



# VCU

College of Engineering

## CS 25-328 - Tenant Self-Service Portal

### Project Proposal

Prepared for

Mahesh Nair

Capital One

Alan Dorn, Austin Glass, Prakhar Mathur, Krish Patel

Under the supervision of

Preetam Ghosh

12/9/24

## Executive Summary

The Tenant Self-Service Portal project aims to streamline and automate the onboarding and dataset management processes for platform tenants at Capital One. Currently, these processes require manual intervention from platform engineers, leading to inefficiencies, delays, and repeated back-and-forth communications. Our goal is to simplify this workflow by developing a self-service web portal that allows clients to manage their datasets and use cases independently, reducing dependency on engineers and improving overall operational efficiency.

The proposed solution involves building a multi-tenant platform where users (Clients) can perform tasks such as dataset registration, data mapping creation, batch process tracking, and document lifecycle management. The portal will also include an approval system where Admins can review and provide feedback on submitted datasets and use cases. This portal will enhance user experience by allowing users to save progress during dataset creation, view approval status, and make necessary updates based on feedback.

Key project objectives include:

- Developing a self-service portal for clients to manage datasets and use cases.
- Enabling an approval workflow for dataset submissions, with Admin feedback loops.
- Building a user-friendly interface that supports navigation, progress-saving, and dataset updates.
- Ensuring the solution is scalable and secure, adhering to multi-tenant platform management principles.

The project will be implemented using open-source UI stacks and databases such as MongoDB. The focus will first be on building the Dataset Registry, potentially followed by additional features. This project presents a significant opportunity for students to gain hands-on experience in full-stack development, platform management, and enterprise-level solutions. It will also benefit Capital One by reducing manual workload, improving data management efficiency, and providing a scalable solution to support multiple business functions.

The expected outcome is a robust, reliable, and user-centric portal that will reduce time loss, minimize redundancy, and improve the overall onboarding experience for platform tenants at Capital One.

## **Table of Contents**

Section A. Problem Statement	4
Section B. Engineering Design Requirements	6
B.1 Project Goals (i.e. Client Needs)	6
B.2 Design Objectives	6
B.3 Design Specifications and Constraints	7
Section C. Scope of Work	9
C.1 Deliverables	9
C.2 Milestones	10
C.3 Resources	10
Section D. Frontend Design	12
Section E: API Design and Data Models	13
Appendix 1: Project Timeline	15
Appendix 2: Team Contract (i.e. Team Organization)	16
Appendix 3: Workflow Diagram	19

## **Section A. Problem Statement**

The Tenant Self-Service Portal initiative aims to address a critical inefficiency in the current onboarding process for platform tenants at Capital One. The existing workflow relies heavily on manual intervention, necessitating frequent communication between tenants and platform engineers for routine tasks such as dataset registration, data mapping creation, and configuration updates. This manual approach introduces operational delays, increases the potential for human error, and places an undue burden on engineering resources with tasks that are prime candidates for automation. By streamlining and automating these workflows, the project seeks to mitigate operational redundancies, enhance efficiency, and significantly reduce the time required for data submissions and approvals.

The issue impacts two primary stakeholder groups: platform tenants (clients) who require dataset registration and use case implementation, and platform engineers responsible for facilitating these processes. For clients, the absence of a self-service option results in extended wait times for routine task completion, delaying service access. Concurrently, engineers find themselves encumbered by repetitive manual work that could be automated, thereby reducing their productivity and limiting their capacity to focus on high-priority initiatives. These inefficiencies are pervasive across the platform, affecting numerous tenants and internal teams. Consequently, the problem is widespread, and the delays incurred by this outdated process can lead to substantial time losses, increased operational costs, and potential missed business opportunities.

Capital One serves as the primary client for this project, with stakeholders spanning multiple teams whose workflows depend on efficient and timely dataset onboarding. These stakeholders require a solution that is both technically robust and scalable to ensure the platform can accommodate future growth. The objective is to develop an automated self-service portal that empowers tenants to manage key tasks such as dataset registration and updates autonomously, without relying on engineering intervention. By automating the approval process and equipping clients with tools to manage their submissions, the solution will address a critical need in Capital One's platform operations.

