Student Name: Ertan Kaya

Course Name: SWE 573 – Software Development Practice

Date: 03.01.2023

E-Pocket Social Media App Project

System Introduction Video URL: https://youtu.be/JU48iP_THuA

Git Repository: https://github.com/ertantherock/SWE-573

Git Tag Version: https://github.com/ertantherock/SWE-573/releases/tag/v0.9

Deployment URI: http://34.170.4.216/

HONOR CODE

Related to the submission of all the project deliverables for the Swe573 2022 Fall semester project reported in this report, I Ertan Kaya declare that: - I am a student in the Software Engineering MS program at Bogazici University and am registered for Swe573 course during the 2022 Fall semester. - All the material that I am submitting related to my project (including but not limited to the project repository, the final project report, and supplementary documents) have been exclusively prepared by myself. - I have prepared this material individually without the assistance of anyone else with the exception of permitted peer assistance which I have explicitly disclosed in this report.

Student Name-Surname: Ertan Kaya

Signature:

Contents

1. Test User Name and Password

Username: suzanuskudarli

Password: SuzanUskudarli1

2. Project Declaration

This project which I am submiting is performed by myself and there is no need to declarate any software licenses.

3. Project Details

3.1 Overview

This project is a social media app that user can share posts including content, label, link and post title.

Project also has a sign up page that user can sign up to the system and also login page that with required credantials, user can login to the system.

User also can change profile picture and his/her username at the profile page.

3.2 Software Requirements Specification

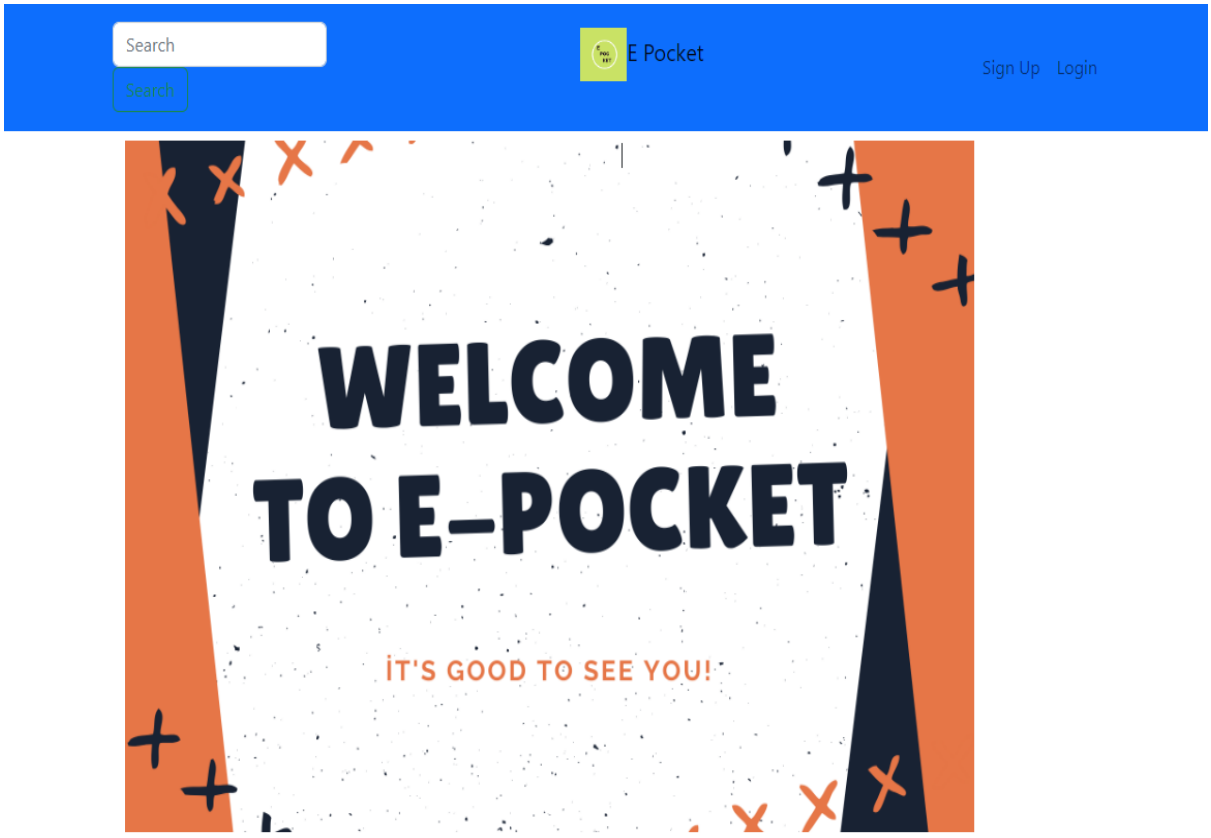
- The user can sign up to the system.
- The user can login to the system.
- The user can share post links.
- The user can share content about post.
- The user can share title with post
- The user can share label about post.
- The user can change his/her profile picture.
- The user can change his user name.

