Winter Internship Report

On

Chemical Process Flow Diagram (PFD) Desktop Application Development

Submitted by

Karan Y Yede

Under the guidance of

Prof. Prabhu Ramachandran
Department of Aerospace Engineering
IIT Bombay

Feb 20, 2026

Acknowledgment

I would like to express my sincere gratitude to the FOSSEE team at IIT Bombay for the incredible opportunity to be a part of the FOSSEE Winter Internship program. This experience has been both intellectually rewarding and personally inspiring. Working on the Chemical Process Flow Diagram (PFD) Desktop Application Development allowed me to dive deep into the process desktop application development. The internship opportunity I had with the FOSSEE Team was an excellent learning and professional development opportunity. I consider myself fortunate to have been given the chance to contribute to a project that directly supports engineering education across India.

I am using this opportunity to express my deepest gratitude to the FOSSEE team for providing me with this platform. I extend my sincere thanks to my Lead Developer, Chirag, for his continuous technical guidance, thorough code reviews, and mentorship throughout the internship. His feedback was instrumental in improving the quality of my work.

I am also grateful to my teammates Nitish and Parshv for their cooperation, collaborative spirit, and patience during integration of various modules. Working alongside them was both productive and enriching.

I perceive this internship as a significant milestone in my career development. The skills and knowledge I have gained here, both technical and professional, will continue to guide me in my future endeavours. I hope to continue contributing to open-source education projects in the future.

.

Contents

Chapter 1

Introduction

The FOSSEE (Free/Libre and Open-Source Software for Education) project at IIT Bombay, an initiative of the Ministry of Education, Government of India, is committed to promoting the widespread adoption of open-source software in aca-demic and research institutions. By reducing dependence on expensive proprietary tools, FOSSEE empowers individuals and institutions to explore free/libre alterna-tives that are equally capable and accessible.

1.1 Overview

This report documents the work carried out during a semester-long internship with FOSSEE, IIT Bombay, focused on the development of a desktop application for creating Chemical Process Flow Diagrams (PFDs). The internship task covered seven primary areas of contribution.

The first area involved implementing the functional component sidebar to display all available components, initially loading from local CSV files and later migrating to backend API integration. The second area covered the development of a multi-tab interface allowing users to work on multiple diagrams simultaneously. The third area involved creating a themed landing page with authentication, recent projects display, and logout functionality. The fourth area encompassed save and export functionalities supporting multiple file formats (.pfd, .pdf, .jpg, .xlsx). The fifth area focused on open functionality with a dedicated viewer supporting zoom for PDF and image files. The sixth area involved backend synchronization, fixing component fetching bugs, and resolving sync issues. The seventh area included additional contributions to canvas architecture, connection routing, and undo/redo systems.

1.2 Aim of this Internship

I participated in this internship to gain practical experience in building production-grade desktop software in a professional open-source environment. The internship allowed me to apply and extend my Python skills, learn the PyQt5 framework in depth, and understand how to architect a multi-module application that must communicate with a backend API. A key personal aim was to contribute meaningfully to a project that serves the chemical engineering education community in India.

1.3 About FOSSEE and its Aim

FOSSEE (Free/Libre and Open Source Software for Education) is a project under the National Mission on Education through ICT, funded by the Ministry of Education, Government of India and hosted at IIT Bombay. The primary aim of FOSSEE is to promote the use of Free and Open Source Software tools in educational institutions across India, reducing dependency on expensive proprietary software.

FOSSEE develops new FOSS tools and upgrades existing ones to meet academic and research requirements. The fellowship programme associated with FOSSEE introduces students to open-source contributions in various engineering fields, making them part of a large, active community of developers and educators.

1.4 About the Chemical-PFD Project

The Chemical-PFD Web-Desktop project is a cross-platform system designed for creating, editing, and managing Chemical Process Flow Diagrams. The project provides both a browser-based web application (built in React.js) and a native desktop application (built in PyQt5), both powered by a shared Django REST backend.

Process Flow Diagrams are fundamental documents in chemical engineering. They visually represent equipment connectivity (pumps, heat exchangers, reactors, valves), process workflows, and stream connectivity. Engineers use them to communicate design intent and to generate Bills of Materials for procurement.

