NEETPrepGPT: From Architect's Blueprint to Launch 🚀

This document outlines the complete technical and strategic plan for building, launching, and scaling the NEETPrepGPT platform. It merges a deep, module-by-module technical learning path with critical project management principles to ensure we build the right product, the right way.

📋 Project Readiness & Execution Plan

This section provides a high-level overview of our strategic approach, priorities, and readiness for execution.

Quick-Look Readiness Checklist

Status	Item	Notes
✅	**Technical Scope Defined**	Modules 1–6 cover the full technology stack.
🔄	**Definition-of-Done Per Module**	Added below. Provides clear completion criteria.
🔄	**MVP & Sprint Schedule**	High-level MVP defined. Needs conversion to hour-based sprints.
📝	**Budget + Infra Estimate**	Added below. Initial estimates to be refined.
📝	**Legal / Data Ethics Checklist**	Added below. Critical for long-term viability.
📝	**Pilot Plan & KPIs**	Added below. Defines how we measure success with real users.

🎯 Prioritised MVP Backlog (Sprint 0)

Our goal is to ship a paid beta as quickly as possible. The minimal feature set required to deliver core value is:

Priority	Feature	Core Components Involved	Status
P0	**Telegram Bot Interface**	Module 6 (UI), Module 2 (API)	To Do
P0	**Core MCQ Generator**	Module 5 (AI), Module 3 (Data)	To Do
P0	**User Authentication & JWT**	Module 2 (Security)	To Do
P0	**Payment Gateway Integration**	Module 6 (Monetization)	To Do
P1	**Web Dashboard (Admin)**	Module 6 (UI)	To Do

🗓️ Concrete Timelines & Owners

The high-level modules must be converted into hour-based sprints. A rough initial allocation is as follows:

- **Module 1 & 2 (Backend Foundation):** 80 hours
- **Module 3 & 4 (Data & DSA):** 60 hours
- **Module 5 (AI Core):** 50 hours
- **Module 6 & MVP Launch Prep:** 70 hours

(Note: This is a high-level estimate. Detailed sprint planning is the next immediate step.)

Module 1: The Pythonic Foundation & Professional Tooling

🎯 **Objective:** To achieve a deep mastery of the Python language and establish the non-negotiable habits and tools of a professional software engineer.

✅ **Definition of Done:**

- All advanced OOP, Decorator, and Generator concepts implemented in a mini-project.
- Data analysis performed on a sample dataset using Pandas & NumPy.
- Asyncio implemented for at least two concurrent I/O operations.
- Project managed entirely via Git/GitHub with a minimum of 20 commits and a full PR

workflow.
- Pytest suite established with >80% unit test coverage for the mini-project.

1.1: Python Mastery (Beyond the Basics)

- **What to Learn:** Advanced OOP, Pythonic Constructs (Decorators, Generators, Context Managers), Modern Python (Type Hinting, Exception Handling).
- **Why It Matters (The Edge):** This is the difference between a simple script and a scalable, maintainable software application. Clean, object-oriented code is easier to debug, extend, and collaborate on.

1.2: The Data Science Stack

- **What to Learn:** NumPy, Pandas, Matplotlib & Seaborn.
- **Why It Matters (The Edge):** Your AI is only as good as the data it's trained on. This stack is the toolkit for understanding user behavior, analyzing MCQ effectiveness, and making data-driven business decisions.

1.3: Asynchronous Programming with asyncio

- **What to Learn:** async/await syntax, event loop concepts, concurrent I/O.
- **Why It Matters (The Edge):** An async server can handle thousands of concurrent users, drastically improving scalability. This is essential for a high-performance backend.

1.4: Professional Habits & Tooling

- **What to Learn:** Git & GitHub workflows, pytest for comprehensive testing, venv for environment management.
- **Why It Matters (The Edge):** These habits define professionalism. Git is non-negotiable for source control, and automated testing is your safety net.

Module 2: The Production-Grade API Backend

🎯 **Objective:** To build the fast, secure, and reliable API that will serve as the central nervous system for your entire platform.

✅ **Definition of Done:**

- A minimum of 7 CRUD endpoints built with FastAPI.
- JWT/OAuth2 authentication fully implemented and protecting all sensitive endpoints.
- Pydantic models enforcing strict validation for all request and response bodies.
- SQLAlchemy ORM connected to a PostgreSQL database.
- Alembic configured for managing database migrations.

