THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Pinch Analysis as a Modular Tool for the Analysis and Optimization of Industrial Chemical Processes

Ethan MacLeod

This report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering with Honours (BE(Hons)) at The University of Waikato.

Abstract

The industrial process heat sector in New Zealand is a significant contributor to the country's greenhouse gas emissions, primarily due to its reliance on non-sustainable energy sources. As part of the nation's efforts to achieve net-zero emissions by 2050, optimizing energy use in this sector has become a critical focus for reducing carbon output and addressing the challenges posed by aging infrastructure. One effective method for optimizing energy efficiency in industrial processes is Pinch Analysis, a widely used technique for identifying opportunities to minimize energy consumption and maximize heat recovery. Existing software tools that perform Pinch Analysis, though useful for reducing energy usage, are outdated and suffer from significant usability and design issues, highlighting the need for improved software solutions. This project seeks to improve access to insights gained from Pinch Analysis through the design and implementation of an intuitive user interface. Furthermore, this work integrates Pinch Analysis techniques into a broader simulation engine, enhancing the utility of existing software toolkit by increasing the level of information that is available to engineers seeking to optimise existing or designed chemical processes. This is achieved through integrating Pinch Analysis as a module into the Ahuora Adaptive Digital Twin Platform. This system is designed to provide enhanced usability, flexibility, and scalability compared to existing tools - validated through extensive user testing. By leveraging modern software practices, including improved data handling, usable interfaces, and integration with broader simulation software, the module aims to inform decisions regarding energy optimization. The project's focus on modularity ensures that the system can be expanded with future functionality, while its integration with existing infrastructure facilitates easy adoption by industry users. This dissertation outlines the development process, system architecture, and the outcomes of developing a specialized process engineering tool.

Acknowledgements

Many individuals have contributed to the success of this project, both through technical and emotional support. First, I would like to thank my project managers, Tim Walmsley and Mark Apperley. Without their guidance and help, this project wouldn't have been possible. I would be remiss not to mention in particular the invaluable support of Tim Walmsley. His patience in guiding me through the core concepts of Pinch Analysis and his willingness to answer all my questions has been instrumental to my understanding and the overall success of this work.

I would also like to extend my deepest gratitude to everyone at Ahuora - specifically Stephen Burroughs and Ben Lincoln. Their technical expertise, collaborative spirit, and emotional support have been a constant source of encouragement throughout this journey. Whether it was offering guidance on complex issues or simply providing motivation during challenging moments, their contributions have been indispensable to the completion of this project.

Finally, the emotional support and motivation offered by Simon, Tracey, and Asher MacLeod was truly indispensable for me throughout the duration of the project. Without their constant encouragement and belief in my abilities, I would not have been able to persevere through the most challenging times. Their unwavering support has been a crucial part of this journey, and I am deeply grateful.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Carbon Emission in the process heat sector

The New Zealand government is committed to achieving net-zero Greenhouse Gas (GHG) emissions by the year 2050 as a part of its broader climate strategy [1]. This initiative is driven by the need to address climate change and take steps towards a more sustainable future for New Zealand and the world as a whole. To achieve this goal, specific sectors have been identified to reduce GHG emissions, one of these sectors is the energy sector, and more specifically, the industrial process heat sector.

Carbon emissions in the industrial process heat sector are a major contributor to New Zealand's net GHG emissions. In the context of industrial sectors, process heat refers to the heat required for warming and heating, usually in the form of steam, hot water, or hot gases. The primary issue with process heat generation is its reliance on fossil fuels - roughly 55% [2] of the total process heat in New Zealand is generated using fossil fuels and unsustainable energy sources - mostly gas and coal. This accounts for 28% [3] of the total GHG emissions from the energy sector annually. In 2016, the sector produced 31.3 million tonnes of emissions [3], highlighting the significant opportunity for reducing emissions in this area. The New Zealand Ministry for the Environment released a set of policies in 2023 outlining the requirements for air discharge when generating industrial heat [4], demonstrating the importance they place on reduction in this sector.

Some of the challenges faced when discussing how to reduce GHG emissions in the industrial heat sector stem from the dated nature of New Zealand's industrial infrastructure. Most of this infrastructure was built multiple decades ago, which presents an issue in its compatibility with modern, more efficient technologies. Additionally, the cost of replacing this old infrastructure poses a significant monetary barrier for most companies, this including the cost of the equipment as well at the lost profits associated with downtime. Addressing this particular issue is difficult, and so alternative options need to be explored. One viable approach is to optimize existing systems to reduce energy consumption and, consequently, GHG emissions.

1.2 Structure of Dissertation

This dissertation has been organized into ten chapters that cover the various research and development stages that have been completed.

- Section 1 & Section 2 have been included to provide context and motivation for the development of this project, as well as highlighting the specific issues it will be addressing.
- Section 3 describes the previous development in this field, including past iterations of the OpenPinch software as well as the literature review conducted for this project.
- Section 4 outlines the process of development and research conducted throughout the duration of this project, with diagrams and workflows included to illustrate key steps.
- Section 5 to Section 7 documents the specific technical development undertaken, covering both the back-end and front-end components. Research related to testing is discussed in Section 6.3 and Section 7.4.
- Section 8 to Section 10 evaluates the effectiveness of the project, discussing how it meets the objectives set out in the project proposal. Additionally, these sections explore the

broader impacts of this work and outline potential directions for future research.

1.3 System Overview

In an effort to inform the rest of this document, Figure 1 has been included. This figure demonstrates an overview of the entire system and the interactions between its constituent components. The sub-system on the left is the back-end of the Pinch Analysis module as described by Section 5, including the API, the Django back-end, Pinch Analysis calculations, and the database. The right sub-system is a brief overview of the front-end implementation (Section 7), including the cache and client-side React components that make up the Pinch Analysis front-end.

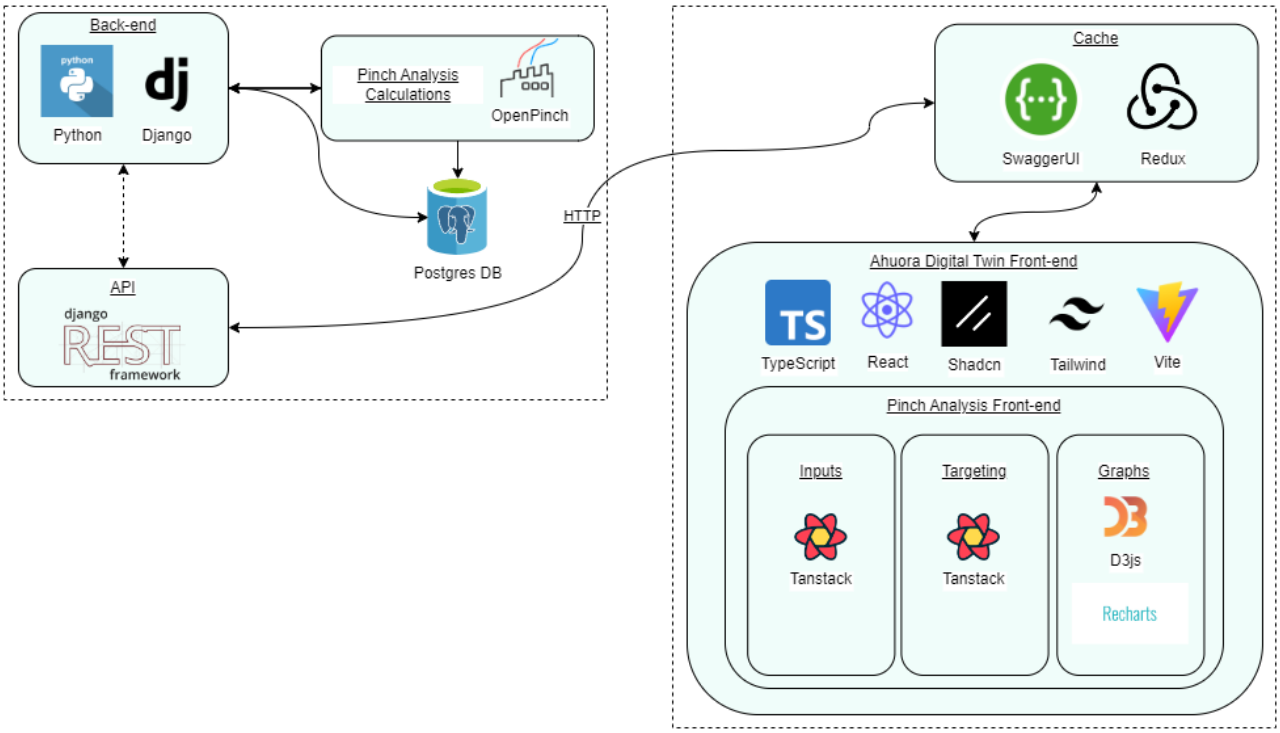

Figure 1: Holistic System Overview

A user is able to interact with the intuitive, browser-based user interface, with the complexity of data structures and analysis functionality abstracted away within the distributed back-end. The utility of this system lies in future development and how users interact with it. By formatting this as an online service, the Pinch Analysis module becomes hardware-agnostic, allowing any process engineer to access its functionality regardless of their location or machine. This modular design also reduces tech debt and future refactoring to accommodate more analysis modules and functionality.

2 Background

The work presented in this report is part of a larger, multi-disciplinary project. Consequently, some of the presented work goes beyond the limited scope of this immediate research, but is still required to achieve outcomes relating to the integration of live data analysis into the broader software platform. As such, this content is still relevant within the context of this specific project. Additionally, some work involves research for future implementations that cannot be completed with the platform's current capabilities. This chapter provides an overview of the Ahuora Digital Twin Platform, to provide context for the rest of the report.

'Project Ahuora' is a Ministry of Business, Innovation and Employment (MBIE) funded project that aims to decarbonise the process heat sector. By decarbonising, New Zealand's greenhouse gas emissions will be reduced. Cost savings from reduced energy consumption are anticipated, along with increased energy independence. This is a multi-disciplinary project that involves researchers from the University of Waikato, University of Auckland, Massey University, and other global universities. Chemical Engineers bring understanding of the chemical processes that are used in industry. Electrical Engineers bring understanding of the grid system and how to integrate renewable energy sources. Mechanical engineers bring understanding of how to design and build more efficient systems. Software Engineers bring understanding of how to model, simulate, and monitor complex systems.

A key objective is to develop a digital twin platform for the chemical processing industry. This platform will allow New Zealand factory operators to model their processes, simulate different scenarios, and monitor process state in real-time. This will enable factory operators to make data-driven and scientifically backed decisions on how to improve their processes. A digital twin can recognise where the factory is under-performing, suggest real-time improvements, and help plain future investments.

2.1 The Ahuora Simulation Platform

A key deliverable of the Ahuora project to date is the Ahuora Simulation Platform. This is a web-based platform that allows users to model a factory or other energy system, and simulate its performance. Much of the analysis functionality is achieved by leveraging the IDAES Process Systems Engineering framework within the back-end. IDAES is a Python library that provides tools for modelling and simulating chemical processes.

Currently, the platform can model a factory at a single point in time. The user specifies the properties of the factory, such as the flow rates of different materials, the temperature and pressure of different streams, and the efficiency of different unit operations. The platform then simulates the factory and provides the user with a report on the factory's performance.

2.1.1 Flowsheet Interface

Figure 2 shows a screenshot of the Ahuora Simulation Platform, as at August 2024. The user interface is divided into three main sections. The left-hand panel shows a list of unit operations from a factory, including pumps, heaters, heat exchangers, reactors, and material streams. The user can drag and drop these unit operations onto the canvas in the centre of the screen. The user can then connect the unit operations together to create a process flow diagram. The right-hand panel shows the properties of the selected unit operation, such as the flow rate of the material stream, the temperature and pressure of the stream, and the efficiency of the unit

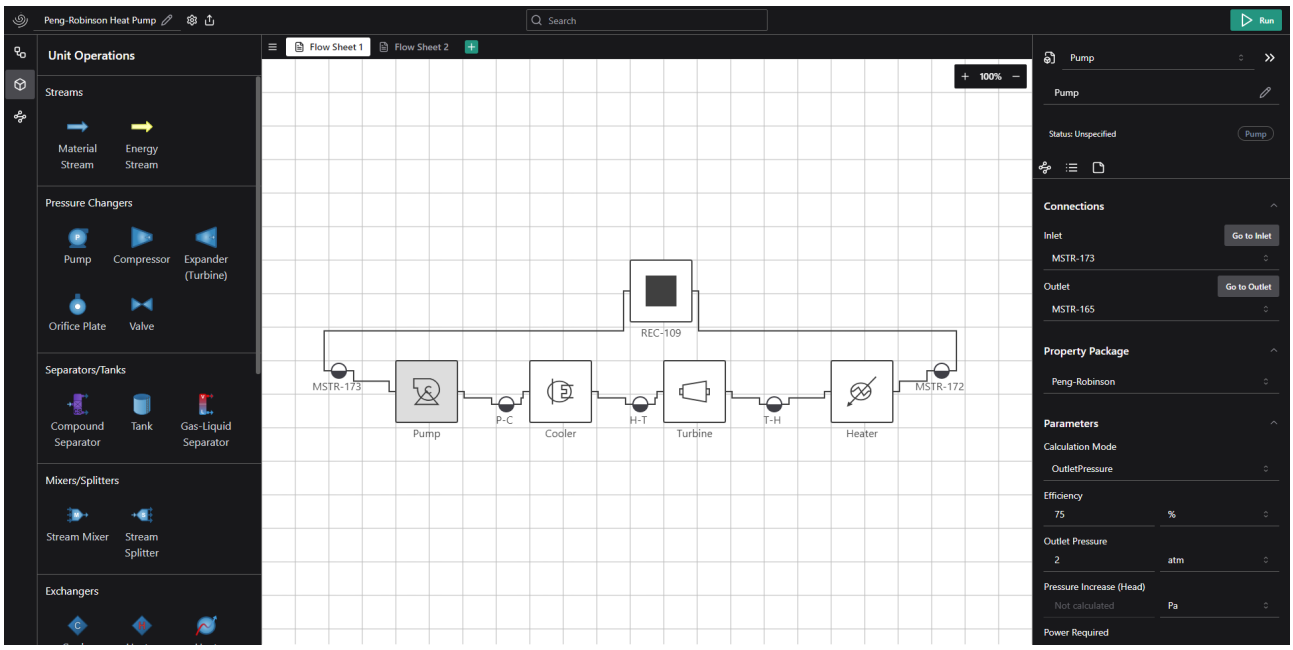

Figure 2: Example Flowsheet in the Ahuora Simulation Platform

operation. The user can edit these properties to simulate different scenarios.

The displayed flowsheet shows a simple heat pump cycle. The block on the top is a "recycle", specifying that the output of the cooler is fed back into the pump. A more complex flowsheet would replace the cooler and heater with heat exchangers, which exchange heat with their environment, but this provides a simple example.

2.1.2 Online Integration

The Ahuora Simulation Platform is designed as a web-based multi-user platform. This offloads processing and data storage to the server, allowing users to access the platform from any device with a web browser. Simulation can be very computationally expensive, particularly in advanced models, so this is a key feature. It enables simulation to be run in parallel on powerful servers, as the simulation is not done within the local web browser, but rather via distributed server infrastructure. This allows the platform to be used in industry without requiring significant upfront investment in hardware.

In future, this will also enable the platform to be used for real-time collaboration between multiple users. Its API allows it to be integrated with other software, enabling enhanced functionality, real-time updates, and broader interactions.

The home page of the platform, shown in Figure 3, provides a list of saved simulations, and allows the user to create a new simulation. This is not publicly accessible, as the platform is still in development, and user account functionality is not yet complete.

2.2 Platform Architecture

The platform is hosted in a Kubernetes cluster, located on-site. The Kubernetes cluster handles all web traffic, deployments from the private GitHub Repository, and scaling of services.

Within the platform there are various containers running through the Docker platform, that can

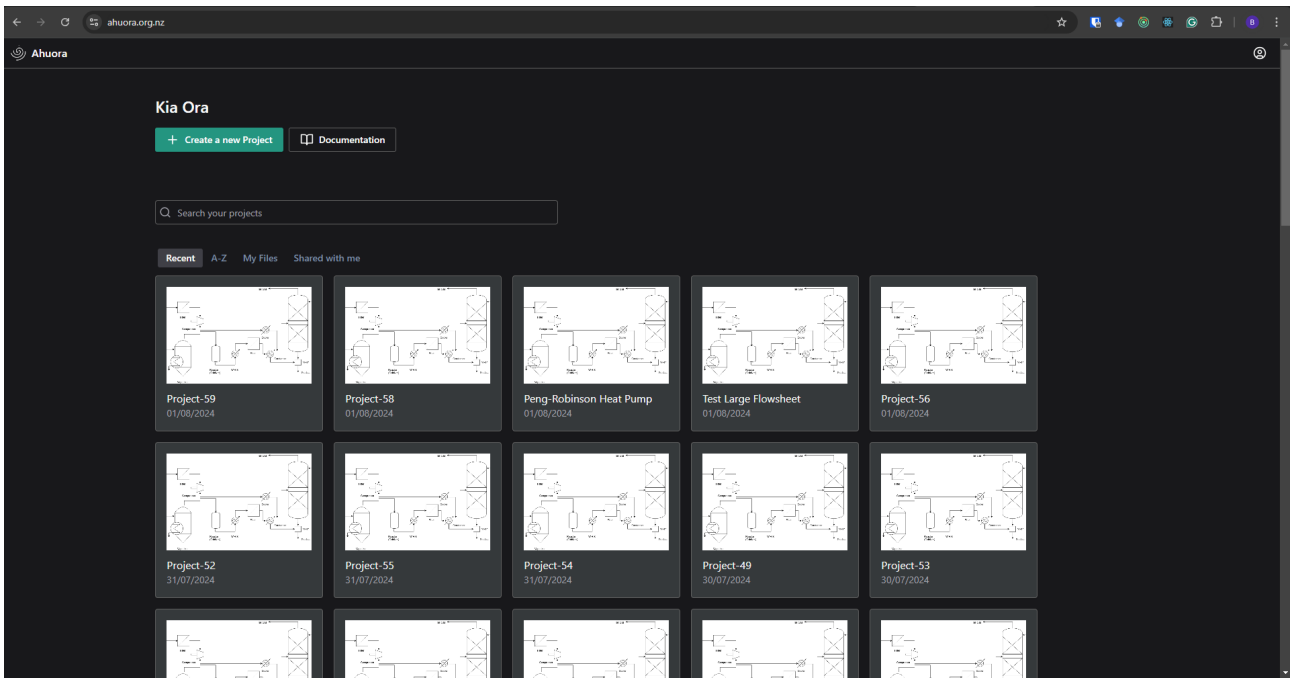

Figure 3: Homepage of The Ahuora Simulation Platform

be replicated as required to scale based on service demand. The Database stores all flowsheets and model data. The solving interface uses the IDAES Process Simulation software [5], built on the Pyomo equation-oriented modelling language [6], to simulate a factory in real time. The User Interface is written in React Typescript, and uses the Shadcn UI Library and Redux Toolkit for state management and communication with the Django back-end. The Django API links all the services together, and handles the business logic for functionality such as Pinch Analysis and Live Data processing.

2.3 Ongoing Projects

Multiple other projects are being completed in parallel by other team members. Each project has a distinct topic, and is largely independent, but collaboration is required to ensure that each other's work is compatible.The projects cover a variety of topics, from analysis and data processing, to usability and deployment.

2.3.1 Live Data Processing

The Ahuora Platform currently supports building and simulating a factory, but it has no functionality to link live data from a factory into the Ahuora Platform. By integrating real-time sensor data into the simulation, the platform can monitor the factories' performance, and suggest tuning that will optimise resource efficiency. The data can also be used to predict and avoid failures and downtime, a key problem where many resources are wasted. Additionally, models created in the Ahuora Simulation Platform during the design phase could also be used during operation, minimising overhead costs.