- The user can like posts.
- The user can search posts.
- The user can comment to the posts.
- The user can delete the post which they shared.
- The user can share images with post
- The user can see other users.

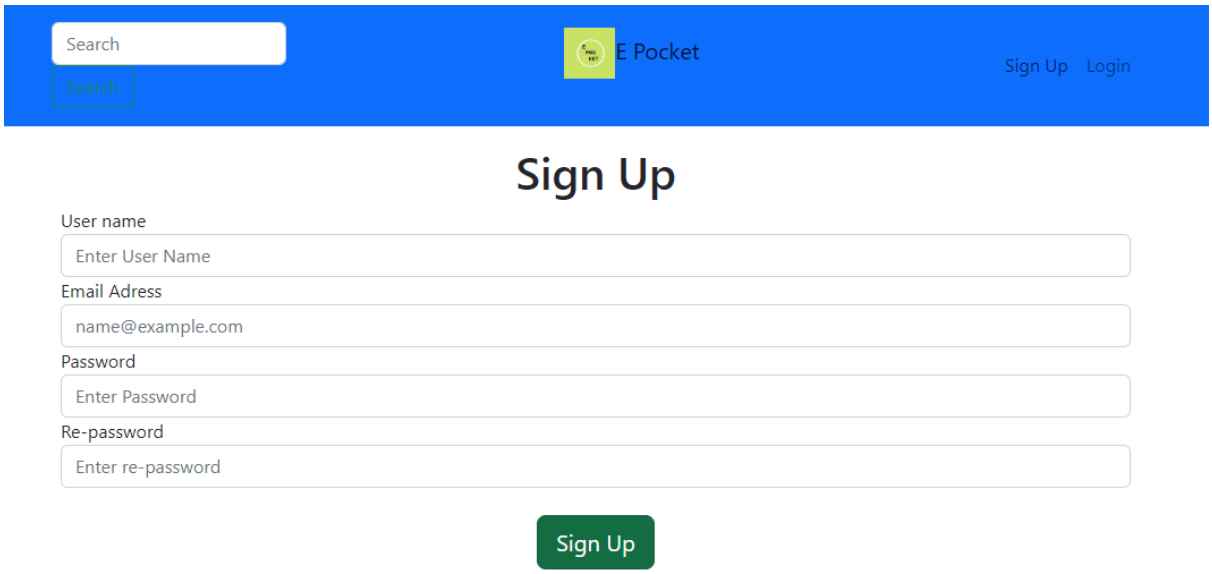
3.3 Project Pictures and UML Diagram

3.3.1 Project Pictures

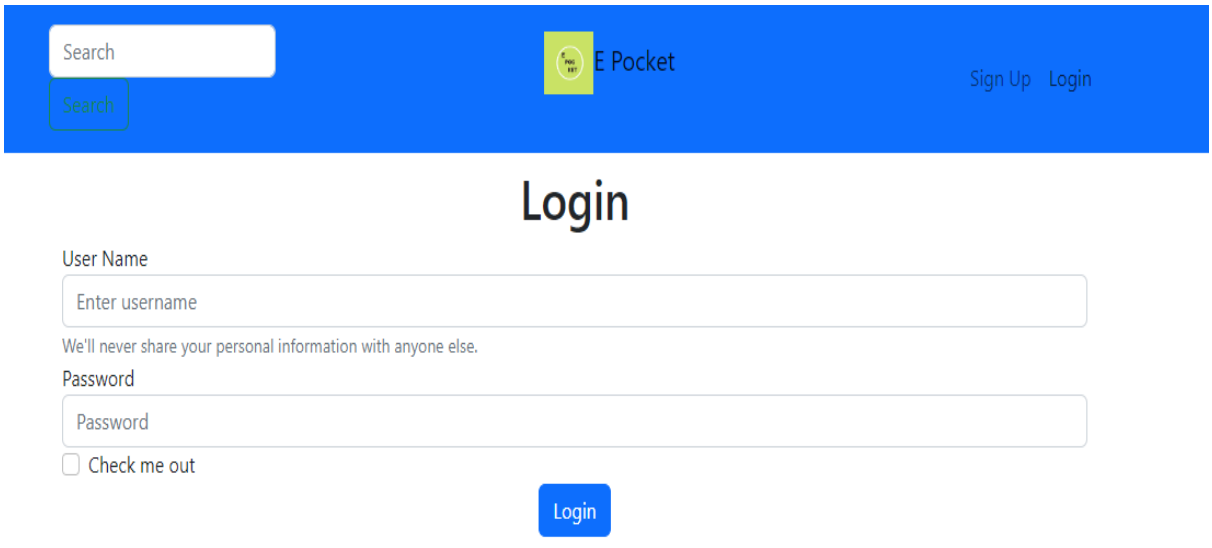
3.3.1.1 Landing Page



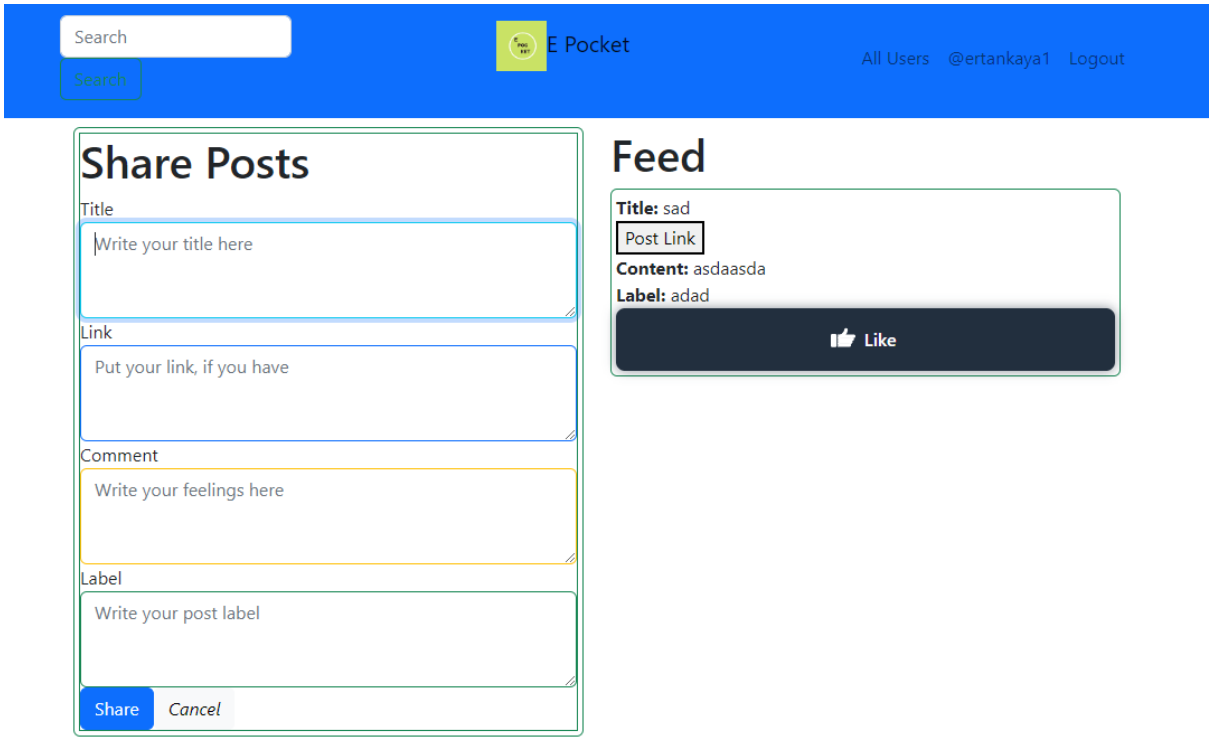
3.3.1.2 Signup Page



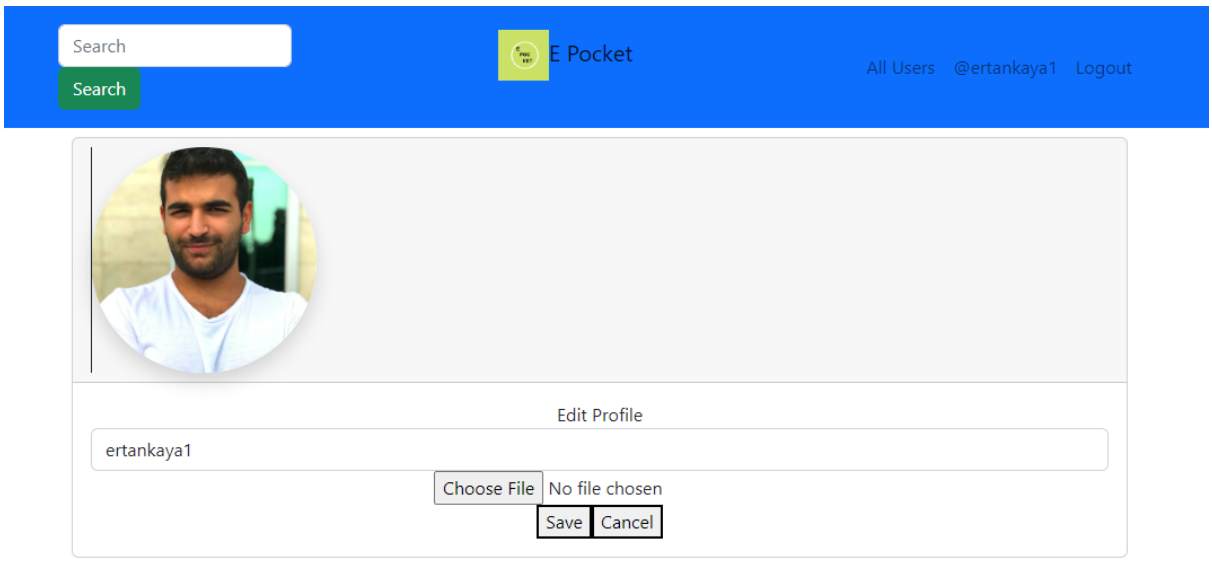
3.3.1.3 Login Page



