IE3100R System Design Project

Optimising Life Cycle of Test Wafers

NUS supervisor: A/Prof Aaron CHIA Eng Seng

Team Members: Cheng Zheng Ting | Lim Shi Pei | Liu Wei | Liu Zhe Pei

Problem Description

To maximise cost efficiency, test wafers are recycled or repurposed from different types of test wafers to prolong their lifespan and avoid the initiation of new test wafers. This process is also known as downgrading. Currently, the lack of a standardised method for downgrading test wafers results in several challenges:

Cost: Our analysis has revealed an excess of more than 30,000 units even after accounting for a 20% buffer, leading to more than **$500,000** in unnecessary expenses.

Time: Another critical factor driving our project is the need to streamline the currently employed time-consuming process which determines possible downgrade paths between wafers. Technicians spend over **4 hours daily** navigating an SQL database to manually identify suitable downgrade paths based on their technical experiences. This process is plagued by inefficiencies, primarily due to its reliance on extensive database knowledge and meticulous examination.

Objectives

Our aim is to optimise inventory management using a wafer downgrade algorithm and a visualisation dashboard for effective inventory management planning, thereby **reducing the costs outlined.**

By implementing a visual representation of the test wafer inventory and its downgrade relationships, we aim to drastically **reduce the time required** for these tasks, enabling more efficient Inventory management and planning.

Key Skills

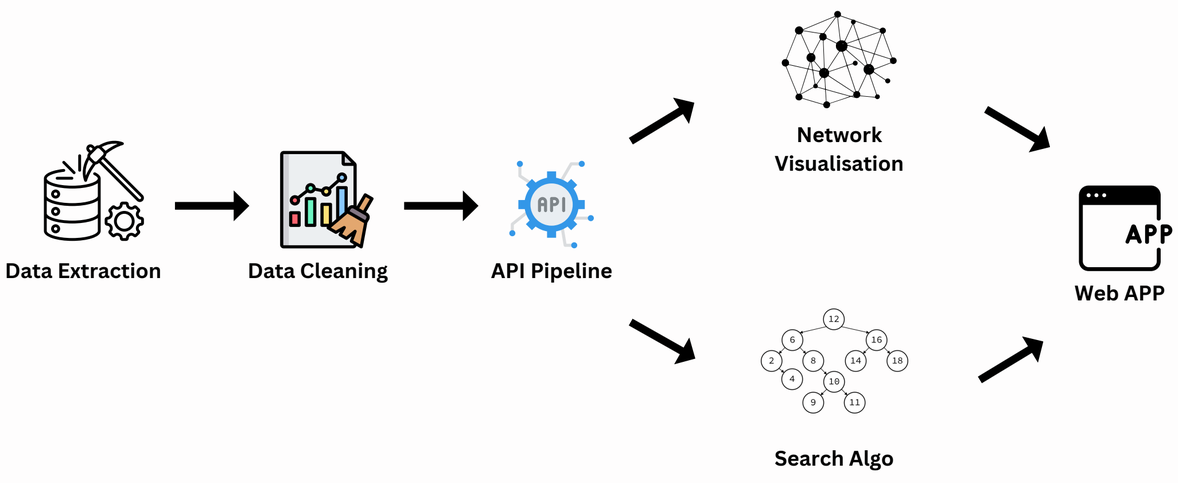
Front End:

Back End:

Project Management:

AGILE project management framework.

Methodology



Data Extraction: We extracted data from the Micron SQL Database to analyse and create a new dataset that will be foundational for our solution.

Data Cleaning: This step involves wrangling the data and organising it into the correct data structure suitable for our API pipeline. This pipeline is specifically designed to facilitate seamless integration between the front end and back end of our solution.

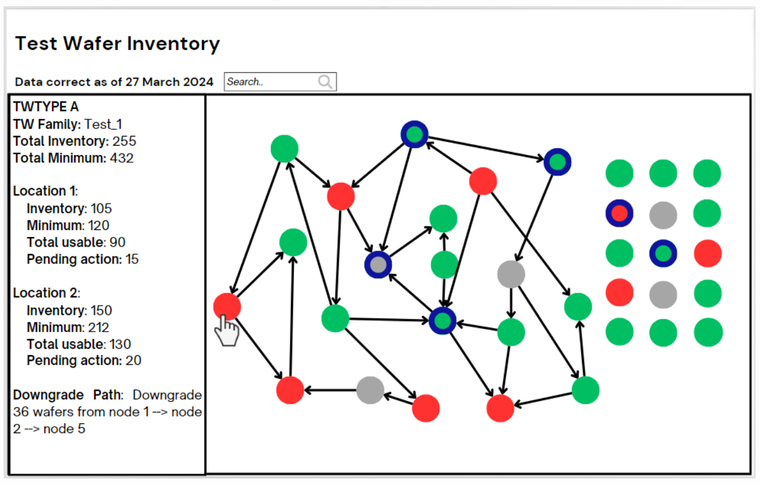
API Pipeline: We aim to establish a pipeline that efficiently delivers packaged, real-time data to both the front end and back end, ensuring timely and accurate data flow.

