Team members:

1. Shrinivas Vasant Shanbhag (UIN: 934008523)
2. Parangjothi Chandrasekaran (UIN: 434007347)

ClientServer

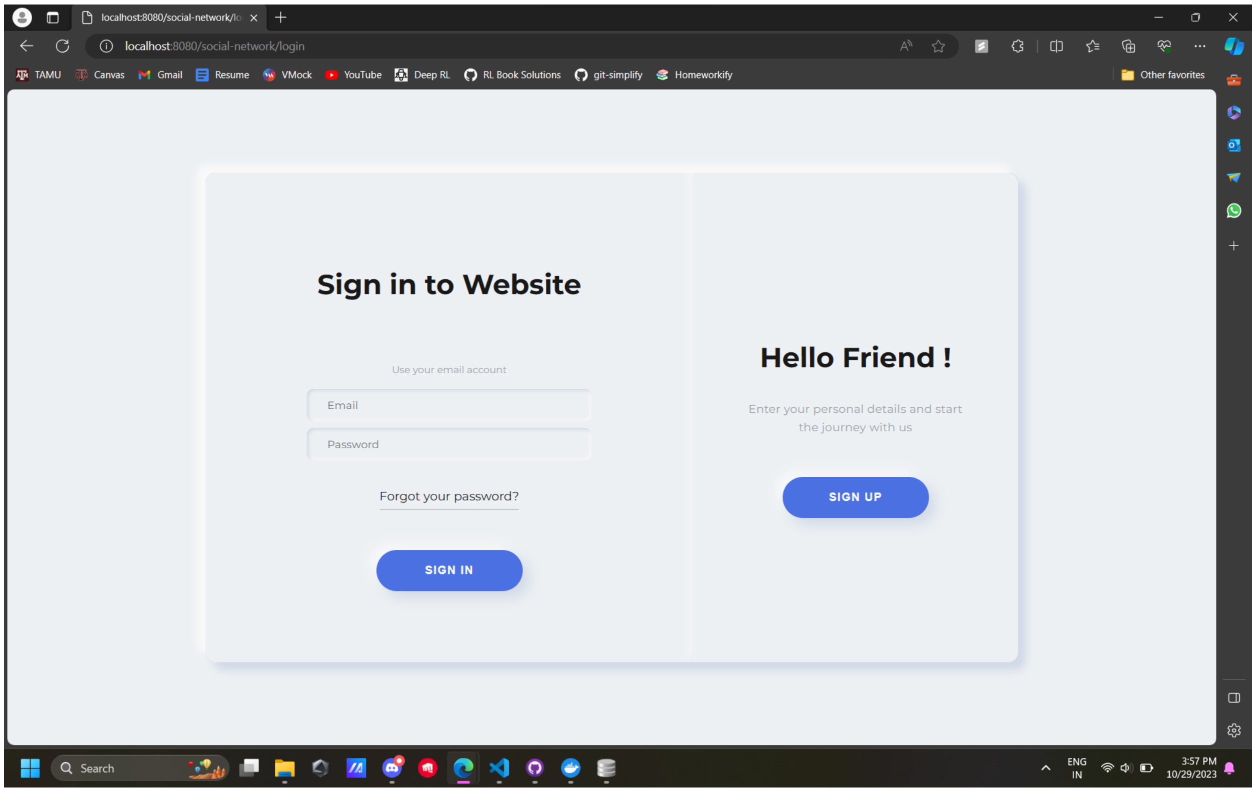
1. Requirements: Description with UI sketch of all use cases. Use cases should be described with the standard format including a short description and a sequence of user actions and system reactions. Each user action or system reaction might include a UI sketch. You should include all use cases, both from Iterations 1 and 2 and the new ones for Iteration 3.
   a. Login user:
      i. Description:
         1. Name: Login existing user into the system
         2. Actors: Normal User, Premium User, and Brand
         3. Goals: Login to the system with email and password, to interact with the website.
         4. Preconditions: the user is logged out of the system.
         5. Summary: User provides registered email id and password, and server will authenticate the user credentials, and either allows user to interact with the system or rejects the login request
         6. Related use cases: Signin and signup the user into the system
         7. Steps:

Actor actions	System responses
1. Visit login page	2. Loads login page
3. Enters registered email id and password	4. Authenticates and loads home page
5. User interacts with homepage to navigate to other pages	

         8. PostConditions: If the user is authenticated then, the user is allowed to access system resources. If the system is not able to authenticate

user, then it will reply with failed login message and return back to the original login page.

9. UI Sketches:



Enter your personal details and start the journey with us

b. Creating/publishing new post:

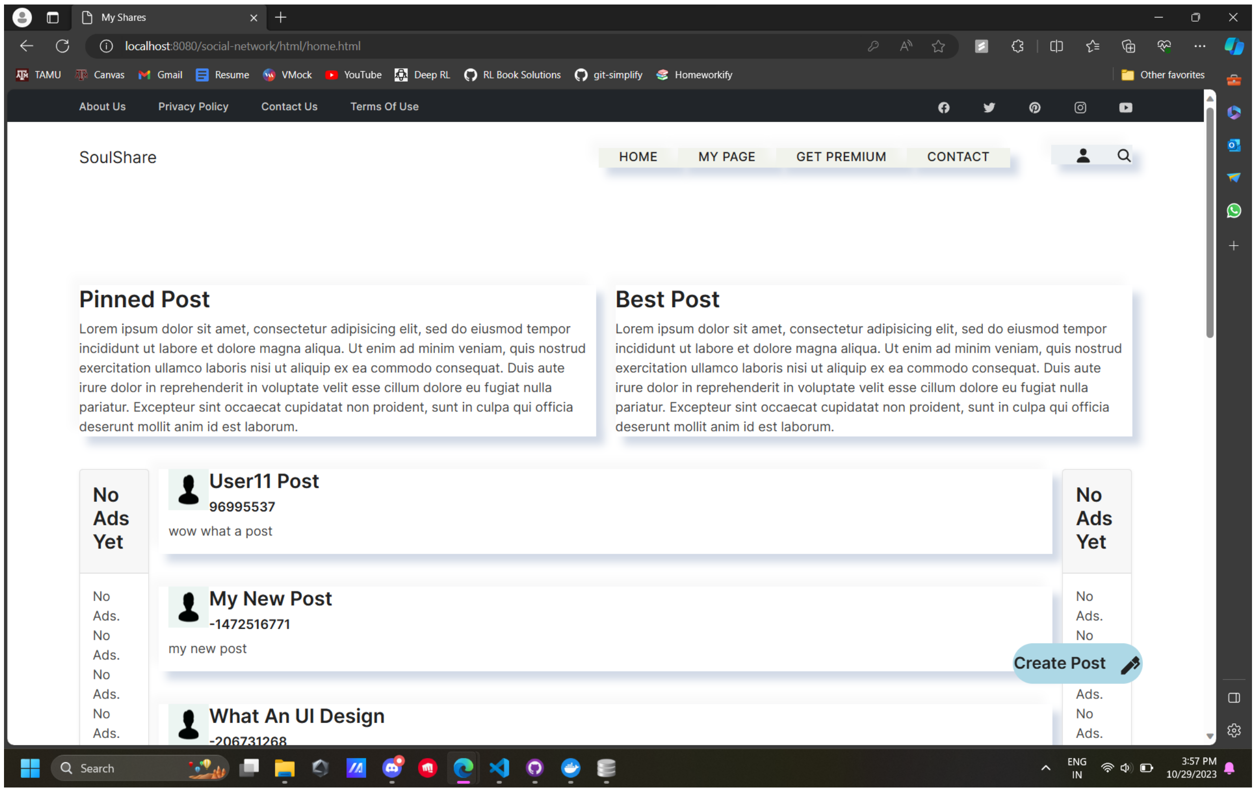
i. Description:

1. Name: Create a new post
2. Actors: Normal User, Premium User, and Brand
3. Goals: Create and publish a new post with title and content. So that is is available to other users
4. Preconditions: the user needs to be logged in.
5. Summary: User will click the post button from the My Page or the Homepage, then in the given prompt the user will add a title to the post and the actual content and upload an image file, then the post will be visible in the post section to the user and others as well.
6. Related use cases: posting from the post page and homepage

7. Steps:

Actor actions	System responses
1. User clicks “Create Post” button in My Page or in Homepage	2. Prompts new web page, with placeholder for post title, content and image
3. User enters title, content and image	
4. User clicks “Post” button	5. Acknowledges the post submission
	6. Makes the post visible in the Home page and in My Page

8. PostConditions: Post will be saved in databases, and post will be made visible in Homepage and My Page.
9. UI Sketches:



Pinned Post

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Best Post

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

No Ads Yet

No Ads. No Ads. No Ads. No Ads.

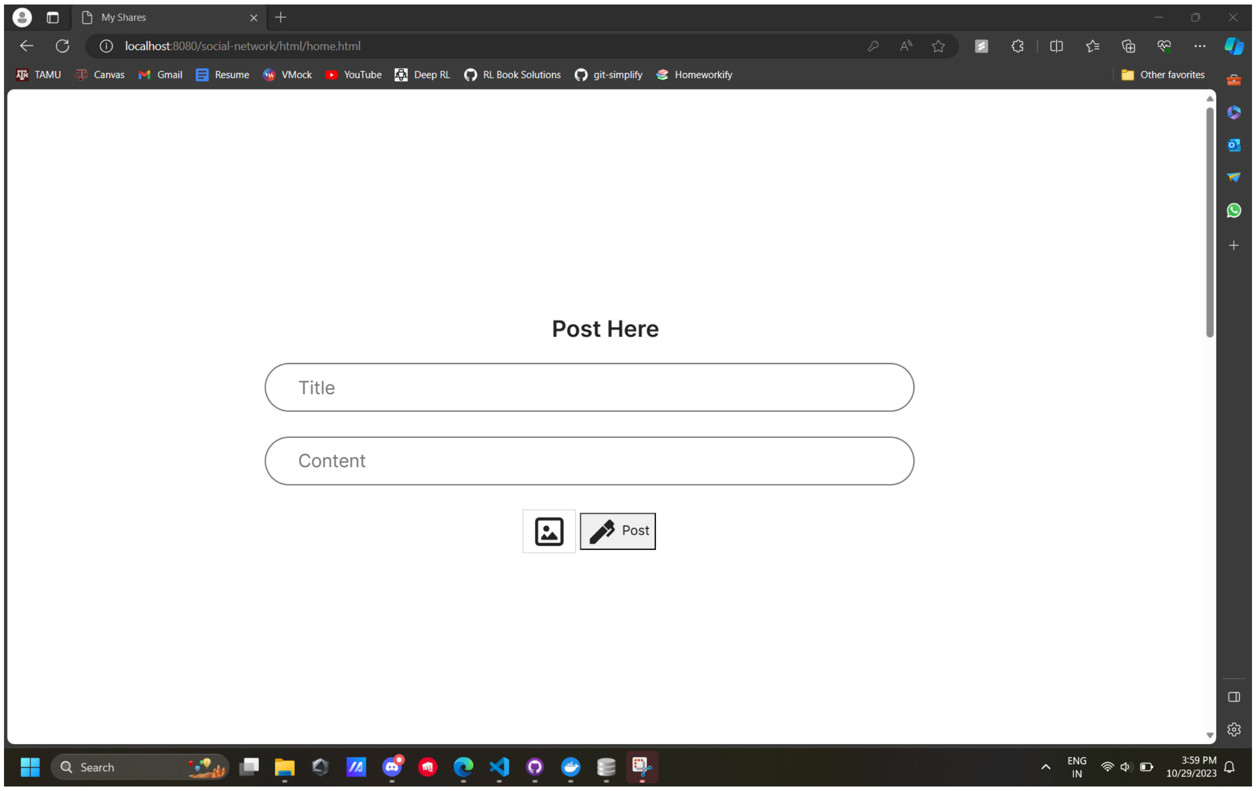
User11 Post
96995537
wow what a post

My New Post
-1472516771
my new post

What An UI Design
-206731268

No Ads Yet

No Ads. No Ads. No Ads.



c. Subscribing to Premium subscription:

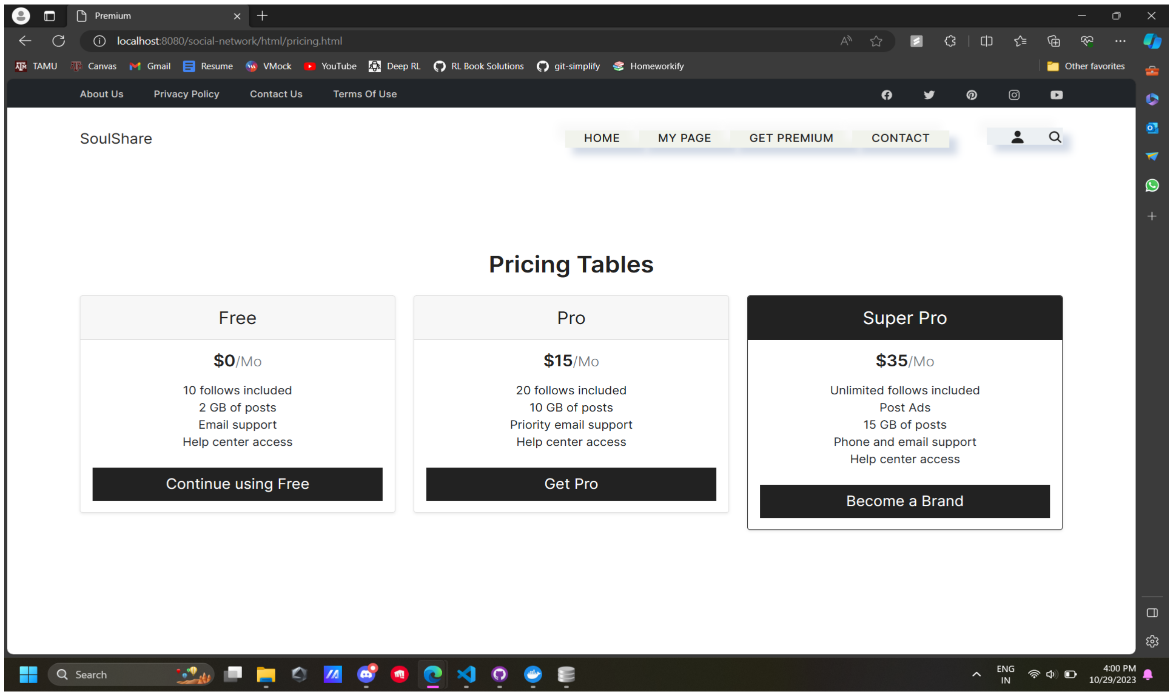
i. Description:

1. Name: Upgrading normal user to premium user.
2. Actors: Normal User, Premium User
3. Goals: Normal users can upgrade to Premium users.
4. Preconditions: the user needs to be logged in, and the user should be a normal user or premium user, so that they can be upgraded to the utmost brand user type.
5. Summary: Normal users will pay $15 to upgrade them to Pro/Premium user, and $35 to upgrade them to SuperPro/Brand user. Premium users can upgrade them to SuperPro/Brand users, so that they can create and manage their advertisements.
6. Related use cases: unique use case
7. Steps:

Actor actions	System responses
1. User clicks on Pricing tab	2. The system loads Pricing page, with three options like, normal, pro, and

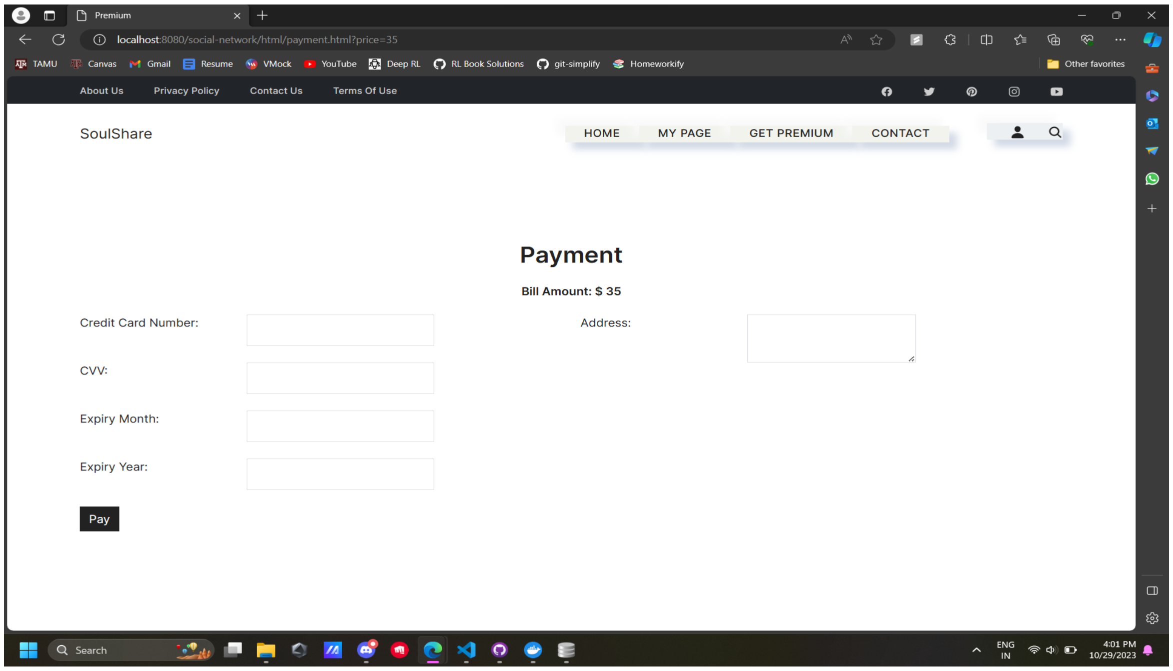
	super pro
3. User selects one of the option	4. System loads payment page, with place holder for credit card details
5. User enters credit card details	6. System processes transaction, and notifies user with user upgraded message

8. PostConditions: User upgraded message will be shown in Pricing page.
9. UI Sketches:



Pricing Tables

Free	Pro	Super Pro
$0/Mo	$15/Mo	$35/Mo
10 follows included 2 GB of posts Email support Help center access	20 follows included 10 GB of posts Priority email support Help center access	Unlimited follows included Post Ads 15 GB of posts Phone and email support Help center access
Continue using Free	Get Pro	Become a Brand



