NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

DEPARTMENT OF

INFORMATION TECHNOLOGY

MINI-PROJECT

TITLE : Social Networking System - Socio

COURSE : Software Engineering

Submitted to : Dr. Biju R Mohan

Submitted by : Shreyas Shankar (16IT138)

Suyash Ghuge (16IT114)

Introduction

This document details the documentation for “Social Networking System - Socio”. Socio is a social networking app where users can create their accounts for free and socialize with others. The primary aim of the document is to detail out/ describe the whole system and clearly lists all its functionalities.

This document is meant for any user around the world who likes to socialize but is bound by geographical or any other barriers.

The main purpose of this document is to describe the features and behavior of the system. It includes a variety of elements that attempts to define the intended functionality required by the user’s system.

The system’s functionality is further described in terms of UML diagrams which include Use Case, Class, ER, State, Navigation, Sequence, Collaboration, Component and Deployment diagrams.

Overview

Socio is a free social networking platform open to users all over the world. The platform allows users to create an account for free and start socializing.

Users can send friend requests to other existing users. The user can communicate with his/her friends via messages. Users can also create posts which would be visible to his/her friends only who can like or comment on these posts. Users can create a group and add members to it. The group has its own posts which are shared only among the group members.

There is also an option for chat. The user can chat with his friends or within his group by entering the respective chat room.

Software Requirements Specifications

1. Introduction

1.1 Purpose

The SRS will provide a detailed description of the requirements for Socio - A social networking system. This SRS will allow for a complete understanding of what is to be expected from the newly introduced system which is to be constructed. The clear understanding of the system and its functionality will allow for the correct software to be developed for the end user and will be used for the development of the further stages of the project.

1.2 Document Conventions

The document is prepared using Google Docs and has used the font type ‘Times New Roman’. The font size that has been used to type this document is 14pt for the headings and 11pt for the corresponding body. Standard IEEE template is the template used to recognize the appearance of the document and its flow.

1.3 Intended Audience and Reading Suggestions

This document is made by keeping in mind different types of readers. This document will be useful for different audience in various ways.

Audience	**Use**
Developers	They will use this document as a guidance for design and implementation phase.
Managers	They will see all the constraints are covered properly. Time and cost is within limits or not.
Marketing Staff	They can use this document to make advertisements for this android app because by reading this document they will know what the system will do? How this system is different from others.
User	By reading the SRS they can ensure whether their needs are being met by the App or not.
Testers	They will test the implementation of the project according to the SRS base.
Documentation Writer	They will use this document during the documentation of the project.

1.4 Product Scope

- The System developed will enable the users to socialize over the internet. The users can make friends, chat with them, create posts to share among friends, like or comment on posts, create groups and have group chats and also share files. Since everyone is leading a busy and stressful life, socializing means a lot to everyone.

1.5 References

- IEEE. IEEE Std. 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

2. Overall Description

2.1 Product Perspective

The SOCIO - A Social Networking System project is a new, self-contained product intended for use on the Web platform. Socio App enables us to interact with other users through chat, posts, comments, messages, groups and many more.

2.2 Product Functions

The above DFD clearly shows various processes associated with the project and how data flows between an entity and database via a specific process

User registers his account and creates/updates profile.
User views different profiles and sends friend requests to make new friends or messages/shares files with existing friends.
User views received friend requests and accepts or declines. A new chat room is created if accepted.
User views received messages and replies back.
User creates group, add/remove his friends to it and grant/take adminship. A new chat room is created/updated for the group.
User selects chat room and views previous chat send new chat messages in that chat room.
User posts images/files on his profile.
User views posts of friends on the home page and can like and comment on the post.

2.3 User Classes and Characteristics

User	**Characteristics**
System Admin	It will be the system administrator. He will maintain the overall App.
Users	It includes the people who use the social networking app.

2.4 Operating Environment

Socio is the software application, which will be limited to a web application. The application is not resource- or graphics-intensive, so there are no practical hardware constraints. The app will rely on several functionalities built into Django's framework and API, so ensuring appropriate usage of the API will be a major concern.

The system shall be deployed on a Heroku platform which is a service providing platform to host with Ver. 5.7.21 MySQL Database to maintain the databases. The system shall be accessed on multiple browsers e.g. Google Chrome, Mozilla Firefox etc. The Operating System on the front-end PCs/Laptops can be MS Windows, Unix, Linux or Apple Mac.

Sr No	**System**	**Environment for development**	**Description**
1.	Software used to develop	Backend: Django 2.0	Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.
		Socket Server : NodeJS, Express	Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
		Database: MYSQL	MySQL is an open-source relational database management system(RDBMS).

		Frontend: HTML, JavaScript, jQuery, AngularJS	HTML: is the standard mark-up language for creating web pages and web applications JavaScript: As a multi-paradigm language, JavaScript supports event driven functional, and imperative (including object-oriented and prototype based) programming style. jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML AngularJS: The AngularJS framework works by first reading the HTML page, which has additional custom tag attributes embedded into it. Angular interprets those attributes as directives to bind input or output parts of the page to a model that is represented by standard JavaScript variables
		Styling: CSS, Bootstrap	Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a mark-up language. Bootstrap is a free and open-source front-end web framework for designing websites and web applications.
		IDE: PyCharm (student version)	It provides code analysis, a graphical debugger, an integrated unit tester, integration with version-control systems (VCSes), and supports web development with Django.

		Version Control: Git, GitHub	GIT is a version-control system for tracking changes in computer-files and coordinating work on those files among multiple people.
		Text Editor: - Sublime	Sublime Text is a cross-platform source code editor with a Python Application programming interface(API)
2.	Server hosting/ Installation (user testing)	Heroku If bugs are found again, coding and unit testing tools will be used along with latest version of software from Git.	Heroku is a cloud Platform as a Service (PaaS) supporting several programming languages. that is used as a web application deployment model.

2.5 Design and Implementation Constraints

The primary design constraint is the web platform. Since the application is designated for any device which can access the web, adaptive screen size and resolution will be a major design consideration. Creating a user interface which is both effective and easily navigable will pose a difficult challenge.

Moreover, since security is a major issue, passwords of the registered users need to be hashed when storing in the database. Also, the privacy of the users needs to be maintained. Only the user who is friends with another user should have the associated rights and not others. Only the group admins should have authority over the groups. Only users of the specific chat room can put up messages in it.User can like a post only once. Any user shouldn't be able to change other's profiles.

2.6 User Documentation

A brochure will be provided describing the functionality of the Web App.

2.7 Assumptions and Dependencies

- The app requires Django framework and REST API.
- A NodeJs server will be required which will be a socket server for real time chats and posts.
- The project requires a host and server, heroku account to publish the app.

3. External Interface Requirements

3.1 User Interfaces

- The Web application will have an initial login page where users would enter their respective login credentials. It would also have an option to allow new users to Sign up.
- The Sign up page would prompt the user to enter email id, username and password.
- Logging in would land the user on the Home Page. Posts would be displayed on the Home page. There would also be an option for the user to create new posts.
- Home page would have a navigation bar having options to view User Profile, Find Friends, Chat, Logout, etc. Also there is a form for creating posts.
- The posts will have an option to like/dislike and a form for commenting. If one's own post, an option to edit/delete the post will be available.
- The chat page would show chat history with the respective person/group and a form to send a chat message.
- The profile page would have an option of sending/cancelling a friend request to that person if he/she is not yet a friend or it would have an option of sending a message or unfriending that person if he/she is a friend.

3.2 Hardware Interfaces

- The System shall be deployed on Heroku Platform. All the Stakeholders are supposed to login into the Socio website where there will be a specific URL to access the System. Hardware Requirements for stakeholders.
- Pentium 4 processor or higher
- Approximately 100 MB of free harddrive space
- Minimum 128 MB RAM
- Hardware Requirements for hosting
- Minimum 1GB database space
- Minimum 2GB RAM
- Wearable devices will not be supported with this application.

3.3 Software Interfaces

Software Requirements for Hosting:

- Django 2.0
- MySQL

Software Requirements for Stakeholders:

- Browser (Google Chrome, Mozilla Firefox, Safari etc.)
- Operating System supporting the above browsers.

3.4 Communications Interfaces

The System will be operational using standard web-browsers like Safari, Google Chrome and Firefox. Users will connect- through a secured encrypted connection over internet. Since the data communicated over internet is confidential it is imperative that encrypted protocols are used to prevent data leakages.

3.5 IDE Interfaces

- IDE Pycharm for development using Django Framework.

4. System Features

4.1 User Signup

Use Case Name	User Signup
Actor	User
Overview	This use case is about registration of the user in.
Pre-condition	The internet connection is working. Server is not down.
Post-condition	The credentials entered are within constraints.

4.2 User Login

Use Case Name	User Login
Actor	User
Overview	This use case is about authorisation of the user.
Pre-condition	The user is registered to the database.
Post-condition	The credentials entered are correct..

4.3 Create/Edit Profile

Use Case Name	Create/Edit Profile
Actor	User
Overview	This use case is about profile creation/updation by the user.
Pre-condition	The user is logged in.
Post-condition	The profile is created/updated.

4.4 View Profiles

Use Case Name	View Profiles
Actor	User
Overview	This use case is about viewing different profiles.
Pre-condition	The user is logged in.
Post-condition	None

4.5 Send Friend Requests

Use Case Name	Send Friend Requests
Actor	User
Overview	This use case is about sending friend requests to different users.
Pre-condition	The user is logged in. The profile user is not user's friend already.
Post-condition	A request is generated with sender and receiver fields.

4.6 Accept Friend Requests

Use Case Name	Accept Friend Requests
Actor	User
Overview	This use case is about accepting friend requests to different users.
Pre-condition	The user is logged in. The profile user is not user's friend already.
Post-condition	The profiles are added to each other's friend list respectively. A new chat room is created with the users as its members.