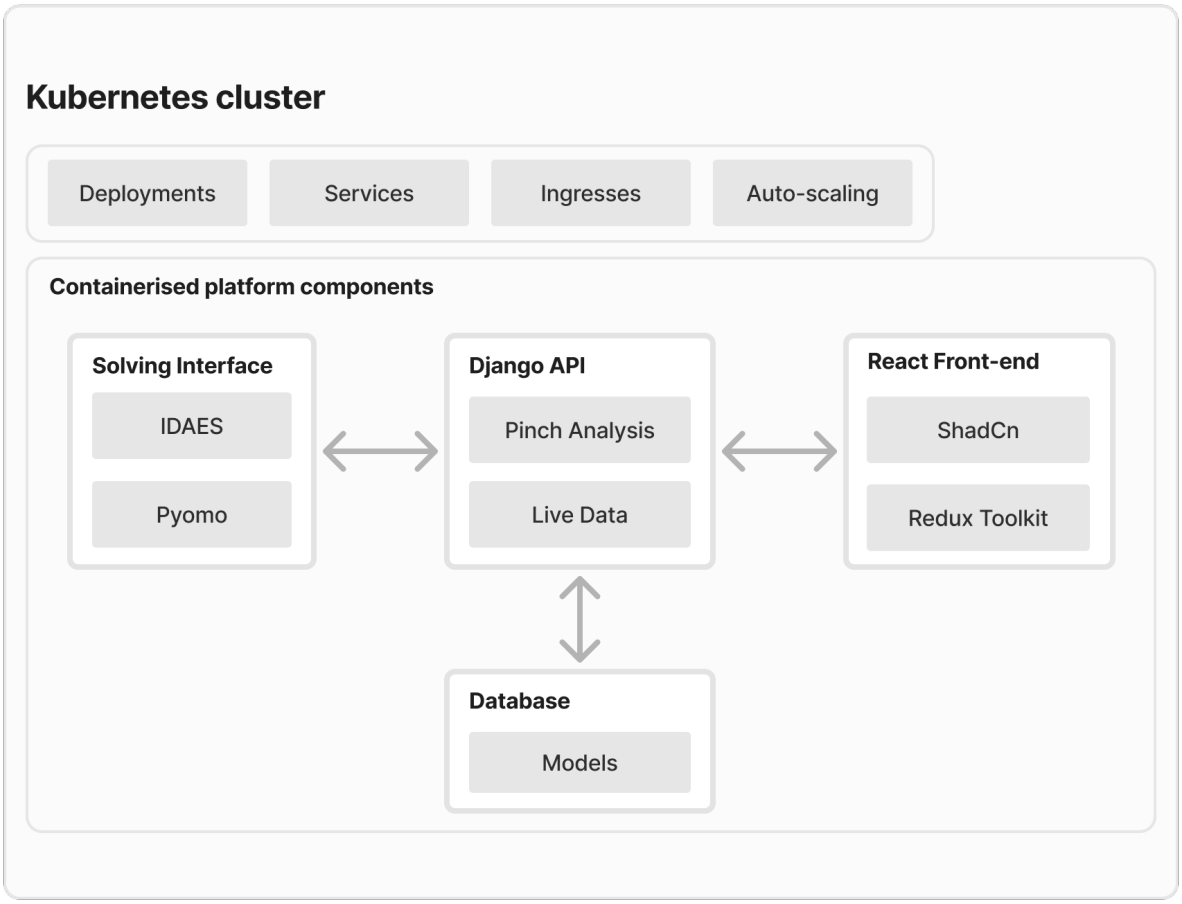

Figure 4: Architecture of the Ahuora Digital Twin Platform

2.3.2 Pinch Analysis

One of the key optimisation tools the Ahuora Platform provides is the Pinch Analysis module. Pinch analysis is the process of identifying heat recovery pockets in a given system, and indicating where heat can be exchanged. The end goal of pinch analysis is reducing the overall heat consumption of a process, which in turn results in lower operational costs for a plant, and less greenhouse gas emissions produced. From the processes that are modelled on the Ahuora Digital Twin Platform, there is a distinct need for integration with process optimization tools like Pinch Analysis to inform the decision-making process for operators and engineers.

2.3.3 User Interface Usability

This project aims to deliver an intuitive user interface and quality user experience in the Ahuora Digital Twin Platform through improving its UI through employing user centred methodologies including A/B testing, interviews, card sorting, the think-Aloud method, and surveys. Additionally, modern diagramming tools were reviewed and end users' knowledge were leveraged to the development of the ADTP UI. This project utilises software technologies and existing design components, Shadcn, for design implementations. Enhancing ADTP's UI increases user satisfaction, increases productivity, and minimises operational error.

2.3.4 Distributed Platform Deployment

Each of the software components making up the platform need to be deployed together within a distributed environment that allows for efficient allocation and use of computational resources. This is achieved within the context of a Kubernetes cluster, on which several replicas of each platform component may run at any given time based on system load. The ability to scale workloads in response to demand increases the capacity of the platform to perform process modelling and simulation, and handle a greater number of users.

2.4 Other Long-term Platform Objectives

The projects listed pave the way for future work on the Ahuora Digital Twin Platform. This includes expanding the number and type of unit operations supported, and supporting a wider range of chemical processes. It is intended that the Ahuora Platform also support Hybrid modelling, where Machine Learning Models are used to represent complex unit operations that do not have an exact mathematical representation. Analysis functionality to be added includes Dynamic Simulation, to model the factory over time and predict the effect of changes in the system, Process Variable Optimisation, to calculate optimal operating conditions, common diagramming and reporting functionality, and scheduling tools.

Due to the long-term context of the Ahuora project, work in this dissertation is constrained by the future requirements of the system. Care has been taken to ensure that the methods outlined can be extended to support the future objectives of the Ahuora Platform.

3 Foundational Work

3.1 Literature Review

As a part of the course requirements for ENGEN582, as well as to inform the development of this project, a literature review was conducted. This literature review managed to provide key information on core concepts, which greatly contributed to the success of the project. Some of the topics covered included Pinch Analysis, UI development and testing, and general software engineering practices.

First, the current state of pinch tools were examined, along with some historical context relating to the progression of graphical process engineering methods. From this, pinch analysis tools were found to be generally lacking in usability and intuitive design, with most of the popular pinch tools having an outdated or minimal UI.

In the design section, some key design principles were explored within the context of applying them to these pinch tools. This section established relevant applicable design processes, alongside appropriate UI development techniques and evaluation methods. Wire framing and prototyping were identified as key techniques that these systems would benefit from and testing methods such as heuristic and A/B testing were explored.

Finally, software principles that indirectly contribute to the user experience, such as maintainability, usability, and scalability were identified within the confines of the project, while graphing libraries used to facilitate the graphical outputs of the system were evaluated against each other.

3.2 Excel Open Pinch

Excel OpenPinch is an open source Pinch Analysis software developed with Microsoft Excel and the VBA programming language. Excel OpenPinch includes many modules for Pinch Analysis and other analysis tools. The software is designed to simplify the process of energy optimization in industrial systems by providing a Graphical User Interface (GUI) and user-friendly platform within Excel.

Excel OpenPinch allows users to input stream and utility data to retrieve Pinch Analysis targets and graphical outputs using the Problem Table Algorithm. These outputs help users identify opportunities for heat recovery and make informed decisions on how to optimize energy use within their systems. By analyzing these results, users can pinpoint areas where energy consumption can be reduced, improve overall system efficiency, and minimize waste, leading to both cost savings and environmental benefits.

Some of the analysis modules packaged with the latest release of OpenPinch include:

- Totally Integrated Site Targeting
- Total Site Heat Integration
- Turbine Shaft Work
- Area and Cost Targeting
- Energy Retrofit
- Thermal Exergy

- Graphing Outputs (For all of the above)
- Interim Problem Table Outputs

Due to the number of diverse modules, Excel OpenPinch provides a comprehensive suite of tools for a chemical or process engineer to analyze, optimize, and design more energy-efficient systems. From identifying pinch points to evaluating utility options and heat integration strategies, the software enables engineers to streamline processes, reduce energy consumption, and enhance sustainability across various industrial applications.

Some of the key issues regarding Excel OpenPinch include its somewhat unintuative GUI, lack of usability considerations, and lack of integration with process modelling software - a key functionality that is required of modern energy optimization workflows. As technology and chemical engineering develops, there is a present need for these problems to be addressed. In this case, being able to integrate the OpenPinch functionality with the aforementioned Ahuora Digital Twin Platform would effectively update the GUI of OpenPinch while also providing it a process simulator to integrate into. These issues specifically provide the motivation behind this project's development.

3.3 Python Open Pinch

Python OpenPinch can be thought of effectively as the Python version of the Excel OpenPinch software. The Project was originally developed as a 1:1 translation of the Excel VBA code as Python code, however this code was based on a now deprecated version of Excel OpenPinch and is missing some of the changes and new modules that can found found in Excel OpenPinch. The modules included in this program are:

- Totally Integrated Site Targeting
- Total Site Heat Integration
- Turbine Shaft Work
- Area and Cost Targeting
- (Partial) Energy Retrofit
- (Partial) Thermal Exergy

Notably, the Python OpenPinch code uses an Excel OpenPinch workbook for it's inputs and outputs. The workflow for the program opens an Excel sheet to allows users to input their streams/utilities, calculates the targets and graphing point using python code, then writes to the workbook for its outputs. This tight coupling with an outdated software is not ideal from a software maintainability and reliability perspective, and is one of the many reasons why the module required an updated version.

Additionally some of the core analysis functionality is missing due to the version discrepancies (v2.30 vs 3.12), the primary example of this is that graphs aren't calculated correctly, and are entirely disconnected from the standard outputs. Beyond this, there is a distinct need for a Pinch Analysis system to be able to efficiently evolve alongside the research and progress within its field and in its current form this is difficult.

3.4 Summary

From examining the previous solutions to Pinch Analysis software, several pain points have been identified that neither of the previous projects have adequately solved. The development of a process simulator such as the Ahuora Digital Twin Platform is the ideal time to address some of these issues with a new project based on the software and principles of the previous work.

The focus when adjusting these systems for the Ahuora Adaptive Digital Twin Platform is to refactor the existing Python OpenPinch to improve upon its stability and flexibility, then to focus on usability and integration into a broader software platform. Some of the specific areas that need to be addressed are:

- Greater usability / GUI
- Loosen coupling and reliance on the Excel OpenPinch program
- Integrate functionality with Ahuora Adaptive Digital Twin Platform
- Fix existing modules
- Refactor outputs for Ahuora Adaptive Digital Twin integration
- Provide a baseline for future modules
- Apply software considerations (maintainability, reliability etc.)

In conclusion, addressing the limitations of previous Pinch Analysis tools within the Ahuora Adaptive Digital Twin Platform is a critical step towards enhancing the usability, flexibility, and integration of insights that Pinch Analysis provides. By refactoring the existing Python OpenPinch to improve stability and reduce reliance on external programs such as Excel, the stability and flexibility of the resulting system is greatly improved. The incorporation of this improved code with refined data structures and a new browser based user interface as a modular component of the broader Ahuora Platform is significant for both the accessibility of Pinch Analysis as a standalone technique, as well as for the overall utility of the platform with regard to process optimisation.

4 Methodology

Figure 5 is a graphic demonstrating the initial estimated timeline for this project. While this figure contains all of the steps that were required, the actual timeline differed significantly from this. The initial timeline follows a traditional waterfall model, where tasks are completed in sequential stages. In contrast, the project was developed using agile techniques, organized into one-week sprints. This allowed for more flexibility and responsiveness, as various sections of the project were often worked on simultaneously. In response to outcomes revealed through testing, requirements and priorities evolved over time, leading to frequent overlaps between development stages.

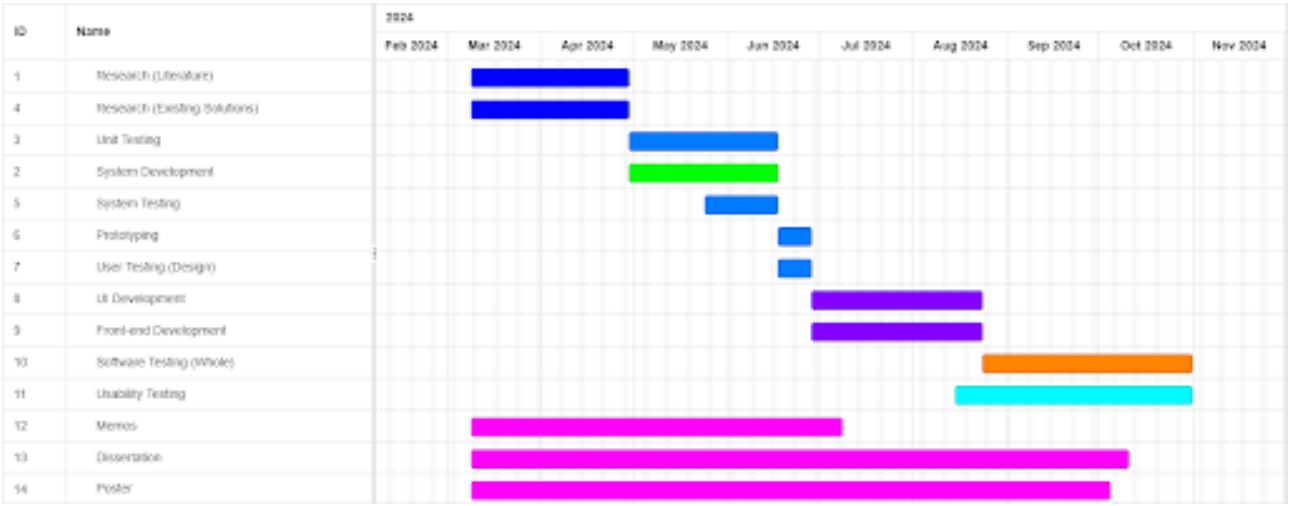

Figure 5: Estimated Project Timeline

As mentioned, the general methodology utilized in this project was an iterative agile design philosophy which organized tasks into one week sprints. Each sprint focused on goals achievable within this one week period, allowing for continuous progress and fluid adjustments. For a significant portion of the project, design and development sprints were alternated. After one or two weeks of development, a design and testing sprint would follow, allowing for feedback from usability testing or design adjustments to be used in subsequent development cycles. The benefits of this were that development cycles would inform designs and testing phases, and in turn development cycles were informed by testing and design. Following these procedures, the project naturally evolved in response to user feedback without losing any of the original requirements for the system.

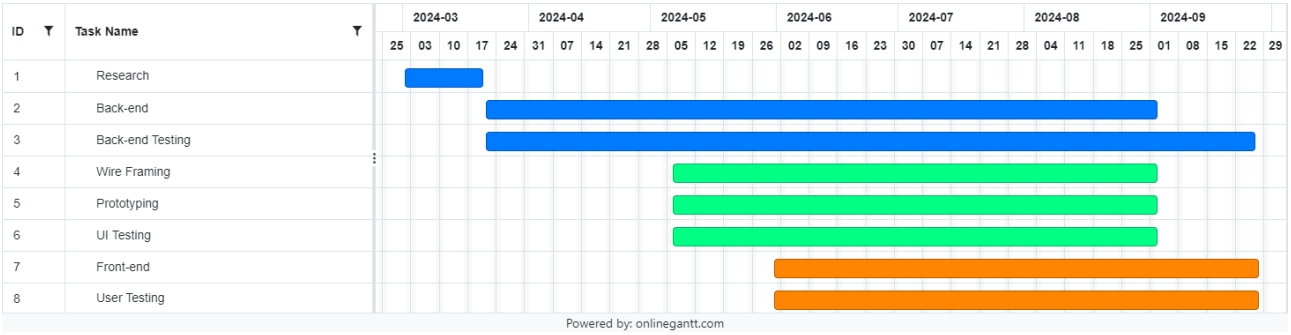

Figure 6: Actual Project Timeline

Major development sections largely remained the same across the proposed and actual timelines. The general work was still broken down into four major portions; back-end, front-end, design, and testing. As the project evolved, the milestones for each of these sections became ambiguous, and a stronger emphasis was placed on a coherent final product rather than incremental "releases" with partial functionality. This is reflected in the actual timeline seen in Figure 6.

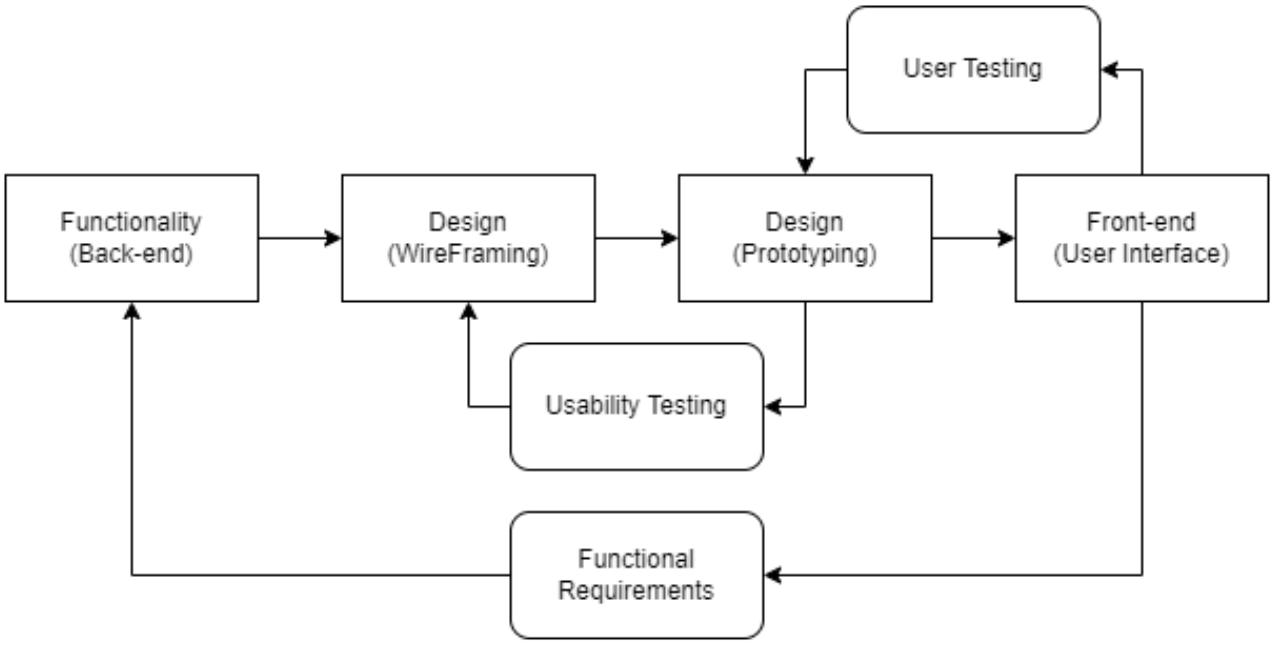

Figure 7: Development Work-Flow

Figure 7 depicts the general workflow undertaken for the development of this project. Firstly, the current functional requirements are developed - these can typically be considered "back-end" developments, however in some rare case these also fall within the scope of the front-end. From these requirements a rough wire-frame is developed. This is a low fidelity model of the user interface that is used for a fast testing cycle. After testing is conducted on the wire-frame, a prototype is developed in Figma [7]. This prototype is much closer to a 1:1 model of the final user interface, and even includes some of the minor functionality (eg. interaction) that the front-end requires. Testing is conducted on this prototype, with changes to the wire-frame and prototype being made through the usability testing cycle. Once a prototype is considered to be usable enough to implement, the front-end is developed based on the prototype. User testing is then conducted on this to restart the prototyping/design cycle.

5 Logic Implementation

5.1 Django

The back-end for this project was created using Python and Django [8]. Django is a high-level web framework designed for web development by providing a clear structure for building applications. For this project, Django was primarily utilized for its powerful database management capabilities. Its Object-Relational Mapping (ORM) system excels at simplifies the process of defining and managing database structures. The decision to use Django was influenced by its ability to integrate into the existing Ahuora Digital Twin back-end which is also built with Django, while also ensuring compatibility with the Python-based OpenPinch work. This combination allowed for an efficient workflow and simple back-end management/integration.

Alongside Django, a library called Django REST Framework [9] (DRF) was used to develop a Django RESTful API. This API allows for Create Read Update Delete (CRUD) operations to be performed on the database, which is the primary method for interaction with the data of the system as a whole. The utility of using a library like DRF is that it allows for fast and smooth integration with the existing data structures, tables, and relationships present through Django. One example of this is the ModelViewSet - a native view set that DRF offers. A ModelViewSet takes a serializer and query set to automatically generates basic CRUD operation views. This saves time for development and provides a basis of consistency across the API for managing operations.

5.1.1 Data Structures

To facilitate the integration Python OpenPinch with the Ahuora Digital Twin Platform, it was necessary to use Django to manage and store data in a PostgreSQL database. The first step was identifying the data categories that needed to be stored. Following this, a comparison was made between the existing data classes in Python OpenPinch and the class structures already in place within the system. This research aimed to align Django's best practices with the existing architecture, ensuring that the data was structured in an intuitive and maintainable way for future development. From this, the following super structures were created:

Name	Description
PinchInputs	Values read from the front-end to the back-end. These are considered the required fields for execution of the OpenPinch module. Included are Streams, Utilities, etc.
PinchOutputs	Values created by the OpenPinch module. Graphs, Targets, etc.
PinchOptions	Optional values passed in that dictate which modules are used, and globally accessed variables. For example which graphs to output.

Table 1: Python OpenPinch Django Super-Structures.

From these super-structures, each contained multiple nested data structures that aligned seamlessly with Django's ORM. These structures were designed to allow for independent extraction of data, meaning that each object's data could be accessed without needing to interact with other related data. This separation of concerns greatly simplified front-end development, as there was no need for additional formatting or manual data extraction. The data could be consumed directly by the front-end, streamlining the integration process and reducing development time. This approach allowed the front-end to remain lightweight as well as development

to be focus on consuming data and usability.

5.1.2 Serializers

Due to the custom Django data structures that were implemented for the various inputs and outputs, there was a need to define serializers to manage these data structures.

Serializers in Django are used to validate the correctness of data being passed into a model, as well as further define relationships between models for external resources. The Ahuora Digital Twin Platform uses a tool called Redux RTK Query for managing the front-end cache and the values inside of it (see Section 7.1). Redux relies on the serializers, not the model definitions, for determining what values on an object should be passed into or returned from a model. This makes the serializers incredibly important for future front-end development, as the serializers need to be defined as they should appear on the front-end.