This initiative falls within the financial technology (fintech) sector, specifically focusing on enterprise platform management and multi-tenant systems. Within the fintech industry, automation has emerged as a key trend, with companies striving to enhance customer experience by reducing manual intervention and optimizing processes. The self-service portal aligns with this trend, providing an automated, user-friendly interface that enables tenants to manage datasets and use cases more efficiently. The project will leverage modern technologies, including open-source UI stacks and MongoDB, allowing for experimentation with various architectures and tools to deliver a scalable solution.

Historically, Capital One and other financial institutions have relied on manual processes for onboarding datasets and services, resulting in inefficiencies and delays. While some internal

tools and scripts have been developed to partially automate these processes, none have fully addressed the need for a comprehensive self-service solution. The proposed portal improves upon prior attempts by empowering clients to manage their submissions independently while maintaining a robust approval process for quality control. Administrators will have the capability to review submissions, provide feedback, and ensure datasets meet required standards, all while significantly reducing the time and resources previously allocated to manual intervention.

In conclusion, the Tenant Self-Service Portal will automate critical tasks within Capital One's platform ecosystem, reducing operational costs, enhancing client experience, and enabling platform engineers to focus on more strategic initiatives. The solution will advance automation efforts within the fintech industry, providing a scalable and efficient platform for managing datasets and use cases in a multi-tenant environment.

## Section B. Engineering Design Requirements

The Tenant Self-Service Portal project aims to automate the onboarding process for platform tenants at Capital One, providing a streamlined experience for dataset registration, updates, and approvals. To ensure the design effectively meets stakeholder expectations and addresses the project's objectives, clear goals, design objectives, and specifications are outlined below.

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

The primary goals of the project are derived from the needs of Capital One and its stakeholders, who require an automated solution to reduce manual intervention and improve the efficiency of the tenant onboarding process. The project will deliver a platform that automates key tasks for tenants while maintaining a feedback and approval system for admins.

The goals of the project include:

- **To create a self-service portal** that allows tenants to independently register, update, and manage datasets without engineering involvement.
- **To reduce manual back-and-forth** between tenants and engineers, minimizing time spent on repetitive tasks and improving overall platform efficiency.
- **To provide transparency** in the onboarding process, ensuring users can track the approval status of submissions, receive feedback, and make necessary adjustments.
- **To streamline the approval workflow** by enabling admins to review and provide feedback in a structured and automated manner.

These goals are designed to improve operational efficiency, reduce time delays, and empower platform users with a user-friendly interface.

### B.2 Design Objectives

The key objectives of the design are focused on creating a robust and scalable self-service platform that meets the operational needs of Capital One's tenants and engineers. These objectives adhere to the SMART framework, ensuring that they are Specific, Measurable, Achievable, Realistic, and Time-bound.

The design will:

- **Provide a step-by-step process** for tenants to submit datasets in a minimum of 9 screens, allowing users to save progress and return later.
- **Automate the approval process**, allowing admin users to review, comment, and provide feedback on tenant submissions within a maximum of 2 business days.
- **Enable dataset updates** where users can view, update, and manage approved datasets, ensuring that only valid, accurate data is processed.

- **Offer a status dashboard** for clients to track the approval status and feedback for all submitted datasets.

These objectives ensure that the portal provides a streamlined, intuitive experience while adhering to performance and usability standards.

### **B.3 Design Specifications and Constraints**

The design of the Tenant Self-Service Portal will be governed by a set of technical and operational constraints, ensuring that the system integrates seamlessly with existing processes while maintaining reliability and scalability. These constraints are measurable and realistic to meet the project's objectives.

#### **Functional Constraints:**

- The platform must be similar to Capital One's existing tech stack, using Node.js for the backend, Vue.js for the frontend, and a MongoDB database (tentatively chosen).
- The platform must allow tenants to submit datasets within a maximum of 15 minutes per complete submission.
- Admins should provide feedback within two business days to ensure a smooth approval process.

#### **Data Constraints:**

- Dataset submissions must adhere to data validation protocols, including format checks, required fields, and compatibility with existing data standards.

#### **Interoperability Constraints:**

- The system must support RESTful API integration with Capital One's internal systems to allow data processing, approval, and feedback sharing.
- It must integrate with Capital One's authentication protocols to ensure secure user access and data protection.

