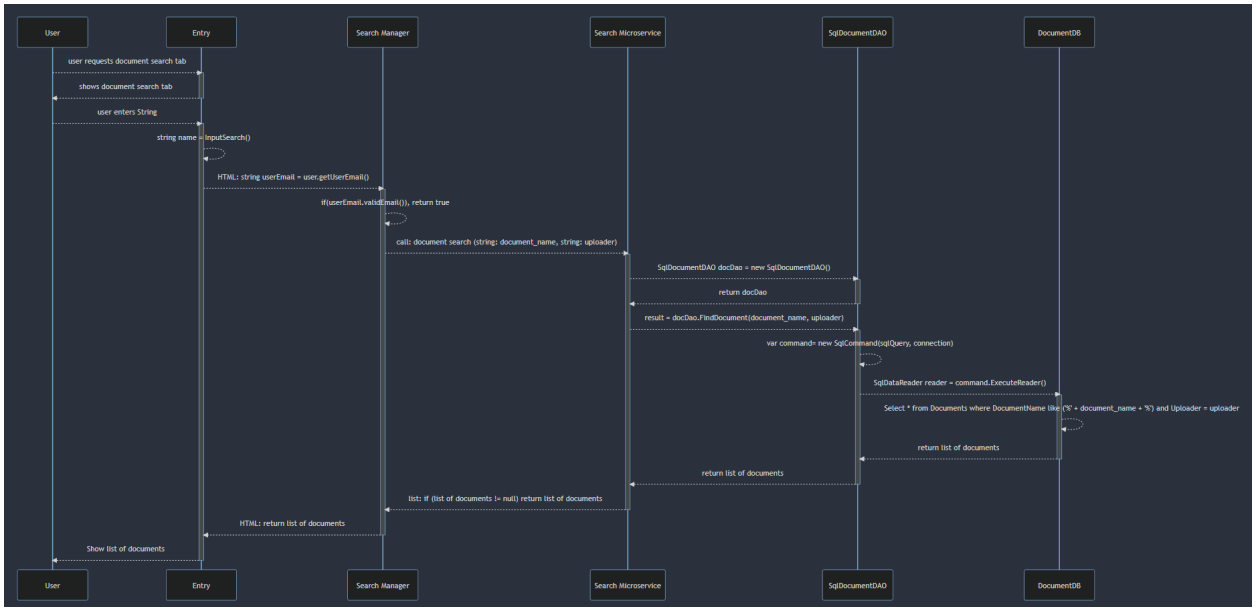
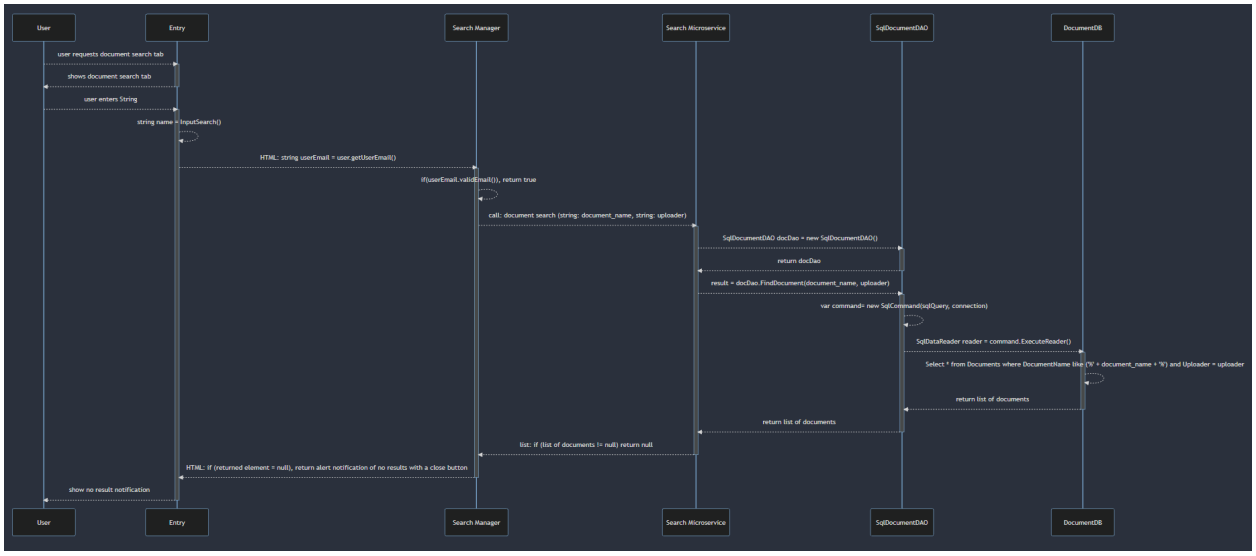
Document Storage Search