4.7 Send Messages/Files

Use Case Name	Send Messages/Files
Actor	User
Overview	This use case is about sending messages/files to friends.
Pre-condition	The user is logged in. The profile is present in user's friend list.
Post-condition	New message is created with sender and receiver fields.

4.8 Reply Messages/Files

Use Case Name	Reply Messages/Files
Actor	User
Overview	This use case is about replying to messages/files received.
Pre-condition	The user is logged in. The user must have received a message/file.
Post-condition	New reply message is created with sender and receiver fields.

4.9 Create Groups

Use Case Name	Create Groups
Actor	User
Overview	This use case is about creating a group. The user can then add/remove members and grant/take away adminship.
Pre-condition	The user is logged in. The user has an active profile.
Post-condition	A group is created with the user as its admin. A new chat room is created for the group.

4.10 Add Members to the Group

Use Case Name	Add Members to the Group
Actor	User
Overview	This use case is about adding members to a group.
Pre-condition	The user is logged in. The user is an admin of the group. The member to be added is present in the user's friend list.
Post-condition	The member is added to the group. The members are added to the group's chat room.

4.11 Remove Members from the Group

Use Case Name	Remove Members from the Group
Actor	User
Overview	This use case is about removing members from a group.
Pre-condition	The user is logged in. The user is an admin of the group. The member to be removed is the group's member.
Post-condition	The member is removed from the group. The member is removed from the group's chat room.

4.12 Add Admins to the Group

Use Case Name	Add Admin to the Group
Actor	User
Overview	This use case is about adding admins to a group.
Pre-condition	The user is logged in. The user is an admin of the group. The member to be given admin rights is the group's member.
Post-condition	The member is given admin rights of the group.

4.13 Remove Admins from the Group

Use Case Name	Remove admins from the Group
Actor	User
Overview	This use case is about removing admins from a group.
Pre-condition	The user is logged in. The user is an admin of the group. The member to be removed as admin is one of the group's admin.
Post-condition	The admin rights of the member are taken away.

4.14 Create Posts

Use Case Name	Create Posts
Actor	User
Overview	This use case is about creating posts. The posts can be posted in one's profile or in a group.
Pre-condition	The user is logged in. The user has an active profile.
Post-condition	New post is created.

4.14 Like/Unlike Posts

Use Case Name	Like/Unlike Posts
Actor	User
Overview	This use case is about liking/unliking posts.
Pre-condition	The user is logged in. The user has an active profile. The user can like/unlike only once. He can toggle between like/unlike.
Post-condition	For like, the user is added to the list of users who like the post. For unlike, the user is removed from the list of users who like the post.

4.15 Comment on Posts

Use Case Name	Comment on Posts
Actor	User
Overview	This use case is about commenting on posts.
Pre-condition	The user is logged in. The user has an active profile.
Post-condition	A comment is created for that post.

4.16 Chat

Use Case Name	Chat
Actor	User
Overview	This use case is about chatting with other users or a group of user.
Pre-condition	The user is logged in. There must be a chat room of which the participants of the chat are members of.
Post-condition	A Chat message is created for that chat room.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

1. Scalability

System should be able to handle a large number of users. For e.g. handling around Thousand users at the same time.

2. Speed

The application should he fast.It should not slow down with increase in the number of users. Search functionality should he fast to enable better end-user experience. The system should be quick enough to be able to respond to the user actions within a short period of time.4.g. the search user functionality of the site should perform a quick search among the users on the database.

5.2 Safety Requirements

There are no specific Safety Requirements.

5.3 Security Requirements

- During user registration, the given email address is validated.
- The password should be at least 8 characters containing at least a small character and a number and a special character.
- Password is stored as a hash value in database.
- We are transferring all data via HTTPS i.e. via SS so that the data is encrypted during the transit. Thus safeguarding the user information.

5.4 Software Quality Attributes

1. Usability

User interface should be simple and clear to beak to understand by any user.

2. Availability

- The system should be available at all times. t should he ensured that there should be minimum or no downtime to ensure better user experience
- The system should be reliable. It should yield correct results if a user performs search for a person. Also, if the user sends message, the system should ensure that the correct message is delivered to the correct destination without any loss or change in content.

3. Maintainability

The system should be developed in such a way that it is extensible. It should be easy to incorporate new features requirements or accommodate a change in the existing requirements.

5.6 Business Rules

1. All Users shall access the system using a login/user-id and password. The login-id/password will be managed in a secured manner.
2. A User cannot add friends, create groups or post without creating a profile.
3. A User can send a message or a file to another user only if he is in the user's friend list.
4. A User can add another user as a group member only if he is in the user's friend list and the user is one of the group's admin.
5. A User can alter adminship of a group member only if he is one of the group's admin.

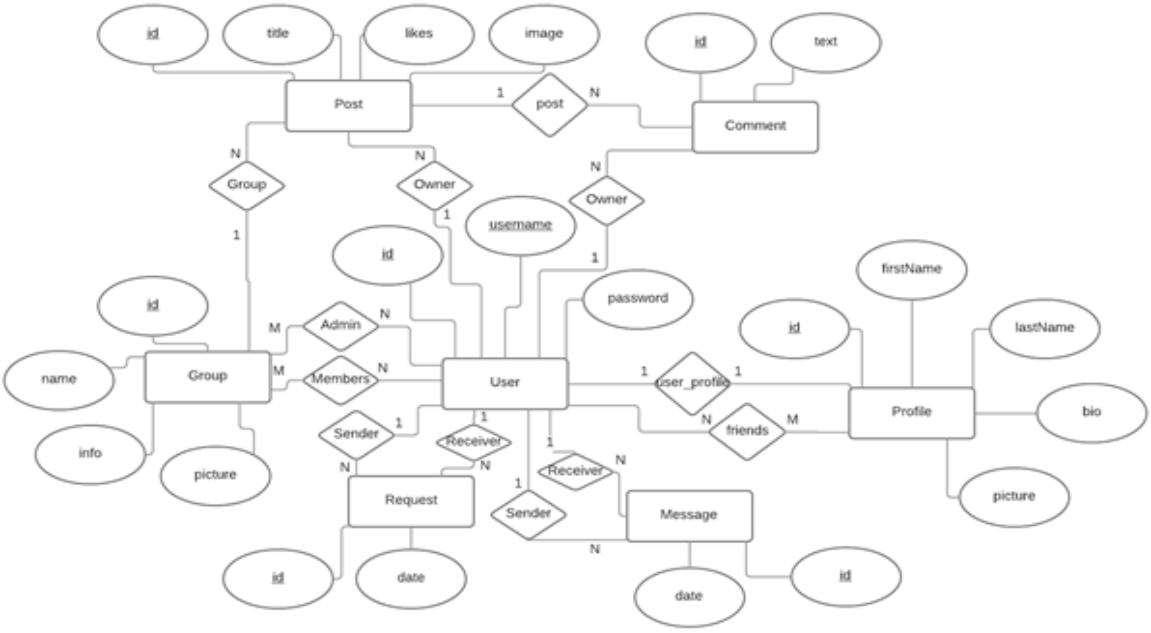
6 Other Requirements

The system data shall be backed up every night (full back-up) with a cycle of 30 days. This essentially means that there will be a provision to rollback by a month. Post back-up everyday the back-up shall be restored on a dummy production system to ensure completeness and correctness of back-up. Post that the dummy production database shall be purged.

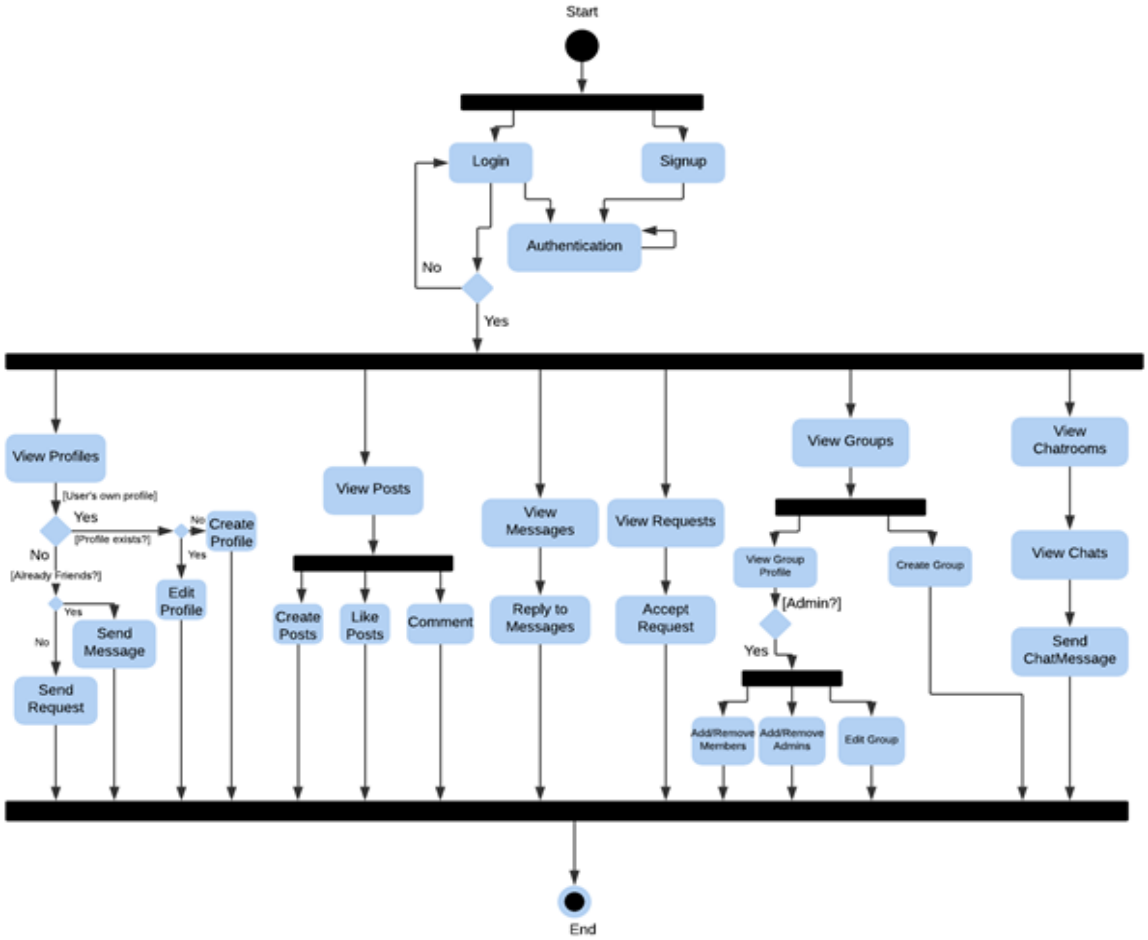
Appendix : Analysis Models

Use Case Diagram:

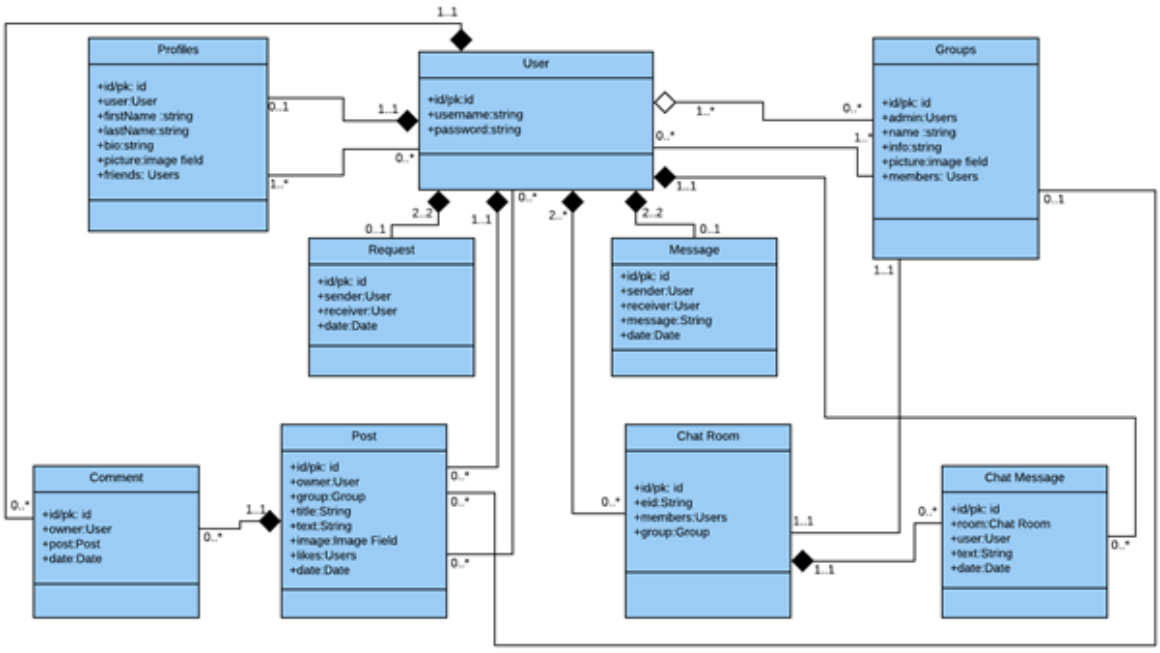
ER Diagram:



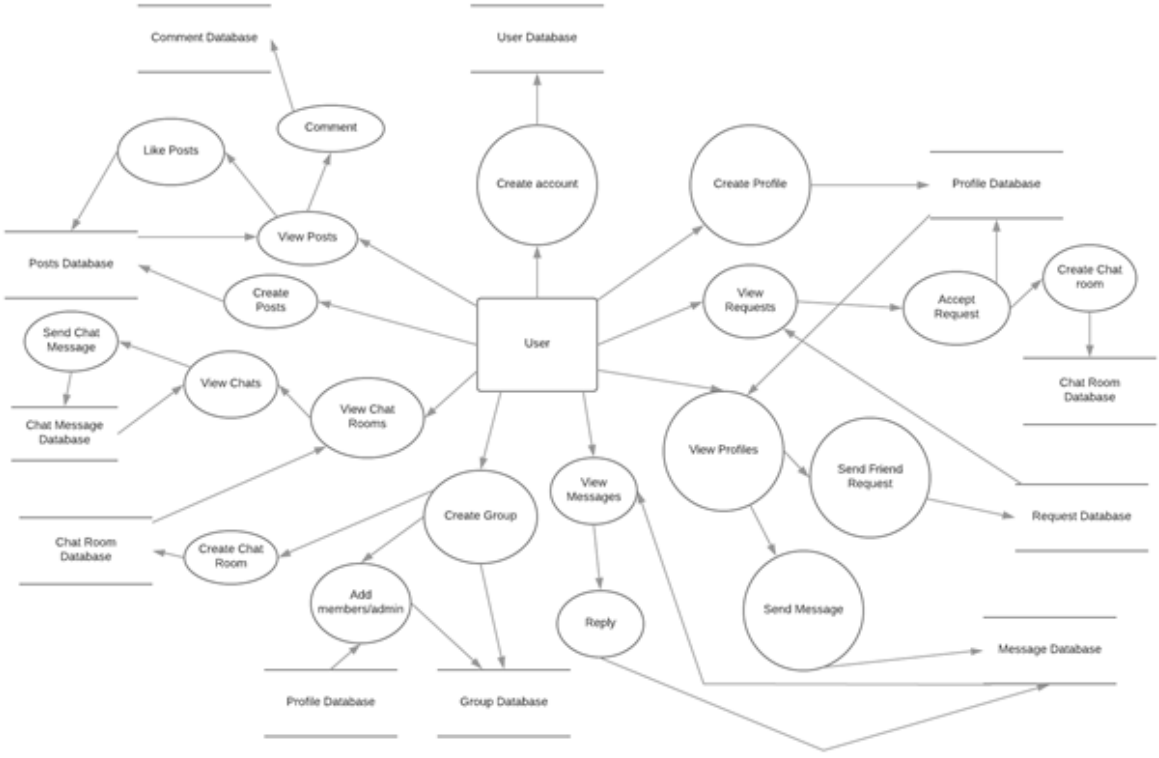
Navigation Diagram



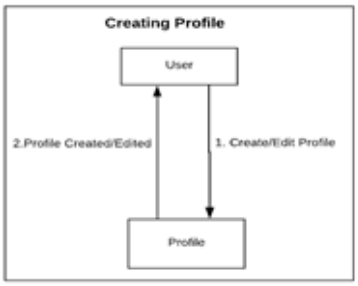
Class Diagram:

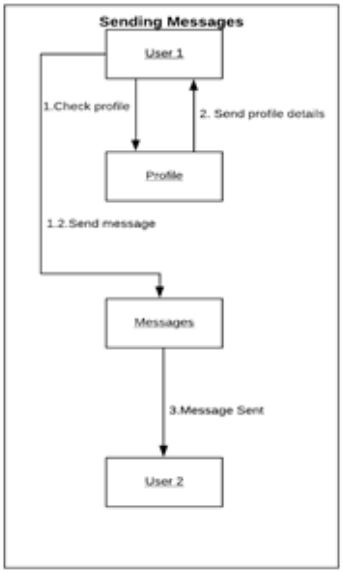


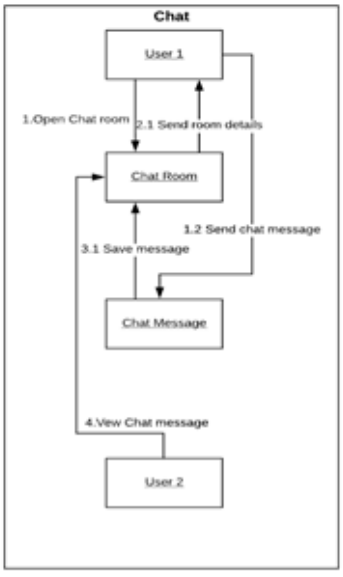
Data Flow Diagram:

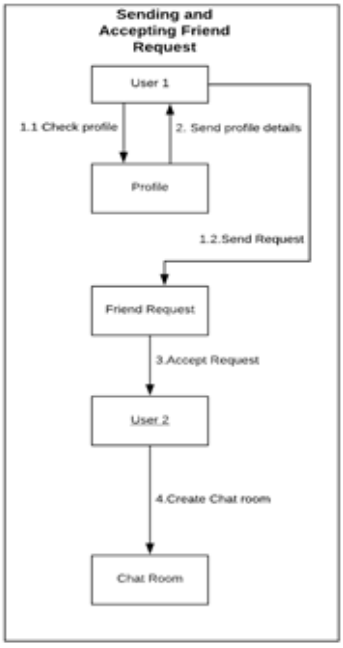


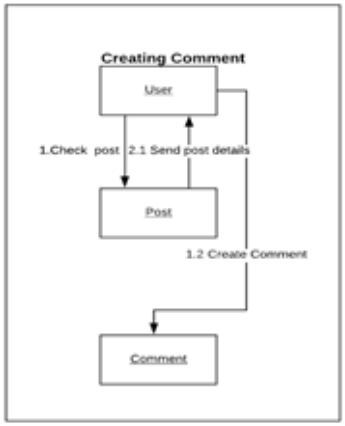
Collaboration Diagram:

