**“*Let’s Catch Up*” – An event organizing and planning website**

A picture containing calendar

Description automatically generated

**Software Project Final Report**

Biswadeep Mazumder

2860920

12th December 2022

Table of Contents:

**1. Introduction**

1.1. Purpose and Scope

1.2. Product Overview (including capabilities, scenarios for using the product, etc.)

1.3. Structure of the Document

1.4. Terms, Acronyms, and Abbreviations

**2. Project Management Plan**

2.1. Project Organization

2.2. Lifecycle Model Used

2.3. Risk Analysis

2.4. Hardware and Software Resource Requirements

2.5. Deliverables and schedule

**3. Requirement Specifications**

3.1. Stakeholders for the system

3.2. Use cases

3.2.1. Graphic use case model

3.2.2. Textual Description for each use case

3.3. Rationale for your use case model

3.4. Non-functional requirements

**4. Architecture**

4.1. Architectural style(s) used

4.2. Architectural model (includes components and their interactions)

4.3. Technology, software, and hardware used

4.4. Rationale for your architectural style and model

**5. Design**

5.1. User Interface design

5.2. Components design (static and dynamic models of each component)

5.3. Database design

5.4. Rationale for your detailed design models

5.5. Traceability from requirements to detailed design models

**6. Test Management**

6.1. A complete list of system test cases

6.2. Techniques used for test case generation

**7. Conclusions**

7.1. Outcomes of the project (are all goals achieved?)

7.2. Lessons learned

7.3. Future development

1. **Introduction**
   1. **Purpose and scope**

A website, which will allow the students and other members of Cleveland State University to post about any events so that other students can get aware of the same and can register themselves to attend the event of the Cleveland State University.

* 1. **Product Capabilities**

This is an event organizing and participation website which will allow the attending members to chat among themselves to further plan for the event. This website is not limited only for any organization specific events, but also, can arrange a group for other activities like going to the gym, going to movies or even organizing a group study session with people of similar interest.

* 1. **Structure of Document**

This document provides the details of the application, how the planning is made to overcome the problem, architectural style, functionalities, Sequence diagrams, EF diagram, lifecycle model, and UML model. This will help an individual to understand the application to the depth.

* 1. **Terms, Acronyms, Abbreviations**

**User** : Can be both the HOST and ATTENDEES.

**Host** : Who hosts an event.

**Attendee** : Who attends an event.

**Event** : The post about which people plan to go together.

1. **Project Management Plan**
   1. **Life Cycle Model Used**

The waterfall methodology is a linear project management approach, where stakeholder and customer requirements are gathered at the beginning of the project, and then a sequential project plan is created to accommodate those requirements. The waterfall model is so named because each phase of the project cascades into the next, following steadily down like a waterfall.

This model is preferred over the other SDLC models as for this project the requirements are firmly mentioned with no further changes soon. This also helped in preparing a detailed development, testing and delivery plan for this class project

Though this project has lots of potential for more addition of functionalities depending upon the market requirements.

Diagram

Description automatically generatedDepending upon future requirements and continuous maintenance of the project simultaneously, a different SDLC method can be used.

* 1. **Risk Analysis**

As we have mostly used open source public libraries along with some 3rd party plugins like Axios(API routing management), MobX(State management) and SignalR(chat functionality),Semantic-UI-react(UI design) and fly-io(deployment server). In future these libraries may be updated, and some existing functionalities may get obsolete, or these products may end their free services, then this app may fail.

Apart from these above mentioned Plugins, there is no such risks.

* 1. **Software Requirements**

The application has been separated into FrontEnd(UI) and BackEnd.

This segregation helped to develop both the units parallelly and an error in any one of the units would not hamper the progress of the other unit.

