Alexandria University
Faculty of Engineering
Computer and Systems Engineering Department
Software Engineering

Milestone 1
E-Shop Web Application

Name	ID
Fareeda Mohamed Ali Abouzed	19016154
Rowan Nasser Edrees	19015686
Bassant Yasser Salah Beshir	19017262
Mayar Ayman Mahmoud Ali	19016744
Nada Mohamed Ibrahim	19016782
Toka Ashraf Abo Elwafa Ahmed	19015539

Table of Contents:

Functional and Non-functional:

a) Functional Requirements:

1. **Register:**

- This method is done only when the user isn't registered before the system and makes a new account. He only needs to fulfill the following fields to register in the system: (First name- Last Name - Password- Telephone Number - Email - Date of birth (he must be 15 or over).

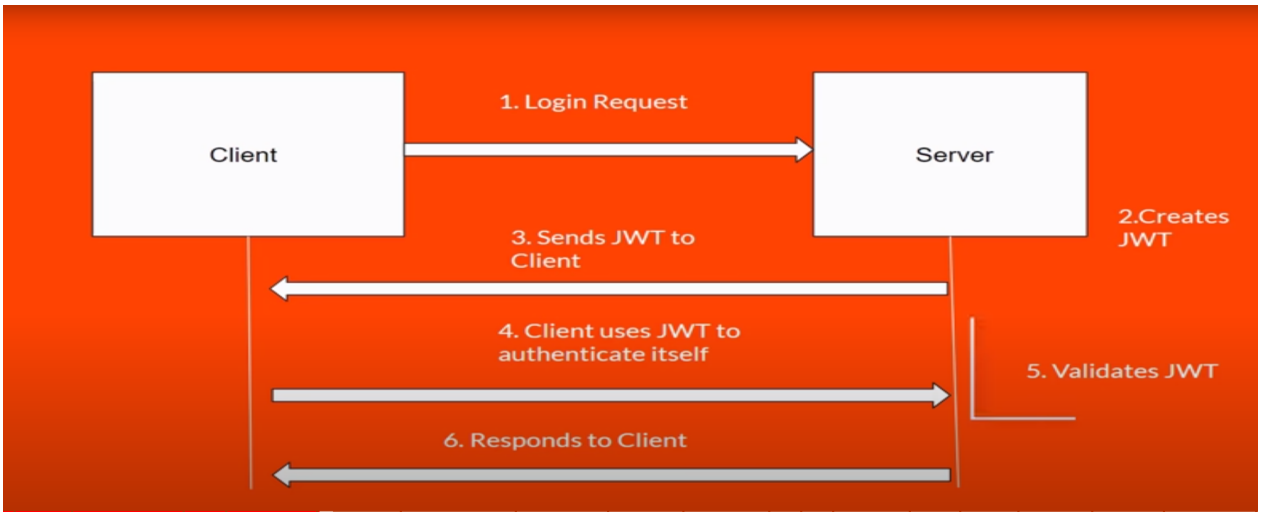
2. **Login:**

- when the user was registered before and wants to log in to his account. He only wants to provide his email and password. the data sent to the backend to be verified (whether it was registered before or not) and give the proper authentication to the user (detailed explanation in Authentication and Authorization part).

b) Non-Functional Requirements:

3. **Authentication and Authorization:**

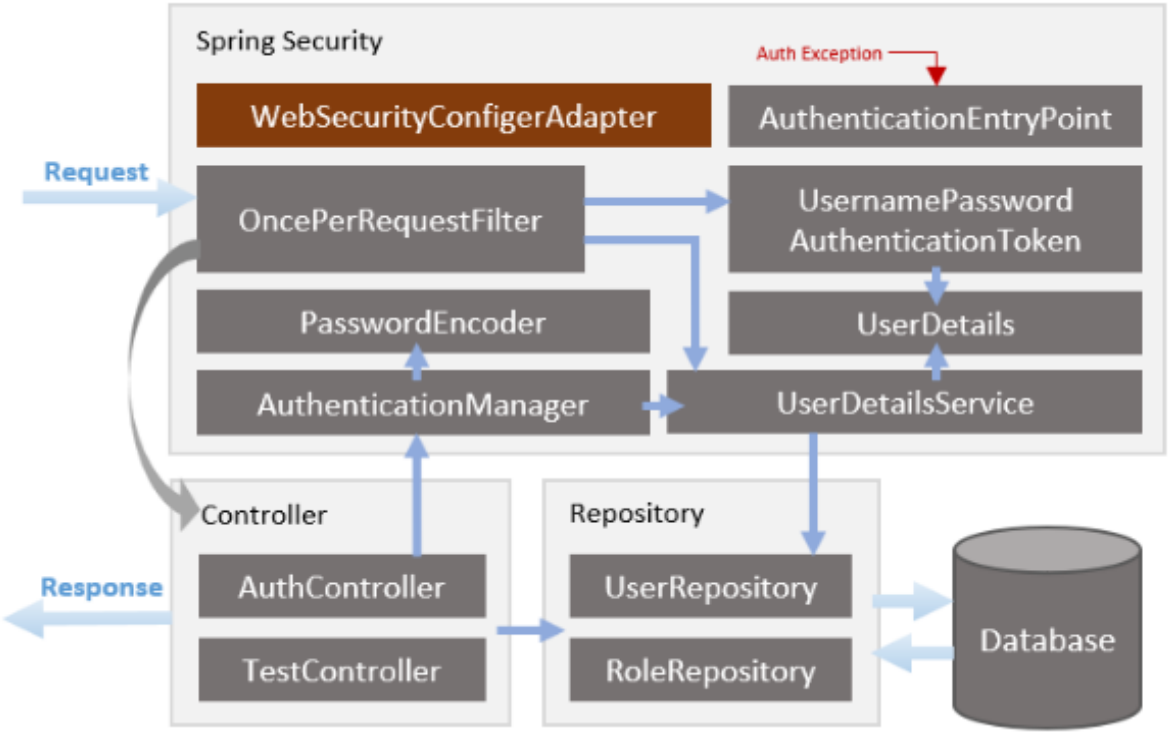
- Authentication is a process that verifies that someone or something is who they say they are. The authentication compares the username and password entered with a record it has on its database. If the information submitted matches, the system assumes you are a valid user and grants you access.
- Authorization which is the function that gives suitable permissions for the authenticated user.

