
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

<i>Management</i>	3
<i>Procedures</i>	3
<i>Plans</i>	24
<i>Previous Sprint Items</i>	24
<i>Deployment</i>	27
<i>Scope</i>	27
<i>Cutover</i>	28
<i>Deployment Plan</i>	28
<i>Testing</i>	30
<i>Testing</i>	30
<i>Bug Tracking</i>	63
<i>Charts/Templates</i>	64
<i>Work breakdown structure</i>	65
<i>Pert Chart</i>	66
<i>Gantt Chart</i>	67
<i>Burndown Chart</i>	68
<i>Meeting Logs</i>	68
<i>Meeting Log#9</i>	70
<i>Meeting Log#10</i>	72

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?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

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.

<i>Validation Questions</i>	Yes	No	NA	Comments
1. Has the application been developed using an Integrated Development Environment that was approved by the OCIO?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

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.).

<i>Validation Questions</i>	Yes	No	NA	Comments
1. Is the presentation layer of the application free from business logic?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2. Has the presentation layer of the application been developed in accordance with prevailing industry standards?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

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.

<i>Validation Questions</i>	Yes	No	NA	Comments
1. Have precautions been taken to avoid data clashes?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

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.

<i>Validation Questions</i>	Yes	No	NA	Comments
1. Does the system have functionality to allow the user to revise their password and force user account expiry?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The user is able to change their password, but does not force user account expiry, because user accounts would be

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
				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?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3. Does the system meet the standard password requirements?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	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?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	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?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
6. Are passwords for privileged accounts different than passwords for non-privileged accounts?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	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?	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
2. Based on the application's Information Security Classification, does the application meet the auditing functional control requirements?	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

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?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

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.

<i>Validation Questions</i>	Yes	No	NA	Comments
2. Has code duplication been avoided?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

8. Consistency of Code

Code sections with similar functionality should be written in a clear, predictable, 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.

<i>Validation Questions</i>	Yes	No	NA	Comments
1. Is the code written in a consistent manner throughout the application?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2. Have all developers followed the same coding style and naming conventions?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
3. Have all developers followed the coding best practices as set out by the organization which owns the technology?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

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.

<i>Validation Questions</i>	Yes	No	NA	Comments
1. Does all application code include sufficient comments for support personnel?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2. Does each code unit have its own brief and accurate description?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

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?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2. Do the error messages contain minimal but meaningful information?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
3. Does the application avoid displaying system information in error messages?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
4. Are the error messages kept in a single location?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	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?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	We are manually logging any errors we find
2. Is logging being done on each server tier?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3. Are the logs kept in a single location / directory / database?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	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?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

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
5. Is the code that logs the error messages written in a modular way?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
6. Are the log files free of personally sensitive or identifiable information?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

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)?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2. Are field values being validated with regular expressions where possible (e.g. validating email addresses and dates for valid formats)?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	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.)?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
4. Are the validations being performed within the business logic, as well as on the presentation layer?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
5. Have the validations been written so that users cannot bypass them?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

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
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?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
7. Are user inputs being sanitized (without exceptions) according to OWASP recommendations?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

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?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	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?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

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?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

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.

<i>Validation Questions</i>	Yes	No	NA	Comments
2. Do all hard-coded values reside exclusively within configuration and constant, centralized locations? (Central Locations that enable changes without recompiling source code)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

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).

<i>Validation Questions</i>	Yes	No	Comments
1. Did the application pass all positive tests?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	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?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3. Have client testers completed the formal test plan in its entirety?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	We have not made a formal test plan yet
4. Did the application pass all tests included in the formal test plan?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
5. Have all positive / negative test cases and test case results been documented?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

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?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2. Have all previous tests been reconducted with the results compared against expected results?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
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?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
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?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

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)?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
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)?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3. Did the system perform well with a large number of concurrent users?	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
4. Did the system perform well with a large number of concurrent transactions?	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
5. Are end-users satisfied with the application's performance and responsiveness during everyday use?	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

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?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
2. Have special software/certificate requirements been documented?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3. Does the documentation provide expiration dates and instructions for renewal?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
4. Is the system / application free from trial versions of software?	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

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?	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	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?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	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?	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	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?	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
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)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

20. Source Code				
Validation Questions	Yes	No	NA	Comments
4. Have the 3 rd 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, 3 rd party products need to be provided)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

21. Database Design				
<p>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.</p>				
Validation Questions	Yes	No	NA	Comments
1. Have the database tables been normalized?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2. Keys based on sequence numbers have unique sequences.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
3. Are all keys and required fields set to 'not null' in all tables of the database?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
4. Have triggers, stored procedures, sequences, and constraints been properly utilized?	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

22. Transition To Support Personnel				
<p>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.</p> <p>The Application Services resources supporting an application should be granted access to development, test/staging, and production environments (as appropriate) prior to transition.</p>				
Validation Questions	Yes	No	NA	Comments
1. Have accounts been created on all servers for the appropriate support personnel?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
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)?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3. Have all server environments (development, test/staging, and production) been fully created?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

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.

<i>Validation Questions</i>	Yes	No	NA	Comments
4. Are all of the server environments entirely consistent with each other?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

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

Change Order Request Form

ID: 4

Detailed Description

The ability to search the employee database using any employee attribute, not just ID and last name. It would make searching the database a lot more flexible.

Change our search database function from just taking ID or last name to be able to take any employee object attribute/property as a parameter and based on that parameter, the function would search the database for that thing. We would then change the radio buttons to a combo box that would show all the potential parameters for the user to choose from.

Approved/Denied: Pending

Date Submitted: 20/03/2022

Man/Hrs: 2hrs

GUI side would be relatively short 20min

Change Order Request Form

ID: 6

Detailed Description

Depending on the user's permission level, a different visual theme will be shown throughout the GUI.

We would implement two distinct themes for each page based on user permission. Research on lighter themes still needs to be done and the specific themes still need to be decided.

Approved/Denied: Pending

Date Submitted: 20/03/2022

Man/Hrs: 2hrs

Should be quick once the research is complete

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.

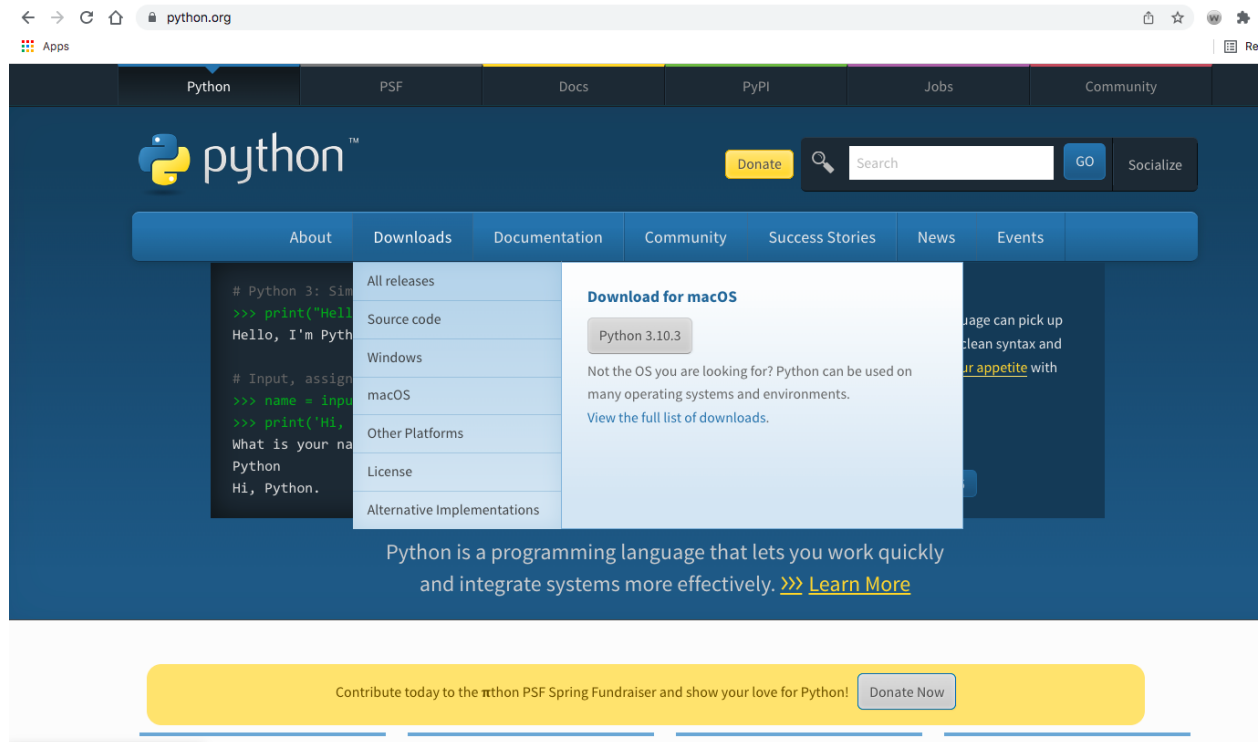
You may refer to this user manual at any time by clicking the “Help” button on the main view page.

1: Installing Python

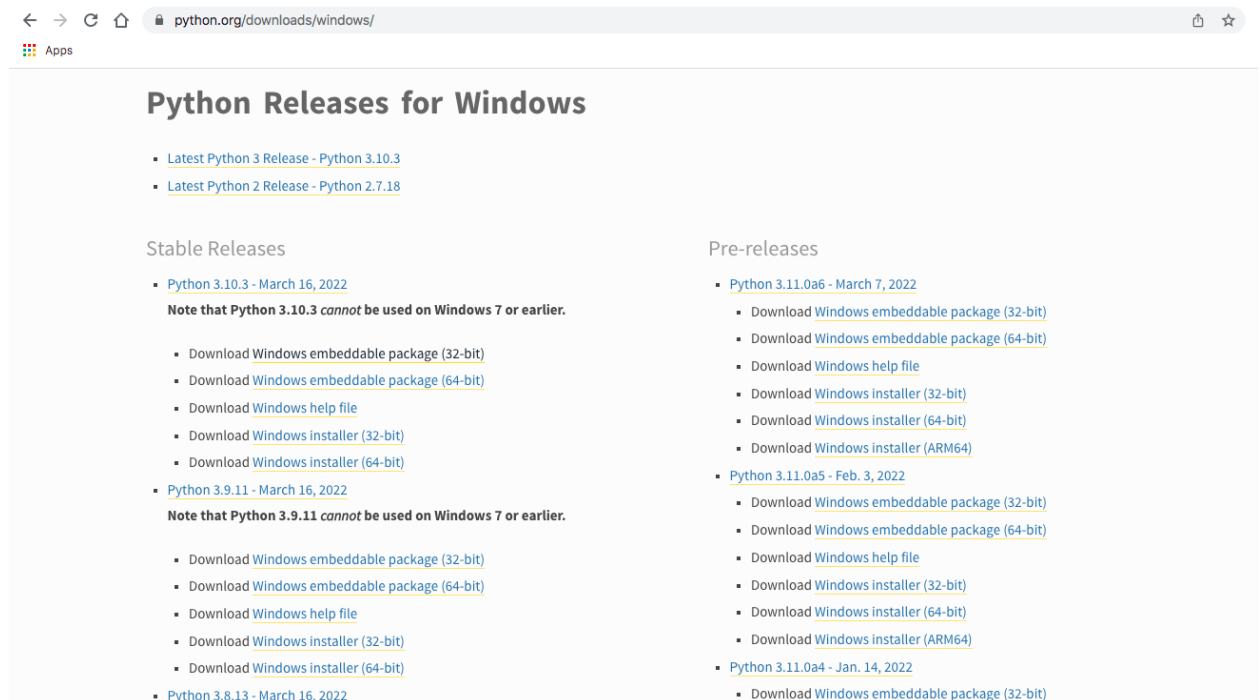
The first step is to install Python onto your computer. If you already have Python installed, then you may skip this step. To install Python, go to the following website:

<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.