Back End Softwares Used:

1. Microsoft Dotnet Core Entity Framework v6.0
2. SQL lite DB for development
3. PostGre SQL for production

Front End Software Used:

1. React v17.0
2. MobX
3. Axios
4. SignalR
5. Semantic-UI-react
   1. **Deliverables and Schedule**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl** | **Main Task** | **Sub Task** | **Status** |
| 1 | Requirements and Gathering | Initial Discussion | Done |
| 2 | Finalize the scope | Done |
| 3 | Tasks Assignment | Done |
| 4 | Design Finalization | Done |
| 5 | Documentation | project plan | Done |
| 6 | Requirement Documentation | Done |
| 7 | Test Plan Documentation | Done |
| 8 | Design Documentation | Done |
| 9 | Development | Development Environment Setup | Done |
| 10 | Front End Development | Done |
| 11 | Back End Development | Done |
| 12 | Unit Testing Front End | Done |
| 13 | Unit Testing back End | Done |
| 14 | Testing | Testing front end | Done |
| 15 | Testing Back end | Done |
| 16 | Website Verification | Done |
| 17 | Final Submission | Project report generation | Done |
| 18 | Project Demonstration | Done |
| 19 | Project Submission | Done |

1. **Requirement Specification**
   1. Stake holders for the system

**Internal Stake Holders** **:** Biswadeep Mazumder

**External Stake Holders :** Cleveland state university students

* 1. **Use Cases**

1. User logins into the Website.
2. User (Author) Creates/Modify/Delete and Event.
3. User (Viewer) View, Jon/Leave and Event.
4. User Can Modify their Profile details
5. User can follow other users depending upon the user’s choice.
   * 1. **Use case model**

Diagram

Description automatically generated

* + 1. **Use case description**

1. User logins into the Website :

*User access the website to view or host events.*

1. User (Author) Creates/Modify/Delete and Event :

*User creates/posts an event which others can join.*

1. User (Viewer) View, Join/Leave and Event.

*All users apart from the host can view, join and leave the event according to the feasibility.*

1. User Can Modify their Profile details:

*A particular user can update their profile details.*

1. User can follow other users depending upon the user’s choice.

*This feature allows one user to follow the other users activity.*

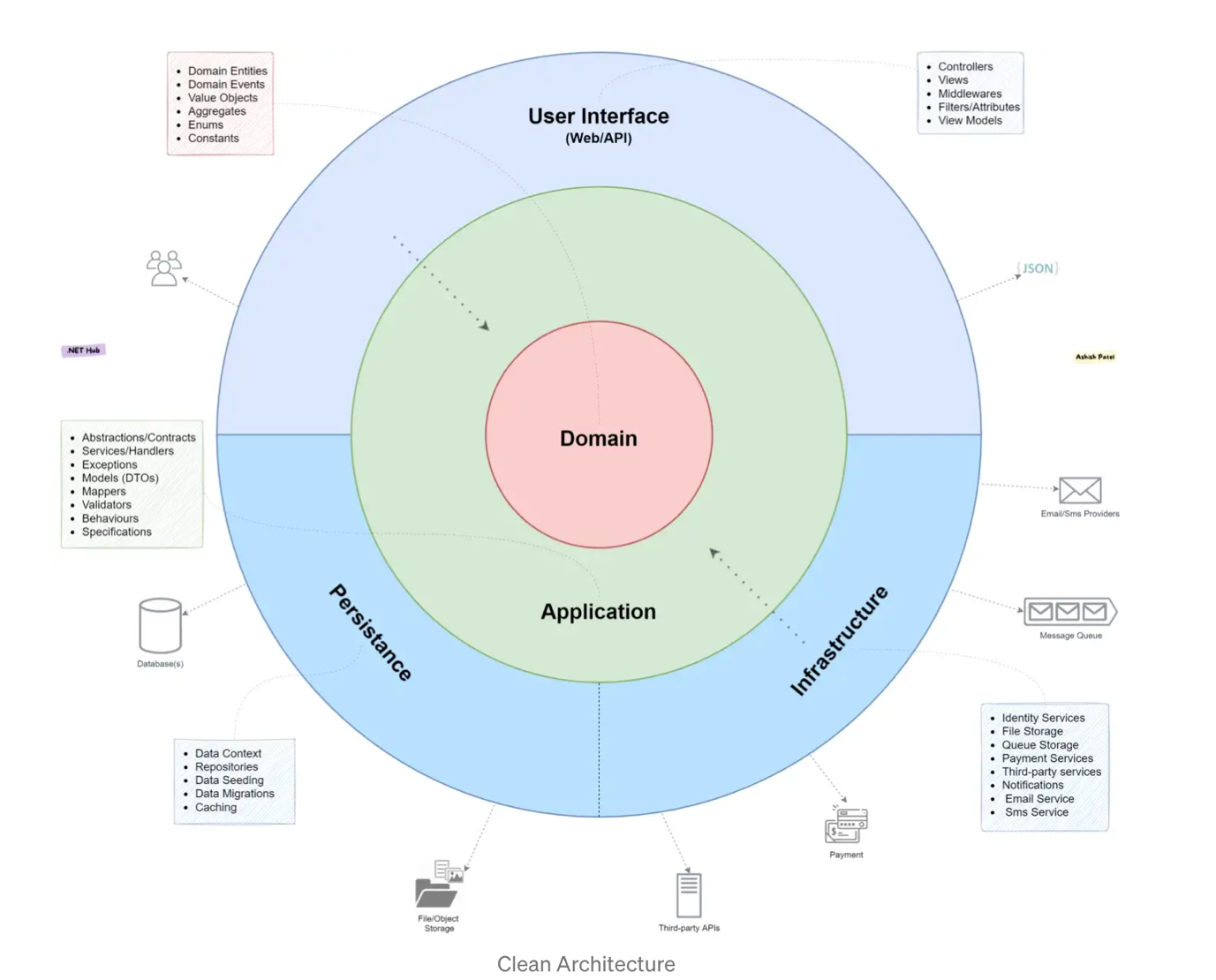
* 1. **Non Functional Requirements**
  2. This app shoul be available 24 X 7.
  3. This app should be able to respond fast.
  4. This app should be able to handle exta load during special events in the university.