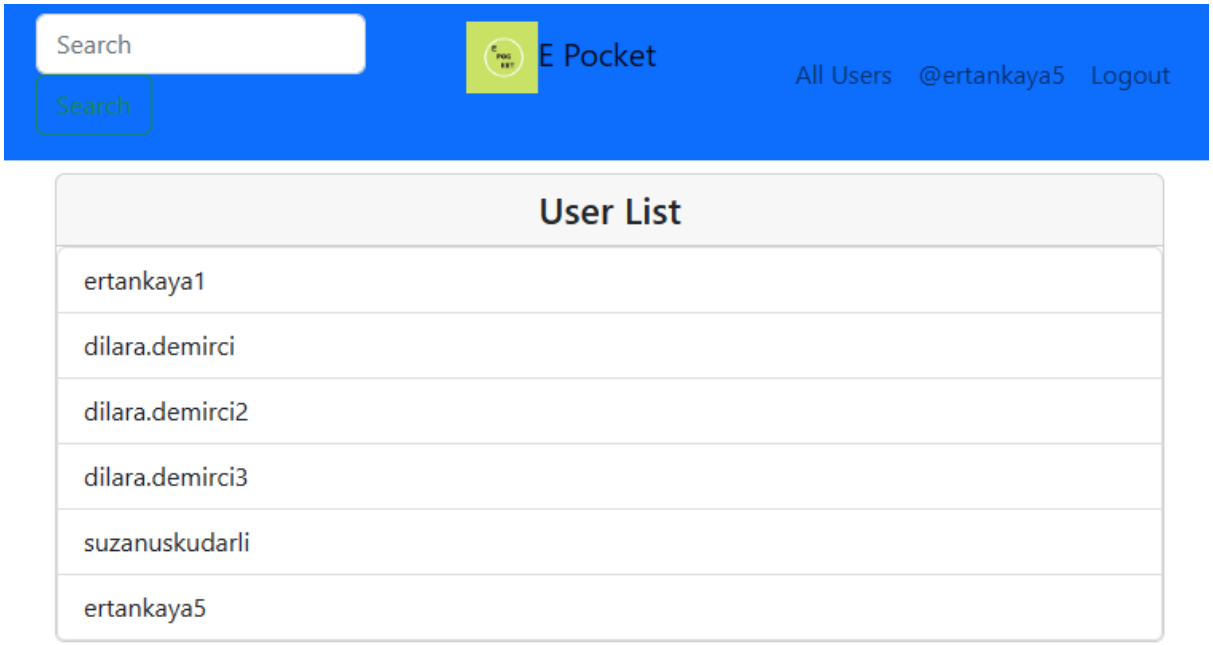
3.3.1.4 Post Page



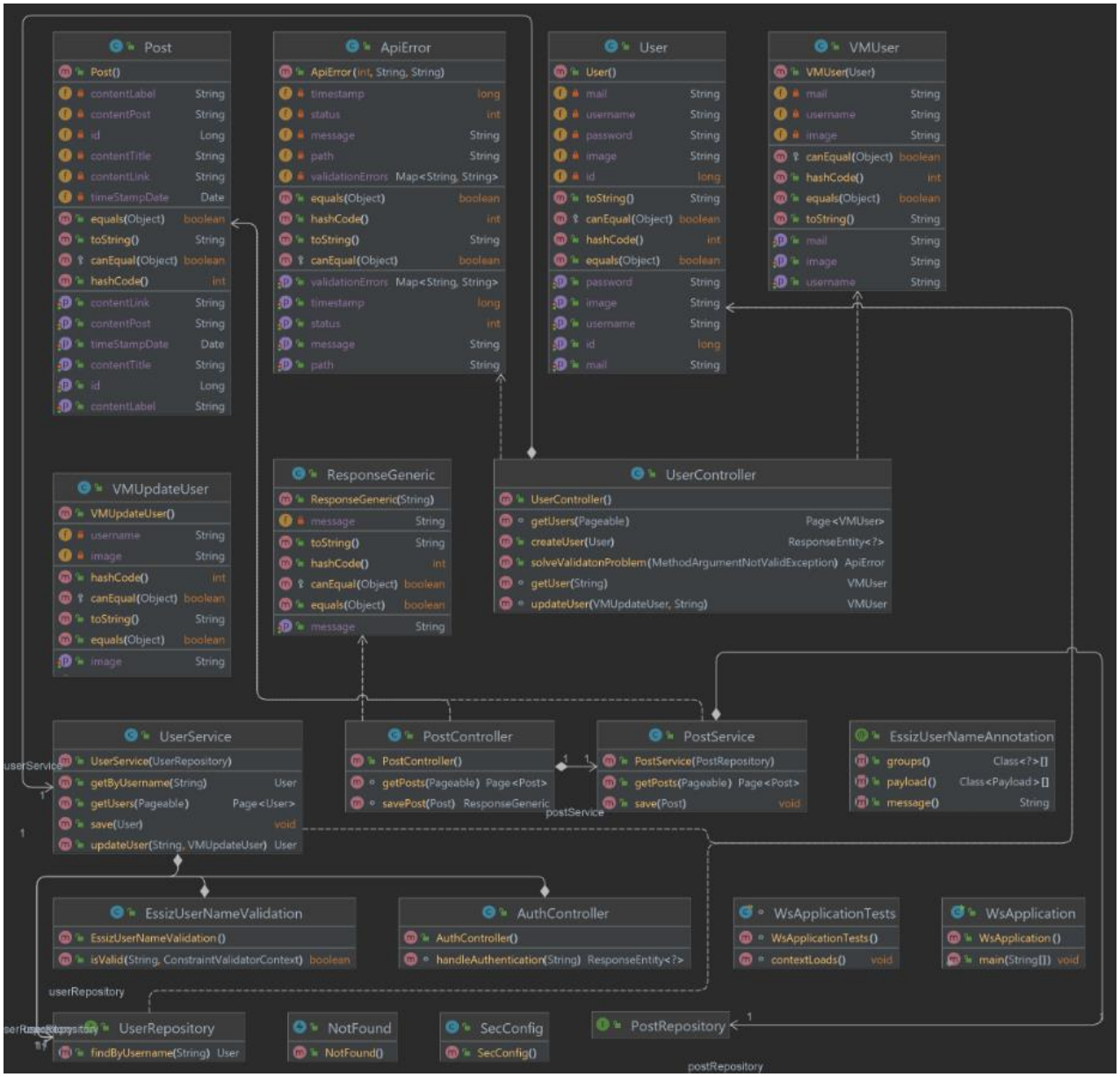
3.3.1.5 User Page



3.3.1.6 All User List Page



3.4 UML Diagram



4. Project Status

Users can sign up and login to the system. Also they can share their posts with Content, Title, Link and Label. They can update their profile pictures and usernames. Uses cannot delete their posts, search bar is not filtering posts, users cannot comment out, users cannot save posts and cannot share images.

