Autodesk – Group 2

Nyein Su Su Thwin

C4033545

**Table of Contents**

[Situation 3](#_Toc157147822)

[Tasks 3](#_Toc157147823)

[Actions 6](#_Toc157147824)

[Result 6](#_Toc157147825)

[Appendices 7](#_Toc157147826)

Situation

The 3D Asset Management System is designed to address the challenges of managing, collaborating on, and version-controlling 3D assets for the automotive design industry. Serving as a centralized platform, it will allow teams to efficiently manage 3D models from concept to production, ensuring streamlined workflows. The system will also support individual projects, incorporating version control and collaboration tools to improve efficiency, minimize errors, and boost productivity.

**Must-have Features:**

* File Management & Version Control: Authorized users (Editors and Project Managers) will have the ability to upload and manage large 3D models, as well as revert to previous versions as needed. Viewers will have read-only access.
* File Previews: Users can preview 3D models without downloading, improving usability.
* Security & Privacy (RBAC): Role-based access control ensures that only authorized users can access or modify assets. File transfers will be encrypted, and two-factor authentication (2FA) will secure user accounts.
* Collaboration & Workflow Management: Real-time notifications will keep users informed about changes. Updates to 3D models require Project Manager approval before being applied. The system will also track project deadlines, asset statuses, and workflows to ensure timely delivery.
* Autodesk Forge API Integration: The API will manage asset versions, making it easy to track, store, and review asset versions.
* Mobile Compatibility: The system will be compatible with mobile devices, supporting various screen resolutions, ensuring accessibility on the go.

**Should-have Features:**

* Project Creation & Asset Management: The system will support various formats, including documents, PDFs, images, and 3D models for diverse asset management.

**Could-have Features:**

* Automated Testing Coverage: Automated tests for 70% of the codebase will ensure maintainability.
* Reset and Amend Features: While rollback is a must-have, features like resetting or amending assets may be considered later.
* Advanced Collaboration Tools: Integration with the Fusion API could enhance tracking of asset changes, though real-time collaboration (e.g., simultaneous editing) is not supported at this stage.

**Key Issues and Mitigation:**

1. Handling Large Files: Autodesk Forge limits model derivatives to 40MB and bucket storage to 100MB. To address this, we will implement a multi-part upload feature, breaking large files into smaller chunks for more efficient uploads. Additionally, file compression will be optimized for faster upload and retrieval speeds.
2. Complexity of Version Control: As asset versions grow, tracking becomes more complex. A user-friendly interface will simplify version history navigation, helping users label and restore versions easily.
3. Security Risks: Unauthorized access to assets can expose sensitive data. To mitigate this, strong security measures, including encryption and Role-Based Access Control (RBAC), will be implemented to safeguard assets and user accounts.
4. Storage Solutions: Autodesk Forge will handle cloud storage for 3D model data using bucket storage. For other data types, such as project files, the storage solution (cloud or local) is undecided and will be determined through a cost-benefit analysis, considering scalability, flexibility, and performance needs.

Tasks

#### **Role and responsibilities**

As the **backend developer**, my primary focus is on developing the server-side of the application, specifically integrating the **Autodesk forge API** to manage 3D models. I chose this role because I am passionate about solving complex problems and wanted to deepen my understanding of Autodesk API and backend development.

****Project planning and initial preparation****

Before development, the team worked on finalizing project specifications. I contributed by drafting non-functional requirements, personas, and user stories. Additionally, I created an initial wireframe for the application's design, which the frontend team later enhanced. I spent time researching Autodesk’s API to ensure smooth integration and also learned PHP to handle server-side operations.

### **Task allocation and collaboration**

To implement the backend effectively, I collaborated with Aryan, another backend developer. We divided responsibilities based on project needs:

* I was responsible for the Upload feature, including file format validation, integrating Autodesk’s API, and storing files in the Autodesk Data Management bucket.
* I also worked on the Viewer feature, integrating Autodesk’s Forge Viewer API for interactive 3D model viewing within the app.
* Aryan handled the Download feature and worked on improving the upload functionality by refining storage and validation mechanisms.

Our collaboration was critical for ensuring smooth integration and resolving any issues during development.

### **Tools and techniques**

For backend development, we utilized the following tools and technologies:

* **PHP** – For writing backend logic and handling API requests.
* **Autodesk forge API** – For managing 3D model uploads, storage, and viewing capabilities.
* **MySQL** – For managing user data and asset metadata.
* **Postman** – For testing API endpoints and debugging API interactions.
* **Git (GitHub)** – For version control and collaborative development.
* **Autodesk data management** – For storing 3D model files in a cloud bucket.
* **Local storage** – For temporarily storing uploaded files before processing with Autodesk API.

### **Software development methodology**

Our team adopted an **Agile** methodology, incorporating **scrum practices** to manage development efficiently.

* **Sprint Planning: We worked in two sprints throughout the semester.**
* **Weekly Scrum Meetings: Discussed progress and issues.**
* **Sprint Review and Client Feedback: Incorporated feedback to refine features.**
* **At Demo Day, we showcased key features like file upload, viewing, and download, incorporating the client’s feedback for further improvements.**

****Conclusion****

**In summary, my backend development contributions, especially in file upload and viewer functionality, were crucial to the project’s success. The AGILE methodology allowed us to adapt and continuously improve, and through collaboration and problem-solving, I gained a deeper understanding of Autodesk’s API and backend systems. This experience has significantly enhanced my development skills and my ability to work in a collaborative team environment.**

Actions

Week 1: Setup & Client Discussion **--** I set up the GitHub repository and participated in defining the project scope. The key challenge was understanding the client’s unclear requirements. We addressed this by drafting specific questions and holding a meeting, as well as communicating via email to clarify the project's purpose, ensuring alignment with both the client and the team.

Week 2: Requirements & Architecture **--** I contributed to finalizing the Requirement Specification document, focusing on non-functional requirements, personas, and user stories. A challenge was aligning technical feasibility with project scope. I addressed this by collaborating with my team to refine the scope and ensure it met both client expectations and technical standards.