Success 1



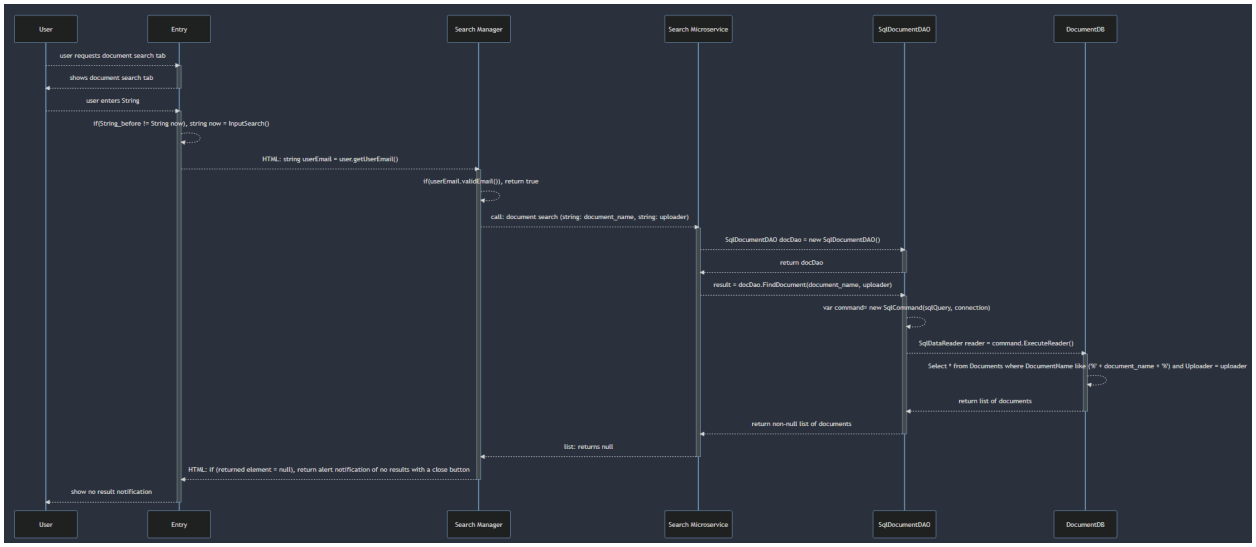
sequenceDiagram
User-->>+Entry: user requests document search tab
Entry-->>-User: shows document search tab
User-->>+Entry: user enters String
Entry-->>Entry: string name = InputSearch()
Entry-->>+Search Manager: HTML: string userEmail = user.getUserEmail()
Search Manager-->>Search Manager: if(userEmail.validEmail()), return true
Search Manager-->>+Search Microservice: call: document search (string: document_name, string: uploader)
Search Microservice-->>+SqlDocumentDAO: SqlDocumentDAO docDao = new SqlDocumentDAO();
SqlDocumentDAO-->>-Search Microservice: return docDao
Search Microservice-->>+SqlDocumentDAO: result = docDao.FindDocument(document_name, uploader)
SqlDocumentDAO-->>SqlDocumentDAO: var command= new SqlCommand(sqlQuery, connection)
SqlDocumentDAO-->>+DocumentDB: SqlDataReader reader = command.ExecuteReader()
DocumentDB-->>DocumentDB: Select * from Documents where DocumentName like ('%' + document_name + '%') and (Uploader = uploader or Shared = uploader)
DocumentDB-->>-SqlDocumentDAO: return list of documents
SqlDocumentDAO-->>-Search Microservice: return list of documents
Search Microservice-->>-Search Manager: list: if (list of documents != null) return list of documents
Search Manager-->>-Entry: HTML: return list of documents
Entry-->>-User: Show list of documents

Success 2



sequenceDiagram
User-->>+Entry: user requests document search tab
Entry-->>-User: shows document search tab
User-->>+Entry: user enters String
Entry-->>Entry: string name = InputSearch()
Entry-->>+Search Manager: HTML: string userEmail = user.getUserEmail()
Search Manager-->>Search Manager: if(userEmail.validEmail()), return true
Search Manager-->>+Search Microservice: call: document search (string: document_name, string: uploader)
Search Microservice-->>+SqlDocumentDAO: SqlDocumentDAO docDao = new SqlDocumentDAO();
SqlDocumentDAO-->>-Search Microservice: return docDao
Search Microservice-->>+SqlDocumentDAO: result = docDao.FindDocument(document_name, uploader)
SqlDocumentDAO-->>SqlDocumentDAO: var command= new SqlCommand(sqlQuery, connection)
SqlDocumentDAO-->>+DocumentDB: SqlDataReader reader = command.ExecuteReader()
DocumentDB-->>DocumentDB: Select * from Documents where DocumentName like ('%' + document_name + '%') and (Uploader = uploader or Shared = uploader)
DocumentDB-->>-SqlDocumentDAO: return list of documents
SqlDocumentDAO-->>-Search Microservice: return list of documents
Search Microservice-->>-Search Manager: list: if (list of documents = null) return null
Search Manager-->>-Entry: HTML: if (returned element = null), return alert notification of no results with a close button
Entry-->>-User: show no result notification

Fail 1



sequenceDiagram

User-->>+Entry: user requests document search tab

Entry-->>-User: shows document search tab

User-->>+Entry: user enters String

Entry-->>Entry: if(String_before != String now), string now = InputSearch()

Entry-->>+Search Manager: HTML: string userEmail = user.getUserEmail()

Search Manager-->>Search Manager: if(userEmail.validEmail()), return true

Search Manager-->>+Search Microservice: call: document search (string: document_name, string: uploader)

Search Microservice-->>+SqlDocumentDAO: SqlDocumentDAO docDao = new SqlDocumentDAO();

SqlDocumentDAO-->>-Search Microservice: return docDao

Search Microservice-->>+SqlDocumentDAO: result = docDao.FindDocument(document_name, uploader)