#### **Cost Constraints:**

- The project is developed using open-source tools (Vue.js, Node.js, MongoDB), resulting in minimal upfront costs during development.
- Future costs for full deployment within the client's proprietary environment are not included in the current phase and will depend on integration with internal systems and possible infrastructure upgrades.

**Maintainability Constraints:**

- The platform is built with open-source tools that have large developer communities, ensuring continued updates and support.
- Maintenance beyond the development phase will depend on integration with the client's internal systems and the handover of long-term upkeep protocols to Capital One's engineering team.

**Usability and Security Constraints:**

- The system must support role-based access control (RBAC), distinguishing between tenant users and admin users.
- The platform must comply with Capital One's data security standards, including encryption of sensitive information and secure data transmission.

By adhering to these specifications and constraints, the Tenant Self-Service Portal will meet the operational needs of Capital One while delivering a scalable, efficient, and secure solution.



## Section C. Scope of Work

The project scope defines the boundaries of the Tenant Self-Service Portal project, outlining the key objectives, timeline, milestones, and deliverables. It delineates the team's responsibilities, including the process by which the proposed work will be verified and approved. Establishing a clear scope facilitates understanding, reduces ambiguities and risks, and manages expectations. Along with stating the team's responsibilities, this section explicitly defines the tasks outside the team's responsibilities. The project scope also sets boundaries on the timeline, available resources, and promised deliverables, preventing scope creep and ensuring control over changes. The section further specifies the development approach to be used, which will follow an agile methodology, allowing for flexibility and regular adjustments as the project progresses. Effective communication with the project sponsor and faculty advisor is essential for keeping the scope, timeline, and budget on track.

### C.1 Deliverables

The project deliverables are the tangible outcomes the project team is responsible for providing to the project sponsor, produced through the engineering design and development process. These include technical components like code, user interfaces, functional diagrams, and documentation. Academic deliverables, required for course completion, are also listed below.

#### Project Deliverables:

- **Functional Dataset Registry:** A portal feature that allows clients to submit datasets, and admins to review and approve them.
- **API Simulation:** RESTful APIs to simulate integration with external systems, demonstrating data processing and approval workflows.
- **User Interfaces:** Vue.js-based front-end for both client and admin users, providing essential interaction points for dataset submission, feedback, and approvals.
- **Documentation:** Full technical and user documentation outlining the developed features, APIs, and system architecture.
- **Testing Suite:** A basic testing framework to validate the core functionalities and ensure the platform operates as intended.

#### Academic Deliverables:

- Team Contract
- Project Proposal
- Preliminary Design Report
- Fall Poster and Presentation
- Final Design Report
- Capstone EXPO Poster and Presentation

## C.2 Milestones

The milestones are major tasks or phases that need to be completed to ensure that the project deliverables are achieved. The following milestones break down the development timeline into manageable tasks and provide a structured approach to completing the project.

### Fall Semester Milestones (Weeks 1-16):

- **Context and Requirements Gathering (Weeks 1-4)**  
Objective: Understand the project scope, gather system requirements, and research relevant technologies.
- **Initial Development of Dataset Registry (Weeks 5-8)**  
Objective: Begin coding the front-end and back-end components of the Dataset Registry.
- **API Simulation and Backend Development (Weeks 9-12)**  
Objective: Implement basic RESTful APIs and finalize core Dataset Registry functionality.
- **Testing and Validation (Weeks 13-16)**  
Objective: Conduct testing, ensure functional accuracy, and refine UI/UX.

### Spring Semester Milestones (Weeks 17-32):

5. **Use Case Registry Development (Weeks 17-20)**  
Objective: Begin developing the Use Case Registry, if time permits.
6. **Refinement and Enhancement (Weeks 21-24)**  
Objective: Refine existing features, based on feedback and testing results.
7. **Final Testing and Documentation (Weeks 25-28)**  
Objective: Finalize testing, prepare the final project for presentation, and complete all documentation.
8. **Final Presentation and Submission (Weeks 29-32)**  
Objective: Present the project and submit all required academic deliverables.

## C.3 Resources

To complete the project, the following resources will be required. These resources will be provided by the project sponsor or sourced from open-source platforms.

