
VCU College of Engineering

# CS 25-329 CCAC Service Catalog

## Preliminary Design Report

Prepared for

Professor Robert Dahlberg

Commonwealth Center for Advanced Computing (CCAC)

By

Akhil Manoj

Bilal Othman

Nick Arthur

Sage Walker

Under the supervision of

Professor Robert Dahlberg

October 1, 2024

## Executive Summary

Four senior computer science students at Virginia Commonwealth University are undertaking a critical project for the Commonwealth Center for Advanced Computing (CCAC) to develop a comprehensive service catalog of pre-configured server images. This initiative aims to streamline CCAC's service delivery capabilities, enabling their customers, primarily Virginia-based universities, to easily provision servers tailored to their specific educational and research needs. The project encompasses identifying stakeholder requirements for various server types (including AI, web, virtual/augmented reality, database, and application servers), ensuring compatibility with IBM z/16 mainframe, IBM Power10 and x86 server architectures, and creating and configuring golden images.

Key deliverables include detailed server specifications, fully configured golden images, and assisting the CCAC team in the implementation of the service catalog system. The team will work closely with CCAC stakeholders throughout the academic year to ensure the final product meets the needs of participating universities, including VCU, UVA, GMU, VSU, JMU, VT, ODU, and Longwood University. This project represents a significant advancement in CCAC's ability to support higher education institutions across Virginia, streamlining access to critical computing resources and fostering innovation in research and education

# Table of Contents

## Section A. Problem Statement

The Commonwealth Center for Advanced Computing (CCAC) faces a significant challenge in efficiently providing its powerful IBM z/16 mainframe, Power10, and x86 server resources to its partners, primarily universities and educational institutions. Despite possessing these advanced computing systems, CCAC currently lacks a streamlined method for users to access and utilize these resources effectively. This project aims to address this issue by developing a comprehensive service catalog that will enable users to provision servers tailored to their specific needs, complete with pre-installed software packages and configurations.

The field of high-performance computing (HPC) is rapidly evolving, with educational institutions increasingly relying on advanced computing resources for research, data analysis, and complex simulations [1]. However, the gap between possessing powerful hardware and making it easily accessible to end-users remains a common challenge in the industry. This project seeks to bridge this gap by creating a user-friendly service catalog system, similar in concept to Amazon Web Services' Lightsail [2], but specifically tailored for CCAC's IBM z/16, Power10, and x86 infrastructure. These institutions stand to benefit significantly from improved access to high-performance computing resources, potentially accelerating research and enhancing educational opportunities in fields requiring substantial computational power [3].

The project addresses several unmet engineering needs, including identifying server types, determining appropriate software packages, investigating compatibility with IBM architectures, and exploring automation techniques like Ansible. The stakeholders include CCAC as the primary client and sponsor, along with its partner universities across Virginia. These institutions stand to benefit significantly from improved access to high-performance computing resources, potentially accelerating research and enhancing educational opportunities in fields requiring substantial computational power.

By addressing these challenges, this project aims to significantly enhance the accessibility and utilization of CCAC's advanced computing resources. The successful implementation of this service catalog will not only improve operational efficiency for CCAC but also potentially accelerate research and innovation across various scientific disciplines in partner institutions.

## Section B. Engineering Design Requirements

The primary goal of this project is to develop a comprehensive service catalog for the Commonwealth Center for Advanced Computing (CCAC), enabling efficient provisioning of virtual servers tailored to various computing needs. The project aims to identify and define a range of server types, including database, application, web, AI, development environment, virtual reality, augmented reality, and natural language processing servers. Each server type will be configured as a "golden image" that users can easily deploy and customize within their CCAC account environment.

Key design requirements include identifying the specific software components for each server type, determining the necessary CPU, RAM, and storage specifications, and establishing compatibility with CCAC's existing infrastructure, particularly the IBM z/16 mainframe and Power10 servers15. The project also requires familiarity with Ansible for server provisioning automation2. Additionally, the team must research and adapt existing Ansible scripts from public or vendor-supported GitHub repositories to streamline the deployment process12. These requirements were established through consultation with CCAC stakeholders and analysis of current high-performance computing trends in academic environments34. The design must prioritize flexibility, scalability, and ease of use to accommodate the diverse needs of CCAC's partner universities while ensuring efficient utilization of CCAC's advanced computing resources

### B.1 Project Goals (i.e. Client Needs)

The Commonwealth Center for Advanced Computing (CCAC) has identified several key goals for this project, aimed at enhancing their ability to provide advanced computing resources to partner institutions. These goals reflect the client's needs for improved accessibility, efficiency, and utilization of their high-performance computing infrastructure. The primary project goals are:

• To develop a comprehensive service catalog that enables easy provisioning of pre-configured virtual servers tailored to various computing needs.

• To optimize the utilization of CCAC's IBM z/16 mainframe and Power10 servers by creating a diverse range of "golden image" server configurations.

• To streamline the process of server deployment and customization for CCAC's partner universities, enhancing their research and educational capabilities in advanced computing fields.

## B.2 Design Objectives

The design objectives for the CCAC service catalog project are as follows:

- The design will identify and define at least seven distinct types of virtual servers (e.g., database, application, web, AI, development environment, virtual/augmented reality, and natural language processing) by the end of the first project phase.
- The design will specify the software components, CPU, RAM, and storage requirements for each server type within the first two months of the project.
- The design will identify which operating systems each server can run on based on its software requirements and compatibility with IBM z/16, Power10, and x86 architectures
- The design will research Ansible and its use case for automating the provisioning of each server by the end of the semester
- The design will incorporate Ansible scripts for automated provisioning of each server type, adapting existing scripts or creating new ones as needed, by the first half of the second semester.
- The design will create a documented process for CCAC staff to update and maintain the service catalog, including procedures for adding new server types or updating existing ones, by the project's completion date.