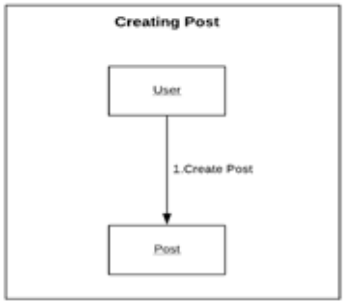


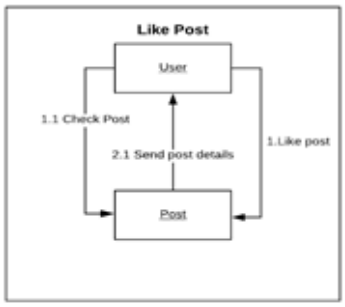


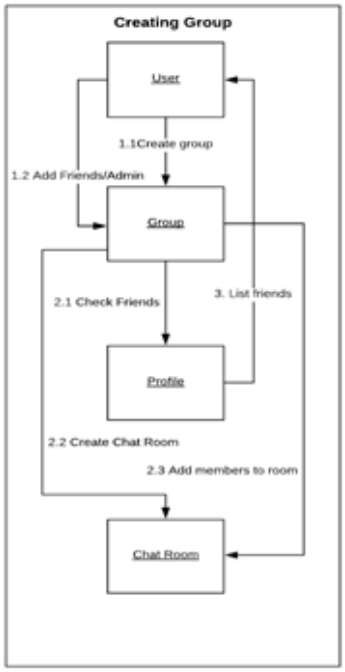




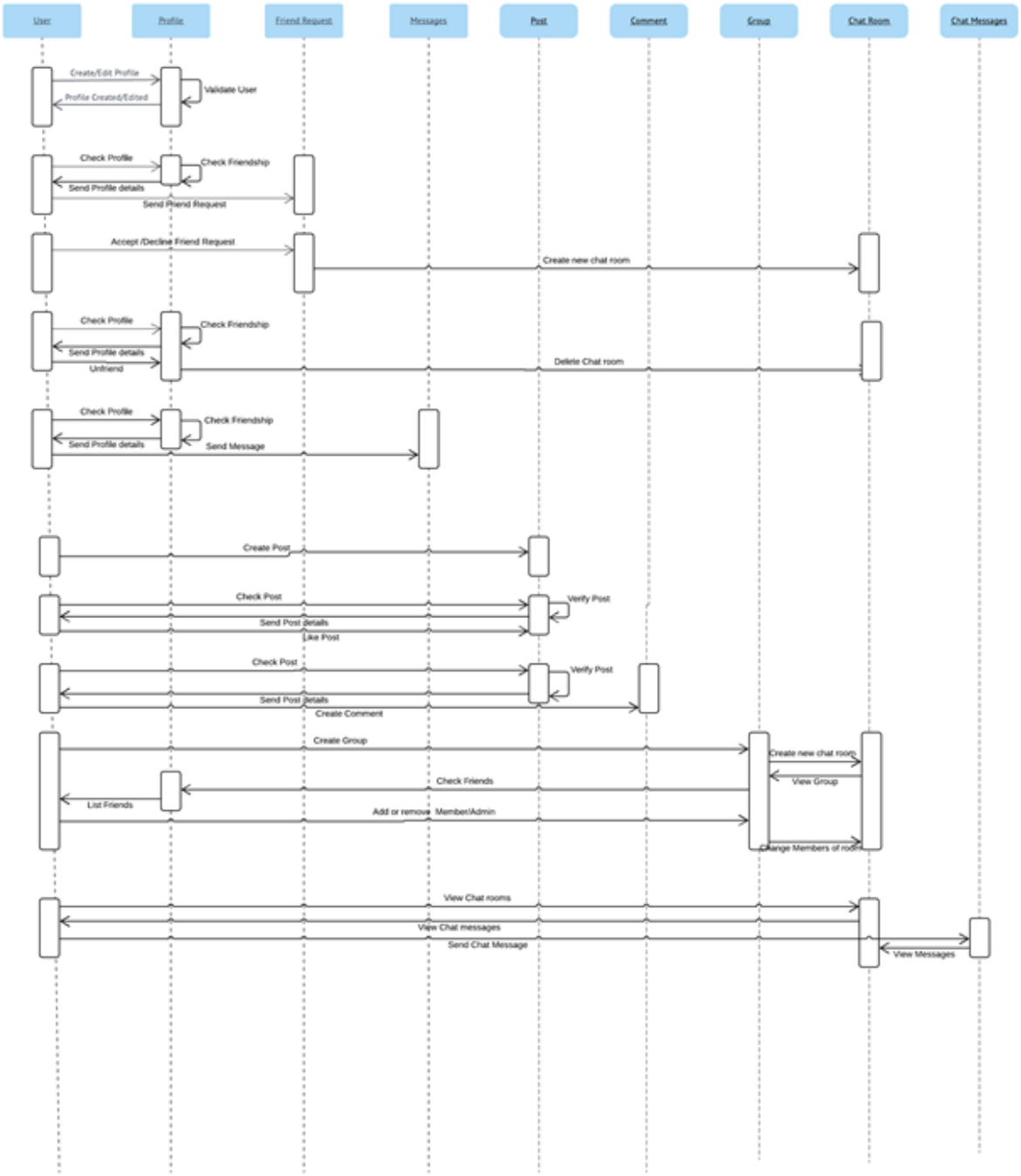




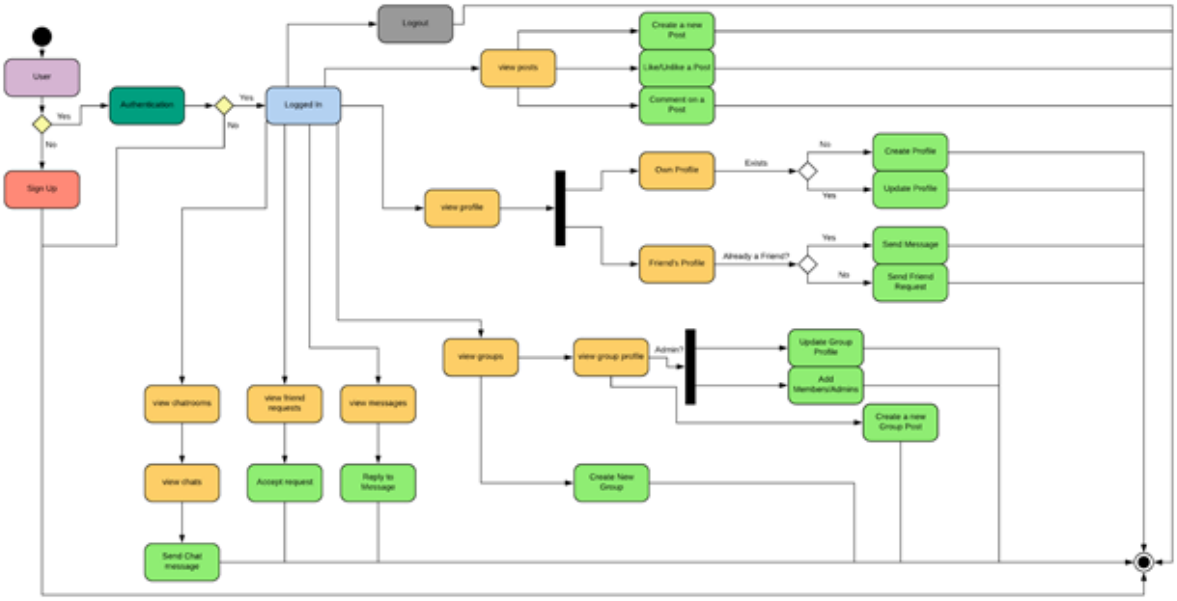




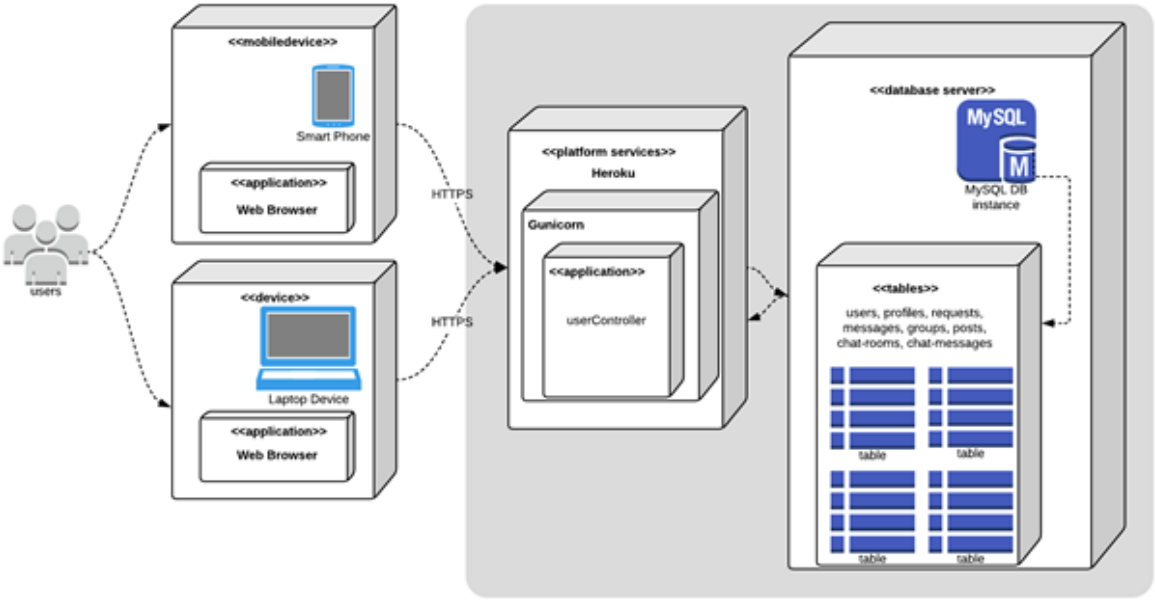
Sequence Diagram:



State Diagram:



Deployment Diagram:



MANUAL TEST CASE DESIGN

Test No	Function	Pre Conditions	Test Description(steps)	Expected Outcome	Outcome
1	Signup	System is setup and functional	1. User clicks on signup button 2. User enter email as username 3. User enters password 4. User reenters password 5. User clicks signup button	Page displays profile page for user and an option to logout	
2	Signup	System is setup and functional	1. User clicks on signup button 2. User enter email as username 3. User enters password 4. User reenters password but different then the above first password 5. User clicks register button	Page displays error message saying second password field doesn't match	
3	Login	System is setup and functional	1. User enter email as username 2. User enters password	Page displays profile page for user and an option to logout	
4	Login	System is setup and functional	1. User enter email as username 2. User enters password	Page displays error message saying password field doesn't match	
5	View Profile	User has logged in the system	1. User searches for a profile 2. User clicks view profile	Page displays the profile to be viewed	
6	View Profile	No user has logged in the system	1. User searches for a profile 2. User clicks view profile	Page renders the login page	
7	View Friend List	User has logged in the system	1. User is on his profile page 2. User clicks friend list	Page displays the friend list of the user	
8	View Friend List	No user has logged in the system	1. User is on his profile page 2. User clicks friend list	Page renders the login page	
9	Friend Request	User has logged in the system	1. User is on a profile page 2. User clicks send friend request	Friend Request is sent	
10	Friend Request	No user has logged in the system	1. User is on a profile page 2. User clicks send friend request	Page renders the login page	
11	Messages	User has logged in the system	1. User is on a profile page 2. User clicks send message	Page renders a message form	
12	Messages	No user has logged in the system	1. User is on a profile page 2. User clicks send message	Page renders the login page	

13	Files	User has logged in the system	1. User is on a profile page 2. User clicks send file	Page renders a file form	
14	Files	No user has logged in the system	1. User is on a profile page 2. User clicks send file	Page renders the login page	
15	Create Groups	User has logged in the system	1. User enters group name 2. User enters group bio 3. User uploads group pic	Group is created	
16	Create Groups	No user has logged in the system	1. User enters group name 2. User enters group bio 3. User uploads group pic	Page renders the login page	
17	View Groups	User has logged in the system	1. User searches for a group 2. User clicks view group	Page renders the group page	
18	View Groups	No user has logged in the system	1. User searches for a profile 2. User clicks view profile	Page renders the login page	
19	View ChatRooms	User has logged in the system	1. User searches for chatroom 2. User clicks chat	Page renders the chat page	
20	View ChatRooms	No user has logged in the system	1. User searches for chatroom 2. User clicks chat	Page renders the login page	
21	Chat	User has logged in the system	1. User enters a message 2. User sends the message	Chat message is sent	
22	Chat	No user has logged in the system	1. User enters a message 2. User sends the message	Page renders the login page	
23	View Posts	User has logged in the system	1. User clicks the home button or is redirected to home page	Page renders the home page	
24	View Posts	No user has logged in the system	1. User clicks the home button or is redirected to home page	Page renders the login page	
25	Create Posts	User has logged in the system	1. User enters post title 2. User enters post text 3. User uploads image/file	New Post is created	
26	Create Posts	No user has logged in the system	1. User enters post title 2. User enters post text 3. User uploads image/file	Page renders the login page	
27	Like Posts	User has logged in the system	1. User clicks the like/dislike button for a post	The likes count is incremented/ decremented	
28	Like Posts	No user has logged in the system	1. User clicks the like/dislike button for a post	Page renders the login page	

29	Comment on Posts	User has logged in the system	1. User enters a comment for the specific post 2. User clicks on comment	The comment is added to the post	
30	Comment on Posts	No user has logged in the system	1. User enters a comment for the specific post 2. User clicks on comment	Page renders the login page	

TOOLS USED

1. Django Unit Testing

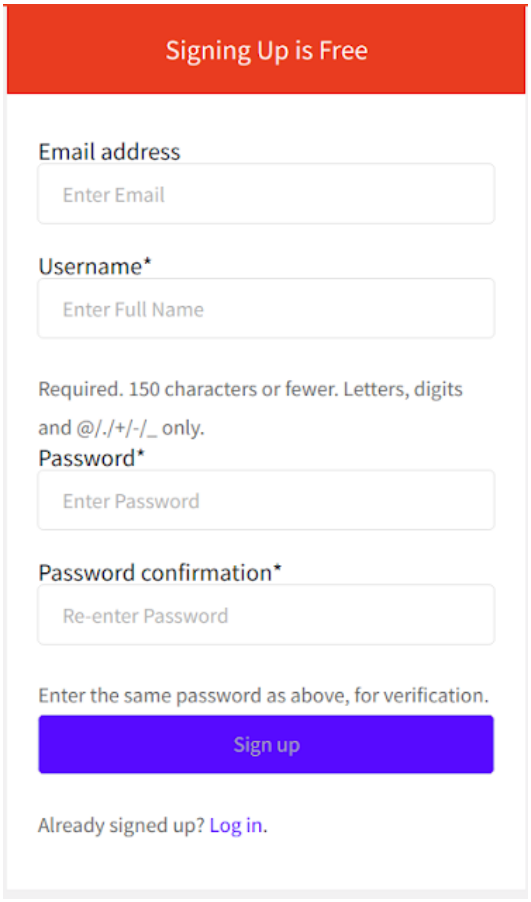
In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.
Automated testing is an extremely useful bug-killing tool for the modern Web developer. We can use a collection of tests – a test suite – to solve, or avoid, a number of problems. When we are writing new code, we can use tests to validate our code works as expected. When we are refactoring or modifying old code, we can use tests to ensure your changes haven't affected the application's behavior unexpectedly.
Testing a Web application is a complex task, because a Web application is made of several layers of logic – from HTTP-level request handling, to form validation and processing, to template rendering. With Django's test-execution framework and assorted utilities, we can simulate requests, insert test data, inspect your application's output and generally verify our code is doing what it should be doing.
The preferred way to write tests in Django is using the unittest module built in to the Python standard library. We can also use any other Python test framework; Django provides an API and tools for that kind of integration.

2. Lighthouse

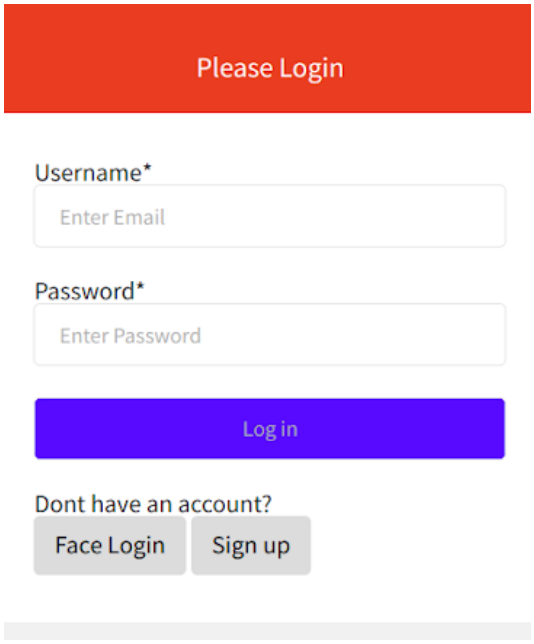
Lighthouse is an open-source, automated tool for improving the quality of web pages. We can run it against any web page, public or requiring authentication. It has audits for performance, accessibility, progressive web apps, and more.
We can run Lighthouse in Chrome DevTools, from the command line, or as a Node module. We give Lighthouse a URL to audit, it runs a series of audits against the page, and then it generates a report on how well the page did. From there, use the failing audits as indicators on how to improve the page. Each audit has a reference doc explaining why the audit is important, as well as how to fix it.
It audits the URL of a web app and generates a report telling you how bad and good our web app is according to web standards and developers best practices. Also attached to each section of the report is documentation explaining why that part of our app was audited, why we should improve that part of the app and how to fix it.

