Currency Transfer

**Introduction:**

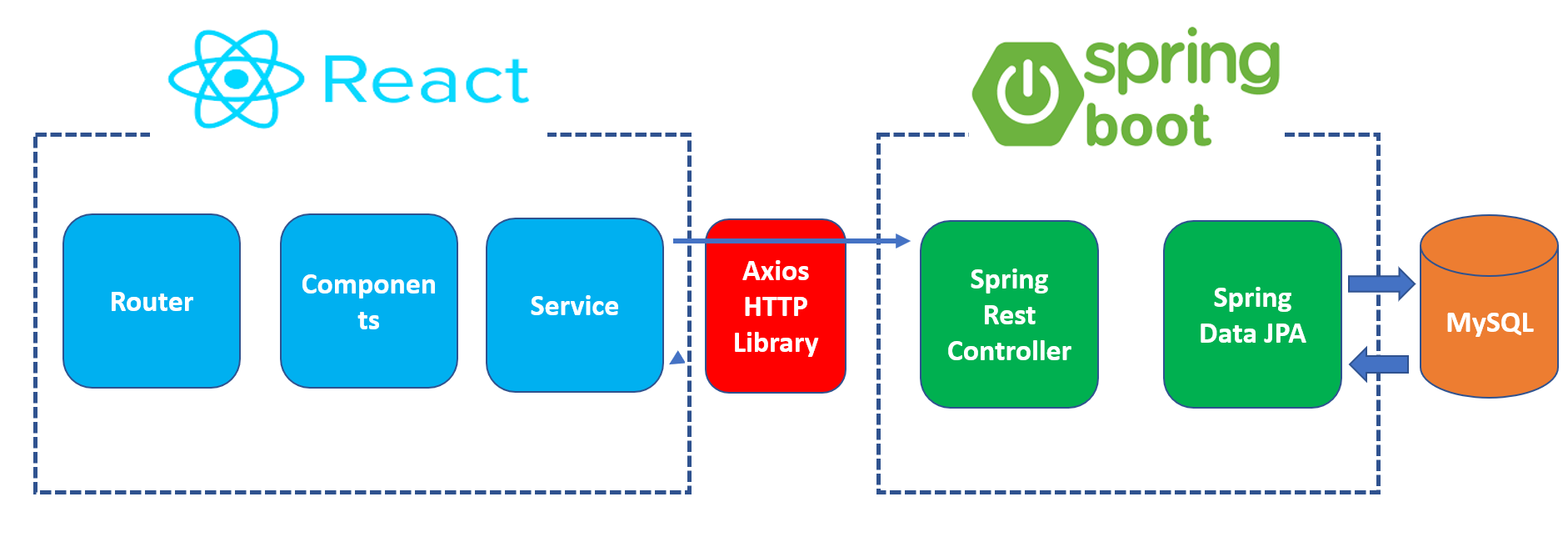
We are creating an application which can transfer funds between the same bank as well as different banks/accounts. Using the said application funds could be transferred between the same currency and the list of currencies given in the application. The application will first convert the domestic currency to the choice of currency from the list of currencies in the application and then it will transfer the funds. For example, if USD is to be transferred to another bank account then INR will be debited from the customer’s account, converted into USD and then transferred to the other account.

**Technology in scope:**

❖ Reactjs for the frontend

❖ Springboot for the backend

❖ MySQL, Monogodb for database

**Project Architecture:**

Above diagram shows the high level structure of the overall application and how different technologies are going to work together.

**Scope of the project:** This application will enable inter account transfer between same currencies.  
This will also enable the transfer between two different currencies, between two different accounts and within the same multi-currency account

**Functional Requirement:** The application will have following functionalities.

1. Customer should login and register with the platform. While registering customer need the following information.

❖ Email

❖Name

❖Address

❖Account number

❖Mobile number

❖Password

1. Customer can access the following services.

**Fund Transfer:** Frontend will have several functionalities such as it would be able to transfer currencies (same currency / other currency) between the same bank accounts as well as other bank accounts. Fund transfer option would also be available in the dashboard. User will have to register and login first to access the fund transfer service. If the user is already registered, user will have to login first to access the fund transfer service.