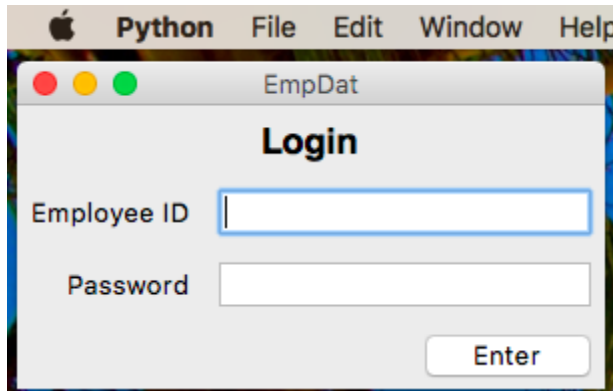
(Image of python website homepage)



(Image of python downloads page)

2: User Login

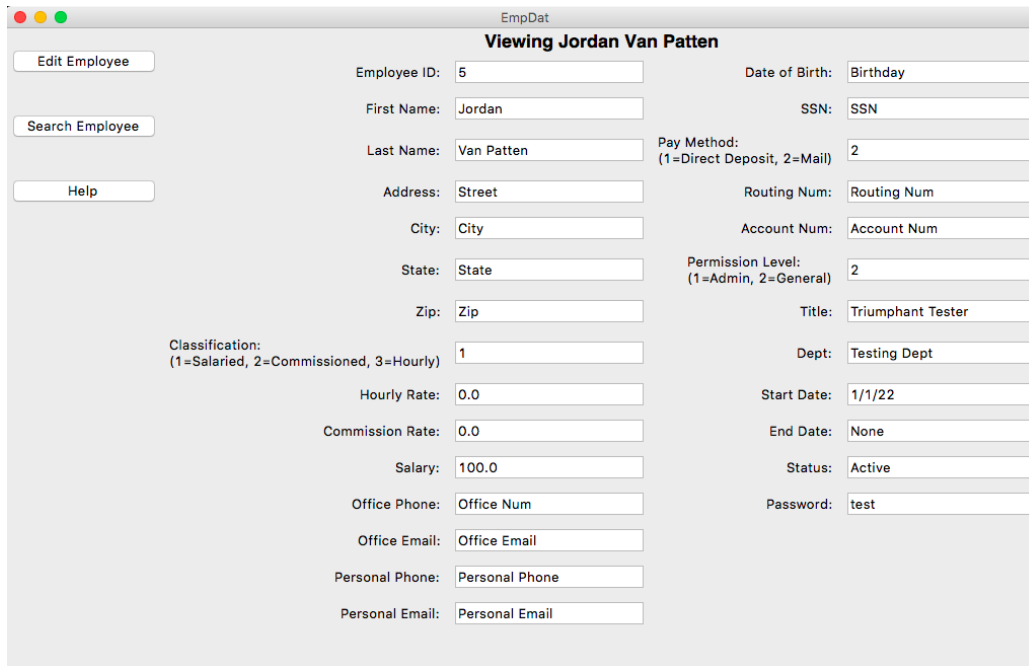
Once you have successfully installed Python and have started the application, you will be taken to the user login page. Here, you will enter your employee ID number and password into the provided fields. Next, click on the “Enter” button and the application will search for your information. If either 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.



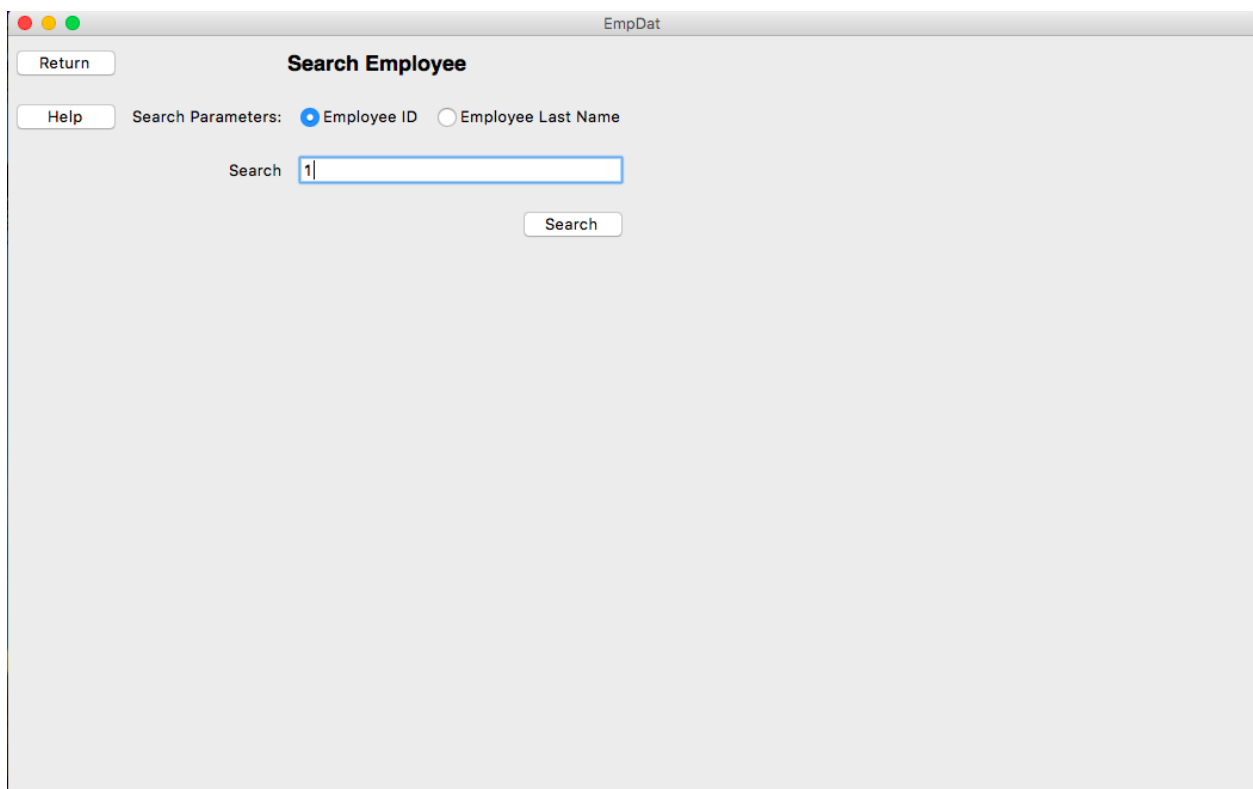
(Image of general view page)

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.

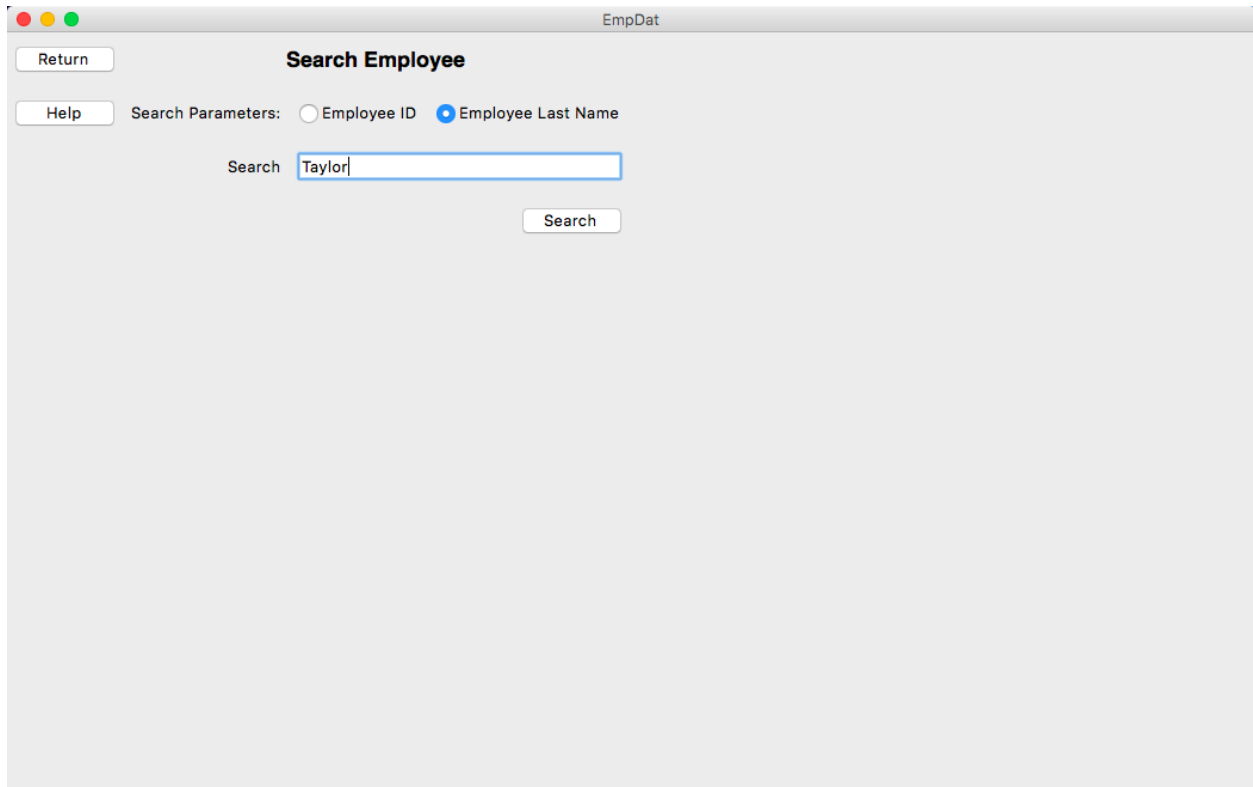


The screenshot shows a window titled "EmpDat" with a "Search Employee" section. On the left, there are two buttons: "Return" and "Help". To the right of these buttons, the text "Search Parameters:" is followed by two radio buttons. The first radio button, labeled "Employee ID", is selected (indicated by a blue dot). The second radio button, labeled "Employee Last Name", is unselected. Below the radio buttons, there is a text input field with the label "Search" to its left. The input field contains the number "1". To the right of the input field is a button labeled "Search".

(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.

The screenshot shows a window titled "EmpDat" with a "Search Employee" section. At the top left of the section is a "Return" button. Below it is a "Help" button. To the right of the "Help" button is the text "Search Parameters:" followed by two radio buttons. The first radio button is labeled "Employee ID" and is unselected. The second radio button is labeled "Employee Last Name" and is selected, indicated by a blue dot. Below the radio buttons is a search bar with the text "Search" to its left and "Taylor" entered inside. To the right of the search bar is a "Search" button.

(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.

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: 10000000000000000

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

(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.

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:

[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.