## B.3 Design Specifications and Constraints

Based on the project requirements and objectives, the following design specifications and constraints have been identified:

- Design must be compatible with IBM z/16 mainframe, Power10, or x86 server architectures, ensuring all server images can run efficiently on CCAC's existing infrastructure.
- Design must include server configurations that can be provisioned with a maximum of 30 minutes of user interaction time, from selection to deployment.
- Design must incorporate Ansible scripts for automated provisioning, with each script thoroughly documented and tested for reliability.
- Design must ensure that all software included in the server images complies with licensing agreements and educational use policies of the respective vendors.
- Design must include a maintenance plan that allows for quarterly updates of software packages and security patches without disrupting active user sessions.

**B.4 Codes and Standards**

The **CCAC Service Catalog** project involves provisioning virtual servers on IBM z/16, Power10, and x86 architectures, using automation tools like Ansible. Below are the relevant codes and standards that should be followed:

**1. IEEE Standards (Institute of Electrical and Electronics Engineers)**

- **IEEE 1012-2016** (Standard for System and Software Verification and Validation):
    - Ensures that the server configurations and Ansible automation scripts undergo proper verification and validation processes before deployment. This guarantees the reliability of the final product.
- **IEEE 14764-2006** (Software Configuration Management):
    - Establishes guidelines for the configuration management of server images, ensuring that any changes to the golden images are traceable, documented, and maintained systematically.

**2. ISO Standards (International Organization for Standardization)**

- **ISO/IEC 27001:2013** (Information Security Management Systems):
    - Since the CCAC involves handling sensitive research data and academic resources, compliance with this standard ensures that information security protocols are maintained for the virtual server environments and data processing activities.
- **ISO/IEC 25010:2011** (System and Software Quality Models):
    - Defines quality characteristics such as performance efficiency, compatibility, usability, and maintainability for the virtual server images and the automation systems.
- **ISO/IEC 17788:2014** (Cloud Computing - Overview and Vocabulary):
    - Provides guidelines to ensure the cloud environment and server provisioning adhere to accepted cloud computing practices, helping with clarity and consistency in terms of cloud terminology and design.

**3. OSHA Codes (Occupational Safety and Health Administration)**

- **OSHA 1910.132** (Personal Protective Equipment):
    - Since CCAC staff may need to physically interact with servers or infrastructure components, operators must follow OSHA guidelines on eye and face protection, especially during maintenance, or if hardware setups are part of the workflow.

---

**4. NIST Standards (National Institute of Standards and Technology)**

- **NIST SP 800-53 Revision 5** (Security and Privacy Controls for Information Systems and Organizations):
    - This standard should guide the implementation of security controls within the CCAC system, particularly for ensuring that user data is protected and that privacy measures are integrated into the cloud infrastructure.
- **NIST SP 500-291** (Cloud Computing Standards Roadmap):
    - Helps guide the selection of cloud service models, deployment models, and cloud security controls. This standard also helps establish best practices for cloud interoperability, ensuring that CCAC's hybrid cloud can integrate seamlessly with other systems if needed.

---

**5. W3C Standards (World Wide Web Consortium)**

- **W3C HTML/CSS Standards**:
    - If the CCAC service catalog is accessible via a web-based platform, the user interface should follow W3C standards for HTML and CSS to ensure compatibility and accessibility across different web browsers.

---

**6. ASME Standards (American Society of Mechanical Engineers)**

- **ASME B31.3** (Process Piping):
    - While typically applied to mechanical systems, if there are any physical connections or infrastructure support required for server hardware (in cases of hybrid cloud configurations that interact with physical computing systems), adherence to this standard will be necessary.

---

**7. IETF (Internet Engineering Task Force)**

- **RFC 793** (Transmission Control Protocol - TCP):
    - Ensures that the CCAC's server images use standardized communication protocols for reliable network communication across different architectures and platforms.
- **RFC 5246** (Transport Layer Security Protocol TLS 1.2):

- Any server provisioning involving network communication must adhere to security protocols like TLS for encryption, especially when dealing with sensitive academic research data or private university resources.

---

**8. Software Licensing and Compliance**

- **GPLv3** (GNU General Public License):
  - If any open-source software is incorporated into the virtual server images, compliance with GPL or similar licenses (e.g., MIT License, Apache License) is necessary to ensure the legal use of code and the freedom to modify and distribute it.
- **Ansible License and Usage**:
  - Since Ansible is a key tool for provisioning, compliance with its license and any relevant third-party software licenses (from GitHub repositories, etc.) must be ensured.

---

**9. Other Codes and Standards**

- **EPA (Environmental Protection Agency)**:
  - Any physical server infrastructure at CCAC must follow **EPA energy guidelines** for data centers to ensure energy-efficient operation.

**Implementation of Codes and Standards in Design:**

- **IEEE and ISO Standards** will govern the quality, security, and reliability of the service catalog and virtual server images.
- **NIST and W3C** standards will ensure security, privacy, and interoperability across CCAC's cloud platform.
- **OSHA** will regulate the safety of personnel working on or interacting with the physical infrastructure, if necessary.
- **IETF Standards** will guarantee that the server images and the networking within the hybrid cloud follow accepted communication protocols, ensuring reliability in data transfers.

Incorporating these standards ensures that the **CCAC Service Catalog** adheres to high-performance computing standards, legal regulations, and best practices in security and

usability. Moreover, it enhances the overall reliability, safety, and scalability of the system, benefiting users across Virginia's academic and research institutions.