Payment

Bill Amount: $ 35

Credit Card Number:

CVV:

Expiry Month:

Expiry Year:

Address:

d. Updating user profile

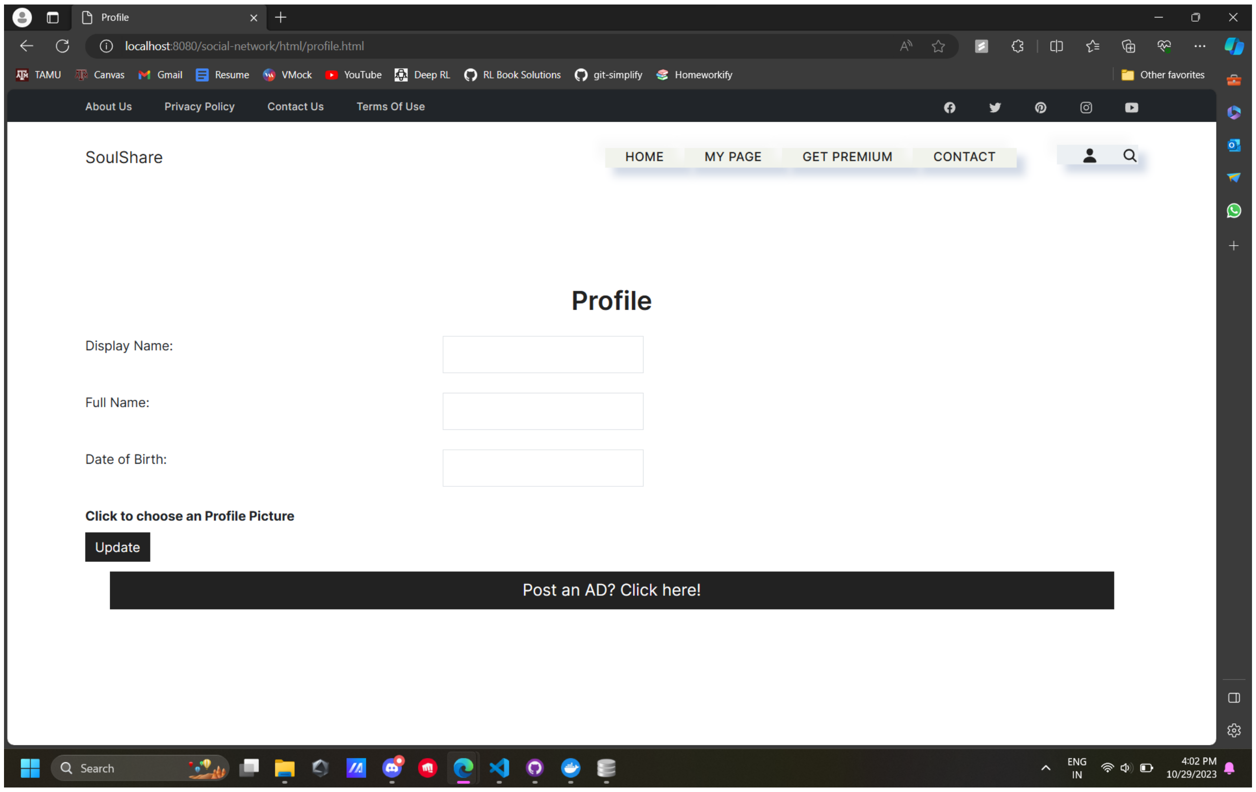
i. Description:

1. Name: Updating user profile.
2. Actors: Normal User, Premium User, and Brand
3. Goals: User will update their own information like display name, full name, date of birth, and profile photo.
4. Preconditions: the user needs to be logged in.
5. Summary: User will visit Profile page, and fills in the updated display name, full name, date of birth and a profile picture, which will overwrite the user profile details in system.
6. Related usecases: unique usecase
7. Steps:

Actor actions	System responses
1. User clicks on Profile tab	2. The system loads Profile page, with placeholders to enter display name, full name, date of birth and a profile picture
3. User fills display name, full name, date of birth and a profile picture, and clicks	4. The system returns the acknowledgement message saying profile is

on update button	updated successfully

8. PostConditions: system updates user profile in data base and it is reflected in the database and posts.
9. UI Sketches:



Profile

Display Name:

Full Name:

Date of Birth:

Click to choose an Profile Picture

Update

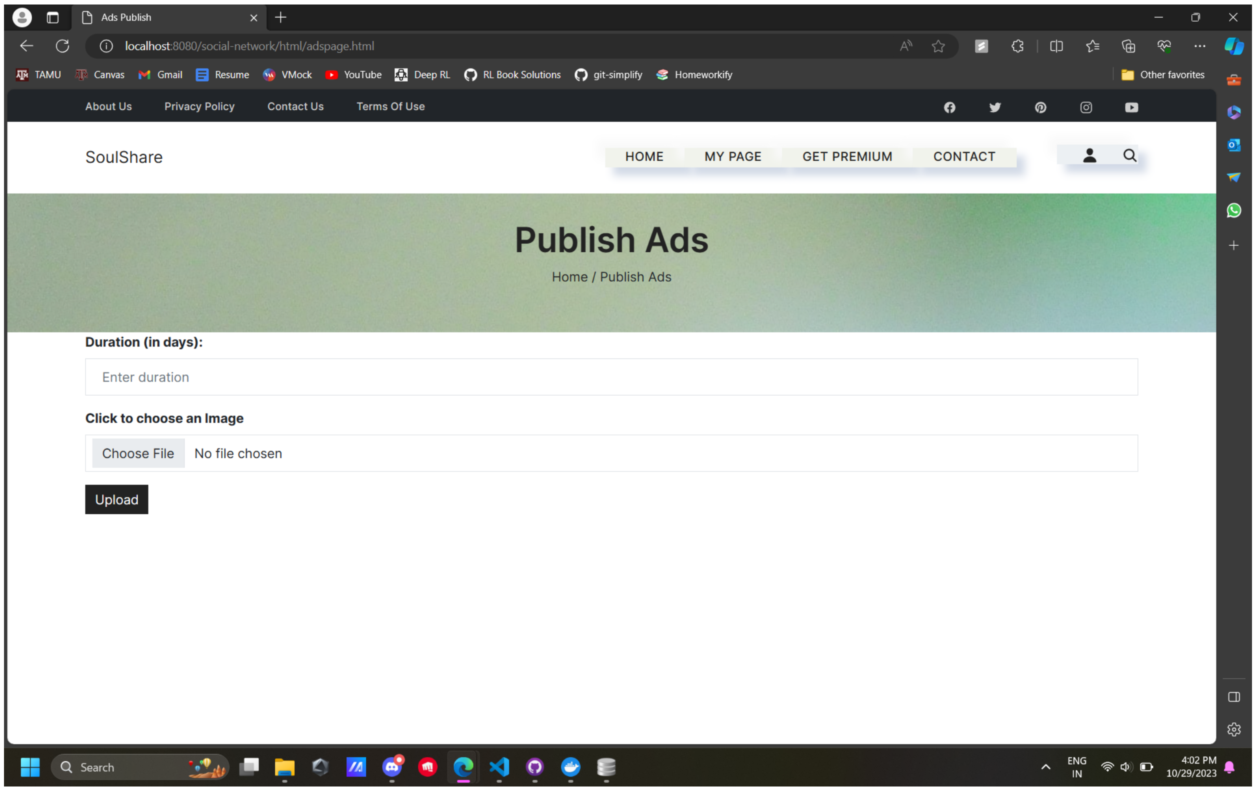
Post an AD? Click here!

