Juice With Ease

Team 6

Vrinda Singhal, Soumyadeep Paul, Larika Sehgal, Sanchit Trivedi

Project goals & Functionalities

- We aim to build **Juice With Ease**, which is a responsive web-app for buying fresh cold pressed juices and beverages online and getting it delivered at your doorstep. It offers one-time delivery and subscription plans.

- **Goals:**
  - The users would be able to explore product categories, see the product listing and add items to cart.
  - They can choose the timings for their delivery, allowing 6hrs hours between the time at which order is placed and the expected delivery time.
  - The website has single order purchase as well as subscription based purchase features.
  - They can then proceed to checkout and pay using multiple payment options.
  - The web-app will also have an Admin interface, which will allow the admin to track customer transactions and product details.

Analysing TechStack Options

SQL vs NoSQL

SQL	NoSQL
SQL is a language used by RDBMS to manipulate, retrieve and store data.	**NoSQL** as the name suggests is the opposite of SQL. NoSQL databases do not follow the RDBMS model.
It uses **Tables** to store data	It uses **Key/Value pairs**, Graphs, Document Object and Column Objects to store data.
The SQL paradigm is suitable for **structured Data**	The NoSQL paradigm is suitable for **unstructured data**
SQL databases require you to **define a schema**	NoSQL database offers **flexibility**, there is no standard schema
Vertically Scalable	**Horizontally** Scalable
Commonly used for **Ecommerce applications** and **Transactional data**	Commonly used for Document based data such as **social media posts, articles** etc.

Since our project is an ecommerce platform with structured data we shall use a **SQL** database

Django vs ExpressJS

Django is a high-level Python Web framework that allows you to build web apps and backends.

Django uses the **MVT (Model View Template)** design pattern

Django provides a much more **structured way** of doing things. For example Django uses its ORM for interfacing with the database.

Django provides **better, easier security** without bothering the developer

Django allows for **rapid development** and achieve a lot of functionality with minimal lines of code and has great community support.

ExpressJS is a web framework that provides you with a simple API, web apps and backends with JavaScript.

Express uses **event-driven programming** in which the entire architecture is driven by “events” or user choices.

ExpressJS allows for more **flexibility.** It is left to the developer to decide how to build an application. For ex. one can select from many libraries to connect to a DB.

To achieve a similar level of security with Express, it requires **experience and confidence** from the developer

Express gives the developer freedom and has great community support.

We chose Django because of speedy development and greater security.

Razorpay vs Stripe

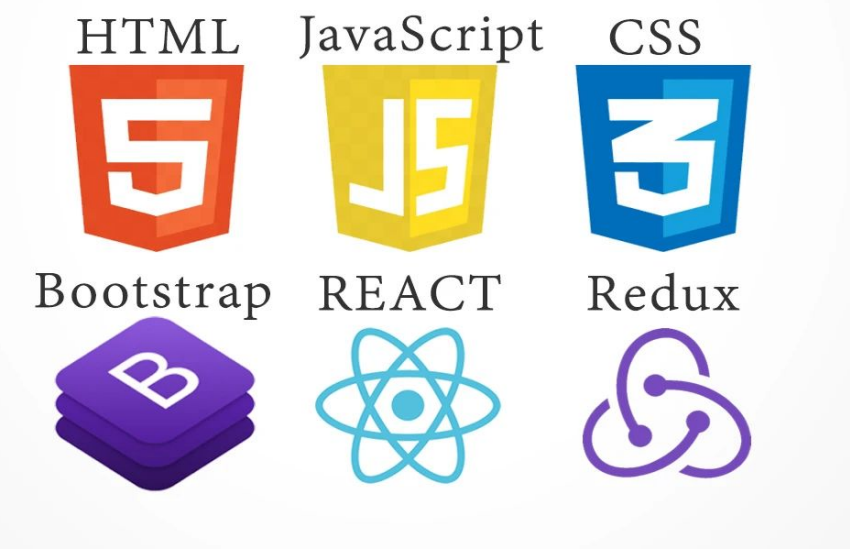
Razorpay	Stripe
Currently the **leading** payment gateway option in **India**.	Currently the **world leader** for integrating payment gateway in a website but unsuitable for Indian scenarios.
Mobile-based customer service is **available**.	**No mobile-based** customer service.
A **small community** around it as it is still developing and also **lack** of well documented beginner-friendly **documentation**.	A **huge community** around it and also **detailed documentation** for developers.
Includes **Debit Cards** and **UPI** services specific to India.	Has **no integration** for **UPI** services in India.
Includes **Indian based** digital wallets - Amazon Pay, Mobikwik, Airtel, JIO money, Freecharge, etc.	Includes **Global based** digital wallets - Apple pay, Alipay, Masterpass, Microsoft pay, etc.

