CS3354 Software Engineering
Final Project Deliverable 1

Asset Management Software

Anna Miranda
Chelsea Heredia
Vincent Nguyen
Pooja Ganapathy
Jacob Guiang
Mason Kuehne
Ross Richards

Due: 21 October 2022

1. [5 POINTS] Please attach here the **Final Project draft description (that contains the instructor feedback)**. It is ok to include a picture of the original document.
Address the feedback provided for your proposal by listing what you did / plan to do to comply with those proposed changes and or requests for additions to your project.

Project Description:
We will be developing asset management software. Some of its features will be handling different user roles with different access levels, linking files together with a tagging system, and capability to invite external users to modify certain files. The software will organize and store files by a hierarchical system. The system will support importing and exporting data with different file types. There will be a search tool that would allow searching by keywords & phrases. The system will also be able to support SQL queries. Another feature would be implementing a customizable data visualization tool within the system.

Instructor Feedback:
Good choice for a topic! Asset management software is a promising tool that will help users wisely and confidently. Lots of useful inherent parts are mentioned such as database search feature, hierarchical storage, ability to allocate a multi-user system which are all making your design very powerful and unique.

A great detailed task delegation, thank you for that.

In the final report, please make sure to include comparison with similar applications -if any-, make sure that you differentiate your design from those, and explicitly specify how.

2. [10 POINTS] Setting up a **Github repository**. Please use your utdallas email accounts only for each group member.

https://github.com/AMirandaUTD/3354-AssetManagement/

1.1. Each team member should create a GitHub account if you don't already have one.
1.2. Create a GitHub repository named 3354-teamName. (whatever your team name will be).
1.3. Add all team members, and the TA as collaborators. Our TA will post his GitHub info on EL
1.4. Make the first commit to the repository (i.e., a README file with [team name] as its content).
1.5. Make another commit including a pdf/txt/doc file named "project_scope". If you choose a predefined topic (one of the 4 topics described in the "Project Topic Ideas" section of this document), the contents of the file should be identical to the corresponding project in this section. If you choose other topics, the contents should follow a similar structure.
1.6. Keep all your project related files in your repository as we will check them. Include the URL of your team project repository into your project deliverable 1 report.
Important Note:
• Tasks 1.3 - 1.5 should be performed by different team members. We will check the commit history for these activities.
• Do not include credentials (e.g., UTD ID) in the repository.
• Only commits performed before the deadline will be considered. Do not forget to push your changes after you have done the work!

3. [5 POINTS] **Delegation of tasks**: Who is doing what. If no contribution, please specify as it will help us grade each group member fairly.

Create Project Scope document (similar to examples in project info doc) - Pooja Ganapathy
What software process model to use & why - Mason Kuehne
Software requirements list (5-7 functional, all non-functional types) - Chelsea Heredia
Use case diagrams (multiple) - Vincent Nguyen
Sequence diagrams (multiple) - Anna Miranda
Architectural design - Jacob Guiang
Class diagram (one only) - Mason, Anna, Pooja, and Chelsea

4. [5 POINTS] Which **software process model** is employed in the project and why. (Ch 2)

We should use the **incremental process model** because our project will end up with many different features that should be implemented at different times. For example, our first iteration may simply store user files, but iterations after that may implement a database search feature, hierarchical storage, and the ability to allocate a multi-user system. Additionally, this process model is not too hard to work with, which will benefit our team considering this is our first time working together.

5. [15 POINTS] Software Requirements including
5.a.) [5 POINTS] **Functional requirements**. To simplify your design, please keep your functional requirements in the range minimum 5 (five) to maximum 7 (seven). (Ch 4)

7 Functional Requirements that describe functionality and system services.

1. The system shall store assets and data files in a hierarchical manner, allowing the storage to be sorted by file name or data modified.
2. The system shall organize data into categories tagged by category labels.
3. A user shall be able to search the database and storage of all assets within the system by specifying a keyword or phrase.
4. Each person using the system shall be uniquely identified by their identification number and login, specifying their access level.
5. The system shall only generate available files based on the cleared access level of the user.
6. The system shall support SQL queries.
7. The system shall support importing and exporting data files of different types, such as .txt, .xlsx, .pdf, .doc, etcetera.