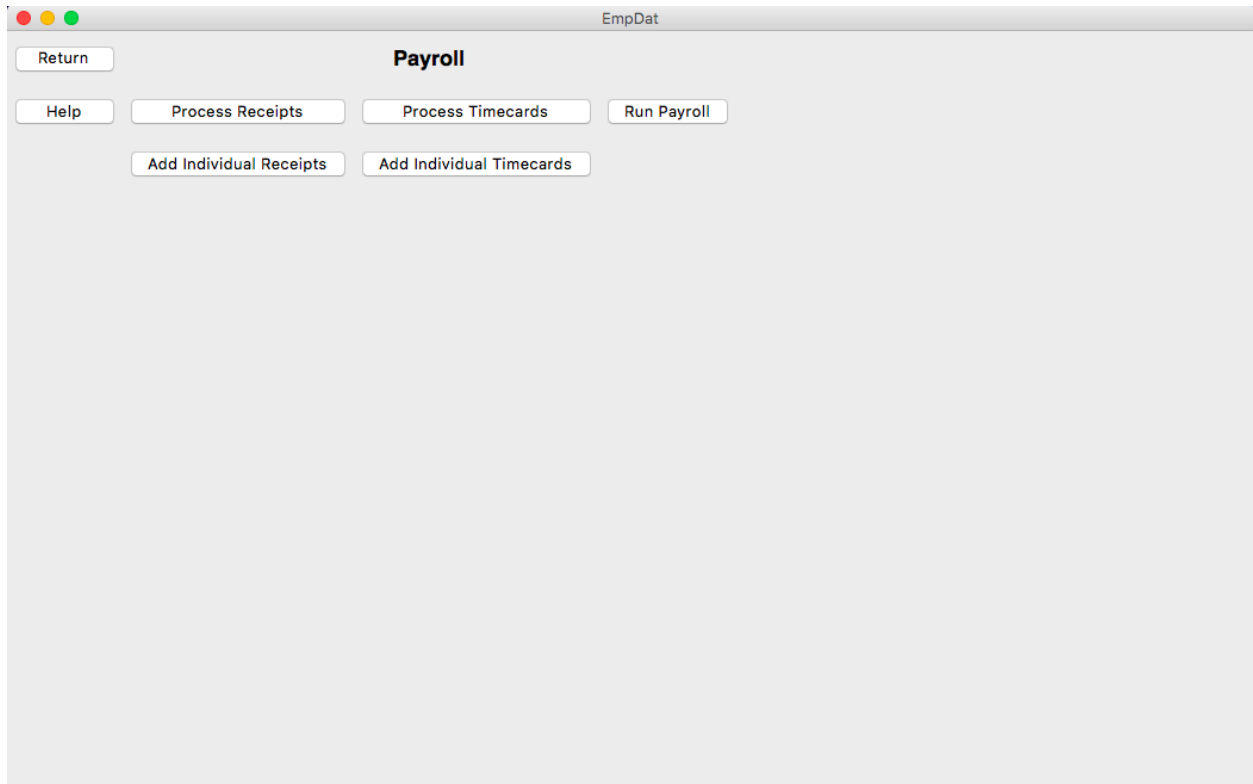
The screenshot shows the EmpDat application window. On the left is a sidebar with buttons: Edit Employee, Add Employee, Deactivate Employee (highlighted in blue), Search Employee, Run Payroll, and Help. The main area contains a form for employee data. A modal dialog box is open in the center with the title 'Do you wish to deactivate this employee?' and a rocket icon. It has 'Cancel' and 'OK' buttons. The form fields are as follows:

Field	Value
Date of Birth:	Birthday
SSN:	SSN
Address:	Street
City:	City
State:	State
Zip:	Zip
Classification:	1
Hourly Rate:	0.0
Commission Rate:	0.0
Salary:	100.0
Office Phone:	Office Num
Office Email:	Office Email
Personal Phone:	Personal Phone
Personal Email:	Personal Email
Routing Num:	Routing Num
Account Num:	Account Num
Permission Level:	2
Title:	Master Manager
Dept:	Managing Dept
Start Date:	1/1/22
End Date:	None
Status:	Active
Password:	test

[Deactivate an employee (requires administrative privileges)]

7: Running Payroll

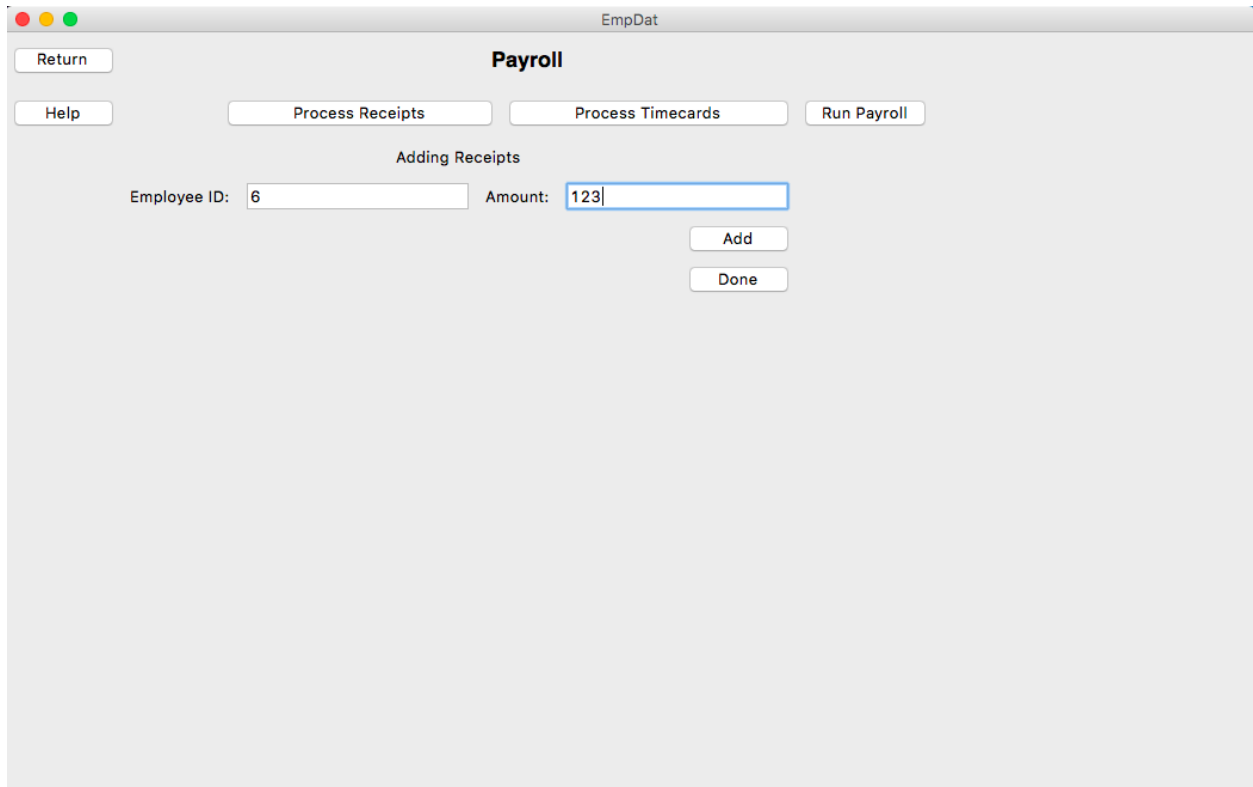
Another administrator-specific function is the ability to generate a payroll report of each employee in the system. The run payroll function generates a pay report using data stored in each employee's profile and exporting the information into a CSV file for your viewing.



(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.



The screenshot shows a window titled "EmpDat" with a "Payroll" section. On the left, there are buttons for "Return" and "Help". In the center, there are three buttons: "Process Receipts", "Process Timecards", and "Run Payroll". Below these, the "Adding Receipts" form is displayed. It includes two input fields: "Employee ID:" with the value "6" and "Amount:" with the value "123". Below the "Amount:" field, there are two buttons: "Add" and "Done".

(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-priority
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	Privileges	Employees are allowed access to data fields that their privileges 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 program

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

The screenshot shows a GUI window titled 'EmpDat' with a sub-header 'Viewing Ethan Taylor'. On the left side, there is a vertical menu with buttons: 'Edit Employee', 'Add Employee', 'Deactivate Employee', 'Search Employee', 'Run Payroll', and 'Help'. The main area contains a form with the following fields and values:

- Employee ID: 6
- First Name: Ethan
- Last Name: Taylor
- Address: Street
- City: City
- State: State
- Zip: Zip
- 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
- Classification: (1=Salaried, 2=Commissioned, 3=Hourly) 2
- Hourly Rate: 1.0
- Commission Rate: 10.0
- Salary: 100.0
- Office Phone: Office Num
- Office Email: Office Email
- Personal Phone: Personal Phone
- Personal Email: Personal Email

EmpDat

Payroll

Return

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

Date of Birth: Birthday

First Name: Ethan

SSN: SSN

Last Name: Taylor

Pay Method: (1=Direct Deposit, 2=Mail) 1

Address: Street

Routing Num: Routing Num

City: City

Account Num: Account Num

State: State

Permission Level: (1=Admin, 2=General) 1

Zip: Zip

Title: GUI Guy

Classification: (1=Salaried, 2=Commissioned, 3=Hourly) 2

Dept: GUI Dept

Hourly Rate: 10000000000000000

Start Date: 1/1/22

Commission Rate: 10.0

End Date: None

Salary: 100.0

Status: Active

Office Phone: Office Num

Password: test

Office Email: Office Email

Personal Phone: Personal Phone

Personal Email: Personal Email

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 an 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"],

```

```

emp["end"], emp["status"], emp["password"])

x = EmpDat_v5.EmpApp(emp_dict)
#CODE FOR TESTING LOGIN (Validate user class only)
def test_correct_1():
    #code that should pass username and password
    x.username.set("8")
    x.user_pass.set("test")
    x.validate_user()
    assert x.msg.get() != ("Invalid ID!") and ("Incorrect password or ID!") != x.msg.get()

def test_correct_2():
    x.username.set("2")
    x.user_pass.set("test")
    x.validate_user()
    assert x.msg.get() != ("Invalid ID!") and ("Incorrect password or ID!") != x.msg.get()

def test_correct_3():
    x.username.set("3")
    x.user_pass.set("test")
    x.validate_user()
    assert ("Invalid ID!") != x.msg.get() and ("Incorrect password or ID!") != x.msg.get()

def test_correct_4():
    x.username.set("4")
    x.user_pass.set("test")
    x.validate_user()
    assert ("Invalid ID!") != x.msg.get() and ("Incorrect password or ID!") != x.msg.get()

def test_correct_5():
    x.username.set("5")
    x.user_pass.set("test")
    x.validate_user()
    assert ("Invalid ID!") != x.msg.get() and ("Incorrect password or ID!") != x.msg.get()

def test_correct_6():
    x.username.set("6")
    x.user_pass.set("test")
    x.validate_user()
    assert ("Invalid ID!") != x.msg.get() and ("Incorrect password or ID!") != x.msg.get()

```



```

def test_correct_7():
    x.username.set("7")
    x.user_pass.set("test")
    x.validate_user()
    assert ("Invalid ID!") != x.msg.get() and ("Incorrect password or ID!") != x.msg.get()

#code testing for incorrect usernames but correct passwords
def test_incorrect_1():
    x.username.set("jim")
    x.user_pass.set("test")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrect_2():
    x.username.set("tony")
    x.user_pass.set("test")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrect_3():
    x.username.set("123")
    x.user_pass.set("test")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrect_4():
    x.username.set("57")
    x.user_pass.set("test")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrect_5():
    x.username.set("gfad")
    x.user_pass.set("test")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrect_6():
    x.username.set("wrong")
    x.user_pass.set("test")

```

```

x.validate_user()
assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrect_7():
    x.username.set("almost")
    x.user_pass.set("test")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrect_8():
    x.username.set("so close")
    x.user_pass.set("test")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

#code that tests for incorrect passwords, but correct username

def test_incorrectP_1():
    x.username.set("1")
    x.user_pass.set("tests")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrectP_2():
    x.username.set("1")
    x.user_pass.set("test!")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrectP_3():
    x.username.set("1")
    x.user_pass.set("tesT")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrectP_4():
    x.username.set("1")
    x.user_pass.set("Test")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

```

