

Information Management System

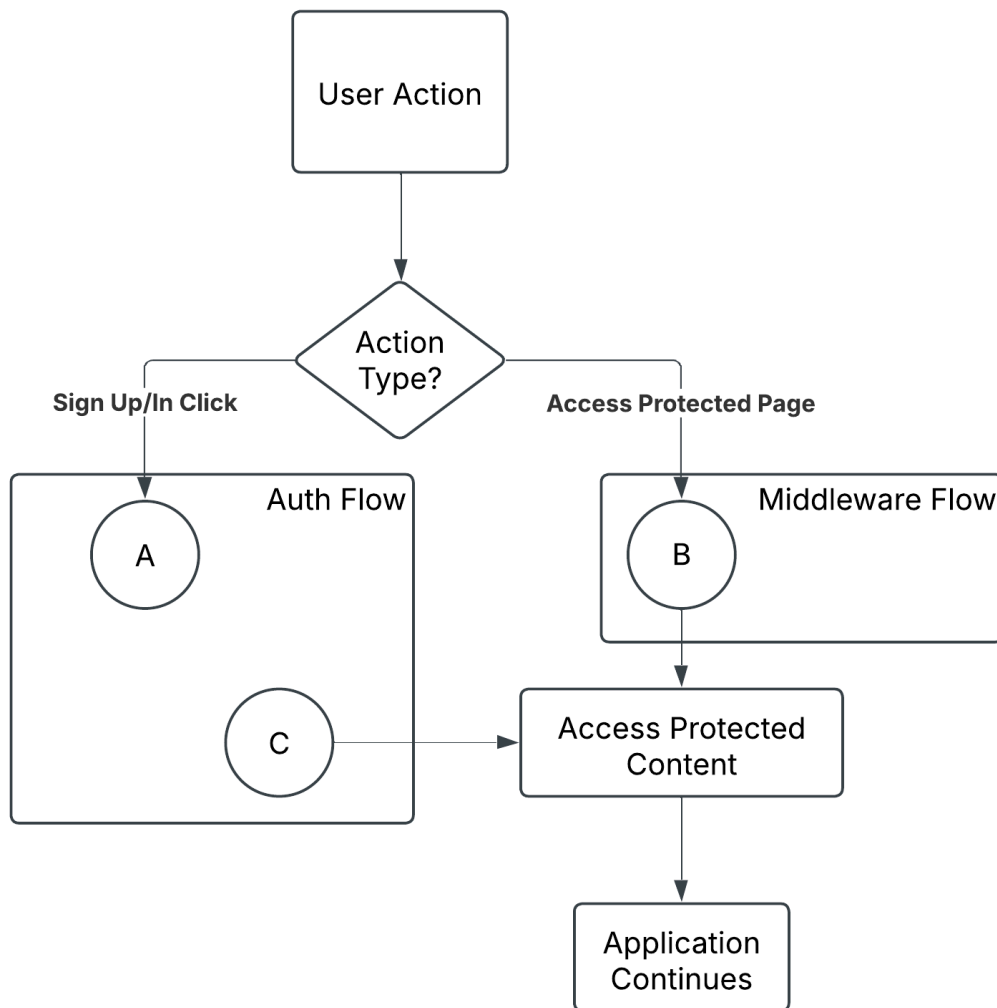
Criterion B - Design

Table of contents:

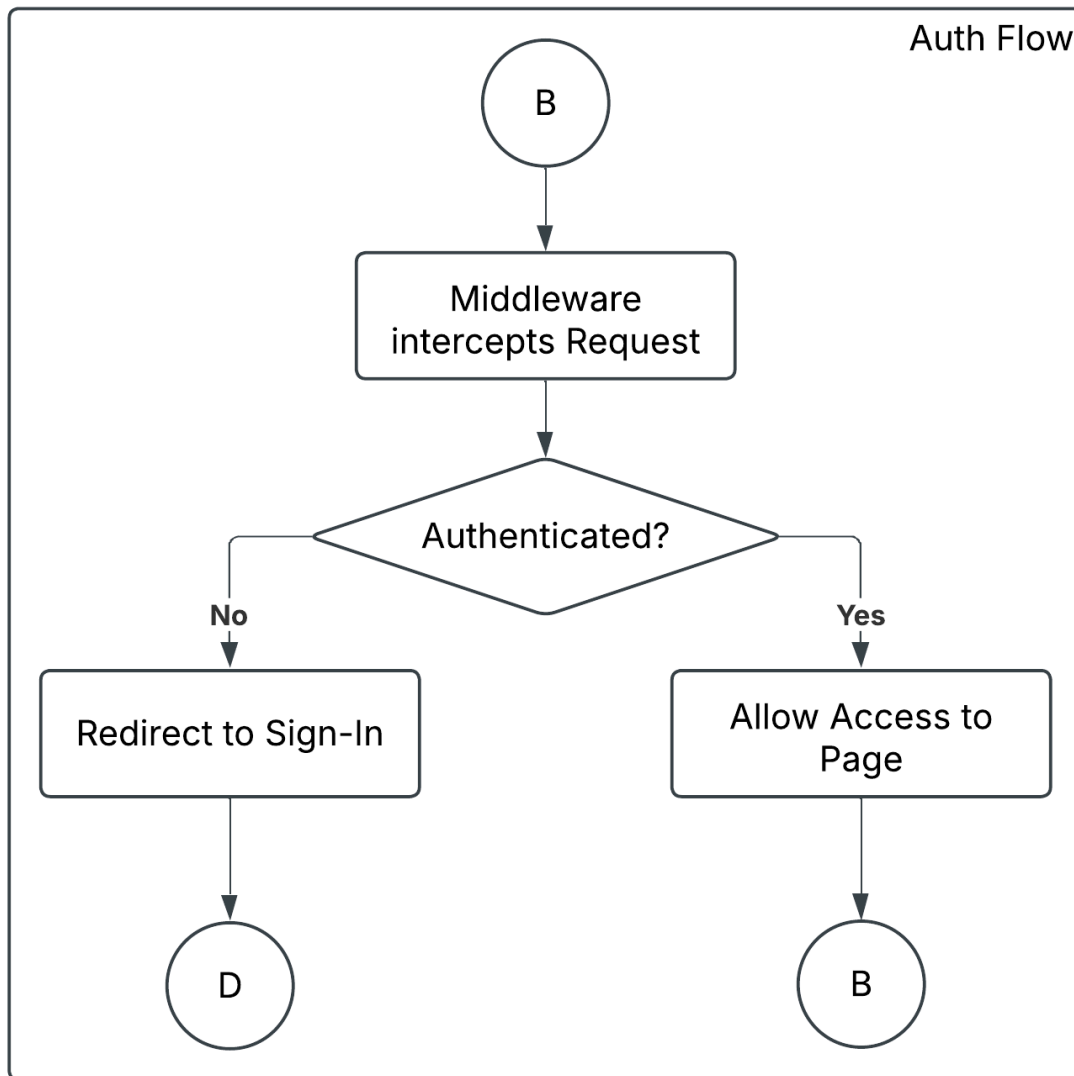
Criterion B - Design.....	1
Table of contents:.....	1
System flowcharts.....	2
Authentication.....	2
CookieJar.....	6
DoubtTracker.....	7
CuriositySpace.....	11
Notebooks.....	15
ToDoList.....	18
StageManager.....	21
Modular Abstraction Diagrams.....	24
Page designs.....	28
Login and registration:.....	28
Database.....	35
CookieJar.....	35
DoubtTracker and CuriositySpace.....	36
Notebooks.....	37
ToDoList.....	38
StageManager.....	39
Validation.....	40
Test Plan.....	46
Testing General and Database Functionality of Information Management System:.....	56