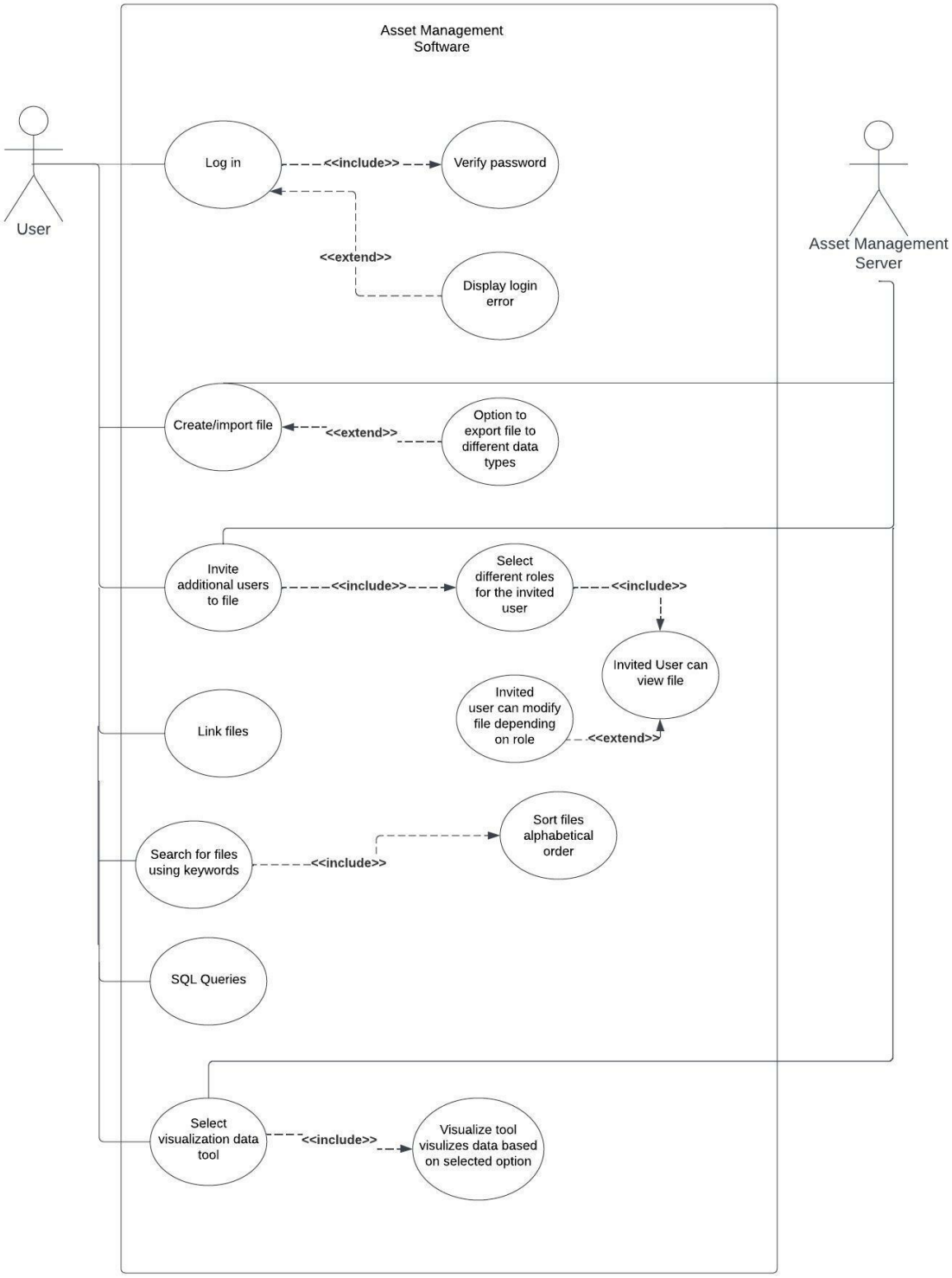
5.b.) [10 POINTS] **Non-functional requirements** (use all non-functional requirement types listed in Figure 4.3 - Ch 4. This means provide one non-functional requirement for each of the leaves of Figure 4.3. You can certainly make assumptions, even make up government/country based rules, requirements to be able to provide one for each. Please explicitly specify if you are considering such assumptions.)

