HAWASSA UNIVERSITY

INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

FINAL PROJECT OF DEGREE PROGRAM

TITLE: JOURNAL SYSTEM

Prepared by:

- **Elihu Getachew** **Tech/0610/10**
- **Eyob Bezu** **Tech/0682/10**
- **Matios Sisay** **Tech/1104/10**

Advisor: - Mr. Muluneh Hailu

Final Project Submitted to the Department of Electrical and Computer Engineering Computer Stream

Date: March 17, 2022

Hawassa, Ethiopia

The paper has been submitted for examination with my approval as university advisor.

____

Mr. Muluneh Hailu

Examining committee members' name, date and signature

Chairman	Date	Signature
Examiner 1	Date	Signature
Examiner 2	Date	Signature
Facilitator	Date	Signature

Acknowledgment

First of all, we would like to thank Almighty God for the strength, he has given us throughout our life and this project, nothing could happen without the help of God. Foremost, we would like to express our sincere gratitude to our advisor **Mr. Muluneh Hailu** for the continuous support on our undergraduate industrial project, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped us in all the time of writing of this document. We could not have imagined having a better advisor and mentor for our industrial project. In addition, we have unforgettable thanks for our coordinator of departments who provide us the information we need to build the system.

Finally, we would like to extend special thanks to our families for their unwavering confidence, encouragement and emotional support which was instrumental to the successful completion of our bachelor degree coursework.

List of Figures

List of Tables

Table of Contents

CHAPTER ONE

INTRODUCTION

1.1 Background

An academic journal, also known as a scholarly journal, is a periodical publication that publishes academic works in a specific discipline. Academic journals provide permanent and open forums for the presentation, scrutiny, and discussion of research findings. Typically, they are peer-reviewed or refereed. Articles presenting original research, review articles, or book reviews are common types of content. They have played a huge role in the further improvement of society starting from the late seventeenth century, by providing a platform where knowledge can be incrementally increased.

In recent times the scholarly communication process as a whole has been subject to profound transformative pressures, driven principally by technology and economics. At the same time, though, the underlying needs of researchers remain largely unchanged changes can be considered under three headings. These are changes to the publishing market (like globalization and the growth of emerging regions), changes to the way researches are conducted (e.g. use of networks; growth of data intensive and data-driven science; globalization of research) and changes to public policy. This makes sure that

The Internet has revolutionized the production of, and access to, academic journals, aiding the aforementioned transformation making their contents available online via services subscribed to by academic libraries. Individual articles are subject-indexed in databases such as Google Scholar. Some of the smallest, most specialized journals are prepared in-house, by an academic department, and published only online – such form of publication has sometimes been in the blog format though some, like the open access journal Internet Archaeology, use the medium to embed searchable datasets, 3D models, and interactive mapping. Currently, there is a movement in higher education encouraging open access, either via self-archiving, whereby the author deposits a paper in a disciplinary or institutional repository where it can be searched for and read, or via publishing it in a free open access journal, which does not charge for subscriptions, being either subsidized or financed by a publication fee. The existence of online publishing platforms facilitates this process.

1.2 Statement of Problem

Our country Ethiopia has long been hailed as the source of a vast amount of knowledge and history in ancient times. Despite this glowing past our recent reputation is nowhere near that in a lot of aspects including the education system. One of the reasons this happened is because we as a society have a very low culture of passing on knowledge and building on it. The proposed system makes sure that high quality researches are critically reviewed raising their standards and afterwards accessible to anyone interested.

1.3 Objective

1.3.1 General Objective

The general objective for the development of this Academic Journaling System is to enable Ethiopian scholars to have access to high quality peer reviewed articles, and also to enable researchers across the globe have access to articles published by Ethiopian researchers without any form of restriction.

1.3.2 Specific Objectives

In order to achieve the general objective, the following specific objectives are identified.

- Provide an online file tracking system.
- Provide online submission and management of all content related to a researcher.
- Provide a secure user and file access management.
- To implement a clear research review process.
- Provide a vast repository that is able to hold various researches.

1.4 Scope

The scope of this project is to develop an Academic Journaling System and provide environment through which researchers can publish their work and get critically reviewed. This project aims to provide online research curating system with information about how far along the review process a certain research is and give access for uploading new studies, view both published and pending works.

Limitation of the project include sandboxed environment was used to develop the system.

1.5 Significance of the Study

The proposed system will provide an easy and convenient way to facilitate the review and publishing of academic researchers conducted by Ethiopian researchers. By doing so it will create a system where researchers can ensure that they can actively track and monitor the stage of reviewing process their work is at. This means that they can hold publishers accountable to publishing their work in a timely manner. In addition to this it'll create a rich community of academics that have a platform to get their work out there. Articles in scientific journals can be used in research and higher education. Scientific articles allow researchers to keep up to date with the developments of their field and direct their own research. This creates a much-needed system which is severely lacking in Ethiopia, with the availability of readily available research material laying the groundwork for further improvement it'll bring about significant growth in the country.

1.6 Feasibility Study

1.6.1. Economic Feasibility

Here is the economic feasibility of developmental and operating costs.

1.6.1.1 Developmental cost

The project will be developed by using easily found software and open-source libraries. We will also use the help of the internet for most of the demands we have. We plan to use our personal computers regarding hard ware tools.so there would be no issues regarding hardware tools.

Another factor that could be considered as a developmental cost is time limit but since this project is our final project, we will devote all of our time and energy to it. We say that the developmental cost is trivial. Hence the proposed system is feasible.

1.6.1.2 Operational Cost

Operational costs may include that student that choose to use the website will be charged for internet usage or mobile data usage (by the ISP of course)

Accordingly, we could say that the proposed system is economically feasible.

1.6.2. Technical Feasibility

• Input: The main input for the proposed system will be basically the Papers.

• Output: The output for this proposed system will be the published journals.

• Software: Software tools are available and found without difficulty.

• Members (personnel): We are working in group of three individuals. We believe that working both separately by dividing tasks and also working together as a team will help us to accomplish the proposed system consequently.

And so, the proposed system can be developed effectively using these existing technologies which makes it technically feasible.

Table 1 Workplan

Month	April				May				June	
Week	1	2	3	4	1	2	3	4	1	2
Literature Review	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Write A Proposal	▨									
Define Requirements		▨	▨							
Design		▨	▨							
Development			▨	▨	▨	▨	▨			
Testing								▨		
Deployment									▨	
Conclusion And Recommendation										▨

1.9 Cost Estimation

Fortunately, most of the tools that are needed are already available for free. We've described the cost needed to develop our system in the below table cost estimation.

Table 2 Budget Breakdown

Items	**Quantity**	**Cost (ETB)**
Web Hosting Per Year	1	10,000
Laptop Computers	2	100,000
Internet Package (Quarterly)	1	2,100
TOTAL		**112,100**

CHAPTER TWO

LITERATURE REVIEW

The STM, reported an overview of scientific and scholarly journal publishing. Journals form a core part of the process of scholarly communication and are an integral part of scientific research itself. Journals do not just disseminate information, they also provide a mechanism for the registration of the author’s precedence; maintain quality through peer review and provide a fixed archival version for future reference. They also provide an important way for scientists to navigate the ever-increasing volume of published material. The proportion of electronic-only journal subscriptions has risen sharply, partly driven by adoption of discounted journal bundles. social media do seem likely to become more important given the rapid growth in membership of the newer scientific social networks (Academia, Mendeley, ResearchGate), trends in general population, and the integration of social features into publishing platforms and other software. The adoption of mobile computing devices in the general population has been, and continues to be extremely rapid, even by the standards of the internet age. There are important technology choices to be made for publishers in addressing the overlapping issues of mobile access that go beyond the scope of this report. Our journal website platform will offer a mobile-optimized interface by using responsive or adaptive design. [1]

ARNO, the ARNO project—Academic Research in the Netherlands Online—has developed software to support the implementation of institutional repositories and link them to distributed repositories worldwide. Project participants include the University of Amsterdam, Tilburg University, and the University of Twente. Released for public use in December 2003, the ARNO system has been in use at the universities of Amsterdam, Maastricht, Rotterdam, Tilburg, and Twente. It is designed to provide a flexible tool for creating, managing, and exposing OAI-compliant archives and repositories. The system supports the centralized creation and administration of repository content, as well as end-user submission. While ARNO offers considerable flexibility as a content management tool, the system does not provide an end-user interface with end-user search capabilities. Our journal repository will be accessible to end users with a broad aspect of functionalities. [2]

CHAPTER THREE

METHODOLOGY

3.1 Methodology Used

Methodology specifies the method and technology used to develop the software system such as, the methods used to gather data, approach used to design the software system, software and hardware requirements used to implement the system. The main goal of this study is to develop NFT marketplace on Ethereum with polygon and next.js by using agile software development method.

Agile software development method

Agile software development refers to software development methodologies centered on the idea of iterative development. The ultimate value in agile development is that is enables teams to deliver value faster, with greater quality and predictability and greater aptitude to respond to change. The benefits of agile are:

- Highly responsive to customers development requests
- Faster time to market
- Project visibility and transparency
- Risk reduction

Agile helps teams deliver high quality software on time and on budget.

ReactJS

React.js is a JavaScript library that was created by Facebook. React is a library for building composable user interfaces. It encourages the creation of reusable UI components, which present data that changes over time. React implements one-way reactive data flow, which reduces the boilerplate and is easier to reason about than traditional data binding. React "reacts" to state changes in your components quickly and automatically to re-render the components in the HTML DOM by utilizing the virtual DOM. The virtual DOM is an in-memory representation of an actual DOM. By doing most of the processing inside the virtual DOM rather than directly in the browser's DOM, React can act quickly and only add, update, and remove components which have changed since the last render cycle occurred.

React provides different features the major features include:

▪ **JSX:** JSX is JavaScript syntax extension. It isn't necessary to use JSX in React development, but it is recommended.

▪ **Virtual DOM:** This characteristic of react helps to speed up the app development process and offers flexibility.

▪ **Component Based Architecture:** React is all about components. You need to think of everything as a component. This will help you maintain the code when working on larger scale projects.

▪ **Unidirectional data flow and Flux:** React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps keeping your data unidirectional.

React is based on components and states. This is what makes react such a popular library. When you want to create an application, you usually break it into simpler parts. When programming with React, you will want to break your interface into its most basic parts, and those will be your React components.

C# (C Sharp)

C# is a general-purpose, object-oriented programming language that is structured and easy to learn. It runs on Microsoft’s .Net Framework and can be compiled on a variety of computer platforms. As the syntax is simple and easy to learn, developers familiar with C, C++, or Java have found a comfort zone within C#.

C# is a modern, general-purpose programming language that can be used to perform a wide range of tasks and objectives that span over a variety of professions. C# is primarily used on the Windows .NET framework, although it can be applied to an open-source platform. This highly versatile programming language is an object-oriented programming language (OOP)—which isn’t very common—and fairly new to the game, yet already a reliable crowd pleaser.

When compared to long-standing languages like Python and PHP, C# is a young addition to the programming family at nearly twenty years old. The language was developed in the year 2000 by Microsoft's Anders Hejlsberg, a Danish software engineer with a history for popular creations. Anders has taken part in the creation of a handful of dependable programming tools and languages, including Microsoft's TypeScript and Delphi, a suitable replacement for Turbo Pascal. C# was originally designed to rival Java. Judging by the quick rise to popularity and the positive response from both new and seasoned developers, it's safe to say that goal has been achieved.

Like other general-purpose programming languages, C# can be used to create a number of different programs and applications: mobile apps, desktop apps, cloud-based services, websites, enterprise software and games. Lots and lots of games. While C# is remarkably versatile, there are three areas in which it is most commonly used.

- **Website development:** C# is often used to develop professional, dynamic websites on the .NET platform, or open-source software. Because this language is object-oriented, it is often utilized to develop websites that are incredibly efficient, easily scalable and a breeze to maintain.
- **Windows applications:** So it's easy to see why it's most popularly used for the development of Windows desktop applications. C# applications require the Windows .NET framework in order to function at their best, so the strongest use case for this language is developing applications and programs that are specific to the architecture of the Microsoft platform.
- **Games:** C# is one of the best programming languages for gaming. This language is heavily used to create fan-favorite games like Rim world on the Unity Game Engine.

Features of C#

The C# programming language has many features that make it more useful and unique when compared to other languages, including:

- **Object-oriented language:** Being object-oriented, C# allows the creation of modular applications and reusable codes, an advantage over C++. As an object-oriented language, C# makes development and maintenance easier when project size grows. It supports all three object-oriented features: data encapsulation, inheritance, interfaces, and polymorphism.
- **Simplicity:** C# is a simple language with a structured approach to problem-solving. Unsafe operations, like direct memory manipulation, are not allowed.
- **Speed:** The compilation and execution time in C# is very powerful and fast.
- **A Modern programming language:** C# programming is used for building scalable and interoperable applications with support for modern features like automatic garbage collection, error handling, debugging, and robust security. It has built-in support for a web service to be invoked from any app running on any platform.
- **Type-safe:** Arrays and objects are zero base indexed and bound checked. There is an automatic checking of the overflow of types. The C# type safety instances support robust programming.
- **Interoperability:** Language interoperability of C# maximizes code reuse for the efficiency of the development process. C# programs can work upon almost anything as a program can call out any native API.

ASP.NET

ASP.NET is a popular, open-source, server-side web-application framework designed for web development to produce dynamic web pages on the .NET platform. It was developed by Microsoft to allow programmers to build dynamic web sites, applications and services. It provides a programming model, a comprehensive software infrastructure and various services required to build up robust web applications for PC, as well as mobile devices. ASP.NET works on top of the HTTP protocol, and uses the HTTP commands and policies to set a browser-to-server bilateral communication and cooperation.

ASP.NET Core is the open-source version of ASP.NET that runs on macOS, Linux, and Windows. It was first released in 2016 and is a re-design of earlier Windows-only versions of ASP.NET. And it is designed to allow runtime components, APIs, compilers, and languages evolve quickly, while still providing a stable and supported platform to keep apps running.

Multiple versions of ASP.NET Core can exist side by side on the same server. Meaning one app can adopt the latest version, while other apps keep running on the version they were tested on. It provides various support lifecycle options to meet the needs of your app.

ASP.NET is a part of Microsoft .Net platform. ASP.NET applications are compiled codes, written using the extensible and reusable components or objects present in .Net framework. These codes can use the entire hierarchy of classes in .Net framework.

The ASP.NET application codes can be written in any of the following languages:

- C#
- Visual Basic.Net
- Jscript
- J#

ASP.NET is used to produce interactive, data-driven web applications over the internet. C# is used to develop the proposed project.

Entity Framework

Entity Framework is an open-source ORM framework for .NET applications supported by Microsoft. It enables developers to work with data using objects of domain specific classes without focusing on the underlying database tables and columns where this data is stored. With the Entity Framework, developers can work at a higher level of abstraction when they deal with data, and they can create and maintain data-oriented applications with less code compared with traditional applications.

Entity Framework is an object-relational mapper (O/RM) that enables .NET developers to work with a database using .NET objects. It eliminates the need for most of the data-access code that developers usually need to write.

Entity Framework (EF) Core is a lightweight, extensible, open source and cross-platform version of the popular Entity Framework data access technology. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and other databases through a provider plugin API.

We use code-first approach in the development of this project as opposed to a database-first approach. In the Code-First approach, you focus on the domain of your application and start creating classes for your domain entity rather than design your database first and then create the classes which match your database design. The code-first approach refers to scaffolding out your database using C# classes, and then using entity framework to migrate your database using built in commands.

Because everything is managed from C# code, when you want additional SQL types such as indexes, constraints or even just complex foreign keys, you can use either the Data Annotation attributes or the Entity Framework Fluent API Configuration.

The benefits of using a Code First approach are numerous:

- Your code is always communicating with a database schema that it has control over, therefore there is rarely a “mismatch” in column types, table definitions etc.
- You have an out of the box way to manage database migrations (So no dodgy “run this script before go live” type go live run sheets)
- Almost anyone can scaffold a database if they know C#, even without having too much knowledge of SQL in general (Although... this can be a negative)
- A code first approach works across different databases such as Postgres, MySQL etc

The MVC framework

Prior to the MVC framework, web programming intermingled database access code with the page's core code. Meaning the user received an HTML page as a result of this, and no abstraction was possible. Server-side language codes are saved in one file that is shared by at least three languages: C#, SQL, and HTML, even if CSS and JavaScript files are placed in external files.

The MVC pattern was established to separate logic from representation and to have a more physical and actual internal architecture. Model-View-Controller (MVC) denotes the three application levels recommended by the paradigm

- **Models:** represent how data is organized in a database. In simple terms, each model describes a database table as well as relationships between other models.
- **Views:** These are the files that contain all of the data that will be sent to the client. They create views that will be used to construct the final HTML content. The HTML code can be linked to the views.
- **Controllers:** These are the files that contain all of the server's actions and are not accessible to the client. The controller can either check for user authentication or create HTML code from a template.

Database Layer: The models are stored in database tables in SQLite.

Components are freely connected, which is a significant benefit of this strategy. Each component of a Django-powered Web application serves a particular purpose and can be updated independently of the others. A developer, for example, can modify the URL for a specific section of an application without impacting the underlying technology. A designer can alter the HTML of a page without touching the Python code that renders it. Instead of having to search and replace across a dozen files, a database administrator can rename a database table and specify the change in one location.

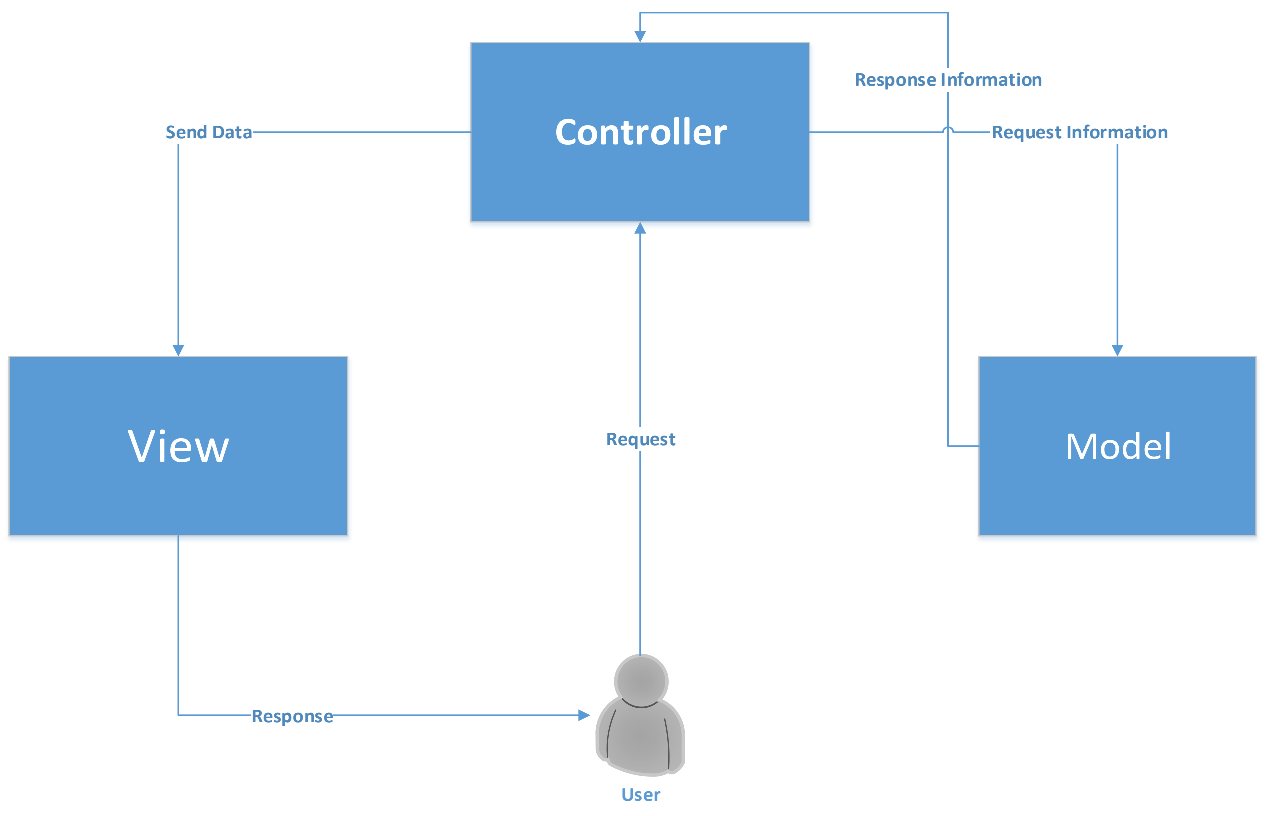


Figure 1 MVC Pattern

The following are the steps that are followed in an application with the MVC pattern:

1. The client sends a request to the server asking to display a page.
2. The controller uses a database through models. It can create, read, update, or delete any record or apply any logic to the retrieved data.
3. The model sends data from the database to the controller; for example, it sends a product list if we have an online shop.
4. The controller injects data into a view to generate it.

5. The view returns its content depending on the data given by the controller to the user.

Front-End Languages

HTML

HTML was created by Sir Tim Berners-Lee in late 1991 but was not released officially, published in 1995 as HTML 2.0. HTML 4.01 was published in late 1999 and was a major version of HTML.

HTML is a markup language that defines the structure of our content. HTML consists of a series of elements, which we use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way. The enclosing tags can make a word or image hyperlink to somewhere else, can italicize words, can make the font bigger or smaller, and so on.

HTML is a very evolving language and has evolved with various versions updating. Long before its revised standards and specifications are carried in, each version has allowed its user to create web pages in a much easier and prettier way and make sites very efficient.