SqlDocumentDAO-->>SqlDocumentDAO: var command= new SqlCommand(sqlQuery, connection)

SqlDocumentDAO-->>+DocumentDB: SqlDataReader reader = command.ExecuteReader()

DocumentDB-->>DocumentDB: Select * from Documents where DocumentName like ('%' + document_name + '%') and (Uploader = uploader or Shared = uploader)

DocumentDB-->>-SqlDocumentDAO: return list of documents

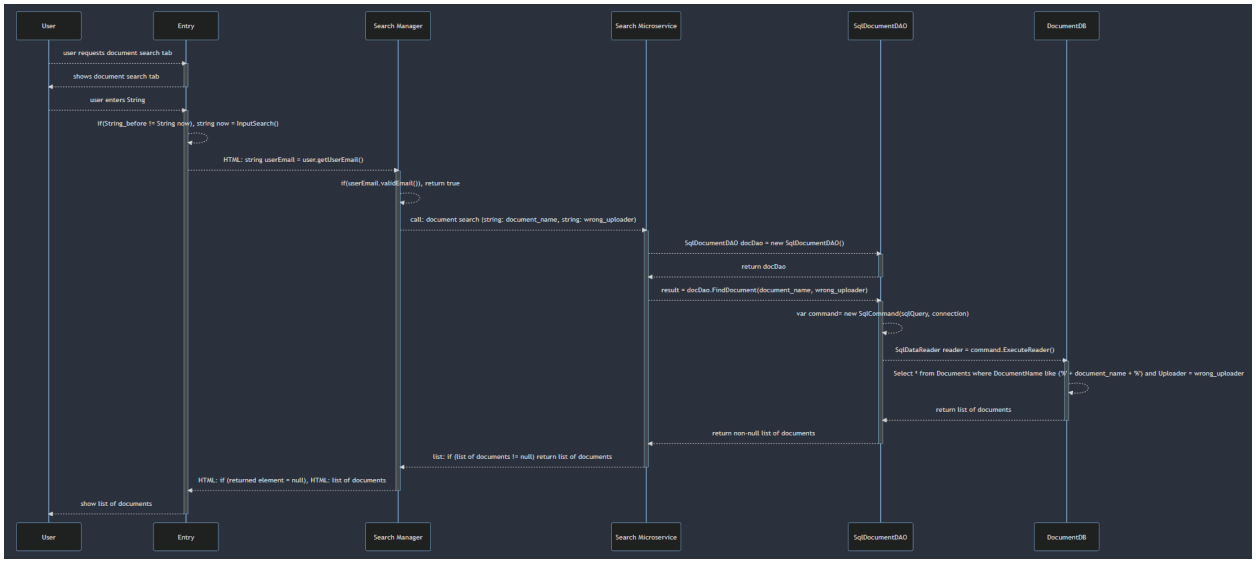
SqlDocumentDAO-->>-Search Microservice: return non-null list of documents

Search Microservice-->>-Search Manager: list: returns null

Search Manager-->>-Entry: HTML: if (returned element = null), return alert notification of no results with a close button

Entry-->>-User: show no result notification

Fail 2



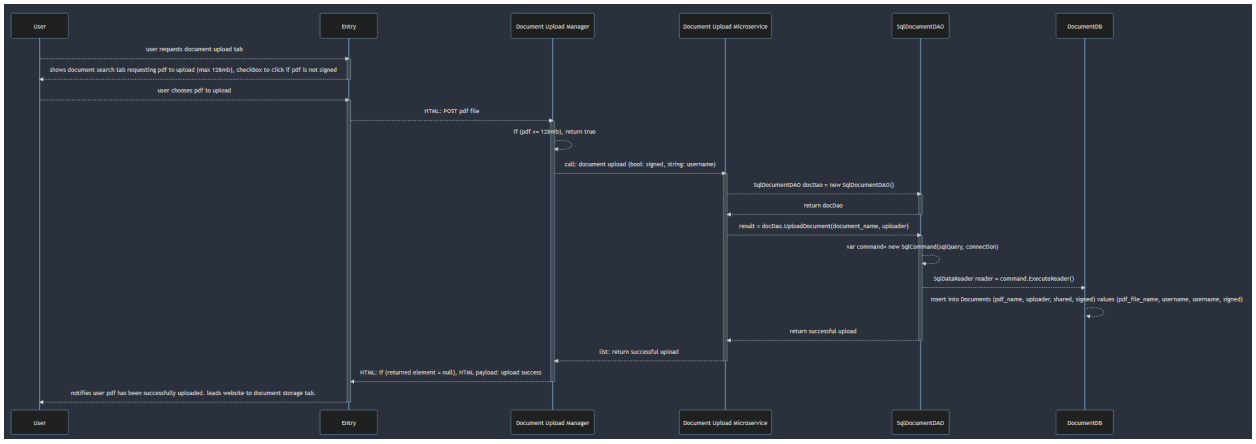
sequenceDiagram
User-->>+Entry: user requests document search tab
Entry-->>-User: shows document search tab
User-->>+Entry: user enters String
Entry-->>Entry: if(String_before != String now), string now = InputSearch()
Entry-->>+Search Manager: HTML: string userEmail = user.getUserEmail()
Search Manager-->>Search Manager: if(userEmail.validEmail()), return true
Search Manager-->>+Search Microservice: call: document search (string: document_name, string: wrong_uploader)
Search Microservice-->>+SqlDocumentDAO: SqlDocumentDAO docDao = new SqlDocumentDAO();
SqlDocumentDAO-->>-Search Microservice: return docDao
Search Microservice-->>+SqlDocumentDAO: result = docDao.FindDocument(document_name, wrong_uploader)
SqlDocumentDAO-->>SqlDocumentDAO: var command= new SqlCommand(sqlQuery, connection)
SqlDocumentDAO-->>+DocumentDB: SqlDataReader reader = command.ExecuteReader()
DocumentDB-->>DocumentDB: Select * from Documents where DocumentName like ('%' + document_name + '%') and (Uploader = wrong_uploader or Shared = wrong_uploader)
DocumentDB-->>-SqlDocumentDAO: return list of documents
SqlDocumentDAO-->>-Search Microservice: return non-null list of documents
Search Microservice-->>-Search Manager: list: if (list of documents != null) return list of documents
Search Manager-->>-Entry: HTML: if (returned element = null), HTML: list of documents
Entry-->>-User: show list of documents