```

def test_incorrectP_5():
    x.username.set("1")
    x.user_pass.set("nope")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrectP_6():
    x.username.set("1")
    x.user_pass.set("tESt")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrectP_7():
    x.username.set("1")
    x.user_pass.set("password")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrectP_8():
    x.username.set("1")
    x.user_pass.set("letMeIn")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrectP_9():
    x.username.set("1")
    x.user_pass.set("please")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrectP_10():
    x.username.set("1")
    x.user_pass.set("prettyplease")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

def test_incorrectP_11():
    x.username.set("1")
    x.user_pass.set("123456")
    x.validate_user()
    assert ("Invalid ID!") == x.msg.get() or ("Incorrect password or ID!") == x.msg.get()

```

#CODE FOR TESTING VIEW PAGE

```
def test_nonAdminPass_1():
    x.username.set("8")
    x.user_pass.set("test")
    x.validate_user()
    v = EmpDat_v5.View_Page(parent=x.container, controller=x)
    if (emp_dict[x.username.get()].is_admin() == True):
        assert v.permission.get() == '1'
    else:
        assert v.permission.get() == '2'
    assert v.id.get() == emp_dict[x.username.get()].get_id()
    assert v.f_name.get() == emp_dict[x.username.get()].get_first_name()
    assert v.l_name.get() == emp_dict[x.username.get()].get_last_name()
    assert v.street.get() == emp_dict[x.username.get()].get_address()
    assert v.city.get() == emp_dict[x.username.get()].get_city()
    assert v.emp_state.get() == emp_dict[x.username.get()].get_state()
    assert v.zip.get() == emp_dict[x.username.get()].get_zip()
    assert v.classification.get() == emp_dict[x.username.get()].get_class()
    assert v.hourly.get() == str(emp_dict[x.username.get()].get_hourly_rate())
    assert v.commissioned.get() == str(emp_dict[x.username.get()].get_commission_rate())
    assert v.salary.get() == str(emp_dict[x.username.get()].get_salary())
    assert v.o_phone.get() == emp_dict[x.username.get()].get_office_phone()
    assert v.o_email.get() == emp_dict[x.username.get()].get_office_email()
    assert v.p_phone.get() == emp_dict[x.username.get()].get_personal_phone()
    assert v.p_email.get() == emp_dict[x.username.get()].get_personal_email()
    assert v.dob.get() == emp_dict[x.username.get()].get_dob()
    assert v.ssn.get() == emp_dict[x.username.get()].get_ssn()
    assert v.pay_type.get() == emp_dict[x.username.get()].get_pay_method()
    assert v.routing_num.get() == emp_dict[x.username.get()].get_routing()
    assert v.account_entry.get() == emp_dict[x.username.get()].get_account()
    assert v.emp_title.get() == emp_dict[x.username.get()].get_title()
    assert v.emp_dept.get() == emp_dict[x.username.get()].get_dept()
    assert v.start_date.get() == emp_dict[x.username.get()].get_start()
    assert v.end_date.get() == str(emp_dict[x.username.get()].get_end())
    if (emp_dict[x.username.get()].get_status() == True):
        assert v.emp_status.get() == "Active"
    else:
```

```

        assert v.emp_status.get() == "Deactivated"

def test_nonAdminPass_2():
    x.username.set("2")
    x.user_pass.set("test")
    x.validate_user()
    v = EmpDat_v5.View_Page(parent=x.container, controller=x)
    if (emp_dict[x.username.get()].is_admin() == True):
        assert v.permission.get() == '1'
    else:
        assert v.permission.get() == '2'
    assert v.id.get() == emp_dict[x.username.get()].get_id()
    assert v.f_name.get() == emp_dict[x.username.get()].get_first_name()
    assert v.l_name.get() == emp_dict[x.username.get()].get_last_name()
    assert v.street.get() == emp_dict[x.username.get()].get_address()
    assert v.city.get() == emp_dict[x.username.get()].get_city()
    assert v.emp_state.get() == emp_dict[x.username.get()].get_state()
    assert v.zip.get() == emp_dict[x.username.get()].get_zip()
    assert v.classification.get() == emp_dict[x.username.get()].get_class()
    assert v.hourly.get() == str(emp_dict[x.username.get()].get_hourly_rate())
    assert v.commissioned.get() == str(emp_dict[x.username.get()].get_commission_rate())
    assert v.salary.get() == str(emp_dict[x.username.get()].get_salary())
    assert v.o_phone.get() == emp_dict[x.username.get()].get_office_phone()
    assert v.o_email.get() == emp_dict[x.username.get()].get_office_email()
    assert v.p_phone.get() == emp_dict[x.username.get()].get_personal_phone()
    assert v.p_email.get() == emp_dict[x.username.get()].get_personal_email()
    assert v.dob.get() == emp_dict[x.username.get()].get_dob()
    assert v.ssn.get() == emp_dict[x.username.get()].get_ssn()
    assert v.pay_type.get() == emp_dict[x.username.get()].get_pay_method()
    assert v.routing_num.get() == emp_dict[x.username.get()].get_routing()
    assert v.account_entry.get() == emp_dict[x.username.get()].get_account()
    assert v.emp_title.get() == emp_dict[x.username.get()].get_title()
    assert v.emp_dept.get() == emp_dict[x.username.get()].get_dept()
    assert v.start_date.get() == emp_dict[x.username.get()].get_start()
    assert v.end_date.get() == str(emp_dict[x.username.get()].get_end())
    if (emp_dict[x.username.get()].get_status() == True):
        assert v.emp_status.get() == "Active"
    else:
        assert v.emp_status.get() == "Deactivated"

def test_nonAdminPass_3():
    x.username.set("3")

```

```

x.user_pass.set("test")
x.validate_user()
v = EmpDat_v5.View_Page(parent=x.container, controller=x)
if (emp_dict[x.username.get()].is_admin() == True):
    assert v.permission.get() == '1'
else:
    assert v.permission.get() == '2'
assert v.id.get() == emp_dict[x.username.get()].get_id()
assert v.f_name.get() == emp_dict[x.username.get()].get_first_name()
assert v.l_name.get() == emp_dict[x.username.get()].get_last_name()
assert v.street.get() == emp_dict[x.username.get()].get_address()
assert v.city.get() == emp_dict[x.username.get()].get_city()
assert v.emp_state.get() == emp_dict[x.username.get()].get_state()
assert v.zip.get() == emp_dict[x.username.get()].get_zip()
assert v.classification.get() == emp_dict[x.username.get()].get_class()
assert v.hourly.get() == str(emp_dict[x.username.get()].get_hourly_rate())
assert v.commissioned.get() == str(emp_dict[x.username.get()].get_commission_rate())
assert v.salary.get() == str(emp_dict[x.username.get()].get_salary())
assert v.o_phone.get() == emp_dict[x.username.get()].get_office_phone()
assert v.o_email.get() == emp_dict[x.username.get()].get_office_email()
assert v.p_phone.get() == emp_dict[x.username.get()].get_personal_phone()
assert v.p_email.get() == emp_dict[x.username.get()].get_personal_email()
assert v.dob.get() == emp_dict[x.username.get()].get_dob()
assert v.ssn.get() == emp_dict[x.username.get()].get_ssn()
assert v.pay_type.get() == emp_dict[x.username.get()].get_pay_method()
assert v.routing_num.get() == emp_dict[x.username.get()].get_routing()
assert v.account_entry.get() == emp_dict[x.username.get()].get_account()
assert v.emp_title.get() == emp_dict[x.username.get()].get_title()
assert v.emp_dept.get() == emp_dict[x.username.get()].get_dept()
assert v.start_date.get() == emp_dict[x.username.get()].get_start()
assert v.end_date.get() == str(emp_dict[x.username.get()].get_end())
if (emp_dict[x.username.get()].get_status() == True):
    assert v.emp_status.get() == "Active"
else:
    assert v.emp_status.get() == "Deactivated"

def test_nonAdminPass_4():
    x.username.set("4")
    x.user_pass.set("test")
    x.validate_user()
    v = EmpDat_v5.View_Page(parent=x.container, controller=x)

```

```

if (emp_dict[x.username.get()].is_admin() == True):
    assert v.permission.get() == '1'
else:
    assert v.permission.get() == '2'
assert v.id.get() == emp_dict[x.username.get()].get_id()
assert v.f_name.get() == emp_dict[x.username.get()].get_first_name()
assert v.l_name.get() == emp_dict[x.username.get()].get_last_name()
assert v.street.get() == emp_dict[x.username.get()].get_address()
assert v.city.get() == emp_dict[x.username.get()].get_city()
assert v.emp_state.get() == emp_dict[x.username.get()].get_state()
assert v.zip.get() == emp_dict[x.username.get()].get_zip()
assert v.classification.get() == emp_dict[x.username.get()].get_class()
assert v.hourly.get() == str(emp_dict[x.username.get()].get_hourly_rate())
assert v.commissioned.get() == str(emp_dict[x.username.get()].get_commission_rate())
assert v.salary.get() == str(emp_dict[x.username.get()].get_salary())
assert v.o_phone.get() == emp_dict[x.username.get()].get_office_phone()
assert v.o_email.get() == emp_dict[x.username.get()].get_office_email()
assert v.p_phone.get() == emp_dict[x.username.get()].get_personal_phone()
assert v.p_email.get() == emp_dict[x.username.get()].get_personal_email()
assert v.dob.get() == emp_dict[x.username.get()].get_dob()
assert v.ssn.get() == emp_dict[x.username.get()].get_ssn()
assert v.pay_type.get() == emp_dict[x.username.get()].get_pay_method()
assert v.routing_num.get() == emp_dict[x.username.get()].get_routing()
assert v.account_entry.get() == emp_dict[x.username.get()].get_account()
assert v.emp_title.get() == emp_dict[x.username.get()].get_title()
assert v.emp_dept.get() == emp_dict[x.username.get()].get_dept()
assert v.start_date.get() == emp_dict[x.username.get()].get_start()
assert v.end_date.get() == str(emp_dict[x.username.get()].get_end())
if (emp_dict[x.username.get()].get_status() == True):
    assert v.emp_status.get() == "Active"
else:
    assert v.emp_status.get() == "Deactivated"

def test_AdminPass_1():
    x.username.set("8")
    x.user_pass.set("test")
    x.validate_user()
    v = EmpDat_v5.View_Page(parent=x.container, controller=x)
    if (emp_dict[x.username.get()].is_admin() == True):
        assert v.permission.get() == '1'
    else:

```