**Edit profile:** One of the component of the application will contain the edit profile option where the user can access the his/her personal details that was submitted by the user during the registration process. The user will be able to see his profile, will able to add and will able to make changes to his profile.

**Dashboard:** There will be a dashboard, which will be accessed by the user after successfully login to the page. The dashboard will contain the balance information of the account, all the available services of the application.

**Balance Status**: User will be able to check it’s account balance in the dashboard section.

**Transaction history:** User will also be able see up to last 5 transaction history done through the application.

**Payee details:** User will be able to add, delete, edit and see the payee details in the application. Once the payee is added only then the user will be able to transfer the funds to the payee. The payee details will be save in the Monogodb data base.

1. Currency exchange protocol should be followed (base currency GBP,USD,Euro) to remaining exotic currencies.

**Data Models:**

**Frontend UI:** Frontend of the application will be in Reactjs. There will be several components in the react frontend application which will take care of different functionalities of the application. These different components will be connected to each other in App.js file through router.

To improve the reusability of code and to improve response time of the application we use Reactjs. In each of the component of the application our Header and Footer will remain the same.

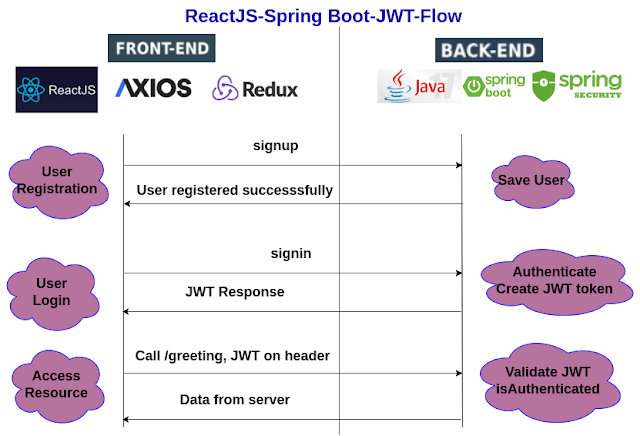
**User Registration and Login**

**Registration:** The user registration data will be saved in MySQL data base. User have to provide details such as Name, Email, Address, Mobile number, Password and confirm Password details while registering.

Proper authentication will be placed in the react user form. All the fields in the user form will be mandatory. Phone number should be of 10 digits. User have to provide proper email address.

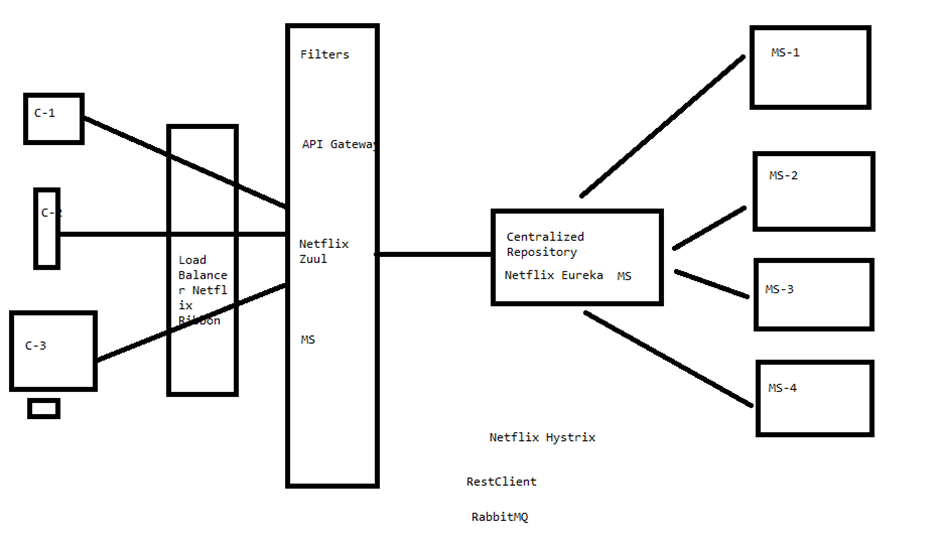
**Login:** Login authentication will also be placed. The user will be able to access the application once he will entry the correct combination of email and password that he entered while registering. In case of entering incorrect password, login page will give an error message to entre the correct email and password.

