- Upon click of "add" button, build new employee object and add it to database, then return to view screen

Deactivate Employee Screen:
- Set up screen window
- Set screen title and label with accompanying entry field for Emp ID
- Set up "deactivate employee" button and help button
- Check if entered Emp ID is in database

- If Emp ID not in database, report Emp ID invalid
- If valid Emp ID, on click of "deactivate" button ask for confirmation
- On confirmation set employee status to deactivated and return to view screen

Search Employees Screen:
- Set up search screen window
- Set up combo-box** with list of search parameters (Must have last name and Emp ID, others optional)
- Set up entry field for search parameters and help button
- Set search entry to read-only until search parameter chosen
- Set up grid geometry to control widget size and placement
- Apply a filter to the entry field that will search the database for potential matches and provide results of this search in a drop-down list as the user types*
- If search returns no matches, report no match found
- Upon selection of searched employee, take employee info to view employee screen

Help Pop-Up Screen:
- On hover over important fields, display a small window with information about that field*

Employee Class:
- Employee object has following attributes: first name, last name, address, city, state, zip, office phone, personal phone, Emp ID, pay type/classification, D.O.B, SS#, start date, bank info(if Direct Deposit), permission level, title, dept., office email, personal email, employment status
- Methods to find employee object by Emp ID and last name, possibly more**
- Methods to change employee attributes

Global Functions:
- Function to read in employee csv into list
- Functions to read in timecard and receipt csv files
- Function to process and output payroll report

Security

Summary: This early on, we have not confirmed what security measures we plan on using. Though we are looking into some basic data encryption that Visual Studio Code offers.

Hardware

Summary: The software will be designed to run on a laptop or personal computer for windows 10. Support for mobile devices is not currently planned.

External Interface

Summary: We are currently using Tkinter for creating the GUI and as we get closer to the coding process we may decide to use GitHub to store our code.

Internal Interface

Summary: The GUI classes will use the properties of the Employee class to fill out the data fields on the GUI screen.

Reports/Outputs

Summary: Our current prototype from CS1410 has the ability to output pay reports to txt files, this will be expanded in our final product to report to output to csv files.

Database Design and Issues

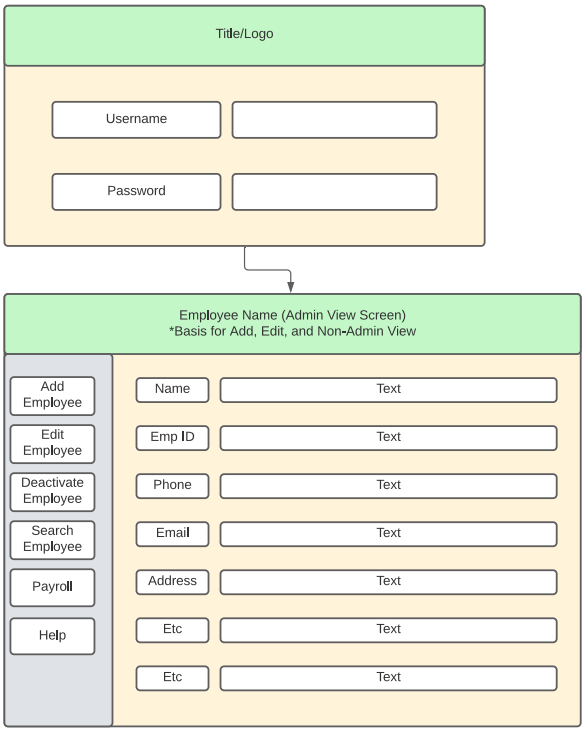
Summary: Our database design is going to be in text because we are going to be using csv files.

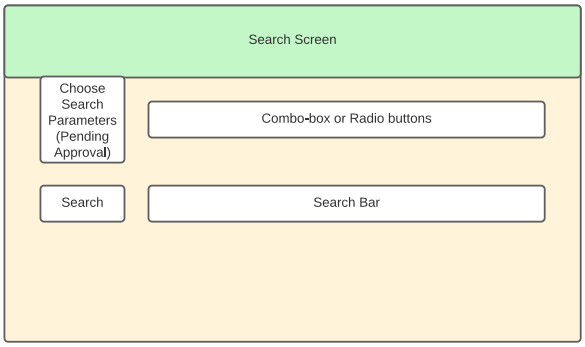
Training

Summary: The user manual will be in a help button on the GUI that will explain to the user how to use the software.

GUI Design

Summary: First version of the template for the GUI we will be creating. Shows the different screens that'll be implemented, and the information shown on each screen. Does not yet have the screen for reports or for deactivating employees





Architecture

Summary: We will be using a monolithic structure as our architecture

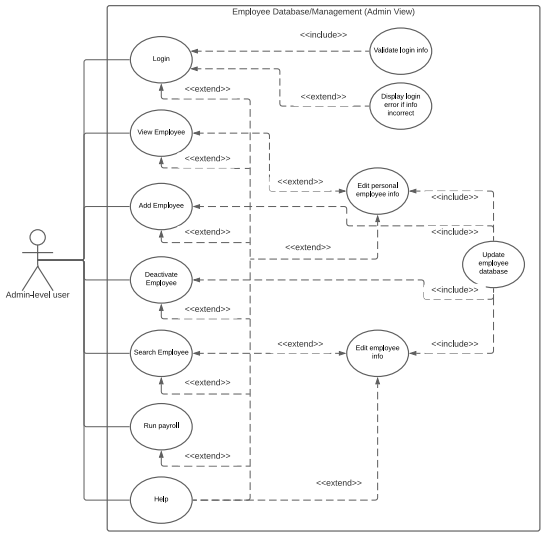
- A single program does everything
  - It displays the user interfaces
  - Accesses data
  - Processes customer orders

Use Case Scenarios

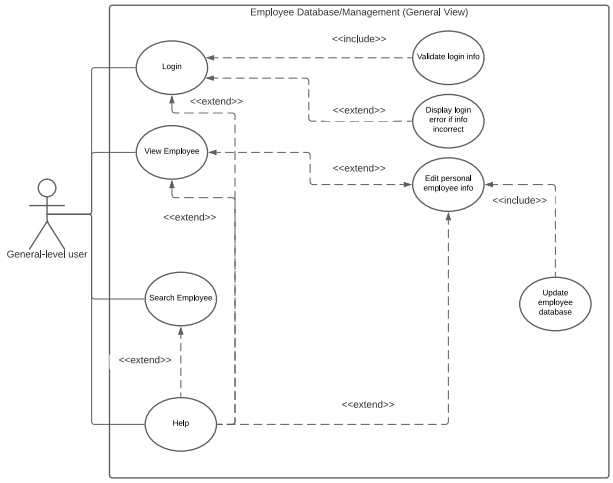
Summary: The use case scenarios are based on the privileges that the user has, whether they are an admin user or a general employee user. The admin user can add employees, edit

employees(including themselves), deactivate employees, print out report, and search employees based on id or last name. The general user can edit employees(themselves), view employees(limited fields). Both have access to the help screen, and both will utilize the login screen.

