SWE 573

Software Development Practice

Software Engineering Department, Boğaziçi University

Project Final Report

Project Name: ***Story App***

HONOR CODE:

Related to the submission of all the project deliverables for the Swe573 2022 Fall semester project reported in this report, I, Salih Yavuz declare that:

- I am a student in the Software Engineering MS program at Bogazici University and am registered for Swe573 course during the Spring 2023 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.

Submitted by: Salih Yavuz
No: 2022719150
salih.yavuz@boun.edu.tr

[illegible]

30.05.2023

Table of Contents

1. Overview

This app is intended to be a platform where people can share stories from any point of their life or stories, they have heard from people around them. If used and populated correctly it can be a valuable gateway to the points of time and places that would otherwise be forgotten.

In this platform people can share their stories while also specifying the date and exact geolocation it passes in. The time and place specification details are what makes this project unique. Since it is unimaginable to have a novel without a time and place, we should also not have stories devoid of a sense of time and place.

This platform is not intended to be just another social media gig, it is intended to be a place where people of all ages, places and life experiences can come together and share their wonderfully unique stories.

Once a user logs in to the app, they can immediately start sharing their stories. They can also see and interact with other people's stories in the forms of liking and commenting.

I have two separate repositories for this project. One is for the UI and the other is for the Backend.

Backend Repository Link:	https://github.com/yavuzsa/swe-573-fall-23
UI Repository Link:	https://github.com/yavuzsa/storyapp-ui

The Backend Repository is the main repository that contains the Wiki and other documents.

2. Requirements and Their Status

I will be explaining the status of each requirement that I have planned.
I have not changed the requirements throughout the semester since my idea is to see how far I have strayed from my original idea and to see whether my original requirements hold up.

Red means a critical point that is missing.
Orange means either a minor misfunction or a drop in requirement by design choice.
Yellow means requirement fully satisfied.

1. Onboarding to the Site

1.1. Users shall be required to specify an email when creating an account.

Not implemented. UI and Backend and Database need minor modifications to accommodate.

1.2. Users shall be required to specify a username when creating an account.

Fully Implemented.

1.3. Users shall be required to specify a password when creating an account.

Fully Implemented.

1.4. Users shall be able to specify their name and surname when creating an account.

Not implemented. Decided that it was not meaningful to request names and surnames from people who are creating an account. Can be implemented with very minor effort if required in the future.

1.5. Users shall be able to set a profile picture when creating an account.

Not implemented. UI and Backend and Database need major modifications to accommodate.

1.6. Users shall be able to log-in to the site with their username and password.

Fully Implemented.

1.7. Users shall be able to log out from the site.

Fully Implemented.

2. Creating a Post

2.1. Users shall be required to sign in to create a post.

Fully Implemented.

2.2. Users shall be able to share posts in a text format.

Fully Implemented.

2.3. Users shall be able to add photos to their posts.

Not implemented. UI and Backend and Database need major modifications to accommodate.

2.4. Users shall be able to tag their posts with certain keywords.

Not implemented. UI and Backend and Database need minor modifications to accommodate.

2.5. Users shall have predefined tags to choose from.

Not implemented. UI and Backend and Database need minor modifications to accommodate.

2.6. Users shall be able to add a title to their posts.

Fully Implemented.

2.7. Users shall be able to specify the geolocation of their posts.

Partially Implemented. Users cannot choose their desired location from a map; they must type it and choose from the prompted autocomplete list. I store the exact coordinates of their input and display it in their story as a number. Currently can not display it inside a map instance.

2.8. Users shall be able to select the date of the story.

Partially Implemented. Users can choose a date in the form dd/mm/yyyy from a calendar. More ways to provide a can be implemented with minor modifications to UI and Backend and Database.

2.9. Users shall be able to select a date interval.

Not implemented. Both UI and Backend and Database need major modifications to accommodate.

2.10. Users shall be required specify at least a year to create a post.

Not implemented. Decided that it was not meaningful to strictly require users to specify a date to create a story.

2.11. Users shall be able to specify the season (winter, summer, etc...) of their post.

Not implemented. UI and Backend and Database need minor modifications to accommodate.

3. Searching for posts

3.1. Users shall be able to search stories by tag name.

Not implemented. Due to being dependent on requirement **2.4.**

3.2. Users shall be able to search for posts by username.

Partially Implemented. Backend has necessary methods ready, meaning that a request can be sent to the server that performs this operation and get an appropriate response. But the UI does not have an interactable component to send this request through.

3.3. Users shall be able to search for posts by date.

Not implemented. UI and Backend and Database need minor modifications to accommodate.

3.4. Users shall be able to search for posts in a range of geolocation.

Not implemented. Both UI and Backend and Database need major modifications to accommodate.

4. Interacting with Posts

4.1. Users shall be able to comment on posts.

Fully Implemented.

4.2. Users shall be able to like posts.

Fully Implemented.

4.4. Users shall be able to edit their posts after posting.

Partially Implemented. Backend has necessary methods ready, meaning that a request can be sent to the server that performs this operation and get an appropriate response. But the UI does not have an interactable component to send this request through.

4.5. Users shall be able to delete their posts after posting.

Partially Implemented. Backend has necessary methods ready, meaning that a request can be sent to the server that performs this operation and get an appropriate response. But the UI does not have an interactable component to send this request through.

5. User Profile and Interacting with Other Users

5.1. Users shall have a profile page.

Not implemented. UI has a "Profile" button that directs to the profile page of the current user but inside that page is empty. UI needs major work and Backend needs minor work to populate this page.

5.2. Users shall be able to change their profile picture.

Not implemented. Both UI and Backend and Database need major modifications to accommodate.

5.3. Users shall be able to see all the previous posts they shared in their profile.

Not implemented. Backend has necessary methods ready, meaning that a request can be sent to the server that performs this operation and get an appropriate response. But the UI does not have an interactable component to send and view this request through.

5.4. Users shall be able to see all the posts they have liked in their profile.

Not implemented. Backend has necessary methods ready, meaning that a request can be sent to the server that performs this operation and get an appropriate response. But the UI does not have an interactable component to send and view this request through.

5.5. Users shall be able to visit other user's profile.

Not implemented. Backend has necessary methods ready, meaning that a request can be sent to the server that performs this operation and get an appropriate response. But the UI does not have an interactable component to send and view this request through.

5.5. Users shall be able to follow other users.

Not implemented. Both UI and Backend and Database need major modifications to accommodate.

5.6. Users shall see the recent posts from the users they follow in their home page.

Not implemented. Both UI and Backend and Database need major modifications to accommodate.

As can be understood from the state of the requirements, the backend services of this project generally require very little extra work to accommodate all the requirements. The UI requires most of the work to complete all the requirements. I believe all the requirements that have not been implemented or partially implemented can be completed in very little time. It is in my best belief that the core structure of this application is very solid. With minor efforts and polishes in the very near future this application will meet all the requirements.

3. Design

Below are the request paths of the app. I am aware that this does not paint the whole picture of the workings but writing these down helped me during my development stage. There are API design and documentation tools such as Swagger to do this task but couldn’t get to using it.

```
POST > …/login
POST > …/register

GET, POST > .../users
GET, PUT, DELETE > .../users/{userId}

GET, POST > .../posts
GET, PUT, DELETE > .../posts/{postId}

GET, POST > .../comments
GET, PUT, DELETE > .../comments{commentId}

GET, POST > .../likes
GET, DELETE > ...likes/{likeId}
```

In *fig.1.* below you can find the Entity Relationship Diagram of the application.

