

## 1. INTRODUCTION

### 1.1 SYNOPSIS

Most modern social media platforms are usually owned and operated by big corporations. These platforms often employ algorithms to rank content favourable for their advertisement partners but not their users. This kind of model is called an advertising-centric product model. These kinds of products are not palatable for normal users as they are not in control over the content they are seeing. Some platforms provide a false sense of control by employing features like ‘add friend’ or ‘subscribe’, but they still don't stick purely to the subscription model. This leads to customer dissatisfaction. In an audience-centric product, the user should have total control over the content they see. This not only includes which posts are shown but also their order of appearance. Thus, to determine the order of the posts, some metric needs to be maintained, which is audience-centric. Making this unique for each user would be harder as the user would have to rate multitudes of posts. The most practical solution for this becomes a community-centric algorithm. This uses other users’ explicit expression of their liking/disliking of a post to construct a sortable metric. The algorithm also needs to take into account the age of the post, so that newer posts are given a chance to get exposure instead of being buried under popular but old posts. This can be done using the ‘hot ranking algorithm’ for the posts so that newer posts are given a higher ranking than older posts, all the while also using the like/dislike counts to assign the score. The comment section of the posts however needs to have a different algorithm so that it does not depend on the age of the comment, since better comments should always be present on the top. This can be achieved by using the ‘confidence sort’ using the lower bound of the ‘Wilson score interval’. This is similar to social platform ‘Reddit’ but differs from it as it uses a subscription-based model of users following other users instead of sourcing through the entire pool of users. Our project deals with these shortcomings and provides a rich, user-friendly, and audience-centric platform that lets users be in-charge of the posts and comments they see.

### 1.2 PROBLEM DEFINITION

Current Social Media Platforms are company / advertiser centric. The content ranking algorithm used by them is proprietary and advertiser-centric and does not benefit the user. Users want to be in control of the posts they see. Users want posts to be ranked according to their merit and age. Users want to be able to follow other users. These are not possible with current platforms. Either the platform does not

prioritize the user, or it doesn't let the user choose whom to follow. The merits of both the types of platforms need to be combined into one platform, devoid of the drawbacks of all of them.

### **1.3 SCOPE OF THE PROJECT**

SCSP tries to create a diverse and equal platform for all users where content is sorted algorithmically using their merit, decided by the community. Users get to choose who they follow. We device a fair algorithm that is based on community taste of a post. New posts should outrank old ones to avoid oversaturation, newer posts should be given higher priority over old posts. User should be able to express opinion let user not only express liking, but also express their disliking of a post or comment, for better ranking. SCSP devices a mix of social platforms, using all the benefits and discarding all disadvantages.

**1.4 MODULES IN THE SOFTWARE:**

1. **Discover** - View content shared by peers whom you follow. Content is sorted so that newer and better content is shown first. User can like, dislike, and comment on the posts, as well as like and dislike the comments of the posts.
2. **Messaging** - Chat with your peers or anyone else in the world. Chats are asynchronous and stored on a centralised server.
3. **User Account** - View / Change account details like name, bio, and upload your display picture. Users can only change their own details post authentication. Users without a display picture are provided with a dynamically generated display picture made of their initials. User can also delete their own account.
4. **Content Upload and Post** - Let users upload and share content. Store the content in the database and let it be accessible to all other users who follow them.
5. **Notification Centre** - View and open notifications received by a user pertaining to his account and posts. Get notified about peers commenting on your posts.
6. **Settings** - Change site settings on a per-user basis. Toggle between light and dark mode. Clear cache and stored cookies.
7. **Donate** - Support the application with donation. Donations are optional and do not affect any functionality of the application.
8. **Explore** – Find new users to follow and see their user profile and posted images.
9. **Admin Dashboard** - Show statistics and data about users, registrations, posts, comments, and other similar insights about the platform.

## **2. SYSTEM STUDY**

### **2.1 EXISTING SYSTEM**

Currently multitudes of social media platforms exist and listing all of them would not be practically possible. However, most of the products like Facebook, Instagram, LinkedIn, etc focus on proprietary algorithms that favour advertisers over users. These algorithms are made to keep the users on the app for longer periods of time and/or to make them see advertisements and click on them. These platforms do not respect the user's time and energy, hence the need of an audience-centric platform. Platforms like Reddit, 4chan, etc are more user-centric but are not social platforms per-se. These platforms either show posts from all users across the globe, or certain sandboxed community pages, which only pertain to a specific topic/thing/person/place. This does not allow the user to follow other users and see only posts made by them, losing the social aspect. These shortcomings are addressed by combining the follower-followee model and the merit-based ranking model.

#### **Problems in Existing System:**

1. The algorithm for ranking is not audience-centric.
2. Users cannot subscribe to individual users.
3. Unpredictable behaviour of the platform.

### **2.2 FEASIBILITY STUDY**

A feasibility study is an analysis of how successfully a project can be completed, accounting for factors that affect it such as economic, technological, legal and scheduling factors. Project managers use feasibility studies to determine potential positive and negative outcomes of a project before investing a considerable amount of time and money into it. A feasibility study tests the viability of an idea, a project or even a new business. The goal of a feasibility study is to place emphasis on potential problems that could occur if a project is pursued and determines if, after all significant factors are considered, the project should be pursued. Feasibility studies also allow a business to address where and how it will operate, potential obstacles, competition and the Funding needed to get the business up and running. Every project is feasible for given unlimited resources and infinite time. Feasibility study is an evaluation of the proposed system regarding its workability, impact on the organization, ability to meet the user needs and effective use of resources. Thus, when a new application is proposed it normally

goes through a feasibility study before it is approved for development. Feasibility and risk analysis are related in many ways. The feasibility analysis in this project has been discussed below based on the above-mentioned components of feasibility.

This project "**Social Content Sharing Platform**" has undergone the following Feasibility study:

- Technical Feasibility
- Economic Feasibility
- Behavioural Feasibility
- Schedule Feasibility

### **1. Technical feasibility:**

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. The project is technically feasible as it doesn't have any known technical bugs or glitches, as shown by the validation testing.

### **2. Economic feasibility:**

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility—helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide. The project is currently technically feasible as it is hosted for free. It can be deployed on a free tier of any deployment platform like Heroku, replit, etc.

### **3. Behavioural feasibility:**

Behavioural feasibility is the analysis of behaviour of the computerized system. In this we analysis that the computerized system is working properly or not. If working then it is communicating

properly with the environment or not. All the matters are analysed and a good computerized system is prepared. The project is currently behaviourally feasible.

#### **4. Schedule feasibility:**

This assessment is the most important for project success; after all, a project will fail if not completed on time. In scheduling feasibility, an organization estimates how much time the project will take to complete. The project SCSP is feasible as it can be developed in a short amount of time (one semester).

### **2.3 PROPOSED SYSTEM**

The proposed system combines the advantages of existing solutions while not incorporating its shortcomings. The project uses a merit-based ranking system with/without considering the age of the item. This does not incorporate any external variables in the ranking of posts, such as advertiser interest, as the platform does not run on ads. The solution utilises the ‘hot ranking algorithm’ for ranking posts using their submission date and time as well as their like and dislike counts as metrics. This causes posts to ‘age’ naturally and slowly rank lower than newer posts. Comments on the platform utilise the ‘confidence sort’ which utilises the ‘Wilson score interval’ to statistically test the hypothesis of how good the comment is, based on the like-dislike data available. This is better than using an absolute score or absolute average score directly from the like-dislike counts. This prevents conflicting situations like posts with one like and zero dislikes being ranked higher than posts with hundreds of likes and one dislike. This causes a more organic flow of comments which are worthy of being shown at the top. The system utilises the ‘subscribe’ model to filter contents to show. This lets the user be in complete control of what they see and what they don’t. Users only see content from people they explicitly ‘follow’.

#### **Advantages over Existing System:**

1. Ranks content systematically based on their merit and age.
2. It lets users choose their circle and author of the posts they see.
3. Incentivises content creation and engaging in liking/disliking of other’s content to improve the community.
4. Incorporates transparency in content ranking algorithm.

### 3. SYSTEM DESIGN

In the design phase the architecture is established. This phase starts with the requirement document delivered by the requirement phase and maps the requirements into architecture. The architecture defines the components, their interfaces and behaviours. The deliverable design document is the architecture. The design document describes a plan to implement the requirements. This phase represents the “how” phase. Details on computer programming languages and environments, machines, packages, application architecture, distributed architecture layering, memory size, platform, algorithms, data structures, global type definitions, interfaces, and many other engineering details are established. The design may include the usage of existing components. Analysing the trade-offs of necessary complexity allows for many things to remain simple which, in turn, will eventually lead to a higher quality product. The architecture team also converts the typical scenarios into a test plan. In our approach we have given a complete requirement document and indicated critical priorities for the implementation team. A critical implementation priority leads to a task that has to be done right. If it fails, the product fails. If it succeeds, the product might succeed. At the very least, the confidence level of the team producing a successful product will increase. This will keep the implementation team focused. Exactly how this information is conveyed is a skill based on experience more than a science based on fundamental foundations. System design is the process of defining the architecture components, modules, interfaces, and data for a system to satisfy specified requirements. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. If the broader topic of product development “blends the perspective of marketing, design, and manufacturing into a single approach to product development,” then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the Process of defining and developing systems to satisfy specified requirements of the user. Until the 1990s, systems design had a crucial and respected role in the data processing industry. In the 1990s, standardization of hardware and software resulted in the ability to build modular systems. The increasing importance of software running on generic platforms has enhanced the discipline of software engineering. Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design. The UML has become the standard language in object-oriented analysis and design. It is widely used for modelling software systems and is increasingly used for high designing non-software systems and organizations.

**LOGICAL DESIGN:**

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modelling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems, designs are included. Logical design includes entity-relationship diagrams (ER diagrams).

**PHYSICAL DESIGN:**

The physical design relates to the actual input and output processes of the system. This is explained in terms of how data is input into a system, how it is verified /authenticated, how it is processed, and how it is displayed. In physical design, the following requirements about the system are decided.

1. Input requirement,
2. Output requirements,
3. Storage requirements,
4. Processing requirements,
5. System control and backup or recovery.

Put another way, the physical portion of system design can generally be broken down into three subtasks:

- 1.** User Interface Design
- 2.** Data Design
- 3.** Process Design

**User Interface:** It is concerned with how users add information to the system and with how the system presents information back to them.

**Data Design:** It is concerned with how the data is represented and stored within the system.

**Process Design:** It is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system.



At the end of the system design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase. Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and control processor. The H/S personal specification is developed for the proposed system.

### 3.1 E-R DIAGRAM

An Entity Relationship Diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases. At first glance an entity relationship diagram looks very much like a flowchart. It is the specialized symbols, and the meanings of those symbols, that make it unique. Because this ER tutorial focuses on beginners below are some tips that will help you build effective ER diagrams:

- Identify all the relevant entities in a given system and determine the relationships among these entities.
- An entity should appear only once in a particular diagram.
- Provide a precise and appropriate name for each entity, attribute, and relationship in the diagram.

Terms that are simple and familiar always beats vague, technical-sounding words. In naming entities, remember to use singular nouns. However, adjectives may be used to distinguish entities belonging to the same class (part-time employee and full-time employee, for example). Meanwhile attribute names must be meaningful, unique, system independent, and easily understandable.

- Remove vague, redundant or unnecessary relationships between entities.
- Never connect a relationship to another relationship.
- Make effective use of colors. You can use colors to classify similar entities or to highlight key areas in your diagrams.

You can draw entity relationship diagrams manually, especially when you are just informally showing simple systems to your peers. However, for more complex systems and for external audiences, you need diagramming software such as Creately's to craft visually engaging and precise ER diagrams. The

ER Diagram Software offered by Creately as an online service is pretty easy to use and is a lot more affordable than purchasing licensed software. It is also perfectly suited for development teams because of its strong support for collaboration.

### The History of Entity Relationship Diagrams

Peter Chen developed ERDs in 1976. Since then, Charles Bachman and James Martin have added some slight refinements to the basic ERD principles.

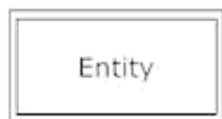
### Structure of an Entity Relationship Diagram with Common ERD Notations

An entity relationship diagram is a means of visualizing how the information a system produces is related. There are five main components of an ERD:

- **Entities**, which are represented by rectangles. An entity is an object or concept about which you want to store information.



- **Weak entity** is an entity that must be defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.



- **Actions**, which are represented by diamond shapes, show how two entities share information in the database. In some cases, entities can be self-linked. For example, employees can supervise other employees.



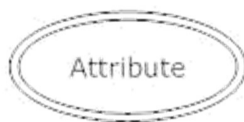
- **Relationship:** The degree of a relationship is the number of entity types that participate in the relationship.



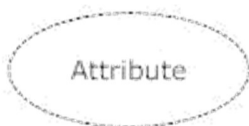
- **Attributes**, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.



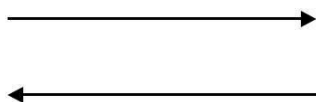
- **Multi-valued attribute** can have more than one value. For example, an employee entity can have multiple skill values.



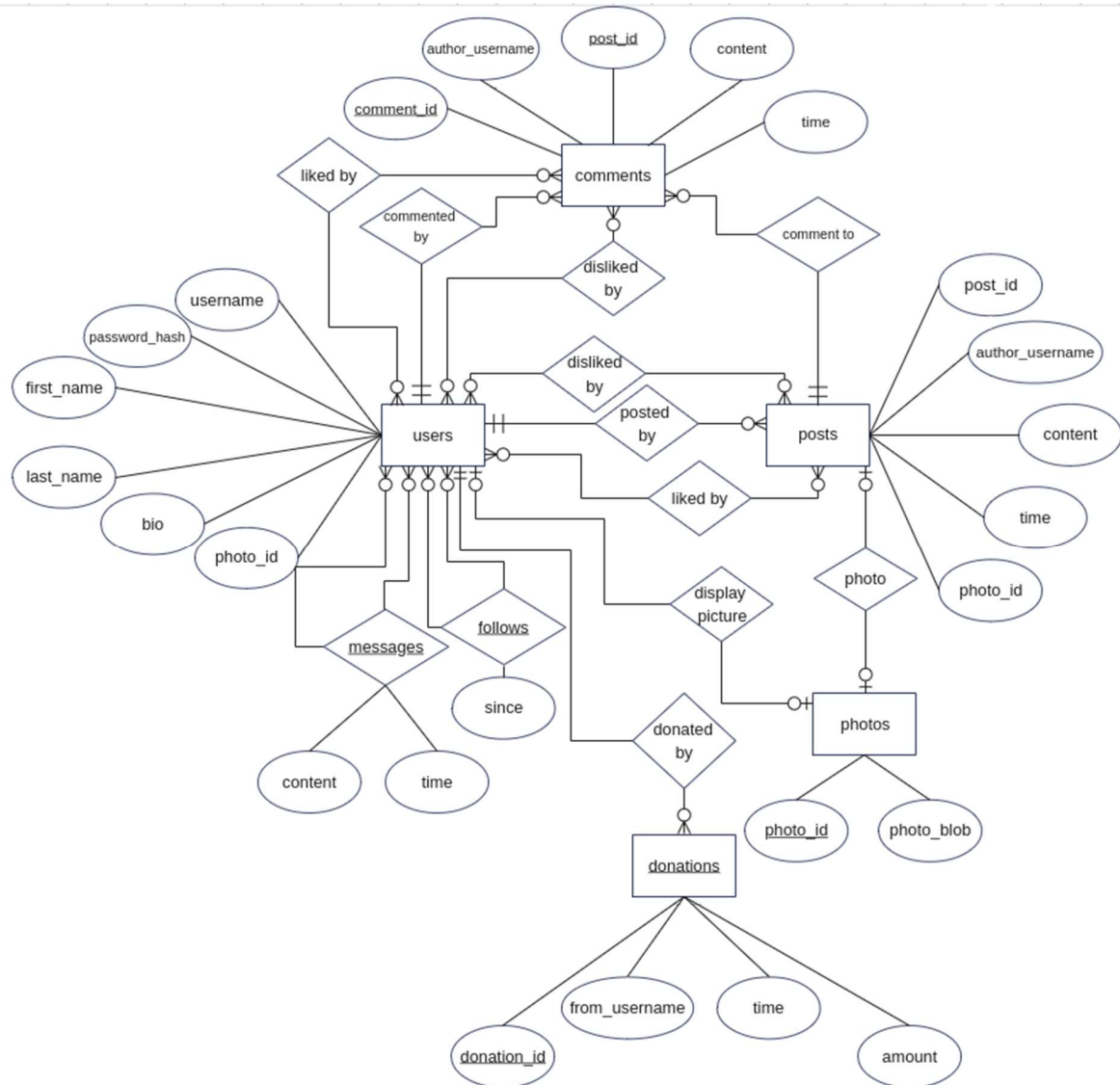
- **Derived attribute** is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.



- **Connecting lines**, solid lines that connect attributes to show the relationships of entities in the diagram.



- **Cardinality** specifies how many instances of an entity relate to one instance of another entity. Ordinarily is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinarily describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and cordiality specifies the absolute minimum number of relationships.
  - One to One
  - One to Many
  - Many to One
  - Many to Many

**E-R DIAGRAM FOR SOCIAL CONTENT SHARING PLATFORM:**

### 3.2 DATA FLOW DIAGRAM (level 0 and level 1)

The Data Flow Diagrams (DFDs) are used for structure analysis and design. DFDs show the flow of data from external entities into the system. DFDs also show how the data moves and is transformed from one process to another, as well as its logical storage. The following symbols are used within DFDs. For clarity, a key has been provided at the bottom of this page. A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

#### HISTORY

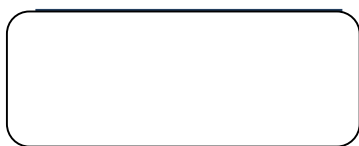
Larry Constantine, the original developer of structured design, based on Martin and Estrin's "Data Flow Graph" model of computation. Starting in the 1970s, data flow diagrams (DFD) became a popular way to visualize the major steps and data involved in software system processes. DFDs were usually used to show data flow in a computer system, although they could in theory be applied to business process modeling. DFD were useful to document the major data flows or to explore a new high-level design in terms of data flow. Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top-down approach to Systems Design. This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system. Data flow diagrams are one of the three essential perspectives of the structured-systems analysis and design method. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data flow diagram, users are able to visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's dataflow diagrams

can be drawn up and compared with the new system's data flow diagrams to draw comparisons to implement a more efficient system. Data flow diagrams can be used to provide the end user with a physical idea of where the data they input ultimately has an effect upon the structure of the whole system from order to dispatch to report. How any system is developed can be determined through a data flow diagram model. There are different notations to draw data flow diagrams defining different visual representations for processes, data stores, data flow, and external entities.

### PHYSICAL VS LOGICAL DFD

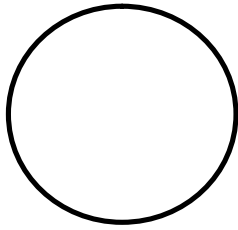
A logical DFD captures the data flows that are necessary for a system to operate. It describes the processes that are undertaken, the data required and produced by each process, and the stores needed to hold the data. On the other hand, a physical DFD shows how the system is actually implemented, either at the moment (Current Physical DFD), or how the designer intends it to be in the future (Required Physical DFD). Thus, a Physical DFD may be used to describe the set of data items that appear on each piece of paper that move around an office, and the fact that a particular set of pieces of paper are stored together in a filing cabinet. It is quite possible that a Physical DFD will include references to data that are duplicated, or redundant, and that the data stores, if implemented as a set of database tables, would constitute an un-normalized (or de-normalized) relational database. In contrast, a Logical DFD attempts to capture the data flow aspects of a system in a form that has neither redundancy nor duplication.

### DATA FLOW SYMBOLS AND THEIR MEANINGS: -



**An entity:** A source of data or a destination for data.

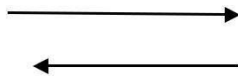
**Source/Sink:** Represented by rectangles in the diagram. Sources and Sinks are external entities which are sources or destinations of data, respectively.



**Process:** Represented by circles in the diagram. Processes are responsible for manipulating the data. They take data as input and output an altered version of the data.



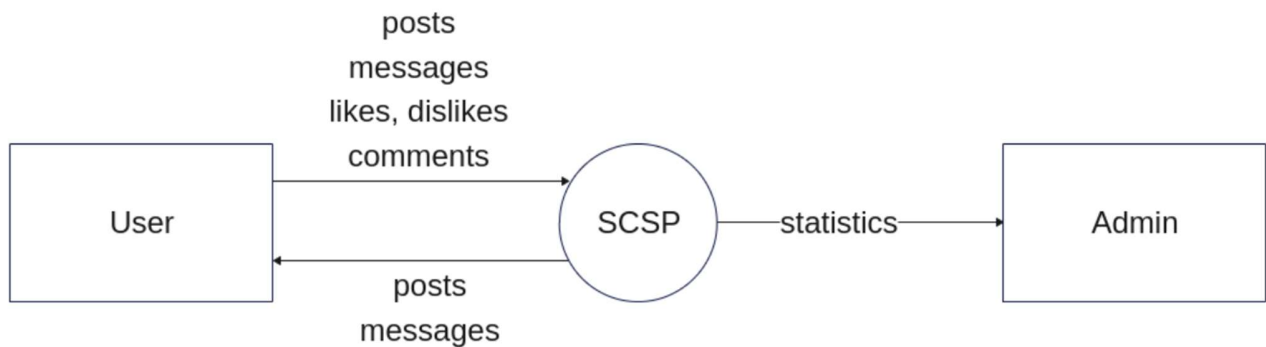
**Data Store:** Represented by a segmented rectangle with an open end on the right. Data Stores are both electronic and physical locations of data. Examples include databases, directories, files, and even filing cabinets and stacks of paper.



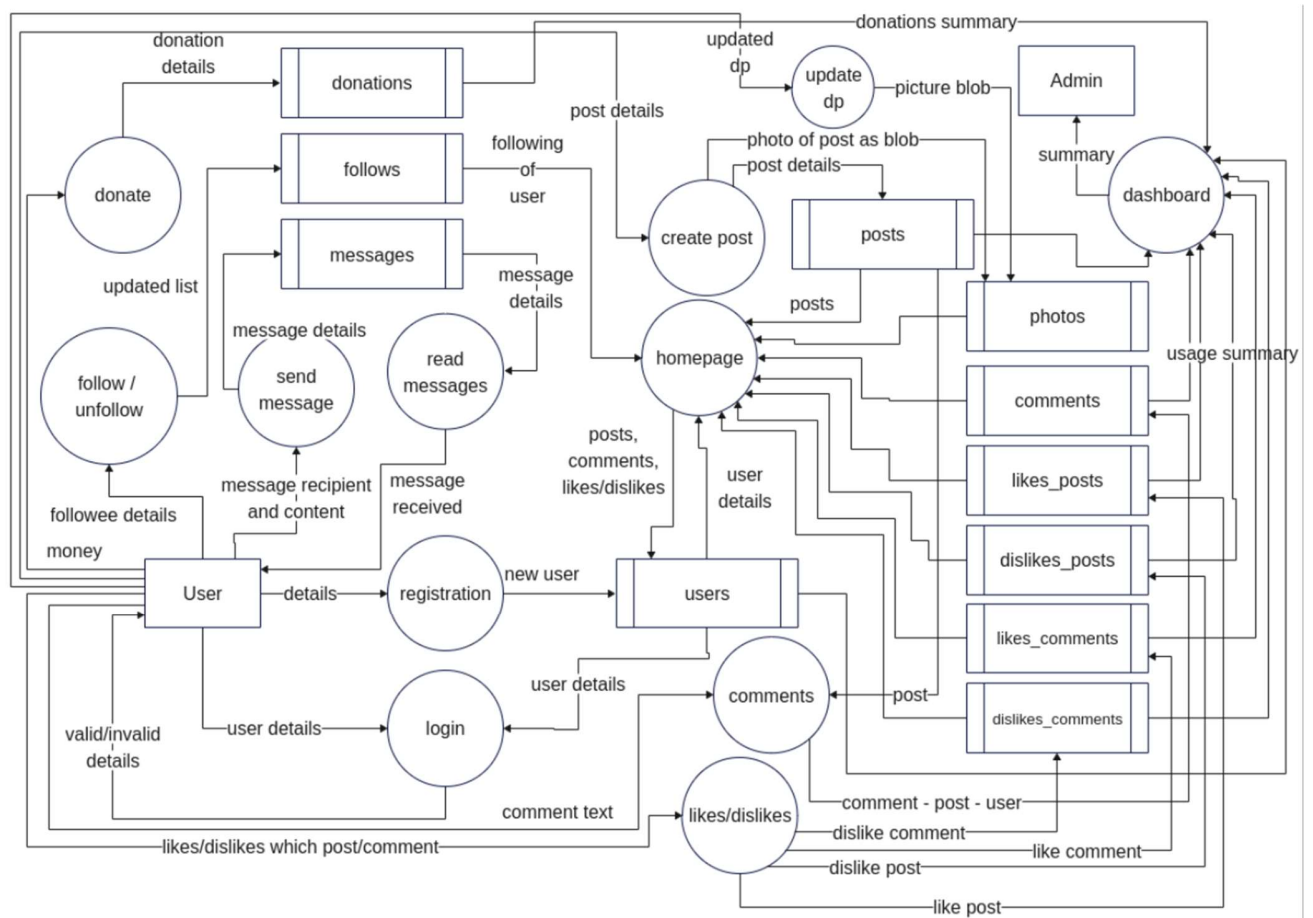
**Data Flow:** Represented by a unidirectional arrow. Data Flows show how data is moved through the System. Data Flows are labelled with a description of the data that is being passed through it.