Screenshots

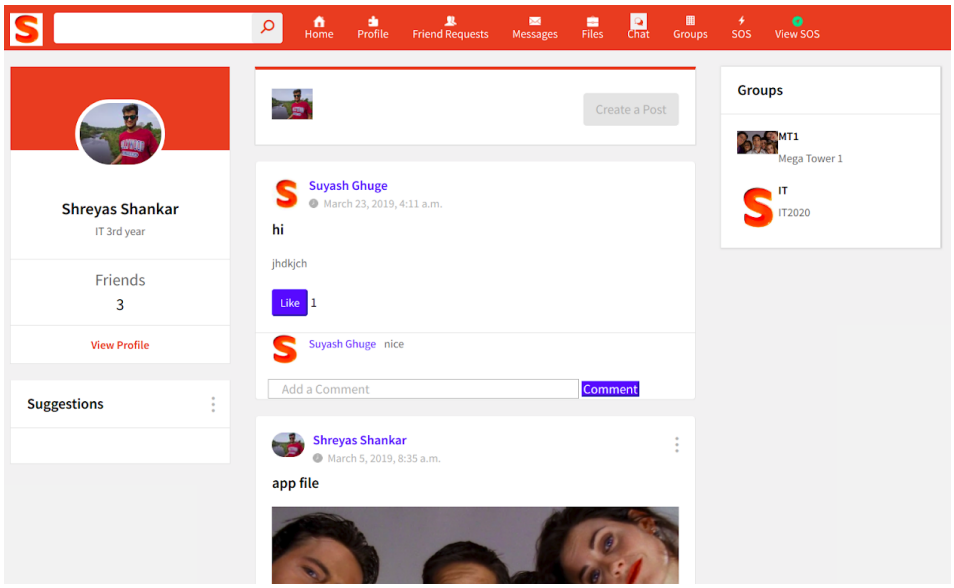
1. Signup Page



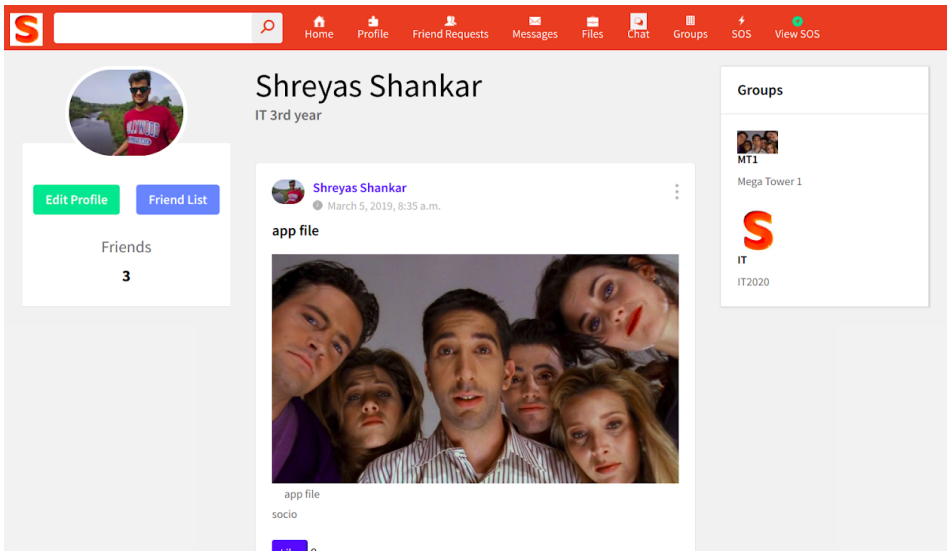
2. Login Page



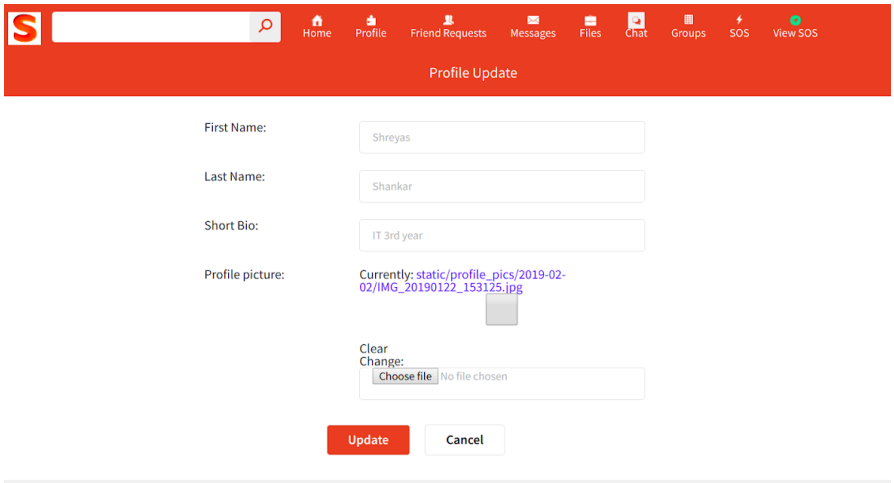
3. Home Page



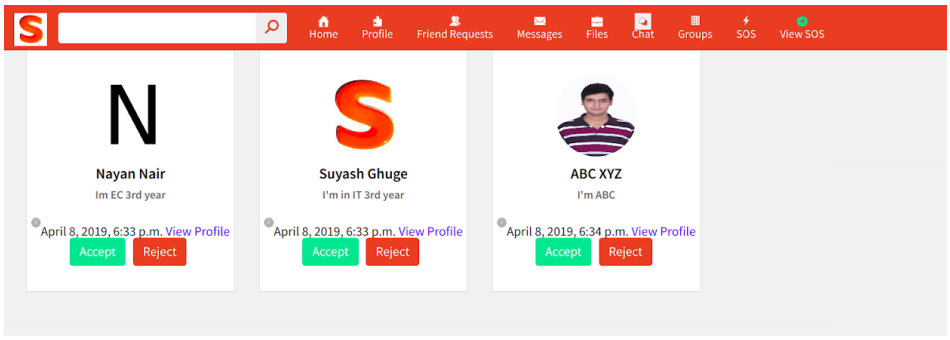
4. Profile Page



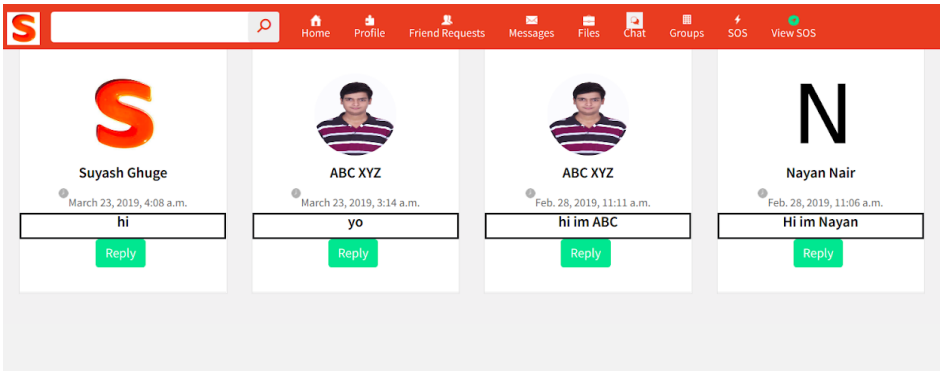
5. Update Profile Page



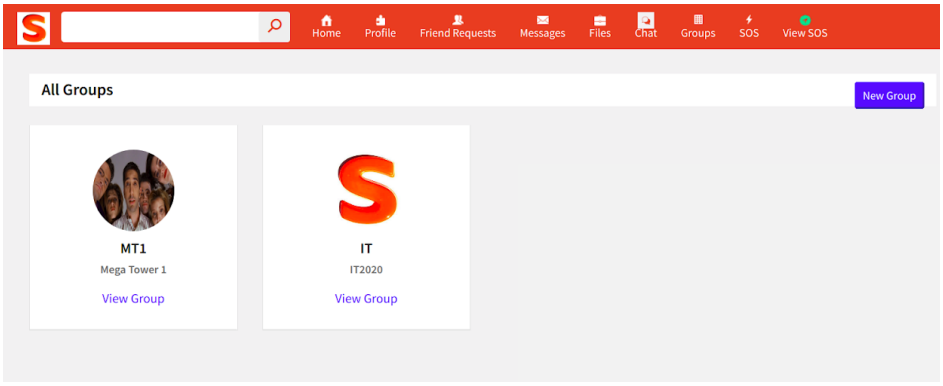
6. Requests Page



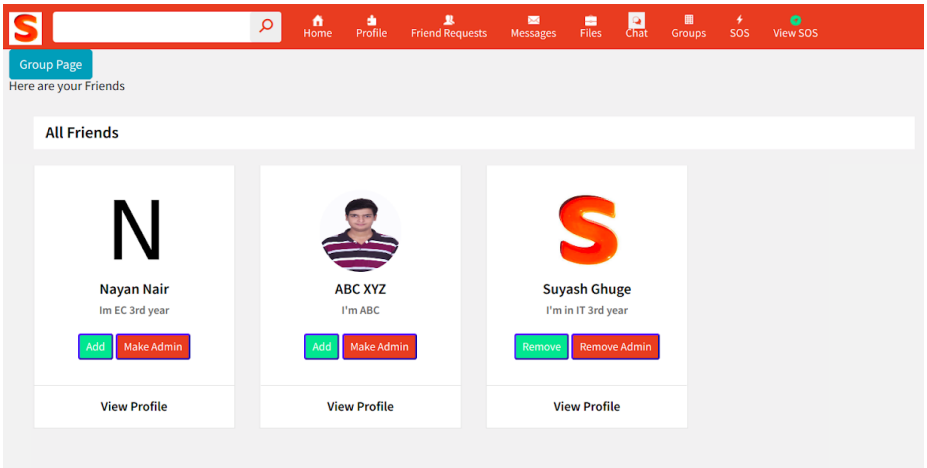
7. Messages Page



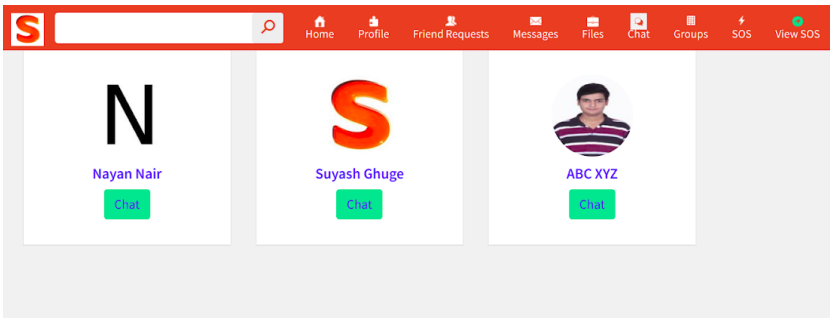
8. Groups Page



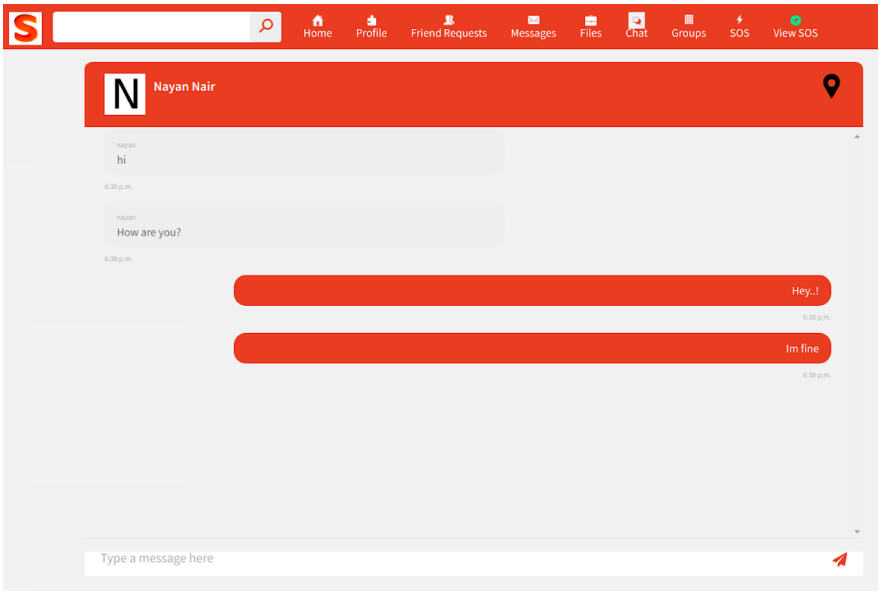
9. Group Members Page



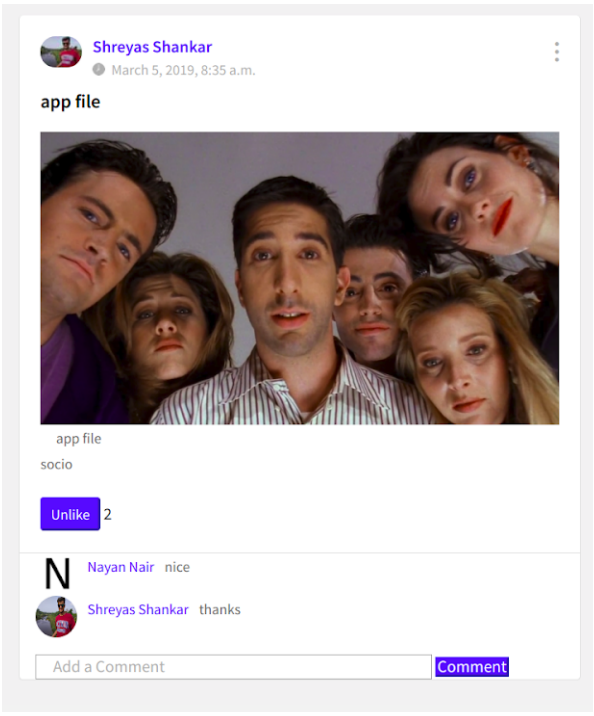
10. Chat Rooms Page



11. Chats Page



12. Post Page



AUTOMATION TESTING

```
(venv) shreyas@shreyas-ubuntu:/mnt/368847BE88477AFF/6th sem/Projects/Socio-SE/Socio$ python manage.py test accounts
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Testing Login Page with invalid credentials

Login failed

.Testing Login Page

Login page verified

.Testing Logout Page

Logout page verified

.Testing Signup Page with invalid credentials

Signup Failed

.Testing SignUp Page

SignUp page verified

.
----------------------------------------------------------------------
Ran 5 tests in 0.468s

OK
Destroying test database for alias 'default'...
```

```
(venv) shreyas@shreyas-ubuntu:/mnt/368847BE88477AFF/6th sem/Projects/Socio-SE/Socio$ python manage.py test profiles
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Testing friendlist Page with login

friendlist page verified

.Testing friendlist Page without login

friendlist page not working without login

.Testing profile Page with login

profile page verified

.Testing profile Page without login

profile page not working without login

.
----------------------------------------------------------------------
Ran 4 tests in 0.375s

OK
Destroying test database for alias 'default'...
```

```
(venv) shreyas@shreyas-ubuntu:/mnt/368847BE88477AFF/6th sem/Projects/Socio-SE/Socio$ python manage.py test requests message
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Testing requests Page with login

requests page verified

.Testing requests Page without login

requests page not working without login

.Testing files Page with login

files page verified

.Testing files Page without login

files page not working without login

.Testing messages Page with login

messages page verified

