Document management

Aleksandra Maksimowska (11031917)

Michał Jagoda (11031910)

Group 156

Pure HTML version

Document management (Pure HTML version) 1/2

The application supports user registration and login via a public page with appropriate forms. Registration requires entering a username, email address, and password, and checks the syntactic validity of the email address and the equality between the "password" and "repeat password" fields, also on the client side. The registration ensures the uniqueness of the username.

A folder has an owner, a name, and a creation date, and can contain other folders and/or documents. A document has an owner, a name, a creation date, a summary, and a type. When the user logs into the application, a HOME PAGE appears containing a tree of their folders and subfolders.

Example of a folder tree on the HOME PAGE

- Folder 1
  - Folder 11
    - Folder 111
  - Folder 12
  - Folder 13
- Folder 2
  - Folder 21
- Folder 3
  - Folder 31
- Folder 111

Contents of the CONTENT PAGE of a folder

- Document 1 > access > move
- Document 2 > access > move
- Document 3 > access > move

Document management (Pure HTML version) 2/2

On the HOME PAGE, the user can select a folder and access a CONTENT PAGE which shows a list of folders and documents in a folder. Each document in the list has two links: access and move. When the user selects the access link, a DOCUMENT PAGE (in the same window and browser tab) appears, showing all the data of the selected document. When the user selects the move link, the HOME PAGE with the folder tree appears; in this case, the page shows the message "You are moving document X from folder Y. Choose the destination folder", the folder to which the document to be moved belongs is NOT selectable and its name is highlighted (for example, with a different color). When the user selects the destination folder, the document is moved from the source folder to the destination folder, and the CONTENT PAGE appears, showing the updated content of the destination folder. Every page, except the HOME PAGE, contains a link to return to the previous page. The application allows the user to log out from any page. A CONTENT MANAGEMENT PAGE accessible from the HOME PAGE allows the user to create a top-level folder, a folder within an existing folder, and a document within a folder. The application does not require the management of document uploads.

Data Analysis (Pure HTML version)

The application supports user registration and login via a public page with appropriate forms. Registration requires entering a username, email address, and password, and checks the syntactic validity of the email address and the equality between the "password" and "repeat password" fields, also on the client side. The registration ensures the uniqueness of the username.

A folder has an owner, a name, and a creation date, and can contain other folders and/or documents. A document has an owner, a name, a creation date, a summary, and a type. When the user logs into the application, a HOME PAGE appears containing a tree of their folders and subfolders.

Entity Relation Attribute

Example of a folder tree on the HOME PAGE

- Folder 1
  - Folder 11
    - Folder 111
  - Folder 12
  - Folder 13
- Folder 2
  - Folder 21
- Folder 3
  - Folder 31
- Folder 111

Contents of the CONTENT PAGE of a folder

- Document 1 > access > move
- Document 2 > access > move
- Document 3 > access > move

Data Analysis (Pure HTML version) 2/2

On the HOME PAGE, the user can select a folder and access a CONTENT PAGE which shows a list of folders and documents in a folder. Each document in the list has two links: access and move. When the user selects the access link, a DOCUMENT PAGE (in the same window and browser tab) appears, showing all the data of the selected document. When the user selects the move link, the HOME PAGE with the folder tree appears; in this case, the page shows the message "You are moving document X from folder Y. Choose the destination folder", the folder to which the document to be moved belongs is NOT selectable and its name is highlighted (for example, with a different color). When the user selects the destination folder, the document is moved from the source folder to the destination folder, and the CONTENT PAGE appears, showing the updated content of the destination folder. Every page, except the HOME PAGE, contains a link to return to the previous page. The application allows the user to log out from any page. A CONTENT MANAGEMENT PAGE accessible from the HOME PAGE allows the user to create a top-level folder, a folder within an existing folder, and a document within a folder. The application does not require the management of document uploads.

Entity Relation Attribute

Requirements analysis (Pure HTML version) 1/2

The application supports user registration and login via a public page with appropriate forms. Registration requires entering a username, email address, and password, and checks the syntactic validity of the email address and the equality between the "password" and "repeat password" fields, also on the client side. The registration ensures the uniqueness of the username.

A folder has an owner, a name, and a creation date, and can contain other folders and/or documents. A document has an owner, a name, a creation date, a summary, and a type. When the user logs into the application, a HOME PAGE appears containing a tree of their folders and subfolders.

Example of a folder tree on the HOME PAGE

- Folder 1
  - Folder 11
    - Folder 111
  - Folder 12
  - Folder 13
- Folder 2
  - Folder 21
- Folder 3
  - Folder 31
- Folder 111

Contents of the CONTENT PAGE of a folder

- Document 1 > access > move
- Document 2 > access > move
- Document 3 > access > move

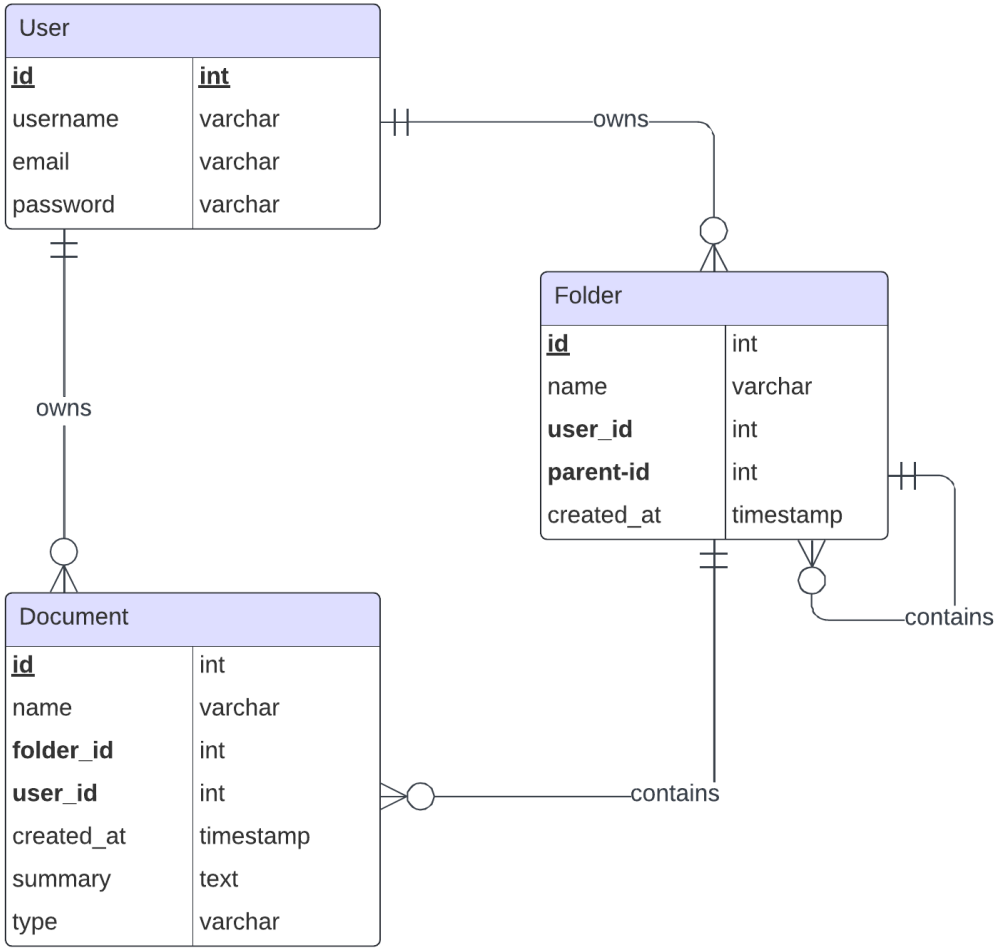
Pages View components Events Actions

Requirements analysis (Pure HTML version) 2/2

On the HOME PAGE, the user can select a folder and access a CONTENT PAGE which shows a list of folders and documents in a folder. Each document in the list has two links: access and move. When the user selects the access link, a DOCUMENT PAGE (in the same window and browser tab) appears, showing all the data of the selected document. When the user selects the move link, the HOME PAGE with the folder tree appears; in this case, the page shows the message "You are moving document X from folder Y. Choose the destination folder", the folder to which the document to be moved belongs is NOT selectable and its name is highlighted (for example, with a different color). When the user selects the destination folder, the document is moved from the source folder to the destination folder, and the CONTENT PAGE appears, showing the updated content of the destination folder. Every page, except the HOME PAGE, contains a link to return to the previous page. The application allows the user to log out from any page. A CONTENT MANAGEMENT PAGE accessible from the HOME PAGE allows the user to create a top-level folder, a folder within an existing folder, and a document within a folder. The application does not require the management of document uploads.

Pages View components Events Actions

Database ERD



Database schema (users)

```
CREATE TABLE `users` (
        `id` int NOT NULL AUTO_INCREMENT,
        `username` varchar(45) NOT NULL,
        `email` varchar(45) NOT NULL,
        `password` varchar(45) NOT NULL,
        PRIMARY KEY (`id`),
        UNIQUE KEY `email_UNIQUE` (`email`),
        UNIQUE KEY `username_UNIQUE` (`username`),
        UNIQUE KEY `id_UNIQUE` (`id`)
        ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Database schema (folders)

