Take Home Assignment

Instructions:

You have 2 days to complete the assignment. Submit your code via a GitHub repository link. You will also be required to demo the working project locally during the interview.

Task: Loan Application Microservice

Your task is to implement a simple microservice that allows customers to:

1. Apply for a loan.
2. View their loan application status.
3. Update the loan application details (such as amount, repayment period).

You'll implement both the backend service and a basic frontend to interact with it.

Requirements:

Backend:

- **Language**: Java (Spring Boot), Node.js, or Python (use whichever you are most comfortable with).
- **Database**: Use PostgreSQL or MongoDB.
- **APIs to implement**:
  a. **POST /api/loans/apply** – Create a new loan application.
    - Request body:

```
{
  "customerId": "string",
  "loanAmount": "number",
  "repaymentPeriod": "number", // in months
  "loanPurpose": "string"
}
```

    - Response:

```
{
  "loanId": "string",
  "status": "PENDING"
}
```

  b. **GET /api/loans/{loanId}** – View loan application status.
    - Response:

```
{
  "loanId": "string",
  "customerId": "string",
  "loanAmount": "number",
  "repaymentPeriod": "number",
  "loanPurpose": "string",
  "status": "PENDING | APPROVED | REJECTED"
}
```

c. **PUT /api/loans/{loanId}** – Update loan application details.

- Request body (any field can be updated):

```
{
  "loanAmount": "number",
  "repaymentPeriod": "number"
}
```

Frontend:

- **Tech stack**: Use React, TypeScript, or Vue for the frontend.
- **UI**: Build a simple web form to:
  a. Apply for a loan.
  b. Check loan status.
  c. Update loan details.
- The form should call the backend APIs you've implemented.

Bonus:

- Implement validation on both frontend and backend.
- Use Docker to set up the backend and database, with a `docker-compose` file to run the services locally.
- Write basic unit tests for the backend APIs.

Deliverables:

- A **GitHub repository** with:
  - **Backend code** in a directory called `backend/`.
  - **Frontend code** in a directory called `frontend/`.
  - **README**: Instructions on how to run the application locally, both backend and frontend.
  - Ensure that the project can be demoed easily without deployment complications.

Evaluation Criteria:

- **Correctness**: Does the solution work as expected?
- **Code quality**: Is the code clean, well-organized, and easy to follow?
- **Efficiency**: Does the code handle the task in an efficient way?
- **UI/UX**: Is the frontend user-friendly and intuitive?
- **Bonus points**: For Docker setup and automated testing.