Week 3: Design & API Testing Preparation **--** I collaborated with my friend to create a Use Case Diagram and designed five mockups (Home, Project, Model Upload, Login, and Sign-Up) using Moqups, as it allowed us to easily share with the client via link, connect to other pages, and draw quickly. We explored 3D modeling websites for inspiration A key challenge was my lack of API knowledge, so I used Postman to understand it, enabling effective model upload and display integration.

Week 4: API Integration & Testing **--** I focused on understanding Autodesk’s Model Derivative API using Postman. I successfully set up authentication, uploaded files, tested translation, and downloaded files. I explored PHP with Guzzle for 3-legged authentication and, after team discussions, opted for 2-legged authentication, which suited our needs better. Then, I encountered issues such as mismatched callback URLs and login page access, resulting in authentication errors like '400 Bad Request' and 'invalid\_grant' when starting to write with PHP. By referring to the API documentation again and checking my app settings on the Autodesk website, I was able to resolve these issues.

Week 5: In Week 5, I focused on integrating the Autodesk Forge API in PHP for uploading and viewing 3D models. I successfully implemented two-legged authentication to obtain access tokens and created functions for file uploads to a bucket, generating signed URLs for the process. I encountered and resolved issues such as incorrectly handling signed URLs and errors during model translation. Additionally, I worked on the viewer functionality, ensuring that once a user uploaded a 3D model, it was processed and displayed correctly in the Forge Viewer. I connected the backend and frontend by securely passing the translated URN, which was extracted on the frontend to load the model. I also converted the PHP code to JavaScript to handle the translated URN and access token. This ensured a seamless upload-to-viewer workflow, allowing users to visualize their 3D models immediately after uploading. The week involved troubleshooting errors, refining the integration, and preparing for the next steps in the project.

Result

The first sprint of our project was a successful starting point, where we met key milestones, particularly in integrating the Autodesk Forge API for uploading and viewing 3D models. We implemented two-legged authentication, handled file uploads to the Forge bucket, and displayed models in the Forge Viewer. These efforts set the stage for further development, and the API integration successfully handled model uploads and viewings.

For the second sprint, we will focus on addressing the client’s feedback and additional internal tasks. Our action plan includes supporting a wider variety of file formats, implementing thumbnail generation, and updating the model timeline to include version history, issues, and filtering options (such as event type, version and date range). We will also refine the user interface and improve the functionality of the collaborator lists, ensuring they are accessible separately from the timeline. Additionally, we will update the 3D modeling component and consult with the project manager for approval when updates are requested by the editor.

The system needs to handle large file sizes—up to 450MB—without affecting performance. Autodesk Forge has a 40MB limit for model derivatives and a 100MB limit for bucket storage. Since the bucket size cannot be increased, I’ll implement chunked uploads to break large files into smaller parts for efficient uploading. This ensures the system remains performant while managing larger files. The focus will be on enhancing the issue tracking system and supporting large files without performance issues.

Reflecting on the first sprint, I learned the importance of balancing backend and frontend development while staying aligned with project specifications. This will help me prioritize tasks and adjust based on client feedback in the next sprint. During the client review, the feedback was positive, with praise for meeting initial requirements. They also suggested improvements, including expanding file format support, updating thumbnails, and enhancing the model timeline. These improvements will be incorporated into the second sprint to ensure a robust and scalable product.

Appendices

Introduction  
Project Overview  
This project centers on the development of a comprehensive 3D asset management system designed to streamline the organization, collaboration, and lifecycle management of 3D model projects. By enhancing efficiency and workflow optimization, the system aims to address the needs of industries such as automotive design, where effective 3D asset management is critical.  
The primary goal of this system is to provide a centralized platform where teams and individuals can seamlessly manage, organize, and collaborate on 3D assets throughout their entire lifecycle. Users will have the ability to create dedicated project repositories, upload 3D model files, and collaborate with others in real-time. The platform will also feature robust version control, enabling users to track changes, update models, and restore previous versions as needed. By prioritizing both functionality and user experience, this 3D asset management system aims to provide a robust, intuitive, and collaborative environment. It will empower users to work more efficiently, reduce workflow bottlenecks, and enhance productivity across industries that rely on 3D modelling and design.

Problem Statement  
Managing 3D assets in creative and technical projects is often fraught with significant challenges that hinder productivity and collaboration. One of the primary issues is inefficient file management, particularly when dealing with large 3D models, which can lead to performance bottlenecks and slow down workflows. Additionally, the lack of robust version control systems makes it difficult for teams to track changes, update models, or revert to previous iterations,  
resulting in potential errors and inconsistencies. Collaboration is further hampered by inadequate tools that fail to support real-time updates, commenting, or seamless communication among team members. This limitation stifles creativity and slows down decision-making processes. Moreover, inconsistent access control mechanisms often lead to security concerns, as there is no clear way to define user roles or restrict access to sensitive files. These challenges collectively create inefficiencies, increase project timelines, and reduce the overall quality of output, highlighting the need for a more streamlined and collaborative 3D asset management solution.

Project Objectives  
The primary objectives of this 3D asset management system are to provide a robust, scalable, and user-friendly platform that addresses the challenges of managing 3D assets while enhancing collaboration and workflow efficiency. The key objectives include:

**Role-Based Access Control:**  
Implement a secure and flexible role-based access system, allowing collaborators to be assigned specific roles such as editors (who can actively modify and contribute to projects) or viewers (who have limited access to view files for a defined period). This ensures tailored collaboration and safeguards sensitive data.

**Version Control and File Management:**Develop a built-in version control system that enables users to upload and download 3D models, track changes, manage versions, and restore previous iterations. This ensures accuracy, consistency, and the ability to recover from errors throughout the project lifecycle.

**File Sharing with Contractors:**  
Facilitate seamless external collaboration by allowing users to share files with contractors. Contractors will have limited access to specific files for a defined period, with the ability to suggest changes and edits, ensuring smooth integration of external contributions.