HTML is a successor of SGML, but it was created so that normal people could use it and not just specialists i.e. since its first days, HTML has had the following advantages:

- Simplicity, which was achieved by means of a small set of structural elements called descriptors (also known as tags). All tags are written in angle brackets, for example <img> and convey some meaning.
- The ability to format a document without reference to means of display (such as a computer monitor or the screen of a phone)

CSS

CSS (Cascading Style Sheets) was first proposed by Håkon Wium Lie on October 10, 1994 while working with Tim Berners-Lee at CERN and the rest is history. CSS was not the only styling language in development at the time, but the very element of cascading and developing sequence is what set it apart from the rest.

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media. CSS is among the core languages of the open web and is standardized across Web browsers according to W3C specifications.

CSS is a domain-specific, declarative programming language; it describes how HTML elements are to be displayed on screen, paper, or in other media; it can control the layout, colors, and fonts of multiple web pages all at once. Cascading is the core feature of CSS since its inception, styles can cascade from one style sheet to another, enabling multiple types of CSS sources to be used on one HTML document. Inline CSS refers to CSS found in an HTML file, found in the head of a document between style tags. Inlining CSS simply means putting your CSS into your HTML file instead of an external CSS file.

Some of the advantages of using CSS are:

- Easier to maintain and update
- Greater consistency in design
- More formatting options
- Lightweight code
- Faster download times

3.2 Hardware Requirement

The only hardware requirement to develop the system is personal computer which will enable us to host a local server in order to see our progress and test the frontend functionalities by using necessary web development software tools.

3.3 Software Requirement

The software tools that are used to design the system are: Visual Studio Code, Postman, Node.js, and Python.

Visual Studio Code: which is also known as **Microsoft Visual Studio** and **VS**, is an integrated development environment for Microsoft Windows. It is a tool for writing computer programs, websites, web apps, and web services. It includes a code editor, debugger, GUI design tool, and database schema designer, and supports most major revision control systems. It is available in both a free "Community" edition and a paid commercial version.

Programming languages supported by Visual Studio are:

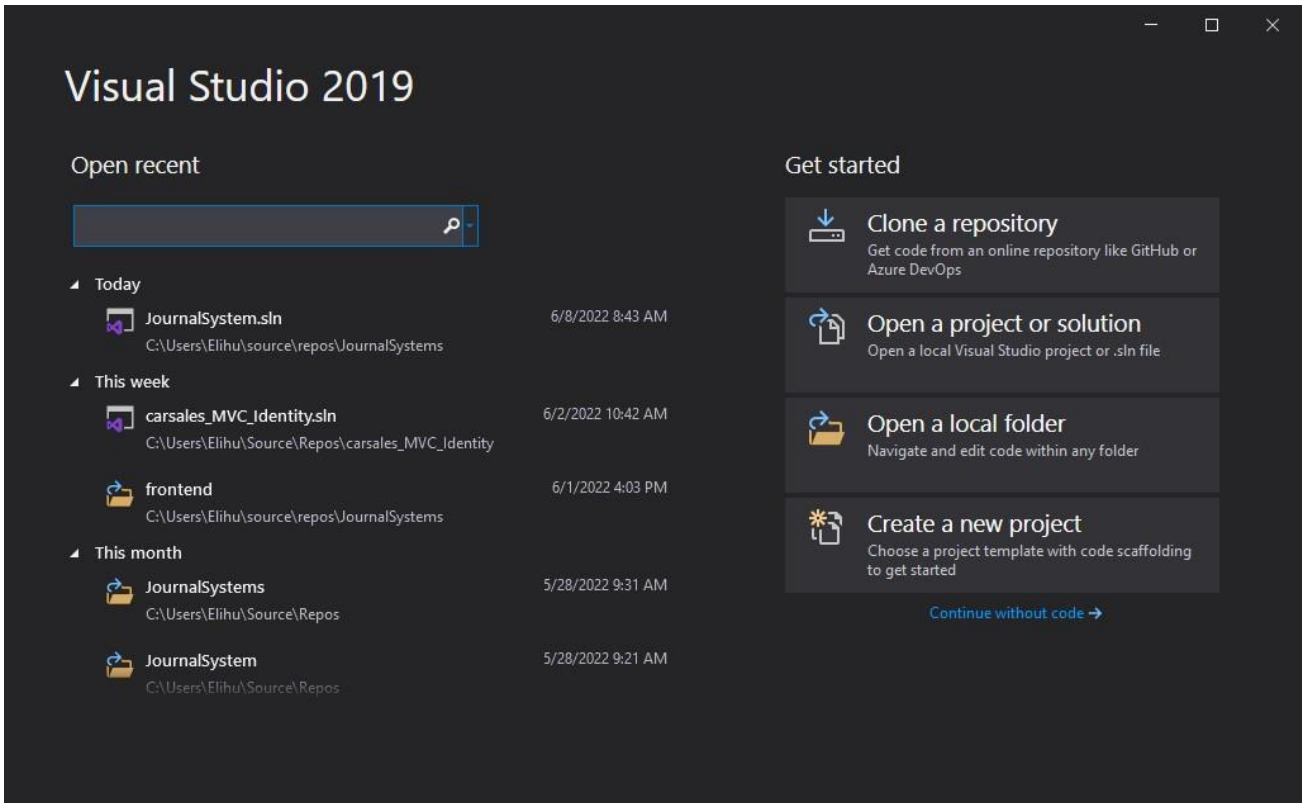
- C
- C++
- C#
- Visual Basic .NET
- F#
- Fossil
- M
- Python
- HTML/XHTML/CSS
- JavaScript

Evolution of Visual Studio: The first version of VS (Visual Studio) was released in 1997, named Visual Studio 97 having version number 5.0. The latest version of Visual Studio is 15.0 which was released on March 7, 2017. It is also termed Visual Studio 2017. The supported *.Net Framework Versions* in the latest Visual Studio are 3.5 to 4.7. Java was supported in old versions of Visual Studio, but the latest version doesn't provide any support for Java language. It is available for Windows as well as for macOS. 24

There are 3 editions of Microsoft Visual Studio as follows:

1. **Community:** It is the **free Microsoft Visual Studio** version, which is announced in 2014.
2. **Professional:** It is the commercial edition of Visual Studio. It comes in Visual Studio 2010 and later versions.
3. **Enterprise:** It is an integrated, end to end solution for teams of any size with demanding quality and scale needs.

The user Interface of visual studio looks like the figure below. You can see a number of tool windows when you will open the Visual Studio.



Visual Studio 2019

Open recent

Today
JournalSystem.sln 6/8/2022 8:43 AM
C:\Users\Elihu\source\repos\JournalSystems

This week
carsales_MVC_Identity.sln 6/2/2022 10:42 AM
C:\Users\Elihu\Source\Repos\carsales_MVC_Identity
frontend 6/1/2022 4:03 PM
C:\Users\Elihu\source\repos\JournalSystems

This month
JournalSystems 5/28/2022 9:31 AM
C:\Users\Elihu\Source\Repos
JournalSystem 5/28/2022 9:21 AM
C:\Users\Elihu\Source\Repos

Get started

Clone a repository
Get code from an online repository like GitHub or Azure DevOps

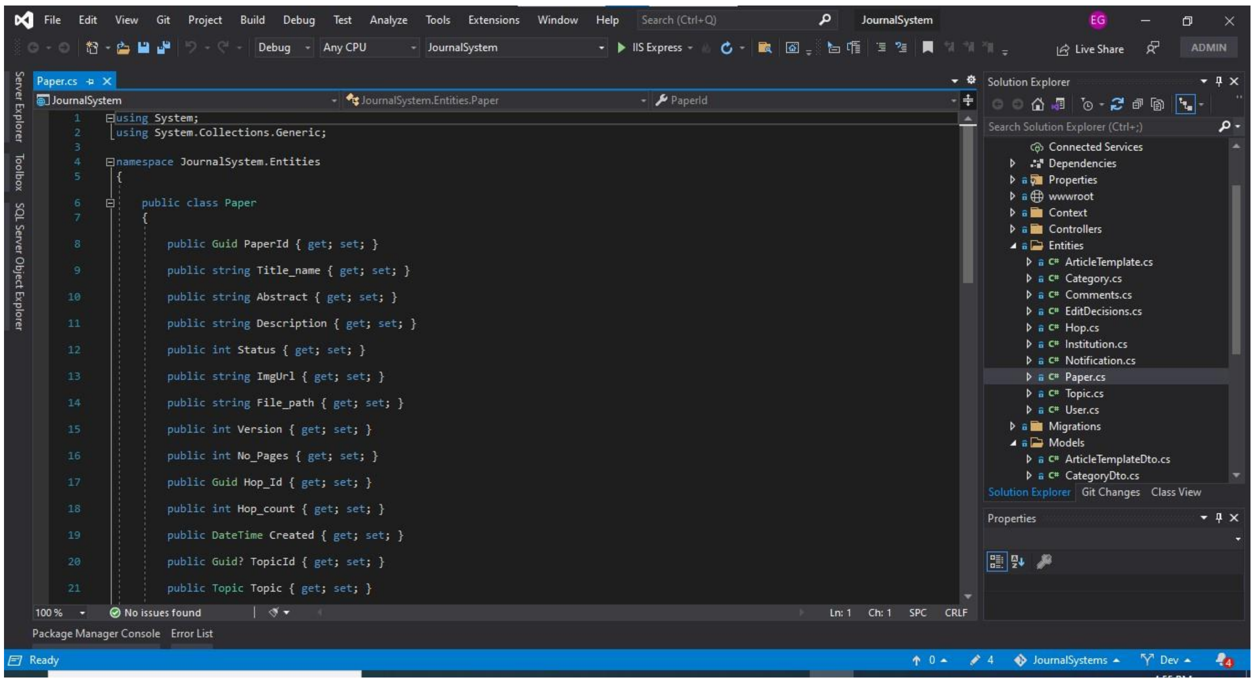
Open a project or solution
Open a local Visual Studio project or .sln file

Open a local folder
Navigate and edit code within any folder

Create a new project
Choose a project template with code scaffolding to get started

Continue without code →

Figure 2 Visual Studio



```
using System;
using System.Collections.Generic;

namespace JournalSystem.Entities
{
    public class Paper
    {
        public Guid PaperId { get; set; }
        public string Title_name { get; set; }
        public string Abstract { get; set; }
        public string Description { get; set; }
        public int Status { get; set; }
        public string ImgUrl { get; set; }
        public string File_path { get; set; }
        public int Version { get; set; }
        public int No_Pages { get; set; }
        public Guid Hop_Id { get; set; }
        public int Hop_count { get; set; }
        public DateTime Created { get; set; }
        public Guid? TopicId { get; set; }
        public Topic Topic { get; set; }
```

Figure 3 Visual Studio Solution Explorer

- **Code Editor:** Where the user will write code.
- **Output Window:** Here the Visual Studio shows the outputs, compiler warnings, error messages and debugging information.
- **Solution Explorer:** It shows the files on which the user is currently working.
- **Properties:** It will give additional information and context about the selected parts of the current project.

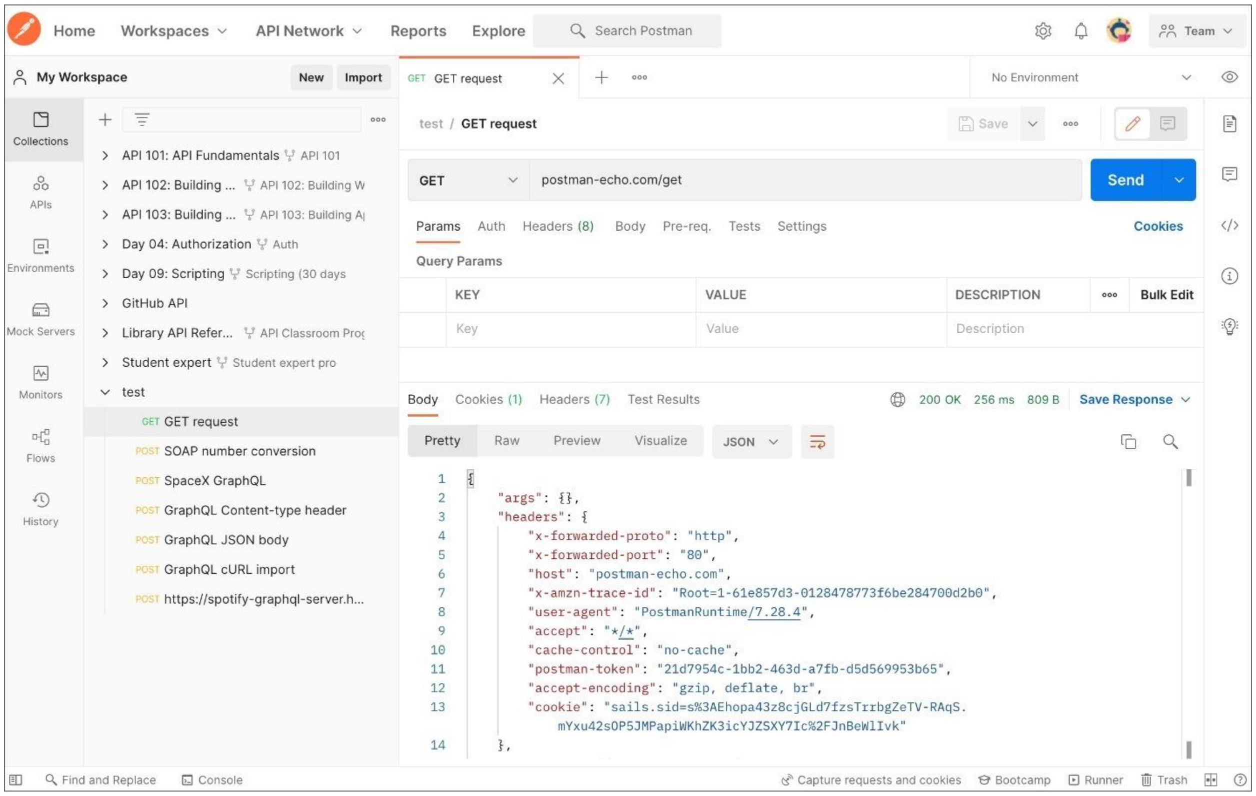
A user can also add windows as per requirement by choosing them from **View** menu. In Visual Studio the tool windows are customizable as a user can add more windows, remove the existing open one or can move windows around to best suit.

Postman: Postman is an API development tool which helps to build, test and modify APIs. It as a web development tool streamlines the development process for API developers. ADE (API Development Environment), a new term created by Postman, is argued to be a logical extension of the IDE (Integrated Development Environment), which provides an integrated set of components to support software development within a User Interface.

Postman has the ability to make various types of HTTP requests like GET, POST, PUT, PATCH, saving environments for later use, converting the API to code for various languages like JavaScript, Python.

The company says that a strong ADE will integrate with software development at multiple points to provide these benefits and more in a single environment:

1. **Testing & Debugging**: ADE provides a single place to debug, create tests and scripts, and run automated tests over time.
2. **Accurate API Documentation**: ADE enables devs to maintain a single source of truth for the API as it gets updated and improved over time.
3. **Collaboration & Version Control**: ADE allows devs to collaborate in real time with effective access to version control.
4. **Flexibility in Specification and Design**: ADE captures multiple forms of existing API specs and allows creation of an API spec.
5. **Ease of Publishing**: ADE helps API publisher get their API in the hands of developers so they can onboard quickly and effectively.



```
{
    "args": {},
    "headers": {
        "x-forwarded-proto": "http",
        "x-forwarded-port": "80",
        "host": "postman-echo.com",
        "x-amzn-trace-id": "Root=1-61e857d3-0128478773f6be284700d2b0",
        "user-agent": "PostmanRuntime/7.28.4",
        "accept": "*/*",
        "cache-control": "no-cache",
        "postman-token": "21d7954c-1bb2-463d-a7fb-d5d569953b65",
        "accept-encoding": "gzip, deflate, br",
        "cookie": "sails.sid=s%3AEhopa43z8cjGLd7fzsTrrbgZeTV-RAqS.
            mYxu42sOP5JMPapiWKhZK3icYJZSXY7Ic%2FJnBeWlIvk"
    },
```

Figure 4 Postman UI

Node.js: Node.js was written initially by Ryan Dahl in 2009, about thirteen years after the introduction of the first server-side JavaScript environment, Netscape’s LiveWire Pro Web. The initial release supported only Linux and Mac OS X. It is an open-source, cross-platform, back-end JavaScript run runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.

Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionalities. Modules are provided for file system I/O, networking (DNS, HTTP, TCP, TLS/SSL, or UDP), and other core functions.

CHAPTER FOUR

SYSTEM DESIGN

4.1 Introduction

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well running system. This phase includes description of each subsystem and the development of the subsystems. Not only had these, the database related to the normalization of the database.

This system design document presents the technical details of the journal management system in the context of Ethiopia, this system design and the document starts with an introduction to the architecture and the requirements to be considered.

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces. The SDD monitors the information necessary to define the suggested system architecture effectively and should provide guidance on the system architecture to be developed.

- To breakdown a system into subsystems that are easy to be handled or controlled without much difficulty.
- To find out the hardware/software stages on which the system will operate.
- To determine access control of each class operation.
- To determine the persistent data.
- To determine the persistent modelling for object-oriented database and the access control.

The system is based on an architecture with MySQL database and C# version handling the server side. The application will be developed in the Microsoft visual studio IDE for both the front and backend, for the front end the design layout has to be of most importance we will be using several frameworks to achieve our requirements and some are listed here jQuery and ajax, bootstrap. The solution will be implemented in one database due to the reason we students don't have the capacity to implement on large infrastructure we will be implementing it only one server.

This designing provides easy and smooth gateway for the implementation phase of the system.

The different modeling incorporated in implementation such as component and deployment modeling are seen in detail.

The procedure we have followed in the development of this system is as follows. As mentioned earlier we followed agile software development methodology while working on this project. This methodology helped us work through the project while figuring out the requirements along the way, since the title was a totally new concept for us that we were not familiar with. In addition to that it helped us play to our strengths by delegating tasks and concurrently working on different parts of the project.

We started our work by looking up other journal systems that are available online and figuring out the basic requirements for this project. This meant that we had written down a product spec and came up with a rough sketch of the UI/UX design. After this we worked on two aspects of the project simultaneously, one of them was designing the SQL database based the Entity-Relationship diagram and after that the use-case diagrams and sequence diagrams were drawn to better illustrate this relationship. Another thing concurrently being worked on at the time was the design of the front-end (UI design) part of the website.

Then we progressed onto the back-end implementation which was made easier because we had already worked on our designs earlier. In addition to that the work done on the UI gave us a clear idea of what fields, tables and columns were missing in our original draft design; meaning we were able to make adjustments along the way. Since we chose code-first approach as the way for our project we designed domain classes for the predetermined entities and generated a database that matched specifications we set in the classes.

And finally, after we tested each part to confirm it was working according to what we had planned, we set the system up for deployment and we were done with the project. Now we will dive into each step in much detail.

4.2 System Features

In this section, we will define the proposed system's functional and nonfunctional requirements, design the class diagram, the use case and its scenario, design the sequence diagram and activity diagram for the use cases, and ultimately design the system's interface.

The requirement definition is a statement of the task that the system should perform and the characteristics that it should possess in order to be accepted by the users. User requirements are statements in natural language with diagrams that describe the services the system is intended to provide to system users as well as the restrictions under which it must operate. This identifies the user objectives or tasks that the technology must enable them to complete.

As a result, user requirements define what the user will be able to perform with the system.

The system's features include:

- A menu-driven user interface with pleasant dialog boxes, drop-down lists, radio buttons, and text boxes for user input.
- In addition to that we will include an attractive dashboard for the actors involved.

The transition from one screen to the next must be simple and with faster loading time. This is implemented using single page applications.

Functional Requirements and Non-functional Requirements are the two sorts of requirements we will be concerned with.

The analysis model uses UML system models to introduce the proposed system. The model of the envisioned system is described in system models. We'll use use-case models and sequence diagrams to create the new system model.

Functional Requirements

Functional requirements pertain to the system's functioning and planned behaviors, or the services it will provide to the user. The system's response to specific inputs and how it should work in specific conditions should be specified in service statements. This behavior and functionality could be specified as services, tasks, or functions that the system must perform. The functional requirements for the recommended system are mentioned below.

- **These are users that interact with the system**.

❖ **End Users:** users who want to view and download research papers and journals

Frequency of use: - whenever it's necessary.

Subset of product function used: - can view, download papers.

Minimum technical expertise: - basic computer skills.

Privilege level: - they can view published papers and download them.

❖ **Amins:** are those users who will monitor the overall status of the system including assigning editors and controlling the user list.

Frequency of use: - whenever it's necessary.

Subset of product function used: - manage the over-all system, view and perform initial screening of submitted papers, assign tasks to reviewers, accept/reject papers, and if accepted prepare them for publishing.

Minimum technical expertise: - high level computer skills and

Privilege level: - see submissions, have access to the list of users of the system, assign reviewers to review the paper, accept/reject papers and publish papers.

❖ **Editors:** are those who will operate the system receiving paper submissions and overseeing the review process.

Frequency of use: - whenever it's necessary.

Subset of product function used: - view and perform initial screening of submitted papers, assign tasks to reviewers, accept/reject papers, and if accepted prepare them for publishing.

Minimum technical expertise: - basic level computer skills, editorial qualifications (to check papers against a standard and to check for plagiarism).

Privilege level: - see submissions, have access to the list of users of the system, assign reviewers to review the paper, accept/reject papers and publish papers.

- ❖ **Authors:** are those who submit their work in order to be published on the journal.

Frequency of use: - whenever it's necessary.

Subset of product function used: - submit papers, view and track the progress of submitted papers.

Minimum technical expertise: - basic computer skills, solid educational background (qualification in a specified field of study).

Privilege level: - submit papers, track submitted papers and view papers submitted by them along with the comments.

- ❖ **Reviewers:** are those who will perform the peer review process on submitted papers that are aligned with their area of expertise.

Frequency of use: - whenever it's necessary.