e. Post Ads
   i. Description:
      1. Name: Post new ads in the system.
      2. Actors: Premium User and Brand
      3. Goals: Premium User and Brand users can create new ads that will be visible on the homepage to all other users.
      4. Preconditions: the user needs to be either a Premium User or Brand type user

5. Summary: User will visit Profile page, and clicks post an ads, then they will publish an ad in the form of picture. This ad will be visible to all the users in their homepage in a timely manner.
6. Related usecases: creating/publishing a post
7. Steps:

Actor actions	System responses
1. User will visit Profile page	2. System provides button to post an add
3. User click the button to post an ad	4. System loads Publisg ads page, where ads image can be uploaded
5. User uploads ads image, and duration saying how long this ad should be visible, then click upload button	6. System acknowledges the upload of new ad into system.

8. PostConditions: System acknowledges the published ads, and makes the ads visible to all other users.
9. UI Sketches:



2. Database design: Description of data entities and relationships, entity-relationship diagram, sample data.
   a. We used online free NoSQL database services: Redis Labs and MongoDB Atlas.
      i. Below is the list of data entities stored in Redis:
         1. USER
         2. CREDIT_CARD
         3. TRANSACTION
         4. FOLLOWS
      ii. Below is the list of data entities stored in MongoDB:
         1. CONTENT
         2. Advertisements
   b. Description of data entities and relationships:
      i. Data Entities:
         1. USER:
            a. Attributes:
               i. ID: it's a primary key, that represents each user uniquely.

ii. USER_NAME: username used by the user to log in, it is different from the actual name of the user.
iii. PASSWORD: user's password, used to authenticate user login.
iv. ADDRESS: user address, captured as part of contact details.
v. DATE_OF_BIRTH: user's date of birth, captured as part of the age-related constraint.
vi. EMAIL_ID: user's email ID, captured as part of contact details.
vii. USER_TYPE: it stores the type of user. Users may be normal users, premium users, or brands.
viii. FULL_NAME: it is the full name of each user.

2. CREDIT_CARD:
   a. Attributes:
      i. ID: the primary key, used to uniquely identify each credit card entry.
      ii. USER_ID: is the USER_ID associated with credit card details. It's a foreign key referencing the USER table.
      iii. CARD_NUMBER: credit card number, as mentioned on the credit card.
      iv. CVV: security code mentioned on the back of credit card.
      v. EXPIRY_MONTH: expiry month as mentioned on the credit card.
      vi. EXPIRY_YEAR: year of expiry of credit card, as mentioned on it.

3. TRANSACTION:
   a. Attributes:
      i. ID: the primary key, is used to uniquely identify each transaction.
      ii. USER_ID: ID of the user, who carries out the transaction. It's a foreign key referencing the USER table.
      iii. CREDIT_CARD_ID: is a credit card ID associated with the credit card used to execute the transaction. It's a foreign key reference to the CREDIT_CARD table.
      iv. AMOUNT: amount paid as part of the transaction
      v. BILLING_ADDRESS: billing address of the transaction.
      vi. TRANSACTION_DATE: is the date and time, to capture the transaction time.
      vii. RECEIPT: The receipt will be stored here.

4. FOLLOWS:
   a. Attributes:
      i. USER_ID: if user A follows B, then it is A's user ID. It's part of the primary key. It is also a foreign key reference to the USER table.

ii. FOLLOWER_USER_ID: if user A follows B, then it is B's user ID. It's part of the primary key. It is also a foreign key reference to the USER table.