Fail 3

sequenceDiagram
User-->>+Entry: user requests document search tab
Entry-->>-User: shows document search tab
User-->>+Entry: user enters String
Entry-->>Entry: if(String_before != String now), string now = InputSearch()
Entry-->>+Search Manager: HTML: string userEmail = user.getUserEmail()
Search Manager-->>Search Manager: if(userEmail.validEmail()), return true
Search Manager-->>+Search Microservice: call: document search (string: document_name, string: wrong_uploader)
Search Microservice-->>+SqlDocumentDAO: SqlDocumentDAO docDao = new SqlDocumentDAO();
SqlDocumentDAO-->>-Search Microservice: return docDao
Search Microservice-->>+SqlDocumentDAO: result = docDao.FindDocument(document_name, wrong_uploader)
SqlDocumentDAO-->>SqlDocumentDAO: var command= new SqlCommand(sqlQuery, connection)
SqlDocumentDAO-->>+DocumentDB: SqlDataReader reader = command.ExecuteReader()
DocumentDB-->>DocumentDB: Select * from Documents where DocumentName like ('%' + document_name + '%')
SqlDocumentDAO-->>-Search Microservice: return non-null list of documents
Search Microservice-->>-Search Manager: list: if (list of documents != null) return list of documents
Search Manager-->>-Entry: HTML: if (returned element = null), HTML: list of documents
Entry-->>-User: show list of documents

Document Storage Upload

Success



sequenceDiagram

User-->>+Entry: user requests document upload tab

Entry-->>-User: shows document search tab requesting pdf to upload (max 128mb), checkbox to click if pdf is not signed

User-->>+Entry: user chooses pdf to upload

Entry-->>+Document Upload Manager: HTML: POST pdf file

Document Upload Manager-->>Document Upload Manager: if (pdf <= 128mb), return true

Document Upload Manager-->>+Document Upload Microservice: call: document upload (bool: signed, string: username)

Document Upload Microservice-->>+SqlDocumentDAO: SqlDocumentDAO docDao = new SqlDocumentDAO();

SqlDocumentDAO-->>-Document Upload Microservice: return docDao

Document Upload Microservice-->>+SqlDocumentDAO: result = docDao.UploadDocument(document_name, uploader)

SqlDocumentDAO-->>SqlDocumentDAO: var command= new SqlCommand(sqlQuery, connection)

SqlDocumentDAO-->>+DocumentDB: SqlDataReader reader = command.ExecuteReader()

DocumentDB-->>DocumentDB: Insert into Documents (pdf_name, uploader, shared, signed) values (pdf_file_name, username, username, signed)

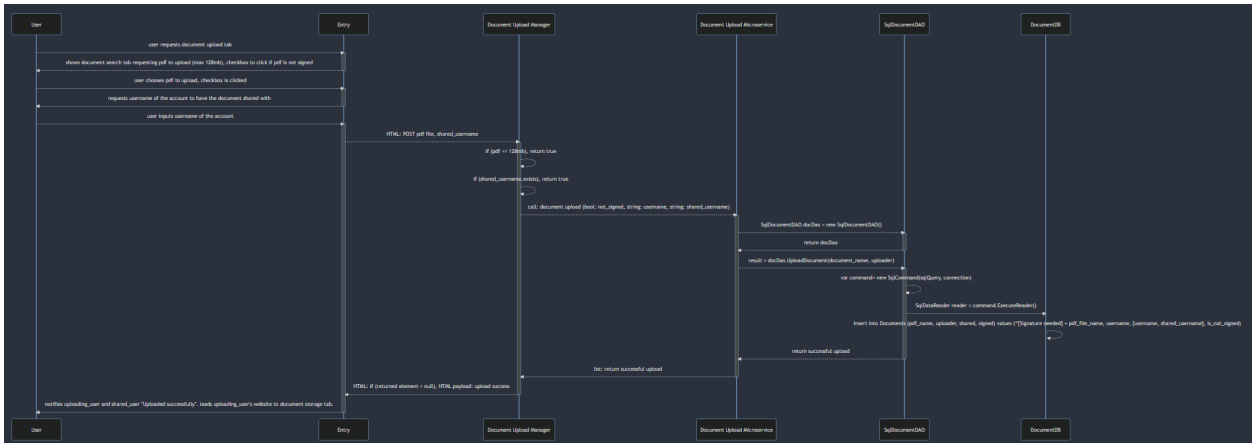
SqlDocumentDAO-->>-Document Upload Microservice: return successful upload