Subset of product function used: - peer review papers and give recommendations to editor on what to do with a submission

Minimum technical expertise: - basic computer skills, high level expertise and reputation in a specific field of study

Privilege level: - see submissions, review submitted papers and give comments on them, recommend whether to accept/reject papers and publish papers.

Table 3 FR-01 Login

ID	FR-01
Name	Login
Summary	Account usage
Description	An existing user can login to use the website
Dependency	FR-02

Table 4 FR-02 Create account/Register

ID	FR-02
Name	Create account / Register
Summary	Account creation
Description	A new user can create an account
Dependency	None

Table 5 FR-03 View Profile Info

ID	FR-03
Name	View profile info
Summary	User can view their profile
Description	This functionality will let user see that he/she has the right credentials, photo, name... on their profile
Dependency	FR-01

Table 6 FR-04 Upload Paper

ID	FR-04
Name	Upload paper
Summary	Upload research paper for review
Description	This functionality will let the author/researcher submit their paper for review and later publication.
Dependency	FR-01

Table 7 FR-05 Track Submitted Paper

ID	FR-05
Name	Track submitted paper
Summary	Shows what stage of review the submitted paper is in
Description	The author in this case will be able to track at what stage of publishing his/her paper is at and hold the journal accountable for a delayed process.
Dependency	FR-01 FR-04

Table 8 FR-06 See Submissions

ID	FR-06
Name	See submissions
Summary	Displays the papers submitted by different authors to the editor.
Description	The editor will have access to all submitted papers and will start the review process by screening among them.
Dependency	FR-01 FR-04

Table 9 FR-07 Assign Reviewer

ID	FR-07
Name	Assign reviewer
Summary	Assign a paper to a reviewer.
Description	It will display reviewers with related area of expertise to a certain paper to review it.
Dependency	FR-01 FR-06

Table 10 FR-08 Edit Profile

ID	FR-08
Name	Edit Profile
Summary	Allows the user (author, editor and reviewer) to edit profile.
Description	It will let the user his/her profile to change to an updated information.
Dependency	FR-01

Table 11 FR-9 Search

ID	FR-9
Name	Search
Summary	Search capability
Description	It will let the end-user find question based on a keyword typed on the search bar.
Dependency	None

Table 12 FR-10 Show Assigned Papers

ID	FR-10
Name	Show assigned papers
Summary	Display the papers assigned to the reviewer
Description	It will show the list of papers assigned to a certain reviewer in order to review.
Dependency	FR-04

Table 13 FR-11 Submit Recommendations

ID	FR-11
Name	Submit recommendations
Summary	Recommendations on decisions to be made regarding papers
Description	The reviewer submits recommendations on the papers he/she reviewed and sends them to the editor.
Dependency	FR-01 FR-10

Table 14 FR-12 Add topic and category

ID	FR-12
Name	Add topic and category
Summary	Topics and categories will be added
Description	Topics and categories will be added based on the needs of the journal
Dependency	FR-01

Table 15 FR-13 Remove topics and categories

ID	FR-13
Name	Remove topics and category
Summary	Outdated topics and categories are removed from the system
Description	Outdated topics and categories that are not inline with current needs are removed from the system.
Dependency	FR-01 FR-12

Table 16 Update topics and categories

ID	FR-14
Name	Update
Summary	Topics and categories are updated
Description	If there is anything to be modified on the topic or category the admin will update specific topics and categories.
Dependency	FR-01 FR-12

Table 17 FR-15 Remove account

ID	FR-14
Name	Remove account
Summary	Accounts are removed
Description	If there are any accounts that we want to remove, we use this method
Dependency	FR-01

Non-Functional Requirements

Non-functional requirements, as the term implies, are those that are not immediately related to the system's unique services provided to its users. On the other hand they have everything to do with emergent system attributes like reliability, response time, or store occupancy.

Non-functional requirements, such as performance, security, or availability, are used to define or constrain the system's overall characteristics.

Good Performance: Taking into account the typical quantities of hardware required to run this system, we will optimize it for optimal performance. In order for the suggested system to have a good response time, we will employ best practices, shorten scripts or use simplified models with latest approaches, and remove superfluous flaws.

GUI: the new system that we are going to develop has a graphical user interface that can interact with users graphically in an easier and more understandable way. The system is user-friendly and easy to use and requires less training to understand.

Availability: The system must operate twenty-four hours a day, seven days a week wherever Internet is available, except for exceptional occasions of maintenance and upgrade.

Portability: The proposed system shall work properly in any operating system and web browser. Because the programming languages we use are platform independent.

Security

Before any kind of inputs are submitted to the database, they should critically be analyzed to block illegal inputs. Malicious deformations of data holding information on a client's case should be avoided.

➢ **User Input Validation**

- All user input must be validated. The system must manage client input controls (based on predefined input parameters) from the server side to the extent possible. All third-party client-side input controls must be well documented and approved by the system.

➢ **Username/Password Based Authentication**

- When usernames and passwords are going to be used as the method for system authentication the following for each must be met:

  - **Username requirements for clients/administrators valuating companies:**

    - ✓ Usernames are unique and are traceable to an individual end-user/editor/reviewer account.
    - ✓ Usernames are NOT to be shared and are never hard-coded into system logic. This is because every user will create their own.
    - ✓ Usernames should be easy to remember because they will be used to login to the account.

  - **Password requirement for clients/administrators valuating companies:**

    - ✓ Are not to be shared.
    - ✓ Must be 8 characters or more in length.
    - ✓ Must NOT be a word found in the dictionary, regardless of language.
    - ✓ Password must NOT be stored in clear text.
    - ✓ Must be changed at least every 60 days.
    - ✓ Must be changed immediately if revealed or compromised.
    - ✓ Passwords must be encrypted using irreversible industry-accepted strong encryption.
    - ✓ Accounts must be locked after 5 failed login attempts.
    - ✓ Must be composed of characters from at least three of the following four groups from the standard keyboard:

- Upper case letters (A-Z);
- Lower case letters (a-z);
- Arabic numerals (0 through 9)

4.3 Writing A Product Spec

In this phase of the project what we mainly focused on was to create an idea of what a journal system entails and what would be required from us when building it. This is where we performed our data gathering from various sources. In order to get ideas about what the system looked like we did extensive research on already existing systems. This was done by asking individuals who were familiar with the system to explain the details to us, in addition to us visiting these sites and making our own observations.

Here all we did was discuss what the web app was going to include, in terms of features. Additionally, we discussed what different types of end users there are and how they would interact with the system based on the roles they play. In this phase what we have done is mostly idea building, which guides the way the project will turn out and gave us an idea of what we are expected to do. Based on this we mapped out our system architecture and did our system analysis.

4.3.1 System Architecture

The system has three main components: frontend, backend and database. The frontend is implemented using react + redux for creating user view page and create our react app. We also use react bootstrap, implement react router and test our server using the local host. Node.js JavaScript is used to create the front end. To some extent we also used C# razor pages MVC in order to implement some pages. In the backend C# ASP.Net Core is implemented, specifically using Entity Framework Core. We started by creating the working environment, installing ASP.Net Core and setting it as an environmental variable rest frame work library for application program interface (API). After both the frontend and backend are created, Axios is used to fetch data from backend to the frontend (to create a link between the two parts of the project.

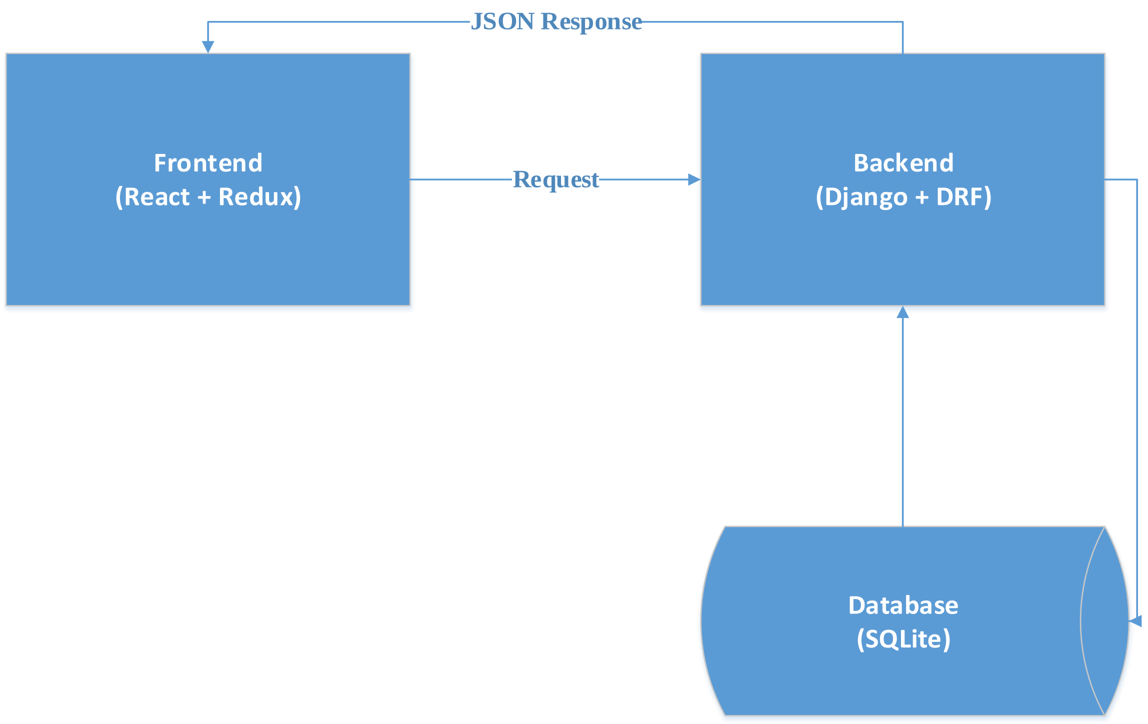


Figure 5 Overall Architecture of the System

The frontend runs on a local server host port: http://127.0.0.1.3000. First we started the project by rendering static data from the react folder and once the backend was in developed was hosted on different port: http://127.0.0.1.44225. After having both servers up and running, we open the aforementioned links on a web browser and once we have that backend with API and database now we can use different methods to make data request from the front end using Axios that will be sent to the backend.

The backend we have implemented has 3 layers of abstraction within itself. These layers are the OAuth layer which handles the security of our system using an integration of ASP.Net Identity and Identity Server, this helps secure our user data and the tokens we will send. Another layer is our MVC layer which handles the association between our views and our datastore. This part of our system is responsible for the creation of some of our views. The final layer here is the Datalink layer which handles the communication of data to and from the database. This creates a sort of abstraction between our database and our backend logic. In our backend we are going to get a request inside of a view through different methods. Now, from that view, we will send the invoked request to our database which is SQL Server. The database will get that data and send it back into our view. And before we return that data to the frontend, we need to serialize our data and give it an attractive structure, because right now it’s just a C# query set and we can only get JSON data in our frontend. So, we serialize that data and then return back to the frontend. The basic methods used in this project to request and modify data are:

- **GET Method:** a request method which is used to retrieve intended data from the database. For example if we just want to get our papers, we send a get request GET /api/paper/ to port 44225.
- **POST Method:** a request method that enables us to create a user or upload a paper onto the database. If we want to upload a paper, we send a post request POST /api/paper/create/ to the backend port
- **PUT Method:** a request method that allow us to update an existing paper or user. If we want to update a paper, we send a put request PUT /api/paper/update/id to the backend port.
- **DELETE Method:** a request method which is used to remove a paper or a user from the data base. If we want to delete a paper, we send a delete request DELETE /api/paper/id/ to the backend port.

4.3.2 System Analysis

This approach uses a unified modeling language (UML) in which it depends on the visual modeling of the system. UML is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

4.3.2.1 Behavioral Diagram

Behavioral diagrams describe the functionality of the software system. It shows how the system interacts with the user and it self. The behavioral diagrams used in this project are:

1. Use case diagram
2. Sequence diagram
3. Activity diagram

Use Case Diagram

A use case diagram is a set of actions that gives an actor quantitative value. Another way to look at it is that a use case shows how a user interacts with the system in the actual world. A business use case is an important use case. This type of use case is a simple, abstract, generalized use case that expresses the user's intent in a technology and implementation agnostic way. Case models are used to document a system's behavioral (functional) requirement, often known as the "what." A fully documented essential use case is a structured narrative that includes a simplified, abstract, technology-free, and implementation-independent description of one job or interaction, presented in the language of the application domain and users.

A use case diagram depicts a system's use cases, their actors, the relationships between the actors and the use cases, and the relationships among the use cases. It also depicts a system function as a series of events that result in a visible consequence for the actors. It is a set of instructions for achieving a goal. The components of the unified modeling language (UML) diagram are listed below.

Use case: an oval-shaped description of a series of actions that deliver something of measurable value to an actor.

Actor: is a person or an external system that plays a part in one or more interactions with the system. And represented with a stick figure, here are the actors of the system

1. End Users
2. Authors (Researchers)
3. Editors
4. Reviewers

Types of relationships in our use case diagram include

- **Association:** represented as a line between an actor and a use case represents that the actor initiates and/or participates in the process.

- **Include:** is a directed relationship between two use cases which is the included use case is required, not optional.

- **Extend:** is a directed relationship between two use cases which is the included use case is optional, not required.