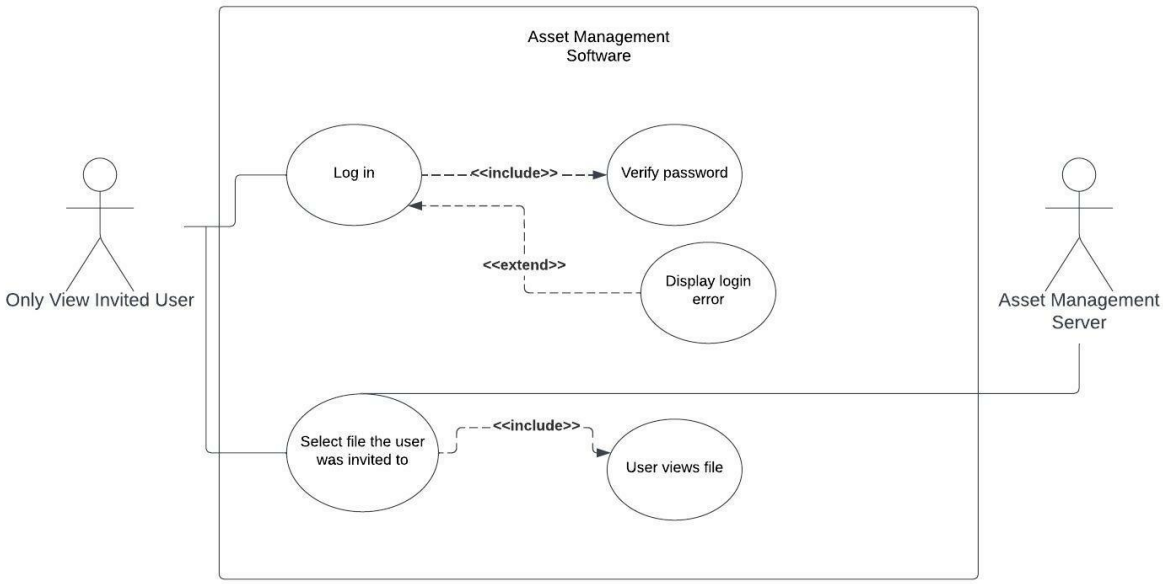
Non-Functional Requirements that define system properties and constraints.

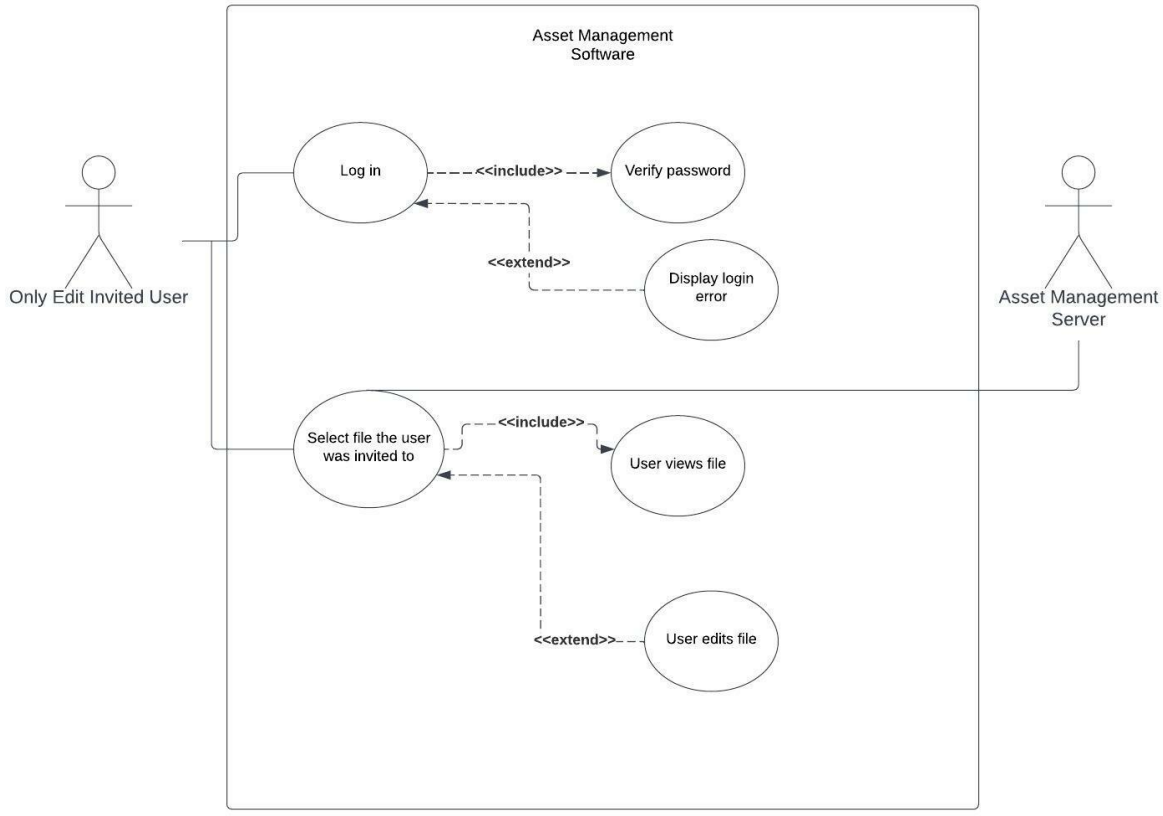
- **Usability Requirement**: Users shall be able to customize data visualization, such as by prioritizing files or selecting a viewing theme.
- **Performance Requirement:** The system shall generate search results of data queries within five seconds of search time.
- **Space Requirement:** The asset management system shall be able to store a maximum of 1TB storage, approximating a storage of 200,000 documents.
- **Dependability Requirement:** The system shall be available to all users during normal waking hours each weekday (Mon-Fri, 05:00-23:59). Downtime shall not exceed five seconds in any one day outside of maintenance hours each week (1x/week, 00:00-04:59).
- **Security Requirement:** The system shall contain encrypted file storage and sharing.
- **Environmental Requirement:** The team shall use the Visual Studio Code IDE for local environment development, and they shall push changes to the remote repository hosted on GitHub.
- **Operational Requirement:** Users shall authenticate themselves using a secure company login, including identification PIN.
- **Development Requirement:** The product shall be developed using common web framework technologies, including but not limited to: React.js, Node.js, jQuery, and/or Express.