Document Upload Microservice-->>-Document Upload Manager: list: return successful upload

Document Upload Manager-->>-Entry: HTML: if (returned element = null), HTML payload: upload success

Entry-->>-User: notifies user pdf has been successfully uploaded. leads website to document storage tab.

success



sequenceDiagram

User-->>+Entry: user requests document upload tab

Entry-->>-User: shows document search tab requesting pdf to upload (max 128mb), checkbox to click if pdf is not signed

User-->>+Entry: user chooses pdf to upload, checkbox is clicked

Entry-->>-User: requests username of the account to have the document shared with

User-->>+Entry: user inputs username of the account

Entry-->>+Document Upload Manager: HTML: POST pdf file, shared_username

Document Upload Manager-->>Document Upload Manager: if (pdf <= 128mb), return true

Document Upload Manager-->>Document Upload Manager: if (shared_username.exists), return true

Document Upload Manager-->>+Document Upload Microservice: call: document upload (bool: not_signed, string: username, string: shared_username)

Document Upload Microservice-->>+SqlDocumentDAO: SqlDocumentDAO docDao = new SqlDocumentDAO();

SqlDocumentDAO-->>-Document Upload Microservice: return docDao

Document Upload Microservice-->>+SqlDocumentDAO: result = docDao.UploadDocument(document_name, uploader)

SqlDocumentDAO-->>SqlDocumentDAO: var command= new SqlCommand(sqlQuery, connection)

SqlDocumentDAO-->>+DocumentDB: SqlDataReader reader = command.ExecuteReader()

DocumentDB-->>DocumentDB: Insert into Documents (pdf_name, uploader, shared, signed) values (“[Signature needed] + pdf_file_name, username, [username, shared_username], is_not_signed)

SqlDocumentDAO-->>-Document Upload Microservice: return successful upload

Document Upload Microservice-->>-Document Upload Manager: list: return successful upload

Document Upload Manager-->>-Entry: HTML: if (returned element = null), HTML payload: upload success

Entry-->>-User: notifies uploading_user and shared_user "Uploaded successfully". leads uploading_user's website to document storage tab.

Fail 1

sequenceDiagram

User-->>+Entry: user requests document upload tab

Entry-->>-User: shows document search tab requesting pdf to upload (max 128mb), checkbox to click if pdf is not signed

User-->>+Entry: user chooses pdf to upload

Entry-->>+Document Upload Manager: HTML: POST pdf file

Document Upload Manager-->>Document Upload Manager: if (pdf <= 128mb), return true

Document Upload Manager-->>+Document Upload Microservice: call: document upload (bool: signed, string: username)

Document Upload Microservice-->>+SqlDocumentDAO: SqlDocumentDAO docDao = new SqlDocumentDAO();

SqlDocumentDAO-->>-Document Upload Microservice: return docDao

Document Upload Microservice-->>+SqlDocumentDAO: result = docDao.UploadDocument(document_name, uploader)

SqlDocumentDAO-->>SqlDocumentDAO: var command= new SqlCommand(sqlQuery, connection)

SqlDocumentDAO-->>+DocumentDB: SqlDataReader reader = command.ExecuteReader()

DocumentDB-->>DocumentDB: Insert into Documents (pdf_name, uploader, shared, signed) values (pdf_file_name, username, username, signed)

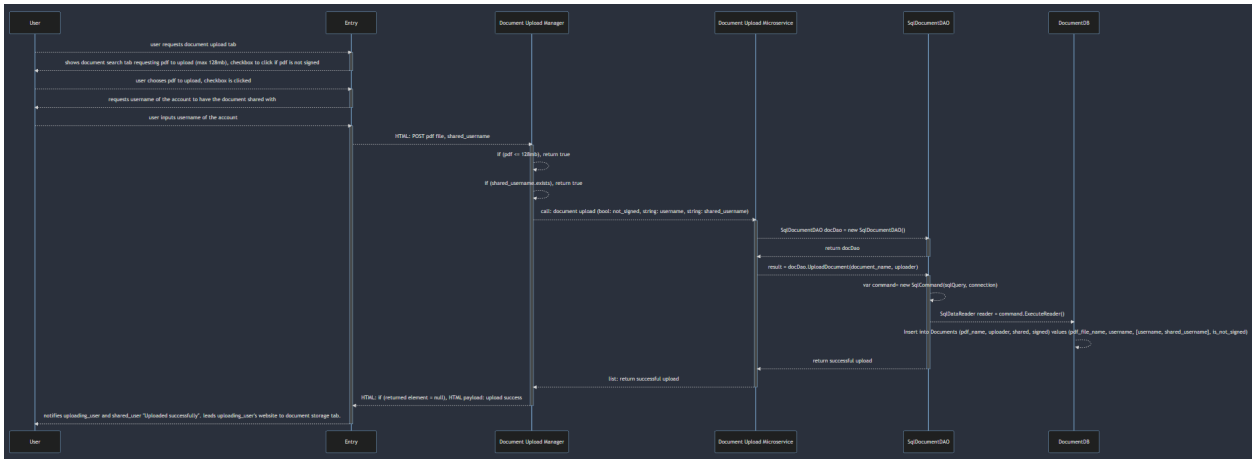
SqlDocumentDAO-->>-Document Upload Microservice: return successful upload

Document Upload Microservice-->>-Document Upload Manager: list: return successful upload

Document Upload Manager-->>-Entry: HTML: if (returned element = null), HTML payload: upload success

Entry-->>-User: leads website to document storage tab.