- Admin Use-Case



- General Use-Case



Docstring Code Outline

Summary: The docstring code outline outlines the components of the software that will be programmed. We have an employee class that creates employee objects that are classified by their pay type(hourly, salary, commission). We have Tkinter based GUI classes that will control the various screens of the components.

#Import any needed modules
#Initiate constants
Employee Class:
"Creates Employee object"

#Establish classification object
#Getters (to fill out view screen)
“Return employee ID”
“Return employee first name”
“Return employee last name”
“Return employee address”
“Return employee city”
“Return employee state”
“Return employee zip”
“Return employee classification”
“Return employee salary”
“Return employee commission rate”
“Return employee hourly rate”
“Return employee office phone”
“Return employee personal phone”
“Return employee office email”
“Return employee personal email”
“Return employee D.O.B”
“Return employee SS#”
“Return employee direct deposit info”
“Return employee permission”
“Return employee title”
“Return employee dept”
“Return employee employment status”
“Return employee start date”
“Return employee end date”

#Setters (to use with edit screen)
“Set employee ID”
“Set employee first name”
“Set employee last name”
“Set employee address”
“Set employee city”
“Set employee state”
“Set employee zip”
“Make employee salaried”
“Make employee commissioned”
“Make employee hourly”
“Set employee office phone”
“Set employee personal phone”
“Set employee office email”
“Set employee personal email”
“Set employee D.O.B”
“Set employee SS#”
“Set employee direct deposit info”
“Set employee permission”
“Set employee title”
“Set employee dept”
“Set employee employment status”
“Set employee start date”
“Set employee end date”

#Other methods
“Pays the employee”
“String representation of employee object”

Classification Class:
“An abstract class of a classification object for an employee”
"Abstract decorator method of computing an employee's payment"

Hourly Class:
"Class that creates an hourly classification"
"Adds timecards to the list of timecards"
"Computes the payment of an hourly employee"

Salaried Class:
"Class that creates a salaried classification for an employee"
"Computes the payment of a salaried employee"

Commissioned Class:
"Constructs a commission classification for an employee"
"Adds receipts to the list of receipts"
"Computes the payment for a commissioned employee"

#Global Functions
"A function that processes employees database file into a useable list"
"Adds receipts to every commissioned employee"
"Adds timecards to every hourly employee"
"Finds an employee by their ID number"
"Finds an employee by their last name"
"Pays each employee"

#GUI Classes
Login Class:
“Controls the login screen of the GUI”

Admin_View Class:
“Controls the admin view screen of the GUI”
#Show all fields and buttons
#On click of deactivate button, show extra screen requesting confirmation

General_View Class:
“Controls the general view screen of the GUI”
#Only show first name, last name, address, city, state, zip, office phone, office email, title, dept, employment status
#Only show edit button if username equals employee, else show search and help buttons only

Edit_Screen Class:
“Controls the edit employee screen of the GUI”

Add_Screen Class:
“Controls the add employee screen of the GUI”

Search_Screen Class:
“Controls the search screen of the GUI”
#If user enters numbers search by EmpID, else if user enters letters search by last name

Testing

Summary: For testing the most likely software we plan on using will be Pytest with Assertion testing. May be using Black as well for making sure the program follows the proper coding standards.

Research

Sources

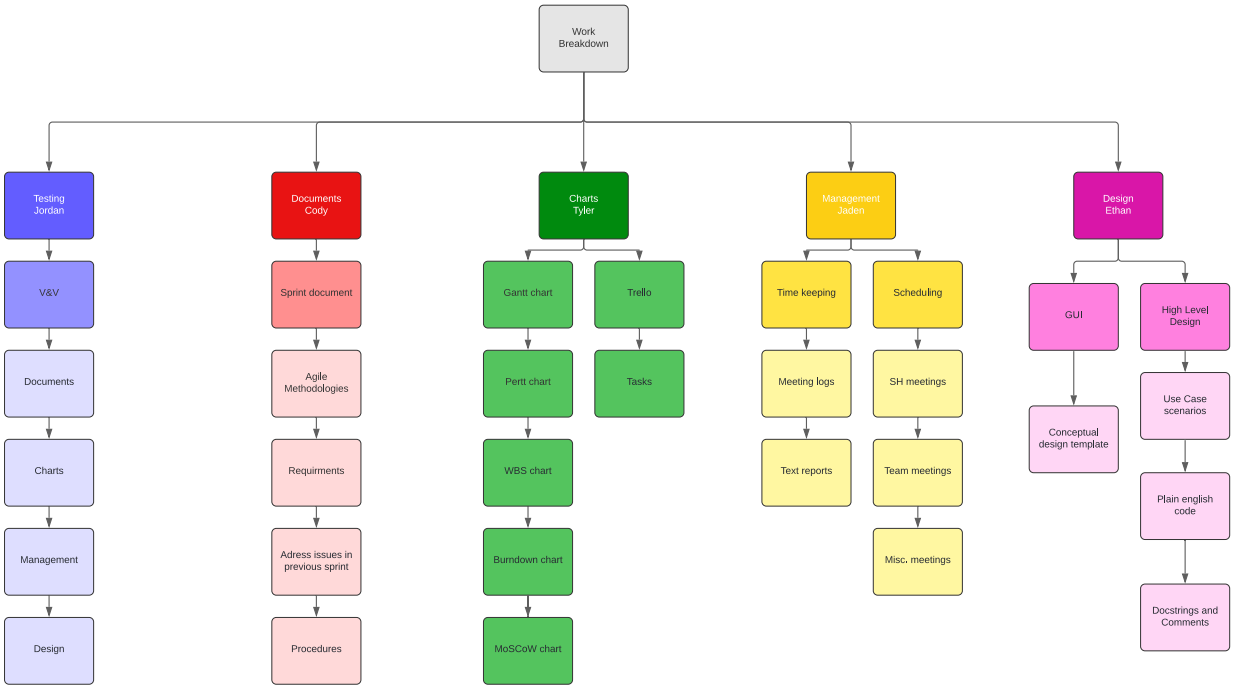
- Python.org
- Beginning Software Engineering(chapter 5)
- Modern Tkinter
- Course material

