

PIXSTAR

WEBSITE

REPORT

PINTERST CLONE

By **Syed Nasiruddin**

Introduction

Within this report, I will meticulously cover the different tabs of the website, various processes involved (sign up, post creation, etc.), the webservices used, client and server side testing, HTML and CSS validation, security, privacy and legal issues.

Through a detailed analysis, I aim to convey the different functionalities offered by the website

Overview

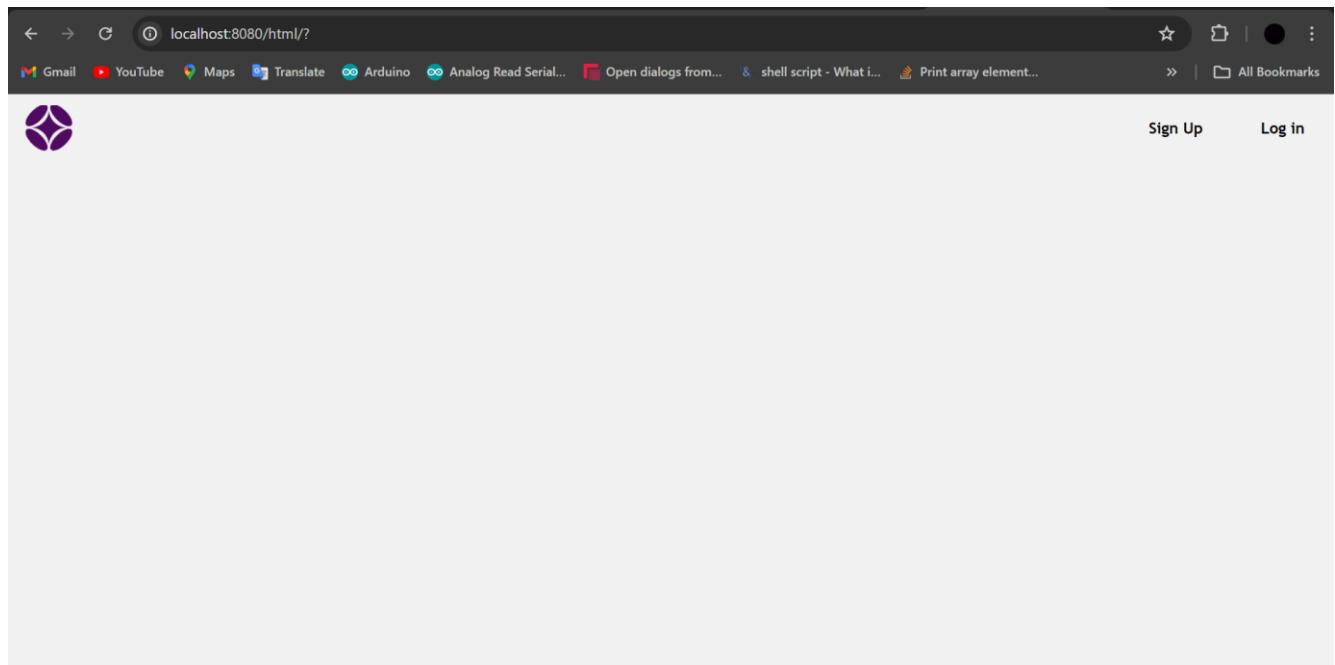
Pixstar is an image-sharing platform that offers a diverse range of functionalities designed to enhance the social networking experience.

Users can connect with friends and followers by following each other's profiles. Once connected, users can create and view posts, allowing for seamless expression of ideas and experiences. It also enables users to explore content from both followed and unfollowed users, broadening their horizons and fostering discovery of potentially new friends! Additionally, users can engage with posts by liking and commenting on each other's posts, sparking meaningful conversations and interactions between each other. The website offers powerful search capabilities as users can easily find posts based on keywords or topics of interest, ensuring they never miss out on relevant content.

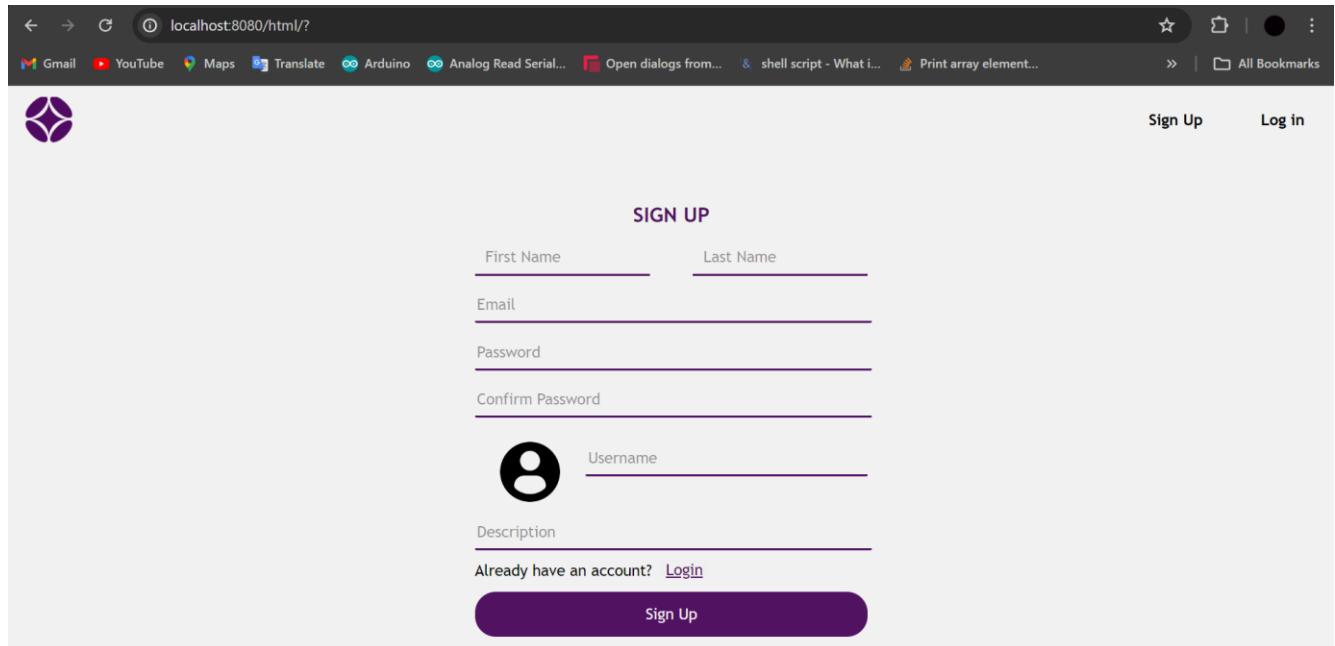
Whether its seeking inspiration, connection, or entertainment, Pixstar provides the tools and features to elevate the social networking experience.

Website Tabs

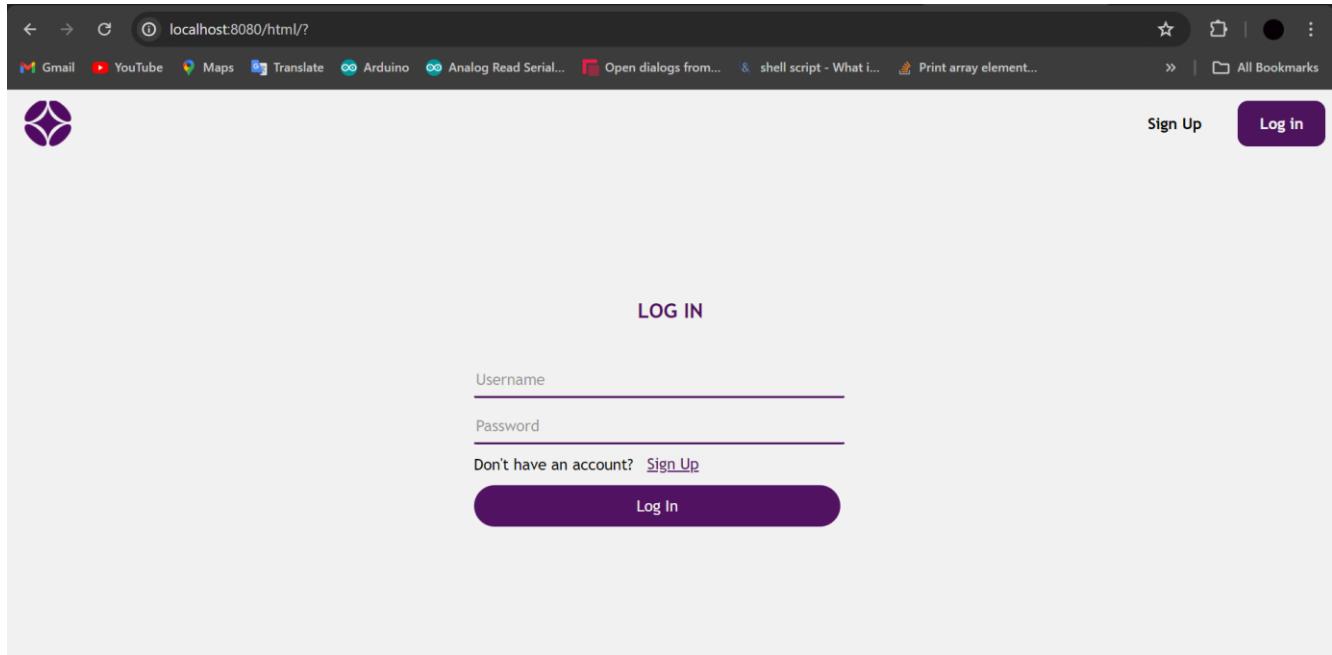
When the user first loads the website, this is what they see:



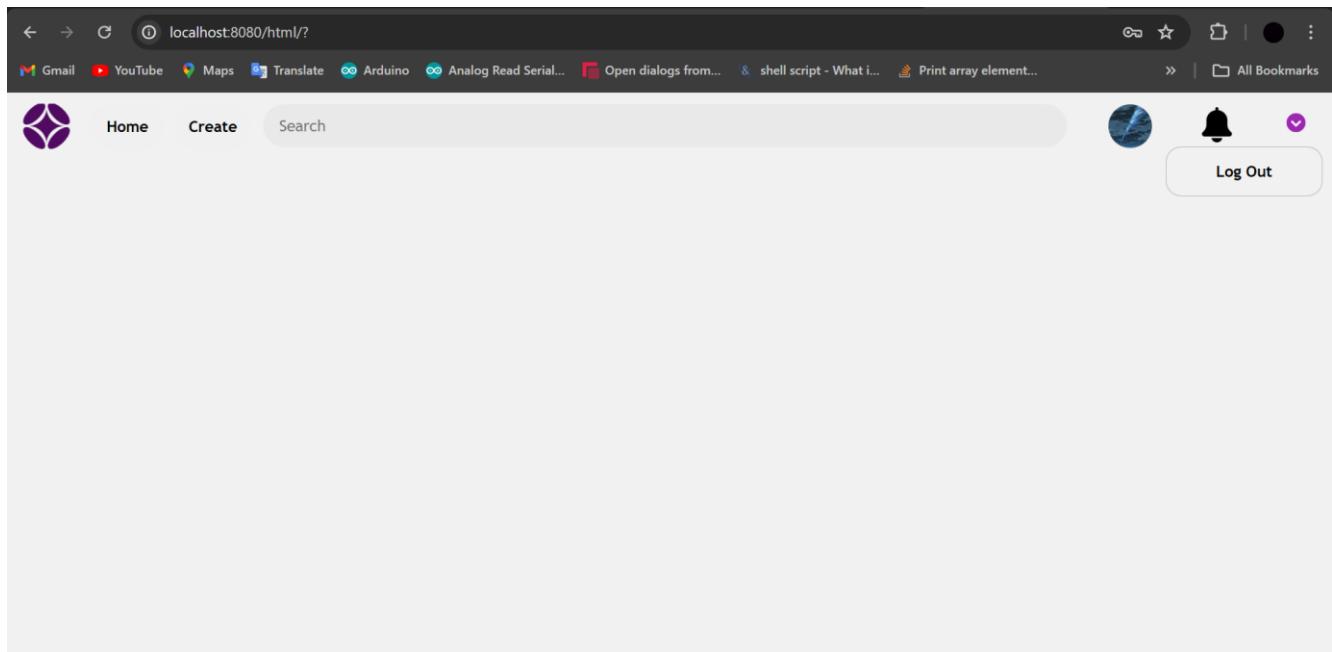
They have the option to sign up and log in, the sign up page is as seen below.



And below is the login page.



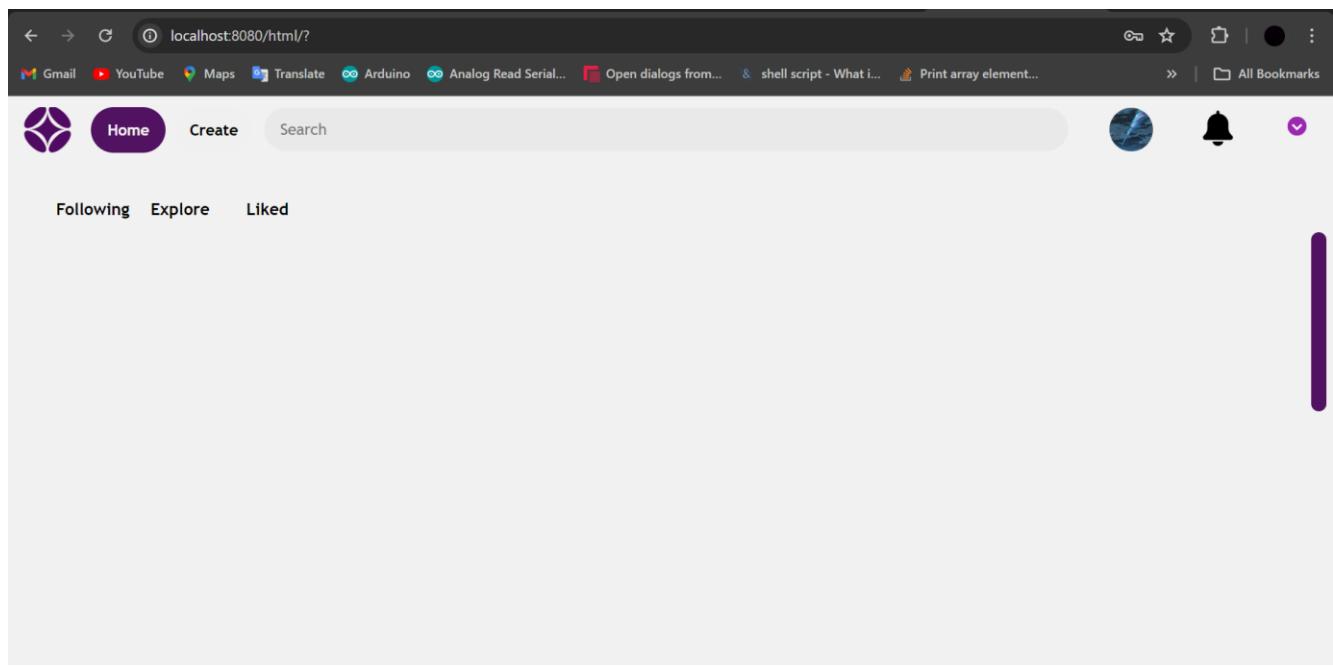
Once the user has logged in, they will be shown this screen:



The page now has a lot more options, in the top left corner are the buttons that lead the user to the home page and the post creation page, in the middle is the search bar where the user can search users in the format “@username” or they

can search posts by title by just typing in the title, on the top right is profile photo of the user which upon clicking it leads the user to the account tab, there is notification tab as well that shown when the user clicks on the bell, and a dropdown that allows the user to log out as and when they require.

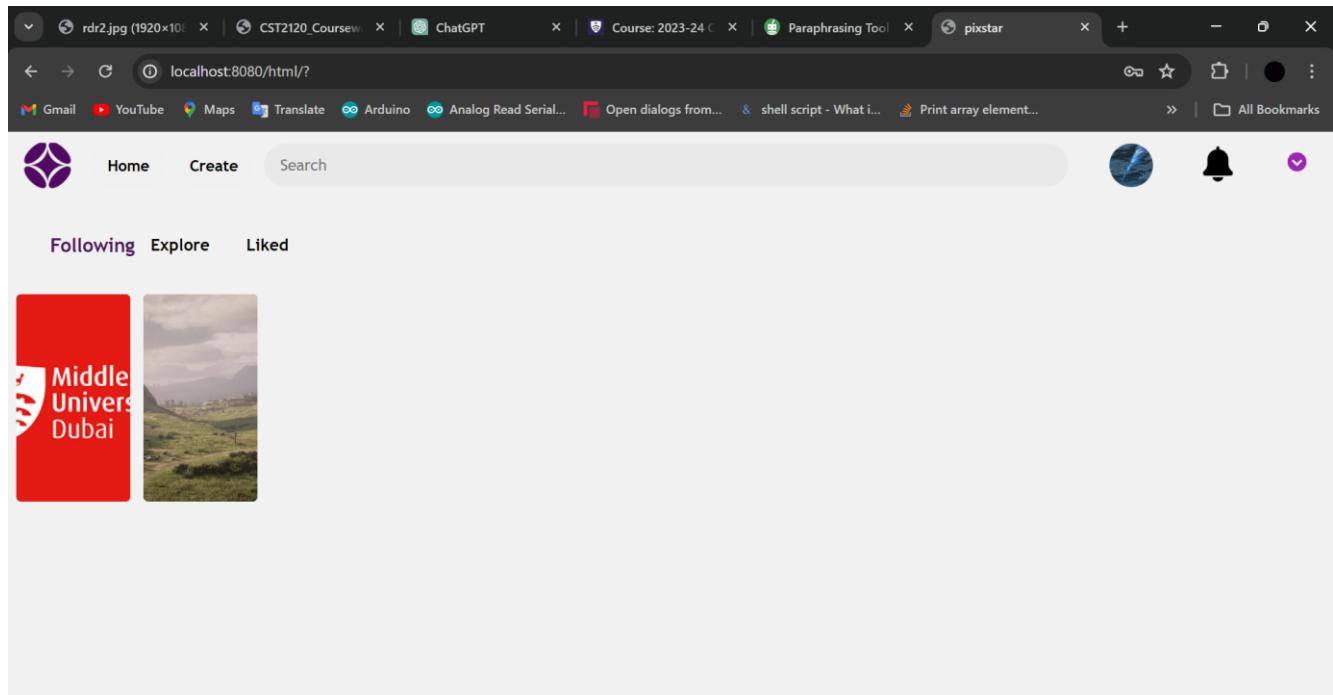
Starting with the home page, once the user clicks the “home” button, the below is shown



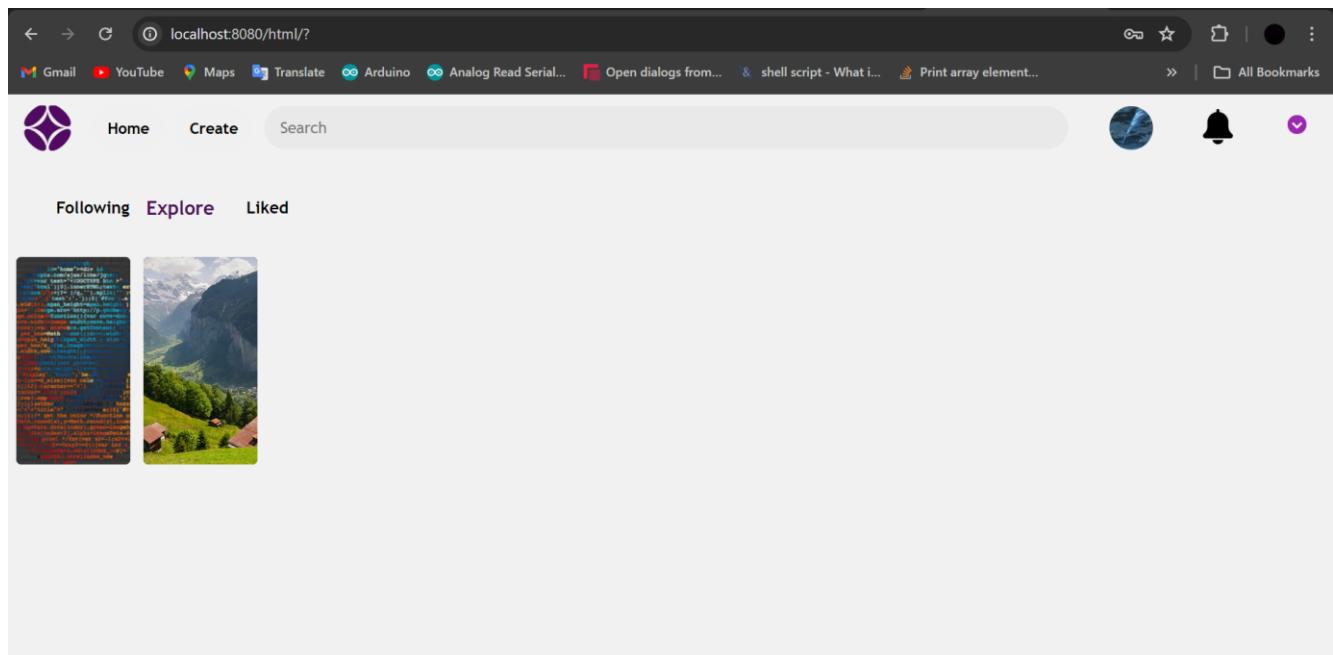
The user can choose which of the three sections they want to view, the following tab consists of all the users the current user is following, the explore tab consists of users that are not followed by the user, and the liked tab includes all the posts the user has liked.

The tabs can be seen below,

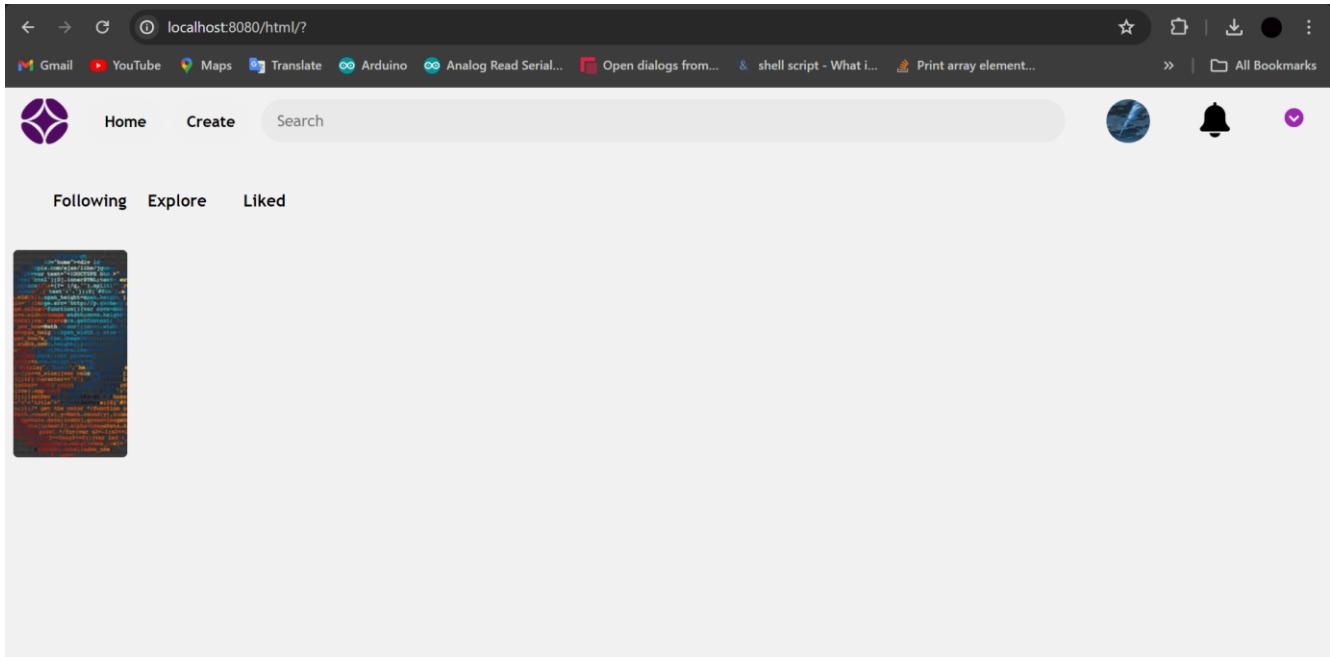
The following tab:



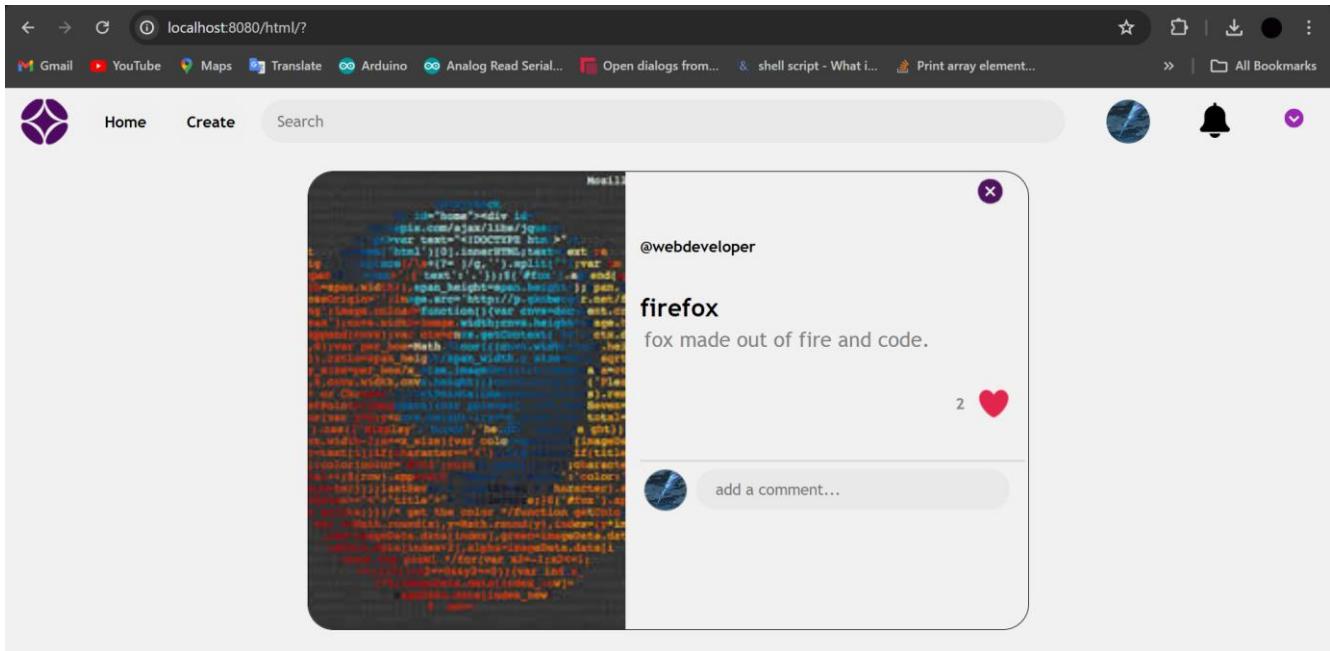
The explore tab:



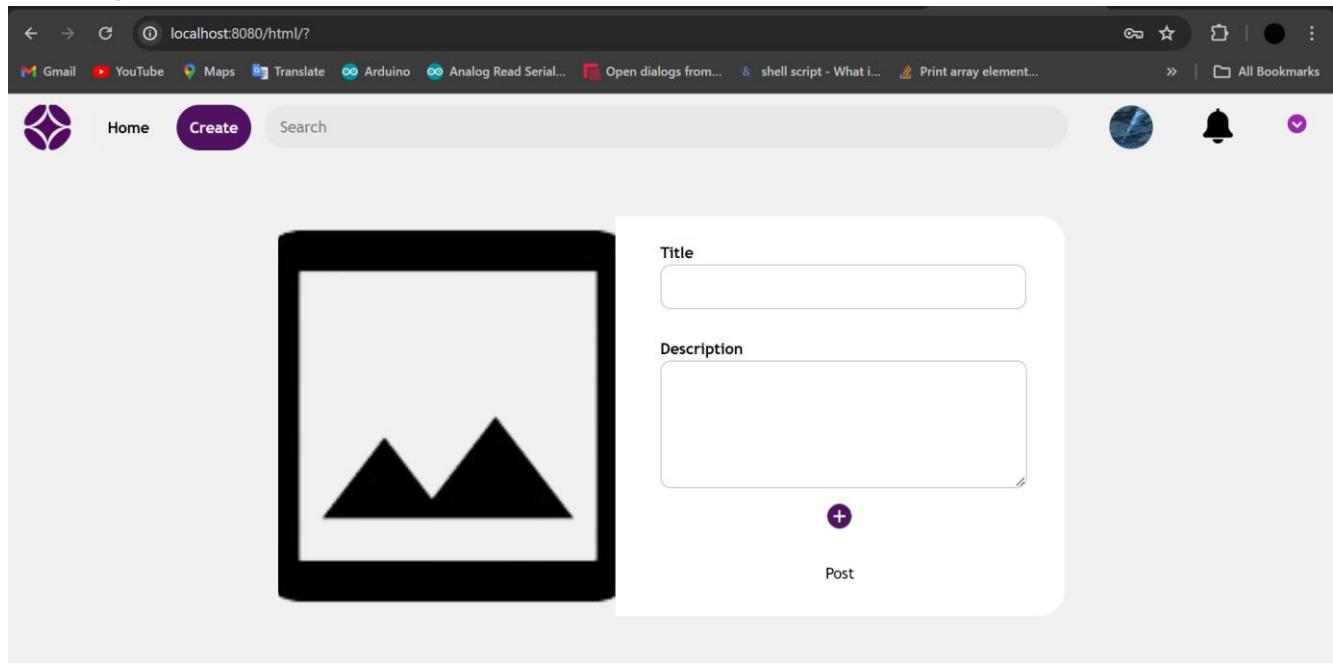
The liked tab:



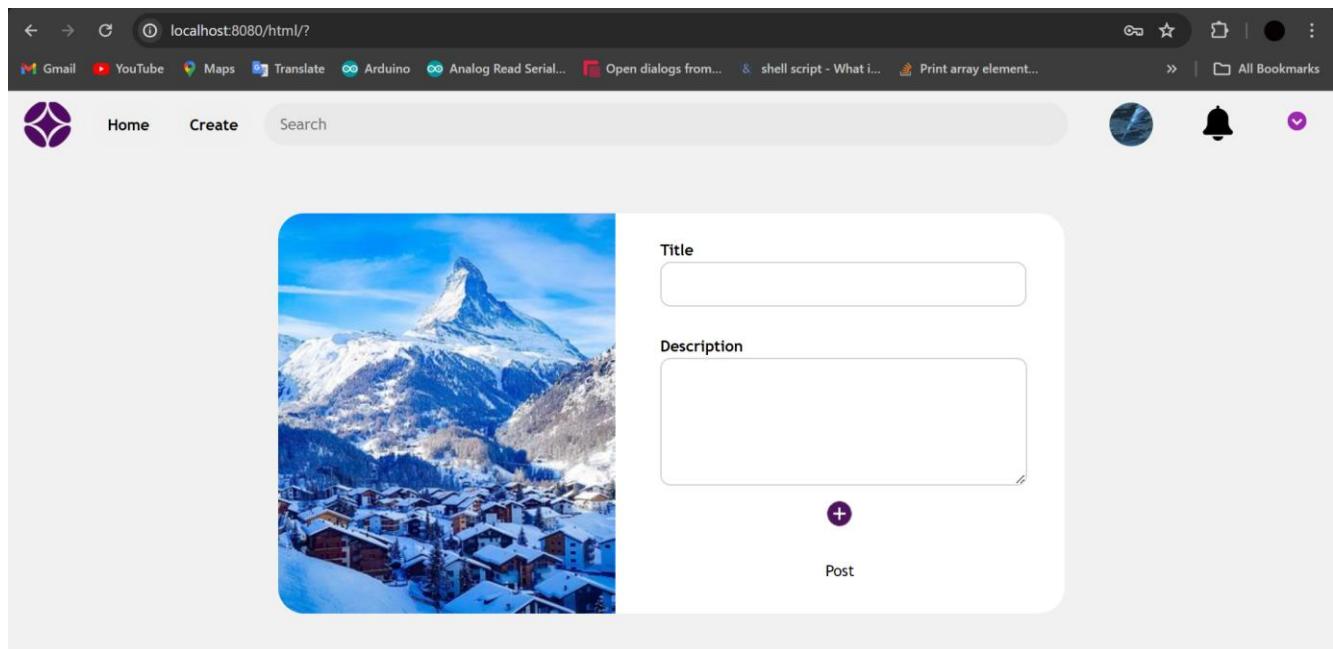
The user can click on the post to view more details, and interact with it by liking and commenting as well.



