

# Information Technology — Control Point Documentation (2025)

Knight Foundation School of Computing and Information Sciences (KFSCIS) — Florida International University

Instructor: Masoud Sadjadi | Version: 2025 Fall | License: MIT (include in your repo)

Poster: 36"×48" horizontal • Videos: Intro & Demo • Presentation: 10–15 min • Scrum: disciplined, evidence-based.

## ACM Student Outcomes (O1–O6)

| Outcome                                      | Description                                                                                        |
|----------------------------------------------|----------------------------------------------------------------------------------------------------|
| O1. Problem Analysis & Requirements          | Elicit and analyze user/stakeholder needs; define constraints and success criteria.                |
| O2. System Design & Implementation           | Design solutions using appropriate abstractions, patterns, and tools; implement maintainable code. |
| O3. Experimentation, Testing & Evaluation    | Devise evaluation methods; collect evidence; interpret results to improve the system.              |
| O4. Communication & Collaboration            | Communicate effectively with technical and non-technical stakeholders; collaborate in teams.       |
| O5. Professional, Ethical, Legal & Societal  | Recognize responsibilities; address ethics, security, privacy, accessibility, and sustainability.  |
| O6. Systems Integration, DevOps & Operations | Discipline-specific depth outcome; addressed in dedicated sections below.                          |

## Outcome & Rubric Crosswalk (Checklist)

Use this to ensure your documentation and artifacts demonstrate each outcome and satisfy the rubric. Mark each ☐ when complete.

| Outcome | Where It Appears in This Document             | External Evidence (Links/Appendices)              | Status ( <input type="checkbox"/> / <input checked="" type="checkbox"/> ) |
|---------|-----------------------------------------------|---------------------------------------------------|---------------------------------------------------------------------------|
| O1      | §2 Problem & Requirements; §3 System Overview | Appendix A (User Research), Backlog, User Stories | <input checked="" type="checkbox"/>                                       |

|    |                                                            |                                                |   |
|----|------------------------------------------------------------|------------------------------------------------|---|
| O2 | §3 System Overview & Architecture; §5 Implementation Notes | Repo structure, Code, CI status badge          | ☑ |
| O3 | §6 Testing & Evaluation                                    | Test reports, coverage, performance dashboards | ☑ |
| O4 | §1 Executive Summary; §9 Limitations & Future Work         | Poster, Slides, Videos                         | ☑ |
| O5 | §7 Security/Privacy/Compliance                             | Threat model / Model card / SBOM               | ☑ |
| O6 | §4 Discipline-Specific Depth                               | Artifacts noted in §4                          | ☑ |

# 1. Executive Summary

## The Problem

Organizations often struggle to track and manage their physical and digital assets across departments, vendors, and locations. Spreadsheets become outdated, financial visibility is limited, and administrators lack an efficient way to monitor assignments, maintenance, disposals, and lifecycle data. This creates inefficiencies, increases costs, and makes audits or planning more difficult.

## Who Benefits

Control Point is designed for businesses, schools, IT departments, and administrative teams responsible for managing assets and employee equipment. Both everyday staff and system administrators benefit. Staff can view and manage their assigned items, while administrators gain complete oversight of organizational assets, costs, users, and supporting reference data.

## Solution Approach

Control Point provides a centralized, SQL-backed web application with a clean, structured interface for managing assets. The dashboard displays real-time visualizations—including Assets by Status, Vendor, and Type—along with cost metrics for admin users. Users can log in to manage assets, employees, assignments, and supporting data through sortable tables, search tools, and CSV import options.

The system organizes related features under intuitive sections: Assets, Employees, Assignments, and an Advanced menu for Maintenance, Disposals, Asset Types, Statuses, Locations, Departments, and Vendors. Each page provides consistent patterns for listing, searching, sorting, adding, and editing data, reducing the learning curve and supporting smooth operations.

## Key Results

Control Point improves asset visibility, standardizes data management, and enhances operational efficiency. Administrators gain clear insight into total and average asset cost, lifecycle processes, and departmental usage. Staff gain a reliable, user-friendly way to access and manage asset data. Overall, the system reduces manual workload, prevents data loss or inconsistency, and provides a scalable foundation for asset tracking across an organization.

# 2. Problem & Requirements (O1)

**Context:** Companies rely on accurate information about their assets, vendors, and assignments to employees. However, many organizations still depend on spreadsheets or other outdated tools, which often lead to inaccurate records. When administrators are unaware of assets that are missing, broken, or in need of restocking, business can slow causing loss of revenue. Control Point was created to provide an intuitive, centralized solution to asset management, ensuring organizations maintain accurate, real-time oversight of their assets.