**Level 0 DFD (Context Diagram):** A level-0 DFD is the most basic form of DFD. It aims to show how the entire system works at a glance. There is only one process in the system and all the data flows either into or out of this process. Level-0 DFD's demonstrates the interactions between the process and external entities. They do not contain Data Stores. When drawing Level-0 DFD's, we must first identify the process, all the external entities and all the data flows. We must also state any assumptions we make about the system. It is advised that we draw the process in the middle of the page. We then draw our external entities in the corners and finally connect our entities to our process with the data flows.



**Level 0 DFD/Context Diagram of SCSP:****Level 1 DFD:**

Level 1 DFD's aim is to give an overview of the full system. They look at the system in more detail. Major processes are broken down into sub-processes. Level 1 DFD's also identifies data stores that are used by the major processes. When constructing a Level 1 DFD we must start by examining the Context Level DFD. We must break up the single process into its subprocesses. We must then pick out the data stores from the text we are given and include them in our DFD. Like the Context Level DFD's, all entities, data stores and processes must be labelled. We must also state any assumptions made from the text.

**Level 1 DFD of SCSP:**

### 3.3 GANTT CHART

A Gantt chart is a type of bar chart, devised by Henry Gantt in the 1910s, that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project. Modern Gantt charts also show the dependency (i.e., precedence network) relationships between activities. Gantt charts can be used to show current schedule status using percent-complete shadings and a vertical "TODAY" line as shown here. Although now regarded as a common charting technique, Gantt charts were considered revolutionary when first introduced. This chart is also used in information technology to represent data that has been collected.

#### **HISTORICAL DEVELOPMENT:**

The first known tool of this type was developed in 1896 by Karol Adamiecki, who called it a Harmon gram. Adamiecki did not publish his chart until 1931, however, and only in Polish, which limited both its adoption and recognition of his authorship. The chart is named after Henry Gantt (1861–1919), who designed his chart around the years 1910–1915. One of the first major applications of Gantt charts was by the United States during World War I, at the instigation of General William Crozier in the 1980s, personal computers allowed widespread creation of complex and elaborate Gantt charts. The first desktop applications were intended mainly for project managers and project schedulers. With the advent of the Internet and increased collaboration over networks at the end of the 1990s, Gantt charts became a common feature of web-based applications, including collaborative groupware.

**GANTT CHART BENEFITS:**

**Clarity:** One of the biggest benefits of a Gantt chart is the tool's ability to boil down multiple tasks and timelines into a single document. Stakeholders throughout an organization can easily understand where teams are in a process while grasping the ways in which independent elements come together toward project completion.

**Communication:** Teams can use Gantt charts to replace meetings and enhance other status updates. Simply clarifying chart positions offers an easy, visual method to help team members understand task progress.

**Motivation:** Some teams or team members become more effective when faced with a form of external motivation. Gantt charts offer teams the ability to focus work at the front of a task timeline, or at the tail end of a chart segment. Both types of team members can find Gantt charts meaningful as they plug their own work habits into the overall project schedule.

**Coordination:** For project managers and resource schedulers, the benefits of a Gantt chart include the ability to sequence events and reduce the potential for overburdening team members. Some project managers even use combinations of charts to break down projects into more manageable sets of tasks.

**Creativity:** Sometimes, a lack of time or resources forces project managers and teams to find creative solutions. Seeing how individual tasks intertwine on Gantt charts often encourages new partnerships and collaborations that might not have evolved under traditional task assignment systems.

**Time Management:** Most managers regard scheduling as one of the major benefits of Gantt charts in a creative environment. Helping teams understand the overall impact of project delays can foster stronger collaboration while encouraging better task organization.

**Flexibility:** Whether you use Excel to generate Gantt charts or you load tasks into a more precise chart generator, the ability to issue new charts as your project evolves lets you react to unexpected changes in project scope or timeline. While revising your project schedule too frequently can eliminate some of the other benefits of Gantt charts, offering a realistic view of a project can help team members recover from setbacks or adjust to other changes.

**Manageability:** For project managers handling complex assignments, like software publishing or event planning, the benefits of Gantt charts include externalizing assignments. By visualizing all of the pieces of a project puzzle, managers can make more focused, effective decisions about resources and timetables.

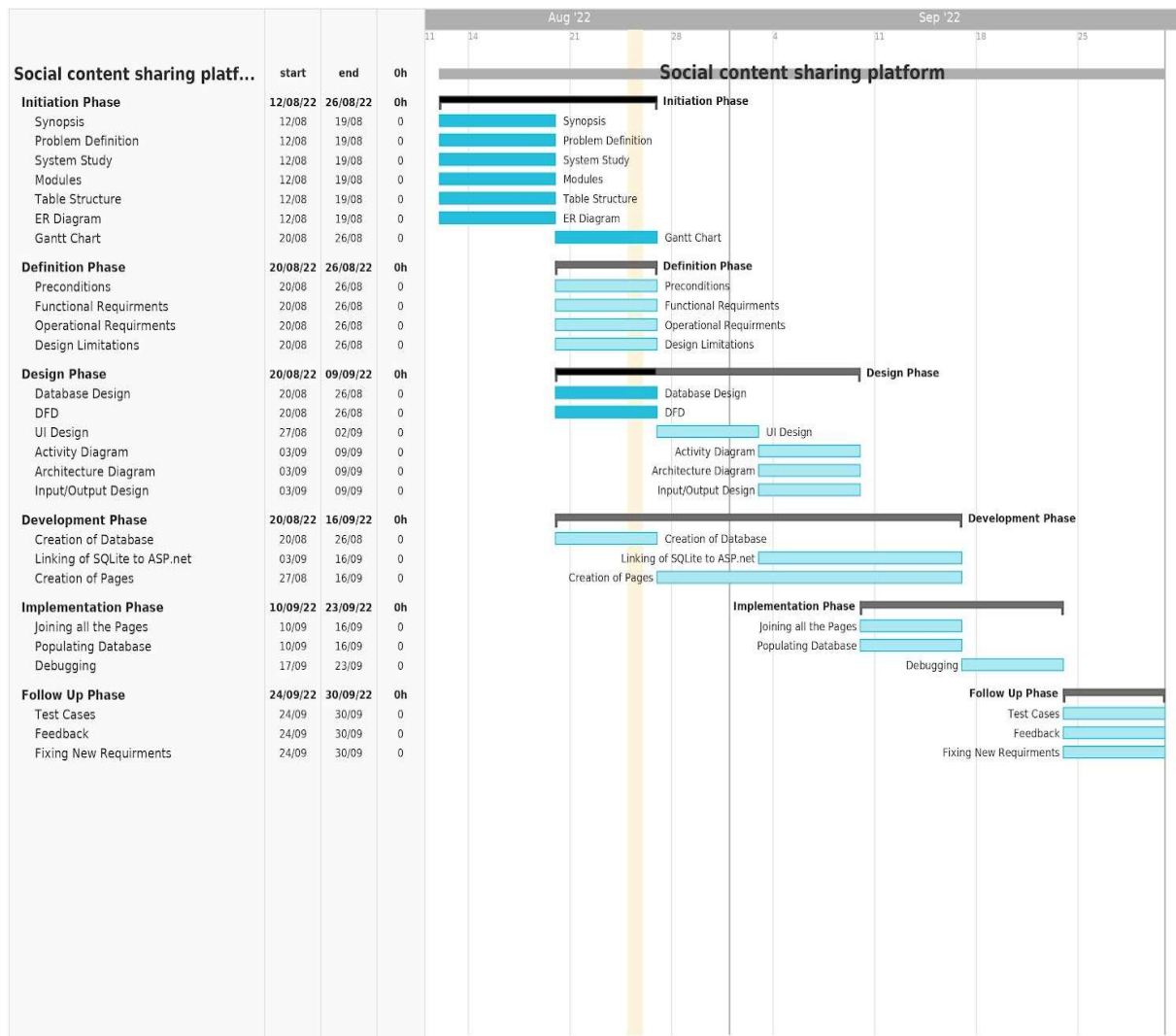
**Efficiency:** Another one of the benefits of Gantt charts is the ability for teams members to leverage each other's deadlines for maximum efficiency. For instance, while one team member waits on the outcome of three other tasks before starting a crucial piece of the assignment, he or she can perform other project tasks. Visualizing resource usage during projects allows managers to make better use of people, places, and things.

**Accountability:** When project teams face major organizational change, documenting effort and outcomes becomes crucial to career success. Using Gantt charts during critical projects allows both project managers and participants to track team progress, highlighting both big wins and major failures during professional review periods; team members who frequently exceed expectations can leverage this documentation into larger raises or bonuses.

**Gantt chart Importance:** The project's summary and terminal elements, which combine to form the project's internal structure, are shown on the Gantt chart. Many charts will also depict the precedence rankings and dependencies of various tasks within the project. The charts can illustrate the start and finish project terminal elements in project management. It can also show summary elements and terminal dependencies. The smallest task tracked as part of the project effort is known as a terminal element. Gantt chart represents the tasks in most modern project scheduling packages. However other management applications use simpler communication tools such as message boards, to-do lists and simple scheduling etc., therefore, they do not use Gantt charts as heavily. The way to create this chart begins by determining and listing the necessary activities. Next, sketch out how you expect the chart to look. List which items depend on others and what activities take place when. For each activity, list how many man-hours it will require, and who is responsible. Lastly, determine the throughput time. This technique's primary advantage is its good graphical overview that is easy to understand for nearly all project participants and stakeholders. Its primary disadvantage is its limited applicability for many projects, since projects are often more complex than can be effectively communicated with this chart.

**GANTT CHART OF SOCIAL CONTENT SHARING PLATFORM:**

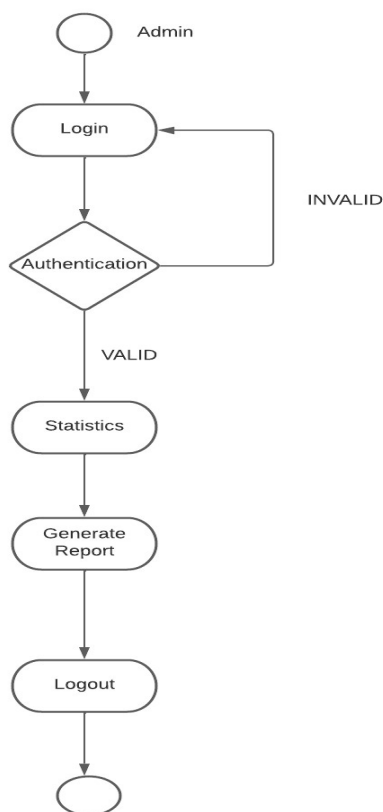
Printed: Aug 25, 2022

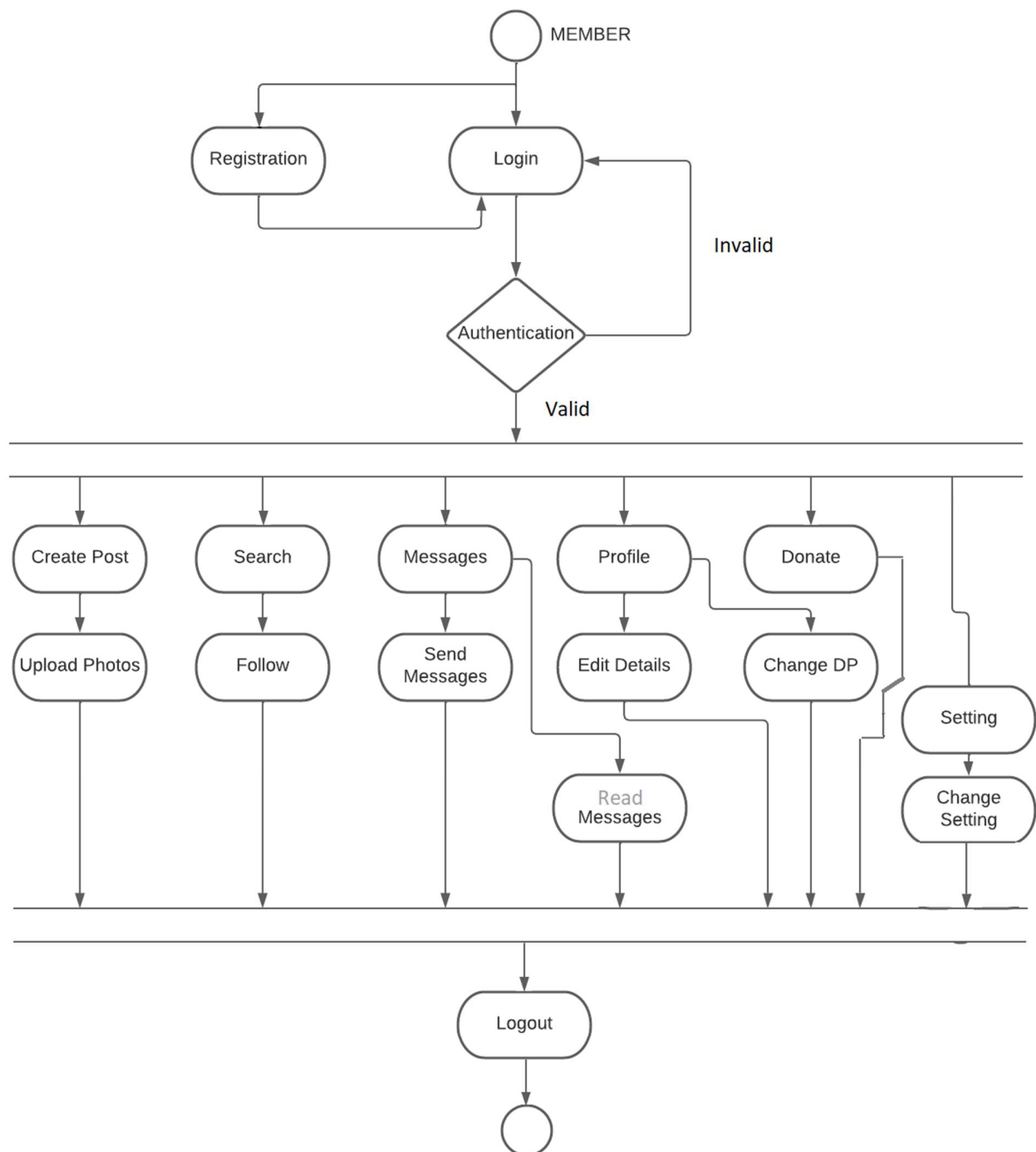


### 3.4 ACTIVITY DIAGRAM

An activity diagram visually represents the series of actions or flow of control in a system similar to a flow chart or data flow diagram. Activity diagram are often used in business processing modelling. They can also describe steps in a used case diagram. The activity diagram for Admin module and User module of College Gadget Booking is given below. Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.

#### ACTIVITY DIAGRAM FOR ADMIN

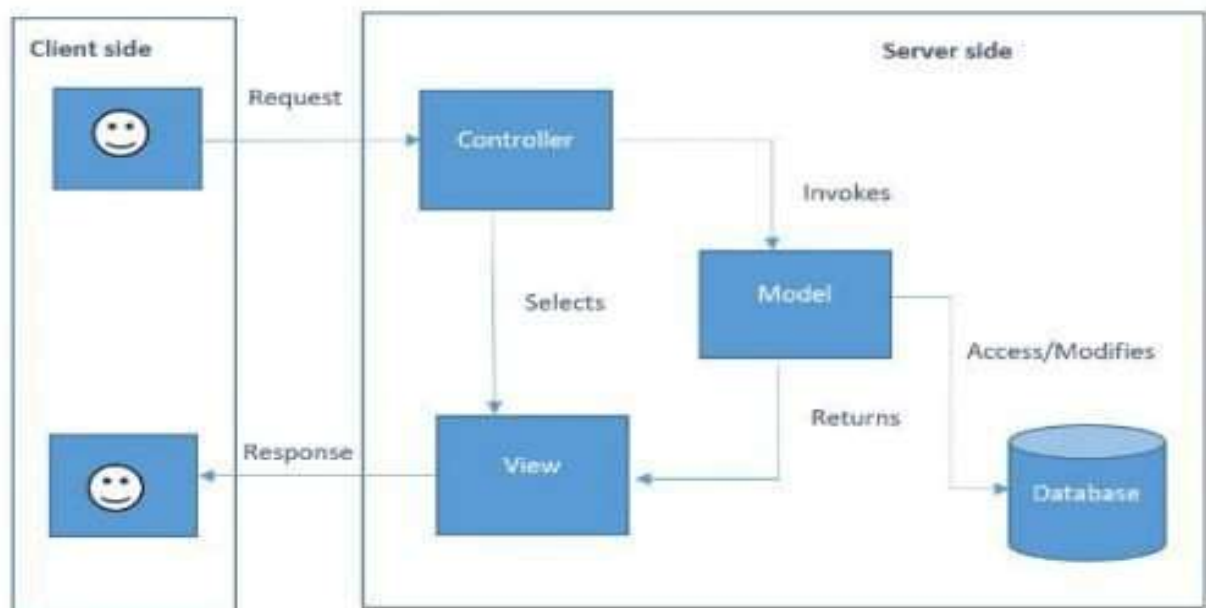


**ACTIVITY DIAGRAM FOR USER**



### 3.5 ARCHITECTURAL DESIGN

An architectural model (in software) is a rich and rigorous diagram, created using available standards, in which the primary concern is to illustrate a specific set of tradeoffs inherent in the structure and design of a system or ecosystem.



## 3.6 INPUT OUTPUT DESIGN

SCSP Home Messages Profile Explore Donations Logout

Social Content Sharing Platform  
Login

Username \*

Password \*

☐ Remember me for 1 day

Login

Not an user yet? [Register](#)

© 2022 - SCSP

SCSP Home Messages Profile Explore Donations Logout

Social Content Sharing Platform  
Register

Username \*  
 sayan

Password \*  
 .....

First Name \*  
 Sayan

Last Name  
 Ghosh

Bio  
 I like to code

Register

Already an user? [Login](#)

© 2022 - SCSP

SCSP Home Messages Profile Explore Donations Logout

Social Content Sharing Platform  
Login

Username \*  
 sayan

Password \*  
 .....

☒ Remember me for 1 day

Login

Not an user yet? [Register](#)

© 2022 - SCSP

SCSP Home Messages Profile Explore Donations Logout

Home  
Welcome Sayan  
[+ New Post](#)

Rohit Raj  
rohit

10/14/2022 11:42:40 PM

hehehehe

1 Likes 0 Dislikes 1 Comments

SCSP

Home

Messages

Profile

Explore

Donations

Logout

Create

Post

Enter the content of your post

hello this is a new post i am making for synchroniz

Upload photo if any

Choose File

No file chosen

Create

Cancel

SCSP

Home

Messages

Profile

Explore

Donations

Logout

Create

Post

Enter the content of your post

hello this is a new post i am making for synchroniz

Upload photo if any

Choose File

promo-reel.png


Create

Cancel

© 2022 - SCSP

© 2022 - SCSP

SCSP Home Messages Profile Explore Donations Logout



Sayan Ghosh

sayan

3

2

2


Posts Followers Following

About

i code stuff

Posts

+ New Post



hello this is a new post i am making for synchronize


0

0

0

Delete

10/15/2022 8:11:22 AM



another post by sayan: our cs coordinator

1


0

0

Delete

10/14/2022 9:06:57 AM

SCSP Home Messages Profile Explore Donations Logout



Sayan Ghosh

sayan

Update Display Picture

Upload New Display Picture

Choose File

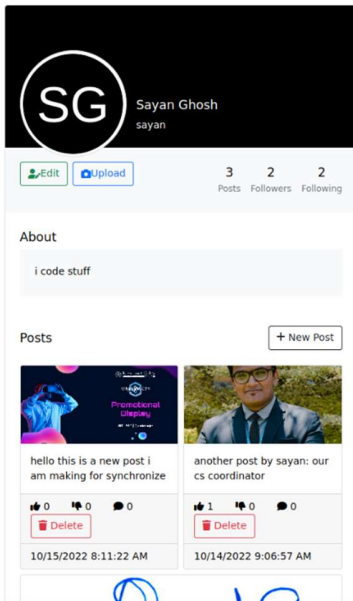
No file chosen

Update

Cancel

© 2022 - SCSP

SCSP Home Messages Profile Explore Donations Logout




SG Sayan Ghosh  
sayan

[Edit](#) [Upload](#) 3 Posts 2 Followers 2 Following

**About**  
i code stuff

**Posts** [+ New Post](#)




hello this is a new post i am making for synchronize

0 Likes 0 Dislikes 0 Comments

[Delete](#)

10/15/2022 8:11:22 AM



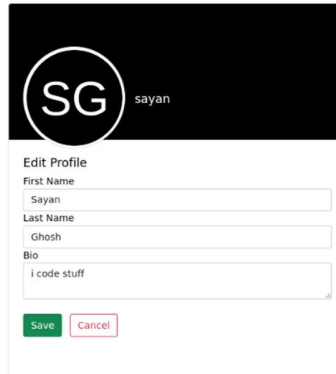
another post by sayan: our cs coordinator

1 Like 0 Dislikes 0 Comments

[Delete](#)

10/14/2022 9:06:57 AM

SCSP Home Messages Profile Explore Donations Logout



SG sayan

**Edit Profile**

First Name  
Sayan

Last Name  
Ghosh

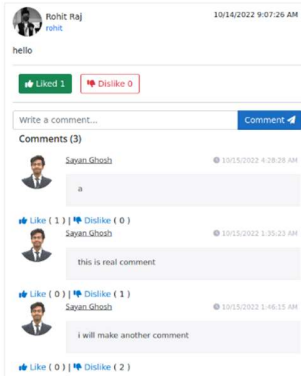
Bio  
i code stuff

[Save](#) [Cancel](#)

© 2022 - SCSP

SCSP Home Messages Profile Explore Donations Logout

## Post Details



Rohit Raj  
rohit


10/14/2022 9:07:26 AM

hello

1 Like 0 Dislikes


[Write a comment...](#) [Comment](#)

**Comments (3)**




Sayan Ghosh  
a

1 Like 0 Dislikes



Sayan Ghosh  
this is real comment

0 Likes 1 Dislike



Sayan Ghosh  
i will make another comment

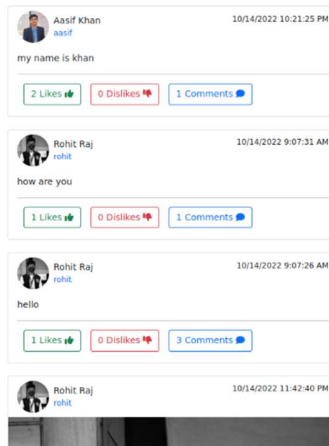
0 Likes 2 Dislikes


© 2022 - SCSP

SCSP Home Messages Profile Explore Donations Logout

## Home

Welcome Sayan

[+ New Post](#)





Asif Khan  
asif

10/14/2022 10:21:25 PM

my name is khan

2 Likes 0 Dislikes 1 Comments




Rohit Raj  
rohit

10/14/2022 9:07:31 AM

how are you

1 Like 0 Dislikes 1 Comments




Rohit Raj  
rohit

10/14/2022 9:07:26 AM

hello



1 Like 0 Dislikes 3 Comments

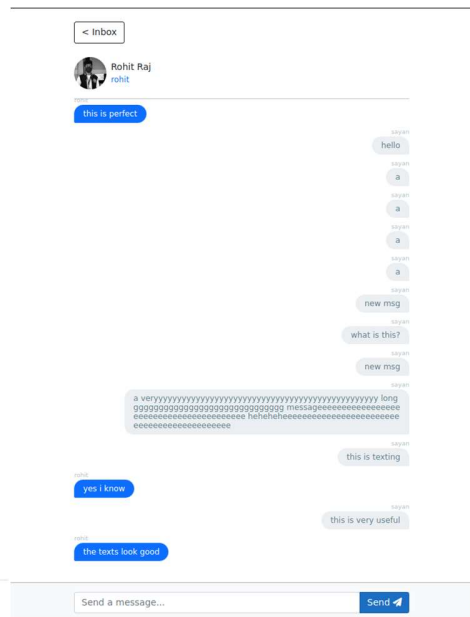


Rohit Raj  
rohit

10/14/2022 11:42:40 PM

## Inbox






	Rohit Raj	21 messages
	Aasif Khan	2 messages



© 2022 - SCSP

Search Here

 Search

Name	Bio	Options
 <b>Sayan Ghosh</b> sayan	I code stuff	<a href="#">Open</a> <a href="#">Message</a>
 <b>Rohit Raj</b> rohit	I do UI/UX	<a href="#">Open</a> <a href="#">Message</a>
 <b>Aasif Khan</b> aasif	competitive programmer	<a href="#">Open</a> <a href="#">Message</a>
 <b>Papon Raj</b> papon		<a href="#">Open</a> <a href="#">Message</a>
 <b>Admin</b> admin		<a href="#">Open</a> <a href="#">Message</a>

## Index

Date	Amount
10/14/2022	15
10/14/2022	100

[+ New Donation](#)

© 2022 - SCSP

© 2022 - SCSP

SCSP [Home](#) [Messages](#) [Profile](#) [Explore](#) [Donations](#) [Logout](#)

Create  
Donation

Amount

[Donate](#) [Cancel](#)

© 2022 - SCSP

## 4. SYSTEM CONFIGURATION

### 4.1 Hardware requirements

CPU	INTEL CORE 2 DUO 2.0 GHz
RAM	2.00 GB
Hard disk	50 GB
Graphics	Integrated Graphics card

### 4.2 Software Requirements:

Front end	HTML, CSS, JS
Back end	ASP.NET CORE 6 and SQLite
Tools	Visual Studio, Visual Studio Code, EDrawMax, Figma, dbDesigner, SQLBrowser
Operating System	Linux or MacOS or Microsoft Windows

## 5. DETAILS OF SOFTWARE

A development process consists of various phases, each phase ending with a defined output. The phases are performed in an order specified by the process model being followed. The main reason for having a phased process is that it breaks the problem of developing software into successfully performing a set of phases, each handling a different concern of software development. This ensures that the cost of development is lower than what it would have been if the whole problem were tackled together. A phased development process is central to the software engineering approach for solving the software crisis.