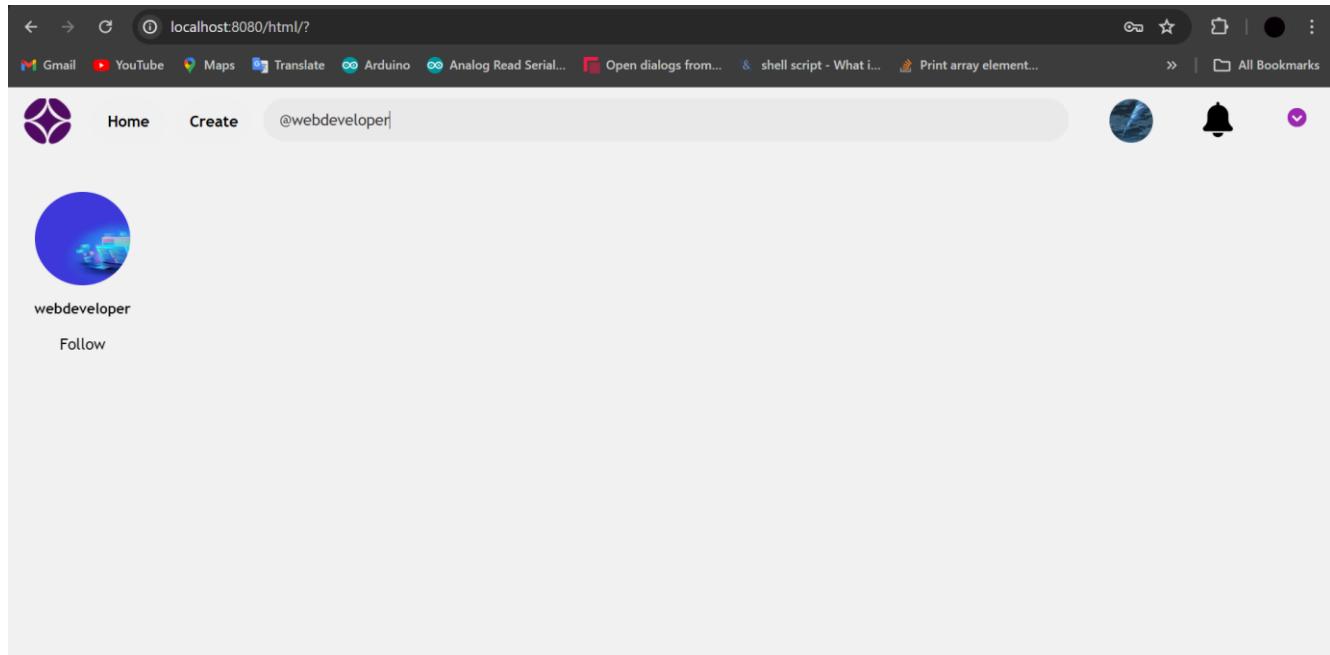
Moving on to the create tab,



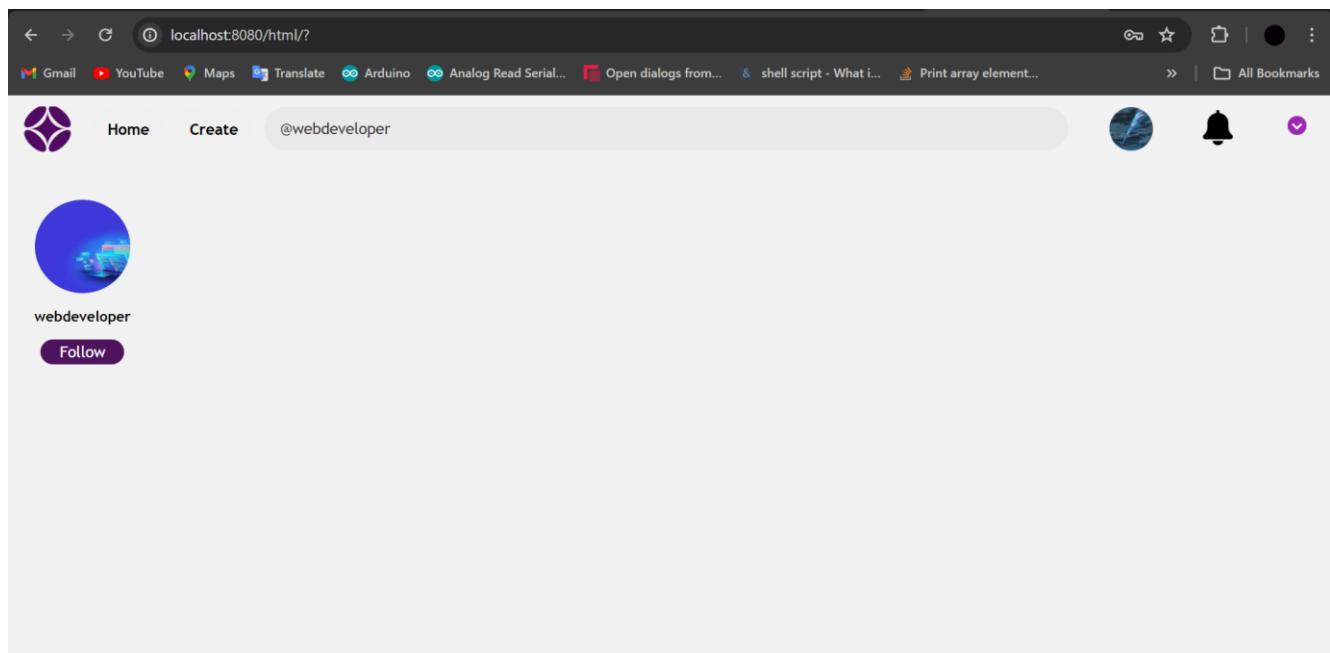
It allows the user to assign a title, description and an image (that should be less than 1 MB). The image is attached by clicking on the purple “plus” symbol, a preview is shown on the left.

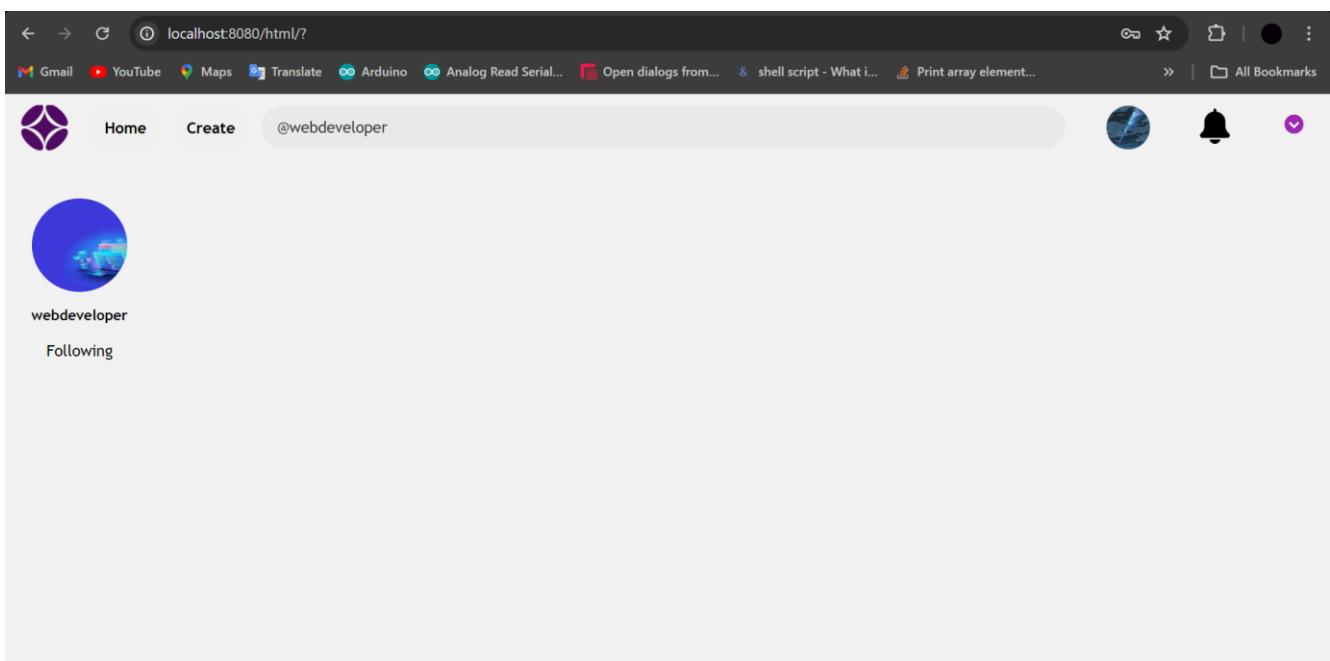


The searchbar provides both user and post searches, the below is the result of a user search,

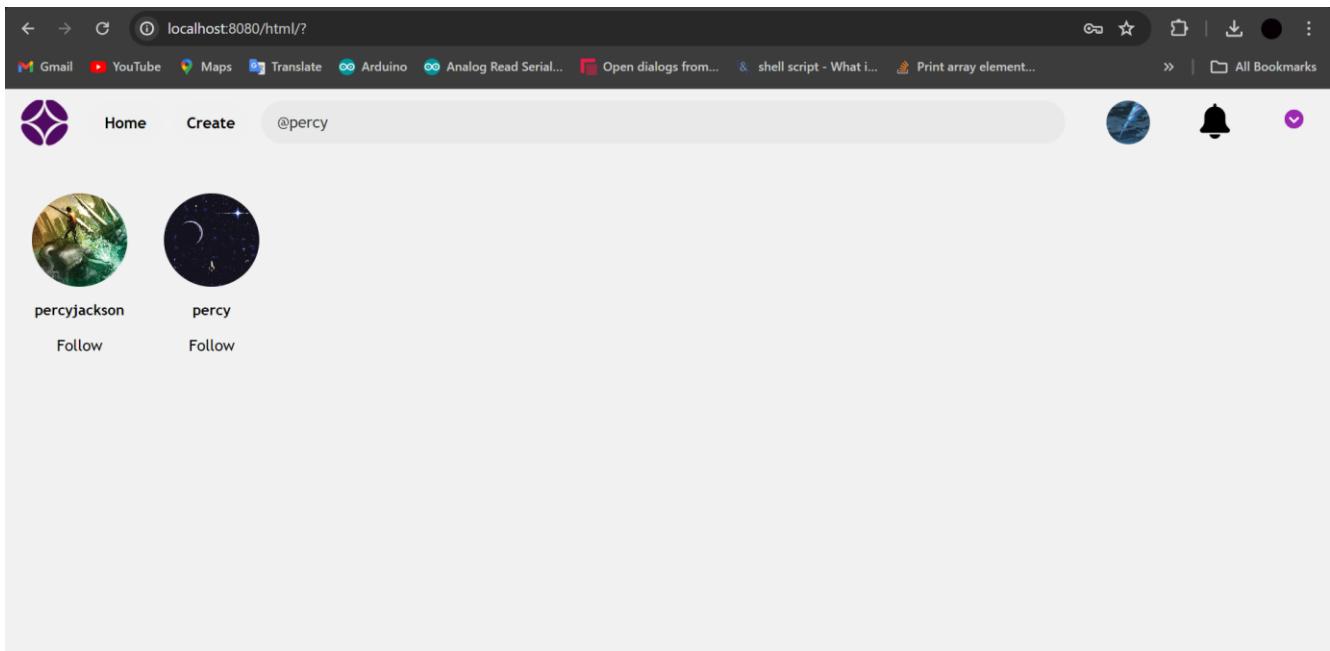


The user can follow or unfollow users after searching them,

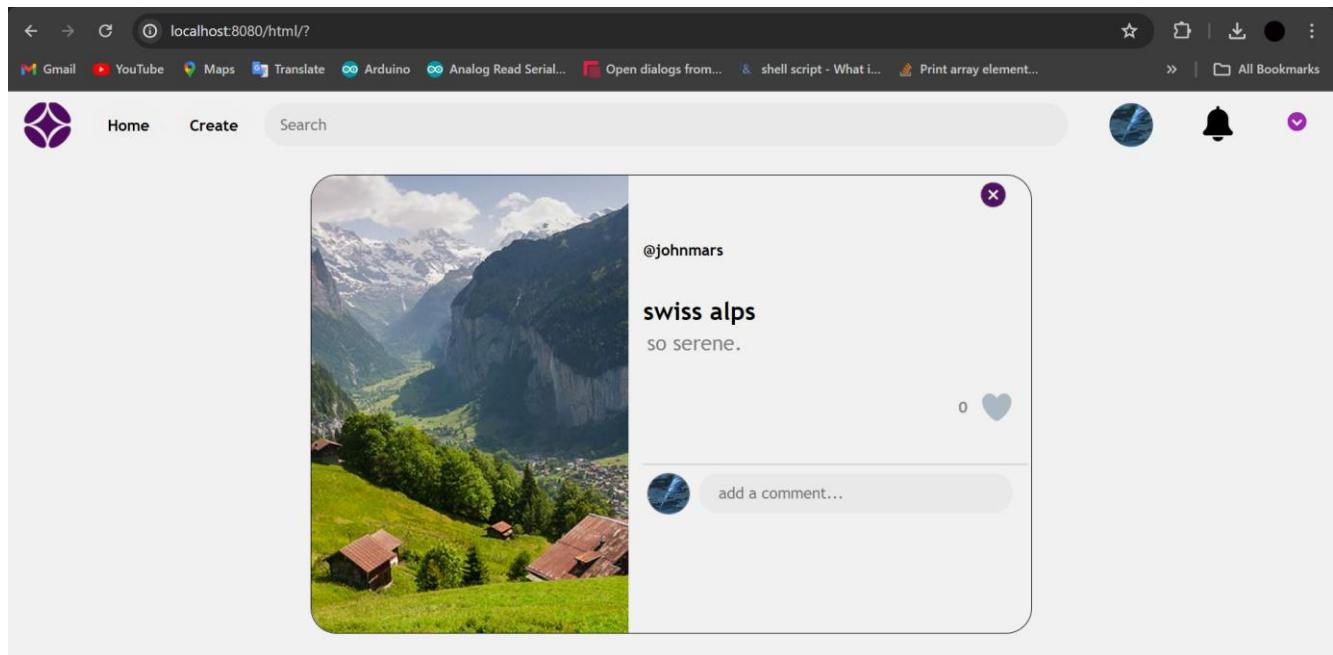
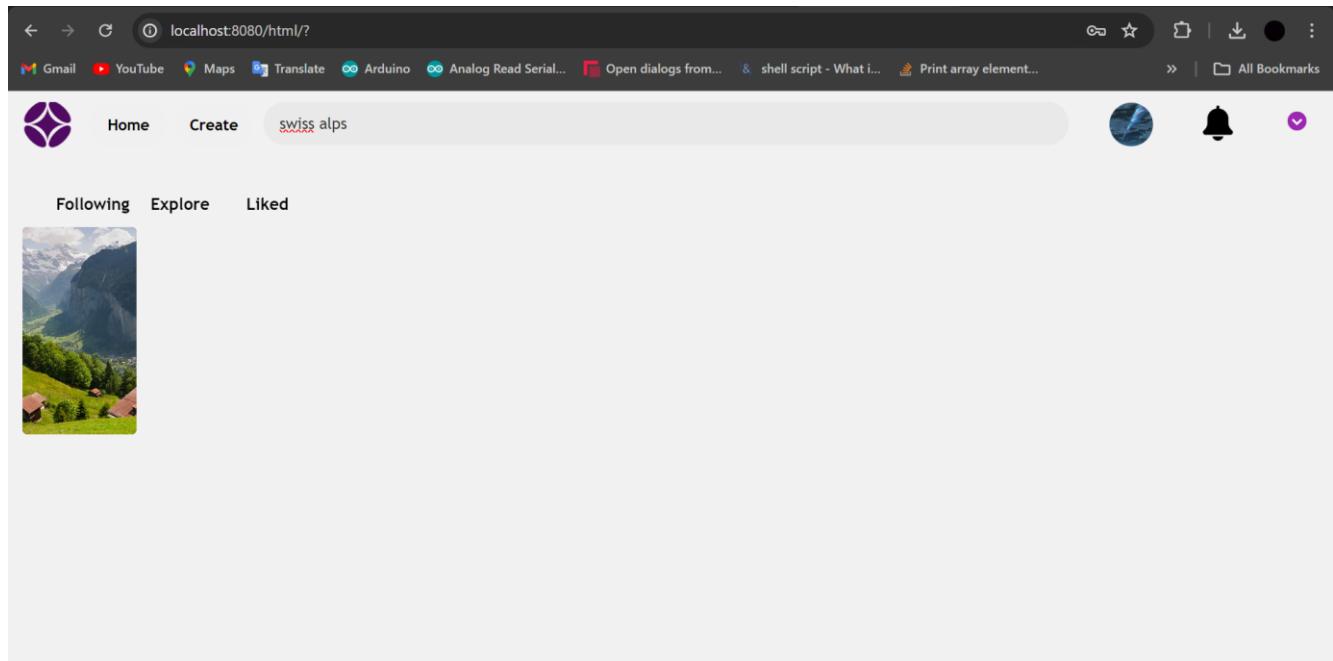




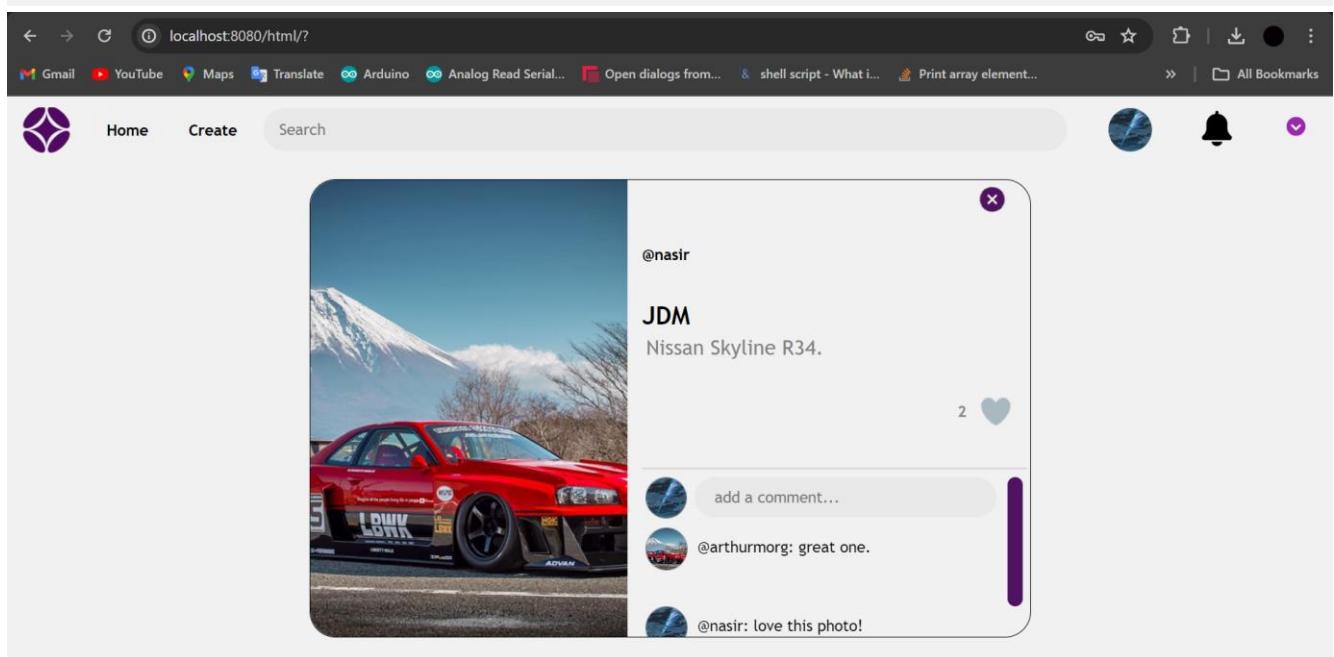
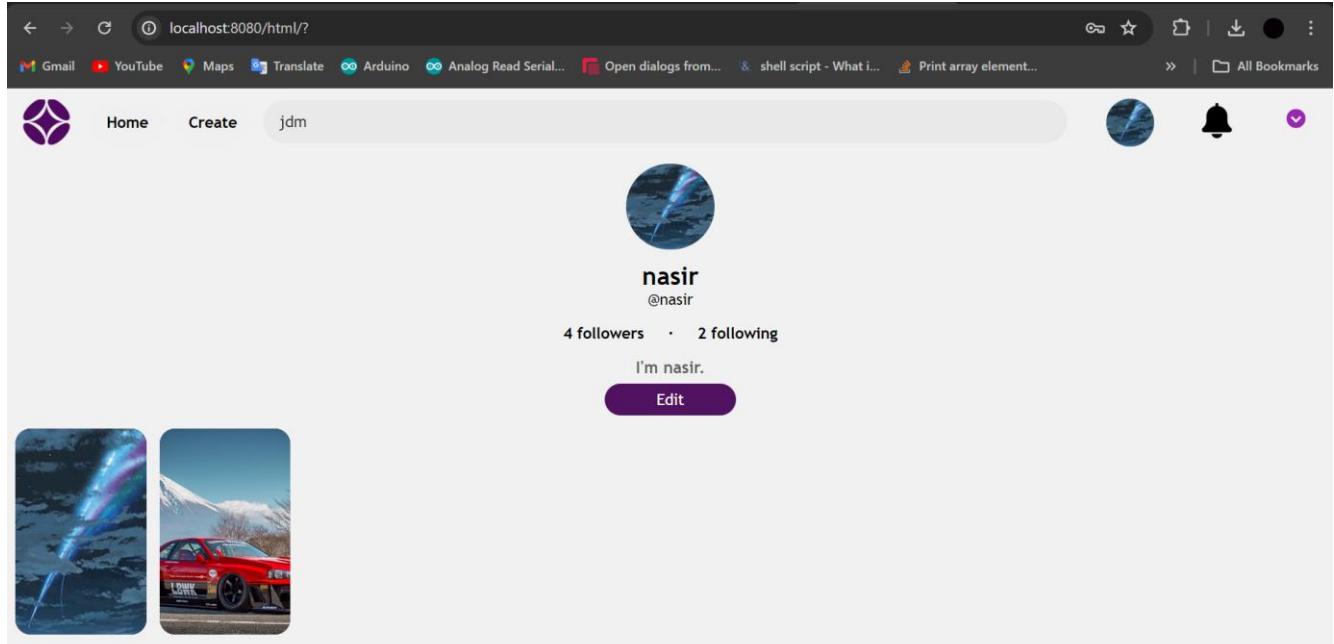
If a few users have a part of the same name, both users are shown:



posts can be searched by searching key words

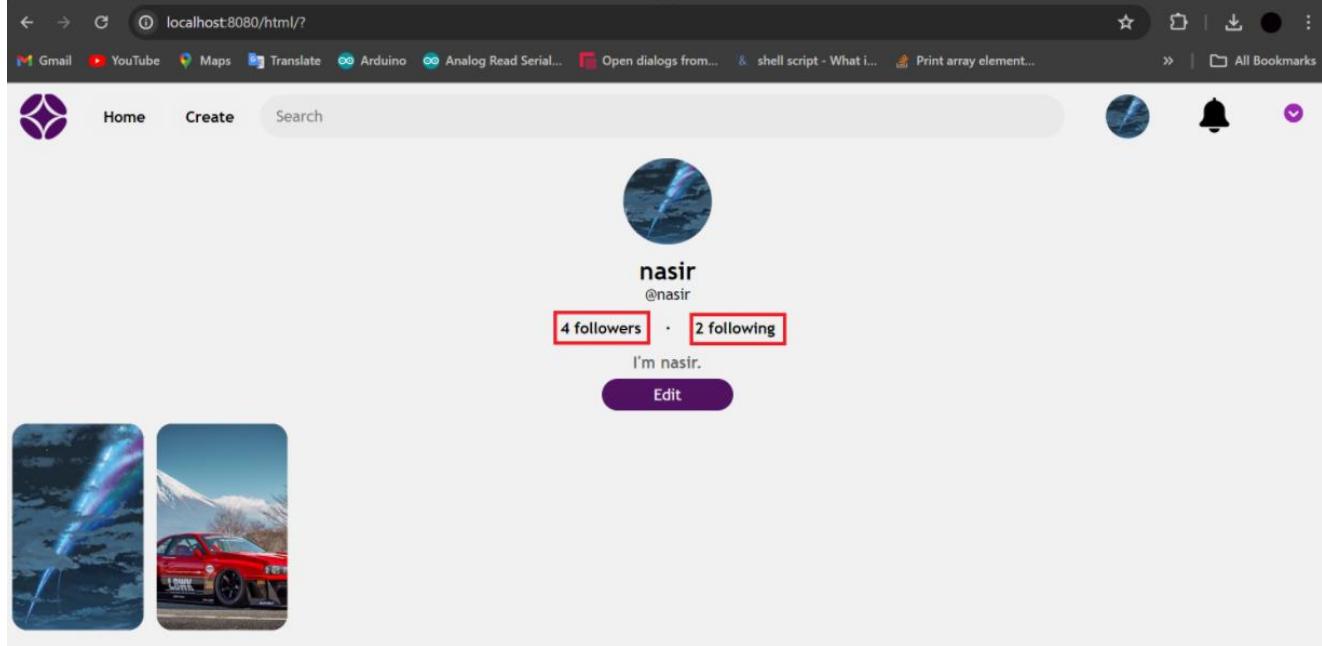


By clicking on the user's profile picture, they can view the account tab, which allows them to see their own posts, followers, and following. They are also given the option to edit their profile.

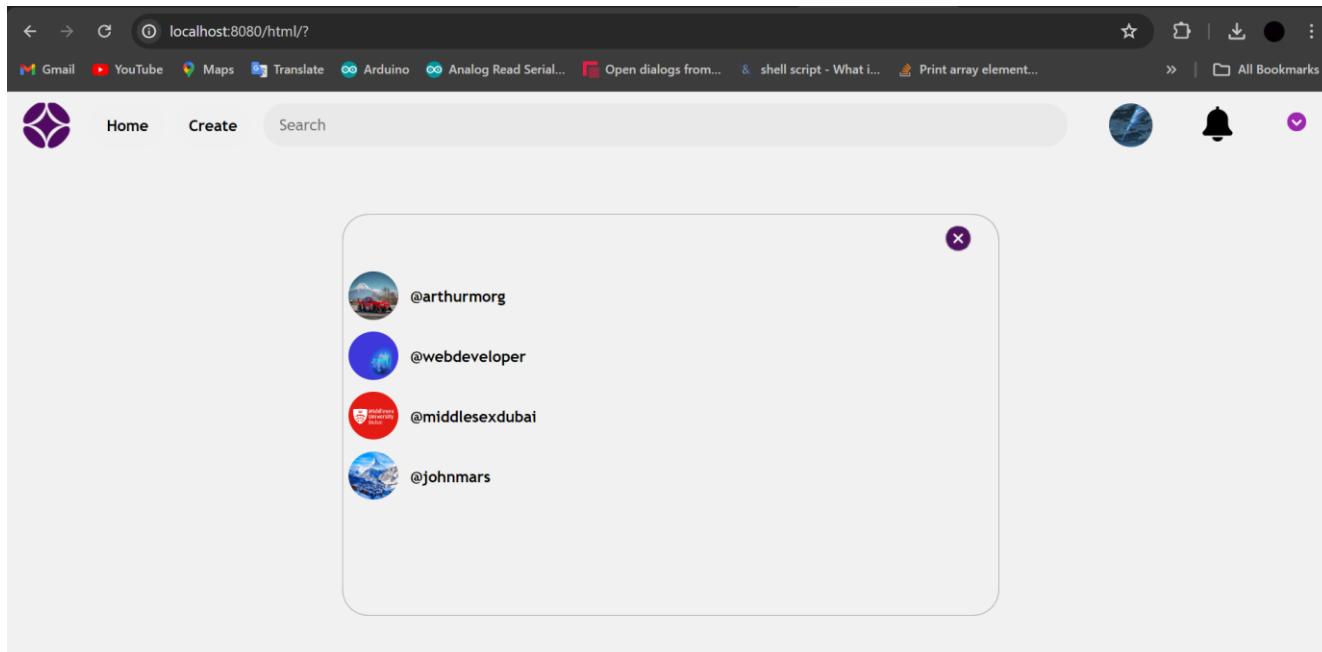


The newer comments appear at the top of the comment section.

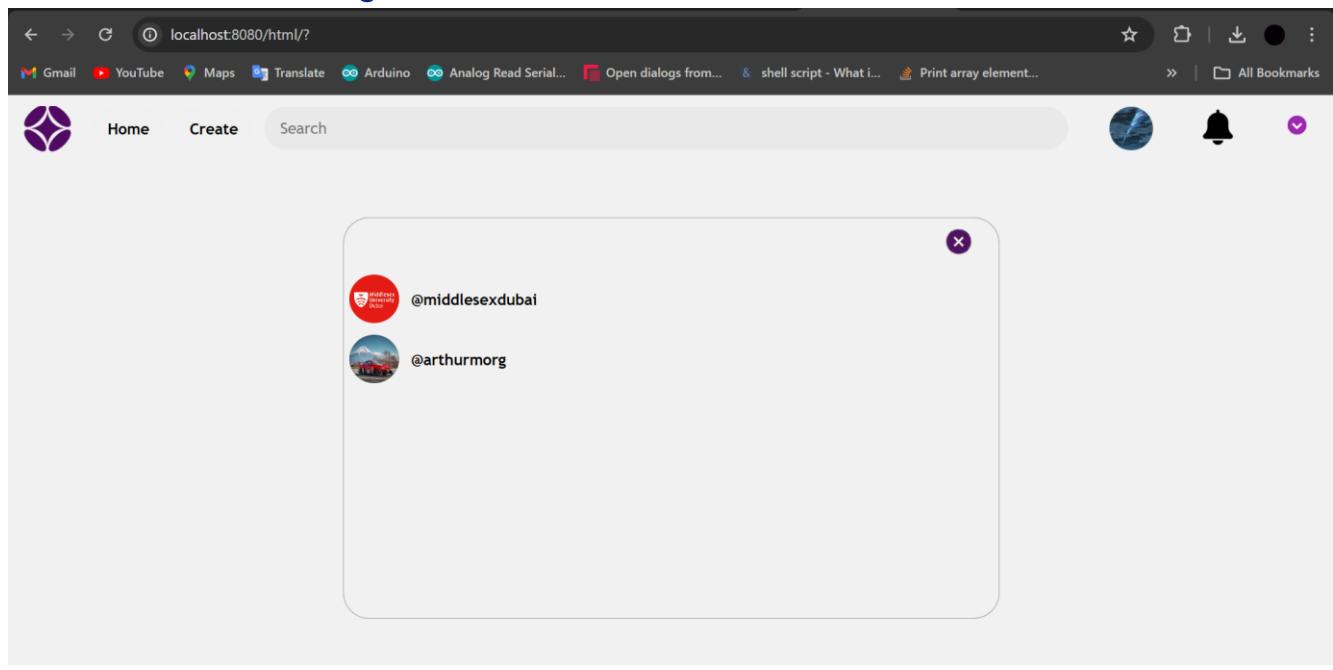
Followers can be viewed by clicking on “followers” on the account tab, similarly, following can be viewed by clicking on “following”, the buttons have been highlighted in red for informative purposes.



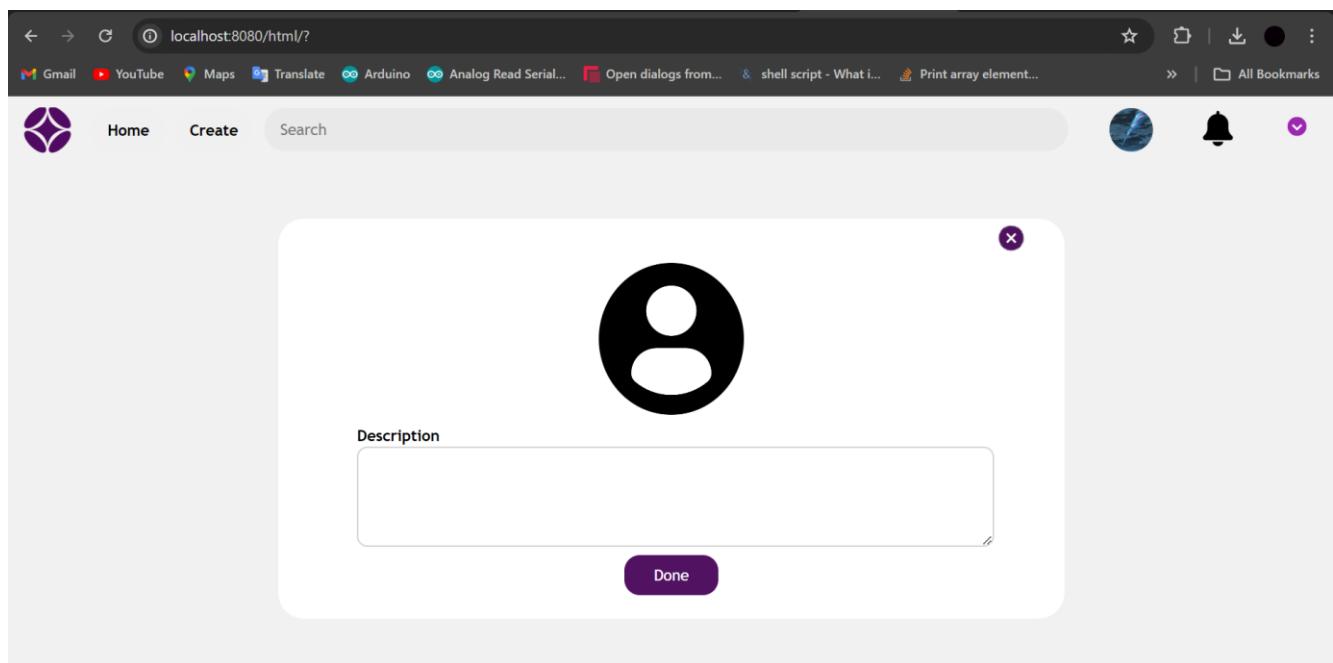
by clicking on the “followers” button, the followers username and profile pictures are shown.



