Lumberjack Balancing

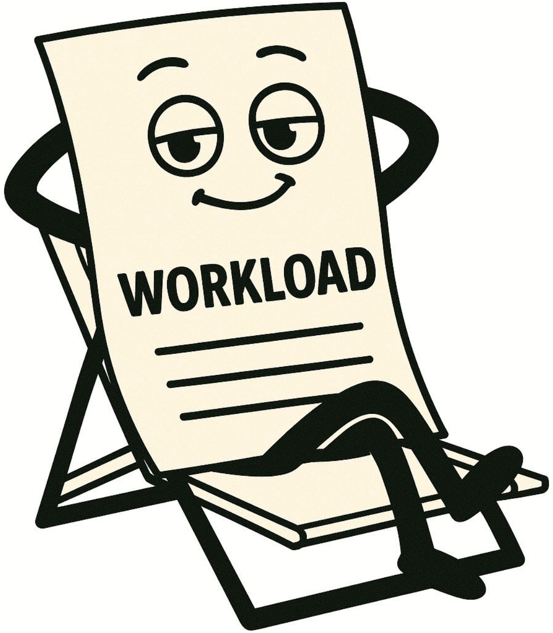
Project Sponsor: Dr. Scot Raab
Project Mentor: Paul Deasy
Team Members: Riley Burke, Cristian Marrufo,
Sergio Rabadan, Braden Wendt

What is Faculty Workload?

- Calculation of how faculty time is distributed across duties
- Teaching, Research, and Service
- Governed by institutional policy
- Typically expressed as a percentage

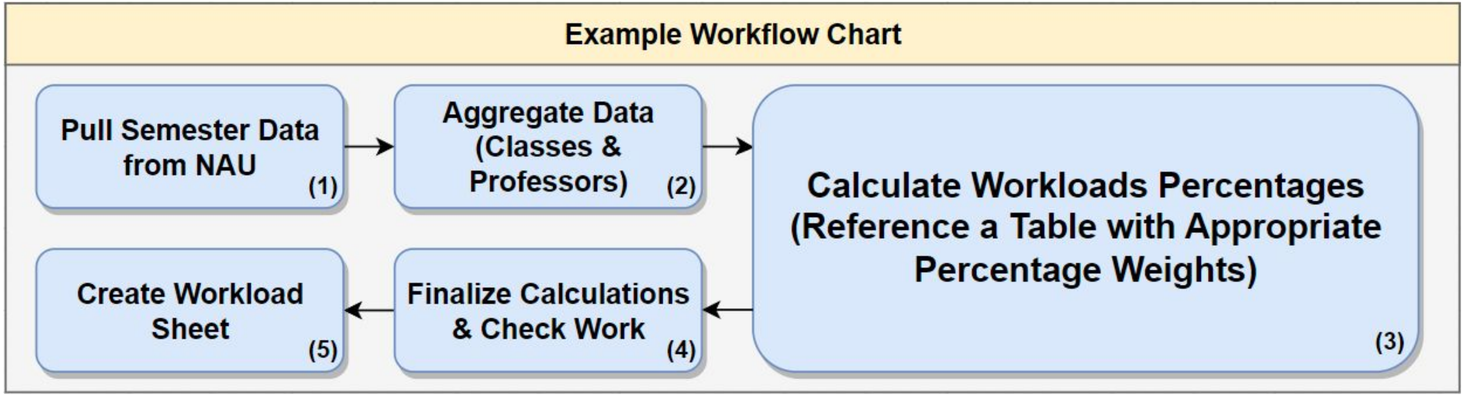
Why is it Important?

- Fairness for all faculty
- Compliance with university policies
- Reduce overwork and underutilization
- Directly affects faculty satisfaction and university effectiveness



Problem Statement

- Crucial administrative task for a large university like NAU
- Accurate workload management ensures equitable teaching responsibilities
- Current process is completely manual, time consuming, and prone to error.



