



College of Engineering

CS 25-328 - Tenant Self-Service Portal

Final Design Report

Prepared for

Mahesh Nair, Ashish Kulkarni, Shannon Hsu

Capital One

Alan Dorn, Austin Glass, Prakhar Mathur, Krish Patel

Under the supervision of

Preetam Ghosh

5/9/25

Executive Summary

The Tenant Self-Service Portal project has successfully streamlined and automated the onboarding and dataset management processes for platform tenants at Capital One. Prior to implementation, these processes required extensive manual intervention from platform engineers, causing inefficiencies, delays, and repetitive communications. Our solution has simplified this workflow through a self-service web portal that enables clients to manage their datasets and use cases independently, significantly reducing dependency on engineers and improving overall operational efficiency.

The implemented multi-tenant platform enables users (Clients) to perform critical tasks such as dataset registration, field configuration, producer/consumer selection, and lifecycle management policy configuration. The portal includes a comprehensive approval system where administrators can review and provide feedback on submitted datasets. User experience is enhanced through progress-saving functionality during dataset creation, status tracking, and streamlined update capabilities based on feedback.

Key achievements of the project include:

- Development of a complete self-service portal for clients to independently manage datasets with minimal engineer involvement
- Implementation of a structured approval workflow for dataset submissions with administrator feedback capabilities
- Creation of an intuitive, user-friendly interface featuring multi-step navigation, progress tracking, and draft-saving functionality
- Deployment of a secure solution that adheres to multi-tenant platform management principles

The project was implemented using Vue.js for the frontend, Node.js for the backend, and a robust database architecture for data persistence. The primary focus was on building a comprehensive Dataset Registry with the flexibility to incorporate additional features in future iterations.

The resulting application delivers a robust, reliable, and user-centric portal that has measurably reduced processing time, minimized operational redundancy, and improved the overall onboarding experience for platform tenants at Capital One. The project successfully achieved all primary objectives while providing valuable hands-on experience in full-stack development, platform management, and enterprise-level solution design.

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
Section F: Challenges and Solutions	15
Section G: Conclusion and Future Work	16
Appendix 1: Project Timeline	17
Appendix 2: Team Contract (i.e. Team Organization)	18
Appendix 3: Workflow Diagram	21

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 were 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 Tenant Self-Service Portal successfully met the following goals:

- Created a self-service portal allowing tenants to independently register, update, and manage datasets
- Reduced manual back-and-forth between tenants and engineers
- Provided transparency in the onboarding process through status tracking and feedback mechanisms
- Streamlined the approval workflow with structured feedback channels

These goals were 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 implemented design achieved these key objectives:

- Developed a step-by-step process for tenants to submit datasets through a minimum of 9 screens
- Created a progress-saving mechanism allowing users to return to incomplete submissions
- Implemented an approval workflow for admin review and feedback
- Built a dataset gallery with filtering capabilities for better dataset management
- Designed a user-friendly interface that supports intuitive navigation

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 was governed by a set of technical and operational constraints that ensured the system integrated seamlessly with existing processes while maintaining reliability and scalability. These specifications were measurable and realistic, helping the project meet its objectives.

Functional Constraints:

- **Technology Stack Implementation:** The platform was successfully built using Vue.js for the frontend and Node.js for the backend, aligning with Capital One's existing technology ecosystem. The database architecture was designed with scalability in mind, supporting complex data relationships.
- **Submission Efficiency:** The multi-step dataset submission process was optimized to allow users to complete submissions, with built-in validation at each step to reduce errors.
- **Feedback:** The administrator interface was designed to facilitate review and feedback, including status tracking and notification capabilities.

Data Constraints:

- **Validation Protocols:** Comprehensive validation protocols were implemented throughout the application, including format verification for email addresses, required field validation, and data type checking for specialized fields (dates, keywords, etc.).
- **Field Contract Management:** The system successfully handles complex field contracts, including managed fields and client fields with various data types, required statuses, and tokenization options.