For this project, serializers have been defined for all major Django models as well as any models with nested data structures. These serializers include:

- Pinch Options - Options have nested `PropertySet` and `PropertyInfos` for appropriately displaying the options on the front-end, these nested objects needed to be included in the serializers to retrieve the specific options value for displaying on the front-end.
- Pinch Inputs - A `PinchInput` is a structure that contains all of the `PinchStream`s and `PinchUtility`s, required for extracting the input data for the Pinch Analysis module as well as creating the objects on the front-end.
- Pinch Outputs - The final super-structure with nested elements, `PinchOutputs` contains the highest degree of nested elements of all the database objects. Contained inside this serializer are the targeting outputs, as well as graphical outputs with graph segments and points. Due to the large number of nested serializers, strictly managing the data and inputs for these objects is particularly important.

5.1.3 Views

The primary mechanism through which interactions with the Pinch Analysis system are managed is through the various API endpoints ("views" in Django REST Framework). Django REST Framework has many types of views that are available to use, in this instance the ones used were an API View - where a developer has fine grained control over the data flow and operations, and a Model View Set - a class of API View that provides a default implementation for CRUD operations on a model.

The only custom view required for this project was a general `Calculate Pinch` endpoint, designed to initiate the pinch targeting algorithm. This view was implemented as a POST operation, where the client sends a `flowsheetID` — the identifier for the object containing the pinch structures. Upon receiving the request, the Python OpenPinch system is initialized with the corresponding inputs, as detailed in Section 5.2.2. The outputs are updated in-place directly within the database, where values can then returned to the front-end as required in an 'on-request' fashion.

Notably, no data beyond the HTTP response headers and status code is returned to the user from the initial pinch analysis request. This decision aligns with Ahuora Digital Twin Platform's

architecture, which is generally structured to not use the response payloads as data for the front-end. Instead of returning the data within the HTTP response, the platform's caching system is leveraged, allowing the front-end to pull updated data by GET requests, which reduces response time and optimizes performance. This design paradigm is utilized for all the views in this project, including the Django model views

The use of the ModelViewSet, expedites the process of creating the necessary API endpoints for some models. From the data structures that were created in this project, a ModelViewSet was made for each of the externally editable super classes. These super classes are the highest level of each table created for this project, except for the pinch outputs, which are only created and edited internally. Figure 8 shows the auto-generated schema for one of these ModelViewSet views. Included in the generated views are all the basic CRUD operations, these being; GET all, POST, GET one, UPDATE all fields, partial UPDATE, and DELETE.

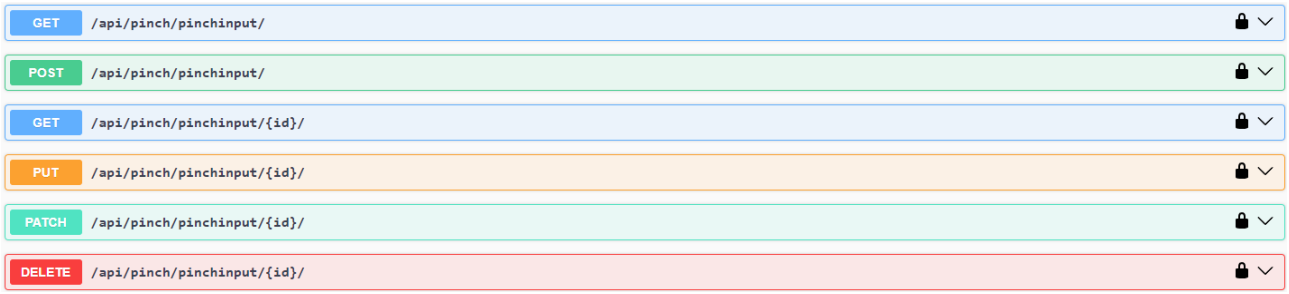

Figure 8: SwaggerUI Schema for ModelViewSet

Overall, these DRF practices expedited the API development process, while also structuring the API in a readable and standard way for future development. This standard was exceptionally useful during front-end development where there was no need to refer to documentation or manual checks to correctly consume the API.

5.2 Pinch Module

5.2.1 Implementation

Figure 9 demonstrates the basic logical progression through the Pinch Analysis module for context purposes. Starting with accessing the module through the Pinch Analysis view, the system will first serialize the input data and create classes from it. From this, the system will initiate the individual modules based on conditional parameters provided by the options data.

5.2.2 I/O

For this project, the Python OpenPinch module was utilized to perform Pinch Analysis and implement the Problem Table algorithm. During the technology review of the module, a significant limitation identified was the systems heavy reliance on a Microsoft Excel workbook for managing inputs and outputs (I/O). While this approach was necessary in previous iterations, it does not align with the project's objectives for usability, maintainability, and scalability. The dependency on Excel introduces limitations in terms of integration with the Ahuora Digital Twin Platform and contains unnecessary complexity for both development and users.

Figure 10 shows the initial I/O workflow that involved the Excel workbook. When the system is started, an Excel workbook would open, allowing the user to enter the stream data, utility

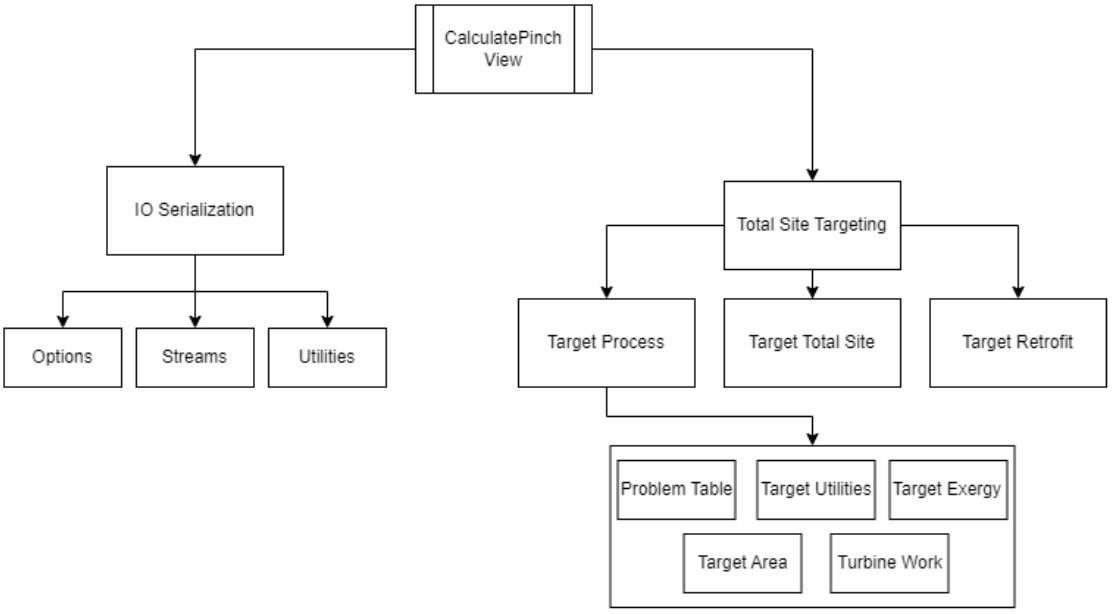

Figure 9: OpenPinch Module Flow

data, and options to perform Pinch Analysis. Once the data has been entered, the Python file extracts the data by iterating over the columns and rows, creating a instance of a class as it goes. After the Problem Table algorithm is applied to this data, the data is reformatted into an appropriate structure and is written to separate output Excel sheets. After the outputs are finished, the user can save and close the Excel workbook.

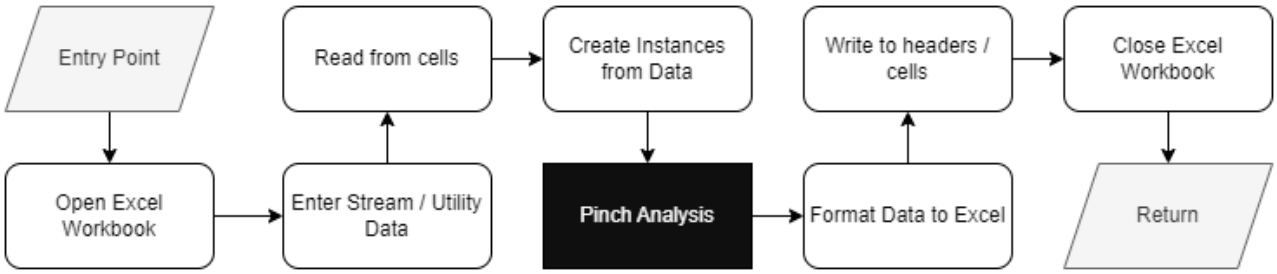

Figure 10: Python OpenPinch I/O Work-Flow

To address this reliance on the OpenPinch workbook, the Python OpenPinch system was initially decoupled from its Excel-based I/O, and modified to leverage the existing database structures and back-end systems within the Ahuora Platform. This development was undertaken through a complete overhaul of the `dataPreparation.py` file that was originally used to extract the stream and utility data present within the Excel workbook. Instead of reading from a file, the data structures for the inputs have been intentionally modelled to replicate the final classes in the Python OpenPinch module. This allows for for database entries to be directly read into OpenPinch class structures through creating instances of the class with Django table data.

Figure 11 demonstrates the improved workflow after the process was streamlined and integrated with the database. Before the Pinch Analysis system is initiated, the user creates the required objects through the front-end platform. Once a request is sent to the Calculate Pinch endpoint, these database rows are read directly into the OpenPinch class structure, and the targets are

calculated. Once calculated, the outputs are established through the creation of database objects. These outputs are not directly returned until requested by the front-end via a separate API call.

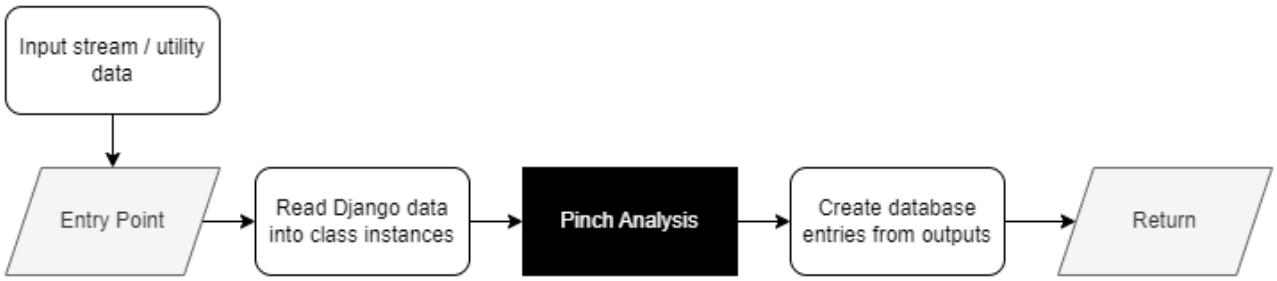

Figure 11: Improved I/O Work-Flow

This migration from Excel-driven to database-driven I/O resulted in a substantial increase in performance for the entire system. Removing the overhead of Excel manipulations for every run reduced the processing time, allowing for faster Pinch Analysis calculations. By directly querying the database and importing structured data into the Python OpenPinch module, the system can handle larger datasets more efficiently while also increasing scalability and maintainability. An additional benefit for this approach to data handling is that it allows developers to leverage the Django unit testing suites. Previously, doing integration and unit testing on the Python OpenPinch system was complicated, as Python OpenPinch was tightly coupled with the Excel workbook. However, now that the system is decoupled from Excel and integrated directly with the database, developers can write unit tests to ensure that each component of the system functions as expected and verify that future developments do not result in broken functionality.

5.2.3 Error Handling

Error handing in the original Python OpenPinch module was not ideal from a usability and maintainability standpoint. The extent of the previous error handling involved logging a specific error message (manually defined by a string) to standard output and exiting the program. This methodology was acceptable for a small scale standalone system, however when integrating with a larger system the error handling needed to be migrated to use the DRF and Ahuora defined management.

The solution to this was to change these specific instances to custom Python exceptions, as recommended by Django. When a Python exception is called inside a view, there is custom handling with Django in place at the view definition level to handle returning valuable data back to the user. By defining a custom exception of an appropriate type, it is possible to return the error data in the request body alongside the correct HTTP status code.

Having this functionality is fundamental for future development and integration testing between the front-end and back-end. This new system also allows for meaningful data to be returned while testing as well as if an error occurs during execution.

5.2.4 Error Mitigation

The Python OpenPinch system was not in a complete state before development on this project started. Many of the modules in the system were incomplete, and in most cases produced

incorrect outputs. Before any design or front-end integration was done, these errors needed to be fixed for testing purposes.

Some of the bugs encountered included:

Issue	Description
Problem Table Calculation	Incorrect array indexing during the transition from VBA to Python led to wrong or missing calculations at specific indexes in the problem table. This table forms the foundation for most of the targeting and subsequent calculations, making it a critical component. Errors caused by this bug included missing graph points, incorrect pinch points, and misaligned energy targets. Fixing this was essential to ensure accurate downstream calculations.
Composite Curves	Plotting composite curves encountered "off by one" indexing errors due to differences in how Python handles arrays compared to VBA. These errors resulted in incorrect graph segments and misleading visualizations of the heat exchange between streams. Correcting the index logic ensured accurate representation of the heat cascade.
Grand Composite Curves	Similar indexing issues affected the grand composite curves (GCCs). Errors in the problem table calculations were especially pronounced here, leading to distorted curves that misrepresented energy flows and opportunities for process integration. These inaccuracies significantly impacted the visual analysis and energy retrofit decisions.
Energy Retrofit Curves	Indexing issues during the calculation of Energy Temperature Difference (ETD) led to incorrect retrofit curves. The ETD is critical for assessing the feasibility of process modifications to improve energy efficiency. Errors in indexing skewed the ETD results, misguiding the retrofit strategy by suggesting incorrect heat recovery options. Correcting these errors ensured reliable identification of energy-saving opportunities.
Stream Data Input	The OpenPinch system automatically adds default heating and cooling utilities to balance the heating requirements of the process. Prior to fixing the bug, these utilities were not being added correctly, causing discrepancies in heat supplier and receiver targets. Proper utility handling is crucial for setting correct boundary conditions and ensuring all streams are balanced for accurate pinch analysis.
Degree of Integration	The degree of integration, a measure of how much process heat is reused between streams, was miscalculated due to accessing the wrong variables. This led to inaccurate assessments of process efficiency and integration potential, undermining the validity of recommendations for heat recovery and process improvements.

Table 2 continued from previous page

Issue	Description
Balanced Composite Curves	The Balanced Composite Curves (BCC) calculations were impacted by indexing issues originating from the problem table. Incorrectly balanced curves led to errors in area and cost calculations, which are essential for evaluating the economic feasibility of proposed heat exchanger networks. Fixing the indexing ensured proper balancing and reliable cost estimations.
Thermal Exergy Targets	The Python OpenPinch module was using an outdated method for calculating thermal exergy targets, compounded by missing exergy calculations. Exergy targets are vital for understanding the true potential for work within the system, and outdated or missing calculations reduced the accuracy of energy efficiency evaluations. Updates to the calculation methods restored correct exergy target assessments.
Area and Cost Calculations	Area and cost calculations were either missing or incorrectly implemented, which affected the evaluation of the economic implications of process modifications. These calculations are crucial for determining the feasibility and return on investment of energy-saving measures. Implementing accurate formulas ensured correct cost-benefit analyses.

Table 2: Issues found in Python OpenPinch

Most of the problems with the Python OpenPinch system can be derived from the array indexing differences between VBA and Python. As previously mentioned, Python OpenPinch was a one-to-one translation of the VBA code. Traditionally, most programming languages index arrays at zero, so a value at index zero is the first in the array, VBA however indexes at index one. The previous developer managed to translate this well for the most part, however at key areas mistakes were made that lead to larger issues. Problem Tables, Balanced Composite Curves, Area and Cost, and all the graphing issues can be attributed to these indexing mistakes.

All the mentioned issues have been fixed for this project, as each specific OpenPinch module is crucial to the functionality of the project as a whole. Resolving these bugs not only restored the integrity of the calculations but also enhanced the overall accuracy and reliability of the system. These fixes have ensured that all heat integration analyses, energy efficiency assessments, and retrofit strategies are based on correct data, providing dependable insights for decision-making.

For systems such as this one that are used to inform decisions on real-world processes, maintaining the integrity of all outputs and calculations is especially important. Any inaccuracies can lead to incorrect decisions, resulting in increased operational costs, missed opportunities for energy savings, or even safety risks in process modifications. The motivation for this project lies in reducing energy consumption, and producing faulty results undermines this objective. Ensuring the accuracy and reliability of every module is critical to building trust in the tool's outputs, which stakeholders depend on for optimizing energy efficiency and achieving sustainability targets.

6 Design

6.1 Wire-framing

To align with the iterative, agile design style outlined in Figure 7, the first step required was wire-framing. In contrast to the other two phases for the front-end work, the wire-framing takes a very short amount of time but comes with the caveat of a low fidelity model without any functionality to examine. This made it perfect for weekly design cycles, and composed the bulk of all design work for this project. Most of the testing phase was also conducted on these wire-frames, as specific requirements and details could quickly be drafted for approval and feedback.

The format of wire-frames for this project was hand-drawn rough sketches of the system. For components and sections with more detail, these would be extracted from the drawing and another design iteration would be conducted on just the singular component.

The specific wire-framing workflow can be seen Figure 12. The process starts with a rough drawing of the user interface for what is currently being worked on. This can be the whole page, specific components, or groups of components. The most common example of this would be a design for a web-page, with separate wire-frames for components that are too intricate to accurately represent with a single wire-frame. Next, a series of tests would be conducted on the wire-frames, which is further explored in Section 6.3. From the results of this testing, one of three things occurred:

1. Component Changes - If test participants indicated that the broad drawing was acceptable, however certain parts need more work, these parts were extracted into their own wire-frame and the workflow was restarted with these as the focus point.
2. Layout Changes - If the general layout of the wire-frame was lacking, the work-flow re-entered another design cycle aimed at exploring ideas from the participants or otherwise overhaul the design as a whole.
3. Passed Testing - if no further valuable feedback was gained from the testing, the design was considered ready for prototyping, and prototyping was then undertaken as defined by Section 6.2.

Some examples of wire-frames that were undertaken for this project are included below as Figure 13. From these examples, the low quality nature of the wire-frames can be seen. These wire-frames provide just enough detail for testing to be conducted on them, without sacrificing any needless time to aesthetics.

As mentioned in Section 4, these wire-framing design cycles were continued up until the conclusion of this project and provided invaluable insight into the direction of the User Interface. Undertaking this wire-framing process allowed for rapid prototyping and design iterations, ensuring that feedback from users and stakeholders could be quickly incorporated. This iterative approach also helped identify potential usability issues early, which expedited development and ensured that the final product was both usable and met the project's objectives. Overall, wire-framing was a crucial tool in refining the interface to meet both functional and aesthetic requirements.

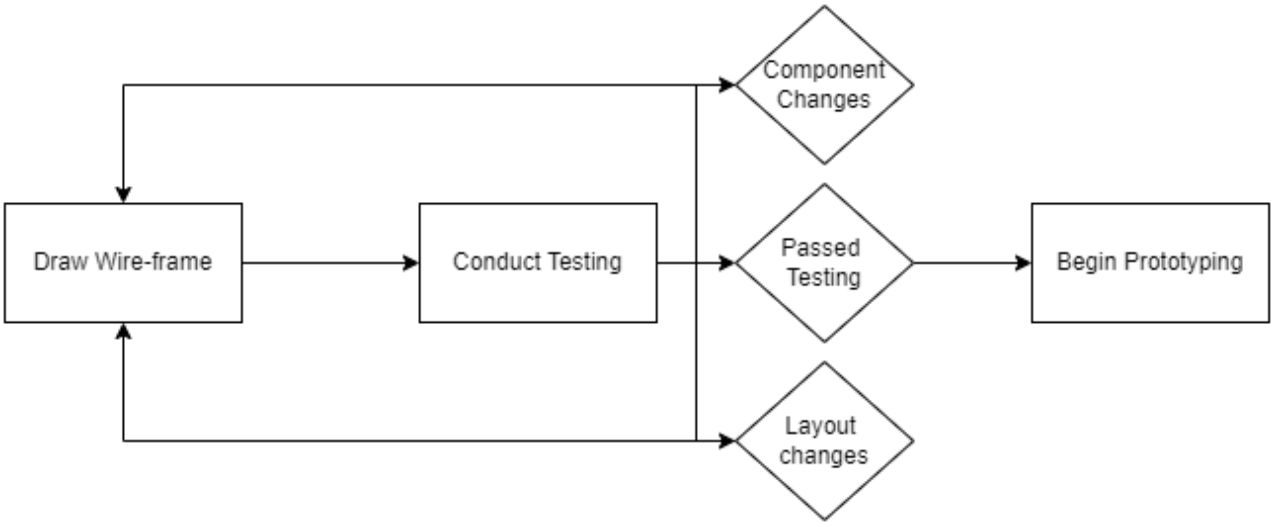

Figure 12: Wire-Framing Work-Flow

6.2 Prototyping

When designing the user interface for a complex system such as this one, there is a need to be able to test the functionality as well as the general design - this is done through prototyping. This allows user to indicate problems beyond visual issues, such as the system workflow and interactions between individual components. The prototyping for this project was done through the online tool Figma. Figma is a web-based collaborative tool for building complex prototypes and includes features that allow for complex functionality to be added to designs. The reasons Figma was chosen for the prototyping for this project are outlined as follows:

- Existing Designs - The prototyping that was completed previously for the Ahuora Digital Twin Platform was already done through Figma. As the final goal for this project was integration with the Ahuora Platform, many of the existing components and layouts were required for the design of the Pinch Analysis module. By reusing common components, prototyping time is reduced drastically, and there is a consistent basis across the prototypes that allows for smooth visual integration between the two. In addition to this, as future design and prototyping work is completed on the Ahuora Platform, these changes can be immediately reflected in the Pinch Analysis module design through Figma's shared components system - ensuring there is no redundant work being done.
- Cloud Serving - Another function that was found to be immensely helpful during this project was the online nature of Figma. Being able to show stakeholders real-time updates to the prototyping was invaluable for communication and transparency. During testing, Figma was occasionally used to conduct testing online over video, removing the requirements for in-person testing that would have slowed the design cycle.
- UI Library Integration - A point that will be discussed later on in Section 7.2 is the component library Shadcn [10]. The primary component library used for front-end development was Shadcn, and Figma integrates well with Shadcn through the Shadcn UI Toolkit [11] - a Figma library that allows developers to use Shadcn components and Tailwindcss [12] colours inside of Figma. This allowed for a direct 1-1 model of the final system to be composed as a prototype.