Example Workflow Chart

Pull Semester Data from NAU (1)

Aggregate Data (Classes & Professors) (2)

Calculate Workloads Percentages (Reference a Table with Appropriate Percentage Weights) (3)

Finalize Calculations & Check Work (4)

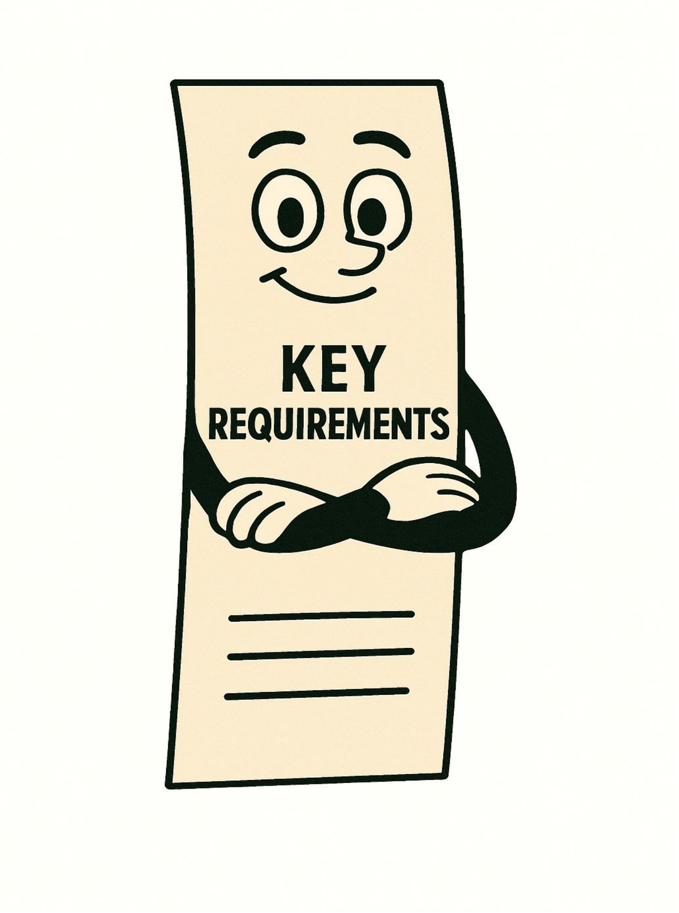
Create Workload Sheet (5)

Solution Overview

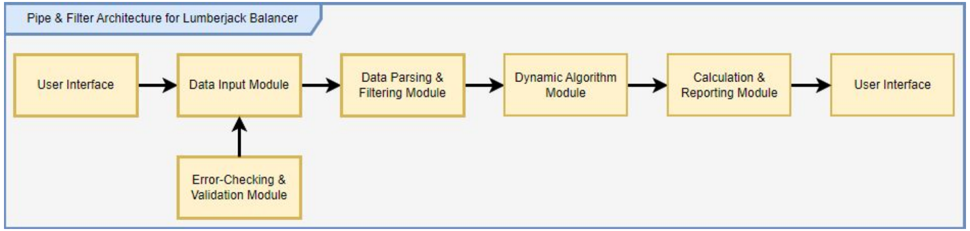
- Python based desktop application that will fully automate workload calculations
- Data validation, customizable workload policies, user-friendly interface
- Eliminate the cumbersome manual process and replace it with an automated Python program

Key Requirements

- Automated Workload Calculation.
- Data Validation
- Centralized Data Management
- Customizable Workload Policies
- Comprehensive Reporting



Architecture Review



Pipe & Filter Architecture for Lumberjack Balancer

User Interface

Data Input Module

Data Parsing & Filtering Module

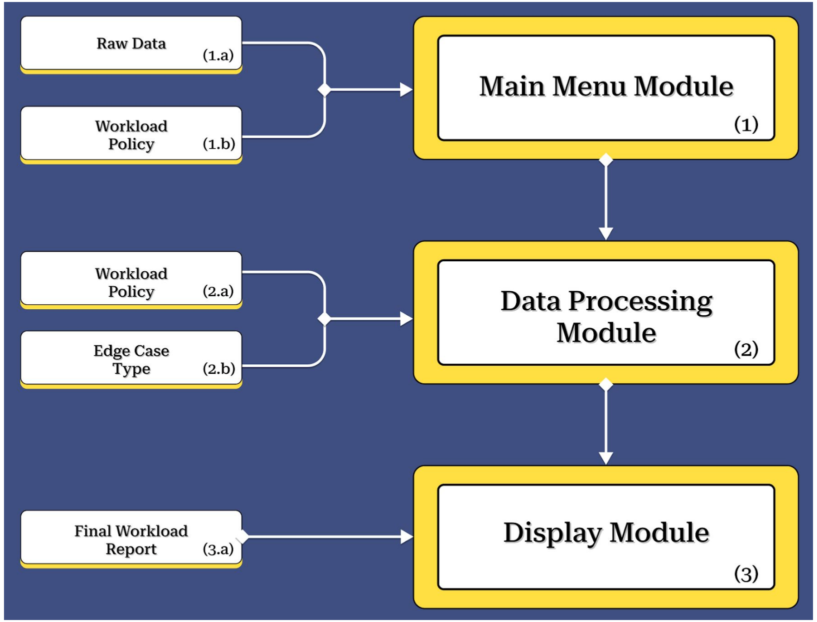
Dynamic Algorithm Module

Calculation & Reporting Module

User Interface

Error-Checking & Validation Module

- Condensed into 3 large modules
- Currently working on the Data Processing Module
- Always refining the Main Menu Module