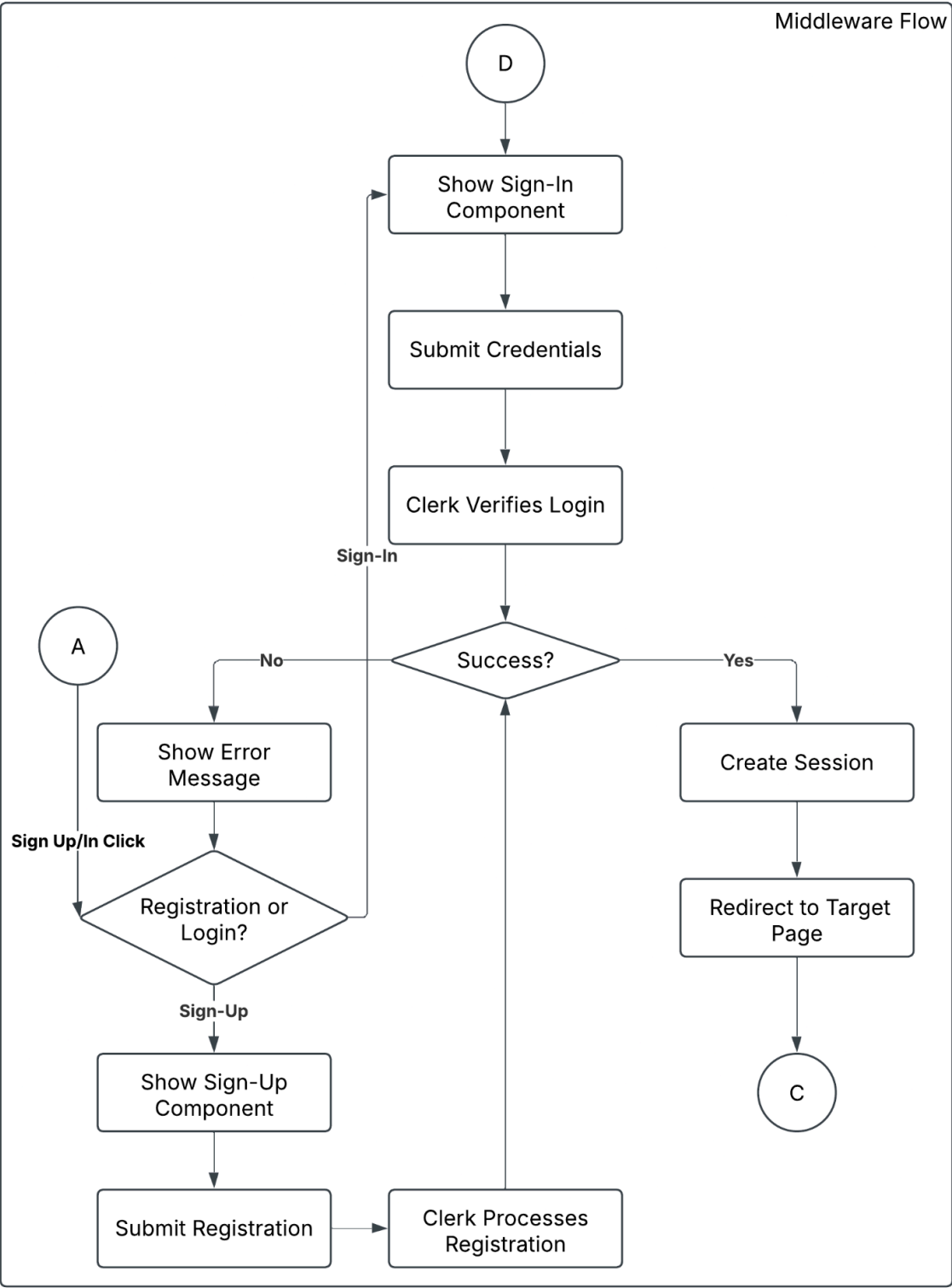
System flowcharts

Authentication

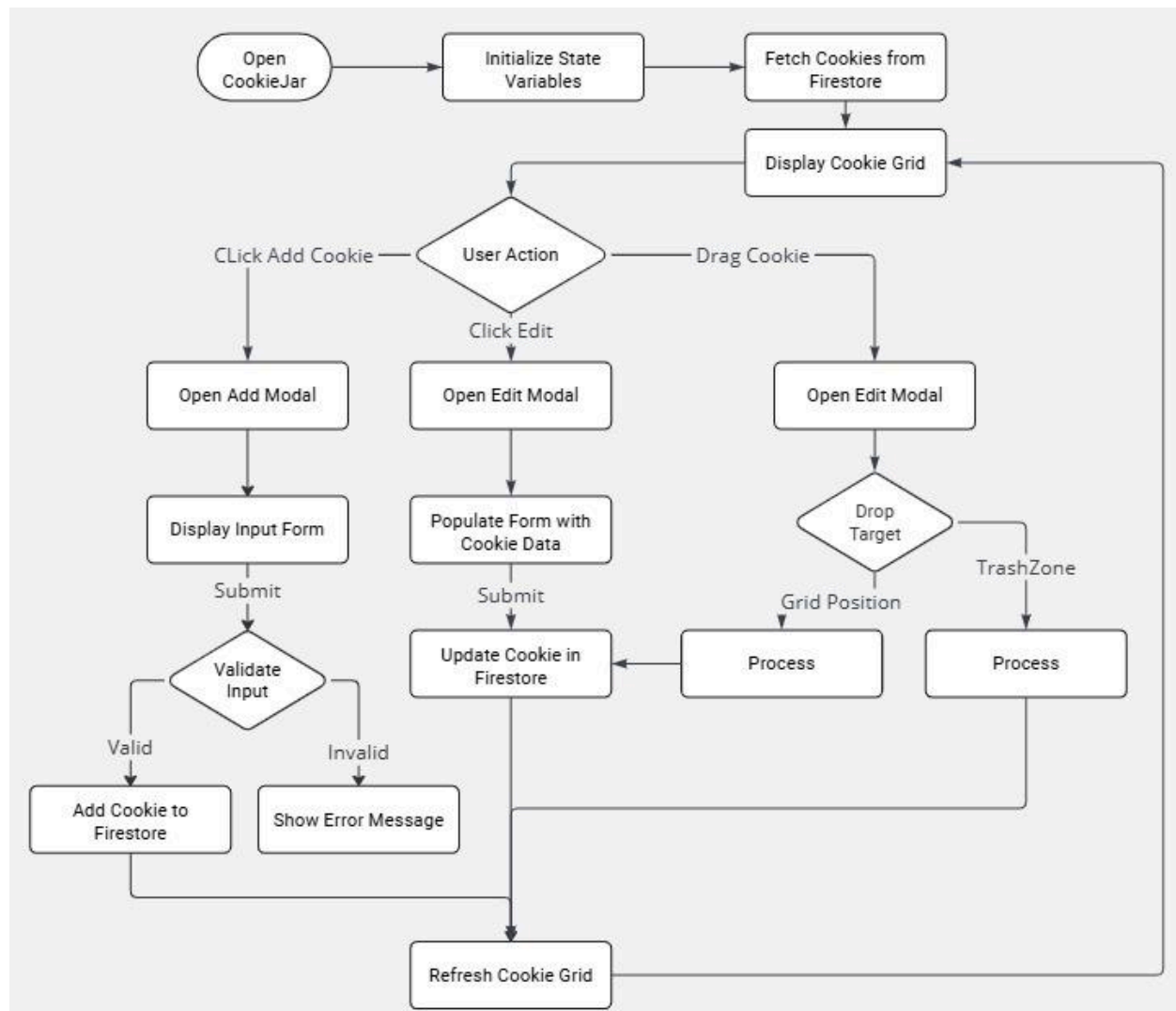


Auth Flow

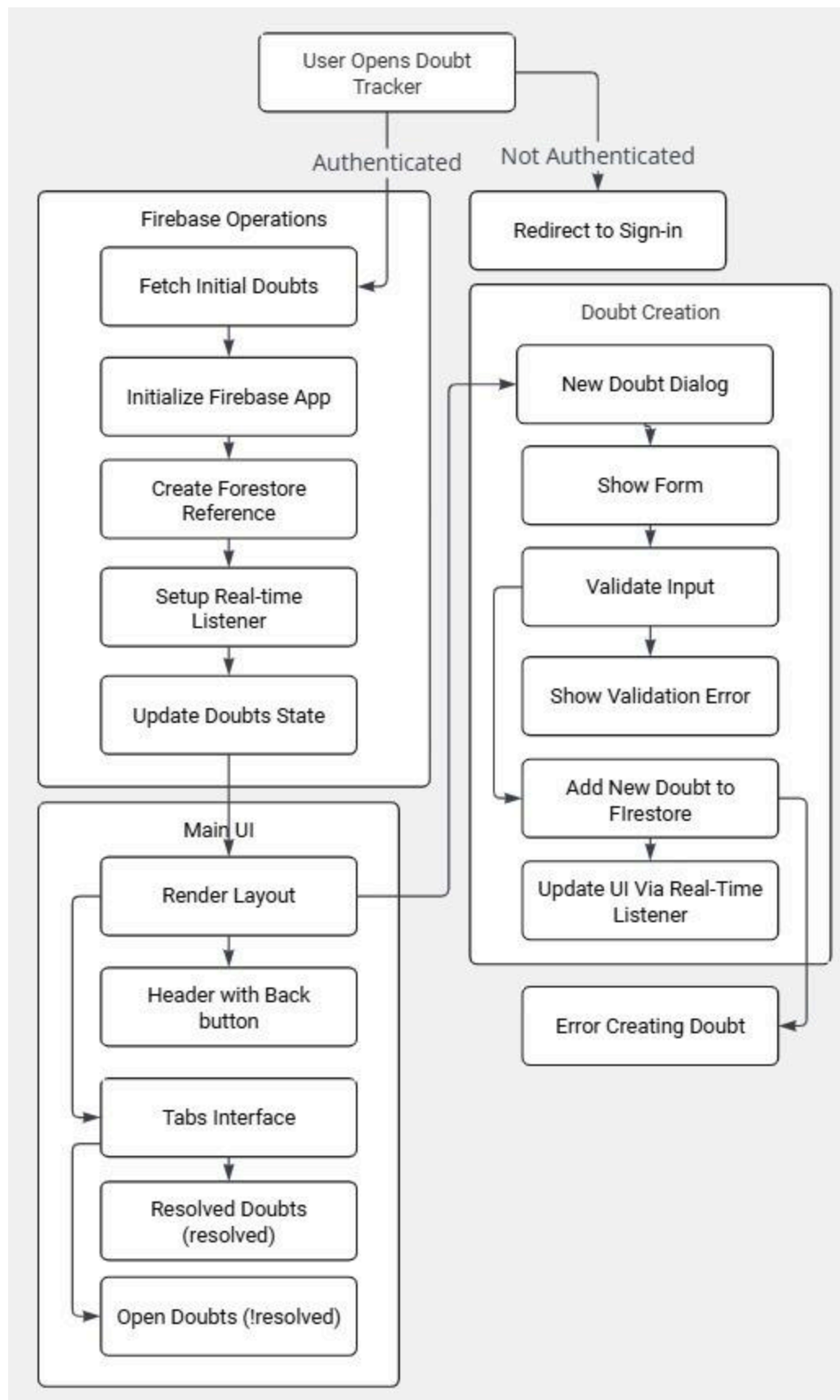


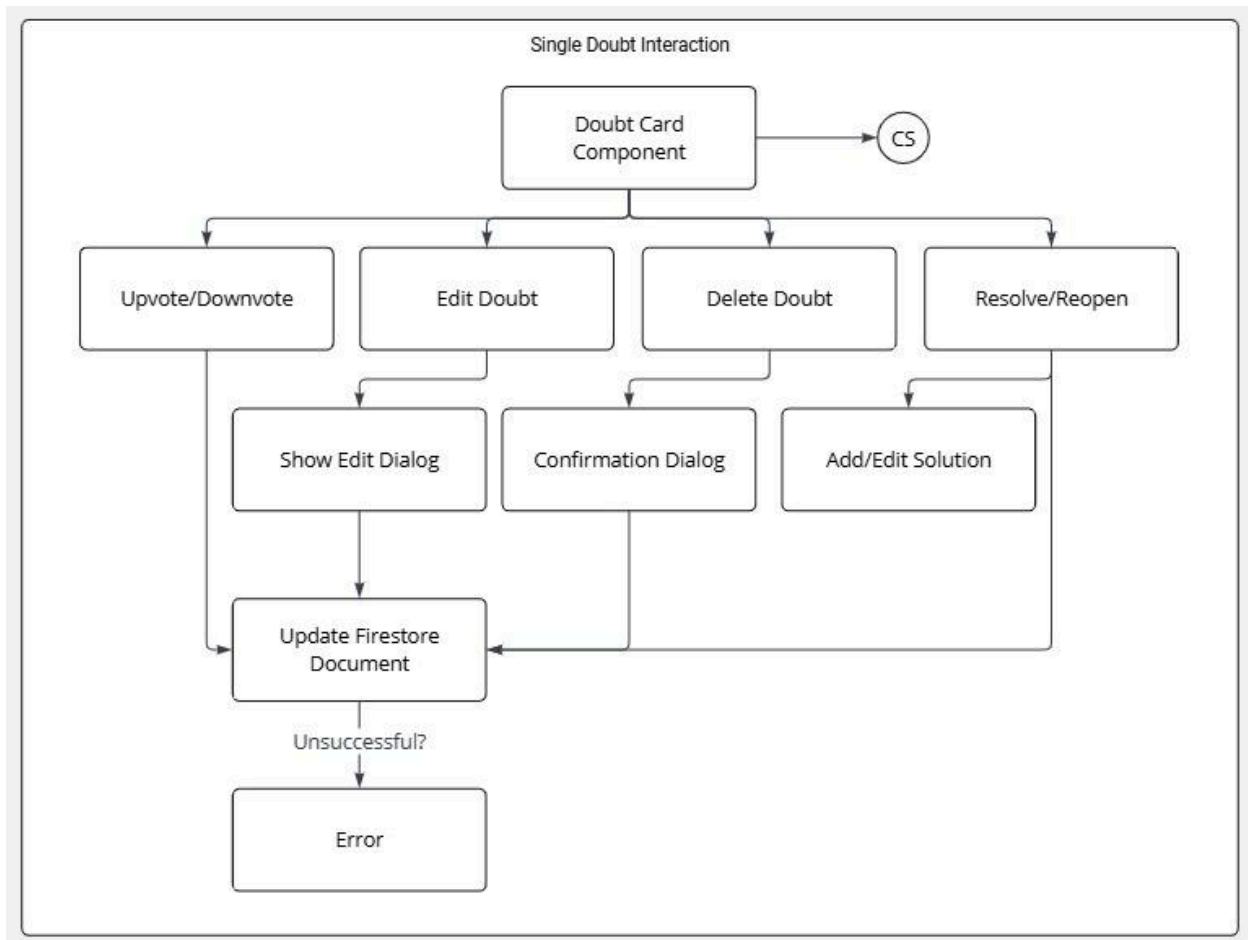


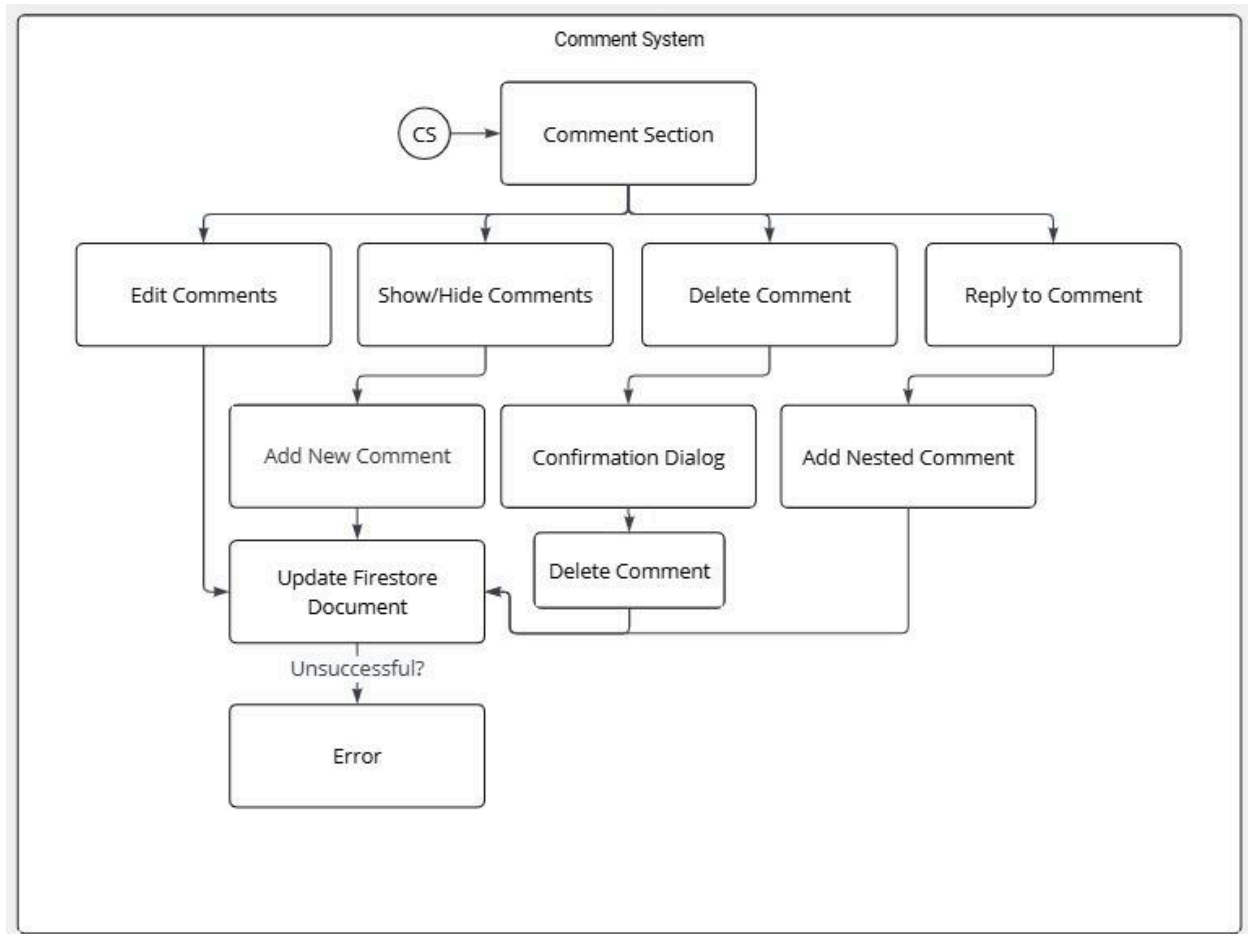
CookieJar



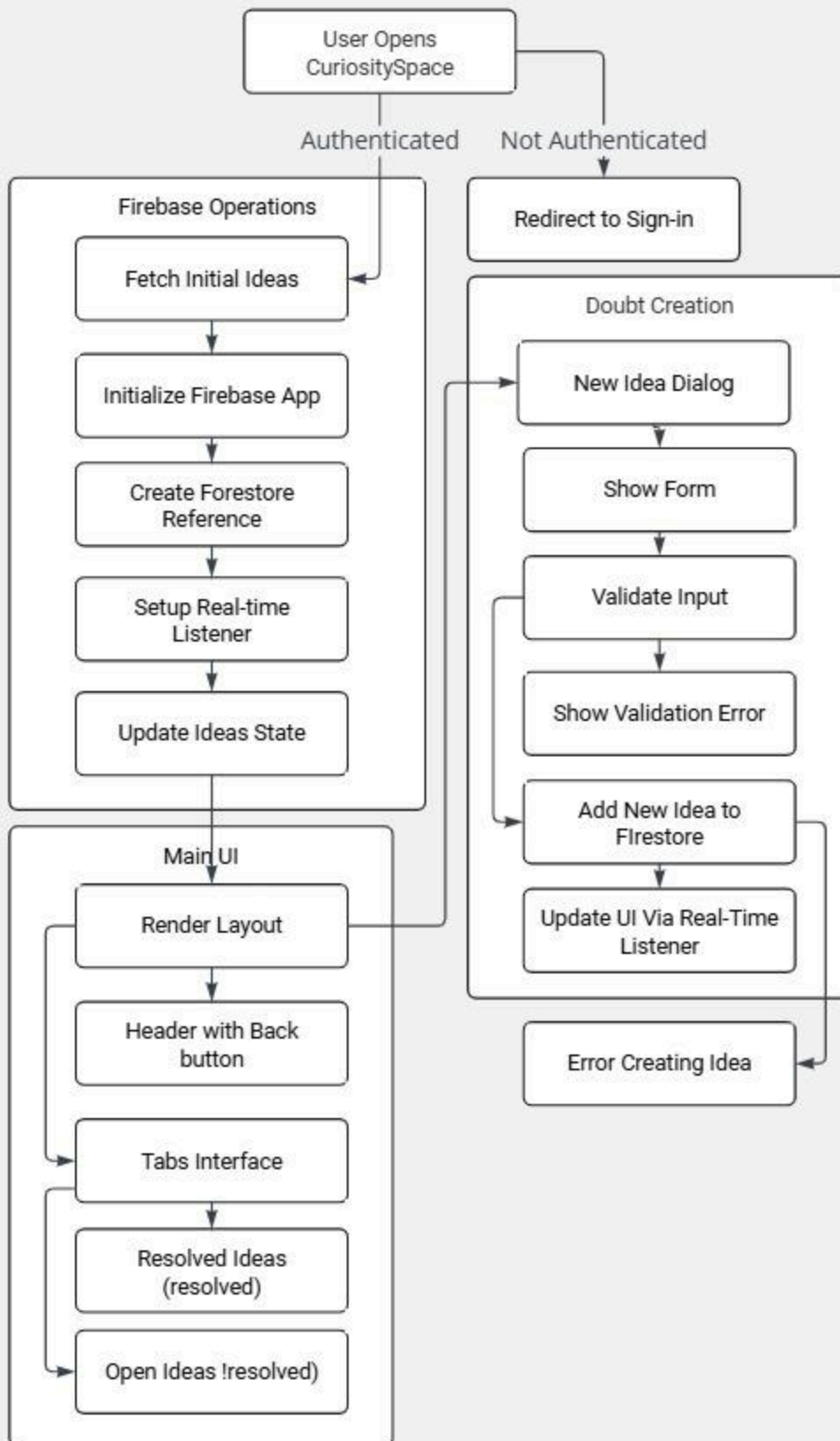
DoubtTracker

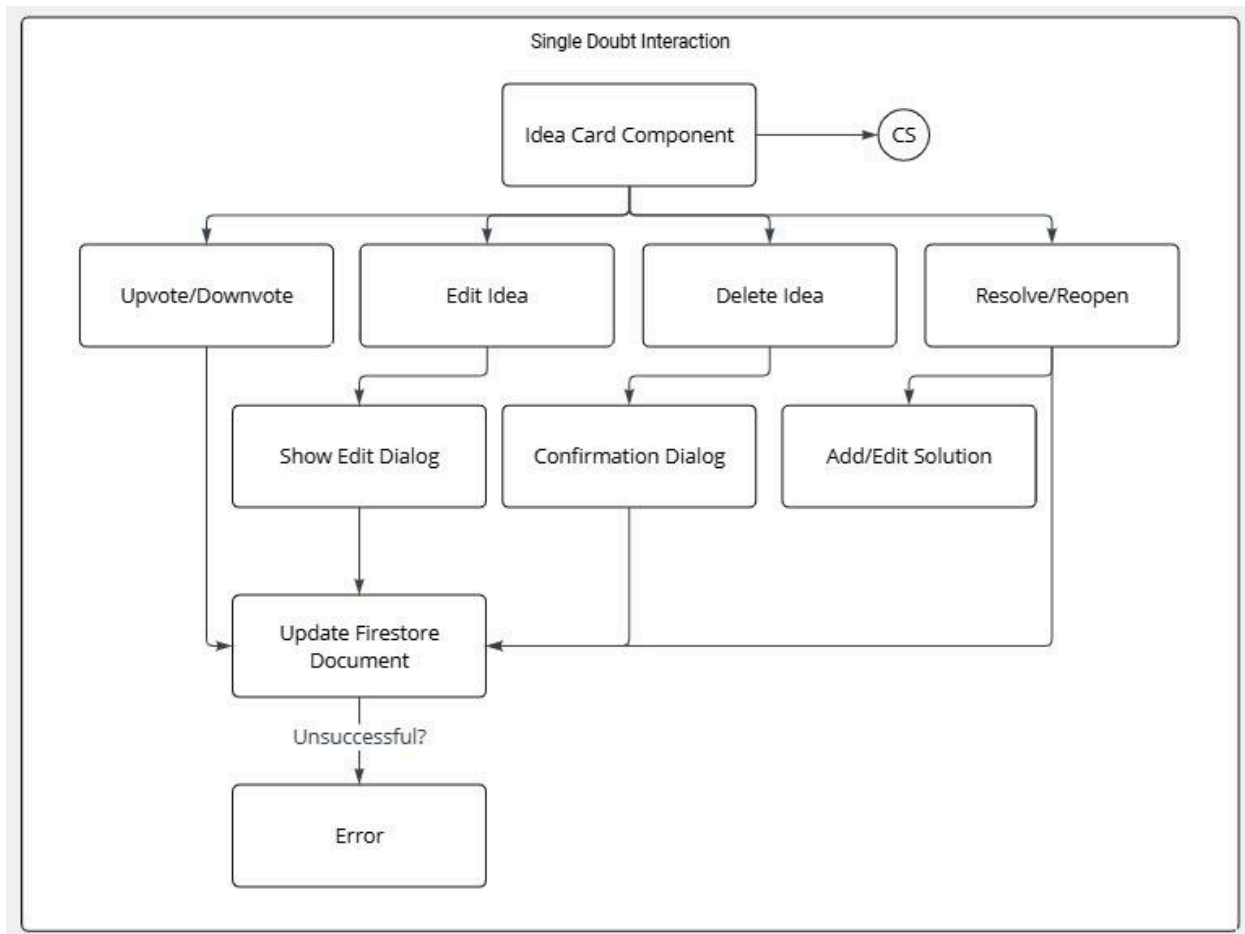