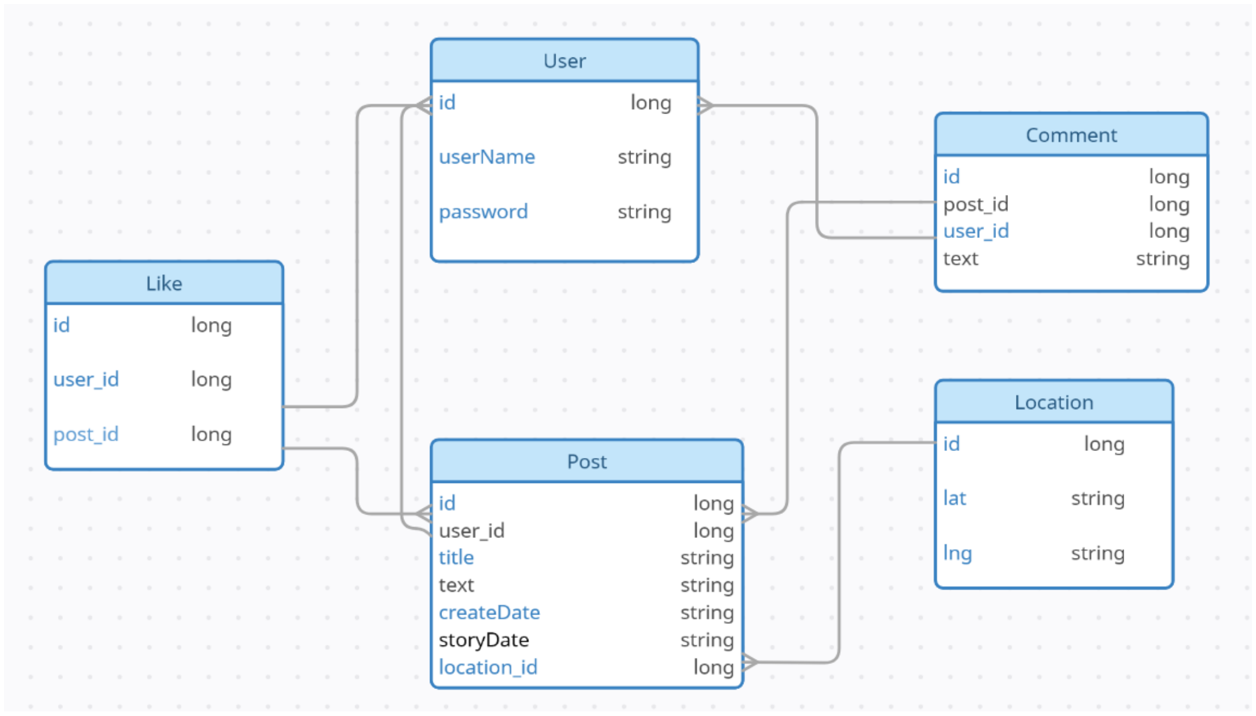
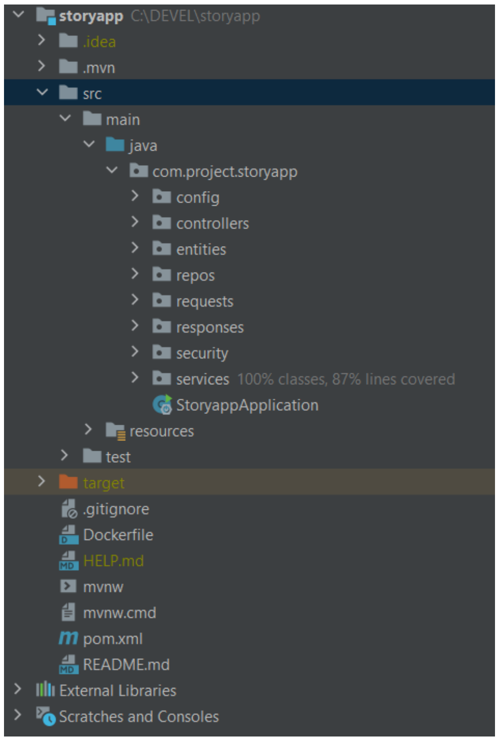
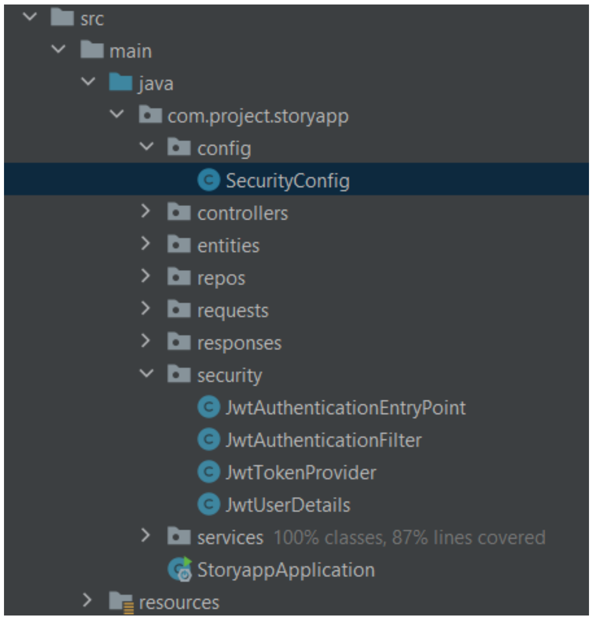


fig. 1. ER diagram of the application.



```
storyapp  C:\DEVEL\storyapp
  .idea
  .mvn
  src
    main
      java
        com.project.storyapp
          config
          controllers
          entities
          repos
          requests
          responses
          security
          services  100% classes, 87% lines covered
          StoryappApplication
      resources
    test
  target
  .gitignore
  Dockerfile
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
  README.md
External Libraries
Scratches and Consoles
```

fig.2. Folder structure of the application.



```
src
  main
    java
      com.project.storyapp
        config
          SecurityConfig
        controllers
        entities
        repos
        requests
        responses
        security
          JwtAuthenticationEntryPoint
          JwtAuthenticationFilter
          JwtTokenProvider
          JwtUserDetails
        services  100% classes, 87% lines covered
        StoryappApplication
    resources
```

fig.3. Files related to security and tokenization.

The backend of the project has been done with Java using Spring Boot and followed a typical Java application folder structure. In *fig.2.* we can see the folder structure associated with it. The main entry point for this application is StoryappApplication.java class which can be seen in *fig.2.*

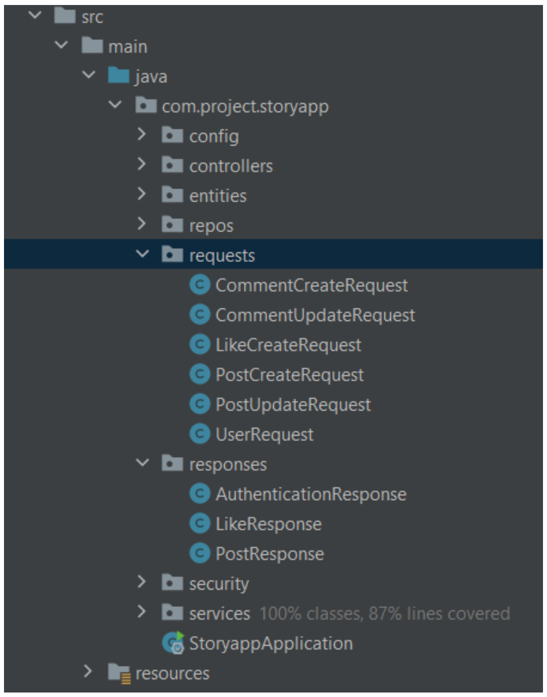
Our main interests will be under the /src folder. As we can see in the file hierarchy, I have separated classes according to the purpose they fulfill.

Under the /config folder we have the main configuration class for spring security. I have used JWT for tokenization. The classes related to assigning a bearer token to users are located under /security folder.

When a user logs in or registers, they are assigned a token which is verified when they try to send requests to the server. This tokenization also determines the UI users are going to be seeing. If a user is not logged in, they won't have a token assigned to so, they won't be able to see functionalities that are only visible to logged in users. These functionalities are: Posting a story, making a comment, liking a story, and having access to a profile page.

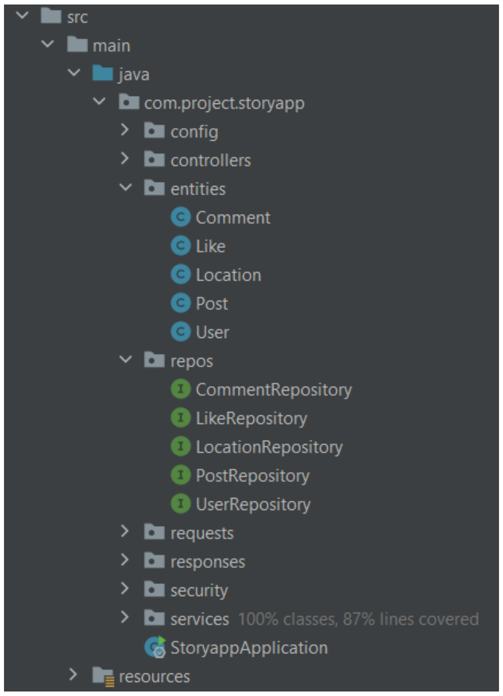
In *fig.4.* we see the contents of the folders containing request and response classes. These classes are used by controllers to receive and respond to the requests coming to the application.