Raw Data (1.a)

Workload Policy (1.b)

Main Menu Module (1)

Workload Policy (2.a)

Edge Case Type (2.b)

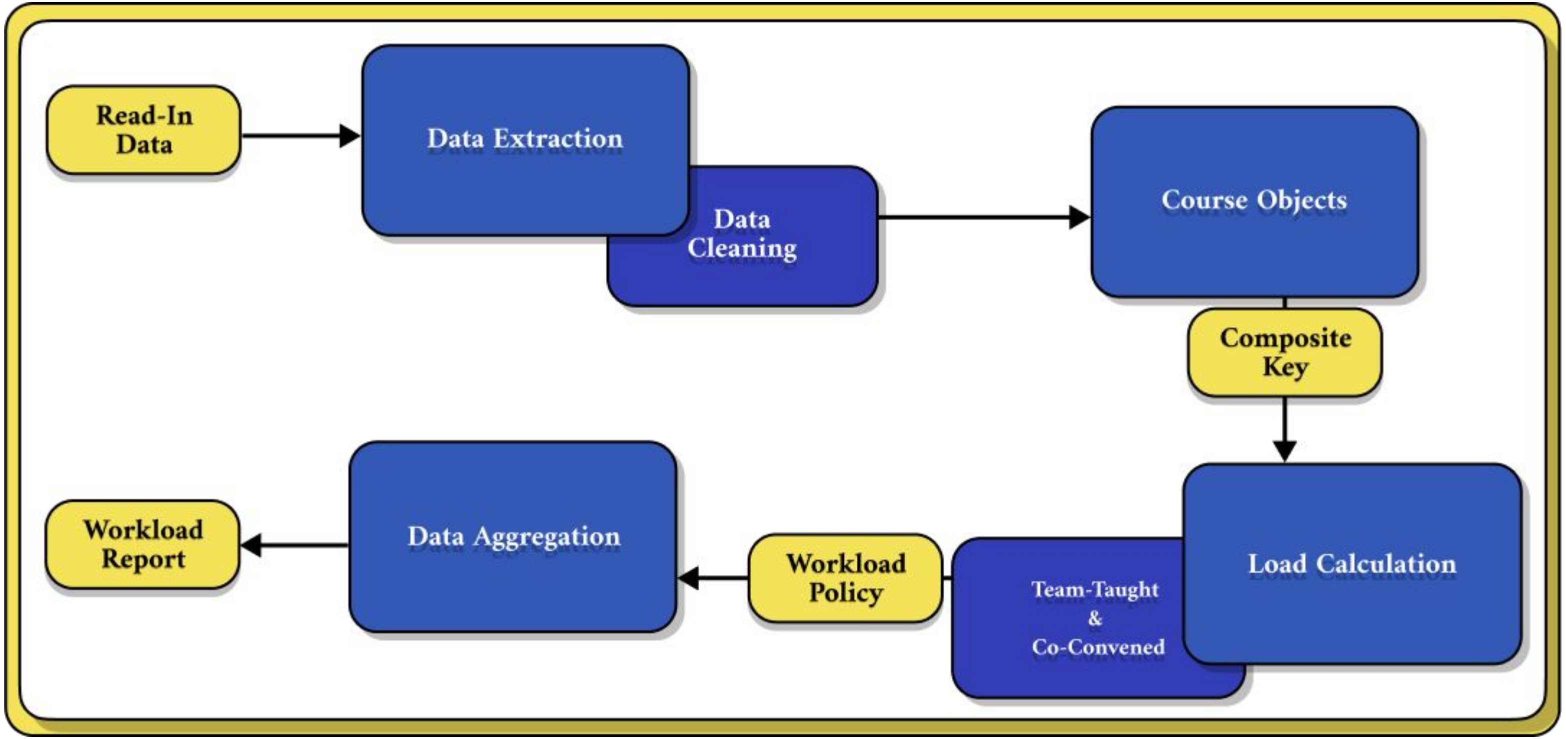
Data Processing Module (2)

Final Workload Report (3.a)

Display Module (3)