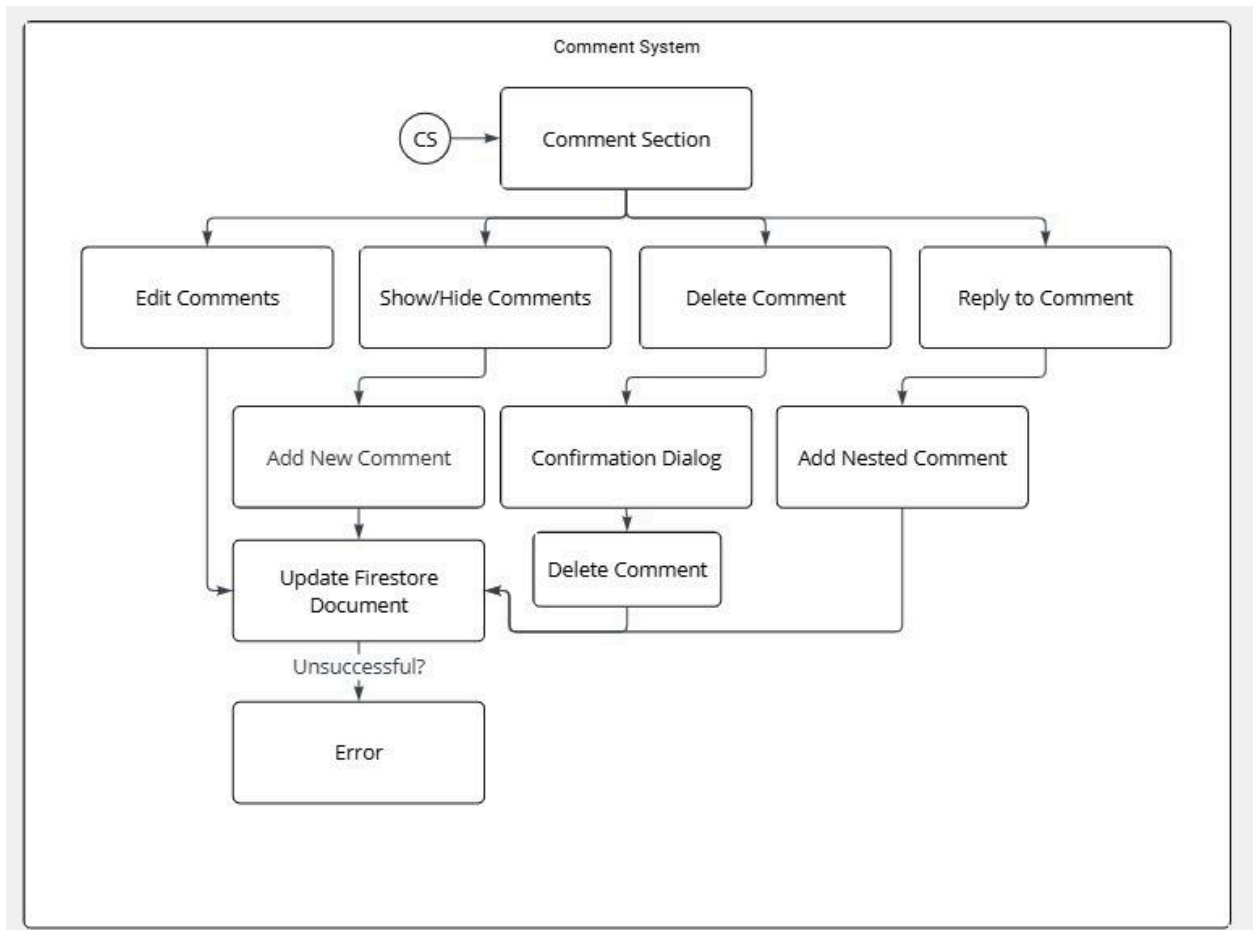




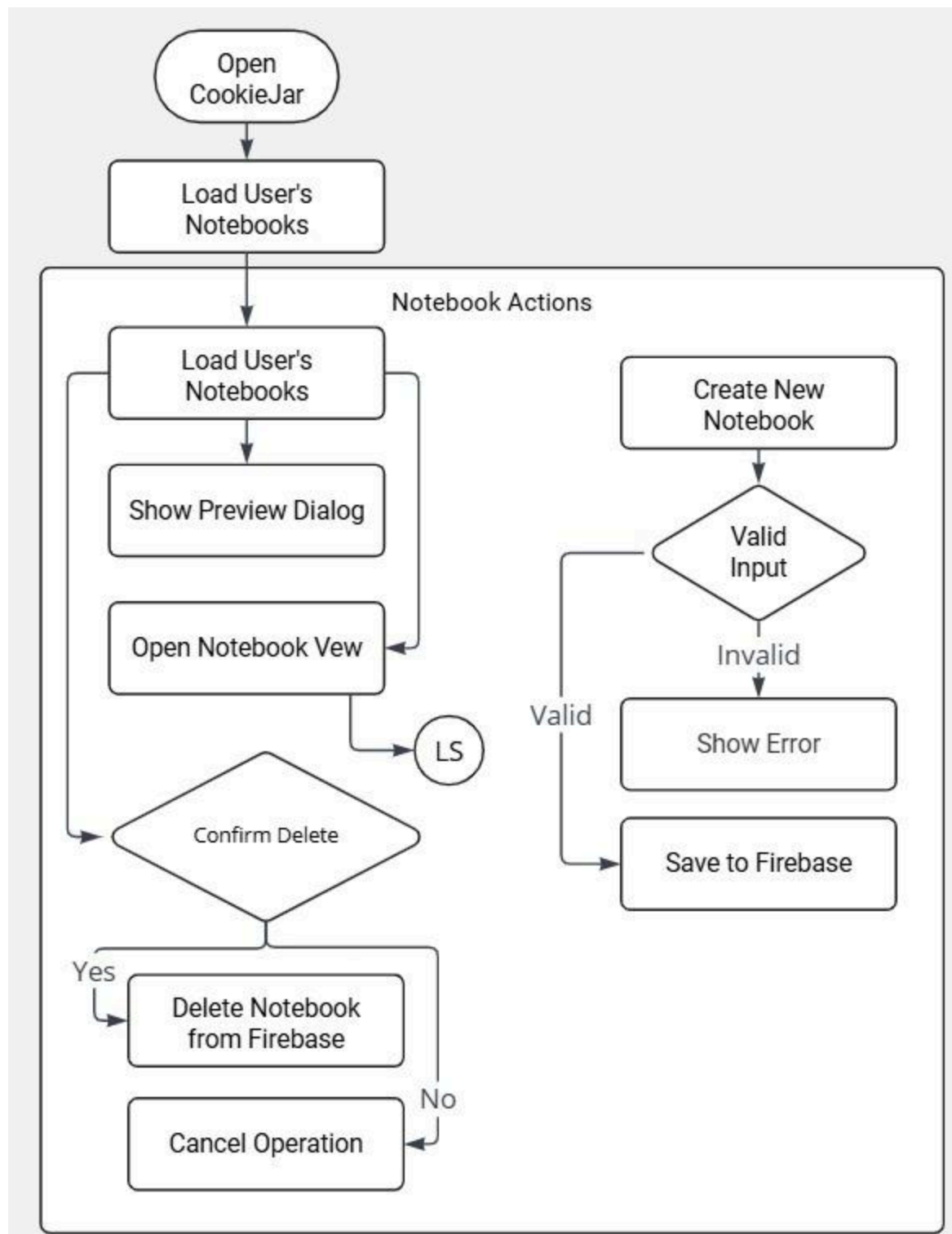
CuriositySpace

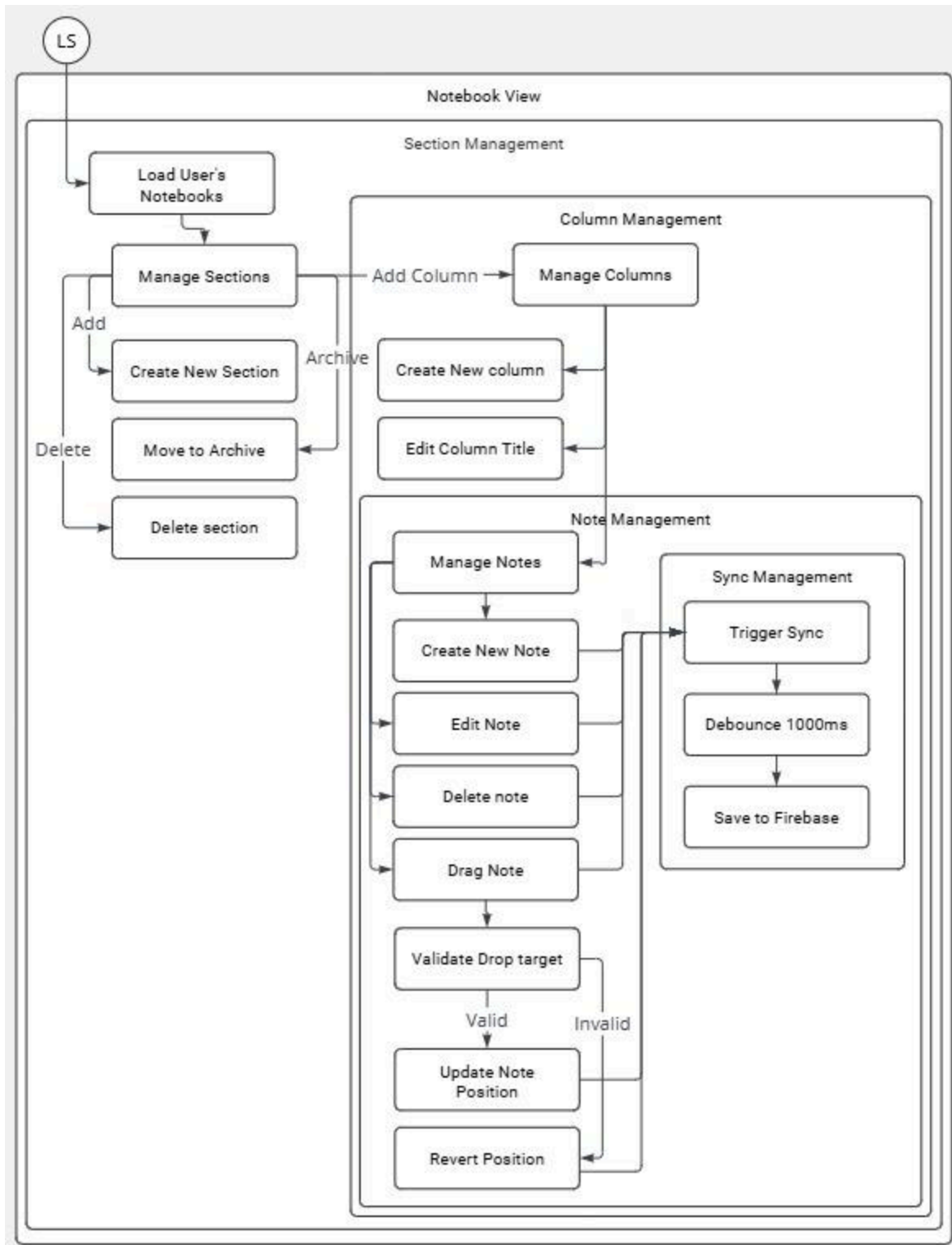




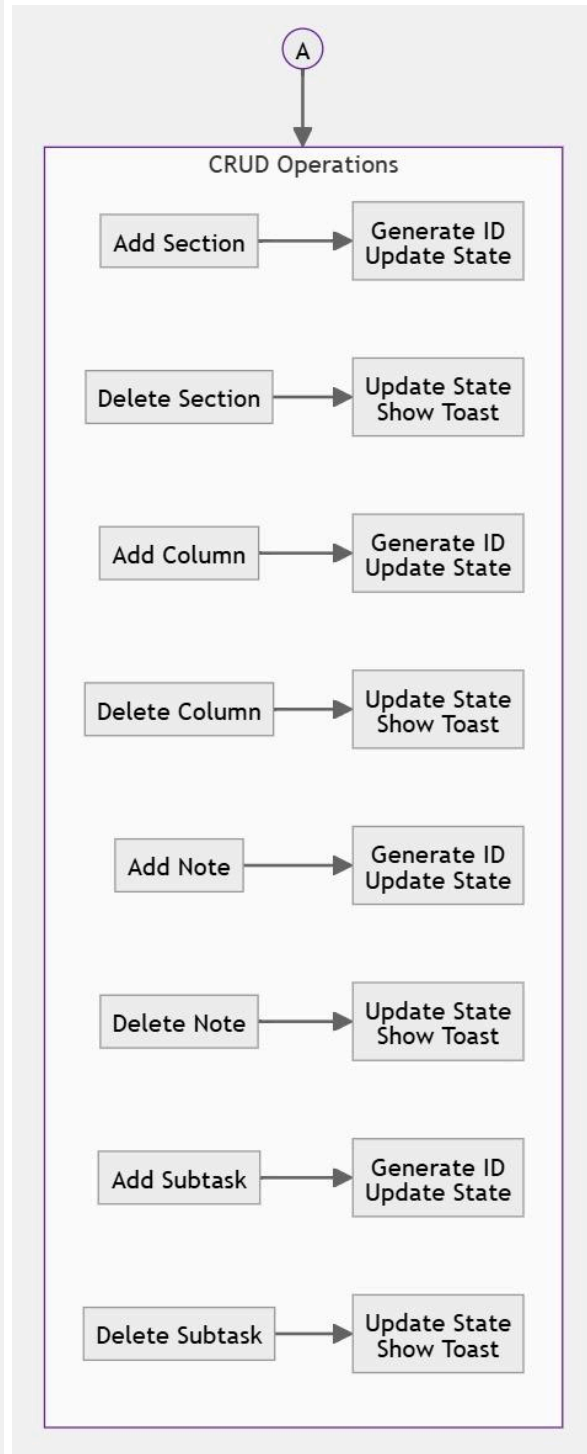
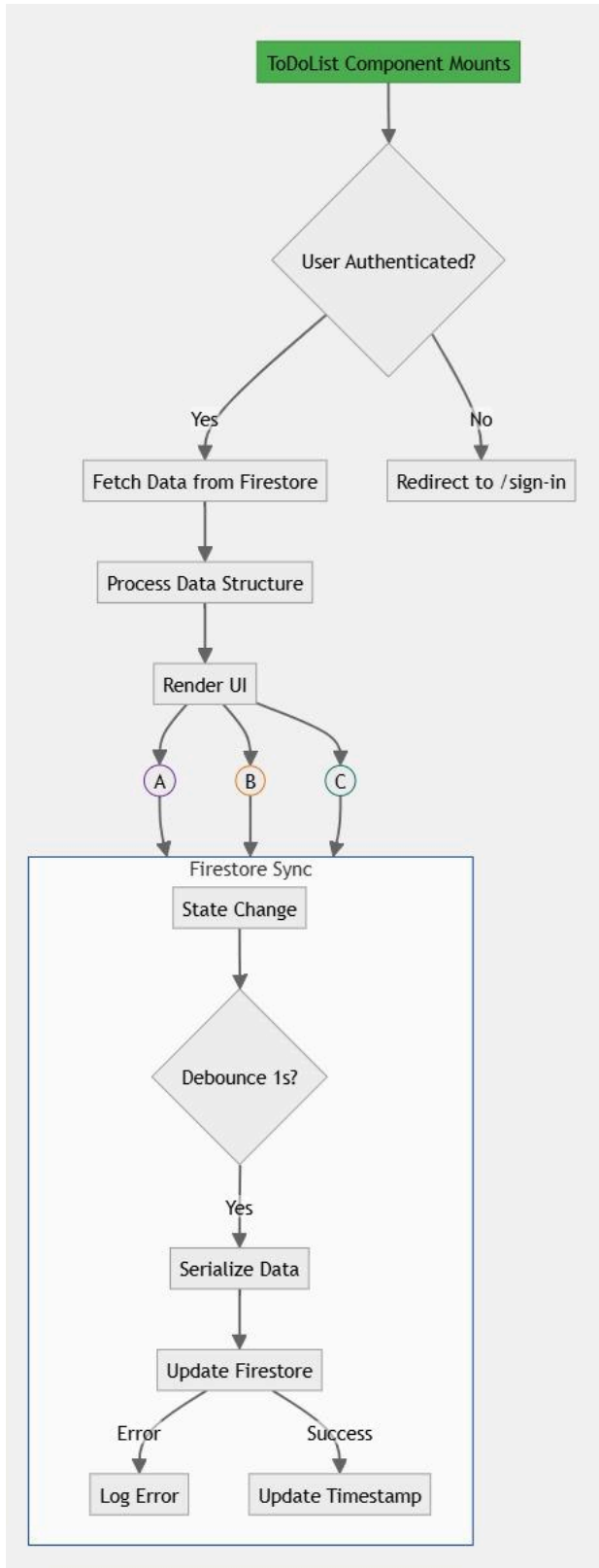


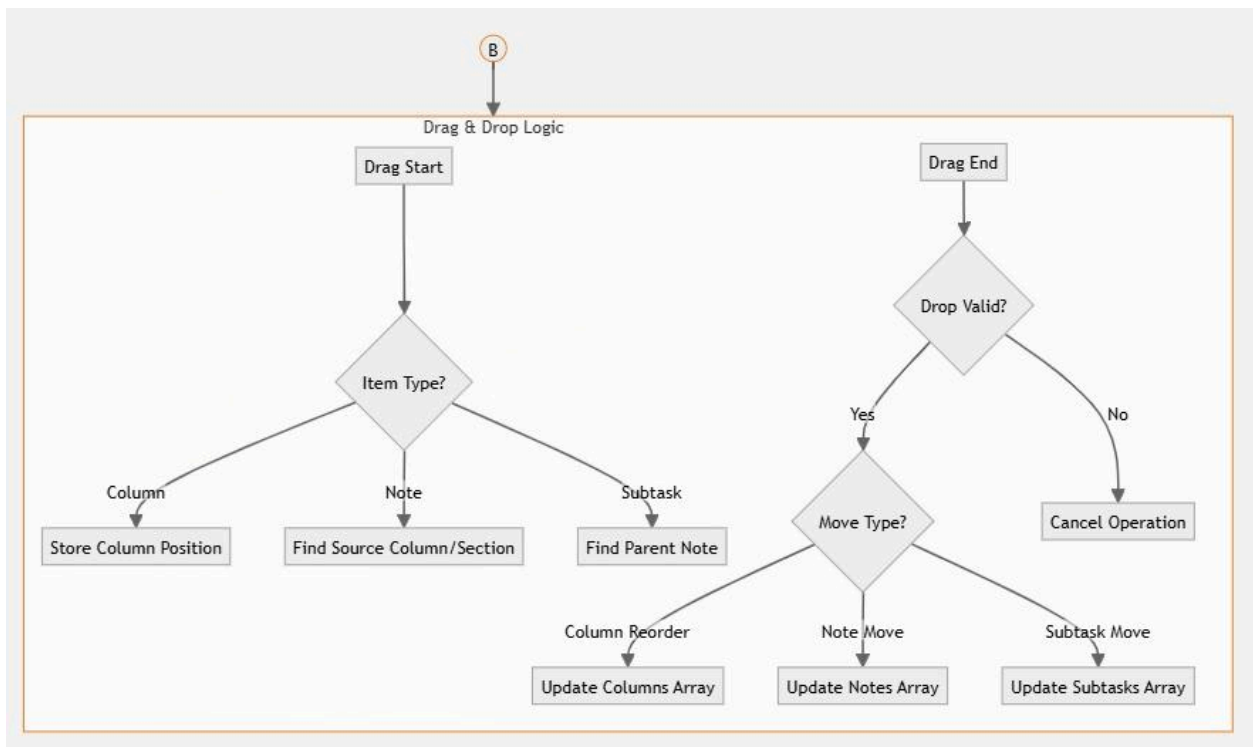
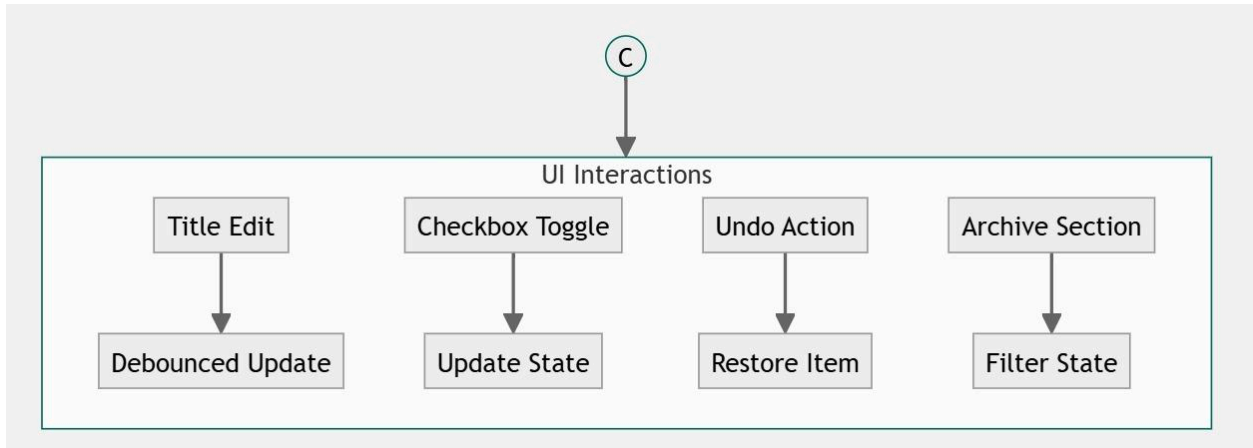
Notebooks



