Project Documentation

Candidate Name: UVAANA FALICICA J

Candidate Email ID: UVAANA.J@itcinfotech.com

Batch Name: .Net Full Stack with I&F

Project Title: Digital Books

Date: 17/10/2022

Project Manager (s): Mr.Rampavan.S (Sr.Project Manager)

TABLE OF CONTENTS

TITLE PAGE NO

LIST OF FIGURES

CHAPTER 1 - INTRODUCTION

1.1 Objective 3

1.2 Chapter wise summary 4

CHAPTER 2 - Business Specification/Requirements

2.1 Architecture diagram 5

2.2 Product Deliverables 6

CHAPTER 3 - Functional Output/Requirements

3.1 Data Management 7

3.2 Authorization Levels 7

3.3 Testing 8

CHAPTER 4 - Tool Chain/Technical Coverage 9

CHAPTER 5 – Risk & issues

5.1 Bugs and Errors 15

5.2 Solution 16

CHAPTER 6 – Conclusion 17

CHAPTER-1

INTRODUCTION

In the fullest sense of the term, a digital book is a file containing **text and images** suitable for distributing **electronically and displaying on screens** in a similar way to a printed book. It is possible to create a **digital book by converting a printer's source files** or by using a database or a collection of text files that were not created specifically for printing. Most Digital books are **distributed via the Internet as downloadable files that can be read offline**, as live **Web pages that must be read online, or as cached Web pages that a browser can view offline.** A retailer may provide customers with information about, buy e-books, but they will be required to **download the e-books directly from the publisher's or distributor's servers**, which may be on another continent. Despite being hidden from the customer, this distinction is very important for businesses. In spite of the fact that customers may find them listed for sale or loan in a variety of places, the e-book files are stored and managed in a single place (or very few places). A closed, proprietary distribution system would result in all Digital books being purchased or borrowed directly from a small number of owners of e-book files without this distinction.

1.1 Objective

- Creating a web app app using React Js for a Digital Books store, so that the users can order or subscribe the books as per their preference.
- The guest user can login or register from the home page and able to search the books available in the app from the home page.
- The logging part has two roles. Author and Reader. The Author can able to add, edit and block books.
- The reader can able to search the books and can subscribe or order the books which is not in blocked state.

1.2 Chapter Wise Summary:

- Chapter 1: This section contains **the project goal, the tasks had completed**, and a clear understanding of the modules and specifications.

- Chapter 2: This chapter includes the **complete needs of the project which to be implemented during the working.**

- Chapter 3: It offers a functional requirements of the project

- Chapter 4: This part clearly explains the Tools and IDE's used during the project implementation.

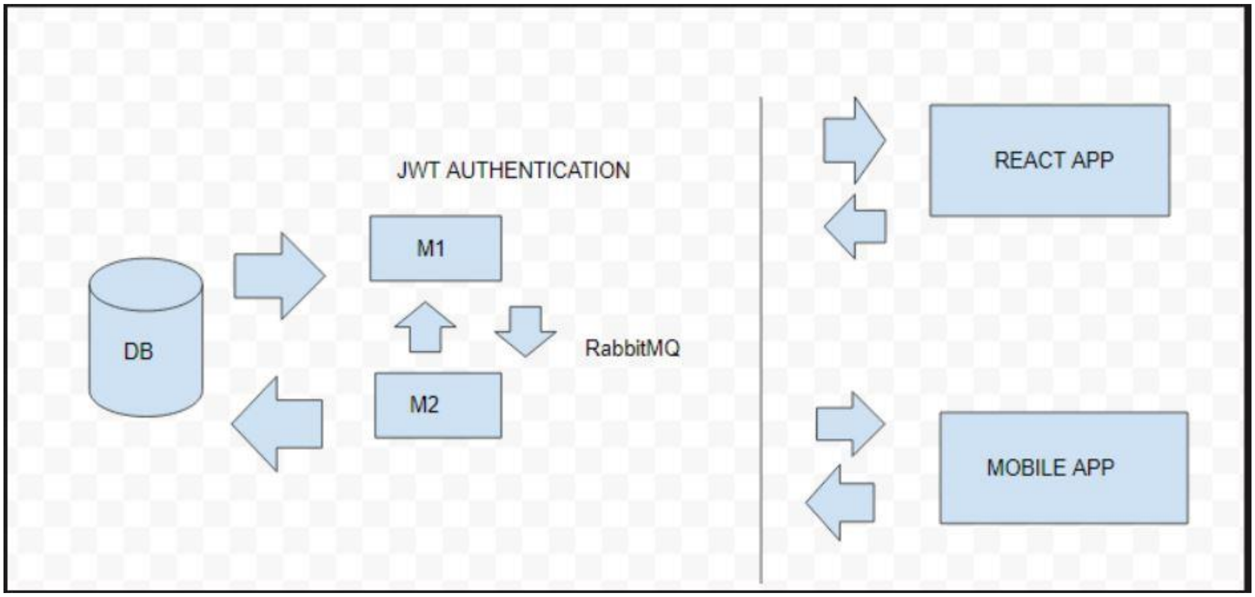
- Chapter 5: It describes the **risks faced during the project implementation** and the solution for the bugs and errors