**Stakeholders:** IT Administrators, Department Managers, Employees

**Personas:**

- Admin User
  - Responsibilities: Manage assets, users, assignments, maintenance, and configuration
  - Needs: Full access to cost metrics, users, and assets
- Standard User
  - Responsibilities: View of assets
  - Needs: Navigation tools, access to their assignments, ability to sort and search

**Functional requirements:** Authentication, role-based access, ability to create, edit, delete, assign, search, sort, import and export data, navigation between pages

**Non-functional requirements:** Consistent UI, intuitive design, fast and reliable

**Constraints:** Linux based, installation necessary, limited time, team size, and expertise

**Success criteria:** Users are able to create, edit, delete users, assets, vendors, locations etc., navigation works properly, graphs update in real time and remain interactive, permissions prevent unauthorized access

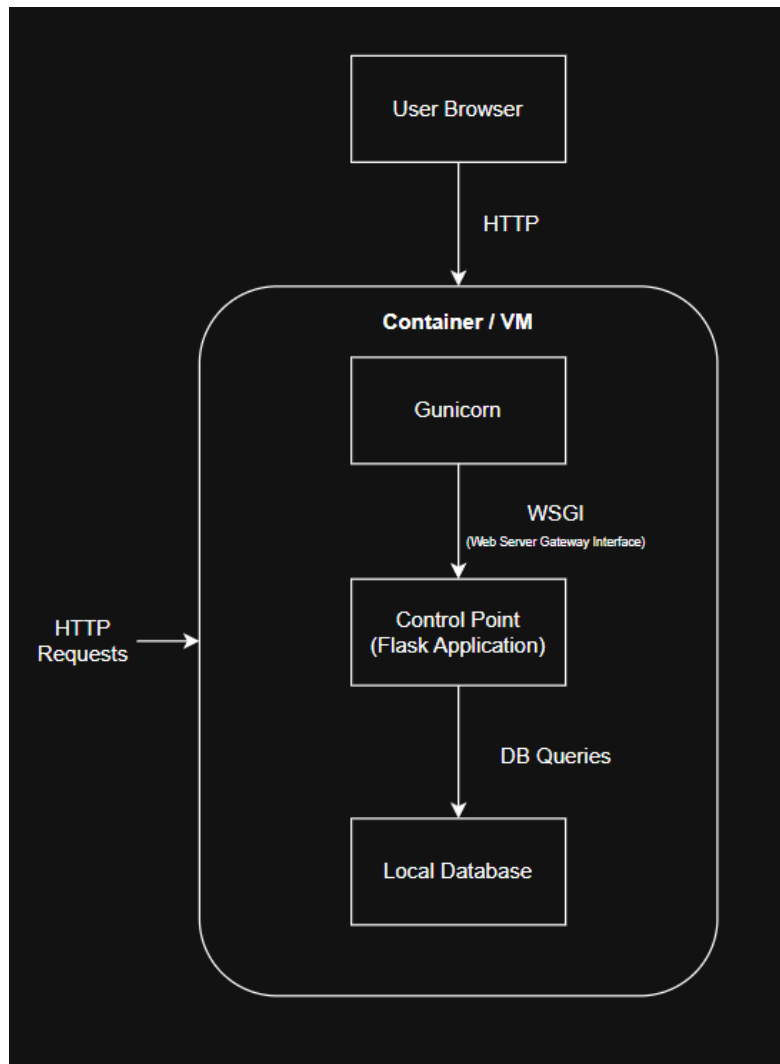
**Risks:** Data corruption, unaccounted security vulnerabilities, human error

Include a prioritized backlog (top 10–15 items) and a link to your full backlog board.

[Backlog Board](#)

### 3. System Overview & Architecture (O2)

Describe the overall architecture. Paste a current architecture diagram (C4 containers/components or equivalent).



List key technologies, frameworks, and services. Include API surface (OpenAPI/GraphQL) and data/storage overview.

- Python
- Flask
- Chart.js
- HTML
- CSS
- Werkzeug
- SQLAlchemy
- MariaDB
- SQL
- Debian Linux
- Bootstrap

#### 4. Discipline-Specific Depth (O6)

- **Architecture & Integration:** network topology, services, and platform choices (cloud/on-prem, containers).

MariaDB database and Flask applications run on the same machine and communicate using the loopback address and a preconfigured DB administrator account. MariaDB runs on its own service that is created on installation while the Webapp runs on a gunicorn system service that is created during the application installation.