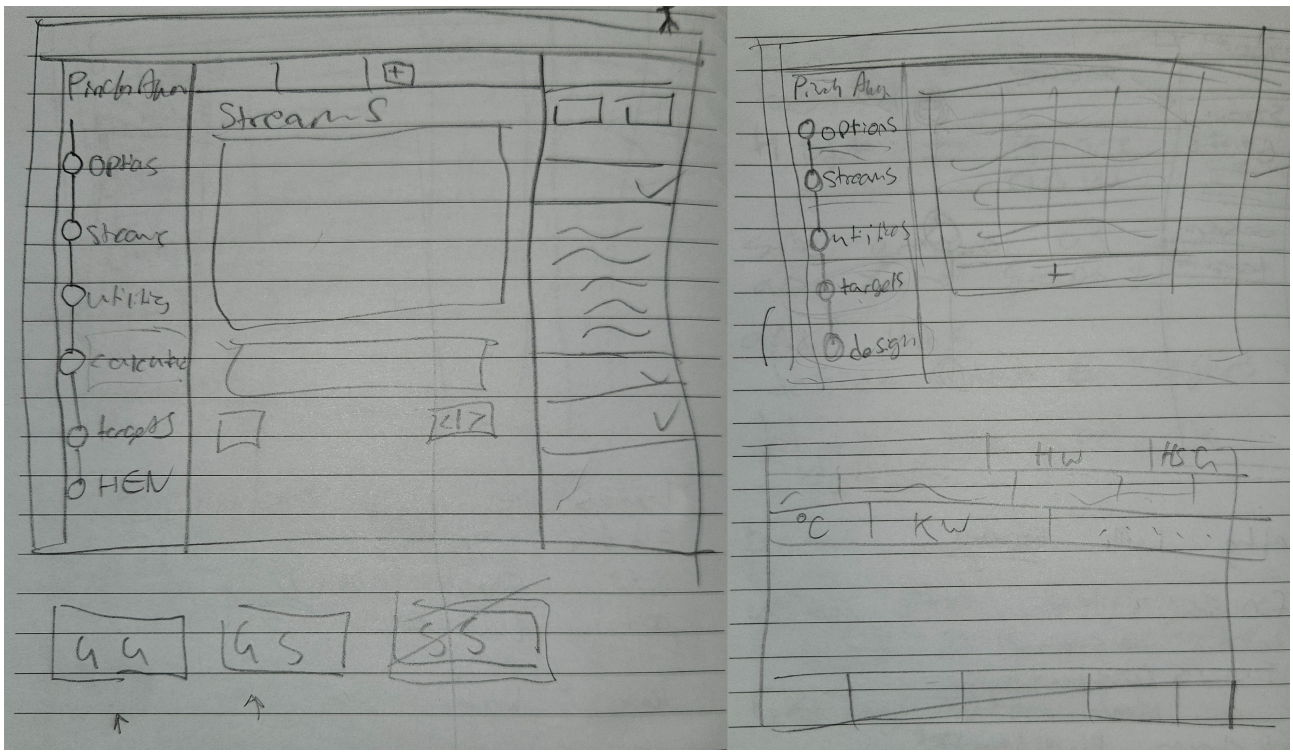

Figure 13: Wire-framing Examples

An important prototyping paradigm that was followed was leveraging Figma's component functionality where possible. This involves separating out individual components to design and change, then compiling all these components into a single design. The reason behind this was that it allows future changes to be done more easily, and removes many of the problems of making global changes to the prototype. Anything designed in Figma can be turned into a component, then instances of that component can be used in other sections or designs. Due to the fact that this was done previously in the larger Ahuora prototyping, the concurrent design changes that were done were immediately reflected on the designs that used them, saving a lot of time when matching the OpenPinch system with the greater platform. Figure 14 demonstrates how this was done for this project, where all the individual components are separated out to be used to build other components and designs. This system also matched well with how the wire-framing and testing was conducted.

At the end of this project, three major prototyping iterations have been completed, the third of which reflects the final implementation as seen on the front-end. Beyond these three major iterations, many smaller component level changes were done to polish and further the designs. This prototyping phase was invaluable to the development of the front-end, as changes to the front-end development would have been costly in development time, and the prototyping allowed the system to be matched to already tested and verified designs in a much faster and more efficient manner. Figure 15 demonstrates the second prototype that was completed for this project. Another key point that helped a lot when migrating the prototype designs to the actual system was that there was no need to check the technical feasibility of the system. Using the Shadcn toolkit for Figma allowed all the designs to be immediately implemented without many struggles developing the individual components.

One notable change that would be made if prototyping was undertaken again, is more time would be spent on wire-framing rather than prototyping. Despite this section being successful

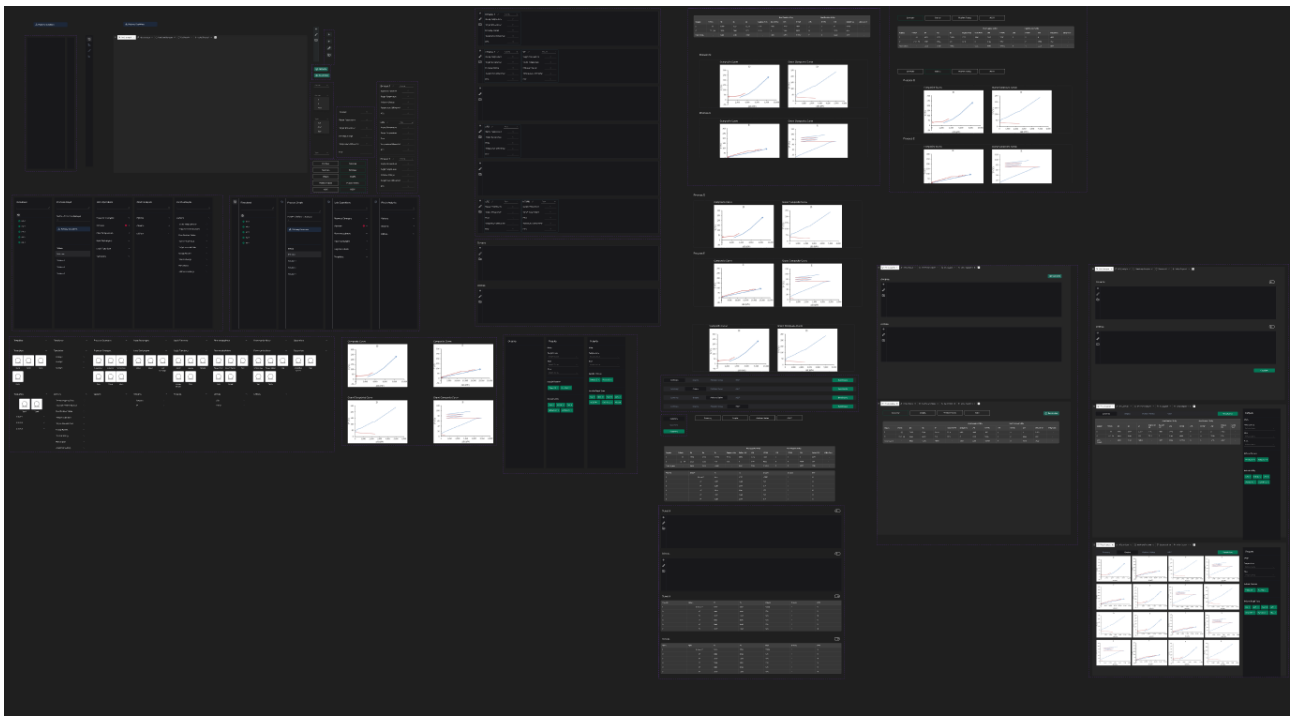

Figure 14: Prototyping Shared Components

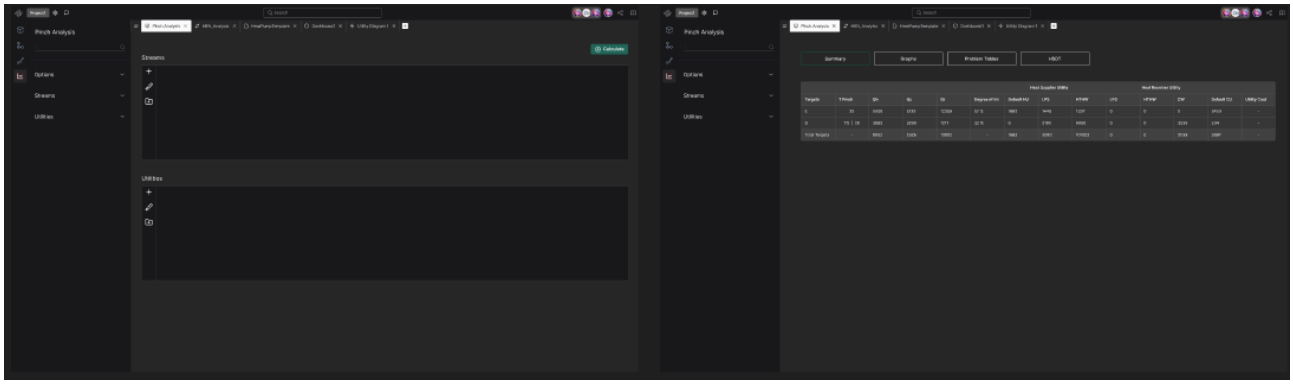

Figure 15: Prototype Iteration 2

overall, there was a fair amount of wasted time identifying issues with the prototype that could have been caught earlier with more intensive wire-frame testing and design.

6.3 Testing

In an effort to progress through the design work and validate designs, a series of tests were conducted for each unique section. This was in an effort to allow valuable user feedback to be incorporated into the designs from an early stage.

The first step for design testing was to apply for and receive an ethics approval from the Waikato University division of Health, Engineering, Computing, and Science (HECS). This step was crucial, and needed to be done before any of the testing was conducted to maintain impartial results and ethical conduct with participants. In this application was a preliminary outline of the methodology that would be used for testing, alongside a participation form. This form was given to every applicant and outlines the steps that have been taken to ensure

anonymity, the responsibilities of the conductor of the testing, and the rights of the interviewee.

An excerpt from the main body of the form is as follows:

Excerpt from Participant Information Sheet

Project Title
Designing the user interface of pinch analysis system to integrate into an existing application.

Purpose
This session of testing a digital prototype is conducted as a partial fulfillment of the requirements of an ENGEN582 capstone project. Low fidelity designs are required to be evaluated and refined as a part of the capstone project.

What is this research project about?
The overall aim of this research project is to use UI/UX testing techniques to evaluate the effectiveness of a digital prototype. This will involve exploring design and receiving feedback from participants to refine the final product.

What will you have to do and how long will it take?
The student will be providing you with a digital prototype and an environment where you will be able to interact with it. You will be asked to complete a set of instructions using the digital prototype after giving consent prior to the instructions, the student will be recording notes and observing how you navigate the system. After each minor task of the interview, you may be asked to give feedback and identify pain points that you encountered. After the interview is finished, you will be asked again to provide your consent to use the findings of the evaluation in a formal report. This process should take no more than 30 minutes. No personal or confidential information will be asked for or used during the interview.

What will happen to the information collected?
The information collected will be used by the student for their research and development project in ENGEN582-24X. This project involves presentations and a capstone portfolio. Only the researcher and supervisor will be privy to the notes, documents, and the paper written. Afterwards, this information will be destroyed, and erased. The researcher will keep transcriptions of the recordings and a copy of the paper but will treat them with the strictest confidentiality. No participants will be named in the publications and every effort will be made to disguise their identity.

Declaration to participants
If you take part in the study, you have the right to:

- Refuse to answer any particular question, and to withdraw from the study before 10th October 2024 when the report is due to be submitted.
- Ask any further questions about the study that occurred to you during your participation.
- Be given access to a summary of findings from the study when it is concluded.

This project was approved by HECS shortly after its submission without any major changes required to the proposal and methodology. Once this approval came through, testing began immediately, and all participants were given the participant information sheet to read and sign. In total, 5 individuals were tested, the background of these participants varied, with some coming from a chemical engineering background while others were purely software engineers and designers. This variety allowed for a broad range of feedback. In general it was found that design and software engineers provided more in-depth feedback on website specific areas, such as identifying where the layout or components were not usable, while chemical engineers focused more on functionality and the systems relationship with products and software they had used previously. Both of these categories of feedback were valuable, and including these groups positively affected the end product

The wire-frame testing followed a less rigid format compared to the other testing conducted in this project. Participants were asked to review the designs and identify pain points or areas needing improvement with minimal prompts or specific workflows. This approach was particularly useful for the early wire-frames, which were created to explore the system's initial concepts. By using this flexible format, a wider range of ideas emerged — ideas that would have been harder to implement as the designs became more refined due to requiring more refactoring of prototypes and user interfaces. This allowed for quick experimentation and testing of different concepts, which were later polished through more testing.

For prototype testing, interviewees were given a loose workflow to follow. This workflow was structured in a way for testers to be able to see every part of the system being tested - this could be working their way through the whole system, or simply interacting with every part of a specific component. The prompts and workflow were left vague on purpose, as a key goal for this testing was to check that the system was intuitive and usable. If many users got stuck on a prompt or took too long to navigate to a certain area, this was noted down as a bottleneck to productivity and the next design cycle would address the concerns. This also serves to indicate exact changes that should be made - for example if users always try to click a certain element to navigate, then that element should have the functionality to navigate. An example of this would be something like "add a stream" then the users would be left to figure out how to navigate to that screen and input the data.

In total, 7 weeks of wire-frame and prototyping testing iterations were conducted across the duration of this project. In the beginning, this designing and testing was done during the back-end development to gauge practical design from the work being done, then later the design and testing was more focused addressing the front-end testing feedback, as noted in Section 7.4.

6.4 Results

The design phase played a significant role in shaping the front-end development and the final structure of the system. The initial vision for the project underwent a large variety of transformations during the wire-framing and testing stages, highlighting the immense influence that design testing and iterative practices can have on a project. Notably, the integration of design patterns and iterative workflows, developed during the literature review phase streamlined the development cycle and heavily influenced the overall design approach and project direction.

A key challenge during the design and testing process was the need to analyze and prioritize feedback effectively. Testing often resulted in conflicting opinions and general comments, which required careful analysis and additional rounds of testing to validate. Some feedback, such as

suggestions on color schemes or component layouts, had to be disregarded entirely, as they did not align with the established branding and design consistency set by the Ahuora Digital Twin Platform.

A notable subset of feedback that was present in the prototype portion of testing that was less common in others, is that participants were more open to suggesting novel features to the system. During wire-frame testing most participants would comment on the general layout of the UI and component positions, and during usability testing the focus was placed more on the workflow of the system rather than missing features. One such example of this was the suggestion to add loading from CSV files that came through during prototype testing.

An area that would be explored if this project were to be undertaken again is heuristics testing. This was an area that was noted in the literature review conducted (see Section 3.1) and would have improved the final design substantially - particularly near the end of development. Heuristics testing is a usability evaluation method in which a set of design principles, or "heuristics," are used to identify usability issues within a user interface. Typically performed by usability experts and designers, the evaluation is focused on how well the user interface adheres to these predefined principles. Incorporating the results from this testing would have brought an alternative perspective to the design work, and could have identified even more issues. Due to time and personnel constraints, heuristics testing wasn't viable for this project, as the usability experts weren't available during the time to perform this evaluation.

7 Front-end Implementation

7.1 Caching and Hooks

As previously mentioned, the Ahuora Platform that the Pinch Analysis system is being integrated into leverages a caching process through the Redux toolkit - RTK Query. In the context of web applications, caching allows data fetched from an API to be stored so that subsequent requests for the same data can be served faster, without having to re-perform the computation or fetch from the server again. The specific motivation behind using this technology in this project is twofold. Firstly, having a central cache to retrieve data from abstracts out some of the finer state management issues that can be found in a large web applications - especially in the context of React. Instead of passing values and state across components and sections of the platform which may lead to inconsistencies and a heavy performance toll as well as contributing to difficulties when maintaining or modifying code, each component can retrieve the required data individually, as well as make changes to the cache through manual optimistic updates and re-fetch requests.

The second reason has a lot to do with the general structure of the platform. In an effort to make the front-end "dumb" - reducing computation - the front-end has been structured to perform as few logical operations as possible. Where this can best be seen is how the graphic object positioning on a flowsheet is stored and read from the database. When a flowsheet is loaded, these values are read from the database and stored in the cache. When a unit operation is moved on the front-end the values in the cache are first optimistically updated, then a request is sent to the back-end to update it. This allows the values to be stored on the back-end while also achieving immediate visual updates with the cache. When the request to update the database has been fulfilled, the cache will quietly update the cache values with those returned from the request - which theoretically should be the same values.

All the data and functionality for the Pinch Analysis front-end are stored as database objects on the back-end (see Section 5.1.1). This approach enables direct management of functionality, inputs, and outputs via cache manipulation. Static features that do not strictly require user interaction — such as the pinch options — are created when the flowsheet is initialized. This setup allows the front-end to request these objects from the back-end, and any future updates can modify the values of these objects rather than creating new ones.

Dynamic inputs, such as pinch streams and utilities, are fetched as a set of objects to be displayed. When these objects are added, they are first optimistically created in the cache so that they are immediately visible to the user. Then, a request is sent to the API with the input data, and Django creates the corresponding database entry. This methodology ensures a seamless user experience, with data changes being reflected instantly while maintaining consistent data entries in the back-end database.

This approach leads to several advantages from a software reliability standpoint. By leveraging caching for immediate user feedback, the platform ensures that interactions remain fluid, even as data is processed in the background. At the same time, the integrity of the database is upheld, as the cache serves as a temporary layer before each read/write operation - checked against the serializers defined with DRF. Finally, the cache invalidation strategies mentioned earlier help to keep the cache synchronized with the underlying data, avoiding potential issues with outdated or stale information being presented to the user.

Sending requests to the API is also handle by RTK Query through the use of some auto

generated hooks. The complexity of making requests is mostly abstracted by RTK Query, as hooks are auto-generated based on the the defined serializers and their associated SwaggerUI schemas (see Section 5.1.2). Some specific example of these hooks for this project are:

Hook	**HTTP Request Header**	**Database Object**
usePinchinputRetrieveQuery	GET	Pinch Stream / Utility Parent Object
usePinchPinchstreamPartialUpdateMutation	PATCH	Pinch Stream
usePinchPinchstreamCreateMutation	POST	Pinch Stream
usePinchPinchstreamDestroyMutation	DELETE	Pinch Stream

Table 3: Cache Hooks to Request Mapping

For example when updating a pinch stream, the usePinchPinchstreamPartialUpdateMutation hook can be called by passing in the stream's ID along with the new data and a PATCH request will be sent. The real utility of these hooks is their error handling and functionality. The RTK Query hooks allow for some variables to be accessed directly, namely isError and isLoading. As their names suggest, these are status variables that can be accessed to determine the success of the request. For this project, isLoading was used to display a spinner when loading to indicate to the user that a request is being processed - the nature of the cache means that this would only happen the first time a request was being made however this was a good feature from a usability perspective. The isError variable allowed for simple error handling without redundant boilerplate code across components.

Another feature that was utilized was the skip parameter passed in to fetch (GET) requests. The skip parameter allows for the request to be conditionally made based on a function or condition. This was primarily used as a error prevention technique, as sometimes there were chained requests being made: requests that depend on the output of other requests. Should any of these requests fail, many of the downstream components will then in turn throw exceptions or produce faulty results. If a required variable isn't present or incorrect, the hook will instead account for this via the skip parameter, skipping these requests and retrying upon the next re-render.