- **Regulatory Requirement:** The system shall allow efficient and effective modifications to development of existing datasets and files without compromising system quality or other assets, as approved by regulators in upper management.
- **Ethical Requirement**: The system shall be implemented in the highest standard acceptable to user expectations and to the general public, as abiding by the IEEE Code of Ethics.
- **Accounting Requirement:** The system's user data collection will abide by national laws enforced by the U.S. Federal Trade Commission, including but not limited to: The Children's Online Privacy Protection Act (COPPA), the Gramm Leach Bliley Act (GLBA), and the Fair Credit Reporting Act (FCRA).
- **Safety/Security Requirement:** The system shall allow user authentication and password management for user account safety and security.

6. [15 POINTS] **Use case diagram** – Provide a use case diagram (similar to Figure 5.5) for your project. Please note that there can be more than one use case diagrams as your project might be very comprehensive. (Ch 5 and Ch 7)

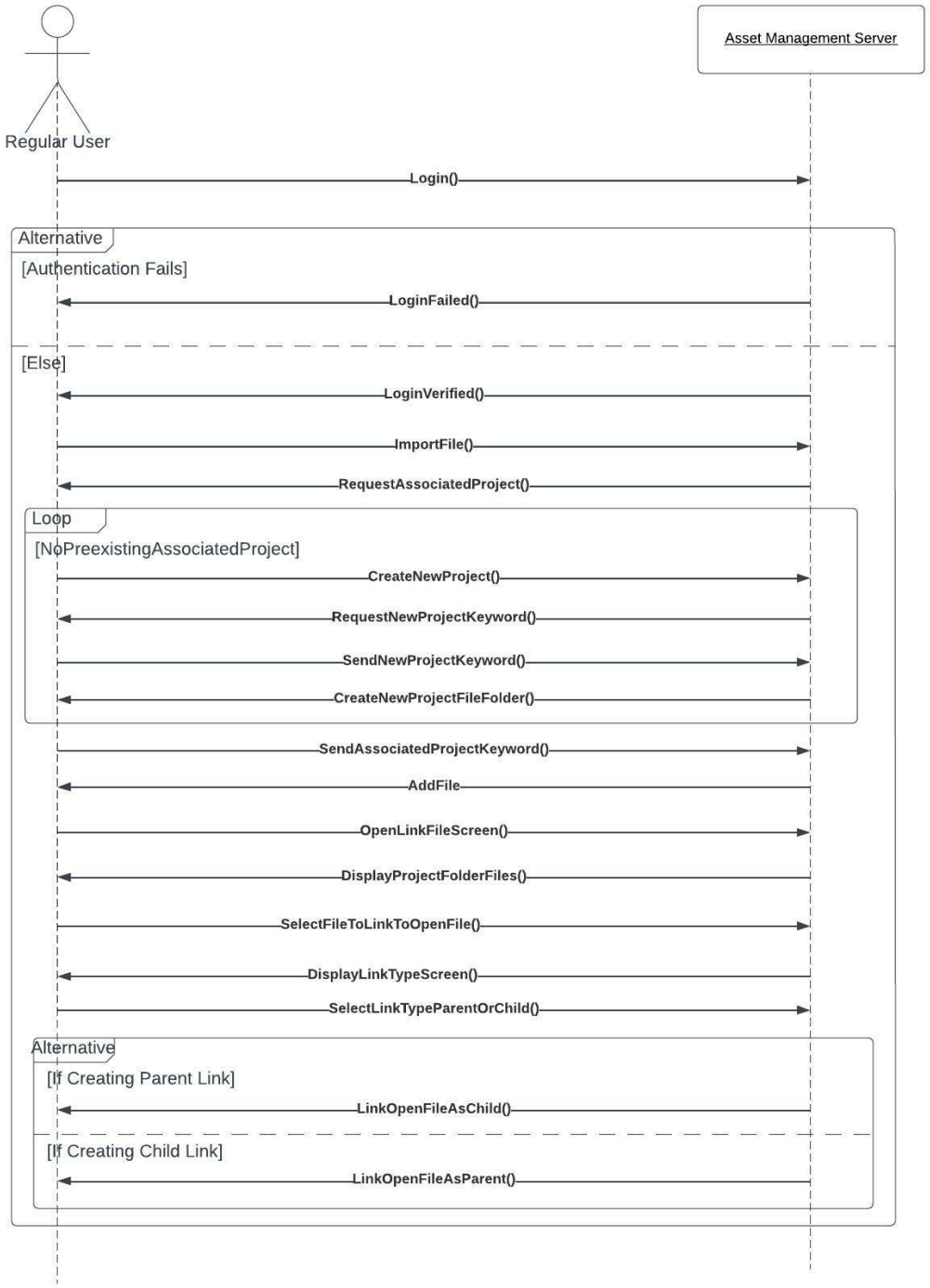

Asset Management Software
User
Asset Management Server
Log in
<<include>>
Verify password
<<extend>>
Display login error
Create/import file
<<extend>>
Option to export file to different data types
Invite additional users to file
<<include>>
Select different roles for the invited user
<<include>>
Invited User can view file
Invited user can modify file depending on role
<<extend>>
Link files
Search for files using keywords
<<include>>
Sort files alphabetical order
SQL Queries
Select visualization data tool
<<include>>
Visualize tool visulizes data based on selected option