We chose Razorpay as it is more suitable for India.

Final Technology Stack

Front-End

- **HTML** (Hypertext Markup Language) is the standard markup language for documents designed to be displayed in a web browser.
- HTML is styled by technologies **CSS** (Cascading Style Sheets) and scripting using, **JavaScript**.
- **React** is an open-source, front end, JavaScript library for building user interfaces or UI components.
- **Bootstrap** is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.
- **Redux** is an open-source JavaScript library for managing application state.



Back-End

- **Node.js** is a javascript runtime environment that allows for asynchronous event driven task execution designed to build scalable network applications
- **Npm** is a package manager for the JavaScript programming language and is the default package manager for the node.js runtime environment.
- **JavaScript** is a dynamically typed, interpreted programming language used both on the client-side and server-side that allows you to make web pages interactive.
- **Django** is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. It takes care of much of the hassle of Web development, without needing to reinvent the wheel. It's free and open source.

Database

- **PostgreSQL**, also known as Postgres, is an advanced free and open-source object-relational database management system emphasizing extensibility and SQL compliance.

Payment Gateway

- **Razorpay**, accept payments from customers.

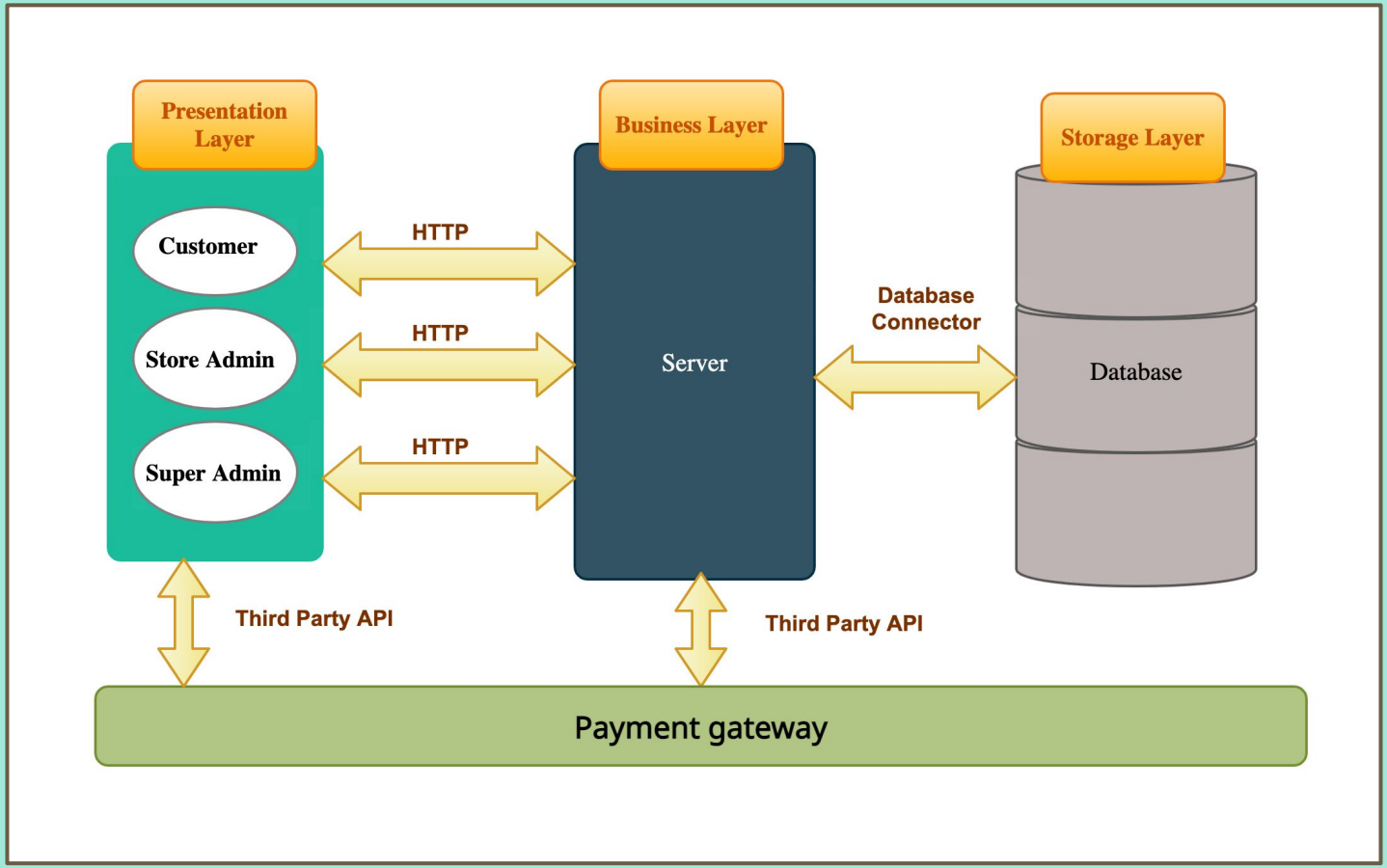
  Automate payouts to vendors & employees.

  Never run out of working capital.
- Suitable for use in India

Architecture

- Follows **3-Tier, Client Server Model**
- The presentation layer has **three different components**: Super Admin, Customer and Store Admin
- **Database** will have data for: Products, Catalogues, Customers, Orders etc. It will serve requests that come from the client web-app and return the result.