Charts/Templates

SPR-2 Work Breakdown Structure

Chart

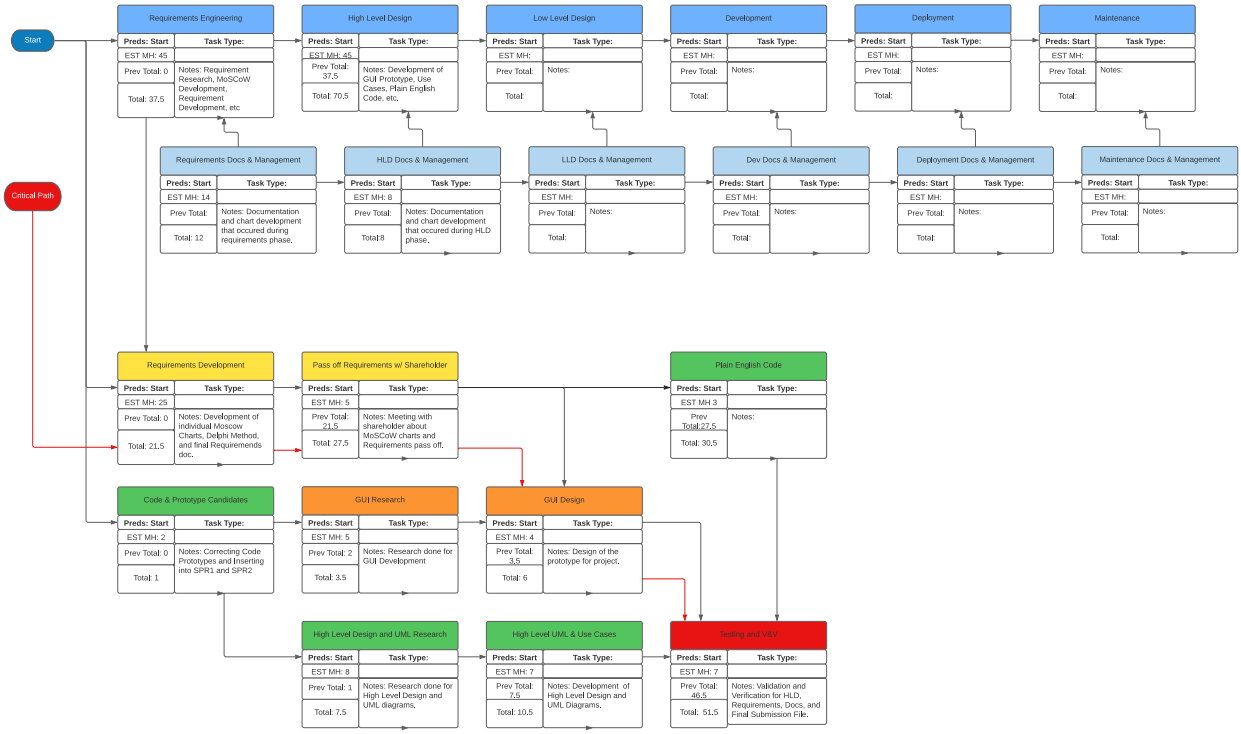
Summary: This is a Work Breakdown Structure for our five members. It details each major tasks that need to be completed and who is in charge of managing said tasks, making sure that they are completed. All tasks listed are relevant to SPR3 and should be completed by the end of the sprint



SPR-2 Pert Chart

Chart

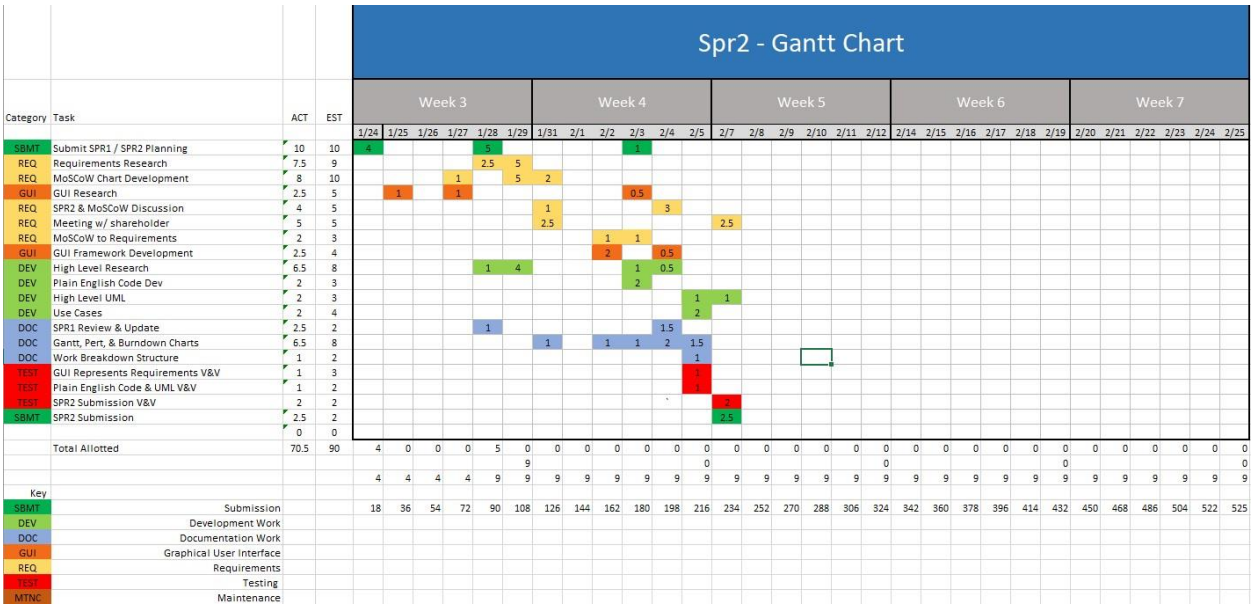
Summary: This is a Pert Chart Template, for each of the stage. It has the estimated time we expect each task to take and come up with a total amount of time for the project. The critical path came out to 41hours