The request classes can be thought as "what the server expects from a request" and the response classes can be thought as "what the user expects from the server".



```
src
  main
    java
      com.project.storyapp
        config
        controllers
        entities
        repos
        requests
          CommentCreateRequest
          CommentUpdateRequest
          LikeCreateRequest
          PostCreateRequest
          PostUpdateRequest
          UserRequest
        responses
          AuthenticationResponse
          LikeResponse
          PostResponse
        security
        services 100% classes, 87% lines covered
        StoryappApplication
    resources
```

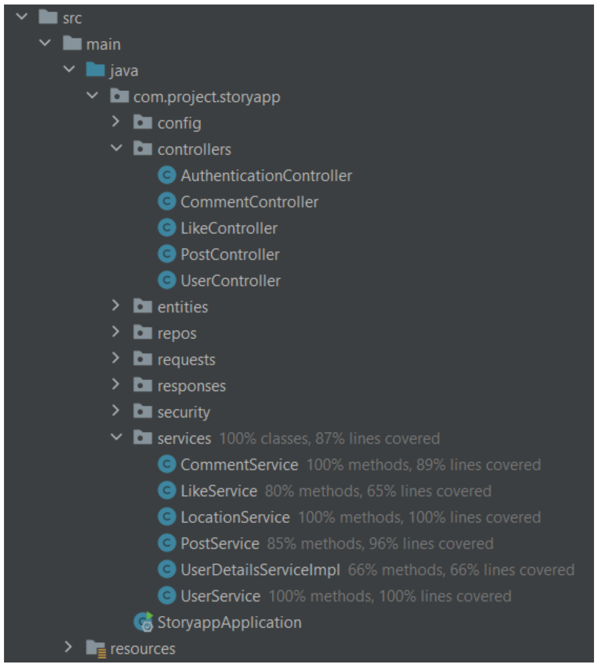
fig.4. Request and Response classes.



```
src
  main
    java
      com.project.storyapp
        config
        controllers
        entities
          Comment
          Like
          Location
          Post
          User
        repos
          CommentRepository
          LikeRepository
          LocationRepository
          PostRepository
          UserRepository
        requests
        responses
        security
        services 100% classes, 87% lines covered
        StoryappApplication
    resources
```

fig.5. Entities and Repositories

As can be seen in *fig.5*, we also have entity classes and repository interfaces. Repositories allow us to communicate with the database. I have extended JPA (Java Persistence API) repository inside these repositories for my database operations. Entities are essentially the objects that we hold in our database. Other than location, all entities represent a table in the relational database.



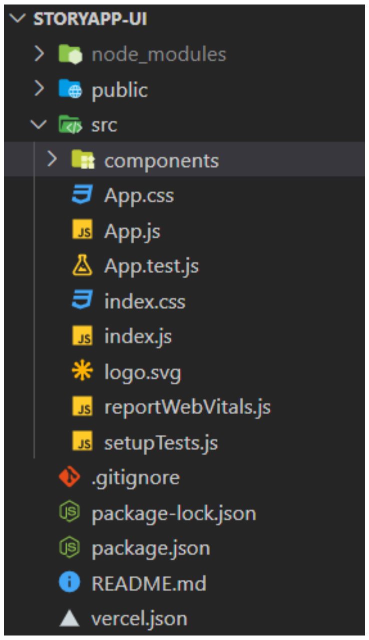
```
src
  main
    java
      com.project.storyapp
        config
        controllers
          AuthenticationController
          CommentController
          LikeController
          PostController
          UserController
        entities
        repos
        requests
        responses
        security
        services 100% classes, 87% lines covered
          CommentService 100% methods, 89% lines covered
          LikeService 80% methods, 65% lines covered
          LocationService 100% methods, 100% lines covered
          PostService 85% methods, 96% lines covered
          UserDetailsServiceImpl 66% methods, 66% lines covered
          UserService 100% methods, 100% lines covered
        StoryappApplication
    resources
```

fig.6. Service and Controller classes.

In *fig.6. we* see Service classes that handle the business logic of our application and Controller classes that handle the requests coming to our application. The controllers contain methods to handle upcoming requests with an appropriate request body. Necessary services are autowired inside the controllers and method calls are made to handle coming requests.

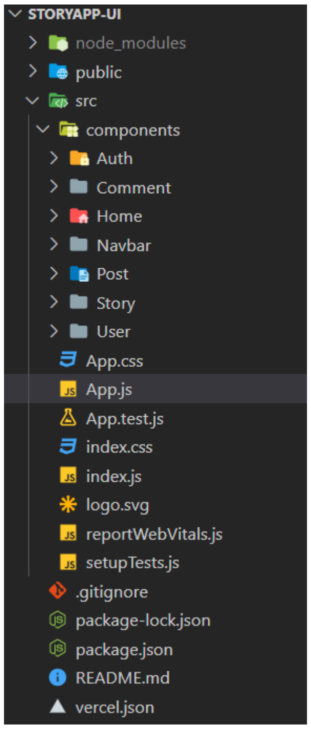
We have an 'application.properties' file under the resources folder. This file contains several variables for database settings, token generation seed and token expire time.

This was the backend structure of the application. Now let's move to UI.



```
STORYAPP-UI
  > node_modules
  > public
  v src
    > components
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
      setupTests.js
    .gitignore
    package-lock.json
    package.json
    README.md
    vercel.json
```

Fig. 7. Folder structure of the UI



```
STORYAPP-UI
  > node_modules
  > public
  v src
    v components
      > Auth
      > Comment
      > Home
      > Navbar
      > Post
      > Story
      > User
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
      setupTests.js
    .gitignore
    package-lock.json
    package.json
    README.md
    vercel.json
```

Fig. 8. Components of the UI

I used React for UI development. The followed react folder structure practices. The main entry point for the UI is App.js file. Under the components folder I have defined my components which will are rendered in the UI when needed for.

For UI, I have used the default react '.gitignore' file that is generated by the 'create-react-app' command. For the backend I have used the default '.gitignore' file that is generated by spring initializr. Spring initializr is a website where we can generate a basic template for a spring boot application. While generation this template we select the Java and Spring Boot version we will be using and which automation tool we will be using for our dependencies. The URL for the website is https://start.spring.io/

4. Status of Deployment

Docker: **Dockerized**

Docker has been used to containerize the Java Spring backend of the application. Dockerfile can be found directly inside the project destination. The dockerfile takes the latest '.jar' file which is generated after a successful build of the application under the /target folder and creates a container.

Backend and Database Deployment: **Deployed**

Backend is deployed to a free service called ***Clever Cloud***. My configuration in clever cloud takes the latest commint to the main branch and builds it. Then the dockerfile is invoked which takes latest jar and containerizeses it. The container is then exposed to a public URL.
When a new commit is made to the main branch of the repository, a new pipeline automatically starts to build, containerize and deploy the application in its latest form.

The URI backend is deployed at: http://app_dd90c65e-7942-424f-b5b9-0ac19252d021

Note this this URL does not contain the UI. The UI is deployed separately. But this URL can be used to send requests directly to the backend by using tools such as Postman or Soap UI.

