BANK APPLICATION PROJECT KICK OFF DOCUMENT

Project Name: SUREBANK WEB APPLICATION PROJECT

Definition: The banking application is a web application where you can access the details of your bank account and complete transactions from web browsers. The Bank application has some features include the ability to view your balance, see transaction history, transfers to other accounts at your bank, deposit and withdraw money. In addition to these. Administrator of the bank application can list the users of the application, update their information and see current bank statement and transactions.

BANK WEB APPLICATION CUSTOMER REQUIREMENTS

1. The Bank Application must be a web application.
2. The application must have user friendly UI. (Non functional requirement)
3. The application must be secure. (Non functional requirement)
4. Language of the application must be English. (Non functional requirement)
5. The application must display response of user request nearby for 5 seconds. (Non functional requirement)
6. Application should contain banner, about, products, prices and contact pages that are accessible to everyone. Users who will make a process must register the application.
7. There should be able two types of roles for authorized user: Customer, Admin
8. Authorized user is able to have at least one role or both roles
9. Users who will make a process should be able to register this application with their own information.
10. Registered users should be able to login to the application.
11. Users who has the customer or admin role should be able to have a bank account.
12. The user who has the customer role should be able to deposit, withdraw and transfer.
13. The user who has the customer role should be able to have recipients.
14. The user who has the customer role should be able to create a new repicient
15. The user who has the customer role should be able to list transactions
16. The user who has the customer role should be able to see current balance, total deposit, total transfer and total withdraw for last 15 days
17. The user who has the customer role should be able to see transactions between two dates

18. The user who has the admin role should be able to see user list.

19. The user who has the admin role should be able to update some user information.

20. The user who has the admin role should be able to list all transactions.

21. The user who has the admin role should be able to see current bank statement. (Total money in the bank etc.)

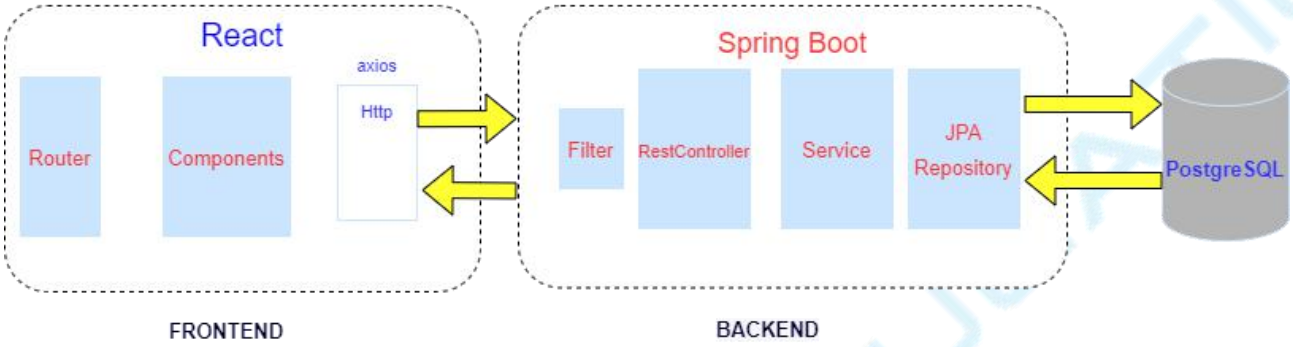
What is Use Case Diagram: In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

You will see use case diagrams of the project Graphic 1.

Graphic 1- Use Case Diagrams of the application

Frontend: The part of a web site or web application that the user interacts with directly. In this project, React Library will be used for frontend.

Backend: It is the server site the wep application. It stores and arranges data. It is the part of the web app that you can't see and interact with. In this project, Sprint boot framework will be used as a backend tecknology. Moreover PostgreSQL will be used as relational database.