## Section C. Scope of Work

The **Scope of Work** for the Commonwealth Center for Advanced Computing (CCAC) Service Catalog project focuses on creating a detailed plan outlining the project's key objectives, timeline, and deliverables. The primary goal is to design and implement a service catalog of pre-configured virtual servers for educational and research purposes, with compatibility across IBM z/16, Power10, and x86 architectures. The project team is responsible for identifying server requirements, automating server provisioning using Ansible, and ensuring that all configurations meet the specific needs of Virginia-based universities. The scope also includes a clear timeline and milestone structure to keep the project on track. The project will follow a waterfall methodology for the initial design phase and transition to an Agile approach for iterative implementation. Responsibilities also include testing, documentation, and delivering a fully functional catalog by the end of the academic year, while keeping communication with stakeholders consistent to ensure the final product aligns with expectations.
The scope explicitly excludes tasks such as hardware maintenance, long-term support, or the management of infrastructure upgrades. The project timeline and budget will be adhered to closely, with all changes outside the agreed-upon deliverables managed to prevent scope creep. Clear boundaries will be established to manage expectations, and approvals will be obtained through ongoing collaboration with CCAC stakeholders and formal validation of deliverables at key project stages. The focus is on delivering a user-friendly, scalable service catalog within the established project constraints.

### C.1 Deliverables

The **Deliverables** for the Commonwealth Center for Advanced Computing (CCAC) Service Catalog project include a set of outputs that the project team is responsible for delivering to the sponsor. These deliverables primarily consist of pre-configured virtual server images compatible with IBM z/16, Power10, and x86 architectures, along with automated provisioning scripts developed using Ansible. Additionally, the project will provide comprehensive documentation, including a maintenance plan, user manuals, and training materials to ensure CCAC staff can manage and update the service catalog in the future. Academic deliverables include the project proposal, design reports, a fall poster, and presentations, culminating in a final Capstone EXPO poster and presentation.

To mitigate risks associated with deliverable completion, the team must consider potential obstacles such as access to campus facilities and resources, ensuring that any work that can be done remotely is properly supported through software licenses and shared drives. Some deliverables may also require components or software from third-party vendors, with careful planning needed to avoid delays due to supply chain disruptions. The team will maintain open

communication with CCAC sponsors and faculty advisors to address these risks and ensure all deliverables are provided on time and meet project specifications.

**C.2 Milestones**

The project involves creating a comprehensive service catalog for the Commonwealth Center for Advanced Computing (CCAC) that will allow universities and businesses in Virginia to easily provision and deploy virtual servers. The catalog will feature pre-configured "golden images" for various server types, including AI, web, database, development environments, virtual reality, and natural language processing servers, compatible with IBM z/16 mainframes, Power10, and x86 platforms. Key tasks include identifying server requirements, ensuring compatibility, creating automation scripts using Ansible, and testing deployment processes.

**Milestones:**

**Phase 1: Initial Planning and Requirements Gathering (2-3 Weeks)**

- **Task:** Meet with stakeholders (CCAC, university partners) to finalize user requirements and server types.
- **Deliverables:**
    - Stakeholder meeting minutes and approval of server types (e.g., database, web, AI, VR).
    - Detailed documentation of software packages required for each server type.
    - Compatibility analysis with IBM z/16, Power10, and x86 architectures.

**Phase 2: Architecture Design and Ansible Familiarization (2 Weeks)**

- **Task:** Design the overall architecture for the service catalog and familiarize the team with Ansible for automated server provisioning.
- **Deliverables:**
    - Complete architectural blueprint for the service catalog.
    - Training materials or sessions on Ansible automation.
    - Initial repository of Ansible scripts from public or vendor-supported GitHub repositories.

**Phase 3: Development of Golden Images for Servers (3 Weeks)**

- **Task:** Begin configuring the “golden images” for different types of servers based on gathered requirements.
- **Deliverables:**
    - Initial versions of at least 4 virtual servers (e.g., database, AI, development environment, web server).
    - CPU, RAM, storage specifications, and OS compatibility for each server.
    - Draft Ansible scripts for provisioning these servers.

**Phase 4: Automation and Testing of Server Provisioning (2-3 Weeks)**

- **Task:** Develop Ansible scripts for fully automated server provisioning and test the deployment process.
- **Deliverables:**
    - Final Ansible scripts for server provisioning.
    - Successful testing of server deployment within 30 minutes.
    - Bug reports and optimizations documented.

**Phase 5: Expansion of Server Types and Further Testing (3 Weeks)**

- **Task:** Expand the catalog to include additional server types (e.g., augmented reality, NLP).
- **Deliverables:**
    - Fully configured additional servers (e.g., VR, AR, NLP).
    - Extended testing of server compatibility across all platforms.
    - Reports on user experience and provisioning speed.

**Phase 6: Final Integration and Documentation (2 Weeks)**

- **Task:** Final integration of the service catalog and detailed documentation for CCAC staff.
- **Deliverables:**
    - Final version of the service catalog with all servers and provisioning scripts.
    - Comprehensive user guide for future updates and maintenance.
    - Final documentation of Ansible scripts for future CCAC use.

**Phase 7: Final Testing and Deployment (2 Weeks)**

- **Task:** Perform final tests across all platforms and prepare for deployment.
- **Deliverables:**
    - Full report on testing results.
    - Sign-off from CCAC stakeholders and partner universities.
    - Go-live of the service catalog for CCAC and its partners.