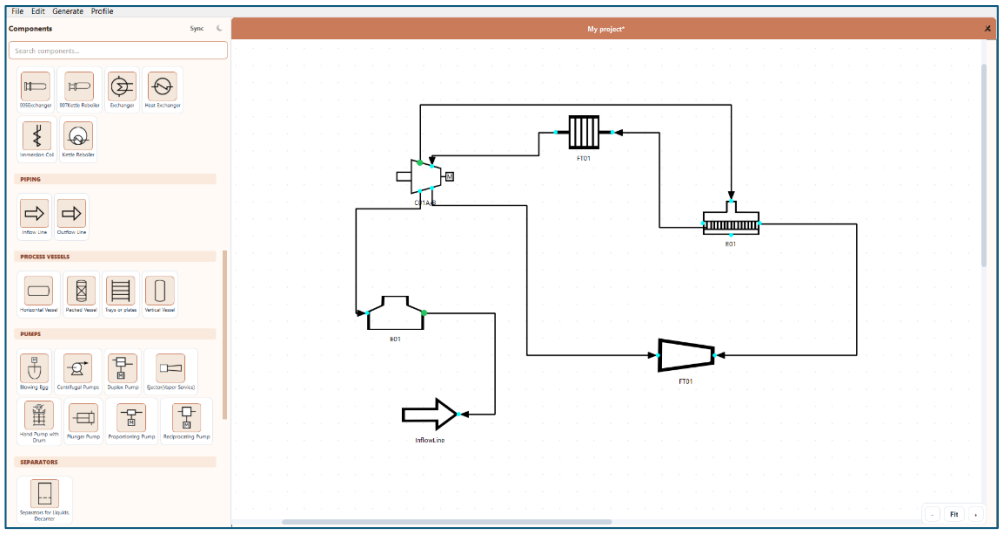

Figure 1.1: Chemical-PFD Desktop Application

1.5 Initial State of the Codebase

When I joined the project, the desktop application was at an early prototype stage. Component data was loaded from static local Excel and CSV files. There was no Save or Load functionality. The application only supported a single window. There were frequent cross-platform compatibility issues.

1.6 Technology Stack

The desktop application is built using the following technologies:

- Python 3.8+
- PyQt5 5.15+ – GUI framework
- Django REST API – Backend service
- Pandas and OpenPyXL – Excel export
- PyMuPDF – PDF handling
- ReportLab – PDF report generation
- Requests – HTTP client
- JSON – File serialization

Chapter 2

Component Sidebar Implementation

2.1 About the Component Sidebar Task

One of the first major features I implemented was the functional component sidebar that displays all available equipment components that users can drag onto the canvas. This sidebar serves as the primary interface for accessing the component library.

2.2 Initial CSV-Based Component Loading

Learning Objectives

- Load component data from CSV files using Pandas
- Parse component metadata (name, category, file paths)
- Display components in a scrollable sidebar with thumbnails

Task

In the initial implementation, I used Pandas to read the Component_Details.csv file which contained columns for component name, category, SVG file path, PNG thumbnail path, and description. The sidebar was implemented as a QScrollArea containing a grid layout of component widgets. Each component widget displayed the PNG thumbnail and component name.

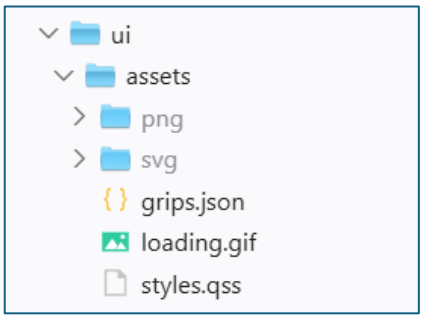

Figure 2.1: Initial component sidebar with CSV data

The sidebar was organized by categories (Pumps, Heat Exchangers, Reactors, Vessels, Valves). Users could expand or collapse categories. The implementation included search functionality allowing users to filter components by name in real-time.

2.3 Migration to Backend API

Learning Objectives

- Replace static CSV with dynamic API endpoint
- Implement automatic asset downloading
- Cache API responses locally

Task

After the backend API was ready, I refactored the component loading system to fetch data from the /api/components/ endpoint. The migration involved creating an API client module, implementing a caching mechanism, and adding automatic asset downloading for SVG and PNG files.

The new system checks for a local cache file on startup. If the cache exists, components load from cache immediately. A background sync operation then checks the backend for updates. New components are marked with a 'NEW' badge.