**Scalability and Performance:**  
Design the platform to handle large volumes of 3D assets efficiently, ensuring optimal performance even for complex, multi-faceted projects. The system will be scalable to accommodate individual creators, small teams, and large organizations, while maintaining data security and reliability.

**Collaboration and Permissions:**  
Enable easy and efficient collaboration through role-based permissions, ensuring that users with different levels of access (e.g., editors, viewers) can work together seamlessly. This includes tools for real-time effective communication to enhance teamwork.

**User Experience and Accessibility:**  
Provide a user-friendly and intuitive interface that meets accessibility standards, ensuring the platform is usable by a wide range of users, including those with disabilities. The system will prioritize ease of use without compromising on advanced functionality.

**Collaboration Workflow Management:**  
Include features to track project deadlines, monitor asset statuses, and manage workflows effectively. This will help teams stay organized, meet project milestones, and ensure timely delivery of 3D assets.

# System Requirements Non-Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Theme** | **Description** | **Priority (MoSCoW)** |
| **NFR-01** | Performance | The system must handle file uploads and downloads of up to **450MB** in size within **10 seconds** under normal load conditions. For files larger than 450MB, the system should provide some form of progress tracking. | Must |
| **NFR-02** | Security | All file uploads, downloads and metadata changes must be encrypted.  User authentication must support **two-factor authentication (2FA)** and enforce strong password policies. | Must |
| **NFR-03** | Privacy | The system must enforce **role-based access control (RBAC)** to ensure private projects and files are accessible only to authorised users. | Must |
| **NFR-04** | Usability | The user interface must be intuitive, allowing first-time users to perform key actions (e.g., upload, view, comment on files) within **2 minutes** without external documentation. | Must |
| **NFR-05** | Collaboration | The system must support real-time collaboration features, including **comments, notifications, and activity feeds**. Notifications must be delivered within **5 seconds** of an update or change. | Must |
| **NFR-06** | Version Control | The system must store and manage **all versions of 3D models**, including metadata (e.g., timestamp, user who made the change, and a brief description of changes). Select users must be able to restore previous versions within **3 clicks**. | Must |
| **NFR-07** | Storage | The system must ensure data durability with 99.99% reliability, maintaining data integrity during multi-user collaborations. | Must |
| **NFR-08** | Compatibility | The system must be compatible with **modern browsers** (Chrome, Firefox, Edge, Safari) and support devices with screen resolutions starting from **1280x720 pixels**. It must also support **mobile responsiveness** for screens as small as **320px**. | Must |
| **NFR-9** | File Preview | The system should support **file previews** for common 3D model formats (e.g., .obj, .stl) without requiring users to download the file. | Must |
| **NFR-10** | Maintainability | The system must be modular, with clear separation of concerns. At least **70% of the codebase** must be covered by automated tests. | Could |

**User Roles**

System – Level Roles

|  |  |
| --- | --- |
| **Site – User Role** | **Description** |
| System Administrator | A registered user with elevated privileges to manage the system, users, and projects.  Abilities:  User Management: • Add, suspend, ban, or delete registered users. • Manage user profiles (e.g., update details, reset passwords). • Approve or reject user registration requests based on system policies.  Content Moderation: • Receive notifications about project or user reports (e.g., inappropriate content, user disputes). • Review and take action on user-generated reports. • Remove or restrict access to inappropriate or policy-violating content.  System Maintenance: • Monitor system performance (e.g., server health, storage).  Policy Enforcement: • Ensure compliance with site policies, guidelines, and legal requirements. • Audit user activity logs for suspicious or policy-violating behavior.  Reporting and Analytics: • Generate system-wide reports (e.g., user activity,). |
| Contractors | Non-registered users who are invited and verified via email to access specific projects and their contents. Contractors are granted temporary access by Project Administrators or Project Managers and have limited interaction with the system.  Abilities:  Access Control:  • Access is granted via a verified email link invitation. • Time-limited access to specific files or projects (e.g., 7 days, 30 days). • Access can be revoked by the Project Administrator at any time.  Limited Interaction: • Can only view or download files as permitted by the Project Administrator or Project Manager. • Cannot create, delete, or manage projects. • Cannot invite other users or contractors to the project.  Editing Capabilities: • Can create edits or comments on files. • Edits are submitted as a commit change request and must be approved by a Project Administrator or Project Manager before being finalized. • Cannot directly modify or overwrite files without approval.  Notifications: • Receive notifications about project updates or changes relevant to their access. |
| Registered Users | Users with registered accounts who can participate in and create projects. They are further divided into four roles within projects: Project Administrator, Project Manager, Project Editor, and Project Viewer.  Abilities:  • Manage their user profiles and details. • Create and manage projects. • Participate in multiple projects with varying roles. • Receive notifications about project updates, comments etc. |
| Site Guest | A non-registered user who can browse the public-facing pages of the site and create an account to become a Registered User. Site Guests have limited access to the system and are primarily intended to explore the platform before committing to registration. • Sign up to become a registered user. • Browse the site (view FAQs, Privacy Policy etc.) |

Organisation – Level Roles

|  |  |
| --- | --- |
| Organization - User Roles | Description |
| Organization Creator | A registered user, who creates the organization and has full control over its settings and members.  Organization Management: • Create, edit, or delete the organization. • Set the organization’s name and details. • Transfer ownership of the organization to another member.  Member Management: • Create accounts for new members. • Assign roles (e.g., Organization Manager, Organization Member) during account creation. • Edit or deactivate member accounts. • Resend invite emails if needed. • Suspend or ban members from the organization.  Project Management: • Create, archive, or delete organization-level projects. |
| Organization Manager | A registered user with elevated permissions to manage the organization’s members, projects, and content.  Member Management: • Create accounts for new members. • Assign roles (e.g., Organization Member) during account creation. • Resend invite emails if needed.  Notifications: • Receive notifications about project activity (e.g., new uploads, version updates). |
| Organization Member | • Registered users, who are regular members under the organization, that can participate in the organization’s projects.  Member Management: • Cannot create accounts for members.  Project Participation: • Access and participate in organization projects based on their assigned roles (e.g., Project Editor, Project Viewer). • Cannot create or delete organization-level projects.  Notifications: • Receive notifications about project updates or changes relevant to their access. |