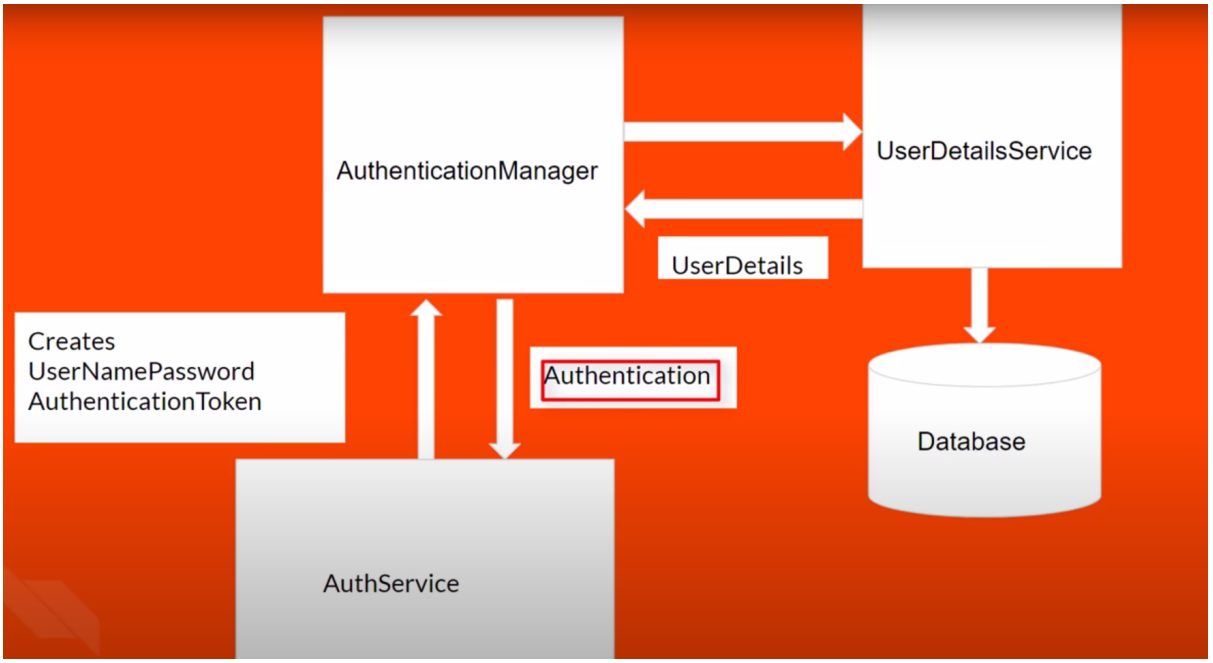


The flow of the Authentication process:

- the user will make a request to the backend with his registered password and Gmail.
- the server will take this data given and validate whether it is in the database or not.
- After finding the data in the database the server generates a Jwt token to send it to the client again which is the user that requests the data (JWT token which is JSON web Token that is used to share data between the client and server in a secure way).
- This generated data will be sent back to the client to be able to get authorized to make changes and to have suitable permission in the web application.
- the client will use this token to authenticate his data and authorize his sign-in session, this will happen by sending the token again and the system will make sure either the token sent by the client is the one generated by the server that validates his data or not.

The previous explanation was an overview of the flow of the authentication process, we are going to explain more about authentication using `Spring Security` in spring boot.


Spring Security
Auth Exception
WebSecurityConfigerAdapter
AuthenticationEntryPoint
Request
OncePerRequestFilter
UsernamePassword
AuthenticationToken
PasswordEncoder
UserDetails
AuthenticationManager
UserDetailsService
Controller
Repository
Response
AuthController
UserRepository
TestController
RoleRepository
Database


AuthenticationManager
UserDetailsService
UserDetails
Creates
UserNamePassword
AuthenticationToken
Authentication
Database
AuthService

How it works in the backend:

```
Authentication authentication = authenticationManager.authenticate(
        new UsernamePasswordAuthenticationToken(loginRequest.getEmail(), loginRequest.getPassword())
);
```

a) **Generates UserNamePassword:**

- First of all, we take from the front the data entered by the user which is the (Email and password)and generate a token to send to AuthenticationManager to send it to UserDetailsService which is a class that extracts the data of the user from the given token and finds whether it exists in the database or not.

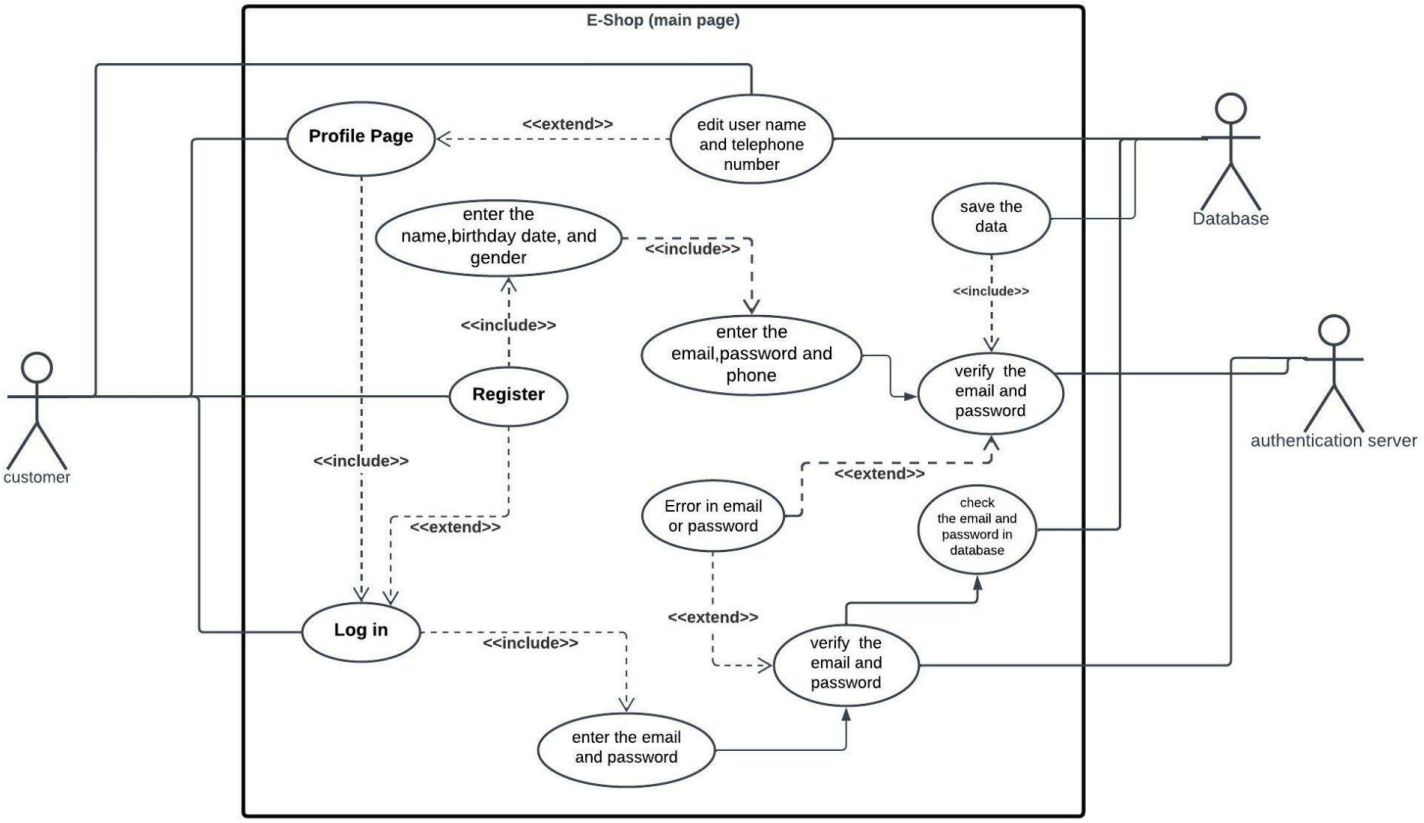
b) **AuthenticationManager:** is the main strategy interface for authentication. If the principal of the input authentication is valid and verified, AuthenticationManager returns an instance with the authenticated flag = true. It throws an AuthenticationException in case of invalidation of the token given. For the last case, it returns null if it can't decide. It delegates the authentication process to a list of AuthenticationProvider instances. We can set up a global or local AuthenticationManager if we create a SecurityFilterChain bean. For a local AuthenticationManager, we could create an AuthenticationManager bean, accessing AuthenticationManagerBuilder through HttpSecurity.

c) **AuthService:** after authentication of the user from the database. A Jwt token will be generated in order to determine the authority and give specific responsibility, expire at (session), etc.