```
CREATE TABLE `folders` (
        `id` int NOT NULL AUTO_INCREMENT,
        `name` varchar(255) NOT NULL,
        `user_id` int NOT NULL,
        `parent_id` int DEFAULT NULL,
        `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
        PRIMARY KEY (`id`),
        KEY `fk_folders_users` (`user_id`),
        KEY `fk_folders_parent` (`parent_id`),
        CONSTRAINT `fk_folders_parent` FOREIGN KEY (`parent_id`) REFERENCES `folders` (`id`),
        CONSTRAINT `fk_folders_users` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
        ) ENGINE=InnoDB AUTO_INCREMENT=52 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Database schema (documents)

```
CREATE TABLE `documents` (
        `id` int NOT NULL AUTO_INCREMENT,
        `name` varchar(255) NOT NULL,
        `folder_id` int DEFAULT NULL,
        `user_id` int NOT NULL,
        `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
        `summary` text, `type` varchar(255) DEFAULT NULL,
        PRIMARY KEY (`id`), KEY `folder_id` (`folder_id`),
        KEY `fk_documents_users` (`user_id`),
        CONSTRAINT `documents_ibfk_1` FOREIGN KEY (`folder_id`) REFERENCES `folders` (`id`) ON DELETE CASCADE,
        CONSTRAINT `fk_documents_users` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
        ) ENGINE=InnoDB AUTO_INCREMENT=29 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

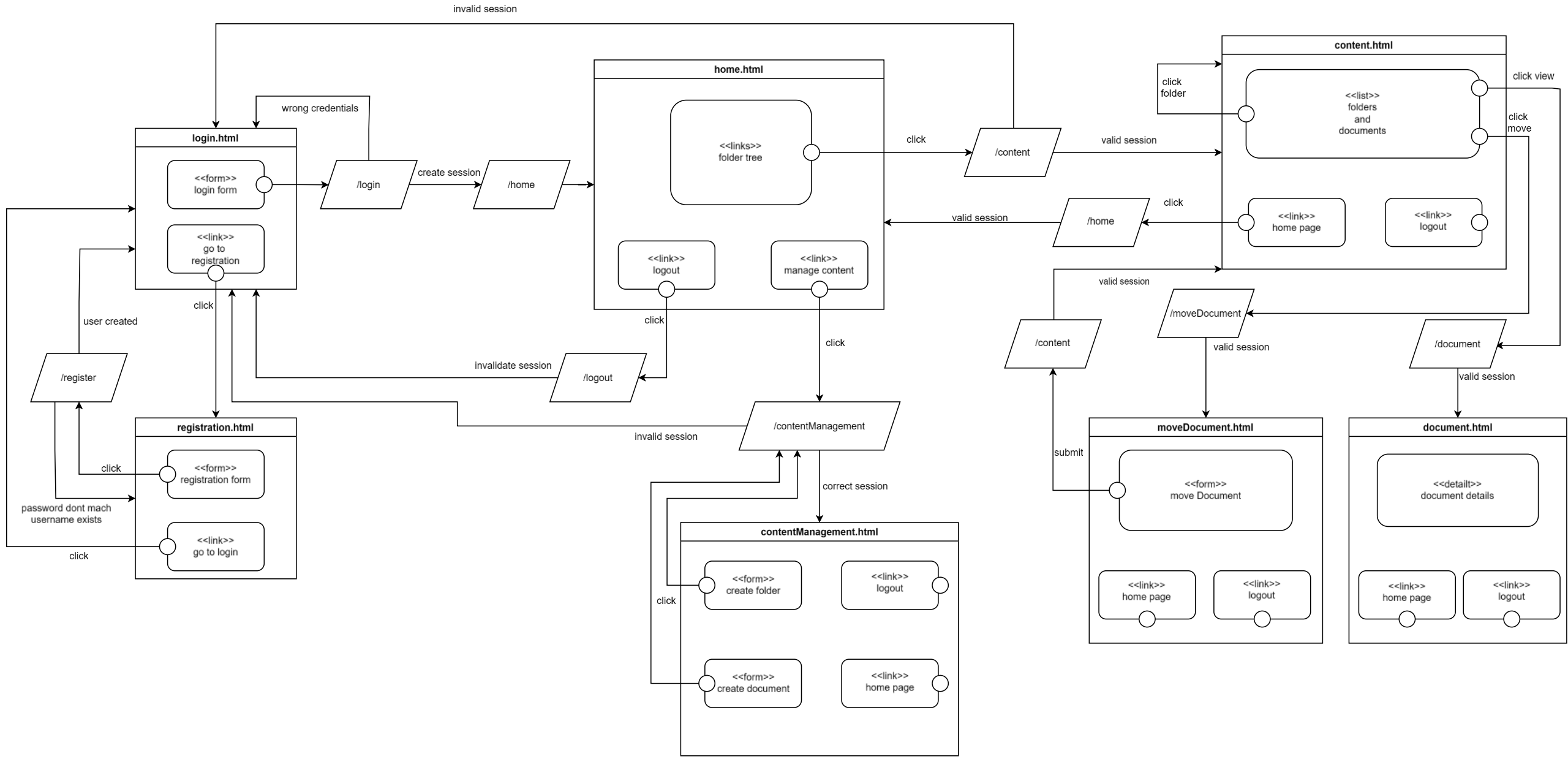
Components

- Model objects (Beans)
  - User
  - Document
  - Folder
- Views
  - login.html
  - register.html
  - home.html
- Filters
  - AuthenticationFilter
  - CsrfFilter
  - LoggingFilter
- Controllers (Servlets)
  - ContentManagementServlet
  - ContentServlet
  - DocumetnServlet
  - FolderServlet
  - HomeServlet
  - LoginServlet
  - LogoutServlet
  - MoveDocumentServlet
  - RegisterServlet
  - ViewDocumentServlet
- Utils
  - DatabaseUtils
  - SecurityUtils
  - ThymeleafConfig

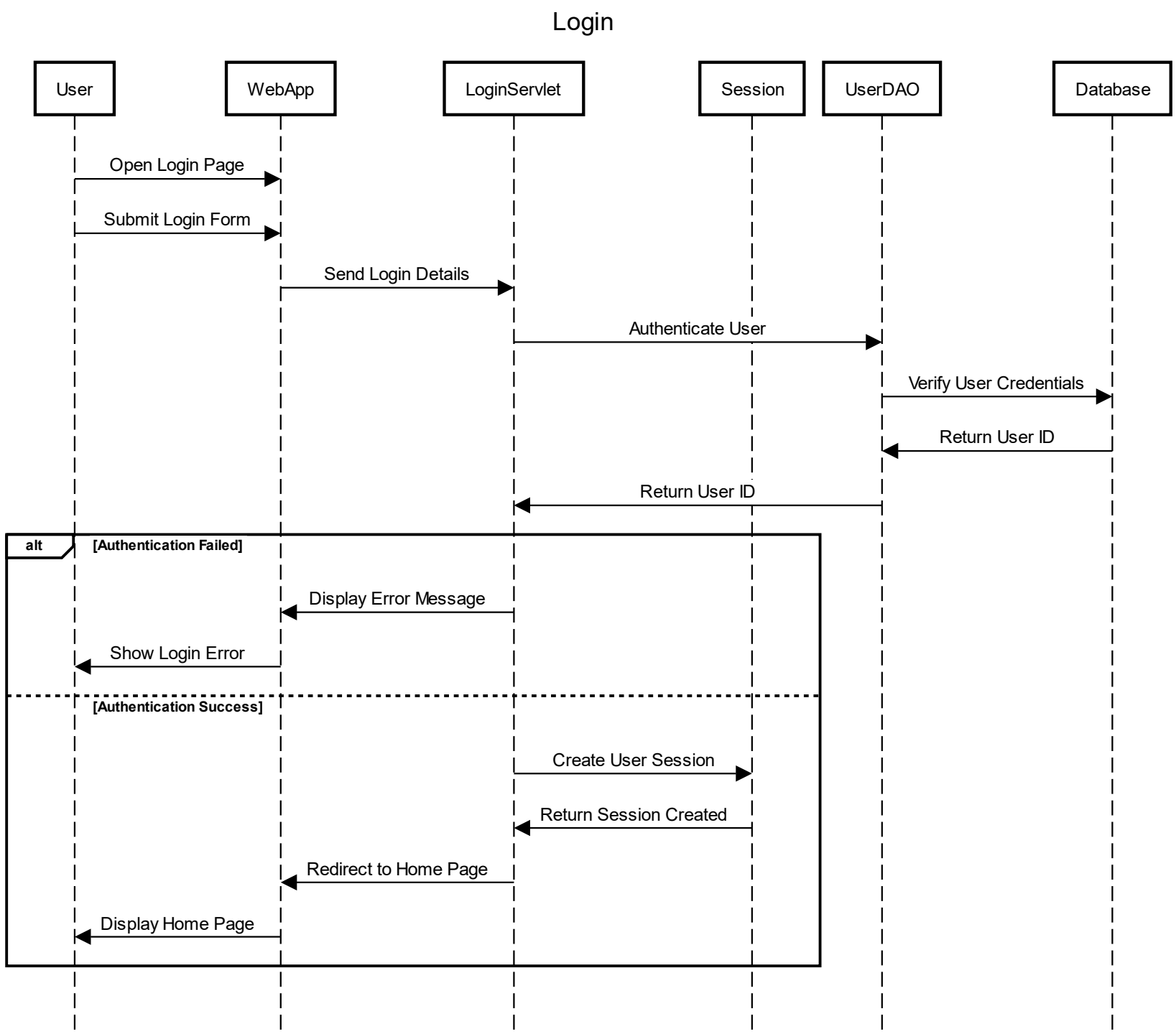
Components

- Views (templates):
  - contentManagement.html
  - content.html
  - document.html
  - fragments.html
  - home.html
  - login.html
  - moveDocument.html
  - register.html
  - viewDocument.html

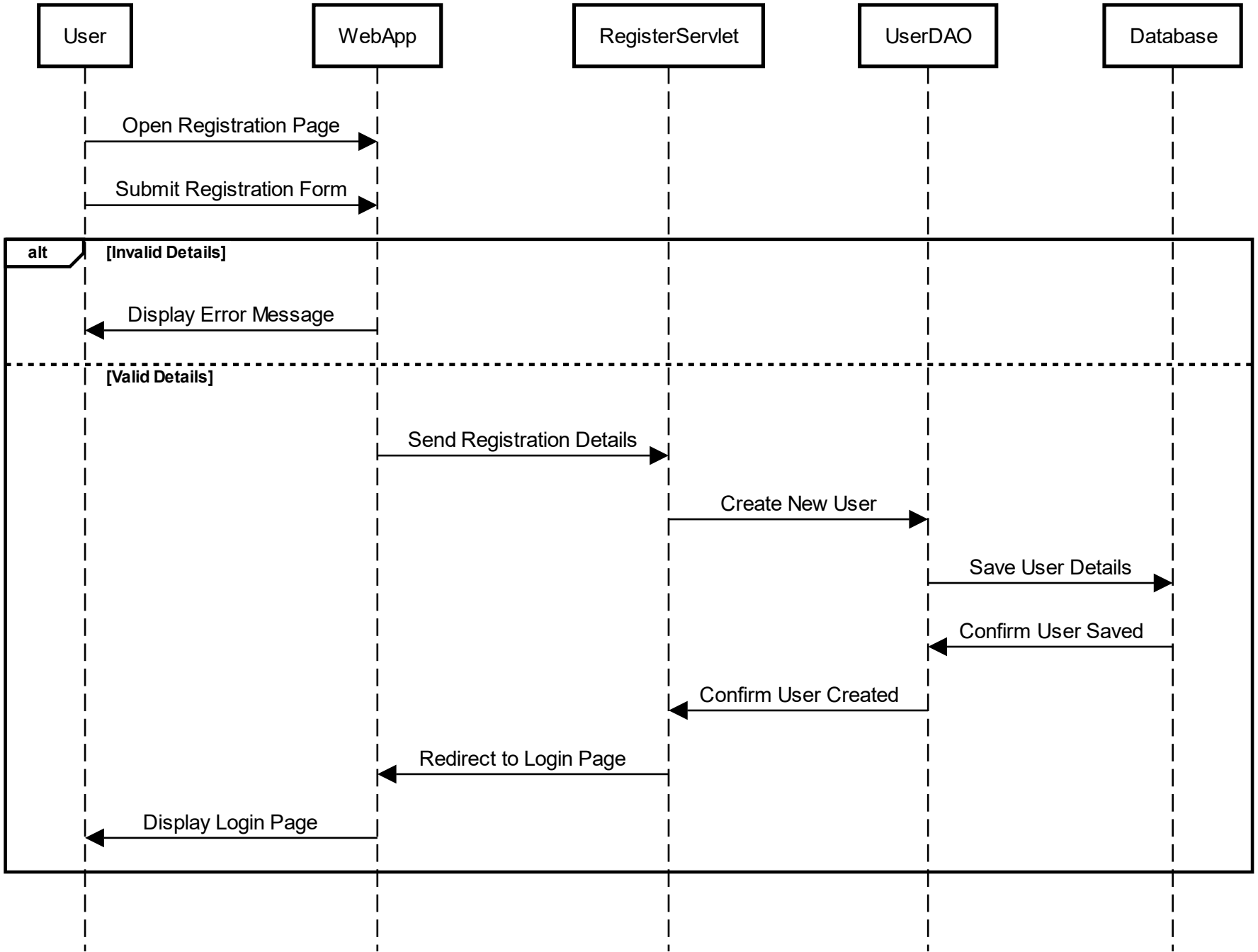
IFML diagram (pure HTML version)


invalid session
login.html
<<form>>
login form
<<link>>
go to
registration
wrong credentials
/login
create session
/home
home.html
<<links>>
folder tree
<<link>>
logout
<<link>>
manage content
click
/content
valid session
content.html
click
folder
<<list>>
folders
and
documents
click view
click
move
<<link>>
home page
<<link>>
logout
valid session
/home
click
click
user created
/register
invalidate session
/logout
click
click
valid session
/content
/moveDocument
/document
valid session
valid session
registration.html
<<form>>
registration form
<<link>>
go to login
click
password dont mach
username exists
click
/contentManagement
invalid session
correct session
contentManagement.html
<<form>>
create folder
<<link>>
logout
<<form>>
create document
<<link>>
home page
click
submit
moveDocument.html
<<form>>
move Document
<<link>>
home page
<<link>>
logout
document.html
<<detailt>>
document details
<<link>>
home page
<<link>>
logout

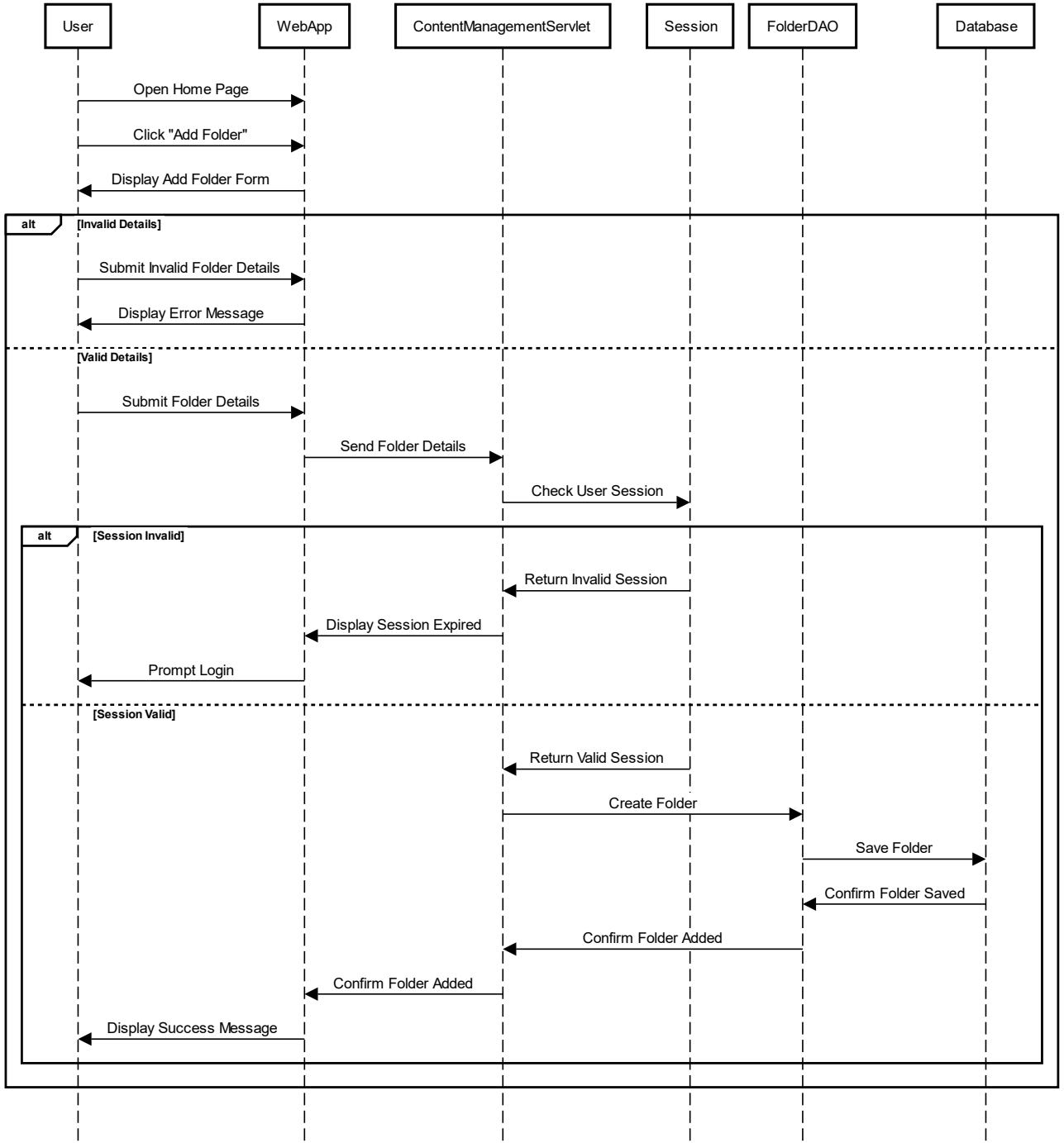
Sequence diagrams (pure HTML version)


Login
User
WebApp