SPR-2 Gantt Chart

Chart

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



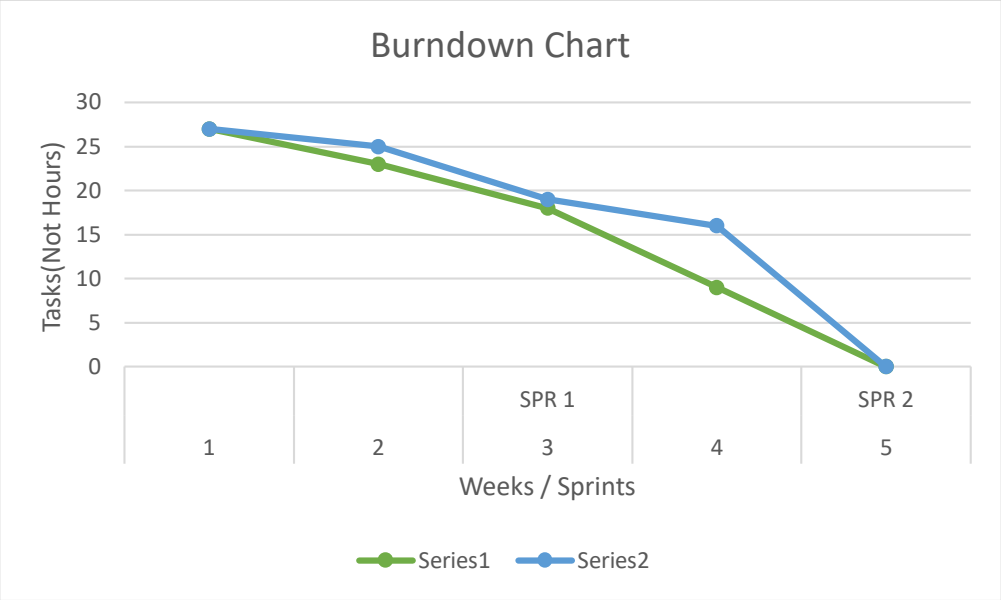
Spr2 - Gantt Chart

Category	Task	ACT	EST	Week 3						Week 4						Week 5						Week 6						Week 7					
				1/24	1/25	1/26	1/27	1/28	1/29	1/31	2/1	2/2	2/3	2/4	2/5	2/7	2/8	2/9	2/10	2/11	2/12	2/14	2/15	2/16	2/17	2/18	2/19	2/20	2/21	2/22	2/23	2/24	2/25
SBMT	Submit SPR1 / SPR2 Planning	10	10	4				5					1																				
REQ	Requirements Research	7.5	9					2.5	5																								
REQ	MoSCoW Chart Development	8	10				1		5	2																							
GUI	GUI Research	2.5	5		1		1						0.5																				
REQ	SPR2 & MoSCoW Discussion	4	5							1				3																			
REQ	Meeting w/ shareholder	5	5							2.5						2.5																	
REQ	MoSCoW to Requirements	2	3									1	1																				
GUI	GUI Framework Development	2.5	4									2		0.5																			
DEV	High Level Research	6.5	8					1	4				1	0.5																			
DEV	Plain English Code Dev	2	3										2																				
DEV	High Level UML	2	3												1	1																	
DEV	Use Cases	2	4												2																		
DOC	SPR1 Review & Update	2.5	2					1						1.5																			
DOC	Gantt, Pert, & Burndown Charts	6.5	8							1		1	1	2	1.5																		
DOC	Work Breakdown Structure	1	2												1																		
TEST	GUI Represents Requirements V&V	1	3												1																		
TEST	Plain English Code & UML V&V	1	2												1																		
TEST	SPR2 Submission V&V	2	2													2																	
SBMT	SPR2 Submission	2.5	2													2.5																	
		0	0																														
	Total Allotted	70.5	90	4	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									9						0						0						0						0
				4	4	4	4	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
Key																																	
SBMT	Submission			18	36	54	72	90	108	126	144	162	180	198	216	234	252	270	288	306	324	342	360	378	396	414	432	450	468	486	504	522	525
DEV	Development Work																																
DOC	Documentation Work																																
GUI	Graphical User Interface																																
REQ	Requirements																																
TEST	Testing																																
MTNC	Maintenance																																

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. There are two versions of the chart, the regular burndown chart, and the backwards burndown chart.



Research

Sources

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

Meeting Logs

Meeting Log#4

Team #: T2-002
Meeting log #: 4
Date: January 28, 2022
Time: 7:00pm – 8:05pm
Location: MS Teams (watch video here)
Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Tyler Deschamps, Jordan Van Patten
Current Sprint: SPR2

Sprint Progress

Ethan Taylor:
- Read all SPR2 documentation
- Finished personal MoSCoW chart (30 minutes)

Jaden Albrecht:
- Read all SPR2 documentation
- Finished personal MoSCoW chart (30 minutes)

Cody Strange:
- Read all SPR2 documentation
- Finished personal MoSCoW chart (30 minutes)

Tyler Deschamps:
- Read all SPR2 documentation
- Finished personal MoSCoW chart (30 minutes)
- Created Trello board for task management

Jordan Van Patten:
- Read all SPR2 documentation
- Finished personal MoSCoW chart (30 minutes)

Topics Discussed
- Next shareholder meeting
- MoSCoW charts (Delphi method)
- Testing methods
- Task assignment
- Preparations for high-level design

Obstacles Encountered
- No obstacles encountered

Finished Items
- All team members have thoroughly read and grasped the requirements for SPR2

- Each team member constructed individual MoSCoW charts based on user requirements discussed in chapter 4
- Assigned Jordan as tester/V&V/QA consultant
- Scheduled next team meeting for January 31, 2022

Unfinished Items

- Final requirements spec
- High-level design (plain English description)
- Create classes for GUI
- Use-case scenarios for high-level design
- Develop pert, Gantt, work breakdown structure, and MoSCoW charts
- SPR1 prototype candidate
- Establish connection between tasks and requirements
- V&V
- Schedule next shareholder meeting

Tasks Until Next Meeting

- Jaden is responsible for reviewing each individual MoSCoW chart and combining them into two separate charts
- Each chart will be distributed to one of two subgroups of the team and will be reviewed by each subgroup
- The final MoSCoW chart will be assembled after both subgroups have made their changes to each chart

Notes

- There was not a whole lot to work with at this point in the development cycle
- Once the final MoSCoW chart is assembled, each team member will have a clearer understanding of what is required of them throughout the remainder of the current sprint

Meeting Log#5

Team #: T2-002
Meeting log #: 5
Date: January 31, 2022
Time: 6:55pm – 8:10pm
Location: MS Teams (watch video here)
Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Tyler Deschamps, Jordan Van Patten
Current Sprint: SPR2