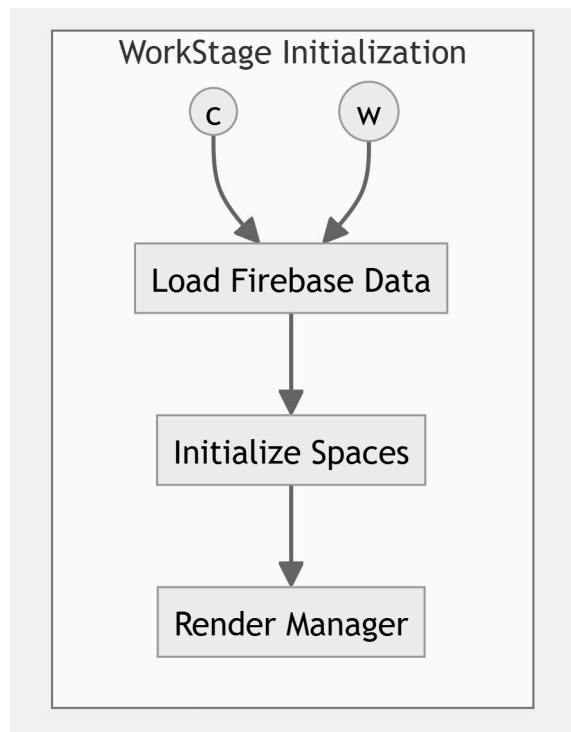
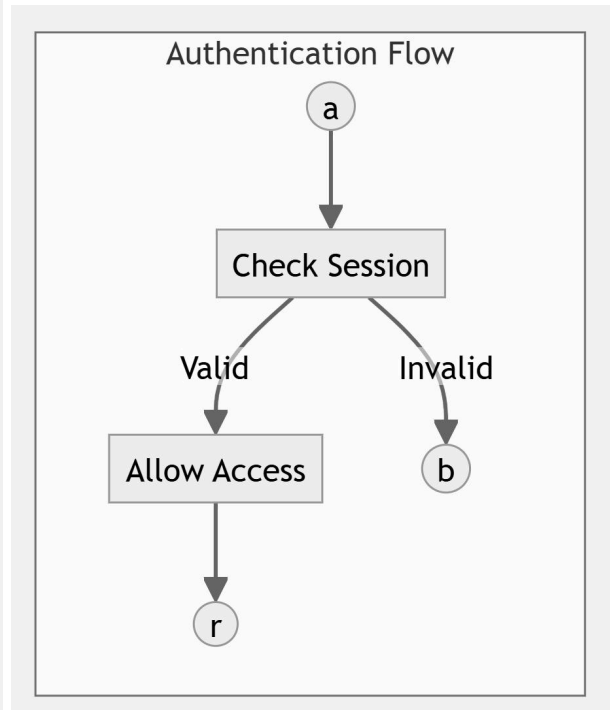
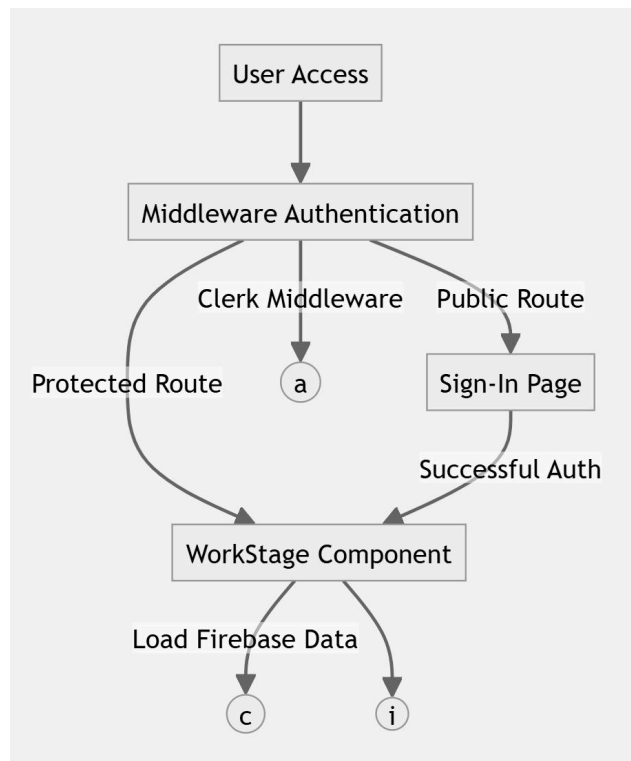


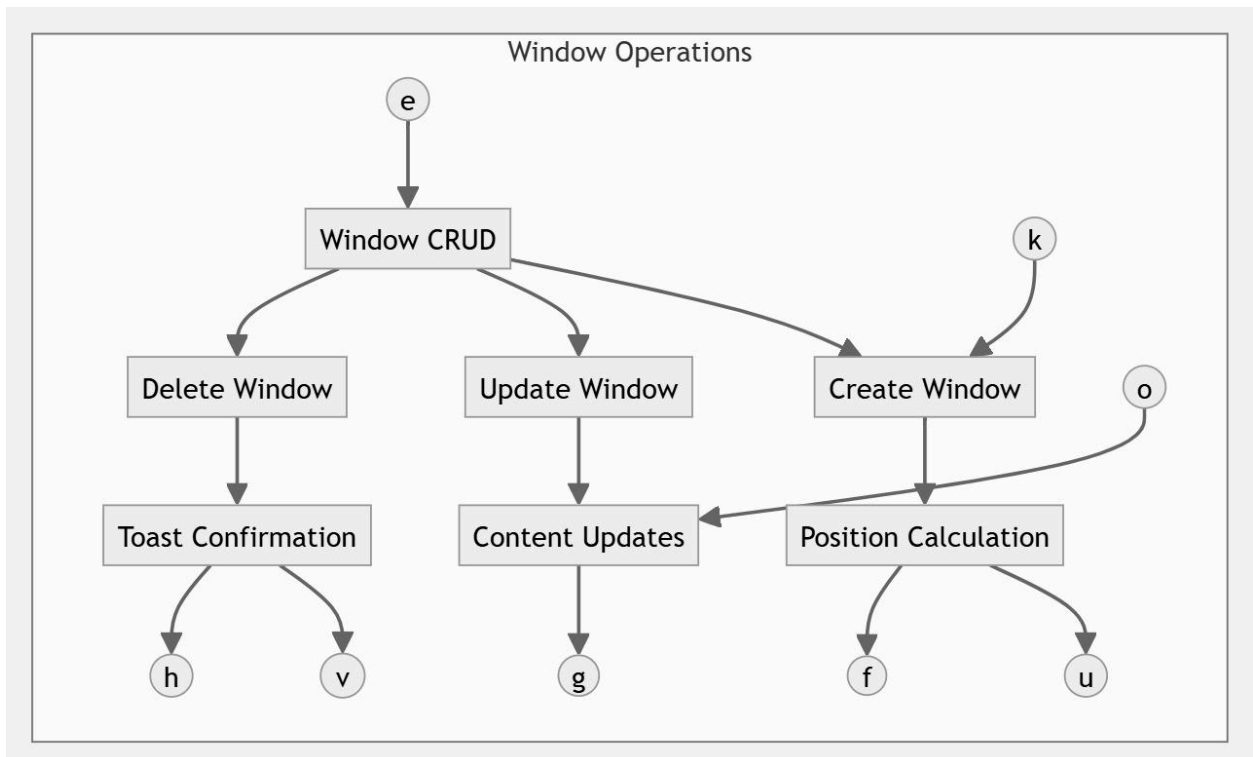
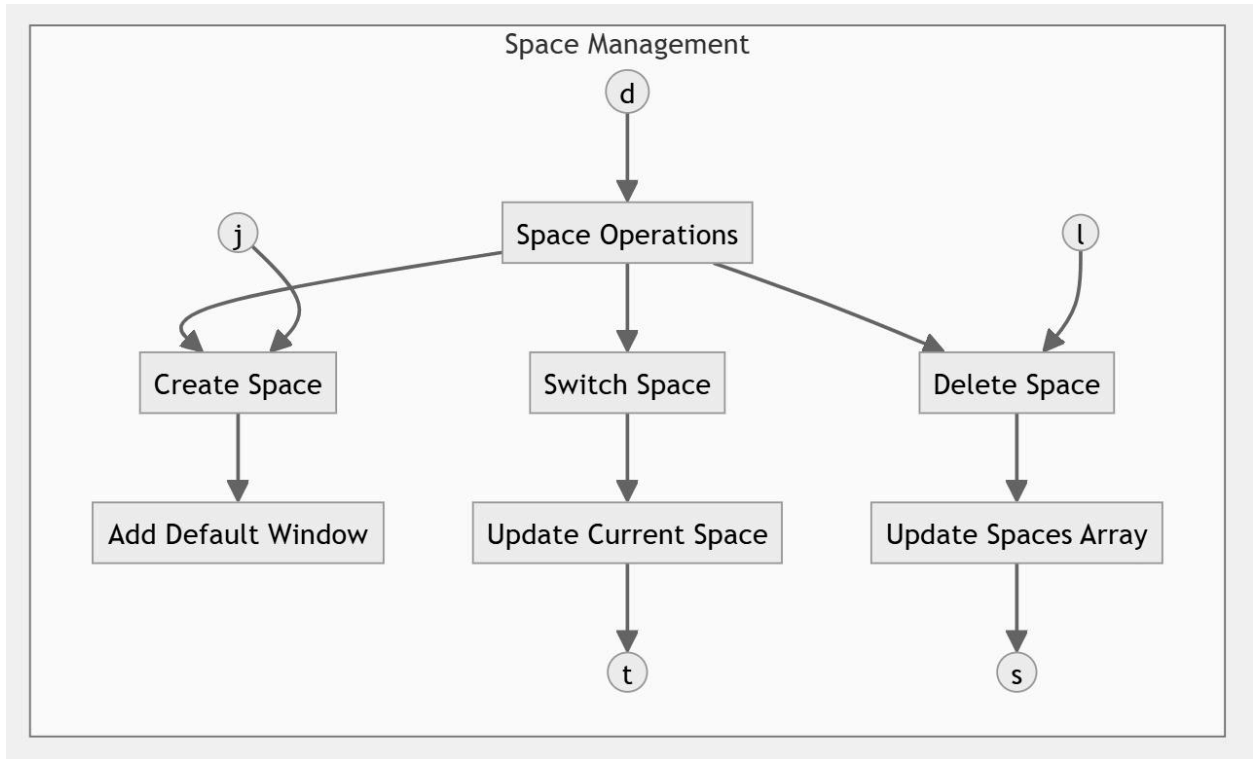
ToDoList

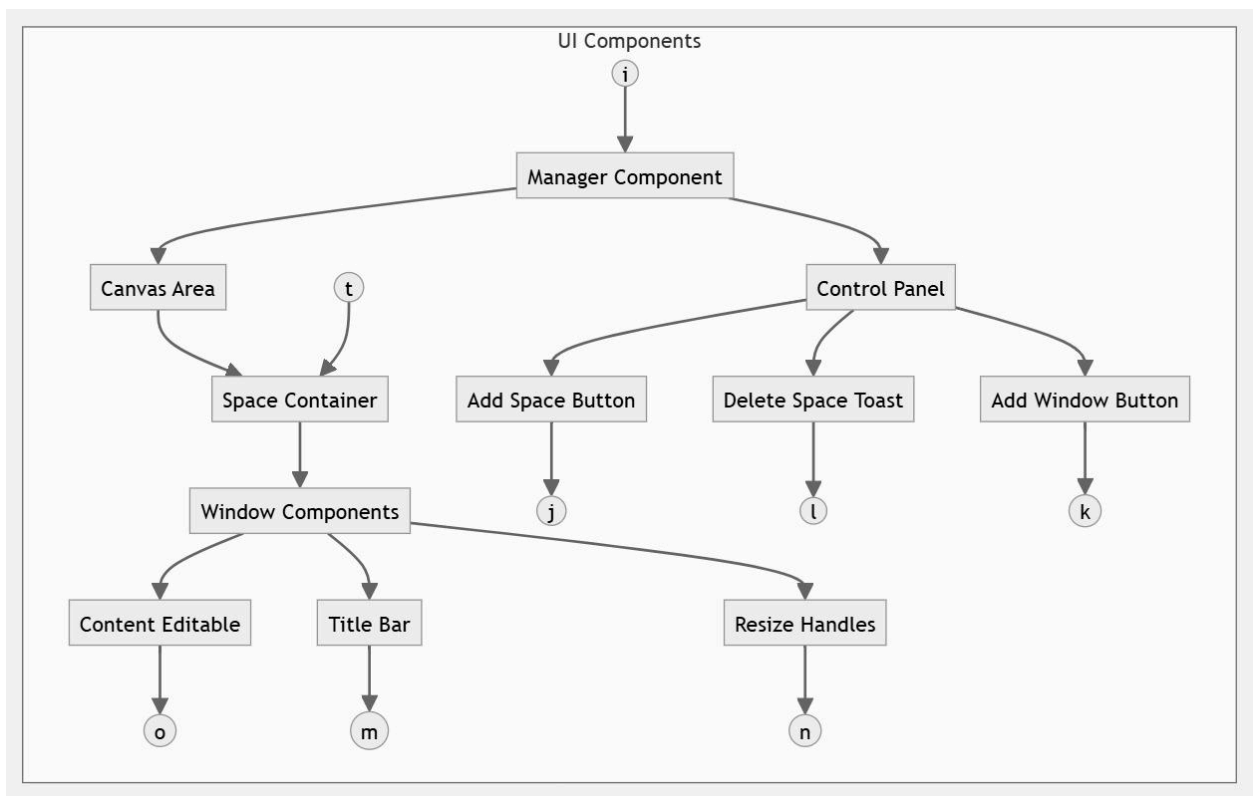
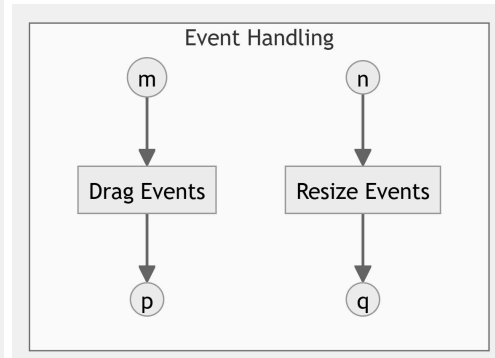
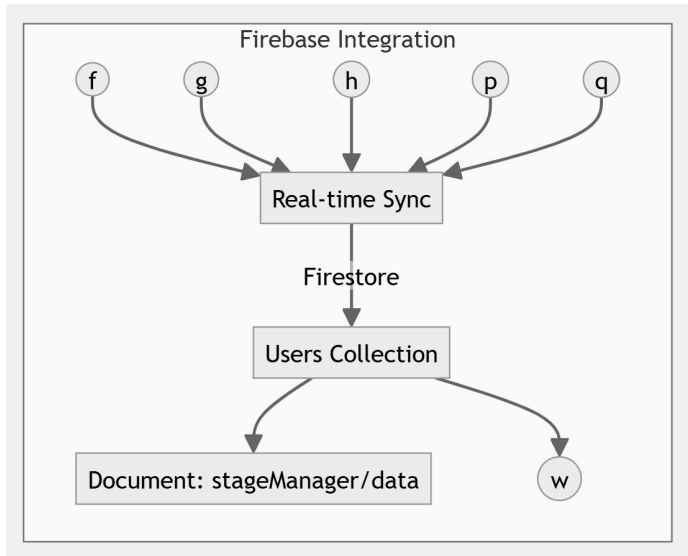