### 5.1 Overview of Front End

**HTML:** HTML is a standard markup language for web page creation. It allows the creation and structure of sections, paragraphs, and links using HTML elements (the building blocks of a web page) such as tags and attributes.

HTML has a lot of use cases, namely:

- **Web development.** Developers use HTML code to design how a browser displays web page elements, such as text, hyperlinks, and media files.
- **Internet navigation.** Users can easily navigate and insert links between related pages and websites as HTML is heavily used to embed hyperlinks.
- **Web documentation.** HTML makes it possible to organize and format documents, similarly to Microsoft Word.

It's also worth noting that HTML is not considered a programming language as it can't create dynamic functionality. It is now considered an official web standard. The **World Wide Web Consortium (W3C)** maintains and develops HTML specifications, along with providing regular updates. This article will go over the basics of HTML, including how it works, its pros and cons, and how it relates to CSS and JavaScript.

**CSS:** Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout,



colors, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting. Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device. The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable. The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents. In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL.

### **JAVASCRIPT:**

JavaScript often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices. JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM). The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O. JavaScript engines were originally used only in web browsers, but are now core components of some servers and a variety of applications. The most popular runtime system for this usage is Node.js. Although Java and JavaScript are similar in name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design.

## 5.2 Overview of Back End

### ASP.NET CORE:

ASP.NET Core is a free and open-source web framework and successor to ASP.NET, developed by Microsoft. It is a modular framework that runs on both the full .NET Framework, on Windows, and the cross-platform .NET. However ASP.NET Core version 3 works only on .NET Core dropping support of the .NET Framework. The framework is a complete rewrite that unites the previously separate ASP.NET MVC and ASP.NET Web API into a single programming model. Despite being a new framework, built on a new web stack, it does have a high degree of concept compatibility with ASP.NET. The ASP.NET Core framework supports side-by-side versioning so that different applications being developed on a single machine can target different versions of ASP.NET Core. This is not possible with previous versions of ASP.NET. Blazor is a recent (optional) component to support WebAssembly and since version 5.0 it is dropping support for some old web browsers. While current Microsoft Edge works, the legacy version of it, i.e. "Microsoft Edge Legacy" and Internet Explorer 11 are dropped when you use Blazor.

- **Base framework for processing web requests in C#**
- **Web-page templating syntax**, known as Razor, for building dynamic web pages using C#
- **Libraries for common web patterns**, such as Model View Controller (MVC)
- **Authentication system** that includes libraries, a database, and template pages for handling logins, including multi-factor authentication and external authentication with Google, Twitter, and more.
- **Editor extensions** to provide syntax highlighting, code completion, and other functionality specifically for developing web pages

**SQLITE3:**

SQLite is a database engine written in the C programming language. It is not a standalone app; rather, it is a library that software developers embed in their apps. As such, it belongs to the family of embedded databases. It is the most widely deployed database engine, as it is used by several of the top web browsers, operating systems, mobile phones, and other embedded systems. Many programming languages have bindings to the SQLite library. It generally follows PostgreSQL syntax, but does not enforce type checking by default. This means that one can, for example, insert a string into a column defined as an integer.

**5.3 About the Platform**

.NET Framework (pronounced as "dot net") is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library named Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for .NET Framework execute in a software environment (in contrast to a hardware environment) named Common Language Runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling. As such, computer code written using .NET Framework is called "managed code". FCL and CLR together constitute the .NET Framework. FCL provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. Programmers produce software by combining their source code with .NET Framework and other libraries. The framework is intended to be used by newest applications created for the Windows platform. Microsoft also produces an integrated development environment largely for .NET software called Visual Studio. .NET Framework began as proprietary software, although the firm worked to standardize the software stack almost immediately, even before its first release. Despite the standardization efforts, developers, mainly those in the free and open-source software communities, expressed their unease with the selected terms and the prospects.

## 6. TESTING

Testing is a vital part of software development, and it is important to start it as early as possible, and to make testing a part of the process of deciding requirements. To get the most useful perspective on your development project, it is worthwhile devoting some thought to the entire lifecycle including how feedback from users will influence the future of the application. The tools and techniques we've discussed in this book should help your team to be more responsive to changes without extra cost, despite the necessarily wide variety of different development processes. Nevertheless, new tools and process improvements should be adopted gradually, assessing the results after each step. Testing is part of a lifecycle. The software development lifecycle is one in which you hear of a need, you write some code to fulfil it, and then you check to see whether you have pleased the stakeholders—the users, owners, and other people who have an interest in what the software does. Hopefully they like it, but would also like some additions or changes, so you update or augment your code; and so, the cycle continues, or every few years,

### SOFTWARE DEVELOPMENT LIFE CYCLE

Testing is a proxy for the customer. You could conceivably do your testing by releasing it into the wild and waiting for the complaints and compliments to come back. Some companies have been accused of having such a strategy as their business model even before it became fashionable. But on the whole, the books are better balanced by trying to make sure that the software will satisfy the customer before we hand it over. This portal “**Social Content Sharing Platform**” is developed using Incremental Model and Spiral Model as Modules are made, tested, debugged, and next module is created, which is integrated with the rest of the application with integration testing and then debugged again.

**SOFTWARE TESTING TYPES:****1. FUNTIONAL TESTING**

This type of testing ignores the internal parts and focus on the output is as per requirement or not.

They are:

**Black box testing** – Internal system design is not considered in this type of testing. Tests are based on requirements and functionality.

**White box testing** – This testing is based on knowledge of the internal logic of an application's code. Also known as Glass box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions.

**Unit testing** – Testing of individual software components or modules. Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. May require developing test drive modules or test harnesses.

**Integration Testing:** – Testing of integrated modules to verify combined functionality after integration. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.

**System testing** – Entire system is tested as per the requirements. Black-box type testing that is based on overall requirements specifications, covers all combined parts of a system.

**Acceptance testing** -Normally this type of testing is done to verify if system meets the customer specified requirements. User or customers do this testing to determine whether to accept application.

**Alpha testing** – In house virtual user environment can be created for this type of testing. Testing is done at the end of development. Still minor design changes may be made as a result of such testing.

**Beta testing** – Testing typically done by end-users or others. Final testing before releasing application for commercial purpose.

## 2. NON-FUNCTIONAL TESTING

**Security testing** – Can system be penetrated by any hacking way. Testing how well the system protects against unauthorized internal or external access. Checked if system, database is safe from external attacks.

**Usability testing** – User-friendliness check. Application flow is tested, Can new user understand the application easily, Proper help documented whenever user stuck at any point, basically system navigation is checked in this testing.

**Recovery testing** – Testing how well a system recovers from crashes, hardware failures, or other catastrophic problems.

**Install/uninstall testing** – Tested for full, partial, or upgrade install/uninstall processes on different operating systems under different hardware, software environment

## 7. CONCLUSION AND FUTURE ENHANCEMENT

The system is developed with easy navigation, keeping in mind that the primary audience are non-technically inclined people. The system is designed with simple UI so that it stays away from the functionality. The system provides an intuitive UI to all the users. The project displays each component in its separate page to clearly distinguish elements. Multiple users can login at the same time and get served customised website for each of them.

The system can further be enhanced as follows:

- Deployed Server to have more stability.
- Minor UI updates.
- Send push notification when a notification appears.

## 8. BIBILIOGRAPHY

### Book References:

1. Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages by Adam Freeman, Published by APress.
2. Getting Started with SQL: A Hands-On Approach for Beginners by Thomas Nield

### Web References:

1. <https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-6.0>
2. <https://learn.microsoft.com/en-us/ef/>
3. <https://avatars.dicebear.com/docs/http-api>



## 9. APPENDICES A – Table Structure

Table Name	Attribute	Data type	Constraints	Description
User	UserID	TEXT	primary key, unique, not null	unique username of each user
	PasswordHash	TEXT	not null	hash of username+password
	FirstName	TEXT	not null	first name of user
	LName	TEXT	Nullable	last name of user
	Bio	TEXT	Nullable	user's short biography
	Photo	TEXT	Nullable	Stores base64 encoding of the photo data bytes
Post	PostID	INTEGER	primary key, unique, not null, autoincrement	unique id of each post
	AuthorId	TEXT	references User.UserID, not null	username of user who posted the post
	Content	TEXT	not null	text part of post
	Time	TEXT	not null	time of posting
	Photo	TEXT	Nullable	photo of post (if any) in base64 encoding

User	FolloweeId	TEXT	not null, unique, references User.UserID	username of person whom he follows
	FollowerId	TEXT		username of person who follows
	ID	INTEGER	Primary key, not null, unique	Primary key for unique identification

Comment	CommentID	INTEGER	primary key, unique, not null, autoincrement	unique id of each comment
	AuthorId	TEXT	references User.UserID, not null	username of user who posted this comment
	PostId	INTEGER	references Post.PostID, not null	unique id of post to which comment belongs
	Content	TEXT	not null	text content of comment
	Time	TEXT	not null	time of posting of comment

Message	MessageID	INTEGER	primary key, unique, not null, autoincrement	unique id of each chat
	FromId	TEXT	references User.UserID, not null	username of user who sends message
	ToId	TEXT	references User.UserID, not null	username of user who receive message
	Content	TEXT	not null	text content of message
	Time	TEXT	not null	date of message sent

LikePost	LikeID	INTEGER	primary key, unique, not null, autoincrement	unique id of the like
	AuthorId	TEXT	references User.UserID, not null	username of user who liked the post
	PostID	INTEGER	references Post.PostID, not null	unique id of the post liked
DislikePost	DislikeID	INTEGER	primary key, unique, not null, autoincrement	unique id of the dislike
	AuthorId	TEXT	references User.UserID, not null	username of user who disliked the post
	PostID	INTEGER	references Post.PostID, not null	unique id of the post disliked
LikeComment	LikeID	INTEGER	primary key, unique, not null, autoincrement	unique id of the like
	AuthorId	TEXT	references User.UserID, not null	username of user who liked the comment
	CommentID	INTEGER	references Comment.CommentID, not null	unique id of the comment liked
DislikeComment	DislikeID	INTEGER	primary key, unique, not null, autoincrement	unique id of the dislike
	AuthorId	TEXT	references User.UserID, not null	username who disliked the comment
	CommentID	INTEGER	references Comment.CommentID, not null	unique id of the comment disliked

Donation	DonationID	INTEGER	primary key, unique, not null, autoincrement	unique id of each donation transaction
	UserID	TEXT	references User.UserID, not null	username of user who donated
	Time	TEXT	not null	time of transaction
	Amount	REAL	not null	amount of money donated

**10. APPENDICES B – Test Cases**

Serial Number	Test Case ID	Test Description	Test Data	Expected Result	Actual Result	Status
1	Case 1	Correct username and password	sayan and sayanghosh	Login Success	Login Success	PASS
2	Case 2	Correct username and wrong password	sayan and sayanghosh1	Incorrect Username or Password	Incorrect Username or Password	PASS
3	Case 3	Wrong username and correct password	Sayang and sayanghosh	Incorrect Username or Password	Incorrect Username or Password	PASS
4	Case 4	Wrong username and wrong password	Sayang and sayanghosh1	Incorrect Username or Password	Incorrect Username or Password	PASS
5	Case 5	Short password	sayan	Password should be atleast 8 characters	Password should be atleast 8 characters	PASS
6	Case 6	Disliking a Liked Post	Dislike	Like is removed automatically and dislike is registered	Like is removed automatically and dislike is registered	PASS

7	Case 7	Upload image bigger than 1 MB	image.png	File is too big. Max size is 1MB.	File is too big. Max size is 1MB.	PASS
8	Case 8	Upload file which is not image	document.pdf	Only .jpg or .png files are allowed	Only .jpg or .png files are allowed	PASS
9	Case 9	Upload non-image file bigger than 1 MB	document.pdf	File is too big. Max size is 1MB.Only .jpg or .png files are allowed	File is too big. Max size is 1MB.Only .jpg or .png files are allowed	PASS
10	Case 10	Create post with empty content	""	Please fill out this field	Please fill out this field	PASS
11	Case 11	Send empty message	""	Please fill out this field	Please fill out this field	PASS
12	Case 12	Enter negative amount in donation	-5	Amount cannot be negative or zero	Amount cannot be negative or zero	PASS

## 11.APPENDICES C – SOURCE CODE

./wwwroot/css/home.css

```
.card{      max-width: 60rem;      margin: 20px auto; } .card:hover{      transform:
scale(1.02);      border: 2px solid var(--bs-gray-500);      transition: all 0.1s
ease-in-out; } .card-img-top{      width: 100%;      height: 15vw;      margin: 10px
auto;      object-fit: contain; } .stats{      padding-left: 10px;      padding-
right: 10px;      margin-bottom: 10px; } .stats div{      margin-right: 15px; }
/* comments */      .be-comment-block {      border: 1px solid #edeff2;      border-
radius: 2px;      padding: 5px 10px;      border: 1px solid #ffffff; } .comments-
title {      font-size: 16px;      color: #262626;      margin-bottom: 15px;
font-family: 'sans-serif';      text-align: left; } .be-img-comment {      width:
60px;      height: 60px;      float: left;      margin-bottom: 15px; } .be-ava-
comment {      width: 60px;      height: 60px;      border-radius: 50%; } .be-
comment-content {      margin-left: 80px; } .be-comment-content span {
display: inline-block;      width: 49%;      margin-bottom: 15px; } .be-comment-
name {      font-size: 13px;      font-family: 'sans-serif';      text-align: left;
} .be-comment-content a {      color: #383b43; } .be-comment-content span {
display: inline-block;      width: 49%;      margin-bottom: 15px; } .be-comment-
time {      text-align: right; } .be-comment-time {      font-size: 11px;
color: #b4b7c1; } .be-comment-text {      font-size: 13px;      line-height: 18px;
color: var(--bs-gray-800);      display: block;      background: #f6f6f7;
border: 1px solid #edeff2;      padding: 15px 20px 20px 20px; }
./wwwroot/css/profile.css
```

```
.card-img-top{      width: 100%;      height: 15vw;      object-fit: cover; }
.post-card:hover{      transform: scale(1.02);      border: 2px solid var(--bs-
gray-500);      transition: all 0.05s ease-in-out; } .stats-box{      position:
relative; } .followsyoud{      position: absolute;      bottom: 10px; }
./wwwroot/css/messageinbox.css
```

```
.sideBar {      padding: 0 !important;      margin: 0 !important;      background-
color: #fff;      overflow-y: auto;      border: 1px solid #f7f7f7;      height:
calc(100% - 120px); } .sideBar-body {      position: relative;      padding:
10px !important;      border-bottom: 1px solid #f7f7f7;      height: 72px;
margin: 0 !important;      cursor: pointer; } .sideBar-body:hover {
background-color: #f2f2f2; } .sideBar-avatar {      text-align: center;
padding: 0 !important; } .avatar-icon img {      border-radius: 50%;
height: 49px;      width: 49px; } .sideBar-main {      padding: 0
!important; } .sideBar-main .row {      padding: 0 !important;      margin:
0 !important; } .sideBar-name {      padding: 10px !important; }
.name-meta {      font-size: 100%;      padding: 1% !important;      text-align:
left;      text-overflow: ellipsis;      white-space: nowrap;      color: #000; }
.sideBar-time {      padding: 10px !important; } .time-meta {      text-
align: right;      font-size: 12px;      padding: 1% !important;      color: rgba(0,
0, 0, .4);      vertical-align: baseline; }
./wwwroot/css/login.css
```

```
body {      font-family: "Lato", Arial, sans-serif;      font-size: 16px;      line-
height: 1.8;      font-weight: normal;      background: #f8f9fd;      color: gray; }
a {      -webkit-transition: .3s all ease;      -o-transition: .3s all ease;
transition: .3s all ease;      color: #01d28e;      text-decoration: none; }
a:hover, a:focus {      text-decoration: none !important;      outline: none
!important;      -webkit-box-shadow: none;      box-shadow: none; }
a:hover, a:active, a:focus {      color: #01d28e;      outline: none
```

```

!important; text-decoration: none !important; } h1, h2, h3, h4, h5,
.h1, .h2, .h3, .h4, .h5 { line-height: 1.5; font-weight: 400; font-
family: "Lato", Arial, sans-serif; color: #000; } .bg-primary {
background: #01d28e !important; } .ftco-section { padding: 1em 0; }
.ftco-no-pt { padding-top: 0; } .ftco-no-pb { padding-bottom: 0; }
.heading-section { font-size: 28px; color: #000; } .img {
background-size: cover; background-repeat: no-repeat; background-
position: center center; } .wrap { width: 100%; overflow: hidden;
background: #fff; border-radius: 5px; -webkit-box-shadow: 0px 10px 34px -
15px rgba(0, 0, 0, 0.24); -moz-box-shadow: 0px 10px 34px -15px rgba(0, 0, 0,
0.24); box-shadow: 0px 10px 34px -15px rgba(0, 0, 0, 0.24); } .wrap .img
{ height: 200px; } .login-wrap { position: relative; } .login-
wrap h3 { font-weight: 300; } .form-group { position: relative;
z-index: 0; margin-bottom: 40px !important; } .form-group a {
color: gray; } .form-control { height: 48px; background: #fff;
color: #000; font-size: 16px; border-radius: 5px; -webkit-box-shadow:
none; box-shadow: none; border: 1px solid rgba(0, 0, 0, 0.1); }
.form-control::-webkit-input-placeholder { /* Chrome/Opera/Safari */
color: rgba(0, 0, 0, 0.2) !important; } .form-control::-moz-placeholder {
/* Firefox 19+ */ color: rgba(0, 0, 0, 0.2) !important; } .form-
control:-ms-input-placeholder { /* IE 10+ */ color: rgba(0, 0, 0,
0.2) !important; } .form-control:-moz-placeholder { /* Firefox 18- */
color: rgba(0, 0, 0, 0.2) !important; } .form-control:focus, .form-
control:active { outline: none !important; -webkit-box-shadow: none;
box-shadow: none; border: 1px solid #01d28e; } .field-icon {
position: absolute; top: 50%; right: 15px; -webkit-transform:
translateY(-50%); -ms-transform: translateY(-50%); transform:
translateY(-50%); color: rgba(0, 0, 0, 0.3); } .form-control-placeholder
{ position: absolute; top: 2px; padding: 7px 0 0 13px; -webkit-
transition: all 400ms; -o-transition: all 400ms; transition: all 400ms;
opacity: .6; } .form-control:focus ~ .form-control-placeholder, .form-
control:not(:placeholder-shown) ~ .form-control-placeholder { -webkit-
transform: translate3d(0, -120%, 0); transform: translate3d(0, -120%, 0);
padding: 7px 0 0 0; opacity: 1; text-transform: uppercase; font-size:
12px; letter-spacing: 1px; color: #01d28e; font-weight: 700; }
.form-control:not(:placeholder-shown):invalid { border-color: var(--bs-
danger); } .form-control:not(:placeholder-shown):invalid ~ .form-
control-placeholder { color: var(--bs-danger); } .password-
minlength { display: none; color: var(--bs-danger); }
.form-control:not(:placeholder-shown):invalid ~ .password-minlength {
display: block; } .form-control:not(:placeholder-shown):invalid ~ .field-
icon { -webkit-transform: translateY(-140%); -ms-transform:
translateY(-140%); transform: translateY(-140%); } .social-media {
position: relative; width: 100%; } .social-media .social-icon {
display: block; width: 40px; height: 40px; background:
transparent; border: 1px solid rgba(0, 0, 0, 0.05); font-size: 16px;
margin-right: 5px; border-radius: 50%; } .social-media .social-icon
span { color: #999999; } .social-media .social-icon:hover, .social-
media .social-icon:focus { background: #01d28e; } .social-media
.social-icon:hover span, .social-media .social-icon:focus span { color:
#fff; } .checkbox-wrap { display: block; position: relative;
padding-left: 30px; margin-bottom: 12px; cursor: pointer; font-size:
16px; font-weight: 500; -webkit-user-select: none; -moz-user-select:
none; -ms-user-select: none; user-select: none; } /* Hide the
browser's default checkbox */ .checkbox-wrap input { position: absolute;
opacity: 0; cursor: pointer; height: 0; width: 0; } /* Create a
custom checkbox */ .checkmark { position: absolute; top: 0; left:

```



```

0; } /* Create the checkmark/indicator (hidden when not checked) */
.checkmark:after { content: "\f0c8"; font-family: "FontAwesome";
position: absolute; color: rgba(0, 0, 0, 0.1); font-size: 20px;
margin-top: -4px; -webkit-transition: 0.3s; -o-transition: 0.3s;
transition: 0.3s; } @media (prefers-reduced-motion: reduce) {
.checkmark:after { -webkit-transition: none; -o-transition: none;
transition: none; } } /* Show the checkmark when checked */ .checkbox-wrap
input:checked ~ .checkmark:after { display: block; content: "\f14a";
font-family: "FontAwesome"; color: rgba(0, 0, 0, 0.2); } /* Style the
checkmark/indicator */ .checkbox-primary { color: #01d28e; } .checkbox-
primary input:checked ~ .checkmark:after { color: #01d28e; } .btn {
cursor: pointer; -webkit-box-shadow: none !important; box-shadow: none
!important; font-size: 15px; padding: 10px 20px; } .btn:hover,
.btn:active, .btn:focus { outline: none; } .btn.btn-primary {
background: #01d28e !important; border: 1px solid #01d28e !important;
color: #fff !important; } .btn.btn-primary:hover { border: 1px
solid #01d28e; background: transparent; color: #01d28e; }
.btn.btn-primary.btn-outline-primary { border: 1px solid #01d28e;
background: transparent; color: #01d28e; } .btn.btn-primary.btn-
outline-primary:hover { border: 1px solid transparent;
background: #01d28e; color: #fff; }
./wwwroot/css/explore.css

```

```

td{ background-color: white; transition: all 200ms ease; } td:hover{
background-color: var(--bs-gray-400); }
./wwwroot/css/message.css

```

```

.card{ max-width: 30rem; margin: auto; } .fa-user{ margin-right:
15px; padding: 15px; border: 1px solid gray; border-radius: 50px; }
.topbar{ background-color: white; padding-top: 20px; } header{
display:none; } /* // chat bubbles */ ul { padding: 0; } ul > li {
position: relative; list-style: none; display: block;
margin-top: 1.5rem; margin: 1rem 0; } ul > li:after { display:
table; content: ''; clear: both; } .msg { max-width: 85%;
display: inline-block; padding: 0.5rem 1rem; line-height: 1rem;
min-height: 2rem; font-size: 0.875rem; border-radius: 1rem;
margin-bottom: 0.5rem; word-break: break-all; } .msg.him {
float: left; background-color: var(--bs-primary); color: #fff;
border-bottom-left-radius: 0.125rem; } .msg.you { float: right;
background-color: #ECEFF1; color: #607D8B; border-bottom-right-
radius: 0.125rem; } .msg > span { font-weight: 500; position: absolute;
} .msg > span.partner { color: #B0BEC5; font-size: 0.5rem; top:
0; font-size: 0.675rem; margin-top: -1rem; } .msg > span.time {
color: #CFD8DC; font-size: 0.5rem; bottom: -0.35rem;
display: none; } .msg:hover span.time { display: block; } .msg.him > span
{ left: 0; } .msg.you > span { right: 0; } input:hover, input:focus,
input:active { outline: none !important; border-bottom: 0.125rem
solid var(--bs-primary); }
./wwwroot/css/site.css