### Hardware & Software Resources:

- **Development Environment:** Access to cloud-based IDEs and version control systems (GitHub).
- **Data:** Mock data to simulate tasks and processes
- **APIs:** Access to relevant APIs for simulating system integration (mocked RESTful services).

- **Front-end & Back-end Technologies:** Vue.js for front-end development and Node.js with MongoDB for back-end.
- **Testing Tools:** Basic testing frameworks for validating functionality.

**External Resources:**

- **Libraries and APIs:** Access to libraries for building RESTful APIs and data validation processes.
- **Reference Materials:** Documentation and online resources for implementing authentication, UI components, and secure data handling.

## **Section D. Frontend Design**

### **User flow diagrams for key scenarios**

- Dataset creation wizard
  - Enter basic dataset information (name, description, line of business, data stewards, etc.)
  - Define dataset fields and validation rules
  - Map fields to source systems
  - Review and submit for approval
- Dataset search and browsing
  - View dataset details and version history
- Admin review and approval
  - Review submitted datasets
  - Approve or reject with comments

### **Wireframes of key screens**

- Dataset listing page
  - Table view of datasets with key information (name, status, last updated, etc.)
  - Filters for name, line of business, status
  - Action buttons for editing, deleting, and submitting datasets
- Dataset details page
  - Read-only view of dataset information
  - Version history with changelog
  - Action buttons for editing, deleting, and submitting the dataset
- New dataset wizard
  - Step-by-step interface for creating a new dataset
  - Basic information form with fields for name, description, line of business, data stewards
  - Field definitions form to specify field names, types, validation rules
  - Source mappings form to map dataset fields to source system fields
  - Review and submit page to preview the dataset and submit for approval

## Section E. API Design and Data Models

This section focuses on the dataset creation functionality implemented in the Tenant Self-Service Portal. The API endpoint for creating a new dataset is defined as a POST request to /datasets. The dataset data model includes fields such as dataset name, description, line of business, data stewards, and a flag for international data. The API workflow describes the process of creating a dataset, where the client submits a POST request with the dataset details, and the server validates the request, creates the dataset, and responds with the created dataset object. The frontend integration is achieved through the createDataset function, which uses Axios to send a POST request to the /datasets endpoint with the form data. This functionality lays the foundation for dataset management in the portal, enabling tenants to create datasets seamlessly.

### 1. API Endpoints

#### ○ Dataset Routes

- GET /datasets: Retrieve all datasets
- GET /datasets/{id}: Retrieve a specific dataset by ID
- POST /datasets: Create a new dataset
- PUT /datasets/{id}: Update a specific dataset by ID
- DELETE /datasets/{id}: Delete a specific dataset by ID
- POST /datasets/{id}/submit: Submit a dataset for approval

### 2. Data Models

#### ○ Dataset

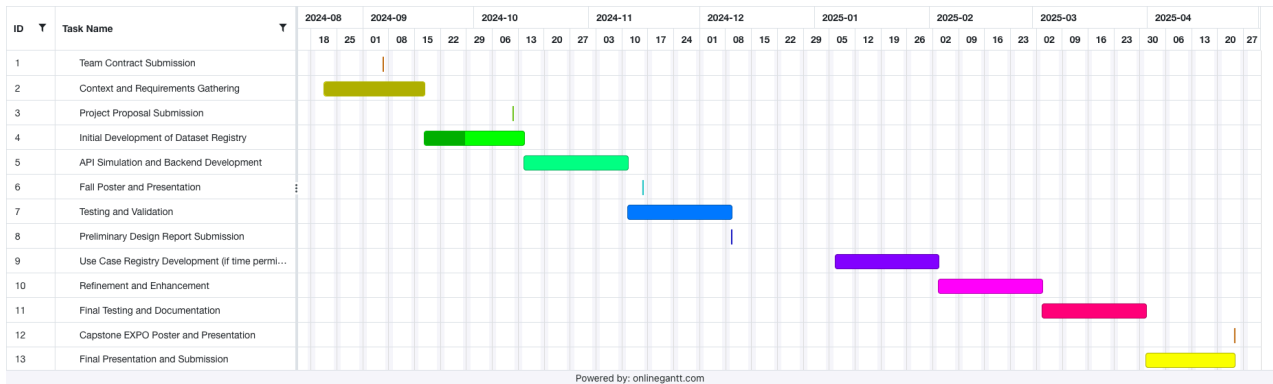
- datasetId: Unique identifier for the dataset
- datasetVersion: Version number of the dataset
- datasetName: Name of the dataset
- lineOfBusiness: Line of business associated with the dataset
- managedFieldContracts: Array of managed field contracts
  - fieldName: Name of the field
  - fieldType: Data type of the field
  - dateFormat: Date format (if applicable)
  - isFieldTokenized: Flag indicating if the field is tokenized
  - keywordValues: Array of keyword values (if applicable)
  - isRequired: Flag indicating if the field is required
  - isMappedFromClientField: Flag indicating if the field is mapped from a client field
  - mappedFromFieldName: Name of the client field (if mapped)
- clientFieldContracts: Array of client field contracts
  - (Similar structure as managedFieldContracts)
- datasetProducers: Array of producer IDs associated with the dataset
- datasetConsumers: Array of consumer IDs associated with the dataset
- dataSources: Array of data source IDs associated with the dataset