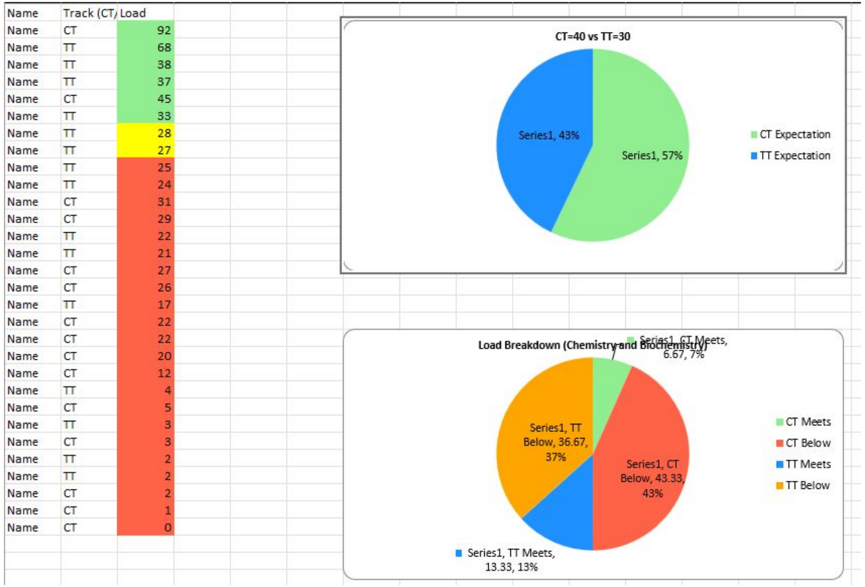
- Gives a clear image of module responsibilities

Processing Layer Overview



Report Output

- Display Professors, TT or CT, and Percentage
- Provide easy visualizations for various scenarios
  - Compare the tracks with ABOR expected
  - Compare with table data