.Testing messages Page without login

messages page not working without login

.
----------------------------------------------------------------------
Ran 6 tests in 0.615s

OK
Destroying test database for alias 'default'...
```

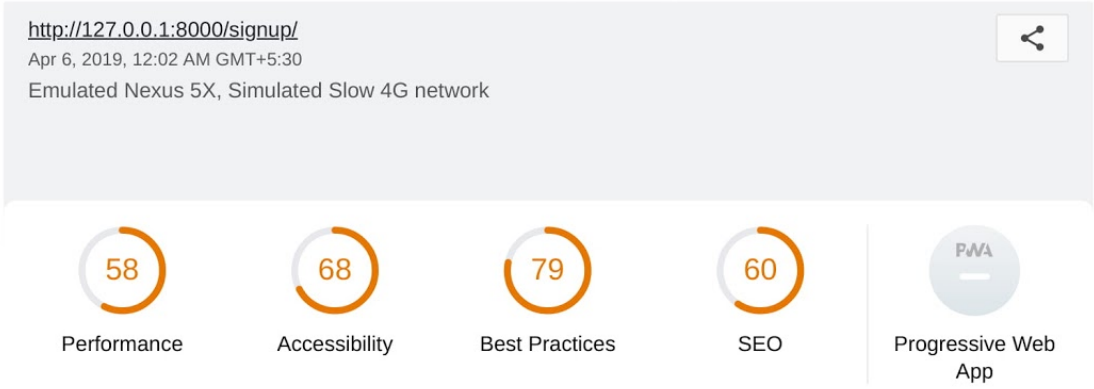
```
(venv) shreyas@shreyas-ubuntu:/mnt/368847BE88477AFF/6th sem/Projects/Socio-SE/Socio$ python manage.py test groups
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Testing groups create Page with login

groups create page verified

.Testing groups create Page without login

groups create page not working without login

.Testing groups Page with login

groups page verified

.Testing groups Page without login

groups page not working without login

.
----------------------------------------------------------------------
Ran 4 tests in 0.356s

OK
Destroying test database for alias 'default'...
```

```
(venv) shreyas@shreyas-ubuntu:/mnt/368847BE88477AFF/6th sem/Projects/Socio-SE/Socio$ python manage.py test chat
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Testing chat Page with login

chat page verified

.Testing chat Page without login

chat page not working without login

.Testing save_message Page with login

save_message page verified

.Testing save_message Page with login

save_message page not working

.
----------------------------------------------------------------------
Ran 4 tests in 0.381s

OK
Destroying test database for alias 'default'...
```

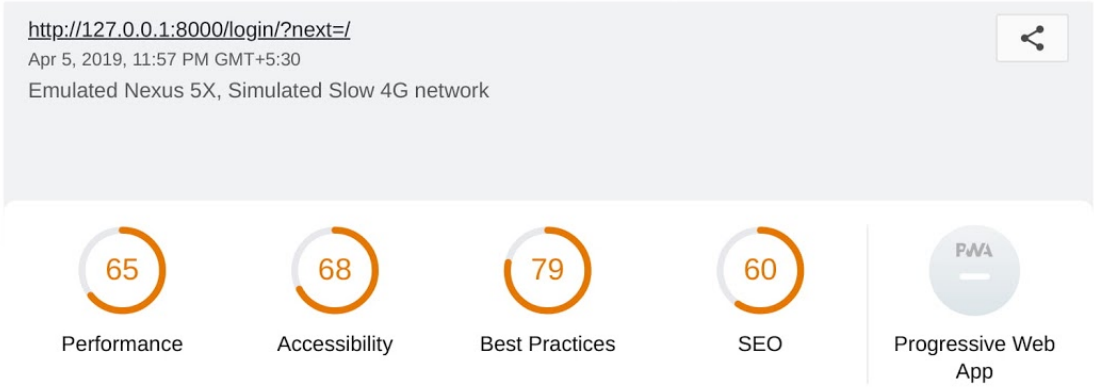
```
(venv) shreyas@shreyas-ubuntu:/mnt/368847BE88477AFF/6th sem/Projects/Socio-SE/Socio$ python manage.py test posts/
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Testing posts comment Page with login

posts comment page verified

.Testing posts comment Page without login

posts comment page not working

.Testing posts create Page with login

posts create page verified

.Testing posts create Page without login

posts create page not working without login

.Testing posts like Page with login

posts like page verified

.Testing posts like Page without login

posts like page not working without login

.Testing posts Page with login

posts page verified

.Testing posts Page without login

posts page not working without login

.
----------------------------------------------------------------------
Ran 8 tests in 0.752s

OK
Destroying test database for alias 'default'...
```