```

        assert v.permission.get() == '2'
    assert v.id.get() == emp_dict[x.username.get()].get_id()
    assert v.f_name.get() == emp_dict[x.username.get()].get_first_name()
    assert v.l_name.get() == emp_dict[x.username.get()].get_last_name()
    assert v.street.get() == emp_dict[x.username.get()].get_address()
    assert v.city.get() == emp_dict[x.username.get()].get_city()
    assert v.emp_state.get() == emp_dict[x.username.get()].get_state()
    assert v.zip.get() == emp_dict[x.username.get()].get_zip()
    assert v.classification.get() == emp_dict[x.username.get()].get_class()
    assert v.hourly.get() == str(emp_dict[x.username.get()].get_hourly_rate())
    assert v.commissioned.get() == str(emp_dict[x.username.get()].get_commission_rate())
    assert v.salary.get() == str(emp_dict[x.username.get()].get_salary())
    assert v.o_phone.get() == emp_dict[x.username.get()].get_office_phone()
    assert v.o_email.get() == emp_dict[x.username.get()].get_office_email()
    assert v.p_phone.get() == emp_dict[x.username.get()].get_personal_phone()
    assert v.p_email.get() == emp_dict[x.username.get()].get_personal_email()
    assert v.dob.get() == emp_dict[x.username.get()].get_dob()
    assert v.ssn.get() == emp_dict[x.username.get()].get_ssn()
    assert v.pay_type.get() == emp_dict[x.username.get()].get_pay_method()
    assert v.routing_num.get() == emp_dict[x.username.get()].get_routing()
    assert v.account_entry.get() == emp_dict[x.username.get()].get_account()
    assert v.emp_title.get() == emp_dict[x.username.get()].get_title()
    assert v.emp_dept.get() == emp_dict[x.username.get()].get_dept()
    assert v.start_date.get() == emp_dict[x.username.get()].get_start()
    assert v.end_date.get() == str(emp_dict[x.username.get()].get_end())
    if (emp_dict[x.username.get()].get_status() == True):
        assert v.emp_status.get() == "Active"
    else:
        assert v.emp_status.get() == "Deactivated"

def test_AdminPass_2():
    x.username.set("6")
    x.user_pass.set("test")
    x.validate_user()
    v = EmpDat_v5.View_Page(parent=x.container, controller=x)
    if (emp_dict[x.username.get()].is_admin() == True):
        assert v.permission.get() == '1'
    else:
        assert v.permission.get() == '2'
    assert v.id.get() == emp_dict[x.username.get()].get_id()
    assert v.f_name.get() == emp_dict[x.username.get()].get_first_name()

```



```

assert v.l_name.get() == emp_dict[x.username.get()].get_last_name()
assert v.street.get() == emp_dict[x.username.get()].get_address()
assert v.city.get() == emp_dict[x.username.get()].get_city()
assert v.emp_state.get() == emp_dict[x.username.get()].get_state()
assert v.zip.get() == emp_dict[x.username.get()].get_zip()
assert v.classification.get() == emp_dict[x.username.get()].get_class()
assert v.hourly.get() == str(emp_dict[x.username.get()].get_hourly_rate())
assert v.commissioned.get() == str(emp_dict[x.username.get()].get_commission_rate())
assert v.salary.get() == str(emp_dict[x.username.get()].get_salary())
assert v.o_phone.get() == emp_dict[x.username.get()].get_office_phone()
assert v.o_email.get() == emp_dict[x.username.get()].get_office_email()
assert v.p_phone.get() == emp_dict[x.username.get()].get_personal_phone()
assert v.p_email.get() == emp_dict[x.username.get()].get_personal_email()
assert v.dob.get() == emp_dict[x.username.get()].get_dob()
assert v.ssn.get() == emp_dict[x.username.get()].get_ssn()
assert v.pay_type.get() == emp_dict[x.username.get()].get_pay_method()
assert v.routing_num.get() == emp_dict[x.username.get()].get_routing()
assert v.account_entry.get() == emp_dict[x.username.get()].get_account()
assert v.emp_title.get() == emp_dict[x.username.get()].get_title()
assert v.emp_dept.get() == emp_dict[x.username.get()].get_dept()
assert v.start_date.get() == emp_dict[x.username.get()].get_start()
assert v.end_date.get() == str(emp_dict[x.username.get()].get_end())
if (emp_dict[x.username.get()].get_status() == True):
    assert v.emp_status.get() == "Active"
else:
    assert v.emp_status.get() == "Deactivated"

```

#CODE FOR TESTING EDIT PAGE

```

def test_edit_1():
    x.username.set("8")
    x.user_pass.set("test")
    x.validate_user()
    v = EmpDat_v5.View_Page(parent=x.container, controller=x)
    e = EmpDat_v5.Edit_Page(v, controller=x)
    e.f_name.set("FirstNameChanged")
    e.l_name.set("LastNameChanged")
    e.street.set("StreetChanged")
    e.city.set("CityChanged")
    e.zip.set("ZIPChanged")
    e.classification.set("2")
    e.pay_type.set("1")

```

```

e.salary.set("100.00")
e.commissioned.set("10.00")
e.hourly.set("1.00")
e.routing_num.set("RoutingChanged")
e.account_num.set("AccountChanged")
e.o_phone.set("OfficePChanged")
e.p_phone.set("PersonalPChanged")
e.o_email.set("OfficeEChanged")
e.p_email.set("PersonalEChanged")
e.dob.set("BirthdayChanged")
e.ssn.set("SSNChanged")
e.emp_title.set("MasterChanged")
e.emp_dept.set("DeptChanged")
e.start_date.set("1/2/2022")
e.update_emp()
if (emp_dict[x.username.get()].is_admin() == True):
    assert v.permission.get() == '1'
else:
    assert v.permission.get() == '2'
assert emp_dict[x.username.get()].get_first_name() == "FirstNameChanged"
assert emp_dict[x.username.get()].get_last_name() == "LastNameChanged"
assert emp_dict[x.username.get()].get_address() == "StreetChanged"
assert emp_dict[x.username.get()].get_city() == "CityChanged"
assert emp_dict[x.username.get()].get_zip() == "ZIPChanged"
assert emp_dict[x.username.get()].get_class() == "2"
assert str(emp_dict[x.username.get()].get_hourly_rate()) == "1.00"
assert str(emp_dict[x.username.get()].get_commission_rate()) == "10.00"
assert str(emp_dict[x.username.get()].get_salary()) == "100.00"
assert emp_dict[x.username.get()].get_office_phone() == "OfficePChanged"
assert emp_dict[x.username.get()].get_office_email() == "OfficeEChanged"
assert emp_dict[x.username.get()].get_personal_phone() == "PersonalPChanged"
assert emp_dict[x.username.get()].get_personal_email() == "PersonalEChanged"
assert emp_dict[x.username.get()].get_dob() == "BirthdayChanged"
assert emp_dict[x.username.get()].get_ssn() == "SSNChanged"
assert emp_dict[x.username.get()].get_pay_method() == "1"
assert emp_dict[x.username.get()].get_routing() == "RoutingChanged"
assert emp_dict[x.username.get()].get_account() == "AccountChanged"
assert emp_dict[x.username.get()].get_title() == "MasterChanged"
assert emp_dict[x.username.get()].get_dept() == "DeptChanged"
assert emp_dict[x.username.get()].get_start() == "1/2/2022"
if (emp_dict[x.username.get()].get_status() == True):

```

```

        assert e.emp_status.get() == "Active"
    else:
        assert e.emp_status.get() == "Deactivated"

def test_edit_2():
    x.username.set("8")
    x.user_pass.set("test")
    x.validate_user()
    v = EmpDat_v5.View_Page(parent=x.container, controller=x)
    e = EmpDat_v5.Edit_Page(v, controller=x)
    e.f_name.set("FirstNameChanged")
    e.l_name.set("LastNameChanged")
    e.street.set("StreetChanged")
    e.city.set("CityChanged")
    e.zip.set("ZIPChanged")
    e.classification.set("2")
    e.pay_type.set("1")
    e.salary.set("100.00")
    e.commissioned.set("10.00")
    e.hourly.set("1.00")
    e.routing_num.set("RoutingChanged")
    e.account_num.set("AccountChanged")
    e.o_phone.set("OfficePChanged")
    e.p_phone.set("PersonalPChanged")
    e.o_email.set("OfficeEChanged")
    e.p_email.set("PersonalEChanged")
    e.dob.set("BirthdayChanged")
    e.ssn.set("SSNChanged")
    e.emp_title.set("MasterChanged")
    e.emp_dept.set("DeptChanged")
    e.start_date.set("1/2/2022")
    e.update_emp()
    if (emp_dict[x.username.get()].is_admin() == True):
        assert v.permission.get() == '1'
    else:
        assert v.permission.get() == '2'
    assert emp_dict[x.username.get()].get_first_name() == "FirstNameChanged"
    assert emp_dict[x.username.get()].get_last_name() == "LastNameChanged"
    assert emp_dict[x.username.get()].get_address() == "StreetChanged"
    assert emp_dict[x.username.get()].get_city() == "CityChanged"
    assert emp_dict[x.username.get()].get_zip() == "ZIPChanged"

```

```

assert emp_dict[x.username.get()].get_class() == "2"
assert str(emp_dict[x.username.get()].get_hourly_rate()) == "1.00"
assert str(emp_dict[x.username.get()].get_commission_rate()) == "10.00"
assert str(emp_dict[x.username.get()].get_salary()) == "100.00"
assert emp_dict[x.username.get()].get_office_phone() == "OfficePChanged"
assert emp_dict[x.username.get()].get_office_email() == "OfficeEChanged"
assert emp_dict[x.username.get()].get_personal_phone() == "PersonalPChanged"
assert emp_dict[x.username.get()].get_personal_email() == "PersonalEChanged"
assert emp_dict[x.username.get()].get_dob() == "BirthdayChanged"
assert emp_dict[x.username.get()].get_ssn() == "SSNChanged"
assert emp_dict[x.username.get()].get_pay_method() == "1"
assert emp_dict[x.username.get()].get_routing() == "RoutingChanged"
assert emp_dict[x.username.get()].get_account() == "AccountChanged"
assert emp_dict[x.username.get()].get_title() == "MasterChanged"
assert emp_dict[x.username.get()].get_dept() == "DeptChanged"
assert emp_dict[x.username.get()].get_start() == "1/2/2022"
if (emp_dict[x.username.get()].get_status() == True):
    assert e.emp_status.get() == "Active"
else:
    assert e.emp_status.get() == "Deactivated"

def test_edit_3():
    x.username.set("2")
    x.user_pass.set("test")
    x.validate_user()
    v = EmpDat_v5.View_Page(parent=x.container, controller=x)
    e = EmpDat_v5.Edit_Page(v, controller=x)
    e.f_name.set("FirstNameChanged")
    e.l_name.set("LastNameChanged")
    e.street.set("StreetChanged")
    e.city.set("CityChanged")
    e.zip.set("ZIPChanged")
    e.classification.set("1")
    e.pay_type.set("1")
    e.salary.set("100.00")
    e.commissioned.set("10.00")
    e.hourly.set("1.00")
    e.routing_num.set("RoutingChanged")
    e.account_num.set("AccountChanged")
    e.o_phone.set("OfficePChanged")
    e.p_phone.set("PersonalPChanged")

```