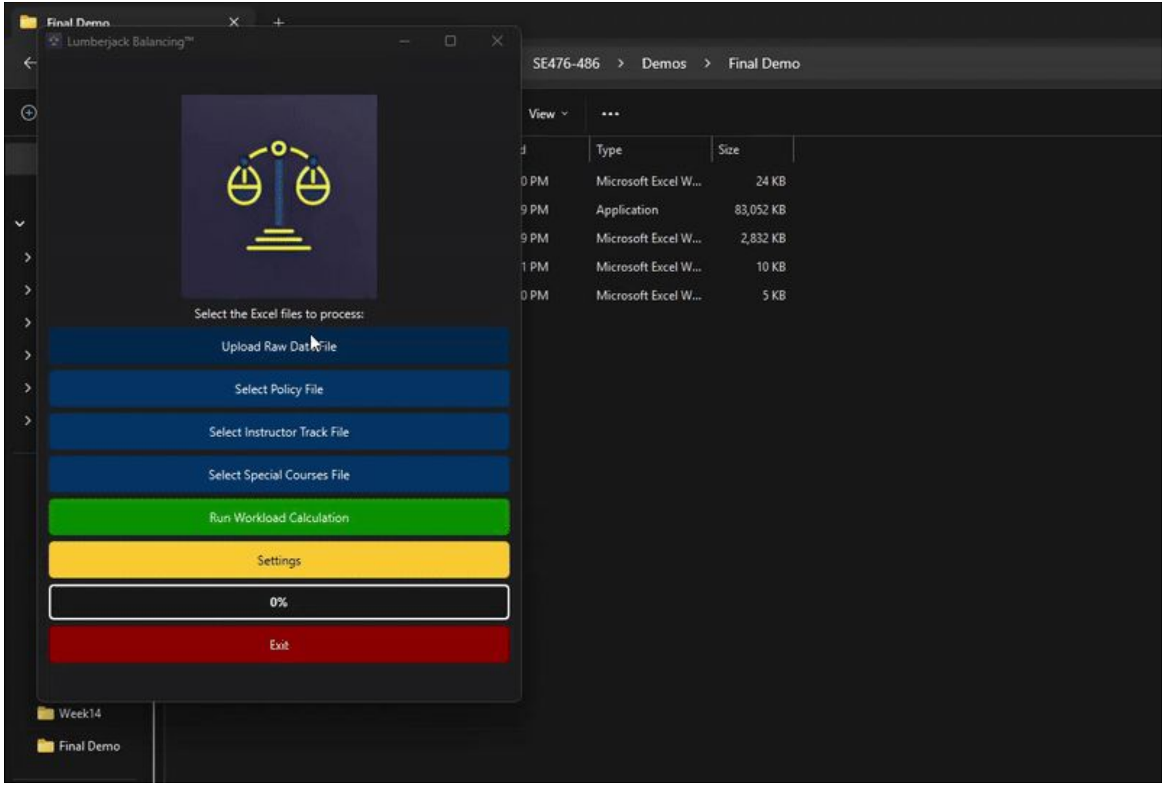


Name	Track (CT/	Load
Name	CT	92
Name	TT	68
Name	TT	38
Name	TT	37
Name	CT	45
Name	TT	33
Name	TT	28
Name	TT	27
Name	TT	25
Name	TT	24
Name	CT	31
Name	CT	29
Name	TT	22
Name	TT	21
Name	CT	27
Name	CT	26
Name	TT	17
Name	CT	22
Name	CT	22
Name	CT	20
Name	CT	12
Name	TT	4
Name	CT	5
Name	TT	3
Name	CT	3
Name	TT	2
Name	TT	2
Name	CT	2
Name	CT	1
Name	CT	0

CT=40 vs TT=30
Series1, 43%
Series1, 57%
CT Expectation
TT Expectation
Load Breakdown (Chemistry and Biochemistry)
Series1, CT Meets, 6.67, 7%
Series1, TT Below, 36.67, 37%
Series1, CT Below, 43.33, 43%
Series1, TT Meets, 13.33, 13%
CT Meets
CT Below
TT Meets
TT Below

Disclaimer: Fake Data points

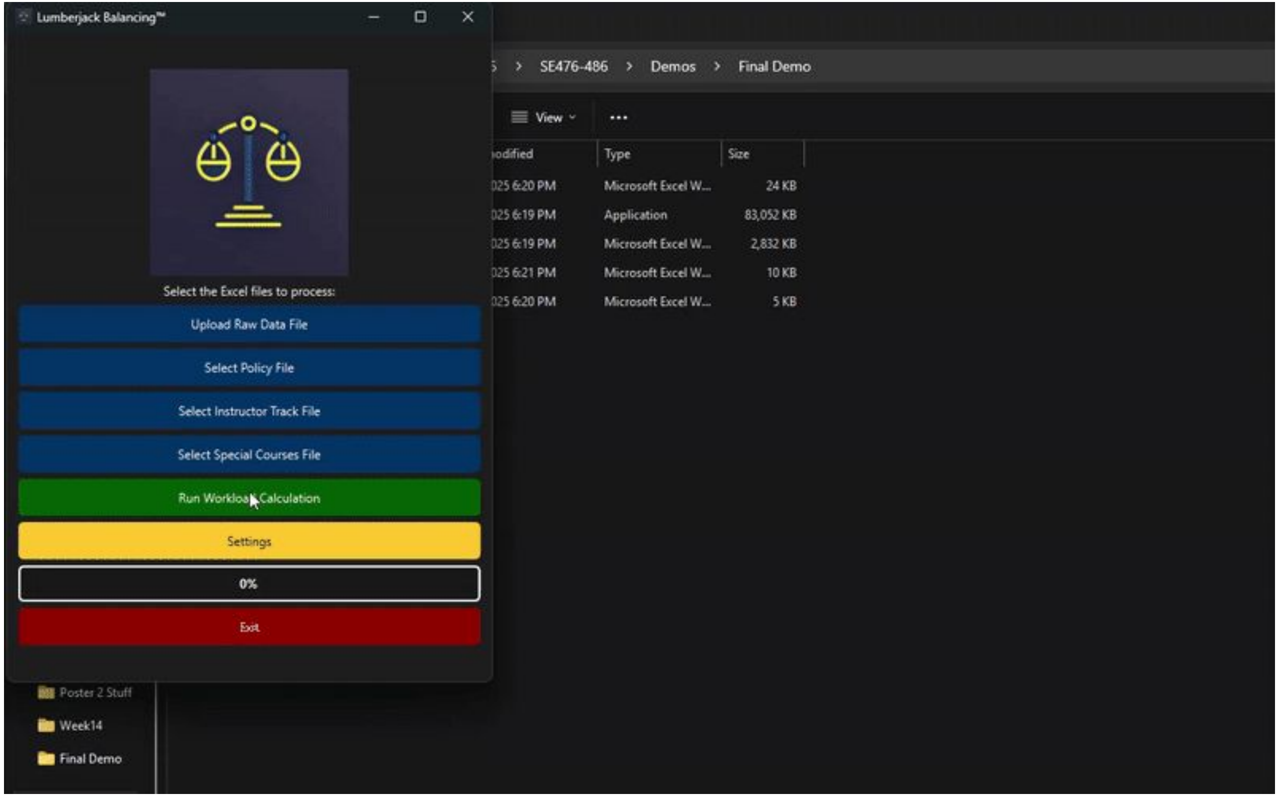
Demo - Uploading Documents



Select the Excel files to process:

d	Type	Size
0 PM	Microsoft Excel W...	24 KB
9 PM	Application	83,052 KB
9 PM	Microsoft Excel W...	2,832 KB
1 PM	Microsoft Excel W...	10 KB
0 PM	Microsoft Excel W...	5 KB

