**NUCAP - University Admission Platform MVP**

**🎯 Executive Summary**

**Product Name:** NUCAP (National University Admission Platform)  
**Target Users:** 1M+ Intermediate (12th grade) students in Pakistan  
**Core Value:** Prevent students from missing admission deadlines and help them find best-fit universities based on their academic profile

**📋 MVP Scope Definition**

**Phase 1: Core Features (4-6 weeks)**

**Must-Have Features:**

1. ✅ Student profile creation with academic credentials
2. ✅ Merit calculator and university matching
3. ✅ Automated data scraping from 3-4 universities
4. ✅ Admission deadline tracking
5. ✅ Basic admin panel for manual updates
6. ✅ User authentication

**Universities for MVP:**

* NUST (National University of Sciences & Technology)
* FAST (Foundation for Advancement of Science & Technology)
* COMSATS University
* One additional public university (e.g., PUNJAB University/PIEAS)

**🏗️ Technical Architecture**

text

┌─────────────────────────────────────────────────────┐

│ FRONTEND (Next.js) │

│ - Student Dashboard - University Explorer │

│ - Merit Calculator - Deadline Tracker │

└──────────────────┬──────────────────────────────────┘

│

↓

┌─────────────────────────────────────────────────────┐

│ BACKEND API (Next.js API Routes) │

│ - Authentication (Clerk) │

│ - Merit Calculation Logic │

│ - University Matching Algorithm │

└──────────────────┬──────────────────────────────────┘

│

┌──────────┴──────────┐

↓ ↓

┌───────────────┐ ┌────────────────────┐

│ DATABASE │ │ WEB SCRAPING │

│ (PostgreSQL/ │ │ SERVICE │

│ MongoDB) │ │ - Jina AI API │

│ │ │ - Cron Jobs │

└───────────────┘ └────────────────────┘

**💾 Database Schema**

TypeScript

// User Collection/Table

User {

id: string (PK)

clerkId: string (unique)

email: string

name: string

phoneNumber: string

createdAt: datetime

lastLogin: datetime

}

// Student Profile

StudentProfile {

id: string (PK)

userId: string (FK)

matricMarks: {

totalMarks: number

obtainedMarks: number

percentage: number

board: string

year: number

}

interMarks: {

totalMarks: number

obtainedMarks: number

percentage: number

board: string

group: string (Pre-Med/Pre-Eng/Commerce/Arts)

year: number

}

testScores: {

nustTest: number (optional, estimated)

fastTest: number (optional, estimated)

ntsTest: number (optional, estimated)

}

preferences: {

preferredCities: string[]

preferredFields: string[]

budgetRange: string

}

createdAt: datetime

updatedAt: datetime

}

// University

University {

id: string (PK)

name: string

shortName: string (NUST/FAST/COMSATS)

location: string

website: string

contactInfo: object

testRequired: string (NUST/FAST/NTS)

isActive: boolean

lastScraped: datetime

createdAt: datetime

}

// Department

Department {

id: string (PK)

universityId: string (FK)

name: string

degree: string (BS/BE/BBA etc.)

duration: string

seats: number

category: string (Engineering/CS/Medical/Business)

}

// Merit List

MeritList {

id: string (PK)

universityId: string (FK)

departmentId: string (FK)

year: number

admissionCycle: string (Fall/Spring)

meritType: string (1st/2nd/3rd/Final)

lastMerit: {

aggregatePercentage: number

matricPercentage: number

interPercentage: number

testScore: number

}

closingMerit: number

publishedDate: datetime

createdAt: datetime

}

// Admission Timeline

AdmissionTimeline {

id: string (PK)

universityId: string (FK)

year: number

cycle: string (Fall 2024/Spring 2025)

applicationStart: datetime

applicationDeadline: datetime

testDate: datetime

firstMeritList: datetime

secondMeritList: datetime

thirdMeritList: datetime

finalMeritList: datetime

isActive: boolean

scrapedData: json

lastUpdated: datetime

updatedBy: string (system/admin)

}

// University Updates (News/Announcements)

UniversityUpdate {

id: string (PK)

universityId: string (FK)

title: string

description: text

updateType: string (deadline/merit/general)

priority: string (high/medium/low)

sourceUrl: string

publishedDate: datetime

isManual: boolean

createdBy: string

createdAt: datetime

}

// Student Matches (Recommendations)