- lifeCycleManagementPolicyIds: Array of lifecycle management policy IDs
- accountableExecutive: Email address of the accountable executive
- performingDataSteward: Email address of the performing data steward
- managingDataSteward: Email address of the managing data steward
- description: Description of the dataset
- hasInternationalData: Flag indicating if the dataset contains international data
- managedFieldDetails: Array of managed field details (placeholder)
- clientFieldDetails: Array of client field details (placeholder)

### 3. API Workflow

- Dataset Creation
  - Client submits a POST request to /datasets with the dataset details
  - Server validates the request payload and creates a new dataset with a generated datasetId and a status of 'DRAFT'
  - Server responds with the created dataset object
- Dataset Submission
  - Client submits a POST request to /datasets/{id}/submit to submit a dataset for approval
  - Server updates the dataset status to 'PENDING' and sets the submittedAt timestamp
  - Server responds with the updated dataset object
- Dataset Search and Browsing
  - Search datasets using filters (e.g., name, line of business, status).
  - View dataset details and version history.
- Admin Review and Approval
  - Retrieve submitted datasets for review.
  - Approve or reject datasets with comments.

Appendix 1: Project Timeline



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

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

<b>Team Member Name</b>	<b>Strengths each member bring to the group</b>	<b>Other Info</b>	<b>Contact Info</b>
<i>Prakhar Mathur</i>	<i>Organization, communication, Professional experience, data science + ML, information systems, software development, IT infrastructure + architecture</i>	<i>Python, R, SQL (Data Science, ML)  Java, C  Statistics, Math, Visualization  Azure DevOps, Project Management</i>	<a href="mailto:mathurp6@vcu.edu">mathurp6@vcu.edu</a>  804-591-7609
<i>Alan Dorn</i>	<i>Effective communication, Organization, software development</i>	<i>Java, Python, C, Databases, SQL, HTML, CSS, JS</i>	<a href="mailto:dornn@vcu.edu">dornn@vcu.edu</a>  703-713-5629
<i>Krish Patel</i>	<i>Workforce experience, organization, software development</i>	<i>Microsoft Power Platform, Java, Python, Databases, HTML, CSS, JS, SQL</i>	<a href="mailto:patelkp7@vcu.edu">patelkp7@vcu.edu</a>  757-386-8642
<i>Austin Glass</i>	<i>Effective communication, backend development,</i>	<i>Java, C, C++, Python, Linux</i>	<a href="mailto:glassal@vcu.edu">glassal@vcu.edu</a>  804-714-7969

<b>Other Stakeholders</b>	<b>Notes</b>	<b>Contact Info</b>
<i>Faculty Advisor</i>	<i>Dr. Ghosh</i>	<a href="mailto:pghosh@vcu.edu">pghosh@vcu.edu</a>
<i>Sponsor</i>	<i>Mahesh Nair</i>	<a href="mailto:mahesh.bahulleyannair@capitalone.com">mahesh.bahulleyannair@capitalone.com</a>
<i>Mentor</i>	<i>Balakrishnan Muthusamy  Ashish Kulkarni  Shannon Hsu</i>	<a href="mailto:balakrishnan.muthusamy@capitalone.com">balakrishnan.muthusamy@capitalone.com</a>  <a href="mailto:ashish.kulkarni2@capitalone.com">ashish.kulkarni2@capitalone.com</a>  <a href="mailto:shannon.hsu@capitalone.com">shannon.hsu@capitalone.com</a>