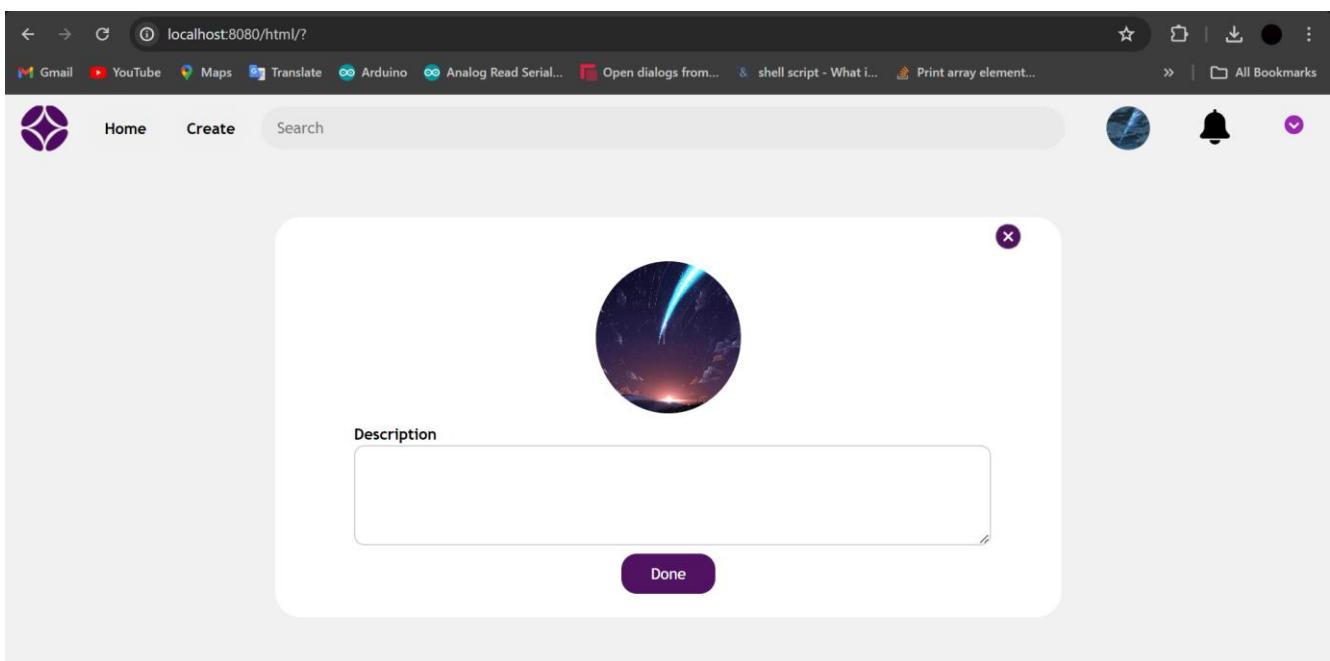
Same for the following button,



the edit profile section can be seen if the user clicks on the “edit” button,

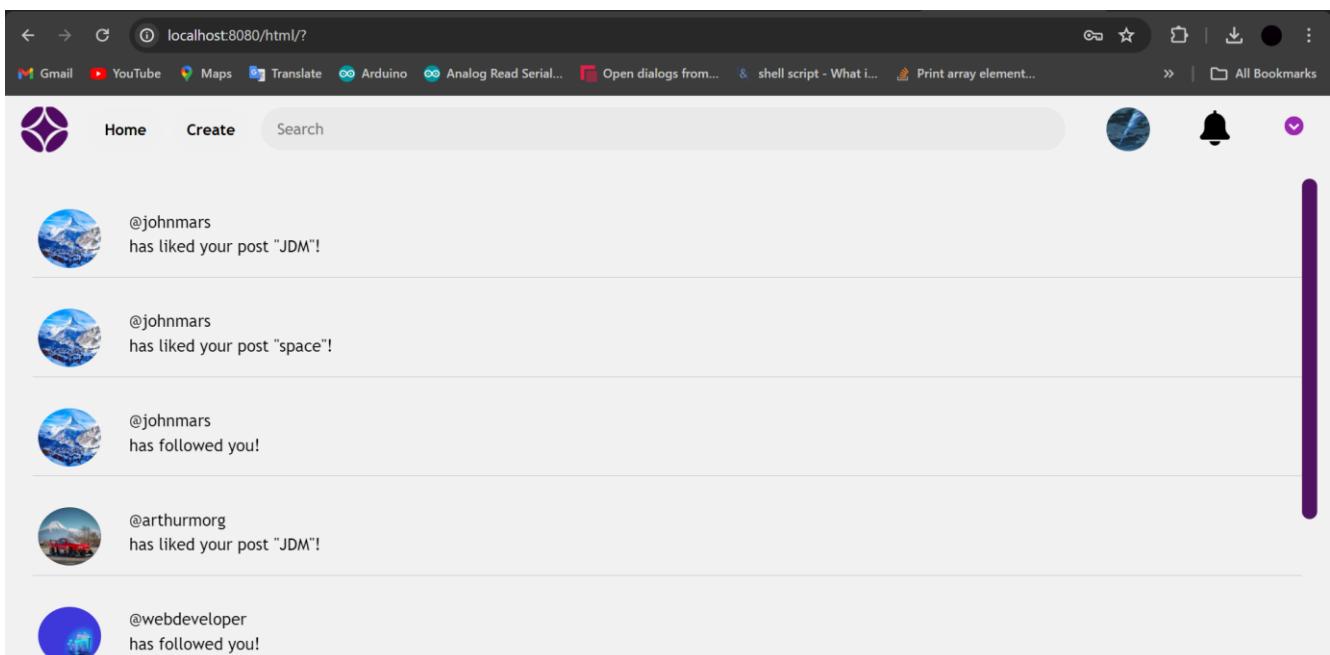


by clicking on the placeholder profile photo, they can attach the new profile photo and it shows a preview as well.



The user can exit by clicking the “x” button on the top right of the window of both the edit profile section and the expanded posts.

The viewer can get notifications about who followed them and liked their posts, if a person unfollows or unlikes their post, this notification is deleted. These notifications are scrollable and the newer notifications appear the top of the page.



localhost:8080/html/?

Gmail YouTube Maps Translate Arduino Analog Read Serial... Open dialogs from... shell script - What i... Print array element... All Bookmarks

Home Create Search

@johnmars has followed you!

@arthurmorg has liked your post "JDM"!

@webdeveloper has followed you!

@middlesexdubai has followed you!

@arthurmorg has followed you!

Finally the user can log out by clicking the button in the dropdown,

localhost:8080/html/?

Gmail YouTube Maps Translate Arduino Analog Read Serial... Open dialogs from... shell script - What i... Print array element... All Bookmarks

Home Create Search

@webdeveloper has followed you!

@arthurmorg has liked your post "JDM"!

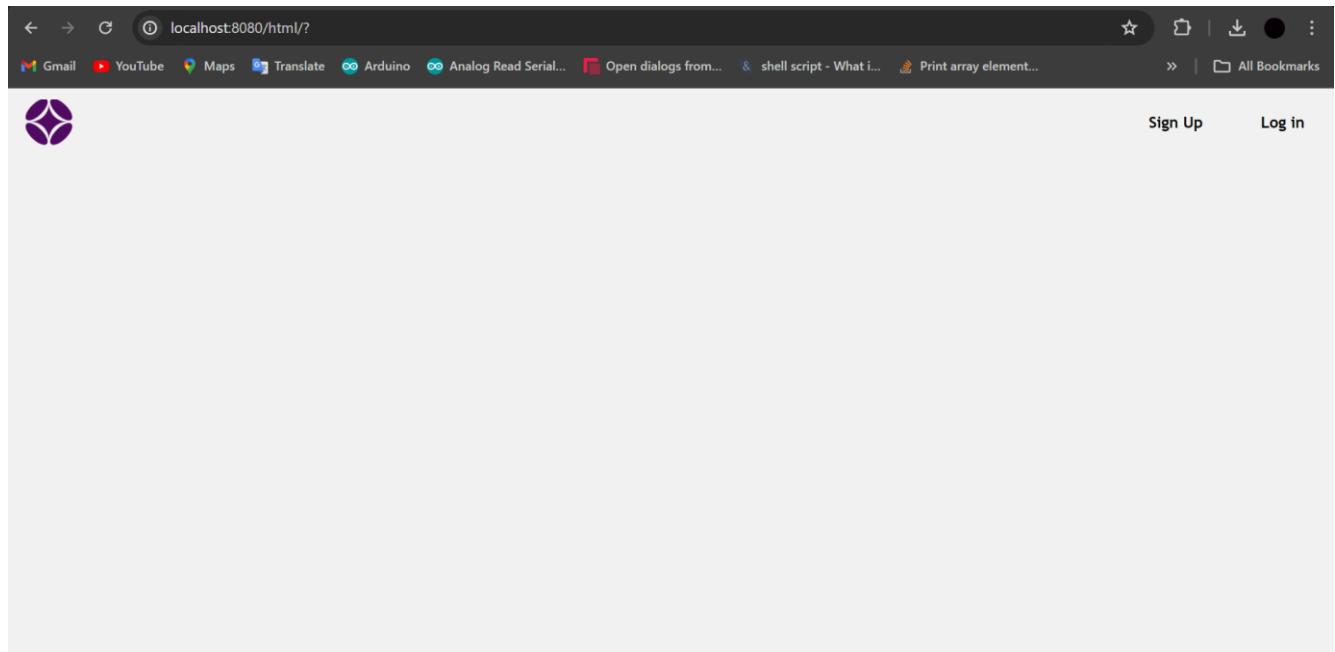
@johnmars has followed you!

@johnmars has liked your post "space"!

@johnmars has liked your post "JDM"!

Log Out

and the user is redirected back to the first page,



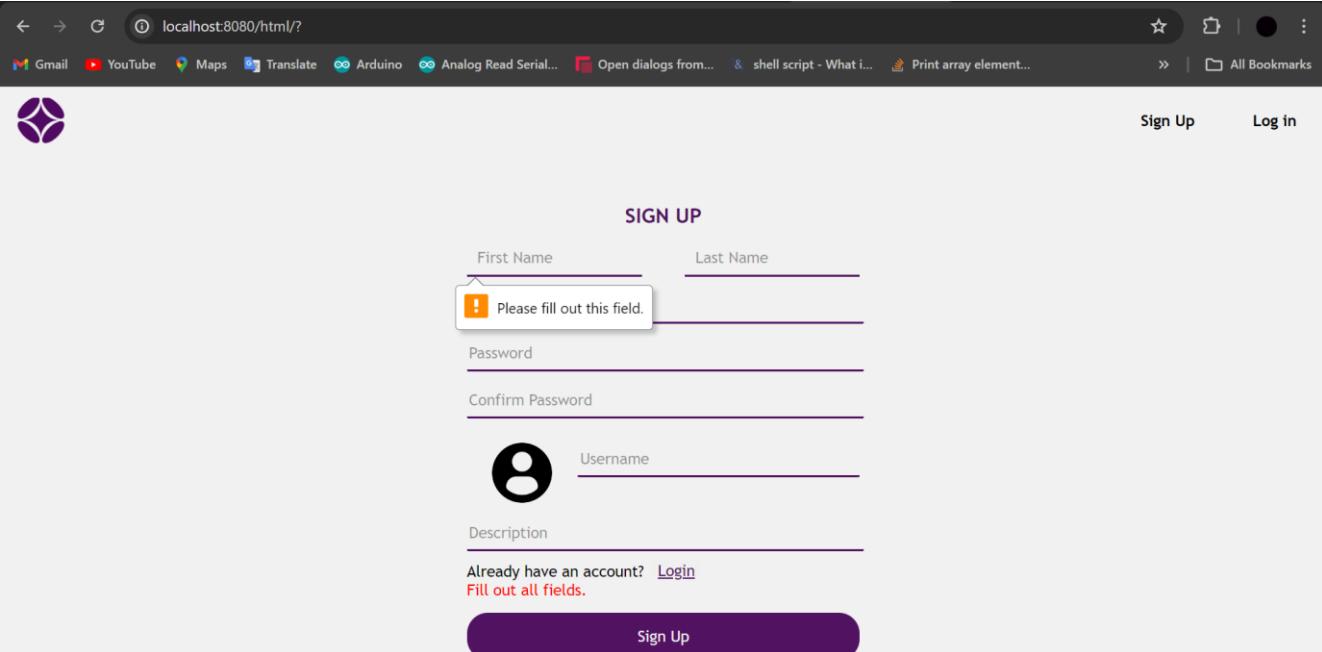
Processes

Sign Up

The user is required to put in their first name, last name, an email with a valid domain (outlook.com, yahoo.com, gmail.com), password, confirm password, profile photo, username, and description.

Before the form is submitted, validation is performed and below are the different types of errors that the form catches.

1) Fields are not filled out.



A screenshot of a web browser window showing a 'SIGN UP' form. The browser's address bar shows 'localhost:8080/html/?'. The page has a purple header with a logo on the left and 'Sign Up' and 'Log in' buttons on the right. The main content area has a white background with purple borders around input fields. The 'First Name' field is highlighted with a red border and contains the error message 'Please fill out this field.' in a red box. Below it are fields for 'Last Name', 'Password', and 'Confirm Password'. To the left of the 'Username' field is a black silhouette icon of a person. The 'Description' field is empty. At the bottom, there is a red note: 'Already have an account? [Login](#)' and 'Fill out all fields.' A large purple 'Sign Up' button is at the bottom.

A screenshot of a web browser window showing a sign-up form. The URL bar shows 'localhost:8080/html/?'. The browser toolbar includes links for Gmail, YouTube, Maps, Translate, Arduino, Analog Read Serial..., Open dialogs from..., shell script - What i..., Print array element..., All Bookmarks, Sign Up, and Log in.

The sign-up form has the following fields:

- First Name: Syed
- Last Name: Nasiruddin
- Email: testforreport@gmail.com
- (two lines)
- (two lines)
- User ID: m00914286
- Description: (empty)

An error message is displayed: "Already have an account? [Login](#) Fill out all fields! Please fill out this field."

The "Sign Up" button is at the bottom.

2) Special characters in the first or last name (for example, brackets []).

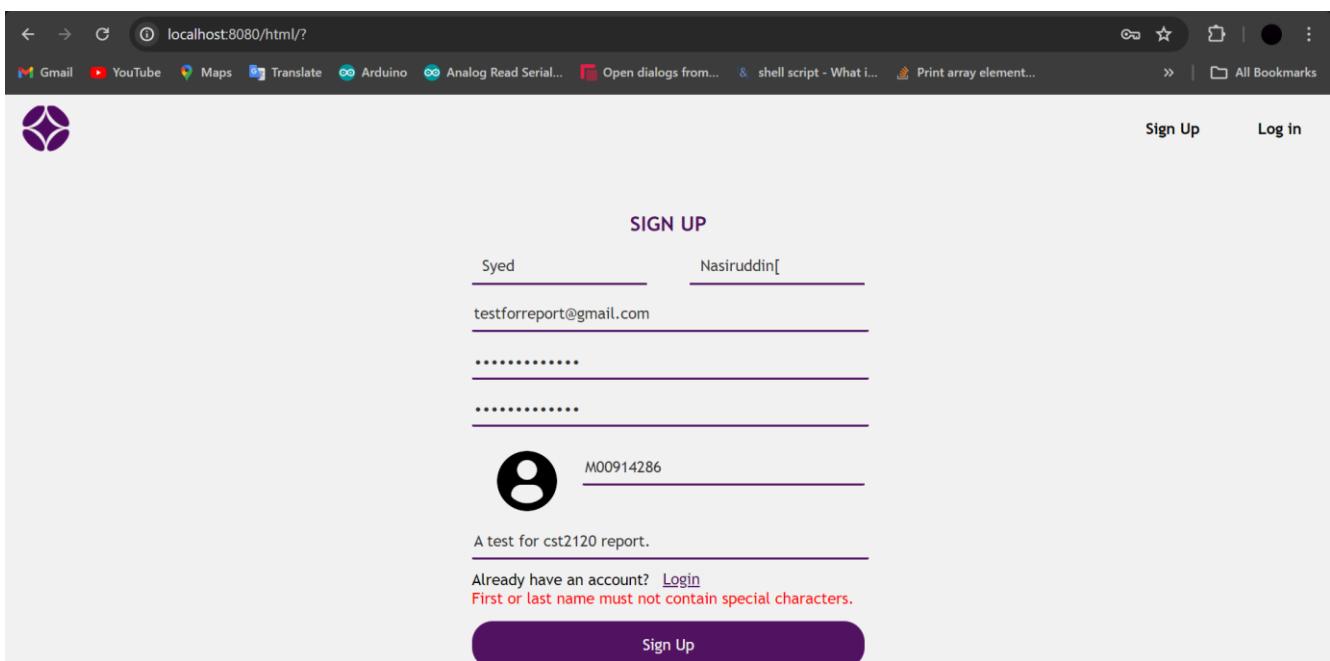
A screenshot of a web browser window showing a sign-up form. The URL bar shows 'localhost:8080/html/?'. The browser toolbar includes links for Gmail, YouTube, Maps, Translate, Arduino, Analog Read Serial..., Open dialogs from..., shell script - What i..., Print array element..., All Bookmarks, Sign Up, and Log in.

The sign-up form has the following fields:

- First Name: Syed[(with a red border)
- Last Name: Nasiruddin
- Email: testforreport@gmail.com
- (two lines)
- (two lines)
- User ID: M00914286
- Description: A test for cst2120 report.

An error message is displayed: "Already have an account? [Login](#) First or last name must not contain special characters."

The "Sign Up" button is at the bottom.



SIGN UP

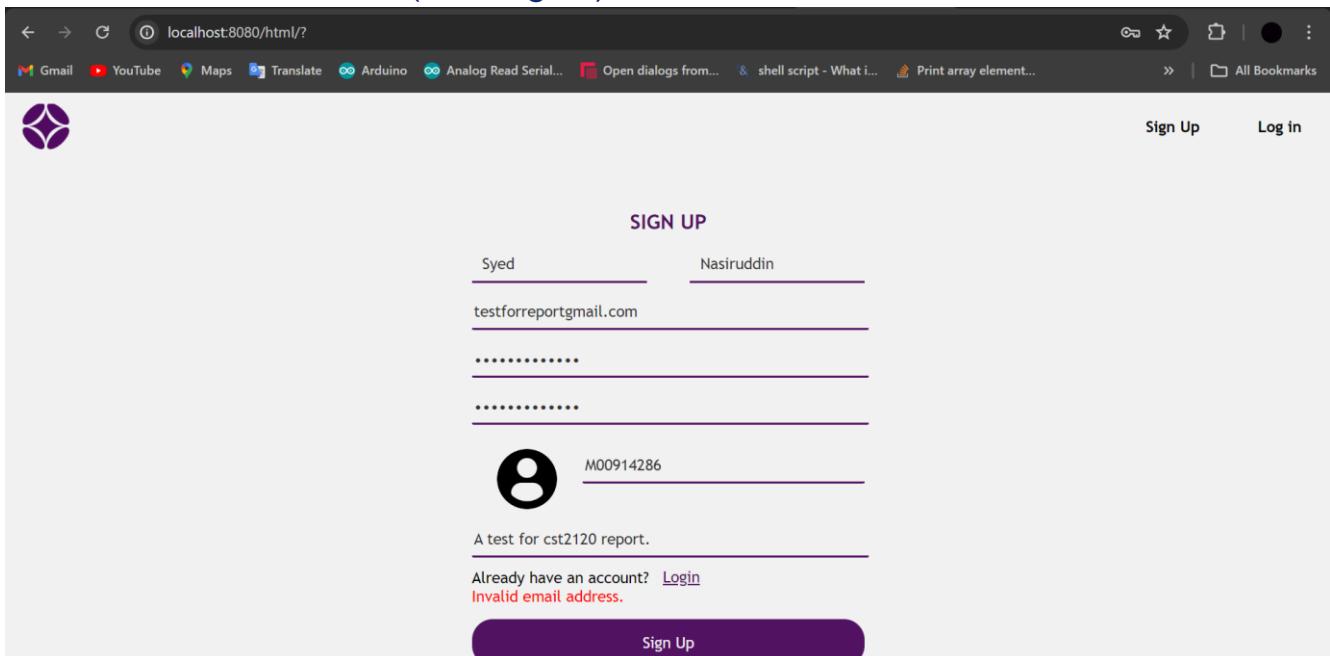
Syed Nasiruddin[
testforreport@gmail.com
.....
.....
 M00914286

A test for cst2120 report.

Already have an account? [Login](#)
First or last name must not contain special characters.

Sign Up

3. Invalid email address (missing @)



SIGN UP

Syed Nasiruddin
testforreportgmail.com
.....
.....
 M00914286

A test for cst2120 report.

Already have an account? [Login](#)
Invalid email address.

Sign Up

3. Invalid email domain. (not gmail, yahoo, outlook, aol, hotmail, mdx.ac.live.uk)

localhost:8080/html/?

Gmail YouTube Maps Translate Arduino Analog Read Serial... Open dialogs from... shell script - What i... Print array element... All Bookmarks Sign Up Log in

Syed Nasiruddin

testforreport@somedomain.com

.....

.....

 M00914286

A test for cst2120 report.

Already have an account? [Login](#)
Invalid email domain.

Sign Up

4. Email already taken.

localhost:8080/html/?

Gmail YouTube Maps Translate Arduino Analog Read Serial... Open dialogs from... shell script - What i... Print array element... All Bookmarks Sign Up Log in

Syed Nasiruddin

syednasir@gmail.com

.....

.....

 M00914286

A test for cst2120 report.

Already have an account? [Login](#)
Email already in use.

Sign Up

pixstarCW2.users

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } or [Generate query](#) [Explain](#) [Reset](#) [Find](#) [Options](#)

[ADD DATA](#) [EXPORT DATA](#)

1–7 of 7 [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#) [14](#) [15](#) [16](#) [17](#) [18](#) [19](#) [20](#) [21](#) [22](#) [23](#) [24](#) [25](#) [26](#) [27](#) [28](#) [29](#) [30](#) [31](#) [32](#) [33](#) [34](#) [35](#) [36](#) [37](#) [38](#) [39](#) [40](#) [41](#) [42](#) [43](#) [44](#) [45](#) [46](#) [47](#) [48](#) [49](#) [50](#) [51](#) [52](#) [53](#) [54](#) [55](#) [56](#) [57](#) [58](#) [59](#) [60](#) [61](#) [62](#) [63](#) [64](#) [65](#) [66](#) [67](#) [68](#) [69](#) [70](#) [71](#) [72](#) [73](#) [74](#) [75](#) [76](#) [77](#) [78](#) [79](#) [80](#) [81](#) [82](#) [83](#) [84](#) [85](#) [86](#) [87](#) [88](#) [89](#) [90](#) [91](#) [92](#) [93](#) [94](#) [95](#) [96](#) [97](#) [98](#) [99](#) [100](#) [101](#) [102](#) [103](#) [104](#) [105](#) [106](#) [107](#) [108](#) [109](#) [110](#) [111](#) [112](#) [113](#) [114](#) [115](#) [116](#) [117](#) [118](#) [119](#) [120](#) [121](#) [122](#) [123](#) [124](#) [125](#) [126](#) [127](#) [128](#) [129](#) [130](#) [131](#) [132](#) [133](#) [134](#) [135](#) [136](#) [137](#) [138](#) [139](#) [140](#) [141](#) [142](#) [143](#) [144](#) [145](#) [146](#) [147](#) [148](#) [149](#) [150](#) [151](#) [152](#) [153](#) [154](#) [155](#) [156](#) [157](#) [158](#) [159](#) [160](#) [161](#) [162](#) [163](#) [164](#) [165](#) [166](#) [167](#) [168](#) [169](#) [170](#) [171](#) [172](#) [173](#) [174](#) [175](#) [176](#) [177](#) [178](#) [179](#) [180](#) [181](#) [182](#) [183](#) [184](#) [185](#) [186](#) [187](#) [188](#) [189](#) [190](#) [191](#) [192](#) [193](#) [194](#) [195](#) [196](#) [197](#) [198](#) [199](#) [200](#) [201](#) [202](#) [203](#) [204](#) [205](#) [206](#) [207](#) [208](#) [209](#) [210](#) [211](#) [212](#) [213](#) [214](#) [215](#) [216](#) [217](#) [218](#) [219](#) [220](#) [221](#) [222](#) [223](#) [224](#) [225](#) [226](#) [227](#) [228](#) [229](#) [230](#) [231](#) [232](#) [233](#) [234](#) [235](#) [236](#) [237](#) [238](#) [239](#) [240](#) [241](#) [242](#) [243](#) [244](#) [245](#) [246](#) [247](#) [248](#) [249](#) [250](#) [251](#) [252](#) [253](#) [254](#) [255](#) [256](#) [257](#) [258](#) [259](#) [260](#) [261](#) [262](#) [263](#) [264](#) [265](#) [266](#) [267](#) [268](#) [269](#) [270](#) [271](#) [272](#) [273](#) [274](#) [275](#) [276](#) [277](#) [278](#) [279](#) [280](#) [281](#) [282](#) [283](#) [284](#) [285](#) [286](#) [287](#) [288](#) [289](#) [290](#) [291](#) [292](#) [293](#) [294](#) [295](#) [296](#) [297](#) [298](#) [299](#) [300](#) [301](#) [302](#) [303](#) [304](#) [305](#) [306](#) [307](#) [308](#) [309](#) [310](#) [311](#) [312](#) [313](#) [314](#) [315](#) [316](#) [317](#) [318](#) [319](#) [320](#) [321](#) [322](#) [323](#) [324](#) [325](#) [326](#) [327](#) [328](#) [329](#) [330](#) [331](#) [332](#) [333](#) [334](#) [335](#) [336](#) [337](#) [338](#) [339](#) [340](#) [341](#) [342](#) [343](#) [344](#) [345](#) [346](#) [347](#) [348](#) [349](#) [350](#) [351](#) [352](#) [353](#) [354](#) [355](#) [356](#) [357](#) [358](#) [359](#) [360](#) [361](#) [362](#) [363](#) [364](#) [365](#) [366](#) [367](#) [368](#) [369](#) [370](#) [371](#) [372](#) [373](#) [374](#) [375](#) [376](#) [377](#) [378](#) [379](#) [380](#) [381](#) [382](#) [383](#) [384](#) [385](#) [386](#) [387](#) [388](#) [389](#) [390](#) [391](#) [392](#) [393](#) [394](#) [395](#) [396](#) [397](#) [398](#) [399](#) [400](#) [401](#) [402](#) [403](#) [404](#) [405](#) [406](#) [407](#) [408](#) [409](#) [410](#) [411](#) [412](#) [413](#) [414](#) [415](#) [416](#) [417](#) [418](#) [419](#) [420](#) [421](#) [422](#) [423](#) [424](#) [425](#) [426](#) [427](#) [428](#) [429](#) [430](#) [431](#) [432](#) [433](#) [434](#) [435](#) [436](#) [437](#) [438](#) [439](#) [440](#) [441](#) [442](#) [443](#) [444](#) [445](#) [446](#) [447](#) [448](#) [449](#) [450](#) [451](#) [452](#) [453](#) [454](#) [455](#) [456](#) [457](#) [458](#) [459](#) [460](#) [461](#) [462](#) [463](#) [464](#) [465](#) [466](#) [467](#) [468](#) [469](#) [470](#) [471](#) [472](#) [473](#) [474](#) [475](#) [476](#) [477](#) [478](#) [479](#) [480](#) [481](#) [482](#) [483](#) [484](#) [485](#) [486](#) [487](#) [488](#) [489](#) [490](#) [491](#) [492](#) [493](#) [494](#) [495](#) [496](#) [497](#) [498](#) [499](#) [500](#) [501](#) [502](#) [503](#) [504](#) [505](#) [506](#) [507](#) [508](#) [509](#) [510](#) [511](#) [512](#) [513](#) [514](#) [515](#) [516](#) [517](#) [518](#) [519](#) [520](#) [521](#) [522](#) [523](#) [524](#) [525](#) [526](#) [527](#) [528](#) [529](#) [530](#) [531](#) [532](#) [533](#) [534](#) [535](#) [536](#) [537](#) [538](#) [539](#) [540](#) [541](#) [542](#) [543](#) [544](#) [545](#) [546](#) [547](#) [548](#) [549](#) [550](#) [551](#) [552](#) [553](#) [554](#) [555](#) [556](#) [557](#) [558](#) [559](#) [560](#) [561](#) [562](#) [563](#) [564](#) [565](#) [566](#) [567](#) [568](#) [569](#) [570](#) [571](#) [572](#) [573](#) [574](#) [575](#) [576](#) [577](#) [578](#) [579](#) [580](#) [581](#) [582](#) [583](#) [584](#) [585](#) [586](#) [587](#) [588](#) [589](#) [590](#) [591](#) [592](#) [593](#) [594](#) [595](#) [596](#) [597](#) [598](#) [599](#) [600](#) [601](#) [602](#) [603](#) [604](#) [605](#) [606](#) [607](#) [608](#) [609](#) [610](#) [611](#) [612](#) [613](#) [614](#) [615](#) [616](#) [617](#) [618](#) [619](#) [620](#) [621](#) [622](#) [623](#) [624](#) [625](#) [626](#) [627](#) [628](#) [629](#) [630](#) [631](#) [632](#) [633](#) [634](#) [635](#) [636](#) [637](#) [638](#) [639](#) [640](#) [641](#) [642](#) [643](#) [644](#) [645](#) [646](#) [647](#) [648](#) [649](#) [650](#) [651](#) [652](#) [653](#) [654](#) [655](#) [656](#) [657](#) [658](#) [659](#) [660](#) [661](#) [662](#) [663](#) [664](#) [665](#) [666](#) [667](#) [668](#) [669](#) [670](#) [671](#) [672](#) [673](#) [674](#) [675](#) [676](#) [677](#) [678](#) [679](#) [680](#) [681](#) [682](#) [683](#) [684](#) [685](#) [686](#) [687](#) [688](#) [689](#) [690](#) [691](#) [692](#) [693](#) [694](#) [695](#) [696](#) [697](#) [698](#) [699](#) [700](#) [701](#) [702](#) [703](#) [704](#) [705](#) [706](#) [707](#) [708](#) [709](#) [710](#) [711](#) [712](#) [713](#) [714](#) [715](#) [716](#) [717](#) [718](#) [719](#) [720](#) [721](#) [722](#) [723](#) [724](#) [725](#) [726](#) [727](#) [728](#) [729](#) [730](#) [731](#) [732](#) [733](#) [734](#) [735](#) [736](#) [737](#) [738](#) [739](#) [740](#) [741](#) [742](#) [743](#) [744](#) [745](#) [746](#) [747](#) [748](#) [749](#) [750](#) [751](#) [752](#) [753](#) [754](#) [755](#) [756](#) [757](#) [758](#) [759](#) [760](#) [761](#) [762](#) [763](#) [764](#) [765](#) [766](#) [767](#) [768](#) [769](#) [770](#) [771](#) [772](#) [773](#) [774](#) [775](#) [776](#) [777](#) [778](#) [779](#) [780](#) [781](#) [782](#) [783](#) [784](#) [785](#) [786](#) [787](#) [788](#) [789](#) [790](#) [791](#) [792](#) [793](#) [794](#) [795](#) [796](#) [797](#) [798](#) [799](#) [800](#) [801](#) [802](#) [803](#) [804](#) [805](#) [806](#) [807](#) [808](#) [809](#) [810](#) [811](#) [812](#) [813](#) [814](#) [815](#) [816](#) [817](#) [818](#) [819](#) [820](#) [821](#) [822](#) [823](#) [824](#) [825](#) [826](#) [827](#) [828](#) [829](#) [830](#) [831](#) [832](#) [833](#) [834](#) [835](#) [836](#) [837](#) [838](#) [839](#) [840](#) [841](#) [842](#) [843](#) [844](#) [845](#) [846](#) [847](#) [848](#) [849](#) [850](#) [851](#) [852](#) [853](#) [854](#) [855](#) [856](#) [857](#) [858](#) [859](#) [860](#) [861](#) [862](#) [863](#) [864](#) [865](#) [866](#) [867](#) [868](#) [869](#) [870](#) [871](#) [872](#) [873](#) [874](#) [875](#) [876](#) [877](#) [878](#) [879](#) [880](#) [881](#) [882](#) [883](#) [884](#) [885](#) [886](#) [887](#) [888](#) [889](#) [890](#) [891](#) [892](#) [893](#) [894](#) [895](#) [896](#) [897](#) [898](#) [899](#) [900](#) [901](#) [902](#) [903](#) [904](#) [905](#) [906](#) [907](#) [908](#) [909](#) [910](#) [911](#) [912](#) [913](#) [914](#) [915](#) [916](#) [917](#) [918](#) [919](#) [920](#) [921](#) [922](#) [923](#) [924](#) [925](#) [926](#) [927](#) [928](#) [929](#) [930](#) [931](#) [932](#) [933](#) [934](#) [935](#) [936](#) [937](#) [938](#) [939](#) [940](#) [941](#) [942](#) [943](#) [944](#) [945](#) [946](#) [947](#) [948](#) [949](#) [950](#) [951](#) [952](#) [953](#) [954](#) [955](#) [956](#) [957](#) [958](#) [959](#) [960](#) [961](#) [962](#) [963](#) [964](#) [965](#) [966](#) [967](#) [968](#) [969](#) [970](#) [971](#) [972](#) [973](#) [974](#) [975](#) [976](#) [977](#) [978](#) [979](#) [980](#) [981](#) [982](#) [983](#) [984](#) [985](#) [986](#) [987](#) [988](#) [989](#) [990](#) [991](#) [992](#) [993](#) [994](#) [995](#) [996](#) [997](#) [998](#) [999](#) [1000](#)

SIGN UP

Syed Nasiruddin

testforreport@gmail.com

.....
.....

