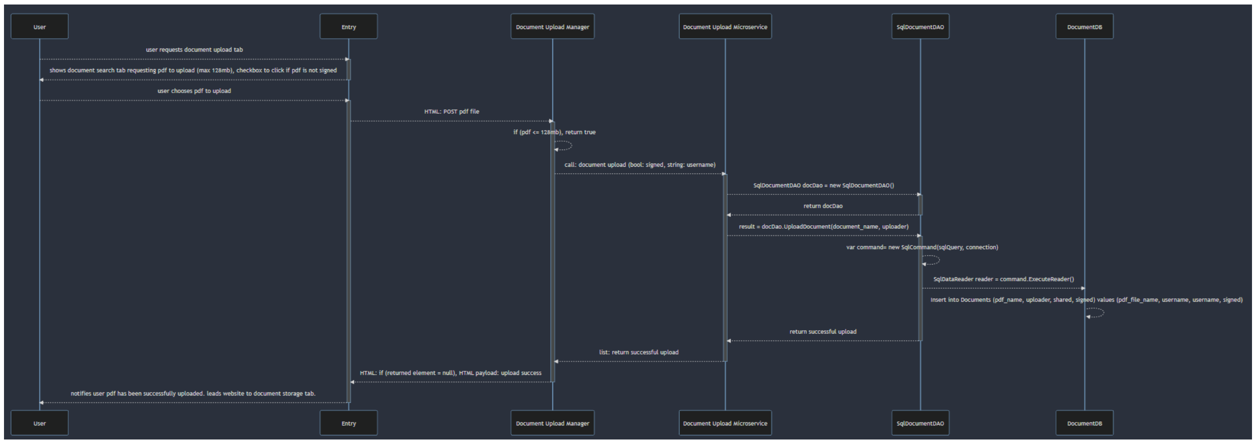
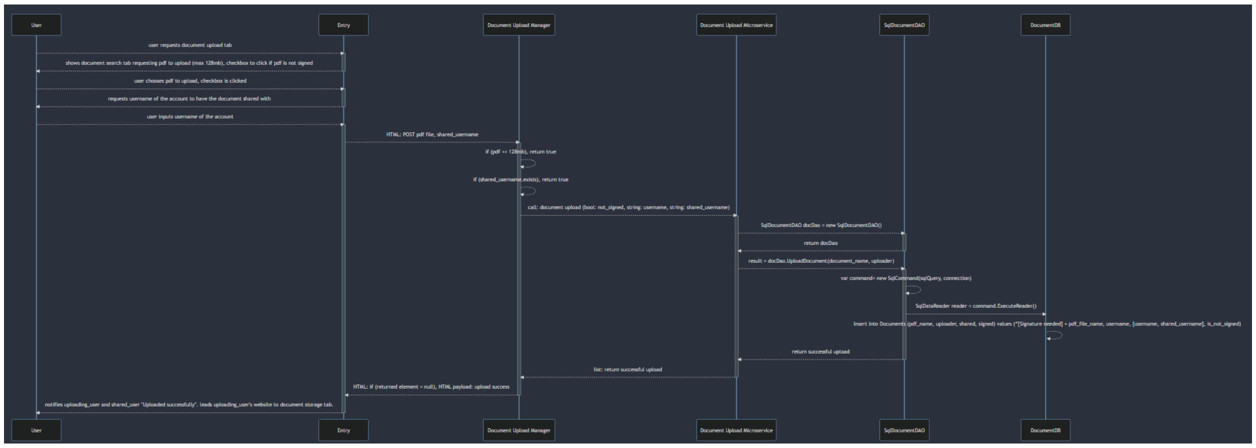
Success



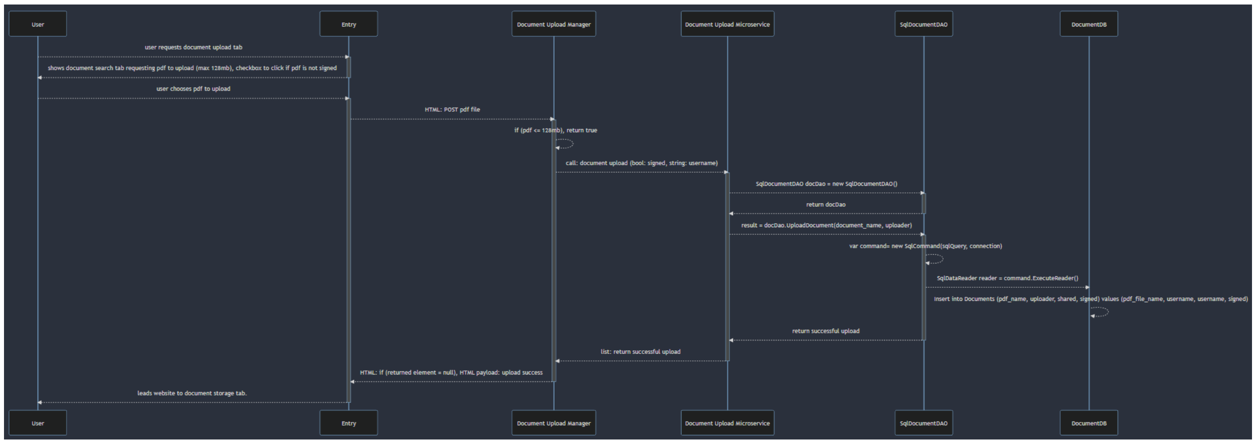
```
sequenceDiagram
    User-->>+Entry: user requests document upload tab
    Entry-->>-User: shows document search tab requesting pdf to upload (max 128mb), checkbox to click if pdf is not signed
    User-->>+Entry: user chooses pdf to upload
    Entry-->>+Document Upload Manager: HTML: POST pdf file