```

```

html { font-size: 14px; } @media (min-width: 768px) { html { font-size:
16px; } } html { position: relative; min-height: 100%; } body { margin-
bottom: 60px; } ::-webkit-scrollbar { width: 20px; } ::-webkit-scrollbar-track
{ background-color: transparent; } ::-webkit-scrollbar-thumb { background-
color: #d6dee1; border-radius: 20px; border: 6px solid transparent;
background-clip: content-box; } ::-webkit-scrollbar-thumb:hover { background-
color: #a8bbbfb; }

```

```
./wwwroot/css/notifications.css
```

```
td{      background-color: white;      transition: all 200ms ease; } td:hover{
background-color: var(--bs-gray-400); }
./ViewModels/ProfileIndexViewModel.cs
```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels {      public
class ProfileIndexViewModel      {      public User currentuser {get; set;} =
new User();      public ICollection<Post> Posts {get; set;} = new
List<Post>();      public ICollection<Foll> Followers {get; set;} = new
List<Foll>();      public ICollection<Foll> Following {get; set;} = new
List<Foll>();      } }
./ViewModels/MessageSendViewModel.cs
```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels {
public class MessageSendViewModel      {      public ICollection<Message>
Messages {get; set;} = new List<Message>();      public string message {get;
set;} = "";      public string errormsg {get; set;} = "";      public User
from {get; set;} = new User();      public User to {get; set;} = new User();
} }
./ViewModels/PostDetailsViewModel.cs
```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels {      public
class PostDetailsViewModel      {      public string AlertMsg {get; set;} = "";
public string AlertType {get; set;} = "danger";      public Post Post {get;
set;} = new Post();      public User currentuser {get; set;} = new User();
public bool likedbyme {get; set;}      public bool dislikedbyme {get; set;}
} }
./ViewModels/ProfileEditViewModel.cs
```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels {      public
class ProfileEditViewModel      {      public User currentuser {get; set;} =
new User();      } }
./ViewModels/PostCreateViewModel.cs
```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels {      public
class PostCreateViewModel      {      public string AlertMsg {get; set;} = "";
public string AlertType {get; set;} = "danger";      public string Content
{get; set;} = "";      public User currentuser {get; set;} = new User();      }
}
./ViewModels/NotificationIndexViewModel.cs
```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels {
public class NotificationIndexViewModel      {      public User currentuser
{get; set;} = new User();      public List<Comment> comments {get; set;} = new
List<Comment>();      } }
./ViewModels/AuthRegisterViewModel.cs
```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels; public
class AuthRegisterViewModel {      public User User {get; set;} = new User();
```

```

public string password {get; set;} = "";      public string Title {get; set;} =
"";      public string AlertMsg {get; set;} = "";      public string AlertType
{get; set;} = "danger"; }
./ViewModels/MessageIndexViewModel.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels {
public class MessageIndexViewModel {      public Dictionary<User, int>
Users {get; set;} = new Dictionary<User, int>();      public string errormsg
{get; set;} = "";      public User from {get; set;} = new User();      } }
./ViewModels/AdminIndexViewModel.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels {
public class AdminIndexViewModel {      public int comments {get; set;}
public int posts {get; set;}      public int registrations {get; set;}
public int messages {get; set;}      public int likes {get; set;}
public int dislikes {get; set;}      public int donations {get; set;}
public double donationTotal {get; set;}      } }
./ViewModels/DonationCreateViewModel.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels {
public class DonationCreateViewModel {      public string AlertMsg {get;
set;} = "";      public string AlertType {get; set;} = "";      public
double Amount {get; set;}      public User currentuser {get; set;} = new
User();      } }
./ViewModels/HomeIndexViewModel.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels {      public
class HomeIndexViewModel {      public User currentuser {get; set;} = new
User();      public ICollection<Post> Posts {get; set;} = new List<Post>(); //
posts of friends      public ICollection<Foll> Followers {get; set;} = new
List<Foll>();      public ICollection<Foll> Following {get; set;} = new
List<Foll>();      } }
./ViewModels/ProfileExploreViewModel.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels {      public
class ProfileExploreViewModel {      public User user {get; set;} = new
User();      public ICollection<Post> Posts {get; set;} = new List<Post>();
public ICollection<Foll> Followers {get; set;} = new List<Foll>();      public
ICollection<Foll> Following {get; set;} = new List<Foll>();      public bool
ifollowhim {get; set;}      public bool hefollowsme {get; set;}      public
User currentuser {get; set;} = new User();      } }
./ViewModels/DonationIndexViewModel.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels {
public class DonationIndexViewModel {      public string AlertMsg {get;
set;} = "";      public string AlertType {get; set;} = "";      public
List<Donation> Donations {get; set;} = new List<Donation>();      public User
currentuser {get; set;} = new User();      } }
./ViewModels/ExploreIndexViewModel.cs

```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels {
public class ExploreIndexViewModel { public string query {get; set;}
= ""; public List<User> Users {get; set;} = new List<User>();
public string AlertMsg {get; set;} = ""; public string AlertType {get;
set;} = ""; public User currentuser {get; set;} = new User();
} }
./ViewModels/AuthLoginViewModel.cs
```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels { public
class AuthLoginViewModel { public User User {get; set;} = new User();
public string password {get; set;} = ""; public string Title {get; set;}
= ""; public string AlertMsg {get; set;} = ""; public string
AlertType {get; set;} = "danger"; } }
./ViewModels/ProfileUpdateDPViewModel.cs
```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using scsp.Models; namespace scsp.ViewModels { public
class ProfileUpdateDPViewModel { public User currentuser {get; set;}
= new User(); public string AlertMsg {get; set;} = ""; public
string AlertType {get; set;} = ""; } }
./Views/Post/Create.cshtml
```

```
@model scsp.ViewModels.PostCreateViewModel @{ ViewData["Title"] = "Create";
ViewBag.currentuser = @Model.currentuser; } <h1>Create</h1> <h4>Post</h4>
@if (!String.IsNullOrEmpty(@Model.AlertMsg)) { string alert =
@Model.AlertType; <div class="alert alert-@alert alert-dismissible fade
show" role="alert"> <h5>@Model.AlertMsg</h5> <button
type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button> </div> } <hr /> <div class="row"> <div
class="col-md-4"> <form method="post" asp-action="Create"
enctype="multipart/form-data"> <div asp-validation-
summary="ModelOnly" class="text-danger"></div> <div class="form-
group"> <label class="control-label">Enter the content of your
post</label> <textarea class="form-control" name="content"
id="content" required maxlength="250">@Model.Content</textarea>
<span class="text-danger"></span> </div> <div
class="form-group"> <label class="control-label">Upload photo if
any</label> <input class="form-control" type="file" name="file"
id="file" /> <span class="text-danger"></span>
</div> <br> <span class="form-group">
<input type="submit" value="Create" class="btn btn-success" required/>
</span> <span> <a class="btn btn-outline-danger"
asp-action="Index" asp-controller="Profile">Cancel</a> </span>
</form> </div> </div> @section Scripts { @await
Html.RenderPartialAsync("_ValidationScriptsPartial"); } }
./Views/Post/Details.cshtml
```

```
@model scsp.ViewModels.PostDetailsViewModel @{ ViewData["Title"] =
"Details"; var Post = @Model.Post; var User = @Post.Author;
ViewBag.currentuser = @Model.currentuser; bool likedby = @Model.likedby;
bool dislikedby = @Model.dislikedby; string like_btn_design = likedby
? "btn-success" : "btn-outline-success"; string dislike_btn_design =
dislikedby ? "btn-danger" : "btn-outline-danger"; string likemsg = "Like"
+ (likedby ? "d" : ""); string dislikemsg = "Dislike" + (dislikedby ?
"d" : ""); } @section Styles { <link
```

```

href="@Url.Content("~/css/home.css")" rel="stylesheet" type="text/css" /> }
<h1>Post Details</h1> <div> <div class="card"> <div class="d-
flex justify-content-between p-2 px-3"> <div class="d-flex flex-row
align-items-center"> <a class="text-reset text-decoration-none"
asp-action="Explore" asp-controller="Profile" asp-route-id="@Post.AuthorId">
 </a>
<div class="d-flex flex-column ms-2"> <span class="font-
weight-bold"> <a class="text-reset text-decoration-none"
asp-action="Explore" asp-controller="Profile" asp-route-id="@Post.AuthorId">
@Post.Author.FName @Post.Author.LName </a>
</span> <small class="text-primary">
<a class="text-reset text-decoration-none" asp-action="Explore" asp-
controller="Profile" asp-route-id="@Post.AuthorId">
@Post.AuthorId </a> </small>
</div> </div> <div class="d-flex flex-row mt-1
ellipsis"> <small class="mr-2 text-muted"> @ {
var td = DateTime.Now - Post.Time; var days = td.Days;
var hours = td.Hours; var mins = td.Minutes;
var secs = td.Seconds; var timetext = "";
if(days > 0){ timetext = days + " day" + (days == 1 ? ""
: "s") + " ago"; } else if(hours > 0){
timetext = hours + " hour" + (hours == 1 ? "" : "s") + " ago";
} else if(mins > 0){ timetext =
mins + " minute" + (mins == 1 ? "" : "s") + " ago"; }
else if(secs > 0){ timetext = secs + " second" + (secs
== 1 ? "" : "s") + " ago"; } else {
timetext = "Now"; } }
@timetext </small> </div> </div> @if
(!String.IsNullOrEmpty(Post.Photo)) {  }
<div class="ps-3 pt-2 pb-2"> <p class="" style="text-align:
left;">@Post.Content</p> <hr> <div class="d-flex
justify-content-between align-items-center stats"> <div
class="d-flex flex-row icons align-items-center"> <div
class="likes"> <a asp-action="Like" asp-route-
id="@Post.PostID" class="btn @like_btn_design"><i class="fa fa-thumbs-up"></i>
@likemsg @Post.Likes.Count </a> </div>
<div class="dislikes"> <a asp-action="Dislike" asp-route-
id="@Post.PostID" class="btn @dislike_btn_design"><i class="fa fa-thumbs-
down"></i> @dislikemsg
@Post.Dislikes.Count</a> </div> </div>
</div> <hr> <form asp-action="Create" asp-
controller="Comment" asp-route-id="@Post.PostID"> <div
class="commentbox input-group"> <input class="form-control
rounded" placeholder="Write a comment..." name="comment" id="comment"
autocomplete="off" required maxlength="250"/> <button
type="submit" value="Comment" class="btn btn-primary me-2">Comment <i class="fas
fa-paper-plane"></i></button> </div> </form>
<div class="be-comment-block"> <h1 class="comments-
title">Comments (@Post.Comments.Count)</h1> @foreach (var
Comment in Post.Comments) { <div class="be-
comment"> <div class="be-img-comment">
<a asp-action="Explore" asp-controller="Profile" asp-route-
id="@Comment.Author.UserID">  </a> </div>

```

```

<div class="be-comment-content">
    <span class="be-comment-name">
        <a asp-action="Explore" asp-controller="Profile" asp-route-id="@Comment.Author.UserID">@User.FName
        @User.LName</a>
    </span>
    <span class="be-comment-time">
        <i class="fa fa-clock"></i>
        @{
            td = DateTime.Now - Comment.Time;
            days = td.Days;
            hours = td.Hours;
            mins = td.Minutes;
            secs = td.Seconds;
            timetext = "";
            if(days > 0){
                timetext = days + " day" + (days == 1 ? "" : "s") + " ago";
            } else if(hours > 0){
                timetext = hours + " hour" + (hours == 1 ? "" : "s") + " ago";
            } else if(mins > 0){
                timetext = mins + " minute" + (mins == 1 ? "" : "s") + " ago";
            } else if(secs > 0){
                timetext = secs + " second" + (secs == 1 ? "" : "s") + " ago";
            } else {
                timetext = "Now";
            }
        }
    </span>
    <p class="be-comment-text">
        @Comment.Content
    </p>
</div>
<div class="likedislike">
    <span class="likes">
        <a class="text-decoration-none" asp-action="Like" asp-controller="Comment" asp-route-id="@Comment.CommentID"><i class="fa fa-thumbs-up"></i> Like</a>
        (@Comment.Likes.Count)
    </span> |
    <span class="dislikes">
        <a class="text-decoration-none" asp-action="Dislike" asp-controller="Comment" asp-route-id="@Comment.CommentID"><i class="fa fa-thumbs-down"></i> Dislike</a>
        (@Comment.Dislikes.Count)
    </span>
</div>
</div>
</div>
</div>
./Views/Post/Delete.cshtml

@model scsp.Models.Post    @{    ViewData["Title"] = "Delete";
ViewBag.currentuser = @Model.Author; }    <h1>Delete</h1>    <h3>Are you sure
you want to delete this?</h3>    <div>    <h4>Post</h4>    <hr />    <dl
class="row">    <dt class = "col-sm-2">
@Html.DisplayNameFor(model => model.Content)    </dt>    <dd class =
"col-sm-10">    @Html.DisplayFor(model => model.Content)    </dd>
<dt class = "col-sm-2">    @Html.DisplayNameFor(model => model.Time)
</dt>    <dd class = "col-sm-10">    @Html.DisplayFor(model =>
model.Time)    </dd>    </dl>    <form asp-action="Delete">
<input type="hidden" asp-for="PostID" />    <input type="submit"
value="Delete" class="btn btn-danger" />    </form>    </div>
./Views/Post/Index.cshtml

@model IEnumerable<scsp.Models.Post>    @{    ViewData["Title"] = "Index"; }
<h1>Index</h1>    <p>    <a asp-action="Create">Create New</a>    </p>    <table
class="table">    <thead>    <tr>    <th>
@Html.DisplayNameFor(model => model.PostID)    </th>    <th>
@Html.DisplayNameFor(model => model.Content)    </th>    <th>
@Html.DisplayNameFor(model => model.Time)    </th>    <th>
@Html.DisplayNameFor(model => model.Author)    </th>    <th>
@Html.DisplayNameFor(model => model.Photo)    </th>
<th>Options</th>    </tr>    </thead>    <tbody>    @foreach (var item in
Model) {    <tr>    <td>

```

```

@Html.DisplayFor(modelItem => item.PostID)                </td>                <td>
@Html.DisplayFor(modelItem => item.Content)                </td>                <td>
@Html.DisplayFor(modelItem => item.Time)                   </td>                <td>
@item.Author.UserID                                     </td>                <td>
(String.IsNullOrEmpty(item.Photo))                       {
<span>@item.Photo</span>                                } else{
<span>@item.Photo.Substring(0,10) ...</span>              }
</td>            <td>            <a asp-action="Details" asp-route-
id="@item.PostID">Details</a> |                          <a asp-action="Delete" asp-
route-id="@item.PostID">Delete</a>                        </td>            </tr>    }
</tbody> </table>
./Views/Post/Edit.cshtml

@model scsp.Models.Post    @{      ViewData["Title"] = "Edit";  }
<h1>Edit</h1>    <h4>Post</h4> <hr /> <div class="row">    <div class="col-
md-4">        <form asp-action="Edit">            <div asp-validation-
summary="ModelOnly" class="text-danger"></div>            <input type="hidden"
asp-for="PostID" />            <div class="form-group">                <label
asp-for="Content" class="control-label"></label>                <input asp-
for="Content" class="form-control" />                <span asp-validation-
for="Content" class="text-danger"></span>            </div>            <div
class="form-group">                <label asp-for="Time" class="control-
label"></label>                <input asp-for="Time" class="form-control" />
<span asp-validation-for="Time" class="text-danger"></span>            </div>
<div class="form-group">                <input type="submit" value="Save"
class="btn btn-primary" />            </div>        </form>    </div>
</div>    <div>        <a asp-action="Index">Back to List</a>    </div>    @section
Scripts {        @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}    }
./Views/User/Create.cshtml

@model scsp.Models.User    @{      ViewData["Title"] = "Create";  }
<h1>Create</h1>    <h4>User</h4> <hr /> <div class="row">    <div class="col-
md-4">        <form asp-action="Create">            <div asp-validation-
summary="ModelOnly" class="text-danger"></div>            <div class="form-
group">                <label asp-for="UserID" class="control-label"></label>
<input asp-for="UserID" class="form-control" />                <span asp-
validation-for="UserID" class="text-danger"></span>            </div>
<div class="form-group">                <label asp-for="PasswordHash"
class="control-label"></label>                <input asp-for="PasswordHash"
class="form-control" />                <span asp-validation-for="PasswordHash"
class="text-danger"></span>            </div>            <div class="form-
group">                <label asp-for="FName" class="control-label"></label>
<input asp-for="FName" class="form-control" />                <span asp-
validation-for="FName" class="text-danger"></span>            </div>
<div class="form-group">                <label asp-for="LName" class="control-
label"></label>                <input asp-for="LName" class="form-control" />
<span asp-validation-for="LName" class="text-danger"></span>            </div>
<div class="form-group">                <label asp-for="Bio" class="control-
label"></label>                <input asp-for="Bio" class="form-control" />
<span asp-validation-for="Bio" class="text-danger"></span>            </div>
<div class="form-group">                <input type="submit" value="Create"
class="btn btn-primary" />            </div>        </form>    </div>
</div>    <div>        <a asp-action="Index">Back to List</a>    </div>    @section
Scripts {        @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}    }
./Views/User/Details.cshtml

```

```

@model scsp.Models.User @ {      ViewData["Title"] = "Details"; }
<h1>Details</h1> <div> <h4>User</h4> <hr /> <dl class="row">
<dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
model.PasswordHash) </dt> <dd class = "col-sm-10">
@Html.DisplayFor(model => model.PasswordHash) </dd> <dt class =
"col-sm-2"> @Html.DisplayNameFor(model => model.FName)
</dt> <dd class = "col-sm-10"> @Html.DisplayFor(model =>
model.FName) </dd> <dt class = "col-sm-2">
@Html.DisplayNameFor(model => model.LName) </dt> <dd class =
"col-sm-10"> @Html.DisplayFor(model => model.LName) </dd>
<dt class = "col-sm-2"> @Html.DisplayNameFor(model => model.Bio)
</dt> <dd class = "col-sm-10"> @Html.DisplayFor(model =>
model.Bio) </dd> </dl> </div> <div> <a asp-action="Edit"
asp-route-id="@Model?.UserID">Edit</a> | <a asp-action="Index">Back to
List</a> </div>
./Views/User/Delete.cshtml

```

```

@model scsp.Models.User @ {      ViewData["Title"] = "Delete"; }
<h1>Delete</h1> <h3>Are you sure you want to delete your account?</h3> <div>
<h4>Deletion cannot be undone</h4> <hr /> <dl class="row"> <dt
class = "col-sm-2"> Username: </dt> <dd class =
"col-sm-10"> @Html.DisplayFor(model => model.UserID) </dd>
<dt class = "col-sm-2"> First Name: </dt> <dd
class = "col-sm-10"> @Html.DisplayFor(model => model.FName)
</dd> <dt class = "col-sm-2"> Last Name: </dt>
<dd class = "col-sm-10"> @Html.DisplayFor(model => model.LName)
</dd> <dt class = "col-sm-2"> Bio: </dt>
<dd class = "col-sm-10"> @Html.DisplayFor(model => model.Bio)
</dd> </dl> <form asp-action="Delete"> <input
type="hidden" asp-for="UserID" /> <input type="submit" value="Delete"
class="btn btn-danger" /> <a asp-action="Index" asp-controller="Profile"
class="btn btn-outline-danger">Cancel</a> </form> </div>
./Views/User/Index.cshtml

```

```

@model IEnumerable<scsp.Models.User> @ {      ViewData["Title"] = "Index"; }
<h1>Index</h1> <p> <a asp-action="Create">Create New</a> </p> <table
class="table"> <thead> <tr> <th>username</th>
<th>First Name</th> <th>Last Name</th> <th>Bio</th>
<th>Options</th> </tr> </thead> <tbody> @foreach (var item in
Model) { <tr> <td>
@Html.DisplayFor(modelItem => item.UserID) </td> <td>
@Html.DisplayFor(modelItem => item.FName) </td> <td>
@Html.DisplayFor(modelItem => item.LName) </td> <td>
@Html.DisplayFor(modelItem => item.Bio) </td> <td>
<a asp-action="Edit" asp-route-id="@item.UserID">Edit</a> |
<a asp-action="Details" asp-route-id="@item.UserID">Details</a> |
<a asp-action="Delete" asp-route-id="@item.UserID">Delete</a>
</td>
</tr> } </tbody> </table>
./Views/User/Edit.cshtml

```

```

@model scsp.Models.User @ {      ViewData["Title"] = "Edit"; }
<h1>Edit</h1> <h4>User</h4> <hr /> <div class="row"> <div class="col-
md-4"> <form asp-action="Edit"> <div asp-validation-
summary="ModelOnly" class="text-danger"></div> <input type="hidden"
asp-for="UserID" /> <div class="form-group"> <label
asp-for="PasswordHash" class="control-label"></label> <input
asp-for="PasswordHash" class="form-control" /> <span asp-

```



```

validation-for="PasswordHash" class="text-danger"></span>                </div>
<div class="form-group">                <label asp-for="FName" class="control-
label"></label>                <input asp-for="FName" class="form-control" />
<span asp-validation-for="FName" class="text-danger"></span>                </div>
<div class="form-group">                <label asp-for="LName" class="control-
label"></label>                <input asp-for="LName" class="form-control" />
<span asp-validation-for="LName" class="text-danger"></span>                </div>
<div class="form-group">                <label asp-for="Bio" class="control-
label"></label>                <input asp-for="Bio" class="form-control" />
<span asp-validation-for="Bio" class="text-danger"></span>                </div>
<div class="form-group">                <input type="submit" value="Save"
class="btn btn-primary" />                </div>                </form>                </div>
</div>    <div>        <a asp-action="Index">Back to List</a>    </div>    @section
Scripts {        @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}    }
./Views/Home/Index.cshtml

@model scsp.ViewModels.HomeIndexViewModel @{    ViewData["Title"] = "Home";
var Posts = @Model.Posts;    ViewBag.currentuser = @Model.currentuser; }
@section Styles {    <link href="@Url.Content("~/css/home.css")"
rel="stylesheet" type="text/css" /> }    <div class="text-center">        <h1
class="display-4">Feed</h1>        <h5 class="text-secondary">Welcome
@Model.currentuser.FName</h1>        <a class="btn btn-outline-primary" asp-
action="Create" asp-controller="Post"><i class="fa-solid fa-plus"></i> New
Post</a>        <br><br><br>        @foreach (var Post in @Posts)            {
<a class="text-reset text-decoration-none" asp-action="Details" asp-
controller="Post" asp-route-id="@Post.PostID">                <div class="card">
<div class="d-flex justify-content-between p-2 px-3">                    <div
class="d-flex flex-row align-items-center">                                                <div
class="d-flex flex-column ms-2">                            <span class="font-
weight-bold">@Post.Author.FName @Post.Author.LName</span> <small class="text-
primary" style="text-align: left">@Post.AuthorId</small>
</div>                            </div>                            <div class="d-flex flex-row mt-1
ellipsis"> <small class="mr-2 text-muted">                                @
var td = DateTime.Now - Post.Time;                                var days = td.Days;
var hours = td.Hours;                                var mins = td.Minutes;
var secs = td.Seconds;                                var timetext = "";
if(days > 0){                                timetext = days + " day" + (days == 1 ?
"" : "s") + " ago";                                } else if(hours > 0){
timetext = hours + " hour" + (hours == 1 ? "" : "s") + " ago";
}                                else if(mins > 0){
timetext = mins + " minute" + (mins == 1 ? "" : "s") + " ago";
}                                else if(secs > 0){
timetext = secs + " second" + (secs == 1 ? "" : "s") + " ago";
} else {                                timetext = "Now";                                }
}                                @timetext                                </small> </div>
</div>                                @if (!String.IsNullOrEmpty(Post.Photo))                {

}                                <div class="ps-3 pt-2 pb-2 pe-3">                                <p class=""
style="text-align: left;">@Post.Content</p>                                <hr>
<div class="d-flex justify-content-left align-items-center stats">
<div class="likes">                                <span class="text-success text-
decoration-none">@Post.Likes.Count Likes <i class="fa fa-thumbs-up"></i></span>
</div>                                <div class="dislikes">                                <span
class="text-danger text-decoration-none">@Post.Dislikes.Count Dislikes <i
class="fa fa-thumbs-down ms-2"></i> </span>                                </div>