Project – Level Roles

|  |  |
| --- | --- |
| Project - User Roles | Description |
| Project Administrator | A registered user with elevated permissions to manage a specific project they have created.  They inherit all the abilities of the Project Manager and Project Editor roles.  Abilities:  Project Creation and Management: • Create new projects. • Define project metadata (e.g., name, description). • Archive (to asset library) or delete projects when no longer needed.  Team Management: • Invite and manage project members. • Assign roles (Project Manager, Editor, Viewer, Contractor) to collaborators. • Set permissions for each collaborator (e.g., viewer, editor, manager or contractor).  Notifications: • Receive notifications about project activity (e.g., new uploads, version updates, member invitations). |
| Project Manager | A registered user with elevated permissions to manage project workflows and team members. They inherit all the abilities of the Project Editor role.  Abilities:  Version Control: • Rollback to previous versions of files (must provide a comment explaining the reason for rollback). • Manage version history and ensure version integrity. • Approve or reject version commit requests and edits.  Content Management: • Oversee file organization and naming conventions.  Notifications: • Receive notifications about file changes, comments, or team activity. |
| Project Editors | A registered user with permissions to upload, organize, and manage project files.  Abilities:  File Management: • Push a request to upload, rename, move, organize or delete files within the project (must provide a comment explaining the changes). • Download project files locally.  Version Control: • View previous commit versions, comments and issues of the project. • Push a request for newly uploaded versions of files with clear version comments (through a version commit).  Collaboration: • Collaborate with other collaborators to ensure file integrity.  Notifications: • Receive notifications about changes to the project (e.g., new file uploads, version updates). |
| Project Viewers | A registered user with read-only access to project files.  Abilities:  View-Only Access: • Can view or download files as permitted by the Project Administrator. • Cannot make edits, upload files, or modify project content.  Interaction: • Can view commits and comments or raise issues (e.g., raise issue comments). • Cannot manage or organize files.  Notifications: • Receive notifications about project updates or changes relevant to their access. |

**Personas**

|  |  |
| --- | --- |
| **John** | |
| * Age: 22 * Occupation: Computer Science Student * Location: Sheffield, UK * Project Type: Team Project – Collaborative 3D Modelling * Disability: None * Quote: “I need a platform where I can easily manage and collaborate on my 3D models with my team, without worrying about complicated workflows or inaccessible tools.” | |
| **Background** | John is a second-year Computer Science student with a passion for 3D modelling. He frequently works on team projects, where he collaborates with peers to create and refine 3D models for assignments and personal development. John values efficiency, clear communication, and seamless collaboration. He often uses tools, like Fusion 360, for modelling but struggles with the lack of integration between his tools and collaboration platforms. |
| **Goals** | * Easily upload, organise, and manage 3D models for team projects. * Collaborate effectively with team members by sharing files, providing feedback, and tracking changes. * Review comments for versions from peers for models submitted to the platform. * Access all platform features quickly and intuitively without unnecessary steps or confusion. * Stay updated with real-time notifications about changes in team projects and feedback from peers. * Maintain a clear version history of 3D models to avoid confusion and ensure everyone is working on the latest version. |
| **Frustrations** | * **Inefficient Workflow**: John dislikes that he can’t update 3D models directly on the platform. He must download, modify, and re-upload, which disrupts his workflow. * **Version Control Issues**: Managing multiple versions is confusing, especially when he needs to track updates after repeatedly downloading and uploading models. * **Lack of Real-Time Collaboration**: It’s difficult to collaborate and get feedback from team members, as there’s no real-time way to discuss model updates on the platform. * **Limited Integration**: The platform doesn’t integrate well with his preferred 3D modelling tools (e.g., Fusion 360), forcing him to use workarounds. * **Poor Feedback System**: The current feedback system is not suitable, making it hard to track comments and suggestions from teammates. |
| **Technology** | **Devices**: Laptop (Windows, powerful enough for 3D modelling software)  **Technology**: Fusion 360 (for creating 3D models) |
| **Scenario** | John logs into the platform to upload a new version of his model for team review. The intuitive drag-and-drop interface makes it easy to upload files, and he receives real-time notifications about each upload. After sharing the model, he uses the built-in commenting system to discuss changes with his team. John checks the version history to ensure everyone is working on the latest version and reviews any feedback to gauge his team’s perception of the model. He appreciates the seamless integration with his external applications (e.g. Fusion 360), which allows him to make quick edits without leaving the platform. |

**User Stories**

|  |  |  |
| --- | --- | --- |
| **Title**: Upload 3D Model | **Priority**: Must | **Estimate:** 5 points |
| **As a:** Registered User  **I want to:**Upload my 3D model file into an already created project repository  **So that:**I can access it via the site and possibly share it with my team. | | |
| **Acceptance Criteria**  **Given,** I am on the project repository page,  **When,**I click the "Upload Files" button and select a or drag in a 3D model file (e.g., .obj, .stl),  **Then,**thesystemshould**:**   * Validate the file format (e.g., .obj, .stl) and size (maximum **450MB**). * Display an error message if the file format or size is invalid. * Successfully upload the file to the correct repository. * Notify me with a success message (e.g., "File uploaded successfully"). * Update the repository file list to include the uploaded model. * Provide options to **view**, **download**, or **update** the uploaded model. | | |

|  |  |  |
| --- | --- | --- |
| **Title**: Download 3D Model | **Priority**: Must | **Estimate:** 5 points |
| **As a:** Registered User  **I want to:**Download a 3D model file from my project repository  **So that:**I can save it locally to my device and work on it offline. | | |
| **Acceptance Criteria**  **Given,**I am on the project repository page,  **When,**I select a 3D model file and click the "Download" button below the viewer,  **Then,**the system should:   * Validate that the file exists and is accessible. * Initiate the download process and save the file to my device in its original format. * Notify me with a success message (e.g., "Download complete"). * Ensure the downloaded file is not corrupted and matches the original file. | | |

|  |  |  |
| --- | --- | --- |
| **Title**: Version Control | **Priority**: Must | **Estimate:** 5 points |
| **As a:** Registered User  **I want to:**View and manage previous versions of my 3D model files  **So that:**I can track changes and revert to an older version if needed. | | |
| **Acceptance Criteria**  **Given,**I am on the project repository page,  **When,**I upload a new version of a 3D model file,  **Then,**the system should:   * Automatically create a new version entry in the version history. * Store metadata for each version (e.g., timestamp, user who uploaded it, and a brief description of changes). * Allow me to view and compare previous versions. * Provide an option to restore a previous version with one click. * Notify team members of the new version upload. | | |