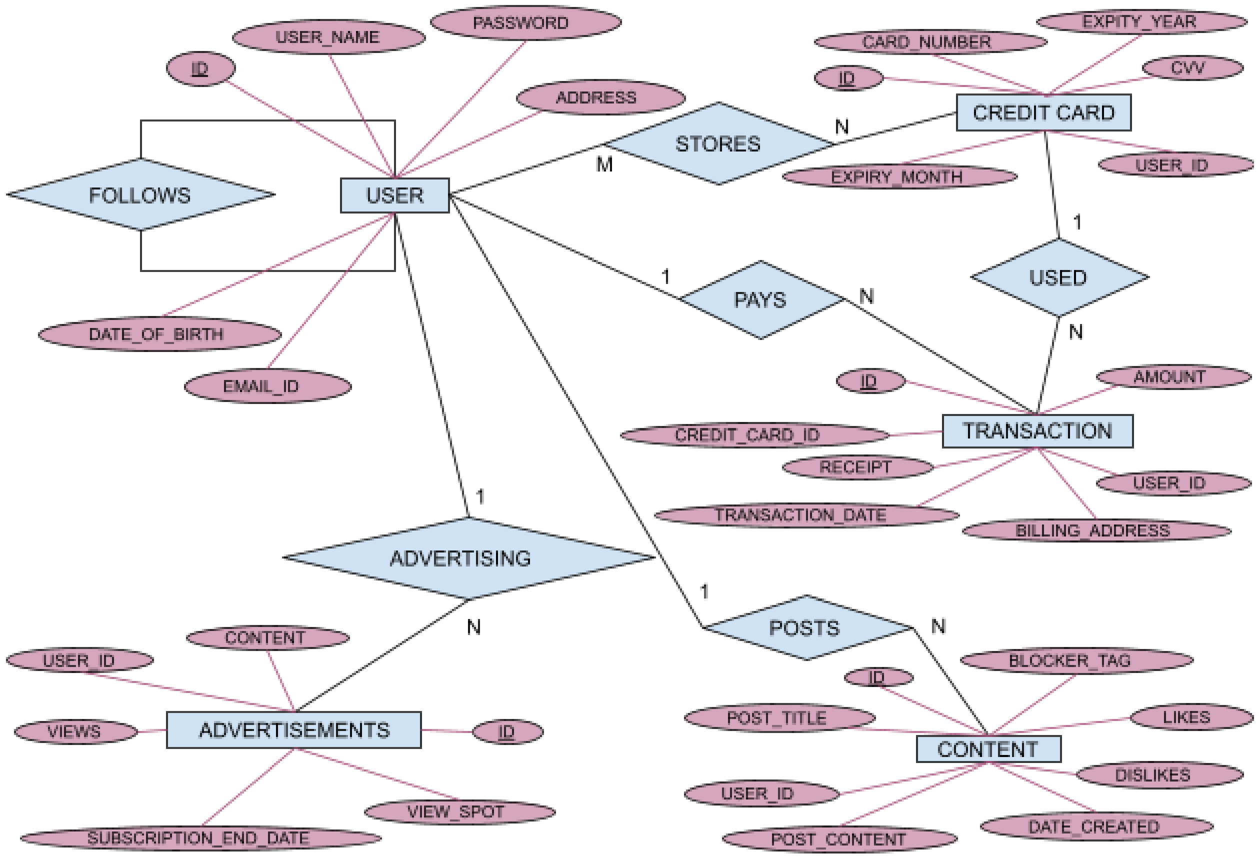
5. <u>CONTENT</u>:
   a. Attributes:
      i. ID: the primary key, is used to uniquely identify each post's content.
      ii. USER_ID: the id of the user, who posted the content. It is a foreign key reference to the USER table.
      iii. POST_CONTENT: it is used to store the contents of the post
      iv. POST_TITLE: it's the subject line of the post.
      v. BLOCKER_TAG: is the tag, which says whether a particular post is blocked by the moderator or not.
      vi. DATE_CREATED: post creation time is saved here.
      vii. LIKES: counts how many likes each post obtained.
      viii. DISLIKES: counts how many dislikes each post has.

6. <u>Advertisements</u>:
   a. Attributes:
      i. ID: primary key, used to uniquely identify each advertisement.
      ii. USER_ID: its foreign key reference to USER table. This stores the USER_ID of the user who created this advertisement content.
      iii. CONTENT: it stores the content of advertisements.
      iv. VIEWS: number of views each advertisement has
      v. VIEW_SPOT: it is a spot information, i.e. spot in the UI, where exactly the advertisement is shown.
      vi. SUBSCRIPTION_END_DATE: The user needs to pay for each advertisement to view it on our website for a certain period of time. This is the end date of this advertisement subscription.

ii. Relationships:
1. One user can create many content items.
2. One content item can be created by only one user.
3. One user can own many advertisements.
4. One advertisement can be owned by only one user.
5. One user can have multiple credit cards.
6. One credit card can be owned by multiple users.
7. One transaction can be associated with one credit card.
8. One credit card can be used in multiple transactions.
9. One user can have multiple transactions.

10. One transaction is associated with only one user.

c. Entity-Relationship Diagram:



d. SQL code to design database:

i. Creating USER table:

```
CREATE TABLE "USER" (
     "ID"  INTEGER,
     "USER_NAME"      TEXT,
     "PASSWORD" TEXT,
      "FULL_NAME"      TEXT,
      "ADDRESS"   TEXT,
      "EMAIL_ID" TEXT,
      "USER_TYPE"      TEXT,
      "DATE_OF_BIRTH"  TEXT,
      PRIMARY KEY("ID")
)
```

ii. Creating TRANSACTION table:

```
CREATE TABLE "TRANSACTION" (
      "ID"  INTEGER,
      "USER_ID"   INTEGER,
      "CREDIT_CARD_ID" INTEGER,
      "AMOUNT"    REAL,
      "BILLING_ADDRESS"     TEXT,
      "TRANSACTION_DATE"    TEXT,
      "RECEIPT"   TEXT,
      PRIMARY KEY("ID"),
      FOREIGN KEY("USER_ID") REFERENCES "USER"("ID"),
      FOREIGN     KEY("CREDIT_CARD_ID")     REFERENCES
"CREDIT_CARD"("ID")
)
```

iii. Creating FOLLOWS table:

```
CREATE TABLE "FOLLOWS" (
      "USER_ID"   INTEGER,
      "FOLLOWER_USER_ID"    INTEGER,
      FOREIGN KEY("USER_ID") REFERENCES "USER"("ID"),
      FOREIGN    KEY("FOLLOWER_USER_ID")    REFERENCES
"USER"("ID"),
      PRIMARY KEY("USER_ID","FOLLOWER_USER_ID")
)
```

iv. Creating CREDIT_CARD table:

```
CREATE TABLE "CREDIT_CARD" (
      "ID"  INTEGER,
      "USER_ID"  INTEGER,
      "CARD_NUMBER"    TEXT,
      "CVV" INTEGER,
      "EXPIRY_MONTH"   INTEGER,
      "EXPIRY_YEAR"    INTEGER,
      FOREIGN KEY("USER_ID") REFERENCES "USER"("ID"),
      PRIMARY KEY("ID")
)
```

v. Creating CONTENT table:

```
CREATE TABLE "CONTENT" (
      "ID"  INTEGER,
      "USER_ID"  INTEGER,
      "POST_CONTENT"   BLOB,
      "POST_TITLE"     TEXT,
      "BLOCKER_TAG"    TEXT,
      "DATE_CREATED"   TEXT,
      "LIKES"    INTEGER,
      "DISLIKES" INTEGER,
      FOREIGN KEY("USER_ID") REFERENCES "USER"("ID"),
      PRIMARY KEY("ID")
)
```