```

```
<div class="comments">
    <span class="text-primary text-decoration-none"><@Post.Comments.Count Comments <i class="fa fa-comment ms-2"></i></span>
    </div>
</div>
</div>
</a>
} </div>
./Views/_ViewImports.cshtml

@using scsp @using scsp.Models @addTagHelper *,
Microsoft.AspNetCore.Mvc.TagHelpers
./Views/Profile/Explore.cshtml

@model scsp.ViewModels.ProfileExploreViewModel @{      ViewData["Title"] =
"Profile";      var User = @Model.user;      ViewBag.currentUser =
@Model.currentUser;      string username = @User.UserID;      string fname =
@User.FName;      string lname = @User.LName ?? "";      string bio = @User.Bio ??
"";      string name = fname + " " + lname;      var ifollowhim =
@Model.ifollowhim;      var hefollowsme = @Model.hefollowsme;      var
follow_design = ifollowhim ? "btn-success" : "btn-outline-success"; } @section
Styles {      <link href="@Url.Content("~/css/profile.css")" rel="stylesheet"
type="text/css" /> }      <section class="h-100 gradient-custom-2">      <div
class="container py-5 h-100">      <div class="row d-flex justify-content-center
align-items-center h-100">      <div class="col col-lg-9 col-xl-7">      <div
class="card">      <div class="rounded-top text-white d-flex flex-row"
style="background-color: #000; height:200px;">      <div class="ms-4 mt-5
d-flex flex-column" style="width: 150px;">      <img
src='@Model.user.Photo' alt="profile photo" class="img-fluid img-thumbnail mt-4 mb-2"
style="width: 150px; height: 150px; object-fit: cover; z-index: 1; border-radius: 50%">
      <div class="buttons d-flex flex-row" style="z-index: 1;">      <a type="button" class="btn
@follow design me-2 d-flex flex-row align-items-center" asp-action="Follow" asp-controller="Profile" asp-route-id="@username">
      <i class="fas fa-user-plus"></i>      @if (ifollowhim)
      {
      <div>Following</div>
      }else{
      <a type="button" class="btn btn-outline-primary d-flex flex-row align-items-center" asp-action="Send" asp-controller="Message" asp-route-id="@username">
      <i class="fas fa-comments"></i> Message
      </a>
      </div>
      </div>
      <div class="ms-3" style="margin-top: 130px;">      <h5>@name</h5>
      <p>@username</p>
      </div>
      </div>
      <div class="p-4 text-black stats-box" style="background-color: #f8f9fa;">      @if
(hefollowsme)
      {
      <div class="followsyou text-muted">
      follows you
      </div>
      }
      <div class="d-flex justify-content-end text-center py-1">
      <div>
      <p class="mb-1 h5">@Model.Posts.Count</p>
      <p class="small text-muted">@Model.Posts</p>
      </div>
      <div class="mb-1 h5">@Model.Followers.Count</p>
      <p class="small text-muted">Followers</p>
      </div>
      <div>
      <p class="mb-1 h5">@Model.Following.Count</p>
      <p class="small text-muted">Following</p>
      </div>
      </div>
      @* TODO: Make this a partial view *@
      <div class="card-body p-4 text-black">
      <div class="mb-5">
      <p class="lead fw-normal mb-1">About</p>
      <div class="p-4" style="background-color: #f8f9fa;">
      <p class="font-italic mb-1">@bio</p>
      </div>
      </div>
      <div class="d-flex justify-content-between align-items-center mb-4">
      <p class="lead fw-normal mb-0">Posts</p>
      @* <p class="mb-0"><a asp-action="Create" asp-controller="Post" class="btn btn-outline-dark"><i class="fa-solid fa-plus"></i> New Post</a></p>
      *@
      </div>
      @if (@Model.Posts.Count == 0)
      {
      <p
```

59 | Page

```

class="form-group">                                <label class="control-label">Upload New
Display Picture</label>                             <input type="file" class="form-
control" name="file" id="file" />                   </div>                                <br>
<span class="form-group">                           <input type="submit" value="Update"
class="btn btn-success me-2" />                     </span>                                <span>
<a asp-action="Index" class="btn btn-outline-danger">Cancel</a>
</span>                                </div>                                </div> </form>
./Views/Profile/Index.cshtml

```

```
@model Models.ProfileIndexViewModel @{ ViewData["Title"] = "Profile"; var User = @Model.currentUser; ViewBag.currentUser = User; }@section Styles { <link href="@Url.Content("~/css/profile.css")" rel="stylesheet" type="text/css" /> } <section class="h-100 gradient-custom-2"><div class="container py-5 h-100"> <div class="row d-flex justify-content-center align-items-center h-100"> <div class="col col-lg-9 col-xl-7"><div class="card"> <div class="rounded-top text-white d-flex flex-row" style="background-color: #000; height:200px;"> <div class="ms-4 mt-5 d-flex flex-column" style="width: 150px;"> <img src='@Model.currentUser.Photo' alt="profile photo" class="img-fluid img-thumbnail mt-4 mb-2" style="width: 150px; height: 150px; object-fit: cover; z-index: 1; border-radius: 50%"> <div class="buttons d-flex flex-row" style="z-index: 1;"> <a type="button" class="btn btn-outline-success me-2 d-flex flex-row align-items-center justify-content-left" asp-action="Edit"> <i class="fas fa-user-edit"></i> Edit </a> <a type="button" class="btn btn-outline-primary d-flex flex-row align-items-center justify-content-left" asp-action="UpdateDP"> <i class="fas fa-camera"></i> Upload </a> </div> </div> <div class="ms-3" style="margin-top: 130px;"> <h5>@Model.currentUser.FName <p>@Model.currentUser.UserID</p> </div> <div class="p-4 text-black" style="background-color: #f8f9fa;"> <div class="d-flex justify-content-end text-center py-1"> <div> <p class="mb-1 h5">@Model.Posts.Count</p> <p class="small text-muted mb-0">Posts</p> </div> <div class="px-3"> <p class="mb-1 h5">@Model.Followers.Count</p> <p class="small text-muted mb-0">Followers</p> </div> <p class="mb-1 h5">@Model.Following.Count</p> <p class="small text-muted mb-0">Following</p> </div> <div class="card-body p-4 text-black"> <div class="mb-5"> <p class="lead fw-normal mb-1">About</p> <div class="p-4" style="background-color: #f8f9fa;"> <p class="font-italic mb-1">@Model.currentUser.Bio</p> </div> <div class="d-flex justify-content-between align-items-center mb-4"> <p class="lead fw-normal mb-0">Posts</p> <p class="mb-0"><a asp-action="Create" asp-controller="Post" class="btn btn-outline-dark"><i class="fa-solid fa-plus"></i> New Post</a></p> </div> @for (int i = 0; i < ( @Model.Posts.Count + 1 ) / 2; i++){ <div class="row g-2 mb-2"> @for (int j = 0; j < 2 && i*2 + j < @Model.Posts.Count; j++){ var Post = Model.Posts.ToList()[i*2+j]; <div class="col"> <a class="text-reset text-decoration-none" asp-action="Details" asp-controller="Post" asp-route-id="@Post.PostID"> <div class="card"> <div>  </div> <div class="card-body"> @* <h5 class="card-title">Card
```

```
./Views/Profile/Edit.cshtml
```

```

@model scsp.ViewModels.ProfileEditViewModel @{
    ViewData["Title"] = "Edit
Profile";
    ViewBag.currentUser = @Model.currentUser; } <form asp-
action="Edit">
    <div asp-validation-summary="ModelOnly" class="text-
danger"></div>
    <input type="hidden" asp-for="currentUser.UserID"
name="username" id="username" />
    <div class="container py-5 h-100">
    <div class="row d-flex justify-content-center align-items-center h-100">
    <div class="col col-lg-9 col-xl-7">
        <div class="card">
        <div class="rounded-top text-white d-flex flex-row" style="background-color:
#000; height:200px;">
            <div class="ms-4 mt-5 d-flex flex-column"
style="width: 150px;">
                <img src='@Model.currentUser.Photo'
alt="profile photo" class="img-fluid img-thumbnail mt-4 mb-2"
style="width: 150px; height: 150px; object-fit: cover; z-index: 1; border-radius:
50%">
            </div>
            <div class="ms-3" style="margin-top:
130px;">
                <h5>@Model.currentUser.UserID</h5>
            </div>
            <div class="card-body p-4 text-black">
                <div class="mb-5">
                    <p class="lead fw-normal mb-1">Edit
Profile</p>
                    <div class="form-group">
                        <label
asp-for="currentUser.FName" class="control-label">First Name</label>
                        <input asp-for="currentUser.FName" placeholder="First Name" class="form-control"
name="fname" id="fname" required maxlength="32"/>
                        <span asp-
validation-for="currentUser.FName" class="text-danger"></span>
                    </div>
                    <div class="form-group">
                        <label asp-
for="currentUser.LName" class="control-label">Last Name</label>
                        <input asp-for="currentUser.LName" placeholder="Last Name" class="form-control"
name="lname" id="lname" maxlength="32" />
                        <span asp-

```

```

validation-for="currentuser.LName" class="text-danger"></span>
</div>
<div class="form-group">
<label asp-
for="currentuser.Bio" class="control-label"></label>
<textarea asp-for="currentuser.Bio" placeholder="Bio" class="form-control"
name="bio" id="bio" maxlength="250"></textarea>
<span asp-
validation-for="currentuser.Bio" class="text-danger"></span>
</div>
<br>
<span class="form-group">
<input type="submit" value="Save" class="btn btn-success me-2" />
</span>
<span>
<a asp-action="Index"
class="btn btn-outline-danger">Cancel</a>
</span>
</div>
</div>
</div>
</form>
@section Scripts {
@{await
Html.RenderPartialAsync("_ValidationScriptsPartial");} }
./Views/Shared/_ValidationScriptsPartial.cshtml

<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>
<script src="~/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.min.js"></script>
./Views/Shared/_Layout.cshtml

<!DOCTYPE html> <html lang="en"> <head>
<meta charset="utf-8" />
<meta
name="viewport" content="width=device-width, initial-scale=1.0" />
<title>@ViewData["Title"] - SCSP</title>
<link rel="stylesheet"
href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
<link rel="stylesheet"
href="~/css/site.css" asp-append-version="true" />
<link
href="~/lib/fontawesome/css/fontawesome.css" rel="stylesheet">
<link
href="~/lib/fontawesome/css/brands.css" rel="stylesheet">
<link
href="~/lib/fontawesome/css/solid.css" rel="stylesheet">
@RenderSection("Styles", false)
<style>
a.navbar-brand { white-space:
normal; text-align: center; word-break: break-all; }
a { color: #0077cc; }
.btn-primary { color: #fff; background-color: #1b6ec2; border-color:
#1861ac; }
.nav-pills .nav-link.active, .nav-pills .show > .nav-link { color:
#fff; background-color: #1b6ec2; border-color: #1861ac; }
.border-top { border-top: 1px solid #e5e5e5; }
.border-bottom { border-bottom: 1px solid
#e5e5e5; }
.box-shadow { box-shadow: 0 .25rem .75rem rgba(0, 0, .05); }
button.accept-policy { font-size: 1rem; line-height: inherit; }
.footer { position: absolute; bottom: 0; width: 100%; white-space: nowrap; line-
height: 60px; text-align: center; }
.normallinks{ display: inherit; }
.adminlinks{ display: none; }
</style>
</head>
<body>
<header
class="sticky-top">
<nav class="navbar navbar-expand-sm navbar-
toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
<div class="container-fluid">
<a class="navbar-brand" asp-area=""
asp-controller="Home" asp-action="Index">SCSP</a>
<button
class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target=".navbar-collapse" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
<span
class="navbar-toggler-icon"></span>
</button>
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
<ul class="navbar-nav flex-grow-1">
<li class="nav-item normallinks">
<a class="nav-link text-dark" asp-
area="" asp-controller="Home" asp-action="Index">Home</a>
</li>
<li class="nav-item normallinks">
<a class="nav-link text-dark" asp-area="" asp-controller="Message" asp-
action="Index">Messages</a>
</li>
@* <li class="nav-item normallinks">
<a class="nav-
link text-dark" asp-area="" asp-controller="Profile" asp-
action="Index">Profile</a>
</li>
* @
<li class="nav-item normallinks">
<a class="nav-link

```

```

text-dark" asp-area="" asp-controller="Explore" asp-action="Index">Explore</a>
</li>
<li class="nav-item normallinks">
<a class="nav-link text-dark" asp-area="" asp-controller="Notification" asp-
action="Index">Notifications</a>
</li>
<li class="nav-item normallinks">
<a class="nav-link
text-dark" asp-area="" asp-controller="Donation" asp-action="Index">Donations</a>
</li>
<li class="nav-item normallinks">
<a class="nav-link text-dark" asp-area="" asp-controller="Authentication" asp-
action="Logout">Logout</a>
</li>
<li class="nav-item adminlinks">
<a class="nav-link
text-dark" asp-area="" asp-controller="Admin" asp-action="Index">User Admin</a>
</li>
<li class="nav-item adminlinks">
<a class="nav-link text-dark" asp-area="" asp-controller="Authentication" asp-
action="Logout">Logout</a>
</li>
</ul>
@{
    var User =
    (User)ViewBag.currentuser;
    if (User != null)
    {
        @* profile photo *@
        <div class="me-2">
            <a class="text-decoration-none
            text-dark" asp-controller="Profile" asp-action="Index">
            
            @User.FName
            </div>
        }
    }
</div>
</div>
</nav>
</header>
<div
class="container">
    <main role="main" class="pb-3">
    @RenderBody()
    </main>
    </div>
    <footer class="border-top footer
    text-muted footer-fixed">
    <div class="container">
        &copy; 2022
    - SCSIP
    </div>
    </footer>
    <script
    src="~/lib/jquery/dist/jquery.min.js"></script>
    <script
    src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script
    src="~/js/site.js" asp-append-version="true"></script>
    @await
    RenderSectionAsync("Scripts", required: false)
</body>
</html>
./Views/Shared/Error.cshtml

@model ErrorViewModel @{
    ViewData["Title"] = "Error";
}
<h1 class="text-
danger">Error.</h1>
<h2 class="text-danger">An error occurred while processing
your request.</h2>
@if (Model.ShowRequestId) {
    <p>
        <strong>Request
        ID:</strong>
        <code>@Model.RequestId</code>
    </p>
}
<h3>Development Mode</h3>
<p>
    Swapping to <strong>Development</strong> environment will display more
detailed information about the error that occurred.
</p>
<p>
    <strong>The
    Development environment shouldn't be enabled for deployed applications.</strong>
It can result in displaying sensitive information from exceptions to end users.
For local debugging, enable the <strong>Development</strong> environment by
setting the <strong>ASPNETCORE_ENVIRONMENT</strong> environment variable to
<strong>Development</strong> and restarting the app.
</p>
./Views/Shared/_Layout.cshtml.css

/* Please see documentation at https://docs.microsoft.com/aspnet/core/client-
side/bundling-and-minification for details on configuring this project to bundle
and minify static web assets. */
a.navbar-brand {
    white-space: normal;
    text-align: center;
    word-break: break-all;
}
a {
    color: #0077cc;
}
.btn-primary {
    color: #fff;
    background-color: #1b6ec2;
    border-color: #1861ac;
}
.nav-pills .nav-link.active, .nav-pills .show > .nav-link {
    color: #fff;
    background-color: #1b6ec2;
    border-color: #1861ac;
}
.border-top {
    border-top: 1px solid #e5e5e5;
}
.border-bottom {
    border-bottom: 1px solid #e5e5e5;
}
.box-shadow {
    box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);
}
button.accept-

```

```
policy { font-size: 1rem; line-height: inherit; } .footer { position:
absolute; bottom: 0; width: 100%; white-space: nowrap; line-height: 60px;
}
./Views/Donation/Create.cshtml
```

```
@model scsp.ViewModels.DonationCreateViewModel @{ ViewData["Title"] =
"Create"; ViewBag.currentuser = @Model.currentuser; } <h1>Create</h1>
<h4>Donation</h4> @if (!String.IsNullOrEmpty(@Model.AlertMsg)) { string
alert = @Model.AlertType; <div class="alert alert-@alert alert-dismissible
fade show" role="alert"> <h5>@Model.AlertMsg</h5> <button
type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button> </div> } <hr /> <div class="row"> <div
class="col-md-4"> <form asp-action="Create"> <div
class="form-group"> <label class="control-label">Amount</label>
<input class="form-control" type="number" required name="amount" id="amount"
value="@Model.Amount"/> <span class="text-danger"></span>
</div> <br> <span class="form-group">
<input type="submit" value="Donate" class="btn btn-primary" id="submit"
onclick="opensite()" /> </span> <span>
<a asp-action="Index" class="btn btn-outline-danger">Cancel</a>
</span> </form> </div> </div> @section Scripts{ <script>
function opensite(){ window.open('https://rzp.io/l/tDfWB9zd',
' _blank'); } </script> }
./Views/Donation/Index.cshtml
```

```
@model scsp.ViewModels.DonationIndexViewModel @{ ViewData["Title"] =
"Index"; ViewBag.currentuser = @Model.currentuser; } <h1>Index</h1>
<table class="table"> <thead> <tr> <th>
Date </th> <th> Amount
</th> </tr> </thead> <tbody> @foreach (var item in
Model.Donations) { <tr> <td>
@Html.DisplayFor(modelItem => item.Time) </td> <td>
@Html.DisplayFor(modelItem => item.Amount) </td> </tr> }
</tbody> </table> <p> <a asp-action="Create" class="btn btn-outline-
primary"><i class="fa-solid fa-plus"></i> New Donation</a> </p>
./Views/Explore/Index.cshtml
```

```
@model scsp.ViewModels.ExploreIndexViewModel @{ ViewData["Title"] =
"Explore"; ViewBag.currentuser = @Model.currentuser; } @section Styles {
<link href="@Url.Content("~/css/explore.css")" rel="stylesheet" type="text/css"
/> } <form asp-action="Index" class="form"> <div class="input-group mb-3">
<input type="text" class="form-control form-control-lg" placeholder="Search Here"
name="query" id="query" value="@Model.query"/> <button type="submit"
class="input-group-text btn-primary"><i class="fa fa-search me-2"></i>
Search</button> </div> </form> <div class="results"> <table class="table
align-middle mb-0 bg-white"> <thead class="bg-light"> <tr>
<th>Users</th> </tr> </thead> <tbody> @foreach (var User
in @Model.Users) { <tr> <td> <a
asp-action="Explore" asp-controller="Profile" asp-route-id="@User.UserID"
class="text-reset text-decoration-none"> <div class="d-flex
align-items-center">  <div class="ms-3">
<p class="fw-bold mb-1">@User.FName @User.LName</p> <p
class="text-muted mb-0">@User.UserID</p> </div> </div>
</td> </tr> }
</tbody> </table> </div> @section Scripts{ <script> </script> }
```



```
./Views/Comment/Create.cshtml
```

```
@model scsp.Models.Comment    @{      ViewData["Title"] = "Create";  }
<h1>Create</h1>    <h4>Comment</h4>  <hr />  <div class="row">    <div
class="col-md-4">    <form asp-action="Create">    <div asp-
validation-summary="ModelOnly" class="text-danger"></div>    <div
class="form-group">    <label asp-for="Content" class="control-
label"></label>    <input asp-for="Content" class="form-control" />
<span asp-validation-for="Content" class="text-danger"></span>
</div>    <div class="form-group">    <label asp-
for="Time" class="control-label"></label>    <input asp-for="Time"
class="form-control" />    <span asp-validation-for="Time"
class="text-danger"></span>    </div>    <div class="form-
group">    <input type="submit" value="Create" class="btn btn-
primary" />    </div>    </form>    </div>    </div>    <div>
<a asp-action="Index">Back to List</a> </div>    @section Scripts {    @{await
Html.RenderPartialAsync("_ValidationScriptsPartial");}  }
./Views/Comment/Details.cshtml
```

```
@model scsp.Models.Comment    @{      ViewData["Title"] = "Details";  }
<h1>Details</h1>    <div>    <h4>Comment</h4>    <hr />    <dl class="row">
<dt class = "col-sm-2">    @Html.DisplayNameFor(model => model.Content)
</dt>    <dd class = "col-sm-10">    @Html.DisplayFor(model =>
model.Content)    </dd>    <dt class = "col-sm-2">
@Html.DisplayNameFor(model => model.Time)    </dt>    <dd class =
"col-sm-10">    @Html.DisplayFor(model => model.Time)    </dd>
</dl> </div> <div>    <a asp-action="Edit" asp-route-
id="@Model?.CommentID">Edit</a> |    <a asp-action="Index">Back to List</a>
</div>
./Views/Comment/Delete.cshtml
```

```
@model scsp.Models.Comment    @{      ViewData["Title"] = "Delete";  }
<h1>Delete</h1>    <h3>Are you sure you want to delete this?</h3> <div>
<h4>Comment</h4>    <hr />    <dl class="row">    <dt class = "col-sm-
2">    @Html.DisplayNameFor(model => model.Content)    </dt>
<dd class = "col-sm-10">    @Html.DisplayFor(model => model.Content)
</dd>    <dt class = "col-sm-2">    @Html.DisplayNameFor(model =>
model.Time)    </dt>    <dd class = "col-sm-10">
@Html.DisplayFor(model => model.Time)    </dd>    </dl>    <form
asp-action="Delete">    <input type="hidden" asp-for="CommentID" />
<input type="submit" value="Delete" class="btn btn-danger" /> |    <a asp-
action="Index">Back to List</a>    </form> </div>
./Views/Comment/Index.cshtml
```

```
@model IEnumerable<scsp.Models.Comment>    @{      ViewData["Title"] = "Index";
}    <h1>Index</h1>    <p>    <a asp-action="Create">Create New</a>    </p>
<table class="table">    <thead>    <tr>    <th>
@Html.DisplayNameFor(model => model.Content)    </th>    <th>
@Html.DisplayNameFor(model => model.Time)    </th>
<th></th>    </tr>    </thead>    <tbody>    @foreach (var item in Model)
{    <tr>    <td>    @Html.DisplayFor(modelItem =>
item.Content)    </td>    <td>
@Html.DisplayFor(modelItem => item.Time)    </td>    <td>
<a asp-action="Edit" asp-route-id="@item.CommentID">Edit</a> |
<a asp-action="Details" asp-route-id="@item.CommentID">Details</a> |
<a asp-action="Delete" asp-route-id="@item.CommentID">Delete</a>
</td>    </tr>    }    </tbody> </table>
```

```
./Views/Comment/Edit.cshtml
```

```
@model scsp.Models.Comment @ { ViewData["Title"] = "Edit"; }
<h1>Edit</h1> <h4>Comment</h4> <hr /> <div class="row"> <div
class="col-md-4"> <form asp-action="Edit"> <div asp-
validation-summary="ModelOnly" class="text-danger"></div> <input
type="hidden" asp-for="CommentID" /> <div class="form-group">
<label asp-for="Content" class="control-label"></label> <input
asp-for="Content" class="form-control" /> <span asp-validation-
for="Content" class="text-danger"></span> </div> <div
class="form-group"> <label asp-for="Time" class="control-
label"></label> <input asp-for="Time" class="form-control" />
<span asp-validation-for="Time" class="text-danger"></span> </div>
<div class="form-group"> <input type="submit" value="Save"
class="btn btn-primary" /> </div> </form> </div>
</div> <div> <a asp-action="Index">Back to List</a> </div> @section
Scripts { @await Html.RenderPartialAsync("_ValidationScriptsPartial"); } }
./Views/_ViewStart.cshtml
```

```
@ { Layout = "_Layout"; }
./Views/Notification/Index.cshtml
```

```
@model scsp.ViewModels.NotificationIndexViewModel @ { ViewData["Title"] =
"Notifications"; var currentuser = @Model.currentuser;
ViewBag.currentuser = currentuser; } @section Styles { <link
href="@Url.Content("~/css/notifications.css")" rel="stylesheet" type="text/css"
/> } <div class="results"> <table class="table align-middle mb-0 bg-white">
<thead class="bg-light"> <tr> <th>Notifications</th>
</tr> </thead> <tbody> @foreach (var Comment in @Model.comments)
{ var User = @Comment.Author; <tr> <td>
<a asp-controller="Post" asp-action="Details" asp-route-id="@Comment.PostId"
class="text-reset text-decoration-none"> <div class="d-flex
align-items-center justify-content-between"> <div
class="notification-content d-flex flex-row align-items-center">
<div class="photo"> 
</div> <div class="ms-3"> <p
class="mb-1"> @if (@User == @currentuser)
{ <strong>You</strong>
} else { <strong>
@User.FName @User.LName </strong>
commented
"@Comment.Content" on your post titled
<em> "@Comment.Post.Content"
</em> </div>
</div> @* <div class="postphoto me-5">
@if(!String.IsNullOrEmpty(Comment.Post.Photo)){  }
</div> *@ <div class="time text-muted ms-3" style="text-
align: right; flex-shrink: 0"> @ {
var td = DateTime.Now - Comment.Time; var days =
td.Days; var hours = td.Hours;
var mins = td.Minutes; var secs = td.Seconds;
var timetext = ""; if(days > 0){
timetext = days + " day" + (days == 1 ? "" : "s") + " ago";
} else if(hours > 0){ timetext = hours + " hour"
```

```

+ (hours == 1 ? "" : "s") + " ago";
else if(mins > 0){
    (mins == 1 ? "" : "s") + " ago";
else if(secs > 0){
    (secs == 1 ? "" : "s") + " ago";
    timetext = "Now";
}
@timetext
</a>
</td>
</tr>
}
</tbody>
</table>
</div>
@section Scripts{
<script>
</script>
}
./Views/Admin/Index.cshtml

@{
    ViewData["Title"] = "Admin Page";
}
@section Styles {
    <link
href="@Url.Content("~/css/admin.css")" rel="stylesheet" type="text/css" />
<style>
        .normallinks{
            display: none !important;
        }
        .adminlinks{
            display: inherit !important;
        }
    </style>
}
<div
class="text-center">
    <h1 class="display-4">Admin Report</h1>
</div>
<hr />
<dl class="row">
    <dt class = "col-sm-6">Number of
Comments</dt>
    <dd class = "col-sm-1">@Model.comments</dd>
    <dt
class = "col-sm-6">Number of Posts</dt>
    <dd class = "col-sm-
1">@Model.posts</dd>
    <dt class = "col-sm-6">Number of Registration</dt>
    <dd class = "col-sm-1">@Model.registrations</dd>
    <dt class = "col-sm-6">Number of Messages</dt>
    <dd class = "col-sm-1">@Model.messages</dd>
    <dt class = "col-sm-6">Number of Likes</dt>
    <dd class = "col-sm-
1">@Model.likes</dd>
    <dt class = "col-sm-6">Number of Dislikes</dt>
    <dd class = "col-sm-1">@Model.dislikes</dd>
    <dt class = "col-sm-6">Number
of Donations</dt>
    <dd class = "col-sm-1">@Model.donations</dd>
    <dt class = "col-sm-6">Total Amount of Donations</dt>
    <dd class = "col-
sm-1">@Model.donationTotal</dd>
</dl>
<a href="javascript:window.print()"
class="btn btn-outline-primary">Print</a>
</div>
</div>
./Views/Message/Index.cshtml

@model scsp.ViewModels.MessageIndexViewModel
@{
    ViewData["Title"] =
"Inbox";
    ViewBag.currentuser = @Model.from;
}
@if
(!String.IsNullOrEmpty(@Model.errormsg)) {
    string alert = "danger";
    <div class="alert alert-@alert alert-dismissible fade show" role="alert">
    <h5>@Model.errormsg</h5>
    <button type="button" class="btn-close" data-
bs-dismiss="alert" aria-label="Close"></button>
    </div>
}
<h1>Inbox</h1>
<div class="row sideBar">
    @foreach (var kv in Model.Users) {
        var User =
kv.Key;
        var Messages = kv.Value;
        string username =
@User.UserID;
        string fname = @User.FName;
        string lname =
@User.LName ?? "";
        <a class="text-reset text-decoration-none" asp-
action="Send" asp-route-id="@User.UserID" style="padding: 0px">
        <div
class="row sideBar-body">
            <div class="col-sm-3 col-xs-3 sideBar-
avatar">
                <div class="avatar-icon">
                    
                </div>
            <div class="col-sm-9 col-xs-9 sideBar-main">
                <div class="row">
                    <div class="col-sm-8 col-xs-8 sideBar-
name">
                        <span class="name-meta">@fname @lname
</span>
                    </div>
                    <div class="col-sm-4
col-xs-4 pull-right sideBar-time">
                        <span class="time-
meta pull-right">@Messages.messages
</span>
                    </div>
                </div>
            </div>
        </div>
    }
</div>
@section Styles{
    <link
href="@Url.Content("~/css/messageinbox.css")" rel="stylesheet" type="text/css" />
}
./Views/Message/Send.cshtml