Demo - Running the Application

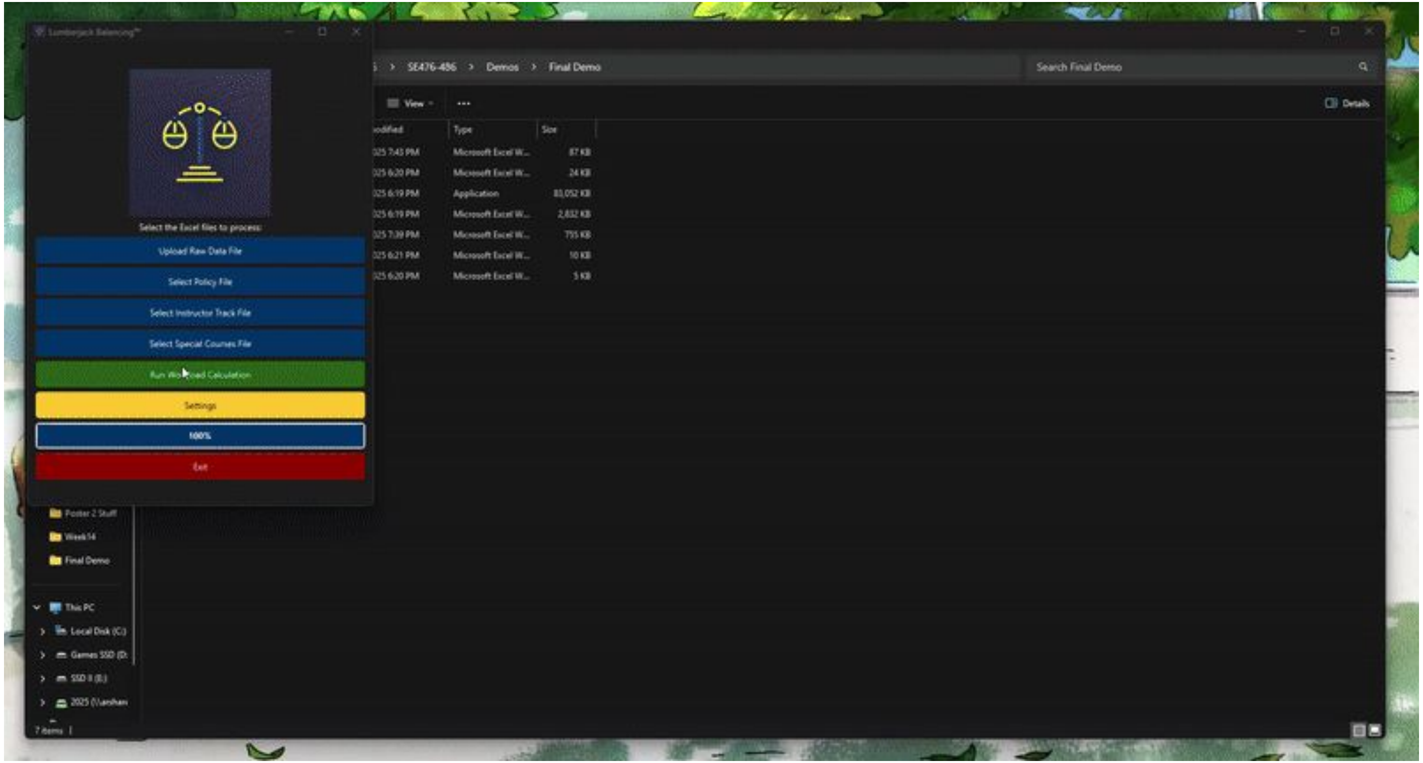


Select the Excel files to process:

odified	Type	Size
025 6:20 PM	Microsoft Excel W...	24 KB
025 6:19 PM	Application	83,052 KB
025 6:19 PM	Microsoft Excel W...	2,832 KB
025 6:21 PM	Microsoft Excel W...	10 KB
025 6:20 PM	Microsoft Excel W...	5 KB

Demo - Reports

Disclaimer: Names hidden for privacy purposes



odified	Type	Size
025 7:43 PM	Microsoft Excel W...	87 KB
025 6:20 PM	Microsoft Excel W...	24 KB
025 6:19 PM	Application	83,052 KB
025 6:19 PM	Microsoft Excel W...	2,832 KB
025 7:39 PM	Microsoft Excel W...	755 KB
025 6:21 PM	Microsoft Excel W...	10 KB
025 6:20 PM	Microsoft Excel W...	5 KB

Testing - Overall

- Get example data from Client
- Run the data in our application
- Compare results

Testing - Unit

- Verify Correctness:

  Ensure that each method or procedure produces the expected output given specific inputs.