Fail 2



sequenceDiagram
User-->>+Entry: user requests document upload tab
Entry-->>-User: shows document search tab requesting pdf to upload (max 128mb), checkbox to click if pdf is not signed
User-->>+Entry: user chooses pdf to upload, checkbox is clicked
Entry-->>-User: requests username of the account to have the document shared with
User-->>+Entry: user inputs username of the account
Entry-->>+Document Upload Manager: HTML: POST pdf file, shared_username
Document Upload Manager-->>Document Upload Manager: if (pdf <= 128mb), return true
Document Upload Manager-->>Document Upload Manager: if (shared_username.exists), return true
Document Upload Manager-->>+Document Upload Microservice: call: document upload (bool: not_signed, string: username, string: shared_username)
Document Upload Microservice-->>+SqlDocumentDAO: SqlDocumentDAO docDao = new SqlDocumentDAO();
SqlDocumentDAO-->>-Document Upload Microservice: return docDao
Document Upload Microservice-->>+SqlDocumentDAO: result = docDao.UploadDocument(document_name, uploader)
SqlDocumentDAO-->>SqlDocumentDAO: var command= new SqlCommand(sqlQuery, connection)
SqlDocumentDAO-->>+DocumentDB: SqlDataReader reader = command.ExecuteReader()
DocumentDB-->>DocumentDB: Insert into Documents (pdf_name, uploader, shared, signed) values (pdf_file_name, username, [username, shared_username], is_not_signed)
SqlDocumentDAO-->>-Document Upload Microservice: return successful upload
Document Upload Microservice-->>-Document Upload Manager: list: return successful upload
Document Upload Manager-->>-Entry: HTML: if (returned element = null), HTML payload: upload success
Entry-->>-User: notifies uploading_user and shared_user "Uploaded successfully". leads uploading_user's website to document storage tab.

Document Storage Delete

success

sequenceDiagram

User-->>+Entry: user requests document storage tab

Entry-->>-User: shows document search tab requesting pdf to upload (max 128mb), checkbox to click if pdf is not signed

User-->>+Entry: user chooses pdf to delete

Entry-->>+Document Delete Manager: HTML: POST pdf file

Document Delete Manager-->>+Document Delete Microservice: call: document delete (PDFName = pdfname, uploader=username)

Document Delete Microservice-->>+SqlDocumentDAO: SqlDocumentDAO docDao = new SqlDocumentDAO();

SqlDocumentDAO-->>-Document Delete Microservice: return docDao

Document Delete Microservice-->>+SqlDocumentDAO: result = docDao.DeleteDocument(document_name, uploader)

SqlDocumentDAO-->>SqlDocumentDAO: var command= new SqlCommand(sqlQuery, connection)

SqlDocumentDAO-->>+DocumentDB: SqlDataReader reader = command.ExecuteReader()

DocumentDB-->>DocumentDB: Delete from Documents where PDFName = pdfname and uploader = username

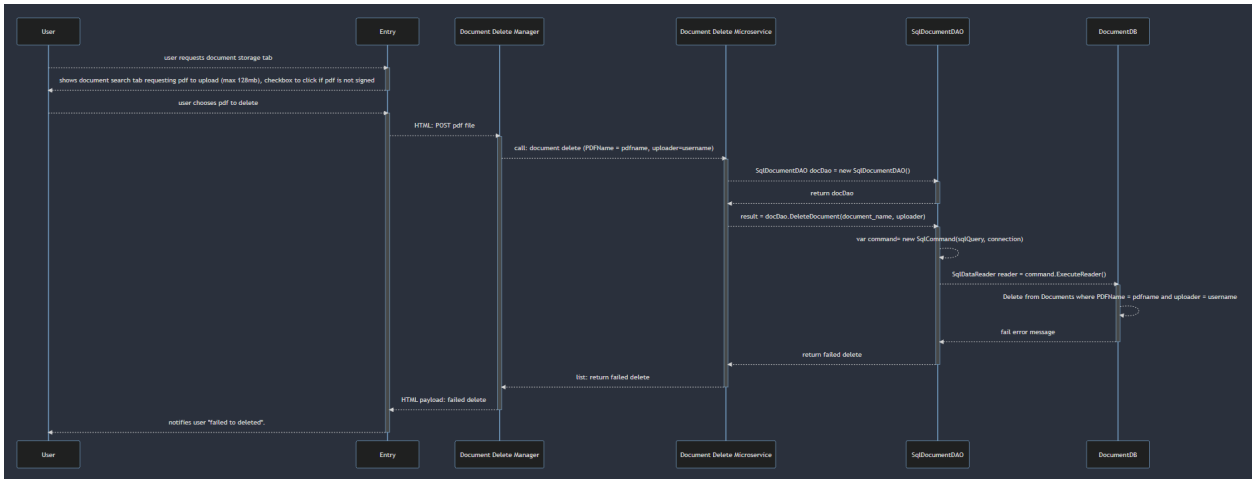
SqlDocumentDAO-->>-Document Delete Microservice: return successful delete