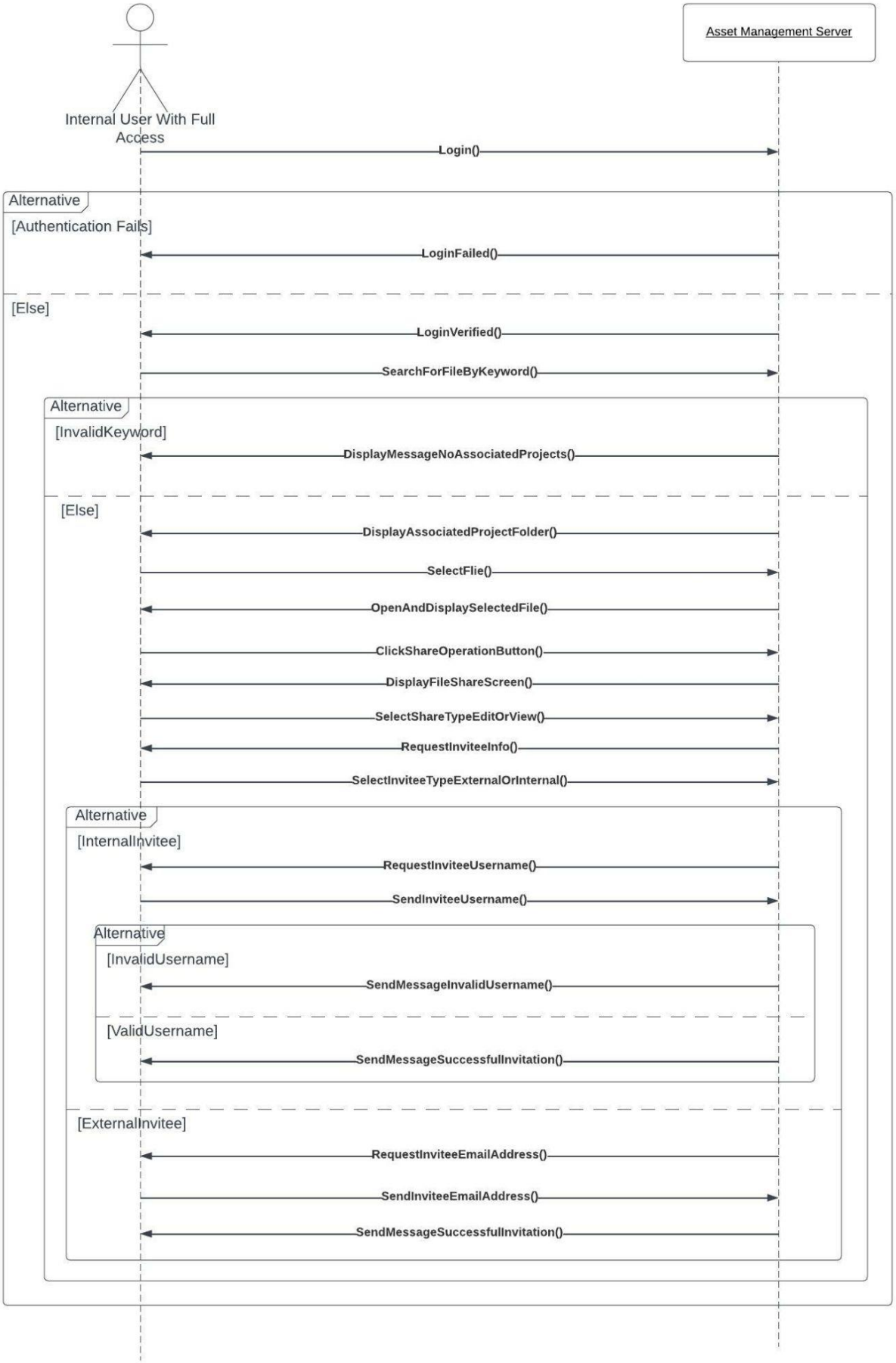

Asset Management Software
Log in
<<include>>
Verify password
<<extend>>
Display login error
Only View Invited User
Asset Management Server
Select file the user was invited to
<<include>>
User views file