5. Status of Deployment

I have deployed the application 2 times in 2 different platform. First one was on Azure but Microsoft has closed my Student account because I have used all of my free credits. Before the deploy the application on Azure, I have dockerized my application on Docker and published it on docker.hub. After Azure, I have deployed my application on google cloud. I have created a Virtual Machine on Google Cloud and setup my .jar file on this Virtual Machine. This VM is working 7/24 hours.

URL of my deployed application is: http://34.170.4.216/.

6. Requirements for Dockerization

- Build React project with npm run build command.
- Copy files from Build to Java-Resources-Static folder.
- Build .jar file with mvn package command.
- Create a Docker file in Java project named Dockerfile.
- Dockerfile needs to get below instructions:
  - FROM openjdk:17
  - LABEL mainteiner="javaguides.net"
  - ADD target/ws-0.0.1-SNAPSHOT.jar epocket-docker.jar
  - ENTRYPOINT ["java","-jar","epocket-docker.jar"]

Run docker command: docker run -t -p 8080:8080 --name epocket-docker-container spring-mvc-sample-image .

7. User Manual

For the local usage:

Users can run .jar file with the command java -jar ws-0.0.1-SNAPSHOT.jar and the application will starts. Users can sign up and login to the system with localhost:8080 port. When user uses the

project in their local, they will not see any data on the website. So they need to sign up and after that login to platform.