```

```

@model scsp.ViewModels.MessageSendViewModel @{} var messages =
@Model.Messages; var message = @Model.message; var errormsg =
@Model.errormsg; var from = @Model.from; var to = @Model.to; var
User = to; ViewData["Title"] = "Chat - " + to.UserID; } <div class="topbar
sticky-top"> <a asp-action="Index" class="btn btn-outline-dark">&lt;
Inbox</a> <br><br> <div class="d-flex flex-row align-items-center">
<a class="text-reset text-decoration-none" asp-action="Explore" asp-
controller="Profile" asp-route-id="@to.UserID"> 
</a> <div class="d-flex flex-column ms-2"> <span
class="font-weight-bold"> <a class="text-reset text-
decoration-none" asp-action="Explore" asp-controller="Profile" asp-route-
id="@to.UserID"> @to.FName @to.LName
</a> </span> <small class="text-primary">
<a class="text-reset text-decoration-none" asp-action="Explore" asp-
controller="Profile" asp-route-id="@to.UserID">
@to.UserID </a> </small>
</div> </div> <hr /> </div> <div class="chats"> <ul>
@foreach (var msg in @messages) { var direction = @msg.From ==
@Model.from ? "you" : "him"; <li> <div class="msg
@direction"><span class="partner">@msg.From.UserID</span>@msg.Content<span
class="time">@msg.Time</span></div> </li> } </ul> </div> <form
asp-action="Send" class=""> <div class="bottom fixed-bottom"> @if
(!String.IsNullOrEmpty(@Model.errormsg)){ string alert = "danger";
<div class="alert alert-@alert alert-dismissible fade show" role="alert">
<h5>@Model.errormsg</h5> <button type="button" class="btn-close"
data-bs-dismiss="alert" aria-label="Close"></button> </div>
} <div class="input-group pb-3 pt-3 container sendtext">
<input class="form-control rounded" placeholder="Send a message..."
name="message" id="message" autocomplete="off" required maxlength="250"/>
<button type="submit" value="Send" class="btn btn-primary">Send <i class="fas fa-
paper-plane"></i></button> </div> </div> </form> @section
Styles{ <style> .sendtext{ min-height: 60px;
} .footer{ display: none; } .bottom{
background-color: var(--bs-light); border-top: 1px solid var(--bs-
gray-500); } </style> <link
href="@Url.Content("~/css/message.css")" rel="stylesheet" type="text/css" />
@section Scripts{ <script> window.scrollTo(0,
document.body.scrollHeight); </script> }
./Views/Authentication/Error.cshtml

```

```

@model scsp.Models.ErrorViewModel @{} ViewData["Title"] = "Error"; }
<h1>@ViewData["Title"]</h1> <div class="alert alert-danger" role="alert">
<h2>Error Occured while registering</h2> @if(Model.Title != null){
<h4>@Model.Title</h4> } @if(Model.Details != null){
<h5>@Model.Details</h5> } </div>
./Views/Authentication/Register.cshtml

```

```

@model scsp.ViewModels.AuthRegisterViewModel @{} ViewData["Title"] =
Model.Title; } @section Styles { <link
href="@Url.Content("~/css/login.css")" rel="stylesheet" type="text/css" /> }
<div class="text-center"> @if (!String.IsNullOrEmpty(@Model.AlertMsg)) {
string alert = @Model.AlertType; <div class="alert alert-@alert alert-
dismissible fade show" role="alert"> <h5>@Model.AlertMsg</h5>
<button type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button> </div> } </div> <section class="ftco-
section"> <div class="container"> <div class="row justify-content-

```

```

center">
class="wrap">
<div class="d-flex">
<h3 class="mb-4">Register</h3>
</div>
class="signin-form">
<input placeholder="Username" type="text" class="form-control" required
maxlength="32" minlength="1" name="username" id="username">
<label class="form-control-placeholder" for="username">Username <span class="text
text-danger">*</span></label>
</div>
<input id="password" type="password" class="form-control" required minlength="8"
name="password" placeholder="Password">
<label class="form-control-placeholder" for="password">Password <span class="text text-
danger">*</span></label>
<span toggle="#password"
class="fa fa-fw fa-eye field-icon toggle-password"></span>
<span class="password-minlength">Password should be atleast 8 characters</span>
</div>
<input placeholder="First Name" type="text" class="form-control" required
maxlength="32" minlength="1" name="FName" id="FName">
<label class="form-control-placeholder" for="FName">First Name <span class="text
text-danger">*</span></label>
</div>
<input placeholder="Last Name" type="text" class="form-control" maxlength="32"
name="LName" id="LName">
<label class="form-
control-placeholder" for="FName">Last Name</label>
</div>
<input placeholder="Bio" type="text" class="form-control" maxlength="250"
name="Bio" id="Bio">
<label class="form-control-
placeholder" for="FName">Bio</label>
</div>
<button type="submit" class="form-control btn btn-primary rounded submit px-
3">Register</button>
</form>
<p class="text-center">Already a member? <a asp-
action="Login">Login</a></p>
</div>
</div>
</div>
</section>
@* <div class="row">
<div class="col-md-4">
<form asp-action="Register" id="register-form">
<div class="form-group">
<label asp-for="User.UserID"
class="control-label">Username</label>
<span class="text text-
danger">*</span>
<input asp-for="User.UserID" class="form-
control" required maxlength="32" minlength="1" name="username" id="username"/>
</div>
<div class="form-group">
<label asp-
for="password" class="control-label">Password</label>
<span class="text text-danger">*</span>
<input asp-for="password"
class="form-control" type="password" required minlength="8" name="password"
id="password"/>
</div>
<div class="form-group">
<label asp-for="User.FName" class="control-label">First Name</label>
<span class="text text-danger">*</span>
<input asp-
for="User.FName" class="form-control" required maxlength="32" name="FName"
id="FName"/>
</div>
<div class="form-group">
<label asp-for="User.LName" class="control-label">Last Name</label>
<input asp-for="User.LName" class="form-control" maxlength="32" name="LName"
id="LName"/>
</div>
<div class="form-group">
<label asp-for="User.Bio" class="control-label"></label>
<textarea asp-for="User.Bio" class="form-control" type="text" maxlength="250"
form="register-form" name="Bio" id="Bio"></textarea>
</div>
<br>
<div class="form-group">
<input type="submit"

```

```

value="Register" class="btn btn-primary" />                                </div>                                </form>
</div> </div> <br><hr><br> <div class="text-center">                                Already an user? <a asp-
action="Login">Login</a> </div> *@ @section Scripts {                                }
./Views/Authentication/Index.cshtml

```

```

@{    ViewData["Title"] = "Authentication"; } @section Styles {    <link
href="@Url.Content("~/css/auth.css")" rel="stylesheet" type="text/css" /> }
<div class="text-center">    <h1 class="display-4">Social Content Sharing
Platform</h1>    <a asp-controller="Authentication" asp-action="Login"
class="btn btn-primary">Login</a>    <a asp-controller="Authentication" asp-
action="Register" class="btn btn-primary">Register</a>    <a asp-
controller="Authentication" asp-action="Logout" class="btn btn-primary">Log
Out</a>    <br><hr><br>    <a asp-controller="User" asp-action="Index">User
Admin</a> </div>
./Views/Authentication/Login.cshtml

```

```

@model scsp.ViewModels.AuthLoginViewModel @{    ViewData["Title"] = Model !=
null ? Model.Title : "L"; } @section Styles {    <link
href="@Url.Content("~/css/login.css")" rel="stylesheet" type="text/css" /> }
<div class="text-center">    @if (Model != null && !
String.IsNullOrEmpty(Model.AlertMsg))    {        string alert =
Model.AlertType;        <div class="alert alert-@alert alert-dismissible fade
show" role="alert">            <h5>@Model.AlertMsg</h5>            <button
type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button>        </div>    } </div> <section class="ftco-
section">    <div class="container">        <div class="row justify-content-
center">            <div class="col-md-7 col-lg-5">                <div
class="wrap">                    <div class="login-wrap p-4 p-md-5">
<div class="d-flex">                        <div class="w-100">
<h3 class="mb-4">Sign In</h3>                                </div>
</div>                                <form asp-action="Login" id="login-form"
class="signin-form">                                <div class="form-group mt-3">
<input placeholder="Username" type="text" class="form-control" required
maxlength="32" minlength="1" name="username" id="username">
<label class="form-control-placeholder" for="username">Username <span class="text
text-danger">*</span></label>
</div>                                <div class="form-group">
<input id="password" type="password" class="form-control" required minlength="8"
name="password" placeholder="Password">                                <label
class="form-control-placeholder" for="password">Password <span class="text text-
danger">*</span></label>                                <span toggle="#password"
class="fa fa-fw fa-eye field-icon toggle-password"></span>
<span class="password-minlength">Password should be atleast 8 characters</span>
</div>                                <div class="form-group d-md-flex">
<div class="w-50 text-left">                                <label
class="checkbox-wrap checkbox-primary mb-0">Remember me for 1 day
<input type="checkbox" checked="checked" name="remember" id="remember">
<span class="checkmark"></span>                                </label>
</div>                                </div>                                <div
class="form-group">                                <button type="submit"
class="form-control btn btn-primary rounded submit px-3">Log In</button>
</div>                                </form>                                <p class="text-
center">Not a member? <a asp-action="Register">Register</a></p>
</div>                                </div>                                </div>                                </div>
</section> @section Scripts{    <script>        (function($) {
"use strict";        $(".toggle-password").click(function() {
$(this).toggleClass("fa-eye fa-eye-slash");        var input =

```

```

$(this).attr("toggle"));
{
    input.attr("type", "text");
} else {
input.attr("type", "password");
}
})(jQuery);
</script>
}
./Controllers/AuthenticationController.cs

```

```

using System; using System.Collections.Generic; using System.Diagnostics; using
System.Linq; using System.Threading.Tasks; using Microsoft.AspNetCore.Mvc; using
Microsoft.Extensions.Logging; using scsp.Models; using scsp.ViewModels; using
System.Security.Cryptography; using System.Security.Claims; using
Microsoft.AspNetCore.Authentication; using
Microsoft.AspNetCore.Authentication.Cookies; namespace scsp.Controllers {
public class AuthenticationController : Controller { private readonly
SCSPDataContext _context; public
AuthenticationController(SCSPDataContext context) { _context
= context; } public IActionResult Index() {
return View(); } public IActionResult Login(string message = "")
{
    AuthLoginViewModel vm = new AuthLoginViewModel() {
        Title = "Login",
        AlertMsg = message,
        AlertType =
        "danger",
        password = "",
        User = new User()
    };
    return View(vm); } public IActionResult
Register(string message, string alert = "danger")
{
    AuthRegisterViewModel vm = new AuthRegisterViewModel() {
        Title =
        "Register",
        AlertMsg = message,
        AlertType =
        alert,
        password = "",
        User = new User()
    };
    return View(vm); } public IActionResult
Error(string title, string details) {
    return View("Error",
    new ErrorViewModel { Title = title, Details = details }); } //
POST: Authentication/Register // To protect from overposting attacks,
enable the specific properties you want to bind to. // For more details,
see http://go.microsoft.com/fwlink/?LinkId=317598. [HttpPost]
[ValidateAntiForgeryToken] public async Task<IActionResult>
Register(string username, string fname, string lname, string bio, string
password) {
    if(username == null)
    return Register("username cannot be empty");
    if(password == null)
    return Register("password cannot be empty");
    if(fname == null)
    return Register("First Name cannot be empty");
    if(username.Length >
    32)
    return Register("Username cannot be greater than 32
    characters");
    if(password.Length < 8)
    return
    Register("Password cannot be smaller than 8 characters");
    if(fname.Length > 32)
    return Register("First Name cannot be
    greater than 32 characters");
    if(lname != null && lname.Length > 32)
    return Register("Last Name cannot be greater than 32 characters");
    if(bio != null && bio.Length > 250)
    return Register("Bio cannot be
    greater than 250 characters");
    User user = new User();
    user.UserID = username;
    user.FName = fname;
    user.LName =
    lname ?? "";
    user.Bio = bio ?? "";
    string initials =
    (String.IsNullOrEmpty(fname) ? "" : fname[0]) + " " + (String.IsNullOrEmpty(lname)
    ? "" : lname[0]);
    user.Photo =
    "https://avatars.dicebear.com/api/initials/"+initials+".svg?r=50&b=black";
    user.Follows = new List<Foll>();
    user.FollowedBy = new List<Foll>();
    user.Posts = new List<Post>();
    user.Donations = new List<Donation>();
    user.MessagesSent = new List<Message>();
    user.MessagesRecieved = new
    List<Message>();
    byte[] passwordbytes =
    System.Text.Encoding.UTF8.GetBytes(username + password);
    byte[]
    result;
    SHA512 shaM = SHA512.Create();
    result =
    shaM.ComputeHash(passwordbytes);
    string passwordhash =

```

```

Convert.ToBase64String(result);                user.PasswordHash = passwordhash;
try {                                           var userifalreadyexists =
_context.User.FirstOrDefault(m => m.UserID == username);
if(userifalreadyexists == null){               _context.Add(user);
await _context.SaveChangesAsync();           return
Register("Registration successful", "success"); }
else{                                         return Register("User with the username (" + username +
") already exists, please choose some other username."); }
} catch(Exception e){                       Console.WriteLine("Invalid");
Console.WriteLine(e);                       return Error("Registration Error", "Error
occured while registering"); } } // POST:
Authentication/Register // To protect from overposting attacks, enable
the specific properties you want to bind to. // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598. [HttpPost]
[ValidateAntiForgeryToken] public async Task<IActionResult> Login(string
username, string password, string remember) { if(username ==
null) return Login("Username cannot be empty"); if(password ==
null) return Login("Password cannot be empty");
Console.WriteLine("Remember Me:" + remember); string passwordhash;
// finding the password hash from username+password {
byte[] passwordbytes = System.Text.Encoding.UTF8.GetBytes(username + password);
byte[] result; SHA512 shaM = SHA512.Create();
result = shaM.ComputeHash(passwordbytes); passwordhash =
Convert.ToBase64String(result); } try {
var user = _context.User.FirstOrDefault(m => m.UserID == username);
if(user == null || user.PasswordHash != passwordhash){ return
Login("Incorrect Username or Password"); } else{
var claims = new List<Claim>{ new Claim(ClaimTypes.Name, username) };
var claimsIdentity = new ClaimsIdentity(claims,
CookieAuthenticationDefaults.AuthenticationScheme); var
authProperties = new AuthenticationProperties{
AllowRefresh = true, IsPersistent = true,
ExpiresUtc = DateTime.UtcNow.AddMinutes(remember == "on" ? 86400 : 30)
}; await
HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new
ClaimsPrincipal(claimsIdentity), authProperties); return
RedirectToAction("Index", username == "admin" ? "Admin" : "Home");
} catch(Exception e){ Console.WriteLine(e);
return Error("Login Error", "Error occured while logging in"); }
} public async Task<IActionResult> Logout(){ await
HttpContext.SignOutAsync(); return RedirectToAction("Index", "Home");
} } }
./Controllers/AdminController.cs

```

```

using System.Diagnostics; using Microsoft.AspNetCore.Authorization; using
Microsoft.AspNetCore.Mvc; using scsp.Models; using scsp.ViewModels; namespace
scsp.Controllers; public class AdminController : Controller { private
readonly ILogger<HomeController> _logger; private readonly SCSPDataContext
_context; public AdminController(ILogger<HomeController> logger,
SCSPDataContext context) { _logger = logger; _context =
context; } [Authorize] public IActionResult Index() {
var identity = HttpContext.User.Identity; var username = identity != null
? identity.Name : null; var user = _context.User.FirstOrDefault(m =>
m.UserID == username); if(user == null){ return
RedirectToAction("Logout", "Authentication"); } if(username !=
"admin"){ return Content("Not Authorized"); } var vm

```



```

= new AdminIndexViewModel{
posts = _context.Post.Count(),
messages = _context.Message.Count(),
_context.LikePost.Count(),
donations = _context.Donation.Count(),
_context.Donation.Sum(i => i.Amount),
comments = _context.Comment.Count(),
registrations = _context.User.Count(),
likes =
dislikes = _context.DislikePost.Count(),
donationTotal =
}; return View(vm); } }
./Controllers/HomeController.cs

```

```

using System.Diagnostics; using Microsoft.AspNetCore.Authorization; using
Microsoft.AspNetCore.Mvc; using scsp.Models; using scsp.ViewModels; namespace
scsp.Controllers; public class HomeController : Controller { private
readonly ILogger<HomeController> _logger; private readonly SCSPDataContext
_context; public HomeController(ILogger<HomeController> logger,
SCSPDataContext context) { _logger = logger; _context =
context; } [Authorize] public IActionResult Index() {
var identity = HttpContext.User.Identity; var username = identity != null
? identity.Name : null; var user = _context.User.FirstOrDefault(m =>
m.UserID == username); if(user == null){ return
RedirectToAction("Logout", "Authentication"); } var followers =
_context.UserUser.Where(uu => uu.Followee == user).ToList(); var
following = _context.UserUser.Where(uu => uu.Follower == user).ToList();
var posts = _context.Post.ToList(); var frndposts = new List<Post>();
foreach (var post in posts) { foreach (var frnd in following)
{ if(post.AuthorId == frnd.FolloweeId){
post.Author = _context.User.FirstOrDefault(u => u.UserID == post.AuthorId) ??
post.Author; var Comments = _context.Comment.Where(c =>
c.Post == post).ToList() ?? new List<Comment>(); var Likes =
_context.LikePost.Where( lp => lp.Post == post).ToList() ?? new List<LikePost>();
var Dislikes = _context.DislikePost.Where( dlp => dlp.Post == post).ToList() ??
new List<DislikePost>(); post.Comments = Comments;
post.Likes = Likes; post.Dislikes = Dislikes;
frndposts.Add(post); break; } }
} frndposts.Sort((a,b) =>
HelperFunctions.hotrank(a.Likes.Count,a.Dislikes.Count, a.Time)
.CompareTo(HelperFunctions.hotrank(b.Likes.Count, b.Dislikes.Count, b.Time)));
frndposts.Reverse(); HomeIndexViewModel vm = new HomeIndexViewModel{
currentuser = user, Followers = followers, Following =
following, Posts = frndposts }; return View(vm);
} }
./Controllers/DonationController.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using Microsoft.AspNetCore.Authorization; using
Microsoft.AspNetCore.Mvc; using Microsoft.AspNetCore.Mvc.Rendering; using
Microsoft.EntityFrameworkCore; using scsp.Models; using scsp.ViewModels;
namespace scsp.Controllers { public class DonationController : Controller
{ private readonly SCSPDataContext _context; public
DonationController(SCSPDataContext context) { _context =
context; } // GET: Donation [Authorize]
public async Task<IActionResult> Index() { var identity =
HttpContext.User.Identity; var username = identity != null ?
identity.Name : null; var user = _context.User.FirstOrDefault(m =>
m.UserID == username); if(user == null){ return
RedirectToAction("Logout", "Authentication"); }
return _context.Donation != null ? View(new
DonationIndexViewModel { currentuser = user,
Donations = await _context.Donation.Where(d => d.User == user).ToListAsync() }) :

```

```

Content("Entity set 'SCSPDataContext.Donation' is null.");
// GET: Donation/Create [Authorize] public IActionResult
Create(string message = "", string alert = "", double amount = 0.0) {
var identity = HttpContext.User.Identity; var username = identity !=
null ? identity.Name : null; var user =
_context.User.FirstOrDefault(m => m.UserID == username); if(user ==
null){ return RedirectToAction("Logout", "Authentication");
} DonationCreateViewModel vm = new DonationCreateViewModel(){
AlertMsg = message, AlertType = alert, Amount =
amount, currentuser = user, }; return
View(vm); } // POST: Donation/Create // To protect
from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[Authorize] [HttpPost] [ValidateAntiForgeryToken]
public async Task<IActionResult> Create(double amount) {
if(amount <= 0){ return Create("Amount cannot be negative or
zero", "danger", amount); } var identity =
HttpContext.User.Identity; var username = identity != null ?
identity.Name : null; var user = _context.User.FirstOrDefault(m =>
m.UserID == username); if(user == null){ return
RedirectToAction("Logout", "Authentication"); }
Donation donation = new Donation{ User = user,
Amount = amount, Time = DateTime.Now };
_context.Add(donation); await _context.SaveChangesAsync();
return RedirectToAction(nameof(Index)); } private
bool DonationExists(int id) { return
(_context.Donation?.Any(e => e.DonationID == id)).GetValueOrDefault();
} }
./Controllers/ExploreController.cs