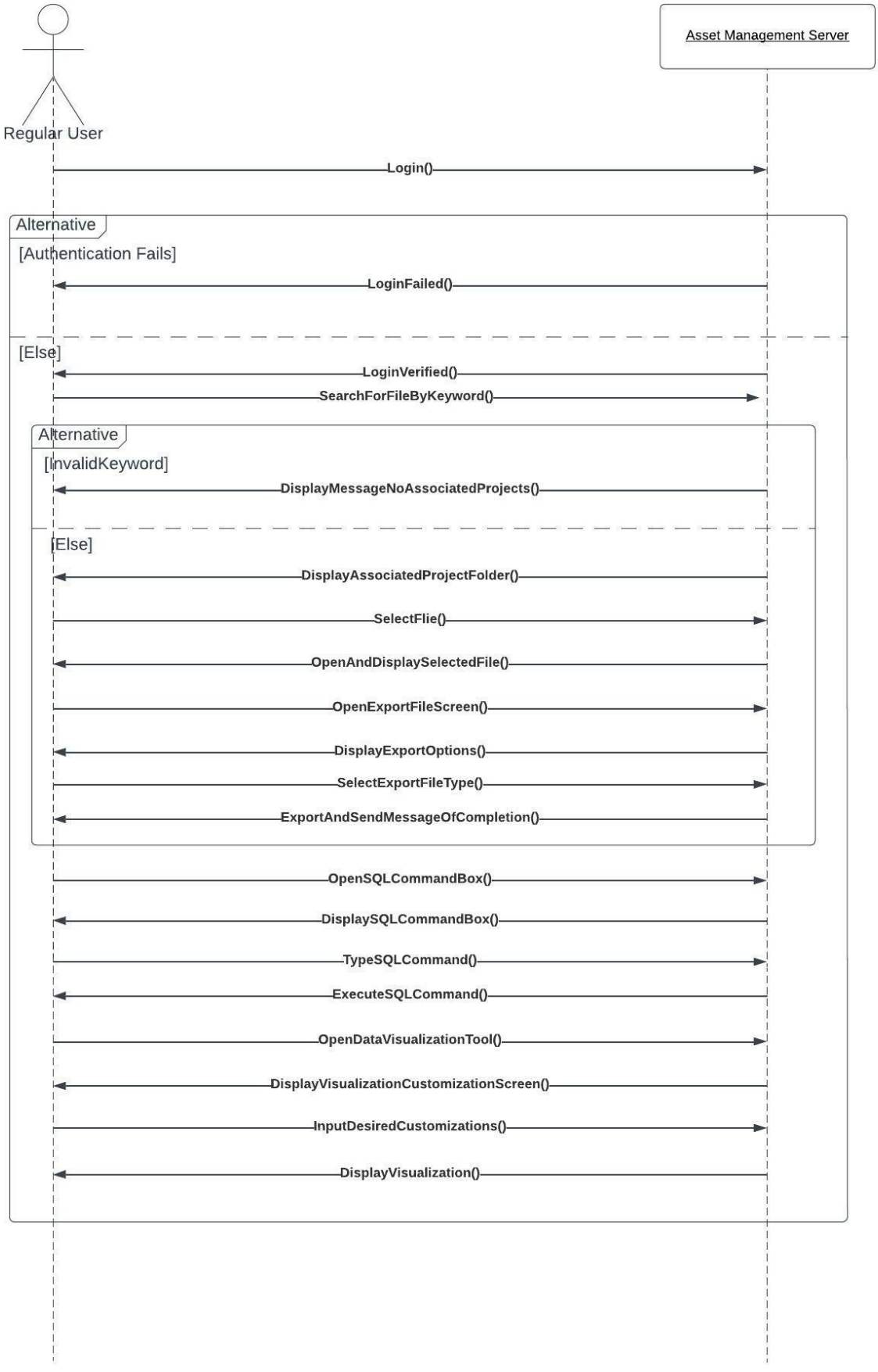

Asset Management Software
Log in
<<include>>
Verify password
<<extend>>
Display login error
Only Edit Invited User
Asset Management Server
Select file the user was invited to
<<include>>
User views file
<<extend>>
User edits file

7. [15 POINTS] **Sequence diagram** – Provide sequence diagrams (similar to Figure 5.6 and Figure 5.7) for each use case of your project. Please note that there should be an individual sequence diagram for each use case of your project. (Ch 5 and Ch 7)