1. **Architecture**
   1. **Architectural Style**

This application is developed using the CLEAN architecture. Clean architecture has a domain layer, Application Layer, Infrastructure Layer, and Framework Layer. The domain and application layer are always the center of the design and are known as the core of the system. The core will be independent of the data access and infrastructure concerns.

Benefit of this style:

1. Independent of frameworks
2. Testable
3. Independent of UI
4. Independent of database
5. Independent of anything external



All the dependencies of the application are Independent/Inwards, and the Core system has no dependencies on any other layer of the system. So, in the future, if we want to change the UI/ OR framework of the system, we can do it easily because all the other dependencies of the system are not dependent on the core of the system.

* 1. **Architectural Model**

Diagram

Description automatically generated

* 1. **Technologies, Hardware and software used for development**

The application is developed on

1. Operating system: MAC OS
2. IDE: Visual Studio Code
3. Framework: Dotnet Entity Framework Core v6.0, React v17
4. Database: Sql Lite(development), PostgreSql(production)

The application has been separated into FrontEnd(UI) and BackEnd.

This segregation helped to develop both the units parallelly and an error in any one of the units would not hamper the progress of the other unit.

Back End Softwares Used:

1. Microsoft Dotnet Core Entity Framework v6.0
2. SQL lite DB for development
3. PostGre SQL for production

Front End Software Used:

* + 1. React v17.0
    2. MobX
    3. Axios
  1. SignalR
  2. Semantic-UI-react
  3. **Why this architectural style is used**

This style is useful to segregate the Logical, business and implementation layer. This helped me to achieve the separation for developing the UI and backend separately. This helped me to achieve my goal where I acted as a Full Stack Developer and any bug was easily identified and helped me to complete the project on time as I have focused only on the units I was developing at that moment

1. **Design**
   1. **User Interface Design**
2. Login Page

Graphical user interface, application

Description automatically generated

1. Sign in Page

Graphical user interface, application

Description automatically generated

1. Graphical user interface, application

   Description automatically generatedEvent Creation Page
2. Event Join/Leave Page

Graphical user interface, application, website

Description automatically generatedGraphical user interface, website

Description automatically generated

1. User Home Page

Graphical user interface, application

Description automatically generated

1. Chat in an event page

Graphical user interface, application

Description automatically generated

1. Graphical user interface, text, application, email

   Description automatically generatedEdit Event
2. Follow / Un-follow User

Graphical user interface, application

Description automatically generated

1. Graphical user interface, application

   Description automatically generatedUser Profile Edit
   1. **Components Design**

Diagram

Description automatically generated

* 1. **Database Design and Why I chose this design**

For designing the database, I have used a code first approach where I have defined the table structures in class file and with the use of dot net core entity framework migrations, these tables have been created on the localhost server