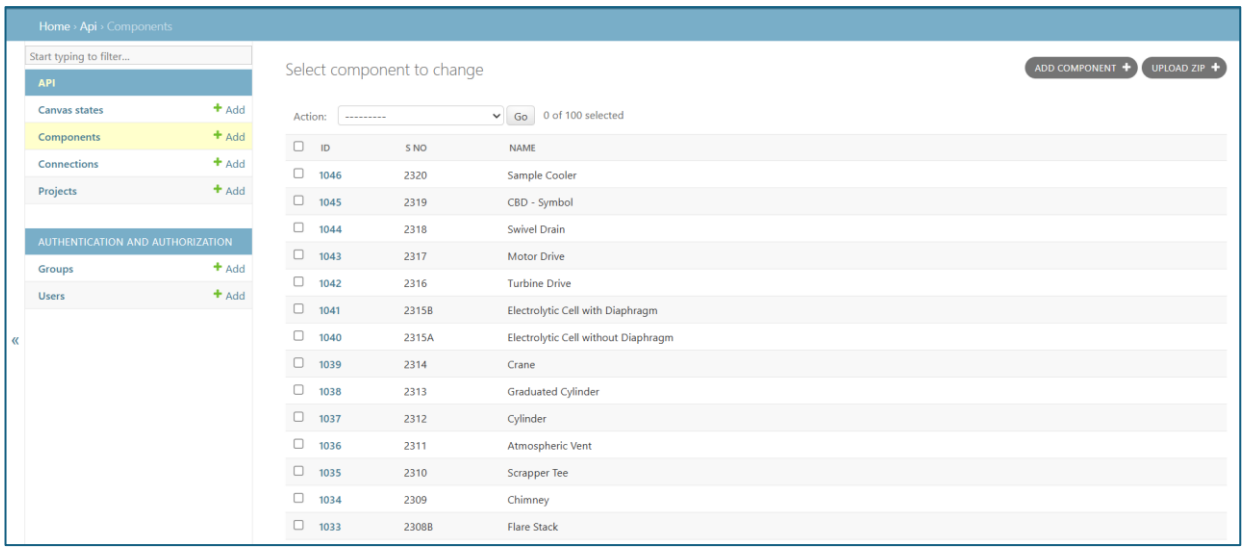

Figure 2.2: Component API integration

2.4 Component Display and Organization

Learning Objectives

- Design an intuitive component browsing interface
- Implement category-based organization with search
- Support drag-and-drop from sidebar to canvas

Task

The component display system uses a flow layout. Each component is represented by a card widget containing the PNG thumbnail, component name, and category label. The search bar filters components in real-time. A ‘Sync’ button triggers manual refresh from the backend.

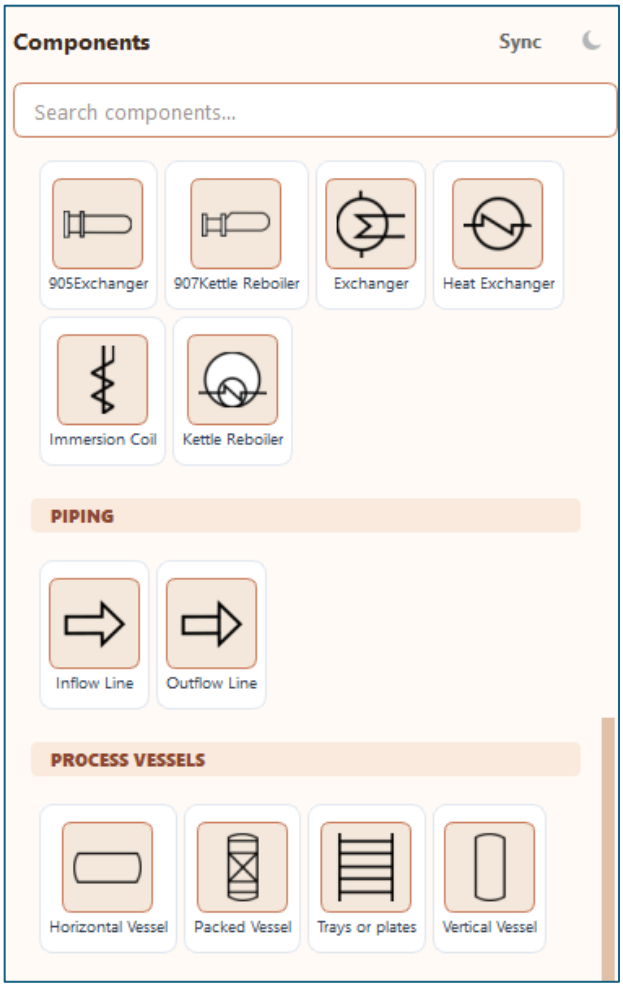

Figure 2.3: Component sidebar search and filter

Chapter 3

Multi-Tab Interface

3.1 About the Multi-Tab Feature

The original prototype was a single-window application. I implemented a multi-tab interface that allows users to open multiple projects in separate tabs within the same application window.