**Phase 8: Presentation and Reporting (2 Weeks)**

- **Task:** Prepare and present the final results to stakeholders.
- **Deliverables:**
    - Final presentation for CCAC and university partners.
    - Complete project report, including technical specifications and user feedback.
    - Recommendations for future improvements and maintenance.

**C.3 Resources**

This project will need specific hardware, software, cloud services, and data tools.

**Hardware Resources**:

- **HPCs or Servers**: IBM z/16 mainframe, Power10, and x86 servers are necessary to test compatibility and deploy golden images for the service catalog.
- **Local Testing Environment**: A dedicated system for internal testing of configurations before pushing to the CCAC environment.

2. **Software Resources**:
   - **Integrated Development Environments (IDEs)**: Tools such as PyCharm or Visual Studio for development, debugging, and script management.
   - **Version Control System**: GitHub or GitLab for collaborative work, version control of Ansible scripts, and code reviews.
   - **Server Provisioning Tools**: Ansible for automated server provisioning.
   - **Operating Systems**: z/OS, z/VM, Linux, and Windows for testing compatibility across platforms.
3. **Cloud Computing Resources**:
   - **CCAC Cloud Access**: To deploy and manage virtual servers within the hybrid cloud environment.
   - **Scalable Cloud Platforms**: Potential access to additional cloud computing resources (e.g., AWS or Azure) if CCAC's capacity is insufficient for testing large-scale deployment or specific use cases.
4. **Data Resources**:
   - **Operational Databases**: Access to test datasets or pre-existing databases within the CCAC environment for testing server configurations and deployment speed.
   - **Testing Data**: Any public or client-provided data sets necessary to validate server functionality, especially for AI and NLP servers.
5. **Libraries and APIs**:
   - **Machine Learning Libraries**: TensorFlow, PyTorch, or other relevant APIs needed for AI servers.
   - **Server Automation Scripts**: Existing Ansible repositories from public or vendor-supported GitHub projects.
   - **API Integrations**: For seamless integration of third-party tools into the virtual environments (e.g., APIs for NLP, VR, or database management).

## Section D. Concept Generation

### Modular Golden Images

Here, we organize server types based on categories with modular software components that can be added or removed based on the server's needs and functions.

- Each server has a base image that contains the operating system and necessary security configurations.
- In addition to the base operating system software components are grouped into categories (Database, Web Server, AI, VR, AR, etc.), and each category contains specific packages (MongoDB for databases or Apache for web servers).
- A modular approach allows easy customization, where each server can be tailored by stacking relevant modules based on its role.

### Pros

- Flexibility in updating or changing software without rebuilding entire images.
- Easier maintenance as different servers can share common base images and only differ by the modules applied.
- Potential to reduce storage overhead by sharing common image layers.

### Cons

- Complexity in managing dependencies between different modules.
- Testing becomes more challenging as each combination of the base image and modules must be validated.

### Potential Risks

- Dependency conflicts between different modules.
- Lack of uniformity across servers if modules are not applied consistently.

### Tiered Golden Image Framework Based on Server Role

Here, golden images are predefined for specific server roles with three tiers based on complexity and usage.

- Servers are grouped into three tiers: Tier 1 (Basic), Tier 2 (Standard), and Tier 3 (Advanced).
- Tier 1 includes lightweight installations meant for simple, singular-purpose servers (e.g. MongoDB)
- Tier 2 includes standard software stacks for medium-complexity services (e.g. SQL servers, application servers)
- Tier 3 is reserved for highly specialized services requiring more comprehensive software stacks (e.g. AR/VR servers, high-performance computing nodes).

**Pros**

- Streamlined process for server provisioning based on predefined roles.
- Simplifies the selection process when setting up new servers—just select the appropriate tier.
- Scalable design that works well in both small and large environments.

**Cons**

- Limited flexibility if a server doesn't neatly fit into one of the three tiers.
- Modifying a tier requires rebuilding the entire image, which may lead to downtime.

**Potential Risks**

- Servers may become over-provisioned or under-provisioned if the tiers are too rigid.
- Some tiers may become outdated faster than others due to changing software needs.

**Ansible Alternatives**

Instead of using Ansible to provision and deploy the servers, we could also use Helm, Chef, or Puppet. These are fine alternatives, but then we would need licenses to use the software. These alternatives don't offer any significant benefits over Ansible, and we aren't familiar with the software as much as Ansible. This means if we choose to use the other software, then we would need to get familiar with the software and verify if it will be compatible with the specific requirements that we have.

## Section E. Concept Evaluation and Selection

Using a systematic decision-making process, evaluate each of the design concepts and choose the one that is most likely to succeed in meeting the design objectives and constraints. A Decision Matrix, or Pugh Matrix, helps to analyze alternatives, eliminate biases, and make rational decisions through thought and structure. First, work to develop a set of selection criteria for which to evaluate the previously generated design concepts. Selection criteria often include concepts of performance, cost, safety, reliability, risk, etc. Note that the selection criteria developed here will likely be more general than the project design objectives. As with the design objectives, conversations with the client help define appropriate selection criteria.

In many cases, the client may value the selection criteria differently, preferring that more emphasis be placed on some than others. In this case, weighting factors may be used to place more or less importance on the various criteria in the decision-making process. Again, conversations with the client can be used to define criteria weighting factors. Oftentimes, these conversations must be analyzed and interpreted by the team to determine which criteria are more

important to the client and by how much. Feel free to discuss the assigned weighting factors with the client to see if they seem accurate.