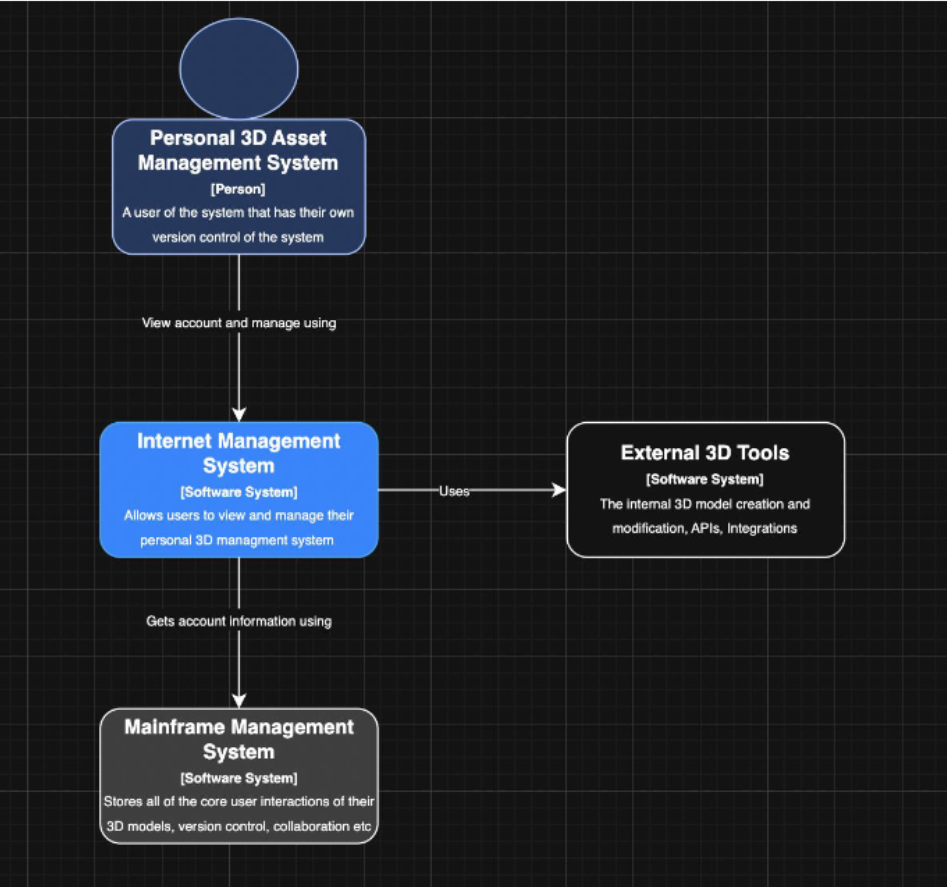
|  |  |  |
| --- | --- | --- |
| **Title**: Version History and Rollback | **Priority**: Must | **Estimate:  8** points |
| **As a:** Project Manager  **I want to:** View the version history of a 3D model and restore a previous version **So that:** I can recover from mistakes or compare changes | | |
| **Acceptance Criteria**  **Given,** I am on the project repository page,  **When,** I select a 3D model file and scroll to “Version History”, **Then,**thesystemshould**:**  • Display a list of all versions with metadata (timestamp, user description) • Allow me to preview and compare versions, with their relevant comments & raised issues • Provide a “Restore” button to revert to a previous version, requiring a commit comment • Notify team members of the rollback with a reason | | |

|  |  |  |
| --- | --- | --- |
| **Title**: Navigate Pages | **Priority**: Must | **Estimate:  4** points |
| **As a:** Project Administrator or Project Manager  **I want to:** Easily navigate between different pages (e.g. project repository, settings, activity log)  **So that:** I can quickly access the information or tools I need | | |
| **Acceptance Criteria**  **Given,** I am logged into the platform **When,** I click on a navigation link (e.g. “Projects”, “Create”)  **Then,**the system should:  • Load the requested page within 2 seconds • Highlight the active page in the navigation menu • Display the correct content for the selected page • Maintain my session and permissions throughout the navigation | | |

