PAYMENT SYSTEM

CONTENTS

1. Problem Statement & Business Requirements

The purpose of this document is to employ use cases to frame what a realistic vision looks like for payments on the Web Platform. The end result is not meant to replace existing payment systems, but augment and simplify the interface to each system via the Web. The purpose of this initiative is to enable as many of the current payment schemes in use today (such as electronic cheques, credit cards, direct debit, and cryptocurrencies) to be used more easily and securely on the Web while ensuring that future payment schemes could be added with little effort.

Biller is an entity (vendor / service provider) to whom payments can be made from bank by existing account holders. Each Biller will be uniquely identified by Biller Code. Bank will maintain a master list of billers. (E.g Biller could be Mobile Service Provider (Airtel, Vi, Jio etc) or Electricity Companies etc

- If Account Holder intends to make payment from the account to the biller, he / she can register the biller by selecting biller from master list and consumer number. Consumer Number could be mobile number or electric meter number etc for which account holder intends to pay bill through the account. While registration account holder can choose whether to auto-pay the bill. Auto-pay feature would pay the bill automatically 3 days before the due date. Account Holder can set a limit on maximum auto-pay amount. If auto-pay is ON and bill amount is greater than auto-pay limit bill will not be paid automatically.
- Bank Manager will register a new bill by entering the following information Biller Code, Consumer Number, Bill Number, Bill Amount, and Due Date. Email Notification will be sent to the account holder from Biller + Consumer Number Combination. New bills will have the status of PENDING.

• Account Holder should be able to

- ✓ See a list of his / her own registered billers, and create a new one (See Above)
- ✓ See bills uploaded and scheduled payments
- ✓ Manually pay the bill (override auto-pay if present)
- ✓ See past payments done for all or selected biller
- ✓ Export into CSV all payments done within specified date range

• When bill payment is done account will be debited and email will be sent to account holder. Bill status will be set to PAID.

All payment transactions would be recorded in transaction table with unique transaction reference / id

Functionalities/Suggested classes:

This software will have following functionalities

- Online payment through debit card
- Customer will be able to get his/her payment status online while sitting at home by accessing the database of the bank using his/her password and account no. allotted him by the bank.
- Save or view up to 1 year past history of transaction
- It will be easy for the customer to view the updates about payment.
- Customer should get notifications for payment and after pay success status with transaction details should be notified.

Suggested Classes

• Role – Role ID , Name
• Accounts – Sequence ID, Account Number, Name, Email Address, Current Balance (Populate via backend SQL)
• Account Transaction (Sequence ID, Transaction Reference, Date Time, Amount, Debit/Credit, Description, Bill Reference Number)
• Users (Login ID, Password, Role, Linked Account Sequence ID (Mandatory for Account Holder Role) (Populate via backend SQL), Role ID
• Master Billers – Biller Code, Name (Populate via backend SQL)
• Registered Billers – Biller Sequence ID, Biller Code, Consumer Number, Account Number, Auto Pay (True/False), Auto Pay Limit (Default NULL)
• Bills – Bill Sequence ID, Biller Code, Consumer Number, Amount, Due Date, Status

User Characteristics:

There are various kinds of users for the App. Usually web apps are visited by various users for different reasons.

The users include:

- ✓ Chancellor who will be acting as the controller and he will have all the privileges of administrator.
- ✓ All the persons who need to perform banking.

Generals Constraints:

Some general constraints should be defined which will have a great part in the overall succession of the online banking project.

Hardware Requirements:

As this system is an online Web-based application so a client server will be the most suitable Organizational style for this system. Computer systems will be needed by each of the actor as well as that user must be connected to the internet.

Safety and Security:

This Project must be safe and secure because customers will directly contact their account through the internet. Software will have to identify the valid customer according to his/her bank details and password. So it is a difficult task to prevent the system by major disasters by preventing the unauthorized access to the system.

Assumptions and Dependencies:

Following are the assumptions and dependencies which are related to this **online payment app** project.