Next, define an associated metric to represent each criteria. Metrics should be specific and quantifiable, providing numerical values that quantify the often vague concepts of the selection criteria. Metrics can be obtained, generated, or estimated through a number of methods including simple background research, preliminary design calculations, or basic analyses. Note that these metrics do not need to specifically align with the design specifications although there may be some commonality between the two. Provide a brief discussion of the rationale for selecting each of the assigned metrics.

Using the defined metrics, evaluated each design concept against all selection criteria by filling out a Decision Matrix. Design concepts can be compared by using simple rank scoring, raw scoring, or weighted scoring techniques and design concept with which to move forward can be selected. This type of process provides a meaningful, unbiased means for choosing a preliminary design concept prior to moving forward with more comprehensive, detailed analyses as provided in the design methodology section below. The results of this process should be discussed with the project client prior to moving forward with the selected design. Table 1 provides an example of a simple decision matrix.

Here, we select the most appropriate design based on the following requirements.

1. Identify the types of virtual servers needed in the service catalog (see the high level design below)
    a. Determine the software needed to create the virtual server - provide links to download
    b. Determine the CPU, RAM, and storage requirements for the software needed
    c. Determine the OS and platform the virtual server will run on (Z-Series, P-Series, x86)
2. Get familiar with Ansible – a server provisioning scripting language.
3. Find public or vendor-supported GitHub repositories for Ansible deployment for the identified virtual servers that will go into the service catalog
4. Create or modify the Ansible script to provision the identified virtual servers

The idea is that users will be able to pick one or more virtual servers they need, then deploy and update their version of the server to meet their particular needs and requirements.

A modular golden image framework is most ideal for this because

- Each server is grouped into its own category.
    - This is convenient because now everything is self-contained and each category contains its specific packages.
- It allows for easy customization since software can be stacked on top of each other
- It gives users the flexibility to tailor the server to their needs.

- The hardware requirements are predetermined based on the particular software being installed on the server.

Therefore, the modular golden image design is the most optimal since it meets all specified requirements.

**Table 1. Decision Matrix.**

Here, we rank the preceding design concepts based on a few criteria. We assign a score from 1 (worst) to 5 (best) based on how well the particular design concept meets the desired criteria.

| | Design 1 | Design 2 | Design 3 |
|---|---|---|---|
| Ease of use | 4 | 2 | 2 |
| Scalability | 5 | 3 | 2 |
| Modularity | 4 | 3 | 3 |
| | | | |
| Total | 9 | 8 | 7 |

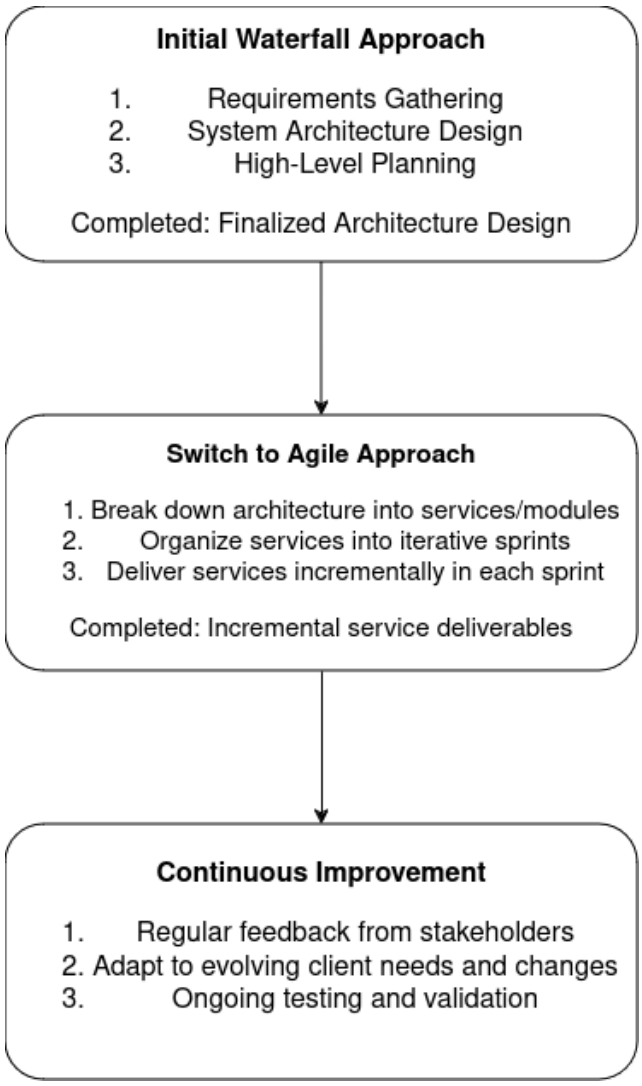
Notice that Design 1, Modular golden images plus Ansible, has the highest score out of all three designs.

## Section F. Design Methodology

We begin by using a **waterfall** approach to outline the overall architecture of the project. The waterfall model allows us to define the project's scope and ensure that all requirements are fully understood before moving forward. This initial phase includes gathering requirements, developing the architecture design, and creating a detailed plan for breaking the project into manageable components.

After the architectural design is complete, we will transition into an **agile** approach. This methodology will allow for continuous evaluation and iteration through short sprints, where each sprint is focused on completing specific services or modules. Agile's flexibility will help ensure that the project can adapt to changes and improvements as they arise throughout development.

Below is a flowchart-style figure that demonstrates the process.



**F.1 Architecture/High-level Design**

CCAC is made up of 7 tenants:

- Teaching
- Academic Research
- Medical Research
- Business Partnerships
- VA Cyber Range
- Incubators and Infrastructure

Each domain is made up of one or more Organizations (VRF). This means that in the teaching domain, all the universities will have their own VRF and in the Business Partnership domain, each business will have their own VRF.