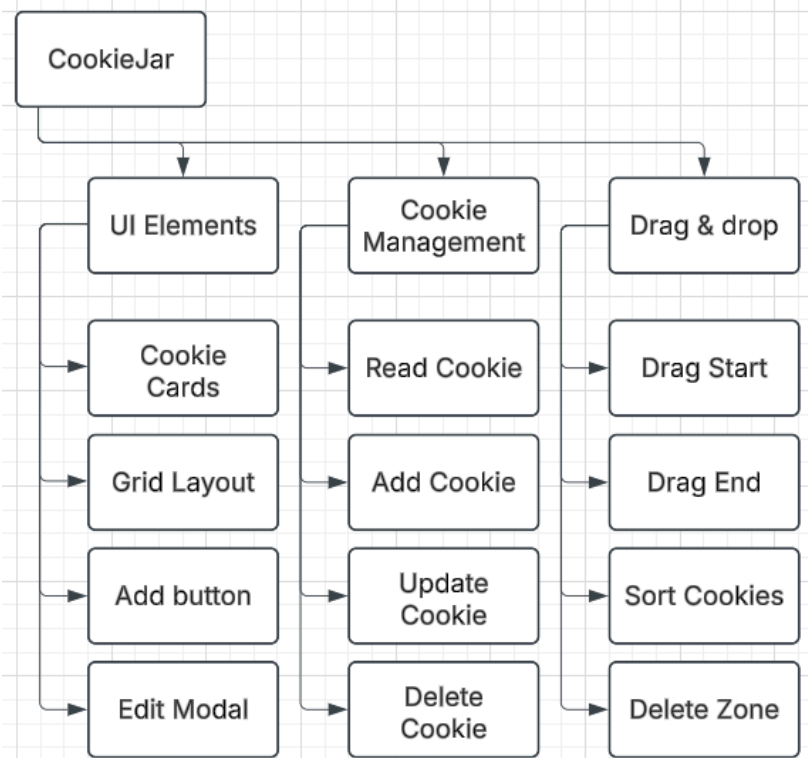
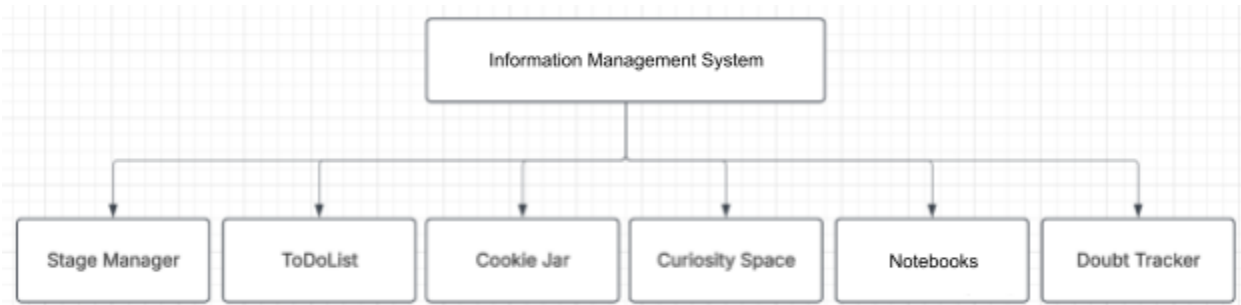
StageManager

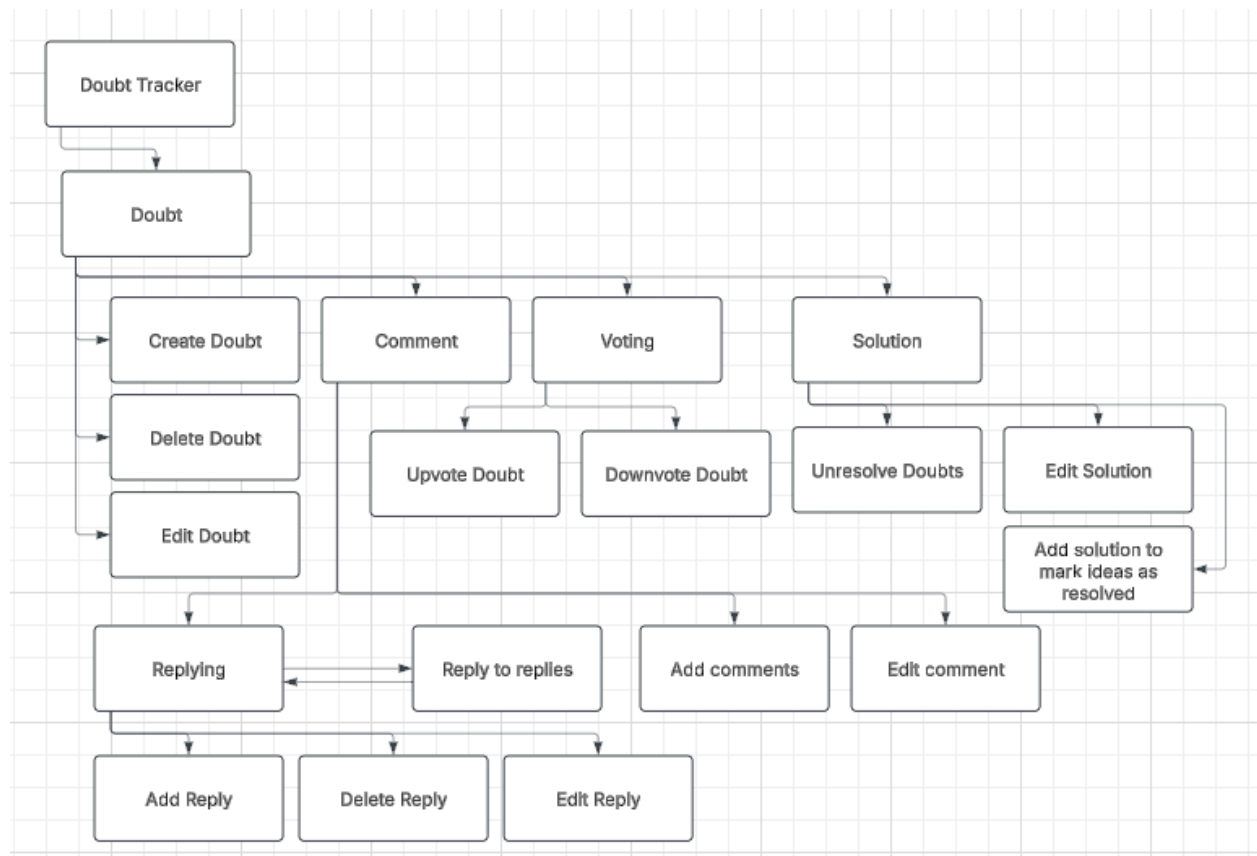


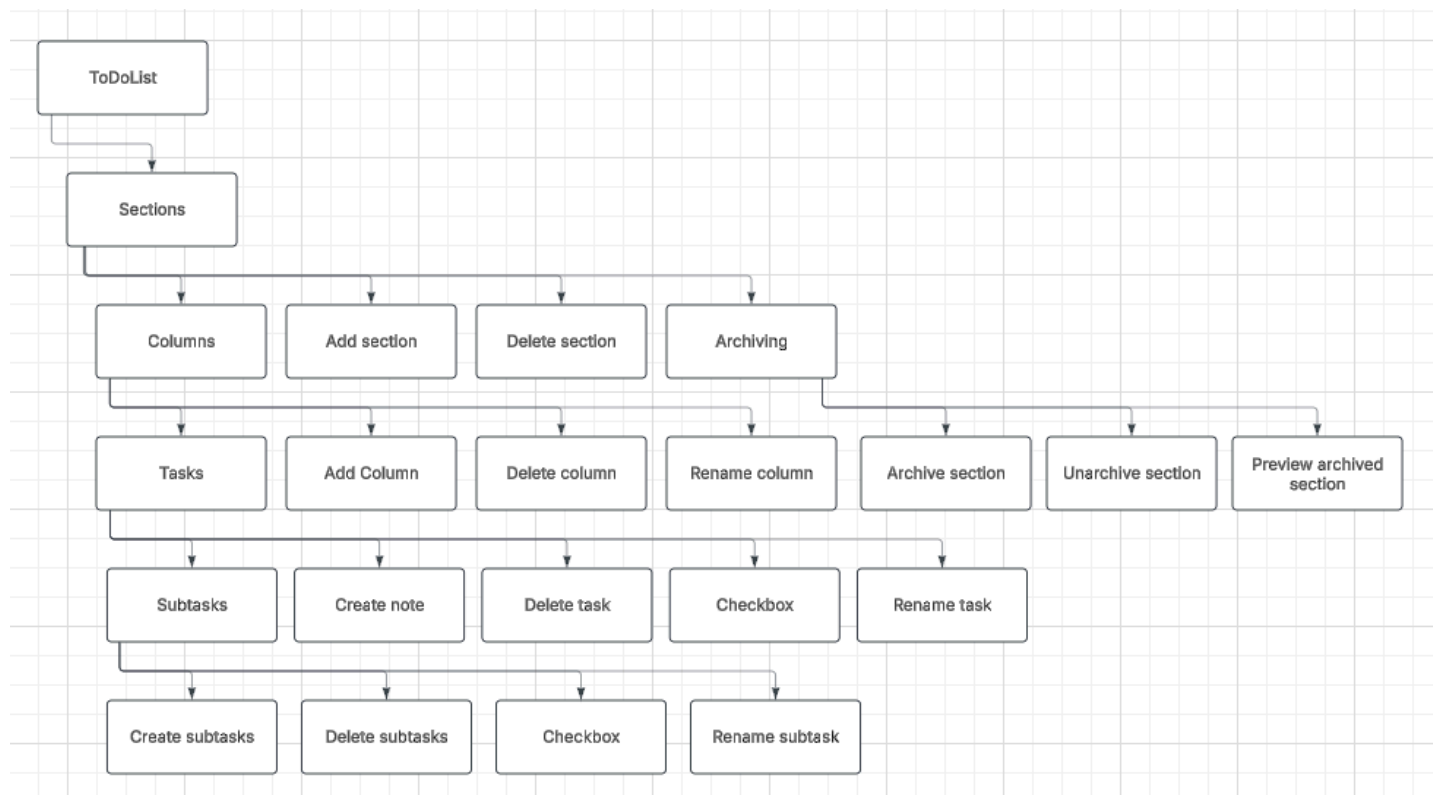
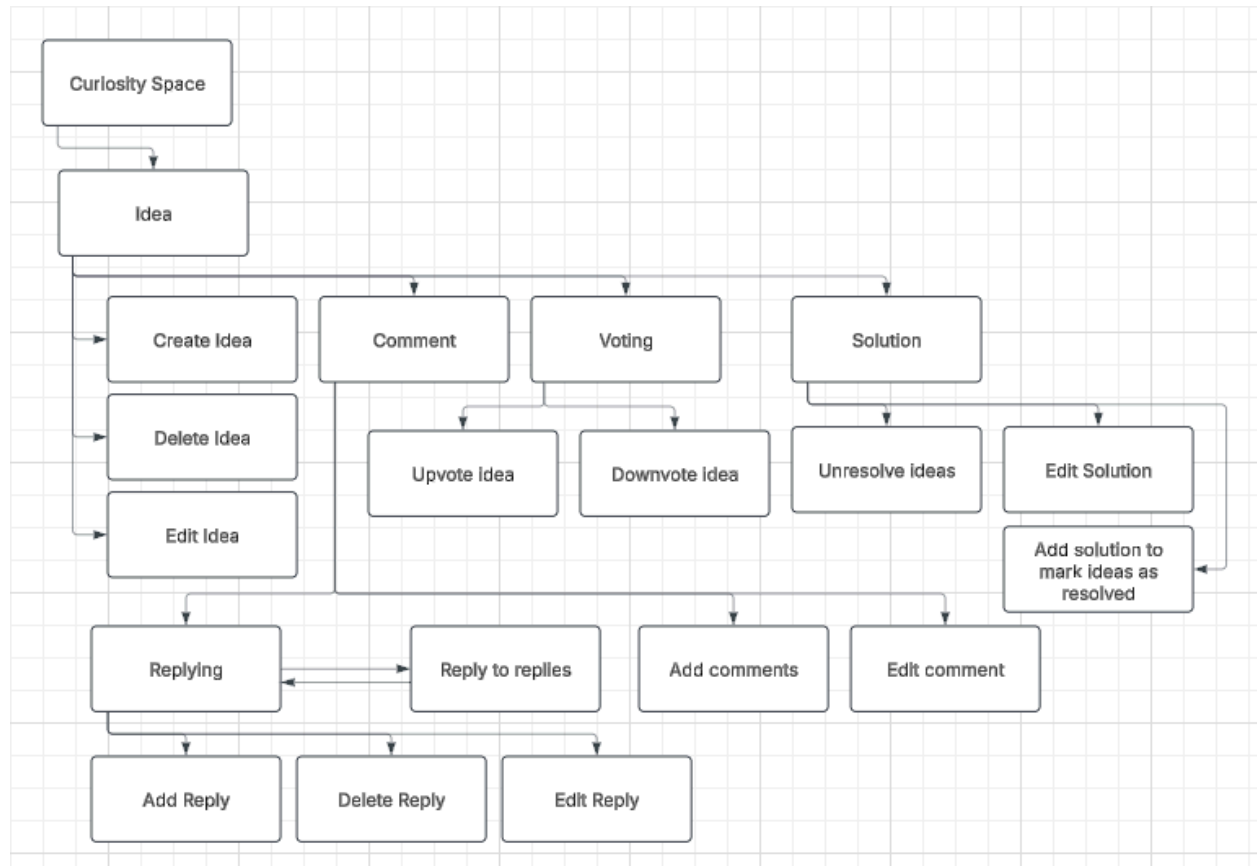


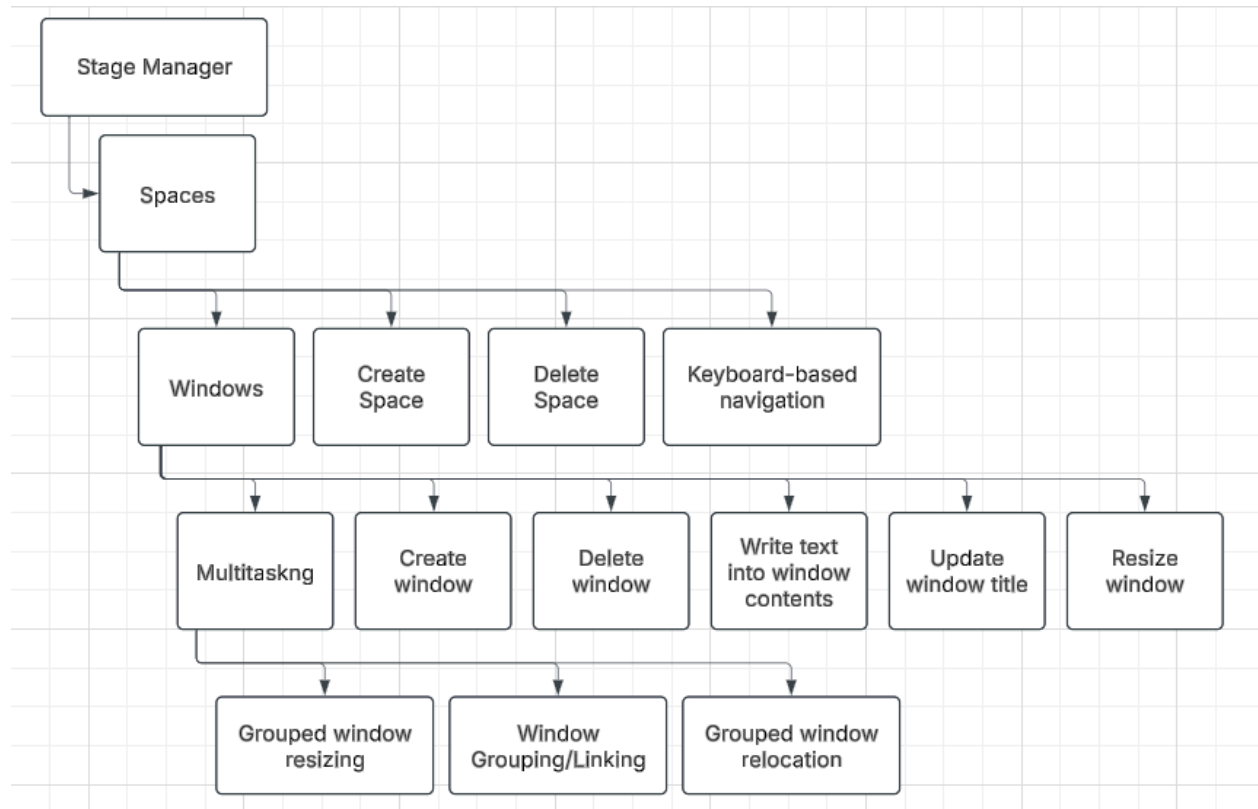


Modular Abstraction Diagrams









Page designs

Login and registration:

Login

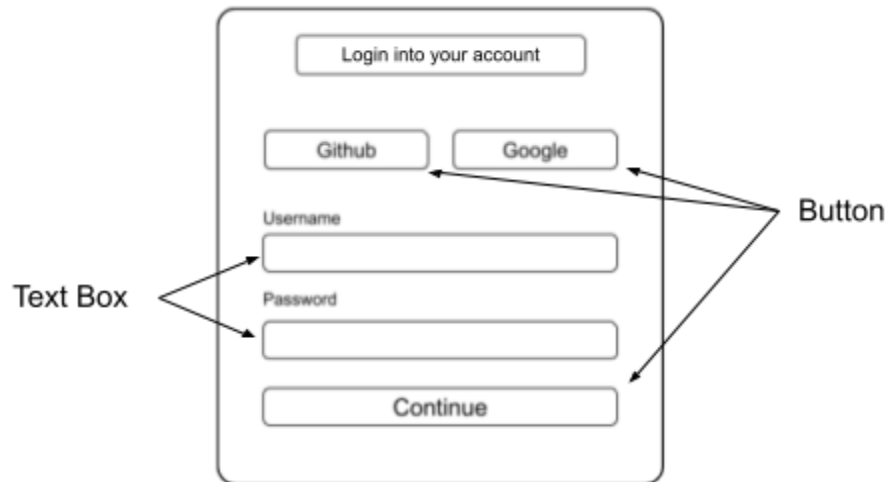


Diagram of a Login form. The form is enclosed in a rounded rectangle. At the top is a button labeled "Login into your account". Below it are two buttons labeled "Github" and "Google". Under these are two text boxes labeled "Username" and "Password". At the bottom is a button labeled "Continue". Arrows point from the labels "Text Box" and "Button" to their respective elements in the form.