- **Connectivity**. Connectivity requirements vary according to use case and include Internet Connectivity and proximity connection over a technology such as NFC or Bluetooth. Some use cases assume no connectivity (e.g., user is temporarily out of cell phone connectivity).
- **Registered Payment Instruments**. For the user to select and apply payment instruments, they must be registered in some way and discoverable by a browser, native application, or other software.
- **Security**. Keys, encryption, and other security technology will be used to secure sensitive information.
- **Exceptions**. This document does not explicitly address various exceptions that might happen such as transactions failing. The architecture will need to take these into account (e.g., if a Payer applies a coupon and the transaction fails, the Payer's coupon would be restored).
- @@others?@@

Specific Requirements:

How the online banking will interact with the environment, what will be the functional and non-functional requirement. These all the steps should be defined here for providing a powerful base to the design phase. The design of the project will completely depend on the functional and non-functional requirements. So these should be defined clearly and accurately for the effectiveness.

Sample Data

• Suggested Master List of Billers with the bank

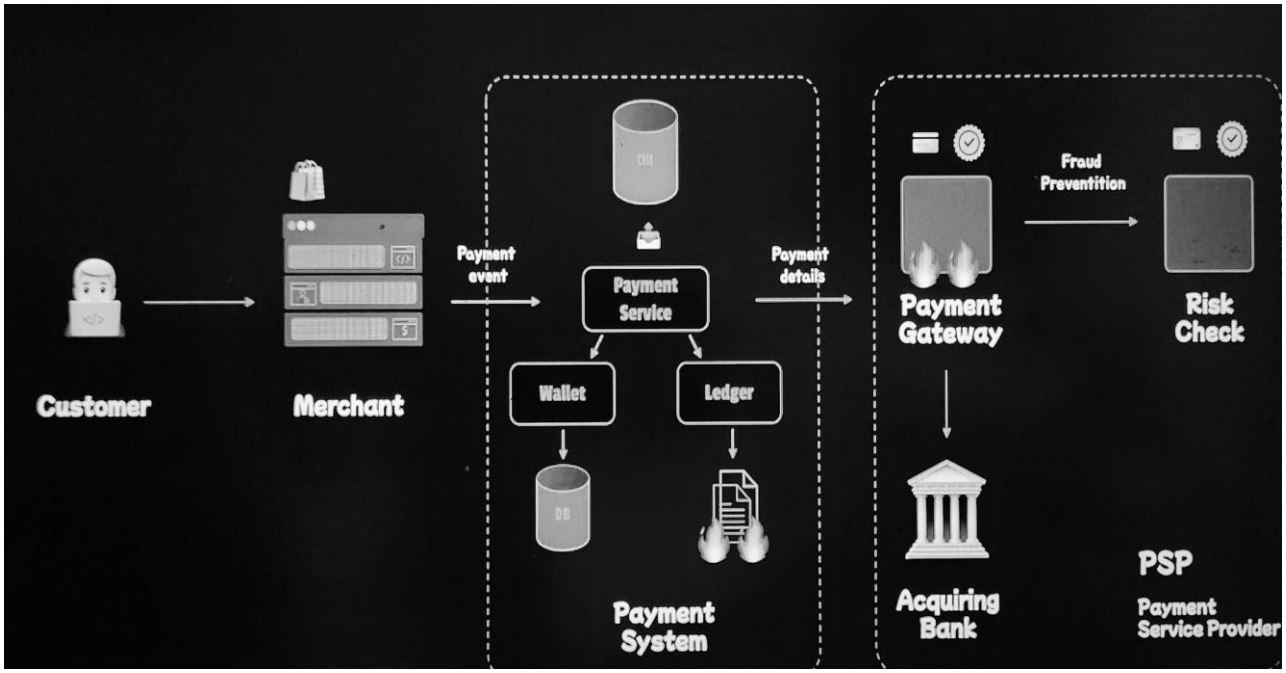
Biller Code	Name
B001	Airtel
B002	Jio
B003	Vi
B011	Tata Electric

Following are the services which this system will provide. These are the facilities and functions required by the customer.