Under each Organization (VRF) there are one or more Courses/Projects (domain bridge). Each Course/Project will have multiple user/student (EPG or End Point Group) accounts with privileges. Each EPG can either work on the Course/Project environment account, or establish their own environment for learning/development. The EPG is where the user policy is defined, determining what each user has authority to do in the environment.

The idea is that users can select from the *service catalog* and select *virtual servers/containers* to use in their account environment. These virtual servers/containers are defined in the hybrid cloud as golden images that the users can provision and modify to their needs. The golden image virtual servers/containers are targeted to run on one of three processor platforms:

- Z-Series
- P-Series
- x86

Supported operating systems are z/OS, z/VM, Linux and Windows.

Then, we will use Ansible to provision and deploy the server. Ansible gives users the ability to tailor their particular server and packages to their needs. We will use public or vendor-supported GitHub repositories for Ansible deployment.

**F.2 Validation Procedure**

We will schedule a final design review meeting with the client, targeted for mid-March. This meeting will serve as an opportunity to

- Present the completed design.
- Demonstrate a working prototype to confirm the design's performance.
- Provide a comprehensive overview of how the final design addresses the client's requirements, including functionality, usability, and any additional design objectives set during the initial phase.

During this review, the design team will walk the client through the key features and operational aspects of the design, ensuring that all technical specifications have been met.

## Section G. Results and Design Details

Use this section to highlight the major results of the design methodology described above including important analytical, computational, experimental, modeling, assembly, and testing results. This section should be one of the most substantial sections of the report showcasing all of the hard work and effort that went into the completion of the final design and delivery of the project deliverables. Show how the identified problem was solved.

Highlight the prominent features of the final design through analysis results, modeling, drawings, renderings, circuit schematics, instrumentation diagrams, flow and piping diagrams, etc. to show that the design functions as intended and meets all design objectives and constraints. Overview designs such as data flow diagrams, process flow, swim lane diagrams, as well as presentation-layer designs (e.g. storyboards for front-ends) should be included here. Detailed designs such as database designs, software designs, procedure flowcharts, or pseudocode should be included here. Support computational and experimental results with key plots and figures. All supporting figures should be clearly labeled and annotated to highlight the most important points of the figure (i.e. explicitly point out what the reader should focus on or understand about the image).

Note that while all results should be used to help inform design decisions, not all results may be necessary to include in the main body of the report. Extraneous supporting results (e.g. graphs, data, design renderings, drawings, etc.) that are not necessary for presenting the fundamental findings can be placed in one or more appendices. Detailed documentation of each program module can be provided as an appendix.

### G.1 Modeling Results (example subsection)

### G.2 Experimental Results (example subsection)

### G.3 Prototyping and Testing Results (example subsection)

### G.4. Final Design Details/Specifications (example subsection)

Note that while the design constraints and specifications may have provided minimum or maximum values, or ranges or values, that the design needed to meet, the final design specifications should be listed here showing that the required design values were met. A list of final design details can also be included to demonstrate fulfillment of the design objectives.

***Note:*** Preliminary results should be included in the Preliminary Design Report to show the progress made of the selected design concept to-date. This section should be updated for the Final Design Report to include documentation of all of the work that was completed on the project throughout the entirety of the academic year.

## Section H. Societal Impacts of Design

In addition to technical design considerations, contemporary engineers must consider the broader impacts that their design choices have on the world around them. These impacts include the consideration of public health, safety, and welfare as well as the potential societal, political/regulatory, economic, environmental, global, and ethical impacts of the design. As appropriate for the project design, discuss how each of these considerations influenced design choices in separate subsections. How will the design change the way people interact with each other? What are the political implications of the design? Does the technology have the potential to impact or shift markets? Does the design have any positive or negative effects on the environment? Don't forget to consider unintended consequences such as process or manufacturing byproducts. What impacts might the design have on global markets and trade? Are there any ethical questions related to the design?

While it is hard to forecast the various impacts of a technology, it is important to consider these potential impacts throughout the engineering design process. When considered during the early stages of the design phase, consideration of these impacts can help determine design objectives, constraints, and specifications and help drive design choices that may mitigate any potential negative impacts or unintended consequences.

***Note:*** A minimum of 4 of these design considerations, including the consideration of public health, safety, and welfare, are required for the Preliminary Design Report while a section for all considerations must be included in the final design report.

### H.1 Public Health, Safety, and Welfare

Public Health, Safety, and Welfare will be considered and noted in the Preliminary Design Report.

### H.2 Societal Impacts

Societal impacts will be considered and noted in the Preliminary Design Report.

### H.3 Political/Regulatory Impacts

Political/Regulatory impacts will be considered and noted in the Preliminary Design Report.

### H.4. Economic Impacts

Economic impacts will be considered and noted in the Preliminary Design Report.

### H.5 Environmental Impacts

Environmental impacts will be considered and noted in the Final Design Report.

### H.6 Global Impacts

Global impacts will be considered and noted in the Final Design Report.

### H.7. Ethical Considerations

Ethical Considerations will be considered and noted in the Final Design Report.

## Section I. Cost Analysis

Provide a simple cost analysis of the project that includes a list of all expenditures related to the project. If an experimental test set-up or prototype was developed, provide a Bill of Materials that includes part numbers, vendor names, unit costs, quantity, total costs, delivery times, dates received, etc. Do not forget to include all manufacturing costs incurred throughout the completion of the project. If the design is expected to become a commercial product, provide a production cost estimate including fixed capital, raw materials, manufacturing (including tooling and/or casting), and labor costs to produce and package the device. Note that this type of detailed cost analysis may be listed as a project deliverable.