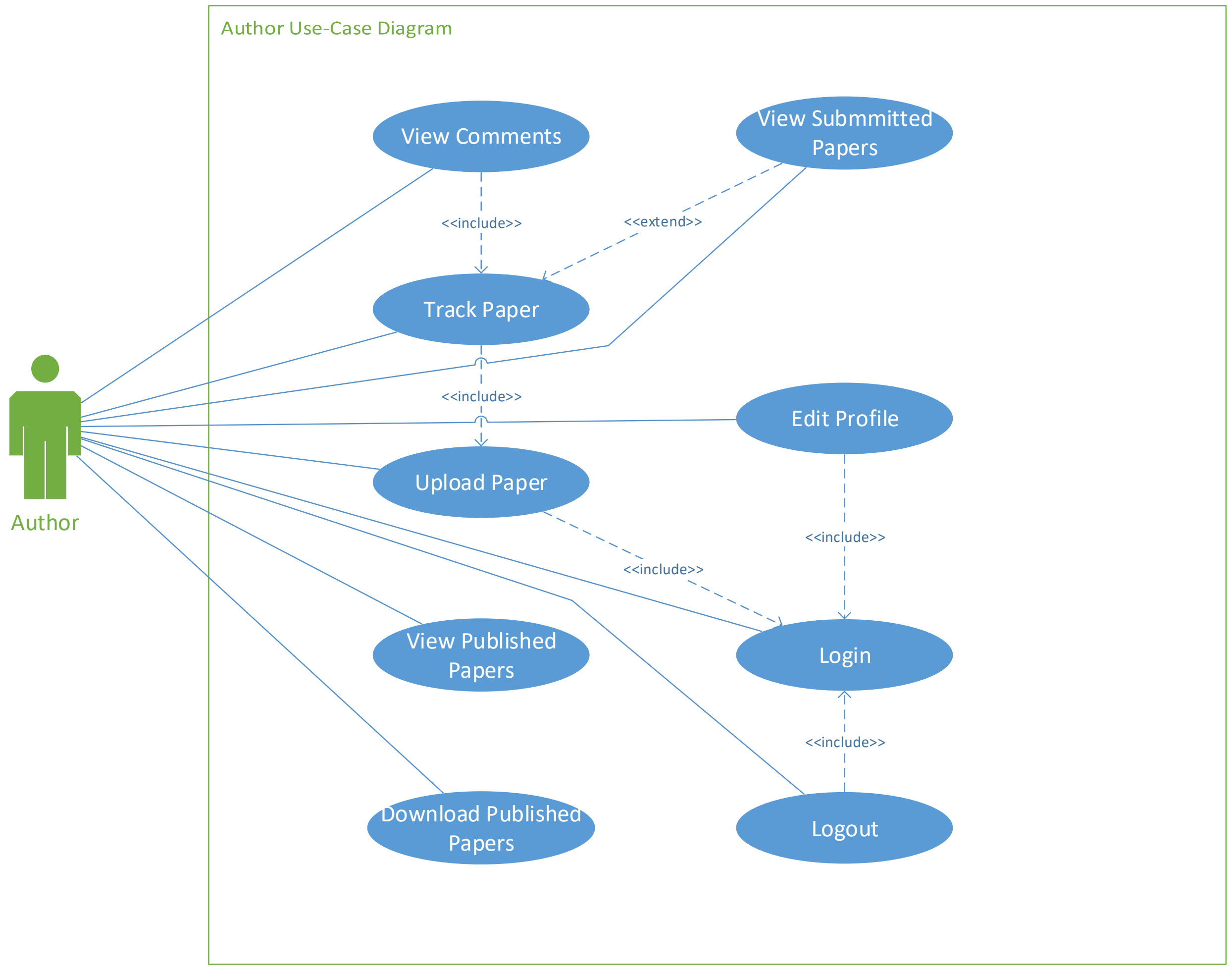


Figure 6 Author Use-Case Diagram

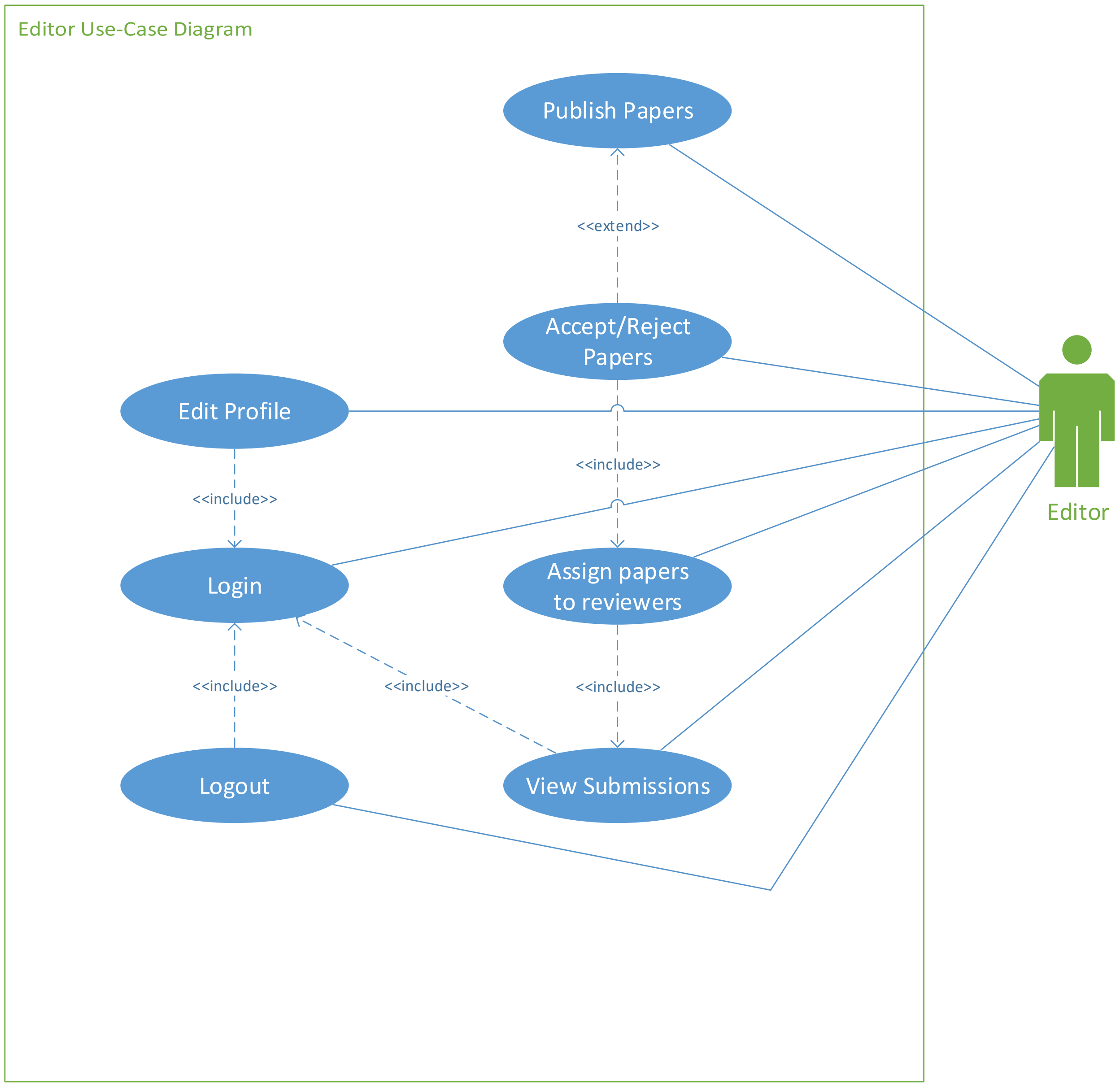


Figure 7 Editor Use-Case Diagram

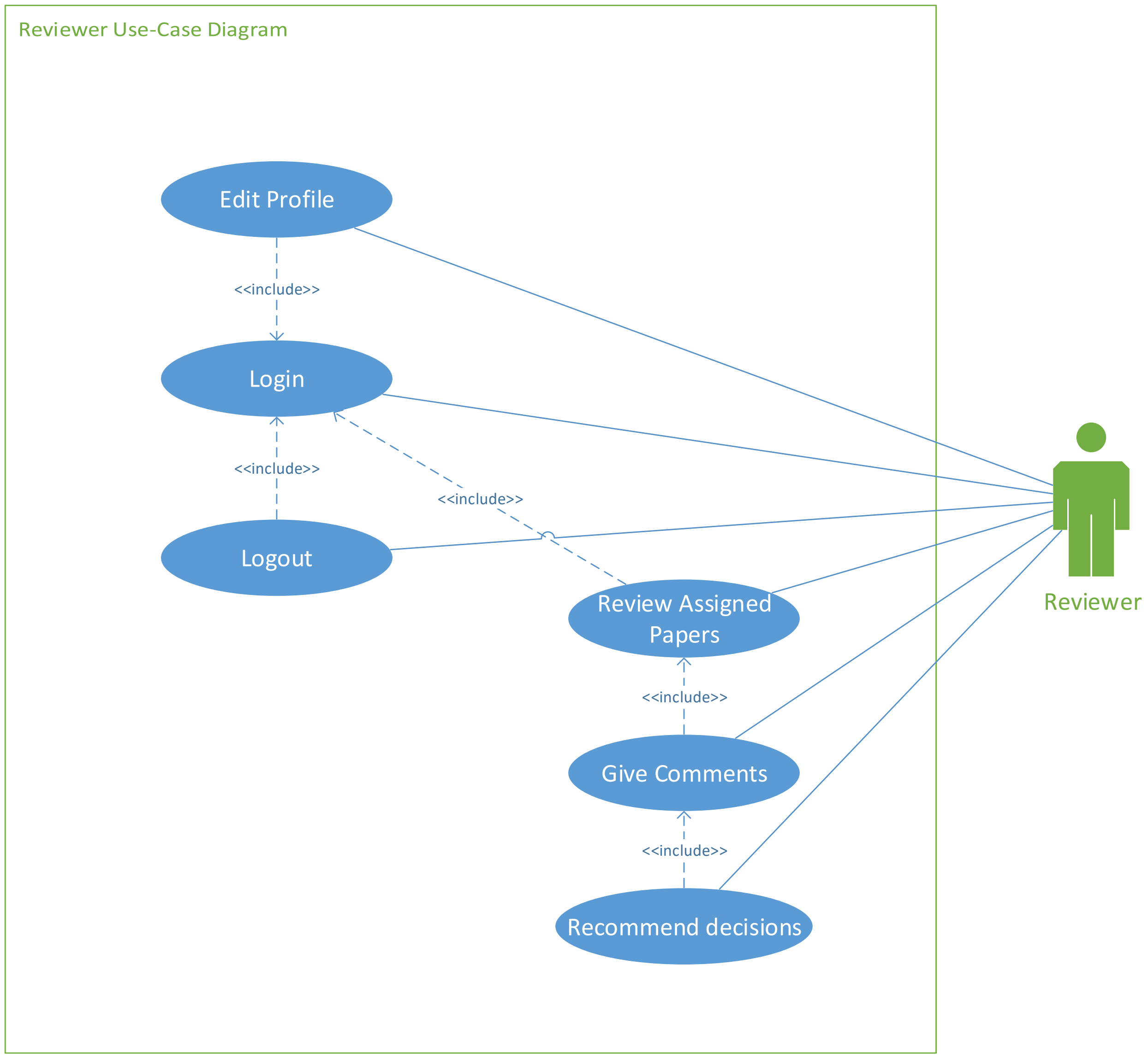


Figure 8 Reviewer Use-Case Diagram

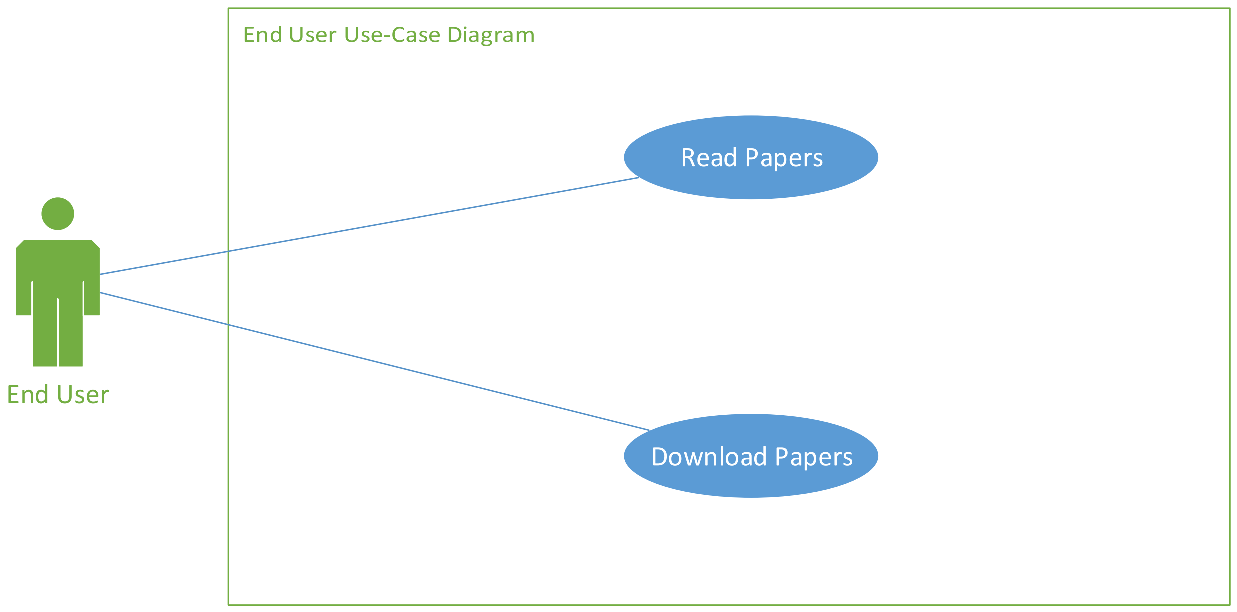


Figure 9 End User Use-Case Diagram

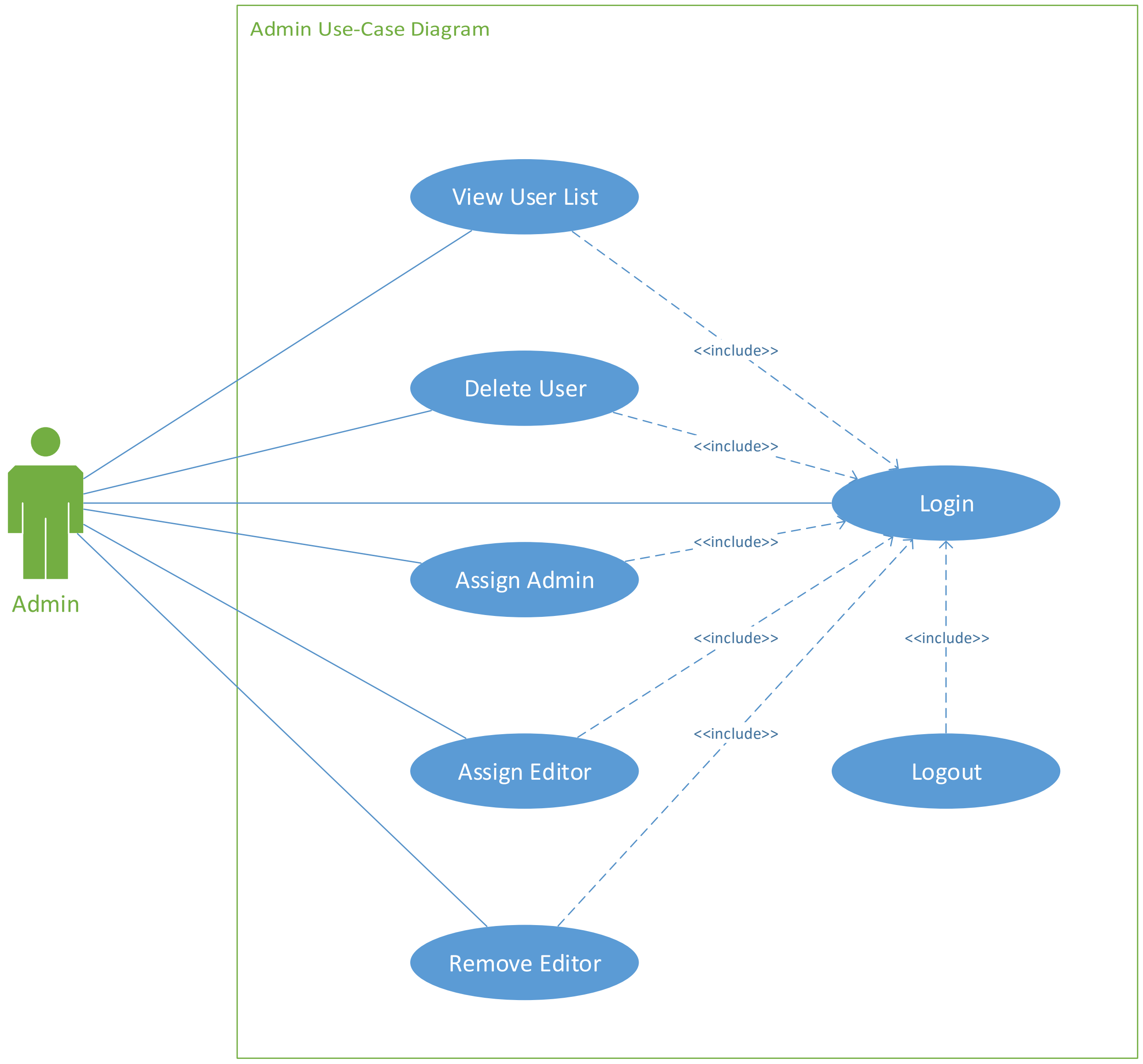


Figure 10 Admin Use-Case Diagram

Sequence Diagram

Sequence diagrams depict the interactions between the objects involved in a particular use case. They're helpful for finding items that weren't detected in the analysis object model. The sequence diagram of each identified use case is shown below to show the interaction between objects. The use case is graphically documented using sequence diagrams, which indicate the classes, messages, and message timing. Sequence diagrams are sometimes known as event diagrams or event scenarios.

An interaction is an activity that relies on the observable exchange of information between objects.

A lifeline represents an object's engagement in a certain interaction.

The head is located on top of a vertical dashed line known as the stem, which indicates the chronology for the object's instance.

A dashed line with an arrowhead denotes a reply message.

A solid line with a solid arrowhead denotes a synchronous call or signal message. Until the destination lifeline responds, the source lifeline is stopped from performing any more tasks.

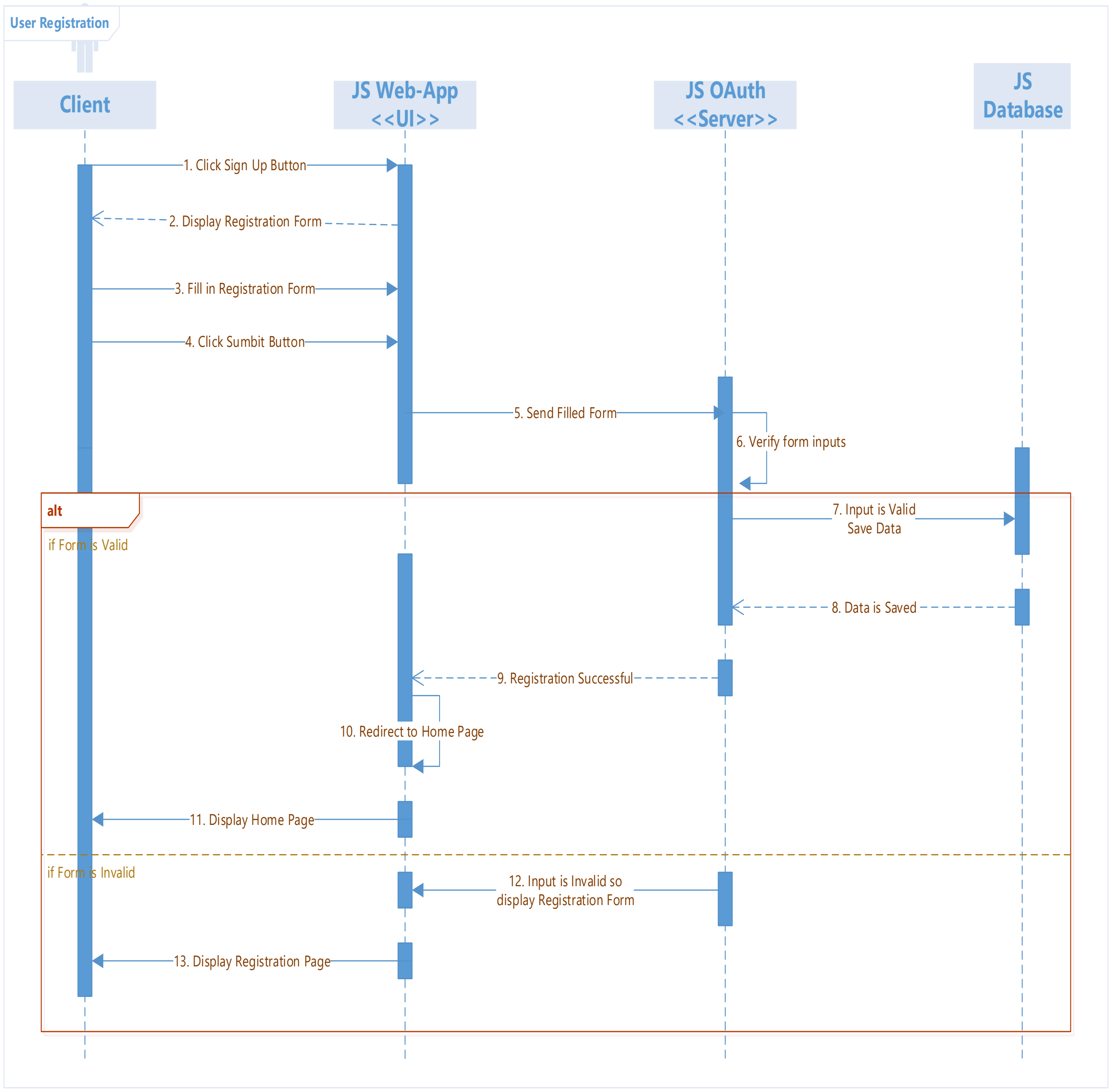


Figure 11 User Registration Sequence Diagram

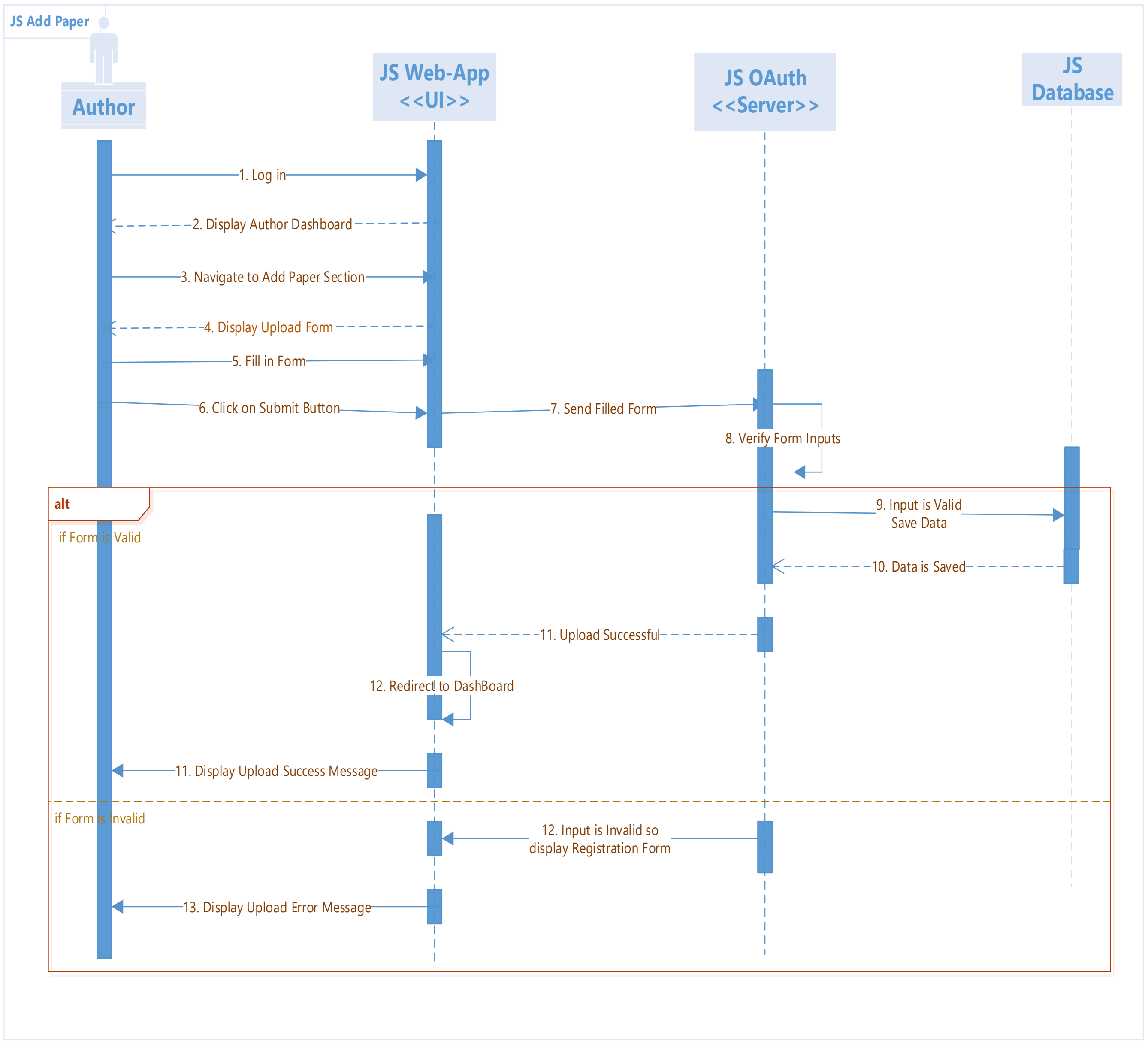

JS Add Paper
Author
JS Web-App <<UI>>
JS OAuth <<Server>>
JS Database
1. Log in
2. Display Author Dashboard
3. Navigate to Add Paper Section
4. Display Upload Form
5. Fill in Form
6. Click on Submit Button
7. Send Filled Form
8. Verify Form Inputs
alt
if Form is Valid
9. Input is Valid Save Data
10. Data is Saved
11. Upload Successful
12. Redirect to DashBoard
11. Display Upload Success Message
if Form is Invalid
12. Input is Invalid so display Registration Form
13. Display Upload Error Message