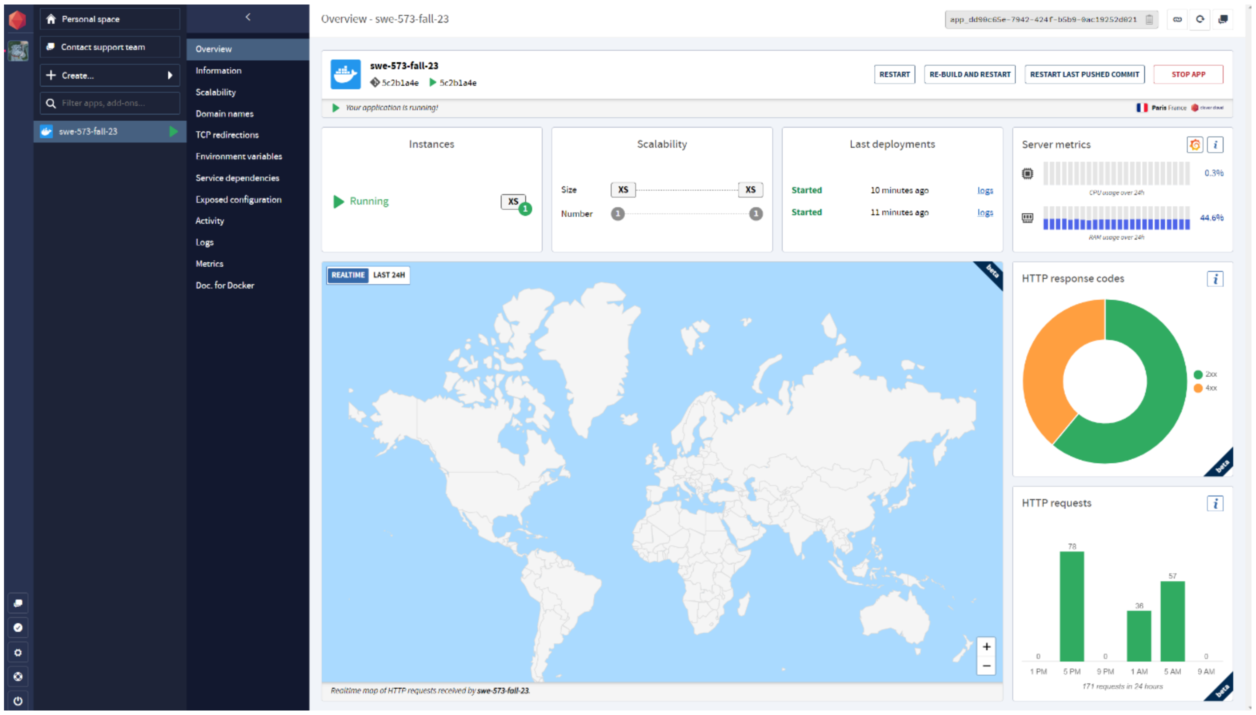
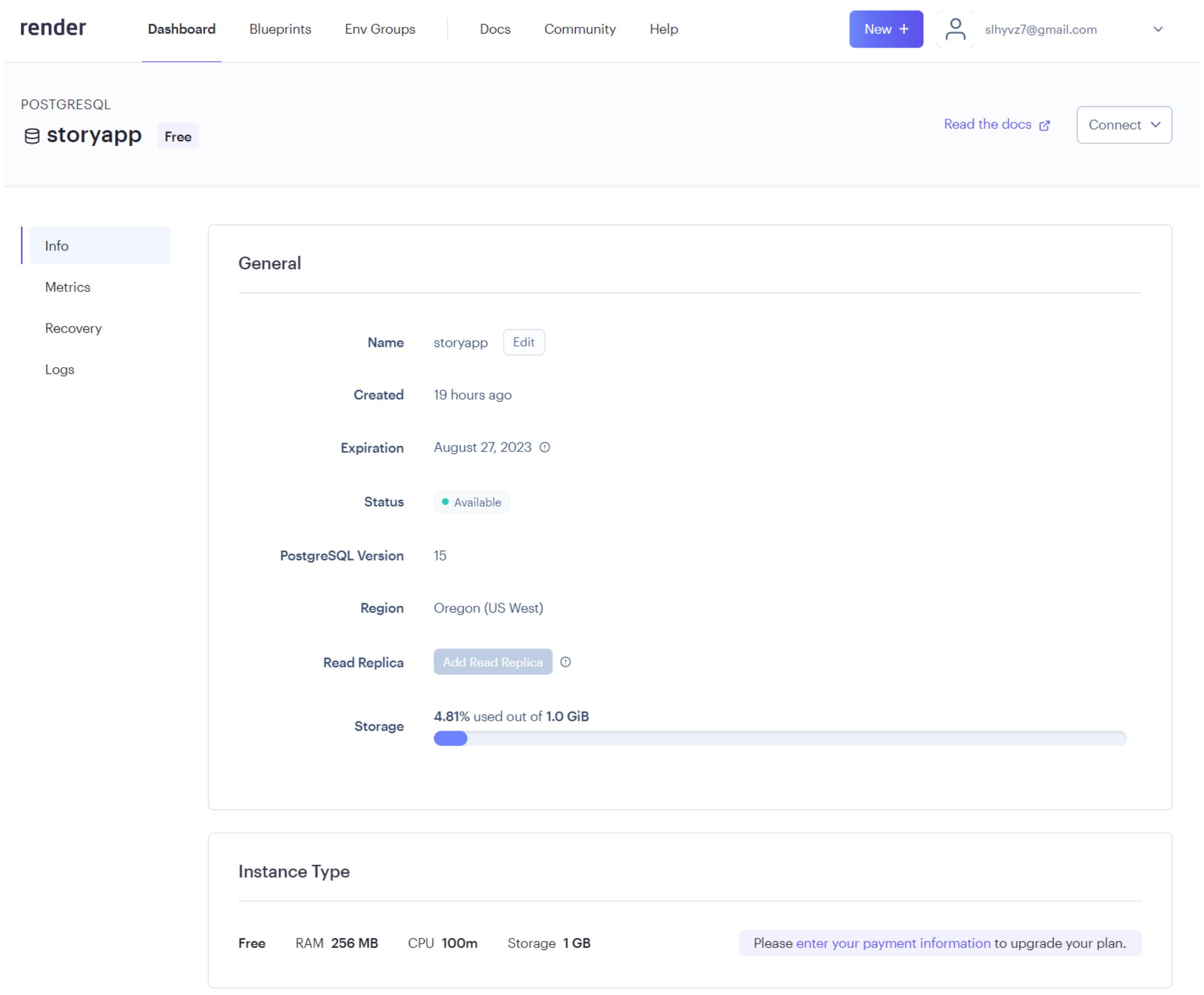


Fig 9. Clever Cloud overview screen of the deployed backend.

The database is deployed to ***render.com*** which hosts our PostgreSQL database for free and provides a public URL. This URL is not meaningfull on its own. It is set as the datasource URL inside our backend while configure database properties.

Datasource URL: dpg-chqc9kik728ivvsj8j70-a.oregon-postgres.render.com:5432/story_app



POSTGRESQL

storyapp Free

General

Name	storyapp Edit
Created	19 hours ago
Expiration	August 27, 2023
Status	Available
PostgreSQL Version	15
Region	Oregon (US West)
Read Replica	Add Read Replica
Storage	4.81% used out of 1.0 GiB

Instance Type

Free RAM 256 MB CPU 100m Storage 1 GB

Please enter your payment information to upgrade your plan.

Fig.10. Render.com deployed database info screen.

UI deployment: **Deployed** but the is a problem while signing into the application from the deployed UI address which I could not resolve in time. So, we cannot use the deployed front-end URL for testing with a logged in user.