Sprint Progress
Jaden Albrecht:
- Distribution of MoSCoW charts for Delphi (2 hours:55 minutes)

Topics Discussed
- Final MoSCoW chart
- Next team meeting
- Beginning high-level design
- Beginning requirements spec
- Begin development of pert, Gantt, and work breakdown structure (WBS) charts
- Methods for tracking tasks and requirements
- V&V

Obstacles Encountered
- Jaden is unable to edit or create new cards on team Trello board

Finished Items
- Distributed MoSCoW for Delphi
- Added Jordan to team Google drive, Trello board, and text chat
- Submitted list of disapprovals from shareholder from SPR1
- Scheduled next team meeting for February 4, 2022
- Scheduled next shareholder meeting for February 1, 2022

Unfinished Items
- Requirement spec
- Plain-English description of high-level design
- Classes for GUI
- Use-case scenarios for high-level design
- V&V
- Burndown chart
- Up-to-date pert/Gantt charts
- Schedule next shareholder meeting

Tasks Until Next Team Meeting
- Finalize MoSCoW chart
- Read SPR2 documentation
- Work on plain-English description for high-level design
- Develop classes for GUI

- Develop use-case scenarios for GUI
- SPR1 prototype candidate
- Create requirement spec
- Establish connection between requirements and tasks
- Update meeting logs to new template
- Begin developing burn-down chart
- Develop Gantt chart
- Logging for Gantt/pert charts
- V&V covering requirements and documentation
- Continue referring to chapters 4-6 for more help

Notes

- At this point, we have our final MoSCoW chart complete and ready to discuss with the shareholder
- From now until our next team meeting (not with the shareholder), Tyler will begin development of burn-down chart and begin tracking minutes on Gantt chart
- Cody will make final changes to MoSCoW chart according to preferences of the shareholder and will also begin the requirement spec
- Ethan, Jaden, and Jordan will research tkinter and begin high-level design

Meeting Log#6

Team #: T2-002
Meeting log #: 6
Date: February 4, 2022
Time: 7:05pm – 8:00pm
Location: MS Teams (watch video here)
Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Tyler Deschamps
Current Sprint: SPR2

Sprint Progress

Ethan Taylor:

- Review requirements for SPR2
- High-level design research (1 hour)
- Plain English description for high-level design (2 hours, 5 minutes)
- GUI template (30 minutes)
- Develop use-case scenarios based on requirements (1 hour, 45 minutes)

Jaden Albrecht:

- Review requirements for SPR2
- Review plain-English description of high-level design (30 minutes)
- Establish connection between requirements and tasks (1 hour)
- Changes to meeting logs and developed method for minute tracking (3 hours:30 minutes)
- Plan for next team meeting (5 minutes)

Cody Strange

- Review requirements for SPR2
- Create requirements spec (1 hour)
- Changes to final MoSCoW chart (1 hour)
- Final revisions to SPR1 document (2 hours, 30 minutes)
- SPR1 prototype candidate (50 minutes)

Tyler Deschamps:

- Review requirements for SPR2
- Develop beginning/working burn-down chart (1 hour)
- Gantt chart development (2 hours)
- Logging for Gantt/pert charts (1 hour)

Jordan Van Patten:

- Review requirements for SPR2
- V&V covering requirements and documentation (3 hours)

Topics Discussed

- Task management
- Overview of pert chart
- Overview of Gantt chart
- Overview of work breakdown structure (WBS)
- Overview of high-level design document (rough draft)
- Use-case scenarios for high-level design

- V&V
- Final revisions to requirements spec
- Schedule next shareholder meeting
- Options for testing software

Obstacles Encountered

- Jaden found new solution for tracking tasks on team Trello board, but is still unable to edit tasks on his own list/card

Finished Items

- High-level design
- Fully viable work breakdown structure and MoSCoW charts
- Updated meeting logs to fit new template
- Scheduled next team/shareholder meeting for February 7, 2022

Unfinished Items

- Final requirements spec
- Use-cases for high-level design
- Create classes for GUI
- Finish developing pert/Gantt charts

Tasks Until Next Meeting

- Create classes for GUI
- Requirement spec
- Use-case scenarios for high-level design
- V&V
- Develop pert/Gantt chart

Notes

- At the start of SPR3, our team will start holding short meetings after class on Wednesdays

SPRINT ONE

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

Introduction

Procedures

Document versioning

Summary: We currently have an archive folder in our google drive that we put any old documents into when we get a new one. Also in the title's of documents we put _v(x) for whatever version of that document is passed the initial version of the document.

Scheduling

Summary: We have the given deadlines from the customer for scheduling when our sprints need to be completed by. Jaden Albrecht is our team manager and schedules our meetings with the stakeholder. We have two meetings scheduled each week at 7pm on Monday and Friday.

Verification & Validation

Summary: We have processes for both verification and validation. Both are in early stages of implementation and are likely to change as we see what works and what doesn't as the sprints progress.

- Verification: We meet the Friday after the sprint begins and discuss our initial thoughts on the requirements of this sprint. We then have the weekend to mull over our ideas and figure out what we missed. Then we meet back on Monday and confirm what we agree needs to be done. We then schedule a meeting with the stakeholder so that we can get a final confirmation on what all needs to be done.
- Validation: We have a V&V folder that when we finish a document we submit it into that folder. Jordan then validates that the documents meet the proper requirements and are ready to be reviewed by the professor. Jordan then either sends them back to us to revise or to the review folder for the professor.

Timekeeping

Summary: We have the Gantt chart and the Burndown chart that shows the estimated time and the total time that is spent on the sprint, and tasks in the sprint. We also have a text message group that each day we send all the tasks we worked on and the amount of time we have spent on them.

Requirements Gathering

Identify Customer

Summary: Currently it is unknown exactly who our customer is. While we know it is Craig Sharp, we do not know enough about him to identify him. We will have to have a stakeholder meeting in order to get the required information.

Requirements

Summary: Requirements gathered from meeting with customer, creating MoSCoW charts, research. Requirements have been split between functional and non-functional. There are also the must-have requirements and the change orders that have been requested.