Figure 12 Add Paper Sequence Diagram

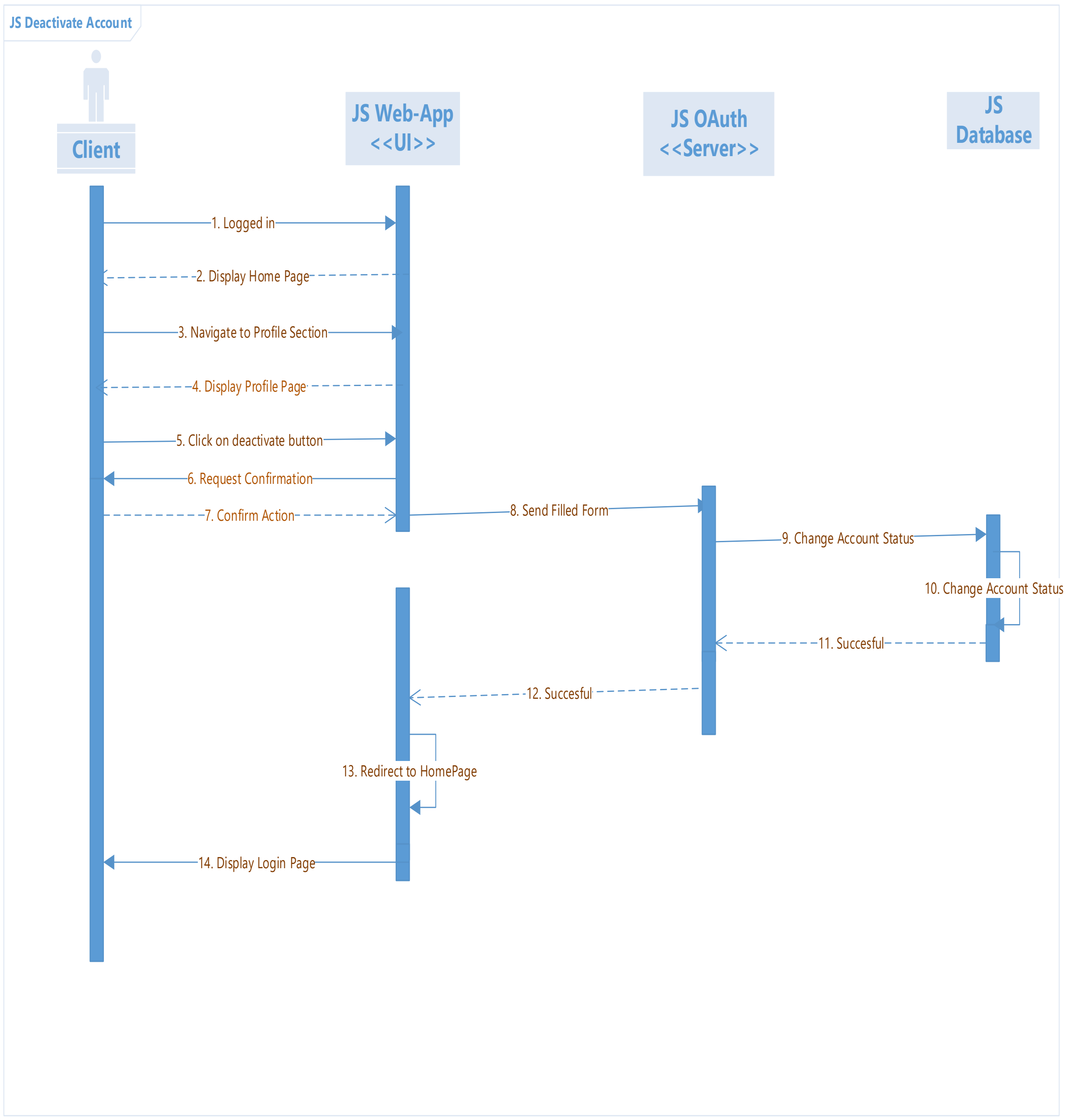


Figure 13 Deactivate Account Sequence Diagram

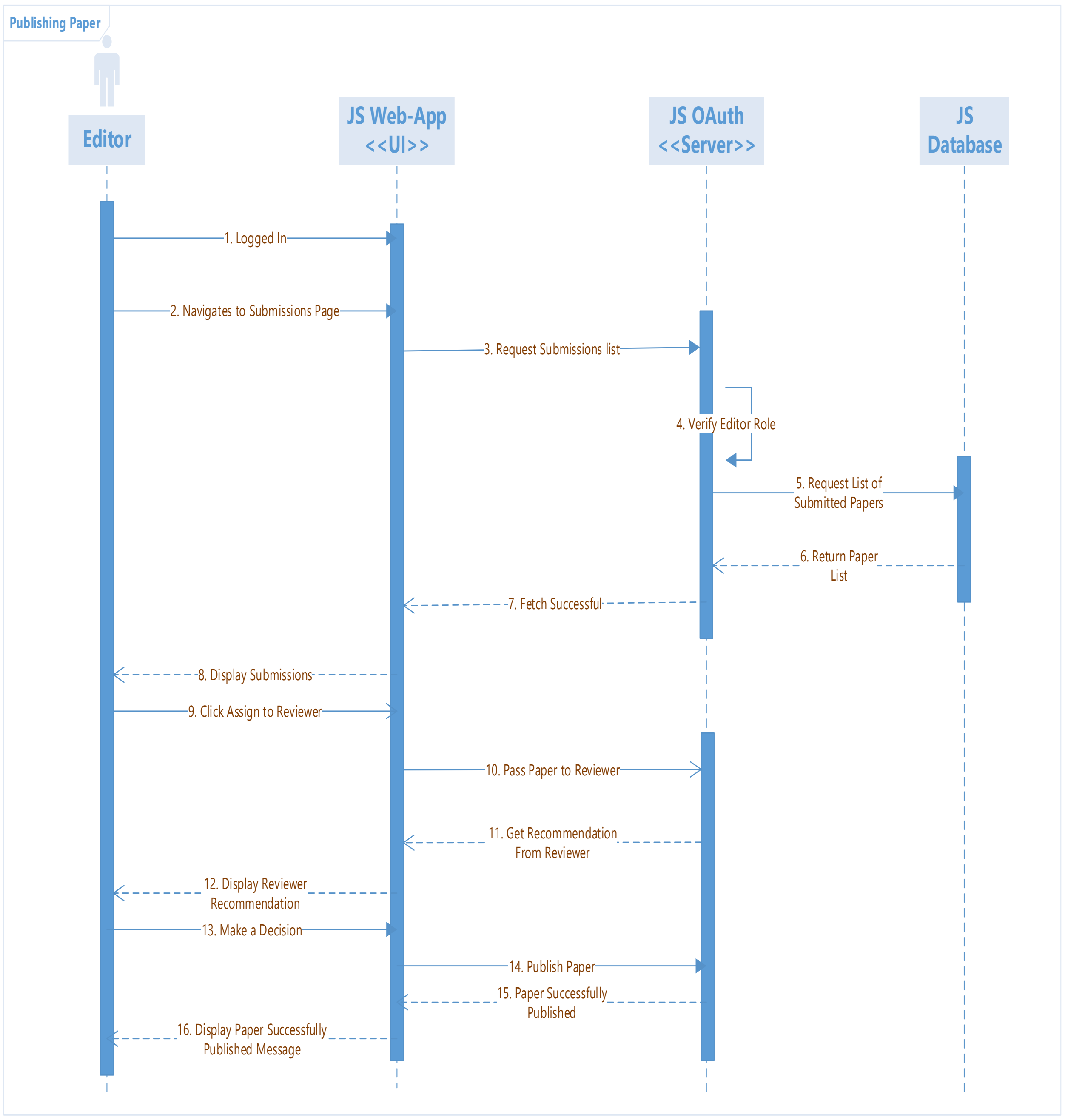

Publishing Paper
Editor
JS Web-App <<UI>>
JS OAuth <<Server>>
JS Database
1. Logged In
2. Navigates to Submissions Page
3. Request Submissions list
4. Verify Editor Role
5. Request List of Submitted Papers
6. Return Paper List
7. Fetch Successful
8. Display Submissions
9. Click Assign to Reviewer
10. Pass Paper to Reviewer
11. Get Recommendation From Reviewer
12. Display Reviewer Recommendation
13. Make a Decision
14. Publish Paper
15. Paper Successfully Published
16. Display Paper Successfully Published Message

Figure 14 Paper Publishing Sequence Diagram

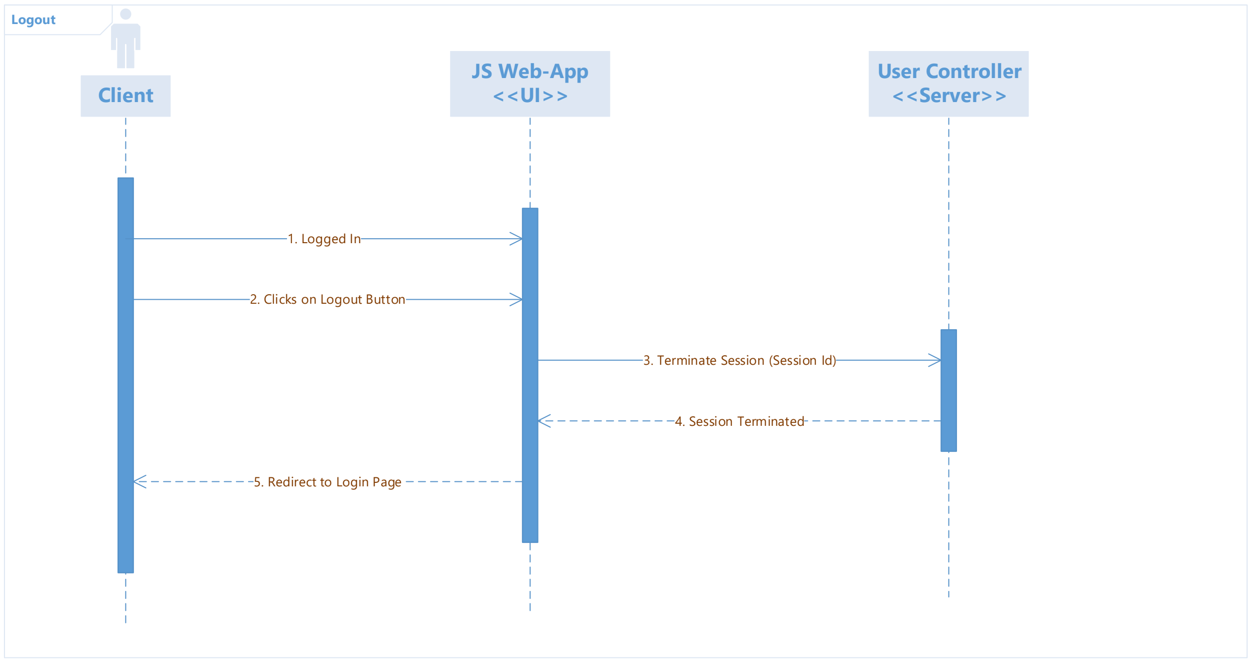


Figure 15 Logout Sequence Diagram

Activity Diagram

An activity diagram is a visual representation of a system's control flow and a reference to the stages involved in performing a use case. We model sequential and concurrent activities using activity diagrams. As a result, we employ an activity diagram to illustrate operations in a clear manner. The state of flow and the order in which it occurs are the emphasis of an activity diagram. We describe or show what causes a certain occurrence using an activity diagram.

An activity diagram depicts the control flow from a start point to an end point, as well as the multiple decision pathways that exist while the activity is being conducted. An activity diagram can depict both sequential and concurrent activity processing.

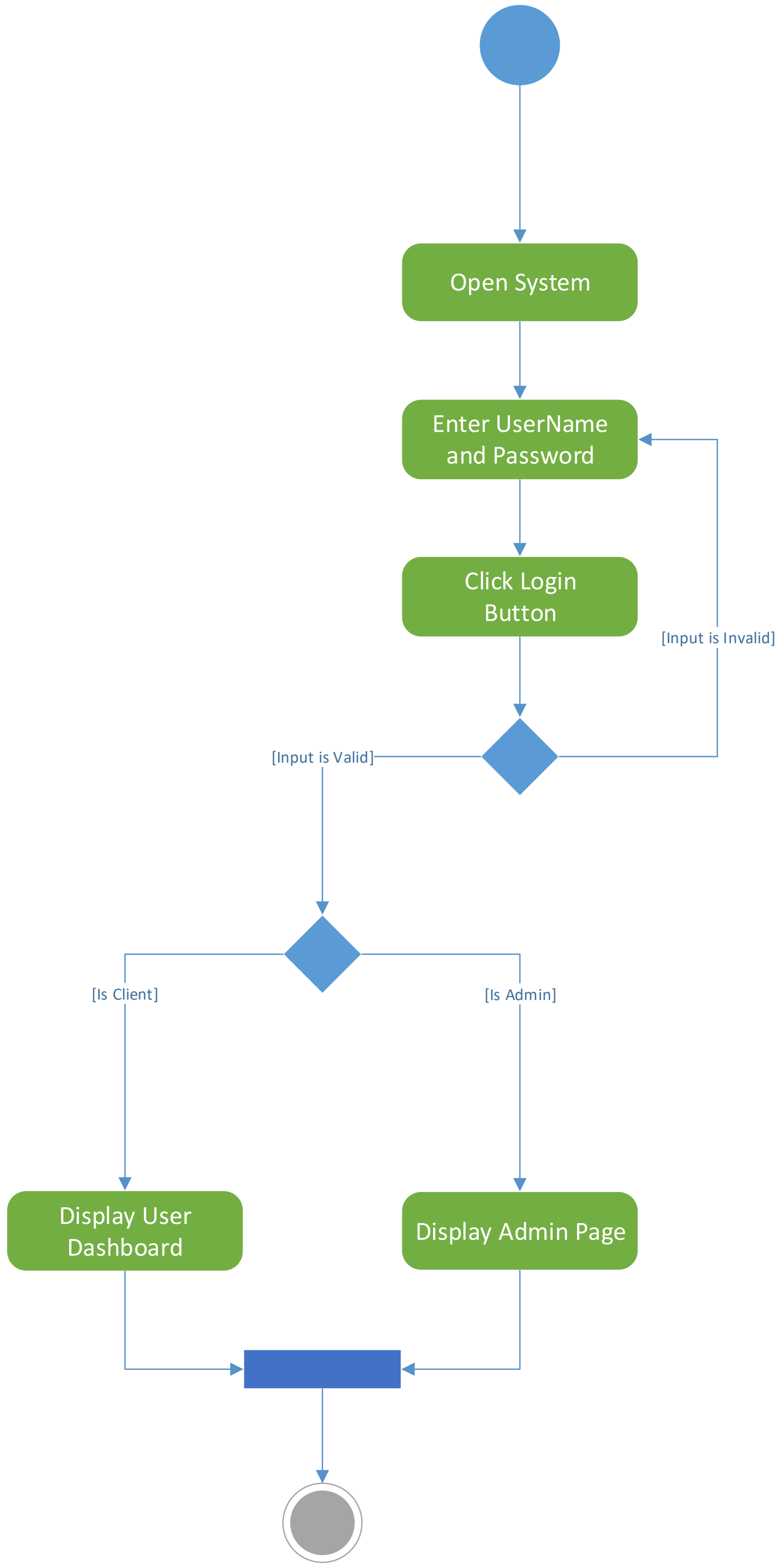
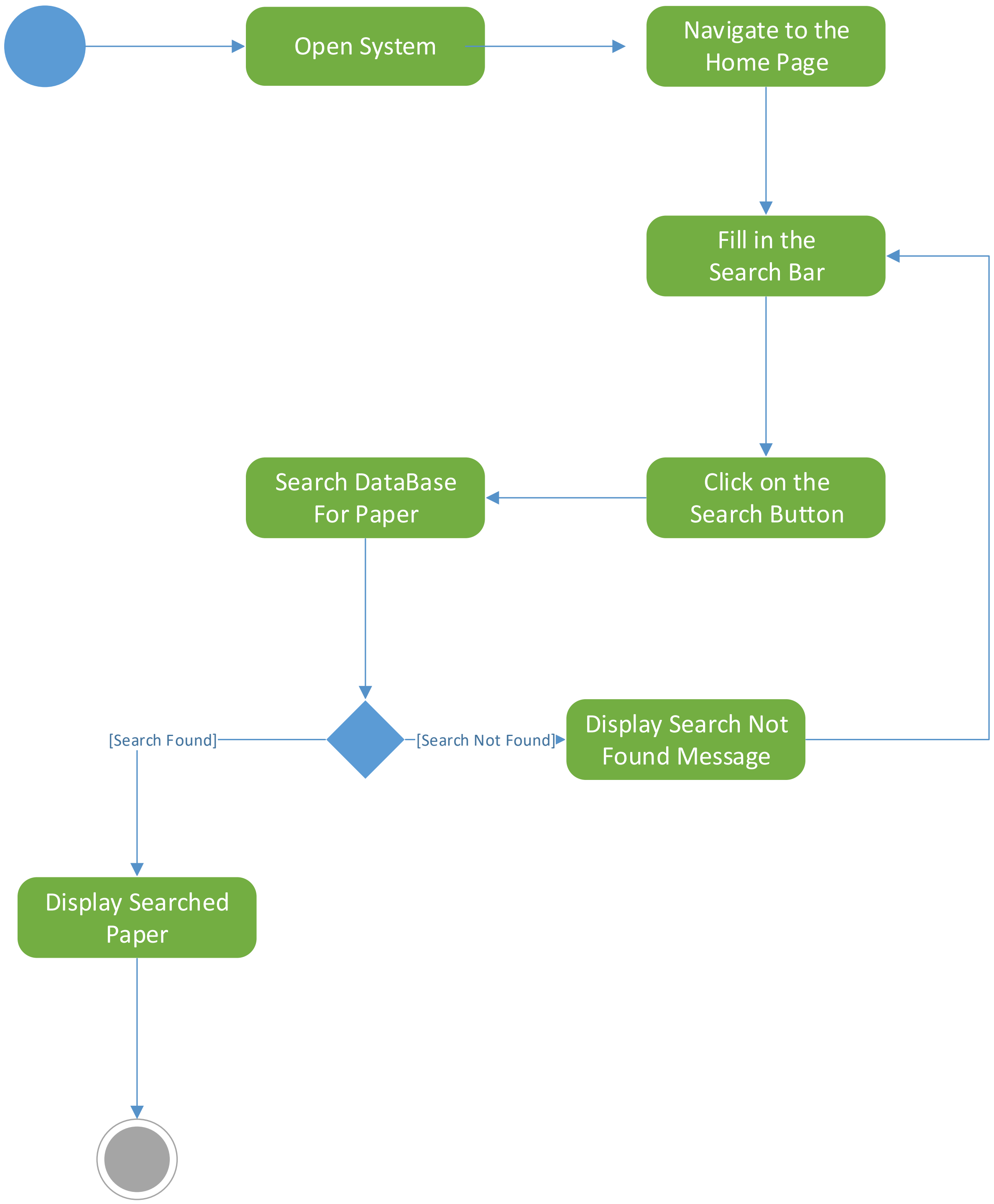
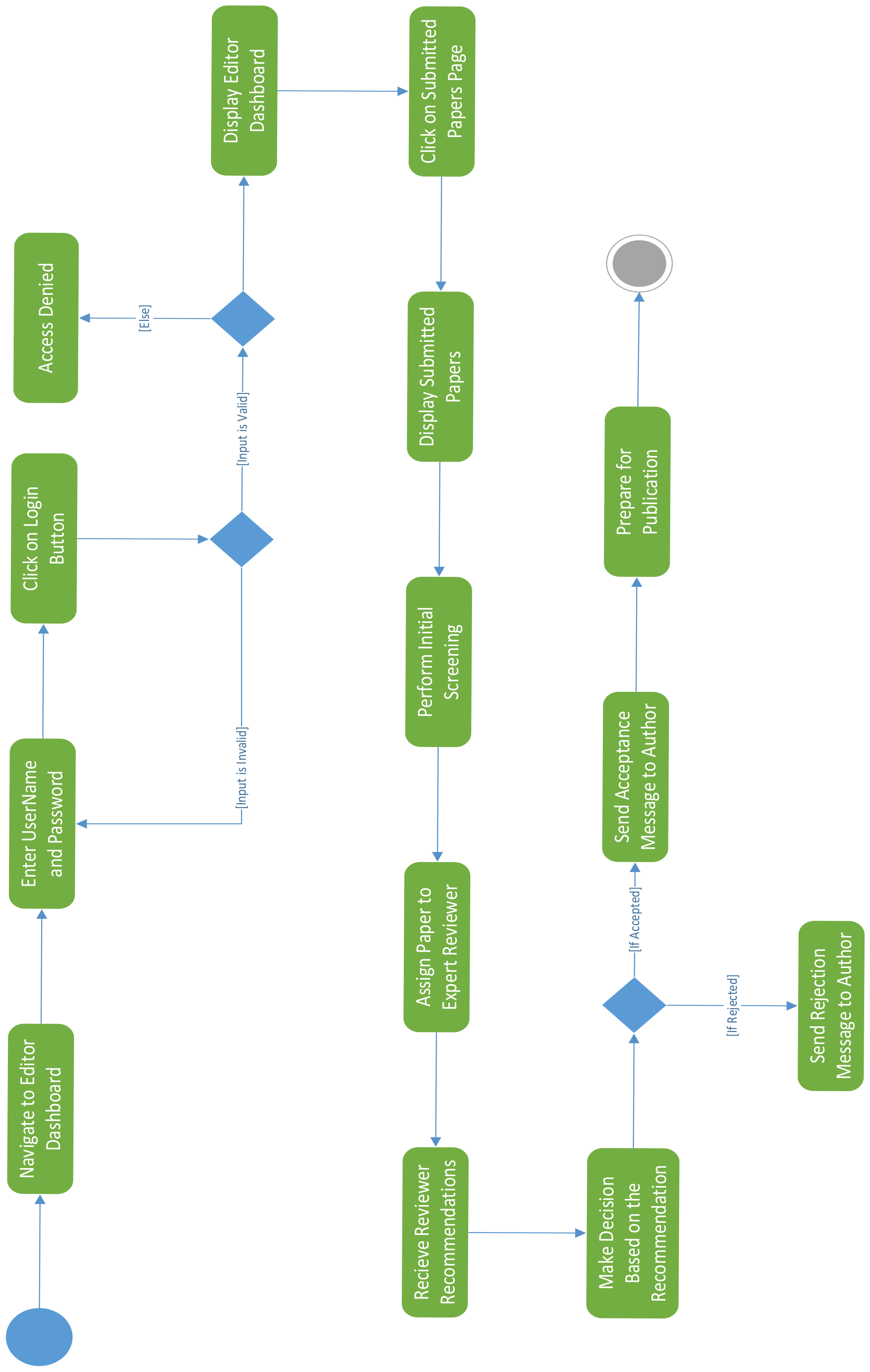


Figure 16 Login Activity Diagram



Open System
Navigate to the Home Page
Fill in the Search Bar
Click on the Search Button
Search DataBase For Paper
[Search Found]
[Search Not Found]
Display Search Not Found Message
Display Searched Paper

Figure 17 Search Paper Activity Diagram



Navigate to Editor Dashboard
Enter UserName and Password
Click on Login Button
[Input is Invalid]
[Input is Valid]
[Else]
Access Denied
Display Editor Dashboard
Click on Submitted Papers Page
Display Submitted Papers
Perform Initial Screening
Assign Paper to Expert Reviewer
Recieve Reviewer Recommendations
Make Decision Based on the Recommendation
[If Accepted]
[If Rejected]
Send Rejection Message to Author
Send Acceptance Message to Author
Prepare for Publication

Figure 18 Publish Paper Activity Diagram

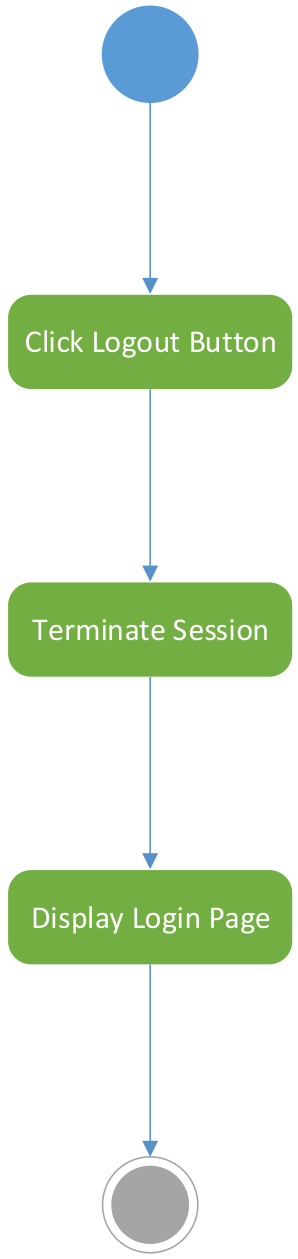


Figure 19 Logout Activity Diagram

4.3.2.2 Structural Diagram

Structural diagram provides a logical overview of all or parts of a software system. It acts as a look inside a given structured classifier, defining, its configuration classes, interfaces, packages, and the relationships between them at a micro-level. The structural diagram used in this project is Class diagram.

Class Diagram

Class diagram masgebat ezi gar

4.3.2.3 Entity Relationship Diagram

Our database design was made based on the entities drawn up from our initial product spec phase and those added later when we came up with a UI/UX sketch. Using this knowledge of entities involved we came up with an Entity-Relationship diagram (ER diagram) that represented entities and relations between them.

When designing the ER diagram what we have done is, we have laid out what all the entities involved in the design of this system are going to look like and the attributes each of them are going to have. This stage helped give us an obvious understanding of the number of entities we have and the relationship between them, which was a very foundational part of our project architecture when we got to the back-end implementation (specifically the model designing part) phase.