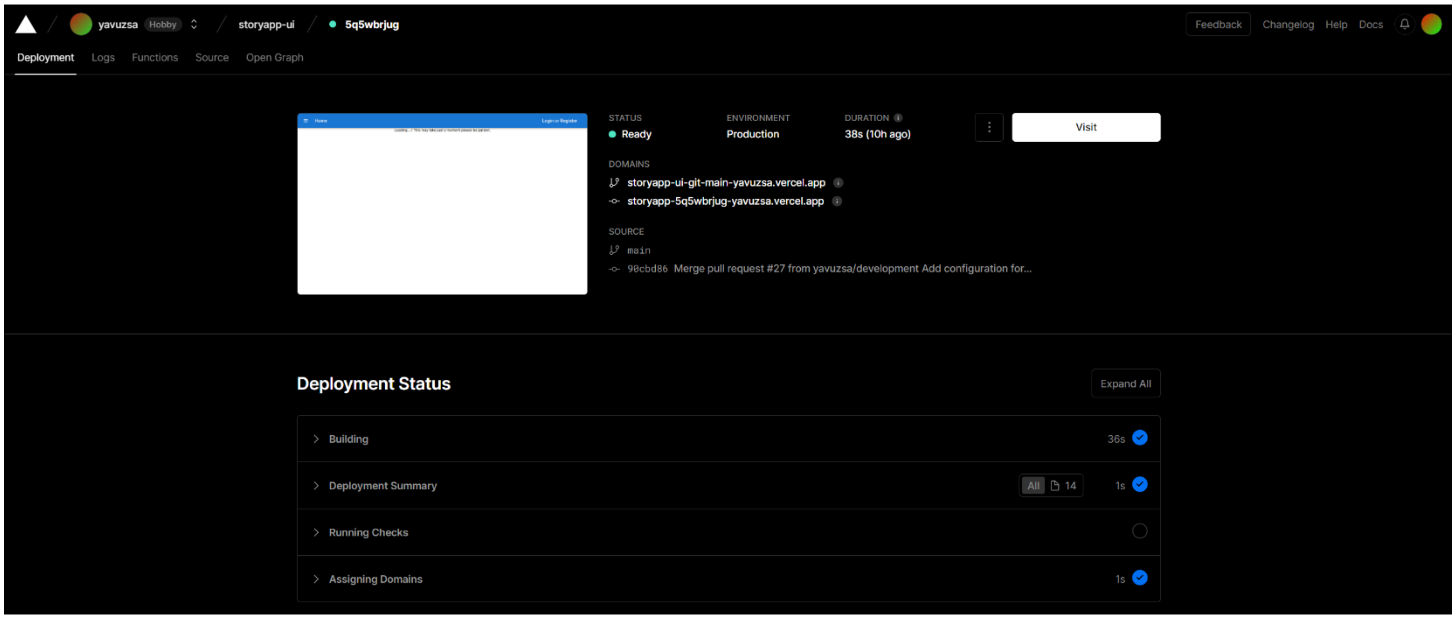
Meaning that we cannot create stories, make comments or like stories. These operations can be tested by running UI in a local machine. The UI is already connected to the deployed backend so there won't be need to run the backend. But the UI needs to be run on a local machine to log in or register to the system.

Viewing the UI without logging in is possible on the public URL.

In order to run the UI on the local machine and communicate with the deployed backend, you must checkout to the 'local-ui' branch. I will be providing more information on upcoming part about running the application.

The UI is deployed to ***Vercel,*** which is a free platform where we can conveniently host the UI our application. When a new commit is made to the repository of the UI, an new build automatically starts and deploys the application again.

The URL to access the UI: https://storyapp-5q5wbrjug-yavuzsa.vercel.app/



STATUS: Ready
ENVIRONMENT: Production
DURATION: 38s (10h ago)

DOMAINS
storyapp-ui-git-main-yavuzsa.vercel.app
storyapp-5q5wbrjug-yavuzsa.vercel.app

SOURCE
main
98cbd86 Merge pull request #27 from yavuzsa/development Add configuration for...

Deployment Status

Building	36s
Deployment Summary	1s
Running Checks	
Assigning Domains	1s

Fig.11. The deployment overview screen of Vercel.

5. System Manual

Requirements for Backend:

- **Java 17**
- **PostgreSql**
- **Docker**

Instructions to run Backend:

Clone the repository:

```
git clone https://github.com/yavuzsa/swe-573-fall-23.git
```

Build java project:

If you have Java 17 installed on your system. You can build using any regular method of building a Java application that uses maven. I decided to use IntelliJ IDE for building the project. Inside the IDE you need to reload all maven project. This allows maven to download required dependencies automatically. There is no need to install additional dependencies since the dependencies specified inside the pom.xml file will be handled by maven. Then by using the build feature of the IDE you gen generate a .jar file after a successful build.

Build container image and run:

Run the dockerfile by using the command below. The -t flag enables us to specift a name for our container image. The '.' at the tells Docker to look for the Dockerfile in the current directory.

```
docker build -t <your_name> .
```

After building the container image we can start an app container. You can use the below command run the application.

```
docker run -dp 8080:8080 <your_name>
```

The -d flag runs the container in detached mode and the -p flag maps the host's port and container's port together.

Requirements for UI:

- **Node v.18.13 installation**

Instructions to run the UI:

Clone the repository:

```
git clone https://github.com/yavuzsa/storyapp-ui.git
```

As I have stated in the UI deployment part, deployment UI has a problem where it does not allow users to sign into the application. I am aware of the problem and trying to fix it.

Meanwhile, running the UI only on the local machine does not have this problem. It can successfully communicate with the deployed Backend and allow users to log in and perform all sorts of actions.

So, in order to be able to test the system with a logged in user you must start the UI on your local machine.

To do this, you MUST checkout to the ‘local-ui’ branch that I have created for this purpose. This is necessary because the main branch is configured for the UI deployment platform. I have reverted those changes in the ‘local-ui’ branch.

To checkout into ‘local-ui’ branch use the following command.

```
git checkout local-ui
```

Install the dependencies and run:

Running `npm install` command should install all the required dependencies.

Then you can run `npm start` command to get the application up and running on

The UI should be live at `http://localhost:3000`

If you encounter any dependency errors, try running all of the commands bellow then try running `npm start` again.

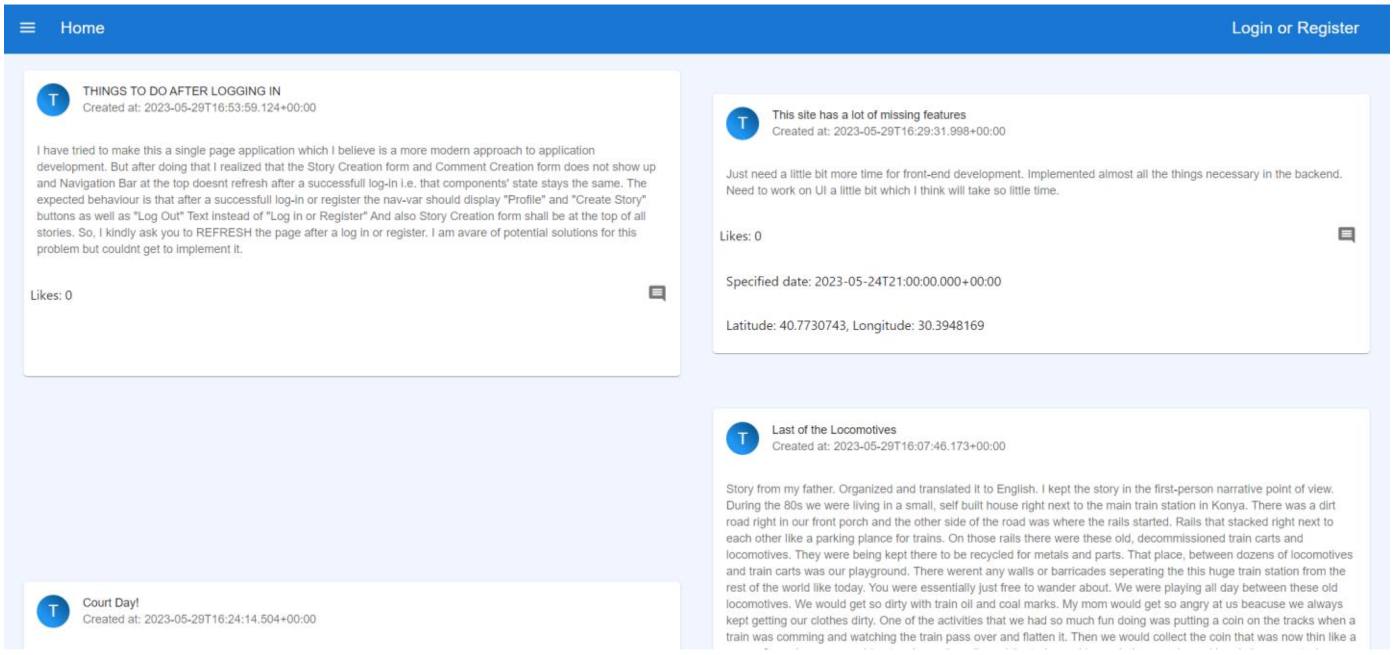
```
npm install react-datepicker –save
npm install react-router-dom
npm install @mui/styles
npm install @mui/icons-material
npm install @mui/material @emotion/react @emotion/styled --force
npm install @material-ui/core
```

6. User Manual

Users can either register or log-in to the site to access all its features.