**User Registration and Login**



**Target Audience/Users of the system:** Retail customers and organisations involved in transferring money.

**Springboot Backend:**  Backend of the application will be made in springboot. Springboot have different layers. React frontend will be connected to the springboot backend. This connection will be made using by calling an API. Cross origin annotation will be used to call the API, which will connect the react frontend with springboot backend.



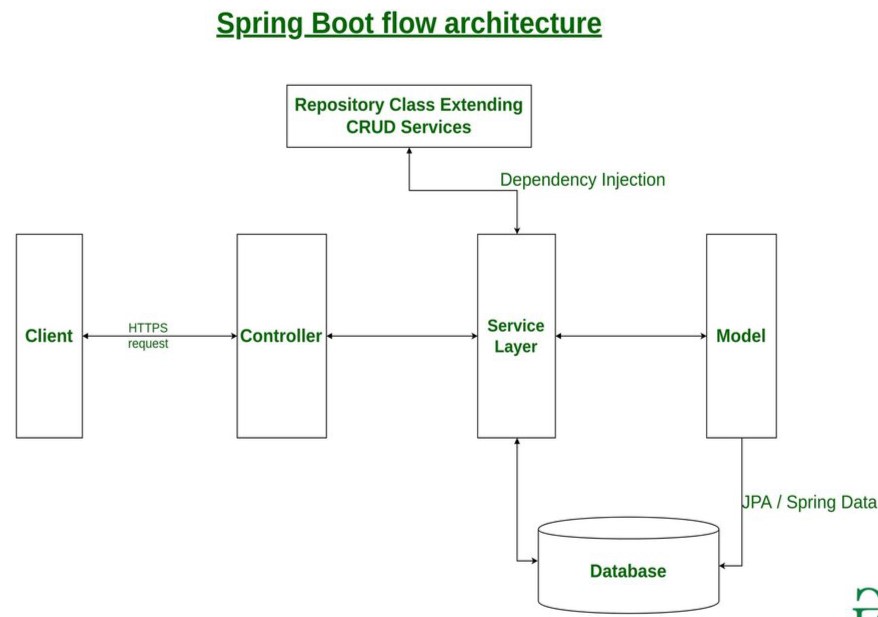
Above picture shows the springboot layers and their functions and how logically these layers are connected to one another.

**Services:**

**Customer:** User will be able to register himself. User will be able to login in the application after authenticating the credentials which the user has entered while doing the registration. Post authentication use will be able to access his profile details and will also be able to make changes to it.

**Balance:** Springboot will provide a middle layer for the UI and the MySQL database. This service will be created to add the functionality to show the balance to the customer when he sends a request for it.

**Transaction:** Springboot will provide a middle layer for the UI and the Mongodb database. This service will be created to add the functionality to show the transaction history to the customer when he sends a request for it.



There will be three layers in the springboot backend namely as controller, service and repository.

**Controller**: Will interact with UI. Frontend will be linked to the controller using crossorigin annotation. This layer will be further linked to Service layer using autowire.

**Service:**  This layer will hold the business logic for all the service of the application. Example: checking balance and transaction history. This layer will be connected to the repository layer.

**Repository:** This layer will extends the JPA repository where we will write the logic to connect the backend of the application with the MySQL database.

Test cases will also be written for the these components.

**Database:** we will be using My SQL and Mongodb databases. These databases will be used to store all the important data such as user registration information. Since MySQL stores data in tabular format, it will be storing user registration data and Payee information. Mongodb data will be used to store the transaction history.