M00914286

A test for cst2120 report.

Already have an account? [Login](#)

Password must contain only alphanumeric, dollar sign, ampersand, and period.

[Sign Up](#)

6. Password less than eight characters.

A screenshot of a web browser window showing a sign-up form. The URL bar shows "localhost:8080/html/?". The page title is "SIGN UP". There are two input fields: "Syed" and "Nasiruddin". Below them is an email input field containing "testforreport@gmail.com". Underneath the email field are two password input fields, both containing "....". To the left of the second password field is a user icon. Next to the second password field is the text "M00914286". Below the password fields is a note: "A test for cst2120 report.". At the bottom, there is a message: "Already have an account? [Login](#)" and "Password must be at least 8 characters long". A large purple "Sign Up" button is at the bottom.

7. Username contains invalid characters.

A screenshot of a web browser window showing a sign-up form. The URL bar shows "localhost:8080/html/?". The page title is "SIGN UP". There are two input fields: "Syed" and "Nasiruddin". Below them is an email input field containing "testforreport@gmail.com". Underneath the email field are two password input fields, both containing ".....". To the left of the second password field is a user icon. Next to the second password field is the text "@@M00914286". Below the password fields is a note: "A test for cst2120 report.". At the bottom, there is a message: "Already have an account? [Login](#)" and "Username contains invalid character(s.)". A large purple "Sign Up" button is at the bottom.

8. Username already taken.

The screenshot shows a web browser window and a MongoDB Compass interface side-by-side.

Web Browser (localhost:8080/html/?)

- Sign Up Form:**
 - First Name: Syed
 - Last Name: Nasiruddin
 - Email: testforreport@gmail.com
 - Two masked password fields (*****)
 - Profile Photo: A small circular image of a person's face.
 - Username: nasir
 - Description: A test for cst2120 report.
 - Already have an account? [Login](#)
 - Error message: Username already exists.
 - Sign Up button

MongoDB Compass - localhost:27017/pixstarCW2.users

- Left Sidebar:** Shows databases: admin, config, cst2120, local, pixstar, pixstarCW2 (selected), comments, followers, following, notifications, posts, users.
- Top Bar:** Connect, Edit, View, Collection, Help.
- Collection Overview:** pixstarCW2.users (7 DOCUMENTS, 1 INDEXES).
- Documents Tab:** Shows two document cards with user data.
- Document Data:**

```
_id: ObjectId('662592a02d2eca24a20b6235')
username: "nasir"
password: "ab5293d47f22729bb25a44f31a62940e"
email: "syednasir@gmail.com"
firstName: "Syed"
lastName: "Nasiruddin"
description: "I'm nasir."
profilePhoto: "/9j/4AAQSkZJRgABAQAAAQABAAAD/4TGIRXhpZgAATU0AKgAAAAgABQEaAUAAAAABAAAAAg=="
```



```
_id: ObjectId('662592d82d2eca24a20b6236')
username: "middlesexdubai"
password: "7d58e8d9ae3f4ed2722a5b7d9a3db111"
email: "mdxdubai@gmail.com"
firstName: "Middlesex"
lastName: "Dubai"
description: "Middlesex University Dubai"
profilePhoto: "iVBORw0KGeoAAAANSUhEUgAAAQEEAADhCAMAAAAJbSJIAABUFBMVExkGxT///8///v="
```

once again, the username is already taken by “nasir”.

9. Description too long.

A screenshot of a web browser window showing a sign-up form. The URL bar shows "localhost:8080/html/?". The page title is "SIGN UP". The form fields include:

- First Name: Syed
- Last Name: Nasiruddin
- Email: testforreport@gmail.com
- Two password fields, both containing "*****".
- User ID: M00914286
- Description: "A test for cst2120 report, i hope it goes well because" (This is a very long description, exceeding the 40-character limit mentioned in the note.)

Below the form, there is a message: "Already have an account? [Login](#)". A red error message says "Description must not exceed 40 characters.". At the bottom is a purple "Sign Up" button.

10. Profile photo not chosen.

A screenshot of a web browser window showing a sign-up form. The URL bar shows "localhost:8080/html/?". The page title is "SIGN UP". The form fields include:

- First Name: Syed
- Last Name: Nasiruddin
- Email: testforreport@gmail.com
- Two password fields, both containing "*****".
- User ID: M00914286
- Description: "A test for cst2120 report."

Below the form, there is a message: "Already have an account? [Login](#)". A red error message says "Choose a profile photo. (JPEG, JPG, PNG or so., size should be < 1MB)".

NOTE: if a user attaches a file that is not an image or exceeds one MB, it is not set and if the user presses submit the same error is shown.

Once all the validation checks are passed, the form is submitted and the users collection is updated.

The screenshot shows the MongoDB Compass interface. The left sidebar lists databases and collections, with 'pixstarCW2' selected. The main area displays the 'users' collection. It shows 8 documents and 1 index. One document is selected, displaying its contents:

```
username: "percy"
password: "e8913f9d518ff1f4fc3591b4b9c431c4"
email: "percy@yahoo.com"
firstName: "percy"
lastName: "smith"
description: "shoot for the stars."
profilePhoto: "iVBORw0KGgoAAAANSUhEUgAAQAAAAEACAYAAABccqhmAAAdnU1EQVR4n02dF4xdXXHDz..."
```

Below the document, there is another document listed:

```
_id: ObjectId('6626a24a65017f38fe1819db')
username: "M00914286"
password: "e8d8beba3c0e98b05360eb0ad8c245e9"
email: "testforreport@gmail.com"
firstName: "Syed"
lastName: "Nasiruddin"
description: "A test for cst2120 report."
profilePhoto: "iVBORw0KGgoAAAANSUhEUgAAAlgAAAJYCAYAAAC+ZpjcAAABhGLDQ1BjQ0MgcHJvZmlsZQ..."
```

Login

The following outlines the login process, and the validation that takes place.

1. If all the fields are not filled out.

A screenshot of a web browser window showing a login form. The URL bar indicates the page is at `localhost:8080/html/?`. The browser toolbar includes links for Gmail, YouTube, Maps, Translate, Arduino, Analog Read Serial..., Open dialogs from..., shell script - What i..., Print array element..., Sign Up, and Log in. The main content area features a purple header with the word "LOG IN". Below it is a "Username" input field containing "M00914286". An "Password" input field is empty and highlighted with a red border, accompanied by a yellow warning icon and the text "Please fill out this field.". Below the fields is a link "Don't have an account? [Sign Up](#)". A note "Fill out all fields." is also present. A purple "Log In" button is at the bottom.

A screenshot of a web browser window showing a login form. The URL bar indicates the page is at `localhost:8080/html/?`. The browser toolbar includes links for Gmail, YouTube, Maps, Translate, Arduino, Analog Read Serial..., Open dialogs from..., shell script - What i..., Print array element..., Sign Up, and Log in. The main content area features a purple header with the word "LOG IN". Below it is a "Username" input field containing "M00914286". An "Password" input field is empty and highlighted with a red border, accompanied by a yellow warning icon and the text "Please fill out this field.". Below the fields is a link "Don't have a [Sign Up](#)". A note "Fill out all fields." is also present. A purple "Log In" button is at the bottom.

2. If the user doesn't exist.

A screenshot of a web browser window. The address bar shows "localhost:8080/html/?". The header bar is purple and contains a logo, a search icon, and several bookmarked links: Gmail, YouTube, Maps, Translate, Arduino, Analog Read Serial..., Open dialogs from..., shell script - What i..., Print array element..., Sign Up, and Log in. Below the header is a purple navigation bar with icons for back, forward, search, and other functions. The main content area has a light gray background. At the top center is a purple "LOG IN" button. Below it are two input fields: the first contains "M00000000" and the second contains "*****". Underneath the second field is the text "Don't have an account? [Sign Up](#)". In red text, it says "User not found". At the bottom is a purple "Log In" button.

3. If passwords don't match.

A screenshot of a web browser window, identical to the one above but with different input values. The address bar shows "localhost:8080/html/?". The header bar and purple navigation bar are also present. The main content area shows the "LOG IN" button at the top. Below it are two input fields: the first contains "M00914286" and the second contains "*****|". Underneath the second field is the text "Don't have an account? [Sign Up](#)". In red text, it says "Incorrect password". At the bottom is a purple "Log In" button.

Else the user will be logged and a session is created in which the user's username is stored.

The screenshot shows a browser window with the URL `localhost:8080/html/?`. On the left, there is a login form with fields for 'M00914286' and a password, and links for 'Sign Up' and 'Log In'. On the right, the browser's developer tools are open, specifically the Application tab under Storage. The Cookies section is expanded, showing a list of cookies for the domain `http://localhost:8080`. One cookie, named 'con...', is selected, and its value is displayed in the details panel below. The value is a long string of characters: `s%3A4wjr-UTjsVZkhUECp0k62MhSmPiYRfzJPrnkvKBX9Px2wE%2Boz2yoBqWUzVUMLY1t%2FBT9ht65o`.

NOTE: prior to logging in, no session or cookie is created.

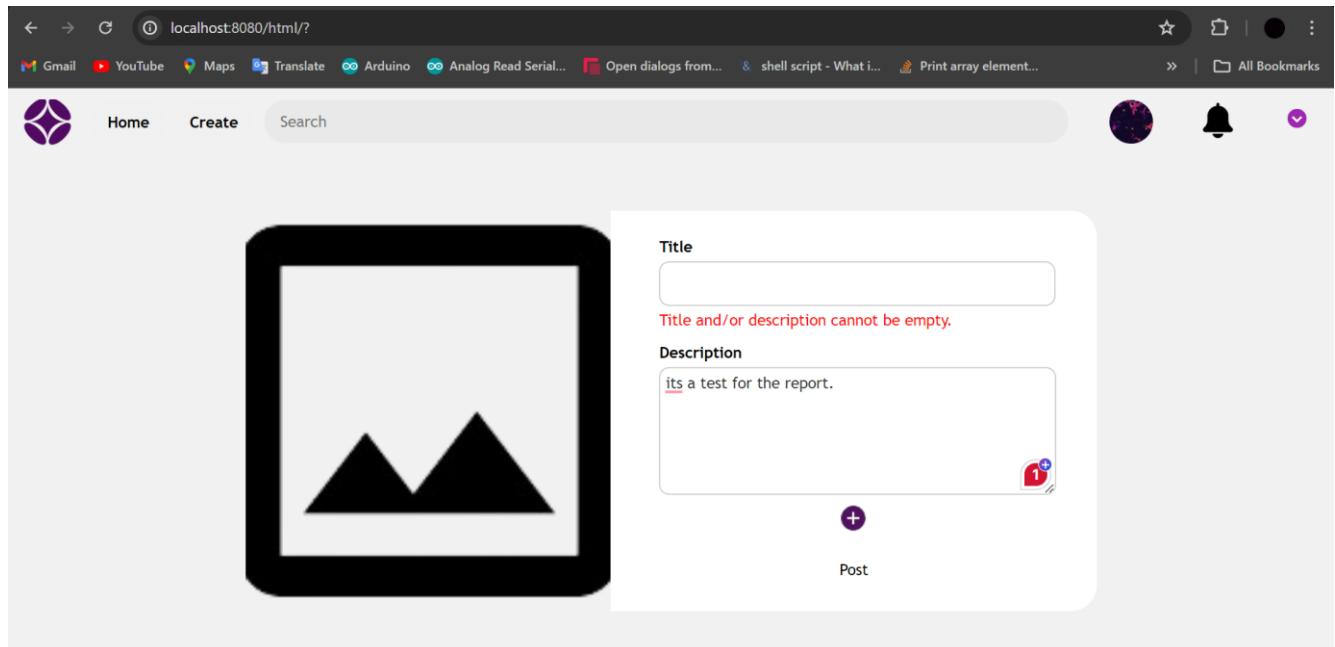
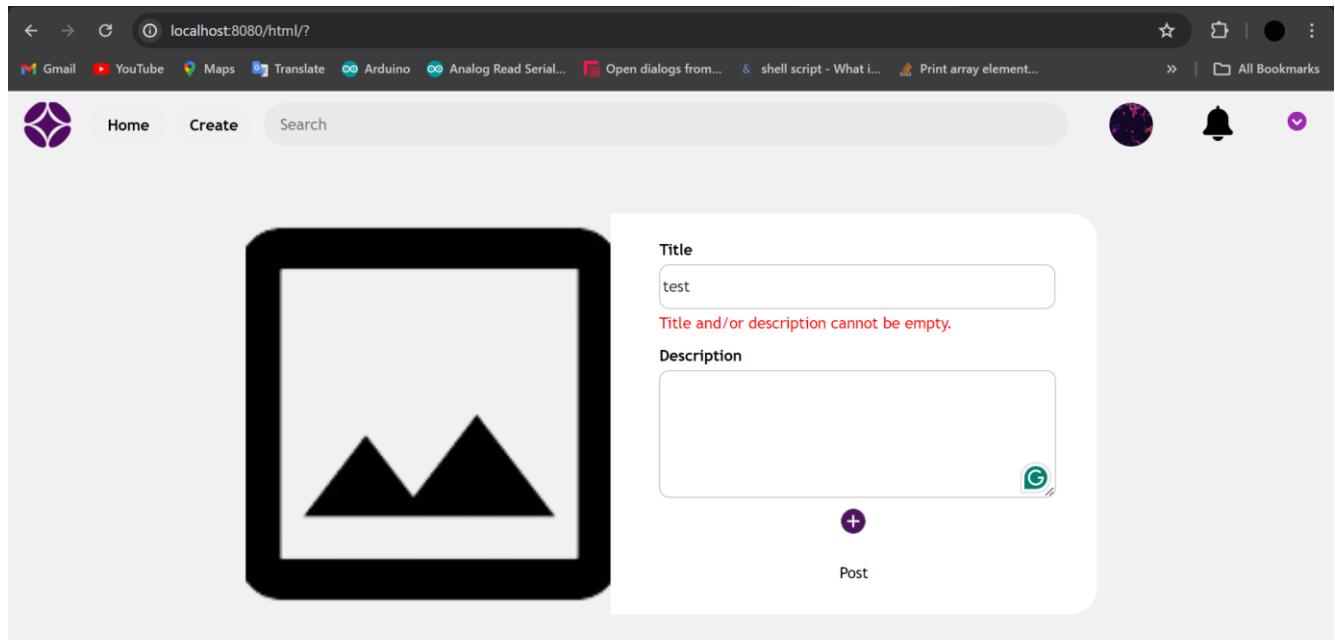
This screenshot is similar to the one above, showing the browser developer tools Application tab for the same URL. The cookies list shows the same cookie 'con...' with the same value. The 'Cookie Value' field at the bottom of the tool panel also displays the value: `s%3A4wjr-UTjsVZkhUECp0k62MhSmPiYRfzJPrnkvKBX9Px2wE%2Boz2yoBqWUzVUMLY1t%2FBT9ht65o`.

When the user logs in, the session and a cookie is made.

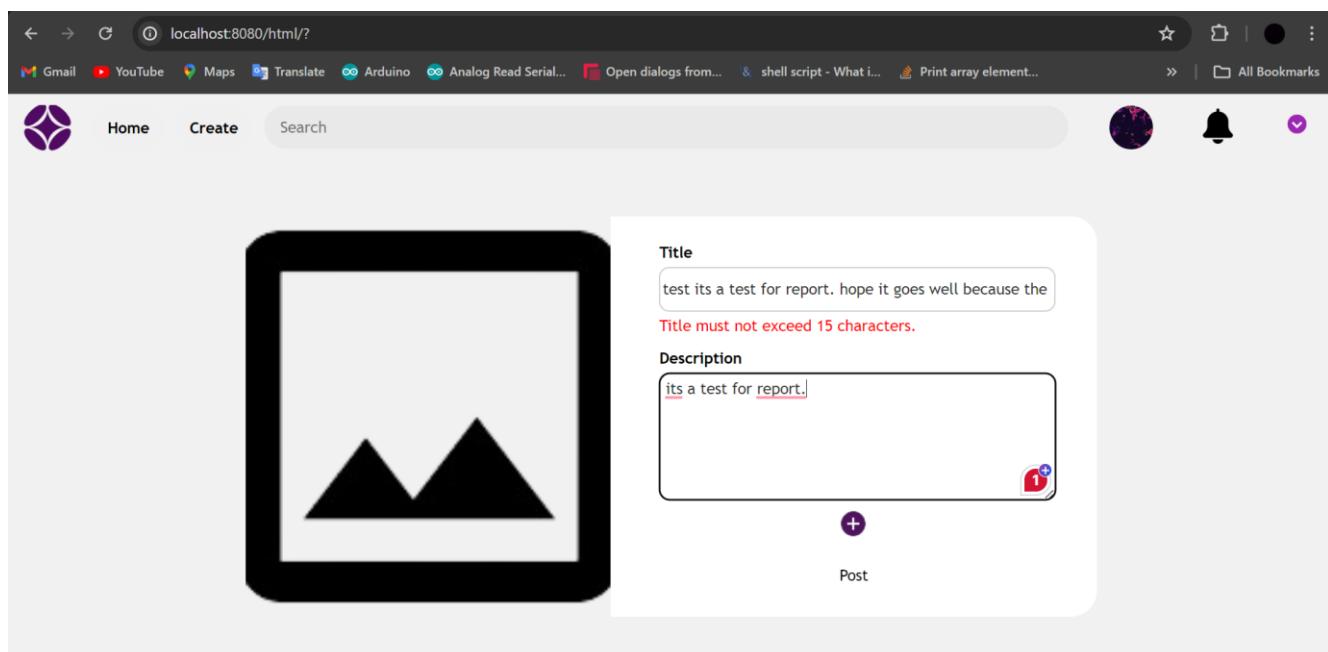
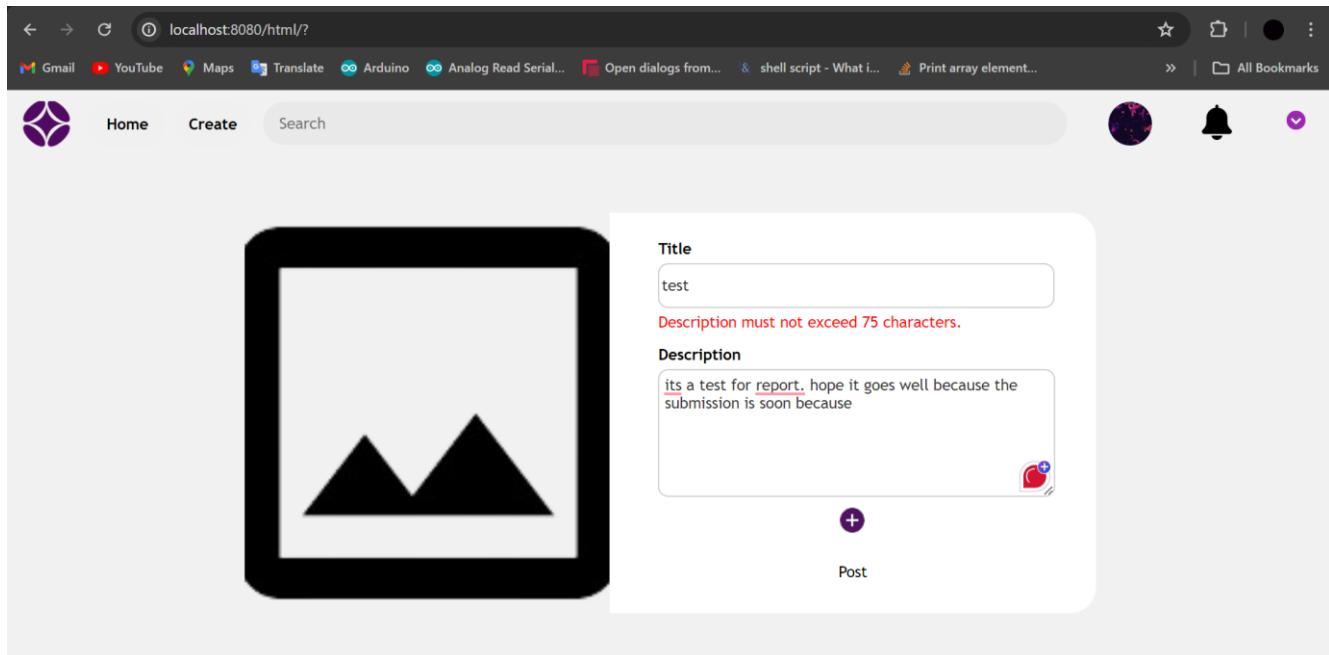
Create Post

The following outlines the post creation process and the validation that takes place.

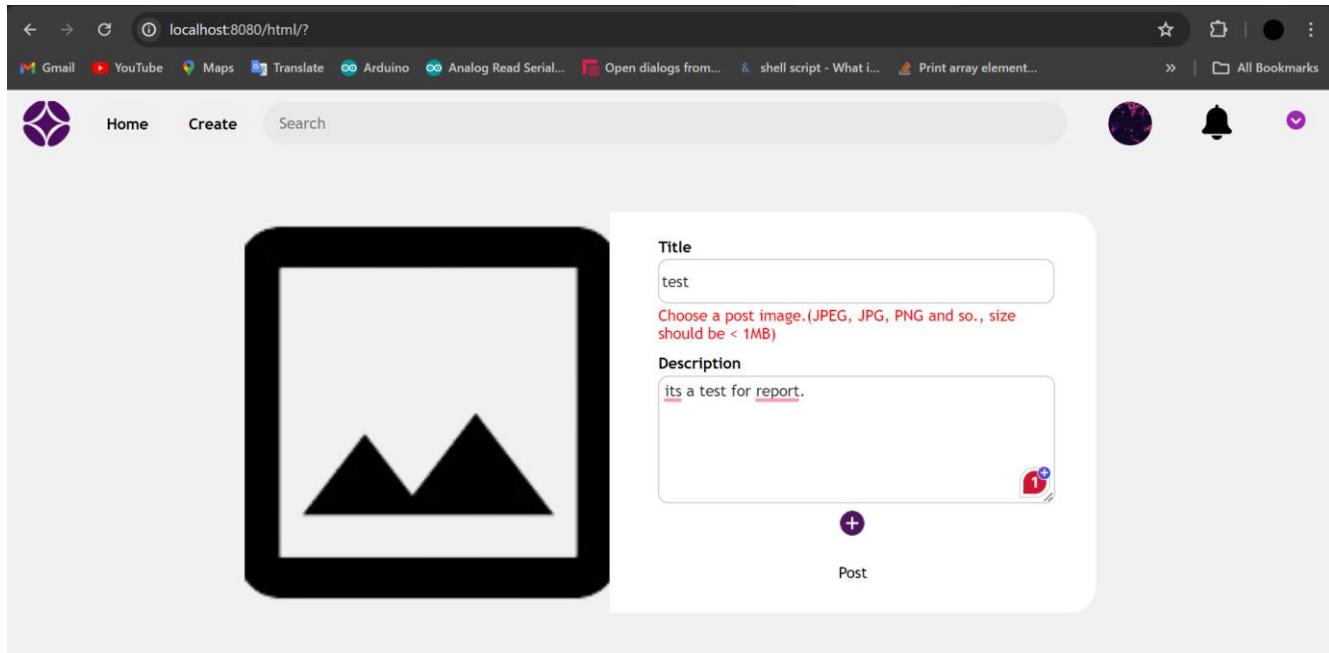
- 1) If all the fields are not filled out.



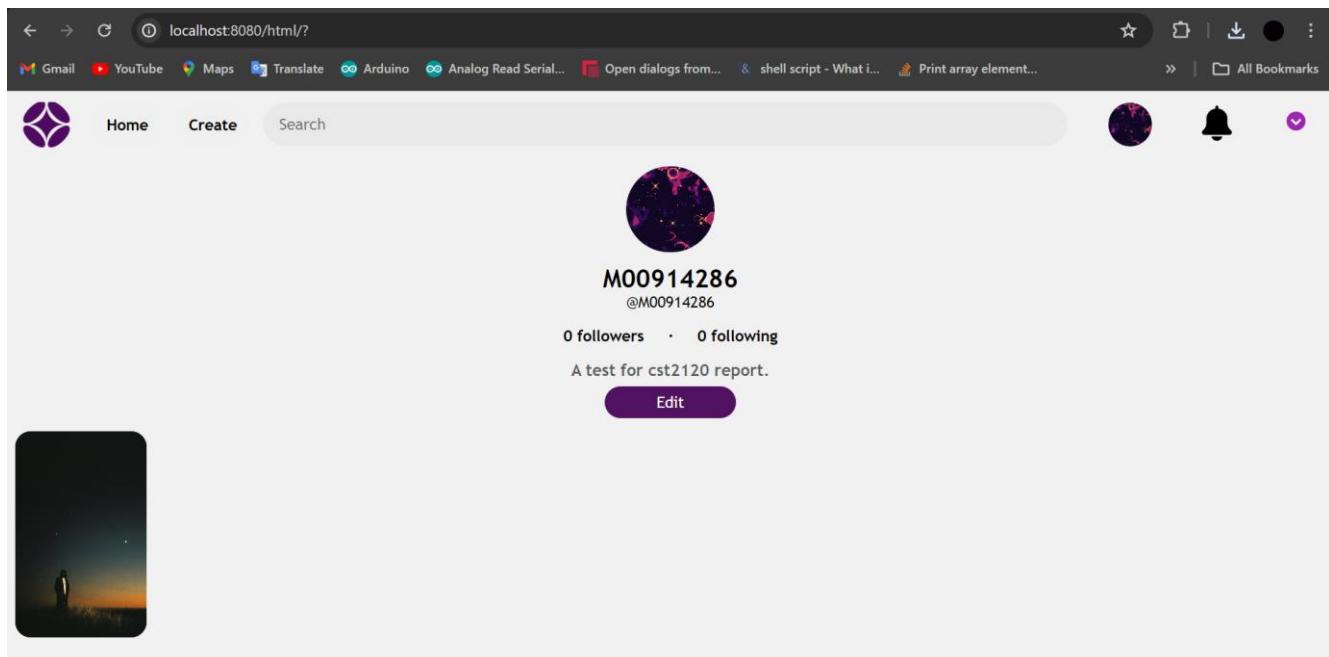
2) if any of the fields exceed the character limit.

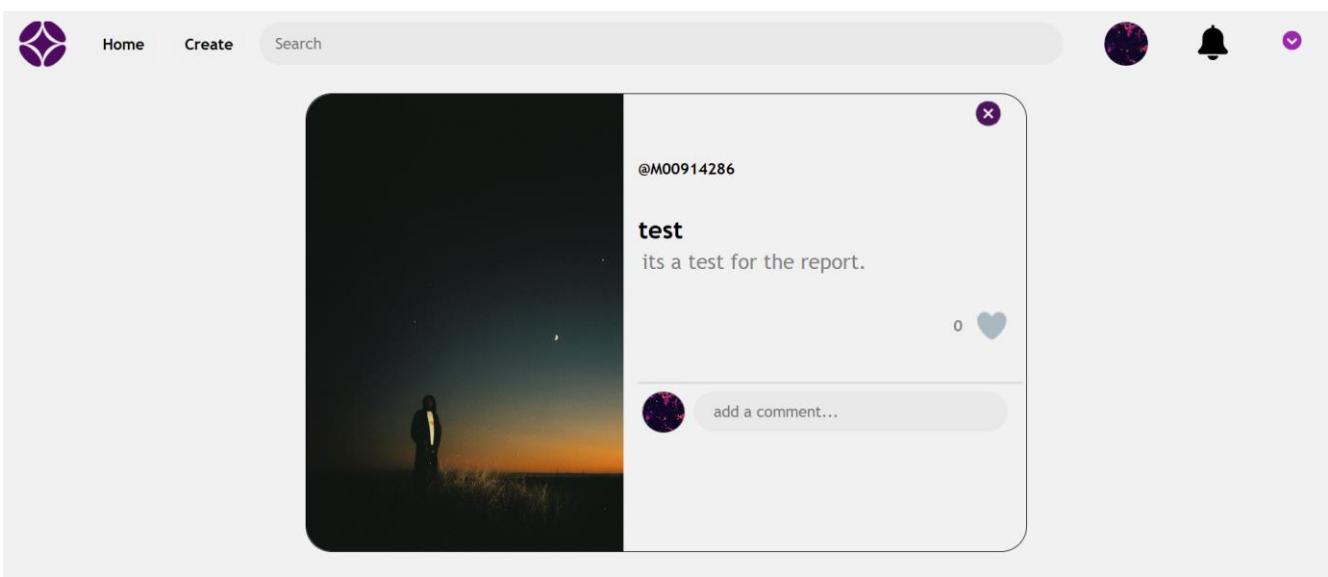


3. If a post image is not chosen.



If all the validation checks are passed, the post is made and the user can view it in their account page.





A document is made with the new posts details.

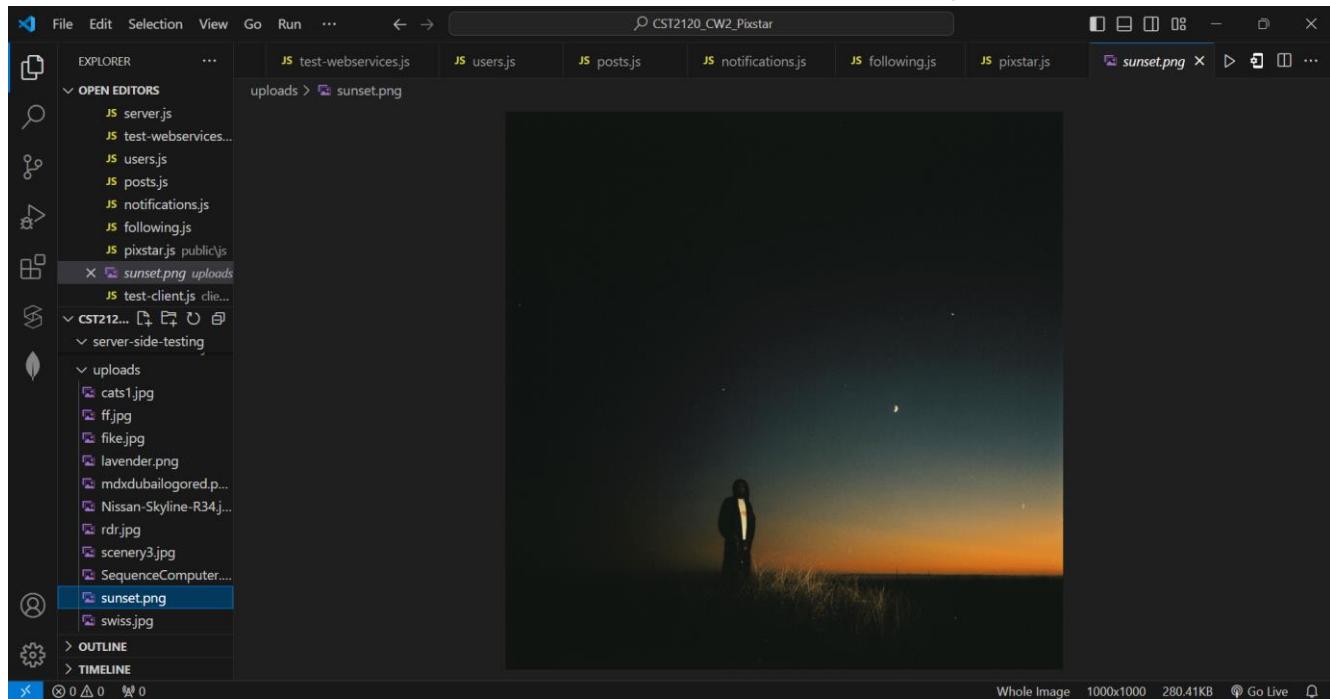
```

{
  "_id": ObjectId("662664a30b4a5826311221f1"),
  "user": "johnmars",
  "title": "swiss alps",
  "description": "so serene.",
  "likes": 0,
  "postImg": "/9j/4QlQaHR0cDovL25zIxFkb2J1LmNvbS94YXAvMS4wLwA8P3hwYNrZXQgYmVnaW49Iu..."
}

{
  "_id": ObjectId("6626c55c7f8a9082752e4bc"),
  "user": "M00914286",
  "title": "test",
  "description": "its a test for the report.",
  "likes": 0,
  "postImg": "iVBORw0KGgoAAAANSUhEUgAAA+gAAAPoCAMAAAB6fSTWAAA9lBMVEUPFBlo267RwYTlkD..."
}

```

The post photo is uploaded to the “uploads” file in the project.

