BETA TEST PLAN – BEDROCK

1. Core Functionalities for Beta Version

Feature Name	Description	Priority (High/Medium/Low)	Changes Since Tech3
Account access - Email	The user is able to use theiremail address to register and login to Bedrock	High	Moved from Privy to thirdweb as login provider
Account access - Wallet	The user is able to use a blockchain wallet provider (i.e. Metamask, Ledger) to register and login into Bedrock	High	Moved from Privy to thirdweb as login provider
File storage - Upload	The user is able to upload their files to Bedrock to be stored securely on the Aleph Network	High	None
File storage - Download	The user is able to retrieve their files by downloading them from the Aleph Network	High	None
File storage - Organization	The user is able to organize their files by creating folders and moving, copying, duplicating or renaming files through folders	Medium	None
File storage - Access	The user is able to list their files and use the interface to look for specific files	Medium	None
File storage - Bulk	The user is able to perform an action (move, copy, delete) on multiple files at a time	Medium	None
File storage - Deletion	The user is able to retrieve a file that has been deleted thanks to a recycle bin feature that will keep any deleted file for 30 days	Medium	None
File storage - Sharing (private)	The user is able to share their private files with other Bedrock users and with read-only or read-write permissions	High	None
File storage - Sharing (public)	The user is able to give public access to their files	Low	None
User interaction - Usernames	The user is able to be identified by creating a username that will be registered on the blockchain	Medium	None
User interaction - Contacts	The user is able to retrieve and register the users he interacted with as contacts	Medium	None

Feature Name	Description	Priority (High/Medium/Low)	Changes Since Tech3
Artificial Intelligence - Chat	The user is able to chat with a private LibertAI model integrated to Bedrock	Medium	None
Artificial Intelligence - Context	The user is able to pass a list of files that will be used by LibertAI as a Knowledge base. The user is able to create multiple Knowledge bases and choose which one they want to use when they interact with the model	Medium	None
Balance - Top up	The user is able to top up their Bedrock balance	Low	Aleph is still free to use for our use case / LibertAI is starting to introduce payments since Apr/May 2025

---

2. Beta Testing Scenarios

2.1 User Roles

Role Name	Description
User.	A user with an account connected on Bedrock (via any available method such as an EVM wallet, an email or social login)
Guest	A visitor of the app that interacts with it without being connected

2.2 Test Scenarios

For each core functionality, provide detailed test scenarios.

Scenario 1: Account access - Email

- **Role Involved:** Guest
- **Objective:** Be able to create an account or login with an email address
- **Preconditions:** The visitor is not already connected with an account
- **Test Steps:**
    1. The Guest accesses the Bedrock page and should be redirected to the authentication page, and inputs their email

2. The Guest checks his email inbox for the code he received from Thirdweb
3. The user enters the code he received in the Thirdweb modal on the Bedrock app
4. If it's an account creation, the user fills the modal to select his username and clicks the button to submit his choice
5. Opening https://app.ens.domains/USERNAME.bedrock-app.eth (https://app.ens.domains/USERNAME.bedrock-app.eth) and replacing USERNAME with the actual username should show a registered username with an address linked that is the address of the user's wallet generated by Thirdweb for his email

- **Expected Outcome:** The Guest should become a User by having an account connected on Bedrock by using their email

Scenario 2: Account access - Wallet

- **Role Involved:** Guest
- **Objective:** Be able to create an account or login with an EVM wallet (injected browser extension like Rabby / Metamask, any WalletConnect wallet...)
- **Preconditions:** The visitor is not already connected with an account
- **Test Steps:**
    1. The Guest accesses the Bedrock page and should be redirected to the authentication page, and uses their provider to connect their wallet
    2. The guest signs the message showed in their wallet
    3. If it's an account creation, the user fills the modal to select an username and clicks the button to submit his choice
    4. Opening https://app.ens.domains/USERNAME.bedrock-app.eth (https://app.ens.domains/USERNAME.bedrock-app.eth) and replacing USERNAME with the actual username should show a registered username with an address linked that is the address of the user's wallet
- **Expected Outcome:** The Guest should become an User by having an account connected on Bedrock by using their wallet

