**Architecture Document**

## 1. Project Overview: SharePlate

**SharePlate** is a food redistribution system designed to connect food donors, such as restaurants, grocery stores, and individuals, with charities, food banks, and people in need. The system ensures surplus food is efficiently redistributed, minimizing food waste and helping communities.

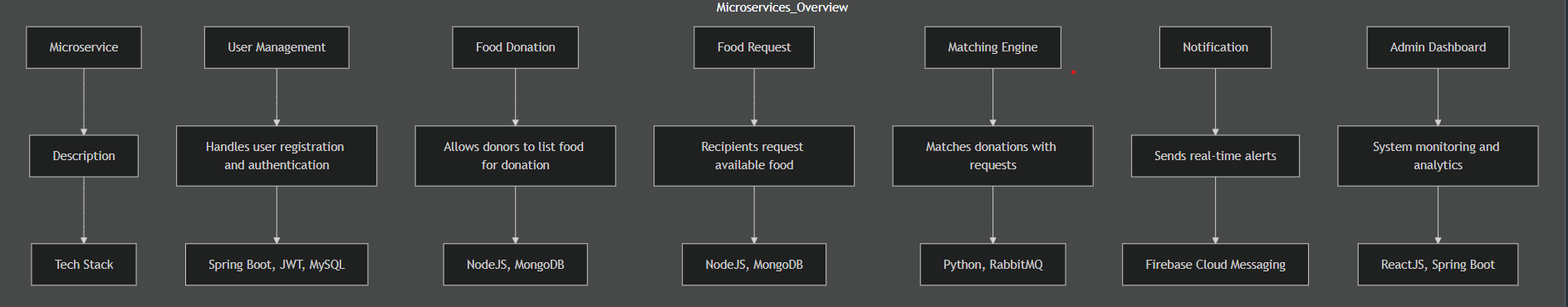
## 2. Chosen Architecture: Microservices Architecture

### 2.1 Justification for Microservices

We selected **Microservices Architecture** for SharePlate because of the following reasons:

1. **Scalability:** Each service can scale independently based on load. For example, the order matching service can scale during peak donation hours.
2. **Flexibility:** Different services (donor, recipient, and admin services) can be developed, deployed, and updated independently.
3. **Technology Diversity:** Each microservice can use the most suitable technology stack, such as Node.js for APIs, React for frontend, and Python for machine learning-based food matching.
4. **Resilience:** A failure in one microservice (e.g., notification service) does not crash the entire system.
5. **Faster Development:** Smaller, manageable services allow different teams to work simultaneously, speeding up development and deployment.

## 3. System Components (Microservices Breakdown)



## 

## 4. System Workflow

1. **Donor:** Registers, lists surplus food with quantity, location, and expiration details.
2. **Recipient:** Registers, searches for available food, and requests items.
3. **Matching Engine:** Matches donations and requests based on proximity and urgency.
4. **Notification Service:** Notifies both parties of a successful match.
5. **Admin Panel:** Monitors system performance and tracks donations.