PERFORMANCE TESTING

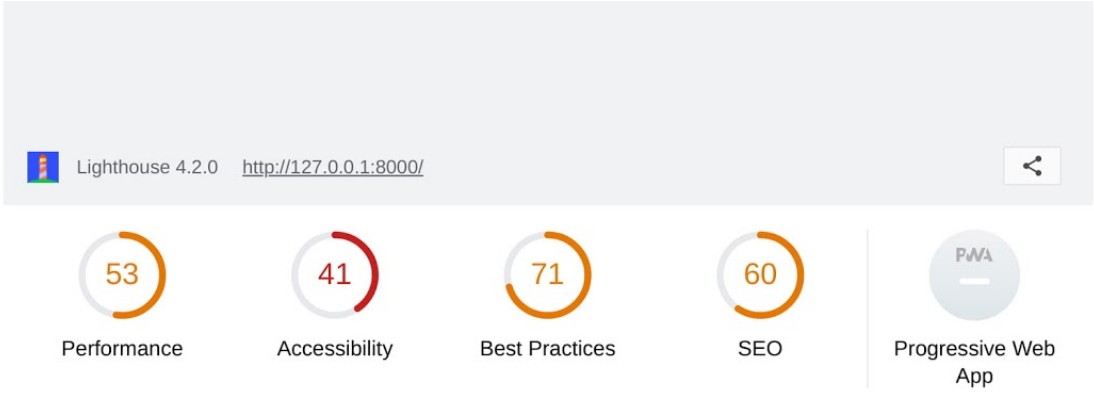
1. Signup



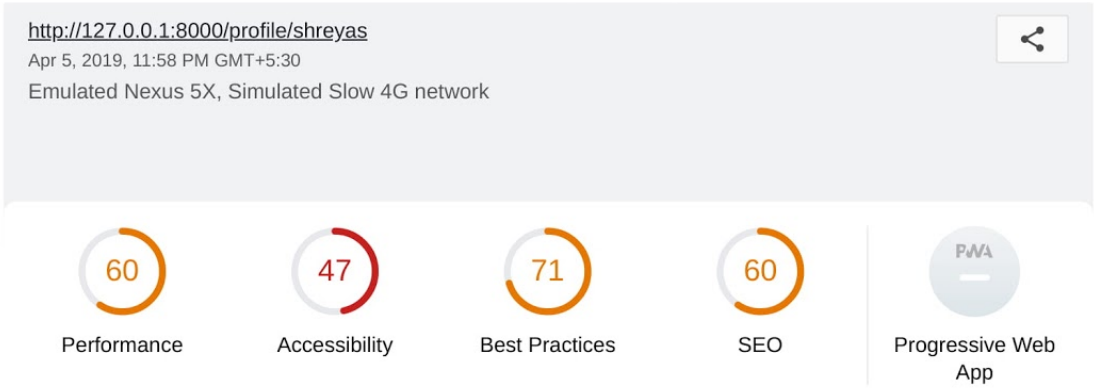
2. Login



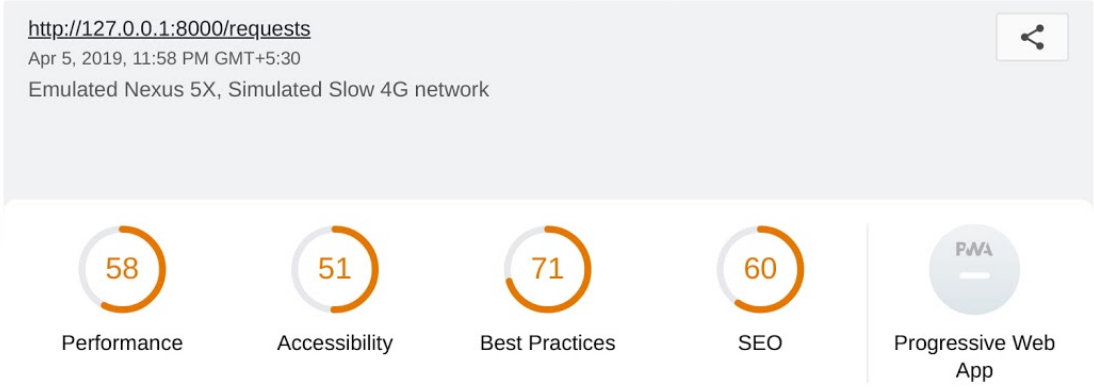
3. Posts



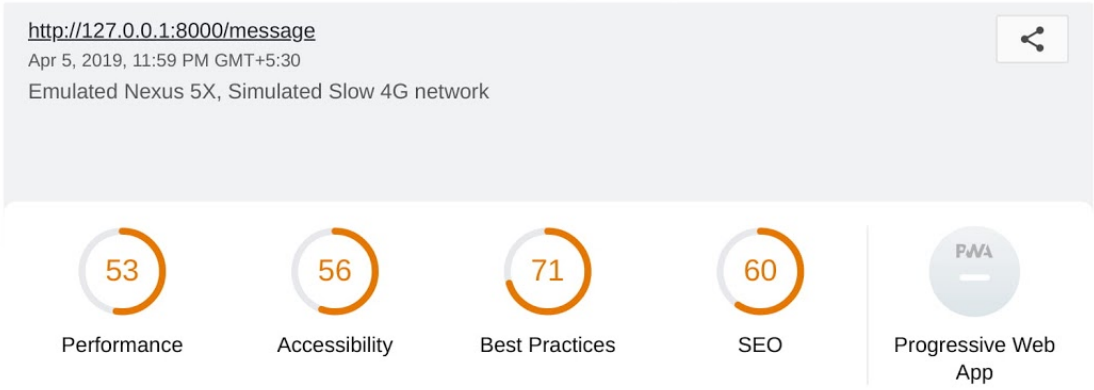
4. Profile



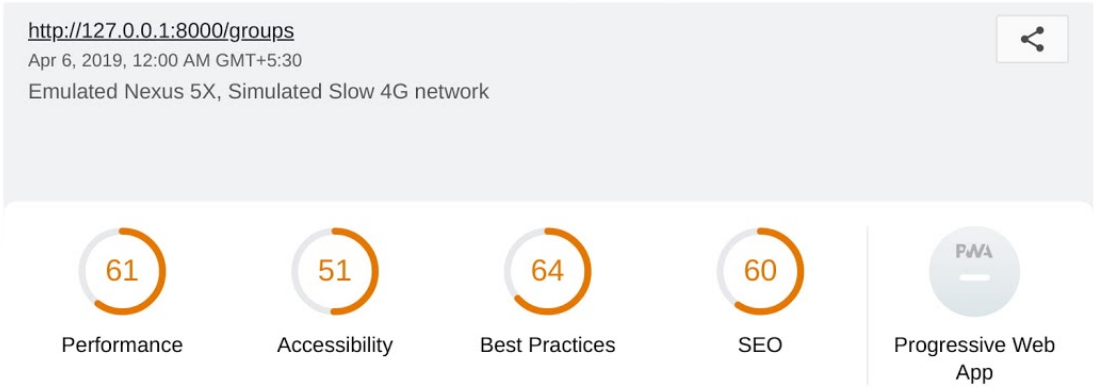
5. Requests



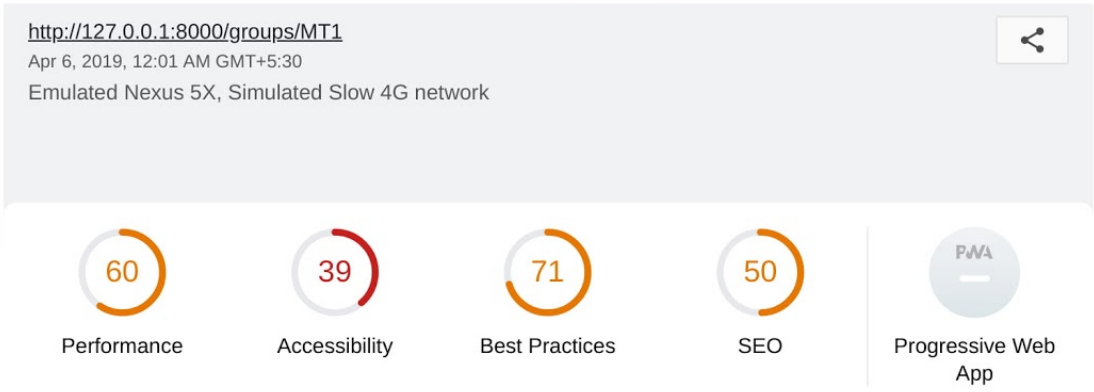
6. Messages



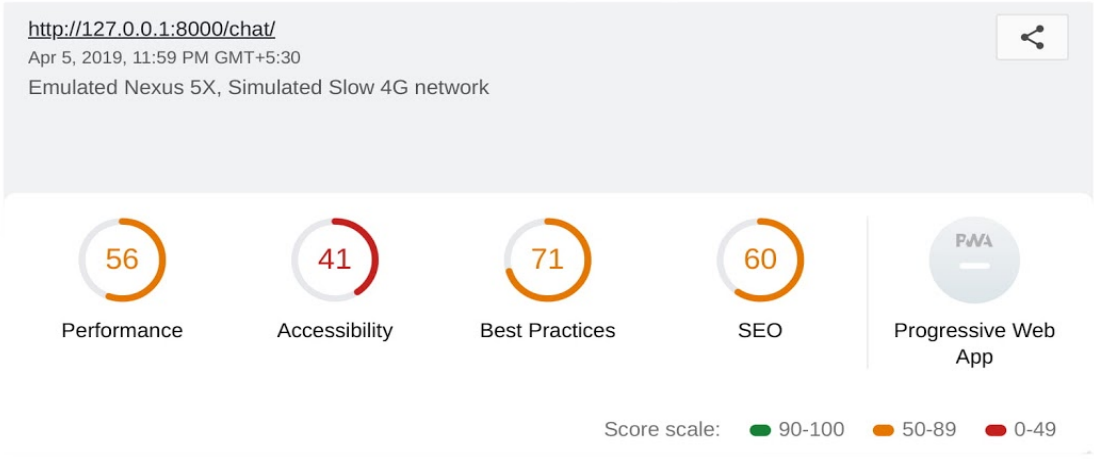
7. Groups



8. Group_page



9. Chatrooms



10. Chats