2.PROPOSED ONLINE PAYMENT APP WIREFRAME

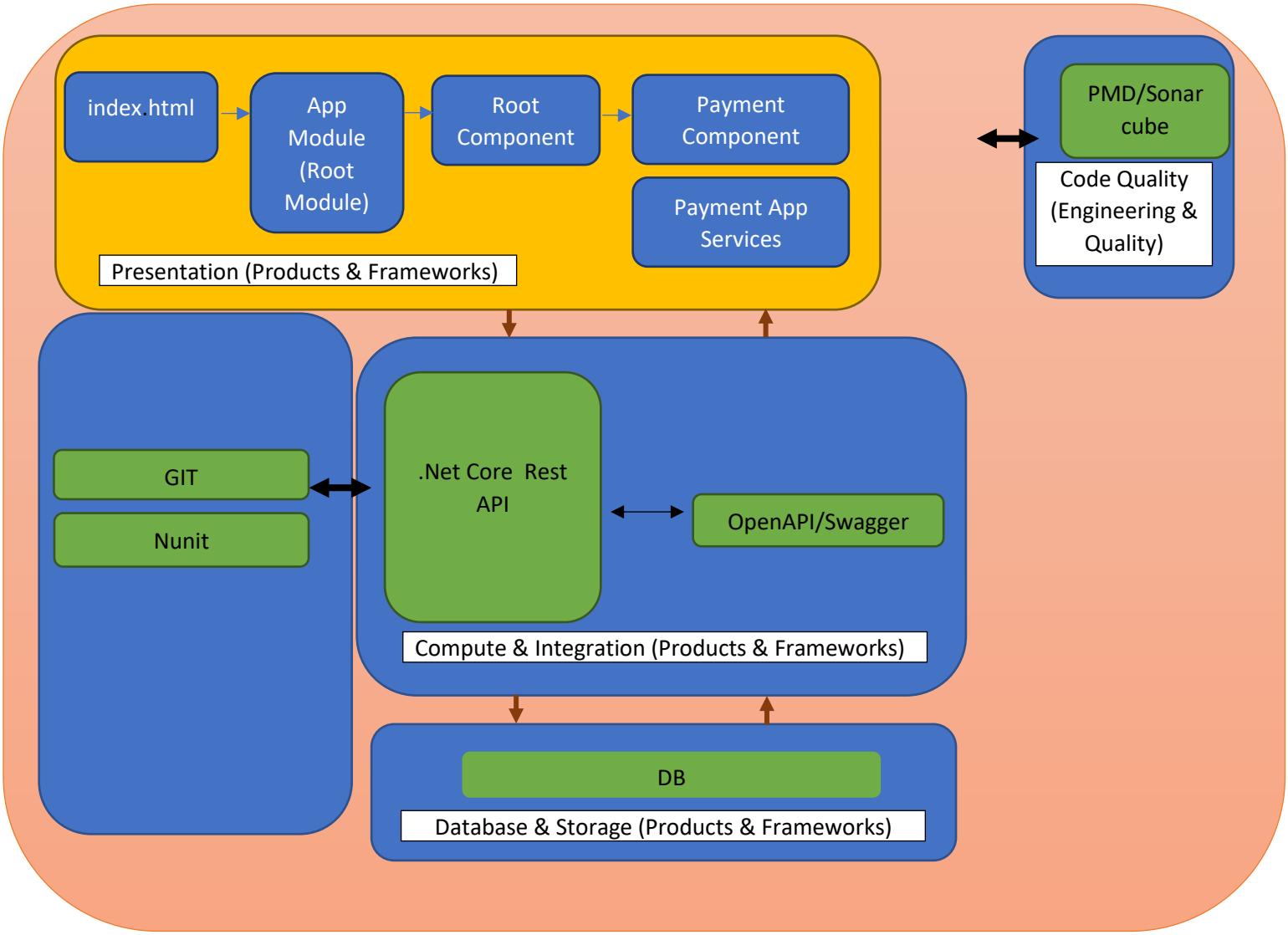
Biller is an entity (vendor / service provider) to whom payments can be made from bank by existing account holders. Each Biller will be uniquely identified by Biller Code. Bank will maintain a master list of billers. (E.g Biller could be Mobile Service Provider (Airtel, Vi, Jio etc) or Electricity Companies etc

UI needs improvisation and modification as per given use case.



- The above diagram is showing the entire use case for Online Payment System, where customer is paying the bill via Biller/Merchant from there Merchant is providing payment service either from Wallet or Ledger after verification of the customer payment will be done then the next phase payment details will take place where the information about sender, receiver, amount, date and time should be mentioned.
- Also, this application will take care of fraud prevention by checking the risk.
- Spring Test should be used for writing the test cases for each component.
- As its payment system so before connecting with Merchant or Biller ,verification and validation of the merchant should be done where before deducting the amount the registered mobile no should get the one time password(OTP) ,once the user is approving the amount by entering the OTP then only transaction should take place.
- If OTP entered is wrong payment will be discarded, or OTP is for limited time if it crosses time the payment should be declined.

3 APPLICATION ARCHITECTURE



4. TOOL CHAIN

Competency	Skill	Skill Detail
Engineering Mindset	Networking and Content Delivery	
	Ways of Working	
	Consulting Mindset	
Programming Languages	Application Language	C#
Products & Frameworks	Presentation	Angular/ React & Redux
		Karma & Jasmine
	Compute & Integration	EF Core
		Nunit Test
	Database & Storage	MongoDB
		Oracle SQL/PostgreSQL
	Governance & Tooling	Git,Nunit
Engineering Quality	Code Quality	Sonar Cube