- Must haves:
    - Database Merging: recognize when there are incomplete fields and making a flag pop up that says "this is an incomplete employee"
    - Add employee: Page(B) of the GUI, Admin access, button on page(A), input all required fields and add employee to database
    - Edit employee: Page(C) of the GUI, General access, button on page(A), list of employee information to edit(First name, Last name, Address, Office phone, Personal phone, Bank info, Office email, Personal email)
    - Edit employee: Page(C) of the GUI, Admin access, button on page(A), list of employee information to edit(SS#, D.O.B, Pay type, Title Dept., Permission level)
    - Search employee by last name or ID: Page(A) of the GUI, General access, button on page(A), if user inputs numbers will search based on ID, if user inputs letters will search based off of last name, pulls up list of names and numbers of every that matches the information input
    - View employee: Page(A) of GUI, Admin access, list of employees names and IDs, when user clicks on name/ID pulls up all information related to that employee, option to view/hide deactivated employees
    - Deactivate employee: Page(D) of GUI, Admin access, button on page(A), input ID of employee that the user wants to deactivate, confirmation message pops up along with employee information to make sure the user wants to deactivate this specific employee
    - Win 10: The program will be able to run on Windows 10
    - Reports: Page(E) of GUI, Admin access, button on page(A), options to produce various reports(pay, reimbursables, employees)
    - Export Reports: Page(E) of GUI, Admin access, button on page(A), option to export any report as a csv
    - Secure records to only admin permissions: Certain information/records will require the user to be flagged as an Admin to see.
    - Intuitive GUI: Mouse over input boxes to see what information is required, help buttons on each page
    - User Manual: Page(F) of GUI, General access, help button on page(A), information on what each page can do
    - Readme.txt: A short description of what data the code contains
    - Simple just download installation that runs: Download software and then the user is good to go

    - Garbage proof entries: Checks each field of data to make sure that all information coming in isn't junk
    - Warn user of empty data fields: When user adds/edits an employee, issue a warning if any data fields are left empty or incomplete
    - Employee can view all personal fields: Page(A) of GUI, when employee logs in with his/her ID that ID's corresponding information is pulled up
    - Bug free: Software will go through extensive testing to minimize/eliminate as many bugs as possible
    - Requirements for employee information: First name, Last name, Address(use separated fields), Office phone, Personal phone, Emp ID(Specific length, only numbers), Pay type(commission, hourly, salary), D.O.B, SS#(Specific length, only numbers), Start date, Bank Info(if Direct Deposit), Permission level, Title Dept., Office email, Personal email
- Functional
    - Add employee
    - Edit employee
    - Search employee
    - View employee
    - Deactivate employee
    - Reports
    - Export Reports
    - Secure records to only admin permissions
    - Warn user of empty data fields
    - Employee can view all personal fields
    - Requirements for employee information
- Non-functional
    - Win 10
    - Bug free
    - Garbage proof entries
    - Simple just download installation that runs
    - Readme.txt
    - User Manual
    - Intuitive GUI
    - Database merging
- Change orders
    - N/A

Prototype candidate

Jaden Albrecht

Summary: The program can read in a csv file of employee information, each employee is required to have an Id, first name, last name, Address, City, State, zip code, and pay

classification. The program holds every employee's information in a list that can be accessed to look up an individual's information. The program also requires timecard, receipt, and pay log files in order to calculate each employee's pay. This is the prototype that we chose to use.

Current Features

- Calculate an employee's pay by their total salary, commission, or hourly wage
- Add employees to the database
- Find employees based on their ID number
- Reads in an csv file and out puts a list of employee objects
- Return specific employee information such as ID, name, address(city, state, zip code), classification(hourly pay, commissioned pay, salary pay)
- Alter an employee's classification

Needed Features

- Recognize when there are incomplete fields and make a flag pop up that says "this is an incomplete employee"
- Add an employee's:
    - First name
    - Last name
    - Office phone
    - Personal phone
    - Bank info
    - Office email
    - Personal email
    - SS#
    - D.O.B
    - Title Dept.
    - Permission level
- Search employees by last name
- Deactivate employees
- Export reports as a csv
- Secure records to only admin permissions
- GUI

Resources

Communication

Summary: We primarily use text messaging group to stay in contact with each other outside of planned meetings. We use it to ask for quick updates on tasks that may affect what we are currently doing. MS teams is what we use for our team meetings, we are currently planning to meet twice a week Monday and Friday at 7:00pm. There is also the project email and google drive that is being used to send documents to each other and to store files for the project.

- MS teams
- Text group chat
- Project email and Google drive

Software

Summary: We are using Lucid Charts as the software to create the PERTT, Gantt, MoSCow, and Work Breakdown charts. Microsoft word is what we are using for documentation for our meeting logs, notes, and sprint documents. We are using Trello to help us organize a to-do list that is similar to Kanban. Python is the programming language that we will be working in, and we are still not sure whether we will be using Thonny or Visual Studio Code as the IDE. Tkinter will be used for the GUI. We will be using a group Gmail to send information to the group and Google Drive to save all of our documents so that the entire group and the shareholder will have access to them.

- Lucid Charts
- Microsoft word
- Trello
- Python
- Thonny/VS code
- Tkinter
- Debugging Software*
- Timekeeping software*
- Gmail
- Google Drive

Research

Sources

- Beginning Software Engineering(chapter 4)
- CS2450 Lecture #6(2/2/2022)
- Customer Meeting #2(2/1/2022)
- CS2450 Lecture #5(1/31/2022)

Agile Methodologies

Scrum

Overview

Summary: As the project sprints are designed around the Scrum methodology, we will be using Scrum in our project management. We will be completing our project in incremental iterations so that at the end of each sprint we will have a fully tested and approved piece of software. We still need to decide exactly what parts of Scrum will and won't work for our team.

- Scrum focuses on breaking large problems into smaller tasks and completing tasks with speed and efficiency.
- Scrum focuses on creating a small workable product soon and building new and more complex versions of that product every sprint.
- Scrum is flexible, allowing constant changes throughout development to meet customers everchanging requirements.

Crystal Clear

Overview

Summary: Along with Scrum will be incorporating part of the Crystal Clear development methodology. Crystal Clear like scrum focuses on frequent releases that get additional features as time goes on. Crystal Clear is very heavy on team communication, it recognizes that in small teams it is valuable to have everyone pitch in on ideas and that rigid structures are not necessary.

- Much more informal methodology
- Expected to deliver production every 2-3 weeks, possibly shorter with non-released development iterations
- Take into consideration what the program will be used for when deciding how strict certain requirements such as testing need to be
- Emphasis on communication and in person collaboration whenever possible

Kanban

Overview

Summary: The last methodology that we will be using is Kanban, and we will be using Trello to implement this. We use the Kanban board to visualize what needs to be done, what is currently being worked on, and what has been completed.