3.2 Tab Management System

Learning Objectives

- Implement a QTabWidget-based multi-document interface
- Manage independent canvas states for each tab
- Handle tab creation, switching, and closing

Task

The multi-tab system is built using PyQt5's QTabWidget. Each tab contains its own independent CanvasWidget with a separate undo stack. When a user clicks 'New Project', a new tab is created. When they click 'Open Project', the file loads into a new tab.

Tab switching preserves the state of each canvas including zoom level and scroll position. Each tab displays its project name with an asterisk (*) if there are unsaved changes. The close button triggers a check for unsaved changes.

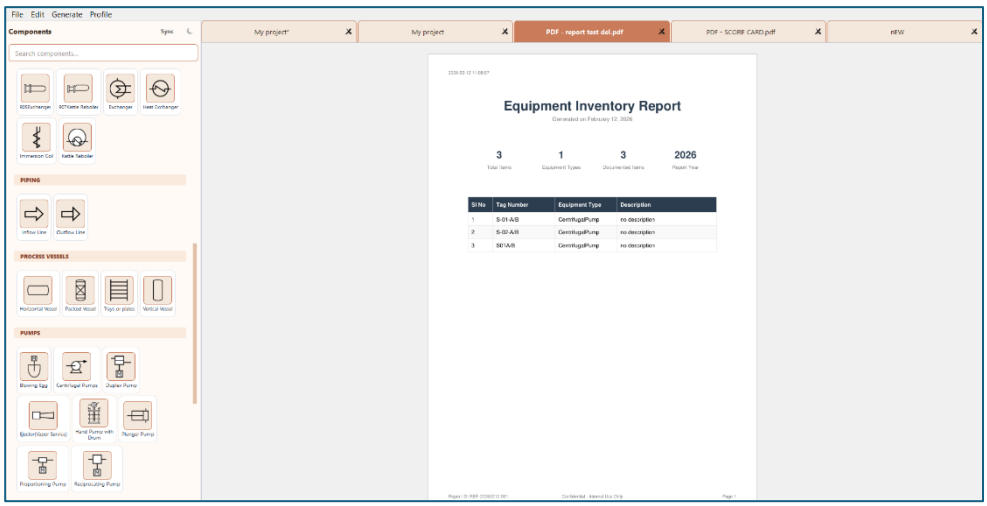

Figure 3.1: Multi-tab interface

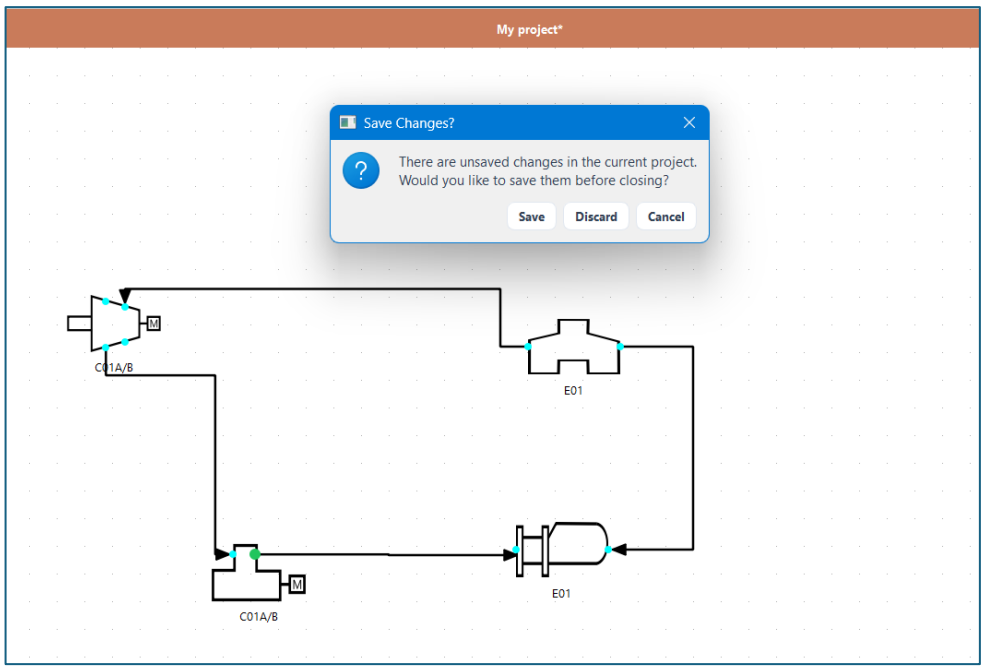

Figure 3.2: Unsaved changes warning

The implementation includes keyboard shortcuts (Ctrl+Tab, Ctrl+Shift+Tab) and a tab context menu with Close Tab, Close Other Tabs, and Close All Tabs options.

Chapter 4

Landing Page and Authentication

4.1 About the Landing Page Task

To improve the user experience, I created a dedicated landing page that appears after successful login. The landing page displays recent projects, provides quick access to create new projects, and offers account management options.