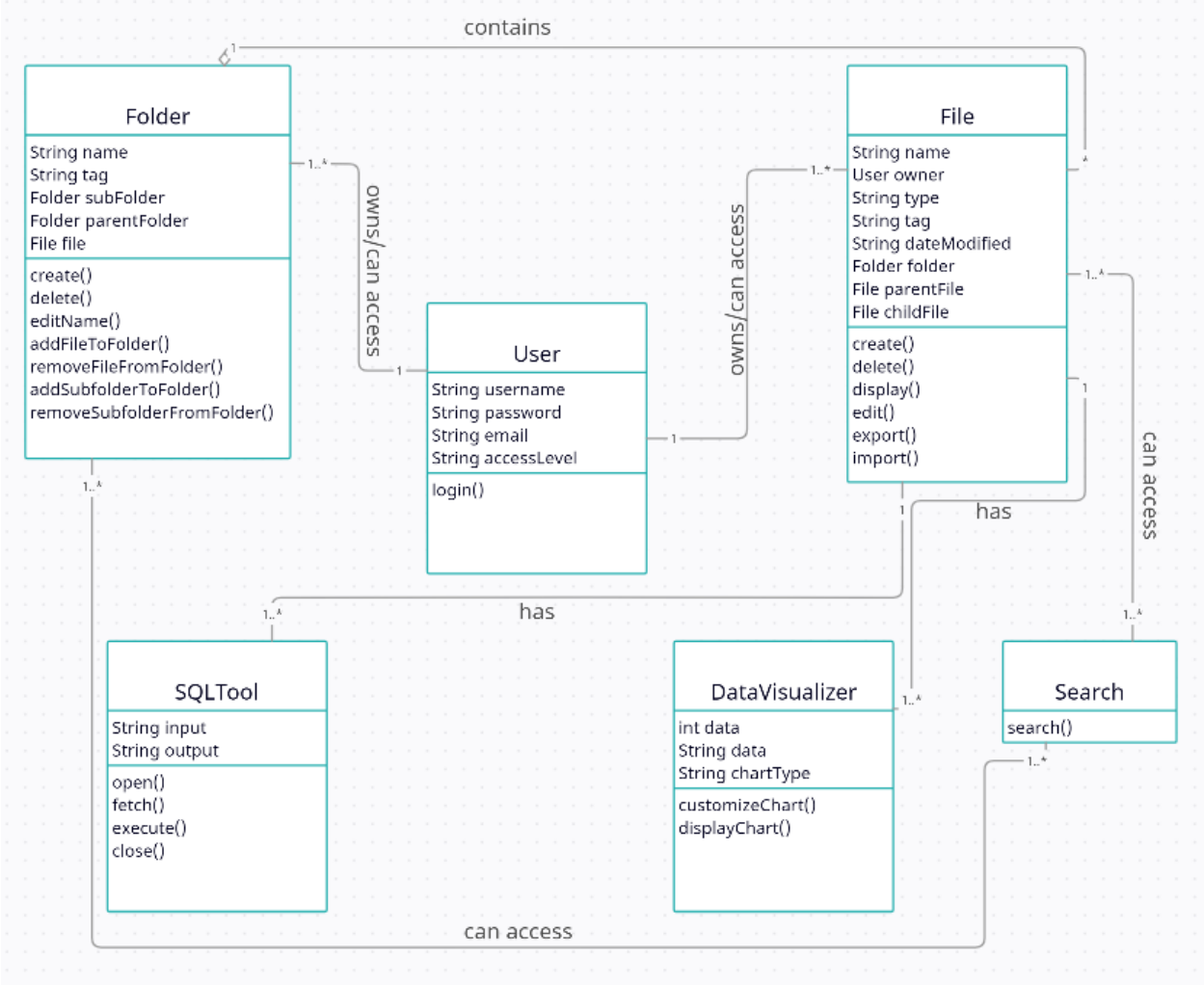
Three sequence diagrams have been provided below. The first includes the actions for importing and linking files. The second shows the actions for inviting collaborators on files and how the search function would work. The third shows how to export files, use SQL commands, and use the data visualization tool. Each "project" would be a different asset in the system, each of which has its own designated keyword or tag.




Asset Management Server
Internal User With Full Access
Login()
Alternative
[Authentication Fails]
LoginFailed()
[Else]
LoginVerified()
SearchForFileByKeyword()
Alternative
[InvalidKeyword]
DisplayMessageNoAssociatedProjects()
[Else]
DisplayAssociatedProjectFolder()
SelectFlie()
OpenAndDisplaySelectedFile()
ClickShareOperationButton()
DisplayFileShareScreen()
SelectShareTypeEditOrView()
RequestInviteeInfo()
SelectInviteeTypeExternalOrInternal()
Alternative
[InternalInvitee]
RequestInviteeUsername()
SendInviteeUsername()
Alternative
[InvalidUsername]
SendMessageInvalidUsername()
[ValidUsername]
SendMessageSuccessfulInvitation()
[ExternalInvitee]
RequestInviteeEmailAddress()
SendInviteeEmailAddress()
SendMessageSuccessfulInvitation()


Asset Management Server
Regular User
Login()
Alternative