As mentioned previously, the RTK Query configuration had to be amended to use the cache efficiently. The most important of these was the cache invalidation. RTK Query has a system that allows for automatic re-fetching of data based on events that would indicate that the data stored in the cache is stale (outdated). These options need to be manually set based on the generated hooks from SwaggerUI. The process for doing this is to first set a marker for different data structures - for example a pinch stream - then define which hooks will cause data to become stale, and which hooks will re-fetch that specific data. In the example of a pinch stream the `PinchstreamCreate` hook will validate the data, however any of `PinchstreamPartialUpdate`, `PinchstreamDestroy`, and others will invalidate the cache. So in the case where the `PinchstreamPartialUpdate` hook is called, the `PinchstreamCreate` hook will be called automatically afterwards. A visual representation of this can be seen in Figure 16.

Using the cache and these hooks greatly streamlined development. A key goal for this project was to ensure maintainability and readability for future work. Employing these practices made a valuable contribution toward meeting those objectives and enhanced the overall system by adhering to software best practices.

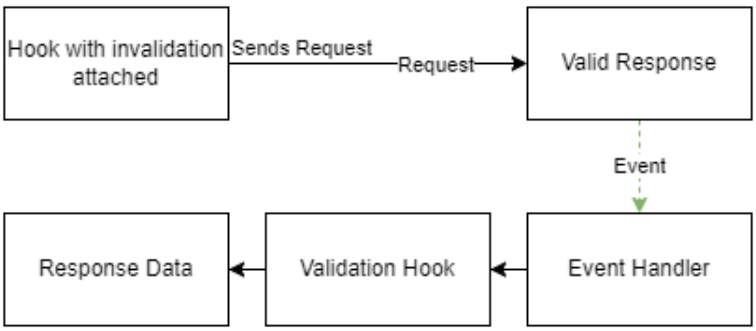

Figure 16: Cache Invalidation Steps

7.2 User Interface

After the prototype was at a point where it was fully functional and no further tests were required, development was begun on the front-end of the system. Having a usable user interface for the Pinch Analysis module was a key outcome to achieve for this project, as this was one of the features that was notably lacking in other Pinch Analysis tools. In order to provide actionable insights for users to inform their processes, the user interface was required to be clear and easy to use - a criteria that was filled by the prototyping and wire-framing stages of design.

The Ahuora Digital Twin Platform already had a front-end from prior development, and this project's front-end work was built on the same code-base. Since the Pinch Analysis system was to be integrated into this platform, it was able to be added seamlessly into the existing framework. The system was logically organized into its own file structure within the Ahuora code-base, ensuring it fit within the platform's architecture. It was further divided into sub-directories based on hierarchy, with separate folders for components, table column definitions, and other key elements to maintain a clean readable structure.

The initial phase of development involved making adjustments to the existing system to incorporate access to the new Pinch Analysis module. The Ahuora platform operates as a single-page application (SPA), which utilizes toolbars and navigation elements to allow users to access various functionalities. Icons and navigation paths needed to be added to these areas to provide seamless access to the pinch analysis tools. Thanks to the SPA architecture, global components like the navigation bar, footer, and key features were easily managed and updated across the separate modules.

Once these integration points were addressed, focus shifted to developing the main content area for the Pinch Analysis system. Some of the main areas developed were:

- Options - Based on the `PinchOptions` data structures, this component allows the user to manage which OpenPinch modules are executed at calculation time. For example exergy, area and cost targeting, turbine options etc.
- Stream and Utility Inputs - Manages the creation and editing of the inputs (`PinchStream` / `PinchUtility`) via a custom "Excel-like" editable table.
- Target Outputs - Table containing all the targeting outputs for the system. Made using Shadcn and TanStack.
- Graph Outputs - Discussed further in Section 7.3

This development portion leveraged several libraries ad technologies in its implementation. Included are:

Technology	Description
React	Javascript library for developing front-end components.
TypeScript	programming language that provides strong typing for Javascript.
Shadcn	React component library for using pre-built items.
TanStack	React library that provides a complex table implementation.
Recharts	React library for building graphs and charts.
Tailwindcss	Open source CSS framework for styling.

Table 4: Technologies Used During Front-End Development

While not a strict criteria for this project, the user interface was designed with relative sizes and positioning in mind for future development of a responsive UI. This involved structuring components with flexible containers and component sizes that would translate effectively to a mobile or smaller screen view. Scroll-able containers and objects were also utilized for the sake sake of readability when considering projects with a large or variable amount of data, especially in the case of the input tables.

The final client-side user interface was developed effectively according to the prototyping designs. All major functionality required for the final product were implemented successfully and to a high-quality standard. This achievement not only enhances the user experience by providing a clear and intuitive interface but also establishes a solid foundation for future developments. By prioritizing usability and integration with the Ahuora Digital Twin Platform, the Pinch Analysis module manages to resolve some of the key limitations found in other software.

7.3 Graph Outputs

The graphs that are produced by the Pinch Analysis system on the back-end are a crucial component that is used to inform decisions. As such, making sure the graphs are correct and easily digestible was a key goal for this project.

While developing this project, the original goal was to use a library called D3 to develop the graphs. This decision was made as a part of the literature review (Section 3.1) where a technology review was conducted on some popular react graphing libraries. Soon after development started however, a better solution presented itself. Shadcn - the component library that was used for the Ahuora Platform - contains the functionality for creating graphs. These graph components were built on top of Recharts, a JavaScript library made using D3 that abstracts away a lot of the complexity. So while the project did still technically use D3, none of the boilerplate was required when building the graphs.

As mentioned, the graphs on the front-end have been built with Shadcn/Recharts. This approach simplified the initial set up immensely. To display the graphs to the user, the targeting calculations must first be done and the graph structures must be present in the RTK Query cache. From this, the appropriate data can be extracted from the cache and mapped to individual Shadcn `Graph` objects that have the required customization - these have been called `PinchGraphs`. A `PinchGraph` is a superstructure of Shadcn's `Graph` object that has been modified to accept the back-end `PinchGraph` data structure. Alongside the default implementation of a `Graph`, the `PinchGraph` has functionality that includes:

- Custom Domain Handling - Custom x/y domains are generated from the provided data that provides a better implementation for a linear domain than what Recharts currently uses.
- Custom Tool-tip - Recharts usually displays all the x/y values and each line name when the graph is hovered over. This data does not provide any value in this use-case and has been changed to only display the current x/y value that is being hovered over.
- Scale/Size - In consideration for future development and changes, all graphs are assigned a uniform scale at the `PinchGraph` level that allows a developer to change the scale from a central location
- Integration with Data Structures - The graphs being displayed have some additional requirements for colour and arrows that have been created when the data is passed in to the `PinchGraph` objects.

Once these graphs have been successfully created, they are displayed to the user on the appropriate page, Figure 17 is an example of how these graphs look with a simple four stream problem.

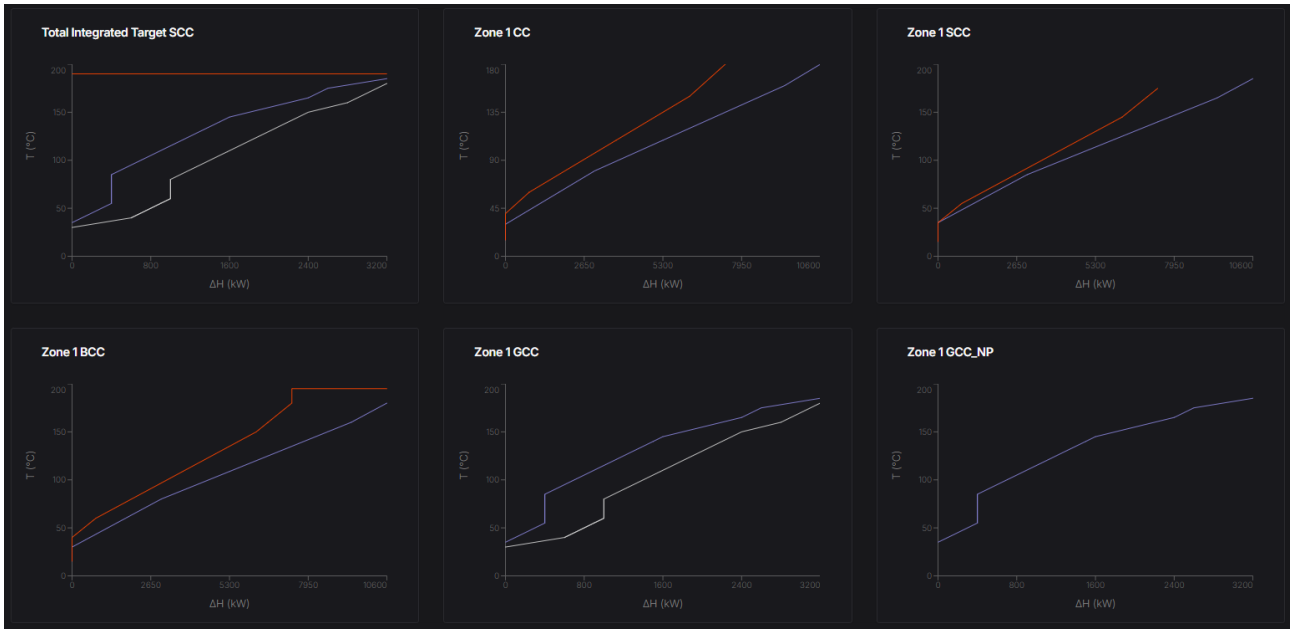

Figure 17: Four Stream Problem Graph Outputs

One of the major difficulties encountered while developing the graphs was the lack of overall functionality of the graphs offered by Recharts. While Recharts is convenient to set up graphs quickly, the major use case for these graphs is for data visualization - particularly for bar charts and others. This means that most of the features were catered towards these types of graphs, and the line charts that were offered aren't typically supposed to be used for accurate scientific line graphs. This problem led to a lot more development than was initially estimated, primarily through adding custom functionality that was required of the pinch graphs. One example that highlights this point is a custom function was required for displaying a properly scaled domain - by default Recharts displays the x/y domains by taking the value of a point at some interval. While this is useful in some cases, a linear domain calculated from zero to the maximum plotted value provides more valuable information to users that inform decisions. These were multiple areas that required custom fixes like this one.

As a general reflection on the current state of the graphing outputs, they are being displayed correctly according to the data that they are given. While there are some minor formatting issues that need to be addressed, these graphs can be used to infer meaningful conclusions from the prior targeting step. The next steps to take for the development of the graphs in the future will be to migrate the Recharts library to D3 instead. This will fix some of the performance issues that have been encountered when displaying many (more than 20) graphs. Alongside this, by switching to D3 some of the optional functionality that was indicated in user testing can be more easily developed.

7.4 Testing

Many testing phases were conducted on at the prototyping stage of this project, however there exists key metrics that cannot be accounted for during the prototyping stage. This is primarily due to the fact that while the prototype "fakes" the functionality of the system, the real-world system can produce unexpected results and requirements can change as users use the real thing.

The motivation for conducting this testing was twofold: firstly, to validate the design findings in a practical, real-world environment, ensuring that the interface meets the expected usability standards; and secondly, to further test the system's functionality, particularly in terms of its integration with the back-end and the overall performance of the front-end components. Similarly to the prototype testing, a series of usability tests were conducted on the front-end design over the course of development in the same format. Interviewee's were given access to the system and a series of prompts were drafted to encourage the user to progress through the system. Notes were be taken for how the user would interact with the UI, as well as the answers to questions asked by the interviewer at key milestones.

The results from these usability tests helped inform the prototype, and in turn the front-end development. Many points of feedback that weren't identified during the prototype testing phases were encountered during this stage of the project, the most notable of which were concerning integration with the larger system.

In stark contrast to the prototype testing, rarely were interviewees asked to specifically test only a single component on the front-end. This was primarily due to the large effects component changes can have to the system. The initial phase of the user navigating throughout the user interface to get to the component being testing often provided valuable feedback to how the singular component effects the system as a whole.

8 Results

The final product that has been developed throughout the course of this project has successfully managed to incorporate the several software practices that were outlined by the requirements and literature review.

For the logic implementation, the system that was previously reliant on outdated and slow technologies has been migrated to use modern technologies instead. The system now exists as a service on a maintainable and scalable distributed system that can be managed and developed more easily in the future. The many prevalent issues that the Pinch Analysis system already had have been effectively addressed through correct error handling and bug fixes, resulting in a more correct system to inform users.

The Pinch Analysis system now also contains a usable user interface, informed by extensive user testing and feedback, as seen in Figure 18. The future UI development that will be conducted for the Ahuora Platform is able to leverage the modular components and structures created through the development of this UI, facilitating fast and uniform future functionality. The user testing that was conducted for this project has resulted in smooth and predictable navigation, with an emphasis on what is expected from similar software in industry. Components are intuitive to use and contain all of the functionality that is required to properly leverage the suggestions made by this system. Overall, this UI is far superior to what is common in the industry.

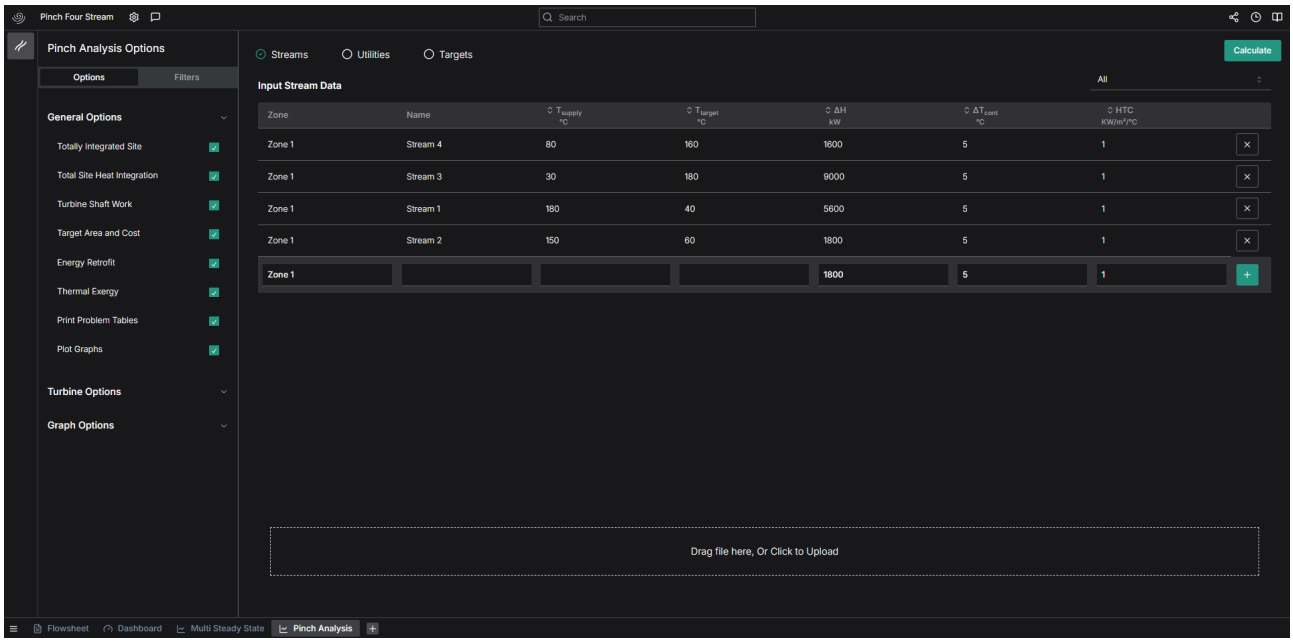

Figure 18: Final User Interface Iteration

9 Discussion

The development of this module has significant implications for the context it exists within. It meaningfully extends the foundational work it is based on and will continue to develop alongside these advancing technologies. Additionally, there are important relationships between this project's completion and its broader impact on the community. However, to fully understand its potential and impact, there is a need to clearly define the specific contexts in which this project is most effective.

9.1 Relationship to Previous Work

As mentioned in Section 5, a potion of the development of this project was spent on updating some of the lacking or inoperable modules contained in the Python OpenPinch system. Significantly, errors in calculation routines were addressed. Data structures and visualisation functionalities were entirely overhauled and implemented as separate components within the new Pinch Analysis module. In addition to this, many sections of redundant or inefficient code was removed entirely - making the system as a whole more performant.

As research and progress is made in the Pinch Analysis and process optimization field, there is a distinct need for these principles to be integrated into existing software and technologies. Within the context of OpenPinch, the system that has been developed for this project fills the role that was previously held by Python OpenPinch - that being a Python Pinch Analysis module. As Pinch Analysis and process optimization develops, new functionality and updates will most likely be done to this system in place of the Python OpenPinch module. There is no evidence that this would be the same for the Excel OpenPinch system however, as there is no intention to lock users into using the Ahuora Digital Twin system to have access to the latest updates and features, despite the advantages this system has over the Excel version.

In an addition to this, the Pinch Analysis module was the first of many planned process optimization techniques to be incorporated into the Ahuora Digital Twin platform. Having this module in the Ahuora Platform raises its value to industry partners and contributes to the final goal set out by Ahuora for its functionality. In addition to this, having the Pinch Analysis module as a service on the Ahuora Platform opens up the possibilities of integrating results with existing and planned internal functionalities - facilitating multi-faceted optimization.

9.2 Limitations

As is always the case with software that models real-world processes, the outputs can never be assured to be 100% accurate. This system relies heavily on input data that is often subject to variability, measurement errors, or incomplete information. Any small inaccuracies in the data, such as incorrect stream and utility data, can lead to incorrect recommendations to the user. This reliance on data integrity is a key limitation, as any inaccuracies in the input data can cause large variability in output results. In an effort to combat these inaccuracies, steps are currently being taken to in corporate live data into the Ahuora Digital Twin Platform that can ideally remove some of these discrepancies (Section 2).

In an effort to make the system more usable and abstract away some of the finer details required for Pinch Analysis, certain assumptions are made. These assumptions aim to simplify the user experience by reducing the complexity involved in setting up and running analyses, while still maintaining the accuracy and integrity of the results. For instance, default values for common

parameters such as heat capacity flow rates have been initialized based on industry standards. This allows users to focus on key inputs specific to their processes without needing an in-depth understanding of every technical aspect. Similarly to input errors however, these assumptions can result in tangible differences between the modelled system and real-world processes.

In addition to this, when programmatically performing Pinch Analysis and using the Problem Table algorithm, the system is subject to the inaccuracies that are often found using programming languages such as Python. One such common issue is related to floating point arithmetic, where minor precision errors can accumulate, potentially affecting the accuracy of heat balance calculations. These inaccuracies arise due to the way floating point numbers are represented in computer memory, which can result in rounding errors during arithmetic operations. In a process as sensitive as Pinch Analysis, where precise temperature and heat flow calculations are essential for targeting energy efficiencies, these small errors can lead to incorrect results. In an effort to combat these issues, a larger number of decimal places for floating point numbers has been used for this system, however these issues will still be present - as is the nature of the system. From extensive testing between the Excel OpenPinch system and manual Pinch Analysis calculations, the outputs from this module slightly differ due to this.

The nature of this system is to inform decision-making rather than to implement changes directly. As such, the responsibility lies with the user to interpret the provided data and make informed decisions regarding their processes. This intentional design choice ensures that the system serves as an advisory tool, offering valuable insights and suggested changes without exerting direct control over process modifications. As such there is a strict limitation for the benefits of this system placed on the engineer using it, as assumptions are made for the competence of the user.

9.3 Impacts

9.3.1 Environmental

The primary goal of this project was to reduce GHG emissions in the process heat generation sector. The system developed achieves this by optimizing energy use and minimizing waste through the application of pinch analysis. By identifying inefficiencies in heat processes, the system enables more efficient heat recovery, reducing the need for additional energy input and decreasing dependence on non-renewable energy sources. As a result, the system directly lowers GHG emissions in industrial facilities by reducing both overall energy consumption and emissions associated with process heat generation. In addition to this, the system can help companies align further with New Zealand's climate goals through the aforementioned emission reductions.

9.3.2 Economical

While not the direct goal behind this project, many companies and industrial plants may find the economic benefits of Pinch Analysis even more compelling than the environmental considerations. By conserving energy through optimized heat exchange, plants can significantly reduce their energy consumption and operational costs. Although the initial investment in heat exchangers and other necessary infrastructure may be costly, for many companies, this investment leads to long-term cost savings that far outweigh the upfront costs.

9.3.3 Social

Through the reduction of GHG emissions, many positive effects can be seen on the community. Improving air quality has a consistent positive impact on a communities health and well being. As well as this, adoption and publication of a company using energy-saving techniques like those provided by this system can help improve their public perception within the community. By showcasing their commitment to sustainability and responsible energy use, the company can position itself as an environmental leader, gaining positive recognition from both local residents and wider audiences. This can enhance the company's reputation, fostering goodwill and trust among stakeholders, customers, and regulators.

9.3.4 Technological

As discussed in Section 3.1, the current state of process optimization tools, and more specifically Pinch Analysis tools, is far from what modern technology allows for. Through the integration with the Ahuora Digital Twin Platform, the diving factor was to bring this technology to a scalable, intuitive, and modern platform. The hope behind this choice is to elevate the standard for process engineering tools to a new height, instead of the efficient and outdated tools common in industry. Raising the standard of what's expected from these tools can allow industries to achieve greater efficiency, sustainability, and operational insight and force development in a new direction.