LoginServlet
Session
UserDAO
Database
Open Login Page
Submit Login Form
Send Login Details
Authenticate User
Verify User Credentials
Return User ID
Return User ID
alt
[Authentication Failed]
Display Error Message
Show Login Error
[Authentication Success]
Create User Session
Return Session Created
Redirect to Home Page
Display Home Page

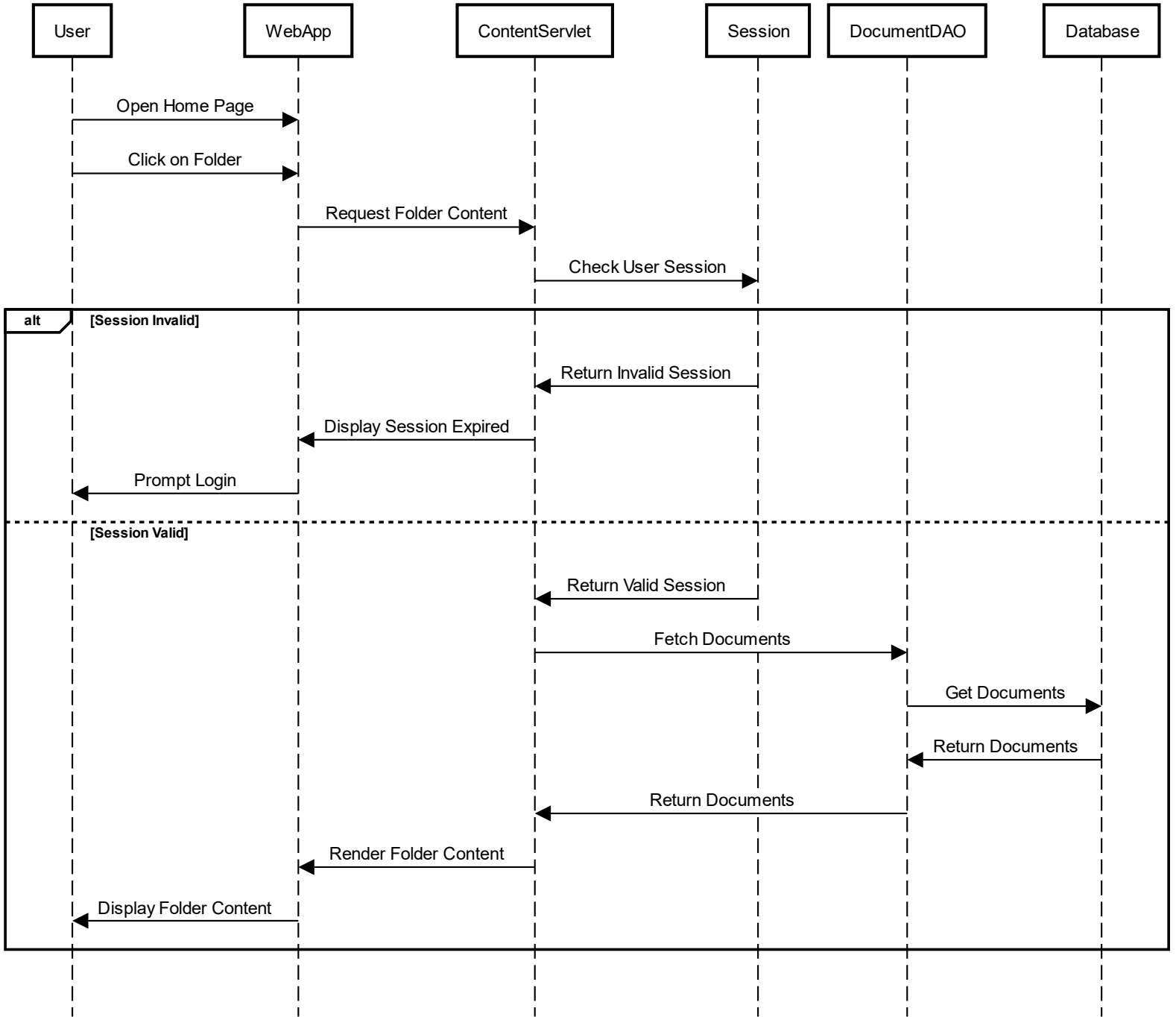
Registration



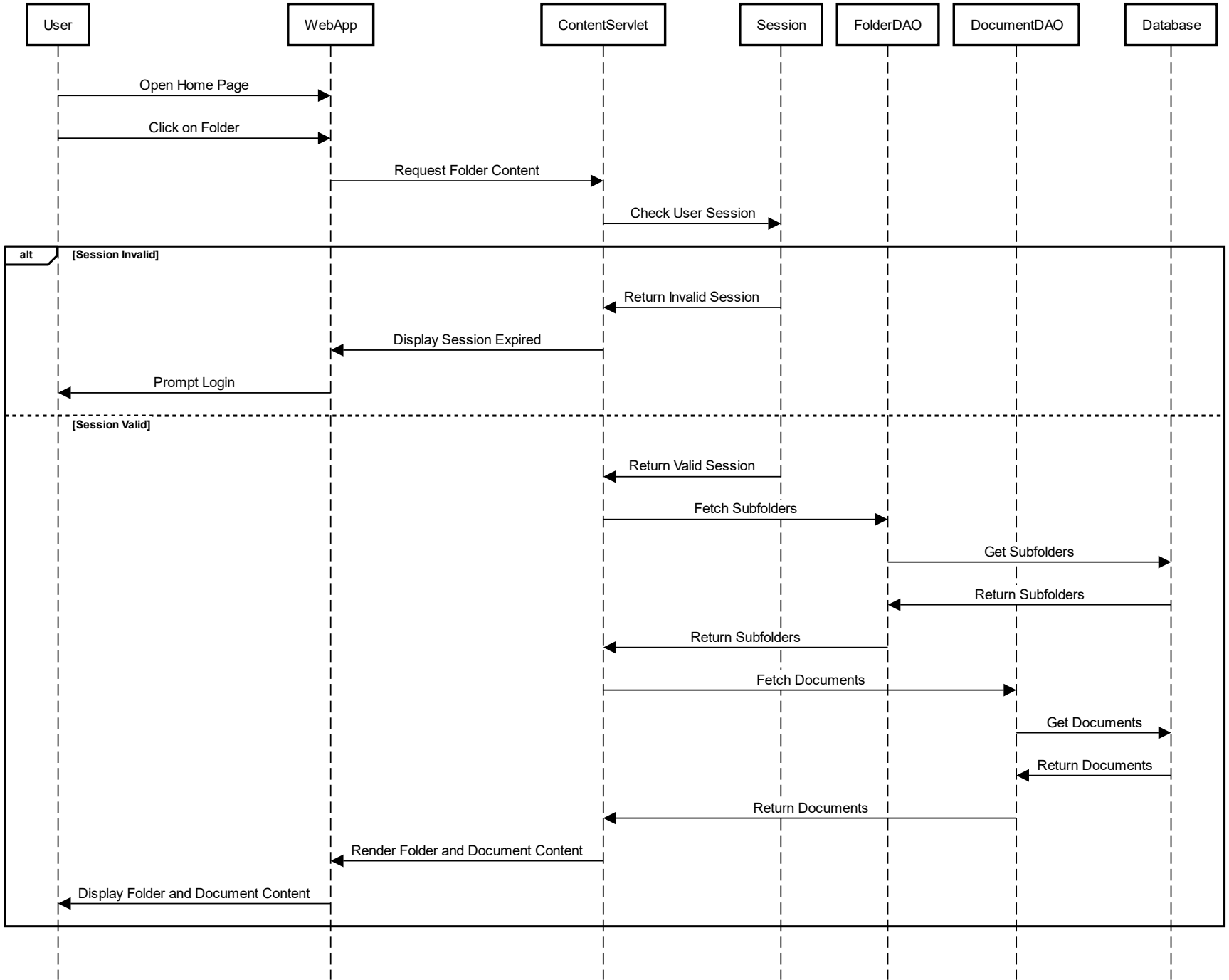
Add Folder


User
WebApp
ContentManagementServlet
Session
FolderDAO
Database
Open Home Page
Click "Add Folder"
Display Add Folder Form
alt
[Invalid Details]
Submit Invalid Folder Details
Display Error Message
[Valid Details]
Submit Folder Details
Send Folder Details
Check User Session
alt
[Session Invalid]
Return Invalid Session
Display Session Expired
Prompt Login
[Session Valid]
Return Valid Session
Create Folder
Save Folder
Confirm Folder Saved
Confirm Folder Added
Confirm Folder Added
Display Success Message

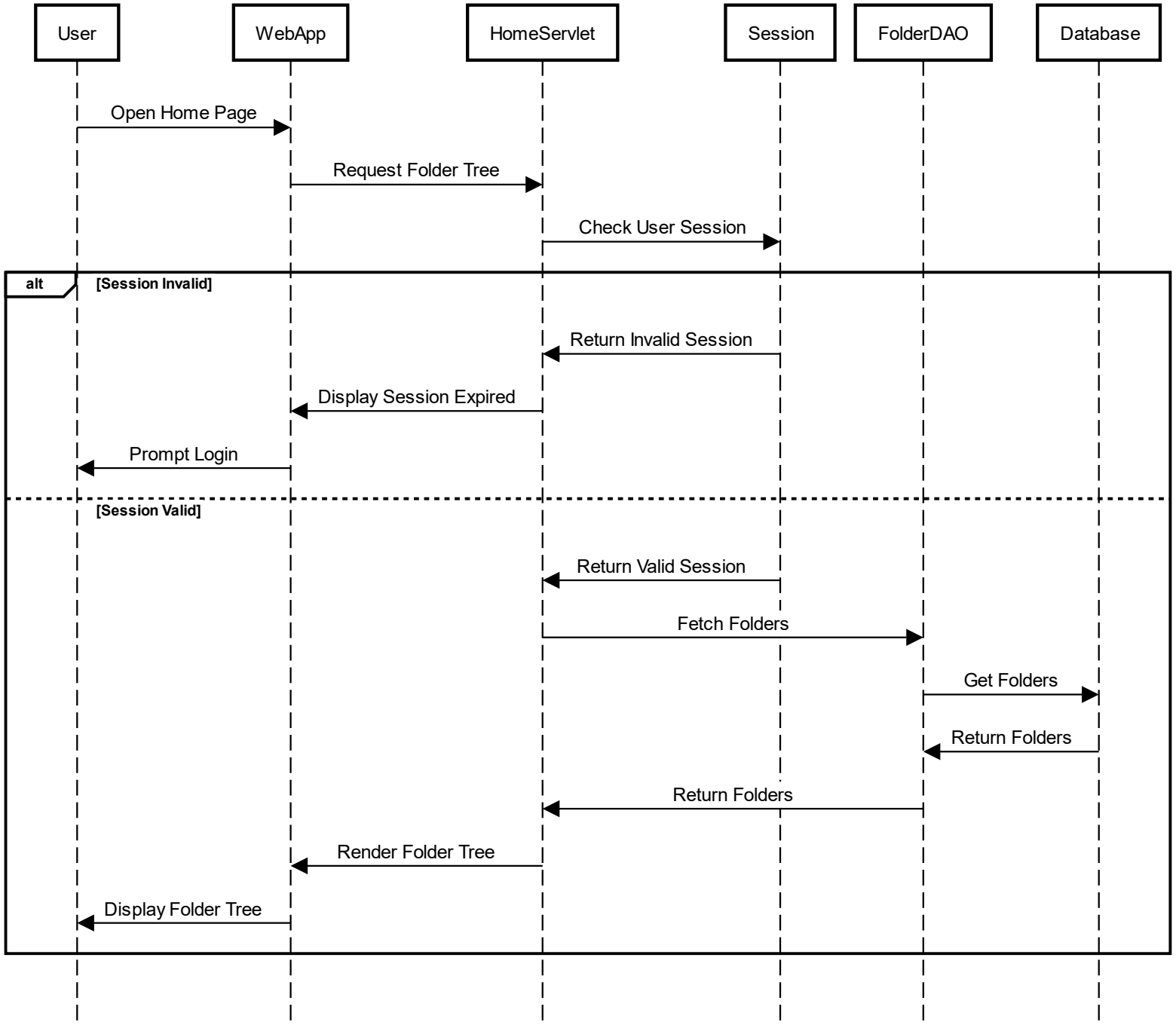
Load Documents



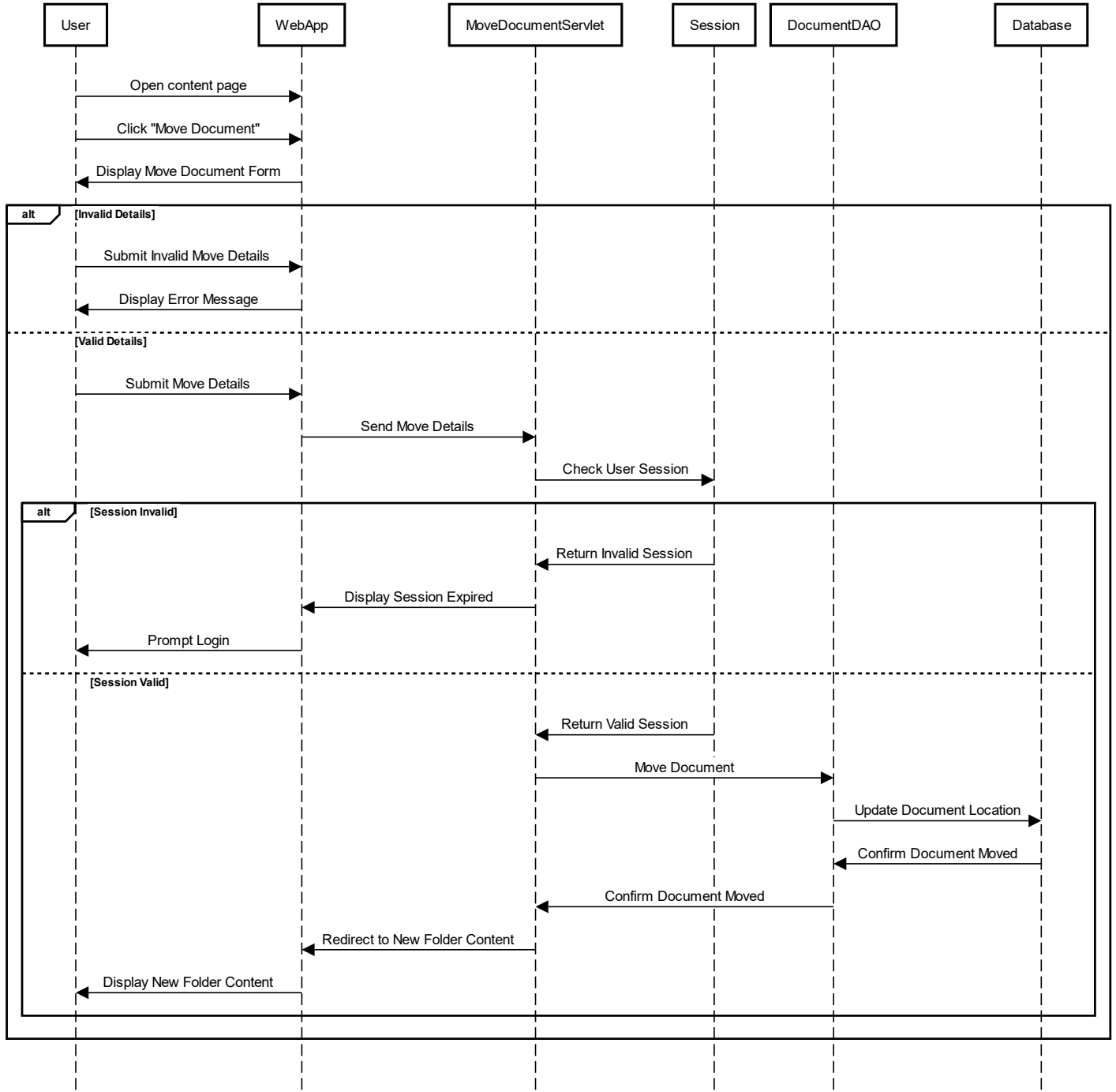
Load Folders and Documents



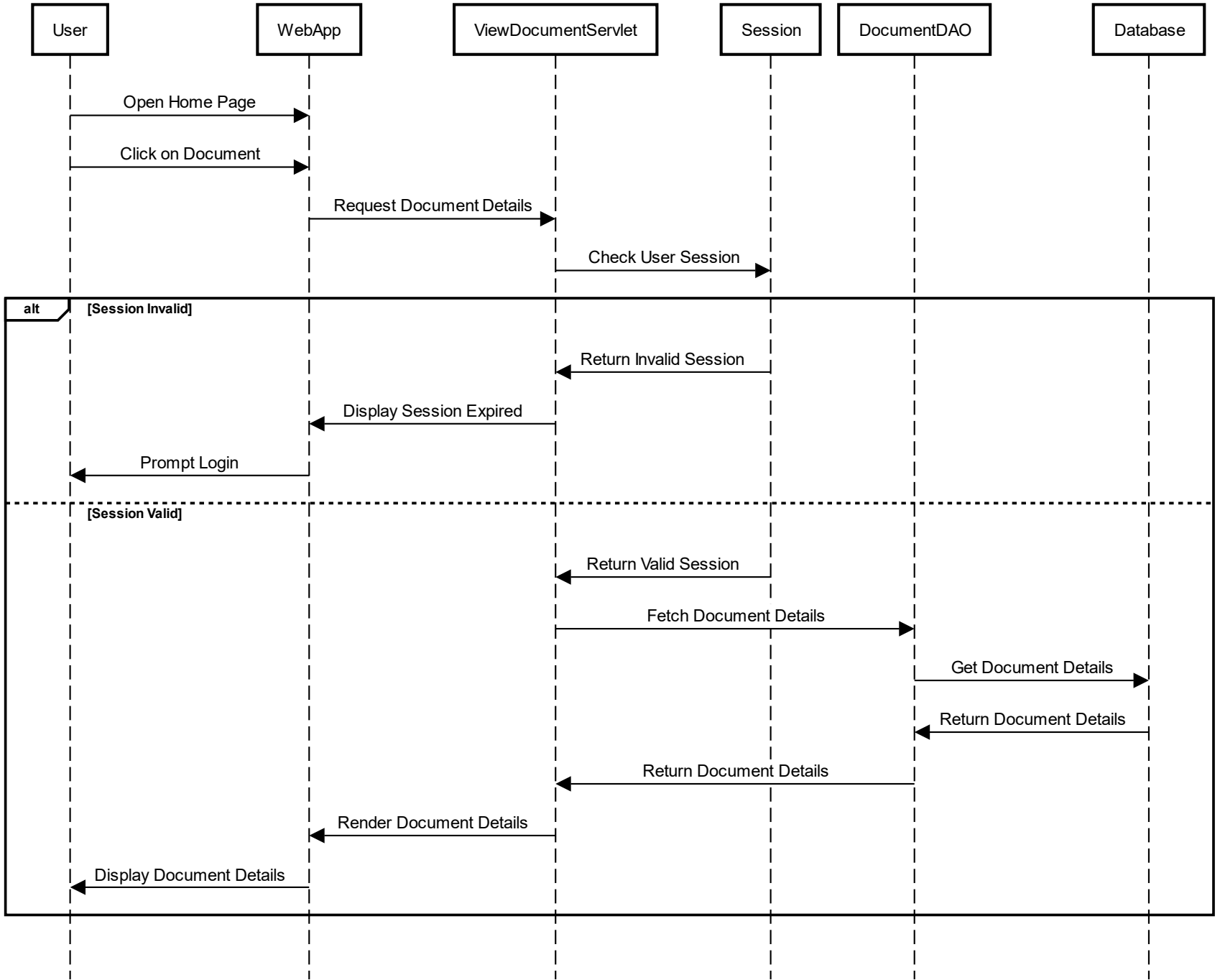
Load Folders

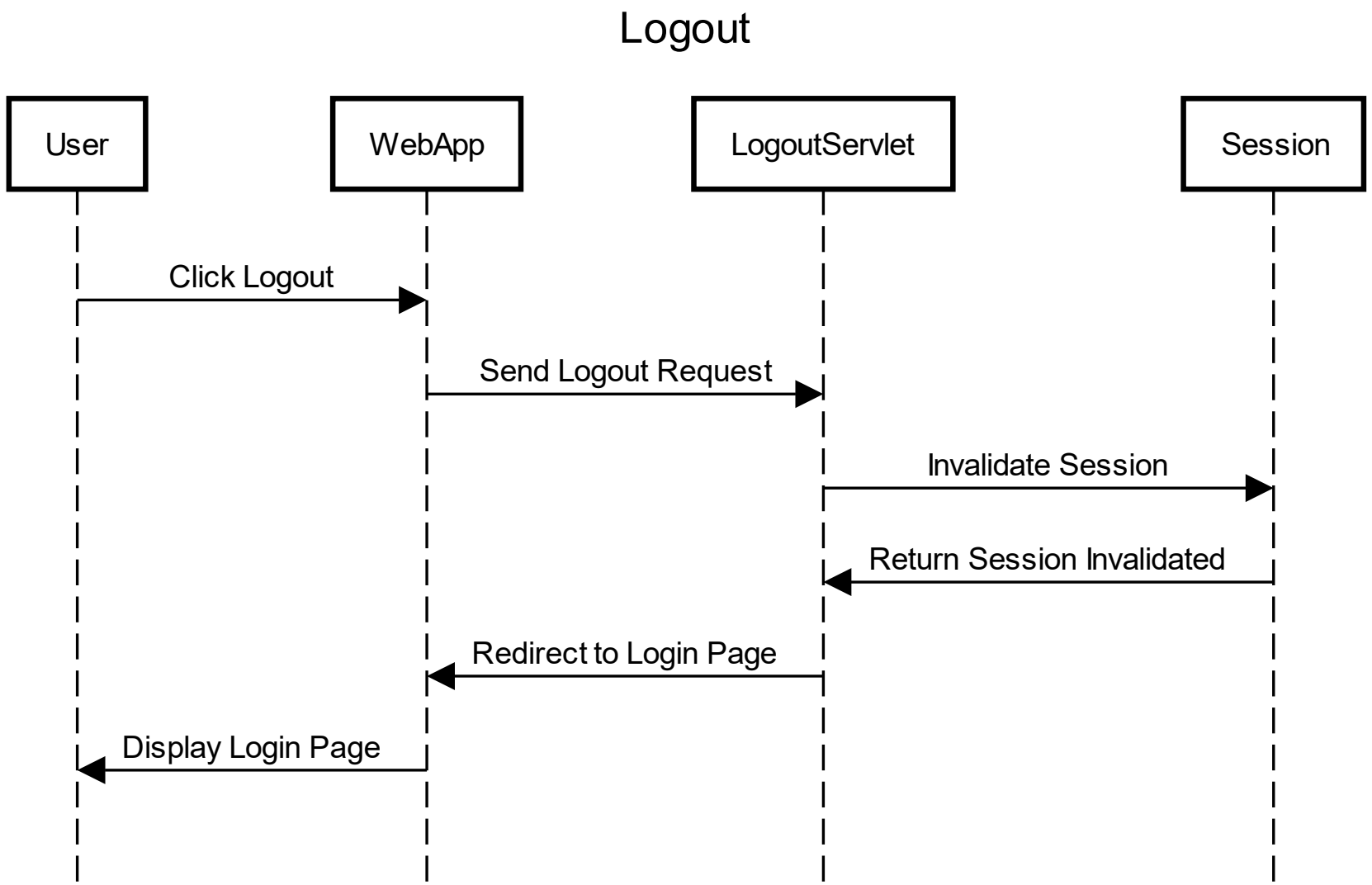


Move Document



View Document




Logout
User
WebApp
LogoutServlet
Session
Click Logout
Send Logout Request
Invalidate Session
Return Session Invalidated
Redirect to Login Page
Display Login Page

JavaScript version

Document management (JavaScript version)

- Implement a client-server web application that modifies the previous specifications as follows:
- The application supports registration and login via a public page with appropriate forms. Registration checks the syntactic validity of the email address and the equality between the "password" and "repeat password" fields, also on the client side. Registration ensures the uniqueness of the username.
- After user login, the entire application is realized with a single page.
- Every user interaction is handled without fully reloading the page, but produces an asynchronous invocation of the server and potentially modifies the content to be updated following the event.
- Server-side errors must be reported through an alert message inside the page.
- The document moving function is implemented via drag and drop.
- The function of creating a subfolder is implemented on the HOME PAGE with an ADD SUBFOLDER button placed next to each folder. Pressing the button brings up an input field to enter the name of the folder to be inserted.
- The function of creating a document is implemented on the HOME PAGE with an ADD DOCUMENT button placed next to each folder. Pressing the button brings up an input form to enter the document data.
- A folder called "trash" is added. Dragging and dropping a document or folder into the trash results in deletion. Before sending the delete command to the server, the user sees a confirmation modal window and can decide whether to cancel the operation or proceed. Deleting a folder results in the complete and recursive deletion of the contents from the database (documents and folders).

Data Analysis (JavaScript version)

- Implement a client-server web application that modifies the previous specifications as follows:
- The application supports registration and login via a public page with appropriate forms. Registration checks the syntactic validity of the email address and the equality between the "password" and "repeat password" fields, also on the client side. Registration ensures the uniqueness of the username.
- After user login, the entire application is realized with a single page.
- Every user interaction is handled without fully reloading the page, but produces an asynchronous invocation of the server and potentially modifies the content to be updated following the event.
- Server-side errors must be reported through an alert message inside the page.
- The document moving function is implemented via drag and drop.
- The function of creating a subfolder is implemented on the HOME PAGE with an ADD SUBFOLDER button placed next to each folder. Pressing the button brings up an input field to enter the name of the folder to be inserted.
- The function of creating a document is implemented on the HOME PAGE with an ADD DOCUMENT button placed next to each folder. Pressing the button brings up an input form to enter the document data.
- A folder called "trash" is added. Dragging and dropping a document or folder into the trash results in deletion. Before sending the delete command to the server, the user sees a confirmation modal window and can decide whether to cancel the operation or proceed. Deleting a folder results in the complete and recursive deletion of the contents from the database (documents and folders).

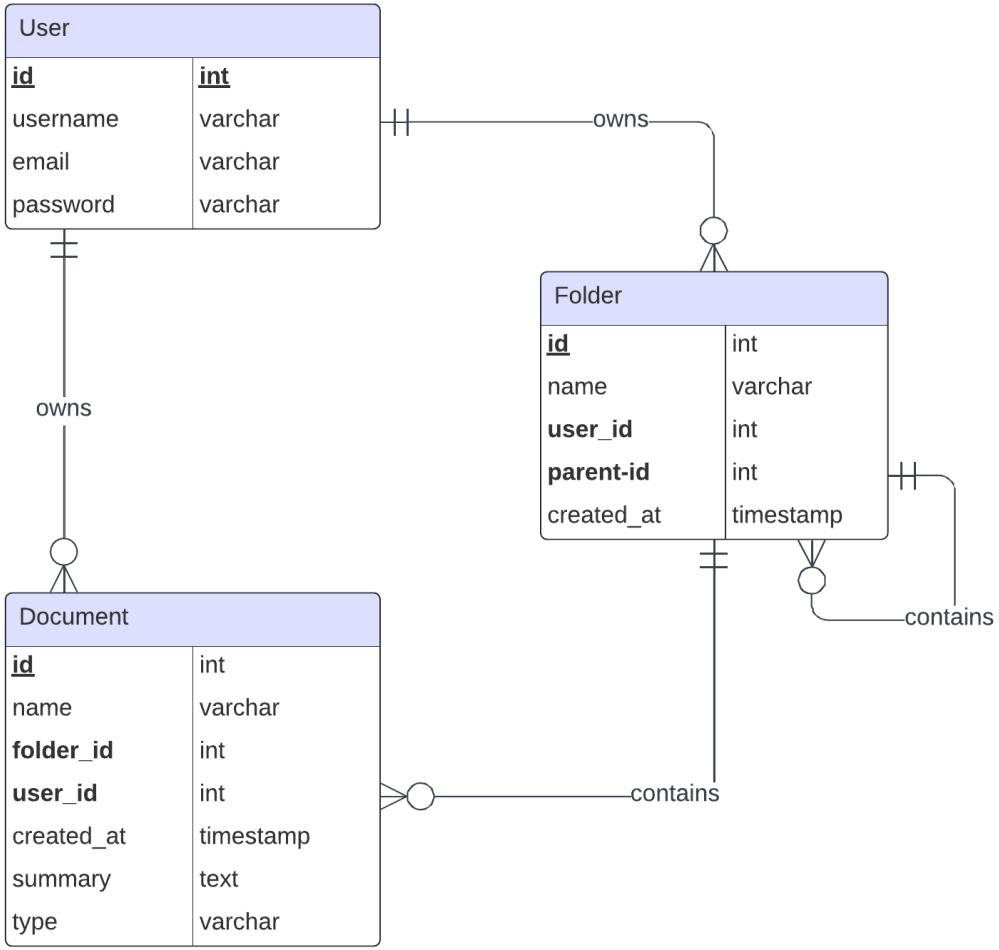
Entity Relation Attribute

Requirements analysis (JavaScript version)

- Implement a client-server web application that modifies the previous specifications as follows:
- The application supports registration and login via a public page with appropriate forms. Registration checks the syntactic validity of the email address and the equality between the "password" and "repeat password" fields, also on the client side. Registration ensures the uniqueness of the username.
- After user login, the entire application is realized with a single page.
- Every user interaction is handled without fully reloading the page, but produces an asynchronous invocation of the server and potentially modifies the content to be updated following the event.
- Server-side errors must be reported through an alert message inside the page.
- The document moving function is implemented via drag and drop.
- The function of creating a subfolder is implemented on the HOME PAGE with an ADD SUBFOLDER button placed next to each folder. Pressing the button brings up an input field to enter the name of the folder to be inserted.
- The function of creating a document is implemented on the HOME PAGE with an ADD DOCUMENT button placed next to each folder. Pressing the button brings up an input form to enter the document data.
- A folder called "trash" is added. Dragging and dropping a document or folder into the trash results in deletion. Before sending the delete command to the server, the user sees a confirmation modal window and can decide whether to cancel the operation or proceed. Deleting a folder results in the complete and recursive deletion of the contents from the database (documents and folders).

Pages View components Events Actions

Database ERD



Database schema (users)