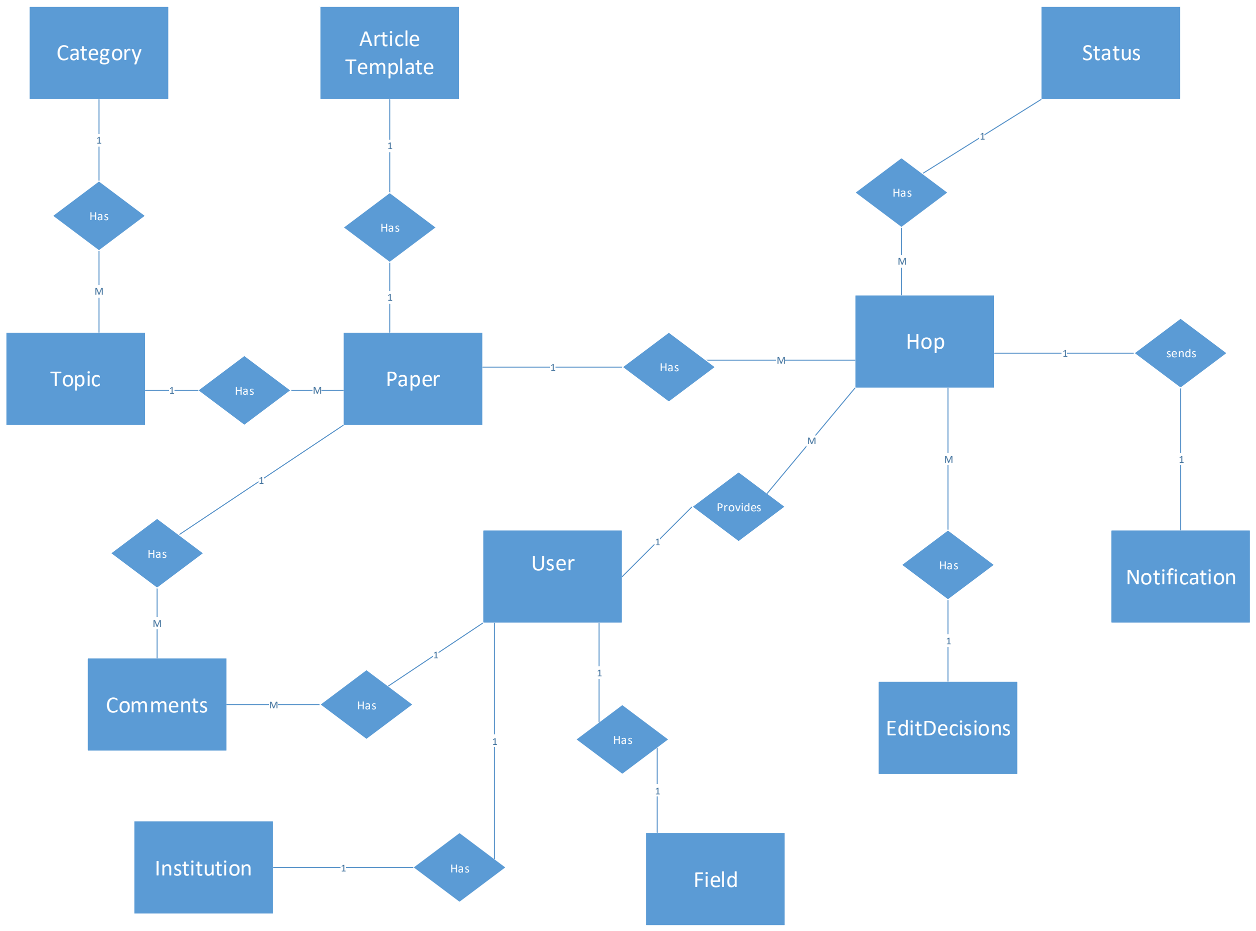


Figure 20 ER-Diagram

4.4 User Interface Design and Prototyping

Here we drew up the initial draft of what the user interface would look like along with the user experience that came with it. This step was crucial in our development because it drew out what our web page would look like in the end. In addition to that we used it to clearly state the fields, tables and columns that were required for our back-end implementation which came later. Doing this helped us visualize the needs that we were not aware of and we were able to fill those needs.

In order to get an overview of what our user interface will look like we had to decide on how many pages we are going to have and what features will have views of their own. After defining these properties, we decided that we will build the home page and other pages the end-user will access from scratch using react-js, while using templates in order to build the other dashboards and role specific pages. We searched online for bootstrap templates and the one we went with was Material UI template.

Material UI offers a comprehensive suite of UI tools that help us ship new features faster. We started with Material UI, using its fully-loaded component library, and we brought about our own design system to our production-ready components. It helped facilitate our design process by giving us an idea of what is to be implemented.

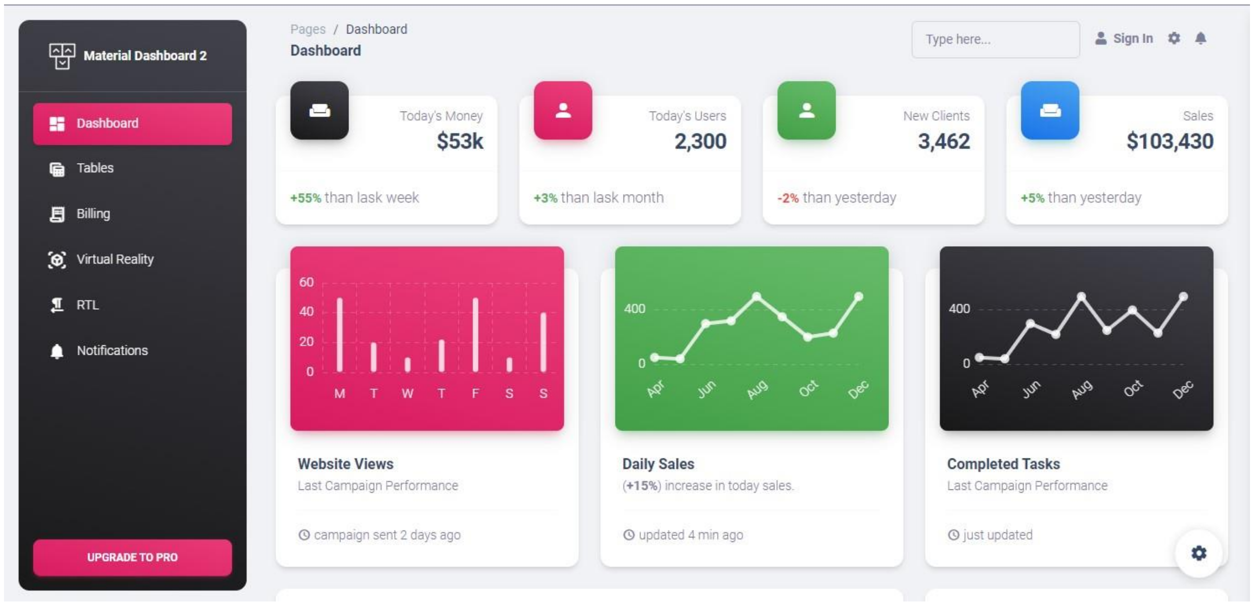


Figure 21 Dashboard Page

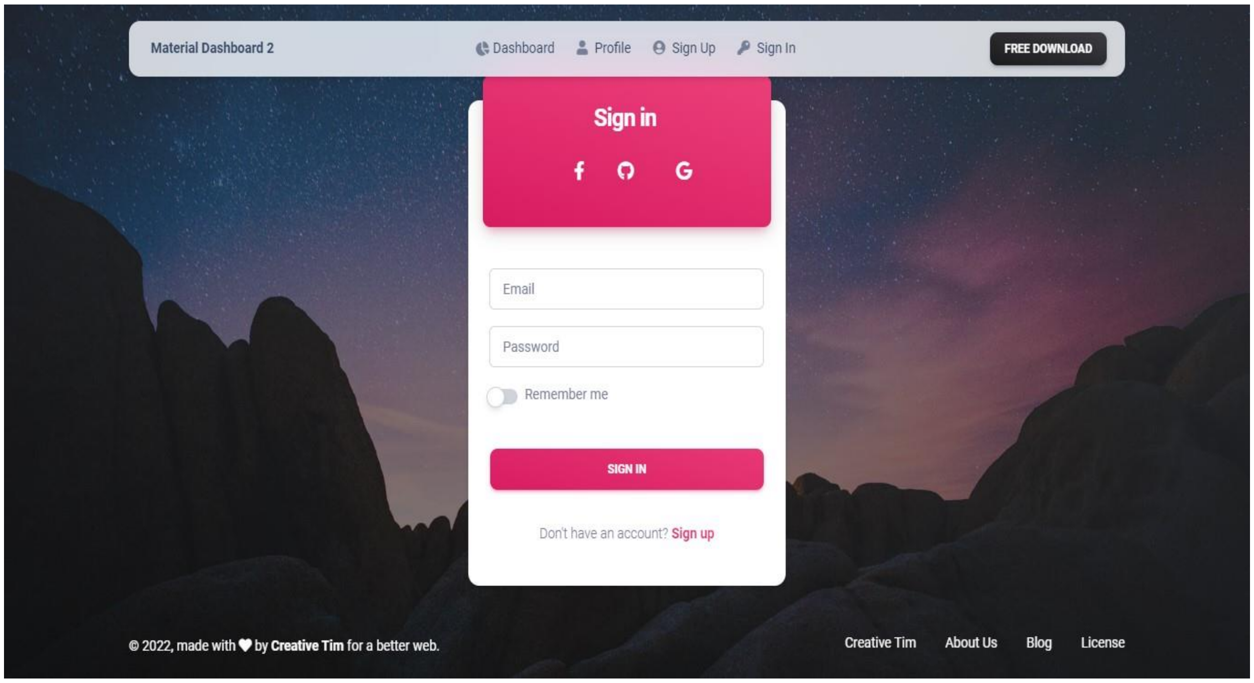


Figure 22 Sign in Page

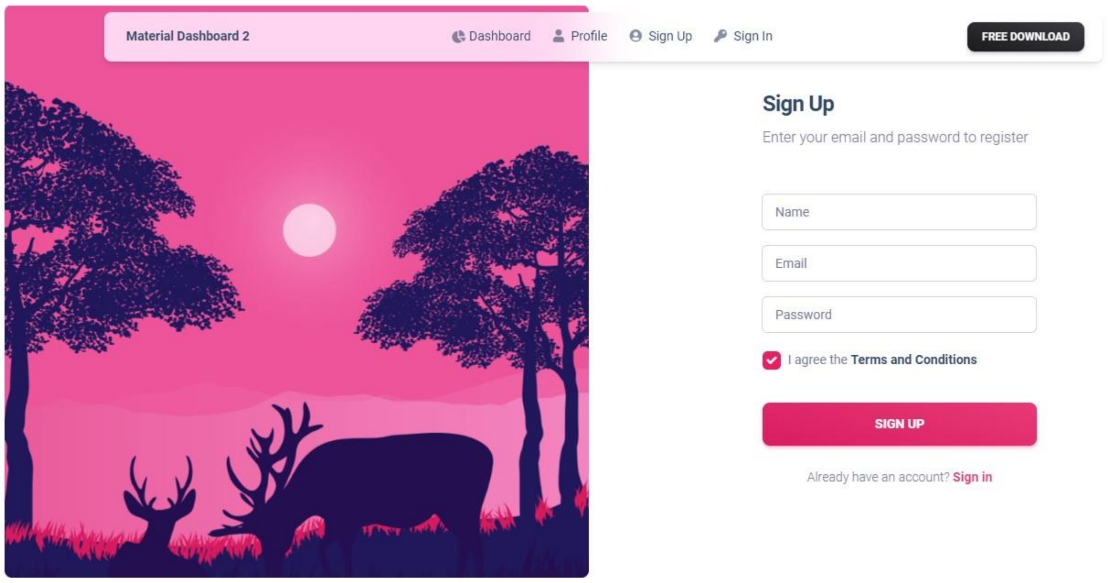
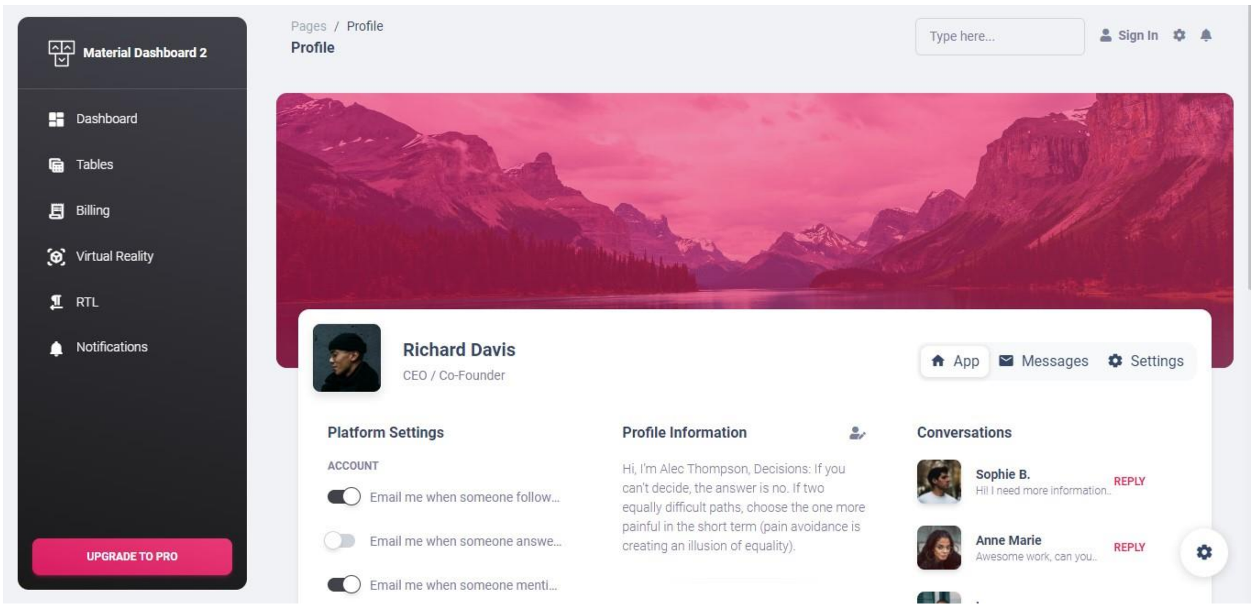

Material Dashboard 2
Dashboard
Profile
Sign Up
Sign In
FREE DOWNLOAD
Sign Up
Enter your email and password to register
Name
Email
Password
I agree the Terms and Conditions
SIGN UP
Already have an account? Sign in

Figure 23 Sign up Page



Richard Davis
CEO / Co-Founder

Platform Settings

ACCOUNT

Email me when someone follow...

Email me when someone answe...

Email me when someone menti...

Profile Information

Hi, I'm Alec Thompson, Decisions: If you can't decide, the answer is no. If two equally difficult paths, choose the one more painful in the short term (pain avoidance is creating an illusion of equality).

Conversations

Sophie B.
Hi! I need more information..

Anne Marie
Awesome work, can you..

Figure 24 Profile Page

The screenshots shown above depicts the template before we took the components according to our needs and modified them.

CHAPTER FIVE

IMPLEMENTATION

5.1 Introduction

After the completion of our system design in which we included the requirements analysis and system architecture planning phase we then proceeded to the implementation stage. In this stage we started with implementing the front-end, which helped us clearly visualize the final end goal of the project as defined in the system design section. After a while we then proceeded to work on the back-end implementation simultaneously.

5.2 Front-end Implementation

Since we already had a sketch of what our front-end would look like we used this rough sketch to build our UI accordingly. We used React.js to implement this part of the system. The final implementation of the front-end of our project is as follows

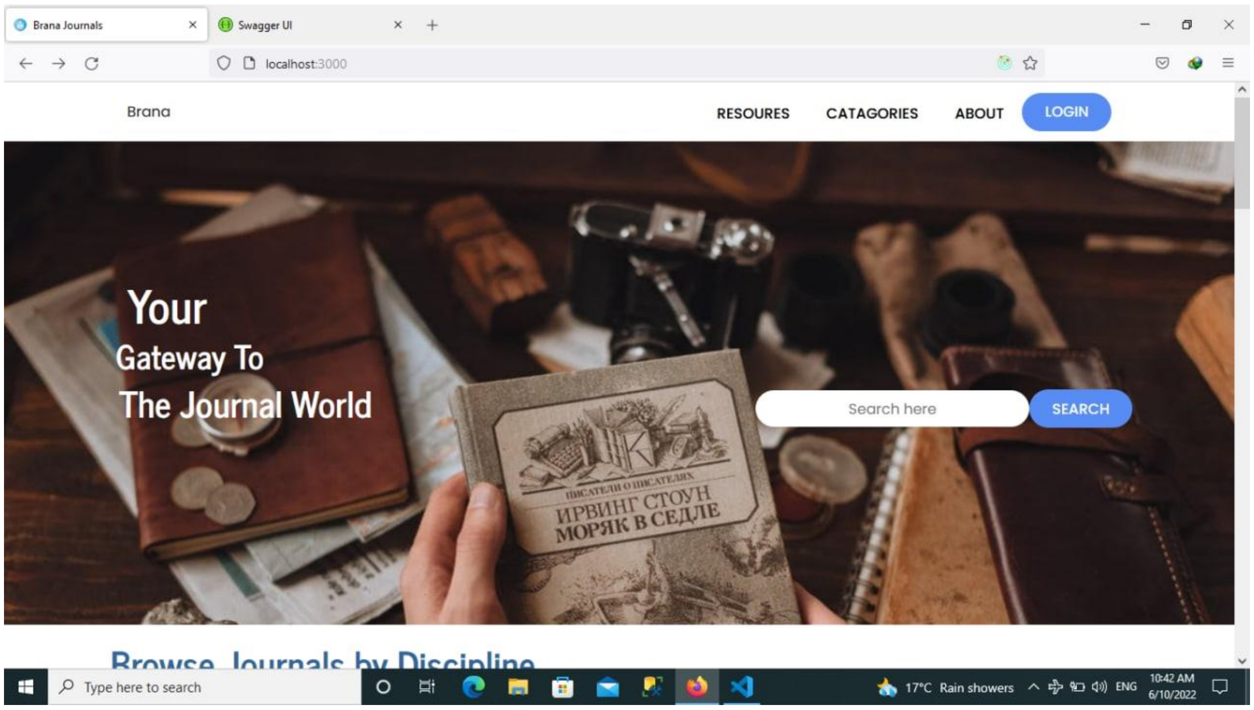


Figure 25 Home Page

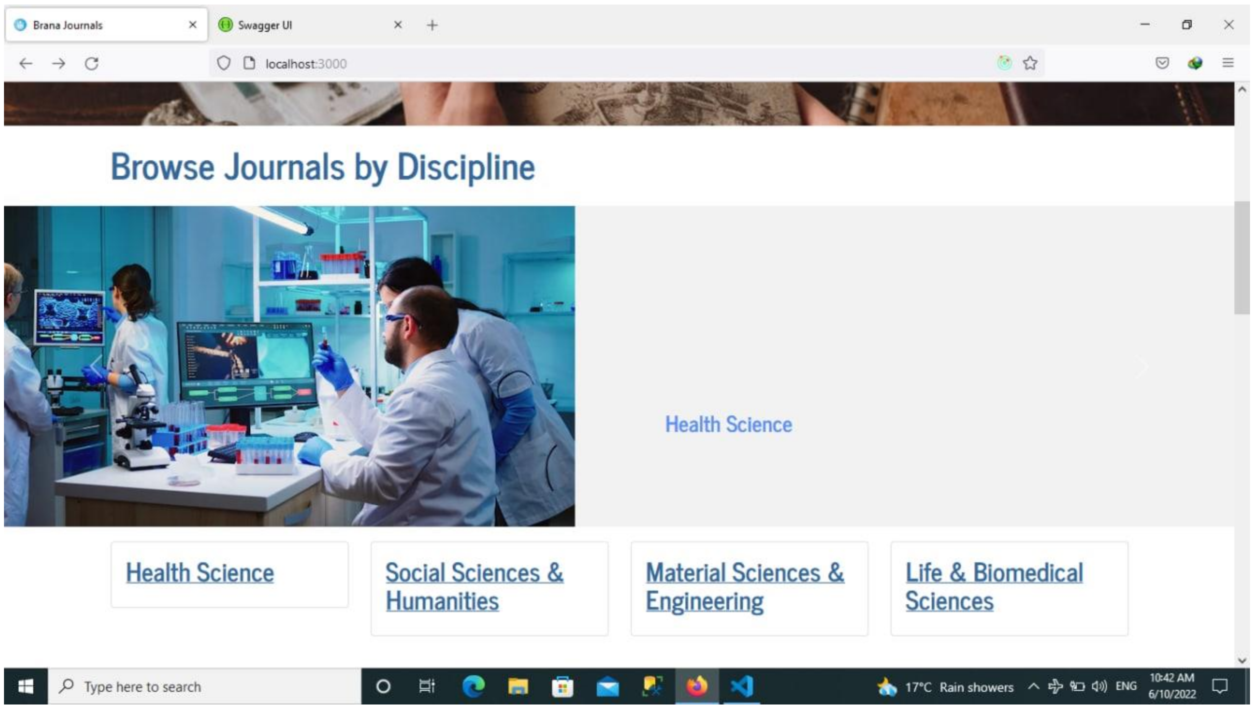
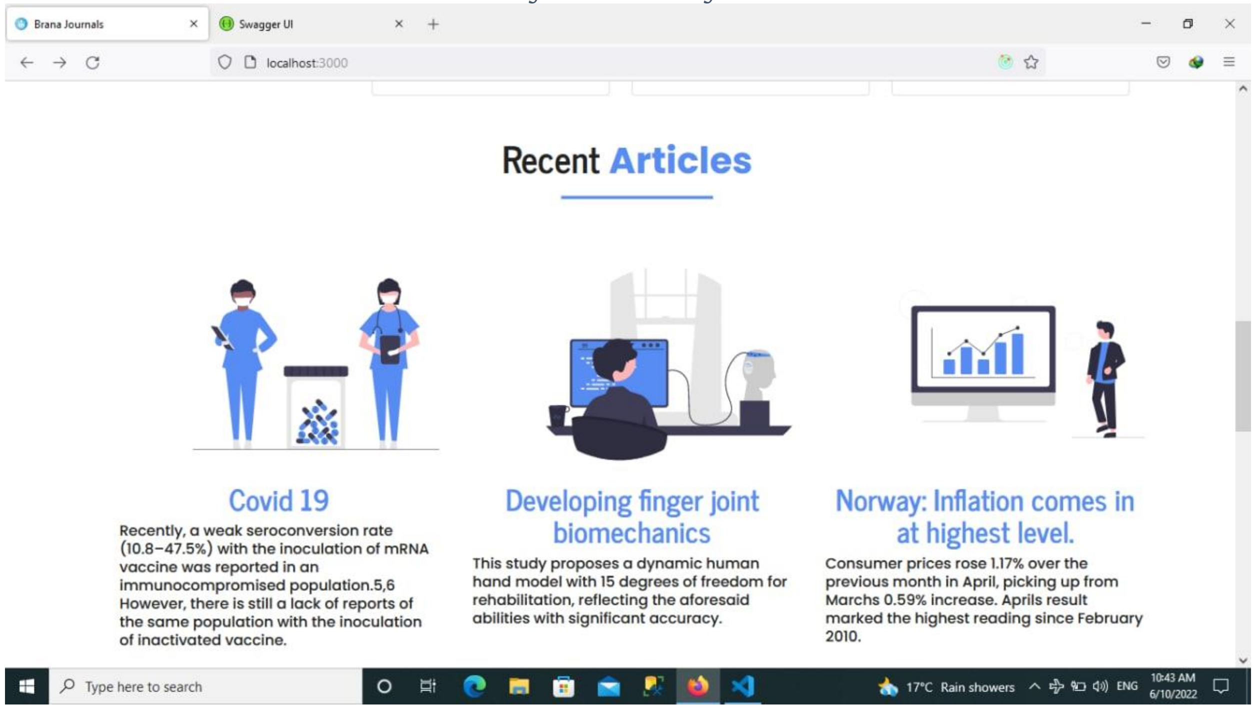


Figure 26 Home Page-2



Covid 19

Recently, a weak seroconversion rate (10.8–47.5%) with the inoculation of mRNA vaccine was reported in an immunocompromised population.5,6 However, there is still a lack of reports of the same population with the inoculation of inactivated vaccine.