A pre-registered test user is, **Username: testuser1   Password: testuser1**

Without a log in users can see the stories posted by other users but cannot create stories of their own or make comments and likes to the stories.



THINGS TO DO AFTER LOGGING IN
Created at: 2023-05-29T16:53:59.124+00:00

I have tried to make this a single page application which I believe is a more modern approach to application development. But after doing that I realized that the Story Creation form and Comment Creation form does not show up and Navigation Bar at the top doesnt refresh after a successfull log-in i.e. that components' state stays the same. The expected behaviour is that after a successfull log-in or register the nav-var should display "Profile" and "Create Story" buttons as well as "Log Out" Text instead of "Log in or Register" And also Story Creation form shall be at the top of all stories. So, I kindly ask you to REFRESH the page after a log in or register. I am avare of potential solutions for this problem but couldnt get to implement it.

Likes: 0

This site has a lot of missing features
Created at: 2023-05-29T16:29:31.998+00:00

Just need a little bit more time for front-end development. Implemented almost all the things necessary in the backend. Need to work on UI a little bit which I think will take so little time.

Likes: 0

Specified date: 2023-05-24T21:00:00.000+00:00

Latitude: 40.7730743, Longitude: 30.3948169

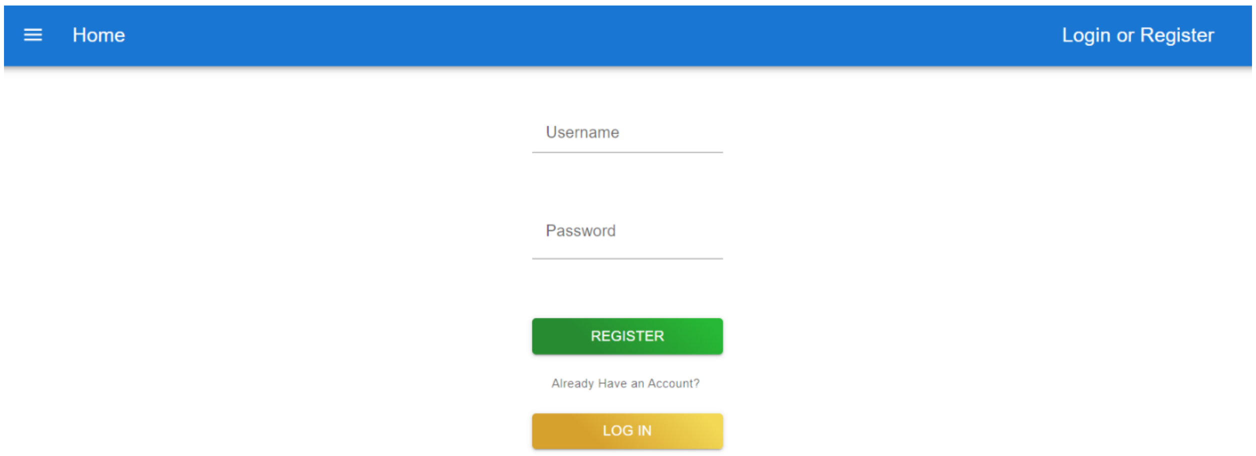
Last of the Locomotives
Created at: 2023-05-29T16:07:46.173+00:00

Story from my father. Organized and translated it to English. I kept the story in the first-person narrative point of view. During the 80s we were living in a small, self built house right next to the main train station in Konya. There was a dirt road right in our front porch and the other side of the road was where the rails started. Rails that stacked right next to each other like a parking plance for trains. On those rails there were these old, decommissioned train carts and locomotives. They were being kept there to be recycled for metals and parts. That place, between dozens of locomotives and train carts was our playground. There werent any walls or barricades seperating the this huge train station from the rest of the world like today. You were essentially just free to wander about. We were playing all day between these old locomotives. We would get so dirty with train oil and coal marks. My mom would get so angry at us beacuse we always kept getting our clothes dirty. One of the activities that we had so much fun doing was putting a coin on the tracks when a train was comming and watching the train pass over and flatten it. Then we would collect the coin that was now thin like a

Court Day!
Created at: 2023-05-29T16:24:14.504+00:00

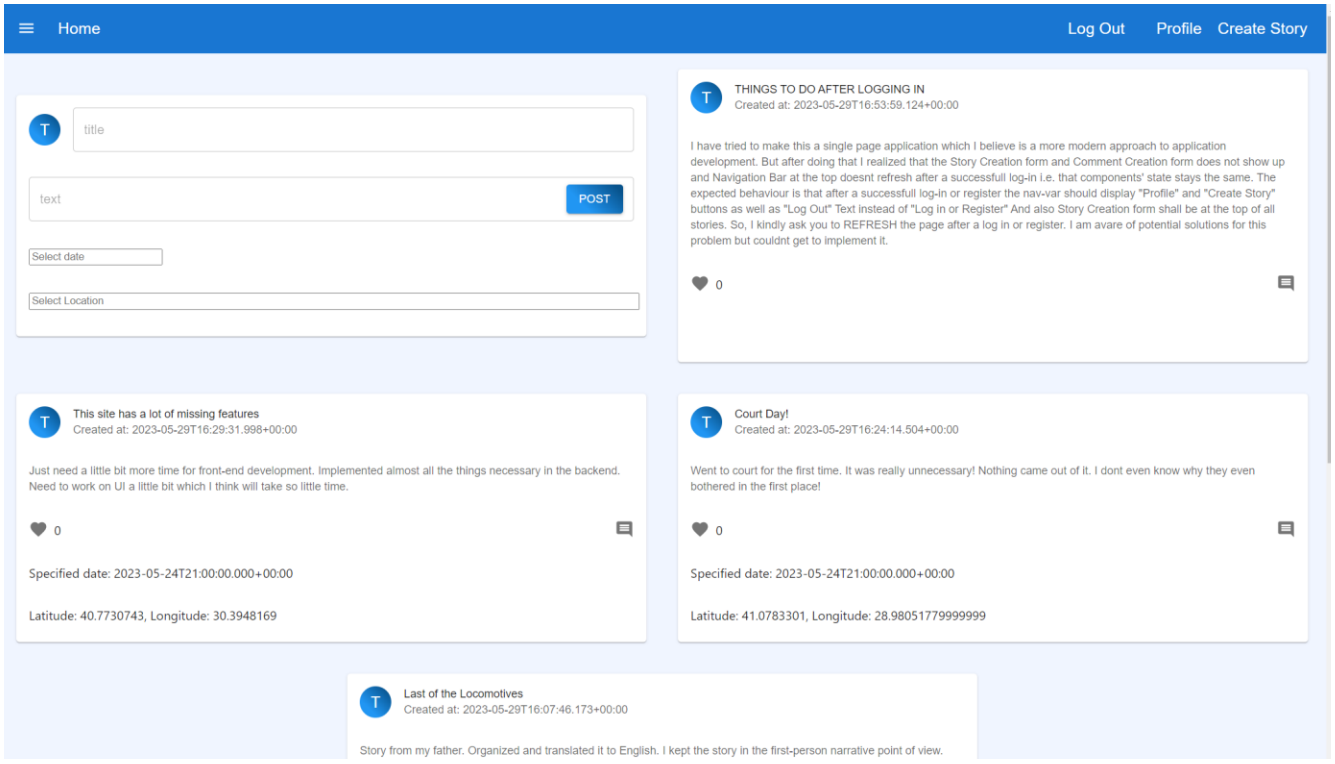
Fig.12. How the home page looks without a log-in

As we can see we can only see the posts created by other users. We get a ‘Login or Register’ button at the navigation bar.



The registration step does not require any additional steps. Just enter a username and password and click register instead of login to create an account. You will be redirected to the home screen as a registered user.

You can always prefer to use **Username: testuser1 and Password: testuser1**



T THINGS TO DO AFTER LOGGING IN
Created at: 2023-05-29T16:53:59.124+00:00

I have tried to make this a single page application which I believe is a more modern approach to application development. But after doing that I realized that the Story Creation form and Comment Creation form does not show up and Navigation Bar at the top doesnt refresh after a successfull log-in i.e. that components' state stays the same. The expected behaviour is that after a successfull log-in or register the nav-var should display "Profile" and "Create Story" buttons as well as "Log Out" Text instead of "Log in or Register" And also Story Creation form shall be at the top of all stories. So, I kindly ask you to REFRESH the page after a log in or register. I am avare of potential solutions for this problem but couldnt get to implement it.