Scenario 3: File storage - Upload

- **Role Involved:** User
- **Objective:** The user need to be able to upload their files on the aleph network
- **Preconditions:** To have a file available to upload on his local system
- **Test Steps:**
    1. The user clicks on the "+" button and chooses the add file action
    2. The user selects the file to be uploaded from their computer
- **Expected Outcome:** The user can see the file in the file list when the file has finished uploading

Scenario 4: File storage - Download

- **Role Involved:** User
- **Objective:** Be able to donwload previously uploaded files
- **Preconditions:** To have uploaded at least one file
- **Test Steps:**

1. The user selects a file to download
2. The user clicks on the Download action

- **Expected Outcome:** The file is downloaded on the user's local system

Scenario 5: File storage - Organization (Create folder)

- **Role Involved:** User
- **Objective:** Be able to create folders
- **Preconditions:** To have uploaded at least one file
- **Test Steps:**
    1. The user clicks on the "+" button and chooses the add folder action
    2. The user inputs the name of the folder
    3. The user moves files into their newly created folder
- **Expected Outcome:** The folder is created taking in consideration the current user path, and clicking on it shows the files inside the folder

Scenario 6: File storage - Organization (Delete folder)

- **Role Involved:** User
- **Objective:** Delete a folder
- **Preconditions:** To have uploaded at least one file and moved it into a folder
- **Test Steps:**
    1. The user selects a folder to delete
    2. The user click on the Delete action
    3. The user goes to the Trash page and sees their folder
    4. The user permanently deletes the folder
- **Expected Outcome:** The folder and every file in it are deleted from the Aleph Network and not accessible anymore

Scenario 7: File storage - Organization (Rename a file or folder)

- **Role Involved:** User
- **Objective:** Be able to rename a folder or a file
- **Preconditions:** To have at least one file uploaded or one folder created
- **Test Steps:**
    1. The user selects the file or folder
    2. The user clicks on the Rename action
    3. The user chooses the new name
- **Expected Outcome:** The name is correctly changed and the file or folder appears with the new name

Scenario 8: File storage - Organization (Move a file or folder)

- **Role Involved:** User
- **Objective:** Be able to move a folder or a file
- **Preconditions:** To have at least one file uploaded and one folder created or two folders created
- **Test Steps:**

    1. The user selects the file or folder
    2. The user clicks on the move action
    3. The user chooses the new path that must include a correct folder
- **Expected Outcome:** The path is correctly changed and the file or folder appears within the new location

Scenario 9: File storage - Organization (Copy a file or folder)

- **Role Involved:** User
- **Objective:** Be able to copy a folder or a file
- **Preconditions:** To have at least one file or folder
- **Test Steps:**
    1. The user selects the file or folder
    2. The user clicks on the copy action
- **Expected Outcome:** The file or the folder and all of it's content is copied

Scenario 10: File storage - Organization (Copy a file or folder)

- **Role Involved:** User
- **Objective:** Be able to paste a folder or a file
- **Preconditions:** Having already copied a file or folder
- **Test Steps:**
    1. The user selects the file or folder
    2. The user clicks on the paste action
- **Expected Outcome:** The file or the folder and all of it's content is pasted

Scenario 11: File storage - Organization (duplicate a file or folder)

- **Role Involved:** User
- **Objective:** Be able to duplicate a folder or a file
- **Preconditions:** To have at least one file or folder or two folders
- **Test Steps:**
    1. The user selects the file or folder
    2. The user clicks on the duplicate action
- **Expected Outcome:** The file or the folder and all of it's content is duplicated

Scenario 12: File storage - Access

- **Role Involved:** User
- **Objective:** Be able to list their files and use the interface to look for specific files
- **Preconditions:** Having files
- **Test Steps:**
    1. The user clicks on the search bar
    2. The user chooses the corresponding name
- **Expected Outcome:** The file appears in the list

Scenario 13: File storage - Bulk

- **Role Involved:** User
- **Objective:** Be able to use an action on multiple target at once
- **Preconditions:** having multiple files or folders
- **Test Steps:**
    1. The user selects all the files or folders he wants
    2. The user click on the action he wants
- **Expected Outcome:** The action should be executed for all targets

Scenario 14: File storage - Deletion

- **Role Involved:** User
- **Objective:** Be able to delete a file
- **Preconditions:** Having a file
- **Test Steps:**
    1. The user selects a folder to file
    2. The user click on the Delete action
    3. The user goes to the Trash page and sees their file
    4. The user permanently deletes the file
- **Expected Outcome:** The file is deleted from the Aleph Network and not accessible anymore