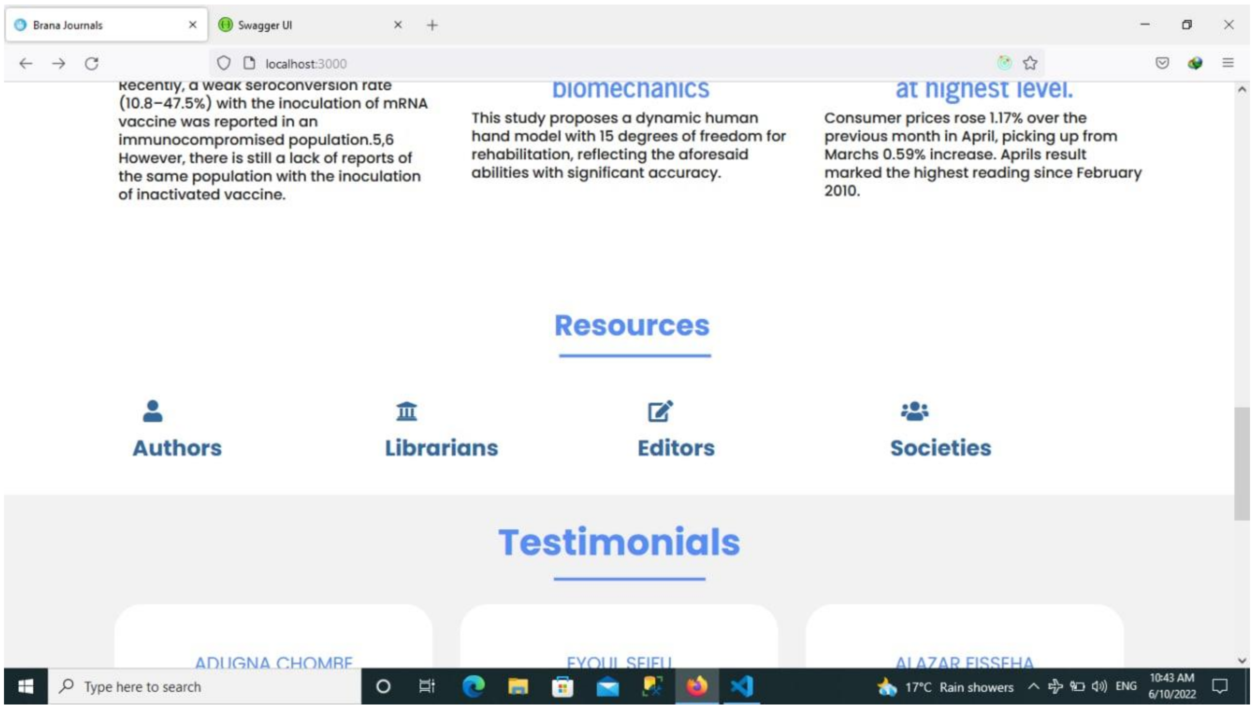
Developing finger joint biomechanics

This study proposes a dynamic human hand model with 15 degrees of freedom for rehabilitation, reflecting the aforesaid abilities with significant accuracy.

Norway: Inflation comes in at highest level.

Consumer prices rose 1.17% over the previous month in April, picking up from Marchs 0.59% increase. Aprils result marked the highest reading since February 2010.

Figure 27 Home Page-3



Recently, a weak seroconversion rate (10.8–47.5%) with the inoculation of mRNA vaccine was reported in an immunocompromised population.5,6 However, there is still a lack of reports of the same population with the inoculation of inactivated vaccine.

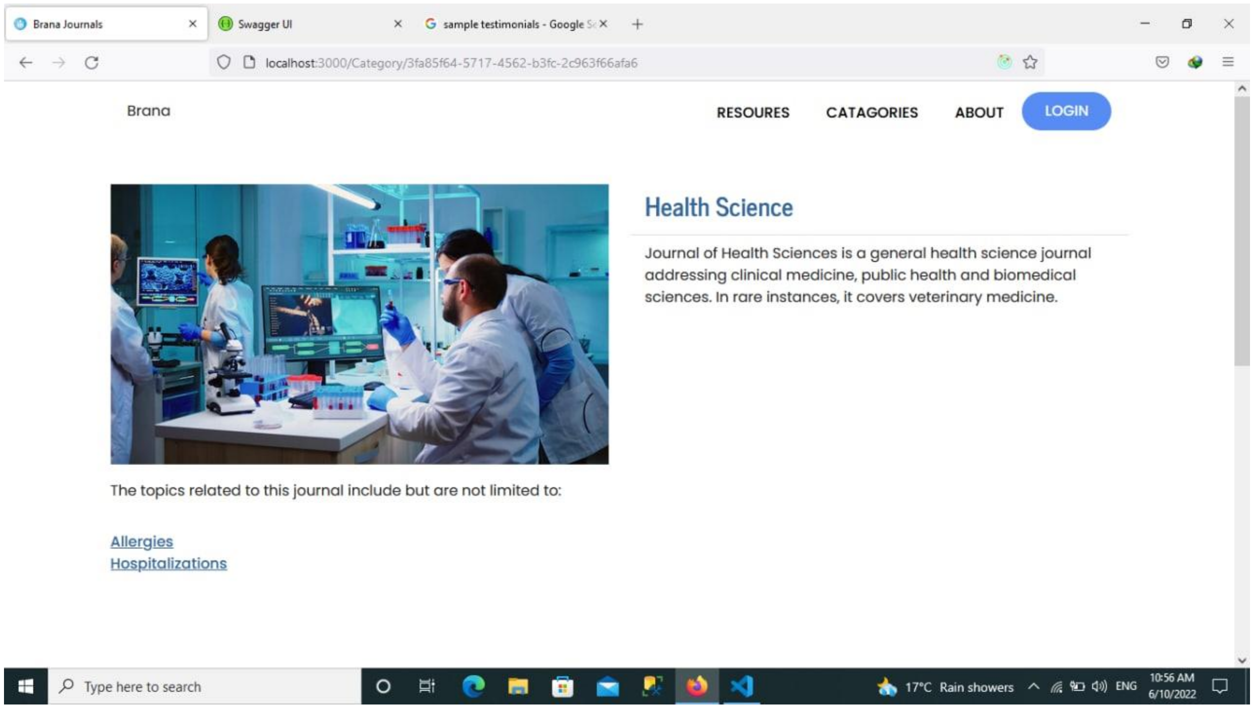
biomechanics

This study proposes a dynamic human hand model with 15 degrees of freedom for rehabilitation, reflecting the aforesaid abilities with significant accuracy.

at highest level.

Consumer prices rose 1.17% over the previous month in April, picking up from Marchs 0.59% increase. Aprils result marked the highest reading since February 2010.

Figure 28 Home Page-4



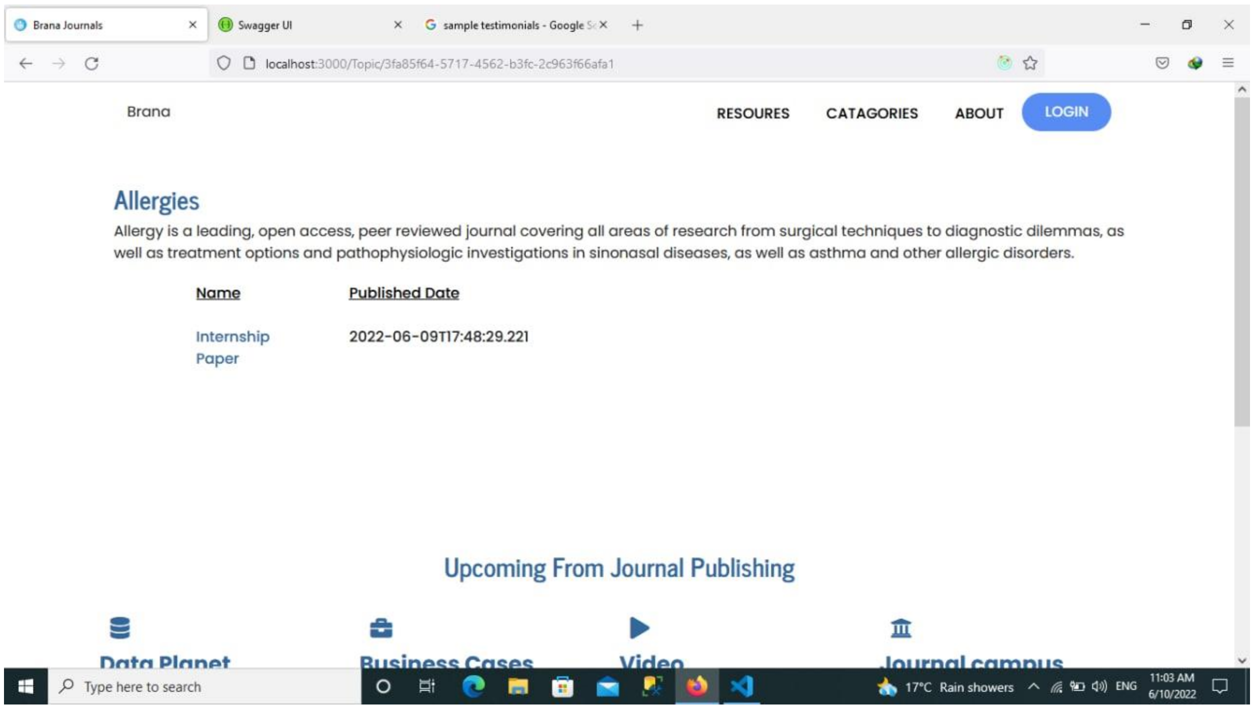
Health Science

Journal of Health Sciences is a general health science journal addressing clinical medicine, public health and biomedical sciences. In rare instances, it covers veterinary medicine.

The topics related to this journal include but are not limited to:

Allergies
Hospitalizations

Figure 29 Disciplines page


Brana Journals
Swagger UI
sample testimonials - Google S
localhost:3000/Topic/3fa85f64-5717-4562-b3fc-2c963f66afa1
Brana
RESOURES
CATAGORIES
ABOUT
LOGIN

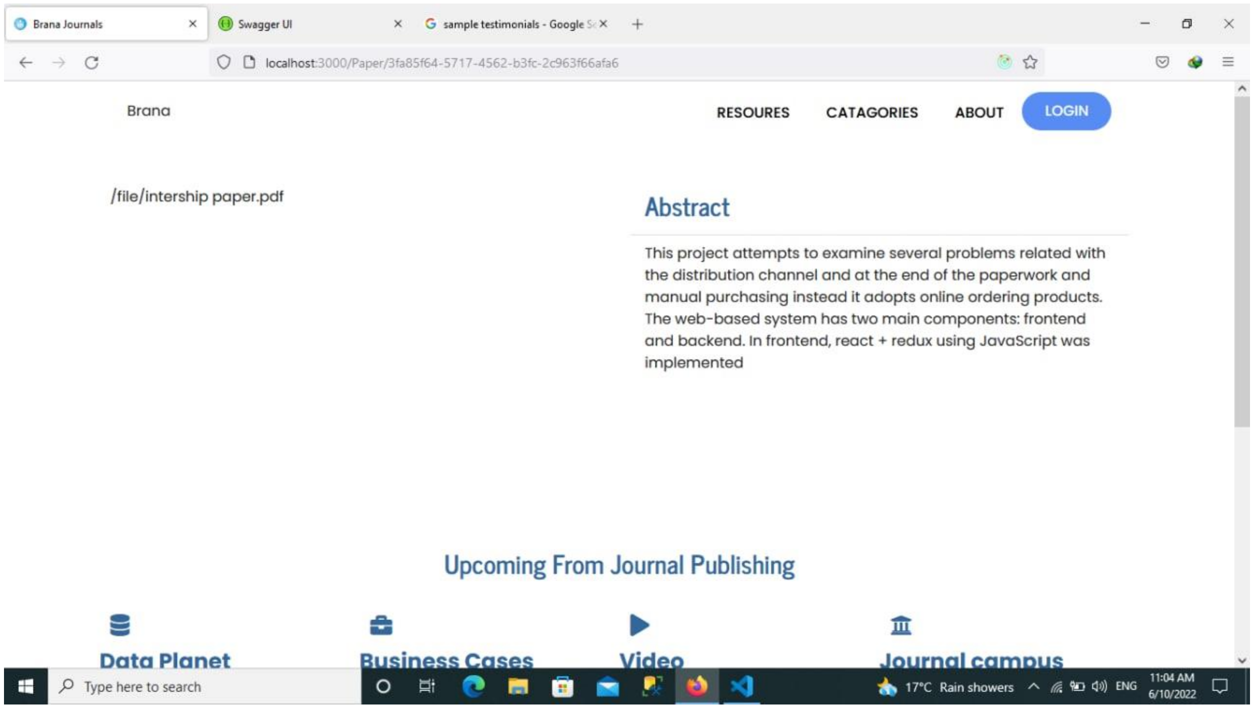
Allergies

Allergy is a leading, open access, peer reviewed journal covering all areas of research from surgical techniques to diagnostic dilemmas, as well as treatment options and pathophysiologic investigations in sinonasal diseases, as well as asthma and other allergic disorders.

Name	Published Date
Internship Paper	2022-06-09T17:48:29.221

Upcoming From Journal Publishing

Figure 30 Topics Page


Brana Journals
Swagger UI
sample testimonials - Google S
localhost:3000/Paper/3fa85f64-5717-4562-b3fc-2c963f66afa6
Brana
RESOURES
CATAGORIES
ABOUT
LOGIN

/file/intership paper.pdf

Abstract

This project attempts to examine several problems related with the distribution channel and at the end of the paperwork and manual purchasing instead it adopts online ordering products. The web-based system has two main components: frontend and backend. In frontend, react + redux using JavaScript was implemented

Upcoming From Journal Publishing

Figure 31 View Paper Page

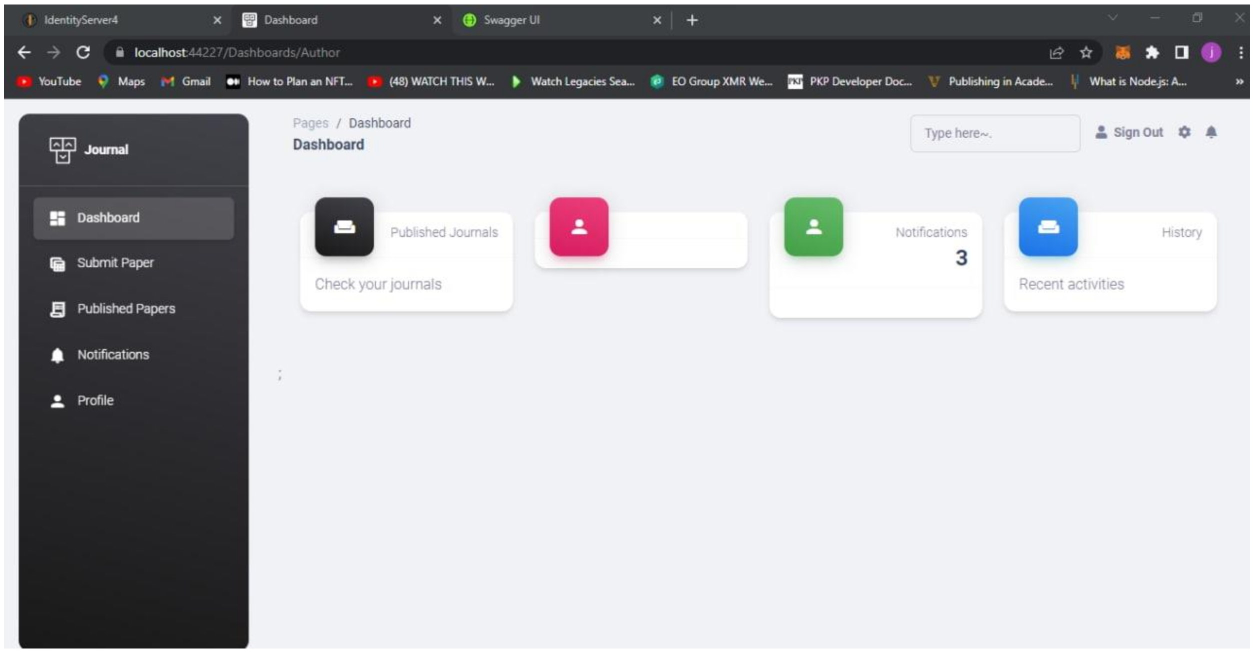


Figure 32 User Dashboard

5.3 Back-end Implementation

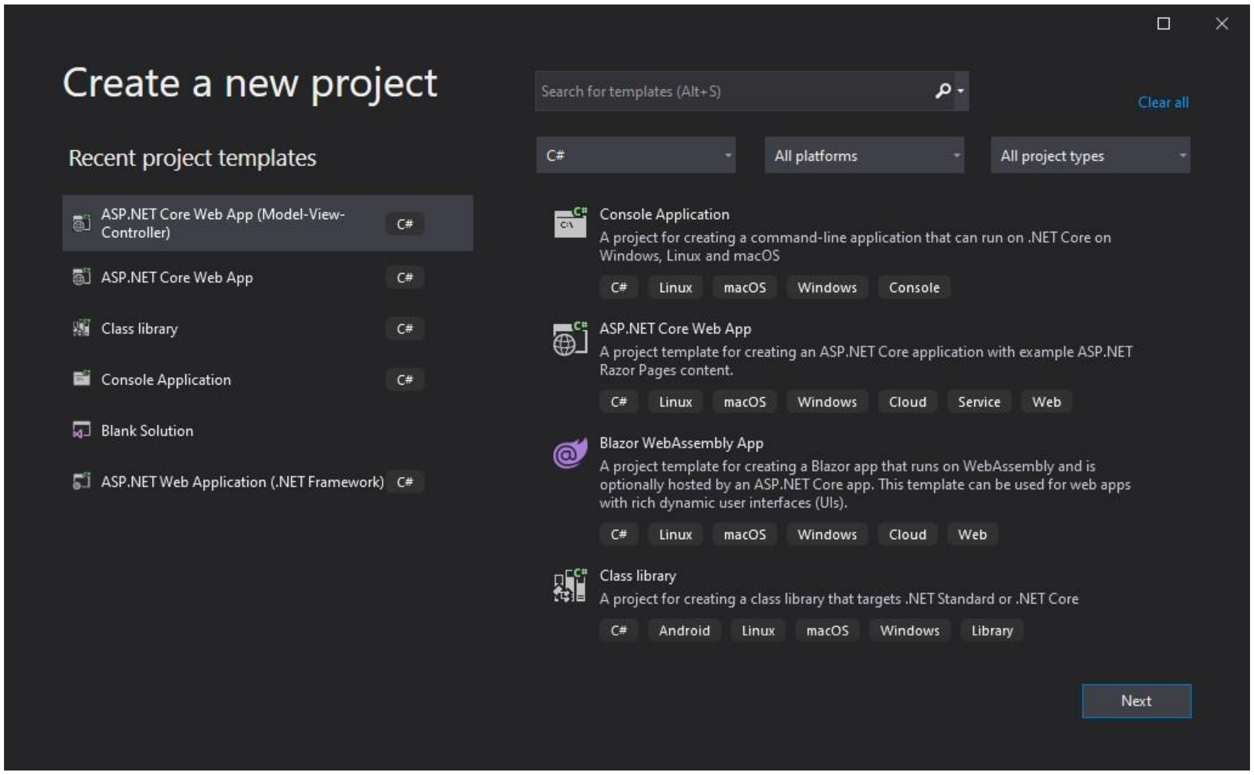
A system architecture is a conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. This representation of the system with all the different aspects was listed in section 4.2 of this chapter.