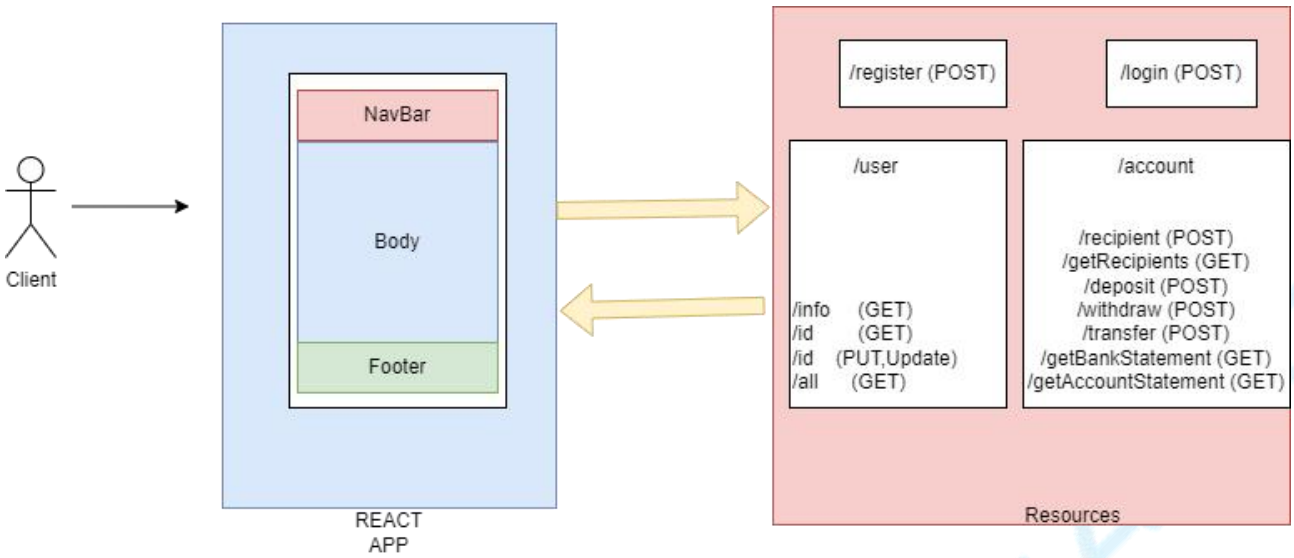


Graphic-2 General system structure of the application

In Spring's approach to building RESTful web services, HTTP requests are handled by a controller. These components are identified by the @RestController annotation. Example UserController shown in below. UserController is an example of Resource Controller.

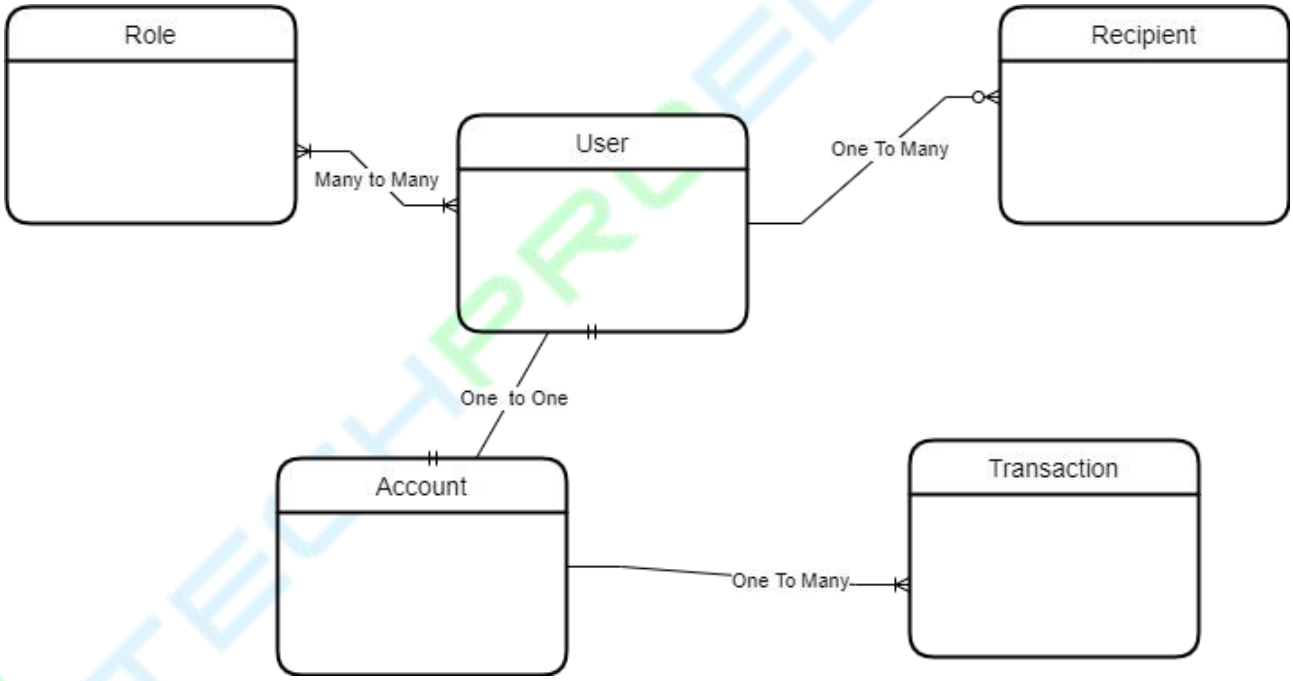
```
@RestController
@RequestMapping("/user")
public class UserController {

	@Autowired
	private UserService userService;

	@GetMapping("/info")
	@PreAuthorize("hasRole('ADMIN') or hasRole('CUSTOMER')")
	public ResponseEntity<UserInfoResponse> getUserInfo(){
		User user = userService.findOneWithAuthoritiesByUserName();
		UserDTO userDTO=convertToDTO(user);
		UserInfoResponse userInfoResponse = new UserInfoResponse(userDTO);
		return ResponseEntity.ok(userInfoResponse);
	}

}
```

