**Tech Stack Documentation for SaaS Application**

**Frontend**

**Next.js**

Next.js is a React-based framework that supports **Server-Side Rendering (SSR)** and **Static Site Generation (SSG)**. This ensures better **SEO performance** and faster **page load times**, which is crucial for an exam platform where students frequently search for “APTRANSCO mock tests” or “Banking practice questions.”

**Example in our SaaS:**

* Mock test listing pages are rendered with SSR so that Google indexes them properly.
* Students searching for “Banking Aptitude Mock Tests” find our platform quickly.

**Comparison:**

* **React alone**: Poor SEO, only client-side rendering.
* **Next.js**: SEO + pre-rendering + API routes.  
  👉 Chosen: **Next.js**.

**Tailwind CSS**

Tailwind is a **utility-first CSS framework** that allows us to build responsive UI quickly.

**Example in our SaaS:**

* Styling the **Performance Analytics Dashboard** for students where progress graphs resize perfectly on mobile and desktop.

**Comparison:**

* **Normal CSS**: More reusable but slower for large apps.
* **Bootstrap**: Fixed components, less customizable.
* **Tailwind**: Faster, consistent, modern.  
  👉 Chosen: **Tailwind CSS**.

**Redux Toolkit**

Redux is a **state management library** for managing global data across the frontend.

**Example in our SaaS:**

* A student starts a mock test → Redux stores answers in state.
* If the student refreshes, progress is still available until submission.

**Comparison:**

* **Context API**: Good for small apps, but not scalable.
* **Redux**: Better debugging, large-scale management.  
  👉 Chosen: **Redux Toolkit**.

**Chart.js**

Libraries to build **interactive charts and graphs**.

**Example in our SaaS:**

* Students see **bar charts of their accuracy in Quant vs Reasoning**.
* Line charts show **performance improvement over weeks**.

**Comparison:**

* **D3.js**: More powerful but very complex.
* **Chart.js**: Easy, lightweight, good enough for exam analytics.  
  👉 Chosen: **Chart.js**

**Backend**

**FastAPI**

FastAPI is a **high-performance Python framework** known for async support and ease of use with AI/ML.

**Example in our SaaS:**

* After a mock test submission, FastAPI calculates the score and generates **AI feedback** (weak areas, suggested practice questions).

**Comparison:**

* **Django**: Heavier, better for monoliths.
* **Flask**: Lightweight but less modern.
* **Falcon**: Fast but less community support.
* **FastAPI**: AI-friendly, async-ready, faster.  
  👉 Chosen: **FastAPI**.

**JWT & OAuth2**

* **JWT**: JSON Web Tokens ensure secure login sessions.
* **OAuth2**: Secure third-party authentication.

**Example in our SaaS:**

* JWT secures API requests for fetching student results.
* OAuth2 allows login with Google (optional).

**Redis**

Redis is an **in-memory data store** used for caching and leaderboards.

**Example in our SaaS:**

* Stores **real-time contest rankings** during state-level competitions.
* Caches frequently accessed questions to reduce DB load.

**Comparison:**

* **Memcached**: Only caching.
* **Redis**: Caching + pub/sub + leaderboard management.  
  👉 Chosen: **Redis**.

**Celery / AWS SNS**

For **background tasks and notifications**.

**Example in our SaaS:**

* A student finishes a mock test → Celery/SNS sends **email with results**.
* AI analysis runs in background without slowing UI.

**Comparison:**

* **Celery**: Good for self-hosted queues.
* **AWS SNS**: Cloud-native, scalable notifications.  
  👉 Chosen: **Celery**

**Database & Storage**

**PostgreSQL**

PostgreSQL is a **relational database** with strong support for complex queries.

**Example in our SaaS:**

* Stores **question banks, user details, subscriptions**.
* Handles complex queries like “Fetch all Banking aptitude questions from 2020–2025.”

**Comparison:**

* **MySQL**: Simpler but weaker JSON/AI support.
* **Postgres**: Rich features, better for scaling exam data.  
  👉 Chosen: **PostgreSQL**.

**FAISS**

FAISS is a **vector search engine** for AI similarity queries.