```

```

using System.Diagnostics; using Microsoft.AspNetCore.Authorization; using
Microsoft.AspNetCore.Mvc; using scsp.Models; using scsp.ViewModels; namespace
scsp.Controllers; public class ExploreController : Controller { private
readonly ILogger<HomeController> _logger; private readonly SCSPDataContext
_context; public ExploreController(ILogger<HomeController> logger,
SCSPDataContext context) { _logger = logger; _context =
context; } [Authorize] public IActionResult Index() {
var identity = HttpContext.User.Identity; var username = identity != null
? identity.Name : null; var user = _context.User.FirstOrDefault(m =>
m.UserID == username); if(user == null){ return
RedirectToAction("Logout", "Authentication"); }
ExploreIndexViewModel vm = new ExploreIndexViewModel{ Users =
_context.User.ToList(), currentuser = user }; return
View(vm); } [Authorize] [HttpPost] public IActionResult
Index(string query) { query ??= ""; query = query.ToLower();
var identity = HttpContext.User.Identity; var username = identity != null
? identity.Name : null; var user = _context.User.FirstOrDefault(m =>
m.UserID == username); if(user == null){ return
RedirectToAction("Logout", "Authentication"); }
ExploreIndexViewModel vm = new ExploreIndexViewModel{ query = query,
currentuser = user, Users = _context.User.Where(u =>
u.UserID.ToLower().Contains(query) || u.FName.ToLower().Contains(query) ||
(u.LName?? "").ToLower().Contains(query)).ToList() }; return
View(vm); } }
./Controllers/CommentController.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using Microsoft.AspNetCore.Authorization; using
Microsoft.AspNetCore.Mvc; using Microsoft.AspNetCore.Mvc.Rendering; using
Microsoft.EntityFrameworkCore; using scsp.Models; namespace scsp.Controllers
{
    public class CommentController : Controller
    {
        private
        readonly SCSPDataContext _context;
        public
        CommentController(SCSPDataContext context)
        {
            _context =
            context;
        }
        // GET: Comment
        [Authorize]
        public async Task<IActionResult> Index()
        {
            return
            _context.Comment != null ?
            View(await
            _context.Comment.ToListAsync()) :
            Problem("Entity set
            'SCSPDataContext.Comment'
            is null.");
        }
        // GET:
        Comment/Details/5
        [Authorize]
        public async Task<IActionResult>
        Details(int? id)
        {
            if (id == null || _context.Comment ==
            null)
            {
                return NotFound();
            }
            var comment = await _context.Comment
            .FirstOrDefaultAsync(m =>
            m.CommentID == id);
            if (comment == null)
            {
                return NotFound();
            }
            return View(comment);
        }
        // GET: Comment/Create
        // public IActionResult Create()
        // {
        //     return View();
        // }
        // POST: Comment/Create
        // To protect from overposting attacks, enable the specific properties you want
        // to bind to.
        // For more details, see
        http://go.microsoft.com/fwlink/?LinkId=317598.
        [Authorize]
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async
        Task<IActionResult> Create(int id, string comment)
        {
            return Content("Comment
            cannot be empty");
        }
        if (comment.Length > 250) {
            return Content("Comment cannot be longer than 250 characters");
        }
        var identity = HttpContext.User.Identity;
        var username = identity !=
        null ? identity.Name : null;
        var user =
        _context.User.FirstOrDefault(m => m.UserID == username);
        if (user ==
        null) {
            return RedirectToAction("Logout", "Authentication");
        }
        var post = _context.Post.FirstOrDefault(p => p.PostID == id);
        if (post == null) {
            return Content("Post does not exist");
        }
        var Comment = new Comment {
            Post = post,
            Content = comment,
            Author = user,
            Time =
            DateTime.Now
        };
        _context.Add(Comment);
        await _context.SaveChangesAsync();
        return
        RedirectToAction(nameof(Details), "Post", new {id = id});
        // GET: Comment/Edit/5
        [Authorize]
        public async
        Task<IActionResult> Edit(int? id)
        {
            if (id == null ||
            _context.Comment == null)
            {
                return NotFound();
            }
            var comment = await _context.Comment.FindAsync(id);
            if (comment == null)
            {
                return NotFound();
            }
            return View(comment);
        }
        // POST:
        Comment/Edit/5
        // To protect from overposting attacks, enable the
        // specific properties you want to bind to.
        // For more details, see
        http://go.microsoft.com/fwlink/?LinkId=317598.
        [Authorize]
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async
        Task<IActionResult> Edit(int id, [Bind("CommentID,Content,Time")] Comment
        comment)
        {
            if (id != comment.CommentID)
            {
                return NotFound();
            }
            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(comment);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if
                    (!CommentExists(comment.CommentID))
                    {
                        return NotFound();
                    }
                    else

```

```

        throw;
    }
    return RedirectToAction(nameof(Index));
}
View(comment);
} // GET: Comment/Delete/5
[Authorize] public async Task<IActionResult> Delete(int? id)
{
    if (id == null || _context.Comment == null)
    {
        return NotFound();
    }
    var comment = await
        _context.Comment
            .FirstOrDefaultAsync(m => m.CommentID == id);
    if (comment == null)
    {
        return NotFound();
    }
    return View(comment);
} // POST:
Comment/Delete/5 [Authorize] [HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken] public async Task<IActionResult>
DeleteConfirmed(int id)
{
    if (_context.Comment == null)
    {
        return Problem("Entity set 'SCSPDataContext.Comment' is
        null.");
    }
    var comment = await
        _context.Comment.FindAsync(id);
    if (comment != null)
    {
        await
            _context.Comment.Remove(comment);
        _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
} // GET: Comment/Like/5
[Authorize] public async
Task<IActionResult> Like(int id)
{
    if (_context.Comment ==
    null)
    {
        return Problem("Entity set
        'SCSPDataContext.Comment' is null.");
    }
    var comment =
        _context.Comment.FirstOrDefault(c => c.CommentID == id);
    if (comment
    == null){
        return NotFound();
    }
    var
    identity = HttpContext.User.Identity;
    var username = identity !=
    null ? identity.Name : null;
    var user =
        _context.User.FirstOrDefault(m => m.UserID == username);
    if (user ==
    null || username == null){
        return RedirectToAction("Logout",
        "Authentication");
    }
    DislikeComment? dlp =
        _context.DislikeComment.FirstOrDefault(dlp => dlp.Comment == comment &&
        dlp.Author == user);
    if (dlp != null){
        _context.Comment.Update(comment);
        await
            _context.DislikeComment.Remove(dlp);
        _context.SaveChangesAsync();
    }
    LikeComment? lp =
        _context.LikeComment.FirstOrDefault(lp => lp.Comment == comment && lp.Author ==
        user);
    if (lp != null){
        return
        RedirectToAction(nameof(UnLike), new {id = id});
    }
    lp =
    new LikeComment{
        Author = user,
        AuthorId =
        username,
        Comment = comment
    };
    _context.LikeComment.Add(lp);
    await _context.SaveChangesAsync();
    comment.Likes.Add(lp);
    _context.Comment.Update(comment);
    await _context.SaveChangesAsync();
    return
    RedirectToAction(nameof(Details), "Post", new {id = comment.PostId});
}
// GET: Comment/Dislike/5
[Authorize] public async
Task<IActionResult> Dislike(int id)
{
    if (_context.Comment
    == null)
    {
        return Problem("Entity set
        'SCSPDataContext.Comment' is null.");
    }
    var comment =
    await _context.Comment.FindAsync(id);
    if (comment == null){
        return
        NotFound();
    }
    var identity =
    HttpContext.User.Identity;
    var username = identity != null ?
    identity.Name : null;
    var user = _context.User.FirstOrDefault(m =>
    m.UserID == username);
    if (user == null || username == null){
        return RedirectToAction("Logout", "Authentication");
    }
    LikeComment? lp = _context.LikeComment.FirstOrDefault(lp => lp.Comment == comment
    && lp.Author == user);
    if (lp != null){
        _context.Comment.Update(comment);
        await
            _context.LikeComment.Remove(lp);
        _context.SaveChangesAsync();
    }
    DislikeComment? dlp =

```

```

_context.DislikeComment.FirstOrDefault(dlp => dlp.Comment == comment &&
dlp.Author == user);          if(dlp != null){          return
RedirectToAction(nameof(UnDislike), new {id = id});          }
dlp = new DislikeComment{          Author = user,
AuthorId = username,          Comment = comment          };
_context.DislikeComment.Add(dlp);          await _context.SaveChangesAsync();
comment.Dislikes.Add(dlp);          _context.Comment.Update(comment);
await _context.SaveChangesAsync();          return
RedirectToAction(nameof(Details), "Post", new {id = comment.PostId});          }
// GET: Comment/UnLike/5          [Authorize]          public async
Task<IActionResult> UnLike(int id)          {          if (_context.Comment
== null)          {          return Problem("Entity set
'SCSPDataContext.Comment' is null.");          }          var comment =
await _context.Comment.FindAsync(id);          if (comment == null){
return NotFound();          }          var identity =
HttpContext.User.Identity;          var username = identity != null ?
identity.Name : null;          var user = _context.User.FirstOrDefault(m =>
m.UserID == username);          if(user == null || username == null){
return RedirectToAction("Logout", "Authentication");          }
LikeComment? lp = _context.LikeComment.FirstOrDefault(lp => lp.Comment == comment
&& lp.Author == user);          if(lp != null){
comment.Likes.Remove(lp);          _context.Comment.Update(comment);
_context.LikeComment.Remove(lp);          await
_context.SaveChangesAsync();          }          else return
RedirectToAction(nameof(Like), new {id = id});          return
RedirectToAction(nameof(Details), "Post", new {id = comment.PostId});          }
// GET: Comment/UnDislike/5          [Authorize]          public async
Task<IActionResult> UnDislike(int id)          {          if
(_context.Comment == null)          {          return Problem("Entity
set 'SCSPDataContext.Comment' is null.");          }          var
comment = await _context.Comment.FindAsync(id);          if (comment ==
null){          return NotFound();          }          var
identity = HttpContext.User.Identity;          var username = identity !=
null ? identity.Name : null;          var user =
_context.User.FirstOrDefault(m => m.UserID == username);          if(user ==
null || username == null){          return RedirectToAction("Logout",
"Authentication");          }          DislikeComment? dlp =
_context.DislikeComment.FirstOrDefault(dlp => dlp.Comment == comment &&
dlp.Author == user);          if(dlp != null){
comment.Dislikes.Remove(dlp);          _context.Comment.Update(comment);
_context.DislikeComment.Remove(dlp);          await
_context.SaveChangesAsync();          } else return
RedirectToAction(nameof(Dislike), new {id = id});          return
RedirectToAction(nameof(Details), "Post", new {id = comment.PostId});          }
private bool CommentExists(int id)          {          return
(_context.Comment?.Any(e => e.CommentID == id)).GetValueOrDefault();          }
} }
./Controllers/HelperFunctions.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; namespace scsp.Controllers { public class
HelperFunctions { public static double epoch_seconds(DateTime date){
var td = date - DateTime.UnixEpoch;          return td.Days * 86400 +
td.Seconds + ((double)(td.Milliseconds) / 1000);          }          public static
double hotrank(int likes, int dislikes, DateTime date){          double s =
likes - dislikes;          double order = Math.Log(Math.Max(Math.Abs(s), 1),
10);          double sign = ( s > 0 ? 1 : ( s < 0 ? -1 : 0 ) );

```

```

using System.Diagnostics; using Microsoft.AspNetCore.Authorization; using
Microsoft.AspNetCore.Mvc; using Microsoft.EntityFrameworkCore; using scsp.Models;
using scsp.ViewModels; namespace scsp.Controllers; public class
ProfileController : Controller { private readonly ILogger<HomeController>
_logger; private readonly SCSPDataContext _context; public
ProfileController(ILogger<HomeController> logger, SCSPDataContext context) {
_logger = logger; _context = context; } [Authorize] public
IActionResult Index() { var identity = HttpContext.User.Identity;
var username = identity != null ? identity.Name : null; var user =
_context.User.FirstOrDefault(m => m.UserID == username); if(user ==
null){ return RedirectToAction("Logout", "Authentication"); }
var posts = _context.Post.Where(p => p.Author == user).OrderBy(post =>
post.Time).Reverse().ToList(); foreach (var post in posts) {
post.Likes = _context.LikePost.Where(l => l.Post == post).ToList();
post.Dislikes = _context.DislikePost.Where(d => d.Post == post).ToList();
post.Comments = _context.Comment.Where(c => c.Post == post).ToList(); }
var followers = _context.UserUser.Where(uu => uu.Followee == user).ToList();
var following = _context.UserUser.Where(uu => uu.Follower == user).ToList();
ProfileIndexViewModel vm = new ProfileIndexViewModel{ currentuser =
user, Posts = posts, Followers = followers,
Following = following, }; return View(vm); } [Authorize]
public IActionResult Explore(string id) { var identity =
HttpContext.User.Identity; var username = identity != null ?
identity.Name : null; var cuser = _context.User.FirstOrDefault(m =>
m.UserID == username); if(cuser == null){ return
RedirectToAction("Logout", "Authentication"); } if(username ==
id) return RedirectToAction(nameof(Index)); var username =
id; var user = _context.User.FirstOrDefault(m => m.UserID == username);
if(user == null){ return Content("User (" + username + ") not
found"); } var posts = _context.Post.Where(p => p.Author ==
user).OrderBy(post => post.Time).Reverse().ToList(); var followers =
_context.UserUser.Where(uu => uu.Followee == user).ToList(); var
following = _context.UserUser.Where(uu => uu.Follower == user).ToList();
bool ifollowhim = false, hefollowsme = false; foreach (var follower in
followers) { ifollowhim = ifollowhim || (follower.Follower ==
cuser); } foreach (var followee in following) {
hefollowsme = hefollowsme || followee.Followee == cuser; }
foreach (var post in posts) { post.Likes =
_context.LikePost.Where(l => l.Post == post).ToList(); post.Dislikes
= _context.DislikePost.Where(d => d.Post == post).ToList();
post.Comments = _context.Comment.Where(c => c.Post == post).ToList(); }
ProfileExploreViewModel vm = new ProfileExploreViewModel{ user =
user, Posts = posts, Followers = followers,
Following = following, ifollowhim = ifollowhim,
hefollowsme = hefollowsme, currentuser = cuser };
return View(vm); } // edit profile [Authorize] public async
Task<IActionResult> Edit() { var identity =
HttpContext.User.Identity; var username = identity != null ?

```

```

identity.Name : null;                var user = await
_context.User.FindAsync(username);    if (user == null)                {
return RedirectToAction("Logout", "Authentication");                }
ProfileEditViewModel vm = new ProfileEditViewModel{                currentuser =
user                };                return View(vm);                }                // POST:
User/Edit/5                // To protect from overposting attacks, enable the specific
properties you want to bind to.                // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.                [Authorize]
[HttpPost]                [ValidateAntiForgeryToken]                public async
Task<IActionResult> Edit(string username, string fname, string lname, string bio)
{                var identity = HttpContext.User.Identity;                var username
= identity != null ? identity.Name : null;                var user = await
_context.User.FindAsync(username);                if (user == null || username !=
username){                return RedirectToAction("Logout", "Authentication");
}                user.FName = fname;                user.LName = lname;
user.Bio = bio;                try{                _context.Update(user);
await _context.SaveChangesAsync();                }                catch
(DbUpdateConcurrencyException){                return NotFound();                }
return RedirectToAction(nameof(Index));                }                [Authorize]                public
async Task<IActionResult> UpdateDP(string msg = "")                {                var
identity = HttpContext.User.Identity;                var username = identity != null
? identity.Name : null;                var user = await
_context.User.FindAsync(username);                if (user == null)                {
return RedirectToAction("Logout", "Authentication");                }
ProfileUpdateDPViewModel vm = new ProfileUpdateDPViewModel{
currentuser = user,                AlertMsg = msg,                AlertType =
"danger"                };                return View(vm);                }                // POST:
User/Edit/5                // To protect from overposting attacks, enable the specific
properties you want to bind to.                // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.                [Authorize]
[HttpPost]                [ValidateAntiForgeryToken]                public async
Task<IActionResult> UpdateDP(string username, IFormFile file)                {
var identity = HttpContext.User.Identity;                var username = identity !=
null ? identity.Name : null;                var user = await
_context.User.FindAsync(username);                if (user == null || username !=
username){                return RedirectToAction("Logout", "Authentication");
}                string initials = (String.IsNullOrEmpty(user.FName) ? "" :
user.FName[0]) + "" + (String.IsNullOrEmpty(user.LName) ? "" : user.LName[0]);
string photo =
"https://avatars.dicebear.com/api/initials/"+initials+".svg?r=50&b=black";
if(file != null && file.Length > 0){                if(file.Length > 1 * 1000 *
1000) return RedirectToAction(nameof(UpdatedP), new {msg="File is too big. Max
size is 1MB"});                if(!file.FileName.ToLower().EndsWith(".png") &&
!file.FileName.ToLower().EndsWith(".jpg") )                return
RedirectToAction(nameof(UpdatedP), new {msg="Only .jpg or .png files are
allowed"});                // return RedirectToAction(nameof(UpdatedP), new
{msg="Roadblock temporary"});                using(var target = new
MemoryStream()){                file.CopyTo(target);                var
barray = target.ToArray();                photo = "data:image/;base64," +
Convert.ToBase64String(barray);                }                }
user.Photo = photo;                try{                _context.Update(user);
await _context.SaveChangesAsync();                }                catch
(DbUpdateConcurrencyException){                return NotFound();                }
return RedirectToAction(nameof(Index));                }                [Authorize]
public async Task<IActionResult> Follow(string id)                {                var
username = id;                var identity = HttpContext.User.Identity;
var username = identity != null ? identity.Name : "";                username ??=

```

```

"";                var user = await _context.User.FirstOrDefaultAsync(e => e.UserID
== username);      var cuser = await _context.User.FirstOrDefaultAsync(e
=> e.UserID == cusername);      if (cuser == null){      return
RedirectToAction("Logout", "Authentication");      }      if(user
== null){      return Content("User " + username + "does not exist");
}      if(cuser == user){      return Content("Cannot follow
ownself");      }      Foll? relation =
_context.UserUser.FirstOrDefault(uu => uu.Follower == cuser && uu.Followee ==
user);      if(relation != null) return RedirectToAction(nameof(Unfollow),
new {id = id});      relation = new Foll{Follower = cuser, Followee =
user};      cuser.Follows.Add(relation);
user.FollowedBy.Add(relation);      try{
_context.Update(user);      _context.Update(cuser);
await _context.SaveChangesAsync();      }      catch (Exception e){
return Content("Concurrency Exception: " + e.Message);      }
return RedirectToAction(nameof(Explore), new {id = id});      }
[Authorize]      public async Task<IActionResult> Unfollow(string id)
{      var username = id;      var identity =
HttpContext.User.Identity;      var cusername = identity != null ?
identity.Name : "";      cusername ??= "";      var user = await
_context.User.FirstOrDefaultAsync(e => e.UserID == username);      var
cuser = await _context.User.FirstOrDefaultAsync(e => e.UserID == cusername);
if (cuser == null){      return RedirectToAction("Logout",
"Authentication");      }      if(user == null){
return Content("User " + username + "does not exist");      }
if(cuser == user){      return Content("Cannot follow ownself");
}      Foll? relation = _context.UserUser.FirstOrDefault(uu => uu.Follower
== cuser && uu.Followee == user);      if(relation == null) return
Content("Already not following");      cuser.Follows.Remove(relation);
user.FollowedBy.Remove(relation);      try{
_context.Update(user);      _context.Update(cuser);
_context.Remove(relation);      await _context.SaveChangesAsync();
}      catch (Exception e){      return Content("Concurrency
Exception: " + e.Message);      }      return
RedirectToAction(nameof(Explore), new {id = id});      }      private
bool UserExists(string id){      return (_context.User?.Any(e => e.UserID ==
id)).GetValueOrDefault();      } }
./Controllers/NotificationController.cs

```

```

using System.Diagnostics; using Microsoft.AspNetCore.Authorization; using
Microsoft.AspNetCore.Mvc; using scsp.Models; using scsp.ViewModels; namespace
scsp.Controllers; public class NotificationController : Controller {      private
readonly ILogger<HomeController> _logger;      private readonly SCSPDataContext
_context;      public NotificationController(ILogger<HomeController> logger,
SCSPDataContext context)      {      _logger = logger;      _context =
context;      }      [Authorize]      public IActionResult Index()      {
var identity = HttpContext.User.Identity;      var username = identity != null
? identity.Name : null;      var user = _context.User.FirstOrDefault(m =>
m.UserID == username);      if(user == null){      return
RedirectToAction("Logout", "Authentication");      }      var comments =
new List<Comment>();      var posts = _context.Post.Where(p => p.AuthorId ==
username).ToList();      foreach (var post in posts)      {      var
cms = _context.Comment.Where(c => c.PostId == post.PostID);      foreach
(var c in cms)      {      c.Author =
_context.User.Find(c.AuthorId) ?? new Models.User();      }
comments.AddRange(cms);      }      comments.Sort((c1,c2) =>
c1.Time.CompareTo(c2.Time));      comments.Reverse();      var vm = new

```



```

NotificationIndexViewModel{                currentuser = user,                comments
= comments                };                return View(vm);                } }
./Controllers/MessageController.cs

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using Microsoft.AspNetCore.Authorization; using
Microsoft.AspNetCore.Mvc; using Microsoft.AspNetCore.Mvc.Rendering; using
Microsoft.EntityFrameworkCore; using scsp.Models; using scsp.ViewModels;
namespace scsp.Controllers {                public class MessageController : Controller
{                private readonly SCSPDataContext _context;                public
MessageController(SCSPDataContext context)                {                _context =
context;                }                // GET: Message                [Authorize]
public IActionResult Index(string errormsg="")                {                var
identity = HttpContext.User.Identity;                var username = identity !=
null ? identity.Name : null;                var user =
_context.User.FirstOrDefault(m => m.UserID == username);                if(user ==
null){                return RedirectToAction("Logout", "Authentication");
}                var messages = _context.Message.Where(m => m.From == user || m.To
== user).ToList();                var users = new Dictionary<User, int>();
foreach (var message in messages)                {                User frnd;
if( message.FromId == username) frnd = _context.User.FirstOrDefault(u => u.UserID
== message.ToId) ?? user;                else frnd =
_context.User.FirstOrDefault(u => u.UserID == message.FromId) ?? user;
if(frnd == user) continue;                if(! users.Contains(frnd))
users.Add(frnd, 1);                else users[frnd]++;                }
var vm = new MessageIndexViewModel(){                errormsg = errormsg,
Users = users,                from = user                };                return
_context.Message != null ? View(vm) : Content("Entity set
'SCSPDataContext.Message' is null.");                }                [Authorize]
public IActionResult Send(string id, string message="", string errormsg = "")
{                var identity = HttpContext.User.Identity;                var
username = identity != null ? identity.Name : null;                var user =
_context.User.FirstOrDefault(m => m.UserID == username);                if(user ==
null){                return RedirectToAction("Logout", "Authentication");
}                var to_username = id;                var to_user =
_context.User.FirstOrDefault(m => m.UserID == to_username);
if(to_user == null){                return Content("User " + id + " not
found");                }                var messages = _context.Message
.Where(m => ( m.From == user && m.To == to_user ) || ( m.To == user && m.From ==
to_user ) )                .OrderBy(m => m.Time)                .ToList();
var vm = new MessageSendViewModel(){                errormsg = errormsg,
Messages = messages,                message = message,                from =
user,                to = to_user                };                return
_context.Message != null ? View(vm) : Content("Entity set
'SCSPDataContext.Message' is null.");                }                // POST:
Message/Send                // To protect from overposting attacks, enable the specific
properties you want to bind to.                // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.                [HttpPost]
[ValidateAntiForgeryToken]                [Authorize]                public async
Task<IActionResult> Send(string id, string message)                {                var
identity = HttpContext.User.Identity;                var username = identity !=
null ? identity.Name : null;                var user =
_context.User.FirstOrDefault(m => m.UserID == username);                if(user ==
null){                return RedirectToAction("Logout", "Authentication");
}                var to_username = id;                var to_user =
_context.User.FirstOrDefault(m => m.UserID == to_username);
if(to_user == null){                return Content("User " + id + " not