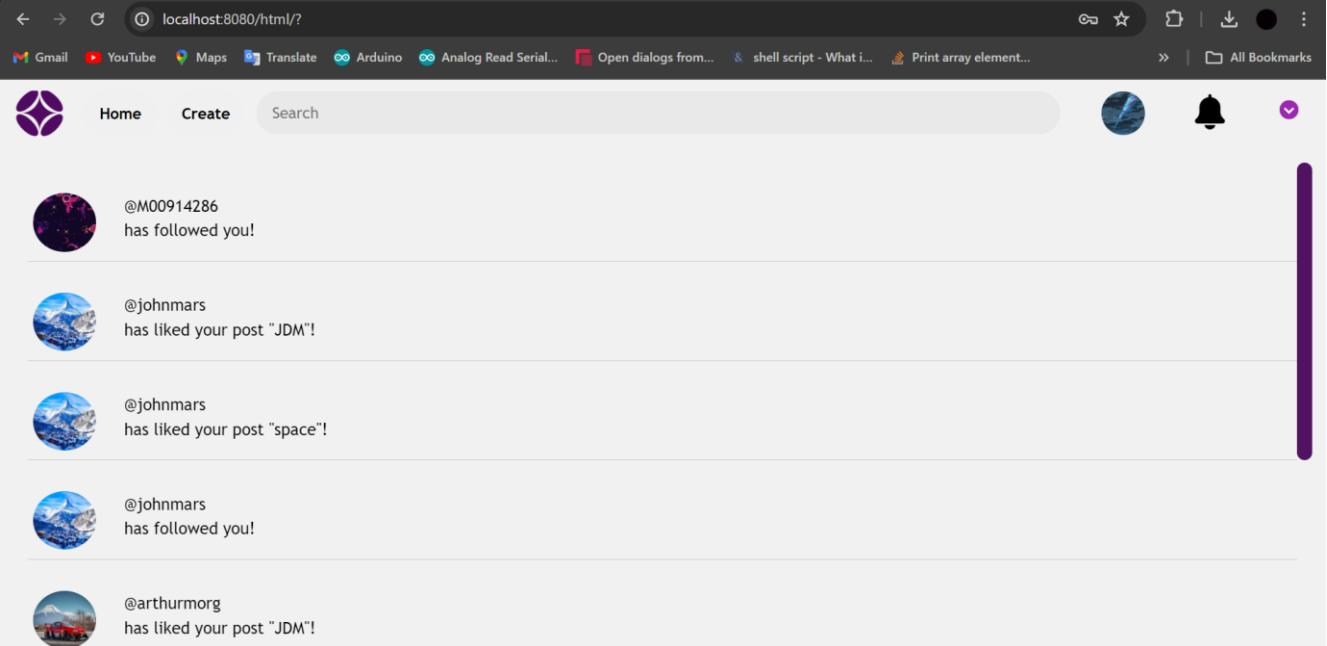


Edit Profile

The following outlines the profile edit process and the validation that takes place.

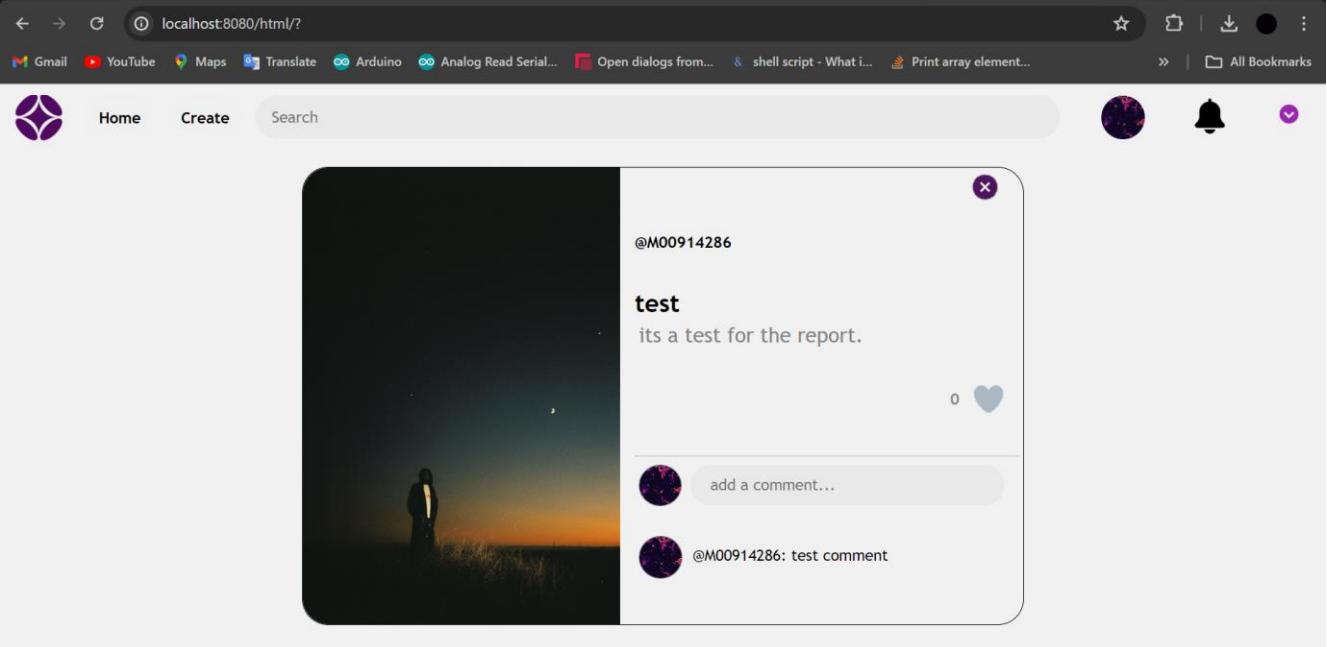
A screenshot of the MongoDB Compass interface. The left sidebar shows databases like "admin", "config", "cst2120", "local", "pixstar", and "pixstarCW2", with "users" selected. The main pane displays the "pixstarCW2.users" collection. It shows 8 documents and 1 index. A document is selected, showing fields such as "username: 'percy'", "password: 'e8913f9d518ff1f4fc3591b4b9c431c4'", "email: 'percy@yahoo.com'", "firstName: 'percy'", "lastName: 'smith'", "description: 'shoot for the stars.'", and "profilePhoto: 'iVBORw0KGgoAAAANSUhEUgAAQAAAAEACAYAAABccqhmAAAdnULEQVR4n02df4xdXHxDz...'". There are buttons for "ADD DATA" and "EXPORT DATA". The bottom status bar says "> MONGOSH".

Above is the document in the “users” collection for the “M00914286” user, while updating the profile photo the description and stored profile photo will be changed.



The screenshot shows a web browser window with the URL `localhost:8080/html/` in the address bar. The page displays a feed of notifications. Each notification includes a profile picture, a handle (@username), and a message indicating an action (e.g., followed you, liked your post). The notifications are as follows:

- @M00914286 has followed you!
- @johnmars has liked your post "JDM"!
- @johnmars has liked your post "space"!
- @johnmars has followed you!
- @arthurmorg has liked your post "JDM"!



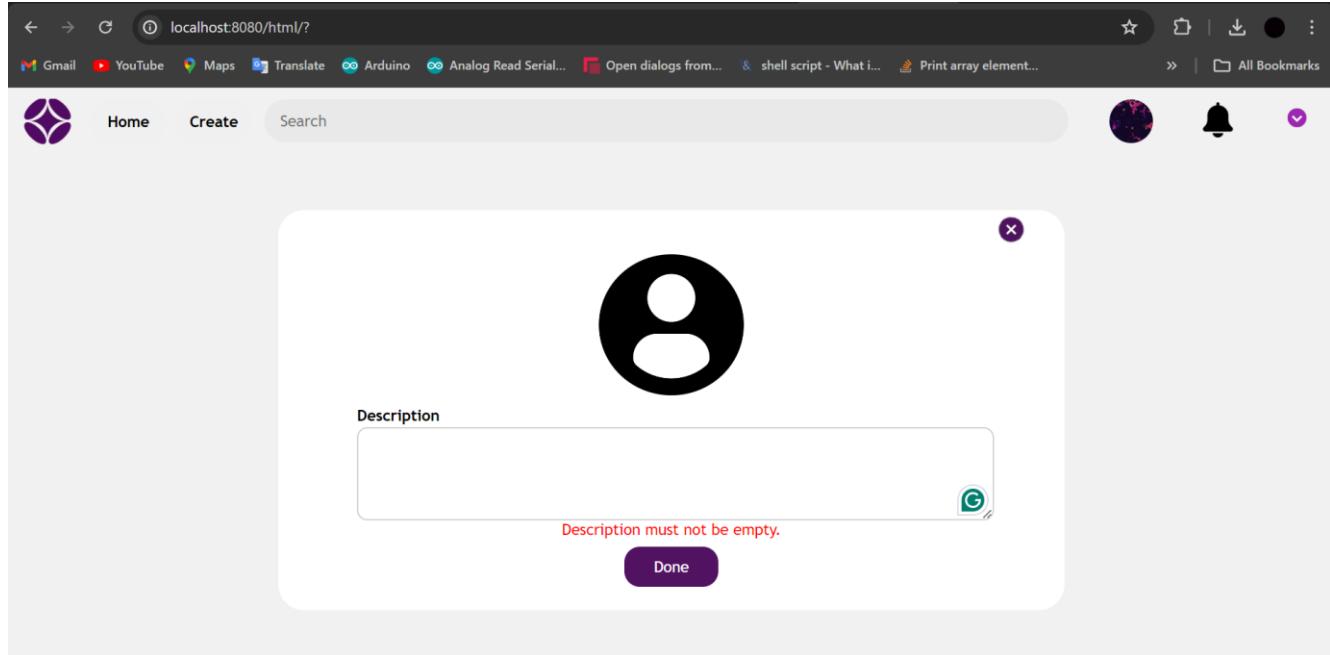
The screenshot shows a web browser window with the URL `localhost:8080/html/` in the address bar. The page displays a post detail view. On the left is a large image of a person standing in a field at sunset. To the right is a card containing the following information:

- Profile picture of the user: @M00914286
- Post title: test
- Post content: its a test for the report.
- Interaction count: 0
- Like icon: a heart
- Comment input field: add a comment...
- Comment history: @M00914286: test comment

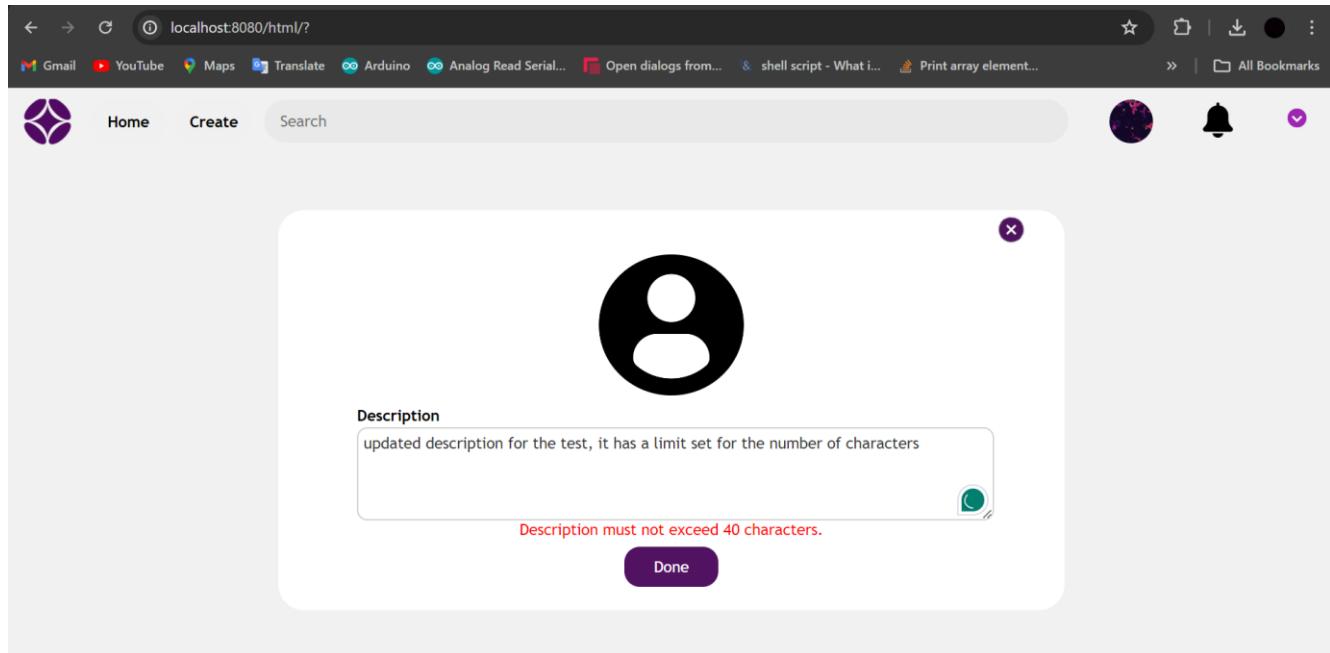
When a user changes their profile photo, all notifications and comments that

include the user are updated with the new profile photo. The above is before editing the profile.

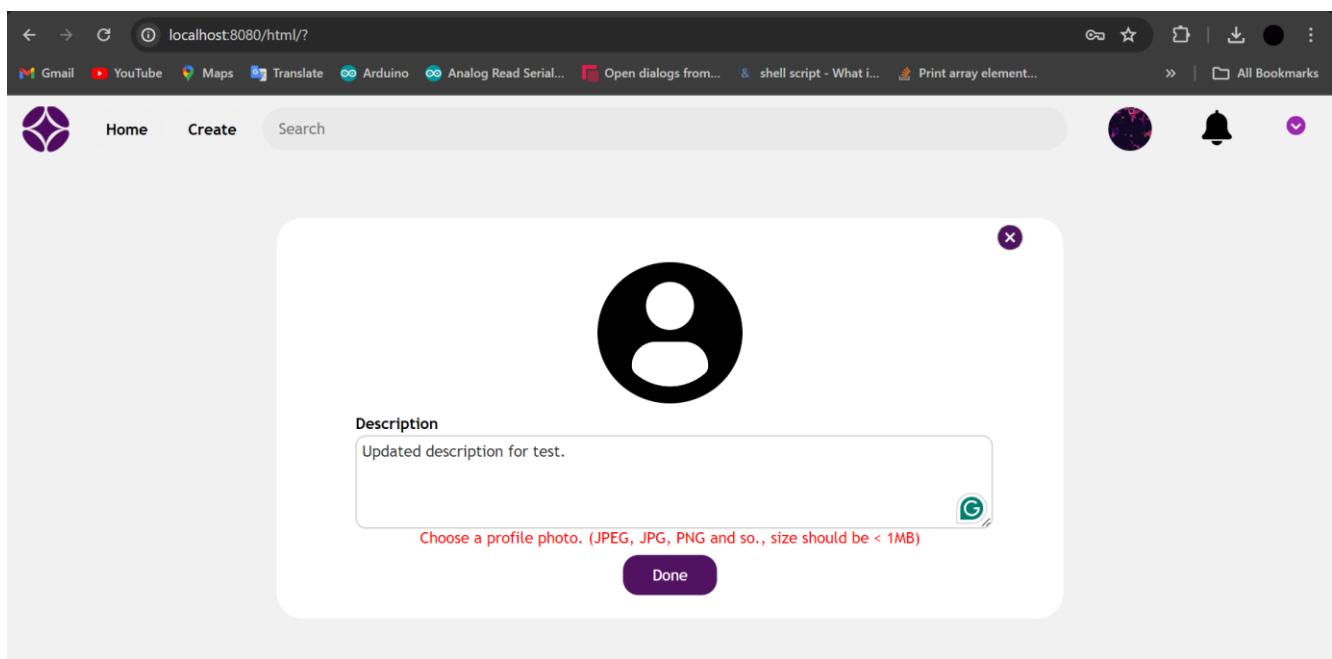
1) If the field is empty.



2) If the character limit has exceeded.

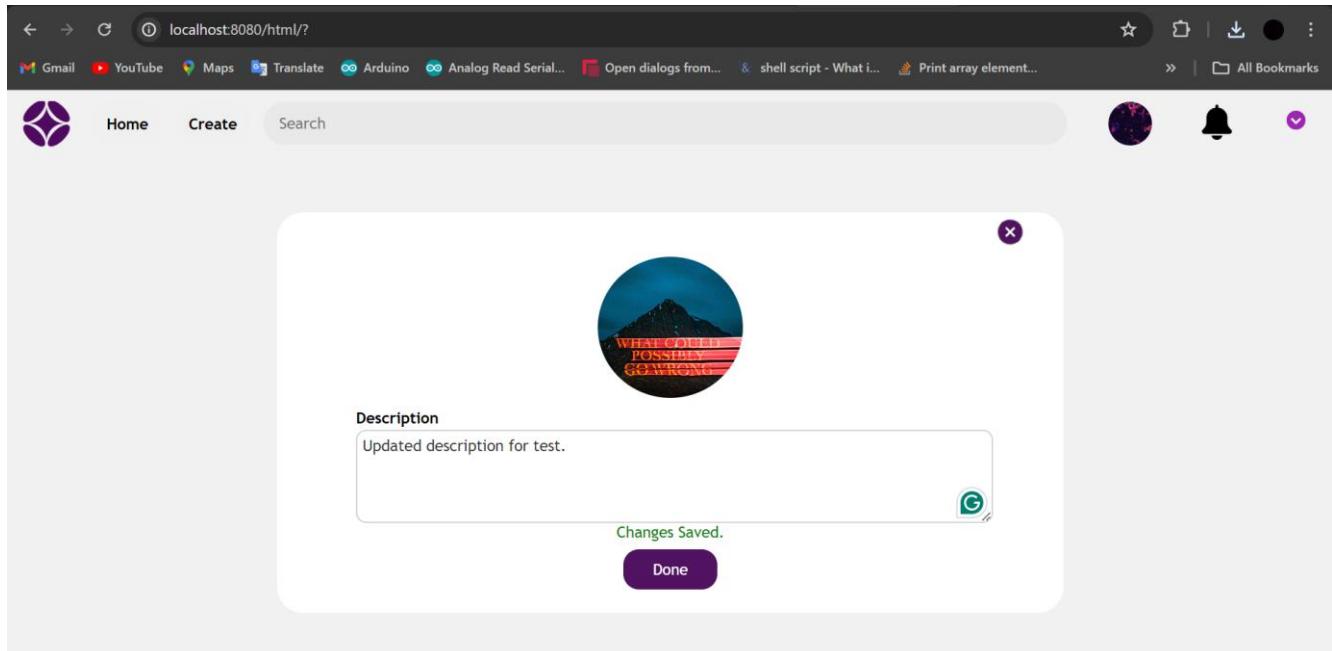


3. Profile photo not chosen.

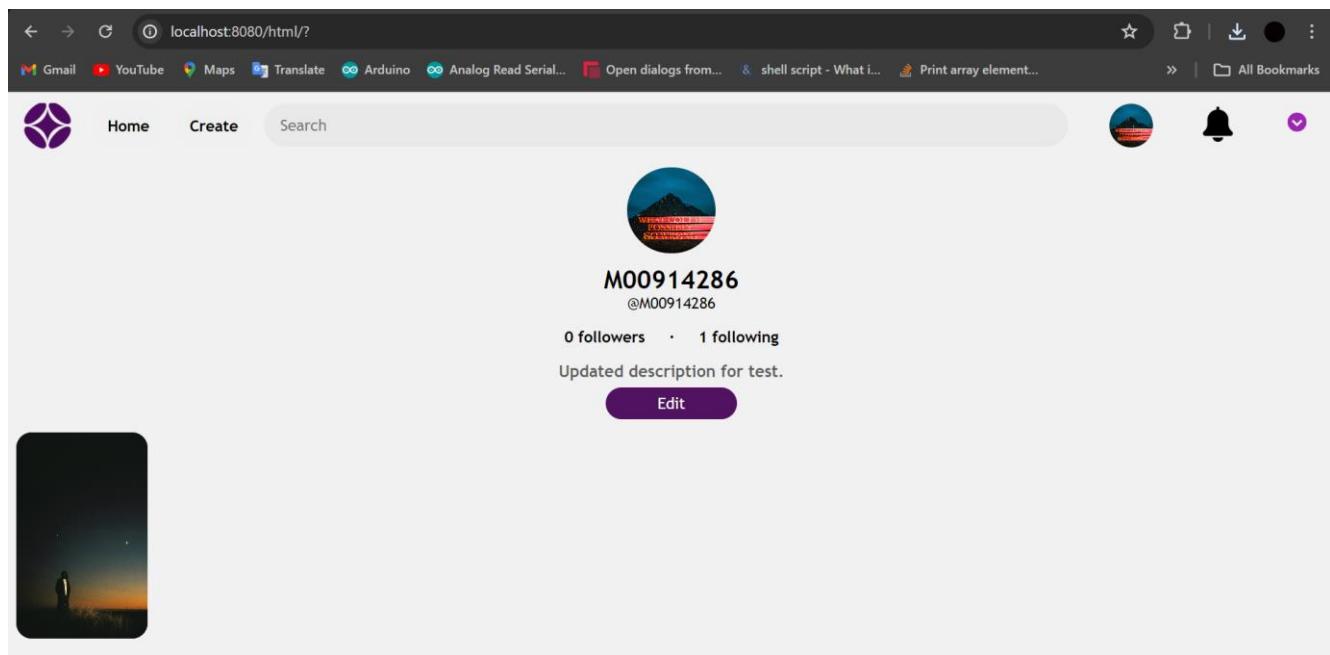


NOTE: if a user attaches a file that is not an image or exceeds one MB, it is not set and if the user presses submit the same error is shown.

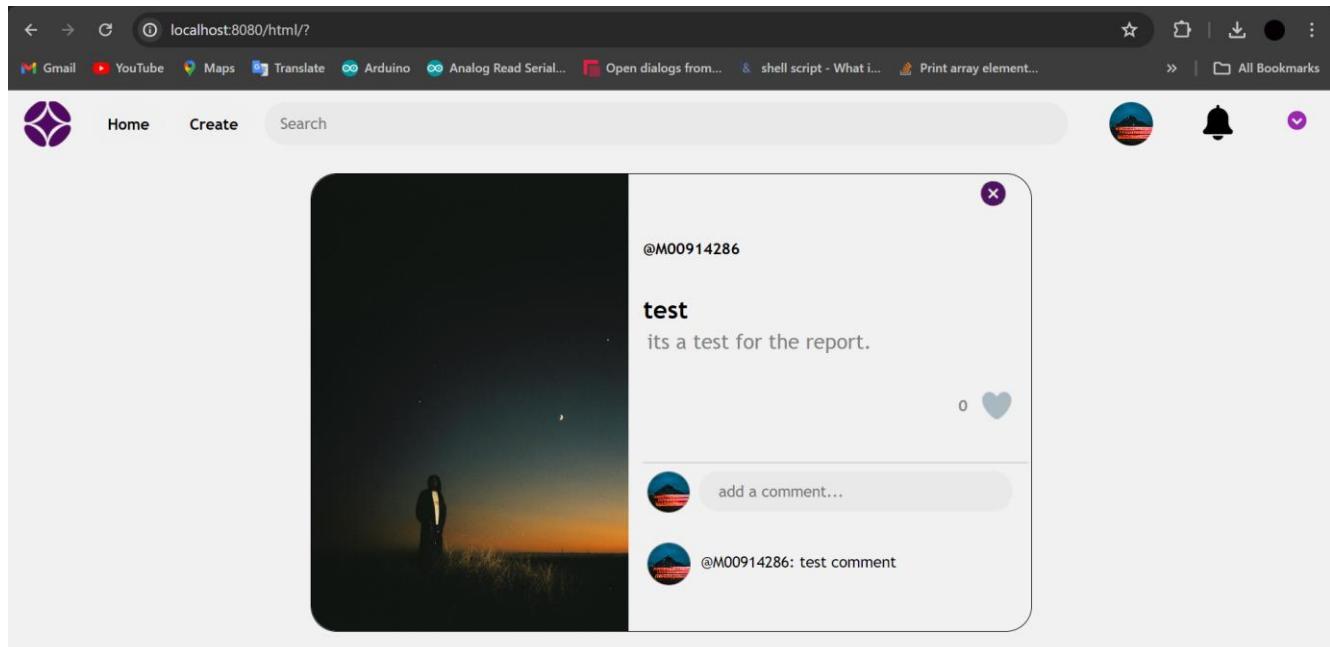
If all validation test are passed the user can change their profile photo and description successfully, any comments or notifications that included the user are updated with the new profile photo and can be seen below.



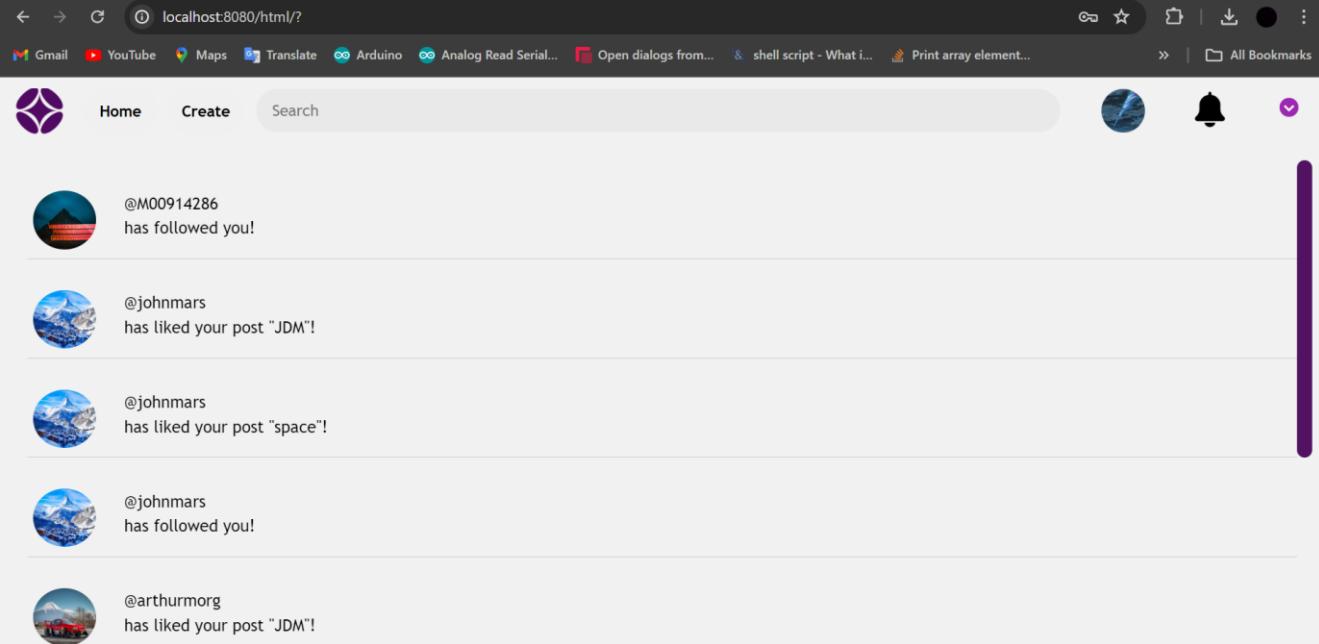
The user can then exit the section by clicking the purple “x” in the top right corner or edit the profile again.



The comment left by the user is updated with the new profile photo.



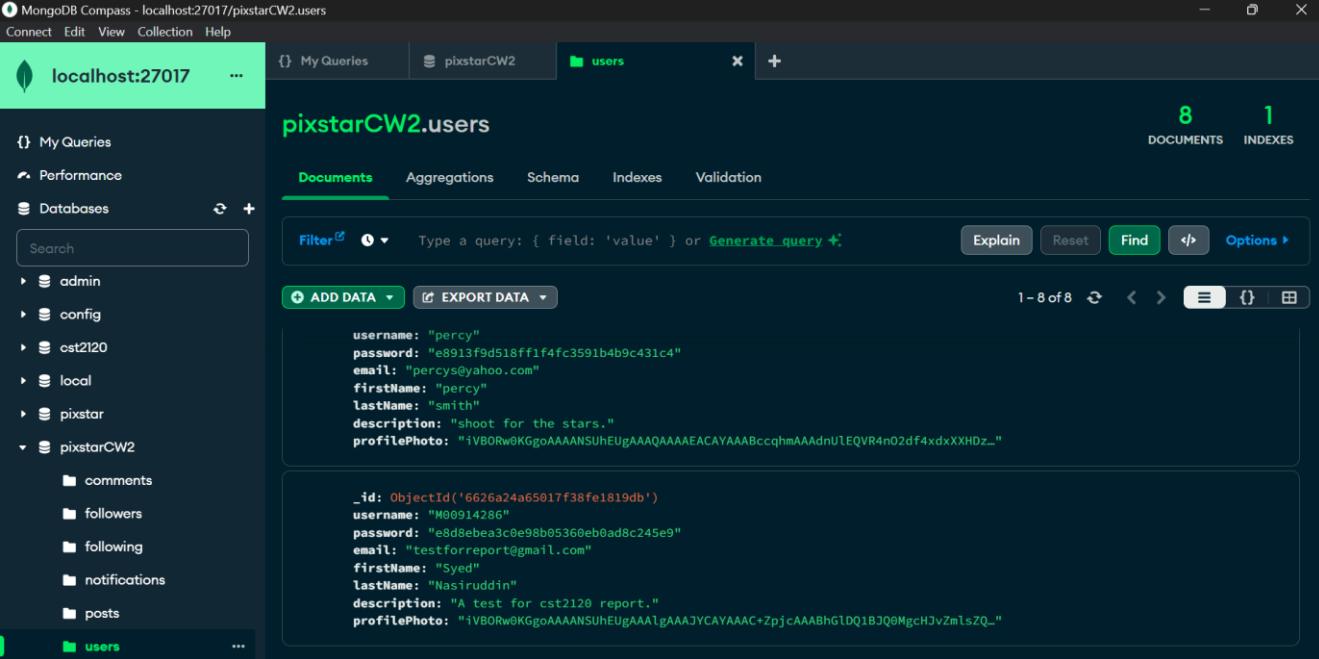
The user's profile photo in notifications has also been updated.



The screenshot shows a web browser window with a feed of notifications. Each notification includes a small profile picture next to the user handle and message. The notifications are as follows:

- @M00914286 has followed you!
- @johnmars has liked your post "JDM"!
- @johnmars has liked your post "space"!
- @johnmars has followed you!
- @arthurmorg has liked your post "JDM"!

The document in mongoDB in the users collection has been updated.



The screenshot shows the MongoDB Compass interface with the 'users' collection selected. It displays two documents with the following data:

```
username: "percy"
password: "e8913fd518ff1f4fc3591b4b9c431c4"
email: "percy@yahoo.com"
firstName: "percy"
lastName: "smith"
description: "shoot for the stars."
profilePhoto: "iVBORw0KGgoAAAANSUhEUgAAQAAAAEACAYAAABccqhmAAAdnU1EQVR4nO2dF4xdXXHDz="

_id: ObjectId('6626a24a65017f38fe1819db')
username: "M00914286"
password: "e8d8bea3c0e98b05360eb0ad8c245e9"
email: "testforreport@gmail.com"
firstName: "Syed"
lastName: "Nasiruddin"
description: "A test for cst2120 report."
profilePhoto: "iVBORw0KGgoAAAANSUhEUgAAAlgAAAJYCAYAAC+ZpjcAAABhGLDQ1BJQ0MgcHJvZmlsZQ=
```

the documents in the notifications and comments collections have also been updated with the user's new profile photo.

MongoDB Compass - localhost:27017/pixstarCW2.notifications

Connect Edit View Collection Help

localhost:27017 ...

My Queries Performance Databases +

Search

- admin
- config
- cst2120
- local
- pixstar
- pixstarCW2**
 - comments
 - followers
 - following
 - notifications**
 - posts
 - users

>_MONGOSH

pixstarCW2.notifications

18 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } or Generate_query ↗ Explain Reset Find Options ↗

ADD DATA EXPORT DATA

1 - 19 of 19

```
_id: ObjectId('6626945e65017f38fe1819da')
user: "nasir"
notifiedUser: "webdeveloper"
notificationText: "has liked your post \"firefox\"!"
userProfilePhoto: "/9j/4QACiRXhpZgAASUkqAAgAAAAAGABIBAwABAAAAAQAABoBBQABAAAAYgAAABsBBQABAA..."
```

```
_id: ObjectId('6626cd7d7f8a9082752ef4bf')
user: "nasir"
notifiedUser: "nasir"
notificationText: "has followed you!"
userProfilePhoto: "/9j/4QACiRXhpZgAATU0AKgAAAAgABQEaAAUAAAABAAAASg..."
```

```
_id: ObjectId('662655e53158ed69557672ef')
user: "M00914286"
commentedPostId: "662639ab86bb3ac3a8fc42b0"
text: "great one."
userPhoto: "/9j/4AAQSkZJRgABAQAAAQABAAAD/4TGIRXhpZgAATU0AKgAAAAgABQEaAAUAAAABAAAASg..."
```

1 - 3 of 3

```
_id: ObjectId('6626c1ea7f8a9082752ef4c0')
commentedPostId: "6626c55c7f8a9082752ef4bc"
user: "M00914286"
text: "test comment"
userPhoto: "/9j/4QACiRXhpZgAASUkqAAgAAAAABIBAwABAAAAAQAAAAAAAD/4QAC/9sAhAAMCAgNCQ..."
```

MongoDB Compass - localhost:27017/pixstarCW2.comments

Connect Edit View Collection Help

localhost:27017 ...