Document Delete Microservice-->>-Document Delete Manager: list: return successful delete

Document Delete Manager-->>-Entry: HTML payload: successful delete

Entry-->>-User: notifies user "successfully deleted". refreshes document storage tab

Fail 1



sequenceDiagram

User-->>+Entry: user requests document storage tab

Entry-->>-User: shows document search tab requesting pdf to upload (max 128mb), checkbox to click if pdf is not signed

User-->>+Entry: user chooses pdf to delete

Entry-->>+Document Delete Manager: HTML: POST pdf file

Document Delete Manager-->>+Document Delete Microservice: call: document delete (PDFName = pdfname, uploader=username)

Document Delete Microservice-->>+SqlDocumentDAO: SqlDocumentDAO docDao = new SqlDocumentDAO();

SqlDocumentDAO-->>-Document Delete Microservice: return docDao

Document Delete Microservice-->>+SqlDocumentDAO: result = docDao.DeleteDocument(document_name, uploader)

SqlDocumentDAO-->>SqlDocumentDAO: var command= new SqlCommand(sqlQuery, connection)

SqlDocumentDAO-->>+DocumentDB: SqlDataReader reader = command.ExecuteReader()

DocumentDB-->>DocumentDB: Delete from Documents where PDFName = pdfname and uploader = username

DocumentDB-->>-SqlDocumentDAO: fail error message

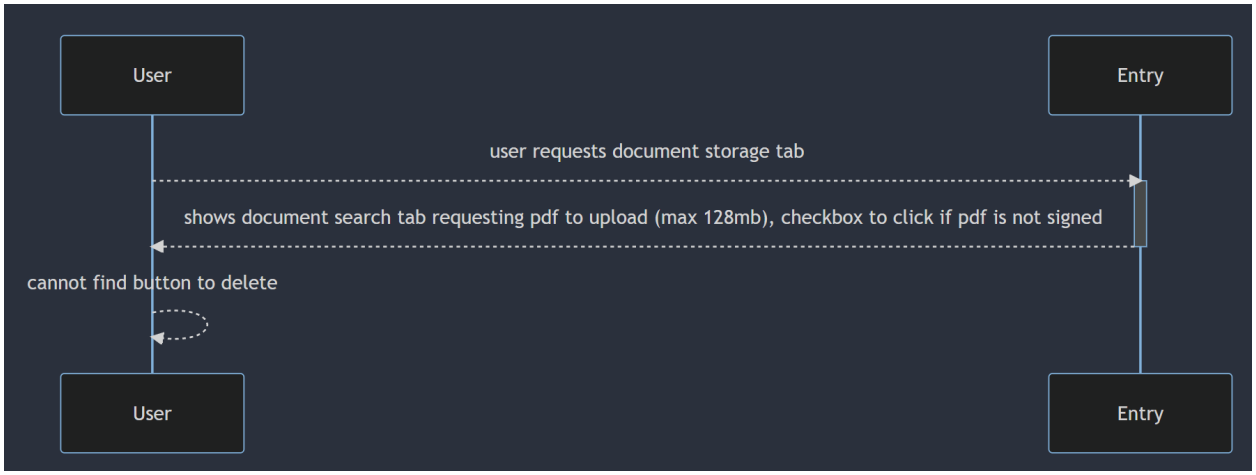
SqlDocumentDAO-->>-Document Delete Microservice: return failed delete

Document Delete Microservice-->>-Document Delete Manager: list: return failed delete

Document Delete Manager-->>-Entry: HTML payload: failed delete

Entry-->>-User: notifies user "failed to delete".

Fail 2



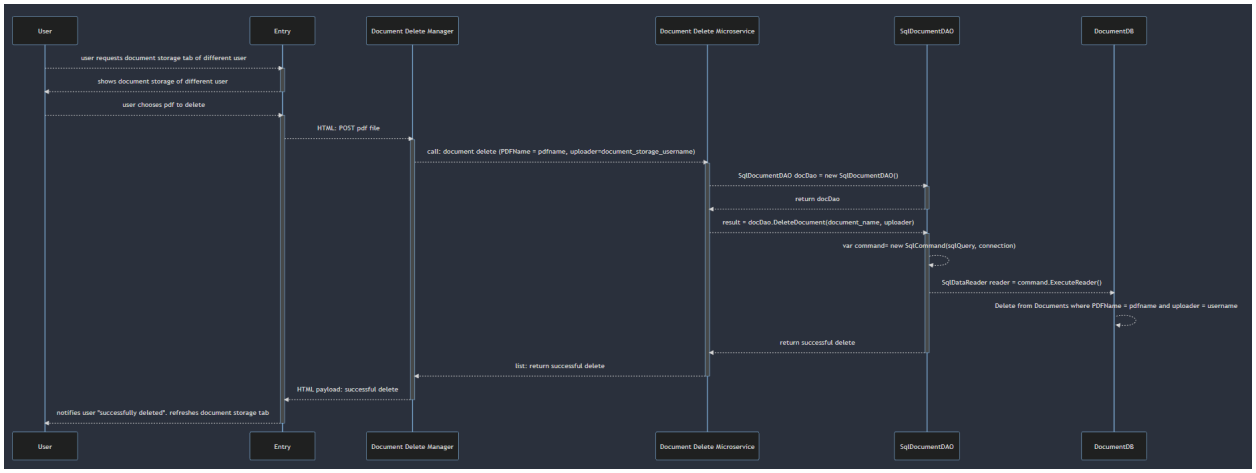
sequenceDiagram

User-->>+Entry: user requests document storage tab

Entry-->>-User: shows document search tab requesting pdf to upload (max 128mb), checkbox to click if pdf is not signed