0

T This site has a lot of missing features
Created at: 2023-05-29T16:29:31.998+00:00

Just need a little bit more time for front-end development. Implemented almost all the things necessary in the backend. Need to work on UI a little bit which I think will take so little time.

0

Specified date: 2023-05-24T21:00:00.000+00:00

Latitude: 40.7730743, Longitude: 30.3948169

T Court Day!
Created at: 2023-05-29T16:24:14.504+00:00

Went to court for the first time. It was really unnecessary! Nothing came out of it. I dont even know why they even bothered in the first place!

0

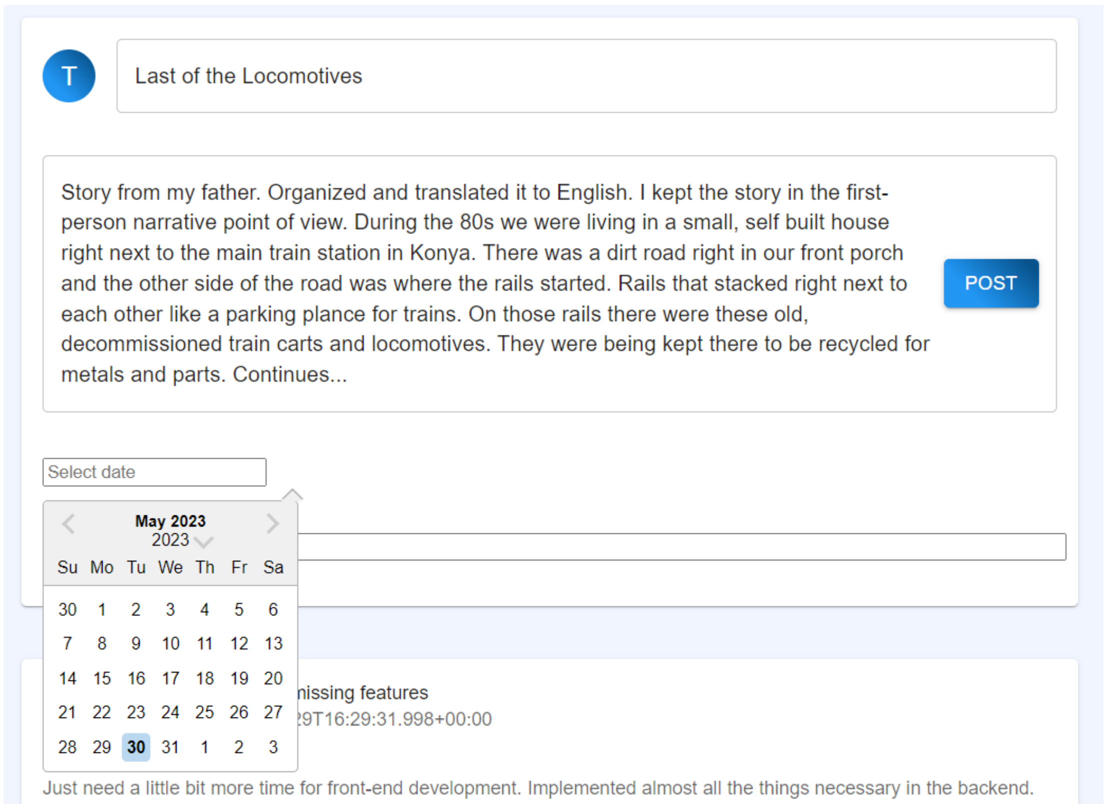
Specified date: 2023-05-24T21:00:00.000+00:00

Latitude: 41.0783301, Longitude: 28.98051779999999

T Last of the Locomotives
Created at: 2023-05-29T16:07:46.173+00:00

Story from my father. Organized and translated it to English. I kept the story in the first-person narrative point of view.

Fig.14. Home screen after logging in or registering.



T

Last of the Locomotives

Story from my father. Organized and translated it to English. I kept the story in the first-person narrative point of view. During the 80s we were living in a small, self built house right next to the main train station in Konya. There was a dirt road right in our front porch and the other side of the road was where the rails started. Rails that stacked right next to each other like a parking plance for trains. On those rails there were these old, decommissioned train carts and locomotives. They were being kept there to be recycled for metals and parts. Continues...

POST

Select date

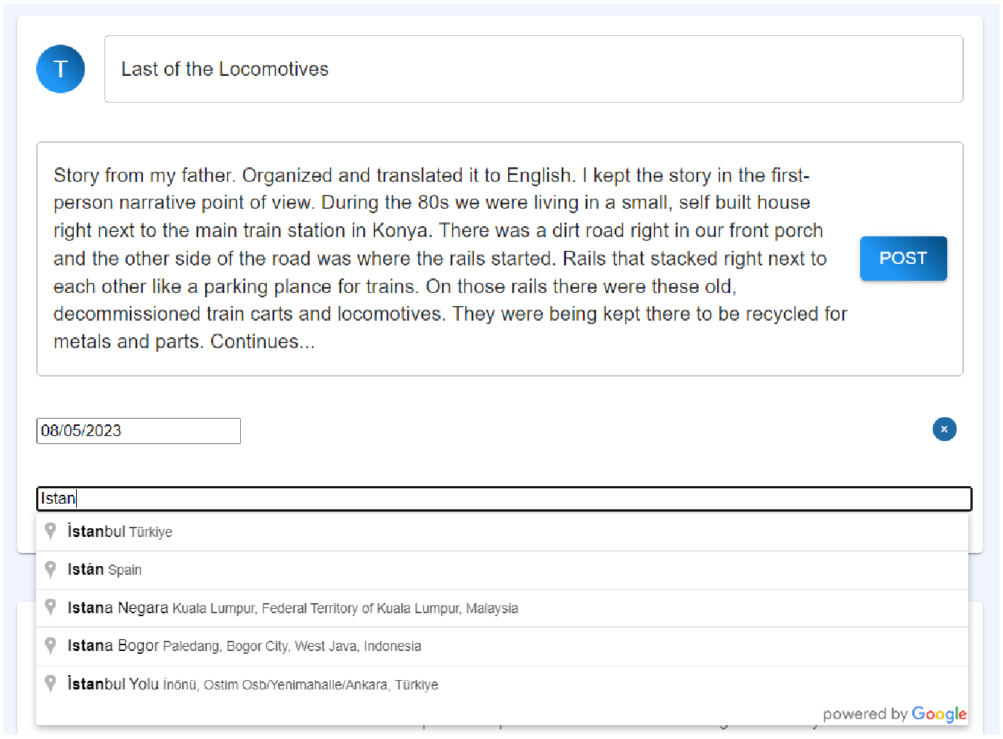
May 2023
2023

Su	Mo	Tu	We	Th	Fr	Sa
30	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	1	2	3

nissing features
29T16:29:31.998+00:00

Just need a little bit more time for front-end development. Implemented almost all the things necessary in the backend.

Fig. 15. The part where we create stories and date picker window open.



T

Last of the Locomotives

Story from my father. Organized and translated it to English. I kept the story in the first-person narrative point of view. During the 80s we were living in a small, self built house right next to the main train station in Konya. There was a dirt road right in our front porch and the other side of the road was where the rails started. Rails that stacked right next to each other like a parking plance for trains. On those rails there were these old, decommissioned train carts and locomotives. They were being kept there to be recycled for metals and parts. Continues...