4.2 Landing Page Design and Theme Support

Learning Objectives

- Design a landing page matching the application theme
- Implement light and dark theme support
- Create a responsive layout

Task

The landing page is implemented as a custom QWidget with a header section, recent projects section, and action buttons. The header displays the logo, welcome message with the user's name, and current date.

Both light and dark themes are fully supported. Theme switching dynamically updates all colors using Qt stylesheets. The light theme uses a white background with blue accents, while the dark theme uses dark gray with cyan accents.

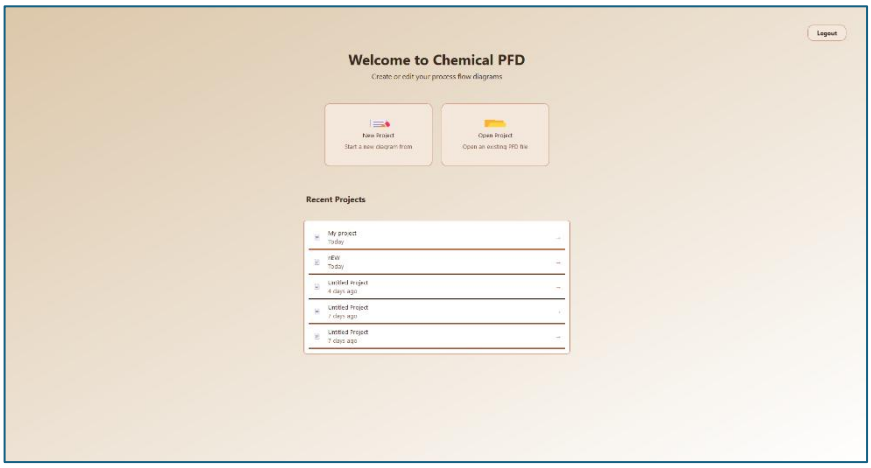

Figure 4.1: Landing page – light theme

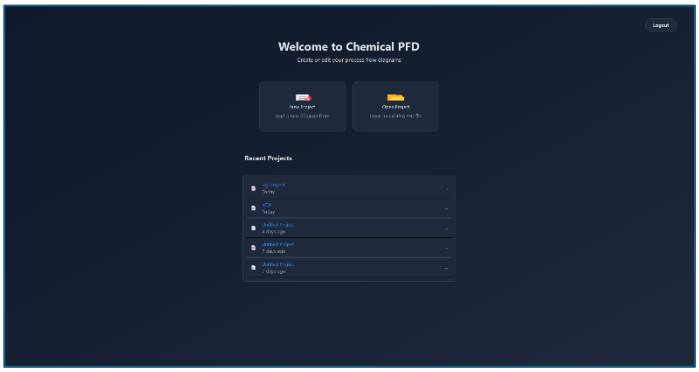

Figure 4.2: Landing page – dark theme

4.3 Recent Projects Display

Learning Objectives

- Display recently opened projects with thumbnails
- Implement quick-open functionality
- Integrate logout functionality

Task

The recent projects section displays up to 6 recently opened projects in a grid layout. Each project card shows a thumbnail, project name, last modified date, and file size. Clicking a card opens that project in a new tab.