```

e.o_email.set("OfficeEChanged")
e.p_email.set("PersonalEChanged")
e.dob.set("BirthdayChanged")
e.ssn.set("SSNChanged")
e.emp_title.set("MasterChanged")
e.emp_dept.set("DeptChanged")
e.start_date.set("1/2/2022")
e.update_emp()
if (emp_dict[x.username.get()].is_admin() == True):
    assert v.permission.get() == '1'
else:
    assert v.permission.get() == '2'
assert emp_dict[x.username.get()].get_first_name() == "FirstNameChanged"
assert emp_dict[x.username.get()].get_last_name() == "LastNameChanged"
assert emp_dict[x.username.get()].get_address() == "StreetChanged"
assert emp_dict[x.username.get()].get_city() == "CityChanged"
assert emp_dict[x.username.get()].get_zip() == "ZIPChanged"
assert emp_dict[x.username.get()].get_class() == "1"
assert str(emp_dict[x.username.get()].get_hourly_rate()) == "1.00"
assert str(emp_dict[x.username.get()].get_commission_rate()) == "10.00"
assert str(emp_dict[x.username.get()].get_salary()) == "100.00"
assert emp_dict[x.username.get()].get_office_phone() == "OfficePChanged"
assert emp_dict[x.username.get()].get_office_email() == "OfficeEChanged"
assert emp_dict[x.username.get()].get_personal_phone() == "PersonalPChanged"
assert emp_dict[x.username.get()].get_personal_email() == "PersonalEChanged"
assert emp_dict[x.username.get()].get_dob() == "BirthdayChanged"
assert emp_dict[x.username.get()].get_ssn() == "SSNChanged"
assert emp_dict[x.username.get()].get_pay_method() == "1"
assert emp_dict[x.username.get()].get_routing() == "RoutingChanged"
assert emp_dict[x.username.get()].get_account() == "AccountChanged"
assert emp_dict[x.username.get()].get_title() == "MasterChanged"
assert emp_dict[x.username.get()].get_dept() == "DeptChanged"
assert emp_dict[x.username.get()].get_start() == "1/2/2022"
if (emp_dict[x.username.get()].get_status() == True):
    assert e.emp_status.get() == "Active"
else:
    assert e.emp_status.get() == "Deactivated"

def test_edit_4():
    x.username.set("3")
    x.user_pass.set("test")

```

```

x.validate_user()
v = EmpDat_v5.View_Page(parent=x.container, controller=x)
e = EmpDat_v5.Edit_Page(v, controller=x)
e.f_name.set("FirstNameChanged")
e.l_name.set("LastNameChanged")
e.street.set("StreetChanged")
e.city.set("CityChanged")
e.zip.set("ZIPChanged")
e.classification.set("2")
e.pay_type.set("1")
e.salary.set("100.00")
e.commissioned.set("10.00")
e.hourly.set("1.00")
e.routing_num.set("RoutingChanged")
e.account_num.set("AccountChanged")
e.o_phone.set("OfficePChanged")
e.p_phone.set("PersonalPChanged")
e.o_email.set("OfficeEChanged")
e.p_email.set("PersonalEChanged")
e.dob.set("BirthdayChanged")
e.ssn.set("SSNChanged")
e.emp_title.set("MasterChanged")
e.emp_dept.set("DeptChanged")
e.start_date.set("1/2/2022")
e.update_emp()
if (emp_dict[x.username.get()].is_admin() == True):
    assert v.permission.get() == '1'
else:
    assert v.permission.get() == '2'
assert emp_dict[x.username.get()].get_first_name() == "FirstNameChanged"
assert emp_dict[x.username.get()].get_last_name() == "LastNameChanged"
assert emp_dict[x.username.get()].get_address() == "StreetChanged"
assert emp_dict[x.username.get()].get_city() == "CityChanged"
assert emp_dict[x.username.get()].get_zip() == "ZIPChanged"
assert emp_dict[x.username.get()].get_class() == "2"
assert str(emp_dict[x.username.get()].get_hourly_rate()) == "1.00"
assert str(emp_dict[x.username.get()].get_commission_rate()) == "10.00"
assert str(emp_dict[x.username.get()].get_salary()) == "100.00"
assert emp_dict[x.username.get()].get_office_phone() == "OfficePChanged"
assert emp_dict[x.username.get()].get_office_email() == "OfficeEChanged"
assert emp_dict[x.username.get()].get_personal_phone() == "PersonalPChanged"

```

```

assert emp_dict[x.username.get()].get_personal_email() == "PersonalEChanged"
assert emp_dict[x.username.get()].get_dob() == "BirthdayChanged"
assert emp_dict[x.username.get()].get_ssn() == "SSNChanged"
assert emp_dict[x.username.get()].get_pay_method() == "1"
assert emp_dict[x.username.get()].get_routing() == "RoutingChanged"
assert emp_dict[x.username.get()].get_account() == "AccountChanged"
assert emp_dict[x.username.get()].get_title() == "MasterChanged"
assert emp_dict[x.username.get()].get_dept() == "DeptChanged"
assert emp_dict[x.username.get()].get_start() == "1/2/2022"
if (emp_dict[x.username.get()].get_status() == True):
    assert e.emp_status.get() == "Active"
else:
    assert e.emp_status.get() == "Deactivated"

def test_edit_5():
    x.username.set("4")
    x.user_pass.set("test")
    x.validate_user()
    v = EmpDat_v5.View_Page(parent=x.container, controller=x)
    e = EmpDat_v5.Edit_Page(v, controller=x)
    e.f_name.set("FirstNameChanged")
    e.l_name.set("LastNameChanged")
    e.street.set("StreetChanged")
    e.city.set("CityChanged")
    e.zip.set("ZIPChanged")
    e.classification.set("2")
    e.pay_type.set("1")
    e.salary.set("100.00")
    e.commissioned.set("10.00")
    e.hourly.set("1.00")
    e.routing_num.set("RoutingChanged")
    e.account_num.set("AccountChanged")
    e.o_phone.set("OfficePChanged")
    e.p_phone.set("PersonalPChanged")
    e.o_email.set("OfficeEChanged")
    e.p_email.set("PersonalEChanged")
    e.dob.set("BirthdayChanged")
    e.ssn.set("SSNChanged")
    e.emp_title.set("MasterChanged")
    e.emp_dept.set("DeptChanged")
    e.start_date.set("1/2/2022")

```

```

e.update_emp()
if (emp_dict[x.username.get()].is_admin() == True):
    assert v.permission.get() == '1'
else:
    assert v.permission.get() == '2'
assert emp_dict[x.username.get()].get_first_name() == "FirstNameChanged"
assert emp_dict[x.username.get()].get_last_name() == "LastNameChanged"
assert emp_dict[x.username.get()].get_address() == "StreetChanged"
assert emp_dict[x.username.get()].get_city() == "CityChanged"
assert emp_dict[x.username.get()].get_zip() == "ZIPChanged"
assert emp_dict[x.username.get()].get_class() == "2"
assert str(emp_dict[x.username.get()].get_hourly_rate()) == "1.00"
assert str(emp_dict[x.username.get()].get_commission_rate()) == "10.00"
assert str(emp_dict[x.username.get()].get_salary()) == "100.00"
assert emp_dict[x.username.get()].get_office_phone() == "OfficePChanged"
assert emp_dict[x.username.get()].get_office_email() == "OfficeEChanged"
assert emp_dict[x.username.get()].get_personal_phone() == "PersonalPChanged"
assert emp_dict[x.username.get()].get_personal_email() == "PersonalEChanged"
assert emp_dict[x.username.get()].get_dob() == "BirthdayChanged"
assert emp_dict[x.username.get()].get_ssn() == "SSNChanged"
assert emp_dict[x.username.get()].get_pay_method() == "1"
assert emp_dict[x.username.get()].get_routing() == "RoutingChanged"
assert emp_dict[x.username.get()].get_account() == "AccountChanged"
assert emp_dict[x.username.get()].get_title() == "MasterChanged"
assert emp_dict[x.username.get()].get_dept() == "DeptChanged"
assert emp_dict[x.username.get()].get_start() == "1/2/2022"
if (emp_dict[x.username.get()].get_status() == True):
    assert e.emp_status.get() == "Active"
else:
    assert e.emp_status.get() == "Deactivated"

def test_edit_6():
    x.username.set("6")
    x.user_pass.set("test")
    x.validate_user()
    v = EmpDat_v5.View_Page(parent=x.container, controller=x)
    e = EmpDat_v5.Edit_Page(v, controller=x)
    e.f_name.set("FirstNameChanged")
    e.l_name.set("LastNameChanged")
    e.street.set("StreetChanged")
    e.city.set("CityChanged")

```



```

e.zip.set("ZIPChanged")
e.classification.set("1")
e.pay_type.set("1")
e.salary.set("100.00")
e.commissioned.set("10.00")
e.hourly.set("1.00")
e.routing_num.set("RoutingChanged")
e.account_num.set("AccountChanged")
e.o_phone.set("OfficePChanged")
e.p_phone.set("PersonalPChanged")
e.o_email.set("OfficeEChanged")
e.p_email.set("PersonalEChanged")
e.dob.set("BirthdayChanged")
e.ssn.set("SSNChanged")
e.emp_title.set("MasterChanged")
e.emp_dept.set("DeptChanged")
e.start_date.set("1/2/2022")
e.update_emp()
if (emp_dict[x.username.get()].is_admin() == True):
    assert v.permission.get() == '1'
else:
    assert v.permission.get() == '2'
assert emp_dict[x.username.get()].get_first_name() == "FirstNameChanged"
assert emp_dict[x.username.get()].get_last_name() == "LastNameChanged"
assert emp_dict[x.username.get()].get_address() == "StreetChanged"
assert emp_dict[x.username.get()].get_city() == "CityChanged"
assert emp_dict[x.username.get()].get_zip() == "ZIPChanged"
assert emp_dict[x.username.get()].get_class() == "1"
assert str(emp_dict[x.username.get()].get_hourly_rate()) == "1.00"
assert str(emp_dict[x.username.get()].get_commission_rate()) == "10.00"
assert str(emp_dict[x.username.get()].get_salary()) == "100.00"
assert emp_dict[x.username.get()].get_office_phone() == "OfficePChanged"
assert emp_dict[x.username.get()].get_office_email() == "OfficeEChanged"
assert emp_dict[x.username.get()].get_personal_phone() == "PersonalPChanged"
assert emp_dict[x.username.get()].get_personal_email() == "PersonalEChanged"
assert emp_dict[x.username.get()].get_dob() == "BirthdayChanged"
assert emp_dict[x.username.get()].get_ssn() == "SSNChanged"
assert emp_dict[x.username.get()].get_pay_method() == "1"
assert emp_dict[x.username.get()].get_routing() == "RoutingChanged"
assert emp_dict[x.username.get()].get_account() == "AccountChanged"
assert emp_dict[x.username.get()].get_title() == "MasterChanged"

```

```

assert emp_dict[x.username.get()].get_dept() == "DeptChanged"
assert emp_dict[x.username.get()].get_start() == "1/2/2022"
if (emp_dict[x.username.get()].get_status() == True):
    assert e.emp_status.get() == "Active"
else:
    assert e.emp_status.get() == "Deactivated"

#This 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
def test_edit_invalid_variable_1():
    x.username.set("5")
    x.user_pass.set("test")
    x.validate_user()
    v = EmpDat_v5.View_Page(parent=x.container, controller=x)
    e = EmpDat_v5.Edit_Page(v, controller=x)
    e.f_name.set("123")
    e.l_name.set("123")
    e.street.set("123")
    e.city.set("123")
    e.zip.set("123")
    e.classification.set("asd")
    e.pay_type.set("asd")
    e.salary.set("asd")
    e.commissioned.set("asd")
    e.hourly.set("asd")
    e.routing_num.set("123")
    e.account_num.set("123")
    e.o_phone.set("123")
    e.p_phone.set("123")
    e.o_email.set("123")
    e.p_email.set("123")
    e.dob.set("123")
    e.ssn.set("123")
    e.emp_title.set("123")
    e.emp_dept.set("123")
    e.start_date.set("asd")
    e.update_emp()
    assert e.page_error.get() == "One or more entries invalid!"

    assert emp_dict[x.username.get()].get_first_name() != "123"

```

```

assert emp_dict[x.username.get()].get_last_name() != "123"
assert emp_dict[x.username.get()].get_address() != "123"
assert emp_dict[x.username.get()].get_city() != "123"
assert emp_dict[x.username.get()].get_zip() != "123"
assert emp_dict[x.username.get()].get_class() != "asd"
assert str(emp_dict[x.username.get()].get_hourly_rate()) != "asd"
assert str(emp_dict[x.username.get()].get_commission_rate()) != "asd"
assert str(emp_dict[x.username.get()].get_salary()) != "asd"
assert emp_dict[x.username.get()].get_office_phone() != "123"
assert emp_dict[x.username.get()].get_office_email() != "123"
assert emp_dict[x.username.get()].get_personal_phone() != "123"
assert emp_dict[x.username.get()].get_personal_email() != "123"
assert emp_dict[x.username.get()].get_dob() != "123"
assert emp_dict[x.username.get()].get_ssn() != "123"
assert emp_dict[x.username.get()].get_pay_method() != "abc"
assert emp_dict[x.username.get()].get_routing() != "123"
assert emp_dict[x.username.get()].get_account() != "123"
assert emp_dict[x.username.get()].get_title() != "123"
assert emp_dict[x.username.get()].get_dept() != "123"
assert emp_dict[x.username.get()].get_start() != "asd"

```

