DESCRIPTION

Create a dynamic and responsive Python online banking web application to deposit, withdraw, and transfer the money between the accounts.

**Background of the problem statement:**

Abc is one of the top banking firms that accepts deposits from the public for the purpose of lending loans to the public. It also invests an amount in securities.

Recently, the business analysts noticed a drop in the number of customers of the bank. They found out that online banking systems of banks like AXIS and American Express are gaining more profits by eliminating middlemen from the equation. As a result, the team decided to hire a Full Stack developer who can develop an online banking web application with a rich and user-friendly interface.

You are hired as one of the Full Stack Python developers and have been asked to develop the web application. The management team has provided you the requirements and their business model so that you can easily arrange different components of the application.

**Features of the application:**

1. Registration
2. Login
3. Account transactions
4. Transfers
5. Savings Pass Book Details
6. Profile settings
7. Requesting cheque books
8. Requesting Home/Personal/Car/Gold Loans
9. Create RD/FD
10. Request for Debit/Credit Card
11. About us/Contact us Information
12. Compliments and Complaints

**Recommended technologies:**

1. Database management: MongoDB
2. Back-end logic: Python programming, Django framework
3. Front-end development: HTML/CSS, Django
4. Automation and testing technologies: Pyunit/Pytest
5. DevOps and production technologies: Git, GitHub, Jenkins, Docker, and AWS

**Project development guidelines:**

* The project will be delivered within three sprints with every sprint delivering a minimal viable product.
* It is mandatory to perform proper sprint planning with user stories to develop all the components of the project.
* The learner can use any technology from the above-mentioned technologies for different layers of the project.
* The web application should be responsive and should fetch or send data dynamically without hardcoded values.
* The learner must maintain the version of the application over GitHub and every new change should be sent to the repository.
* The learner must implement a CI/CD pipeline using Jenkins.
* The learner should also deploy and host the application on an AWS EC2 instance.
* The learner should also implement automation testing before the application enters the CI/CD pipeline.
* The learner should use Git branching to separately perform the basic automation testing of application.
* The learner should make a rich front-end of the application, which is user- friendly and easy for the user to navigate through the application.
* There will be two portals in the application, namely the admin and user portal.

**Admin Portal:**

It deals with all the back-end data generation and product information. The admin user should be able to:

* Authorize the roles and guidelines for the user
* Grant access to the user regarding money transfer, deposits, and withdrawal
* Grant access to the user regarding RD/FD account
* Block the user account in case of any threat
* Authorize the cheque book requests
* Authorize the Home/Persona/Car/Gold Loan requests

**User Portal:**

It deals with the user activities. The user should be able to:

* Register or log in to the application to maintain a record of activities
* Deposit and withdraw money from the account
* View transactions and balance in the primary and savings account
* Transfer funds between different accounts and add recipients
* Request cheque books for different accounts
* Request debit/Credit card
* Request Home/Personal/Car/Gold Loan

**Recommended technologies:**

* **Database:** MongoDB
* **Backend:** Django and Python
* **Frontend:** HTML, CSS, JavaScript, and ReactJS
* **DevOps tools/technologies:** Git and GitHub

**Tools and technologies used:**

1. **HTML (Hypertext Markup Language):**

It is the standard markup language for creating web pages and web applications, providing the structure and content for web pages.

This project will use HTML to define the structure and layout of the various UI components, such as forms, buttons, and lists.

1. **CSS (Cascading Style Sheets):**

It is a stylesheet language used for describing the look and formatting of a document written in HTML. It allows developers to separate content from presentation, making it easier to maintain and update the design of a web application.

In this project, CSS will style the UI components and create a visually appealing and user-friendly design.

1. **React:**

It is a popular open-source JS library for developing dynamic websites. It provides a set of pre-built libraries that make it easier to create consistent and modern web applications.

React SPA (single-page application) features, reusable components, and routing will be used to create a dynamic website. We will make use of JavaScript-fetched APIs to communicate with the backend application created using the Django framework.

1. **Python** : Python is one of the open source programming language which help to write the business logic. Using Python with third party library we can develop the Rest API.

1. **Django:** It is a popular Python framework for building dynamic web applications. It allows developers to create reusable templates and views. It also enforces the separation of model, views, and controller when implementing the MVC architecture.

In this project, Django will be used for creating and managing the various UI components, such as the registration form, login, order, dashboard, and so on.

1. **MongoDB:** It is a non-relational database that persists data. It provides a flexible way to store and manage data in a database using NoSQL and the concept of collection. It can be easily integrated with Python for backend applications to communicate with the database.

In this project, MongoDB will be used to manage the application state.

**Deployment:**

1. **Git and GitHub** 
   1. create local repository for both the Project ie Python backend with Django and react js frontend.
2. **Dockerize application**
   1. Create the Dockerfile for the Python backend and push the image on Docker Hub
   2. Create the Dockerfile for React front-end and push the image on Docker Hub
   3. Create the Docker-compose file which is responsible to run the 3 container ie react js front end container, python with Django backend container and mongo db as database container.
3. **Push the code to github account** 
   1. Push the code to user-defined github repository which contains python with Django backend code, react js frontend code, docker image for backend and frontend and docker-compose
4. **Login to AWS account** 
   1. Create the EC2 instance in AWS Account.
   2. Install required software as
      1. Git : to pull the code from docker hub account
      2. Java : required to run the Jenkin tool.
      3. Jenkin : responsible to run the Jenkin job.
      4. Docker : to run the docker images
      5. Docker-compose : to run more than one container.
5. **Deploy** 
   1. Create Jenkin job which is responsible to pull the project from github account and run the docker-compose file which is responsible to run three container ie frontend – react js, backend – python with Django and mongo db database.
   2. Access that application using public id address or domain provided by ec2 instance.