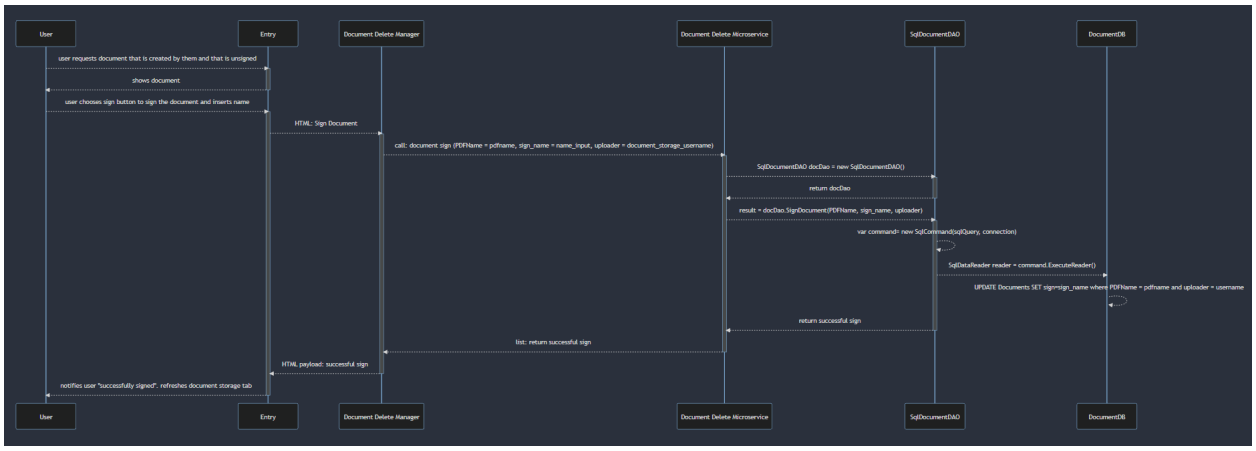
User-->>User: cannot find button to delete specific pdf

Fail 3



sequenceDiagram
User-->>+Entry: user requests document storage tab of different user
Entry-->>-User: shows document storage of different user
User-->>+Entry: user chooses pdf to delete
Entry-->>+Document Delete Manager: HTML: POST pdf file
Document Delete Manager-->>+Document Delete Microservice: call: document delete (PDFName = pdfname, uploader=document_storage_username)
Document Delete Microservice-->>+SqlDocumentDAO: SqlDocumentDAO docDao = new SqlDocumentDAO();
SqlDocumentDAO-->>-Document Delete Microservice: return docDao
Document Delete Microservice-->>+SqlDocumentDAO: result = docDao.DeleteDocument(document_name, uploader)
SqlDocumentDAO-->>SqlDocumentDAO: var command= new SqlCommand(sqlQuery, connection)
SqlDocumentDAO-->>+DocumentDB: SqlDataReader reader = command.ExecuteReader()
DocumentDB-->>DocumentDB: Delete from Documents where PDFName = pdfname and uploader = username
SqlDocumentDAO-->>-Document Delete Microservice: return successful delete
Document Delete Microservice-->>-Document Delete Manager: list: return successful delete
Document Delete Manager-->>-Entry: HTML payload: successful delete
Entry-->>-User: notifies user "successfully deleted". refreshes document storage tab

Document Storage Sign



sequenceDiagram

User-->>+Entry: user requests document that is created by them and that is unsigned

Entry-->>-User: shows document

User-->>+Entry: user chooses sign button to sign the document and inserts name

Entry-->>+Document Delete Manager: HTML: Sign Document

Document Delete Manager-->>+Document Delete Microservice: call: document sign (PDFName = pdfname, sign_name = name_input, uploader = document_storage_username)

Document Delete Microservice-->>+SqlDocumentDAO: SqlDocumentDAO docDao = new SqlDocumentDAO();

SqlDocumentDAO-->>-Document Delete Microservice: return docDao

Document Delete Microservice-->>+SqlDocumentDAO: result = docDao.SignDocument(PDFName, sign_name, uploader)

SqlDocumentDAO-->>SqlDocumentDAO: var command= new SqlCommand(sqlQuery, connection)

SqlDocumentDAO-->>+DocumentDB: SqlDataReader reader = command.ExecuteReader()

DocumentDB-->>DocumentDB: UPDATE Documents SET sign=sign_name where PDFName = pdfname and uploader = username

SqlDocumentDAO-->>-Document Delete Microservice: return successful sign

Document Delete Microservice-->>-Document Delete Manager: list: return successful sign

Document Delete Manager-->>-Entry: HTML payload: successful sign

Entry-->>-User: notifies user "successfully signed". refreshes document storage tab