- Facilitate Maintenance:

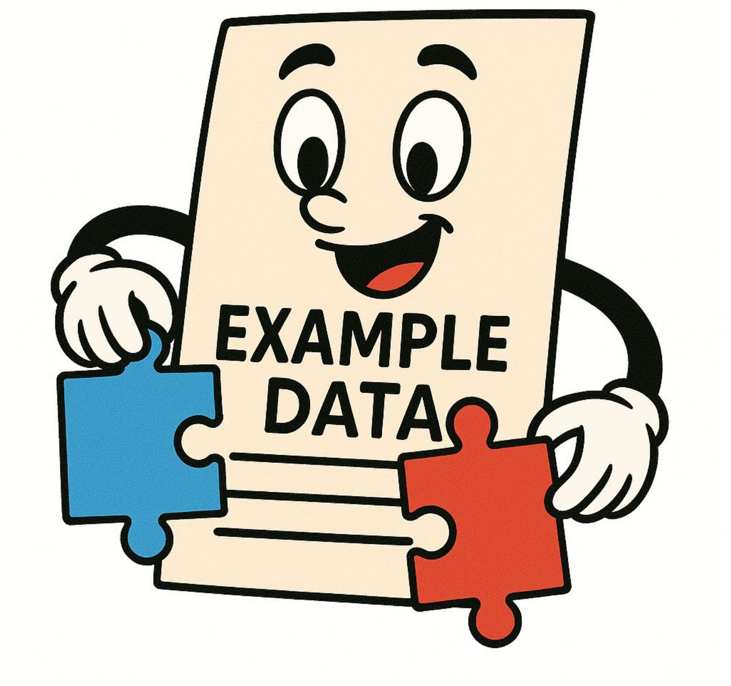
  Detect changes or regressions in functionality early as code evolves.

- Document Behaviour:

  Provide a living specification of how the code is intended to work.

Testing - Integration

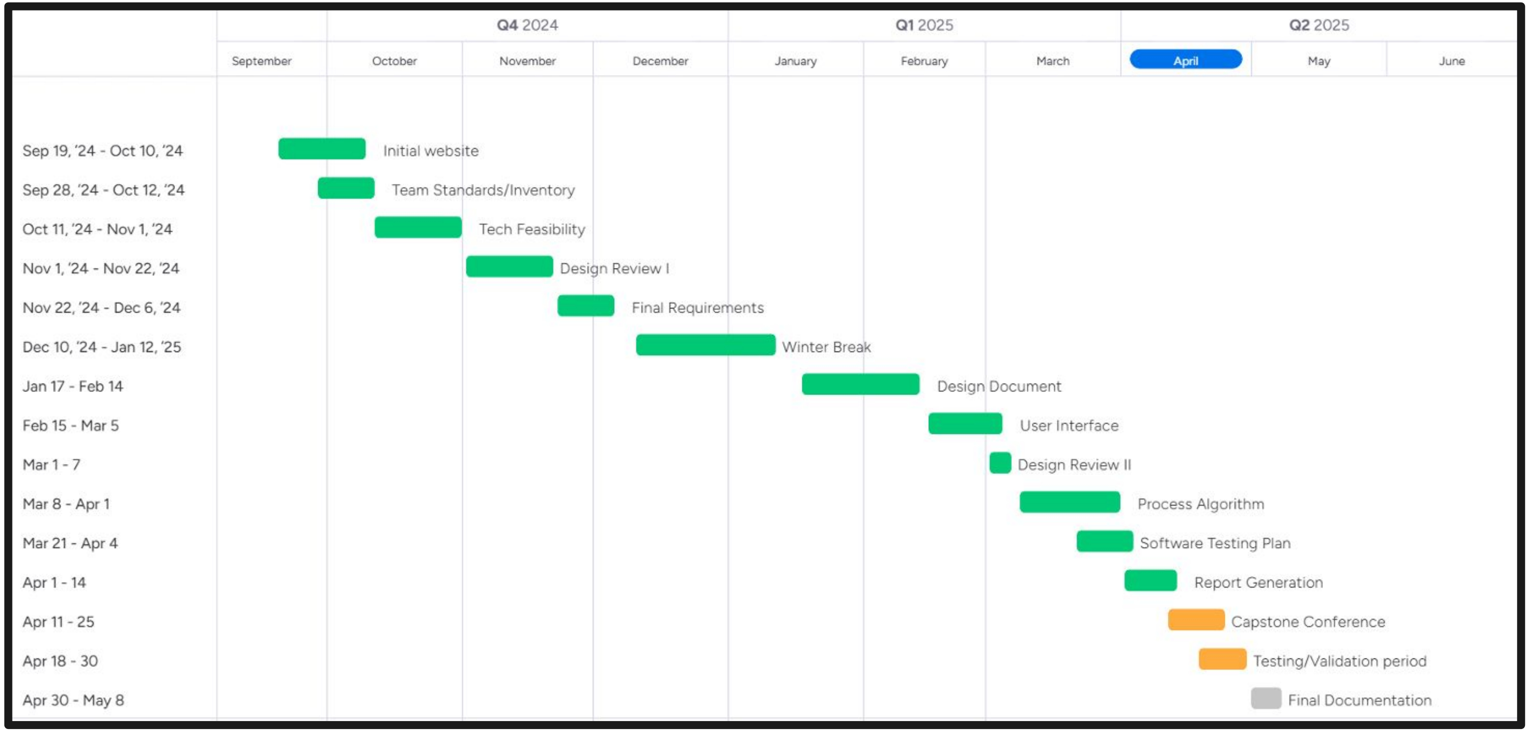
- Confirm that data read from external sources such are handed off to the appropriate processing modules.
- Validate that interfaces between modules correctly exchange parameters and return values.
- Detect issues in the wiring that may appear during unit testing of isolated components.