vi. Creating ADVERTISEMENTS table:

```
CREATE TABLE "ADVERTISEMENTS" (
      "ID"  INTEGER,
      "USER_ID"  INTEGER,
      "CONTENT"  BLOB,
      "VIEWS"    INTEGER,
      "VIEW_SPOT"      INTEGER,
      "SUBSCRIPTION_END_DATE"     TEXT,
      PRIMARY KEY("ID"),
      FOREIGN KEY("USER_ID") REFERENCES "USER"("ID")
)
```

e. Sample data:

i. Inserting data into ADVERTISEMENTS table:

```
INSERT INTO ADVERTISEMENTS (ID, USER_ID, CONTENT, VIEWS,
VIEW_SPOT, SUBSCRIPTION_END_DATE)
VALUES (1, 10, 'my_ad_content', 100, 1, '2023-10-31');

INSERT INTO ADVERTISEMENTS (ID, USER_ID, CONTENT, VIEWS,
VIEW_SPOT, SUBSCRIPTION_END_DATE)
VALUES (2, 20, 'another_ad_content', 200, 2, '2023-11-30');
```

ii. Inserting data into CONTENT table:

```
INSERT INTO CONTENT (ID, USER_ID, POST_CONTENT,
POST_TITLE, BLOCKER_TAG, DATE_CREATED, LIKES, DISLIKES)
VALUES (1, 10, 'my_post_content', 'My Post Title',
'block', '2023-09-29', 10, 5);

INSERT INTO CONTENT (ID, USER_ID, POST_CONTENT,
POST_TITLE, BLOCKER_TAG, DATE_CREATED, LIKES, DISLIKES)
VALUES (2, 20, 'another_post_content', 'Another Post
Title', '', '2023-09-29', 20, 10);
```

iii. Inserting data into CREDIT_CARD table

```
INSERT INTO CREDIT_CARD (ID, USER_ID, CARD_NUMBER, CVV,
EXPIRY_MONTH, EXPIRY_YEAR)
VALUES (1, 10, '1234-5678-9012-3456', 123, 10, 2024);

INSERT INTO CREDIT_CARD (ID, USER_ID, CARD_NUMBER, CVV,
EXPIRY_MONTH, EXPIRY_YEAR)
VALUES (2, 20, '7890-1234-5678-9012', 456, 11, 2025);
```

iv. Inserting data into FOLLOWS table

```
INSERT INTO FOLLOWS (USER_ID, FOLLOWER_USER_ID)
VALUES (10, 20);

INSERT INTO FOLLOWS (USER_ID, FOLLOWER_USER_ID)
```

```
VALUES (20, 10);
```

v. Inserting data into TRANSACTION table

```
INSERT INTO TRANSACTION (ID, USER_ID, CREDIT_CARD_ID,
AMOUNT, BILLING_ADDRESS, TRANSACTION_DATE, RECEIPT)
VALUES (1, 10, 1, 100, '123 Main Street', '2023-09-29',
'my_receipt');

INSERT INTO TRANSACTION (ID, USER_ID, CREDIT_CARD_ID,
AMOUNT, BILLING_ADDRESS, TRANSACTION_DATE, RECEIPT)
VALUES (2, 20, 2, 200, '456 Elm Street', '2023-09-29',
'another_receipt');
```

vi. Inserting data into USER table

```
INSERT INTO USER (ID, USER_NAME, PASSWORD, FULL_NAME,
ADDRESS, EMAIL_ID, USER_TYPE, DATE_OF_BIRTH)
VALUES (10, 'user10', 'password10', 'John Doe', '123
Main Street', 'john.doe@example.com', 'general',
'1980-01-01');

INSERT INTO USER (ID, USER_NAME, PASSWORD, FULL_NAME,
ADDRESS, EMAIL_ID, USER_TYPE, DATE_OF_BIRTH)
VALUES (20, 'user20', 'password20', 'Jane Doe', '456 Elm
Street', 'jane.doe@example.com', 'general',
'1981-02-02');
```

3. Architectural design: Description of client and server components; communication protocol (including data format), overall diagram of the system architecture.
   a. Server components:
      i. We used Tomcat servlets to handle the incoming requests.
      ii. Each servlet is designed to serve different operations like login, signin, signup, transaction, post_handler, etc.
      iii. We store user information, transaction details, user posts,

advertisements, and other details in the database.

iv. We have designed a database facade to access Redis and MongoDB database services.

v. Our server code runs over the Tomcat:11 docker image.

vi. Components:

Class Name	Function
LoginServlet.java	Serves "/login" path to load login page to web client
SigninServlet.java	Serves "/signin" path to sign in the user
SignupServlet.java	Serves "/signup" path to sign up new user
CreatePostServlet.java	Serves "/post" path to store new posts
PostServlet.java	Serves "/getpost" path to read stored posts from database
UserPostServlet.java	Serves "/getuserpost" posts exclusive to a specific user
UserPostAdsServlet.java	Serves "/postads" path to store ads in the system
UserTypeServlet.java	Serves “/usertype” path to get the user type
ProfileServlet.java	Serves “/profile” path to update user details
TransactionServlet.java	Serves “/transaction” path to handle payment and update of user type
DataBaseService.java	Service class to connect to redis and mongodb database services
AuthenticationService.java	Service class used to authenticate and authorize user data
Advertisement.java	Entity class to represent

	advertisement detail
CreditCard.java	Entity class to represent credit card information
Post.java	Entity class to represent posts
Transaction.java	Entity class to represent Transactions
User.java	Entity class to represent user details
UserType.java	Enum to represent user types

