LabD.co.in plan, Backend Architecture and ERD

This document consists of the full fledged plan for our upcoming business Labd.co.in.

What is LabD?

Labd is an online diagnostic platform where users can access all types of blood profiling while sitting at home. LabD is a platform where users can register and choose from various types of blood profiling and book a test. Where our executive will come at their place and collect the blood sample within 1 hour of booking confirmation via call and send the report via email and Whatsapp within 24 hours. From booking confirmation to report, users should be able to track the status as well.

Backend Blueprint

This section consists of the Backend Architecture for this project.

User Authentication and Registration

- We will use a user authentication library named as passport.js along with google Oauth 2.0.
- Implement registration functionality allowing users to sign up with an email and password also.
- We will use Bcrypt to hash and securely store the passwords.

Blood Profiling Services

- We will use firebase to store the service details and images as well.
- Implement a page where users can view available services fetched from the database.
- Since we are going to use firebase, we will use the cloud messaging feature provided by the firebase.

Booking Management

- We will use PostgreSQL to store the Booking ID, User ID, Test ID, Date, Time, Status.

- Implement a booking form where registered users can select a test and schedule a booking.
- Upon form submission, validate the input(Call API) and insert booking details into the database.

Confirmation and Executive Dispatch

- We will integrate a call API (Twilio or Nexmo, I will look for prices) to trigger a confirmation call upon booking.
- We can implement a mechanism to track the time elapsed since booking confirmation and dispatch an executive if it's within 1 hour by utilizing the “status” field from the database.
- Update the booking status in the database to reflect the current status.

Status Tracking

- A page where users can view the status of their bookings.
- Fetch booking details from the database and display them in a user-friendly format.
- We will implement real-time updates using Socket.IO JavaScript library to reflect status changes dynamically.

Sample Collection and Processing

- Upon simulating sample collection, update the booking status in the database.

Report Generation and Delivery

- Upon report generation, we will send it to the user's email using Nodemailer or Mailgun.
- We will integrate WhatsApp Business API to send reports via WhatsApp also.

Payment Integration

- Since our company is based in India, it is better to use either Phonepe or Paytm Payments Gateway.
- We will implement a secure checkout system as well.

Admin Dashboard

- We will have an admin interface that will be made using ReactJs.

- Allow admins to manage user accounts, bookings, services, and upload and send reports.

Security Measures

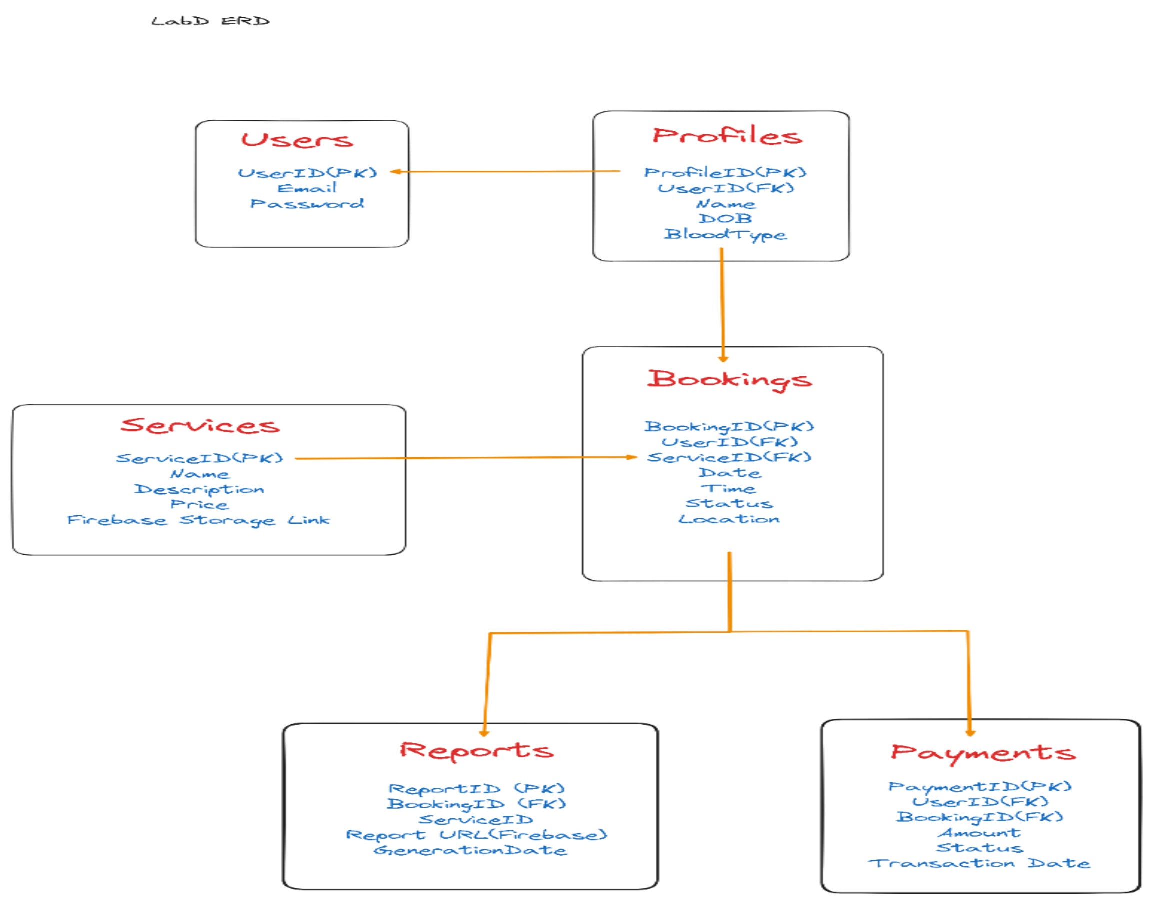
- We have to Implement CSRF protection, input validation, and output encoding to prevent common security vulnerabilities (XSS, SQL injection).
- We will use HTTPS to encrypt data transmitted between the client and server.