Text Box

Button

Registration

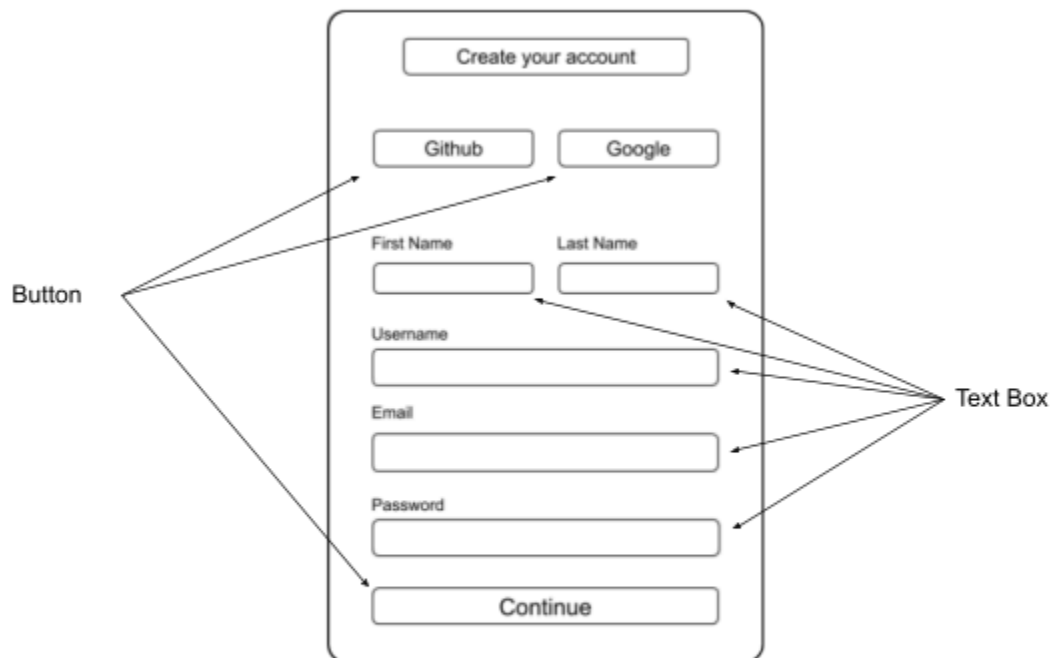
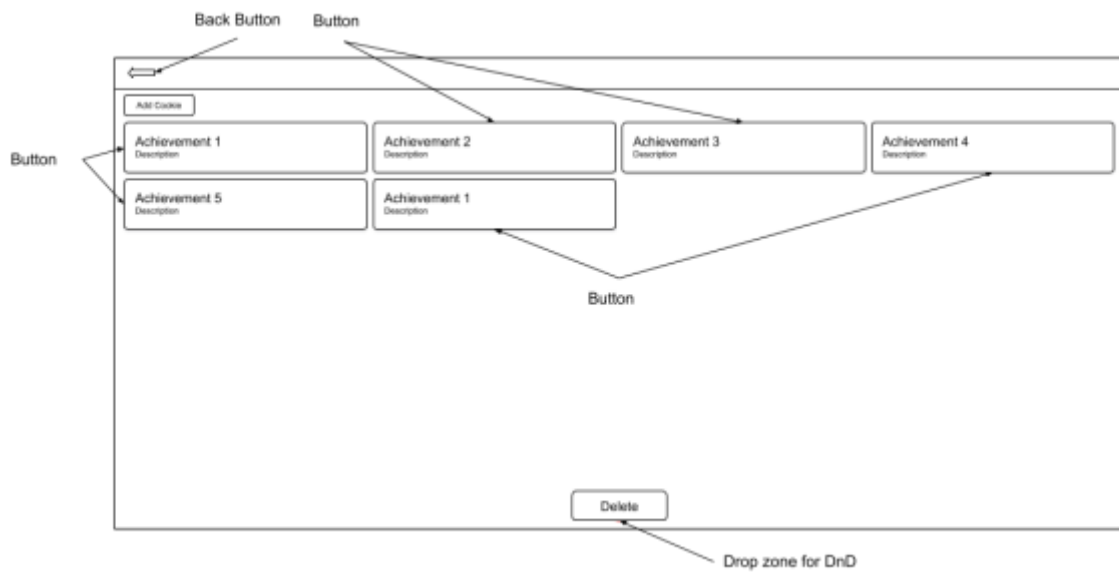


Diagram of a Registration form. The form is enclosed in a rounded rectangle. At the top is a button labeled "Create your account". Below it are two buttons labeled "Github" and "Google". Under these are two text boxes labeled "First Name" and "Last Name". Below those are four text boxes labeled "Username", "Email", "Password", and "Continue". Arrows point from the labels "Button" and "Text Box" to their respective elements in the form.