My Queries Performance Databases +

Search

- admin
- config
- cst2120
- local
- pixstar
- pixstarCW2**
 - comments
 - followers
 - following
 - notifications
 - posts
 - users

>_MONOOSH

pixstarCW2.comments

2 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } or Generate_query ↗ Explain Reset Find Options ↗

ADD DATA EXPORT DATA

1 - 3 of 3

```
user: "nasir"
text: "love this photo!"
userPhoto: "/9j/4AAQSkZJRgABAQAAAQABAAAD/4TGIRXhpZgAATU0AKgAAAAgABQEaAAUAAAABAAAASg..."
```

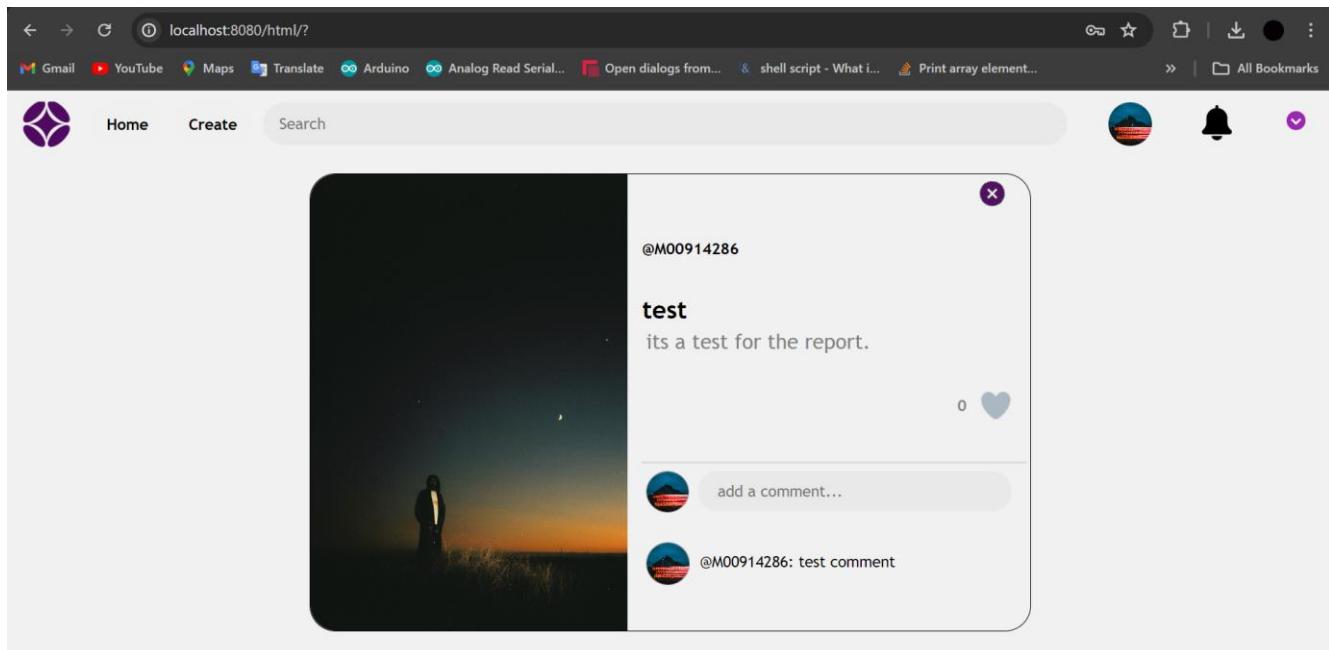
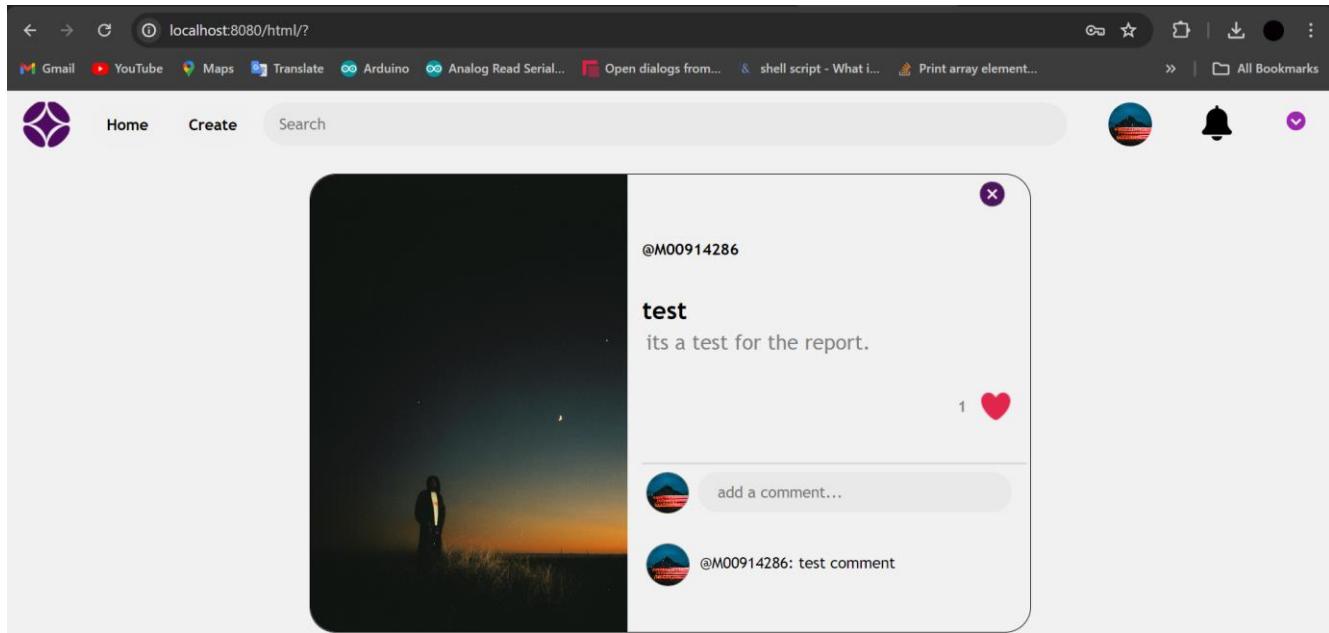
```
_id: ObjectId('662655e53158ed69557672ef')
commentedPostId: "662639ab86bb3ac3a8fc42b0"
user: "arthurmorg"
text: "great one."
userPhoto: "/9j/4AAQSkZJRgABAQAAAQABAAAD/4TGIRXhpZgAATU0AKgAAAAgABQEaAAUAAAABAAAASg..."
```

```
_id: ObjectId('6626c1ea7f8a9082752ef4c0')
commentedPostId: "6626c55c7f8a9082752ef4bc"
user: "M00914286"
text: "test comment"
userPhoto: "/9j/4QACiRXhpZgAASUkqAAgAAAAABIBAwABAAAAAQAAAAAAAD/4QAC/9sAhAAMCAgNCQ..."
```

Likes And Comments

Likes

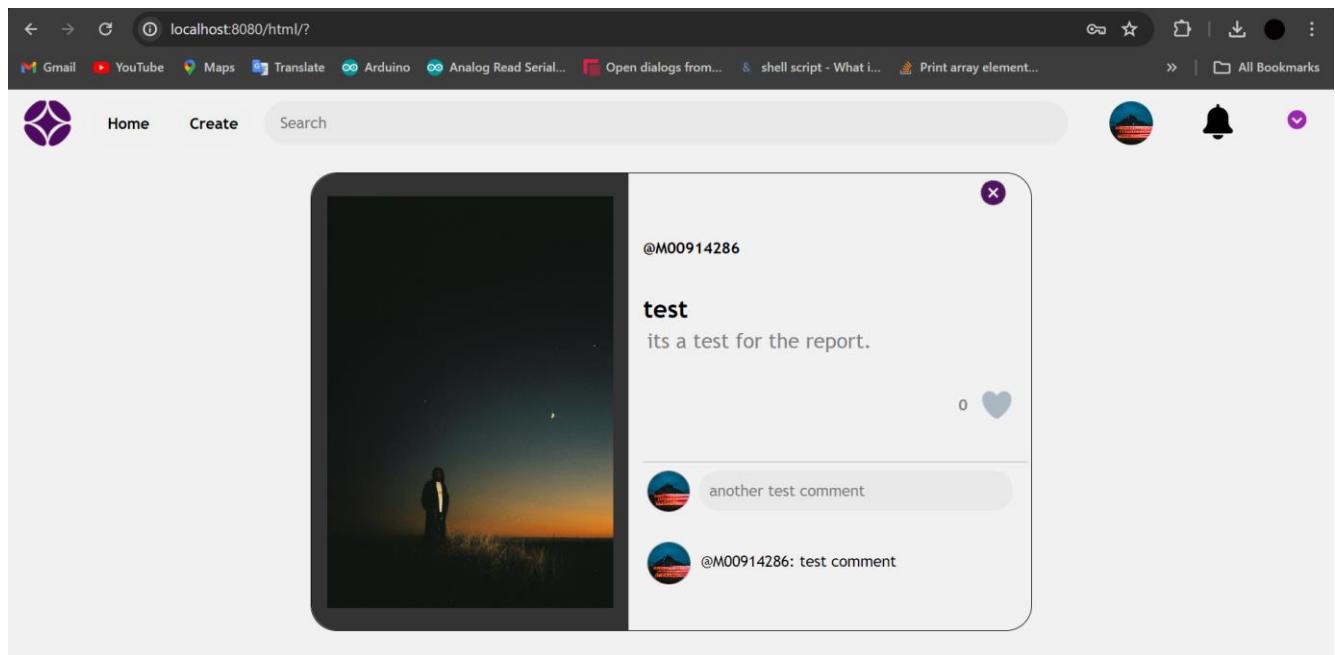
For the likes, whenever the user is viewing a post, they can click on the heart to like and unlike the post. The heart is filled when the user has liked the post, and empty when they haven't, the likes are adjusted accordingly.



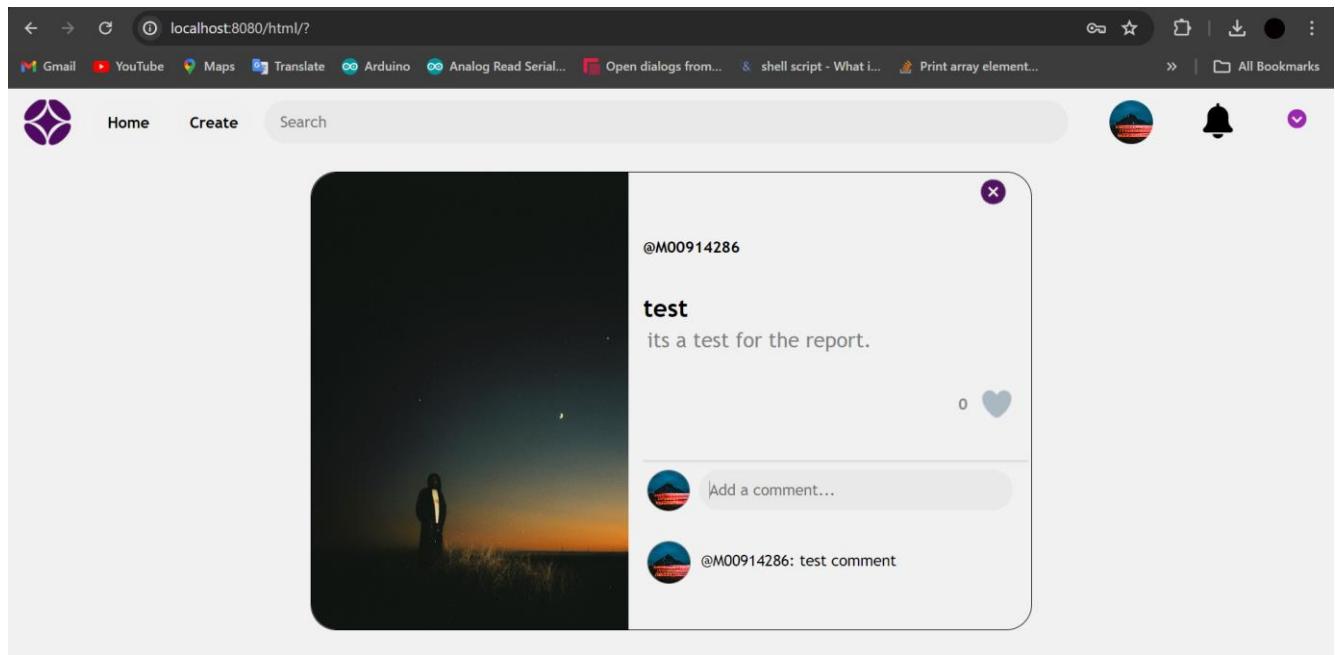
The state of the like button is determined everytime a user view a post by checking the likedArray in the mongoDB database.

Comments

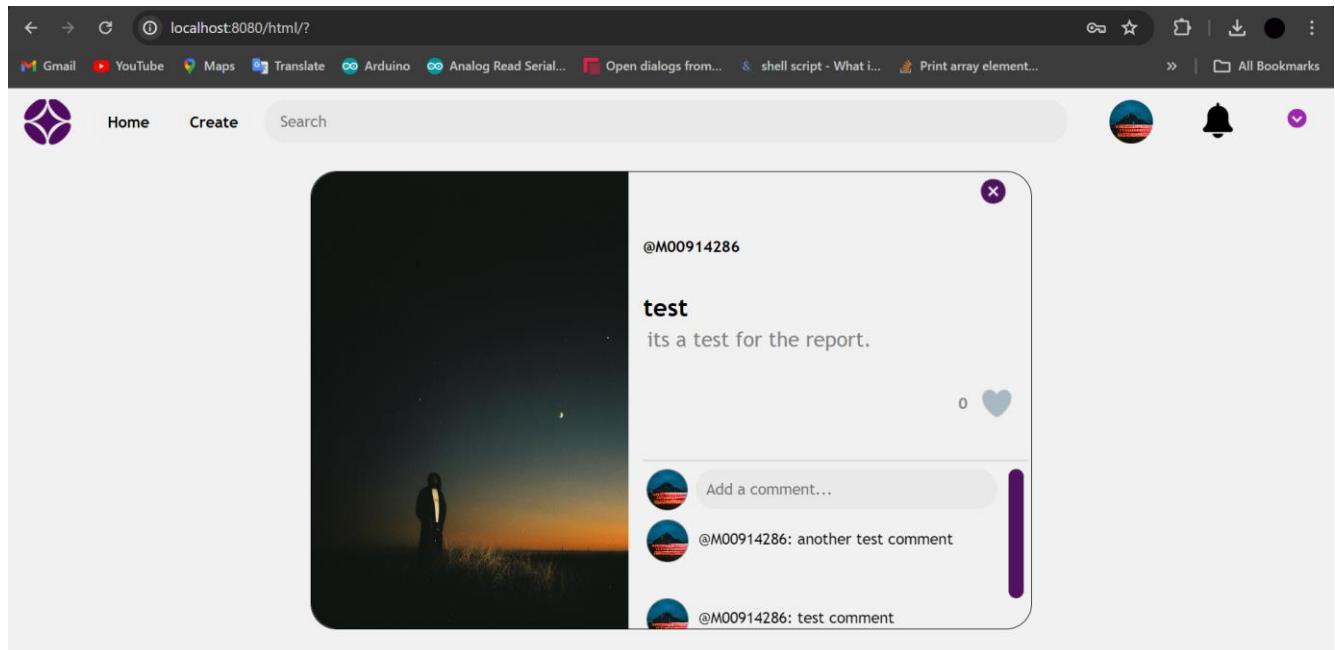
The user can comment on posts by typing their comment and hitting enter, to view their comments they can exit the post and open it again.



After hitting enter, the field is cleared and the user can write another comment, the user is free to comment how many ever time they like.



The comment appears after the user leaves and re-opens the post.



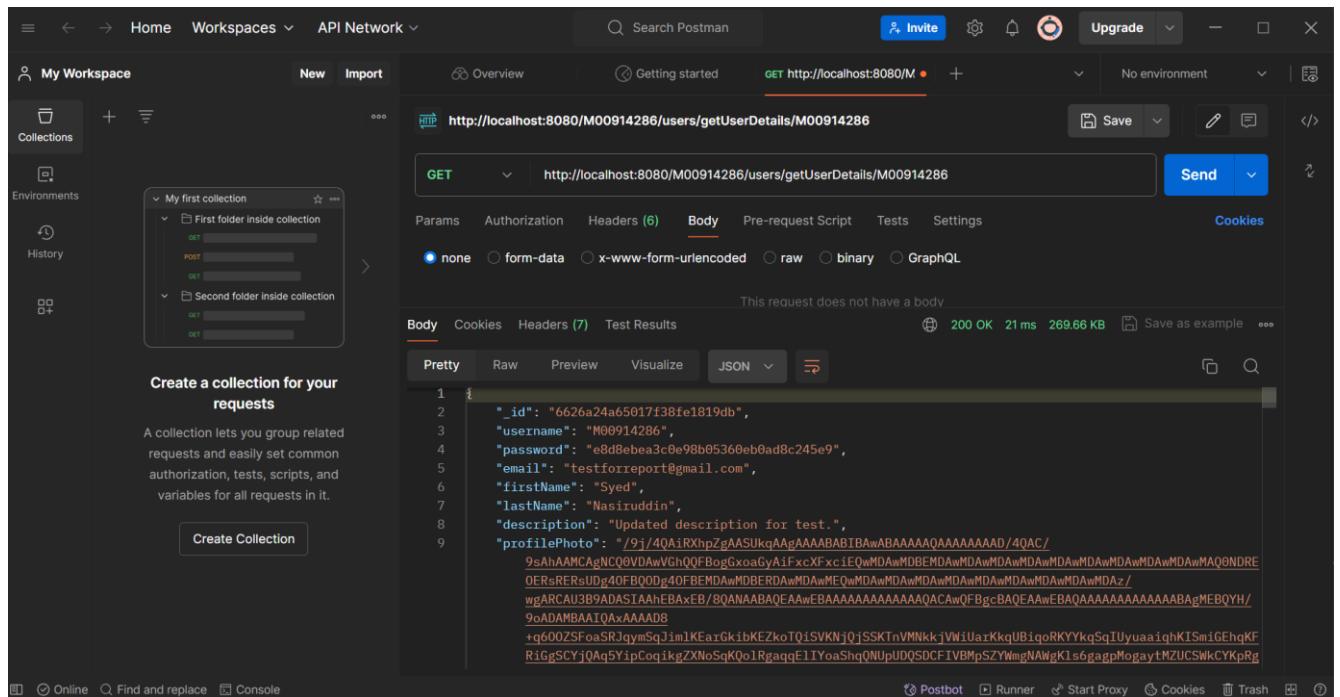
Webservices

Users Collection

The following webservices are related to the users collection:

1. `getUserDetails`

This webservice gets the provided user's details from the users collection.



2. searchUsers

This webservice searches for users with the provided username, even users with a part of the provided username in their username are returned, the users username a profile photo is returned.

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections for Collections, Environments, and History. A central panel displays a collection named "My Workspace" containing several requests grouped into folders like "First folder inside collection" and "Second folder inside collection". Below this, a section titled "Create a collection for your requests" explains what collections are used for. The main workspace shows a search result for the query "GET http://localhost:8080/M00914286/users/search/percy". The results table has one row with the following details:

Method	URL	Status	Time	Size
GET	http://localhost:8080/M00914286/users/search/percy	200 OK	17 ms	417.32 KB

The "Body" tab of the request details shows the JSON response:

```
1 {
2   "username": "percyjackson",
3   "profilePhoto": "9/j/"
4 }
```

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' (Collections, Environments, History), a central area for creating collections, and a detailed view of 'My first collection' containing 'First folder inside collection' and 'Second folder inside collection' with various requests. Below this, a section for creating a collection for requests is shown. To the right, the main workspace shows a request for 'GET http://localhost:8080/M00914286/users/search/percy'. The 'Body' tab is selected, displaying a JSON response with 'username' and 'profilePhoto' fields. Other tabs like Params, Authorization, Headers, and Cookies are also visible.

In the above screenshots, percy is provided and both “percy” and “percyjackson” are returned, this webservice is used for the users search functionality provided by the search bar.

3. checkExistingUser

checks if the provided username in the parameters is already taken by another user in the collection, if yes it sends the below response.

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar is visible with 'My first collection' expanded, showing several requests. The main panel displays a GET request to `http://localhost:8080/M00914286/users/check/M00914286`. The 'Body' tab is selected, showing the response body: `1 "Username already exists!"`. The status bar at the bottom indicates a 409 Conflict response.

if not, then it sends the following response,

The screenshot shows the Postman application interface. The setup is identical to the previous one, but the response body is now `1 "OK"`. The status bar at the bottom indicates a 200 OK response.

4. checkExistingEmail

checks if the provided email in the parameters is already taken by another user in the collection, if yes it sends the below response.

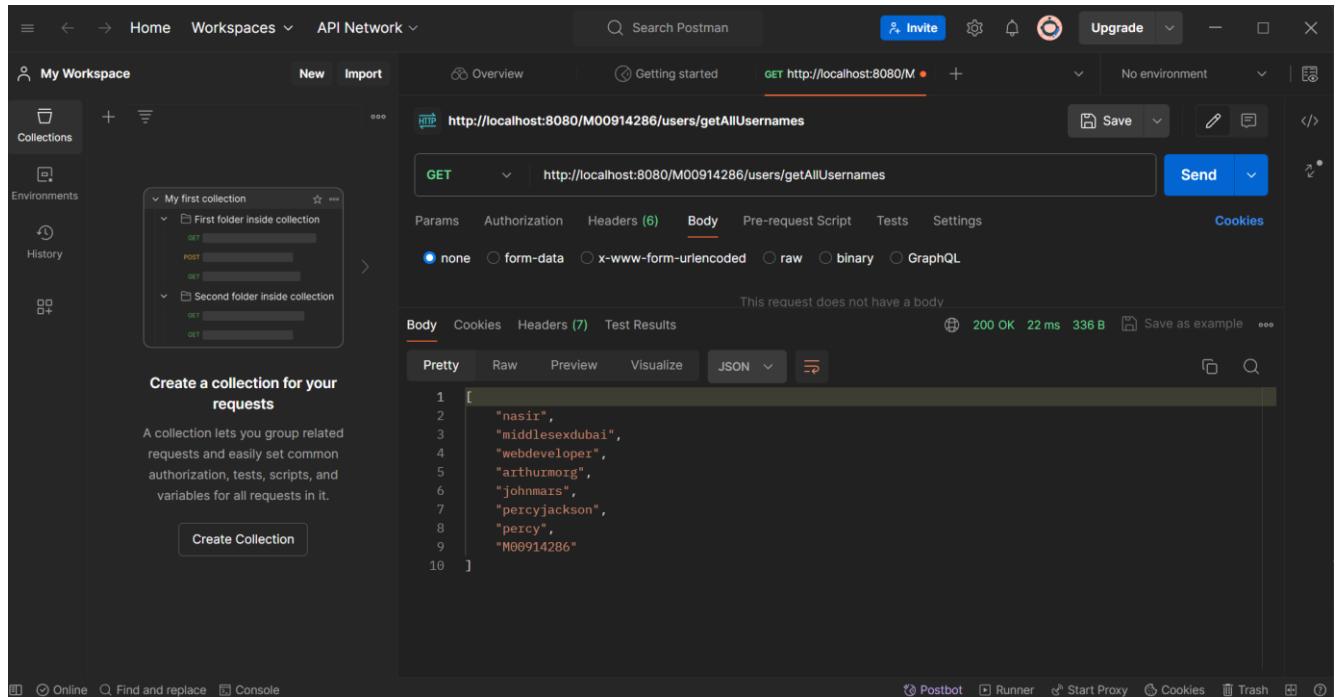
The screenshot shows the Postman interface with a dark theme. On the left, the 'My Workspace' sidebar displays a collection named 'My first collection' containing several requests. The main panel shows a request to 'http://localhost:8080/M00914286/users/checkEmail/testforreport@gmail.com' using a GET method. The 'Body' tab is selected, showing 'none' as the type. The 'Test Results' section indicates a 409 Conflict status with a response body of '1 "Email already in use"'.

If not, then it sends the following response,

The screenshot shows the Postman interface with a dark theme. The 'My Workspace' sidebar is identical to the previous screenshot. The main panel shows a request to 'http://localhost:8080/M00914286/users/checkEmail/anotheritestforreport@gmail.com' using a GET method. The 'Body' tab is selected, showing 'none' as the type. The 'Test Results' section indicates a 200 OK status with a response body of '1 "OK"'.

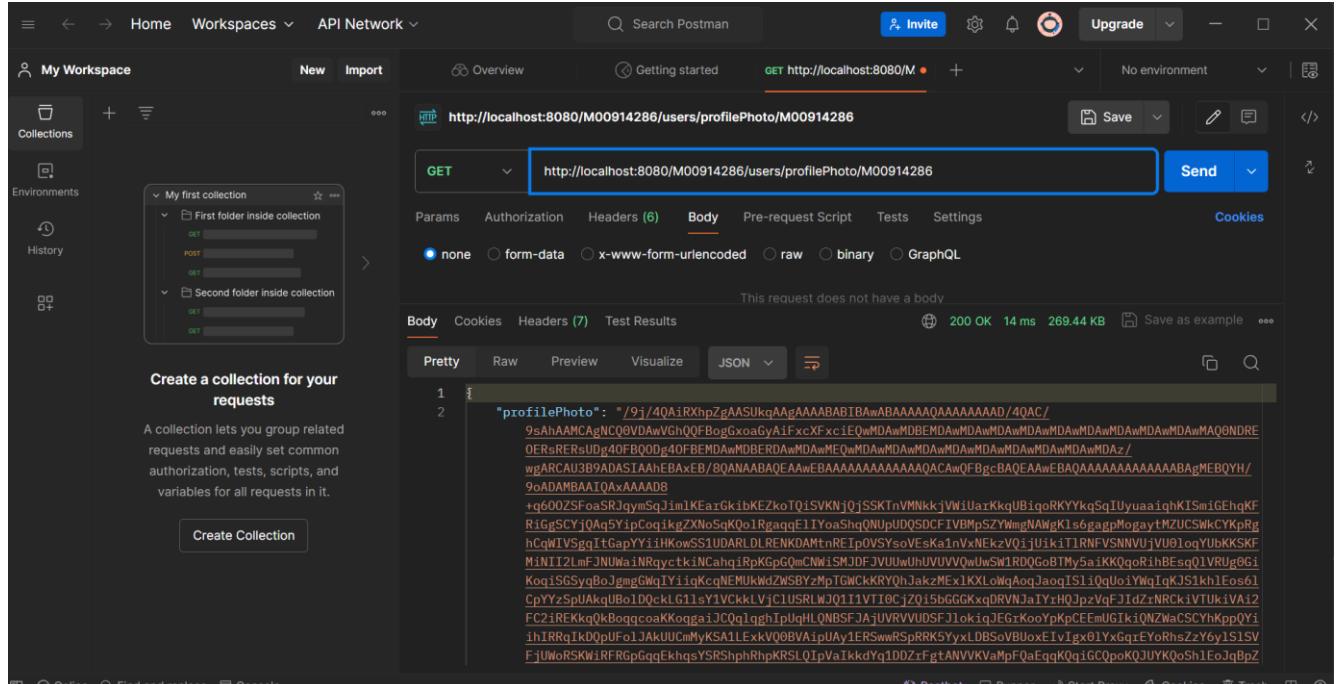
5. getAllUsernames

gets all the usernames of the users stored in the collection.



6. getUserProfilePhoto

gets the provided user's profile photo.



7. `updateUserDescription`
updates the provided user's description.

The screenshot shows the Postman application interface. On the left, the sidebar includes 'My Workspace' (with 'Collections', 'Environments', and 'History' sections), a search bar, and an 'Invite' button. The main area displays a collection named 'My first collection' containing two folders: 'First folder inside collection' and 'Second folder inside collection'. Each folder contains several requests (GET and POST). Below the collection list, there's a section titled 'Create a collection for your requests' with a note about grouping related requests and a 'Create Collection' button. The central workspace shows a POST request to 'http://localhost:8080/M00914286/users/updateDescription'. The 'Body' tab is selected, showing a JSON payload:

```
1 {
2   ...
3   "username": "M00914286",
4   "description": "webservice test"
5 }
```

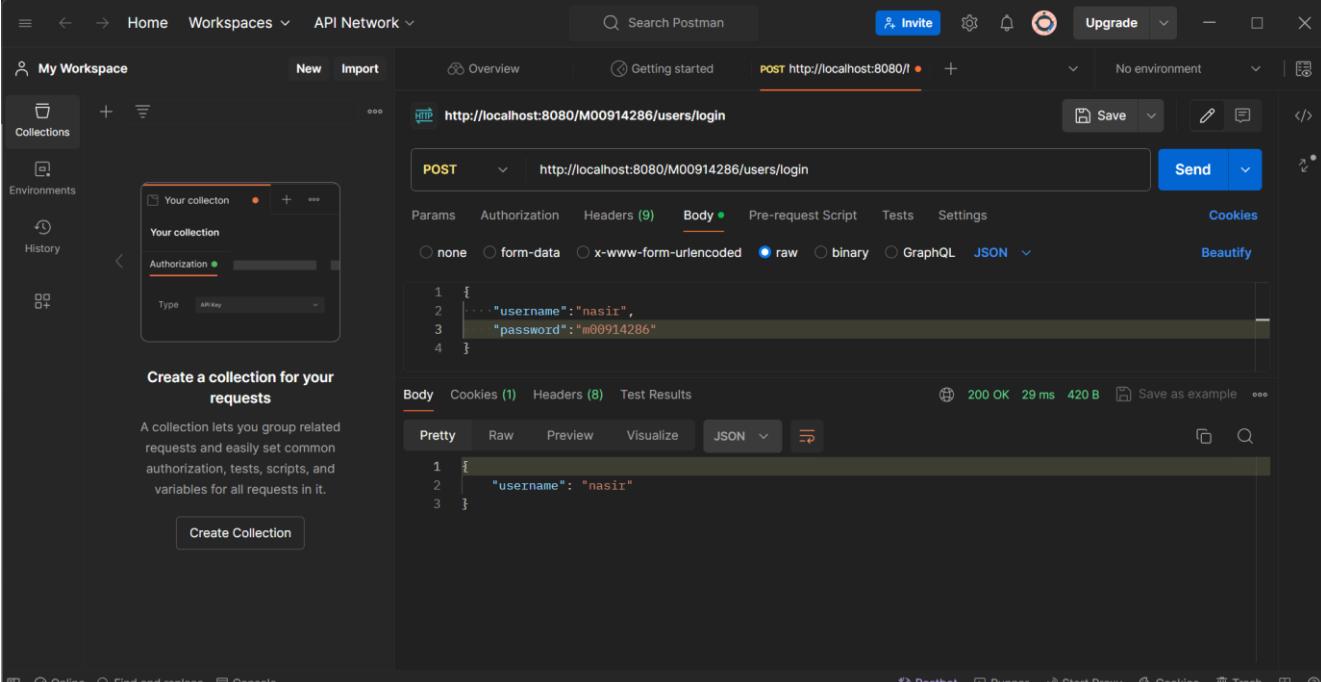
The response section shows a successful 200 OK status with 13 ms latency and 317 B size. The response body is:

```
1 {
2   "postId": "M00914286",
3   "success": true,
4   "message": "Description updated successfully"
5 }
```

the update can be seen below,

8. checkUser

This webservice is used for login, it takes a username and password, checks if the provided password matched the decrypted stored password, if so it returns the stored username.



The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing a collection named 'Your collector' which includes a single request named 'Your collection'. The main panel shows a POST request to 'http://localhost:8080/M00914286/users/login'. The 'Body' tab is selected, showing the following JSON payload:

```
1 {
2   ...
3   "username": "nasir",
4   ...
5   "password": "#00914286"
6 }
```

Below the request, the response status is shown as 200 OK with 29 ms and 420 B. The response body is also displayed as:

```
1 {
2   ...
3   "username": "nasir"
4 }
```

Excluded Webservices

For the following webservices I have not used postman:

- **signupPost**
- **updateProfilePhoto**
- **updateUserProfilePhoto**

The first two work alongside each other, they are passed as request 1 and request 2 to the postToDatabase function, they are used for the creation of a user, signupPost first adds all the text data to the document and then the document is updated with profile photo provided during the sign-up process.

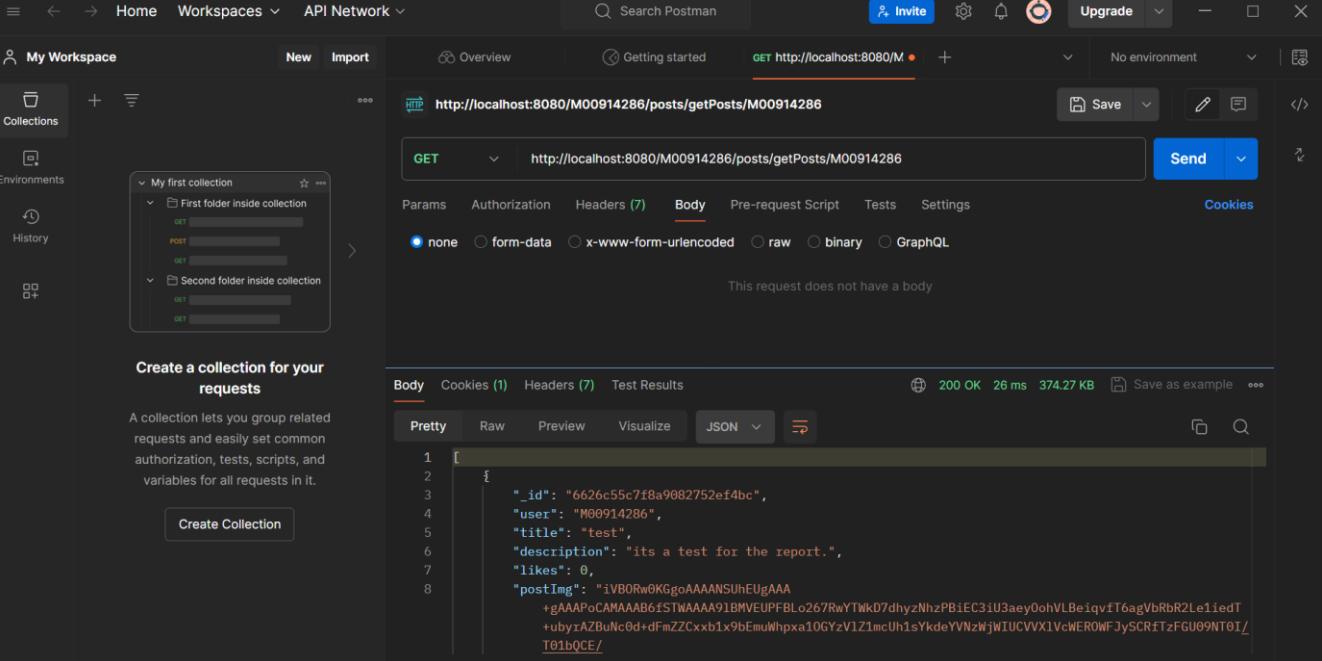
The last of the three, is used when the user is editing their profile, both the profile edit and sign up process will be covered in the video.