There are resources below that we will consume int the project.



Graphic-3 Resources in the Application

Entities represent persistent data stored in a relational database automatically using container-managed persistence. The entities that will be created and used in the project is given below.



Graphic-4 Entities and Relations Between Entity in the Application.

One User can have many roles and One Role can belongs to Many User (Many To Many relation)
One User has an Account (One To One relation)
One User can have many recipients (One To Many relation)
One Account can have many transactions (One To Many relation)

DEVELOPMENT ENVIRONMENT, IDEs and TOOLS – INSTALLATIONS

1. FOR BACKEND

- Java 11 https://www.oracle.com/tr/java/technologies/javase/jdk11-archive-downloads.html
- Spring Tool Suite
  Go to https://spring.io/tools
  - Download Spring Tools 4 for Eclipse depending on your OS.
  - The file that you donwload is a jar file. Extract the file.
  - You will see contents.zip and extract it as well. After extract contents.zip you will see sts-4.13.0.RELEASE
  - Runnable File SpringToolSuite4.exe will be sts-4.13.0.RELEASE folder.
- Postman https://www.postman.com/
- PostgreSQL 12.9 https://www.enterprisedb.com/downloads/postgres-postgresql-downloads
  Guide for installation POSTGRESQL: https://www.postgresqltutorial.com/install-postgresql/

2. FOR FRONTEND

- Visual Studio Code
- Node.js

THE TOPICS THAT YOU SHOULD REVIEW BEFORE STARTING PROJECT

FOR BACKEND (SPRING BOOT PROJECT)

1. Core Java: OOPS, classes, enums, interfaces, exception handling, collections, stream (foreach, filter, map), lambda,optional key word and other fundamentals.
2. Logging (SLF4J, Logback)
3. Regular expressions
4. What is Spring Framework.
5. What is Spring Boot Framework
6. What is Spring Security Framework. (JWT Based Security)
7. What is inversion of control , dependency injection and Spring IOC Container
8. What is JPA, Hibernate and Spring Data JPA
9. What is entity class and how to create it
10. What are OneToOne, OneToMany, ManyToOne, ManyToMany relations on hibernate.
11. How to use @JoinTable, @JoinColum annotations.
12. JPQL (Java Persistence Query Language), Basic SQL knowledge
13. What is REST API
14. How , why to use @Bean, @Autowired, @RestController, @Service, @Repository annotations.
15. Controller-Service-Repository layered structure in spring boot app
16. What is @Transactional annotation in org.springframework.transaction.annotation
17. What is the Data Transfer Object and how to use it.
18. Usage of @ResponseEntity, @ResponseBody, @Request Body, @RequestMapping, @PostMapping, @GetMapping, @DeleteMapping, @PutMapping. @Valid annotations.

19. HTTP Response Status Codes. (200, 201, 400, 404, etc.)
20. Why and How to use @PathVariable and @QueryParam annotations
21. Project Lombok

FOR FRONTEND (REACT PROJECT)

1. Core JavaScript and ES6 features: variable, objects, functions (regular, arrow), ternary operator, export, import, destructuring, promise, (async, await), callback function, error handling (try, catch), Date object
2. Basic html, css knowledge
3. React features: Functional component, props, state, Lists and Keys (rendering multiple components, map), Hooks (**useState, useEffect, useContext, useReducer**), conditional rendering, event handling, routing
4. Browser LocalStorage use
5. **Reactstrap**: UI library like bootstrap
6. **Formik** and **Yup** for forms
7. **Axios:** for api calling
8. **React-Toastify** for notification