Since this is a simple .sh installation script, the server can be installed on any Debian based Linux distribution. When installed, the machine will act as a web server with port 80 being opened on its firewall. The intended deployment is on a container located on a server on-site. It can also be deployed on the cloud as we have done for our demonstration which utilizes a minimal VM running on Google Cloud.

- **DevOps & SRE: CI/CD, IaC snippets, observability (logs/metrics/traces), SLOs/SLIs and error budget policy.**

CI/CD is handled through GitHub tags and a single field on the installation file which specifies which version to install.

- **Operations & Security:** access management, backup/restore, change management, configuration hardening.

RBAC is handled within Control Point with three role options: user, manager, and admin. The setup file installs UFW and only opens necessary ports (80). Unless the container/VM is port forwarded, it is only accessible from a local network.

There are currently only manual backups available to admins. An admin has the ability to take a full DB download and can restore any portion easily through the CSV upload feature.

Version control is handled through a GitHub repository. To update the app, all one needs to do is check the most recent tag on GitHub and replace the existing version number with the new one in the installation file. Running it again will get the relevant files and replace the current installation without changing the database.

## 5. Implementation Notes (O2)

Repository structure, coding conventions, notable patterns, and tradeoffs. Reference the README and module layout.

## Repo structure:

ControlPoint/

```
| -AssetManagment/ # Contains blueprints and logic for pages related to assets
```

```
| -asset assignments.py
```

```
| -asset disposals.py
```

```
| -asset_maintenance.py
| -asset_statuses.py
| -asset_types.py
| -assets.py
| -departments.py
| -employees.py
| -locations.py
| -vendors.py
|
|-MiscPages/                                # Contains blueprints and logic for miscellaneous pages
|
|   |-login.py
|   |-manage_users.py
|
|-Sample Data/                              # Contains CSVs with sample data for demonstrations
|
|-Templates/                                # Contains all HTML files for pages
|
|   |-asset_assignments.html
|   |-asset_disposals.html
|   |-asset_maintenance.html
|   |-asset_status.html
|   |-asset_type.html
|   |-assets.html
|   |-base.html                            # Main navigation bar
|   |-departments.html
|   |-employees.html
|   |-index.html                          # Home page
|   |-locations.html
|   |-login.html
|   |-subnav.html                         # Secondary navigation bar for 'manage assets' pages
```

```

|-users.html
|-vendors.html
|
|-app.py                # Main function
|-misc_fuctions.py      # Contains logic for exporting CSVs
|-models.py             # Contains DB models for SQLAlchemy
|-requirements.txt      # Contains list of packages needed during installation
|-route_decorators.py   # Contains optional fields for blueprints to allow for RBAC

```

### **Coding Conventions / Patterns / Tradeoffs:**

Python version 3.13 with all functions (not associated with specific pages) on the root level. All page logic is contained within labeled folders (MiscPages / AssetManagement) with all HTML files contained in the Templates folder. All files use '\_' instead of spaces and are in lowercase. Database models live entirely in models.py and homepage content is contained in app.py. All individual page logic is constrained to blueprints within their own files labeled with the table of the DB they display.

This tradeoff allows for a simple file structure to navigate with minimal folder clutter. Unfortunately, this has led to some file clutter within the 'Templates' folder. This is mitigated though through clear naming with associated .html and .py using the names of the table they reference in the DB. These similar names based on the tables can become confusing as the blueprints also follow this naming scheme leading to routes such as 'asset.assets'

## **6. Testing & Evaluation (O3)**

Strategy (unit/integration/e2e), coverage, performance and reliability testing; KPIs/metrics; defect summary.

Testing was mostly limited to verifying functionality and preventing major security flaws. Unit testing was used to validate core functions like route responses and utility functions and mainly focused on verifying critical logic in general use. Edge cases were mostly ignored. Integration testing was done through multiple deployments across locally and cloud hosted VMs and containers. Manual tests were performed to verify correct reads/writes to the DB from multiple machines across different networks. Coverage was limited with only manual function testing being performed. Performance validation was informal with the most extensive testing being uploaded several hundred records to the DB in a single operation. Application hung for about a second before operating normally. Could be related to the system resources being extremely limited. Reliability testing was also limited but achieved through running multiple instances across multiple environments (local/cloud) for several days without stopping. This verified that no memory leaks or other bugs that occur within a reasonable amount of time were present.