POST

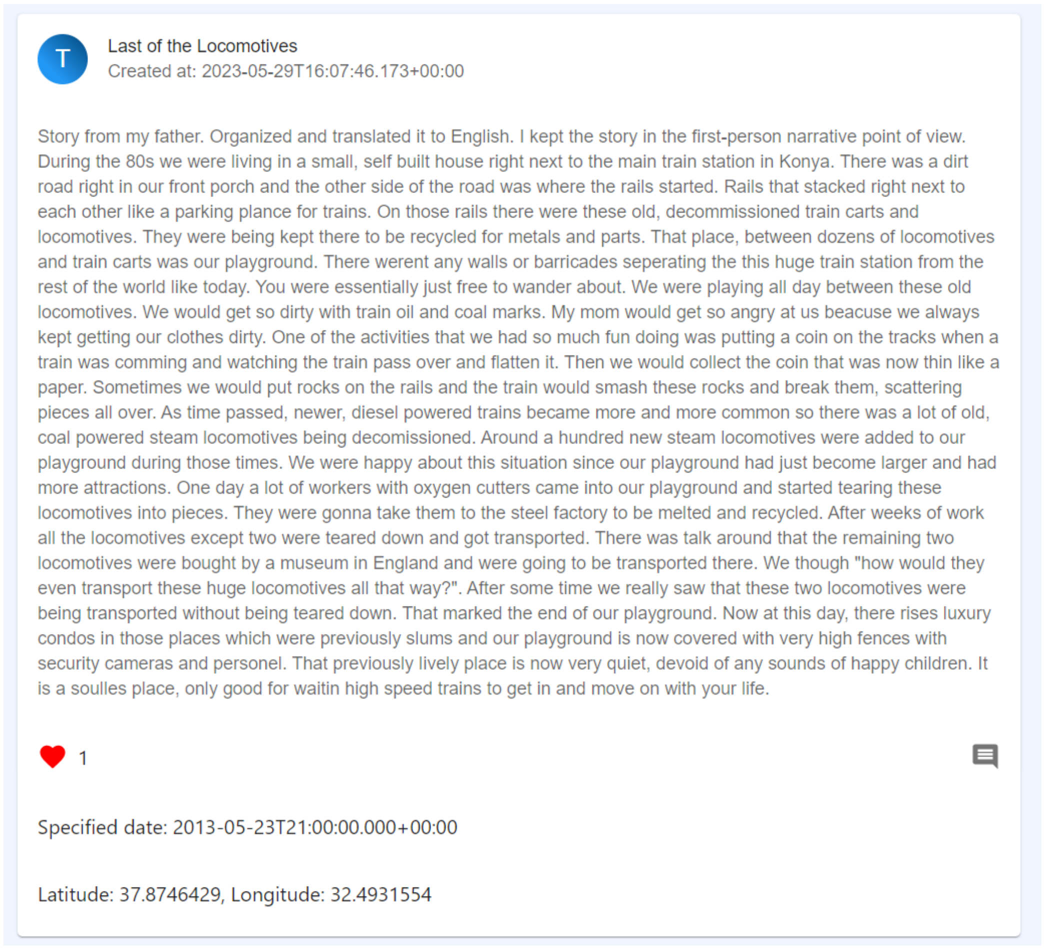
08/05/2023

Istan

- **İstan**bul Türkiye
- **Istán** Spain
- **Istana** Negara Kuala Lumpur, Federal Territory of Kuala Lumpur, Malaysia
- **Istana** Bogor Paledang, Bogor City, West Java, Indonesia
- **İstan**bul Yolu İnönü, Ostim Osb/Yenimahalle/Ankara, Türkiye

powered by Google

Fig. 16. Location selection part


T

Last of the Locomotives
Created at: 2023-05-29T16:07:46.173+00:00

Story from my father. Organized and translated it to English. I kept the story in the first-person narrative point of view. During the 80s we were living in a small, self built house right next to the main train station in Konya. There was a dirt road right in our front porch and the other side of the road was where the rails started. Rails that stacked right next to each other like a parking plance for trains. On those rails there were these old, decommissioned train carts and locomotives. They were being kept there to be recycled for metals and parts. That place, between dozens of locomotives and train carts was our playground. There werent any walls or barricades seperating the this huge train station from the rest of the world like today. You were essentially just free to wander about. We were playing all day between these old locomotives. We would get so dirty with train oil and coal marks. My mom would get so angry at us beacuse we always kept getting our clothes dirty. One of the activities that we had so much fun doing was putting a coin on the tracks when a train was comming and watching the train pass over and flatten it. Then we would collect the coin that was now thin like a paper. Sometimes we would put rocks on the rails and the train would smash these rocks and break them, scattering pieces all over. As time passed, newer, diesel powered trains became more and more common so there was a lot of old, coal powered steam locomotives being decomissioned. Around a hundred new steam locomotives were added to our playground during those times. We were happy about this situation since our playground had just become larger and had more attractions. One day a lot of workers with oxygen cutters came into our playground and started tearing these locomotives into pieces. They were gonna take them to the steel factory to be melted and recycled. After weeks of work all the locomotives except two were teared down and got transported. There was talk around that the remaining two locomotives were bought by a museum in England and were going to be transported there. We though "how would they even transport these huge locomotives all that way?". After some time we really saw that these two locomotives were being transported without being teared down. That marked the end of our playground. Now at this day, there rises luxury condos in those places which were previously slums and our playground is now covered with very high fences with security cameras and personel. That previously lively place is now very quiet, devoid of any sounds of happy children. It is a soulles place, only good for waitin high speed trains to get in and move on with your life.

1

Specified date: 2013-05-23T21:00:00.000+00:00

Latitude: 37.8746429, Longitude: 32.4931554

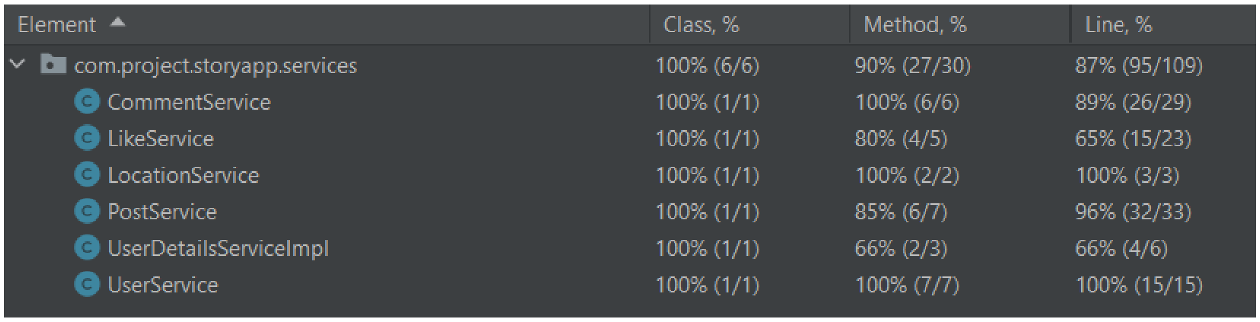
Fig. 17. A posted story with where data and location are specified.

One of the known problems of the UI is that the navigation bar does not refresh change it's conentents after a successful login or register. So it is best to refresh the page manually after logging in or registering.

Another known issue is that after posting a story or a comment to a story, newly created story does not show up without refreshing the page manually. I had fixed this case before but after making additions to the project, the issue still persists. So, I kindly ask to refresh the page after posting a story or comment to see your newly posted story or comment.

7. Test Results

Unit tests have been written for backend services. They can be found under the /tests directory inside the backend repository.



Element	Class, %	Method, %	Line, %
com.project.storyapp.services	100% (6/6)	90% (27/30)	87% (95/109)
CommentService	100% (1/1)	100% (6/6)	89% (26/29)
LikeService	100% (1/1)	80% (4/5)	65% (15/23)
LocationService	100% (1/1)	100% (2/2)	100% (3/3)
PostService	100% (1/1)	85% (6/7)	96% (32/33)
UserDetailsServiceImpl	100% (1/1)	66% (2/3)	66% (4/6)
UserService	100% (1/1)	100% (7/7)	100% (15/15)

Fig.18. Unit test coverage

In *fig.18.* we can see the total coverage of services. The total line coverage is at 87%. LikeService and UserDetailsServiceImpl really drags down the average. I will be looking to increase this coverage.

8. Closing Words

I must say, this semester semester has been the busiest one of my whole education history. And this project claims the title of "most effort I have ever put in a task in my life". Of course, some external life factors during the past semester also contributed to it earning this title but still I can say that I am kind of proud of this project. It has many shortcomings that I am aware and trying to fix but I believe it will be better as I put just a little bit more time to it and polish its rough edges.
This was the first big project that I have done, and it was such a fulfillment. After deploying the website for the first time I sent the URL to my parents and bragged about it. But I still told them not to play around the site too much because I was worried that something could break.

Acknowledgments:

Without the help of the documents on the internet and countless videos and courses that I have watched at 2x speed this project would have been impossible. I send my thanks to all the people who are providing documents and videos to help people learn software development and to all the people who post solutions to annoying errors online.