Testing - User Acceptance

- Two meetings with our Client
  - First Meeting: Initial thoughts and ideas
  - Second Meeting: Our updates based on first meeting

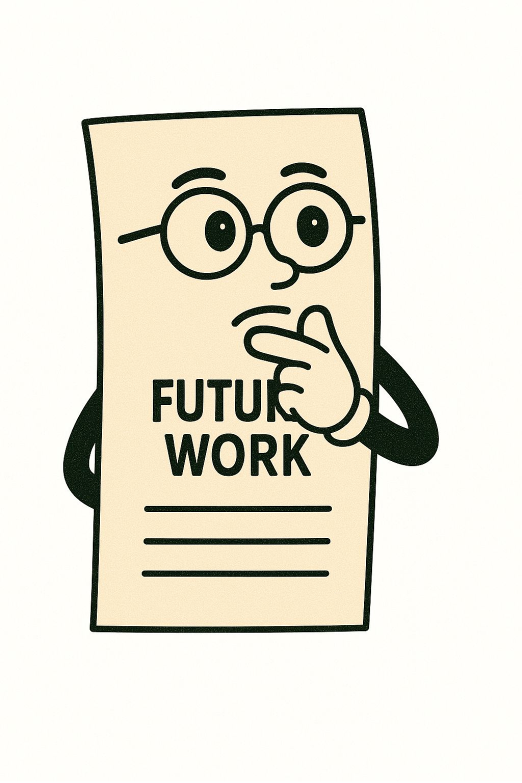
Schedule



Sep 19, '24 - Oct 10, '24	Initial website
Sep 28, '24 - Oct 12, '24	Team Standards/Inventory
Oct 11, '24 - Nov 1, '24	Tech Feasibility
Nov 1, '24 - Nov 22, '24	Design Review I
Nov 22, '24 - Dec 6, '24	Final Requirements
Dec 10, '24 - Jan 12, '25	Winter Break
Jan 17 - Feb 14	Design Document
Feb 15 - Mar 5	User Interface
Mar 1 - 7	Design Review II
Mar 8 - Apr 1	Process Algorithm
Mar 21 - Apr 4	Software Testing Plan
Apr 1 - 14	Report Generation
Apr 11 - 25	Capstone Conference
Apr 18 - 30	Testing/Validation period
Apr 30 - May 8	Final Documentation

Future Work

- Predictive Analytics
- Better Data Visualization Tools
- Direct Policy Editing
- Niche Class Coverage



Conclusion

Lumberjack Balancing is a Python application to automate the faculty workload assessment process for Dr. Scot Raab here at NAU

Thank You!

- Some images are AI generated