```
CREATE TABLE `users` (
        `id` int NOT NULL AUTO_INCREMENT,
        `username` varchar(45) NOT NULL,
        `email` varchar(45) NOT NULL,
        `password` varchar(45) NOT NULL,
        PRIMARY KEY (`id`),
        UNIQUE KEY `email_UNIQUE` (`email`),
        UNIQUE KEY `username_UNIQUE` (`username`),
        UNIQUE KEY `id_UNIQUE` (`id`)
        ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Database schema (folders)

```
CREATE TABLE `folders` (
        `id` int NOT NULL AUTO_INCREMENT,
        `name` varchar(255) NOT NULL,
        `user_id` int NOT NULL,
        `parent_id` int DEFAULT NULL,
        `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
        PRIMARY KEY (`id`),
        KEY `fk_folders_users` (`user_id`),
        KEY `fk_folders_parent` (`parent_id`),
        CONSTRAINT `fk_folders_parent` FOREIGN KEY (`parent_id`) REFERENCES `folders` (`id`),
        CONSTRAINT `fk_folders_users` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
        ) ENGINE=InnoDB AUTO_INCREMENT=52 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Database schema (documents)

```
CREATE TABLE `documents` (
        `id` int NOT NULL AUTO_INCREMENT,
        `name` varchar(255) NOT NULL,
        `folder_id` int DEFAULT NULL,
        `user_id` int NOT NULL,
        `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
        `summary` text, `type` varchar(255) DEFAULT NULL,
        PRIMARY KEY (`id`), KEY `folder_id` (`folder_id`),
        KEY `fk_documents_users` (`user_id`),
        CONSTRAINT `documents_ibfk_1` FOREIGN KEY (`folder_id`) REFERENCES `folders` (`id`) ON DELETE CASCADE,
        CONSTRAINT `fk_documents_users` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
        ) ENGINE=InnoDB AUTO_INCREMENT=29 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Components