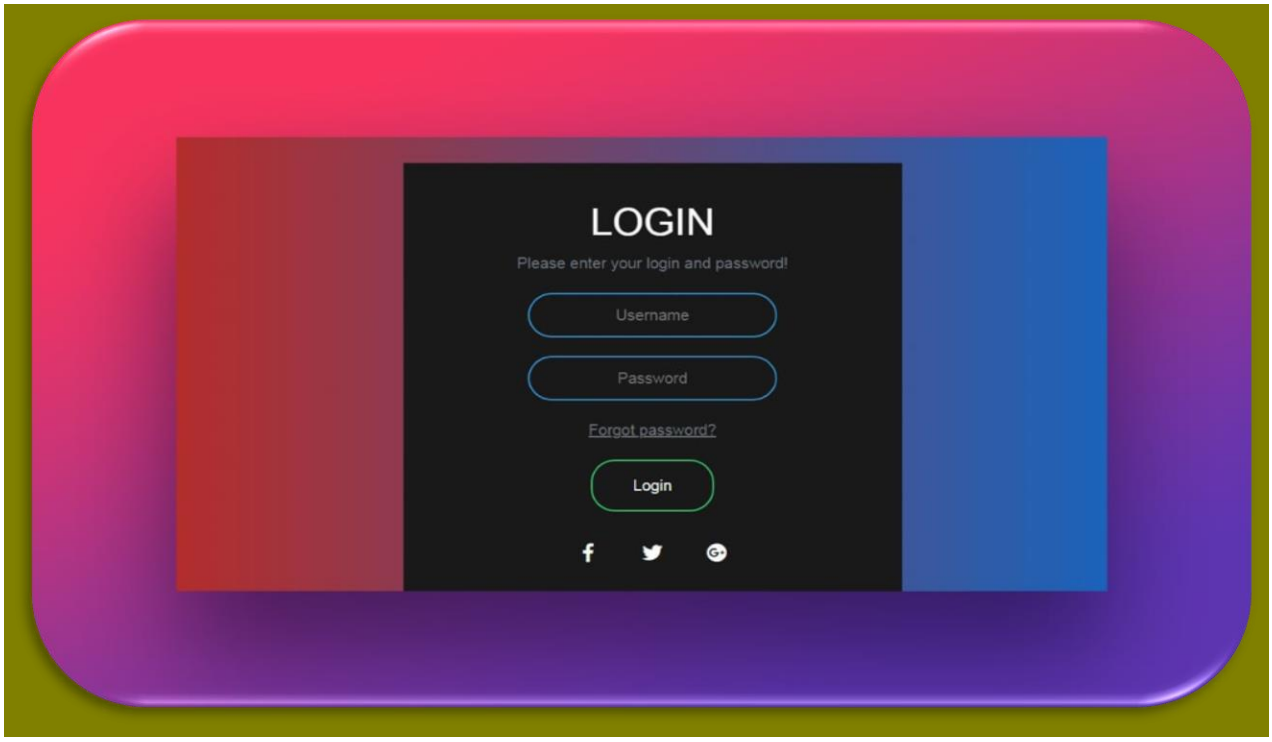
6.DEVELOPMENT FLOW

1	**Backend**	Rest API, Database, Messaging, Log/Monitoring, Testing	**Code Submission and Evaluation, Panel Presentation**
2	**Front End**	Angular/React	

7.PROCESS-REQUIREMENT:

The following are the discussions that describe how a user uses a system to accomplish a particular goal.

• Login :- For user login with credentials.



- Role Bank Manager
  - ✓ Create New Bill
- Role Account Holder
  - ✓ Delete / View Registered Billers.
  - ✓ Add New Biller
  - ✓ View Scheduled Bill Payments
  - ✓ Manual Payment
  - ✓ View Payments done for all or selected biller
  - ✓ Export Payments Data in CSV
- Role System – Batch Job
  - ✓ Execute Auto pay – run once a day
- Mail Provision

User interface:

Application will be accessed through a Browser Interface. The interface would be viewed best using 1024 x 768 and 800 x 600 pixels resolution setting. The software would be fully compatible with Microsoft Internet Explorer for version 6 and above.

No user would be able to access any part of the application without logging on to the system.

8. RUBRICS/EXPECTED DELIVERABLES

a. REST API (PRODUCTS & FRAMEWORKS -> COMPUTE & INTEGRATION):

a. Use Spring Boot to version and implement the REST endpoints.

b. Implement HTTP methods like GET, POST, PUT, DELETE, PATCH to implement RESTful resources:

POST	**/api/v1.0/payment/register**	**Register as a new customer**
GET	**/api/v1.0/ payment/login**	**Login**
GET	**/api/v1.0/payment/<username>/forgot**	**Forgot password**
GET	**/api/v1.0/ payment /domain/all**	**Get all payment details on a particular domain**
POST	**/api/v1.0/ payment /<username>/add**	**Post new payment details**
PUT	**/api/v1.0/payment /<username>/update/<id>**	**Update payment details**
DELETE	**/api/v1.0/ payment/<username>/delete/<id>**	**Delete the details**
PUT	**/api/v1.0/payment/<username>/modify/<id>**	**Modification in payment details**
POST	**/api/v1.0/payment /<username>/reply/<id>**	**Reply on payment information**

c. *username may be partial or complete username

d. Use necessary configuration in place for REST API in application.properties or bootstrap.properties or application.yml; whichever is applicable.

e. Package Structure for Spring Boot Project will be like com.onlinepayment.* with proper naming conventions for package and beans.

f. Use configuration class annotated with @Configuration and @Service for business layer.

g. Use constructor-based dependency injection in few classes and setter-based dependency injection in few classes.

h. Follow Spring Bean Naming Conventions

b. DATABASE (PRODUCTS & FRAMEWORKS -> DATABASE & STORAGE):

1. As an application developer:
   a. Implement ORM (EF CORE) with API Data with MS SQL Server or My sql.
   b. Have necessary configuration in place for REST API in application.properties or bootstrap.properties OR C# based configuration; whichever is applicable.

c. Testing

1. Perform proper testing using Nunit
2. The test coverage should be of 100%
3. Test Suites should cover both positive and exception handlings

9.Frontend

1. Develop the front end for all user stories.
2. Implement using either Angular or React
3. Implement all the Front-End validation rules
4. Proper naming conventions and folder structures
5. Implement using proper SOLID design principles
6. Perform unit and integration testing for the front end application

11.Methodology

d. Agile

1. As an application developer, use project management tool along to update progress as you start implementing solution.
2. As an application developer, the scope of discussion with mentor is limited to:
    a. Q/A
    b. New Ideas, New feature implementations and estimation.
    c. Any development related challenges
    d. Skill Gaps
    e. Any other pointers key to UI/UX and Middleware Development