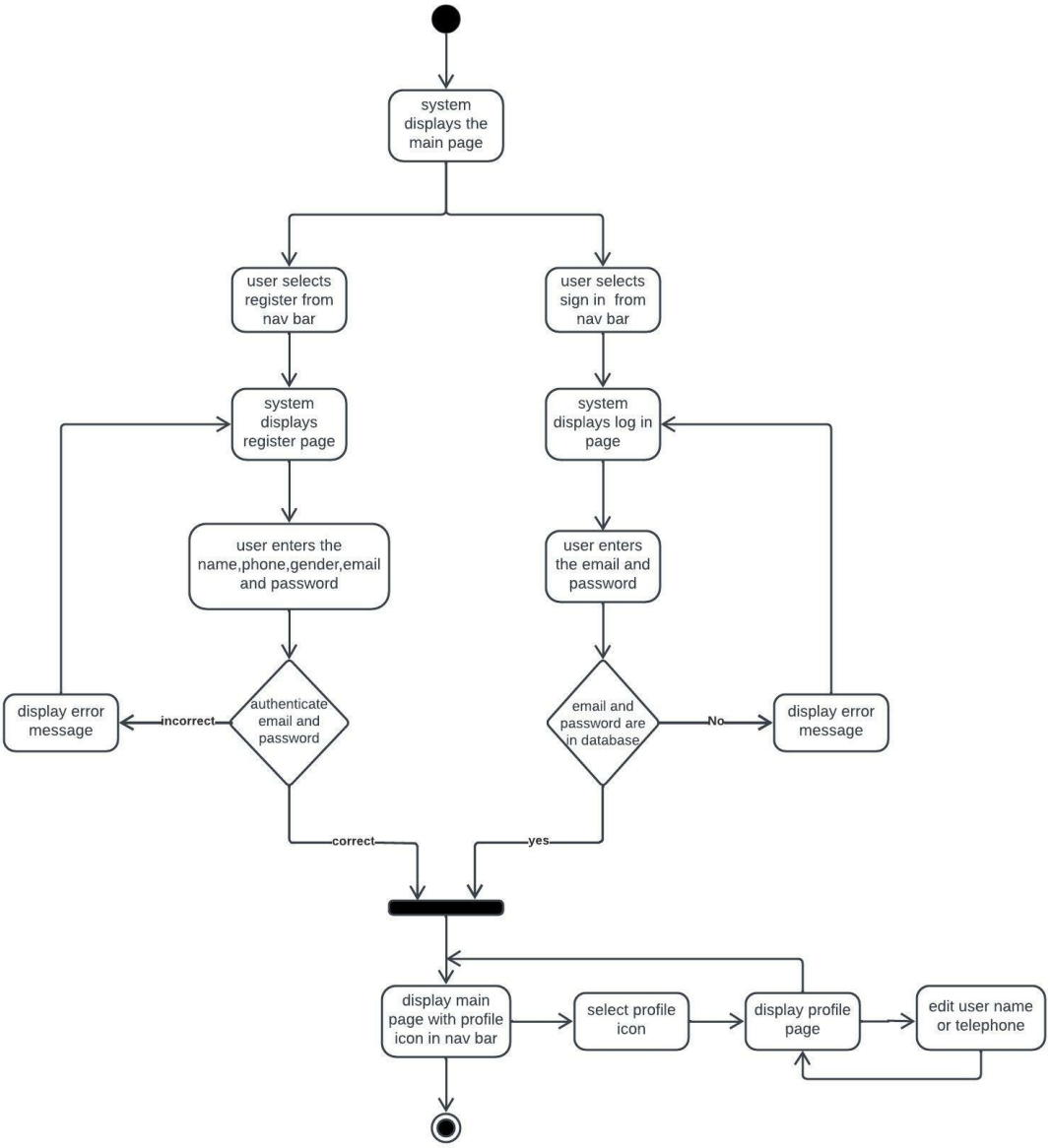
d) **UserDetailsService:** an interface is a core interface in Spring Security framework, which is used to retrieve the user's authentication and authorization information.

4. **Password encoding:** instead of storing the password of the user in plain text which can be easily hacked. There is a generator to take the provided password to encode it using **BCryptPasswordEncoder** which the spring security library supports in spring-boot.
   The security configuration is done in simple steps:
   - injection of the implementation of the users' details service
   - Definition of the authentication provider that references the details service.
   - Finally, enable the password encoder.

Detailed Use case diagrams:

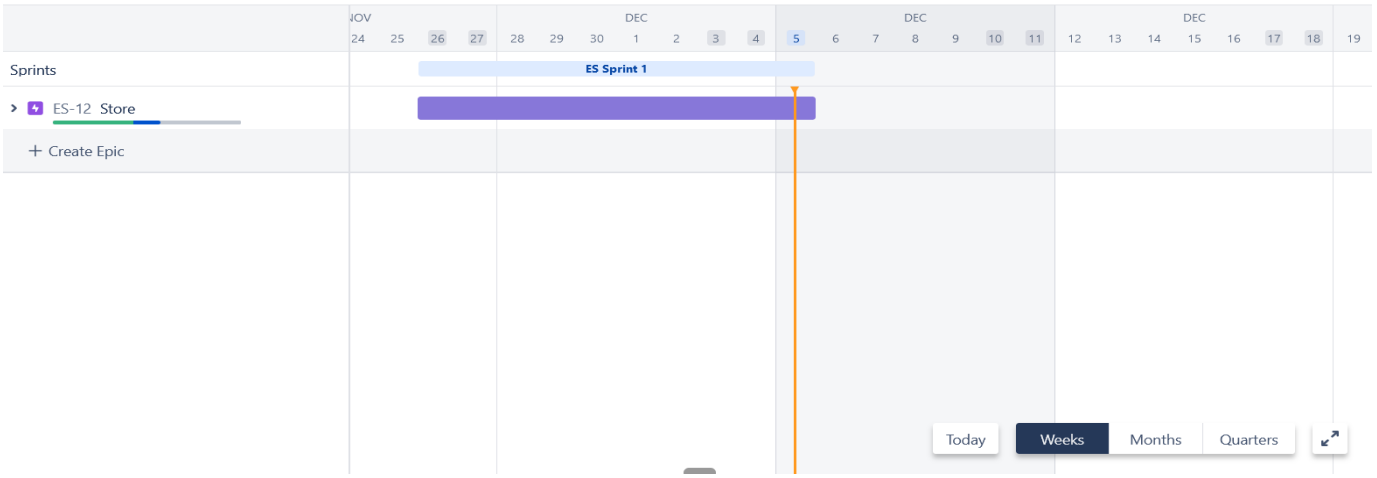


The flow of events for each of the use cases using activity diagrams:

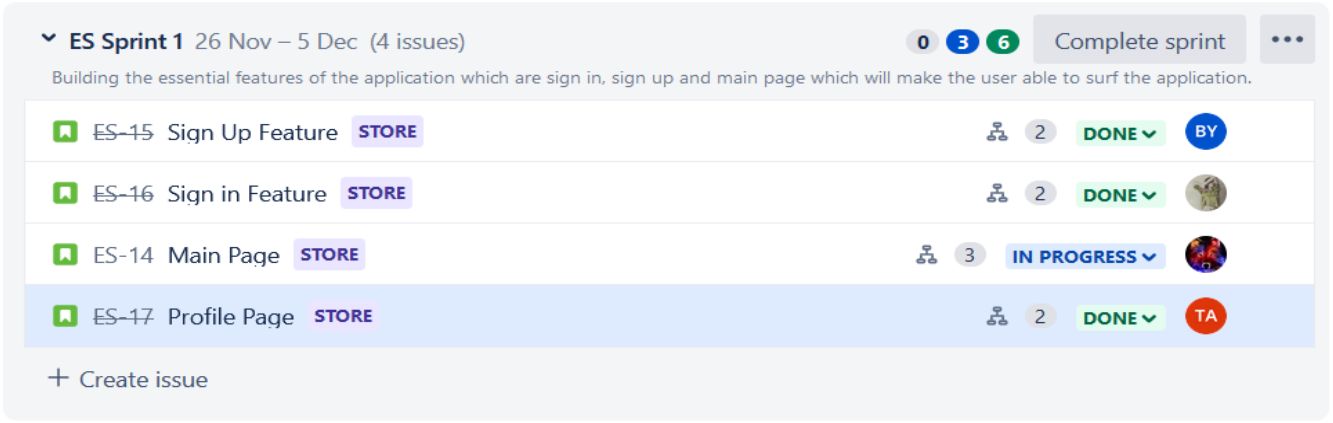


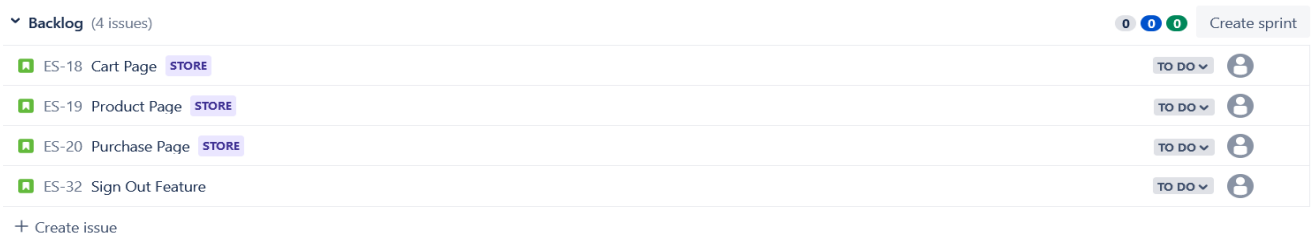
Jira report:

- **Current Jira Roadmap:**

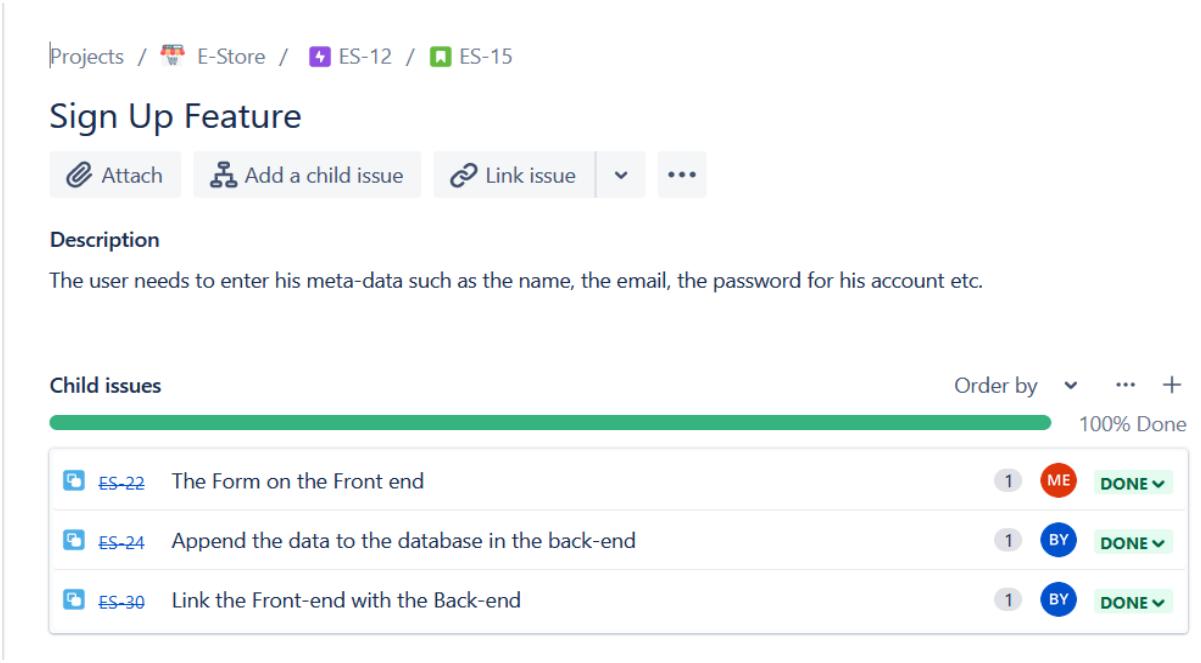


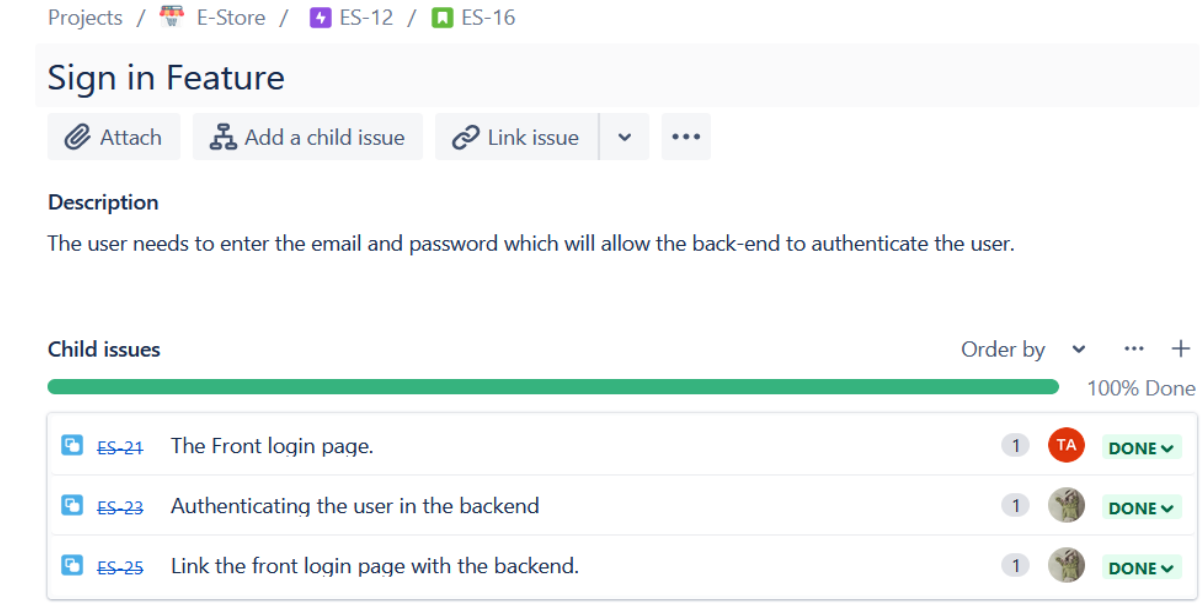
- **Milestone sprint listing the included requirements Jira Stories:**





- **Stories, tasks, and subtasks estimates:**





Projects / E-Store / ES-12 / ES-14

Main Page

Attach | Add a child issue | Link issue

Description

A complete page with a navigation bar, side bar includes the filters and the categories, and the products.

Child issues Order by

50% Done

~~ES-26~~	Navigation Bar	0.5		DONE
~~ES-27~~	Side Bar	0.5		DONE
ES-28	Product Box	0.5		IN PROGRESS
ES-29	Products Database	2		IN PROGRESS
~~ES-31~~	Routing Feature	1	TA	DONE
ES-36	Linking the back-end with the front-end	1		IN PROGRESS

Projects / E-Store / ES-12 / ES-17

Profile Page

Attach | Add a child issue | Link issue

Description

A page containing the full name, the email, the phone number, the date of birth and the gender of the user.