- Model objects (Beans)
  - User
  - Document
  - Folder
- Views
  - login.html
  - register.html
  - home.html
- Filters
  - AuthenticationFilter
- Controllers (Servlets)
  - UserRegisterServlet
  - LoginServlet
  - LogoutServlet
  - FolderServlet
  - DocumentServlet
- Utils
  - DatabaseUtils

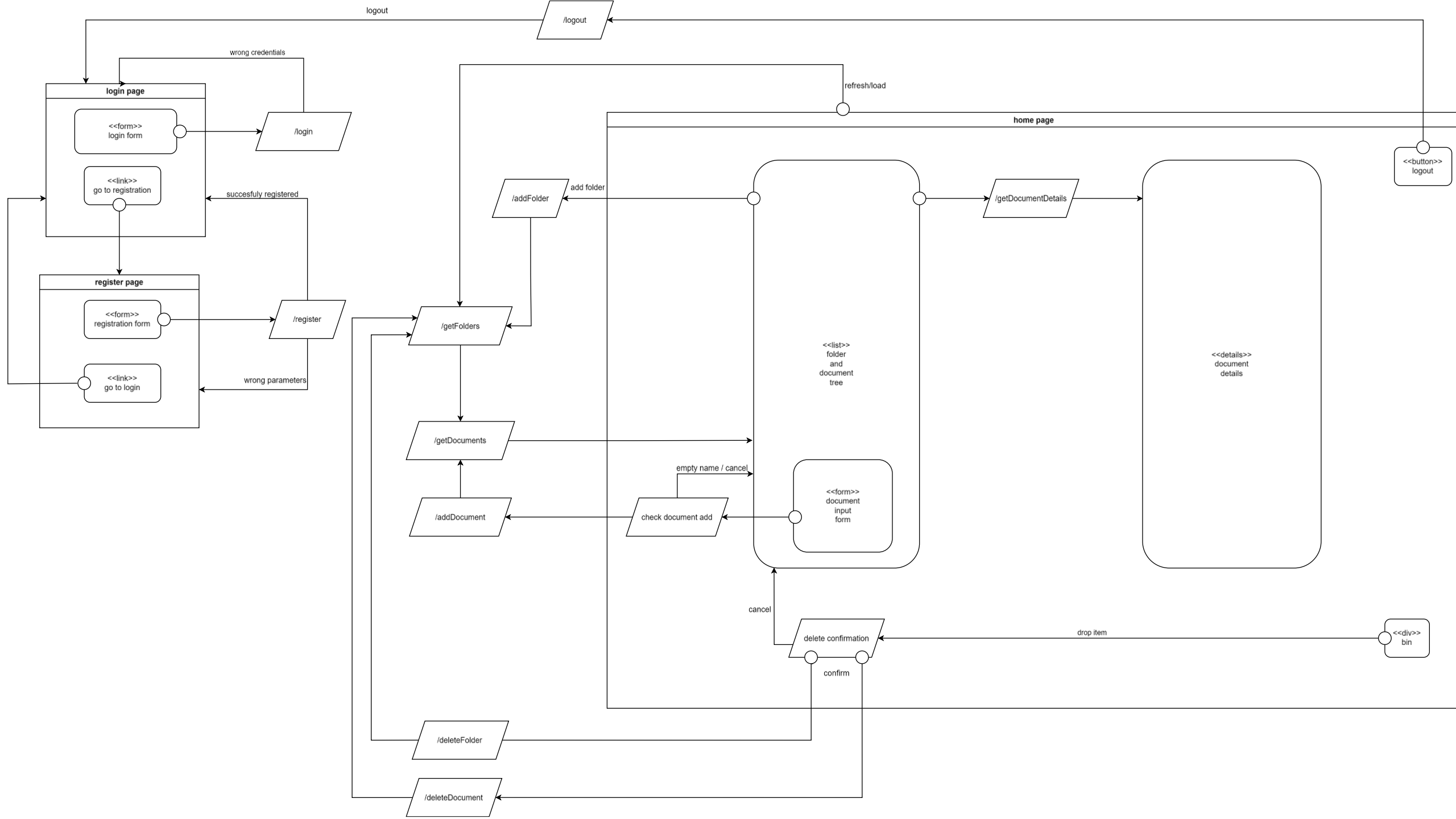
Compnents (DAO)

- UserDAO
  - doesUsernameExist(username)
  - doesEmailExist(email)
  - registerUser(User user)
  - authenticateUser(username, password)
- DocumentDAO
  - getDocumentsByFolder(folderId)
  - getDocumentById(documentId)
  - moveDocument(documentId, newFolderId)
  - addDocument(document)
  - deleteDocument(documentId)
- FolderDAO
  - getAllFolders(userId)
  - addFolder(folder)
  - deleteFolderRecursively(folderId)
  - getSubFolderIds(parentId, conn)
  - deleteDocumentsInFolder(folderId, conn)
  - deleteFolderById(folderId, conn)

Components (JavaScript)

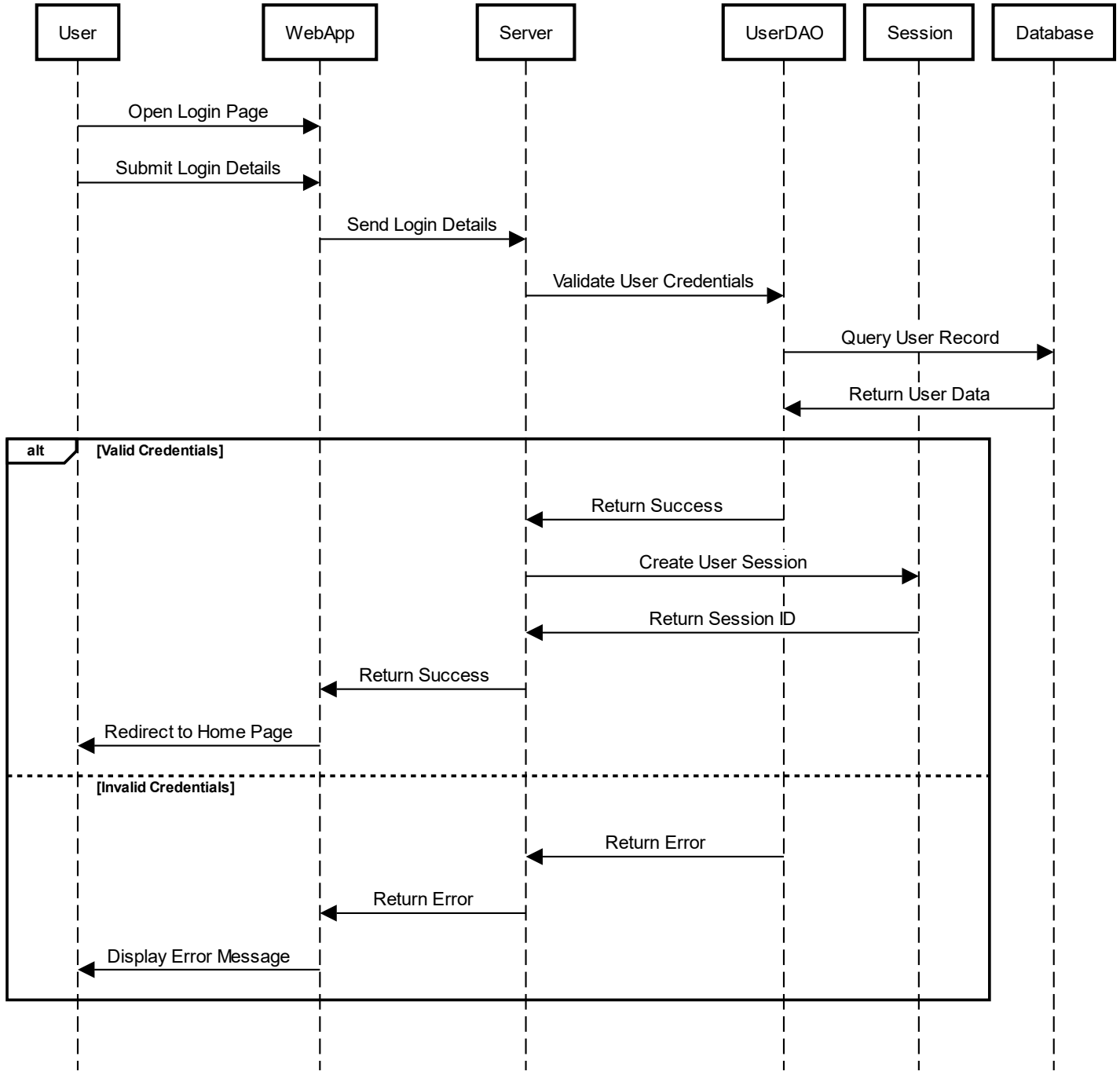
- login.js
- register.js
- home.js
  - handleBinDragOver(event)
  - handleBinDrop(event)
  - handleDragStart(event, itemId, itemType)
  - handleDrop(event, folderId)
  - loadFolders()
  - buildFolderTree(folders)
  - addRootFolder()
  - addSubfolder(parentId)
  - displayFolders(folders, parentElement, level = 0)
  - handleDragOver(event)
  - loadDocuments(folderId, parentElement, level)
  - viewDocument(docId)
  - displayDocumentDetails(document)
  - displayDocuments(documents, parentElement, level)
  - moveDocument(docId, newFolderId)
  - addDocument(folderId)
  - deleteDocument(documentId)
  - deleteFolder(folderId)
  - getCurrentFolderId()
  - logout()

IFML diagram (JavaScript)