**Note:** The Preliminary Design Report should include all costs incurred to date. It is expected that this section will be expanded and updated between the preliminary and final design reports.

**As of 10/11/24, there are no costs incurred to the University and no costs are expected to incur. The CCAC cost analysis will be imported for the preliminary design report, if needed.**

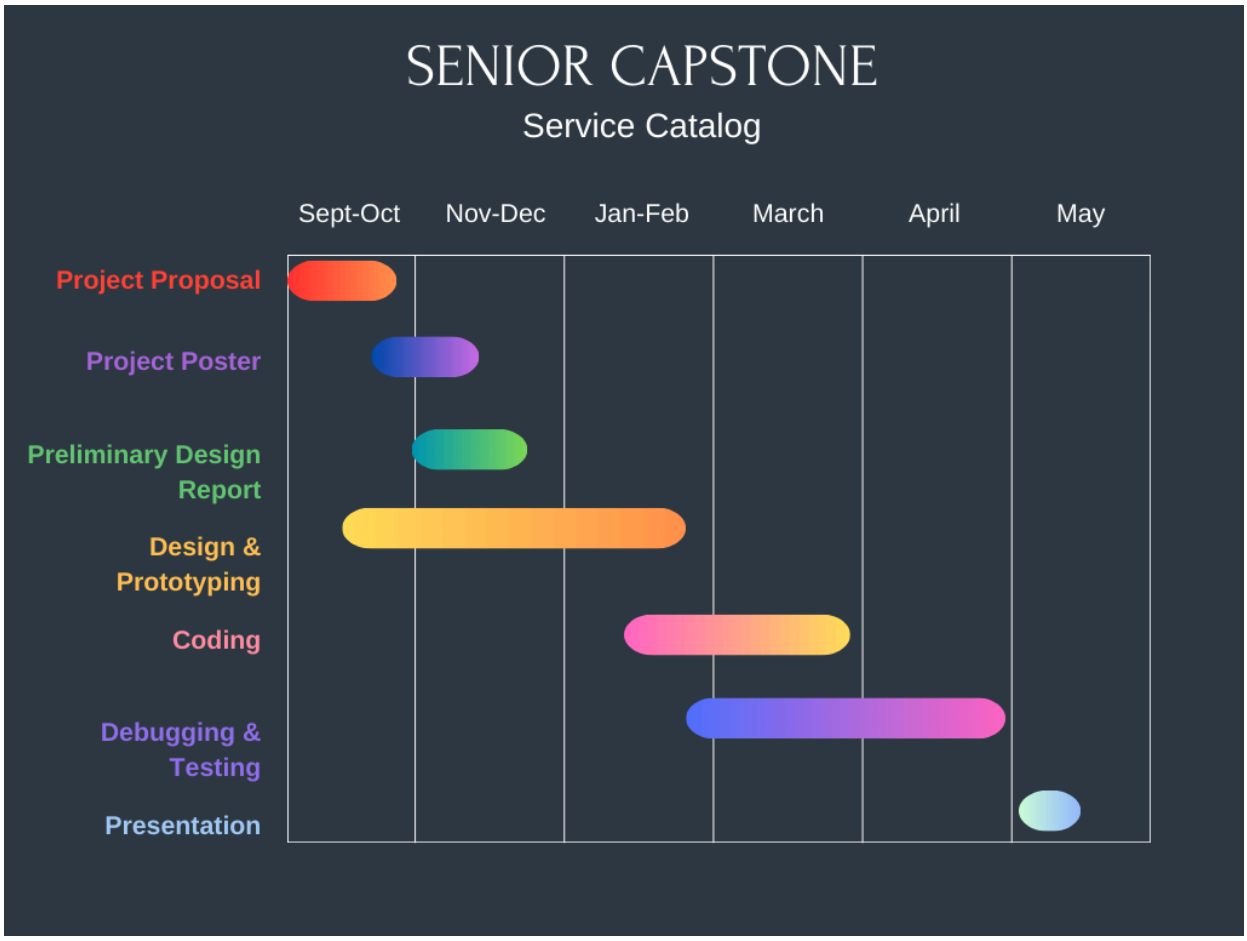
## Section J. Conclusions and Recommendations

Use this section to summarize the story of how the design team arrived at the final design. Focus on the evolution of the design through the use of the engineering design process including lessons learned, obstacles overcome, and triumphs of the final design. Revisit the primary project goals and objectives. Provide a brief summary of the final design details and features paramount to the function of the design in meeting these goals and objectives.

A discussion may be included to discuss how the design could be further advanced or improved in the future. If applicable, summarize any questions or curiosities that the final results/design of this effort bring to mind or leave unanswered. If this project might continue on as a future (continuation) senior design project, detail the major milestones that have been completed to date and include any suggested testing plans, relevant machine drawings, electrical schematics, developed computer code, etc. All relevant information should be included in this section such that future researchers could pick up the project and advance the work in as seamless a manner as possible. Documents such as drawings, schematics, and codes could be referenced here and included in one or more appendices. If digital files are critical for future work, they should be saved on a thumb drive, external hard drive, cloud, etc. and left in the hands of the project advisor and/or client.

## Appendix 1: Project Timeline

Provide a Gantt chart of similarly composed visual timeline showing the start and end dates of all completed tasks and how they are grouped together, overlapped, and linked together. Include all senior design requirements including design reports and Expo materials (i.e. Abstract, Poster, and Presentation). All major milestones should be included in the timeline.