10 Conclusion

This dissertation has explored optimizing energy usage in the industrial process heat sector in New Zealand, particularly in the context of the country's commitment to achieving net-zero greenhouse gas emissions by 2050. Through the development of an innovative Pinch Analysis module integrated into the Ahuora Digital Twin Platform, this project addresses the tangible need for effective tools that facilitates energy efficiency and reduces carbon emissions.

The development work in this project has significantly enhanced the existing Python OpenPinch system though the application of modern error handling and optimizing data flow. Additionally, it has been seamlessly integrated into the back-end of the Ahuora Digital Twin Platform, establishing a solid foundation for future development of process optimization tools. Throughout this project, various bugs were resolved, and superior software practices were employed, ensuring a more scalable and maintainable system.

The extensive design and prototyping iterations have effectively addressed the usability and user-centered design issues commonly found in tools within this industry, while also directing the development. From these designs, a user interface was created that not only validated the findings from the design phase but also provided future users with a comfortable and accessible visual medium to interact with the system.

In summary, this project not only lays the groundwork for improved energy optimization in the industrial process heat sector but also demonstrates the importance of innovative software solutions for sustainability initiatives, particularly within the field of process engineering.

10.1 Contributions

This project makes several contributions to both the industrial sector and the context of the greater Ahuora Platform:

1. Integration of Pinch Analysis with Digital Twin Technology: The successful integration of the Pinch Analysis module into the Ahuora Digital Twin Platform is a major contribution. By enabling real-time optimization of heat recovery in industrial processes, the project advances the practical application of pinch analysis and brings this process optimization tool into a modern, scalable digital environment.
2. Reduction of GHG Emissions: A core goal of the project was to contribute to the reduction of greenhouse gas emissions. By optimizing energy use and facilitating heat recovery in industrial processes, this system directly addresses environmental concerns and supports the global effort to combat climate change.
3. Improved User Interface Design for Process Optimization: Another important contribution is furthering the standards of software tools in the process engineering sector. This system demonstrates the feasibility and necessity of progress towards standardized usable and maintainable systems that are otherwise lacking in this sector.

Overall, this project lays a foundation for future development in the Ahuora Digital Twin Platform, progress towards reducing GHG emissions for industrial heat generation, and provides an exemplar for improving software tools in the process engineering sector.

10.2 Future Work

It is difficult to claim that the Pinch Analysis system is currently completely integrated with the Ahuora Digital Twin Platform due to the fact that it has no interactions with the rest of the system. One of the key areas for future development that was identified was a feature for stream data extraction - an area that would allow complex interactions between the process simulation side of the software and the analysis side. As the system currently stands, users enter the stream data through the interface, however this data needs to be found and compiled manually. This leads to redundant work, where users will be modelling their systems through the process simulator, then also adding the stream data to the Pinch Analysis module. Ideally, there should be an option for the system to automatically extract the meaningful data from the process simulator to be used for Pinch Analysis instead. There are several complex processes involved to develop this system, which is one of the primarily reasons it wasn't completed as a requirement for this project - time constraints being the other reason.

As mentioned in Section 3.3, the Python OpenPinch module is based on OpenPinch v2.30, while the most current version of OpenPinch is v3.12. This means that, although some aspects have been updated to the latest version during development, there are some features that have not yet been migrated to the most recent version of Python OpenPinch / this project. While not an immediate requirement, this functionality is ideal for the final system so emphasis should be placed on starting this migration, incorporating the new features, and implementing the fixes that don't yet exist on Python OpenPinch.

Other features for the Ahuora Digital Twin Platform that is directly related to the work done in this project is OpenHEN. OpenHEN is a module developed by Keegan Hall that leverages Python OpenPinch to generate a Heat Exchanger Network (HEN) Diagram. HEN diagrams are used to inform the design of heat exchange systems, and can be considered the next step from the work done with Pinch Analysis. The next steps in this direction will be to integrate this OpenHEN work into the Ahuora Digital Twin Platform, and make changes to the system to fit in with the existing architecture. One current limitation to this goal is a lack of a queuing system for services on the back-end, another part of the future work required.

As is the case with most systems with a user interface, there is always a need for more development and more testing. As the system and the larger application changes, the UI for this project needs to evolve alongside it, and this is informed through more user testing and feedback. Another point to mention is that the Ahuora Digital Twin Platform is still currently in its beta phase. When real-world users outside the scope of the testing group are exposed to this system, there will most likely be feedback that hasn't been given throughout this project, requiring more development.

References

[1] Ministry for the Environment. *New Zealand's projected greenhouse gas emissions to 2050*. Dec. 2023. URL: https://environment.govt.nz/facts-and-science/climate-change/new-zealands-projected-greenhouse-gas-emissions-to-2050/ (visited on 09/18/2024).

[2] MBIE. *Decarbonising process heat*. Dec. 2022. URL: https://www.mbie.govt.nz/building-and-energy/energy-and-natural-resources/low-emissions-economy/decarbonising-process-heat (visited on 09/18/2024).

[3] PHiNZ. *Process Heat – Overview*. Aug. 2018. URL: https://www.mbie.govt.nz/dmsdocument/152-process-heat-current-state-fact-sheet-pdf (visited on 09/18/2024).

[4] Ministry for the Environment. *National Policy Statement for Greenhouse Gas Emissions from Industrial Process Heat 2023*. June 2023. URL: https://environment.govt.nz/publications/national-policy-statement-for-greenhouse-gas-emissions-from-industrial-process-heat-2023/#:~:text=The%20National%20Policy%20Statement%20for,of%20heat%20for%20industrial%20processes. (visited on 09/18/2024).

[5] Andrew Lee et al. "The IDAES process modeling framework and model library—Flexibility for process simulation and optimization". In: *Journal of advanced manufacturing and processing* 3.3 (2021). Publisher: Wiley Online Library, e10095.

[6] Michael L. Bynum et al. *Pyomo - Optimization Modeling in Python, 3rd Edition*. Springer, 2021.

[7] Dylan Field. *Figma*. 2012. URL: https://www.figma.com/ (visited on 10/07/2024).

[8] Adrian Holovaty. *Django*. May 2024. URL: https://www.djangoproject.com/ (visited on 09/29/2024).

[9] Tom Christie. *Django REST Framework*. June 2024. URL: https://www.django-rest-framework.org/ (visited on 09/25/2024).

[10] *Shadcn*. URL: https://ui.shadcn.com/ (visited on 10/07/2024).

[11] Peitro Schirano. *@shadcn/ui - Design System*. 2022. URL: https://www.figma.com/community/file/1203061493325953101 (visited on 10/07/2024).

[12] Adam Wathan. *Tailwindcss*. 2017. URL: https://tailwindcss.com/ (visited on 10/07/2024).

11 Appendix

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Ahuora Adaptive Digital Twin Platform

Ethan MacLeod

Supervised By:
Tim Walmsley, Mark Apperley

Contents

List of Figures

List of Tables

1 Introduction

New Zealand has committed to achieving net zero Greenhouse Gas (GHG) emissions by the year 2050 through their Carbon Change Response Amendment Act 2019 [8]. One of the largest roadblocks to this eventual goal is the industrial process heat sector, which by itself contributes to 28% of all of New Zealand's energy related GHG emissions [9]. Limiting the amount of process heat that needs to be generated will contribute significantly to net zero GHG emissions, while also reducing the money required to generate this heat. One approach to limiting process heat generation is applying pinch analysis technique to industrial process engineering plants across New Zealand. Pinch analysis is a technique that limits the heat required for a system by finding points in which excess heat can be recycled back through the system to heat other processes - effectively reducing the amount of heat that needs to be applied. Unfortunately the current digital tools that exist fail to provide an ideal user experience, and this has led to a low adoption rate across the industry for applying pinch analysis. The aim of this literature review is to identify the shortcomings of these technologies and find appropriate methodologies and principles that would lead to a better user experience overall, in the hopes that following these principles will increase adoption rates.

Some key research questions that will be address in this review are:

1. How are the current digital pinch tools lacking?
2. Which design techniques and processes should be considered when making a pinch tool?
3. What software design principles and practices should be considered when making a pinch tool?

Research into existing pinch tools are included in Section 2 - pinch analysis. This section is a review on the history of graphical representations for process optimization and how these tools are applied to current digital pinch solutions. The most popular digital pinch analysis tools will be scrutinized primarily on their User Interface (UI) and the nuances between design and retrofit in regards to pinch analysis will be discussed. Section 3 - Design focuses on extracting key UI design paradigms and their application to pinch tools. Design principles, ideal design processes, and methods to evaluate the efficacy of designs will be researched and documented in contrast to weaknesses found in existing tools. The software design section - Section 4 - will explore the best practices to consider when designing complex software. Included in this is discussion of software architecture, general software design principles, and a technology review of some applicable graphing tools to process engineering.

One notable exclusion from this document is the principles and processes behind pinch analysis will not be discussed in-depth. The focus of this review is existing pinch tools and ways to improve them, so an in depth discussion of pinch analysis and how it works is out of scope for this document.

2 Pinch Analysis

Pinch analysis is a methodology used in process engineering to optimize energy efficiency within a system [10]. Pinch analysis does this by determining the 'pinch point' - or point of minimum energy efficiency - and using this to inform the placement and interactions between heat exchangers. This section of the report focuses on the development and critique of existing pinch analysis digital tools and graphical representations, as well as their role within process engineering.

2.1 Historical development of graphical tools

Graphical tools have played a significant role in the evolution of process engineering, particularly in enhancing decision-making when employing pinch techniques. Among these tools, several stand out due to their utility and impact:

The Heat Exchanger Network (HEN) diagram is a visual representation of how heat is being exchanged within a system, and includes heat exchangers, streams, and utility heating/cooling inside it [11]. HEN diagrams are used extensively with heat optimization techniques such as pinch analysis, as it provides a clear overview of the heating within a system. Fig. 1 demonstrates the typical layout of a HEN diagram - where the hot and cold streams are shown as the red and blue lines respectively, the heat exchanger of the network are the yellow utilities (Named E(X)) and the heating and cooling utilities are the blue and red utilities (C and H).

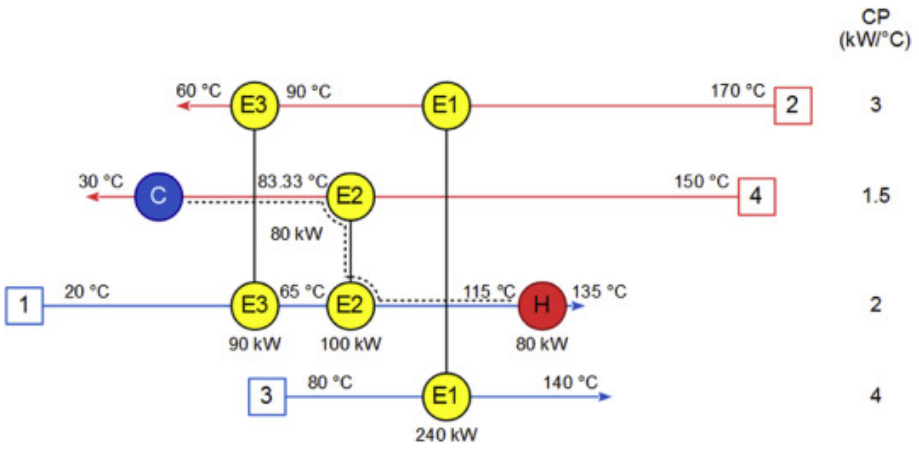

Figure 1: Heat Exchanger Network Diagram [1]

Composite curves, also know as T-H diagrams are used to visualize hot and cold streams on the same graph and also the heat transfer potential between them [12]. A composite curve is used in pinch analysis to identify the pinch point of a system visually, and are instrumental to obtaining the most energy efficient configuration for a HEN - identifying where the minimum temperature difference occurs between streams is a common tactic for targeting with pinch analysis.

Similarly to composite curves, grand composite curves are also used heavily in pinch analysis techniques. A grand composite curve is the graphical representation for how heat flows over temperature intervals - plotted by heating 'duty'. Grand composite curves generally give more information about a system than composite curves and are used to identify heat recovery opportunities within a system.

A Mollier diagram, or Enthalpy - Entropy diagram, was one of the first graphical tools used to represent thermodynamics, more specifically the functional relationship between enthalpy, en-

tropy, temperature, pressure and quality of steam [13]. Adopted in the early 1900s, the Mollier diagram has many applications in process engineering, especially when finding an unknown value in the previously mentioned relationship.

P-graph was another tool developed in the late 1970s as a CAD generation of process structures [14] that was further developed until the main framework for P-graph was finally finished in the 1990s [15]. P-graph uses graphical representation of process networks to optimize and evaluate process structures - and provides optimal alternatives.

Currently there are many tools for visualizing process engineering due to the increase in processing power of modern machines. Programs that provide dynamic simulation and real time data integration are common - such as Aspen Plus, Aspen HYSYS, and DWSIM. These programs can provide outputs utilizing some of the previously mentioned processes while also demonstrating a holistic representation of the entire system.

2.2 Digital Pinch Tools

Some digital tools and libraries have been specifically made integrating pinch analysis techniques. The most popular of these include; SuperTarget [16], OpenPinch [17], and Aspen Energy Analyzer [18]. While these tools address the lack of digital pinch modules, most fall short in terms of their design and usability. This section will be an overview of these tools in terms of their design and functionality.

SuperTarget is a module designed for retrofitting oil refineries and is integrated with the Petro-Sim process simulator. Although it offers high compatibility, allowing users to import files directly from popular process simulators like Aspen HYSYS, Aspen Plus, Petro-Sim, and PRO/II, SuperTarget is often regarded as less relevant in modern settings. This is primarily due to its user interface, which can be considered less intuitive compared to newer tools. Fig. 2 shows the HEN diagram output from SuperTarget 7; the format of the UI is outdated in its design and doesn't provide the breadth of tools needed for interacting with the graph. Despite this, SuperTarget interfacing with other programs allows it to be used without a redesign of the system within the application, meaning it has lots of utility to be used with process simulators that don't have a native pinch module.

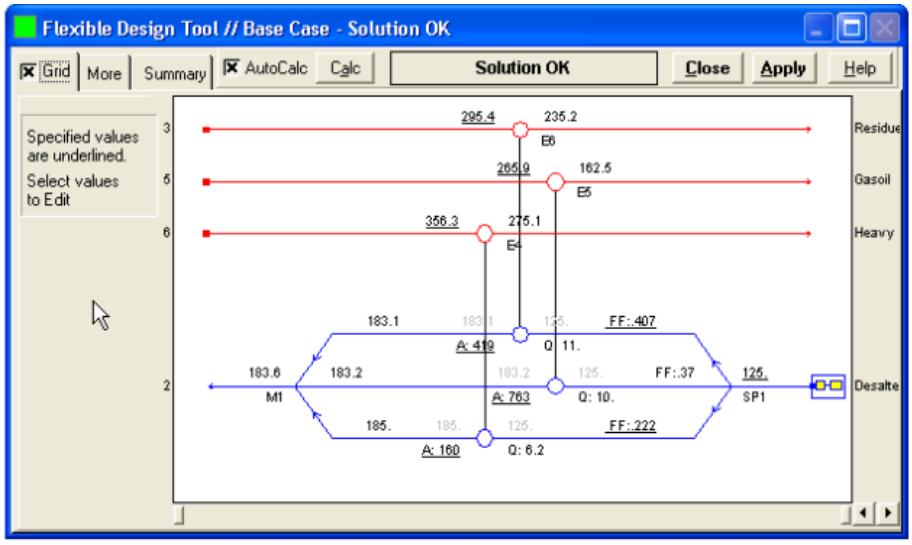

Figure 2: SuperTarget 7 HEN Diagram [2]

OpenPinch is a pinch-based HEN optimizer built on top of Microsoft Excel spreadsheets. Since OpenPinch uses Microsoft Excel and Visual Basic (VB) for its functionality and UI, it is very limited in how it can interact with existing tools. For example, if a user wants to use OpenPinch to evaluate a system made in DWSIM or other process simulators, the user will need to deconstruct the flow-sheet into its base streams then input them manually into the form (Fig. 1). This along with its unintuitive UI makes it hard to navigate for new users, and slows down the workflow of anyone hoping to utilize it. Despite these shortcomings in terms of UI and interfacing with other programs, OpenPinch is among the best pinch-based HEN optimizers for its breadth of functionality and graphical outputs. . In addition to this, Microsoft Excel is known for its visualization of input data, and OpenPinch succeeds at leveraging this to provide clean outputs for graphs such as composite curves and grand composite curves (Fig. 2). Overall OpenPinch places a heavy focus on its functionality while disregarding many UI design principles that it would benefit from, and despite its good output graphs and visualizations, it lacks overall polish.

Run	Subsystem	Stream	T_S °C	T_T °C	ΔH kW	ΔT_{cont} °C	HTC kW/m²/°C
D		Exhaust 1	67.0	40.3	1081.0	8.0	1.0
D		Exhaust 2	40.3	29.7	2425.0	8.0	1.0
D		VF	25.0	60.0	1741.0	8.0	1.0
D		SFB1	60.0	86.5	781.0	8.0	1.0
D		SFB2	86.5	114.0	814.0	8.0	1.0
D		MA	114.0	236.0	2695.0	1.5	1.0
E		Chiller Water	8.0	4.0	456.0	0.5	1.0
E		CIP Water	15.0	84.0	630.0	2.5	1.0
E		Cow Heat	72.1	13.0	38.9	2.5	1.0
E		Direct Use	15.0	55.0	525.0	2.5	1.0
E		E1 vapour bleed	72.0	72.0	3920.5	0.5	1.0
E		E2 Condenser	72.0	71.9	364.1	0.5	1.0
E		HT Flash	87.0	86.9	2060.8	0.5	1.0
E		HT Flash 2	79.3	79.2	1988.8	0.5	1.0
E		Milk Concentrate	65.3	70.0	259.9	2.5	1.0
E		Preheat COW 1	72.0	13.0	10697.8	2.5	1.0
E		Preheat COW 2	86.9	13.0	279.8	2.5	1.0
E		Preheat COW 3	72.0	13.0	412.1	2.5	1.0
E		Preheat COW 4	71.9	13.0	704.4	2.5	1.0
E		Preheat COW 5	79.2	13.0	240.3	2.5	1.0
E		Raw Milk	8.0	95.0	22834.8	2.5	1.0

Figure 3: OpenPinch Stream Data Input

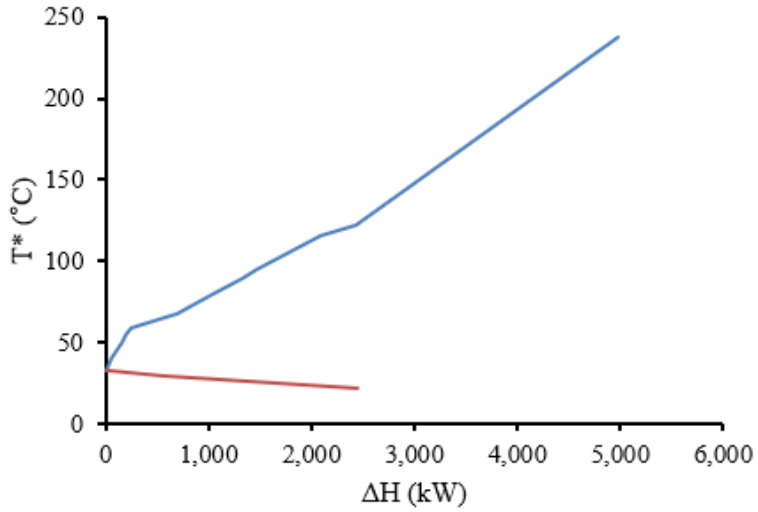

Figure 4: OpenPinch GCC

Aspen Energy Analyzer is the pinch-based HEN module that is integrated with the Aspen process simulation ecosystem. Energy Analyzer as a package is contained with many of Aspen's proprietary software, such as Aspen HYSYS and Aspen Plus - this means for many who use this software, Energy Analyzer is the easiest pinch tool to use. Unfortunately, due to this integration,

the UI for Energy Analyzer follows the direction of the software it is contained in - thus it has the same problems of lacking aesthetics and usability. Aspen software can be seen as mostly functional, but not particularly user-friendly or intuitive, especially for new users or those not accustomed to industrial software interfaces. This can lead to a steeper learning curve for navigating and utilizing the software effectively. This lack of attention to streamlining processes and visual indicators is prevalent in the majority of Aspen software. In terms of functionality, Energy Analyzer is considered generally good, however Ben Maes [19] notes in their case study of applying Aspen software to retrofit problems, that Energy Analyzer has a tendency to fail when confronted with complex HEN designs.

Overall, there are many graphical pinch tools that exist for design and retrofit problems, however the majority of these suffer from poor UI and design in general. One of the main benefits of applying pinch methods to HEN problems is being able to visualize the results and make decisions based on them - so usability and design is a major concern for these software. Time To Adopt (TTA) considerations are also lacking in these systems - while these modules are designed for process engineers who already have a fundamental knowledge of the terminology and techniques, this does not necessarily equate to being able to use the software easily.