The recent projects list persisted in recent projects.json. The list updates automatically when projects are opened or saved. If a project file no longer exists, it is removed from the list.

A 'New Project' button navigates to the canvas and creates a blank project. A 'Logout' button allows users to log out and return to the login screen.

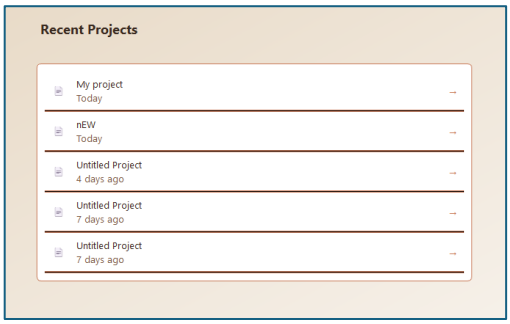

Figure 4.3: Recent projects display

Chapter 5

File Management - Save and Export

5.1 About the File Management Task

I implemented comprehensive file management supporting save and export operations for multiple file formats including .pfd (native project format), .pdf (diagram export), .jpg (image export), and Excel (.xlsx) for equipment lists.

5.2 Save Functionality

Learning Objectives

- Implement Save and Save As operations
- Serialize canvas state to JSON
- Integrate with undo/redo system

Task

The save functionality serializes the complete canvas state including all components with their properties, all connections with routing information, canvas viewport settings, and project metadata.

The Save operation (Ctrl+S) writes to the current file path. Save As (Ctrl+Shift+S) opens a file dialog for a new filename. The .pfd format is JSON-based and human-readable with a version field for compatibility.

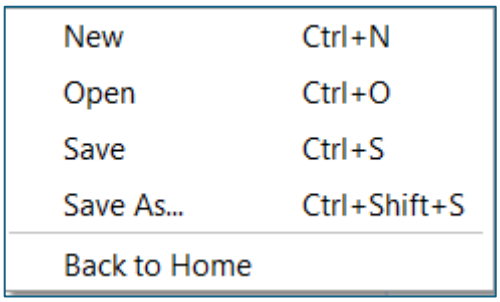

Figure 5.1: Save and Save As menu in NavBar

5.3 Export Functionality

Learning Objectives

- Export canvas as high-resolution PDF
- Export canvas as JPEG image
- Generate Excel equipment list

Task

The export functionality provides three output formats. PDF Export renders the canvas at 4x scale to ensure sharp quality. The page size is calculated from the diagram bounding box.

JPEG Export captures the canvas as a raster image with user-specified resolution and quality. Excel Export generates a Bill of Materials with tag numbers, types, and descriptions using Pandas.

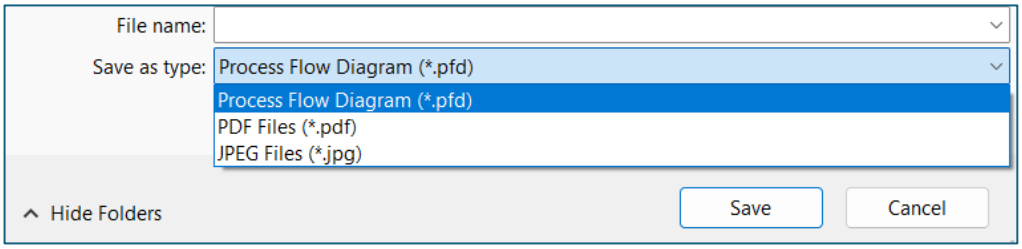

Figure 5.2: Export options

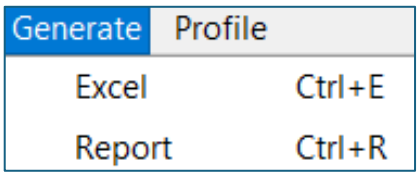

Figure 5.3: Generate Report and Excel Menu

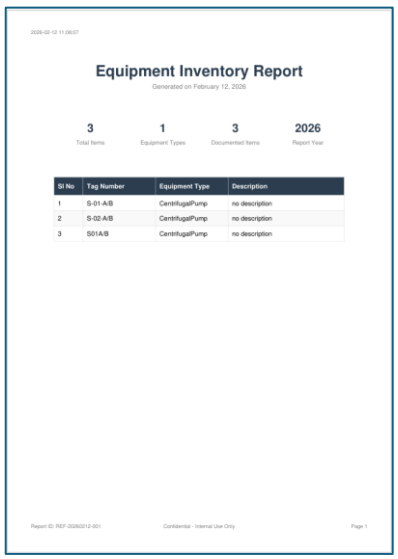

Figure 5.4: Pdf Export

Chapter 6

File Management - Open Functionality

6.1 Open Functionality with Viewer

Learning Objectives

- Implement Open operation for .pfd, .pdf, and .jpg files
- Deserialize .pfd files and reconstruct canvas
- Create dedicated viewer for PDF and images