## Appendix 2: Team Contract (i.e. Team Organization)

### Step 1: Get to Know One Another. Gather Basic Information.

| *Team Member Name* | *Strengths each member brings to the group* | *Other Info* | *Contact Info* |
|---|---|---|---|
| Akhil Manoj | Quick learner, proficient in multiple programming languages, goal-oriented | I enjoy taking on interesting challenges and learning new things | manoja2@vcu.edu<br>804-309-8888 |
| *Bilal Othman* | Adaptability and Continuous Learning. Attention to Detail. Good Background in programming languages | Currently, I work part-time at NNS, where we specialize in constructing aircraft and submarines | Othmanb@vcu.edu<br><br>804-956-8505 |
| Nick Arthur | Motivated and persistent. I like to have things completed before deadlines. | I am a double major: Computer Science and Applied Mathematics | arthurn2@vcu.edu<br>571-577-3143 |
| Sage Walker | Good at resolving interpersonal conflict, hard-working, and good at following schedules. | I am also the treasurer for VCU, Crew, as well as a TA for 254. I am pursuing a minor in Mathematics. | walkers15@vcu.edu<br><br>(703) 232-7876 |

| *Other Stakeholders* | *Notes* | *Contact Info* |
|---|---|---|
| *Robert Dahlberg* | | *dahlbergra@vcu.edu* |

## Step 2: Team Culture. Clarify the Group's Purpose and Culture Goals.

| ***Culture Goals*** | ***Actions*** | ***Warning Signs*** |
|---|---|---|
| Punctuality | - Set up meetings through Zoom<br>- Send a message reminder in Discord the day before the meeting | - Student misses first meeting, warning is granted<br>- Student misses meetings afterwards – issue is brought up with faculty advisor<br>- Student appears stressed, out of the loop |
| Communication & Accountability | - Stay on top of time management (classes, jobs, etc)<br>- Use built-in time buffers when planning deadlines | - Student is unreachable<br>- Student has no quantifiable progress on their tasks from meeting to meeting |
| Respectfulness & Professionalism | - Treat the faculty advisor/sponsor as a real-world client<br>- Treat the other members of the teams as equals and respect each other's opinions | - Student has aggressive tone, noticeable irritability<br>- Other students are uncomfortable with communicating to student in question because of their behavior |

## Step 3: Time Commitments, Meeting Structure, and Communication

| ***Meeting Participants*** | ***Frequency***<br>***Dates and Times / Locations*** | ***Meeting Goals***<br>***Responsible Party*** |
|---|---|---|
| *Students Only* | *As Needed, On Discord Voice Channel* | *Meet to discuss any current roadblocks as needed* |
| *Students Only* | *Weekly on Tuesdays, full attendance is expected* | *Give updates on the current status of each team member's task.*<br>*Discuss any current roadblocks or issues*<br>*Plan out work to be done in the upcoming week* |
| *Students + Faculty advisor* | *Weekly on Thursdays, full attendance is expected* | *Update faculty advisor and get answers to our questions. We will all take notes and participate by asking question (if we have any)* |

## Step 4: Determine Individual Roles and Responsibilities

| ***Team Member*** | ***Role(s)*** | ***Responsibilities*** |
|---|---|---|
| Akhil Manoj | Project Manager | ✔ Be the primary communicator between the team and faculty advisor/project sponsor<br>✔ Make sure everyone understands what is going on |
| Sage Walker | Financial Manager | ✔ conduct pricing analysis and budget justifications<br>✔ log purchases and submit receipts to appropriate entities<br>✔ carry out team purchase requests<br>✔ monitor budget |

| | | |
|---|---|---|
| Nick Arthur | Record keeping | ✔ *Take notes during weekly meeting with Dahlberg*<br>✔ *Save and share notes with the team in GitHub (maybe Discord, but GitHub primarily).* |
| Bilal Othman | Task Coordinator | ✔Provide regular updates to the Project Manager about the status of tasks and any potential delays.<br><br>✔Help team members with any issues or obstacles they might encounter in completing their tasks. |

## Step 5: Agree to the above team contract

*Team Member:* Akhil Manoj *Signature: Akhil Manoj*

*Team Member:* Bilal Othman *Signature: Bilal Othman*

*Team Member:* Nick Arthur *Signature: Nick Arthur*

*Team Member:* Malika Sage Walker *Signature: Malika Sage Walker*

## Appendix 3: [Insert Appendix Title]

Note that additional appendices may be added as needed. Appendices are used for supplementary material considered or used in the design process but not necessary for understanding the fundamental design or results. Lengthy mathematical derivations, ancillary results (e.g. data sets, plots), and detailed mechanical drawings are examples of items that might be placed in an appendix. Multiple appendices may be used to delineate topics and can be labeled using letters or numbers. Each appendix should start on a new page. Reference each appendix and the information it contains in the main text of the report where appropriate.

***Note:*** Delete this page if no additional appendices are included.

## References

Provide a numbered list of all references in order of appearance using APA citation format. The reference page should begin on a new page as shown here.


[1] Amazon. (n.d.). *Free cloud server - free lightsail - AWS*. Amazon Web Services. https://aws.amazon.com/free/compute/lightsail/

[2] Susnjara, S., & Smalley, I. (2024, July 9). *What is high-performance computing (HPC)?*. IBM. https://www.ibm.com/topics/hpc#:~:text=HPC%20is%20a%20technology%20that,computing%20problems%20in%20real%2Dtime.

[3] Raj, R. K., Romanowski, C. J., Impagliazzo, J., Aly, S. G., Becker, B. A., Chen, J., ... & Thota, N. (2020). High performance computing education: Current challenges and future directions. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education* (pp. 51-74).