Interoperability Constraints:

- **API Architecture:** A RESTful API architecture was developed with endpoints for dataset management, authentication, and approval workflows. The implementation includes proper error handling and response formatting.
- **Authentication Integration:** The system incorporates JWT-based authentication with role-based access control, supporting secure user access and protecting sensitive operations.

Cost Constraints:

- **Open-Source Foundation:** Development was completed using open-source technologies (Vue.js, Node.js) as specified, keeping upfront development costs minimal.

- **Future costs:** 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:

- **Component-Based Architecture:** The frontend was built using a modular component-based architecture that enhances maintainability and reusability. Each form step exists as a separate component that can be maintained independently.
- **Centralized Services:** API communications were centralized into service modules, allowing for easier updates and maintenance of data exchange mechanisms.

Usability and Security Constraints:

- **Role-Based Access Control:** The system implements role differentiation between tenant users and administrators, with appropriate access controls for each user type.
- **Progressive Form Submission:** The multi-step form includes progress tracking, draft saving, and intuitive navigation to enhance the user experience.

The implemented design successfully adhered to these specifications and constraints, delivering a solution that meets Capital One's operational needs while providing a secure, efficient, and scalable platform for dataset management.

Section C. Scope of Work

The scope of the Tenant Self-Service Portal project was carefully defined to establish clear boundaries around objectives, timeline, and deliverables. This definition helped maintain focus throughout the development process, reduce ambiguities, and manage expectations effectively. The project scope outlined the team's responsibilities while explicitly noting which tasks fell outside our purview. By clearly defining boundaries around timeline, resources, and deliverables, we successfully prevented scope creep and maintained control over changes.

The development approach followed an agile methodology, allowing for flexibility and regular adjustments as the project progressed. This approach proved valuable as requirements were refined through stakeholder feedback. Throughout the project, the team maintained effective communication with the project sponsor and faculty advisor, enabling the team to keep the scope, timeline, and budget on track.

C.1 Deliverables

The project has successfully produced the following deliverables through our engineering design and development process:

Project Deliverables:

- **Functional Dataset Registry:** We developed a comprehensive portal feature that allows clients to submit datasets and administrators to review and approve them. The registry includes a multi-step submission process with validation at each stage, progress tracking, and draft-saving capabilities.
- **RESTful API Implementation:** We implemented a complete set of RESTful APIs that handle data processing, user authentication, and approval workflows. These APIs provide the foundation for integration with Capital One's systems and demonstrate the workflow functionality through the frontend interface.
- **User Interfaces:** We created Vue.js-based interfaces for both client and administrator users. The client interface includes dataset browsing, multi-step submission forms, and status tracking. The administrator interface provides review and approval capabilities with feedback mechanisms.
- **Testing Infrastructure:** We implemented testing procedures to validate core functionalities, ensuring the platform operates as intended across different use cases and scenarios.

Academic Deliverables Completed:

- Team Contract (Fall Semester)
- Project Proposal (Fall Semester)
- Preliminary Design Report (Fall Semester)
- Fall Poster and Presentation (Fall Semester)
- Final Design Report (Spring Semester)
- Capstone EXPO Poster and Presentation (Spring Semester)

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. **Continued Data Registry Development + Database Configuration (Weeks 17-20)**
Objective: Continued development on Dataset Registry and setup of AWS RDS database.
6. **Refinement and Enhancement (Weeks 21-24)**
Objective: Refine existing features, based on feedback and testing results.
7. **Final Testing and Demo Touch Up (Weeks 25-28)**
Objective: Finalize testing, prepare the final project for presentation, and complete all demo requirements
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:** Figma mockups, sample datasets, 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.

Section F. Challenges and Solutions

Technical Challenges

1. Complex Form State Management:

- **Challenge:** Managing state across multiple form steps while allowing users to save and return to their progress.
- **Solution:** Implemented a centralized form state in DatasetRegistrationWrapper.vue with localStorage persistence.

2. Data Validation Requirements:

- **Challenge:** Implementing comprehensive validation for complex nested form fields.
- **Solution:** Created field-specific validation rules and form-level validation before submission.

3. API Integration:

- **Challenge:** Designing a resilient API communication layer.
- **Solution:** Developed a centralized API service with error handling and retry logic.