```

e.f_name.set("FirstNameChanged")
e.l_name.set("LastNameChanged")
e.street.set("StreetChanged")
e.city.set("CityChanged")
e.zip.set("ZIPChanged")
e.classification.set("2")
e.pay_type.set("1")
e.salary.set("100.00")
e.commissioned.set("10.00")
e.hourly.set("1.00")
e.routing_num.set("RoutingChanged")
e.account_num.set("AccountChanged")
e.o_phone.set("OfficePChanged")
e.p_phone.set("PersonalPChanged")
e.o_email.set("OfficeEChanged")
e.p_email.set("PersonalEChanged")
e.dob.set("BirthdayChanged")
e.ssn.set("SSNChanged")
e.emp_title.set("MasterChanged")
e.emp_dept.set("DeptChanged")

```

```

e.start_date.set("1/2/2022")
e.update_emp()
if (emp_dict[x.username.get()].is_admin() == True):
    assert v.permission.get() == '1'
else:
    assert v.permission.get() == '2'
assert emp_dict[x.username.get()].get_first_name() == "FirstNameChanged"
assert emp_dict[x.username.get()].get_last_name() == "LastNameChanged"
assert emp_dict[x.username.get()].get_address() == "StreetChanged"
assert emp_dict[x.username.get()].get_city() == "CityChanged"
assert emp_dict[x.username.get()].get_zip() == "ZIPChanged"
assert emp_dict[x.username.get()].get_class() == "2"
assert str(emp_dict[x.username.get()].get_hourly_rate()) == "1.00"
assert str(emp_dict[x.username.get()].get_commission_rate()) == "10.00"
assert str(emp_dict[x.username.get()].get_salary()) == "100.00"
assert emp_dict[x.username.get()].get_office_phone() == "OfficePChanged"
assert emp_dict[x.username.get()].get_office_email() == "OfficeEChanged"
assert emp_dict[x.username.get()].get_personal_phone() == "PersonalPChanged"
assert emp_dict[x.username.get()].get_personal_email() == "PersonalEChanged"
assert emp_dict[x.username.get()].get_dob() == "BirthdayChanged"
assert emp_dict[x.username.get()].get_ssn() == "SSNChanged"
assert emp_dict[x.username.get()].get_pay_method() == "1"
assert emp_dict[x.username.get()].get_routing() == "RoutingChanged"
assert emp_dict[x.username.get()].get_account() == "AccountChanged"
assert emp_dict[x.username.get()].get_title() == "MasterChanged"
assert emp_dict[x.username.get()].get_dept() == "DeptChanged"
assert emp_dict[x.username.get()].get_start() == "1/2/2022"
if (emp_dict[x.username.get()].get_status() == True):
    assert e.emp_status.get() == "Active"
else:
    assert e.emp_status.get() == "Deactivated"

```

#CODE TO TEST ADD PAGE

```

def test_add_1():
    x.username.set("5")
    x.user_pass.set("test")
    x.validate_user()
    v = EmpDat_v5.View_Page(parent=x.container, controller=x)
    a = EmpDat_v5.Add_Page(v, controller=x)
    a.f_name.set("FirstNameChanged")

```

```

a.l_name.set("LastNameChanged")
a.street.set("StreetChanged")
a.city.set("CityChanged")
a.emp_state.set("StateChanged")
a.zip.set("ZIPChanged")
a.classification.set("2")
a.pay_type.set("1")
a.salary.set("100.00")
a.commissioned.set("10.00")
a.hourly.set("1.00")
a.routing_num.set("RoutingChanged")
a.account_num.set("AccountChanged")
a.o_phone.set("OfficePChanged")
a.p_phone.set("PersonalPChanged")
a.o_email.set("OfficeEChanged")
a.p_email.set("PersonalEChanged")
a.dob.set("BirthdayChanged")
a.ssn.set("SSNChanged")
a.emp_title.set("MasterChanged")
a.emp_dept.set("DeptChanged")
a.emp_password.set("test")
a.permission.set("1")
a.add_emp()
assert a.page_error.get() != "Missing required fields!"
assert a.page_error.get() != "One or more entries invalid!"
if (emp_dict[x.username.get()].is_admin() == True):
    assert v.permission.get() == '1'
else:
    assert v.permission.get() == '2'
assert emp_dict[str(len(emp_dict.keys()))].get_first_name() == "FirstNameChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_last_name() == "LastNameChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_address() == "StreetChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_city() == "CityChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_zip() == "ZIPChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_class() == "2"
assert str(emp_dict[str(len(emp_dict.keys()))].get_hourly_rate()) == "1.0"
assert str(emp_dict[str(len(emp_dict.keys()))].get_commission_rate()) == "10.0"
assert str(emp_dict[str(len(emp_dict.keys()))].get_salary()) == "100.0"
assert emp_dict[str(len(emp_dict.keys()))].get_office_phone() == "OfficePChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_office_email() == "OfficeEChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_personal_phone() == "PersonalPChanged"

```

```

assert emp_dict[str(len(emp_dict.keys()))].get_personal_email() == "PersonalEChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_dob() == "BirthdayChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_ssn() == "SSNChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_pay_method() == "1"
assert emp_dict[str(len(emp_dict.keys()))].get_routing() == "RoutingChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_account() == "AccountChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_title() == "MasterChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_dept() == "DeptChanged"
if (emp_dict[str(len(emp_dict.keys()))].get_status() == True):
    assert a.emp_status.get() == "Active"
else:
    assert a.emp_status.get() == "Deactivated"

#Test for all correct fields except one, then correct field
def test_add_incorrect_1():
    x.username.set("5")
    x.user_pass.set("test")
    x.validate_user()
    v = EmpDat_v5.View_Page(parent=x.container, controller=x)
    a = EmpDat_v5.Add_Page(v, controller=x)
    a.f_name.set("FirstNameChanged")
    a.l_name.set("LastNameChanged")
    a.street.set("StreetChanged")
    a.city.set("CityChanged")
    a.emp_state.set("StateChanged")
    a.zip.set("ZIPChanged")
    a.classification.set("2")
    a.pay_type.set("1")
    a.salary.set("100.00")
    a.commissioned.set("10.00")
    a.hourly.set("asd")
    a.routing_num.set("RoutingChanged")
    a.account_num.set("AccountChanged")
    a.o_phone.set("OfficePChanged")
    a.p_phone.set("PersonalPChanged")
    a.o_email.set("OfficeEChanged")
    a.p_email.set("PersonalEChanged")
    a.dob.set("BirthdayChanged")
    a.ssn.set("SSNChanged")
    a.emp_title.set("MasterChanged")
    a.emp_dept.set("DeptChanged")

```

```

a.emp_password.set("test")
a.permission.set("1")
a.add_emp()
assert a.page_error.get() != "Missing required fields!"
assert a.page_error.get() == "One or more entries invalid!"

a.hourly.set("1.00")
a.add_emp()

if (emp_dict[x.username.get()].is_admin() == True):
    assert v.permission.get() == '1'
else:
    assert v.permission.get() == '2'
assert emp_dict[str(len(emp_dict.keys()))].get_first_name() == "FirstNameChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_last_name() == "LastNameChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_address() == "StreetChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_city() == "CityChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_zip() == "ZIPChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_class() == "2"
assert str(emp_dict[str(len(emp_dict.keys()))].get_hourly_rate()) == "1.0"
assert str(emp_dict[str(len(emp_dict.keys()))].get_commission_rate()) == "10.0"
assert str(emp_dict[str(len(emp_dict.keys()))].get_salary()) == "100.0"
assert emp_dict[str(len(emp_dict.keys()))].get_office_phone() == "OfficePChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_office_email() == "OfficeEChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_personal_phone() == "PersonalPChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_personal_email() == "PersonalEChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_dob() == "BirthdayChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_ssn() == "SSNChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_pay_method() == "1"
assert emp_dict[str(len(emp_dict.keys()))].get_routing() == "RoutingChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_account() == "AccountChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_title() == "MasterChanged"
assert emp_dict[str(len(emp_dict.keys()))].get_dept() == "DeptChanged"
if (emp_dict[str(len(emp_dict.keys()))].get_status() == True):
    assert a.emp_status.get() == "Active"
else:
    assert a.emp_status.get() == "Deactivated"

```

#CODE FOR TESTING SEARCH PAGE

#Test_1 searches for a user that should be in the database

```

def test_search_1():
    x.username.set("5")
    x.user_pass.set("test")
    x.validate_user()
    s = EmpDat_v5.Search_Page(parent=x.container, controller=x)
    s.search_parameter.set("id")
    s.search.set("5")
    s.search_employees()
    assert s.search.get() == ""
    assert s.msg.get() != "Must choose a search parameter"
    assert s.msg.get() != "ID not found!"
    assert s.msg.get() != "Last Name not found!"

def test_search_2():
    x.username.set("5")
    x.user_pass.set("test")
    x.validate_user()
    s = EmpDat_v5.Search_Page(parent=x.container, controller=x)
    s.search.set("1")
    s.search_employees()
    assert s.msg.get() == "Must choose a search parameter"

def test_search_3():
    x.username.set("5")
    x.user_pass.set("test")
    x.validate_user()
    s = EmpDat_v5.Search_Page(parent=x.container, controller=x)
    s.search_parameter.set("id")
    s.search.set("100")
    s.search_employees()
    assert s.msg.get() == "ID not found!"
    assert s.search.get() != ""

def test_search_4():
    x.username.set("5")
    x.user_pass.set("test")
    x.validate_user()
    s = EmpDat_v5.Search_Page(parent=x.container, controller=x)
    s.search_parameter.set("last")
    s.search.set("johnasdk")
    s.search_employees()

```



```

assert s.msg.get() == "Last Name not found!"

def test_search_4():
    x.username.set("5")
    x.user_pass.set("test")
    x.validate_user()
    s = EmpDat_v5.Search_Page(parent=x.container, controller=x)
    s.search_parameter.set("last")
    s.search.set("LastNameChanged")
    s.search_employees()
    assert s.search.get() == ""
    assert s.msg.get() != "Must choose a search parameter"
    assert s.msg.get() != "ID not found!"
    assert s.msg.get() != "Last Name not found!"