- Kanban enables us to visualize the work that we are doing today in the context of other tasks
- Kanban board

Research

Sources

Summary: Information was gathered from the following sources

- Beginning Software Engineering(chapter 14)

Programming

Coding Standards

PEP-8

Summary: We will be following the PEP style guide for python code.

Brainstorming

Team Meetings

Summary: We have our team meetings on Friday and Monday at 7pm. This allows us to come together on Friday to begin brainstorming and then we have the weekend to think over the ideas that were presented and meet back on Monday to finish the brainstorming.

Research

Sources

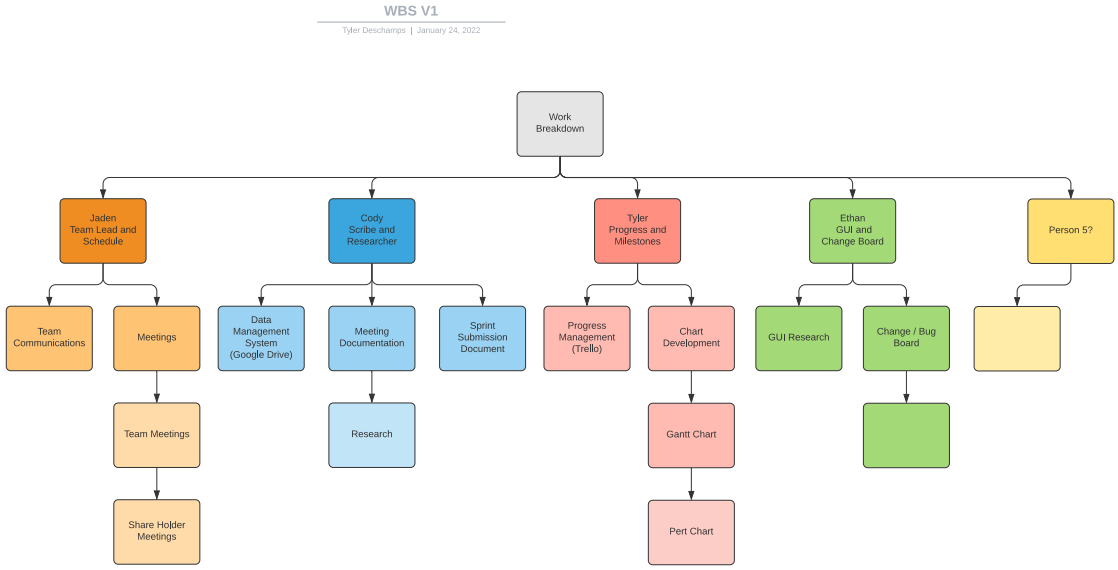
- Python.org

Charts/Templates

Work Breakdown Structure

Chart

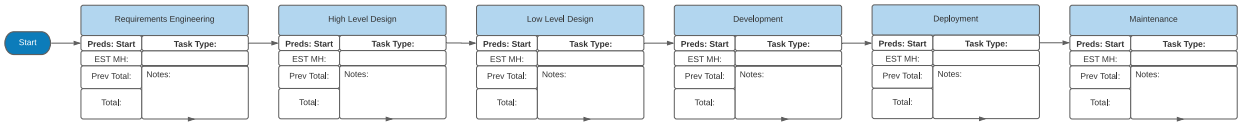
Summary: This is a Work Breakdown Structure for our four (soon to be five) members. It details each of our Team members and their roles along with the general tasks that their role comes with.



PERTT Chart

Template

Summary: This is a PERTT Chart Template, for each of the six stages. We will be able to fill it out with the estimated time we expect each task to take and come up with a total amount of time for the project



Gantt Chart

Template

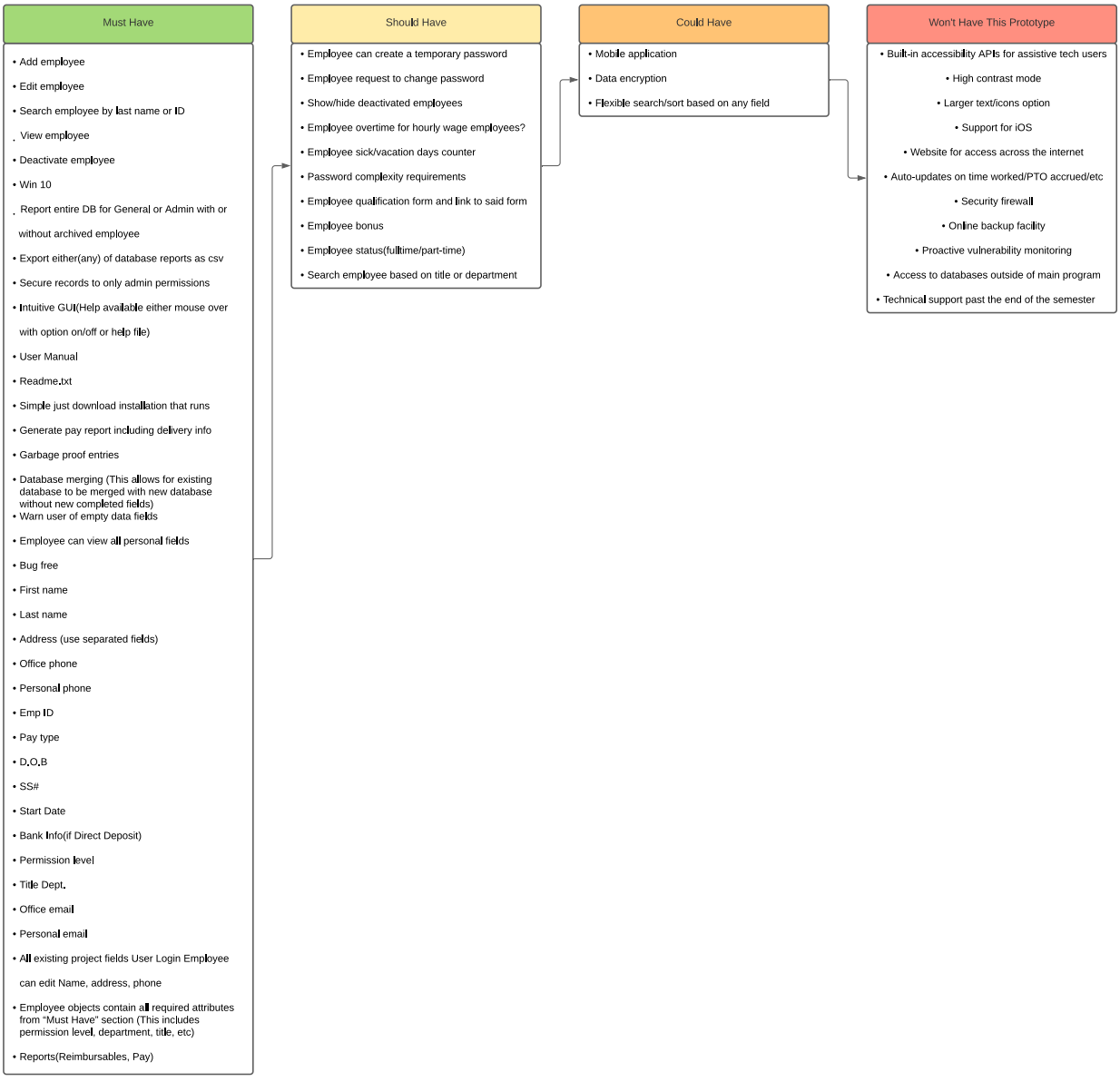
Summary: This is a Gantt chart template; it will list each of the tasks that we will be doing in each sprint.