## **7. Security, Privacy, Accessibility & Compliance (O5)**

Control point is designed to be a secure and reliable way for an organization to manage their assets. The system only stores non-sensitive PII (personally identifiable information) which includes names, phone numbers, email, and business address. It does not store sensitive PII including social security numbers, driver's license, banking information etc. Even though non-sensitive data is collected, Control Point was designed to have proper authentication steps, role based permissions, and was tested against SQL injection methods. Control Point's interface is very simple and intuitive to use, featuring clear navigation, sortable tables, and readable text, making it accessible for users with varying levels of technology proficiency. Due to Control Point being browser-based, it also supports the use of screen readers and other accessibility tools.

## 8. Deployment & Operations

Users are able to log into the application and view and sort all assets, employees, and assignments. Administrators have access to additional tools such as creating, editing, and deleting assets and users. A more thorough explanation is linked below.

[User Manual.docx](#)

## 9. Limitations & Future Work

Known limitations, shortcuts, technical debt, and prioritized roadmap items for future iterations.

### Limitations/Results of shortcuts:

- Assets table does not check for assignments when uploading CSVs and can display incorrect information
- Assets table does not update to reflect disposal or maintenance entries
- App does not have the ability to show past records if an entry is changed
- If app installation file is run multiple times, default admin user can be removed
- No uninstall feature was created
- Database is on the same machine as the webserver and included with installation, which could cause security issues. Done for the sake of simplicity
- Only employees and Assets tables can be downloaded individually
- Sample data for showcases and demos is sometimes nonsensical (assigning an asset type of 'laptop' to a desk) as much of it was generated using algorithms automatically to save time
- Some dashboard information is visible when logged out potentially showing confidential information
- CSV uploading does not detect conflicting values in certain fields
- Employees table and department tables reference each other with no checks when creating/uploading data to check for either
- Application does not support selecting multiple entries

### Shortcuts:

- Relying almost entirely on DB functions to make sure data is correctly deleted
- Widening the screen space to allow asset information to correctly display
- Relying almost entirely on unmodified Bootstrap styling to style app
- Having departments and employees tables reference each other
- Used MariaDB instead of more widely used DB software for simplicity



### Prioritized Roadmap (highest to lowest priority):

- Restrict dashboard information to only those logged in
- Create a managers table to break circular relationship between departments and employees
- Implement some backend logic to correct DB operations
- Implement a feature to store past records for future reference
- Implement a function to select multiple data entries at once
- Implement CSV exporting to each individual table
- Implement backend logic to parse CSV data and compare it to referenced tables where appropriate to correct DB errors stemming from import issues
- Separate the DB from the host server and create separate installation files.
- Fix potential errors resulting from running installation multiple times
- Create uninstall file(s)
- Create custom styling for the application
- Correct included sample data

## 10. References

- [1] W3Schools, "Python Tutorial," *W3schools.com*, 2019. <https://www.w3schools.com/python/default.asp>
- [2] W3Schools, "HTML Tutorial," *W3schools.com*, 2022. <https://www.w3schools.com/html/default.asp>
- [3] Flask, "Welcome to Flask — Flask Documentation (3.0.x)," *Palletsprojects.com*, 2010. <https://flask.palletsprojects.com/en/stable/>
- [4] MariaDB, "MariaDB Documentation | MariaDB Documentation," *Mariadb.com*, Jun. 13, 2025. <https://mariadb.com/docs>
- [5] SQLAlchemy, "SQLAlchemy Documentation — SQLAlchemy 2.0 Documentation," *docs.sqlalchemy.org*, 2025. <https://docs.sqlalchemy.org/en/20/>
- [6] chart.js, "Chart.js | Open source HTML5 Charts for your website," *Chartjs.org*, 2019. <https://www.chartjs.org/>
- [7] Bootstrap, "Get started with Bootstrap," *getbootstrap.com*, 2023. <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
- [8] Werkzeug, "Werkzeug — Werkzeug Documentation (3.1.x)," *Palletsprojects.com*, 2024. <https://werkzeug.palletsprojects.com/en/stable/>
- [9] Canva, "Canva," *Canva*, 2025. <https://www.canva.com/>

# Appendices (Evidence & How-To)

## A. Installation Guide

\*CONTROL POINT IS ONLY OFFICIALLY TESTED ON DEBIAN LINUX

Run these commands in this order as super user or use sudo:

```
chmod +x ControlPoint.sh
```

```
./ControlPoint.sh
```

Enter the desired administrator account password or leave it empty to use the default

[README](#)

## B. User Manual and Admin/Operations Guide

[User Manual](#)

## C. Poster (36"×48" horizontal), Slides, and Video Links (Intro, Demo)

[Poster](#)

[Slides](#)

[Demonstration video](#)

[Introduction video](#)

## D. Scrum Evidence Index (links to Sprint Planning, Dailies, Reviews, Retros)

[Scrum Evidence](#)