2.1: The FastAPI Framework

- **What to Learn:** Path/query parameters, request bodies, Dependency Injection.
- **Why It Matters (The Edge):** FastAPI provides incredible performance and auto-generated interactive API documentation (Swagger UI), accelerating development

and testing.

2.2: Data Validation with Pydantic

- **What to Learn:** Define strict data schemas for all data flows.
- **Why It Matters (The Edge):** Pydantic is your API's first line of defense against malformed or malicious data, eliminating a huge source of bugs and vulnerabilities.

2.3: Database Interaction with SQLAlchemy & Alembic

- **What to Learn:** Use the SQLAlchemy ORM and manage schema changes with Alembic migrations.
- **Why It Matters (The Edge):** The ORM massively speeds up development. Alembic provides a safe, version-controlled way to evolve your database structure.

2.4: Security Deep Dive

- **What to Learn:** Password hashing (bcrypt), token-based authentication (JWT/OAuth2), rate limiting.
- **Why It Matters (The Edge):** In EdTech, user trust is your most valuable asset. A security breach is an extinction-level event.

Module 3: The Data Ingestion & Caching System

🎯 **Objective:** To build a robust pipeline for acquiring your core data and a high-speed caching layer to serve it instantly.

✅ Definition of Done:

- Web scraper built with Requests/BeautifulSoup and Selenium that successfully extracts data from 3 distinct sources.
- PostgreSQL database schema designed and implemented.
- Redis cache implemented for at least 3 frequent database queries, showing a measurable performance improvement.

⚖️ Data Governance + Scraping Checklist

- **Sources:** Explicitly list target websites and forums.
- **Copyright/robots.txt:** Automated check for robots.txt on all target domains before scraping. Log the checks.
- **Rate Limiting:** Implement respectful delays and user-agent rotation to avoid overwhelming target servers.
- **Human Validation:** A plan for a human to review a sample of scraped content for accuracy and relevance before it enters the Vector DB.

3.1: Data Acquisition (Web Scraping)

- **What to Learn:** Requests, BeautifulSoup, and Selenium. Ethical scraping practices.
- **Why It Matters (The Edge):** The quality and uniqueness of your data is your primary

competitive advantage.

3.2: The Relational Database (PostgreSQL)

- **What to Learn:** Clean, normalized schema design and advanced SQL.
- **Why It Matters (The Edge):** Deep SQL knowledge is a developer superpower, allowing for efficient data analysis at the database level.

3.3: High-Speed Caching with Redis

- **What to Learn:** Store frequent query results in memory; implement cache invalidation.
- **Why It Matters (The Edge):** Caching is a primary tool for performance. Serving data from memory (Redis) is orders of magnitude faster than disk (database).

Module 4: Performance Bootcamp: The DSA Core

🎯 **Objective:** To rewire your brain to think algorithmically, enabling you to write maximally efficient code.

✅ **Definition of Done:**

- Implement all core data structures (Hash Maps, Trees, Graphs, etc.) from scratch.
- Solve 20+ problems on a platform like LeetCode covering sorting, searching, and graph traversal.
- Complete all three mini-projects (Autocomplete, Recommendations, Leaderboard).

4.1: Algorithmic Complexity

- **What to Learn:** Master Big O Notation for time and space complexity analysis.
- **Why It Matters (The Edge):** The universal language for measuring code performance, ensuring your app remains fast as it scales.

4.2: Core Data Structures

- **What to Learn:** Arrays, Linked Lists, Stacks, Queues, Hash Maps, Trees, Heaps, and Graphs.
- **Why It Matters (The Edge):** Choosing the right data structure is the difference between an app that runs in milliseconds and one that takes minutes.

4.3: Fundamental Algorithms

- **What to Learn:** Sorting, Searching, Graph Traversal (BFS, DFS), Recursion, Dynamic Programming.
- **Why It Matters (The Edge):** These are the fundamental blueprints for solving any complex computational problem.

4.4: Applied DSA (Mini-Project Week)

- **Projects:** Autocomplete (Trie), Recommendations (Graph Traversal), Leaderboard (Heap).
- **Why It Matters (The Edge):** This crucial step makes abstract knowledge concrete,

permanent, and directly valuable to your product.