StudentMatch {

id: string (PK)

studentProfileId: string (FK)

universityId: string (FK)

departmentId: string (FK)

matchScore: number (0-100)

estimatedMerit: number

requiredMerit: number

admissionChance: string (High/Medium/Low)

meritGap: number

calculatedAt: datetime

}

// Scraping Jobs Log

ScrapingLog {

id: string (PK)

universityId: string (FK)

status: string (success/failed)

dataType: string (merit/deadline/announcement)

recordsUpdated: number

errorMessage: text

executionTime: number

scheduledAt: datetime

completedAt: datetime

}

**🔌 API Structure**

**Student APIs**

TypeScript

// Authentication (Clerk handles these)

POST /api/auth/signup

POST /api/auth/login

POST /api/auth/logout

// Student Profile

POST /api/student/profile // Create profile

GET /api/student/profile // Get profile

PUT /api/student/profile // Update profile

DELETE /api/student/profile // Delete profile

// Merit Calculation

POST /api/student/calculate-merit

// Body: { matricMarks, interMarks, testScores }

// Response: { estimatedMerit, matchedUniversities[] }

// University Matching

GET /api/student/matches

// Query: ?limit=10&sortBy=matchScore

// Response: { matches: StudentMatch[] }

// University Exploration

GET /api/universities // All universities

GET /api/universities/:id // Single university

GET /api/universities/:id/departments

GET /api/universities/:id/merit-lists

GET /api/universities/:id/timeline

// Deadline Tracking

GET /api/deadlines/upcoming

// Query: ?daysAhead=30

// Response: { deadlines: AdmissionTimeline[] }

// Updates/News

GET /api/updates

// Query: ?universityId=xxx&type=deadline&limit=20

**Admin APIs**

TypeScript

// Manual Data Entry

POST /api/admin/universities/:id/merit

POST /api/admin/universities/:id/timeline

POST /api/admin/universities/:id/update

PUT /api/admin/merit/:id

DELETE /api/admin/merit/:id

// Scraping Management

POST /api/admin/scraping/trigger // Manual trigger

GET /api/admin/scraping/logs // View logs

GET /api/admin/scraping/status // Current status

// Dashboard Stats

GET /api/admin/stats

// Response: { totalStudents, activeUniversities, todayMatches }

**🤖 Web Scraping Implementation**

**Scraping Service Architecture**

TypeScript

// /services/scraper/index.ts

import axios from 'axios';

interface ScraperConfig {

universityId: string;

universityName: string;

websiteUrl: string;

selectors: {

admissionPage: string;

meritPage: string;

announcementPage: string;

};

}

class UniversityScraper {

private jinaApiKey: string;

constructor() {

this.jinaApiKey = process.env.JINA\_API\_KEY!;

}

async scrapeUniversity(config: ScraperConfig) {

try {

// Use Jina AI Reader API

const admissionData = await this.scrapeWithJina(

`${config.websiteUrl}/admissions`

);

const meritData = await this.scrapeWithJina(

`${config.websiteUrl}/merit-lists`

);

// Parse and structure data

const structured = await this.parseData(admissionData, meritData);

// Save to database

await this.saveToDatabase(config.universityId, structured);

return { success: true, data: structured };

} catch (error) {

await this.logError(config.universityId, error);

return { success: false, error };

}

}

private async scrapeWithJina(url: string) {

// Jina AI Reader API for web scraping

const response = await axios.get(

`https://r.jina.ai/${url}`,

{

headers: {

'Authorization': `Bearer ${this.jinaApiKey}`,

'X-Return-Format': 'markdown'

}

}

);

return response.data;

}

private async parseData(admissionHtml: string, meritHtml: string) {

// Use GPT/Claude or regex to extract:

// - Admission dates

// - Deadline dates

// - Merit percentages

// - Department names

// You can also use Jina AI Search API

const searchResults = await this.searchWithJina([

"admission deadline 2024",

"merit list closing merit",

"test date schedule"

]);

return {

deadlines: this.extractDeadlines(admissionHtml),

meritLists: this.extractMerit(meritHtml),

updates: this.extractAnnouncements(admissionHtml)

};