- Chapter 6: The final chapter is going to give an conclusion of the project implemented.

CHAPTER 2

BUSINESS SPECIFICATION/REQUIREMENTS

2.1 Architecture diagram



In this project, there are **two microservices Users and Books.** The data of both microservices are stored in a single database. Using the **Mass Transit rabbitmq** the two microservices can **communicate with each other,** it acts as **a message broker** able to pass the message from one end to another end. The Users having the same **authentication levels**, perform the operations(such as CRUD) in Books Microservice. React Js is used to design the frontend UI part of the application. Using the **localhost server we can able to integrate the reactjs with c# code.** So that we can able to run the react app and can perform the operations which are defined. Using Flutter we can able to create a mobile app and integrate code from the backend, and run the application.

2.2 Product Deliverables

Roles: Guest User, Author, Reader.

Freatures of Guest User

- The Homepage consists of **login and signup buttons**. When the guest user comes and signup with the email Id and password, they can choose any one role **either an Author or a Reader.**

- After signing up a **unique token will be generated** from the backend for each user who's signing up. As guest users after registering they could able **to log in to the page.**

- And from the homepage, the guest user can able to **search the books using the title, category, author, and price.** So that the entire details of the book should be displayed.

Features of Author

- After login into the role of the author. The **author has permission to create new books, edit those created books, and Delete books.** The author can able to view all the created books and log off from their page.

Features of Reader

- As a reader role, the reader should be able to **search the available books and they can order or subscribe to the available books.**

- After ordering or subscribing to the books, using the **JWT authentication a unique token** will be generated for the subscription or order placed.

- The reader can read only the subscribed books. As a reader, we can able to see all the subscribed or ordered books and can view invoices using a subscription or order id. Within 24 hours the reader can able to cancel the order or subscription placed.

CHAPTER 3

FUNCTIONAL OUTPUT/REQUIREMENTS

3.1 Data Management

- The data management of this project can be maintained using the **Azure SQL database, Azure Blob Storage.**
- The Azure storage can able **to manage, and load balance the data** we are storing. Due to storing the data in Azure, the data can **be recovered even after the disaster.**
- It consists of many **data centers so that the resources stored in the Azure** SQL can be shared between **nearby regions.** So it's very easy to retrieve the data automatically.

3.2 Authorization Levels

- Using Authorization we can able to **give permissions to the guest user, author, and reader.** So that the author can only able to search and perform CRUD operations.
- The reader can **able to search and order the books, but they can't able to perform the CRUD operations** because of authorization permissions.
- And the guest **user doesn't have permission to perform the author and reader role.**

3.3 Testing

- Swagger is an **open API testing tool,** it is used to validate the API. By installing the packages of swagger, we can able to import and use them for testing purposes.