Task

The Open functionality supports three file types. For .pfd files, the JSON data is parsed and the canvas is reconstructed by creating component widgets and connections according to stored data.

For .pdf and .jpg files, instead of loading them onto the canvas, I implemented a separate viewer window. The viewer displays the file using QGraphicsView with pan and zoom capabilities. Users can reference external diagrams while working on their projects.

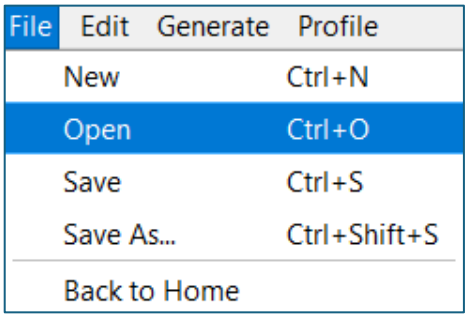

Figure 6.1: Open file menu

6.2 Zoom Support for PDF and Images

Learning Objectives

- Add zoom controls to the viewer window
- Implement pan functionality
- Support multiple zoom levels

Chapter 7

Backend Synchronization and Bug Fixes

7.1 Component Fetching from Backend

Learning Objectives

- Implement robust API communication
- Handle network errors gracefully
- Validate component data from API

Task

The component fetching system makes HTTP GET requests to /api/components/ with JWT authentication. The response is parsed and validated to ensure required fields are present.

Initially there were bugs including missing error handling, incorrect URL construction, and timeout issues. I fixed these by adding try-except blocks, implementing timeouts (10s for list, 30s for downloads), and validating responses.

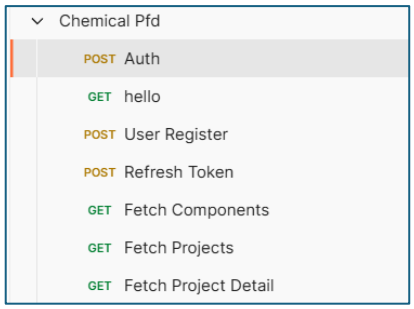

Figure 7.1: Component sync from backend

7.2 Component Sync Issues Resolution

Learning Objectives

- Debug component sync failures
- Implement incremental sync
- Add detailed logging

Task

During testing, several sync issues were discovered including duplicate components, assets not downloading, UI freezing during sync, and 'NEW' badge not appearing.

I resolved these by implementing ID-based deduplication, adding file existence checks, moving sync to a background QThread, and implementing timestamp-based comparison for new components.

I also added detailed logging to sync.log with timestamps, HTTP status codes, and error messages.

7.3 Bug Fixes and Optimizations

Learning Objectives

- Identify and fix critical bugs
- Optimize performance bottlenecks
- Improve error handling

Task

Major bug fixes included cross-platform file path issues, memory leaks in tab closing, coordinate transformation errors during zoom, and race conditions in async loading.

Performance optimizations included caching grip definitions, using QPixmapCache for thumbnails, lazy loading for component SVGs, and optimizing canvas repaint regions.

Chapter 8

Useful Links

- FOSSEE Project: https://fossee.in/
- Chemical-PFD Repository: https://github.com/FOSSEE/Chemical-PFD-Web-Desktop
- My Pull Requests: https://github.com/FOSSEE/Chemical-PFD-Web-Desktop/issues?q=is%3Apr+author%3A%40karanyede
- PyQt5 Documentation: https://doc.qt.io/qt-5/

References

- FOSSEE – https://fossee.in/
- Chemical-PFD Web-Desktop – https://github.com/FOSSEE/Chemical-PFD-Web-Desktop
- PyQt5 Reference Guide – https://doc.qt.io/qt-5/
- Django REST Framework – https://www.django-rest-framework.org/
- Pandas Documentation – https://pandas.pydata.org/docs/