Process Challenges

1. Requirement Evolution:

- **Challenge:** Adapting to changing requirements throughout development.
- **Solution:** Implemented an agile development approach with regular stakeholder reviews.

2. Stakeholder Alignment:

- **Challenge:** Ensuring all stakeholders understood the system capabilities.
- **Solution:** Conducted regular demos and updates of system functionality.

Section G. Conclusion and Future Work

Project Summary

The Tenant Self-Service Portal successfully transformed Capital One's dataset registration process from a manual, engineer-dependent workflow to an automated, user-driven system. The implementation meets all key requirements while providing a foundation for future enhancements.

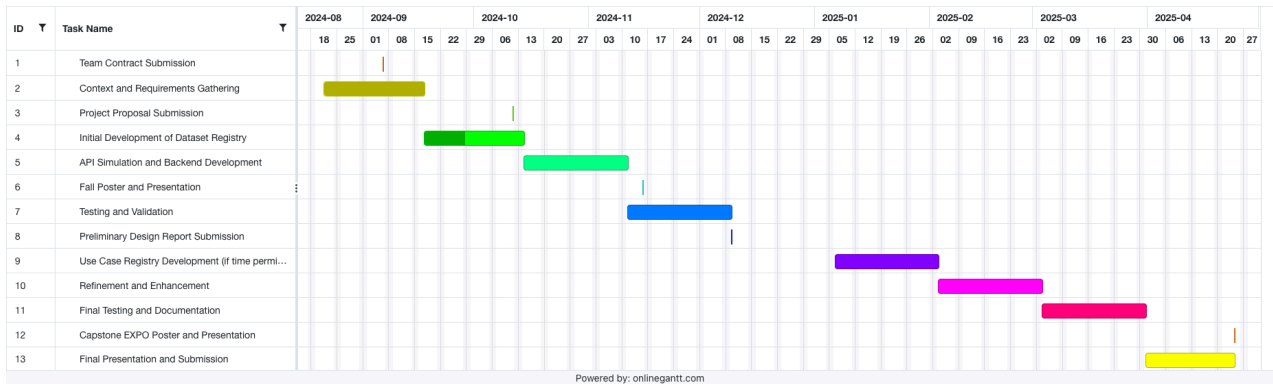
Recommendations for Future Enhancements

1. **Additional Self-Service Capabilities:**
 - Expand the portal to include more tenant management features
 - Develop additional visualization tools for dataset analytics
2. **Integration Enhancements:**
 - Deeper integration with other Capital One systems
 - Automated notification systems for approval status changes
3. **Advanced Analytics:**
 - Add reporting capabilities for dataset usage metrics
 - Implement predictive analysis for resource planning

Lessons Learned

1. **Technical Insights:**
 - Value of component-based architecture for complex forms
 - Importance of centralized state management for multi-step processes
2. **Process Insights:**
 - Benefits of regular stakeholder feedback throughout development
 - Importance of clear documentation for complex workflows

Appendix 1: Project 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 bring to the group	Other Info	Contact Info
<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>	mathurp6@vcu.edu 804-591-7609
<i>Alan Dorn</i>	<i>Effective communication, Organization, software development</i>	<i>Java, Python, C, Databases, SQL, HTML, CSS, JS</i>	dornn@vcu.edu 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>	patelkp7@vcu.edu 757-386-8642
<i>Austin Glass</i>	<i>Effective communication, backend development,</i>	<i>Java, C, C++, Python, Linux</i>	glassal@vcu.edu 804-714-7969

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

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

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

Step 3: Time Commitments, Meeting Structure, and Communication

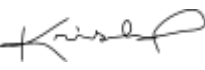
<i>Meeting Participants</i>	<i>Frequency</i> <i>Dates and Times / Locations</i>	<i>Meeting Goals</i> <i>Responsible Party</i>
<i>Students Only</i>	<i>As Needed, On Discord Voice Channel</i>	<i>Update group on day-to-day challenges and accomplishments</i>
<i>Students + Faculty advisor</i>	<i>Meet as needed (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

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

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