2.3 Retrofit Vs Design

Ideal application of pinch analysis is done as a 'blank sheet' design - where the design for the HEN is built from the ground up [10]. Unfortunately in many cases this isn't realistic, and frequently existing systems need to be changed and optimized - this process is known as 'retrofit' or 'retro-vamp'. Truls Gundersen [20] estimates that roughly 70% of all projects in the process sector are for retrofitting existing systems, which shows the importance of this technique. While both of these approaches aim for the same end goal (optimization of a HEN) there are some nuances in the approaches of each. Sreepathi et al [3] in their study of retrofitting methodologies notes their findings of the evolution of retrofitting techniques over time (Fig. 5).

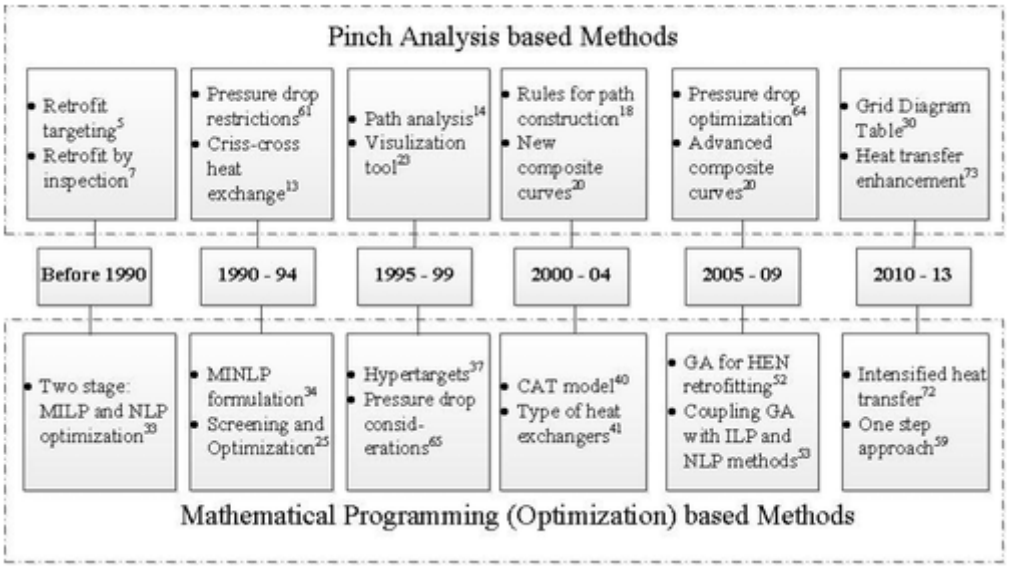

Figure 5: Contributions to HEN retrofitting over 30 years [3]

HEN retrofit can be broadly classified into two groups: thermodynamic-mathematical methods and pure mathematical programming methods [21]. Thermodynamic-mathematical methods, typically involving pinch analysis, employ various graphical analytical tools previously discussed,

such as Grid Diagrams (GD), Composite Curves (CC), and Grand Composite Curves (GCC) for retrofitting. The pinch concept in retrofit methods also incorporates mathematical programming techniques in later stages to optimize the network design. These techniques often aim to minimize costs and enhance energy efficiency, considering the existing constraints of the heat exchanger network (HEN).

Pure mathematical programming based methods - also called optimization methods - can be further reduced to two main processes; deterministic and stochastic optimization. Deterministic optimizations the technique that is employed when all variables and states are known - IE when there are no degrees of freedom in a system, and utilizes linear and non-linear algebra [22]. Stochastic optimization on the other hand supports many unknowns and is more likely to find the global optimum for HEN retrofit problems [23]. Despite these differences, some variations on these methods exist that combine both approaches, such as that proposed by Zhou et al [24] in their case study of a hybrid stochastic-deterministic method for process optimization.

From a graphical tool point of view, retrofitting an existing system requires extra functionality that includes its own design and considerations. The three tools mentioned previously all contain some retrofitting capabilities; OpenPinch requires manual inputs while SuperTarget and Aspen Energy Analyzer interface with their respective process simulators.

3 Design

As discussed in Section 2.2.2 there is a general lack of focus on design principles and usability in existing pinch graphical tools. OpenPinch, for example, is a purely functional program that fulfills all the practical needs of a pinch analysis system while existing in Microsoft Excel sheets. The lack of usability and unintuative design harms the user experience and leads to a system that is hard to navigate and requires a much steeper learning curve. In an effort to demonstrate which principles and methodologies drive great UI and design, this section will discuss the process and considerations that need to be made. Included in this is the general approach of the design process, some key design techniques that are used, and methods to evaluate the efficacy of a UI design.

3.1 Design process

The standard procedure for many large web applications has been that the majority of the UI design is compiled upfront - usually 85% - 95%, then as usability and acceptance testing is conducted, changes are made and drive the development of the project [25]. This process relationship between development and design often comes at the cost of the design iterations failing to keep up with the fast-paced coding sprints found in Extreme Programming (XP) and inevitably leads to a decline in actual development. The solution is thus to have the design process be dependant on what work has been done in the last agile development sprint - 'Development iterations drive usability testing' and then the next development sprint reflects the changes in requirements from usability and acceptance testing - 'Usability Testing Results in Changes in Development'. Using agile project management techniques that integrate both the design and development leads to a symbiotic relationship between these two often separate aspects of a project and allows the requirements of the project to smoothly change as testing is conducted.

One notable limitation of this design pattern is how it is generally tailored for large development teams with separate UI/User Experience (UX) and programming teams. For smaller teams or individual developers, considerations have to be made as to where the majority of the workload will lie - since after applying this methodology each sprint now has two dependant factors, usability testing and development. Existing research is limited on how this development style affects teams of varying sizes and warrants further investigation.

McInerney et al [7] in their case study of common problems that can be identified in many design projects, designed a metric for where efforts should be focused during a design process (Table 1). The design phases from Table 1 can be generalised into key terms: UI architecture is wire-framing and low fidelity designs, UI detail design is prototyping and high fidelity models, and finally UI design change is the process of editing the designs to conform to results from user testing and evaluation. The data indicates that high fidelity models demand the most effort, primarily due to extensive interactions between designers and developers. This study was conducted on many UI professionals and developers to find these values, however in this study no considerations were made to the specifics of the projects and experiences these participants had. Some further research that can be done is to match these values to specific projects - like web design, application design etc - and see how the project determines where the emphasis on effort is placed.

UI Design Phase	Description	Effort
UI architecture	Defining the UI at a gross level, defining the key design direction	20 - 30%
UI detail design	Defining the UI at a level detailed enough to serve as instructions for the programmers	50%
UI design change management & verification	Design rework to address issues that arise after the UI design freeze; also includes work to verify that the UI was implemented as specified	20 - 30%

Table 1: Stages of UI design and the relative effort allocated to each stage [7]

3.2 Design Techniques

During the design process of any large system, there are two key milestones - wire framing and prototyping. Each of these techniques serves their own purpose despite their similarities in execution, and where to use each of them is an important skill for UI/UX developers to learn.

Wire framing is a valuable development tool that interfaces very well with agile management techniques - it involves creating low-fidelity visual representations of UI systems, typically only roughly based on the requirements of the system [26]. Wire frames can be done quickly while still providing valuable information on UI design, and works best after 'round 1' research, where key users and requirements have already been identified and developers know the design metrics [27].

Wireframe	Pros	Cons
Tangible	-Suitable for beginner design user -Increasing soft skills -Spend less time -Low learning curve, no need prior skill	-Limited items for design components -Suitable for Basic design website -Not flexible for experienced user
Digital	- Suitable for Advanced design user or experienced user -Availability of flexible design items -Suitable for complex functionality and detail	-High learning curve -Need prior knowledge/skill for tools -Spent more time -High cost compared to Tangible wireframe

Figure 6: Wire Frame Tangible vs Digital Results [4]

Sutipitakwong et al [4] notes in their study of physical wire framing vs digital wire framing tools (Fig. 6), that tangible wire frames are superior when developing low-tech quick iterations of a design, while digital wire frames can provide much more upfront functionality and can lead straight into prototyping. While this study manages to find some of the strengths and weaknesses of these wire framing styles, continued research into the differences that can be found between specific digital tools and wire framing methods could serve to quantify the differences

further.

Prototyping on the other hand is an approximation of a final design or concept [28]. In contrast to wire framing, prototyping is used much later in the design process, and typically contains far more functionality that is representative of the final system. By making a design that emulates a system without the majority of the infrastructure required for the final implementation of that system, testing becomes easier and issues that cannot be found in a lower fidelity systems can be identified [29]. There are many different techniques and tools that can be utilized when prototyping a system, and choosing the best tool is left to the requirements of the project.

3.3 UI Evaluation

An important portion of the design process is being able to find key metrics and techniques for evaluating the efficacy of a UI design [30]. When evaluating a UI system, the standard testing methods usually fall into two main categories: empirical and inspection [31]. Heuristic evaluation involves experts, or UI professionals, navigating through a user interface identifying problems and pain points, while user testing is done by stakeholders and end users in the same process. Nielson et al [32] created the ten metrics for heuristic testing in 1990, and many other professionals have since expanded on these principles to layout a solid framework for conducting heuristic testing - such as Murillo et al [33] and Allen et al [34].

In general, when conducting usability testing on a system for both empirical and inspection, the most important methodology for gaining valuable feedback is controlling the social dynamics with the test subjects. Carol Barnum [35] notes the main techniques for gaining impartial results - most of these are ways to ensure that the tester is not directly or indirectly affecting the testee. These behaviours can include:

1. **Neutral Prompts** - A prompt to the user should not direct them in what to do.
2. **Comfortable Testing Environment** - The user should feel comfortable to get results that are indicative of their regular behaviour.
3. **Unbiased responses** - A tester should give little to no feedback on users interactions, but where necessary the responses should be unbiased.
4. **Balance Praise** - A tester should not give excessive praise towards user behaviours.

A novel way of utilizing heuristic testing methods that additionally categorizes the findings was presented in the work of Murillo et al [33] in their case study. The first step in this method is to gather some professionals and have them conduct their own individual usability evaluation on the system. This is done to ensure that each participant has the opportunity to express their thoughts without interference from other participants. The next step is to gather the participants and have them unify their findings into a single list following which the the participants are asked to cooperate in ranking the issues based on a metric determined by the test conductor. After all these steps the test conductor will have a ranked list of issues in order of their severity. This method of heuristic testing is ideal when a lot of time can be allocated to the testing phase, but this needs to be determined and used selectively. One noticeable lack of discussion in this study regarding this method of heuristic testing is how it compared to the standard heuristic testing philosophies, as the heuristic testing technique potentially sacrifices post-test processing in favour of a higher upfront time cost of setting up the evaluation. An evaluation of how this style of testing compares to other methods when considering the additional time required could be beneficial to consolidate this method as legitimate - a time to quality/number of results graph

in particular would be beneficial.

A/B testing is another common testing technique found in the industry as noted by Siqueira et al [36]. A/B testing is the process in which two iterations of a design are presented to a test participant and each are evaluated on their own merits. Fig. 7 notes the general process in which this is done from . First, Two variants of a design are made and a hypothesis is determined on which design will perform better under predetermined testing requirements. Second, multiple users are given a random variant and asked to walk through the system through prompts. Finally the data from the experiments are compiled and compared against the hypothesis. A/B testing typically gathers hard empirical results, such as clicks and time spend on UI elements as well as some user engagement metrics, which differs itself from the other testing methods which are typically more subjective and generally contain a lot more stylistic pain points. This also then requires some statistical post-processing of the data to ensure that differences are significant. Another benefit from A/B testing is the 'better' variant will be used in the next testing cycle. There are many A/B testing frameworks that can be used, such as the one outlined by Siqueira et al [36], but these general frameworks often require fine tuning based on what the tester is hoping to achieve from their hypothesis as well as which specific parts of a system they are focusing on. Finding the best metrics for a given project is something that is glossed over in Quin et al [5], but this seems to be very important in generalizing methods for A/B testing.

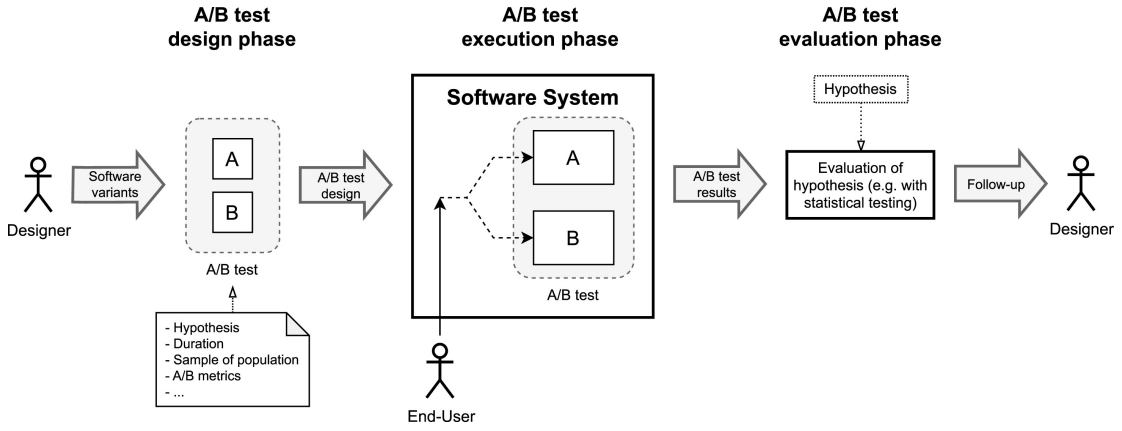

Figure 7: A/B Design Process [5]

The difference in results from heuristic and user testing were identified in a case study presented by Wang et al [37]. The results from the heuristic group demonstrated that many more potential pain points and issues can be identified by UI professionals (an average of 20), but these issues contained a higher than average set of 'false positives' (false alarms) and minor issues that don't have a large affect on how the user interacts with the system. The user testing group however had a much smaller number of identified pain points, but each of these - by definition - are those that needed to be addressed. The difference then can be seen that user testing does well to identify large 'real' pain points that should take priority in the next design and development cycle while heuristic testing are typically contain a large number of minor fixes that should be blocked until extra development resources can be allocated. Both of these methods have their own merits and should be used at the conductors discretion.

In general finding a balance of both empirical and inspection testing is ideal for evaluating a UI system. Both of these methods provide their own valuable points of feedback and favouring one over the other can lead to misidentified or unimportant testing results. In addition to this, the specific process of empirical or inspection testing should be determined by the scope and

time constraints of the testing process. This section has identified two general methods that fit into these categories - but depending on the project more specific evaluation methods may perform better. For example a pinch analysis tool may benefit from additional user testing with process engineers, and have a novel user testing technique based around the workflow of a process simulator.

4 Software

In the realm of process engineering - and specifically pinch analysis techniques - identifying appropriate software tools is crucial. Pinch analysis lends itself well to digital solutions, as pinch analysis uses many Mixed-Integer Linear Programming (MILP) standards. This section analyses various software and software principles through the lens of finding best practices in the development of these systems to achieve usable and intuitive software. Being able to identify the metrics through which high quality software is produced is important during both the development of software and the evaluation of tools to be used in development.

4.1 Software Design Principles

J. R. Mckee [38] estimates that roughly 65%-75% of total support and software development efforts are spent on the maintenance of a system. This is largely due to the lack of frameworks available for measuring the maintainability of a system - and without these metrics it becomes much harder to identify which modules and processes need more considerations for their future maintenance. Aggarwal et al [39] proposes the following three metrics to identify the maintainability of a software system:

1. Readability of source code (RSC)
2. Documentation Quality (DoQ)
3. Understand-ability of Software (UoS)

These key points were identified to have the most impact on maintainability from the stand point of future development of a system - instead of bug fixed and optimizations. RSC is the measure of comments and file structure of the source code, in this case study it was $[NumLinesCode/NumLinesComments]$. The second metric, DoQ measures the quality of documentation of the system using Gunning's Fog Index [40]. The final criteria is a combination of the other two, UoS is how well the documentation (DoQ) equates to the source code (RSC) via how its 'language' is similar between them.

One of the main concepts used by Aggarwal et al [39] to evaluate the maintainability of a system was readability. Readability can be defined as the human judgement of how easy a text is to read - in software engineering principles it applies to code instead [41]. Readability is a design principle that should be prevalent in every section of a application, and is almost always used to facilitate other principles such as maintainability, usability, and general code quality. It's due to this that it becomes important to be able to measure the readability of a system consistently. One of the most popular frameworks for evaluating the readability of a piece of software was captured by Buse et al [42] as Fig. 8 - this framework is an empirical measure of readability and can be automatically calculated by tools that gather and compile the results, making it ideal for CI/CD pipelines. An interesting future research point that could be made is how these readability metrics can be affected or improved by the use of AI. The development or research into an AI tool that can automatically improve the readability of a piece of software would be very valuable, and the discussion around whether it performs better than traditional tools (like those based on Gunning's fog index) could be done.

Avg.	Max.	Feature
•	•	identifier length
•	•	indentation
•	•	line length in characters
•		# blank lines
•		# spaces
•		# comments
•	•	# numbers
•	•	# identifiers
•	•	# keywords
•		# assignments
•		# branches
•		# loops
•		# arithmetic operators
•		# comparison operators
•		# parentheses
•		# periods
	•	# occurrences of any single character
•	•	# occurrences of any single identifier

Figure 8: Readability Metric [6]

Another powerful software design principle is Usability - especially in the case of software development tools. Usability as a design principle is generally thought of as secondary to functionality when developing an Application Programming Interface (API) for any type of service - especially in the case that an API is being developed under agile project management styles, where Minimum Viable Product (MVP) development is often preferred. Myers et al [43] challenge this style of API development due to the fact that usability should be the primary focus for an API, as an API at it's core is built around the interactions it will have with users - both developers and stakeholders. Typically, the goal of any API developer is to minimize support and development costs, reduce deployment time, and maximize adoption, especially for public APIs. On the other hand, developers looking to adopt an API focus on simplicity, consistency, and reliability. Often these requirements conflict and lead to an API that is difficult to consume, or in some cases lead to many production issues when developers cannot use an API in its intended way.

The solution to mitigating these potential issues is to place a heavy focus on the design of the API before its implementation. Taking extra time to conduct usability testing with developers can ensure that the structure of an API follows the 'mental model' that a developer will have when consuming it - this includes components such as class names, endpoint structures, and default values of objects. Having an API that is intuitive to use decreases adoption time, as shown by Stylos et al [44], development time can be decreased from 2.4 - 11.2 times and since productions roll out times will be faster and less prone to bugs, users will also be positively affected.

API development in general conflicts with agile development techniques, as retroactive changes to an API that is in production will lead to legacy code breaking down and familiarity developers have with the API degrading. This demonstrates the importance of appropriate API design before any development iterations. When adoption time for a certain API is too large for a project, other alternatives are available to switch to - this is not the case for internal APIs or niche APIs with no alternatives. While there are many ways one can evaluate the usability of an API, as from Gill et al [45], there is still a overarching lack of systematic patterns that lead

to a usable API - a gap that will need to be filled.

When evaluating how usable an API is, Grill et al [45] used the following metric in their case study:

Name	**Description**
Complexity	An API should not be too complex. Balance complexity and flexibility. Use abstraction.
Naming	Names should be self-documenting and used consistently.
Caller's Perspective	Make the code readable, e.g., `makeTV(Color)` is better than `makeTV(true)`.
Documentation	Provide documentation and examples.
Consistency and Conventions	Design consistent APIs (order of parameters, call semantics) and obey conventions (get/set methods).
Conceptual Correctness	Help programmers use an API properly by using correct elements.
Method Parameters and Return Type	Use few parameters. Return values should indicate the result of the method. Use exceptions when exceptional processing is needed.
Parameterized Constructor	Always provide a default constructor and setters rather than a constructor with multiple parameters.
Factory Pattern	Use the factory pattern only when inevitable.
Data Types	Choose correct data types. Avoid forcing users to use casting. Do not use strings if a better type exists.
Concurrency	Keep concurrent access in mind.
Error Handling and Exceptions	Define class members as public only when necessary. Handle exceptions near where they occur. Error messages should convey sufficient information.
Leftovers for Client Code	Minimize the amount of code the user needs to type.
Multiple Ways to Do One	Do not provide multiple ways to achieve the same thing.
Long Chain of References	Avoid using long complex inheritance hierarchies.
Implementation vs. Interface	Prefer interface dependencies as they are more flexible.

Table 2: API Design Guidelines

Similarly to how Myers et al [43] outlines the success metric when designing the API, Grill et al's [45] criteria primarily revolves around matching potential developer's mental 'image' of

how the API will be set out and ensuring default values are consistent. Method parameters, parameterized constructor, data type, concurrency, multiple ways, and interface implementations are all examples of applying this methodology.

4.2 Monolithic vs Micro-services

The complexity of a system has a large impact on the maintenance time and cost - Banker et al [46] found in their study evaluating software systems based on metrics such as number of modules and module size that maintenance costs can be raised up to 25% when a system is too complex. From this, the software engineering has two general approaches to developing complex systems; monolithic and micro-services.