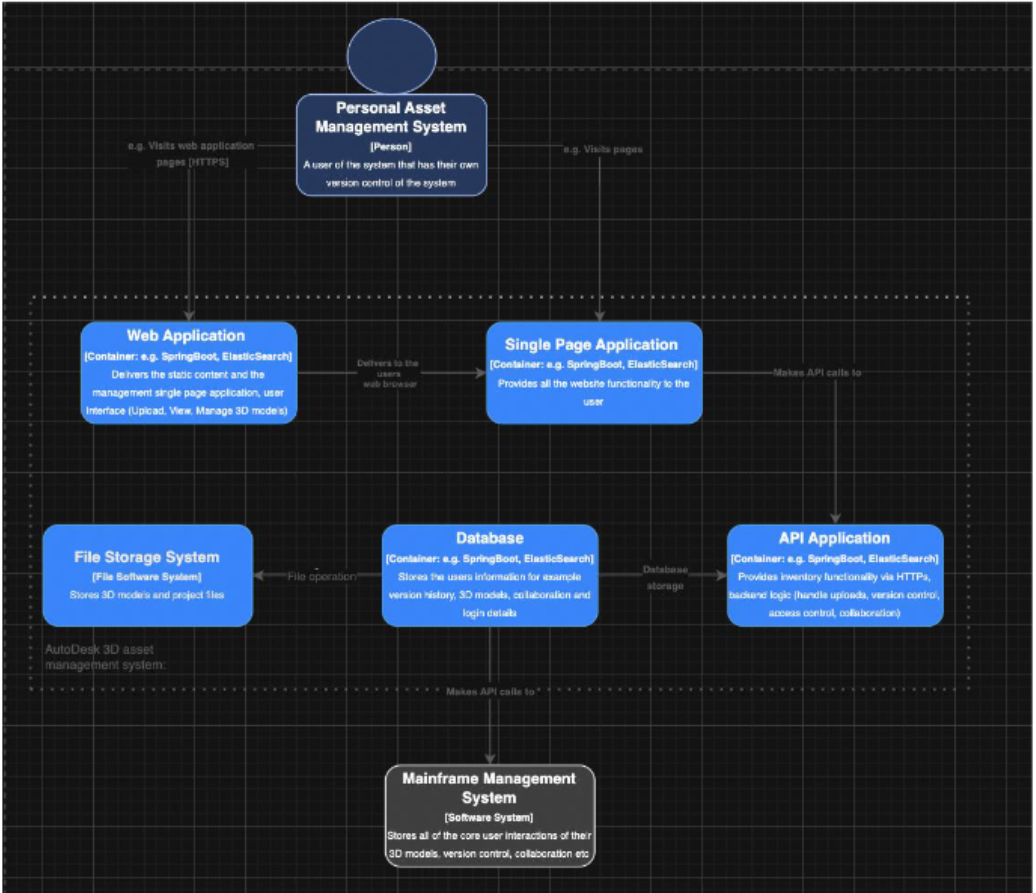
|  |  |  |
| --- | --- | --- |
| **Title**: View Project Details | **Priority**: Must | **Estimate:  4** points |
| **As a:** Project Administrator or Project Manager  **I want to:** View the project name, brief description, and key details when I click on a project **So that:** I can quickly understand what the project is about | | |
| **Acceptance Criteria**  **Given,** I am on the projects list page **When** I click on a project **Then,**the system should:  • Display the project name and brief description at the top of the page • Show key details (e.g. collaborators, creation date, owner) • Load the project details within 2 seconds | | |