#CODE FOR TESTING PAY PAGE
def test_pay_1():
    x.username.set("2")
    x.user_pass.set("test")
    x.validate_user()
    s = EmpDat_v5.Pay_Page(parent=x.container, controller=x)
    if (str(emp_dict["2"].get_classification()) == "Salaried Employee"):
        s.create_add_frame(2)
        s.emp_to_add.set("2")
        s.amount.set("100")
        s.add_item(2)
        assert s.add_error_msg.get() != "Invalid Employee ID!"
        assert s.add_error_msg.get() == "Specified Employee has incorrect classification!"
    elif (str(emp_dict["2"].get_classification()) == "Hourly Employee"):
        s.create_add_frame(2)
        s.emp_to_add.set("2")
        s.amount.set("100")
        s.add_item(2)
        assert s.add_error_msg.get() != "Invalid Employee ID!"
        assert s.add_error_msg.get() != "Specified Employee has incorrect classification!"
        assert s.add_error_msg.get() != "Amount must enter either a number or a decimal!"
    else:
        s.create_add_frame(1)
        s.emp_to_add.set("2")
        s.amount.set("100")

```

```

        s.add_item(1)
        assert s.add_error_msg.get() != "Invalid Employee ID!"
        assert s.add_error_msg.get() != "Specified Employee has incorrect classification!"
        assert s.add_error_msg.get() != "Amount must enter either a number or a decimal!"

def test_pay_2():
    x.username.set("2")
    x.user_pass.set("test")
    x.validate_user()
    s = EmpDat_v5.Pay_Page(parent=x.container, controller=x)
    if (str(emp_dict["3"].get_classification()) == "Salaried Employee"):
        s.create_add_frame(2)
        s.emp_to_add.set("3")
        s.amount.set("100")
        s.add_item(2)
        assert s.add_error_msg.get() != "Invalid Employee ID!"
        assert s.add_error_msg.get() == "Specified Employee has incorrect classification!"
    elif (str(emp_dict["3"].get_classification()) == "Hourly Employee"):
        s.create_add_frame(2)
        s.emp_to_add.set("3")
        s.amount.set("100")
        s.add_item(2)
        assert s.add_error_msg.get() != "Invalid Employee ID!"
        assert s.add_error_msg.get() != "Specified Employee has incorrect classification!"
        assert s.add_error_msg.get() != "Amount must enter either a number or a decimal!"
    else:
        s.create_add_frame(1)
        s.emp_to_add.set("3")
        s.amount.set("100")
        s.add_item(1)
        assert s.add_error_msg.get() != "Invalid Employee ID!"
        assert s.add_error_msg.get() != "Specified Employee has incorrect classification!"
        assert s.add_error_msg.get() != "Amount must enter either a number or a decimal!"

def test_pay_3():
    x.username.set("2")
    x.user_pass.set("test")
    x.validate_user()
    s = EmpDat_v5.Pay_Page(parent=x.container, controller=x)
    if (str(emp_dict["7"].get_classification()) == "Salaried Employee"):
        s.create_add_frame(2)

```

```

s.emp_to_add.set("7")
s.amount.set("100")
s.add_item(2)
assert s.add_error_msg.get() != "Invalid Employee ID!"
assert s.add_error_msg.get() == "Specified Employee has incorrect classification!"
elif (str(emp_dict["7"].get_classification()) == "Hourly Employee"):
    s.create_add_frame(2)
    s.emp_to_add.set("7")
    s.amount.set("100")
    s.add_item(2)
    assert s.add_error_msg.get() != "Invalid Employee ID!"
    assert s.add_error_msg.get() != "Specified Employee has incorrect classification!"
    assert s.add_error_msg.get() != "Amount must enter either a number or a decimal!"
else:
    s.create_add_frame(1)
    s.emp_to_add.set("7")
    s.amount.set("100")
    s.add_item(1)
    assert s.add_error_msg.get() != "Invalid Employee ID!"
    assert s.add_error_msg.get() != "Specified Employee has incorrect classification!"
    assert s.add_error_msg.get() != "Amount must enter either a number or a decimal!"

```

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 database	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 an '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".

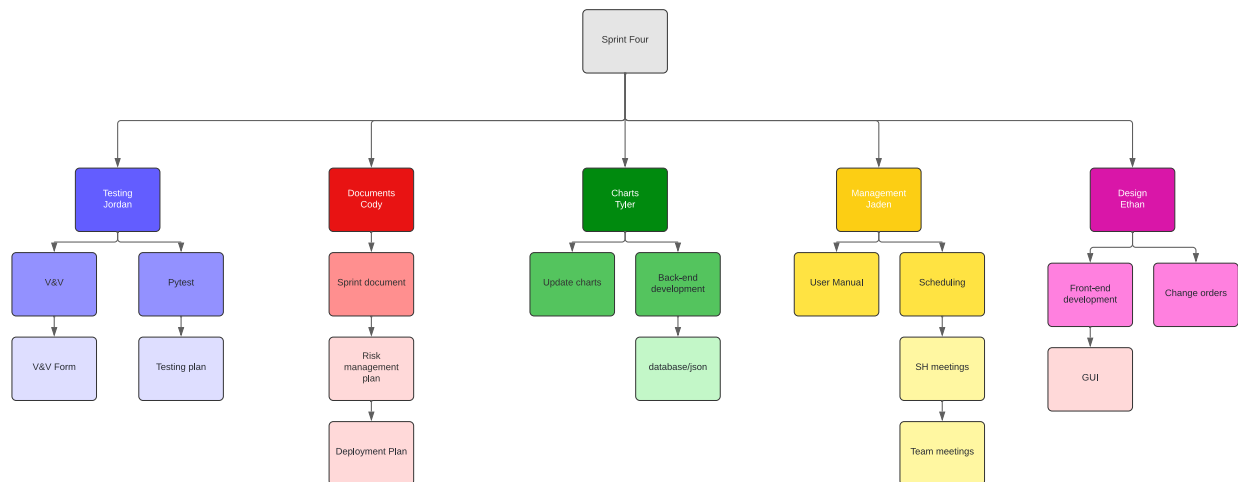
Bug ID	Bug Found Date	Description	Priority	Start Date	Note	End date
1	17/03/2022	When a non-admin employee tries to search for another employee, the program crashes. The error says: "AttributeError: 'Ed@it_Page' object has no attribute 'status_entry'".	high	18/03/2022	FIXED: Bug was caused by tkinter 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

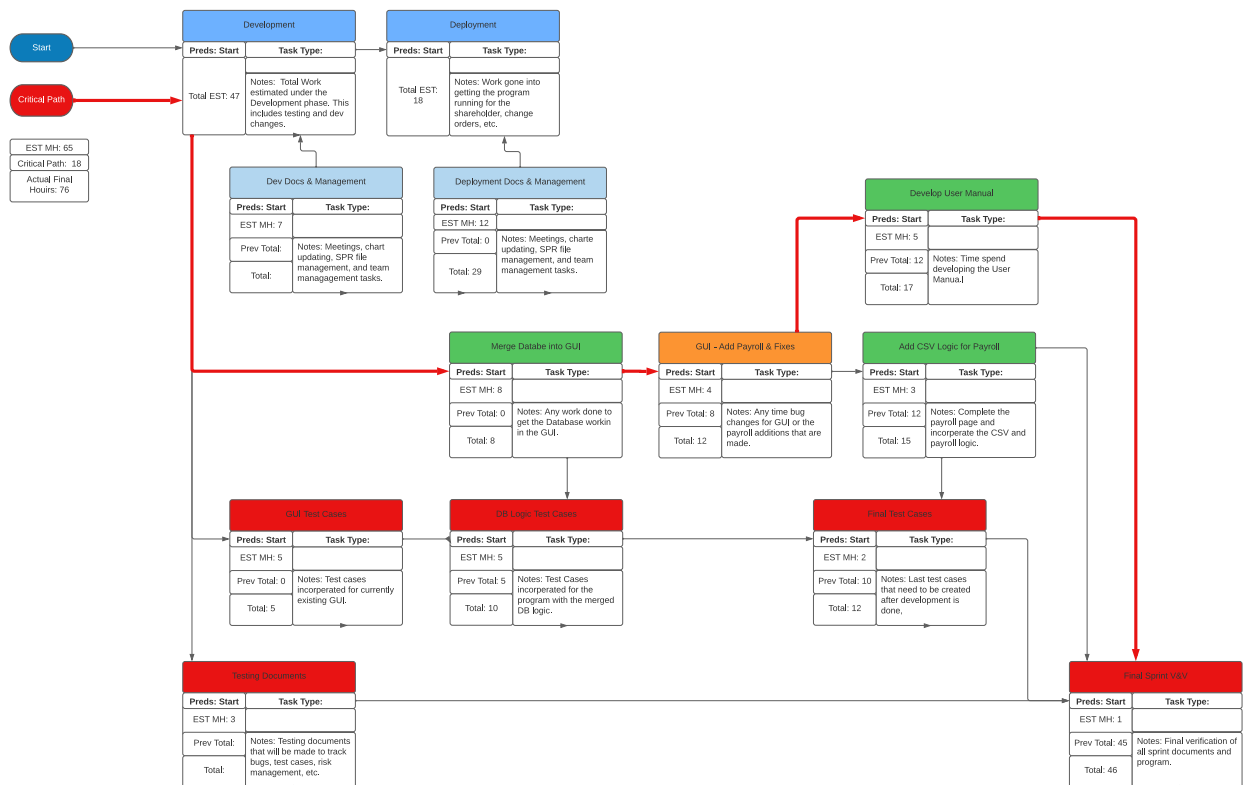
Chart

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



SPR-4 Pert Chart

Chart

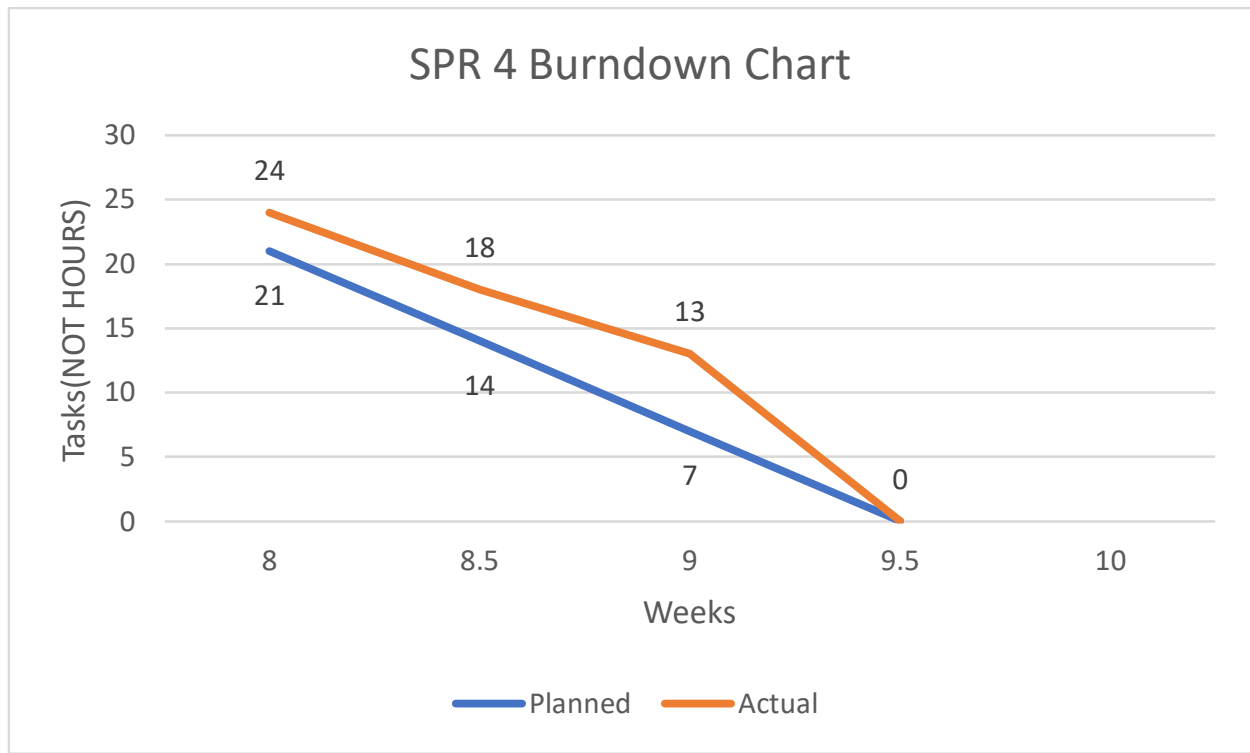


Chart



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