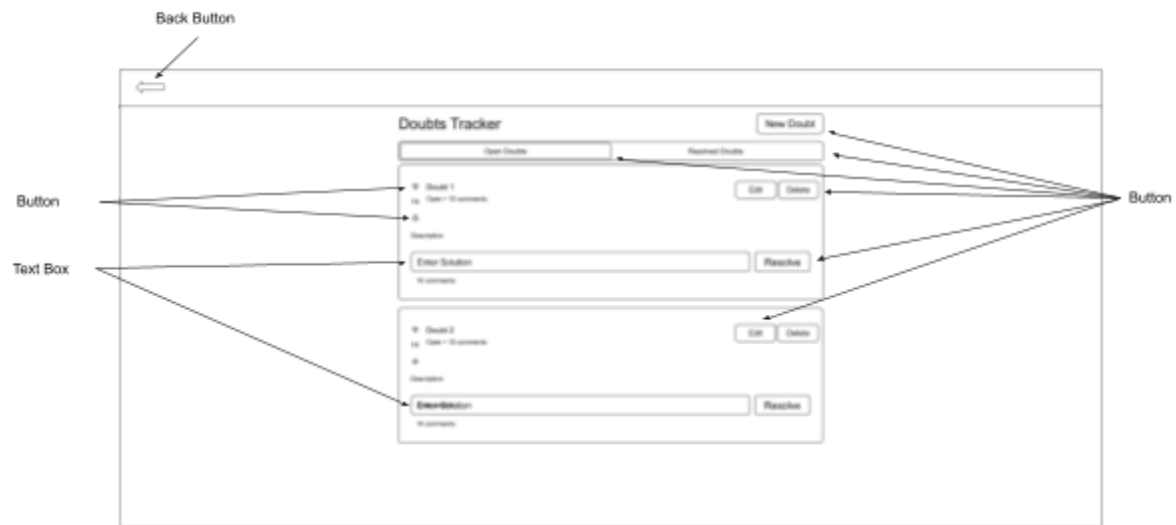
Button

Text Box

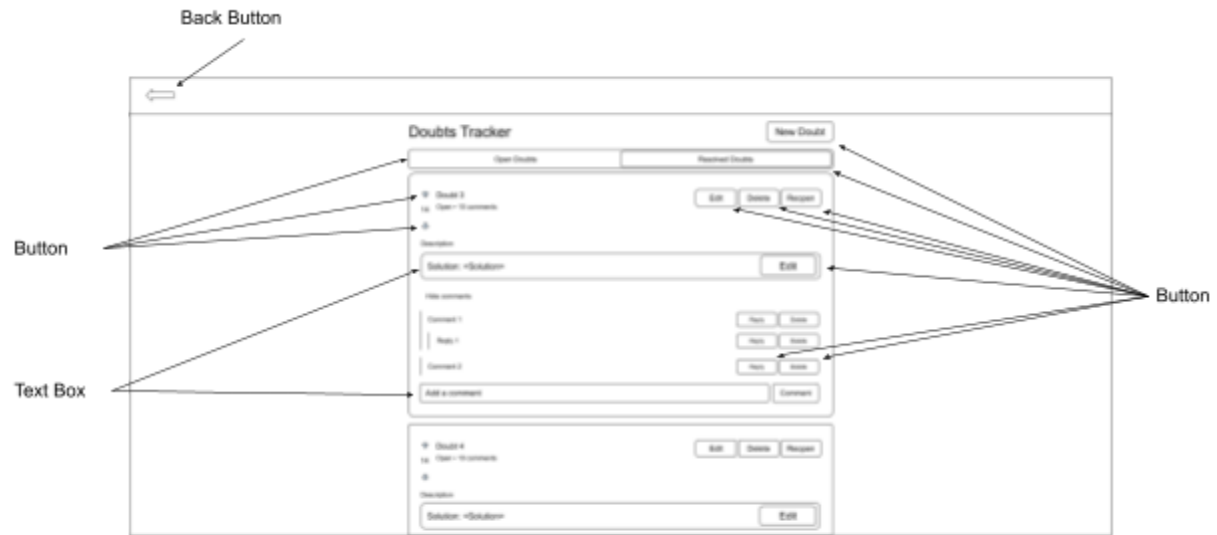
Cookie Jar



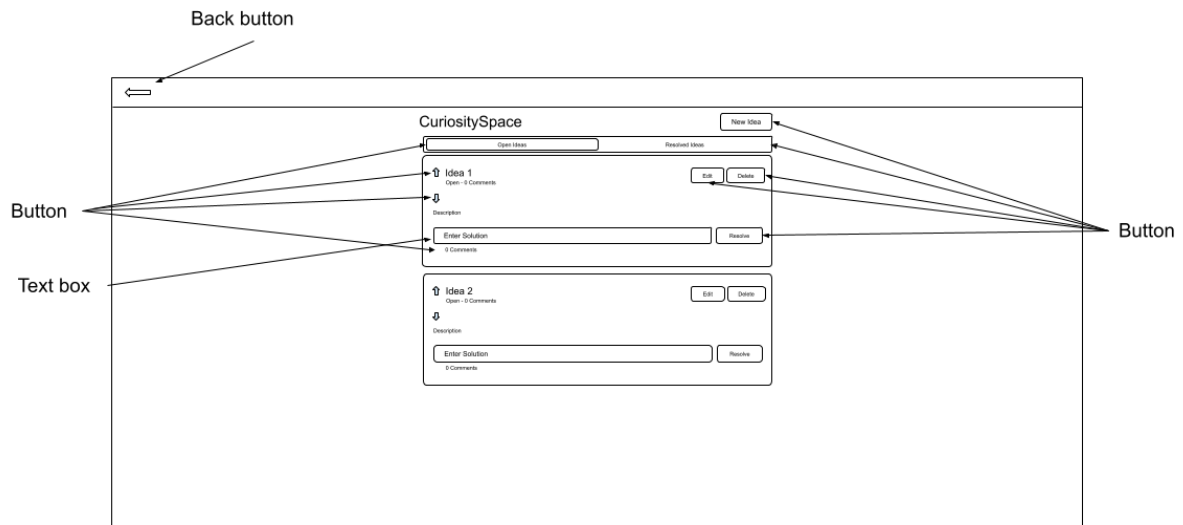
DoubtsTracker Open Doubts



DoubtsTracker Resolved Doubts



CuriositySpace Open Ideas



CuriositySpace Resolved Ideas



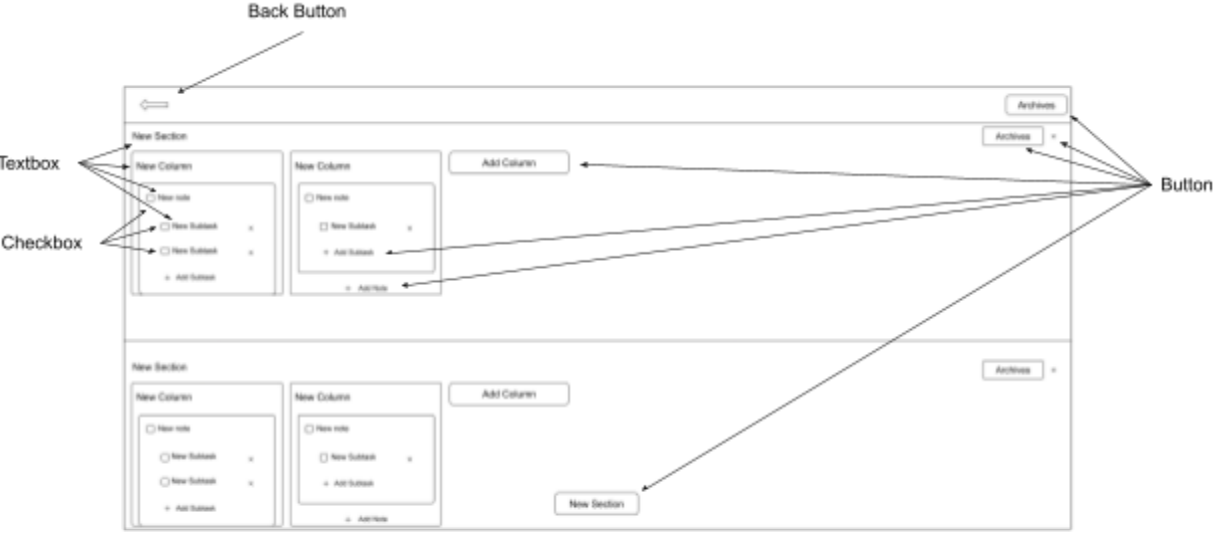
Notebooks



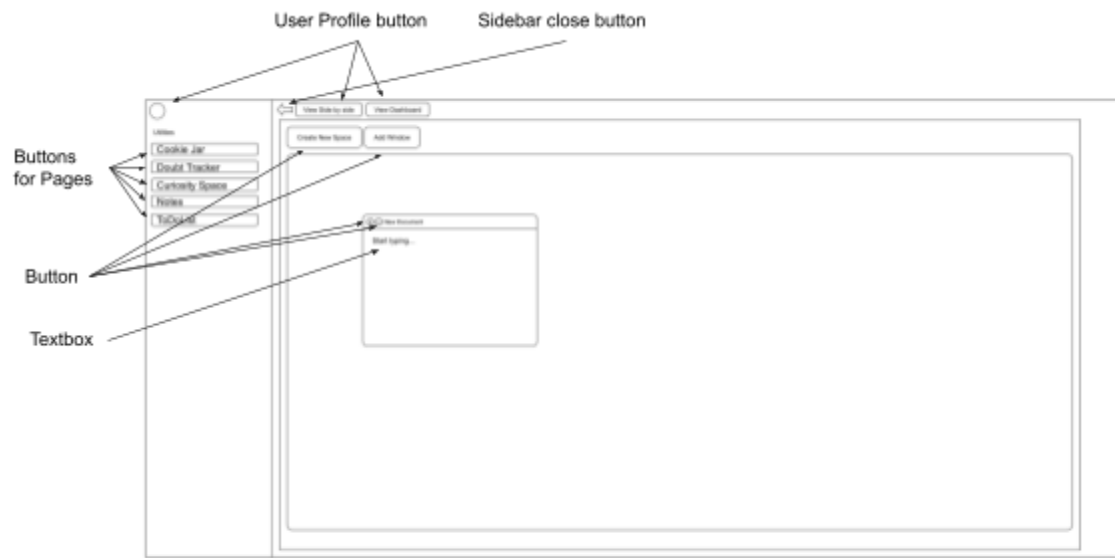
Notes



ToDoList



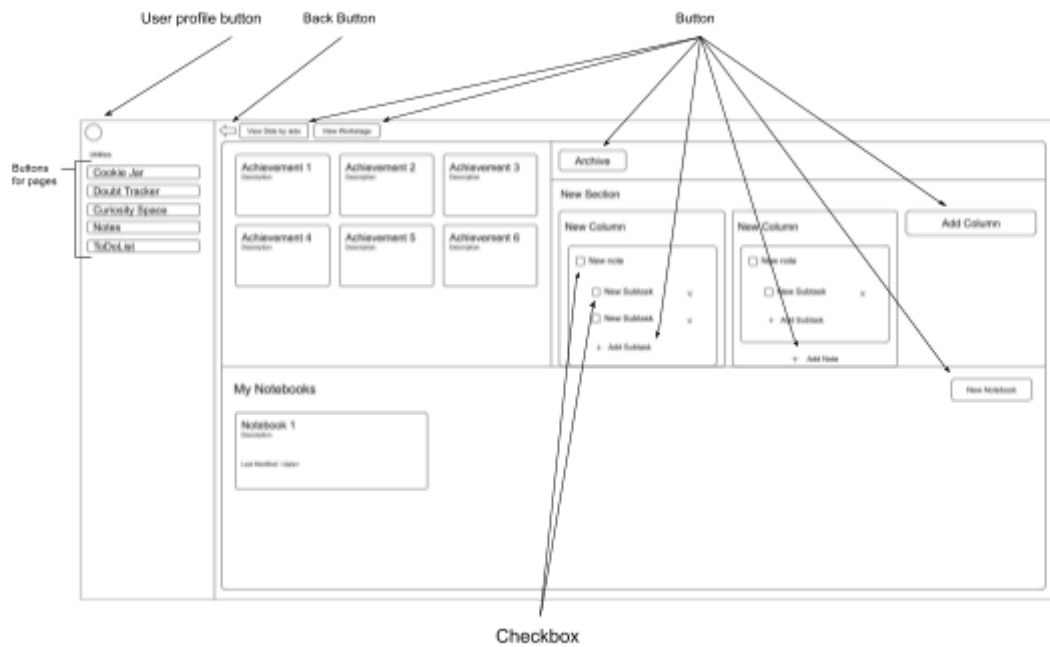
Workstage view main page



Side-by-Side view main page

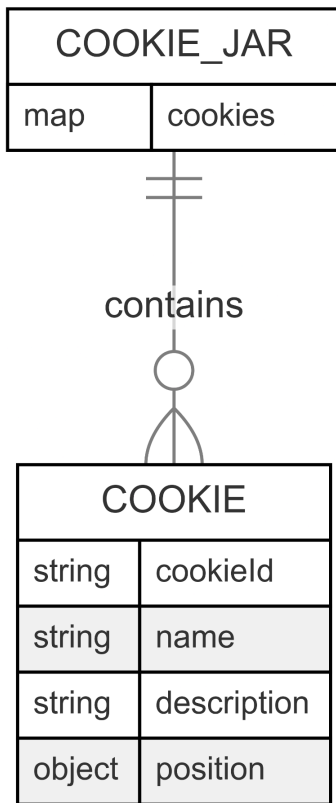


Dashboard view main page

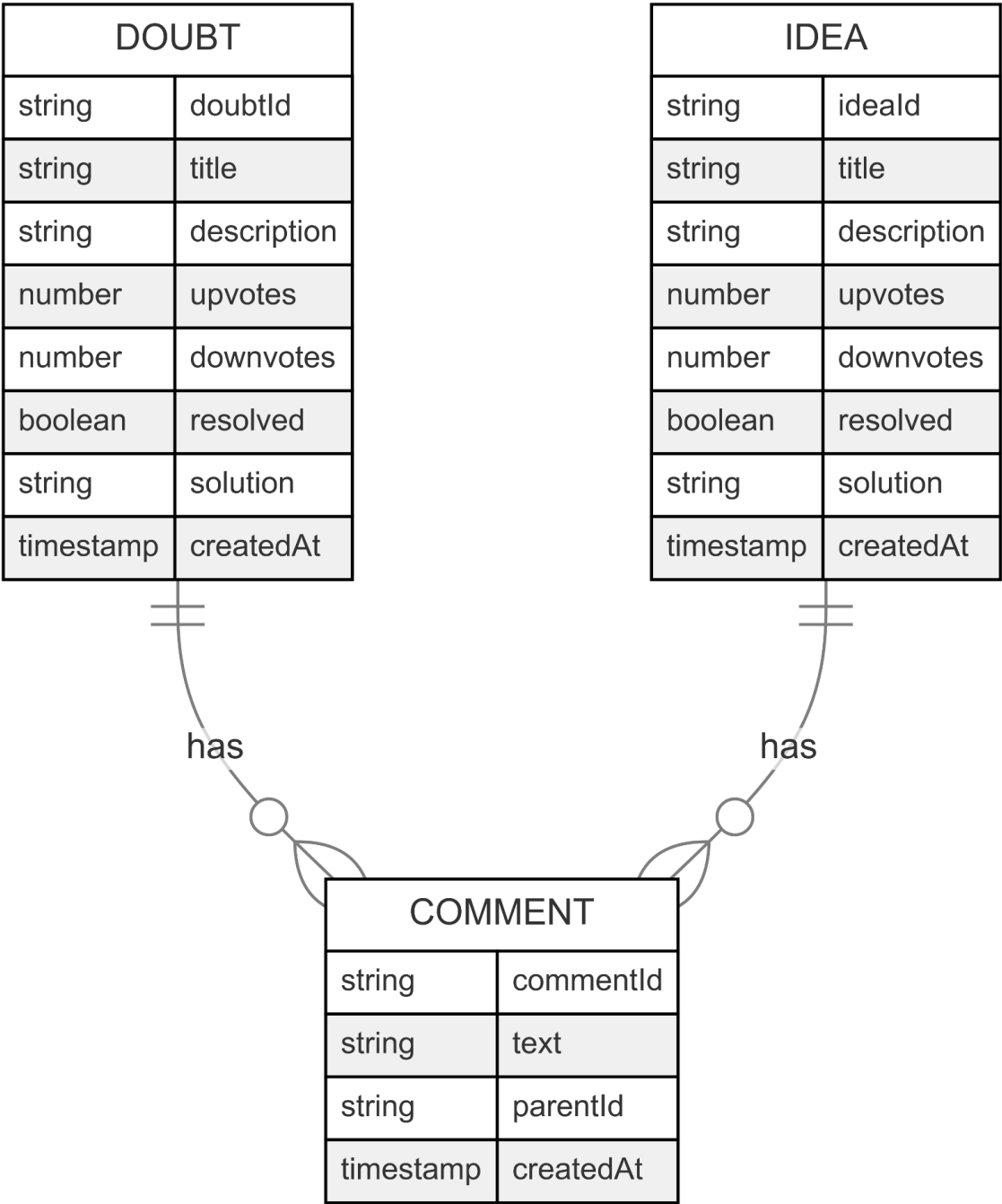


Database

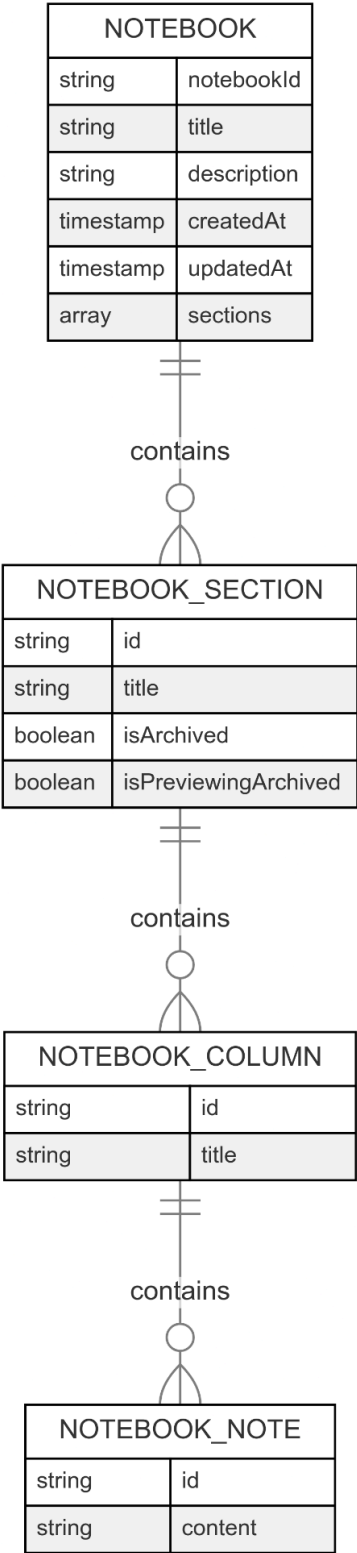
CookieJar



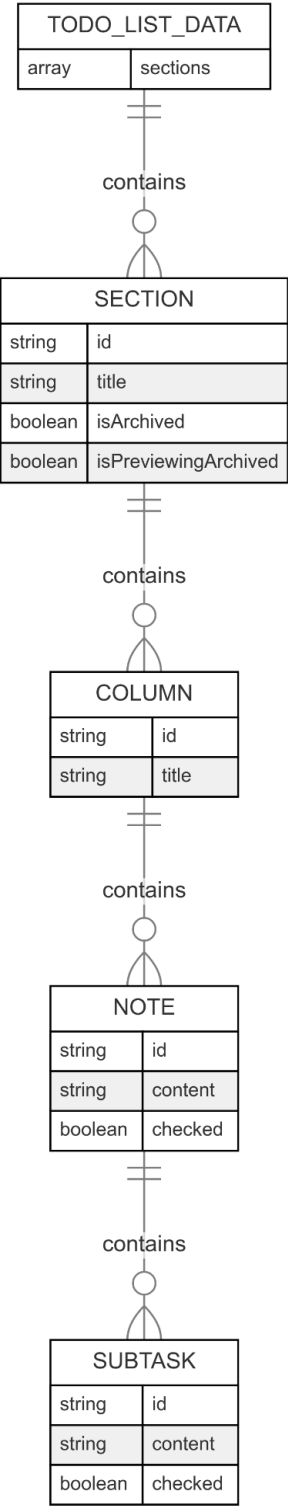
DoubtTracker and CuriositySpace



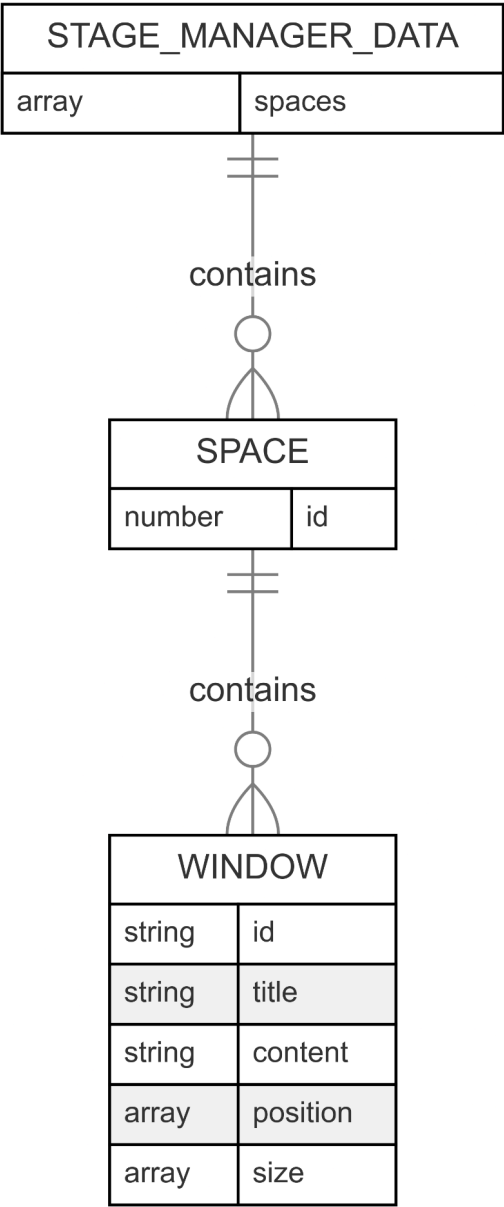
Notebooks



ToDoList



StageManager



Validation

Field Name	Data Validation Type	Explanation
Cookie Jar Module		
Cookie Name	• Presence Check	Must not be empty; User must provide a name for the cookie.
	• Length Check	Must be within a reasonable length (e.g., <= 100 characters).
	• Type Check	Must be text (string).
Cookie Description	• Presence Check	Must not be empty; User must provide a description for the cookie.
	• Length Check	Must be within a reasonable length (e.g., <= 500 characters).
	• Type Check	Must be text (string).
Doubt Tracker Module		
Doubt Title	• Presence Check	Must not be empty; User must provide a title for the doubt.
	• Length Check	Must be within a reasonable length (e.g., <= 100 characters).
	• Type Check	Must be text (string).
Doubt Description	• Presence Check	Must not be empty; User must provide a description for the doubt.
	• Length Check	Must be within a reasonable length (e.g., <= 1000 characters).

Field Name	Data Validation Type	Explanation
	• Type Check	Must be text (string).
Doubt Solution	• Presence Check	Must not be empty when marking doubt as 'Resolved'.
	• Length Check	Must be within a reasonable length (e.g., <= 1000 characters).
	• Type Check	Must be text (string).
Doubt Comment	• Presence Check	Must not be empty when submitting a comment.
	• Length Check	Must be within a reasonable length (e.g., <= 500 characters).
	• Type Check	Must be text (string).
Curiosity Space Module		
Idea Title	• Presence Check	Must not be empty; User must provide a title for the idea.
	• Length Check	Must be within a reasonable length (e.g., <= 100 characters).
	• Type Check	Must be text (string).
Idea Description	• Presence Check	Must not be empty; User must provide a description for the idea.
	• Length Check	Must be within a reasonable length (e.g., <= 1000 characters).
	• Type Check	Must be text (string).
Idea Solution	• Presence Check	Must not be empty when marking idea as 'Resolved'.

Field Name	Data Validation Type	Explanation
	• Length Check	Must be within a reasonable length (e.g., <= 1000 characters).
	• Type Check	Must be text (string).
Idea Comment	• Presence Check	Must not be empty when submitting a comment.
	• Length Check	Must be within a reasonable length (e.g., <= 500 characters).
	• Type Check	Must be text (string).
To-Do List Module		
Section Title	• Presence Check	Must not be empty; User must provide a title for the section.
	• Length Check	Must be within a reasonable length (e.g., <= 100 characters).
	• Type Check	Must be text (string).
Column Title	• Presence Check	Must not be empty; User must provide a title for the column.
	• Length Check	Must be within a reasonable length (e.g., <= 100 characters).
	• Type Check	Must be text (string).
Note Content	• Presence Check	Must not be empty; User must provide content for the note.
	• Length Check	Must be within a reasonable length (e.g., <= 1000 characters).

Field Name	Data Validation Type	Explanation
	• Type Check	Must be text (string).
Subtask Content	• Presence Check	Must not be empty; User must provide content for the subtask.
	• Length Check	Must be within a reasonable length (e.g., <= 500 characters).
	• Type Check	Must be text (string).
Continuous Info Space Module		
Notebook Title	• Presence Check	Must not be empty; User must provide a title for the notebook.
	• Length Check	Must be within a reasonable length (e.g., <= 100 characters).
	• Type Check	Must be text (string).
Section Title	• Presence Check	Must not be empty; User must provide a title for the section.
	• Length Check	Must be within a reasonable length (e.g., <= 100 characters).
	• Type Check	Must be text (string).
Column Title	• Presence Check	Must not be empty; User must provide a title for the column.
	• Length Check	Must be within a reasonable length (e.g., <= 100 characters).
	• Type Check	Must be text (string).

Field Name	Data Validation Type	Explanation
Note Content	• Presence Check	Must not be empty; User must provide content for the note.
	• Length Check	Must be within a reasonable length (e.g., <= 1000 characters).
	• Type Check	Must be text (string).
Notebook Description	• Length Check	Must be within a reasonable length (e.g., <= 500 characters).
	• Type Check	Must be text (string).
Note Description	• Length Check	Must be within a reasonable length (e.g., <= 500 characters).
	• Type Check	Must be text (string).
Stage Manager Module		
Space Name	• Presence Check	Must not be empty; User must provide a name for the Space.
	• Length Check	Must be within a reasonable length (e.g., <= 100 characters).
	• Type Check	Must be text (string).
Window Title	• Presence Check	Must not be empty; User must provide a title for the Window.
	• Length Check	Must be within a reasonable length (e.g., <= 100 characters).
	• Type Check	Must be text (string).

Field Name	Data Validation Type	Explanation
All Text Fields	<ul style="list-style-type: none">• Presence Check (where indicated)	Ensures that required text fields are not left blank.
	<ul style="list-style-type: none">• Type Check	Ensures that the input is of the expected data type (text/string).
	<ul style="list-style-type: none">• Length Check	Ensures that text inputs do not exceed reasonable length limits.

Test Plan

Module/Feature	Test Case	Test Data/Input	Type	Expected Result	Success Criteria (Reference)
Cookie Jar	Create Cookie	Name: "Achieved Goal", Description: "Completed IA Crit A"	Normal	New cookie card "Achieved Goal" with description "Completed IA Crit A" is created and displayed.	1, 2, 9
Cookie Jar	Create Cookie (Missing Name)	Description: "Just a description"	Abnormal	Error message (if implemented, otherwise handle gracefully), Cookie creation is prevented.	1, 9
Cookie Jar	Edit Cookie	Select "Achieved Goal", Change Description to "Revised IA Doc"	Normal	Cookie card "Achieved Goal" description updates to "Revised IA Doc".	3, 9
Cookie Jar	Delete Cookie	Select "Achieved Goal", Click "Delete"	Normal	Cookie card "Achieved Goal" is removed from the Cookie Jar.	4, 9
Cookie Jar	Reorder Cookies	Drag "Achieved Goal" cookie	Normal	"Achieved Goal" cookie is reordered	5, 9

Module/Feature	Test Case	Test Data/Input	Type	Expected Result	Success Criteria (Reference)
	(Drag & Drop)	to a new position		to the new position, and the order persists after refresh.	
Cookie Jar	Persistence Across Sessions	Create cookies, close app, reopen app	Normal	Cookies created in previous session are still present and displayed in the Cookie Jar.	2, 9
Doubt Tracker	Create Doubt	Title: "Maths Doubt", Description: "Integration issue"	Normal	New doubt card "Maths Doubt" with description "Integration issue" is created and displayed.	1, 2, 10
Doubt Tracker	Create Doubt (Missing Title)	Description: "Just a description"	Abnormal	Error message (if implemented, otherwise handle gracefully), Doubt creation is prevented.	1, 10
Doubt Tracker	Resolve Doubt	Select "Maths Doubt", Click "Resolve", Solution:	Normal	"Maths Doubt" card is marked as resolved, Solution "Use	8, 10

Module/Feature	Test Case	Test Data/Input	Type	Expected Result	Success Criteria (Reference)
		"Use substitution"		substitution" is displayed.	
Doubt Tracker	Reopen Doubt	Select "Maths Doubt" (Resolved), Click "Reopen"	Normal	"Maths Doubt" card is marked as open, Solution is hidden.	8, 10
Doubt Tracker	Upvote Doubt	Select "Maths Doubt", Click "Upvote"	Normal	Upvote count for "Maths Doubt" increments by 1.	6, 10
Doubt Tracker	Downvote Doubt	Select "Maths Doubt", Click "Downvote"	Normal	Downvote count for "Maths Doubt" increments by 1.	6, 10
Doubt Tracker	Add Comment	Select "Maths Doubt", Add comment: "Need more details"	Normal	Comment "Need more details" is added to "Maths Doubt" and displayed.	7, 10
Doubt Tracker	Edit Comment	Select comment "Need more details", Edit to "Clarify question"	Normal	Comment text updates to "Clarify question".	7, 10
Doubt Tracker	Delete Comment	Select comment "Clarify"	Normal	Comment "Clarify question" is	7, 10

Module/Feature	Test Case	Test Data/Input	Type	Expected Result	Success Criteria (Reference)
		question", Delete comment		removed from the Doubt card.	
Curiosity Space	Create Idea	Title: "New App Idea", Description: "AI powered note-taking"	Normal	New idea card "New App Idea" with description "AI powered note-taking" is created and displayed.	1, 2, 11
Curiosity Space	Resolve Idea	Select "New App Idea", Click "Resolve", Solution: "Research APIs"	Normal	"New App Idea" card is marked as resolved, Solution "Research APIs" is displayed.	8, 11
To-Do List	Create Section	Click "Add Section"	Normal	New section "New Section" is created and displayed.	1, 2, 12
To-Do List	Create Column	Select "New Section", Click "Add Column"	Normal	New column "New Column" is created within "New Section".	1, 2, 12
To-Do List	Create Note/Task	Select "New Column", Click "Add Note"	Normal	New note "New note" is created	1, 2, 12

Module/Feature	Test Case	Test Data/Input	Type	Expected Result	Success Criteria (Reference)
				within "New Column".	
To-Do List	Check/Uncheck Task	Check checkbox next to "New note"	Normal	Note text is visually marked as completed (e.g., line-through).	12
To-Do List	Delete Section	Select "New Section", Click "Archive"	Normal	"New Section" is archived (removed from main view, accessible in archive).	4, 12
To-Do List	Reorder Sections (Drag & Drop)	Drag "New Section" to a new position	Normal	"New Section" is reordered to the new position.	5, 12
To-Do List	Add Subtask	Select "New note", Click "Add Subtask"	Normal	New subtask "New Subtask" is created under "New note".	1, 2, 12
Continuous Info Space	Create Notebook	Click "New Notebook", Title: "Maths Notes"	Normal	New notebook card "Maths Notes" is created and displayed.	1, 2, 13