- With a **single button click, we can able to validate the API** and can test whether it's working fine or not.
- And also we can able to give the data and test whether it's stored in the backend or not. **Also GET, POST, PUT and DELETE testing parts can be done**.
- And using POSTMAN we can also test the API and get the data. And from the generated token from **the JWT authentication**, we can able to give authorization permissions to the users.
- Here we can able to create our API links using key and value.
- The only difference between swagger and postman is, **swagger should be imported into code and the testing can be done, but a postman is a web-based tool so the request can be sent through the internet.**

CHAPTER 4

Tool Chain/Technical Coverage

Visual Studio

- The Microsoft Visual Studio **Integrated Development Environment (IDE)** facilitates the development of GUI applications, console applications, Web applications, web applications, mobile applications, cloud services, and cloud computing applications.
- You can create native and managed code with the help of this IDE. In addition to Windows Store, Microsoft Silverlight, and Windows API, it also uses the various software development platforms of Microsoft.
- You can use it to write code in many languages, **including C#, C++, VB(Visual Basic), Python, JavaScript, and many more. A total of 36 languages** can be used with it. The software is available for both Windows and Mac. Here the API is developed using .NET core.

Visual Studio Code

- The Visual Studio Code source code editor is free, lightweight, and powerful and available for **Windows, macOS, Linux, and Raspberry Pi OS.**
- The platform supports **JavaScript, TypeScript, and Node.js natively and offers a wide array of extensions (such as C++, C#, Java, Python, PHP, and Go)** for various programming languages, runtimes (like .NET and Unity), environments (like Docker and Kubernetes), and clouds (like Amazon Web Services, Microsoft Azure, and Google Cloud Platform).

ASP.NET Core Web API

ASP.NET Web API is **a framework for building HTTP services that can be accessed from any client including browsers and mobile devices**. It is an ideal platform for building RESTful applications on the . NET Framework.

React Js

React.js is a front-end JavaScript framework **developed by Facebook.** To build composable user interfaces predictably and efficiently using declarative code, we use React. It's an **open-source and component-based framework** responsible for creating the application's view layer.

- ReactJs follows the **Model View Controller (MVC)** architecture, and the view layer is accountable for handling mobile and web apps.
- React is famous for building **single-page applications and mobile apps.**

Swagger UI

- *Swagger is a set of **open source tools for writing REST-based APIs**.* It simplifies the process of writing APIs by notches, specifying the standards & providing the tools required to write beautiful, safe, performant & scalable APIs. When I had to write an API from scratch.

- Well, like there were so many things to get straight, I would go through different articles online. Go through docs **of APIs exposed by Twitter, Facebook & stuff.** Try to get an idea into the standards they were complying to. Hell, It was a time-consuming process.

- Here is where swagger fits in. As I've stated before, it standardizes the whole process of writing APIs. Helps us figure things out which should ideally be spotted in the initial phases of writing software, help us weed them out, saving tons of time of code refactorization.

Postman

- A postman is a **web-based tool,** so users will be able to **access and work with API's anywhere they can get an internet connection.**
- For new users, Postman also provides thorough documentation and web tutorials that make it easy to learn to use the tool.
- Additionally, Postman provides sharing features that make it simple to share **HTTP calls with other members of an organization.**

MassTransit

- MassTransit is a free, open-source, distributed application framework for .NET applications. **It abstracts away the underlying logic required to work with message brokers, such as RabbitMQ, making it easier to create message-based, loosely coupled applications**.
- There are a few benefits to choosing to use a library such as MassTransit, instead of working with the native message broker library.
- Firstly, **by abstracting the underlying message broker logic, we can work with multiple message brokers**, without having to completely rewrite our code.
- This allows us to work with something such as the InMemory transport when working locally, then when deploying our code, use another transport such as Azure Service Bus or Amazon Simple Queue Service.