**System Architecture**

**C4 Context Diagram (level 1):**

****

**C4 Context Diagram (level 2):**



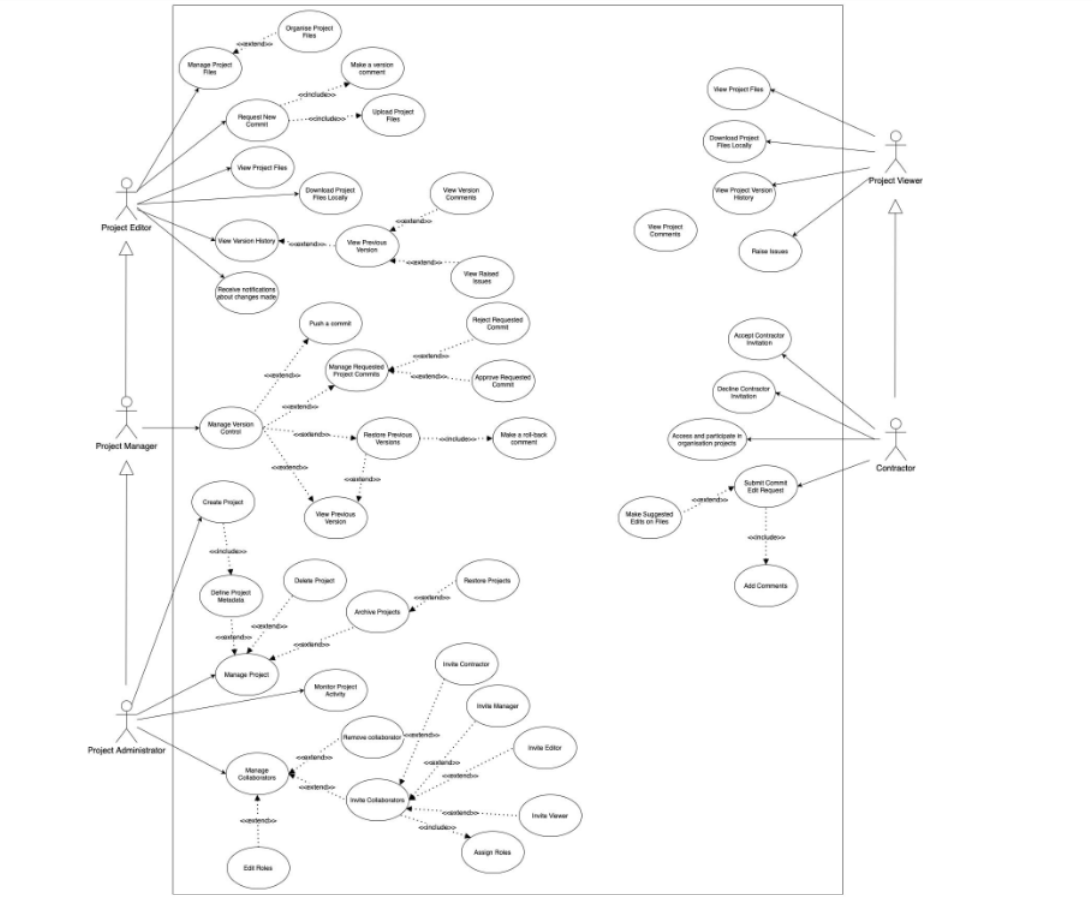
**Use Case Diagram:**

**System Level - Use Case Diagram**

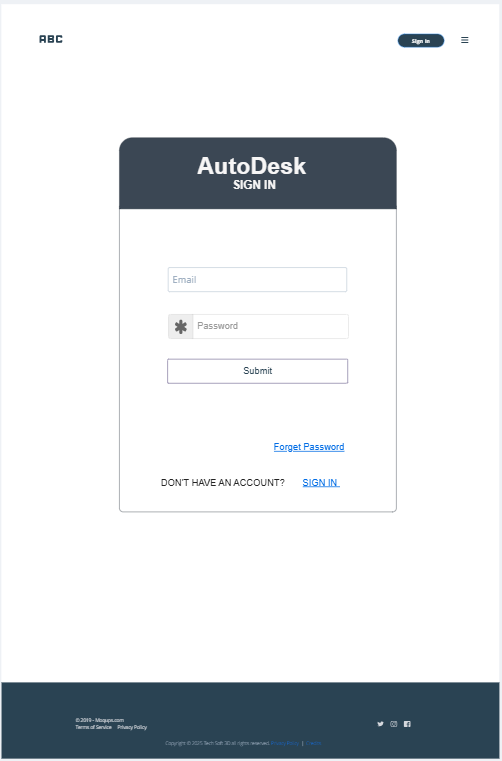
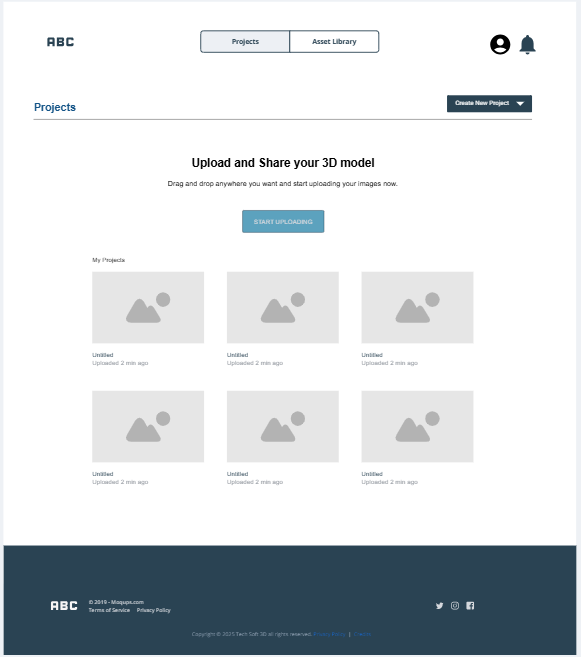
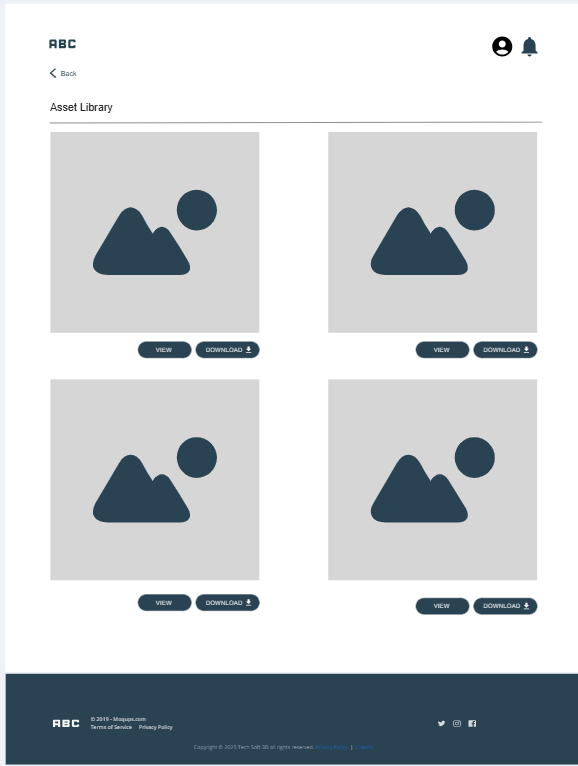
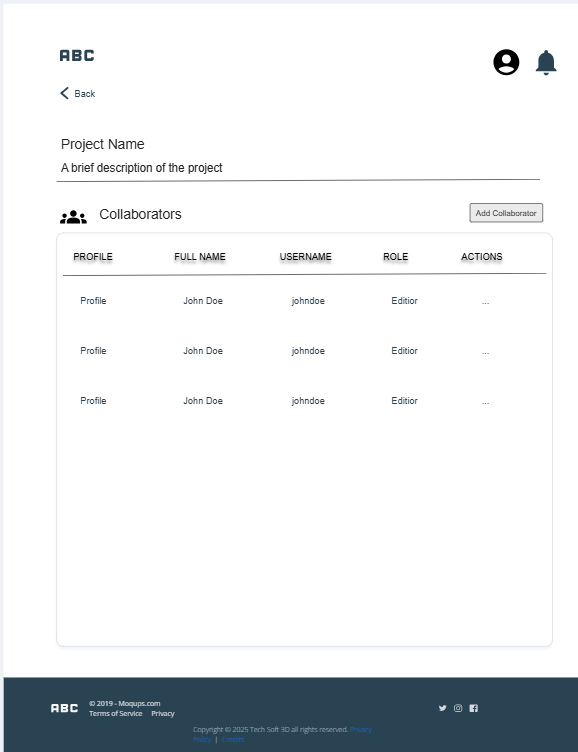
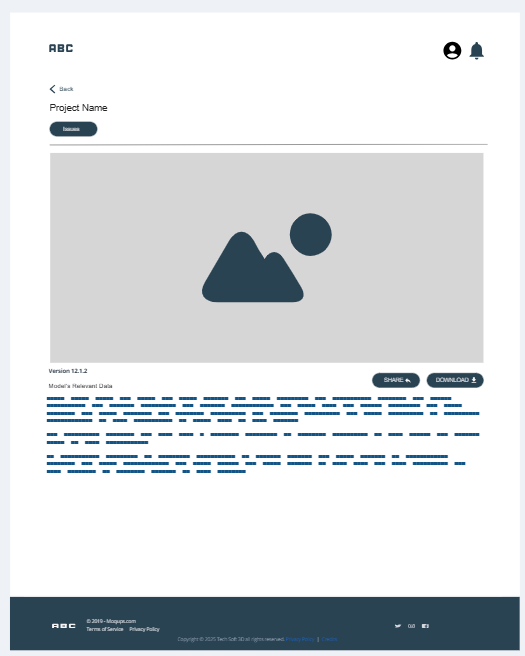
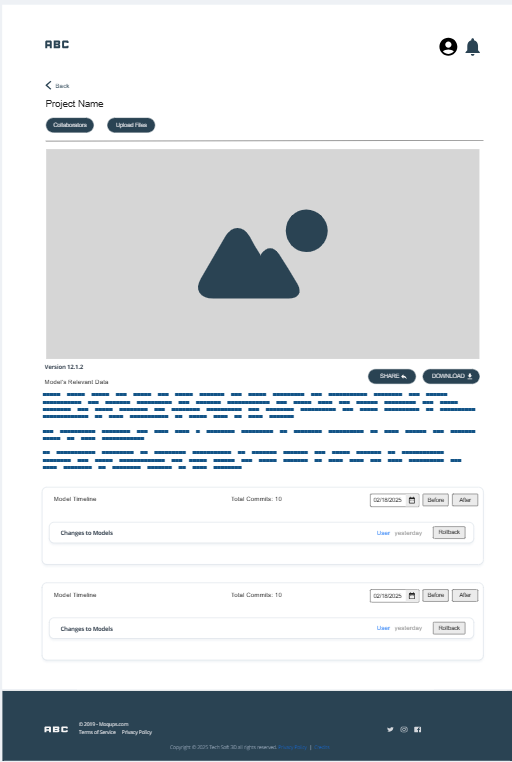
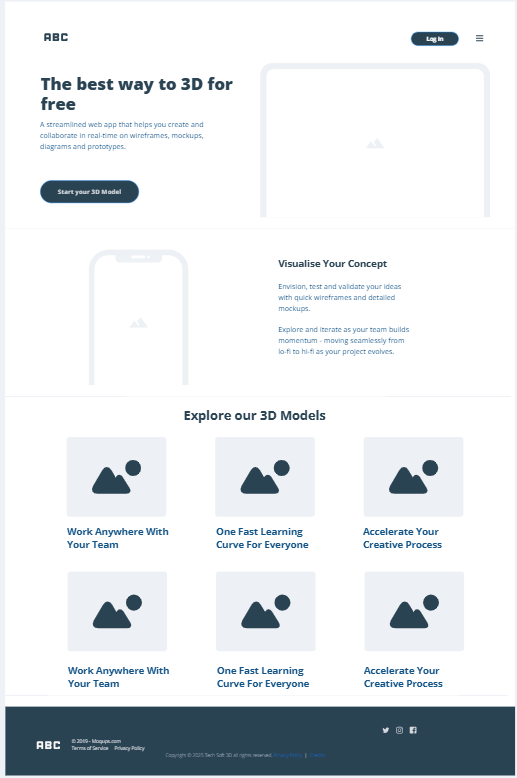