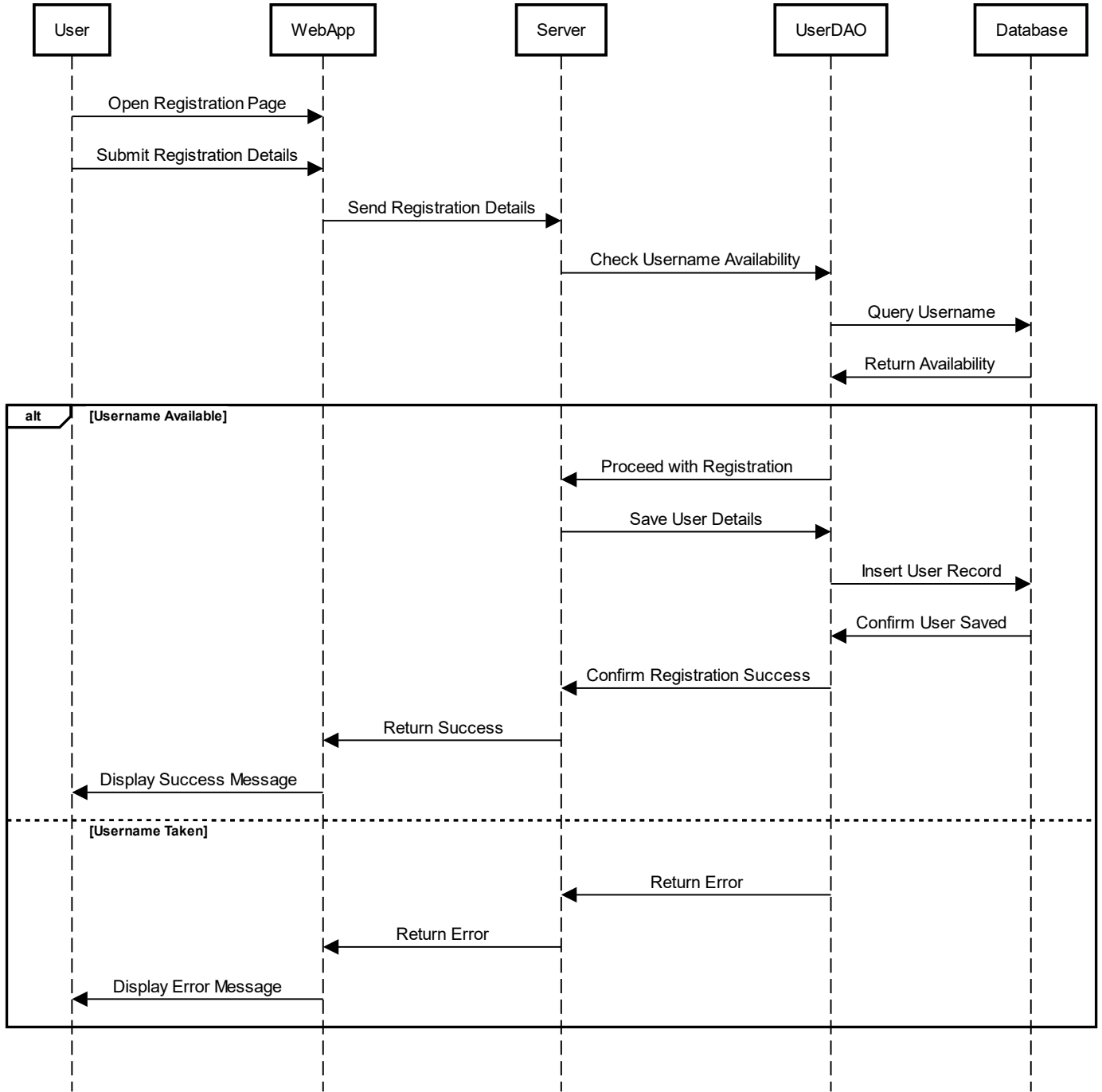

logout
/logout
wrong credentials
login page
<<form>>
login form
/login
<<link>>
go to registration
succesfuly registered
register page
<<form>>
registration form
/register
<<link>>
go to login
wrong parameters
refresh/load
home page
<<button>>
logout
/addFolder
add folder
/getDocumentDetails
/getFolders
<<list>>
folder
and
document
tree
<<details>>
document
details
/getDocuments
empty name / cancel
<<form>>
document
input
form
/addDocument
check document add
cancel
delete confirmation
drop item
< >
bin
confirm
/deleteFolder
/deleteDocument

Sequence diagrams (JavaScript version)

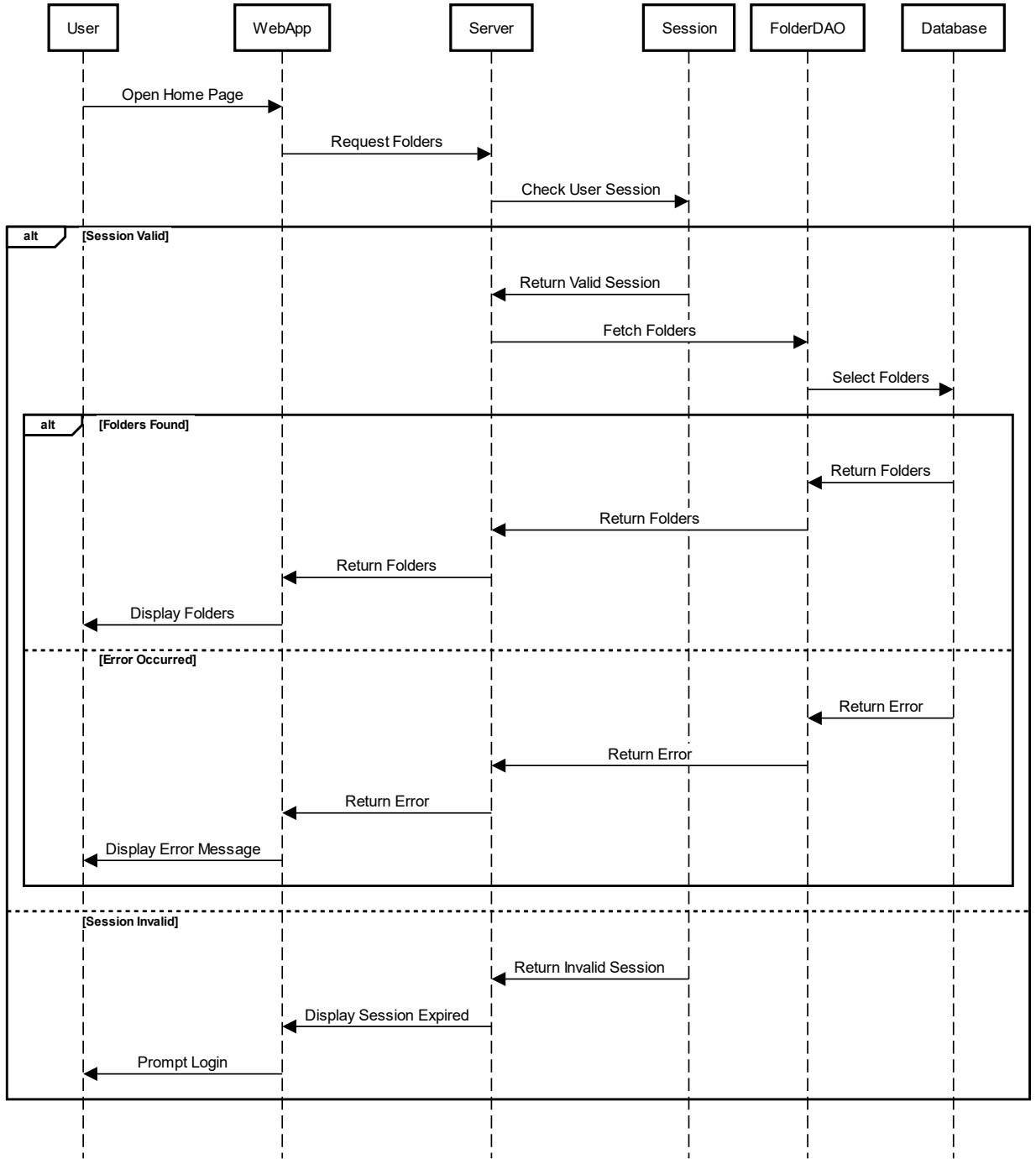
User Login


User
WebApp
Server
UserDAO
Session
Database
Open Login Page
Submit Login Details
Send Login Details
Validate User Credentials
Query User Record
Return User Data
alt
[Valid Credentials]
Return Success
Create User Session
Return Session ID
Return Success
Redirect to Home Page
[Invalid Credentials]
Return Error
Return Error
Display Error Message