RabbitMQ

- RabbitMQ is **lightweight and easy to deploy on premises and in the cloud.** It supports multiple messaging protocols. RabbitMQ can be deployed in distributed and federated configurations to meet high-scale, high-availability requirements.

- RabbitMQ runs on many operating systems and cloud environments, and provides a wide range of developer tools for most popular languages.

Azure Blob Storage

- Blob storage is built from the ground up to support the scale, **security, and availability needs of mobile, web, and cloud-native application developers.**

- Use it as a cornerstone for **serverless architectures such as Azure Functions.**

- Blob storage supports the most popular development frameworks, including **Java, .NET, Python, and Node.js,** and is the only cloud storage service that offers a premium, SSD-based object storage tier for low-latency and interactive scenarios.

Azure SQL DB

- Azure SQL Database is a fully managed **platform as a service (PaaS)** database engine that handles most of the database management functions such as **upgrading, patching, backups, and monitoring** without user involvement.

- Azure SQL Database is always running on the latest stable version of the SQL Server database engine and patched OS with **99.99% availability**

- Azure SQL Database is based on the latest stable version of the Microsoft SQL Server database engine.
- You can use advanced query processing features, such as high-performance in-memory technologies and intelligent query processing.
- In fact, the newest capabilities of SQL Server are released first to Azure SQL Database, and then to SQL Server itself.
- You get the newest SQL Server capabilities with no overhead for patching or upgrading, tested across millions of databases.

Azure VM

- An Azure virtual machine is an on-demand, **scalable computer resource that is available in Azure.**
- Virtual machines are generally used to host applications when the customer requires more control over the computing environment than what is offered by other compute resources.
- When you leverage a virtual machine to host your application, you get the flexibility of virtualization without the need to buy or maintain any underlying physical hardware.

Bootstrap

- Bootstrap is a **free, open source front-end** development framework for the creation of websites and web apps.

- Designed to enable **responsive** development of **mobile-first** websites, Bootstrap provides a collection of syntax for template designs.

- As a framework, Bootstrap includes the basics for responsive web development, so developers only need to insert the code into a pre-defined grid system.

- The **Bootstrap framework** is built on Hypertext Markup Language **(HTML),** cascading style sheets **(CSS)** and **JavaScript**.

- Web developers using Bootstrap can build websites much faster without spending time worrying about basic commands and functions.

CHAPTER 5

Risk & issues

5.1 Bugs and Errors

1. While trying to connect the Azure SQL server from the local server it will generate an error of **Firewall rules is wrong.**

2. While we are trying to extract the tables from the database as models in ASP.NET we will get an error as **Build Failed**

3. The connection strings present in package.json throw an error while executing swagger testing "**the instance of the SQL server you attempted to connect to requires encryption but this machine does not support it".**

4. When we try to hit the button from the front after integrating the backend with UI, it will throw an exception, **Not Found Exception**

5. After creating the CRUD operations in API and React Js, after integration, **there will be no UI can be found after running react app**

5.2 Solution

1. Azure SQL Server, in the networking part we **should add the Firewall Rules.** Then try to log in and connect it from the local DB and check whether **the IPv4 address** is matching the same firewall rule added.

2. This error is because we didn't install the dependencies to import the models from the database. So we should install the packages **Microsoft.EntityFramework.Core, Microsoft.EntityFramework.Core.Tools, Microsoft.EntityFramework.Core.SqlServer.** Then run the command to import models from the database.

3. The encryption present in Azure SQL connection strings asks us permission always because it's encrypted. So make **the encrypt=" False".** So that we can able to store anything in the database from the front or backend.

4. While creating a database in the ID column make an **Identity specification ='yes'**

5. API gets failed due to **the 'CORS'** issue which happens due to both applications running on different ports. **In the 'Program. cs' file has to enable the Cors policy.**

CHAPTER 6

Conclusion

Finally, the conclusion of the project is to understand the API and other tools in a great manner. And should implement and perform testing using swagger, and postman to check whether the API is working. Even we can implement the same project using MVC for better understanding.