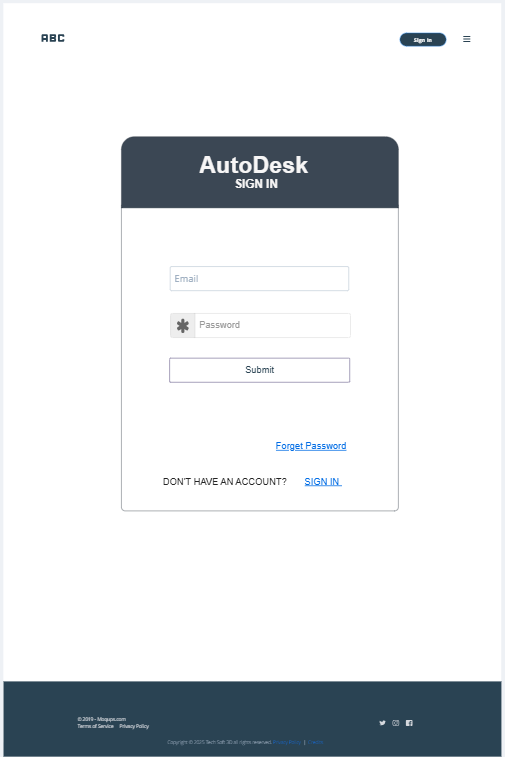
**Organization Level - Use Case Diagram**

**Project Level - Use Case Diagram**



**Wireframes:**





**Appendix 1: Tasks Completed**

*Table 1-Tasks Completed by each Member:*

|  |  |
| --- | --- |
| **Name:** | **Tasks Completed:** |
| **Nadia Sajjad** | - Project Overview/Statement/Objectives - User roles - User personas - User stories - Non-functional Requirements - Use case diagram - Wireframes - Front-End Design - Learned about Autodesk API - Database - Database connection - User Sign Up - User Log In - Assisted in file upload API integration with front-end - Assisted in viewer API integration with front-end - Updated Project Specification |
| **Misbah Khan** | - Project Overview/Statement/Objectives - User Stories - C4 context diagram level 1 - C4 container diagram level 2 - Use case diagram - Wireframes - Front End Design - Introduction - Database - Database connection - Assisted in file upload API integration with front-end - Assisted in viewer API integration with front-end - Updated Project Specification |
| **Aryan Verma** | - Learned about Autodesk API - Upload 3D models feature - Thumbnail Generation for 3D model - Download 3D model feature |
| **Nyein Su Su Thwin** | - Initial Non-Functional Requirements - Initial User personas - Initial User stories - Learned about Autodesk API - Upload 3D models feature - View 3D models page - Wireframes |
| **Su Eaindray Win** | - Learned about Autodesk API - Assisted on API development - Collected sample 3D model - Database connection - C4 container diagram |

**Appendix 2: GitHub Repository**

*Table 2 - GitHub Usernames*

|  |  |
| --- | --- |
| **Student name** | **GitHub name** |
| Nadia Sajjad | nadia-sn3 |
| Nyein Su Su Thwin | nyeinsu4520 |
| Misbah Khan | misbxh0 |
| Aryan Verma | AryanVerma467 |
| Su Eaindray Win | SuEaindrayWin |