Posts Collection

The below webservices are related to the posts collection:

1. getUserPosts

gets the provided users posts that they made.



The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar displays 'Collections' (with 'My first collection' expanded), 'Environments', and 'History'. A 'Create a collection for your requests' section is present. In the center, a request card is shown for a GET request to `http://localhost:8080/M00914286/posts/getPosts/M00914286`. The 'Body' tab is selected, showing the response body as a JSON object:

```
1 [ { 2   "_id": "6626c55c7f8a9082752ef4bc", 3   "user": "M00914286", 4   "title": "test", 5   "description": "its a test for the report.", 6   "likes": 0, 7   "postImg": "iVBORw0KGgoAAAANSUhEUgAAA  +gAAAPoCAMAAB6fSTWAAA91BMVEUPFBLo267RwYTlkD7dhyzNhzPBiEC3iU3aey0ohVLBeiqvFT6agVbRbR2Le1iedT  +ubyxAZBuNc0d+dFmZZCxxb1x9bEmuWhpxa10GYzV1Z1mcUh1sYkdeYVNZWjWIUCVVX1VcWEROWFJySCRftzFGU99NT0I/  T01bQCE/" } ]
```

The response status is 200 OK with 26 ms latency and 374.27 KB size. The 'Pretty' tab is selected in the preview area.

2. `getPostDetailsbyID`

gets a post's details by the value of its `_id` attribute.

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing 'Collections' (selected), 'Environments', 'History', and 'API Keys'. A central panel displays a collection titled 'Your collection' with an 'Authorization' tab selected. Below it, a 'Create a collection for your requests' section is shown with a 'Create Collection' button. The main workspace shows an 'Overview' tab, a 'Getting started' guide, and a request card for a GET request to `http://localhost:8080/M00914286/posts/getPostByID/6626c55c7f8a9082752ef4bc`. The 'Body' tab is active, showing the response body as JSON:

```
1 {
2   "_id": "6626c55c7f8a9082752ef4bc",
3   "user": "M00914286",
4   "title": "test",
5   "description": "its a test for the report.",
6   "likes": 0,
7   "postImg": "iVBORw0KGgoAAAANSUhEUgAAA
     +gAAPoCAMAAB6fSTWAAA91BMEUPFBL0267RwYTWkD7dhyzNhzPBiEC3iU3aeyOohVLBeiqvft6agVbRbR2LeiledT
     +ubyAZBnCg0d+dfmZZCxb1x9bEmuWhpxa10GyV1Z1mcUh1sYkdeYVNzkhWIUCVVXlVcWEROWFJySCRfzFGU09NT0I/
     T01bYzCE/
     SEE3SkpGrjY3REEwREuxPz8P0EzPddCNh8q0jsnNjcvMyo1MjM0Lh0j1jAnLCYhKiwgJxsfJygkIxgd1yUDISIeHxcaHyAZB
```

3. checkIfUserLikedPost

checks if the provided user is in the likedBy array of the post that has the provided post Id.

The screenshot shows the MongoDB Compass interface. The left sidebar lists databases and collections, with 'pixstarCW2' selected. The main area displays the 'posts' collection under the 'pixstarCW2.posts' tab. The 'Documents' tab is active, showing a single document with the following fields and values:

Field	Type
_id	ObjectId
user	String
title	String
description	String
likes	Int32
postImg	String
likedBy	Array

The 'likedBy' array contains two elements: 'johnmars' and 'nasir'. The document ID is '66263e1686bb3ac3a8fc42b5'.

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar is visible with sections for Collections, Environments, and History. A 'Create a collection for your requests' section is present with a 'Create Collection' button. The main workspace shows a request card for a GET request to `http://localhost:8080/M00914286/posts/checkIfUserLikedPost/nasir/66263e1686bb3ac3a8fc...`. The 'Body' tab is selected, showing the response body as JSON:

```
1 {
2   "likedByUser": true,
3   "currentLikes": 2
4 }
```

4. getUserLikedPosts

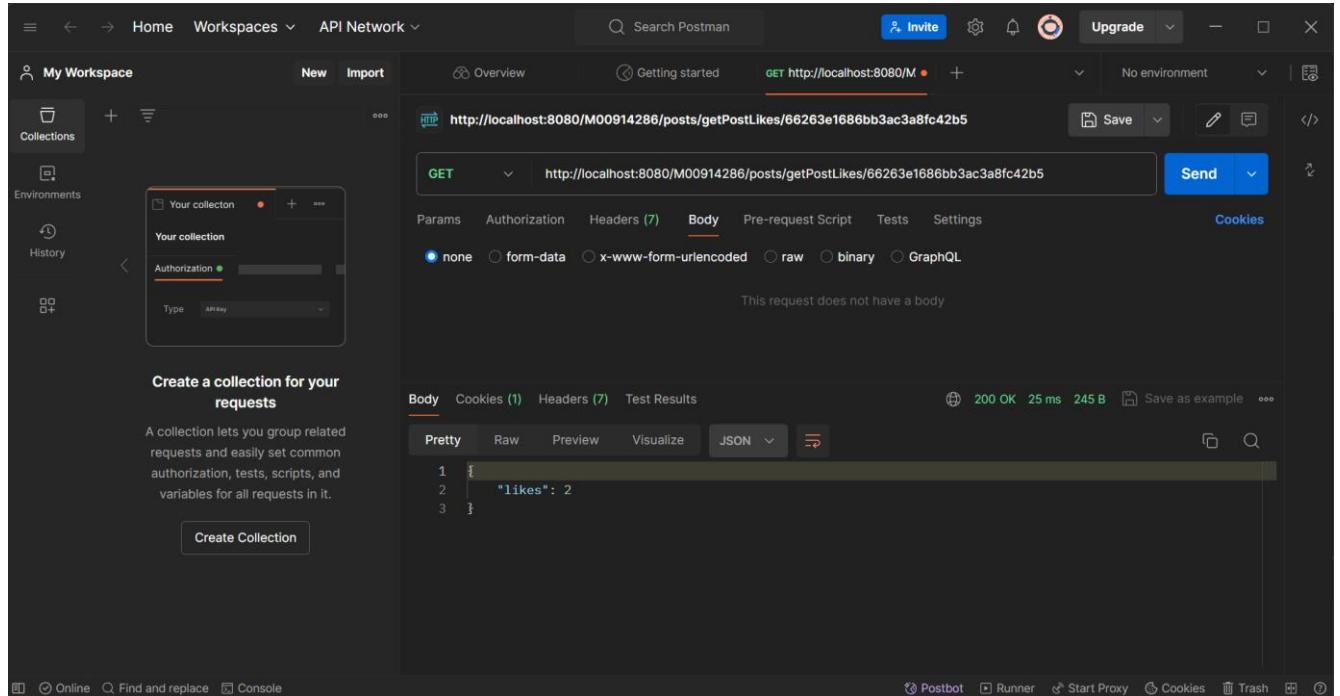
gets the post Ids of the posts that the provided user has liked.

The screenshot shows the Postman application interface. The left sidebar is identical to the previous screenshot. The main workspace shows a request card for a GET request to `http://localhost:8080/M00914286/posts/userLikedPosts/nasir`. The 'Body' tab is selected, showing the response body as JSON:

```
1 {
2   "likedPosts": [
3     "66263e1686bb3ac3a8fc42b5"
4   ]
5 }
```

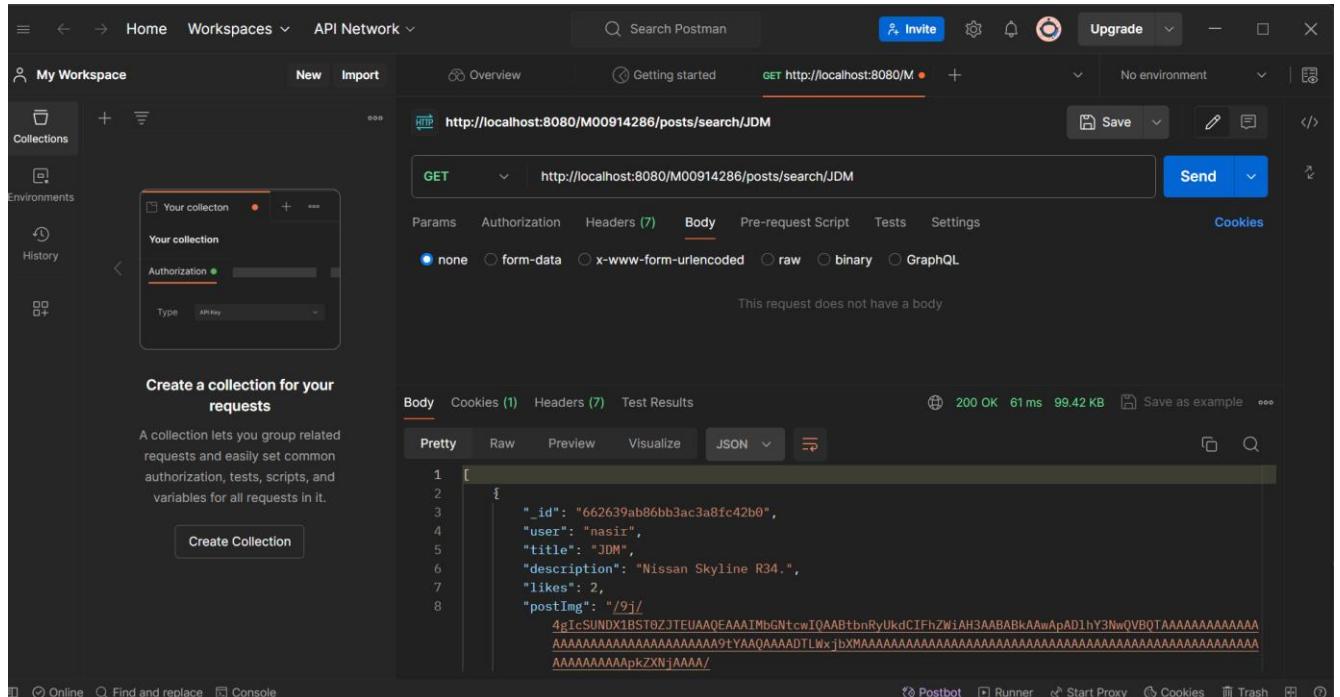
5. getPostLikes

gets the like count of the post that has the provided post Id.



6. searchPostsByTitle

gets posts that have the provided word/phrase in their title.



7. updatePostLikes

updates the post's current liked by adding the adjustment provided, also takes the user and adds them to the likedBy array of the post.

The screenshot shows the Postman interface. The left sidebar displays 'My Workspace' with a collection named 'Your collection'. The main area shows a POST request to `http://localhost:8080/M00914286/posts/updatePostLikes/662639ab86bb3ac3a8fc42b0`. The 'Body' tab contains the following JSON payload:

```
1 {
2   ...
3   "username": "percy",
4   "adjustment": 1
}
```

The response status is 200 OK with a response body:

```
1 {
2   "success": true,
3   "message": "Likes adjusted successfully",
4   "updatedLikes": 3
}
```

The screenshot shows the MongoDB Compass interface connected to `localhost:27017/pixstarCW2.posts`. The left sidebar shows databases like admin, config, cst2120, local, pixstar, and pixstarCW2. The 'posts' collection is selected. The right pane shows the document structure:

```
title: "space"
description: "so vast and unexplored."
likes: 1
postImg: "/9j/4AAQSkZJRGABAQAAAQABAAAD/4TGIRXhpZgAATU0AKgAAAAgABQEaAAUAAAABAAAASg..."
likedBy: Array (1)
```

Below this, another document is shown:

```
_id: ObjectId('662639ab86bb3ac3a8fc42b0')
user: "nasiir"
title: "DDM"
description: "Nissan Skyline R34."
likes: 3
postImg: "/9j/4gICUNDX1BSt0ZJTEUAQAAA1MbGNTcwIQAA8tbnRyUkdC1FhZWiaH3AA8ABkAAw..."
likedBy: Array (3)
0: "arthurmorg"
1: "johnmars"
2: "percy"
```

Excluded Webservices

The following webservices have not been documented using postman:

- **makePost**
- **updatePostImage**

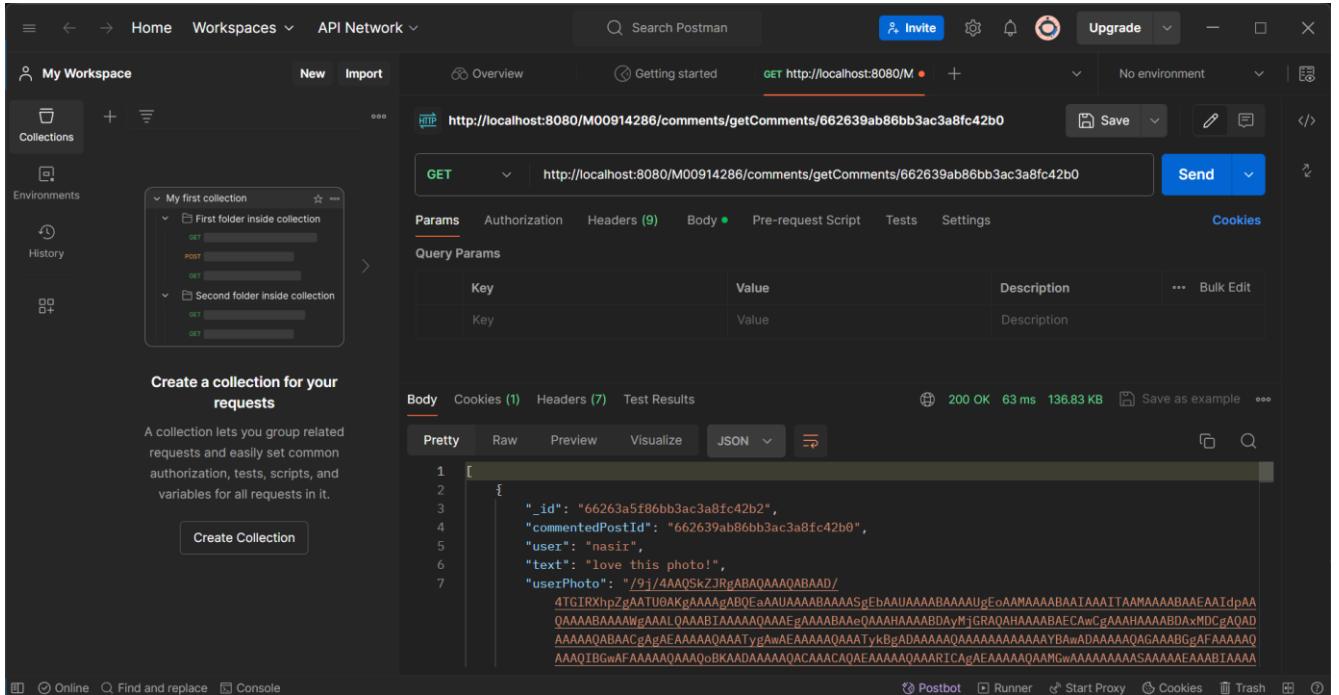
Both of these webservices are involved in the post creation process and are passed as request 1 and request 2 to the postToDatabase function, the post creation process will be covered in the video, a successful creation of the post will indicate the correct working of these webservices.

Comments Collection

These webservices are related to the comments collection:

1. getComments

gets all the comments made for the post that has the provided post Id.



The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing 'Collections', 'Environments', and 'History'. A 'Create a collection for your requests' section is visible. In the center, a request is being made to the URL `http://localhost:8080/M00914286/comments/getComments/662639ab86bb3ac3a8fc42b0`. The 'Params' tab is selected, showing a table with one row: 'Key' (empty) and 'Value' (empty). The 'Body' tab is also selected, showing a JSON response:

```
1 {
2   "_id": "66263a5186bb3ac3a8fc42b2",
3   "commentedPostId": "662639ab86bb3ac3a8fc42b0",
4   "user": "nasir",
5   "text": "love this photo!",
6   "userPhoto": "/9j/4AAQSkzJRgABAQAAAQABAAAD/4TGIRXhpZgAATU0AkgAAAAgABQEaAAUAAAABAAAASgEbAAUAAAABAAAUGEoAAMAAAABAAIAAAITAAAMAAAABAAEAAIdpAAQAAAABAAAAbwAAALQAAABIAAAAQAAAEGAAAABAAeQAAHMAAAABDyMjGRAQAHAAAABAECAwCgAAHMAAAABDxMDCgAQQDAAAAQABAAACgAEAAAAAAQAAAATygbgADAAAAAAQAAAQAAAQAAAQAAAAYBawADAAAAAQAGAAAABgAfFAAAAAQAAAQIBGwAFAAAAQAAAQoBKAADAAAAQACAAACAQAEAAAQAAARICAQgAEAAAAAQAMGwAAAAAAAASAAAAEAAABIAAAA
```

The screenshot displays two applications side-by-side: Postman (top) and MongoDB Compass (bottom).

Postman:

- Left Panel:** Shows "My Workspace" with a collection named "My first collection". This collection contains two folders: "First folder inside collection" and "Second folder inside collection". Each folder has a GET request.
- Right Panel:**
 - Request URL:** http://localhost:8080/M00914286/comments/getComments/662639ab86bb3ac3a8fc42b0
 - Method:** GET
 - Params:** Authorization, Headers (9), Body, Pre-request Script, Tests, Settings
 - Query Params:** Key, Value, Description
 - Body:** Shows a JSON response with fields like _id, commentedPostId, user, text, and userPhoto. The text field contains a large string of placeholder text.
 - Test Results:** Status 200 OK, 63 ms, 136.83 KB

MongoDB Compass:

- Left Panel:** Shows the database structure with collections: config, cst2120, local, pixstar, and pixstarCW2. The "comments" collection is selected.
- Right Panel:**
 - Documents:** Shows two documents in the "comments" collection.
 - Fields:**
 - Document 1: _id: ObjectId('66263a5f86bb3ac3a8fc42b2'), commentedPostId: "662639ab86bb3ac3a8fc42b0", user: "nadir", text: "Love this photo!", userPhoto: "/9j/4AAQSkZJRgABAQAAAQABAAD/4TGIRXhpZgAATU0AKgAAAABQEAUAUAAAABAAAAASg..."
 - Document 2: _id: ObjectId('662655e53158ed69557672ef'), commentedPostId: "662639ab86bb3ac3a8fc42b0", user: "arthurmorg", text: "great one.", userPhoto: "/9j/4GICUNDX1B5T0ZJTEUAAQEAIAIMbGNtcwIQAABtbNyUkdCIFhZWlAH3AABAbkAwApAd1hY3NwQBOTA...AAAAAA" (redacted)

Excluded Webservices

The following webservices have not been documented using postman:

- makeComment
- updateCommentPhoto
- updateUserProfilePhotoInComments

The first two are involved in the comment creation process, and are passed as request 1 and request 2 to the postToDatabase function, a successful creation of the comment (with both text and profile photo in base64 format) will indicate the proper working of these services.

The last of the three is used when the user changes their profile photo by editing it in their account tab, and a change in the profile photo of the comments left by the user on posts would indicate the correct working of this webservice.

This will be covered in the video.

Following Collection

The below webservices are related to the “following” collection:

1. **getFollowingCount**

gets the provided user’s following count (number of people following them).

My Workspace

Overview Getting started [GET http://localhost:8080/M00914286/following/count/nasir](http://localhost:8080/M00914286/following/count/nasir)

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies (1) Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "followingCount": 2
3

```

200 OK 17 ms 255 B Save as example

Create Collection

MongoDB Compass - localhost:27017/pixstarCW2.following

Connect Edit View Collection Help

localhost:27017

My Queries following +

pixstarCW2.following

6 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Type a query: { field: 'value' } or [Generate query](#) Explain Reset Find Options

ADD DATA EXPORT DATA

FOLLOWING: Array (2)

- _id:** ObjectId('662596902d2eca24a20b6248')
- username:** "webdeveloper"
- following:** Array (2)

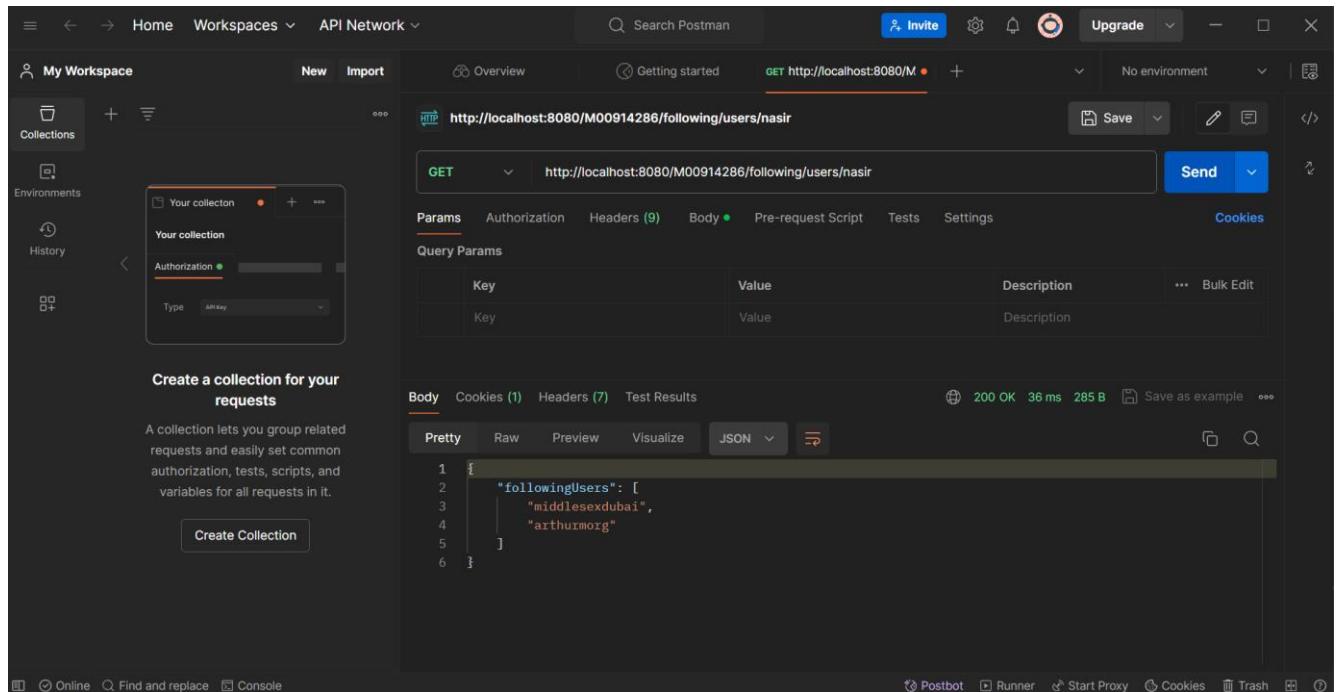
- _id:** ObjectId('6625985f2d2eca24a20b624c')
- username:** "nasir"
- following:** Array (2)
 - 0: "middlesexdubai"
 - 1: "arthurmorg"

- _id:** ObjectId('6626c8ec7f8a9082752ef4bd')
- username:** "M00914286"
- following:** Array (1)

1–6 of 6

2. getUsersFollowing

gets the provided users following.

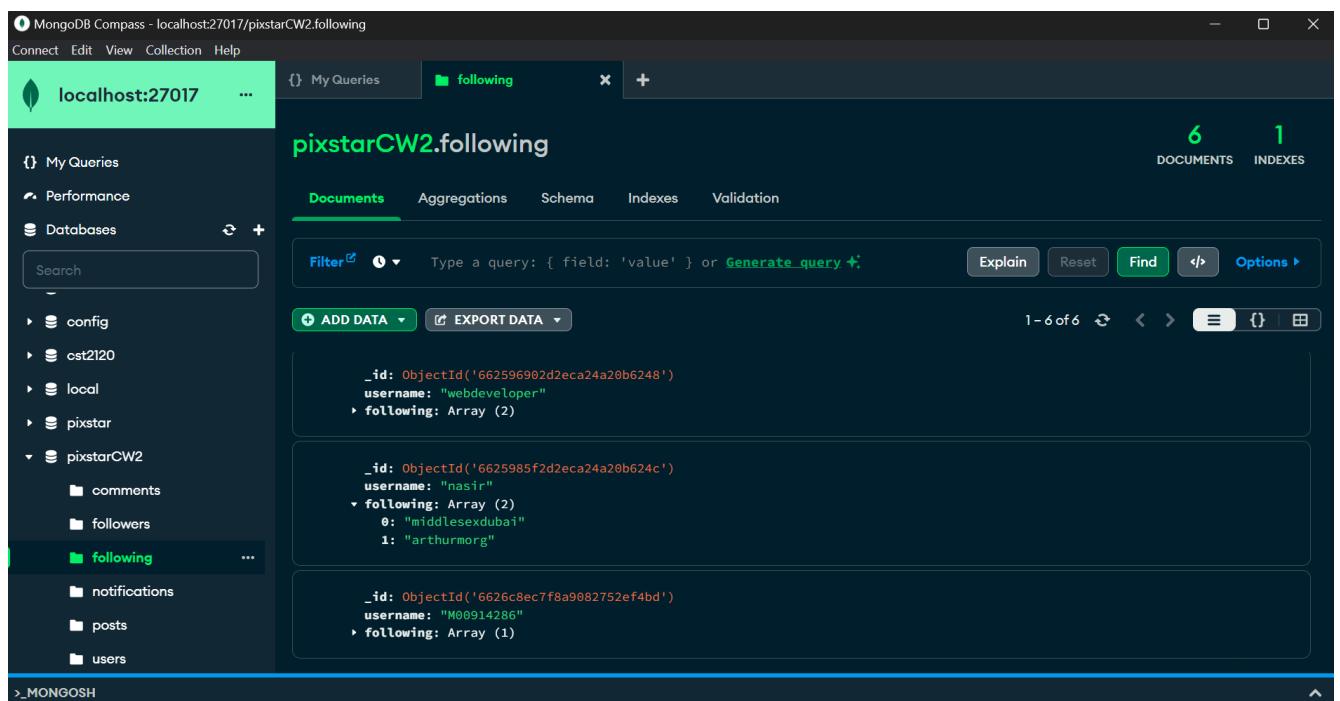


The screenshot shows the Postman interface. On the left, there's a sidebar with 'My Workspace' and a collection named 'Your collection'. Below it, a modal window titled 'Create a collection for your requests' explains what a collection is and has a 'Create Collection' button. The main area shows a GET request to 'http://localhost:8080/M00914286/following/users/nasir'. The 'Body' tab of the request details shows a JSON response:

```
1 {
2   "followingUsers": [
3     "middlesexdubai",
4     "arthurmorg"
5   ]
6 }
```

The response status is 200 OK with 36 ms latency and 285 B size. The response body is also displayed in the main panel:

```
{
  "followingUsers": [
    "middlesexdubai",
    "arthurmorg"
  ]
}
```



The screenshot shows the MongoDB Compass interface connected to 'localhost:27017'. The left sidebar shows databases like 'config', 'cst2120', 'local', 'pixstar', and 'pixstarCW2'. Under 'pixstarCW2', the 'following' collection is selected. The right panel shows the 'Documents' tab with three documents listed:

- `_id: ObjectId('662596902d2eca24a20b6248')`
`username: "webdeveloper"`
`> following: Array (2)`
- `_id: ObjectId('6625985f2d2eca24a20b624c')`
`username: "nasir"`
`> following: Array (2)`
 - 0: "middlesexdubai"
 - 1: "arthurmorg"
- `_id: ObjectId('6626c8ec7f8a9082752ef4bd')`
`username: "M00914286"`
`> following: Array (1)`

3. followUser

adds a user to another user's following list.

My Workspace

POST http://localhost:8080/M00914286/following/followUser

Body

```
{
  "user": "nasir",
  "followedUser": "M00914286"
}
```

Body

```
{
  "success": true,
  "message": "User nasir is now following M00914286"
}
```

MongoDB Compass - localhost:27017/pixstarCW2.following

localhost:27017

following

pixstarCW2.following

7 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Type a query: { field: 'value' } or [Generate query](#)

[ADD DATA](#) [EXPORT DATA](#)

```

_id: ObjectId('662596902d2eca24a20b6248')
username: "webdeveLoper"
following: Array (2)

_id: ObjectId('6625985f2d2eca24a20b624c')
username: "nasir"
following: Array (2)

_id: ObjectId('6626c8ec7f8a9082752ef4bd')
username: "M00914286"
following: Array (1)
  0: "nasir"

```

4. checkIfFollowing

checks if one user is following another, by checking if their in the users following array.

The image displays two screenshots of software interfaces used for API development and database management.

Postman Screenshot:

- Left Panel:** Shows "My Workspace" with a collection named "Your collection". It includes sections for "Collections", "Environments", "History", and a "Create a collection for your requests" section with a note about grouping related requests and setting common authorization, tests, scripts, and variables.
- Right Panel:** A POST request to `http://localhost:8080/M00914286/following/checkIfFollowing`. The "Body" tab shows the following JSON payload:


```
1 {
2   "user": "nasir",
3   "followedUser": "M00914286"
4 }
```
- Status:** Response code 200 OK, 27 ms, 255 B. The response body is:


```
1 {
2   "isFollowing": true
3 }
```

MongoDB Compass Screenshot:

- Left Panel:** Shows the connection status as "MongoDB Compass - localhost:27017/pixstarCW2.following" and navigation links for Connect, Edit, View, Collection, Help.
- Collection List:** Shows collections: "My Queries", "following", and others like "config", "cst2120", "local", "pixstar", "pixstarCW2", "comments", "followers", "notifications", "posts", "users".
- Collection View:** The "following" collection is selected. It shows 7 DOCUMENTS and 1 INDEXES. The "Documents" tab is active, displaying three documents:
 - Document 1: `_id: ObjectId('662596902d2eca24a20b6248')`, `username: "webdeveloper"`, `following: Array (2)`
 - Document 2: `_id: ObjectId('6625985f2d2eca24a20b624c')`, `username: "nasir"`, `following: Array (2)`
 - Document 3: `_id: ObjectId('6626c8ec7f8a9082752ef4bd')`, `username: "M00914286"`, `following: Array (1)`, `0: "nasir"`

5. removeFromFollowing

removes one user from the following of another.

The screenshot displays two applications side-by-side: Postman (top) and MongoDB Compass (bottom).

Postman:

- Left Panel:** Shows "My Workspace" with a collection named "Your collection". It includes sections for "Collections", "Environments", "History", and a "Create a collection for your requests" guide.
- Right Panel:** A POST request to `http://localhost:8080/M00914286/following/removeFromFollowing`. The "Body" tab shows the JSON payload:


```

1 {
2   ...
3   "user": "M00914286",
4   "followedUser": "nasir"
5 }
```
- Bottom Panel:** Response details: 200 OK, 22 ms, 310 B. The response body is:


```

1 {
2   "success": true,
3   "message": "Removed nasir from following list of M00914286"
4 }
```

MongoDB Compass:

- Left Sidebar:** Shows databases: config, cst2120, local, pixstar, pixstarCW2 (selected), comments, followers, notifications, posts, users.
- Top Bar:** MongoDB Compass - localhost:27017/pixstarCW2.following. Includes Connect, Edit, View, Collection, Help.
- Collection View:** "pixstarCW2.following" collection. It shows 6 documents and 1 index. The "Documents" tab is selected.
- Document Preview:** The first document has the following structure:


```

_id: ObjectId('662596902d2eca24a20b6248')
username: "webdeveloper"
following: Array (2)
```
- Second Document Preview:**

```

_id: ObjectId('6625985f2d2eca24a20b624c')
username: "nasir"
following: Array (2)
```
- Third Document Preview:**

```

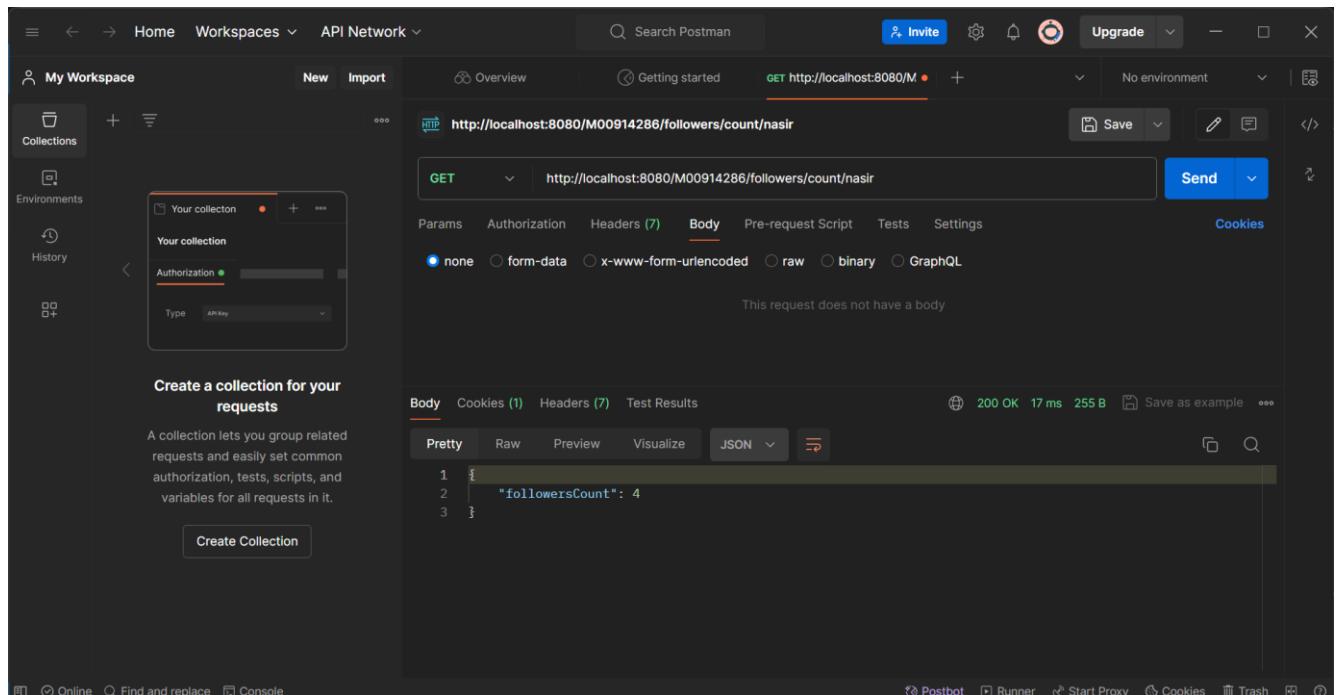
_id: ObjectId('6626c8ec7f8a9082752ef4bd')
username: "M00914286"
following: Array (empty)
```

Followers Collection

The below webservices are related to the “followers” collection:

1. getFollowersCount

gets the count of followers for the provided user.



The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar is visible with a 'Collections' section containing a single item named 'Your collection'. The main workspace displays a GET request to the URL `http://localhost:8080/M00914286/followers/count/nasir`. The 'Body' tab is selected, showing the response body as a JSON object:

```
1 {  
2     "followersCount": 4  
3 }
```

The status bar at the bottom indicates a 200 OK response with 17 ms latency and 255 B size. The bottom right corner includes links for Postbot, Runner, Start Proxy, Cookies, and Trash.

MongoDB Compass - localhost:27017/pixstarCW2.followers

localhost:27017

Databases: config, cst2120, local, pixstar, pixstarCW2 (selected)

Documents (5) Aggregations Schema Indexes Validation

Type a query: { field: 'value' } or [Generate query +](#)

pixstarCW2.followers

5 DOCUMENTS 1 INDEXES

Documents

1 - 5 of 5

Followers: Array (3)

```

_id: ObjectId('662594992d2eca24a20b6240')
username: "johnmars"
followers: Array (1)

```

```

_id: ObjectId('662594a32d2eca24a20b6242')
username: "nasir"
followers: Array (4)
0: "arthurmorg"
1: "middlesexdubai"
2: "webdeveloper"
3: "johnmars"

```

```

_id: ObjectId('662596992d2eca24a20b6244')

```

Actions: Edit, Share, Delete

2. getUserFollowers

gets the provided user's followers (username and profile photo) .

My Workspace

Collections

Environments

History

Create a collection for your requests

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

Create Collection

Overview Getting started GET http://localhost:8080/M00914286/followers/allFollowers/nasir

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies (1) Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "followers": [
3     "arthurmorg",
4     "middlesexdubai",
5     "webdeveloper",
6     "johnmars"
7   ]
8 }

```

200 OK 14 ms 306 B Save as example

Online Find and replace Console Postbot Runner Start Proxy Cookies Trash

MongoDB Compass - localhost:27017/pixstarCW2.followers

localhost:27017 ...

