Sprint Report 01

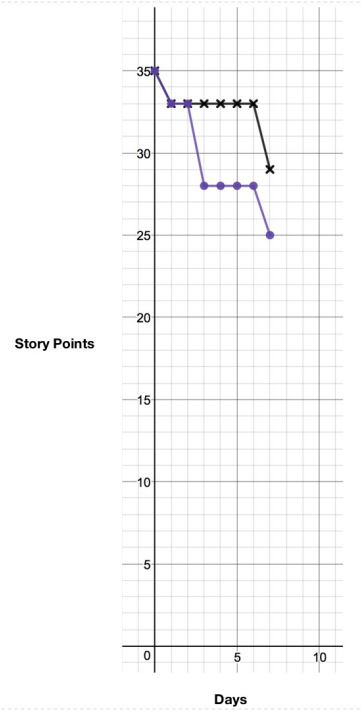
Task Overview

User Stories	Tasks	Dependencies	Story Points
1	1		1
1	2	T1	5
1	3		5
1	4	T2 & T3	1
2	5		2
2	6	T1	2
2	7	T3 & T5	1
3	8		4
3	9	T8	1
4	10		5
4	11		5
4	12	T10 & T11	5
4	13	T11 & T12	5
4	14	T13 & T14	1
5	15		2
5	16		8
6	17		8
6	18	T16 & T17	4
7	19	T17	4

Task Burndown

Days	0	1	2	3	4	5	6	7
Provisional	35	33	33	28	28	28	28	25
Actual	35	33	33	33	33	33	33	29

- Dark Purple line represents the provisional burndown chart
- Black line represents the actual burndown chart



Task Details

Task 1

Story Points: 1

Dependencies: None

Database setup and schema creation: Setup a MySQL database using Python for storing information regarding a the creation of a new agency; Create a schema to depict the relationship between an Agency and Employee using the website draw.io.

Task 2

Story Points: 5

Dependencies: Task 1

Create a new agency: Implement the functionality using Python and MySQL connector for Python to create a new agency's profile. The profile will be stored in a MySQL database.

Task 3

Story Points: 5

Dependencies: None

Interface design: Create a using user interface to accomplish the task of creating a new agency's profile using the PYQT5 Python library.

Task 4

Story Points: 1

Dependencies: Task 2 & Task 3

Connect UI with functionality: Connect the user interface developed with the PYQT5 Python library with the functionality present to create a new agency's profile.

Task 5

Story Points: 2

Dependencies: None

Implement an API function to automatically send emails to agencies. The email will contain the agency employee's username and password. It will be sent when the agency employee is added to the system. The same email will be sent when the agency employee tries to recover their password if they forgot it.
This task will use python's built-in email library. Created a new email account which will act as the sender email. The task needs to retrieve the agency employee's username and password from the agency employee database.

Task 6

Story Points: 2

Dependencies: Task 1

Connect email API to agency employee database. The task needs to retrieve the agency employee's username and password from the agency employee database.

Task 7

Story Points: 1

Dependencies: Task 3 & Task 5

Connect UI for adding an agency employee with email API for sending login information.

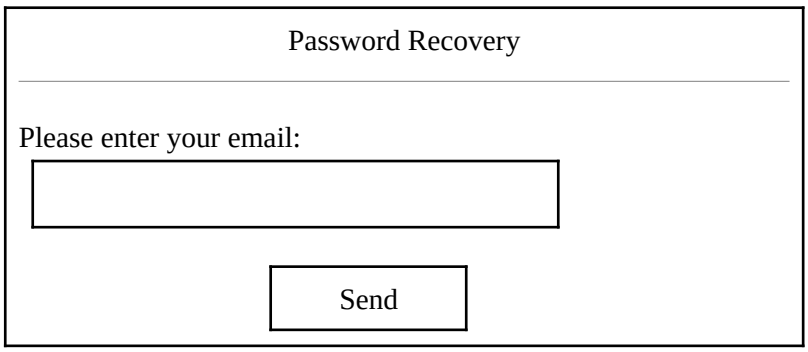
Task 8

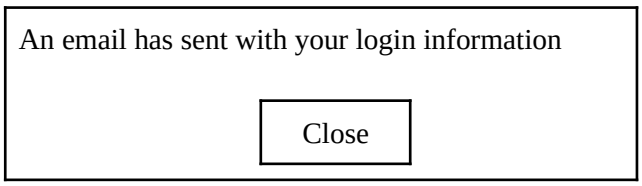
Story Points: 4

Dependencies: None

Design an interface for recovering an agency employee's password. Will use Qt5 to design the UI.