[Authentication Fails]
LoginFailed()
[Else]
LoginVerified()
SearchForFileByKeyword()
Alternative
[InvalidKeyword]
DisplayMessageNoAssociatedProjects()
[Else]
DisplayAssociatedProjectFolder()
SelectFlie()
OpenAndDisplaySelectedFile()
OpenExportFileScreen()
DisplayExportOptions()
SelectExportFileType()
ExportAndSendMessageOfCompletion()
OpenSQLCommandBox()
DisplaySQLCommandBox()
TypeSQLCommand()
ExecuteSQLCommand()
OpenDataVisualizationTool()
DisplayVisualizationCustomizationScreen()
InputDesiredCustomizations()
DisplayVisualization()

8. [15 POINTS] **Class diagram** – Provide a class diagram (similar to Figure 5.9) of your project. The class diagram should be unique (only one) and should include all classes of your project. Please make sure to include cardinalities, and relationship types (such as generalization and aggregation) between classes in your class diagram. Also make sure that each class has class name, attributes, and methods named (Ch 5).



9. [15 POINTS] **Architectural design** – Provide an architectural design of your project. Based on the characteristics of your project, choose and apply only one appropriate architectural pattern from the following list: (Ch 6 section 6.3)
9.1. Model-View-Controller (MVC) pattern (similar to Figure 6.6)