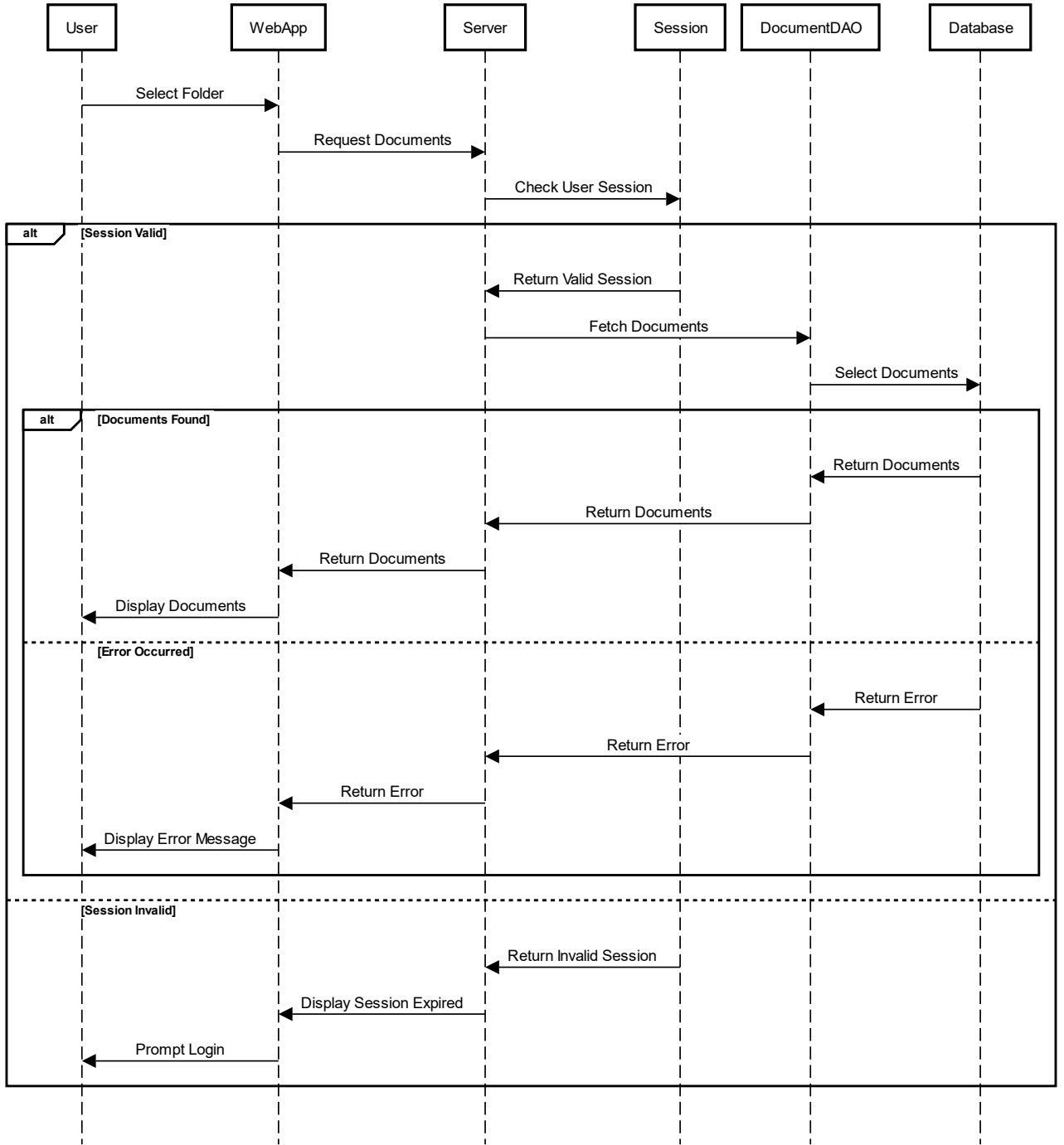
User Registration



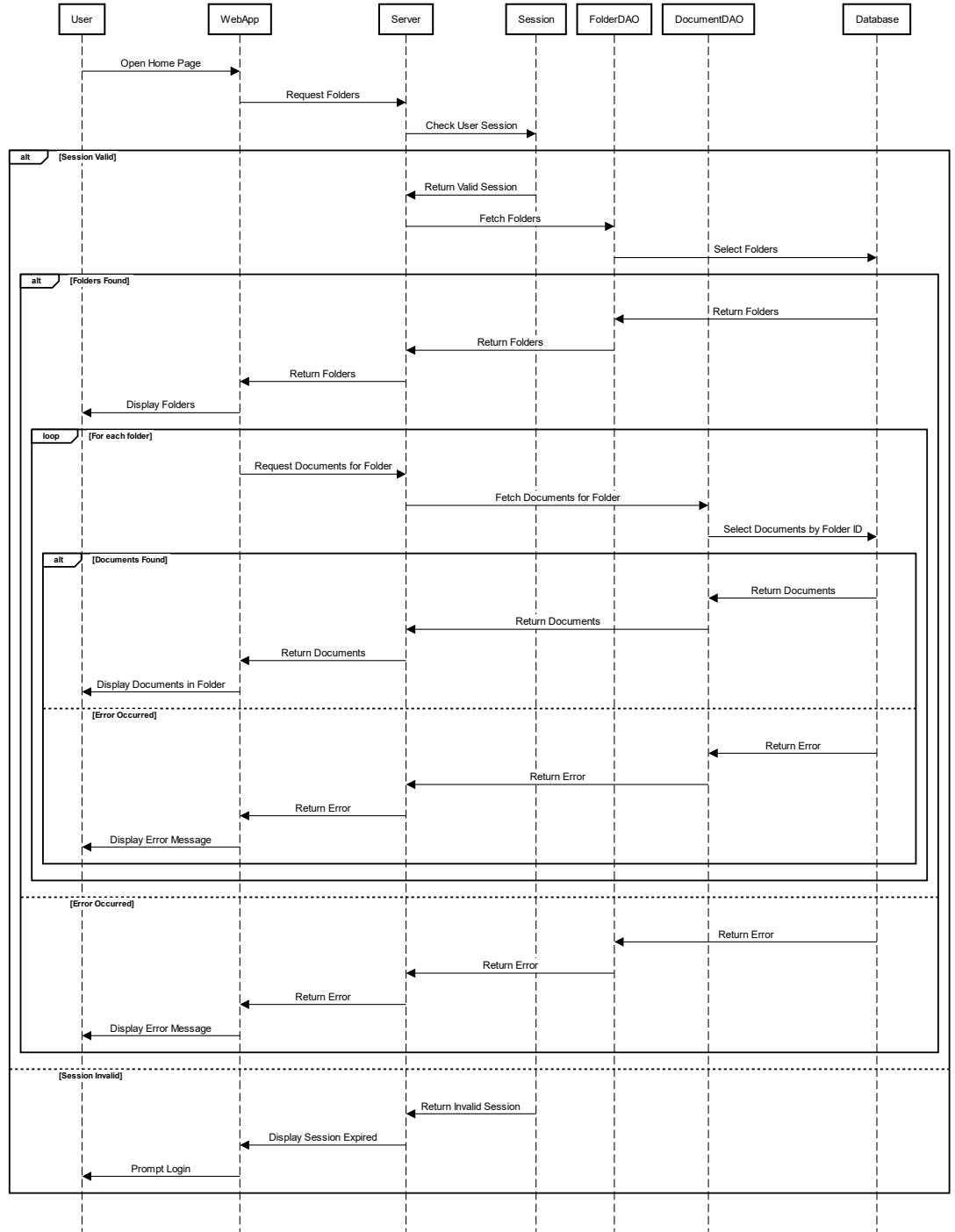
Load Folders



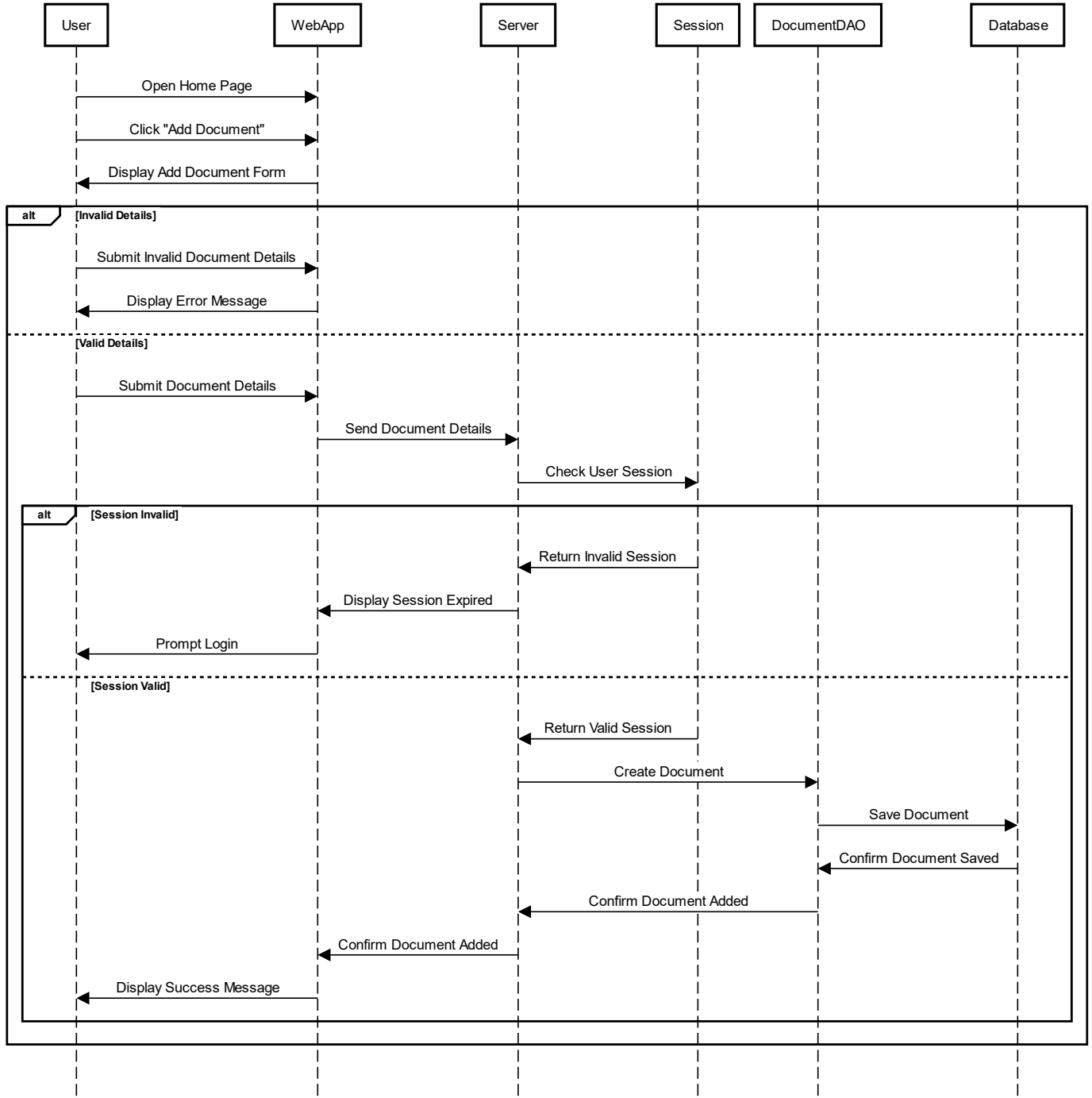
Load Documents



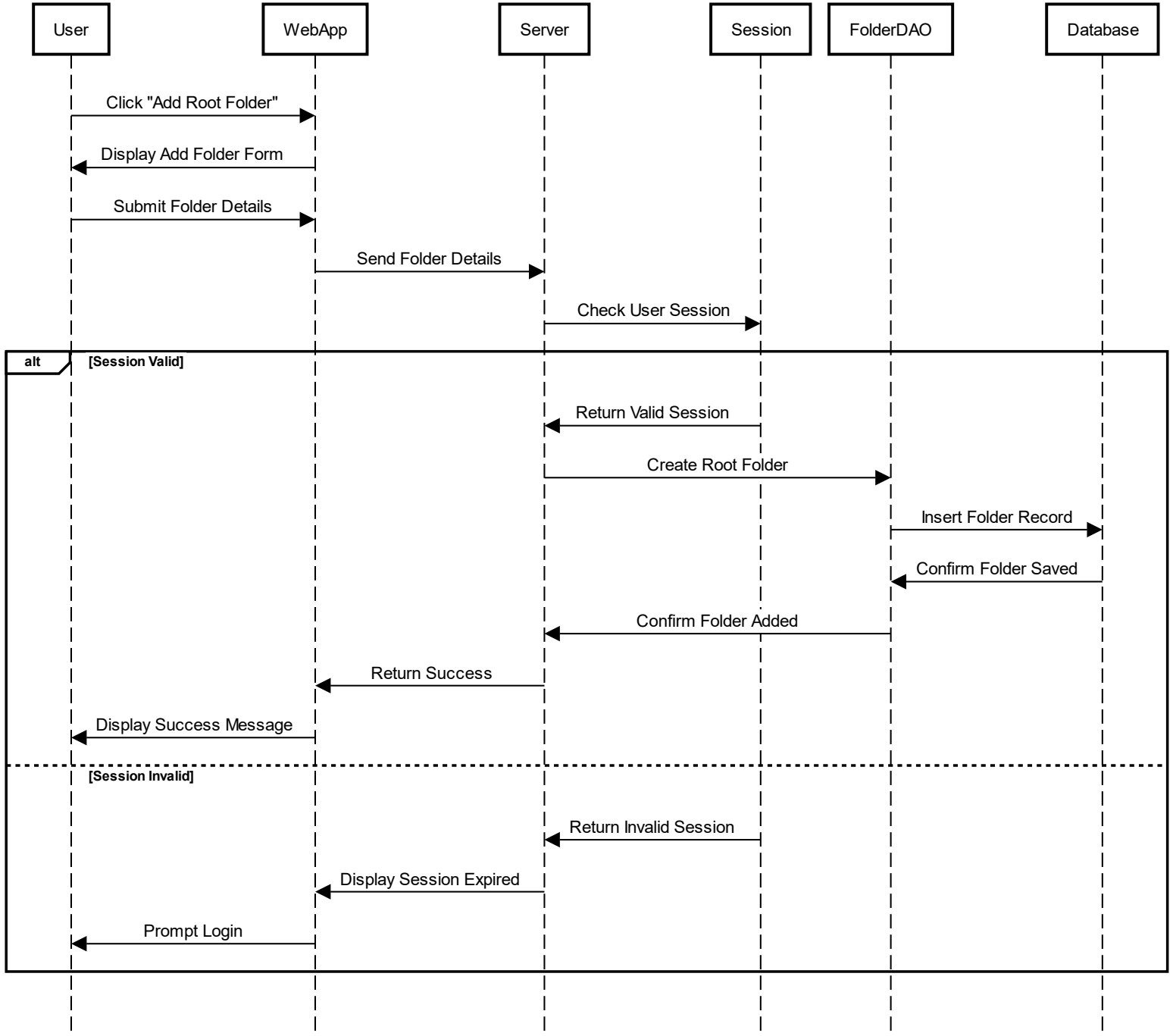
Load Folders and Documents



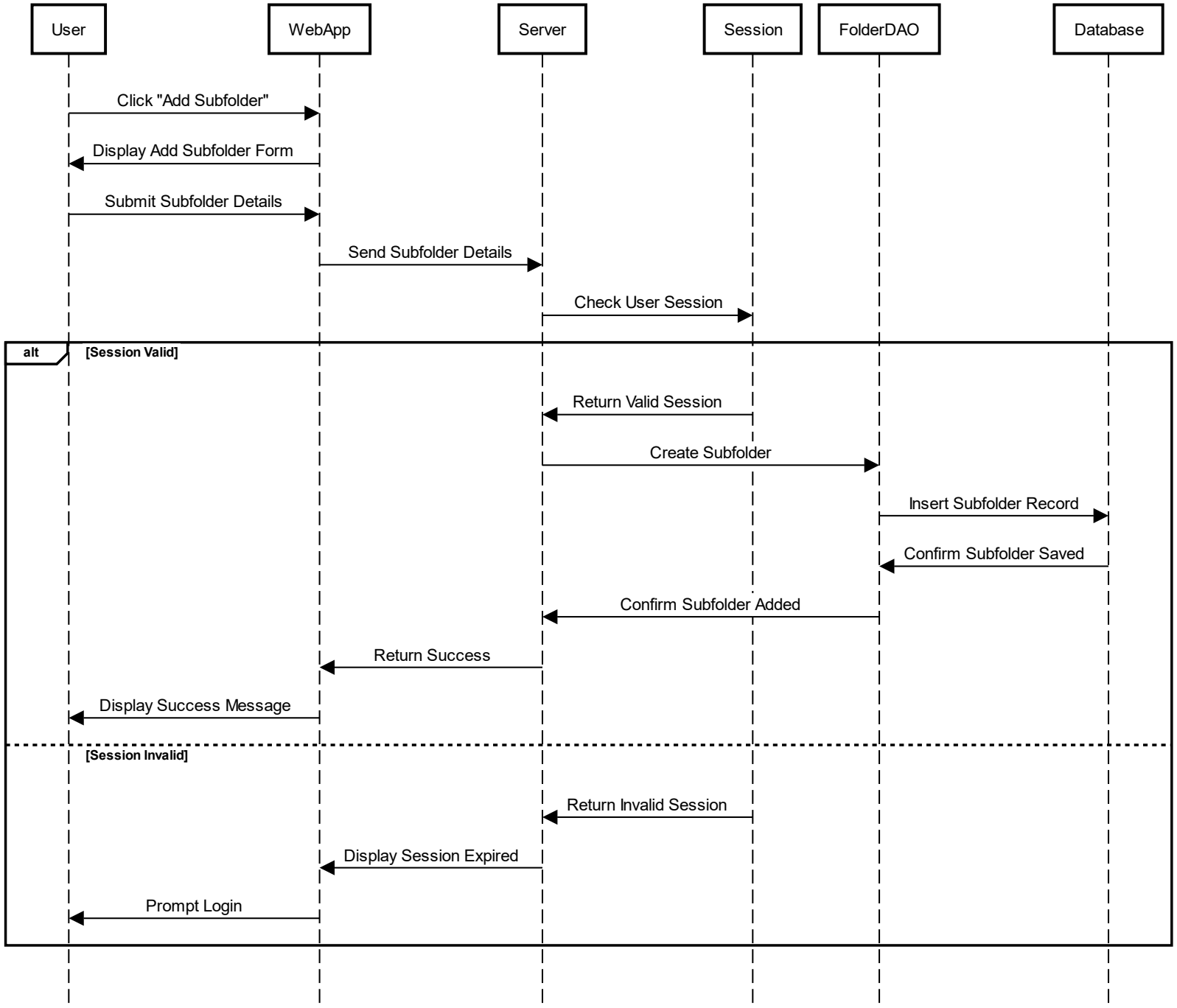
Add Document



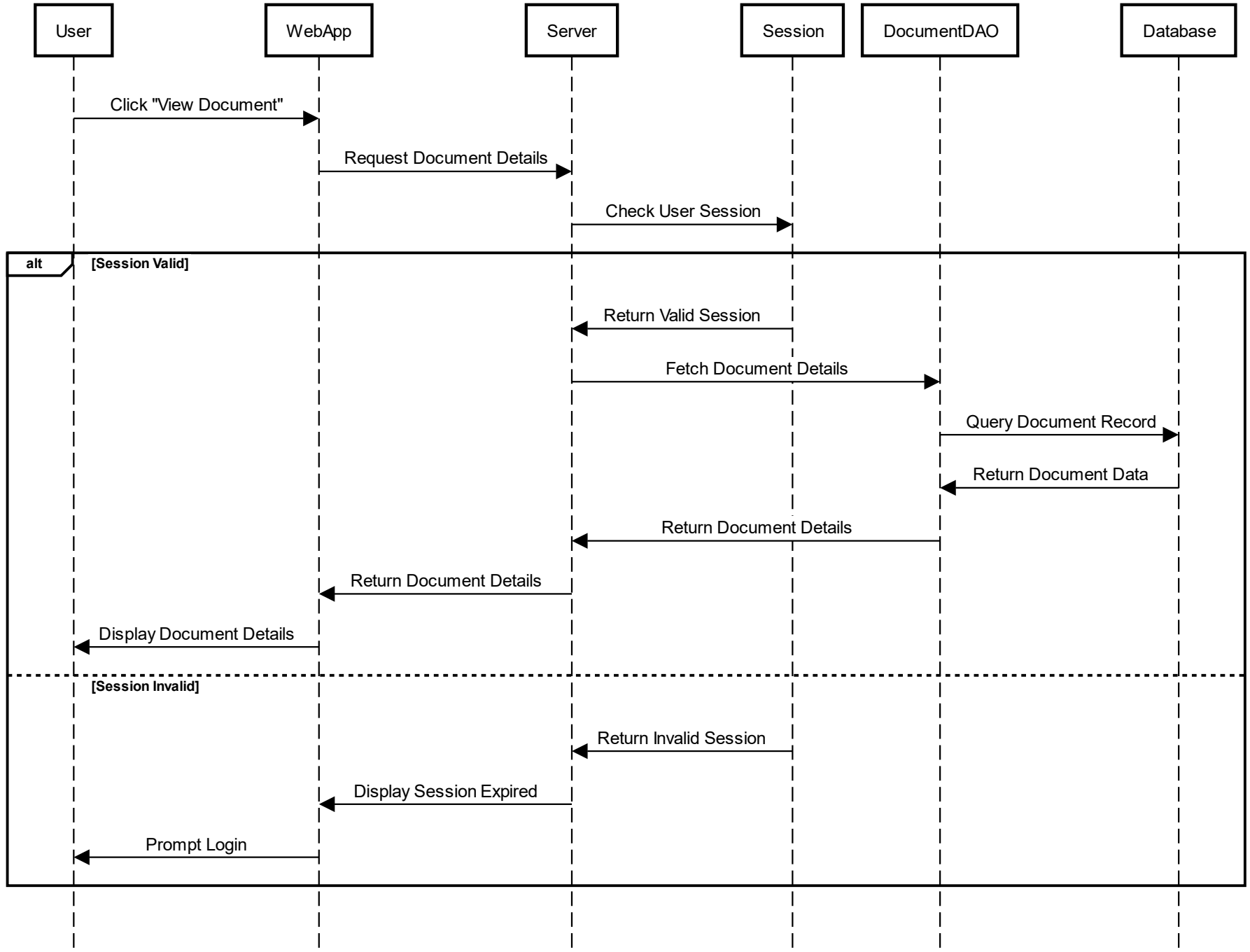
Add Root Folder



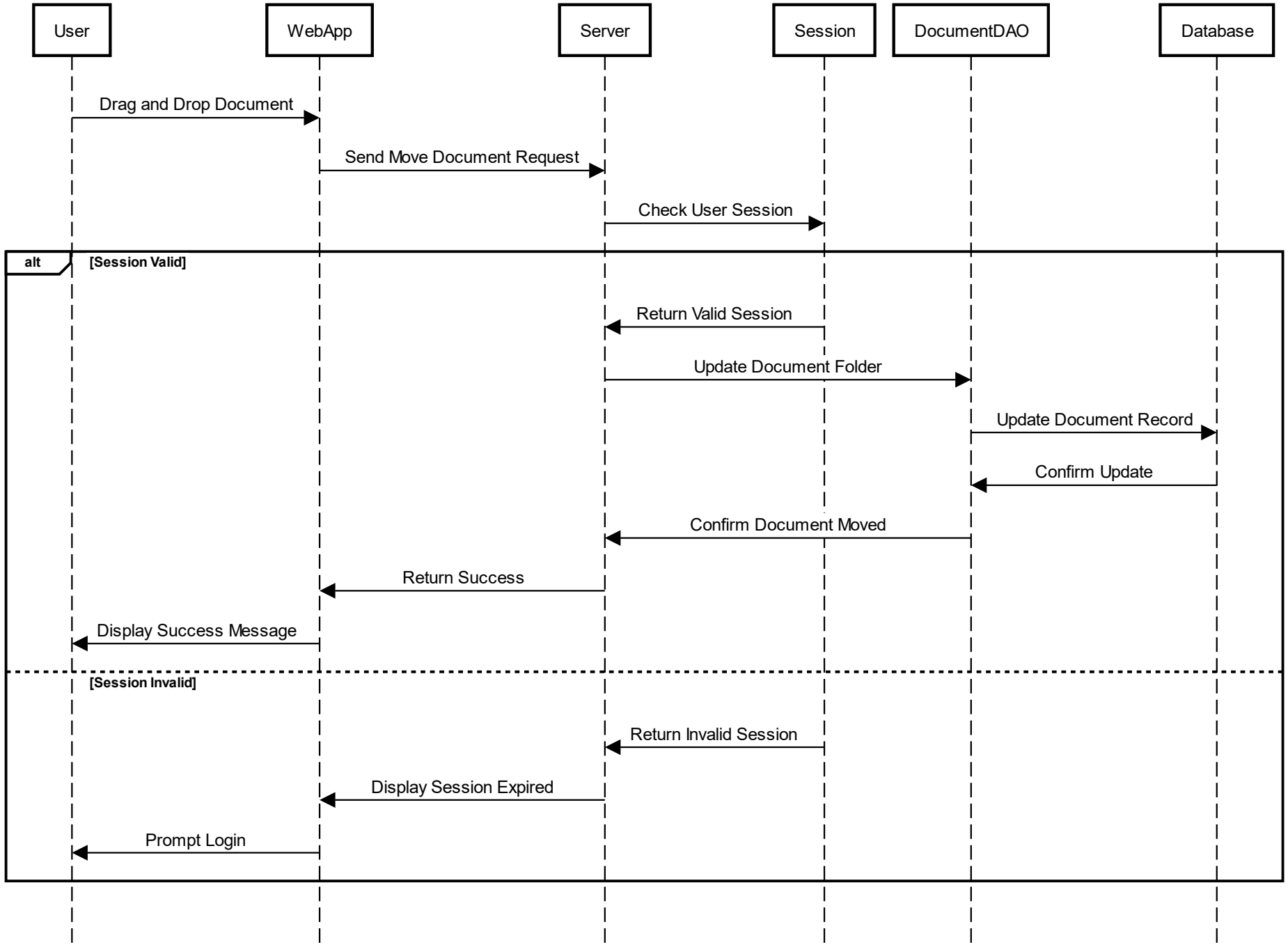
Add Subfolder



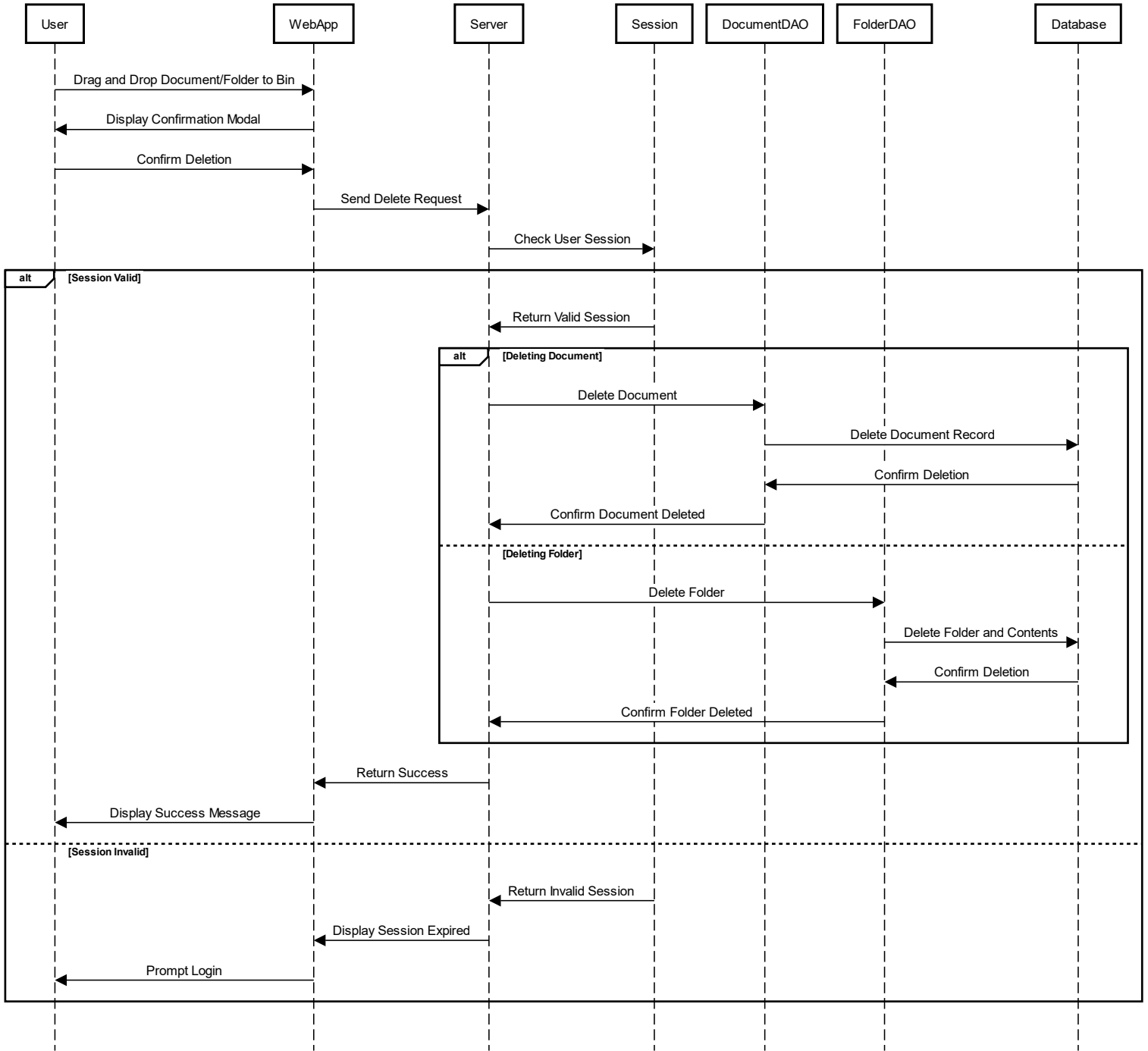
View Document



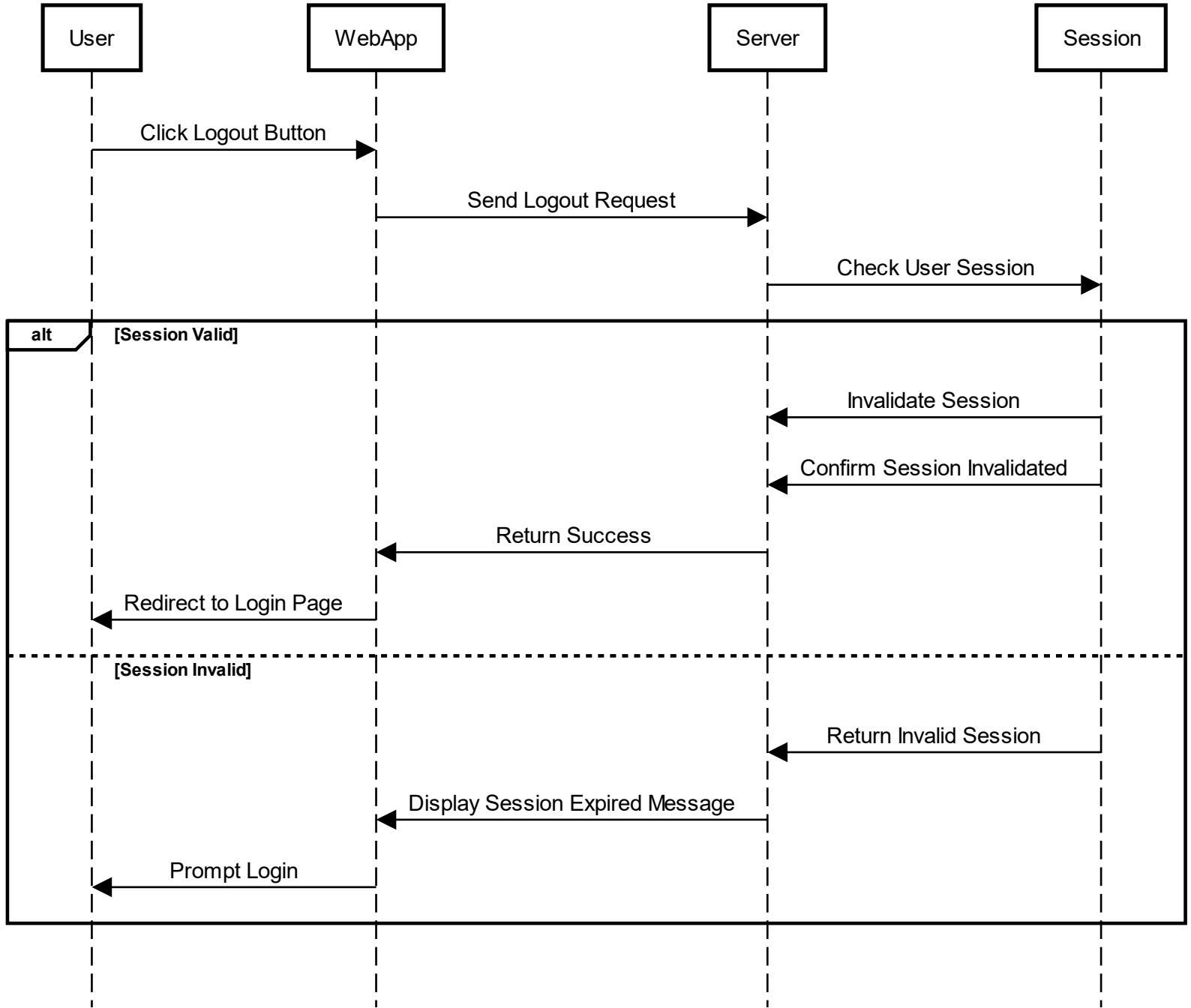
Move Document



Delete Document or Folder



User Logout


User
WebApp
Server
Session
Click Logout Button
Send Logout Request
Check User Session
alt
[Session Valid]
Invalidate Session
Confirm Session Invalidated
Return Success
Redirect to Login Page
[Session Invalid]
Return Invalid Session
Display Session Expired Message
Prompt Login