A monolith is a traditional project style, where a system shares a single code base and typically none of the internal modules are independently executable [47]. All of the functionality, components, and UI elements are contained within a single program platform - which generally make them easier to deploy, develop and test [48]. These benefits make it easy for small to medium size projects, where large scaling and maintenance aren't a factor - but as the complexity of a monolithic program increases, there is a sizeable increase in resource consumption. On the other hand, micro-services are designed to separate concerns into independent modules [49]. The concept of micro-services is not new, however with the industry shift to cloud base approaches, micro-service design has become increasingly popular. Some of the benefits of micro-service design include vast scalability, easy error isolation, and a positive cost-performance return.

An increasingly popular software design architecture that is being adopted by many large tech companies is the 'Modular Monolith' [50]. This architecture is a type of monolithic application that is internally structured into loosely coupled modules. This approach combines the straightforward deployment benefits of a monolithic system with the maintainability and separation of concerns usually associated with a modular system. Although modular monoliths can serve as a standalone architectural choice, they also offer a potential stepping stone to micro-services architectures, making it easier to decouple these individual modules at a later date to operate as their own services. The utility of a modular monolith isn't limited to large tech companies with numerous services that need to be released. Often, when developing a complex system, a modular monolith can provide a scalable yet manageable framework that supports a range of functionalities in a unified environment. This architectural style is particularly beneficial in scenarios where teams are transitioning from a traditional monolithic approach and are not yet ready to adopt a fully distributed micro-services architecture. Additionally, modular monoliths can reduce the complexity and overhead associated with deploying and managing multiple independent services while still allowing for clear boundaries and independence between different functional areas [51]. This makes them ideal for projects where rapid scaling or iterative testing of isolated features is necessary.

The choice of system design process is dependant on the project and needs to be evaluated on a case-by-case basis. For small to medium-sized projects, monolithic and modular-monolithic architectures are generally best, as these projects do not require extensive scalability considerations in the short term. For these projects the ease of deployment, testing, and maintenance is the driving factor for choosing monoliths. When a project is identified to require high scaling in the future, or single modules of a system need to be accessed independently of the entire system, micro-service design is superior. For example a process simulator may benefit from following a micro-services design, where a pinch module is included separately with the functionality of interfacing with other simulators.

One key advantage micro-service design also has over a monolith is how well it lends itself to distributed systems and computing [52]. In the pinch tools previously mentioned, Energy Analyzer and SuperTarget were a part of the monolithic designs of their process simulator - they could not be used independently without them. This integration, while easy, restricts flexibility and scalability that could be enhanced through a micro-services architecture.
A micro-services approach to developing pinch analysis tools could significantly leverage distributed programming to enhance performance and scalability. In a distributed system, different services can run on various servers or even across multiple data centers, allowing for more efficient data processing and management. For pinch analysis, this means that complex calculations for energy optimization, such as the generation and manipulation of composite curves and grand composite curves, could be handled by separate, specialized services. Each service could be optimized for specific tasks like data collection, heat integration analysis, or optimization routines, and scaled independently according to demand.

4.3 Software technology reviews

One intersection between software and pinch analysis techniques is graphical representations. Pinch analysis produces many visual outputs that are used to refine and further optimize HEN networks - and its due to this reason that it becomes important to find software tools that can produce accurate graphs and diagrams for to match these outputs. This section is a review of the most popular React frameworks that support graph creating and management in an effort to find an ideal library for pinch analysis.

4.3.1 D3.Js

D3 [53] is a JavaScript framework for creating interactive data visualizations, and is largely used for displaying large complex charts and graphs. Due to it's high skill ceiling, D3 is among the best when creating custom visualizations for web projects. Table 3 contains the notable pros and cons of using D3.Js in a project.

	Pros	**Cons**
1	High level of customization allows for detailed and tailored visualizations.	Steep learning curve, especially for those not familiar with modern JavaScript and SVG.
2	Strong community support and documentation	The frequent updates and changes in D3.Js versions can make it difficult to find current examples or tutorials that match the latest API.
3	Performant with small-medium datasets	Performance degrades with very large datasets (Like those for machine learning)
4	Integrates well with other libraries	Lacks high level components

Table 3: Pros and Cons of D3.Js

D3's main selling point is the customizability and complexity of designs it can make. This makes it ideal for projects that have unconventional graphs, or need more functionality than other graphing libraries would allow the user to create. This complexity along with the extensive community support, official documentation, and performance are the main pro's that come along with using D3. Some notable limitations of the D3 library, included in the cons section of 3, are

the steep learning curve, lack of high level components, and the constant updates. A side effect of D3's high skill ceiling is it has a very steep learning curve - all of the functionality that D3 offers means that it takes much longer for beginners to learn. Additionally D3 does not package high level components into its library - components such as default graphs and structures need to be created manually by the developer. D3 is a library that has continuously been updated over its 13 years since creation - its because of this that many examples online of D3 projects are outdated and no longer use the current best practices. Overall, D3 is an incredibly powerful graphing framework that comes with the caveat of requiring much more effort and learning than other low-level libraries that place a greater focus on ease of use. Custom graphs such as HEN diagrams is where D3 excels, one such example from Mike Bostock [54] demonstrates this very well - an interactive graph with changing data points and a lot of complexity.

4.3.2 React-vis

React-vis [55] is a React-based data visualization library built on top of D3 and developed by Uber [56], designed to simplify the process of creating charts and graphs. React-vis offers a variety of chart types such as line, bar, scatter, and pie charts, which are optimized for use within React environments.

	Pros	**Cons**
1	Simple API that allows for quick development.	Not much functionality for custom graphs.
2	Strong integration with the React ecosystem.	Lacks depth of graph features.
3	Maintained frequently by Uber.	Performance degrades with very large datasets.
4	Shallow learning curve.	Poor interactivity with graphs.

Table 4: Pros and Cons of React-vis

The strengths of React-vis can also be seen as the majority of its weaknesses. React-vis is primarily designed for fast development, with many React components available to simply add data to and use. This focus leads to a library that lacks customization options and complexity, making it less than ideal for any project that requires a complex graph, but great for projects that just need the graphs that they provide. This is in stark contract to other libraries such as D3 which places a heavier focus on broad functionality of graphs. Table 4 demonstrates this the best - where most of the pros revolve around how many pre-built components that are packaged with React-vis, and the cons are mostly about how its lacking in custom functionality. This lack of custom functionality means that for building HEN and other system diagrams, React-vis falls short for whats needed. For some graphs such as CCs and GCCs React-vis would be an ideal library - as these graphs are simple outputs plotted on a Cartesian plane and this is what React-vis excels at - but React-vis does not have any pre-built HEN diagrams included in it's package.

4.3.3 React-chartjs-2

React-chartjs-2 (RC2) [57] is a React wrapper for the popular JavaScript library, ChartJs [58]. ChartJs is a library that makes it very easy to create simplistic graphs - components such as bar, line, and pie charts are baked into the package and can be easily added to projects. RC2 has

	Pros	Cons
1	Rich visualization options for included graphs.	High learning curve for those now familiar with React.
2	Strong integration with the React ecosystem.	Large overhead and package size.
3	Responsive and supports interactivity.	Limited to ChartJs functionality.
4	Strong community support.	Performance degrades with very large datasets.

Table 5: Pros and Cons of React-chartjs-2

the exact same functionality as ChartJs but is instead designed to be used with React projects. Some of the pros and cons of RC2 are included below in Table 5.
RC2 contains a lot of the same issues found with D3 and React-vis, with degrading performance with large datasets and package size and because RC2 is based on the ChartJs library, all of the functionality that RC2 has is limited to that offered in ChartJs. RC2 has some notable advantages to the other libraries discussed however - ChartJs has existed much longer than other libraries and has much more community support and documentation, and RC2's backwards compatibility with ChartJs means that it benefits from this as well. RC2 also has much of the same capabilities as D3 in its interactive graphing and responsiveness. The lack of custom graphs is again an issue with this framework, similarly to React-vis and is thus not suitable for building HEN diagrams.

4.3.4 Software technology conclusion

From the reviews conducted on the three frameworks available, D3 is the ideal library to be used in conjunction with pinch analysis. React-vis and RC2 both suffer from their lack of support with custom graphs and data structures - meaning that these libraries cannot be used for the entirety of the pinch system and additional libraries will need to be used to fill this gap. Using D3 would mean a higher workload in general - as D3 doesn't natively include support for popular high level graphs that will need to be used in a pinch system and the overall learning curve for D3 is much higher than the other two solutions. Considerations will also have to be made when using D3 to ensure that large amounts of data aren't being used for graphing when they aren't needed - as there is a sharp decrease in performance with larger amounts of data - however this is a common problem between most React graphing libraries and can't be fixed by finding a better solution.

5 Summary

In conclusion, this literature review has examined existing pinch analysis tools through the lens of usability, design, and software principles. The aim of this report has been to note some issues with current pinch analysis tools, then explore the principles and techniques that would produce a higher polished system. First, the current state of pinch tools were examined, along with some historical context of the progression of graphical process engineering methods. From this, pinch analysis tools were found to be generally lacking in usability and intuitive design, with most of the popular pinch tools having an outdated or minimal UI.

In the design section, some key design principles were explored within the context of applying them to these pinch tools. The design process to follow, the techniques in which to develop high-quality user interfaces, and common methods of evaluating the UI system were found. Wire framing and prototyping were the key techniques that these systems would benefit from and testing methods such as heuristic and A/B testing were explored.

Next, software principles that indirectly contribute to the user experience, such as maintainability, usability, and scalability were identified within the confines of the project, while graphing libraries to facilitate the graphical outputs of the system were evaluated against each other.

Conclusions drawn from this literature review will be used to inform decisions around the development of a pinch analysis module in the future. The specific techniques around implementing maintainability and a usable API will be applied in the development stage, while design paradigms and evaluation methods will drive the front-end UI development cycle. In addition to this, the current issues regarding existing pinch tools will be noted and a conscious effort will be made to avoid these pitfalls.

Considering the limitations of current tools regarding scalability, the new pinch analysis module will be designed as a micro-service. This approach will allow it to leverage the benefits of an existing distributed system, enabling it to function efficiently as an independent module within larger systems. This scalability is crucial for accommodating varying load demands that current tools don't have to address, and integrating seamlessly with existing and future technologies in a process simulator.

References

[1] Jun Yow Yong, Petar Sabev Varbanov, and Jiří Jaromír Klemeš. Heat exchanger network retrofit supported by extended grid diagram and heat path development. *Applied Thermal Engineering*, 89:1033–1045, 2015.

[2] KBC Advanced Technologies. Supertarget 7 release notes. `https://www.kbcat.com/php/sd/ST7-Release%20Notes.pdf`, 2023. Accessed: 2024-04-29.

[3] Bhargava Krishna Sreepathi and G. P. Rangaiah. Review of heat exchanger network retrofitting methodologies and their applications. *Industrial Engineering Chemistry Research*, 53(28):11205–11220, 2014.

[4] Sutipong. Sutipitakwong and Pornsuree. Jamsri. Pros and cons of tangible and digital wireframes. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–5, 2020.

[5] Federico Quin, Danny Weyns, Matthias Galster, and Camila Costa Silva. A/b testing: A systematic literature review. *Journal of Systems and Software*, 211:112011, 2024.

[6] Daryl Posnett, Abram Hindle, and Premkumar Devanbu. A simpler model of software readability. In *Proceedings of the 8th Working Conference on Mining Software Repositories*, MSR '11, page 73–82, New York, NY, USA, 2011. Association for Computing Machinery.

[7] Paul McInerney and Rick Sobiesiak. The ui design process. *SIGCHI Bull.*, 32(1):17–21, jan 2000.

[8] New Zealand Government. Climate change response (zero carbon) amendment act 2019. `https://environment.govt.nz/acts-and-regulations/acts/climate-change-response-amendment-act-2019/`, 2019. Accessed: 2024-04-29.

[9] Innovation New Zealand Ministry of Business and Employment. Process heat current state fact sheet. `https://www.mbie.govt.nz/dmsdocument/152-process-heat-current-state-fact-sheet-pdf`, 2022. Accessed: 2024-04-29.

[10] Ian C. Kemp. *Pinch analysis and process integration : a user guide on process integration for the efficient use of energy*. Butterworth-Heinemann, Amsterdam ;, 2nd ed. edition, 2007.

[11] Natchanon Angsutorn and Rungroj Chuvaree. 24th european symposium on computer aided process engineering. In *Computer Aided Chemical Engineering*. Elsevier, 2014.

[12] Assaad Zoughaib. *From Pinch Methodology to Pinch-Energy Integration of Flexible Systems*. ISTE Press - Elsevier, 2017.

[13] S. Bobby Rauf. *Understanding Mollier Diagram*, page 8. River Publishers, 2 edition, 2023.

[14] F. Friedler, T. Blicket, J. Gyenis, and K. Tarjáns. Computerized generation of technological structures. *Computers Chemical Engineering*, 3(1):241–249, 1979.

[15] Ferenc Friedler, Kathleen B Aviso, Botond Bertok, Dominic CY Foo, and Raymond R Tan. Prospects and challenges for chemical process synthesis with p-graph. *Current Opinion in Chemical Engineering*, 26:58–64, 2019. Energy, Environment Sustainability: Sustainability Modeling Reaction engineering and catalysis: Green Reaction Engineering.

[16] KBC Global. Supertarget 7, 2023.

[17] Tim Walmsley. Openpinch. Open Source Software, 2024.

[18] AspenTech. Aspen energy analyzer. Software available from Aspen Technology, Inc., 2023.

[19] Ben Maes. *Exploring Aspen Energy Analyser to Improve Processes*. Master's thesis, Instituto Superior Técnico, June 2018. Thesis to obtain the Master of Science Degree in Chemical Engineering.

[20] T. Gundersen. *Retrofit Process Design - Research and Applications of Systematic Methods*, pages 213–240. CACHE Elsevier, 1990.

[21] Ming Pan, Igor Bulatov, and Robin Smith. Heat transfer intensification for retrofitting heat exchanger networks with considering exchanger detailed performances. *AIChE Journal*, 64(6):2052–2077, 2018.

[22] A. R. Ciric and C. A. Floudas. A retrofit approach for heat exchanger networks. *Computers & Chemical Engineering*, 13(6):703–715, 1989. Accessed through SciFinder.

[23] Robin Smith, Megan Jobson, and Lu Chen. Recent development in the retrofit of heat exchanger networks. *Applied Thermal Engineering*, 30(16):2281–2289, 2010. Selected Papers from the 12th Conference on Process Integration, Modelling and Optimisation for Energy Saving and Pollution Reduction.

[24] Teng Zhou, Yageng Zhou, and Kai Sundmacher. A hybrid stochastic–deterministic optimization approach for integrated solvent and process design. *Chemical Engineering Science*, 159:207–216, 2017. iCAMD – Integrating Computer-Aided Molecular Design into Product and Process Design.

[25] Jennifer Ferreira, James Noble, and Robert Biddle. Agile development iterations and ui design. In *Agile 2007 (AGILE 2007)*, pages 50–58, 2007.

[26] Talita Stockle, editor. *UX Design Process*. Smashing Magazine, Freiburg, Germany, 2013. For information about this resource please contact CLoK@uclan.ac.uk.

[27] Matthew J. Hamm. *Wireframing Essentials*. Packt Publishing, Limited, 2014. ProQuest Ebook Central.

[28] Bradley Camburn, Vimal Viswanathan, Julie Linsey, David Anderson, Daniel Jensen, Richard Crawford, Kevin Otto, and Kristin Wood. Design prototyping methods: state of the art in strategies, techniques, and guidelines. *Design Science*, 3:e13, 2017.

[29] Michael Schrage. The culture(s) of prototyping. *Design Management Journal (Former Series)*, 4(1):55–65, 1993.

[30] Zulfiandri, Silma Novshienza Putri, and Aang Subiyakto. Evaluating user interface of a transport application using usability evaluation methods. In *2021 9th International Conference on Cyber and IT Service Management (CITSM)*, pages 1–7, 2021.

[31] Adrian Fernandez, Emilio Insfran, and Silvia Abrahão. Usability evaluation methods for the web: A systematic mapping study. *Information and Software Technology*, 53(8):789–817, 2011. Advances in functional size measurement and effort estimation - Extended best papers.

[32] Jakob. Nielsen and Rolf. Molich. Heuristic evaluation of user interfaces. In *Conference on Human Factors in Computing Systems - Proceedings*, pages 249–256, 1990.

[33] Braulio Murillo, Jose Pow Sang, and Freddy Paz. Heuristic evaluation and usability testing as complementary methods: A case study. Lima, Peru, 2023. Pontificia Universidad Católica del Perú. Braulio Murillo also affiliated with DHL Express Perú, Callao, Peru.

[34] Mureen Allen, Leanne M. Currie, Suzanne Bakken, Vimla L. Patel, and James J. Cimino. Heuristic evaluation of paper-based web pages: A simplified inspection usability methodology. *Journal of Biomedical Informatics*, 39(4):412–423, 2006.

[35] Carol M. Barnum. *Conducting a Usability Test*, pages 1–2. Elsevier, 2011.

[36] Jorge Gabriel Siqueira and Melise M. V. Paula. Ipead a/b test execution framework. *Proceedings of the XIV Brazilian Symposium on Information Systems*, 2018.

[37] E Wang and B Caldwell. An empirical study of usability testing: Heuristic evaluation vs. user testing. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 46, pages 774–778, 2002.

[38] J. R. McKee. Maintenance as a function of design. In *Proceedings of the AFIPS National Computer Conference*, pages 187–193, Las Vegas, 1980.

[39] K.K. Aggarwal, Y. Singh, and J.K. Chhabra. An integrated measure of software maintainability. In *Annual Reliability and Maintainability Symposium. 2002 Proceedings (Cat. No.02CH37318)*, pages 235–241, 2002.

[40] R. Gunning. *The Technique of Clear Writing*. McGraw-Hill, 1952.

[41] Raymond Buse and Westley Weimer. A metric for software readability. In *ISSTA'08: Proceedings of the 2008 International Symposium on Software Testing and Analysis 2008*, pages 121–130. ACM, 2008.

[42] Raymond P L Buse and Westley R Weimer. Learning a metric for code readability. *IEEE transactions on software engineering*, 36(4):546–558, 2010.

[43] Brad A. Myers and Jeffrey Stylos. Improving api usability. *Communications of the ACM*, 59(6):62 – 69, 2016. Cited by: 123; All Open Access, Green Open Access.

[44] Brad A. Myers and Jeffrey Stylos. The implications of method placement on api learnability. In *Proceedings of the 16th ACM SIGSOFT Symposium on Foundations of Software Engineering*, pages 105–112, New York, 2008. ACM Press. Atlanta, GA, Sept. 23–27.

[45] Thomas Grill, Ondrej Polacek, and Manfred Tscheligi. Methods towards api usability: A structural analysis of usability problem categories. In *Proceedings of the 4th International Conference on Human-Centered Software Engineering*, pages 164–180, Toulouse, France, 2012. Springer.

[46] Rajiv D. Banker, Srikant M. Datar, Chris F. Kemerer, and Dani Zweig. Software complexity and maintenance costs. *Commun. ACM*, 36(11):81–94, nov 1993.

[47] N Bjørndal, Antonio Bucchiarone, Manuel Mazzara, Nicola Dragoni, and Schahram Dustdar. Migration from monolith to microservices : Benchmarking a case study. 03 2020.

[48] Grzegorz Blinowski, Anna Ojdowska, and Adam Przybyłek. Monolithic vs. microservice architecture: A performance and scalability evaluation. *IEEE Access*, 10:20357–20374, 2022.

[49] Alan Sill. The design and architecture of microservices. *IEEE Cloud Computing*, 3(5):76–80, 2016.

[50] Ruoyu Su and Xiaozhou Li. Modular monolith: Is this the trend in software architecture?, 2024. Copyright - © 2024. This work is published under http://creativecommons.org/licenses/by-nc-sa/4.0/ (the "License"). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2024-01-24.

[51] Nuno Gonçalves, Diogo Faustino, António Rito Silva, and Manuel Portela. Monolith modularization towards microservices: Refactoring and performance trade-offs. In *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*, pages 1–8, 2021.

[52] Tasneem Salah, M. Jamal Zemerly, Chan Yeob Yeun, Mahmoud Al-Qutayri, and Yousof Al-Hammadi. The evolution of distributed systems towards microservices architecture. In *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 318–325, 2016.

[53] Michael Bostock. D3.js - data-driven documents. `https://d3js.org/`, 2023. Version 7.5.0.

[54] Mike Bostock. Streamgraph transitions. `https://observablehq.com/@d3/streamgraph-transitions?intent=fork`, August 2023. Accessed: 28-04-2024.

[55] Uber Technologies Inc. React-vis: A composable visualization system built on top of react and d3. `https://uber.github.io/react-vis/`, 2023. Accessed on [02/03/2024].

[56] Uber Technologies Inc. Company overview. `https://www.uber.com`, 2024. Accessed on [02/03/2024].

[57] React-chartjs-2. `https://react-chartjs-2.js.org/`. Accessed: 2024-04-28.

[58] Chart.js. `https://www.chartjs.org/`. Accessed: 2024-04-28.

6 Appendix

Included below is the project proposal for this literature review: