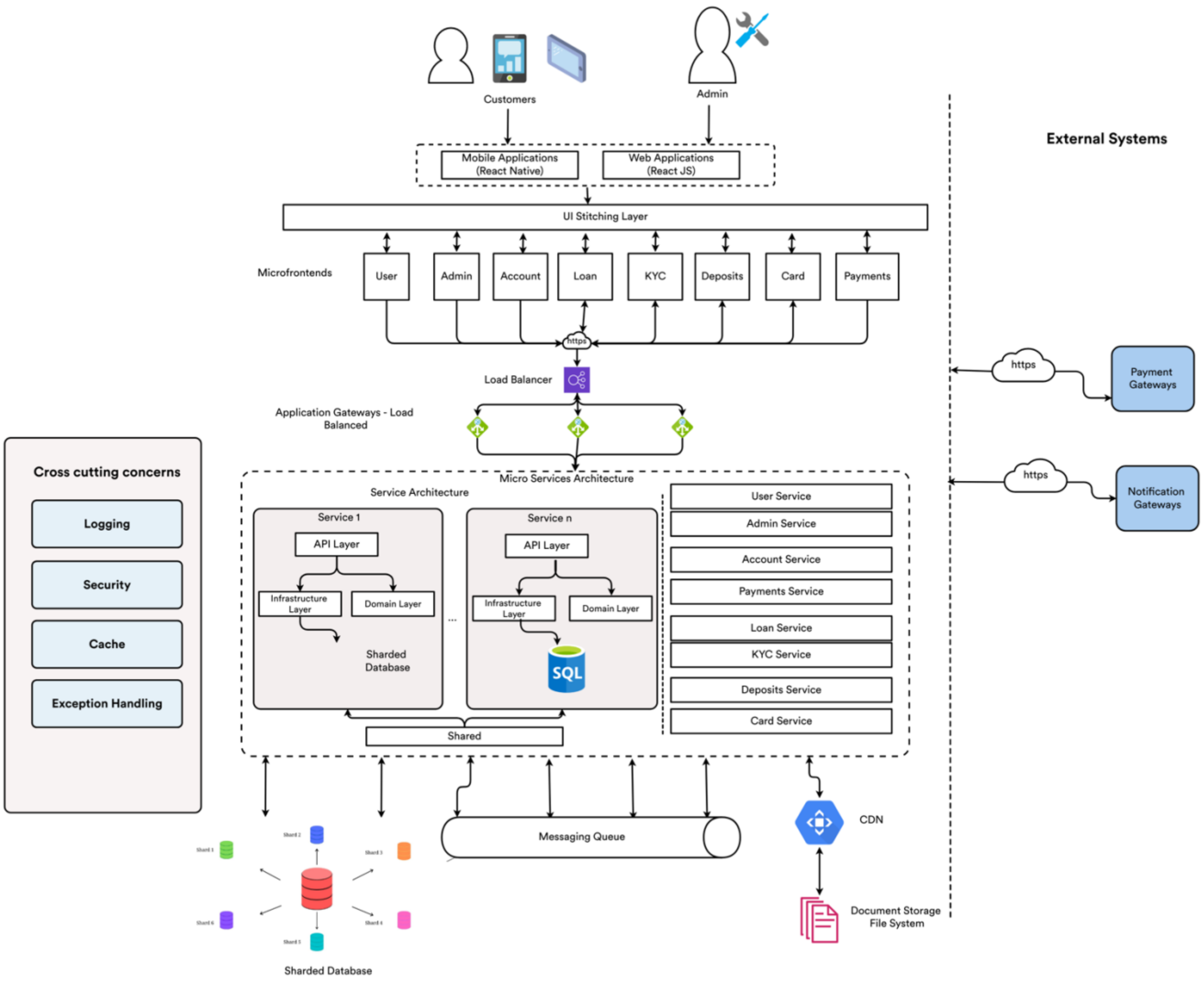
Table of Contents

Architecture Diagram



Architecture Diagram Components:

1. Micro-frontend: The frontend of the application will be rendered using micro-front end architecture. React Js and React Native will be used to support to mobile devices and web browsers.

2. Load Balancer: Load balancer is used for managing the load of the application. It will act as a reverse proxy and distribute traffic evenly across the available servers.

3. Application Gateway: API gateway will be used for connecting the available micro-services. This will act as an external endpoint and all the communication between client and the microservices will happen only through this end point.

4. Cross Cutting Concerns: We have extracted Logging, Security, Cache and Exception Handling as the cross-cutting concerns and these concerns will be common through various micro-services.

5. Messaging Queue: Messaging queue will be used for asynchronous internal and external communications.

6. CDN: We will be using CDN for serving various static data. This static data can be JS/CSS files or the documents that were uploaded to the file system.

7. File System: We will be using AWS S3 for the file storage. It will be storing all the documents of any type on cloud. We will also be storing some sensitive documents in the on premis storage.

8. Sharded Database: We will be using a sharded database sharded horizontally such that the required data can be fetched based on the key that we decide for sharding.

9. Payment Gateways: We will using an external payment gateway.

10. Notification Gateway: We will using an external notification gateway.

Services Identified:

1. User: User service is responsible for the following functionalities:

- Registering a user
- Logging a user
- Checking profile and updating details

2. KYC: KYC service is responsible for the following functionalities:

- E-KYC

3. Admin: Admin service is responsible for the following functionalities:

- Changing site options
- Adding new types of loans/settings
- Changing interest rates etc.

4. Accounts: Accounts service is responsible for the following functionalities:

- Opening a Savings Account
- Opening a Current account
- Check account balance
- Generate pdf statement
- Issue a cheque book
- Generate e-statement
- Email e-statement
- Stop cheque

5. Loan: Loan service is responsible for the following functionalities:

- Types of Loan(Home Loan, Personal Loan etc.)
- Apply for loan
- Handle supporting documents
- Apply for loan on credit card
- Track loan disbursement details
- Check loan account statement

6. Deposits: Deposit service is responsible for the following functionalities:

- Deposit Schemes and related operations
  - Open one/multiple fixed deposit accounts
  - Open one/multiple recurring deposit accounts
  - Select Tenure of deposits
  - Select monthly debit date for recurring deposits
  - Select mode of payment for fixed deposit/recurring deposit
  - Map each deposit account to savings account

7. Card: Card service is responsible for the following functionalities:

- Debit/Credit Card operations
  - Issue a debit card
  - Issue a credit card based on eligibility
  - Block accredit card
  - Increase/decrease credit limit

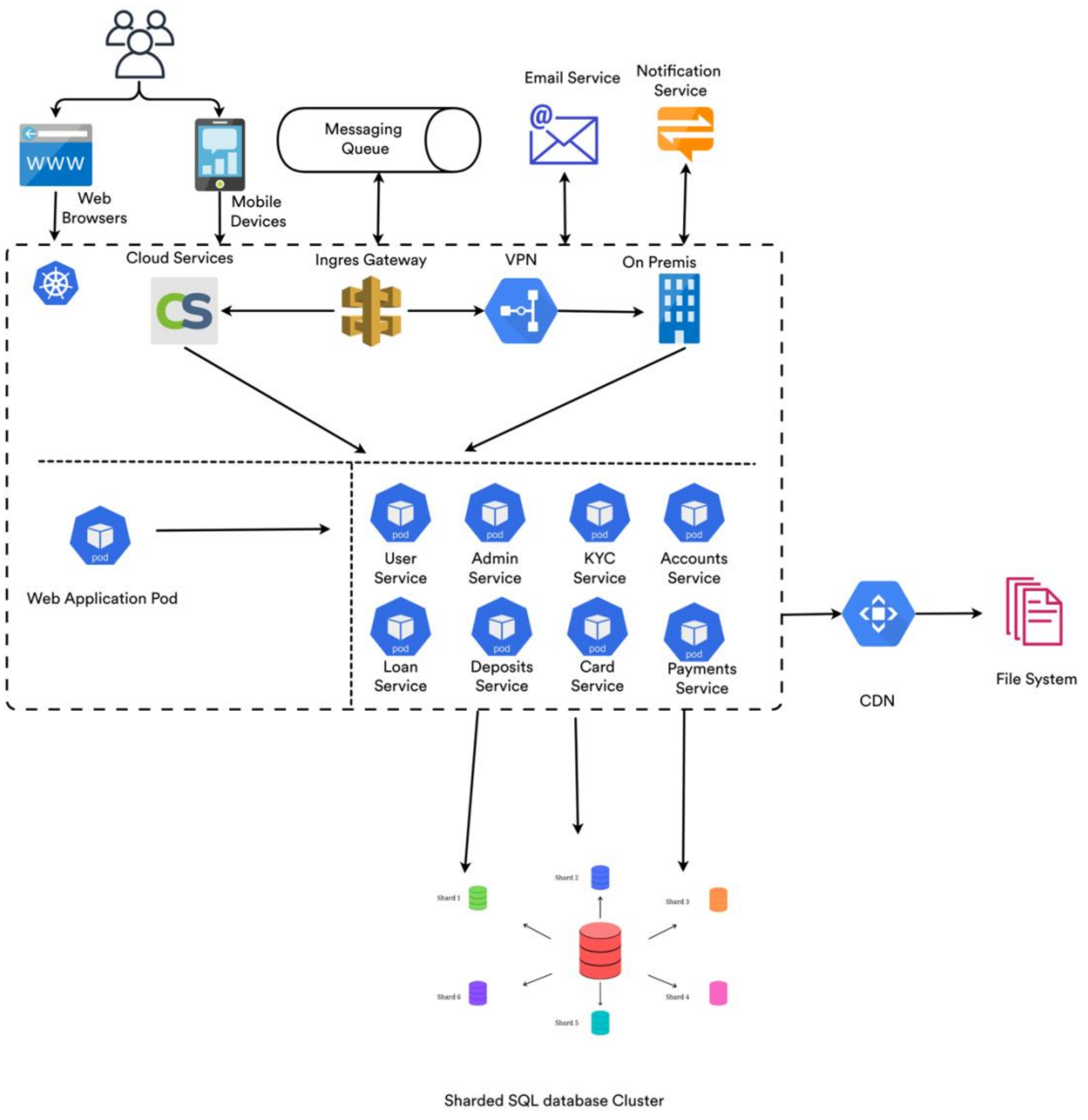
8. Payments: Payments service is responsible for the following functionalities:

- Bank Transfer through RTGS, IMPS, NEFT

Tech Stack for the application

1. Frontend: React Native and React JS with micro frontend:
2. Language: Java
3. Framework: Java Microservices with Spring boot
4. Database: Oracle
5. File Storage: Aws S3
6. Messaging Queue: Azure Queue
7. Code Repo : Azure/GIT
8. CI/CD : Azure pipelines
9. Code Review: Gerrit

Deployment Diagram



Deployment Diagram Components:

1. Ingres Gateway:
   We will be using Ingres Gateway as a load balancer which will be operating at the edge of the service Mesh. This gateway will be responsible for redirecting trafiic to Cloud servers and the On premis servers through VPN. Cloud Services. It is also responsible for balancing the loads and creating new pods/ destroying pods whenever required.

2. VPN: VPN will be used while the traffic is routed from ingres gateway to on premis servers.

3. Servers: We will have two types of servers i.e. On premis and cloud servers. All the data will be served from these servers.

4. Web Application Pods: Ingres gateway with Kubernetes will be responsible for creating new pods whenever required such that the application works as required and the performance of the application is not impacted during peak loads. It will lso be ramping down the pods when the traffic drops.