```

```

found");
    }
    if (String.IsNullOrEmpty(message)) {
return RedirectToAction(nameof(Send), new {id = id, message = message, errormsg =
"Message cannot be empty"});
    }
    if (message.Length >
250) {
return RedirectToAction(nameof(Send), new {id = id,
message = message, errormsg = "Message too long. Max length is 250
characters."});
    }
    Message Message = new Message{
From = user, To = to_user, Content = message, Time = DateTime.Now
    };
    _context.Add(Message);
    await _context.SaveChangesAsync();
return RedirectToAction(nameof(Send), new {id = id});
}
private bool MessageExists(int id)
    {
return
(_context.Message?.Any(e => e.MessageID == id)).GetValueOrDefault();
    }
}
./Controllers/UserController.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using Microsoft.AspNetCore.Authorization; using
Microsoft.AspNetCore.Mvc; using Microsoft.AspNetCore.Mvc.Rendering; using
Microsoft.EntityFrameworkCore; using scsp.Models; namespace scsp.Controllers
{
    public class UserController : Controller
    {
        private readonly
SCSPDataContext _context;
        public UserController(SCSPDataContext
context)
        {
            _context = context;
        }
        //
GET: User
        public async Task<IActionResult> Index()
        {
return _context.User != null ?
            View(await
_context.User.ToListAsync()) :
            Problem("Entity set
'SCSPDataContext.User' is null.");
        }
        // GET: User/Details/5
        public async Task<IActionResult> Details(string id)
        {
            if
(id == null || _context.User == null)
            {
return
NotFound();
            }
            var user = await
_context.User.FirstOrDefaultAsync(m => m.UserID == id);
            if (user ==
null)
            {
return NotFound();
            }
return View(user);
        }
        // GET: User/Create
        public
IActionResult Create()
        {
return View();
        }
        // POST: User/Create
        // To protect from overposting attacks, enable the
specific properties you want to bind to.
        // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult>
Create([Bind("UserID, PasswordHash, FName, LName, Bio")] User user)
        {
            if (ModelState.IsValid)
            {
                _context.Add(user);
await _context.SaveChangesAsync();
return
RedirectToAction(nameof(Index));
            }
            return View(user);
        }
        // GET: User/Edit/5
        public async Task<IActionResult>
Edit(string id)
        {
            if (id == null || _context.User == null)
            {
return NotFound();
            }
            var user =
await _context.User.FindAsync(id);
            if (user == null)
            {
return NotFound();
            }
            return View(user);
        }
        // POST: User/Edit/5
        // To protect from overposting attacks, enable the
specific properties you want to bind to.
        // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(string
id, [Bind("UserID, PasswordHash, FName, LName, Bio")] User user)
        {
            if (id != user.UserID)
            {
return NotFound();
            }
            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(user);
await
_context.SaveChangesAsync();
                }
                catch
(DbUpdateConcurrencyException)
                {
                    if
(!UserExists(user.UserID))
                    {
return
NotFound();
                    }
                    else

```

```

        throw;
    }
    return RedirectToAction(nameof(Index));
}
View(user); // GET: User/Delete/5 [Authorize]
public IActionResult Delete(string id) { if (id == null ||
_context.User == null) { return NotFound(); }
    var identity = HttpContext.User.Identity;
    var username = identity != null ? identity.Name : null;
    var user = _context.User.FirstOrDefault(m => m.UserID == username);
    if (user == null) { return RedirectToAction("Logout", "Authentication"); }
    if (username != id) { return Content("Not
Authorized to delete other users"); }
    View(user); // POST: User/Delete/5 [Authorize]
    [HttpPost, ActionName("Delete")] [ValidateAntiForgeryToken]
    public async Task<IActionResult> DeleteConfirmed(string id) {
        if (id == null || _context.User == null) { return
        NotFound(); }
        var identity = HttpContext.User.Identity;
        var username = identity != null ? identity.Name : null;
        var user = _context.User.FirstOrDefault(m => m.UserID == username);
        if (user == null) { return RedirectToAction("Logout", "Authentication"); }
        if (username != id) { return Content("Not
Authorized to delete other users"); }
        _context.User.Remove(user); await _context.SaveChangesAsync();
        return RedirectToAction("Logout", "Authentication"); }
    private bool UserExists(string id) { return
    (_context.User?.Any(e => e.UserID == id)).GetValueOrDefault(); }
}
./Controllers/PostController.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using Microsoft.AspNetCore.Mvc; using
Microsoft.AspNetCore.Authorization; using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore; using scsp.Models; using scsp.ViewModels;
namespace scsp.Controllers { public class PostController : Controller
{ private readonly SCSPDataContext _context; public
PostController(SCSPDataContext context) { _context =
context; } // GET: Post public async
Task<IActionResult> Index() { return _context.Post !=
null ? View(await _context.Post.ToListAsync()) :
Problem("Entity set 'SCSPDataContext.Post' is null."); } //
GET: Post/Details/5 public async Task<IActionResult> Details(int? id)
{ if (id == null || _context.Post == null) {
return NotFound(); } var post = await _context.Post
.FirstOrDefaultAsync(m => m.PostID == id); if (post == null) {
return Content("Post does not exist"); } var identity =
HttpContext.User.Identity; var username = identity != null ?
identity.Name : null; var user = _context.User.FirstOrDefault(m =>
m.UserID == username); if (user == null) { return
RedirectToAction("Logout", "Authentication"); }
post.Author = _context.User.FirstOrDefault(u => u.UserID == post.AuthorId) ?? new
User(); var Comments = _context.Comment.Where(c => c.Post ==
post).ToList() ?? new List<Comment>(); foreach (var Comment in
Comments) { Comment.Author =
_context.User.FirstOrDefault(u => u.UserID == Comment.AuthorId) ?? new User();
var LikesComment = _context.LikeComment.Where(lc => lc.Comment ==
Comment).ToList() ?? new List<LikeComment>(); var
DislikesComment = _context.DislikeComment.Where(dlc => dlc.Comment ==
Comment).ToList() ?? new List<DislikeComment>(); Comment.Likes =

```

```

LikesComment;                                Comment.Dislikes = DislikesComment;                                }
var Likes = _context.LikePost.Where( lp => lp.Post == post).ToList() ?? new
List<LikePost>();                                var Dislikes = _context.DislikePost.Where( dlp =>
dlp.Post == post).ToList() ?? new List<DislikePost>();                                bool
likedbyme = false, dislikedbyme = false;                                foreach (var Like in Likes)
{
    Like.Author = _context.User.FirstOrDefault(u => u.UserID ==
Like.AuthorId) ?? new User();                                likedbyme = likedbyme ||
Like.Author == user;                                }                                foreach (var Dislike in
Dislikes)                                {                                Dislike.Author =
_context.User.FirstOrDefault(u => u.UserID == Dislike.AuthorId) ?? new User();
dislikedbyme = dislikedbyme || Dislike.Author == user;                                }
Comments.Sort((a,b) =>
HelperFunctions.confidence(a.Likes.Count,a.Dislikes.Count)
.CompareTo(HelperFunctions.confidence(b.Likes.Count, b.Dislikes.Count)));
Comments.Reverse();                                post.Comments = Comments;
post.Likes = Likes;                                post.Dislikes = Dislikes;                                var vm
= new PostDetailsViewModel{                                Post = post,
AlertMsg = "",                                AlertType = "",                                likedbyme =
likedbyme,                                dislikedbyme = dislikedbyme,
currentuser = user,                                };                                return View(vm);                                }
// GET: Post/Create                                [Authorize]                                public IActionResult
Create(string message, string content="", string alert = "danger")                                {
var identity = HttpContext.User.Identity;                                var username = identity !=
null ? identity.Name : null;                                var user =
_context.User.FirstOrDefault(m => m.UserID == username);                                if(user ==
null){                                return RedirectToAction("Logout", "Authentication");
}                                PostCreateViewModel vm = new PostCreateViewModel(){
AlertMsg = message,                                AlertType = alert,                                Content
= content,                                currentuser = user,                                };
return View(vm);                                }                                // POST: Post/Create                                // To
protect from overposting attacks, enable the specific properties you want to bind
to.                                // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.                                [Authorize]
[HttpPost]                                [ValidateAntiForgeryToken]                                public async
Task<IActionResult> Create(string content, IFormFile file)                                {
if(String.IsNullOrEmpty(content)){                                return Create("Content cannot
be empty");                                }                                string photo = "";                                if(file
!= null && file.Length > 0){                                string errormsg = "";
if(file.Length > 1 * 1000 * 1000) errormsg += "File is too big. Max size is
1MB.";                                if(!file.FileName.ToLower().EndsWith(".png") &&
!file.FileName.ToLower().EndsWith(".jpg") )                                errormsg +=
"Only .jpg or .png files are allowed";
if(!String.IsNullOrEmpty(errormsg))                                return
RedirectToAction(nameof(Create), new {message=errormsg, content=content});
using(var target = new MemoryStream()){                                file.CopyTo(target);
var barray = target.ToArray();                                photo =
Convert.ToBase64String(barray);                                }                                }
var identity = HttpContext.User.Identity;                                var username = identity !=
null ? identity.Name : null;                                var user =
_context.User.FirstOrDefault(m => m.UserID == username);                                if(user ==
null){                                return RedirectToAction("Logout", "Authentication");
}                                Post post = new Post{                                Content = content,
Author = user,                                Time = DateTime.Now,                                Photo =
photo                                };                                try{                                _context.Add(post);
await _context.SaveChangesAsync();                                return
RedirectToAction(nameof(Index), "Profile");                                }catch (Exception e){
Console.WriteLine(e);                                PostCreateViewModel vm = new

```

```

PostCreateViewModel{
    e.Message,
    currentUser = user,
}
}
// GET: Post/Delete/5 [Authorize]
public async Task<IActionResult> Delete(int? id) {
    if (id == null || _context.Post == null) {
        return NotFound();
    }
    var post = await _context.Post
        .FirstOrDefaultAsync(m => m.PostID == id);
    if (post == null) {
        return NotFound();
    }
    var identity =
        HttpContext.User.Identity;
    var username = identity != null ?
        identity.Name : null;
    var user = _context.User.FirstOrDefault(m =>
        m.UserID == username);
    if (user == null) {
        return
            RedirectToAction("Logout", "Authentication");
    }
    if (post.Author != user) {
        return Unauthorized();
    }
    return View(post);
}
// POST: Post/Delete/5
[Authorize] [HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult>
DeleteConfirmed(int id) {
    if (_context.Post == null) {
        return Problem("Entity set 'SCSPDataContext.Post' is null.");
    }
    var post = await _context.Post.FindAsync(id);
    if (post == null) {
        return NotFound();
    }
    var identity = HttpContext.User.Identity;
    var username = identity !=
        null ? identity.Name : null;
    var user =
        _context.User.FirstOrDefault(m => m.UserID == username);
    if (user ==
        null) {
        return RedirectToAction("Logout", "Authentication");
    }
    if (post.Author != user) {
        return Unauthorized();
    }
    _context.Post.Remove(post);
    await
        _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index),
        "Profile");
}
// GET: Post/Like/5 [Authorize]
public async Task<IActionResult> Like(int id) {
    if
        (_context.Post == null) {
        return Problem("Entity
            set 'SCSPDataContext.Post' is null.");
    }
    var post =
        await _context.Post.FindAsync(id);
    if (post == null) {
        var identity =
            HttpContext.User.Identity;
        var username = identity != null ?
            var user = _context.User.FirstOrDefault(m =>
                m.UserID == username);
        if (user == null || username == null) {
            return RedirectToAction("Logout", "Authentication");
        }
        DislikePost? dlp = _context.DislikePost.FirstOrDefault(dlp => dlp.Post == post &&
            dlp.Author == user);
        if (dlp != null) {
            _context.Post.Update(post);
            await
                _context.DislikePost.Remove(dlp);
            _context.SaveChangesAsync();
        }
        LikePost? lp =
            _context.LikePost.FirstOrDefault(lp => lp.Post == post && lp.Author == user);
        if (lp != null) {
            return RedirectToAction(nameof(UnLike), new {id
                = id});
        }
        lp = new LikePost{
            Author =
                user,
            AuthorId = username,
            Post = post
        };
        _context.LikePost.Add(lp);
        await
            _context.SaveChangesAsync();
        post.Likes.Add(lp);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Details), "Post", new {id = id});
    }
}
// GET: Post/Dislike/5 [Authorize]
public async
Task<IActionResult> Dislike(int id) {
    if (_context.Post ==
        null) {
        return Problem("Entity set
            'SCSPDataContext.Post' is null.");
    }
    var post = await
        _context.Post.FindAsync(id);
    if (post == null) {
        var identity =
            return NotFound();
    }
    var identity =

```

```

HttpContext.User.Identity;
identity.Name : null;
m.UserID == username);
return RedirectToAction("Logout", "Authentication");
LikePost? lp = _context.LikePost.FirstOrDefault(lp => lp.Post == post &&
lp.Author == user);
post.Likes.Remove(lp);
_context.LikePost.Remove(lp);
}
DislikePost? dlp = _context.DislikePost.FirstOrDefault(dlp =>
dlp.Post == post && dlp.Author == user);
return RedirectToAction(nameof(UnDislike), new {id = id});
dlp = new DislikePost{
    Author = user,
    Post = post
};
_context.DislikePost.Add(dlp);
post.Dislikes.Add(dlp);
await _context.SaveChangesAsync();
return RedirectToAction(nameof(Details), "Post", new {id = id});
// GET: Post/Unlike/5 [Authorize] public async
Task<IActionResult> Unlike(int id) {
    if (_context.Post == null) {
        return Problem("Entity set 'SCSPDataContext.Post' is null.");
    }
    var post = await _context.Post.FindAsync(id);
    if (post == null) {
        return NotFound();
    }
    var identity = HttpContext.User.Identity;
    var username = identity != null ? identity.Name : null;
    var user = _context.User.FirstOrDefault(m => m.UserID == username);
    if(user == null || username == null){
        return RedirectToAction("Logout", "Authentication");
    }
    LikePost? lp = _context.LikePost.FirstOrDefault(lp => lp.Post == post &&
lp.Author == user);
    if(lp != null){
        _context.Post.Update(post);
        await _context.SaveChangesAsync();
    }
    else return RedirectToAction(nameof(Like), new {id = id});
return RedirectToAction(nameof(Details), "Post", new {id = id});
// GET: Post/UnDislike/5 [Authorize] public async
Task<IActionResult> UnDislike(int id) {
    if (_context.Post == null) {
        return Problem("Entity set 'SCSPDataContext.Post' is null.");
    }
    var post = await _context.Post.FindAsync(id);
    if (post == null) {
        return NotFound();
    }
    var identity = HttpContext.User.Identity;
    var username = identity != null ? identity.Name : null;
    var user = _context.User.FirstOrDefault(m => m.UserID == username);
    if(user == null || username == null){
        return RedirectToAction("Logout", "Authentication");
    }
    DislikePost? dlp = _context.DislikePost.FirstOrDefault(dlp => dlp.Post == post &&
dlp.Author == user);
    if(dlp != null){
        _context.Post.Update(post);
        await _context.SaveChangesAsync();
    }
    else return RedirectToAction(nameof(Dislike), new {id = id});
return RedirectToAction(nameof(Details), "Post", new {id = id});
private bool PostExists(int id) {
    return (_context.Post?.Any(e => e.PostID == id)).GetValueOrDefault();
}
}
./Data/SCSPDataContext.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using Microsoft.EntityFrameworkCore; using scsp.Models;

```

```

public class SCSPDataContext : DbContext
{
    public DbSet<scsp.Models.User> User { get; set; } = default!;
    public DbSet<scsp.Models.Foll> UserUser { get; set; } = default!;
    public DbSet<scsp.Models.Post> Post { get; set; } = default!;
    public DbSet<scsp.Models.Message> Message { get; set; } = default!;
    public DbSet<scsp.Models.Donation> Donation { get; set; } = default!;
    public DbSet<scsp.Models.Comment> Comment { get; set; } = default!;
    public DbSet<scsp.Models.LikePost> LikePost { get; set; } = default!;
    public DbSet<scsp.Models.DislikePost> DislikePost { get; set; } = default!;
    public DbSet<scsp.Models.LikeComment> LikeComment { get; set; } = default!;
    public DbSet<scsp.Models.DislikeComment> DislikeComment { get; set; } = default!;
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
    }
}
./Models/User.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using System.ComponentModel.DataAnnotations; using
System.ComponentModel.DataAnnotations.Schema; namespace scsp.Models { public
class User { [Key] [Required] public string UserID {get; set;} = "";
[Required] public string PasswordHash {get; set;} = "";
[Required] public string FName {get; set;} = "";
public string? LName {get; set;} = ""; public string? Bio {get; set;} =
""; public string Photo {get; set;} = ""; // foreign key to
user table (follows) [InverseProperty("Follower")] public
ICollection<Foll> Follows {get; set;} = new List<Foll>();
[InverseProperty("Followee")] public ICollection<Foll> FollowedBy {get;
set;} = new List<Foll>(); // posts created by user
[InverseProperty("Author")] public ICollection<Post> Posts {get; set;} =
new List<Post>(); // donations made by user
[InverseProperty("User")] public ICollection<Donation> Donations {get;
set;} = new List<Donation>(); //messages
[InverseProperty("From")] public ICollection<Message> MessagesSent {get;
set;} = new List<Message>(); [InverseProperty("To")] public
ICollection<Message> MessagesRecieved {get; set;} = new List<Message>(); } }
./Models/LikeComment.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using System.ComponentModel.DataAnnotations; using
System.ComponentModel.DataAnnotations.Schema; namespace scsp.Models { public
class LikeComment { [Key] [Required] public int LikeID {get;
set;} // foreign key [ForeignKey("Author")] public
string AuthorId {get; set;} = ""; [Required] public User Author
{get; set;} = new User(); [Required] public Comment Comment
{get; set;} = new Comment(); } }
./Models/DislikeComment.cs

```

```

using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using System.ComponentModel.DataAnnotations; using
System.ComponentModel.DataAnnotations.Schema; namespace scsp.Models { public
class DislikeComment { [Key] [Required] public int DislikeID
{get; set;} // foreign key [ForeignKey("Author")] public
string AuthorId {get; set;} = ""; [Required] public User Author
{get; set;} = new User(); [Required] public Comment Comment
{get; set;} = new Comment(); } }
./Models/Donation.cs

```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using System.ComponentModel.DataAnnotations; namespace
scsp.Models { public class Donation { [Key] [Required]
public int DonationID {get; set;} [Required]
[DataType(DataType.Date)] public DateTime Time {get; set;} = new
DateTime(); [Required] public double Amount {get; set;}
[Required] public User User {get; set;} = new User(); } }
./Models/DislikePost.cs
```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using System.ComponentModel.DataAnnotations; using
System.ComponentModel.DataAnnotations.Schema; namespace scsp.Models { public
class DislikePost { [Key] [Required] public int DislikeID
{get; set;} // foreign key [ForeignKey("Author")] public
string AuthorId {get; set;} = ""; [Required] public User Author
{get; set;} = new User(); [Required] public Post Post {get; set;}
} = new Post(); } }
./Models/ErrorMessageView.cs
```

```
namespace scsp.Models; public class ErrorMessageView { public string?
RequestId { get; set; } public string? Title {get; set;} public string?
Details {get; set;} public bool ShowRequestId =>
!string.IsNullOrEmpty(RequestId); }
./Models/Post.cs
```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using System.ComponentModel.DataAnnotations; using
System.ComponentModel.DataAnnotations.Schema; namespace scsp.Models { public
class Post { [Key] [Required] public int PostID {get; set;}
[Required] public string Content {get; set;} = ""; [Required]
[DataType(DataType.Date)] public DateTime Time {get; set;} = new
DateTime(); // foreign key [ForeignKey("Author")]
public string AuthorId {get; set;} = ""; // user(author)
[Required] public User Author {get; set;} = new User(); // photo
public string Photo {get; set;} = ""; // comments
[InverseProperty("Post")] public ICollection<Comment> Comments {get;
set;} = new List<Comment>(); [InverseProperty("Post")] public
ICollection<LikePost> Likes {get; set;} = new List<LikePost>();
[InverseProperty("Post")] public ICollection<DislikePost> Dislikes {get;
set;} = new List<DislikePost>(); } }
./Models/Comment.cs
```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using System.ComponentModel.DataAnnotations; using
System.ComponentModel.DataAnnotations.Schema; namespace scsp.Models { public
class Comment { [Key] [Required] public int CommentID {get;
set;} // foreign key [ForeignKey("Author")] public
string AuthorId {get; set;} = ""; [Required] public User Author
{get; set;} = new User(); [ForeignKey("Post")] public int PostId
{get; set;} [Required] public Post Post {get; set;} = new
Post(); [Required] public string Content {get; set;} = "";
[Required] [DataType(DataType.Date)] public DateTime Time {get; set;} =
new DateTime(); [InverseProperty("Comment")] public
ICollection<LikeComment> Likes {get; set;} = new List<LikeComment>();
[InverseProperty("Comment")] public ICollection<DislikeComment> Dislikes
{get; set;} = new List<DislikeComment>(); } }
```



```
./Models/Message.cs
```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using System.ComponentModel.DataAnnotations; using
System.ComponentModel.DataAnnotations.Schema; namespace scsp.Models { public
class Message { [Key] [Required] public int MessageID {get;
set;} [ForeignKey("From")] public string FromId {get; set;} = "";
[Required] public User From {get; set;} = new User();
[ForeignKey("To")] public string ToId {get; set;} = "";
[Required] public User To {get; set;} = new User(); [Required]
public string Content {get; set;} = ""; [Required]
[DataType(DataType.Date)] public DateTime Time {get; set;} = new
DateTime(); } }
./Models/LikePost.cs
```

```
using System; using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks; using System.ComponentModel.DataAnnotations; using
System.ComponentModel.DataAnnotations.Schema; namespace scsp.Models { public
class LikePost { [Key] [Required] public int LikeID {get;
set;} // foreign key [ForeignKey("Author")] public
string AuthorId {get; set;} = ""; [Required] public User Author
{get; set;} = new User(); [Required] public Post Post {get; set;
} = new Post(); } }
./Models/Foll.cs
```

```
using System; using System.Collections.Generic; using
System.ComponentModel.DataAnnotations; using
System.ComponentModel.DataAnnotations.Schema; using System.Linq; using
System.Threading.Tasks; namespace scsp.Models { public class Foll{
[Key] public int ID {get; set;} [ForeignKey("Follower")]
public string FollowerId {get; set;} = ""; [ForeignKey("Followee")]
public string FolloweeId {get; set;} = ""; public User Follower { get;
set; } = new User(); public User Followee { get; set; } = new User();
} }
./Program.cs
```

```
using Microsoft.AspNetCore.Authentication.Cookies; using
Microsoft.EntityFrameworkCore; using Microsoft.Extensions.DependencyInjection;
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddDbContext<SCSPDataContext>(options =>
options.UseSqlite(builder.Configuration.GetConnectionString("SCSPDataContext") ??
throw new InvalidOperationException("Connection string 'SCSPDataContext' not
found."))); // Add services to the container.
builder.Services.AddControllersWithViews();
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationSch
eme).AddCookie(options => { options.LoginPath = "/Authentication/Login"; });
builder.Services.AddMvc(); var app = builder.Build(); // Configure the HTTP
request pipeline. if (!app.Environment.IsDevelopment()) {
app.UseExceptionHandler("/Home/Error"); // The default HSTS value is 30 days.
You may want to change this for production scenarios, see
https://aka.ms/aspnetcore-hsts. app.UseHsts(); } app.UseHttpsRedirection();
app.UseStaticFiles(); app.UseAuthentication(); app.UseRouting();
app.UseAuthorization(); var cookiePolicyOptions = new CookiePolicyOptions{
MinimumSameSitePolicy = Microsoft.AspNetCore.Http.SameSiteMode.Lax };
app.UseCookiePolicy(cookiePolicyOptions); app.MapControllerRoute( name:
"default", pattern: "{controller=Home}/{action=Index}/{id?}"); app.Run();
./scsp.csproj
```

```
<Project Sdk="Microsoft.NET.Sdk.Web">    <PropertyGroup>
<TargetFramework>net6.0</TargetFramework>    <Nullable>enable</Nullable>
<ImplicitUsings>enable</ImplicitUsings>    </PropertyGroup>    <ItemGroup>
<PackageReference Include="Microsoft.AspNetCore.Identity.EntityFrameworkCore"
Version="6.0.9" />    <PackageReference
Include="Microsoft.AspNetCore.Identity.UI" Version="6.0.9" />
<PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="6.0.9">
<IncludeAssets>runtime; build; native; contentfiles; analyzers;
buildtransitive</IncludeAssets>    <PrivateAssets>all</PrivateAssets>
</PackageReference>    <PackageReference
Include="Microsoft.EntityFrameworkCore.Sqlite" Version="6.0.9" />
<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer"
Version="6.0.9" />    <PackageReference
Include="Microsoft.EntityFrameworkCore.Tools" Version="6.0.9">
<IncludeAssets>runtime; build; native; contentfiles; analyzers;
buildtransitive</IncludeAssets>    <PrivateAssets>all</PrivateAssets>
</PackageReference>    <PackageReference
Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="6.0.10" />
</ItemGroup> </Project>
```