Scenario 15: File storage - Sharing (private)

- **Role Involved:** User
- **Objective:** Be able to share a file to a contact
- **Preconditions:** Having a contact already created to share the file to
- **Test Steps:**
    1. The user selects the file to share
    2. The User selects the share action
    3. The user choose the contact to share the file
- **Expected Outcome:** The file is shared to and only to the selected contact

Scenario 16: File storage - Sharing (public)

- **Role Involved:** User
- **Objective:** Be able to share or unshare publicly a file or folder
- **Preconditions:** Having a file
- **Test Steps:**
    1. The user selects a file or folder
    2. The user use the share publicly action
    3. The user enter the given url to check if the file is correctly shared
    4. The user unshare the using the unshare action
    5. The user reload the url
- **Expected Outcome:** The URL should be openly accessible (validating by opening in a private browser window) after the third step but inaccessible at the end

Scenario 17: User interaction - Contacts

- **Role Involved:** User
- **Objective:** Be able to retrieve and register the users he interacted with as contacts
- **Preconditions:** None
- **Test Steps:**
    1. The user goes to the contact page and sees the already existing contacts
    2. The user selects the add contact action
    3. The user selects the username of the account he want to register as contact
- **Expected Outcome:** The contact is well created and appears in the contact section, the shared files will be visible after the other user added the user as contact

Scenario 18: Artificial Intelligence - Chat

- **Role Involved:** User
- **Objective:** Be able to interact with an AI model directly in Bedrock
- **Preconditions:** None
- **Test Steps:**
    1. The user goes to the chat page and sees a text area to start a new conversation, as well as a button to change the AI model among available ones from LibertAI's API.
    2. The user asks something to the AI that will process it, and have a full discussion
- **Expected Outcome:** The user is able to ask AI assistance through Bedrock and is able to leave the page and retrieve later their previous conversations

Scenario 19: Artificial Intelligence - Context

- **Role Involved:** User
- **Objective:** Be able to give any files uploaded as a context for the AI conversations
- **Preconditions:** To have uploaded at least 1 file
- **Test Steps:**
    1. The user goes to the Knowledge base page and creates a new Knowledge base.
    2. The user then adds their uploaded file to their newly created Knowledge base
    3. The user goes to the Chat page, selects their AI model as well as the Knowledge base, and starts chatting with the AI
- **Expected Outcome:** The user is able to ask AI assistance about the file contents and get responses that take into account the file(s) present in their Knowledge base

Scenario 20: Balance - Top up

- **Role Involved:** User
- **Objective:** The user has a credit balance on Bedrock and is able to add money to it
- **Preconditions:** None
- **Test Steps:**
    1. The user goes to the credits page on his profile
    2. The user clicks the "Top up" button
    3. The user chooses his favorite payment method between credit card or crypto (payment possible on several EVM chains with several assets such as ETH, USDC, USDT...)

4. The user follows the flow of the payment modal to enter his credit card information / approve the payment transaction on his crypto wallet
5. A success page and toast are displayed after the payment

- **Expected Outcome:** The balance reflects a new amount that includes the top up that the user just did

---

3. Success Criteria

Any person that already uses an online drive app to store documents & interact with them (Google Drive, Office 365) won't be lost when interacting with Bedrock & will be able to login, manage documents and use the other Bedrock features just as they would with their classic app. Advanced / tech-savy users will be able to understand & verify that their data is securely stored & accessed via different components of Aleph's decentralized cloud.

---

4. Known Issues & Limitations

Issue	Description	Impact	Planned Fix? (Yes/No)
Aleph Network outages	The Aleph network can be unstable, especially when new updates are deployed.	High	No (independent from us)
Aleph sharing permissions	Sharing access to files / messages on Aleph can be restricted via several filters, but these don't include specific message hashes. This means that when sharing a file with editing permissions to someone, he will technically have permissions to edit the whole drive of the user. This is mitigated by the fact that the Bedrock UI limits you to the files really shared, and it can be restricted as soon as Aleph adds this hash filtering capability.	Medium	No (not planned for Aleph this year)

---

5. Conclusion

Bedrock's Beta Test Plan is the first step of ensuring that our project can become a fully usable on-chain workspace solution, with the latest AI advancements available to enhance productivity, making it a private workspace by design, not by promise.