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

<b><i>Culture Goals</i></b>	<b><i>Actions</i></b>	<b><i>Warning Signs</i></b>
<i>Being on time to every meeting</i>	<ul style="list-style-type: none"> <li>- Set up meetings in shared calendar</li> <li>- Send reminder email in day before meeting</li> </ul>	<ul style="list-style-type: none"> <li>- Student misses first meeting, warning is granted</li> <li>- Student misses meetings afterwards – issue is brought up with faculty advisor</li> </ul>
<i>Informing the group of any delays in completing assignments</i>	<ul style="list-style-type: none"> <li>- Stay up to date with each other's project responsibilities</li> <li>- Set reasonable deadlines and note when an extension is needed</li> </ul>	<ul style="list-style-type: none"> <li>- Student shows up for weekly meeting with no considerable work done</li> </ul>
<i>Communicating effectively about delays</i>	<ul style="list-style-type: none"> <li>- Let other members know if you are running behind or are unable to make it</li> </ul>	<ul style="list-style-type: none"> <li>- Student does not communicate whereabouts</li> </ul>

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

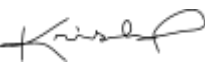
<b><i>Meeting Participants</i></b>	<b><i>Frequency</i></b> <b><i>Dates and Times / Locations</i></b>	<b><i>Meeting Goals</i></b> <b><i>Responsible Party</i></b>
<i>Students Only</i>	<i>As Needed, On Discord Voice Channel</i>	<i>Update group on day-to-day challenges and accomplishments</i>
<i>Students + Faculty advisor</i>	<i>Meet as needed (more of sponsor guidance + involvement)</i>  <i>Weekly/Biweekly</i>	<i>Check-in with Dr. Ghosh on any guidance or expertise needed.</i>
<i>Project Sponsor</i>	<i>Weekday evening (waiting on sponsor communication)</i>  <i>Weekly/Biweekly</i>	<i>Check-in with sponsor for questions (technical/nontechnical), issues, etc.</i>

## Step 4: Determine Individual Roles and Responsibilities

<b>Team Member</b>	<b>Role(s)</b>	<b>Responsibilities</b>
<i>Prakhar Mathur</i>	<i>Project Manager/SDE</i>	<ul style="list-style-type: none"> <li>✓ Define project scope and tasks</li> <li>✓ Facilitate team communication</li> <li>✓ Mitigate risks and allocate resources</li> <li>✓ Monitor progress and quality</li> </ul>
<i>Krish Patel</i>	<i>Full-stack developer/SDE</i>	<ul style="list-style-type: none"> <li>✓ Designing system architecture and UI/UX design</li> <li>✓ Developing backend requirements for software application</li> <li>✓ Debugging and testing software</li> <li>✓ Improving and optimizing accessibility and responsiveness</li> </ul>
<i>Alan Dorn</i>	<i>DevOps</i>  <i>/</i>  <i>Architect</i> <i>personnel</i>	<ul style="list-style-type: none"> <li>✓ Selecting tools</li> <li>✓ Defining infrastructure as code</li> <li>✓ Setting the stage for an automated, efficient pipeline to accelerate innovation</li> </ul>
<i>Austin Glass</i>	<i>Co-Project Manager/SDE</i>	<ul style="list-style-type: none"> <li>✓ Assist in defining project scope and technical tasks</li> <li>✓ Bridge communication between technical and non-technical team members</li> <li>✓ Contribute to coding while managing risks and resources</li> <li>✓ Ensure progress tracking and code quality</li> </ul>

## Step 5: Agree to the above team contract

**Team Member:** *Krish Patel*

**Signature:** 

**Team Member:** *Alan Dorn*

**Signature:** *Alan Dorn*

**Team Member:** *Prakhar Mathur*

**Signature:** *Prakhar Mathur*

**Team Member:** *Austin Glass*

**Signature:** *Austin Glass*

## Appendix 3: Workflow Diagram