The critical link between the design and the engineering of requirements is to identify the main structural components in a system and their relationships. An architecture model describes how the system is organized as a set of communications components in the output of the architectural design process.

We used C# for the back-end design and implementation. This included the object model creation and database design for our system. Since the approach we chose is a code first approach the object model creation comes first and based on this object model we'll create migrations that will translate these changes into changes on the database. We used C# as the language with entity framework as the object-relational mapper.

We partitioned the back-end design into two major solutions one that handled authentication and authorization using identity server (OAuth) and another that handled our business logic. This helped create an abstraction between these layers which should be separate and helps maximize the security of the system.

The step-by-step implementations are as follows



Create a new project

Search for templates (Alt+S)

Clear all

Recent project templates

ASP.NET Core Web App (Model-View-Controller) C#

ASP.NET Core Web App C#

Class library C#

Console Application C#

Blank Solution

ASP.NET Web Application (.NET Framework) C#

C# | All platforms | All project types

Console Application
A project for creating a command-line application that can run on .NET Core on Windows, Linux and macOS
C# Linux macOS Windows Console

ASP.NET Core Web App
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
C# Linux macOS Windows Cloud Service Web

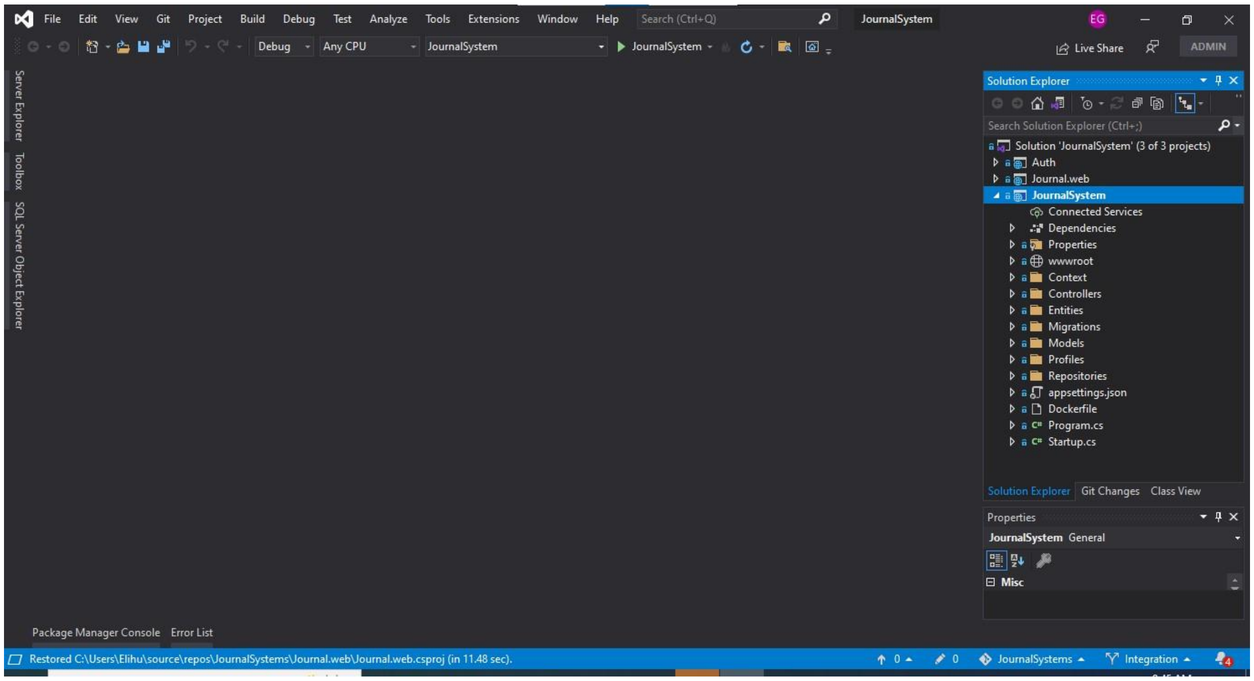
Blazor WebAssembly App
A project template for creating a Blazor app that runs on WebAssembly and is optionally hosted by an ASP.NET Core app. This template can be used for web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Cloud Web

Class library
A project for creating a class library that targets .NET Standard or .NET Core
C# Android Linux macOS Windows Library

Next

Figure 33 Create New Project

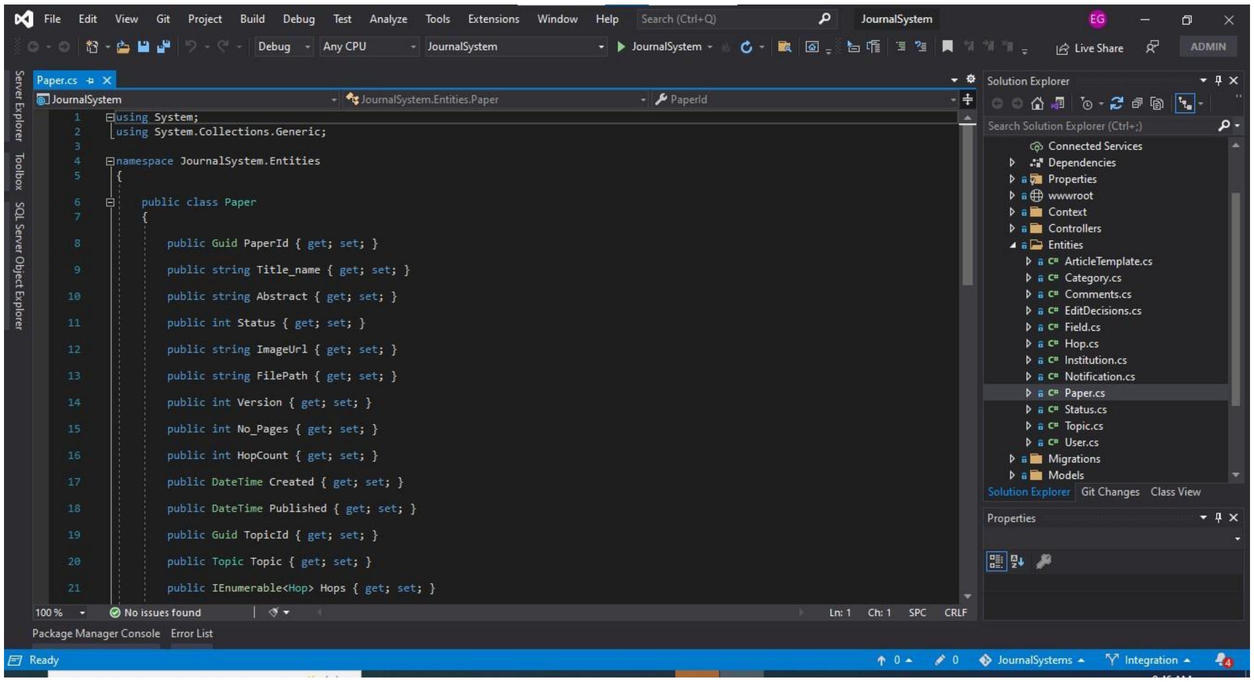
In this step we created a new web app which uses C# as its primary language and runs on any server (Windows, macOS or Linux).



```
Restored C:\Users\Elihu\source\repos\JournalSystems\Journal.web\Journal.web.csproj (in 11.48 sec).
```

Figure 34 Open the Solution

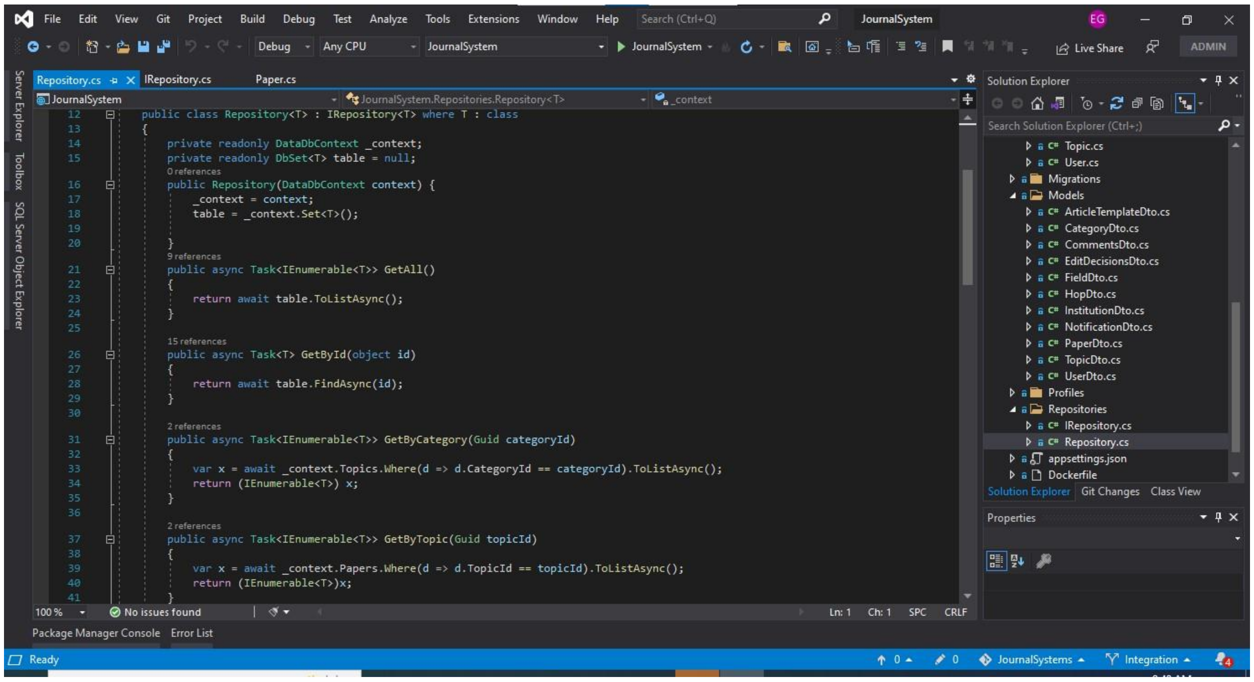
Once created we can navigate through the solution to view our models, our setup files and other configuration files.



```
using System;
using System.Collections.Generic;

namespace JournalSystem.Entities
{
    public class Paper
    {
        public Guid PaperId { get; set; }

        public string Title_name { get; set; }

        public string Abstract { get; set; }

        public int Status { get; set; }

        public string ImageUrl { get; set; }

        public string FilePath { get; set; }

        public int Version { get; set; }

        public int No_Pages { get; set; }

        public int HopCount { get; set; }

        public DateTime Created { get; set; }

        public DateTime Published { get; set; }

        public Guid TopicId { get; set; }

        public Topic Topic { get; set; }

        public IEnumerable<Hop> Hops { get; set; }
```

Figure 35 Create Models

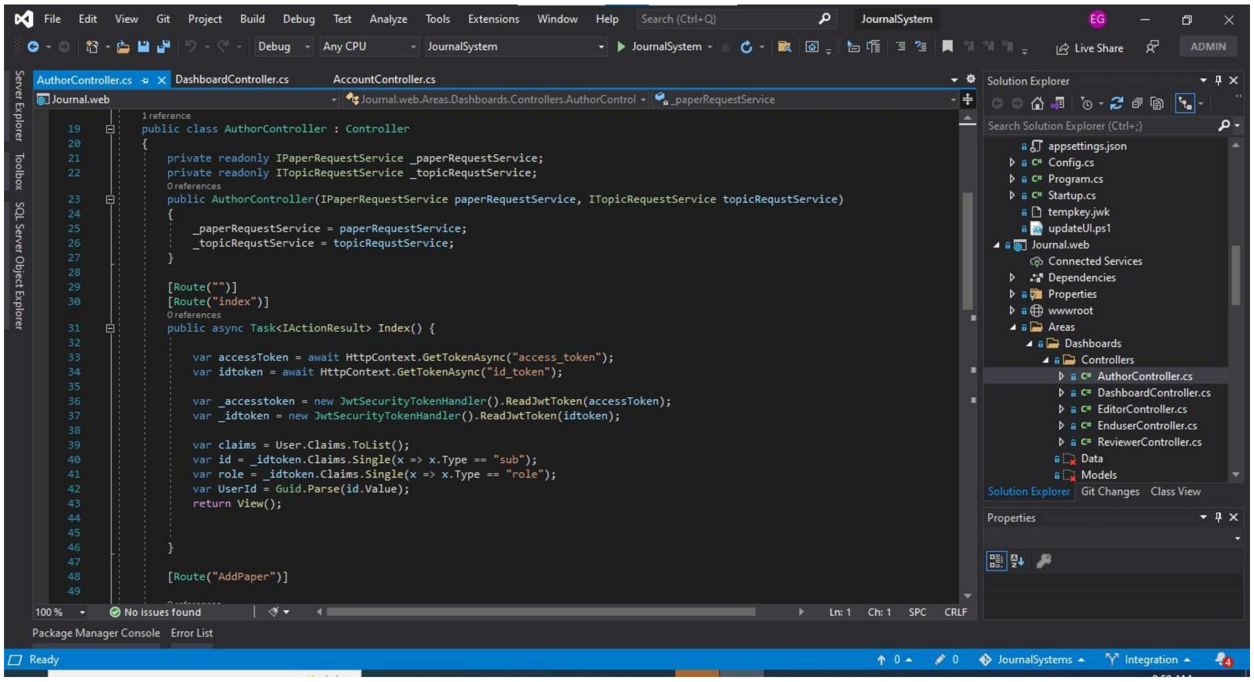
In the solution we will now create models and their views models as Dtos, along with that we'll create migrations for the system and apply them to our database.



```
public class Repository<T> : IRepository<T> where T : class
{
    private readonly DataDbContext _context;
    private readonly DbSet<T> table = null;
    0 references
    public Repository(DataDbContext context) {
        _context = context;
        table = _context.Set<T>();

    }
    9 references
    public async Task<IEnumerable<T>> GetAll()
    {
        return await table.ToListAsync();
    }

    15 references
    public async Task<T> GetById(object id)
    {
        return await table.FindAsync(id);
    }

    2 references
    public async Task<IEnumerable<T>> GetByCategory(Guid categoryId)
    {
        var x = await _context.Topics.Where(d => d.CategoryId == categoryId).ToListAsync();
        return (IEnumerable<T>) x;
    }

    2 references
    public async Task<IEnumerable<T>> GetByTopic(Guid topicId)
    {
        var x = await _context.Papers.Where(d => d.TopicId == topicId).ToListAsync();
        return (IEnumerable<T>)x;
    }
```

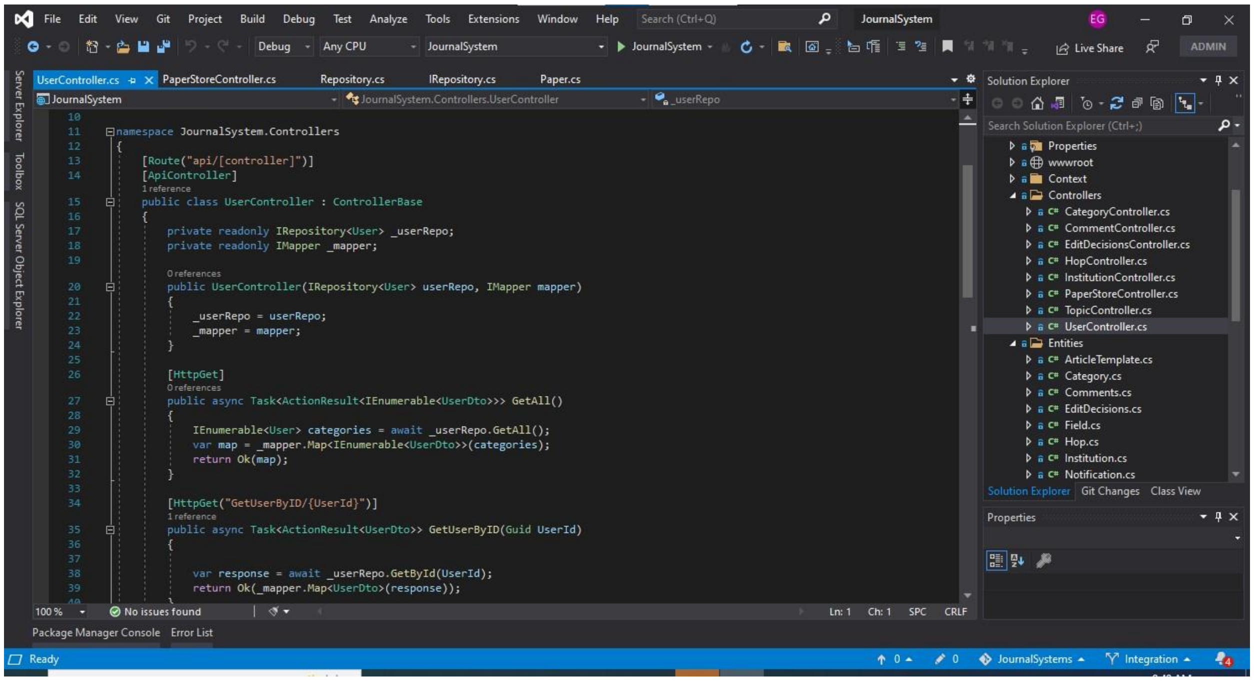
Figure 36 Repository Pattern

We set up a repository file that will handle all our request going through the web traffic, meaning we will handle the implementations of HTTP methods here.



```
1 reference
public class AuthorController : Controller
{
    private readonly IPaperRequestService _paperRequestService;
    private readonly ITopicRequestService _topicRequstService;
    0 references
    public AuthorController(IPaperRequestService paperRequestService, ITopicRequestService topicRequstService)
    {
        _paperRequestService = paperRequestService;
        _topicRequstService = topicRequstService;
    }

    [Route("")]
    [Route("index")]
    0 references
    public async Task<IActionResult> Index() {

        var accessToken = await HttpContext.GetTokenAsync("access_token");
        var idtoken = await HttpContext.GetTokenAsync("id_token");

        var _accesstoken = new JwtSecurityTokenHandler().ReadJwtToken(accessToken);
        var _idtoken = new JwtSecurityTokenHandler().ReadJwtToken(idtoken);

        var claims = User.Claims.ToList();
        var id = _idtoken.Claims.Single(x => x.Type == "sub");
        var role = _idtoken.Claims.Single(x => x.Type == "role");
        var UserId = Guid.Parse(id.Value);
        return View();


    }

    [Route("AddPaper")]
```

Figure 37 Author Controller



```csharp
namespace JournalSystem.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class UserController : ControllerBase
    {
        private readonly IRepository<User> _userRepo;
        private readonly IMapper _mapper;

        public UserController(IRepository<User> userRepo, IMapper mapper)
        {
            _userRepo = userRepo;
            _mapper = mapper;
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<UserDto>>> GetAll()
        {
            IEnumerable<User> categories = await _userRepo.GetAll();
            var map = _mapper.Map<IEnumerable<UserDto>>(categories);
            return Ok(map);
        }

        [HttpGet("GetUserByID/{UserId}")]
        public async Task<ActionResult<UserDto>> GetUserByID(Guid UserId)
        {

            var response = await _userRepo.GetById(UserId);
            return Ok(_mapper.Map<UserDto>(response));
```

Figure 38 User Controller

Shown above is the setup of two of the many controllers designed. These controllers are the locations in which we create the api endpoints. One of them further implements user authorization which is why we felt the need to include them both.

5.4 Testing

The primary objective for test case design is to derive a set of tests that has the highest likelihood for uncovering defects in our software. To accomplish this, we are doing

- **Unit Test** – the task of development is divided into small units (which in our case is that the units could be developed by different members of our team) and those units are going to be tested separately if they behave as they are expected.

- **Integration Testing** – in integration testing, there could be various stages of testing in our proposed system. First, when the units are integrated into a bigger component, an integration test is going to be applied by the team member developing those components. And secondly, those components are going to be integrated with components developed by other team members and integration test is going to be applied here again to ensure that the integrated components behave as required.

- **System Testing** is going to be applied to test the whole system after integrating all the subcomponents.

- Finally we did, **Acceptance Testing** - under this testing there are another sub testing which would be performed: Alpha testing and Beta testing.

Alpha Testing: when our system has been completed, we will examine it by actual users and an independent test team. Hence, representative of the user would come to us and test the system by himself whether it meets their need or not.

Beta Testing: the system will be tested by the users at their own working place whether it meets their needs or not.

CHAPTER SIX

CONCLUSION

Technology has made significant progress over the years to provide various people across the globe easy and free access to up-to date knowledge. This project proposes to bring about these advances into the publishing and research scene making them readily available using state of the art methods. The main problems it solves are those of accessibility of local researches and provides researchers with a chance to be able to actively track and monitor the state of their research review.

The designed journal management system

References

1. Markella-Elpida Tsichl. The Visual Arts in a New Era: Digital Art: Volume 7, Issue 1, 1-5 Pages: American Research Journal of Humanities and Social Sciences
2. Open Society Institute, A Guide to Institutional Repository Software
3. The Benefits of the Ethereum Blockchain | by Marcus Soulsby | Plutus | Medium 'https://medium.com/plutus/the-benefits-of-the-ethereum-blockchain-f332e62f7659'
4. Next.js by Vercel - The React Framework (nextjs.org) Link 'https://nextjs.org/'
5. Polygon Technology | Documentation Link 'https://docs.polygon.technology/'
6. React documentation Link 'reactjs.org'