UI design sketch:





Task 9

Story Points: 1

Dependencies: Task 8

Connect UI for recovering a user's password with email API for sending login information.

Task 10

Story Points: 5

Dependencies: None

Implement an API function to parse .xlsx files.

- Utilize openpyxl
- In particular, parse iCare template files

Task 11

Story Points: 5

Dependencies: None

Implement an API function to insert parsed iCare template data into database.

- Using mysql database
- Need to be able to dynamically create tables
- Will most likely require user to input data type fields

Task 12

Story Points: 5

Dependencies: Task 10 & 11

Connect the API that parses the .xlsx files and the API that inserts parsed iCare template data into the database.

- May affect changes in API

Task 13

Story Points: 5

Dependencies: Task 11 & 12

Design an interface for uploading different iCare templates.

- May require creating a table of values
- Using PyQT5

Task 14

Story Points: 1

Dependencies: Task 12 & 13

Connect the UI for uploading different iCare templates with the API to parse and insert parsed iCare template data.

Task 15

Story Points: 3

Dependencies: None

implement an API function to detect invalid input for a field in an icare template.

- Will involve creating a table of values in our mysql database
- Will involve regex matching
- Will likely modify earlier API created for inputting new icare templates

Task 16

Story Points: 8

Dependencies: None

Implement an API function to run sql query in our database.

- If the user is knowledgeable with SQL, the user is given the ability to submit their own query
- Simple syntax checking will be performed

Task 17

Story Points: 8

Dependencies: None

design an interface for SQL querying and report generation

- Using PYQT5
- Provide textbox for entering query
- May provide drop-down menu/checkbox for join-table/grouping/sorting functions
- Provide some option for conversion to specific file format (button, checkbox, etc)

Task 18

Story Points: 4

Dependencies: T16 & T17

Connect the UI for SQL querying to the API to check bad SQL query input and API to run SQL query

Table 19

Story Points: 4

Dependencies: T17

Implement an API function to turn button presses on SQL query interface into an SQL query

- Using PyPika

Task 20

Story Points: N/A

Dependencies: None

Implement an API that takes SQL query output and outputs the results into a csv file.

Task 21

Story Points: None

Dependencies: Task 18 & Task 20

Append to UI for running SQL queries to output in csv.

Task 22

Story Points: N/A

Dependencies: None

Implement an API to ask user where to save file if necessary.

Task 23

Story Points: N/A

Dependencies: None

Implement an API that takes SQL query output and outputs the results in a graph, into a pdf file.

Task 24

Story Points: N/A

Dependencies: Task 19 & Task 21

Connect UI for running SQL queries to API that outputs results in a pdf file.

Task 25

Story Points: N/A

Dependencies: None

Implement an API that pareses an .xlsx file and creates a table with all the columns specified in the .xlsx file

Task 26

Story Points: N/A

Dependencies: None

Design an interface for uploading .xlsx file and giving the table a name. Will use Qt5 to build the interface.

Task 27

Story Points: N/A

Dependencies: Task 25 & Task 26

Connect UI for uploading .xlsx file and the API that parses an .xlsx file

Task 28

Story Points: N/A

Dependencies: None

Implement an API that parses an .xlsx file and updates a specific table with all the columns specified in the .xlsx file

Task 29

Story Points: N/A

Dependencies:None

Design an interface for uploading .xlsx file and selecting a table to update. Will use Qt5 to build the interface.

Task 30

Story Points: N/A

Dependencies:
Connect the UI for uploading .xlsx file and selecting a table to update with the API that parses an .xlsx file and updates a specific table with all the columns specified in the .xlsx file.

Task 31

Story Points: N/A
Dependencies: None
Implement an API that checks uploaded .xlsx files have valid data and notifies where the error is.

Task 32

Story Points: N/A
Dependencies: None
Design an interface that notifies the user when the data is not valid. Will use Qt5 to build the UI.

Task 33

Story Points: N/A
Dependencies: Task 31 & 32
Connect the UI that notifies the user when the data is not valid and the API that checks for errors.

Note: Tasks 20-33 do not have story points yet. As a group we are focusing on the tasks currently assigned to each of us for this sprint. After completing these tasks and becoming acclimated with the new technologies we chose to use for this project, we will have a better understanding of how quickly we can complete future tasks.