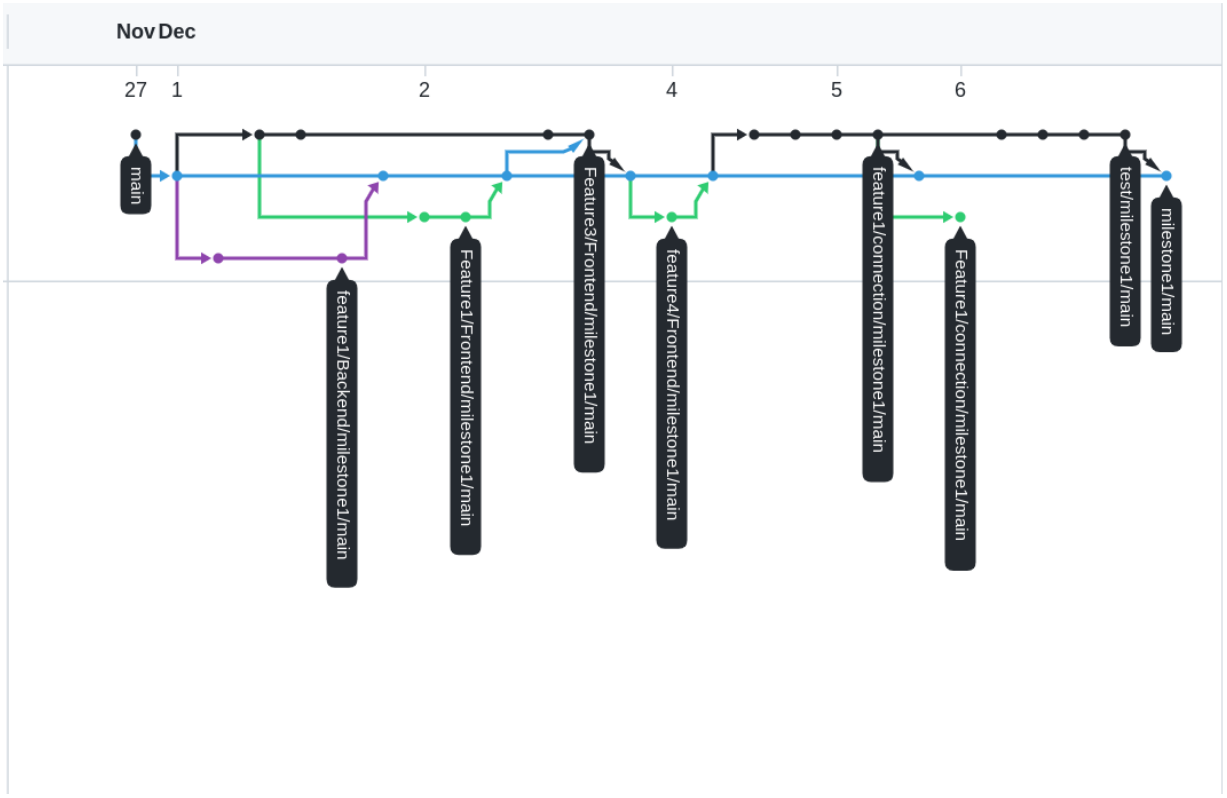
Child issues Order by

100% Done

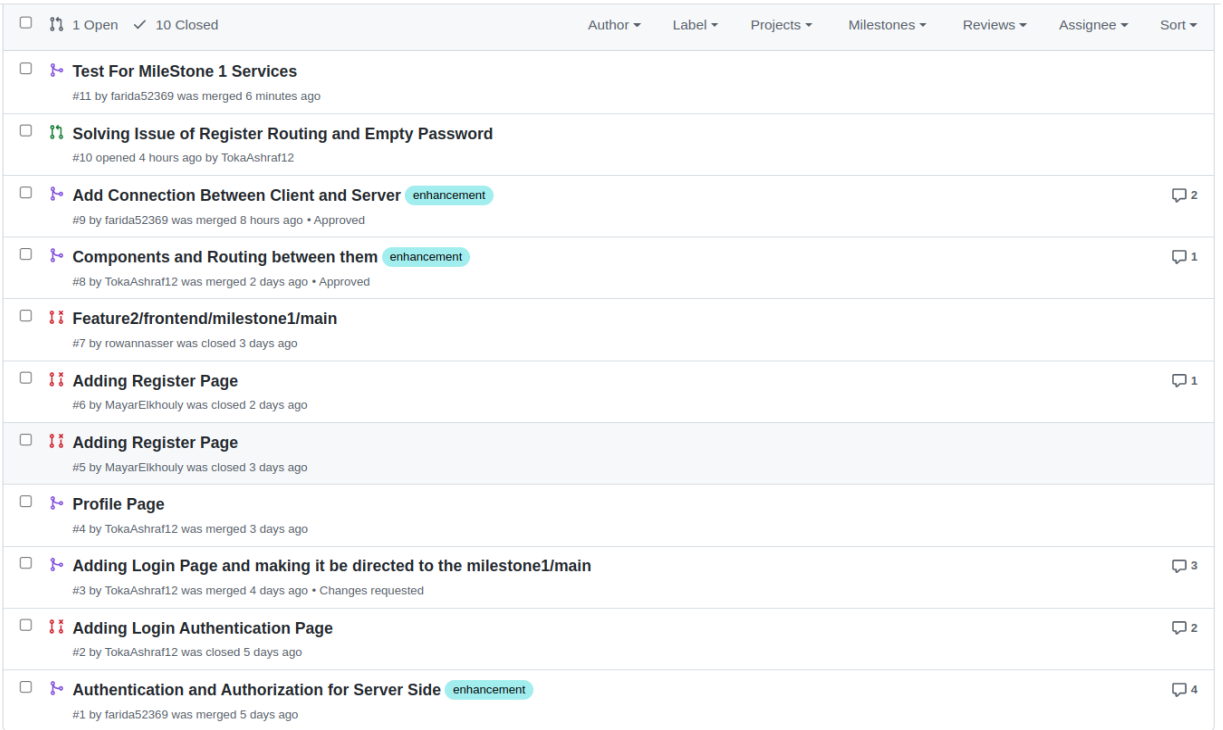
~~ES-33~~	UI in the Front end	1	TA	DONE
~~ES-34~~	The database in the backend	1		DONE
~~ES-35~~	Linking the front-end with the backend	1		DONE

Git & GitHub report showing:

- **branches diagram showing the main branch, milestone branch, and feature branches:**



- **Pull Requests:**



- **Code Reviews comments:**

1. Branch "feature1/Backend/milestone1/main" ← the branch for the Authorization and Authentication feature in the backend.

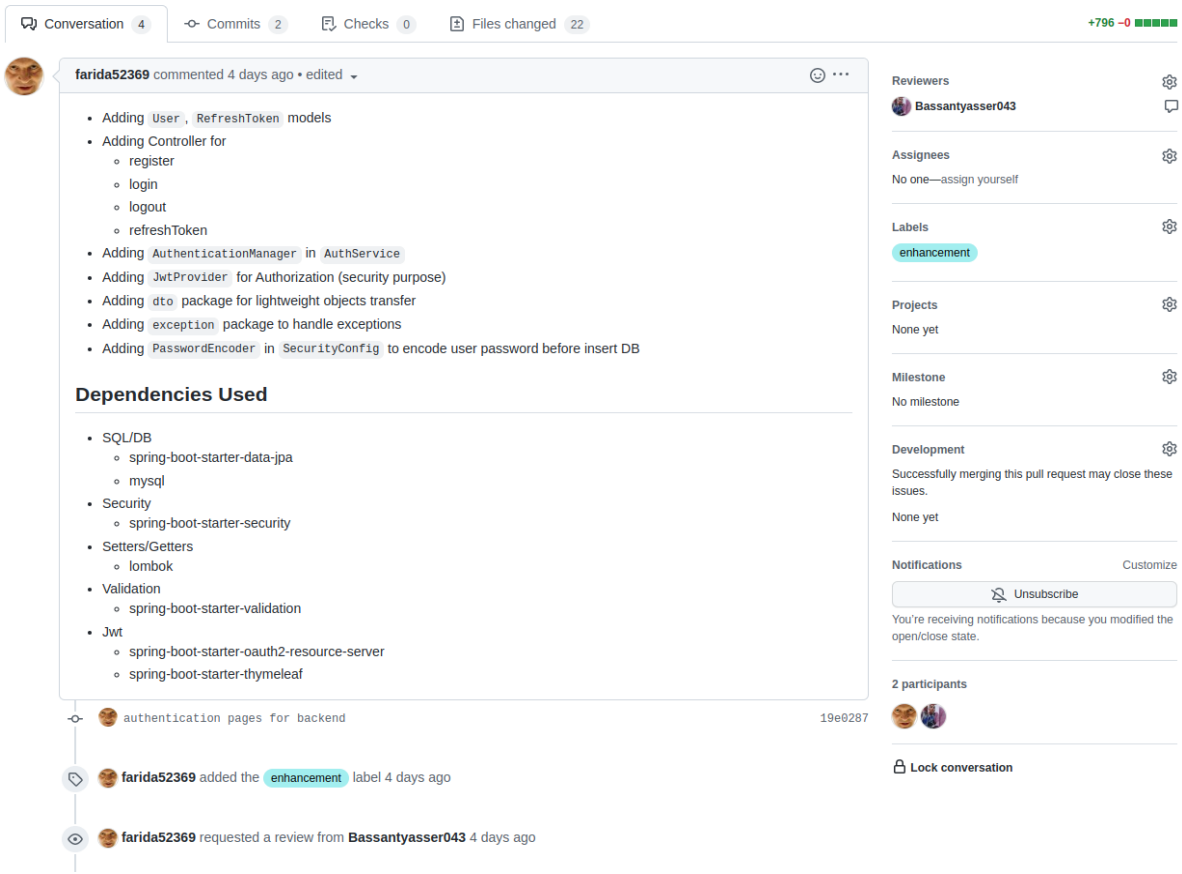


Fig. 1 'Commits of the branch'

