|  |  |
| --- | --- |
| SF Software  a Solution for the coding challenge  .Net core and React js | Abstract  In this document, a solution about how can we upload a file with subfiles between users has been shown and tried to have a reusable and scalable code.  Vahidalizadeh1990@gmail.com  Links:  [Visit SF Application](https://sharing-files-11306.web.app/)  [GitHub](https://github.com/Vahidalizadeh7070/SharingFiles) |

Contents

**Introduction**2

**Technologies**2

**Structure**2

.NetCore API2

React js3

Deployment3

**Architecture3**

Benefits4

**Algorithm**4

**Test**5

**Extra Ideas**5

**Introduction**

This document is about SF application and how it is implemented and what technologies have been used. The SF is software that provides you a platform to share your files through the application. First of all, the framework and 3rd libraries that have been used will be discussed, and then move forward to the architecture, structure, and algorithm which is the base of our application.

**Technologies**

Several frameworks and 3rd party libraries have been used in the SF software to implement a well-done application such as:

1. .Net core Web API: API provides the services that we need to use in the front-end section.
2. Entity framework Core: This is an ORM that enables to work with data.
3. SQL server: This is the main database that store all information that comes through the EF Core.
4. Asp.net Identity Core: This library is responsible for authentication and Authorization.
5. AutoMapper: This library is responsible for mapping the EF Core objects to the DTOs to use in the APIs.
6. JWT: This library provides a JWT token to use in the authentication process.
7. React Js: This JavaScript library is responsible for the front-end section and I use Hooks like useState, useEffect, useCallback, etc. to communicate with the APIs.

**Structure:**

In this section, the structure of the back-end and the front-end section will be discussed.

**.Net Core API:**

In the SF software, SOLID principles, OOP-like polymorphism, and some design patterns like Factory and repository patterns and dependency injection are implemented to develop more useable and scalable APIs. The core of the SF software is repository patterns. The DTO has been used to send a model to the repository and to transfer these DTOs to the main model, AutoMapper has been used. Also, the HttpClient has been used to provide the download files section. For instance, when you click on the download button, a request sends to the API and then the HttpClient sends the file back to the users. In this situation, a user does not have any access to the location of files.

**React Js:**

React Js as a library is picked up to make a component base front-end section and react router dom to redirect users between components. Furthermore, the UI is implemented by bootstrap 5.

**Deployment:**

Microsoft Azure as a service has been selected to host the APIs and migrate all of the models to the SQL server that is running on the Azure. However, there are a lot of ways to deploy the APIs and databases such as the Kubernetes and Docker. Furthermore, the front-end section is hosted on the Google firebase.

**Architecture:**

In this section, we will discuss how the SF works and demonstrate how can we share our files between users, and talk about the SF benefits.

How does it work?

1. Sign up in the SF software. You should use your email address and strong password.
2. Sign in with your email address and password and then you can have access to the dashboard.
3. In the dashboard of the SF, you can see all the files that you have uploaded. A file includes some properties like Title, File, Upload date, and user id. Each file contains subfiles that you can add after uploading a new file. (you can consider each file as a repository like GitHub and …).
4. There are 2 buttons in each of these files. One of them is responsible for deleting a file. Another is responsible for demonstrating the details of each file.
5. In the details section, several options provide for you to communicate with your files. The first option is the Download. If you click on the button, the file will be downloaded. Moreover, you can observe all of your subfiles that have been added to a file. Another one is the share button. If you click on the share button, you can share a file and all of its subfiles with another user.
6. Another section is Add file section. There is a plus button that you can use to add a new file.
7. Finally, there is an inbox in the top menu that able you to observe all of the files and subfiles that have been shared with you. Also, you can download the main file individually or download all subfiles that are related to the main file. Notice that all of the files will be closed after 1 day. If you need a file that is expired, you can ask users to reshare it for you.

**Benefits:**

There are a lot of ways to implement it, like considering a model to handle all of your files. But in that way, users should share those files individually and they should spend a lot of time sharing with other users. But here, a user just shares a file, then other users have access to the subfiles that are related to the main file. In addition, when we use the HttpClient to download a file, the main directory of a file is concealed from any users and the Httpclient will be responsible for having access to the directory of the file.

**Algorithm**

Add subfiles

Details

Remove from the dashboard

Upload file

Download by yourself

Delete file

Add to the dashboard

Dashboard

Sign Up and Sign in

Inbox

Download files and subfiles that a person shared for you

**Test**

The SF has been used 2 main types of testing. The first one is the End-to-End which all the sections such as front-end, back-end, database, DTOs and models, classes and functions, repositories and files, etc. will be tested. The second one is the XUnit test which is a unit test. The main section that has been considered for testing the SF is the controller. But other sections like repositories or extra classes and functions can be tested as a unit test. So in XUnit, a dummy data has been created to play a database role and then all repositories are invoked based on the controller that is going to test.

**Extra Idea**

These technologies and methods can add to the project to improve the life-cycle of the SF app. One of these is Kubernetes or Docker which able to migrates projects to the microservices architecture. Furthermore, CI/CD pipeline is helpful to accelerate the deployment section.