Date	Version	Description	Author
2018 Oct 24	Draft	Design document for Name Sorter	Charith Rodrigo

Assumptions

1. Person names in a text file.
2. A new line character will separate each name.
3. Each name can be consists of several words which can be separate using a single space.
4. Sorting should be ascending.
5. Authentication of the user happens in the base application, which use this service.

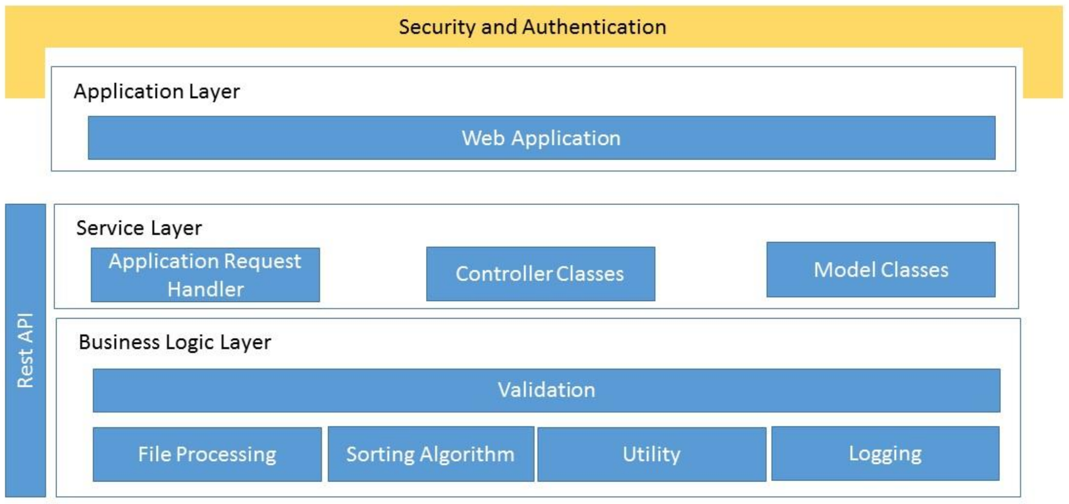
Architecture of the Solution

Scalability

The system will be developed in such a way so that it can be scale balance future loads by introducing new server to the business logic layer, and that will facilitate the user growth. Web API used in the business layer, which can be hosted separately.

Expandability

The application has the ability to responsive for the phones and tab as well without any modification. And we can extend the solution to have subscriptions for the service.



Security and Authentication
I have assumed this could be a pluggable solution to an existing web application which has authentication implemented. Having a encrypted authentication token to be passed through headers for the service layer which can be validated at the service layer will enhance the security further. And also introduction of a login page with .NET identity with necessary roles to users will have to attributed the controller methods to authorize.

Application Layer
A responsive html page will be the view of this application-allowing user to upload a text file, which has several names to sort. Request/Response handled using JQuery with json.

Service Layer
This will be the middle service layer that accept the request with the Text File and pass it through the request validation and number validations in the business layer. Web API has bean used as the endpoint.

Business Logic Layer
This layer will have the request validation and the sorting algorithms. In addition to the key validations, this will include a logger and common error codes and messages to enhance the user experience and maintainability.

Design approach for each requirement

1. Web form should provide an input where text file can be uploaded

   Design Approach
   - HTML5 page with 1 file input + button + notification div
   - Actions will be handled using JQuery (Modular pattern)
   - Page will be responsive using bootstrap

2. Name sorter
   Design Approach
   - Business logic layer will implemented as a .NET Web API service layer, which can hosted in either IIS or Cloud with https.
   - Converted the text content to object list, which can sorted using linq.
   - Sorting method is extendable but cannot modify.

4. Non Functional Requirements

- Single Responsibility Principle (SRP)
  - I have break down key responsibilities to different interfaces and classes

- Open-Closed Principle

    - Sorting will change in future but method should not change for the new requirement so I have separate interfaces for possible other sorting logics.
- Liskov Substitution Principle (LSP)
    - I haven't use explicit casting in any class.
- Interface Segregation Principle (ISP)
    - I have include interfaces for each key function adhering to ISP and SRP
- Dependency Inversion Principle (DIP)
    - I have use dependency injection with unity to avoid the dependency with the concrete object.
- Logger
    - Log4Net used to log the exceptions.
- Unit Testing
    - NUnit used for Unit Testing and separate test classes used for repository and business logic methods.

Deployment
Code is checked in to GitHub and created an Azure DevOps Pipeline, details as follows

Repository
https://github.com/charithGitHub/NameSort

Azure DevOps Pipeline
https://dev.azure.com/CodeAssessment/_git/NameSort