Module/Feature	Test Case	Test Data/Input	Type	Expected Result	Success Criteria (Reference)
Continuous Info Space	Delete Notebook	Select "Maths Notes", Click "Delete"	Normal	"Maths Notes" notebook card is removed.	4, 13
Continuous Info Space	Create Section in Notebook	Open "Maths Notes", Click "Add Section"	Normal	New section "New Section" is created within "Maths Notes" notebook.	1, 2, 13
Continuous Info Space	Create Column in Section	Open "Maths Notes", Select "New Section", Click "Add Column"	Normal	New column "New Column" is created within "New Section" in "Maths Notes".	1, 2, 13
Continuous Info Space	Create Note in Column	Open "Maths Notes", Select "New Column", Click "Add Note"	Normal	New note "New note" is created within "New Column" in "Maths Notes".	1, 2, 13
Stage Manager	Create Space	Click "Create New Space"	Normal	New space "Space 2" (if Space 1 exists) is created.	1, 2, 14
Stage Manager	Delete Space	Select "Space 2", Click "Delete Space"	Normal	"Space 2" is deleted.	4, 14

Module/Feature	Test Case	Test Data/Input	Type	Expected Result	Success Criteria (Reference)
Stage Manager	Create Window	Select "Space 1", Click "Add Window"	Normal	New window "New Document" is created in "Space 1".	1, 2, 14
Stage Manager	Delete Window	Select "New Document", Click "Close" (X)	Normal	"New Document" window is deleted from "Space 1".	4, 14
Stage Manager	Move Window (Drag Title Bar)	Drag "New Document" window	Normal	"New Document" window is moved to the dragged position.	14
Stage Manager	Resize Window (Drag Resizer)	Drag the corner resizer of "New Document" window	Normal	"New Document" window is resized as dragged.	14
Stage Manager	Persist Layout	Create Spaces & Windows, arrange them, close & reopen app	Normal	Spaces and Windows layout (arrangement, positions, sizes) are persisted across sessions.	2, 14
General Application	Intuitive UI Navigation	Navigate through all modules	Normal	User can easily navigate between modules and understand	15

Module/Feature	Test Case	Test Data/Input	Type	Expected Result	Success Criteria (Reference)
				the UI elements without extensive learning.	
General Application	Cross-browser Compatibility (Chrome)	Access application via Chrome browser	Normal	Application functions correctly and is usable in Chrome without errors or UI issues.	16
General Application	Cross-browser Compatibility (Firefox)	Access application via Firefox browser	Normal	Application functions correctly and is usable in Firefox without errors or UI issues.	16
General Application	Cross-browser Compatibility (Safari)	Access application via Safari browser	Normal	Application functions correctly and is usable in Safari without errors or UI issues.	16
General Application	Cross-browser Compatibility (Edge)	Access application via Edge browser	Normal	Application functions correctly and is usable in Edge without errors or UI issues.	16
General Application	Cross-device Compatibility (Desktop)	Access application	Normal	Application functions correctly and	16

Module/Feature	Test Case	Test Data/Input	Type	Expected Result	Success Criteria (Reference)
		on a desktop computer		is usable on a desktop computer, UI is responsive.	
General Application	Cross-device Compatibility (Laptop)	Access application on a laptop	Normal	Application functions correctly and is usable on a laptop, UI is responsive.	16
General Application	Cross-device Compatibility (Tablet)	Access application on a tablet	Normal	Application functions correctly and is usable on a tablet, UI is responsive (if tablet support is implemented)	16
General Application	User Authentication (Sign-in)	Attempt to access application without signing in	Normal	User is redirected to the sign-in page and cannot access application features without authentication.	17
General Application	Real-time Data Sync	Open app on two devices with same account, make	Normal	Changes made on one device are reflected in real-time (or	18

Module/Feature	Test Case	Test Data/Input	Type	Expected Result	Success Criteria (Reference)
		changes on one		near real-time) on the other device.	
General Application	Fast Data Retrieval	Open each module and perform data-loading actions	Normal	Modules and data load quickly with minimal loading time, providing a responsive user experience.	19
All Modules/Features	Input Validation (e.g., long text)	Enter very long text strings in all input fields	Extreme	Application handles long inputs gracefully without crashing or causing UI issues.	Robustness (Implied)
All Modules/Features	No Input	Attempt to perform actions without entering required information	Abnormal	Application prevents actions or provides informative error messages when required input is missing.	Input Validation (Implied)

Testing General and Database Functionality of Information Management System:

User Authentication:

1. **Register New User Account:** Register using a new email and password. Verify successful account creation and redirection to the application dashboard. (Checks Firebase Authentication for new user creation and proper user session setup).
2. **Attempt Duplicate Registration:** Try registering again using the *same* email as in test 1. Verify the application *prevents* duplicate registration and displays an appropriate error message. (Checks Firebase Authentication to ensure duplicate user creation is blocked).
3. **Login with Valid Credentials:** Login using the email and password created in test 1. Verify successful login and redirection to the application dashboard. (Checks Firebase Authentication to match email and password, and establishes user session).
4. **Login with Invalid Credentials:** Attempt login using an *incorrect* password for the registered email. Verify login is *prevented* and an appropriate error message is displayed. (Checks Firebase Authentication to ensure login fails with incorrect credentials).
5. **Multiple Account Isolation:** Log in with two different user accounts. Verify that data (cookies, doubts, ideas, to-do lists, notebooks, spaces/windows) is isolated between the two accounts and not accessible across accounts. (Checks Firebase Firestore rules to ensure data segregation based on user ID).

Cookie Jar Module:

6. **Create New Cookie:** In the Cookie Jar, create a new cookie with a name and description. Verify the cookie card is created and displayed in the Cookie Jar. (Checks Firebase Firestore to ensure new cookie data is stored under the user's cookieJar collection and data is persisted).
7. **Edit Existing Cookie:** Edit the name and description of an existing cookie. Verify the cookie card is updated with the new information in the Cookie Jar. (Checks Firebase Firestore to ensure cookie data is updated in the database and changes are reflected in the UI).
8. **Delete Cookie:** Delete an existing cookie from the Cookie Jar. Verify the cookie card is removed from the Cookie Jar. (Checks Firebase Firestore to ensure cookie data is deleted from the database and UI is updated).
9. **Reorder Cookies (Drag and Drop):** Drag and drop cookie cards to reorder them in the Cookie Jar. Verify the new order is saved and persists after refreshing or reopening the application. (Checks Firebase Firestore to ensure cookie positions are updated in the database and the new order is maintained).

Doubt Tracker & Curiosity Space (Idea Tracker) Modules:

10. **Create New Doubt/Idea:** In the Doubt Tracker/Curiosity Space, create a new doubt/idea with a title and description. Verify the doubt/idea card is created and displayed in the respective module list. (Checks Firebase Firestore to ensure new doubt/idea data is stored under the user's posts/nugget collection respectively and data is persisted).
11. **Edit Existing Doubt/Idea:** Edit the title and description of an existing doubt/idea. Verify the doubt/idea card is updated with the new information. (Checks Firebase Firestore to ensure doubt/idea data is updated in the database and changes are reflected in the UI).
12. **Delete Doubt/Idea:** Delete an existing doubt/idea. Verify the doubt/idea card is removed from the list. (Checks Firebase Firestore to ensure doubt/idea data is deleted from the database and UI is updated).
13. **Resolve Doubt/Idea:** Resolve an open doubt/idea by adding a solution. Verify the doubt/idea is marked as resolved and the solution is displayed. (Checks Firebase Firestore to ensure the 'resolved' status and 'solution' are updated in the database and UI reflects the resolved state).
14. **Reopen Resolved Doubt/Idea:** Reopen a resolved doubt/idea. Verify the doubt/idea is marked as open again and the solution is hidden. (Checks Firebase Firestore to ensure the 'resolved' status and 'solution' are reverted in the database and UI reflects the reopened state).
15. **Upvote/Downvote Doubt/Idea:** Upvote or downvote a doubt/idea. Verify the vote count is updated on the doubt/idea card. (Checks Firebase Firestore to ensure 'upvotes' or 'downvotes' count is incremented in the database and the vote count is updated in the UI).
16. **Add Comment to Doubt/Idea:** Add a comment to a doubt/idea. Verify the comment is displayed in the comment section. (Checks Firebase Firestore to ensure the comment data is stored under the doubt/idea's comments subcollection and the comment is displayed in the UI).
17. **Edit Comment:** Edit an existing comment. Verify the comment text is updated. (Checks Firebase Firestore to ensure the comment data is updated in the database and the edited comment is displayed in the UI).
18. **Delete Comment:** Delete an existing comment. Verify the comment is removed from the comment section. (Checks Firebase Firestore to ensure the comment data is deleted from the database and the comment is removed from the UI).

To-Do List Module:

19. **Create New Section:** In the To-Do List, create a new section. Verify a new section is added to the To-Do List. (Checks Firebase Firestore to ensure the new section data is stored and persisted).
20. **Edit Section Title:** Edit the title of a section. Verify the section title is updated in the To-Do List. (Checks Firebase Firestore to ensure the section title is updated in the database and UI reflects the change).
21. **Delete Section:** Delete an existing section. Verify the section and all its columns and notes are removed from the To-Do List. (Checks Firebase Firestore to ensure the section data and associated data are deleted from the database and UI is updated).

22. **Create New Column:** In a section, create a new column. Verify a new column is added to the section. (Checks Firebase Firestore to ensure the new column data is stored under the section and persisted).
23. **Edit Column Title:** Edit the title of a column. Verify the column title is updated in the To-Do List. (Checks Firebase Firestore to ensure the column title is updated in the database and UI reflects the change).
24. **Delete Column:** Delete an existing column. Verify the column and all its notes are removed from the section. (Checks Firebase Firestore to ensure the column data and associated notes are deleted from the database and UI is updated).
25. **Add New Note/Task:** In a column, add a new note/task. Verify a new note/task is added to the column. (Checks Firebase Firestore to ensure the new note/task data is stored under the column and persisted).
26. **Edit Note/Task Content:** Edit the content of a note/task. Verify the note/task content is updated in the To-Do List. (Checks Firebase Firestore to ensure the note/task content is updated in the database and UI reflects the change).
27. **Delete Note/Task:** Delete an existing note/task. Verify the note/task is removed from the column. (Checks Firebase Firestore to ensure the note/task data is deleted from the database and UI is updated).
28. **Check/Uncheck Note (Task Completion):** Check and uncheck a note/task to mark it as complete/incomplete. Verify the check status is saved and visually indicated (e.g., strikethrough). (Checks Firebase Firestore to ensure the 'checked' status of the note/task is updated and the UI reflects the completion status).
29. **Reorder Columns (Drag and Drop):** Drag and drop columns within a section to reorder them. Verify the new column order is saved and persists. (Checks Firebase Firestore to ensure the column order within the section is updated and the new order is maintained).
30. **Reorder Notes (Drag and Drop):** Drag and drop notes within a column to reorder them. Verify the new note order is saved and persists. (Checks Firebase Firestore to ensure the note order within the column is updated and the new order is maintained).
31. **Move Note between Columns (Drag and Drop):** Drag and drop a note from one column to another (within the same or different section). Verify the note moves to the new column and is removed from the original column. (Checks Firebase Firestore to ensure the note is moved to the new column's data in the database and removed from the original column).
32. **Add Subtask to Note:** Add a subtask to a note. Verify the subtask is added and displayed under the note. (Checks Firebase Firestore to ensure the subtask data is stored under the parent note and persisted).
33. **Delete Subtask:** Delete a subtask. Verify the subtask is removed from the note. (Checks Firebase Firestore to ensure the subtask data is deleted from the database and UI is updated).
34. **Check/Uncheck Subtask:** Check and uncheck a subtask to mark it as complete/incomplete. Verify the subtask's check status is saved and visually indicated. (Checks Firebase Firestore to ensure the 'checked' status of the subtask is updated and the UI reflects the completion status).

35. **Reorder Subtasks (Drag and Drop):** Drag and drop subtasks within a note to reorder them. Verify the new subtask order is saved and persists. (Checks Firebase Firestore to ensure the subtask order within the note is updated and the new order is maintained).
36. **Move Subtask between Notes (Drag and Drop):** Drag and drop a subtask from one note to another (within the same column). Verify the subtask moves to the new note and is removed from the original note. (Checks Firebase Firestore to ensure the subtask is moved to the new note's data in the database and removed from the original note).

Continuous Information Space Module:

37. **Create New Notebook:** In the Continuous Information Space, create a new notebook. Verify a new notebook card is created in the Notebook Manager. (Checks Firebase Firestore to ensure new notebook data is stored under the user's notebooks collection and persisted).
38. **Edit Notebook Title/Description:** Edit the title and description of an existing notebook. Verify the notebook card is updated with the new information. (Checks Firebase Firestore to ensure notebook data is updated in the database and changes are reflected in the UI).
39. **Delete Notebook:** Delete an existing notebook. Verify the notebook card is removed from the Notebook Manager. (Checks Firebase Firestore to ensure notebook data is deleted from the database and UI is updated).
40. **Create New Section in Notebook:** Open a notebook and create a new section. Verify a new section is added to the notebook. (Checks Firebase Firestore to ensure the new section data is stored under the notebook and persisted).
41. **Edit Section Title (Notebook):** Edit the title of a section within a notebook. Verify the section title is updated in the notebook. (Checks Firebase Firestore to ensure the section title is updated in the database and UI reflects the change).
42. **Delete Section (Notebook):** Delete a section from a notebook. Verify the section and all its columns and notes are removed from the notebook. (Checks Firebase Firestore to ensure the section data and associated data are deleted from the database and UI is updated).
43. **Create New Column (Notebook Section):** In a notebook section, create a new column. Verify a new column is added to the section. (Checks Firebase Firestore to ensure the new column data is stored under the section and persisted).
44. **Edit Column Title (Notebook Section):** Edit the title of a column within a notebook section. Verify the column title is updated in the notebook section. (Checks Firebase Firestore to ensure the column title is updated in the database and UI reflects the change).
45. **Delete Column (Notebook Section):** Delete a column from a notebook section. Verify the column and all its notes are removed from the section. (Checks Firebase Firestore to ensure the column data and associated notes are deleted from the database and UI is updated).

- 46. **Add New Note (Notebook Column):** In a notebook column, add a new note. Verify a new note is added to the column. (Checks Firebase Firestore to ensure the new note data is stored under the column and persisted).
- 47. **Edit Note Content (Notebook):** Edit the content of a note within a notebook column. Verify the note content is updated in the notebook. (Checks Firebase Firestore to ensure the note content is updated in the database and UI reflects the change).
- 48. **Delete Note (Notebook):** Delete a note from a notebook column. Verify the note is removed from the column. (Checks Firebase Firestore to ensure the note data is deleted from the database and UI is updated).
- 49. **Reorder Sections (Drag and Drop - Notebook):** Drag and drop sections within a notebook to reorder them. Verify the new section order is saved and persists. (Checks Firebase Firestore to ensure the section order within the notebook is updated and the new order is maintained).
- 50. **Reorder Columns (Drag and Drop - Notebook Section):** Drag and drop columns within a notebook section to reorder them. Verify the new column order is saved and persists. (Checks Firebase Firestore to ensure the column order within the notebook section is updated and the new order is maintained).
- 51. **Reorder Notes (Drag and Drop - Notebook Column):** Drag and drop notes within a notebook column to reorder them. Verify the new note order is saved and persists. (Checks Firebase Firestore to ensure the note order within the notebook column is updated and the new order is maintained).

Stage Manager Module:

- 52. **Create New Space:** In the Stage Manager, create a new space. Verify a new space is added to the space switcher. (Checks Firebase Firestore to ensure new space data is stored and persisted for the user's stage manager data).
- 53. **Delete Space:** Delete an existing space. Verify the space is removed from the space switcher. (Checks Firebase Firestore to ensure space data is deleted from the database and UI is updated).
- 54. **Switch Between Spaces:** Switch between different created spaces. Verify the application switches to the selected space and displays its windows. (Checks UI state to ensure correct space and its windows are loaded and displayed).
- 55. **Create New Window in Space:** In a space, create a new window. Verify a new window is added to the current space. (Checks Firebase Firestore to ensure new window data is stored under the current space and persisted).
- 56. **Delete Window:** Delete an existing window from a space. Verify the window is removed from the space. (Checks Firebase Firestore to ensure window data is deleted from the database and UI is updated).
- 57. **Move Window (Drag):** Drag a window within a space to a new position. Verify the window moves to the new position and the new position is saved. (Checks Firebase Firestore to ensure window position is updated in the database and the new position is maintained).

58. **Resize Window (Drag Edges):** Resize a window by dragging its edges. Verify the window resizes and the new size is saved. (Checks Firebase Firestore to ensure window size is updated in the database and the new size is maintained).
59. **Persistence of Spaces and Windows Layout:** Create spaces and windows, arrange them, and then close and reopen the application (or refresh the page after logging out and back in). Verify that the spaces and windows are restored in the same layout and configuration as before. (Checks Firebase Firestore to ensure the entire stage manager data including spaces and windows layouts is loaded and persisted across sessions).

General Application Functionality:

60. **Cross-Browser Compatibility:** Test all core functionalities (creating, editing, deleting items in each module, drag-and-drop, login/logout, space switching) on major web browsers (Chrome, Firefox, Safari, Edge). Verify that the application functions correctly and without UI issues on each browser.
61. **Cross-Device Compatibility:** Test all core functionalities on different devices (desktop, laptop, tablet - if applicable). Verify that the application is usable and responsive on different screen sizes and input methods.
62. **User Authentication Security:** Attempt to access another user's data by manually changing user IDs in the browser's local storage or making direct Firestore requests (if you know how to do this for testing purposes). Verify that Firebase security rules prevent unauthorized data access. (Verifies Firebase Firestore rules are correctly configured and enforced).
63. **Real-time Data Synchronization:** Open the application on two different devices or browser windows logged in with the same user account. Make changes in one location (e.g., add a cookie, check a to-do item). Verify that the changes are reflected in real-time on the other device/window. (Observes real-time updates in the UI across multiple instances of the application).
64. **Fast Data Retrieval:** Navigate to each module (Cookie Jar, Doubt Tracker, etc.) and observe the loading time for data to appear. Verify that data is retrieved and displayed quickly, ensuring a responsive user experience. (Subjective observation of loading times and application responsiveness).
65. **Intuitive User Interface:** Ask Isht (or another representative user) to perform common tasks within the application (e.g., create a to-do list, add a doubt, organize notebooks). Observe their ease of use and gather feedback on the intuitiveness of the navigation and UI elements. (Subjective usability assessment based on user observation and feedback).