- A **third party service** would be used for processing payments.
- Google Account **Authentication** is used for sign-up/sign-in.

Initial Architecture



Architectural Attributes [1] :

- **Authentication** : Google Login
- **Security** : Backend, database and frontend guarantees prevention of unauthorized access and Cross site Scripting (XSS) attacks.
- **Availability**: 24hrs 365 days
- **Reliability**: As provided by the domain registrar and hosting companies
- **Performance**: Should be able to handle 1000 simultaneous connections

Architectural Attributes [2] :

- **Scalability**: Daily Active Users ~ 100
- **Configurability** : dedicated Admin and Super-Admin interfaces
- **Portability** : Frontend runs on any modern browser on any Operating system. Backend would be deployed on cloud services hence will be ubiquitous.
- **Privacy**: Database privacy is subjected to storage space of hosting company. Customer's personal data won't be accessible to admin or super admin except the order details and address. Passwords are also hashed before storing.

Connectors

- Database connector (server and database)
- APIs (frontend and server)
- Third-party APIs (payment gateway and server)
- Third-party APIs (payment gateway and frontend)

Why a 3-Tier Distributed Architecture?

Our main concerns are security, performance and scalability. Here's how the selected architecture serves our purpose:

- **Security**: A physically separate middle-tier application server can increase security because it adds an extra level of indirection between the web server and the database. This means no direct route from the web server to the database server. If your web server gets hacked, your application server is safe.

- **Performance & Scalability**: A 3-tier architecture allows much more scalability as compared to the normal client-server architecture. By separating out the different layers(presentation, business, data), so we can scale each independently depending on the need at any given time.