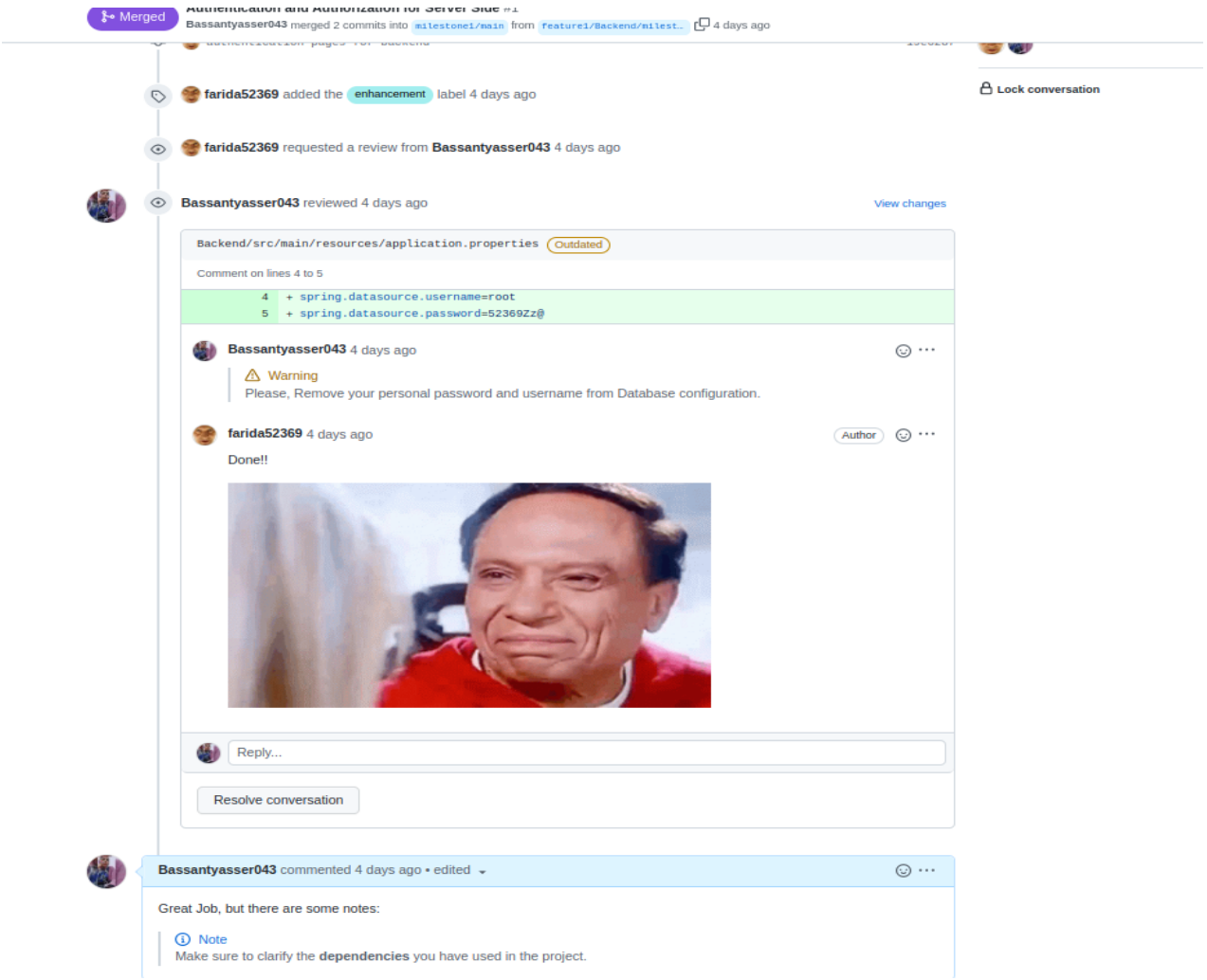


Fig. 2 'Code review for Authentication and Authorization'

2. Branch "[Feature1/Frontend/milestone1/main](#)" ← Branch for committing the sign-in component.

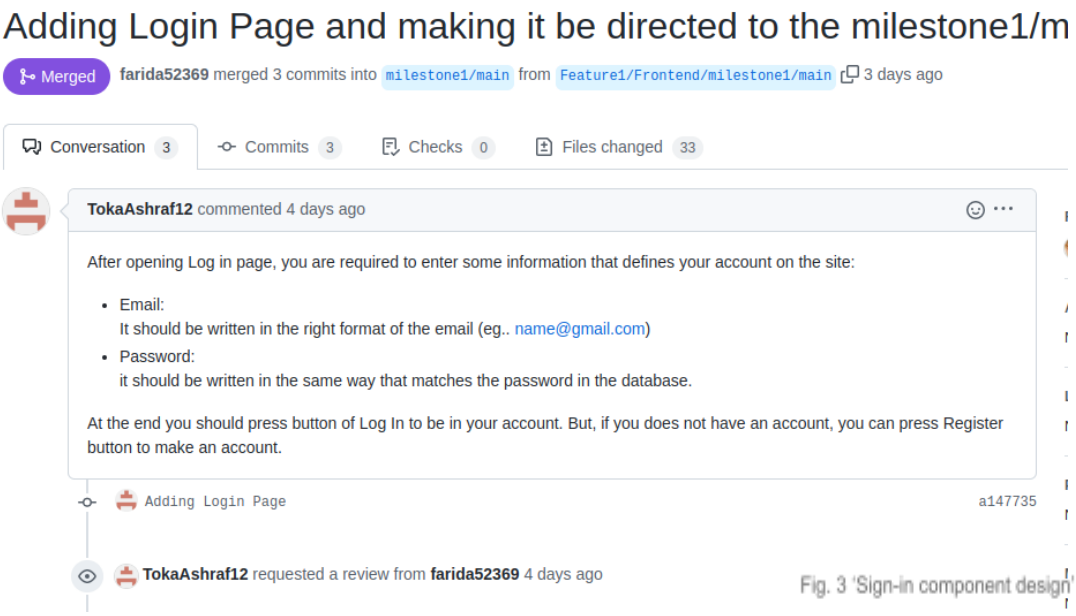


Fig. 3 'Sign-in component design'

Merged **Adding Login Page and making it be directed to the milestone1/main** #3

farida52369 merged 3 commits into `milestone1/main` from `Feature1/Frontend/miles..` 3 days ago

farida52369 commented 4 days ago • edited

Great work, Toka.
The PR performs its functionality perfectly but there are some notes to be considered:

- logo position
- background coloring (size)
- centralized box of `Log In`

They aren't responsive in the design as they differ depending on screen size.

As shown in the screenshots for two different screen sizes:

Merged **Adding Login Page and making it be directed to the milestone1/main** #3

farida52369 merged 3 commits into `milestone1/main` from `Feature1/Frontend/miles..` 3 days ago

farida52369 requested changes 3 days ago View changes

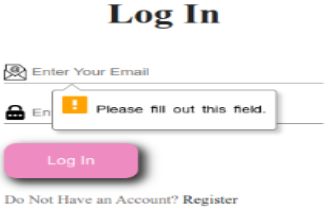
`Frontend/src/app/login/login.component.ts` Outdated