Network Visualisation: Using the vis.js package, we will represent the inventory data and its corresponding downgrade paths in a network format. This visualisation will incorporate various pieces of information, such as inventory levels, to provide a comprehensive view of the system.

Search Algorithm: We have implemented the Breadth-First Search (BFS) algorithm to identify an optimal downgrade path for a test wafer. This approach aims to reduce excess inventory.

Web App: The front-end network visualisation and the back end search algorithm will be integrated into a web application. This app includes additional features such as search capabilities and an information pop-up window to enhance user interaction and accessibility.

Frontend

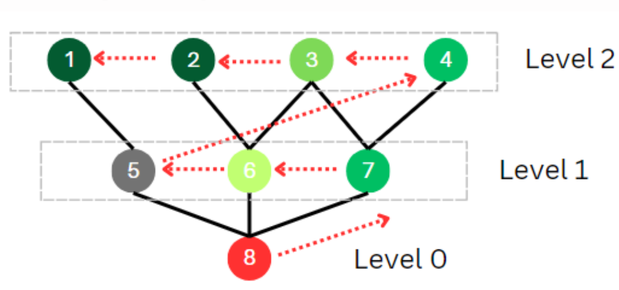


User Interface (illustration)

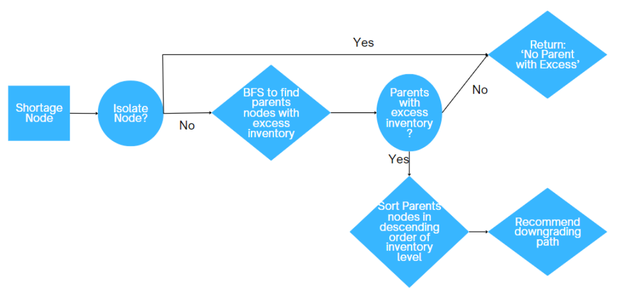
- Utilised HTML, CSS, and JavaScript for seamless integration with Micron's in-house solutions network.
- Dynamic network graphs for displaying wafer connections utilising vis.js library.
- Node color coding: grey (no minimum quantity), red (insufficient inventory), green (surplus).
- Search function for specific wafers or keywords, with auto-highlight and fit.
- Data updates every 15 minutes for real-time accuracy.
- Highlight isolated and family-related nodes for inventory management optimisation opportunities.
- Pop-up display provides detailed wafer information.
- Designed to enhance inventory management and user experience for managers and technicians.

Backend

Downgrade Algorithm Overview:



Breadth First Search (illustration)



Algorithm Logic

Capabilities of our backend solution

(a) Graph Modelling - Unweighted, directed graph using adjacency matrix

(b) Identifying Potential Source Nodes - For each target node, the algorithm initiates BFS traversal with an inverted adjacency matrix

(c) Filtering Potential Source Nodes - From the list of nodes within the BFS traversal, the algorithm filters out the shortage (red) nodes.

(d) Ranking Potential Source Nodes - Remaining excess (green) nodes are ranked in descending order based on their percentage of excess inventory.

Percentage Excess = (Total inventory - minimum inventory) / minimum inventory

(e) Determining Downgrade Amount - With the helper function called 'downgrade', the downgrade amount is calculated as the minimum value between excess inventory in source node and shortage in target node.

(f) Replenishment Process - Replenish the inventory from the first node in the ranked list.

(g) Expanding Range of Search - If the inventory in target node is not fully replenished after processing all source nodes within the specified range, the algorithm will expand the search range and repeat the replenishment process.

(h) Termination Conditions - Target node's inventory is fully replenished or when algorithm exhausts all suitable source nodes within the specified range.

Key Results

- **Cost Savings:** The HTML website enhances user experience with features like regular data updates, search capabilities, wafer information display, and inventory health insights, resulting in annual savings of approximately **USD 500,000** by optimising wafer downgrades and reducing wastage.

- **Improved Efficiency:** Automating downgrade path generation saves around **1,600 work hours** annually, streamlining processes for technicians and improving overall operational efficiency.

Recommendations

Predictive Analytics Integration: Implementing machine learning models to predict test wafer demands, aiming to reduce emergency procurement costs and inventory holding costs, thereby optimising stock levels and enhancing operational efficiency and customer satisfaction.

Customisable Alerts and Notifications: Launching a customisable alert system for real-time inventory oversight, designed to accelerate response times through threshold alerts and actionable recommendations, improving inventory management and overall operational efficiency by minimizing manual monitoring.

Migration to Angular Framework: Transitioning to the Angular framework to leverage its modular structure for scalable, complex application development, facilitating easier integration with systems and APIs, aimed at streamlining operations and future-proofing the platform.