Module 5: The AI Intelligence Layer

🎯 **Objective:** To integrate Large Language Models to build the core intelligent features that define NEETPrepGPT.

✅ **Definition of Done:**

- OpenAI API integrated with secure key management and error handling.
- RAG pipeline fully functional: data is vectorized, stored in ChromaDB, and retrieved as context for prompts.
- A benchmark test suite created with a "golden set" of 50 questions to measure MCQ accuracy, with a target score of >90%.

💰 Cost & Infra Estimate

- **Cloud Services (AWS/GCP):**
  - Compute (EC2/App Runner): ~$30-50/month (starter)
  - Database (RDS for PostgreSQL): ~$20-40/month
  - Cache (ElastiCache for Redis): ~$15-30/month
- **Vector DB (e.g., Pinecone, ChromaDB hosted):** ~$50-70/month (starter tier)
- **OpenAI API:** Variable, budget for ~$100/month initially. Monitor usage closely.
- **Safety Buffer (20%):** ~$40/month
- **Total Estimated Monthly Cost (Initial): ~$250 - $330**

5.1: LLM API Integration (OpenAI)

- **What to Learn:** Securely manage API keys, handle rate limits/errors, understand the cost model.
- **Why It Matters (The Edge):** Doing this securely and cost-effectively is critical for a sustainable business.

5.2: Advanced Prompt Engineering

- **What to Learn:** Few-Shot and Chain-of-Thought prompting to guide the LLM's reasoning for high-quality MCQ generation.
- **Why It Matters (The Edge):** The quality of your prompts is the single biggest determinant of your AI's output. This is your "secret sauce."

5.3: Retrieval-Augmented Generation (RAG)

- **What to Learn:** Convert data to vector embeddings, store in a Vector DB, and build the full RAG pipeline.
- **Why It Matters (The Edge):** RAG is state-of-the-art for making a generic AI an expert in a specific domain, dramatically reducing errors.

5.4: AI Evaluation & Benchmarking

- **What to Learn:** Use frameworks to evaluate generated text, create a "golden set" for

benchmarking, and A/B test prompts.
- **Why It Matters (The Edge):** This transforms your AI from a subjective "magic box" into a scientific instrument you can objectively measure and improve.

Module 6: Interfaces, Deployment & Launch

🎯 **Objective:** To deliver your finished product to real users, handle monetization, and ensure it runs reliably and securely in the cloud.

✅ **Definition of Done:**

- Telegram bot is live and functional for end-to-end MCQ generation and interaction.
- Payment gateway (Razorpay/Stripe) is integrated and has successfully processed a test transaction.
- The entire application is containerized with Docker.
- A CI/CD pipeline is established with GitHub Actions that automatically runs tests and deploys the main branch to the cloud.

🧑‍🔬 User Validation Plan (Pilot Stage)

- **Target Group:** A closed group of 20-30 real NEET students.
- **Methodology:** Provide free access for 4 weeks in exchange for structured feedback (weekly surveys, one-on-one interviews).
- **Key Metrics (KPIs):**
  - **User Retention:** % of students active in Week 4.
  - **Engagement:** Average number of MCQs generated per session.
  - **Accuracy:** User-reported accuracy rate of generated MCQs.
  - **Net Promoter Score (NPS):** "How likely are you to recommend this to a friend?"

6.1: UX & Human Factors in EdTech

- **What to Learn:** Spaced Repetition, Gamification, Adaptive Learning.
- **Why It Matters (The Edge):** The best tech will fail with a poor user experience. A product designed with these principles will be more effective and engaging, driving user retention.

6.2: User Interfaces

- **What to Learn:** Build a conversational interface with python-telegram-bot; create a simple web dashboard with HTML/CSS/JS.
- **Why It Matters (The Edge):** This is how users access the value you've created. Being able to build simple, effective interfaces makes you a full-stack builder.

6.3: Monetization

- **What to Learn:** Integrate a payment gateway like Razorpay or Stripe.
- **Why It Matters (The Edge):** This is the step that turns your project into a business.

6.4: DevOps & Cloud Deployment

- **What to Learn:** Docker for containerization, CI/CD with GitHub Actions, Cloud deployment (AWS/GCP).
- **Why It Matters (The Edge):** This is how modern software is shipped. Automating deployment is safer, faster, and the professional standard.