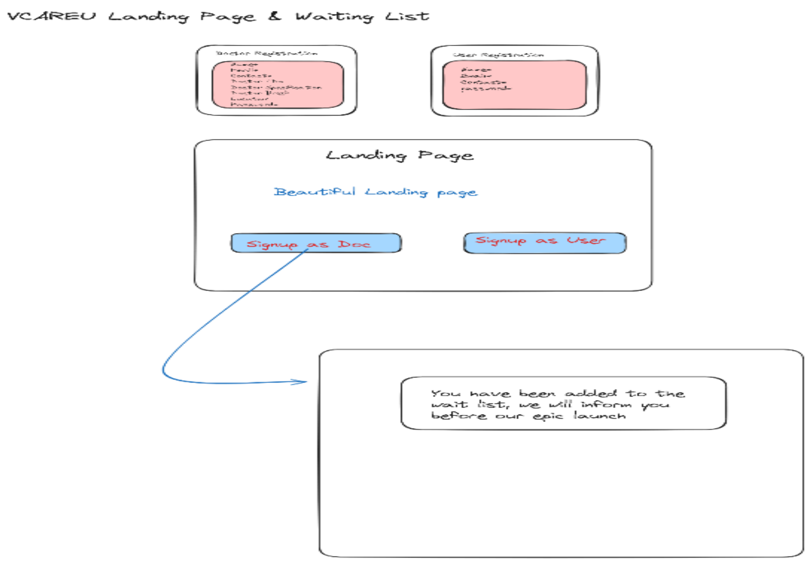
Scalability

We will design the application with scalability in mind by following best practices like modularization, caching, and database optimization.

- Consider deploying the application on AWS to easily scale resources as needed.

Our Entity Relationship Diagram




VCAREU Landing Page & Waiting List
Doctor Registration
User Registration
Landing Page
Beautiful Landing page
Signup as Doc
Signup as User
You have been added to the wait list, we will inform you before our epic launch

[Click Here for the Excalidraw File]

Database Relationship

Here is our complex database relationship that we will follow and systemize our database.

Users

- **One-to-Many with Profiles:** Each user can have multiple profiles.
- **One-to-Many with Bookings:** Each user can make multiple bookings.
- **One-to-Many with Payments:** Each user can make multiple payments.

Profiles

- **Many-to-One with Users:** Each profile belongs to one user.

Services

- **One-to-Many with Bookings:** Each service can be booked multiple times.

Bookings

- **Many-to-One with Users:** Each booking belongs to one user.
- **Many-to-One with Services:** Each booking is for one service.
- **One-to-O**ne with Reports: Each booking can have one report generated.
- **One-to-One with Payment Status:** Each booking can have one payment status.

Reports

- **One-to-One with Bookings:** Each report belongs to one booking.

Payment Status

- **One-to-One with Bookings**: Each payment status belongs to one booking.
- **Many-to-One with Users:** Each payment status belongs to one user.

Payments

- **Many-to-One with Users:** Each payment belongs to one user.