Also user can dockerize the .jar file and start docker image with above instructions.

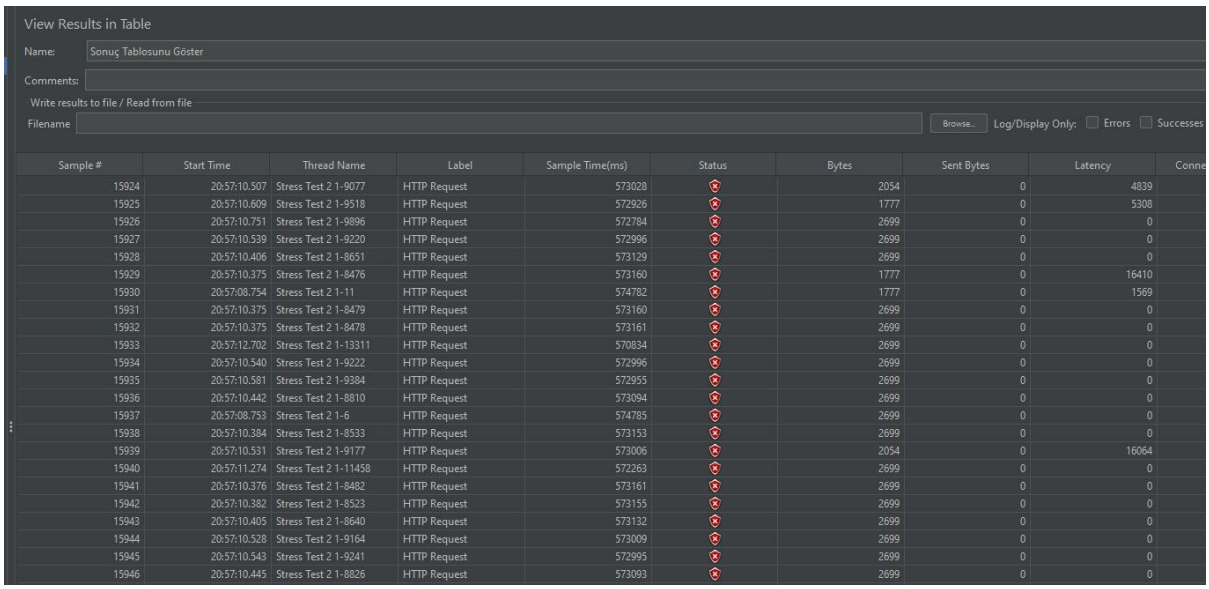
For the deployed link:

Users can connect to website with http://34.170.4.216/ link and they can signup and login with any browser. If user didn't sign up before to system, the user needs to sign up first and then they can login. If user logins with deployed link, user can see posts which shared before and also they can see users who have signed up before.

8. Test Results

Stress Test Results:

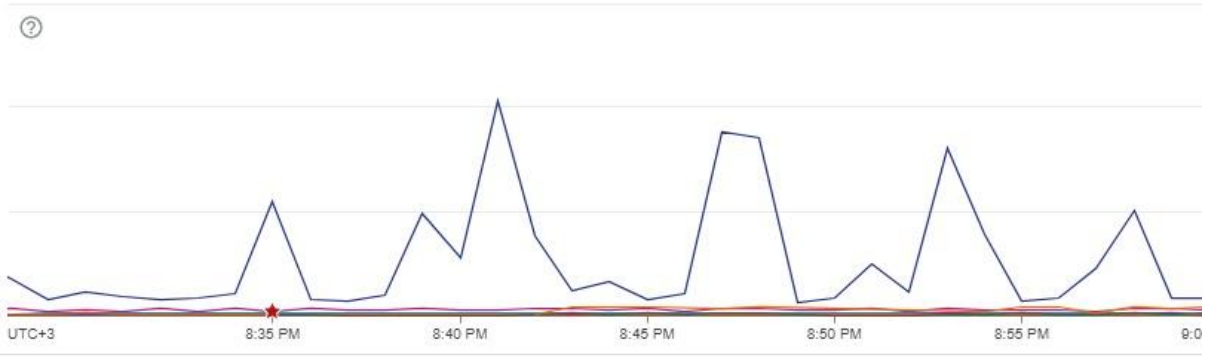
Stress Test made with Apache Jmeter. Up to 1000 Requests, website is handling very well. But after 1000 http requests, website's response time is getting higher and http requests are failing.



Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Conne
15924	20:57:10.507	Stress Test 2 1-9077	HTTP Request	573028		2054	0	4839	
15925	20:57:10.609	Stress Test 2 1-9518	HTTP Request	572926		1777	0	5308	
15926	20:57:10.751	Stress Test 2 1-9896	HTTP Request	572784		2699	0	0	
15927	20:57:10.539	Stress Test 2 1-9220	HTTP Request	572996		2699	0	0	
15928	20:57:10.406	Stress Test 2 1-8651	HTTP Request	573129		2699	0	0	
15929	20:57:10.375	Stress Test 2 1-8476	HTTP Request	573160		1777	0	16410	
15930	20:57:08.754	Stress Test 2 1-11	HTTP Request	574782		1777	0	1569	
15931	20:57:10.375	Stress Test 2 1-8479	HTTP Request	573160		2699	0	0	
15932	20:57:10.375	Stress Test 2 1-8478	HTTP Request	573161		2699	0	0	
15933	20:57:12.702	Stress Test 2 1-13311	HTTP Request	570834		2699	0	0	
15934	20:57:10.540	Stress Test 2 1-9222	HTTP Request	572996		2699	0	0	
15935	20:57:10.581	Stress Test 2 1-9384	HTTP Request	572955		2699	0	0	
15936	20:57:10.442	Stress Test 2 1-8810	HTTP Request	573094		2699	0	0	
15937	20:57:08.753	Stress Test 2 1-6	HTTP Request	574785		2699	0	0	
15938	20:57:10.384	Stress Test 2 1-8533	HTTP Request	573153		2699	0	0	
15939	20:57:10.531	Stress Test 2 1-9177	HTTP Request	573006		2054	0	16064	
15940	20:57:11.274	Stress Test 2 1-11458	HTTP Request	572263		2699	0	0	
15941	20:57:10.376	Stress Test 2 1-8482	HTTP Request	573161		2699	0	0	
15942	20:57:10.382	Stress Test 2 1-8523	HTTP Request	573155		2699	0	0	
15943	20:57:10.405	Stress Test 2 1-8640	HTTP Request	573132		2699	0	0	
15944	20:57:10.528	Stress Test 2 1-9164	HTTP Request	573009		2699	0	0	
15945	20:57:10.543	Stress Test 2 1-9241	HTTP Request	572995		2699	0	0	
15946	20:57:10.445	Stress Test 2 1-8826	HTTP Request	573093		2699	0	0	

:

Top Processes by CPU per VM (Avg. vCPUs)




```
top - 18:06:47 up  3:56,  1 user,  load average: 0.00, 0.00, 0.00
Tasks: 100 total,   1 running,  59 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.0 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1716276 total,   196600 free,   460296 used,  1059380 buff/cache
KiB Swap:        0 total,        0 free,        0 used.  1094632 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 9182 ertanca+  20   0 2621472 292300  28744 S   0.3 17.0   1:21.70 java
    1 root      20   0  159992   9244   6648 S   0.0  0.5   0:04.23 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    3 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 rcu_par_gp
    6 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/0:0H-ev
    8 root       0 -20       0      0      0 I   0.0  0.0   0:00.57 kworker/0:1H-ev
    9 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 mm_percpu_wq
   10 root      20   0       0      0      0 S   0.0  0.0   0:00.47 ksoftirqd/0
   11 root      20   0       0      0      0 I   0.0  0.0   0:00.61 rcu_sched
```