**Example in our SaaS:**

* Student struggles with “Time & Work” → FAISS finds **similar practice questions**.

**AWS S3**

Amazon S3 for **file storage**.

**Example in our SaaS:**

* Stores **strategy videos, daily PDF notes, and live recordings**.
* Provides secure, scalable access.

**Payments**

**Razorpay**

* Supports UPI, cards, net banking.
* Seamless integration with Indian banks.

**Example in our SaaS:**

* A student buys a **monthly subscription** using UPI.

**Paytm**

* Low-cost UPI transactions.

**Example in our SaaS:**

* Contest registration fees collected via Paytm.

**Comparison:**

* **Stripe**: Expensive in India.
* **Razorpay**: Cost-effective, local support.  
  👉 Chosen: **Razorpay**

**Infra & AI**

**AWS Infrastructure**

* EC2 (servers), RDS (databases), S3 (storage), EKS (Kubernetes), CloudWatch (monitoring).

**Example in our SaaS:**

* During exam season, EC2 auto-scales to handle 1M+ students.

**Docker & Kubernetes**

* **Docker**: Packages app into containers.
* **Kubernetes**: Orchestrates scaling.

**Example in our SaaS:**

* Runs separate **AI microservices** (doubt-solving, question generation).

**Hugging Face**

Provides **NLP models**.

**Example in our SaaS:**

* A student asks: “Explain probability shortcut” → Hugging Face model explains.

**LangChain**

Orchestrates multiple AI tools.

**Example in our SaaS:**

* Combines question bank + Hugging Face model to generate a **personalized practice set**.

**PyTorch**

Deep learning framework for AI models.

**Example in our SaaS:**

* Trains **adaptive learning model** → suggests easy/medium/hard questions based on performance.

## Monitoring

## 1. ****Prometheus + Grafana****

* **What it is:**  
  Prometheus is an open-source monitoring system that collects metrics (like CPU usage, memory, and response times). Grafana is a visualization tool that builds dashboards on top of Prometheus data.
* **Example in Our SaaS App:**  
  Imagine during peak exam hours, 10,000 students are attempting mock tests at the same time. Prometheus can track how quickly the backend API is returning results. Grafana can then show a dashboard with “Average Mock Test Submission Time” or “Active Students Right Now.”
* **Pros:**
  + Free and open-source.
  + Real-time metrics monitoring.
  + Highly customizable dashboards.
* **Cons:**
  + Doesn’t provide detailed error logs.
  + Needs extra setup for distributed systems.

## 2. ****ELK Stack (Elasticsearch, Logstash, Kibana)****

* **What it is:**  
  A log management and analytics system. Logstash collects logs, Elasticsearch stores them, and Kibana visualizes them.
* **Example in Our SaaS App:**  
  If a student clicks “Submit Test” and the system throws an error because the database timed out, ELK Stack will capture this log. Later, developers can search logs by student ID or timestamp to debug the issue.
* **Pros:**
  + Powerful log analysis.
  + Helps identify application errors.
  + Scales for large volumes of log data.
* **Cons:**
  + Resource-intensive.
  + More complex to set up and maintain.

## 3. ****Sentry****

* **What it is:**  
  A tool for error tracking and alerting. It notifies developers instantly when something breaks in the frontend (React/Next.js) or backend (FastAPI).
* **Example in Our SaaS App:**  
  Suppose the “Buy Subscription” payment API fails for some users. Sentry will immediately send an alert (Slack/Email) with the exact error trace so the dev team can fix it.
* **Pros:**
  + Easy to integrate with React + FastAPI.
  + Real-time error tracking.
  + Works for frontend and backend.
* **Cons:**
  + Free tier is limited.
  + Doesn’t provide detailed performance metrics like Prometheus.

## Final Tech Stack Summary

**Frontend:** Next.js, Tailwind CSS, Redux, Chart.js  
**Backend:** FastAPI, Redis, Celery/AWS SNS, JWT/OAuth2  
**Database & Storage:** PostgreSQL, FAISS, AWS S3  
**Payments:** Razorpay + Paytm  
**Infra:** AWS, Docker, Kubernetes  
**AI:** Hugging Face, LangChain, PyTorch

­­