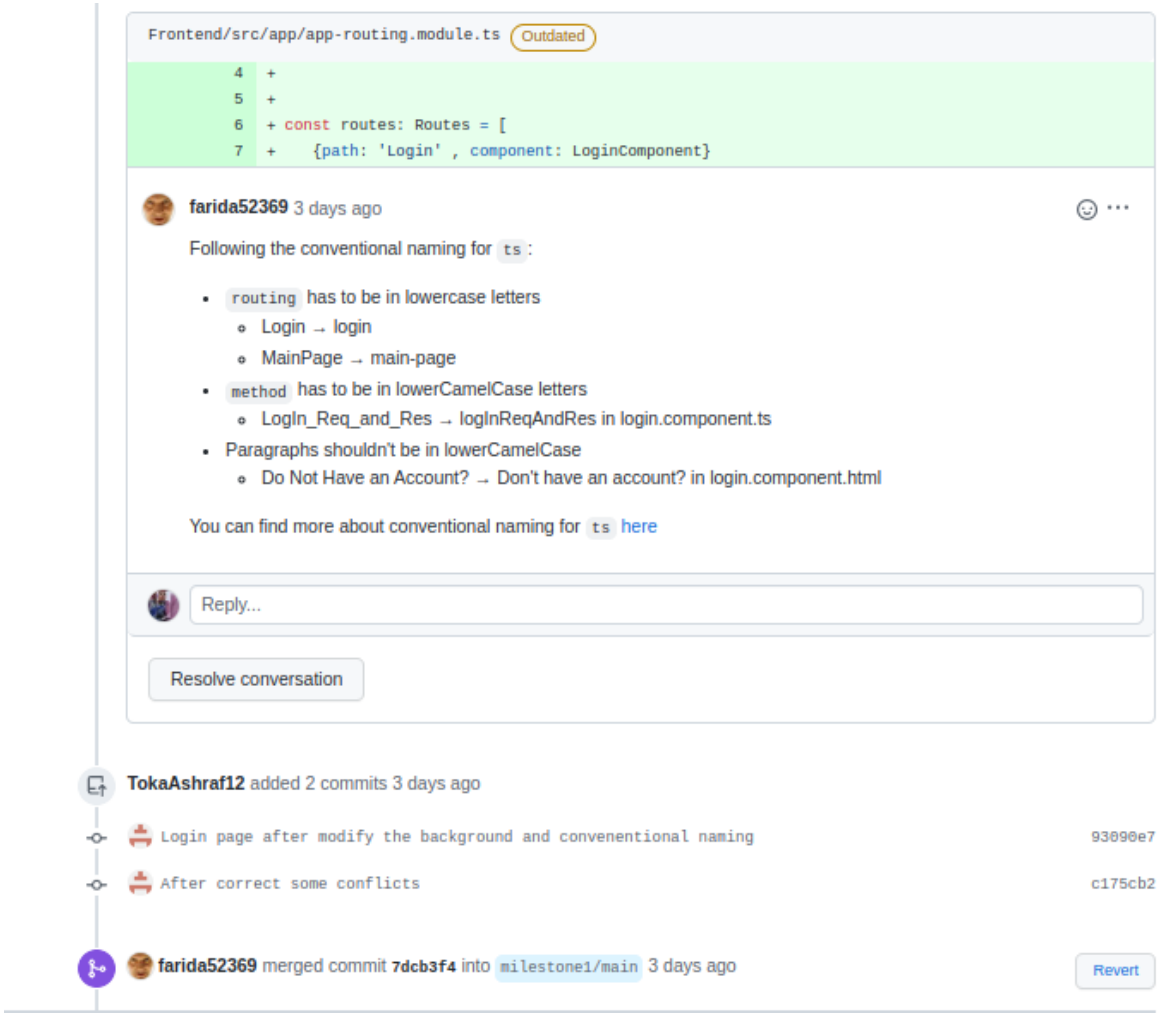
Comment on lines 20 to 23

```
20 +    if (this.password === '' || this.email === '') {
21 +      alert("Please Enter Your Email and Password");
22 +      return;
23 +    }
```

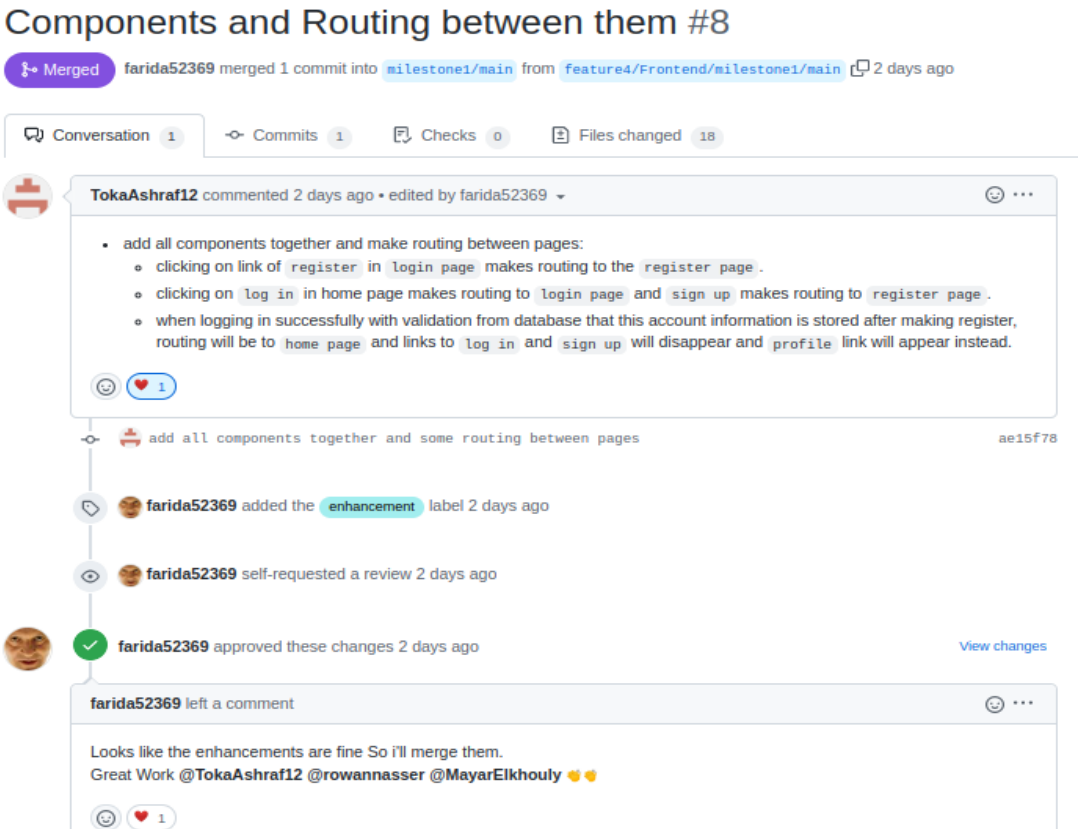
farida52369 3 days ago

`Alert` isn't a good choice to handle the emptiness of `Password` and `Email` fields as it stops the App from doing any action until clicking **OK** so it has a special cases where `Alert` is useful but ours isn't one of them.
+
You already handled the case of emptiness through this note:





3. Branch "feature4/Frontend/milestone1/main" ←Routing between different components in frontend.



4. Branch "`feature1/connection/milestone1/main`" ←Connection between frontend and backend.

Add Connection Between Client and Server #9

Merged **Bassantyasser043** merged 4 commits into `milestone1/main` from `feature1/connection/milestone1/main` 5 hours ago

Conversation 2 | Commits 4 | Checks 0 | Files changed 39

farida52369 commented 15 hours ago

- Adding HTTP connection between Client and Server using `HTTPClient` API in:
  - Login Component
  - Register Component
- Change PK in `User` model → email to have a unique key for each user
- Routing between Different Components `Login`, `Register`, `Home`

> ⚠ Warning
> Register routing doesn't function properly

farida52369 added 4 commits yesterday

`adding environment prod and some services`	`65c4575`
`remove personal info`	`f7bac46`
`add connection between components`	`4043bd9`
`remove personal info`	`ffd7bb8`

farida52369 added the enhancement label 15 hours ago

farida52369 requested a review from **Bassantyasser043** 15 hours ago

Bassantyasser043 approved these changes 5 hours ago View changes

Bassantyasser043 left a comment

- The functionality is good enough.
- The connection between login form in the `front and backend` is working well.
- Authentication method are working well and return to the front whether it is registered or not `(routing to login in case of registred before otherwise return in the same page)`.

well done ⭐⭐⭐🤚🤚💙