My Queries

Databases

Search

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } or Generate query + Explain Reset Find Options

pixstarCW2.followers

5 DOCUMENTS 1 INDEXES

Documents

ADD DATA EXPORT DATA

1 - 5 of 5

_id: ObjectId('6625943b2d2eca24a20b623d')
username: "webdeveloper"
followers: Array (3)

_id: ObjectId('662594992d2eca24a20b6240')
username: "johnmars"
followers: Array (1)

_id: ObjectId('662594a32d2eca24a20b6242')
username: "nasir"
followers: Array (4)
0: "arthurmorg"
1: "middlesexdubai"
2: "webdeveloper"
3: "johnmars"

Followers

following

notifications

posts

users

>_MONGOSH

3. addFollower

adds a user to the follower array of another user.

Home Workspaces API Network

My Workspace

New Import

Overview Getting started POST http://localhost:8080/I + No environment

HTTP http://localhost:8080/M00914286/followers/addFollower

POST http://localhost:8080/M00914286/followers/addFollower

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   ...
3   "user": "nasir",
4   "follower": "M00914286"
5 }
```

Send

Body Cookies (1) Headers (7) Test Results

Pretty Raw Preview Visualize JSON

1 {
2 "success": true,
3 "message": "User nasir now has follower M00914286"
4 }

200 OK 17 ms 301 B Save as example

Online Find and replace Console Postbot Runner Start Proxy Cookies Trash

4. removeFollower

removes the provided follower from the user.

```

POST http://localhost:8080/M00914286/followers/removeFollower
{
  "user": "nasir",
  "follower": "M00914286"
}
{
  "success": true,
  "message": "Removed M00914286 from followers list of nasir"
}
    
```

MongoDB Compass - localhost:27017/pixstarCW2.followers

localhost:27017

My Queries

Databases

Search

config

cst2120

local

pixstar

pixstarCW2

comments

followers

following

notifications

posts

users

>_MONGOSH

pixstarCW2.followers

5 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } or Generate query + Explain Reset Find Options

ADD DATA EXPORT DATA

1 - 5 of 5

_id: ObjectId('662594992d2eca24a20b6240')
username: "johnmars"
followers: Array (1)

_id: ObjectId('662594a32d2eca24a20b6242')
username: "nasir"
followers: Array (4)
0: "arthurmorg"
1: "middlesexdubai"
2: "webdeveloper"
3: "johnmars"

_id: ObjectId('662596992d2eca24a20b624a')

1 2 ... "user": "nasir",
3 "follower": "arthurmorg"
4 }

5. checkIfFollower

checks if the provided follower is in the follower array of the user.

My Workspace

New Import

Overview Getting started

POST http://localhost:8080/M00914286/followers/checkIfFollower

Save Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   ... "user": "nasir",
3   "follower": "arthurmorg"
4 }
```

Body Cookies (1) Headers (7) Test Results

Pretty Raw Preview Visualize JSON

200 OK 13 ms 254 B Save as example

Online Find and replace Console

Postbot Runner Start Proxy Cookies Trash

MongoDB Compass - localhost:27017/pixstarCW2.followers

localhost:27017 ...

My Queries followers +

pixstarCW2.followers

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } or Generate query Explain Reset Find Options

ADD DATA EXPORT DATA

1 – 5 of 5

`_id: ObjectId('6625943b2d2eca24a20b623d')
username: "webdeveloper"
followers: Array (3)`

`_id: ObjectId('662594992d2eca24a20b6240')
username: "johnmars"
followers: Array (1)`

`_id: ObjectId('662594a32d2eca24a20b6242')
username: "nasir"
followers: Array (4)
0: "arthurmorg"
1: "middlesexdubai"
2: "webdeveloper"
3: "johnmars"`

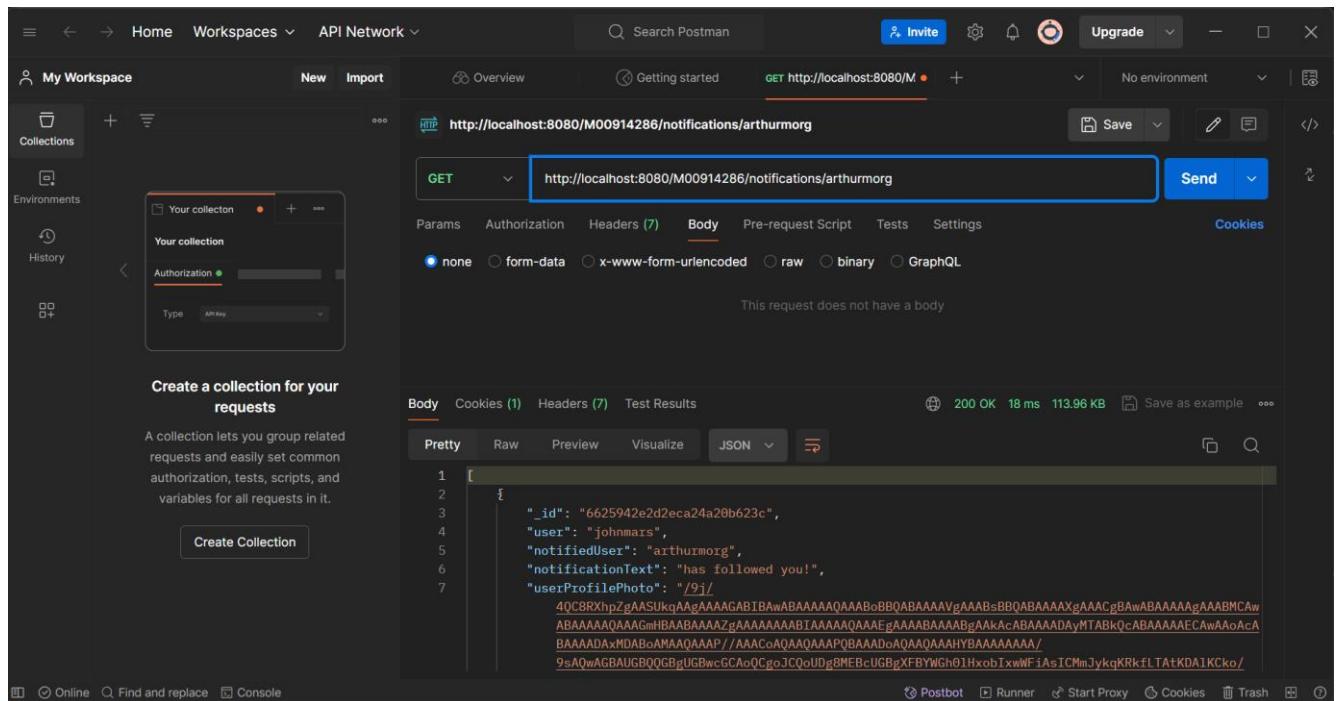
... MONGOSH

Notifications Collection

The following webservices are related to the notifications collection:

1. getNotifications

gets all the notifications for the provided user.



The screenshot shows the Postman interface with a collection named "Your collection". A GET request is defined with the URL `http://localhost:8080/M00914286/notifications/arthurmorg`. The response body is displayed in JSON format:

```
1 {
2   "_id": "6625942e2d2eca24a20b623c",
3   "user": "johnmars",
4   "notifiedUser": "arthurmorg",
5   "notificationText": "has followed you!",
6   "userProfilePhoto": "/9j/
7     4QCB8XhpZgAA5UkqAAgAAAAGAB1BAwABA5AAAQAAABoB80ABAAAAAVgAAABsBBQABAAAAXgAAACpBAwABAAAAAgAAABMCAw
8     ABAAAAAAQAAAGmHBAAABAAA2gAAAAAAABIAAAAQAAAEGAAAAAAAABgAAkAcABAAAADyMTABkQcABAAAAECAwAaOAcA
9     BAAAADxMDAbMAAQAAAP//AACcAQAAQAAAAPQBAAAdoQAQAAQAAHYBAAAAAAA/
9sAwAGBAUGBQ0GBgUGBwGCAoCgoJCo0Dg8MEBcUGBgXFByWGb01Hxb0IxwWFiAsICMmJykqKrkfLTtKDA1KCo/
```

The screenshot shows the Postman interface. On the left, the 'My Workspace' sidebar is visible with 'Collections', 'Environments', and 'History'. In the center, a 'Your collection' card is selected. Below it, a modal window titled 'Create a collection for your requests' explains what a collection is and provides a 'Create Collection' button. The main workspace shows a GET request to `http://localhost:8080/M00914286/notifications/arthurmorg`. The 'Body' tab is selected, showing the response body:

```

{
  "id": "662598662d2eca24a20b624",
  "user": "nasir",
  "notifiedUser": "arthurmorg",
  "notificationText": "has followed you!",
  "userProfilePhoto": "/9j/4AAQSkZJRgABAQAAAQABAD/4TGIRXhpZgAATU0AKgAAAAgABQEaAAUAAAABAAAASgEbAAUAAAABAAAUGEoAAMAAAABAAIAAAIAAAAABAAEAAIdpAA
  QAAAABAAAAGAAALQAAABIAAAQAAAQAAEgAAAABAEeQAAHAAAABDAyMjGraQAHAAAABCAwCgAAHAAAABDaxMDcGAQAD
  AAAAQABAAcAgAEAAAAAAAQAAATygAwEAAAAAQAATAyBqAdAAAAAQAaaaaAAAAYBwAdAAAAAQAAGAAABgAgFAAAAAA
  AAAQIBGwAFAAAAQAAAQoBKAADAAAQQACAAACQAAEAAAQAAMGwAAAAAAAASAAAAAAAASAAAAAAEABIAAAA
  Af/YBAE0gAoCgAwEIAIRaQMRAf/EABBAAAEFQEBQAEBAEEAAAABAAgMEBOYHCakKC/
  EAIHAAATRAwMCRAMR00NEAAAReOcAmAECDHISTTERBhMDVnAePDoa7ChfTMCeEVIIH-HoTOMcgnTfKvYCBnTSvNkPm
}

```

The response status is 200 OK with 18 ms and 113.96 KB. On the right, there are buttons for 'Save as example', 'Postbot', 'Runner', 'Start Proxy', 'Cookies', and 'Trash'.

The screenshot shows the MongoDB Compass interface. The left sidebar shows databases: config, cst2120, local, pixstar, pixstarCW2 (selected), comments, followers, following, notifications (selected), posts, users. The right panel shows the 'pixstarCW2.notifications' collection with 19 documents and 1 index. A filter is applied: `{notifiedUser:"arthurmorg"}`. Two documents are listed:

```

{
  "_id": ObjectId('6625942e2d2eca24a20b623c'),
  "user": "johnmars",
  "notifiedUser": "arthurmorg",
  "notificationText": "has followed you!",
  "userProfilePhoto": "/9j/40C8RXhpZgAASUkqAAGAAAAGABIBAwABAAAAAQAABoBBQABAAAvgAAABsBBQABAA..."
}

{
  "_id": ObjectId('662598662d2eca24a20b624e'),
  "user": "nasir",
  "notifiedUser": "arthurmorg",
  "notificationText": "has followed you!",
  "userProfilePhoto": "/9j/4AAQSkZJRgABAQAAAQABAD/4TGIRXhpZgAATU0AKgAAAAgABQEaAAUAAAABAAAASg...
}

```

2. deleteNotification

deletes a notification that has the specified user, notifiedUser, and notificationText.

MongoDB Compass - localhost:27017/pixstarCW2.notifications

Connect Edit View Collection Help

localhost:27017 ...

My Queries notifications +

pixstarCW2.notifications

19 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter [?] {notifiedUser:"nasir"} Generate query Explain Reset Find Options >

+ ADD DATA EXPORT DATA 1–8 of 8 < > ⌂ ⌂

notificationText: "has liked your post "space!""
userProfilePhoto: "/9j/4QC8RXhpZgAASUkqAqAAAAGABIBAwABAAAAAAQAAABoBBQABAAAAVgAAABsBBQABAA..."

_id: ObjectId('662663d80b4a5826311221f0')
user: "johnmars"
notifiedUser: "nasir"
notificationText: "has liked your post "JDM!""
userProfilePhoto: "/9j/4QC8RXhpZgAASUkqAqAAAAGABIBAwABAAAAAAQAAABoBBQABAAAAVgAAABsBBQABAA..."

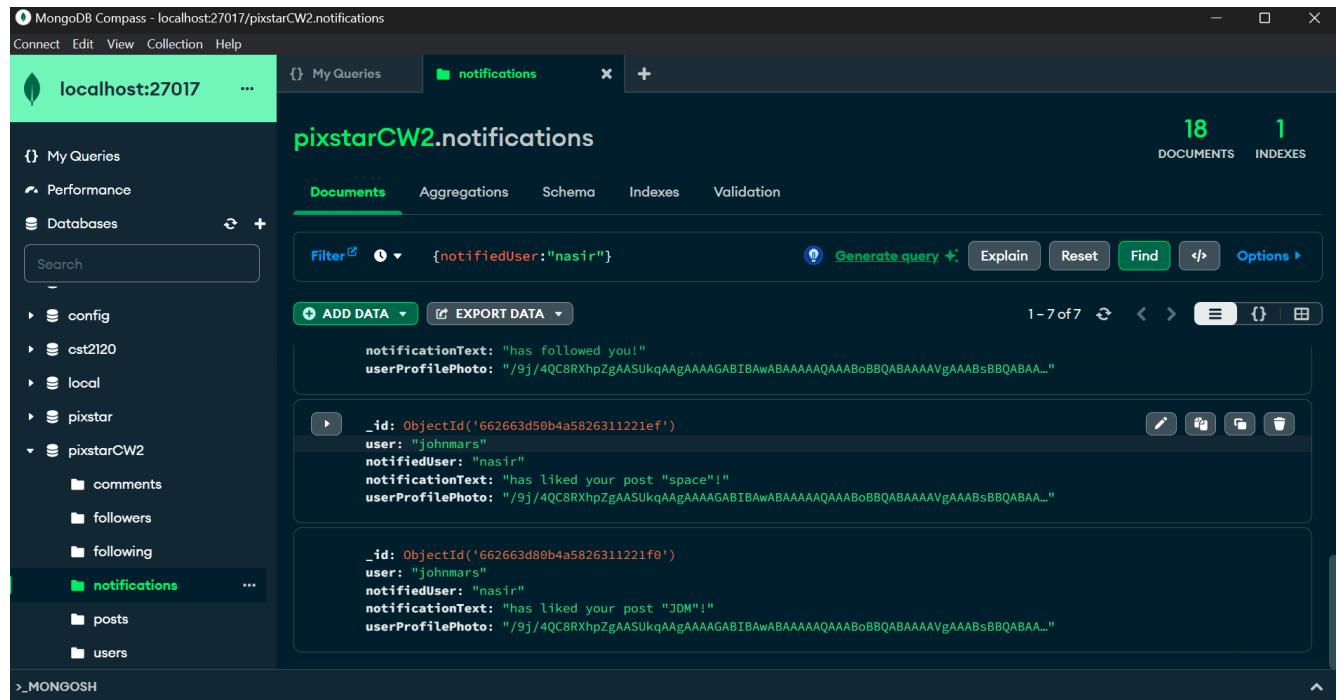
_id: ObjectId('6626cd7d7f8a9082752ef4bf')
user: "M00914286"
notifiedUser: "nasir"
notificationText: "has followed you!"
userProfilePhoto: "/9j/4QAiRXhpZgAASUkqAqAAAABIBAwABAAAAAAQAAAAAAAD/4QAC/9sAhAAMCAGNCQ..."

The screenshot shows the Postman application interface. On the left sidebar, there are sections for 'My Workspace' (Collections, Environments, History), 'Create a collection for your requests' (with a note about collections), and a 'Create Collection' button. The main workspace shows a 'Overview' tab, a 'Getting started' tab, and a selected 'DELETE http://localhost:8080/M00914286/notifications/deleteNotification' tab. The request details panel shows the method as 'DELETE', the URL as 'http://localhost:8080/M00914286/notifications/deleteNotification', and the body type as 'raw' with the following JSON payload:

```
1 {
2   "user": "M00914286",
3   "notifiedUser": "nasir",
4   "notificationText": "has followed you!"}
```

The response panel shows a successful 200 OK status with a response time of 17 ms and a size of 297 B. The response body is:

```
1 {
2   "success": true,
3   "message": "Notification deleted successfully"
4 }
```



Excluded Webservices

The following webservices have not been documented using postman:

- createNotification
 - updateNotificationProfilePhoto
 - newNotificationUserProfilePhoto

The first two are involved in the notification creation process, and a successful creation of the notification (with both text and profile photo in base64 format) will indicate the proper working of these services.

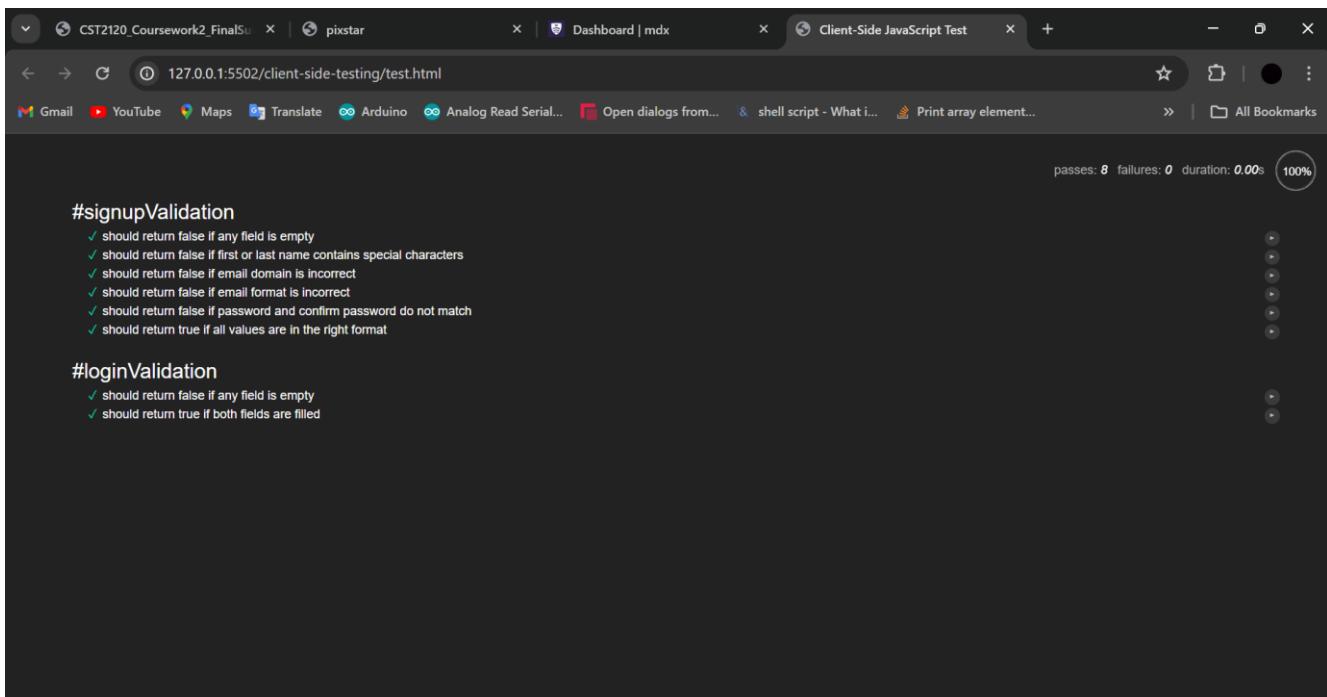
The last of the three is used when the user changes their profile photo by editing it in their account tab, and a change in the profile photo of the notifications in which the user is present would indicate the correct working of this webservice.

This will be covered in the video.

Testing

Client-Side Testing

For client-side testing, there are two files test-client.js that contains the tests and test.html that displays their results, I have tested the validation for the sign up and login processes. By opening the test.html file the results are shown as below,



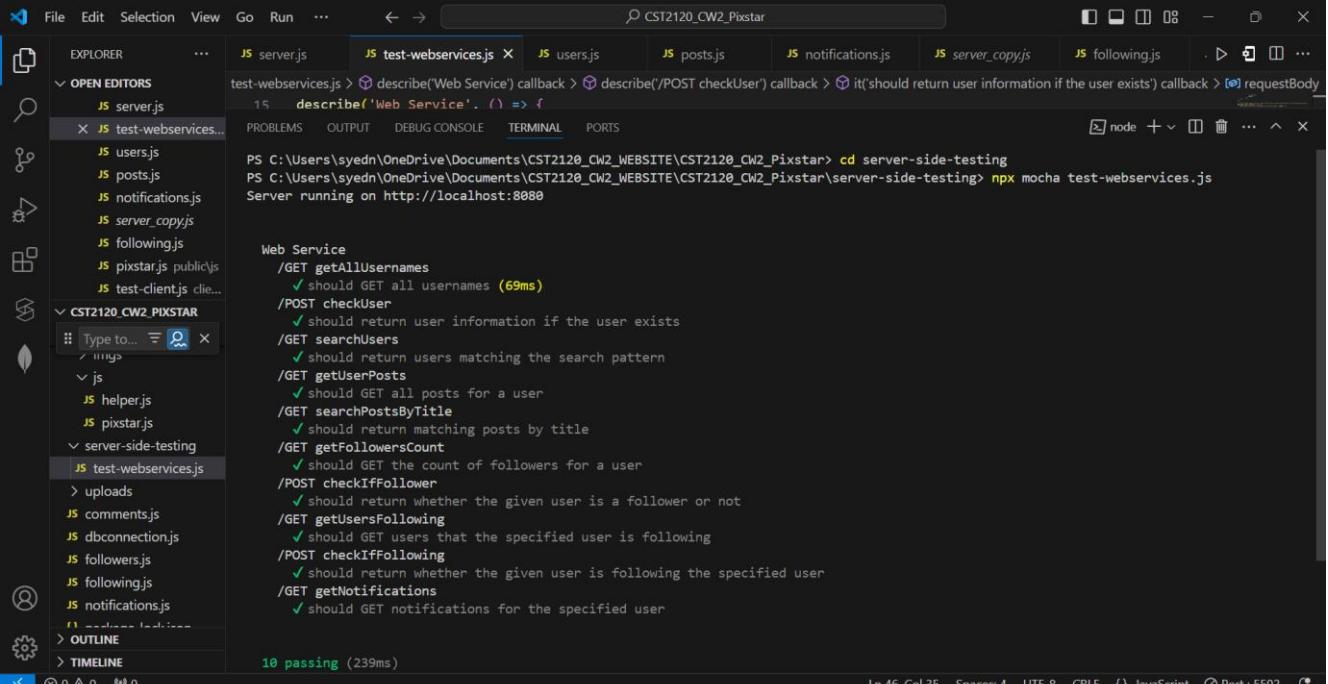
The screenshot shows a browser window with three tabs open: 'CST2120_Coursework2_FinalSu...', 'pixstar', and 'Client-Side JavaScript Test'. The 'Client-Side JavaScript Test' tab is active, displaying the results of a test suite. The results are presented in a table:

Test Suite	Test Case	Status
#signupValidation	should return false if any field is empty	Pass
	should return false if first or last name contains special characters	Pass
	should return false if email domain is incorrect	Pass
	should return false if email format is incorrect	Pass
	should return false if password and confirm password do not match	Pass
	should return true if all values are in the right format	Pass
	#loginValidation	Pass
should return false if any field is empty	Pass	
should return true if both fields are filled	Pass	

At the bottom right of the results table, it says 'passes: 8 failures: 0 duration: 0.00s 100%'.

Server-Side Testing

For server-side testing, I have tested ten of my webservices, containing webservices for all the collections. The tests were run in the terminal by first “cd server-side-testing”, and then using the command “npx mocha test-webservices.js”



The screenshot shows a Visual Studio Code (VS Code) interface with the title bar "CST2120_CW2_Pixstar". The left sidebar displays the file structure of the project, including files like "server.js", "test-webservices.js", "users.js", "posts.js", "notifications.js", "server_copy.js", "following.js", "pixstar.js", "public.js", and "test-client.js". The main editor area shows the content of "test-webservices.js". The terminal tab at the bottom shows the command line output:

```
PS C:\Users\syedn\OneDrive\Documents\CST2120_CW2_WEBSITE\CST2120_CW2_Pixstar> cd server-side-testing
PS C:\Users\syedn\OneDrive\Documents\CST2120_CW2_WEBSITE\CST2120_CW2_Pixstar> npx mocha test-webservices.js
Server running on http://localhost:8080

Web Service
  /GET getAllUsernames
    ✓ should GET all usernames (69ms)
  /POST checkUser
    ✓ should return user information if the user exists
  /GET searchUsers
    ✓ should return users matching the search pattern
  /GET getUserPosts
    ✓ should GET all posts for a user
  /GET searchPostsByTitle
    ✓ should return matching posts by title
  /GET getFollowersCount
    ✓ should GET the count of followers for a user
  /POST checkIfFollower
    ✓ should return whether the given user is a follower or not
  /GET getUsersFollowing
    ✓ should GET users that the specified user is following
  /POST checkIfFollowing
    ✓ should return whether the given user is following the specified user
  /GET getNotifications
    ✓ should GET notifications for the specified user

10 passing (239ms)
```

At the bottom right of the terminal, status indicators show "Ln 46, Col 35", "Spaces: 4", "UTF-8", "CRLF", "JavaScript", "Port: 5502", and a refresh icon.

HTML and CSS Validation

HTML

Attached below is the screenshot of the validation of the index.html page.

The screenshot shows the Nu Html Checker interface running in a browser. The title bar indicates the page is being validated at validator.w3.org/nu/#textarea. The main area displays the HTML code for index.html, which includes a DOCTYPE declaration, a head section with meta tags for charset and title, and a body section containing a single div with id="content". A note in the body states: '<!--the content div contains call the other divs-->'. Below the code editor is a 'Check' button. At the bottom, there's a message filtering section and a green status bar stating 'Document checking completed. No errors or warnings to show.' The source code is also displayed in a scrollable pane at the bottom.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>pixstar</title>
    <link rel="stylesheet" href="../css/style.css">
    <script type="module" src="../js/pixstar.js"></script>
</head>
<body>
    <!--the content div contains call the other divs-->
    <div id="content">
</div>
</body>

```

Document checking completed. No errors or warnings to show.

CSS

Attached below is the screenshots of the validation for the style.css file.

The screenshot shows the Nu Html Checker interface on a web browser. The URL in the address bar is `validator.w3.org/nu/#textarea`. The main area displays the CSS code being checked:

```
/* applied to all */
{
    font-family:'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif;
    font-size: 15px;

}

/* CSS for the body */
body {
    min-height: 100vh; /* fit to window */
    background: rgb(241, 241, 241);
    overflow: hidden;
}
```

Below the code, there is a "Check" button. A message at the bottom states: "Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings." There is also a "Message Filtering" link. A green banner at the bottom indicates: "Document checking completed. No errors or warnings to show."

Source

```
1. /* applied to all */  
2. * {  
3.     font-family:'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif;  
4.     font-size: 15px;  
5. }  
6. }  
7. }  
8. /* CSS for the body */  
9. body {  
10.    min-height: 100vh; /* fit to window */  
11.    background: rgb(241, 241, 241);  
12.    overflow: hidden;  
13. }  
14. }  
15. }  
16. }  
17. /* hide class that is used to display and hide divs according to the button pressed*/  
18. .hide {  
19.     display: none !important;  
20. }  
21. }  
22. /* content div that contains all the other divs*/  
23. #content {  
24. }
```

Security, Legal, and Privacy Issues

Resolved Issues

1. Password Storage

Initially, the passwords were being stored in plaintext format, which poses a huge security risk, this was resolved by storing the password in the following format.

First, the generateHash function is called in the users.js file, and the password is passed as the argument and the hashed password is returned, password is hashed using SHA-256 this will be used as the key for both the encryption and decryption of the password.

Then the encrypt function is called passing both (password, hashedPassword), the function takes sixteen characters from the hashedPassword to be used as the initialization vector (IV) which is a random value used in conjunction with the key for encryption.

A cipher object is created with the AES-256 encryption algorithm and is initialized with the hashed key and IV.

The plaintext password is then updated and encrypted.

The decrypt function works in a similar manner, but converts the encrypted password back to plain text.

2. Session Security

A secret for the session is generated randomly using the crypto library, a random secret for session management enhances security by making it difficult for attackers to predict or manipulate session data.

The cookie has a max age of twelve hours, and is terminated after the time has expired, this time limit supports both usability and security.

Session is also terminated as soon as the user logs out.

Unresolved Issues

There are a few issues that are unresolved, they are listed below:

1. Static Initialization Vector (IV)

Using a static IV derived from the hashed password introduces a potential vulnerability. An IV should be unique for each encryption operation to prevent patterns in the ciphertext and mitigate certain cryptographic attacks.

Generating a random IV for each encryption operation and transmitting it along with the ciphertext would be a much more secure approach.

2. No Block System

Users currently do not have the option to block other users, and so this would mean the users details could potentially be accessed by malicious individuals, that may harass them by commenting on their posts.

3. No Private Profiles

Although the user can remove followers, the following system does not have requests that need to be acknowledged, and so any user can follow another user by just a click of a button and view their posts, every users posts are also displayed in the explore tab.