    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



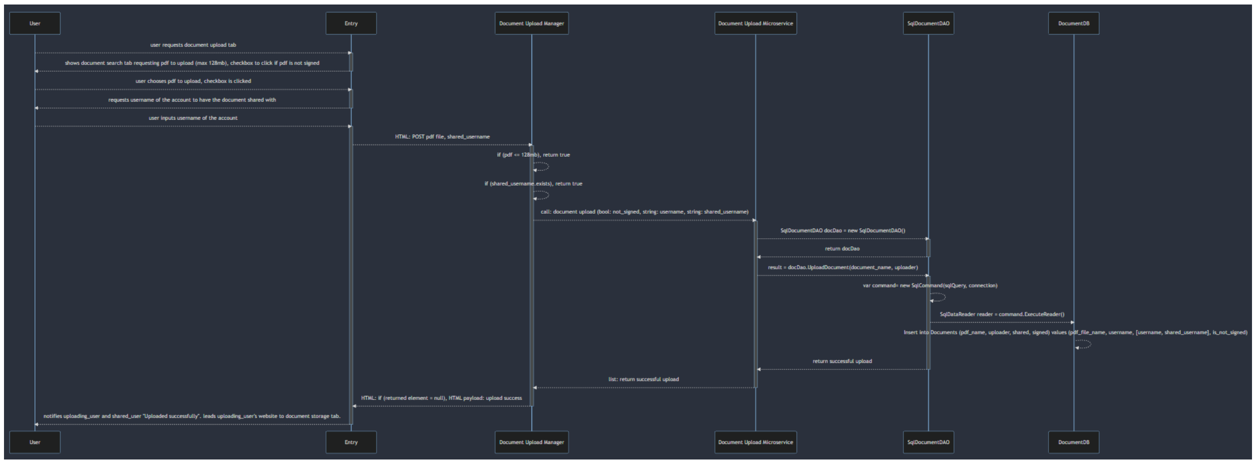
```
sequenceDiagram
    User-->>+Entry: user requests document upload tab
    Entry-->>-User: shows document search tab requesting pdf to upload (max 128mb), checkbox to click if pdf is not signed
    User-->>+Entry: user chooses pdf to upload, checkbox is clicked
    Entry-->>-User: requests username 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.
```