## 5. Detailed Diagrams

### 5.1 Use Case Diagram

****

### 

### 

### 

### 

### 

### 

### 

### 

### 

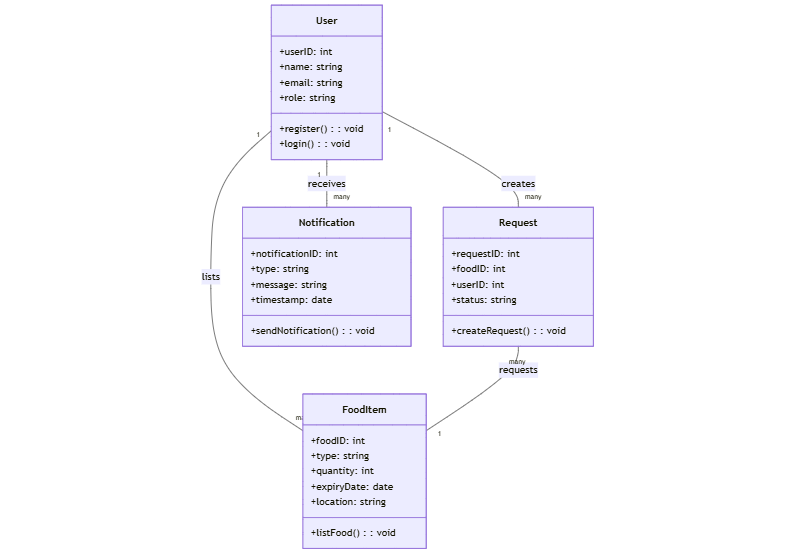
### 

### 

### 

### 

### 5.2 Class Diagram

****

### 

### 

### 

### 

### 

### 

### 

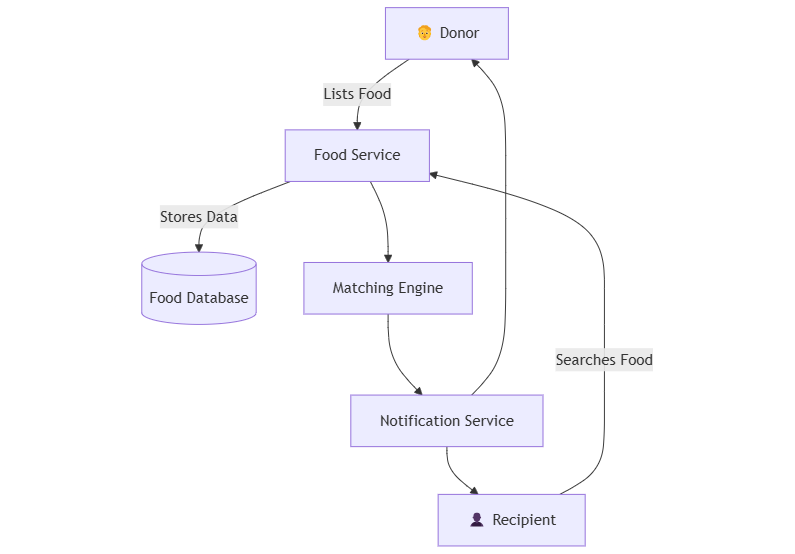
### 

### 

### 

### 5.3 Data Flow Diagram (DFD).

**Level 1:**

****

### 

### 

### 

### 

### 

### 

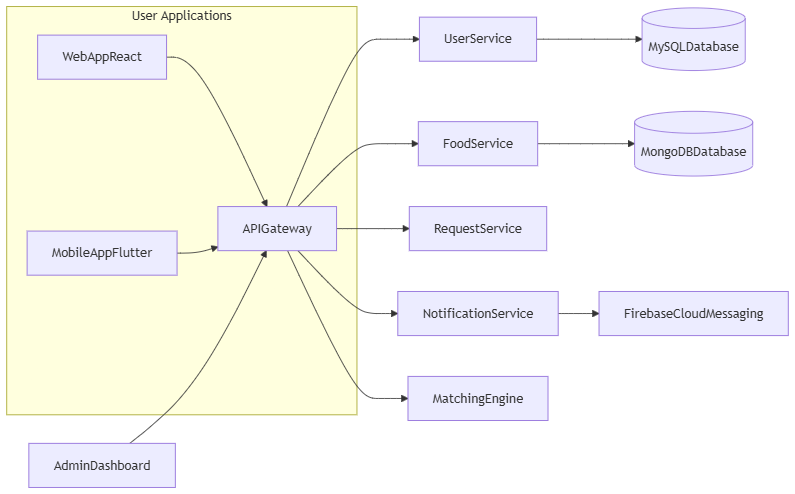
### 

### 

### 5.4 Component Diagram

**Key Components:**

* **Frontend:** Web & Mobile App (React, Flutter)
* **API Gateway:** Manages incoming API requests.
* **Microservices:** User, Donation, Request, Matching, Notification.
* **Databases:** MySQL for user data, MongoDB for food items.



.

### 

### 

### 

### 

### 

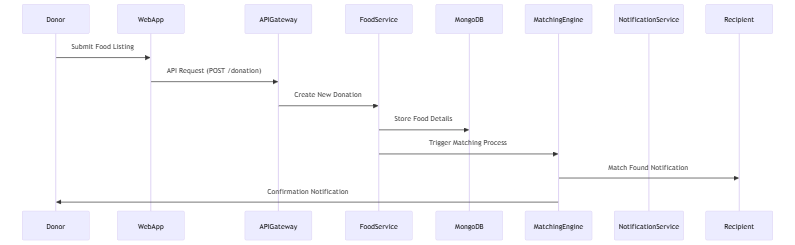
### 

### 

### 5.5 Sequence Diagram (Food Donation Flow)

**Steps:**

1. Donor submits a food listing.
2. System stores it in the database.
3. Matching engine finds a suitable recipient.
4. Notification is sent to both donor and recipient.

**

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

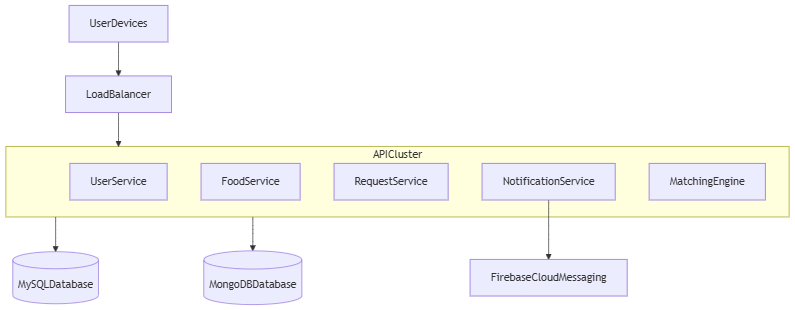
### 

### 

### 5.6 Deployment Diagram

**Environment:**

* AWS Cloud Infrastructure
* Load Balancer for traffic distribution
* Kubernetes for microservice orchestration
* MySQL and MongoDB databases
* Firebase for notifications



## 6. Conclusion

The **Microservices Architecture** is the most suitable choice for SharePlate due to its scalability, flexibility, and resilience. Each service operates independently, ensuring system stability and efficient food redistribution. The architecture also simplifies updates and maintenance, ensuring a seamless user experience for donors, recipients, and administrators.

Would you like me to provide more detailed diagrams or expand on any specific section?