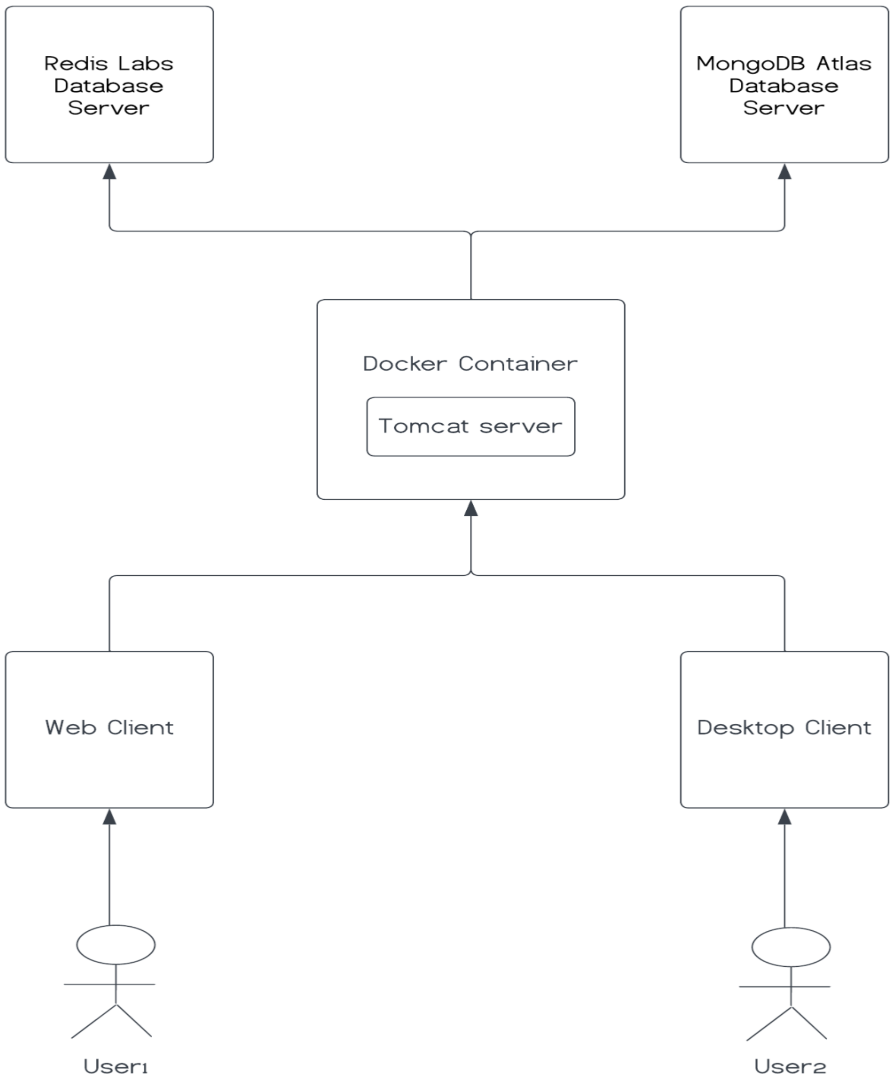
b. Client components:
   i. Desktop Client:
      1. We have created java gui pages for each of the use cases.
      2. We have used Java Swing to design the UI pages
      3. The Swing buttons will trigger the RESTful API calls to the server.
         a. Components:

LoginService.java	Class which handles user sign in and sign up and makes RESTful API calls to server
PostService.java	Class which loads all posts and handles user post creation, and makes RESTful API calls to server
PricingService.java	Class which provides user with subscription choices
ProfileUpdateService.java	Class which allows user to update their details, and makes RESTful API calls to server
TransactionService.java	Class which allows user to complete payment after selecting subscription type, and makes RESTful API calls to server
Application.java	The main runner class

ii. Web Client:
   1. We have created web pages for each of the use cases.
   2. We used HTML, CSS, and JavaScript to design the User Interface/web pages.
   3. Javascript files will make the RESTful API calls to the server on a particular action on UI, for example, a /post request is sent to the client when a user submits a new post.
   4. For each request to the server, javascript code will receive the response back, and based on the response it will change the UI in the browser.
   5. So javascript code along with the web browser will constitute the client part.

c. Communication protocol:
   i. Our architecture used RESTful endpoints between the client and server
   ii. HTTP:1 is the communication protocol used in our RESTful communications
   iii. JSON format is used to serialize the data
   iv. GET and POST requests are the HTTP methods we used to perform the operations on the resources.

d. Overall diagram of the System Architecture



4. A runnable system with GUI-based Desk or web-based client (frontend) and web-based or socket-based server, including database access if needed.
   The Social Network app code is present in the zip, along with the database.

5. Video recordings of user acceptance tests. Each recording is for a use case.

Use Case	Video Link
Login	https://youtu.be/h7kr_utfbcU
Create post	https://youtu.be/hxrsOrIIQPs
Subscribe to Premium	https://youtu.be/4e1gSYzKnNg
Profile Update	https://youtu.be/f1PwryzOKok
Post Ad	https://youtu.be/eEo2db3TLkI

6. Manual for installation and usage with information on necessary libraries, frameworks, and running guidelines.

1. Prerequisites to build and run the system:
    a. Install maven command line.
    b. Install docker for desktop.
2. Building and running Server:
    a. It's a maven project, integrated with plugins to create docker images.
    b. How to generate the docker image:
        i. “Docker for Desktop” should be running in the background.
        ii. Move to the Social-Network folder where the pom.xml file exists:
        iii. Run “mvn clean install”
        iv. It will create a docker image named: “app/social-network”
    c. How to run the code:
        i. Run the below command to run the docker image:
        ii. “docker run -p 8080:8080 app/social-network:latest”
        iii. This will start the server, listening at port 8080.
3. Running Web client:
    a. Once the server is up and running, connect to the server using any web browser and connect with the below URL
        i. “localhost:8080/social-network/login”
4. Running Desktop Client:
    a. It's a maven project, run below command to install the application inside Application folder.
        i. “mvn clean compile assembly:single”
        ii. Run the generated jar
            1. “java -cp

.\target\Application-1.0-SNAPSHOT-jar-with-dependencies jar org.example.Application”

5. Dependencies for Server: (The pom.xml file contains all the dependencies)
    a. jakarta.servlet::jakarta.servlet-api:5.0.0
    b. org.json::json:20210307
    c. commons-io::commons-io:2.11.0
    d. org.mongodb::mongodb-driver-sync:4.11.1
    e. redis.clients::jedis:5.1.0
6. Dependencies for Desktop client: (The pom.xml file contains all the dependencies)
    a. org.apache.httpcomponents::httpcore:4.4.14
    b. org.json::json:20210307