A picture containing text

Description automatically generated

1. **Test management**
   1. **List of test cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **TEST INPUT** | **EXPECTED OUTPUT** | **DESCRIPTION** |
| 1 | SIGN UP | PROFILE CREATED SUCCESSFULLY | ALLOWS FIRST TIME USERS TO SIGN UP TO USE THE APPLICATION |
| 2 | SIGN IN | LOGS IN TO THE APPLICATION | ALLOWS USER TO ACESS THE APPLICATION AND USE THE FEATURES |
| 3 | PROFILE PHOTO UPLOAD | UPLOADS A PICTURE FOR PROFILE | ALLOWS USER TO UPLOAD A PROFILE PICTURE FOR HIS/HER ACOUNT |
| 4 | CREATE EVENT | EVENT CREATED ON A DATE | ALLOWS USER TO CREATE AN POST FOR AN EVENT |
| 5 | MODIFY EVENT | EVENT DETAILS UPDATED | ALLOWS USER TO MODIFY THE DETAILS FOR THE EVENT HE/SHE CREATED |
| 6 | JOIN EVENT | REGISTER FOR A SPECIFIC EVENT | ALLOWS USER TO REGISTER FOR AN EVENT |
| 7 | LEAVE EVENT | DE REGISTER FOR A SPECIFIC EVENT | ALLOWS USER TO DE REGISTER HIM/HER FOR THE EVENT |
| 9 | FOLLOW USER | FOLLOWER/FOLLOWING LIST UPDATED | ALLOWS USER TO FOLLOW AN UN FOLLOW ANY USERS |
| 10 | POST ON ANY EVENT | POSTS SHOWS ON EVENT PAGE | ALLOWS USER TO COMMUNICATE FOR EXTRA DETAILS ABOUT THE EVENT |

* 1. **Technique used for test case generation**

Test cases have been generated based upon the requirement scenarios. These test cases are manually defined and has been tested manually.

1. **Conclusion**
   1. **Outcomes of the Project**

This project has been completed with some minor issues arising because of the constant updates from the 3rd party plugins. This project backend is now running on local host 5000 and the front end on local host 3000. This has been an ambitious project and a lot of days has been spent on understanding the architecture and the software engineering principles.

Git hub url: <https://github.com/BiswadeepMazumder/Reactivities>

* 1. **Lessons learned**

I have taken this project to learn about the latest technology stacks and to learn how a full stack developer works independently. I have learnt how to prepare the requirements, design and test documents.

I am thankful to my professor who gave me an opportunity.

Also I have used Udemy Course <https://www.udemy.com/course/complete-guide-to-building-an-app-with-net-core-and-react/> by Neil Cumming.

A lot of references has been used from the code Neil provided and collaboratively from the other learners.

Collection of data and manually filling the database was hectic, so to save time and focus more on the projects I have used the same data provided in the course.

* 1. **Future Development**

Currently this project is hosted locally. And if run on a different system, it may not work properly.

So to fix this a dockerized version of the project is to be developed so that is remains independent of the other environments and for more protection of the user data, a secure database like Azure database to be used.

Also this project has a huge potential to be independently launched as a product where it can be sold to Organizations and they can use it internally for managing the different events organized by them or even the individuals in those organizations stay aware of the different events.

1. **References**
   * 1. Udemy Course <https://www.udemy.com/course/complete-guide-to-building-an-app-with-net-core-and-react/> by Neil Cumming
     2. Images are gathered from :

https://blog.mindmanager.com/waterfall-methodology/

<https://medium.com/dotnet-hub/clean-architecture>

* + 1. UML tool:

Microsoft visio

* + 1. Dummy Data from :

Course <https://www.udemy.com/course/complete-guide-to-building-an-app-with-net-core-and-react/>