Gantt Chart	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6
Task 1						
Task 2						
Task 3						
Task 4						
Task 5						
Task 6						
Task 7						
Task 8						
Task 9						
Task 10						
Task 11						
Task 12						
Task 13						
Task 14						

MoSCoW Chart

Overview

Summary: This is a MoSCoW chart template; it will have each team member listing what they believe the product Must have, Should have, Could have, and Won't have. We will each make a separate one and eventually combine/narrow them down to one.



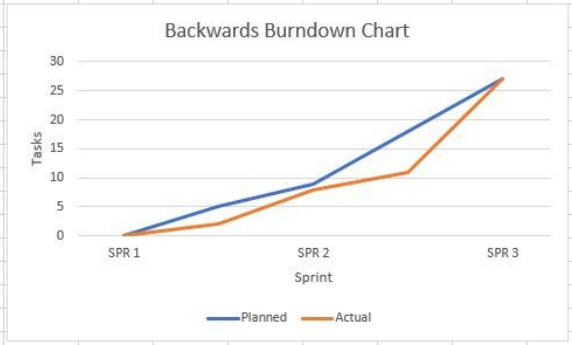
Burndown Chart

Overview

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. There are two versions of the chart, the regular burndown chart and the backwards burndown chart.

Time		Tasks					
Week	Sprint	Planned	Actual		Sprint	Num Tasks	Total
1		27	27		1	9	9
2		23	25		2	18	27
3	SPR 1	18	19		3		
4		9	16		4		
5	SPR 2	0	0		5		
6					6		
7	SPR 3						
8							
9	SPR 4						
10							
11	SPR 5						
12							
13	SPR 6						
14							

Week	Sprint	Planned	Actual
1	SPR 1	0	0
2		5	2
3	SPR 2	9	8
4		18	11
5	SPR 3	27	27
6			
7	SPR 4		
8			
9	SPR 5		
10			
11	SPR 6		
12			
13	Release		
14			



Research

Sources

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

Meeting Logs

Meeting Log#1

Meeting Details

- Organizer: CS2450-002 Team Two
- Date: January 20, 2022
- Time: 7:55pm – 9:07pm
- Location: Microsoft Teams
- Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Tyler Deschamps

Topics Discussed

- Set roles:
  - Ethan Taylor: GUI Developer
  - Jaden Albrecht: Team Manager
  - Cody Strange: Scribe and Milestone document builder
  - Tyler Deschamps: Chart and Milestone document builder
- Tasks:
  - Create project email and google drive[Cody Strange]
  - Research Tkinter[Ethan Taylor]
  - Begin MoSCoW charts[Tyler Deschamps]
  - Research templates for PERT and Gantt charts[Tyler Deschamps]
  - Read chapters 11, 12, 13[Cody Strange, Ethan Taylor, Jaden Albrecht]
  - Set up meeting with Professor[Jaden Albrecht]

Summary

- Notes:
  - Everyone is equally confused about the project, but we have found areas that we can start in and are preparing for our first meeting with the Professor.
  - Took stock of talent and skills of teammates
    - Three have python Experience up to CS2420
    - One has had some workings with GUIs
    - Only four out of six members four out team
    - One has knowledge of Trello to use as a project management tool to start us off with
  - Currently everyone plans on participating in the role of being a programmer, this may vary depending on the workload of certain individuals

- Topics for next time:
  - More detailed description of everyone's strengths
    - Has anyone used Scrum?
    - Years of programming experience
  - What does everyone's schedule look like, how much time can everyone expect to dedicate
  - Clarification on roles

- Next Meeting: January 21, 2020 @ 6:00pm, Microsoft Team

Meeting Log#2

Meeting Details

- Organizer: CS2450-002 Team Two
- Date: January 21, 2022
- Time: 6:00pm – 6:41pm
- Location: Microsoft Teams
- Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Tyler Deschamps

Topics Discussed

- Agile methodologies
    - Crystal clear
    - Scrum
    - XP programming
    - Lean
    - AUP
- Requirements for Sprint one
    - MoSCoW charts
    - Gantt charts
    - PERTT charts
- Programming
    - Everyone is a programmer
    - Everyone has a partner who tests their code for them

Summary

- The Agile methodologies we chose to go with were Scrum and Crystal Clear
- We lack the manpower for XP programming
- File storage and organization complete
    - Google drive and project email created
- Ask about slide highlights from professor
- Team specific tools are established
    - Linked everyone to MS teams, counting shareholder
- Weekly meetings twice a week
    - Monday
    - Friday

Tasks

- Install Tkinter-everyone
- Schedule meeting with professor-Jaden Albrecht
- Next Meeting: January 24, 2020 @ 7:00pm, Microsoft Team

Meeting Log#3

Meeting Details

- Organizer: CS2450-002 Team Two
- Date: January 24, 2022
- Time: 7:00pm – 7:55pm
- Location: Microsoft Teams
- Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Tyler Deschamps

Topics Discussed

- Final revisions and overview of all Sprint#1 documents

Summary

- **Overview of WBS (Work Breakdown Structure)**
- **Overview of Pert chart**
- **Overview of Gantt chart**
- **Initialized MoSCoW chart**
- **Acquired prototypes**
- **Scheduled next team meeting for January 28th**
- Established role for new team member still undecided
- Began document MoSCoW chart ideas for sprint#2

Tasks

- Submit sprint#1
- Read chapters 5-6
- Begin preparations for sprint#2