}

private async searchWithJina(queries: string[]) {

const response = await axios.post(

'https://api.jina.ai/v1/search',

{

queries: queries,

top\_k: 5

},

{

headers: {

'Authorization': `Bearer ${this.jinaApiKey}`,

'Content-Type': 'application/json'

}

}

);

return response.data;

}

}

export default UniversityScraper;

**Cron Job Configuration**

TypeScript

// /app/api/cron/scrape-universities/route.ts

import { NextResponse } from 'next/server';

import UniversityScraper from '@/services/scraper';

// This endpoint will be called by Vercel Cron or similar

export async function GET(request: Request) {

// Verify cron secret

const authHeader = request.headers.get('authorization');

if (authHeader !== `Bearer ${process.env.CRON\_SECRET}`) {

return NextResponse.json({ error: 'Unauthorized' }, { status: 401 });

}

const scraper = new UniversityScraper();

const universities = await getActiveUniversities();

const results = await Promise.all(

universities.map(uni => scraper.scrapeUniversity(uni))

);

return NextResponse.json({

success: true,

timestamp: new Date(),

results

});

}

**Vercel Cron Config (vercel.json):**

JSON

{

"crons": [

{

"path": "/api/cron/scrape-universities",

"schedule": "0 \*/12 \* \* \*"

}

]

}

**🎨 Frontend Implementation**

**Key Pages & Components**

TypeScript

// /app/page.tsx - Landing Page

- Hero section

- Problem statement

- Features showcase

- CTA: "Find Your University"

// /app/dashboard/page.tsx - Student Dashboard

- Profile completion status

- Upcoming deadlines (countdown)

- Matched universities (top 5)

- Recent updates

// /app/profile/page.tsx - Student Profile Form

- Step 1: Personal Info

- Step 2: Matric Marks

- Step 3: Inter Marks

- Step 4: Expected Test Scores

- Step 5: Preferences

// /app/universities/page.tsx - University Explorer

- Filter by: City, Test Type, Field

- Sort by: Deadline, Merit, Match Score

- University cards with key info

// /app/universities/[id]/page.tsx - University Detail

- Departments & Merit Lists

- Admission Timeline

- Merit Calculator for this university

- Latest Updates

// /app/matches/page.tsx - Your Matches

- List of departments with admission chances

- High/Medium/Low probability indicators

- Merit gap analysis

- Action items (Apply/Prepare/Backup)

// /app/admin/page.tsx - Admin Dashboard

- Scraping logs

- Manual data entry forms

- User statistics

- University management

**Merit Calculation Algorithm**

TypeScript

// /lib/meritCalculator.ts

interface MeritInput {

matricPercentage: number;

interPercentage: number;

testScore?: number;

universityId: string;

}

interface MeritCriteria {

matricWeight: number;

interWeight: number;

testWeight: number;

}

// Different universities have different merit formulas

const MERIT\_FORMULAS: Record<string, MeritCriteria> = {

'NUST': {

matricWeight: 0.10, // 10%

interWeight: 0.15, // 15%

testWeight: 0.75 // 75%

},

'FAST': {

matricWeight: 0.10,

interWeight: 0.40,

testWeight: 0.50

},

'COMSATS': {

matricWeight: 0.15,

interWeight: 0.35,

testWeight: 0.50

}

};

export function calculateMerit(input: MeritInput): number {

const criteria = MERIT\_FORMULAS[input.universityId];

if (!criteria) {

throw new Error('Unknown university');

}

// Normalize test score to percentage (assuming 200 max)

const normalizedTest = input.testScore ? (input.testScore / 200) \* 100 : 0;

const merit =

(input.matricPercentage \* criteria.matricWeight) +

(input.interPercentage \* criteria.interWeight) +

(normalizedTest \* criteria.testWeight);

return Math.round(merit \* 100) / 100;

}

export function evaluateAdmissionChance(

studentMerit: number,

lastYearClosingMerit: number

): { chance: string; message: string } {

const gap = studentMerit - lastYearClosingMerit;

if (gap >= 5) {

return {

chance: 'High',

message: 'Your merit is significantly above last year\'s closing merit!'

};

} else if (gap >= 0) {

return {

chance: 'Good',

message: 'You are close to the closing merit. Good chances!'

};

} else if (gap >= -3) {

return {

chance: 'Medium',

message: 'Merit is tight. Keep this as a backup option.'

};

} else {

return {

chance: 'Low',

message: 'Closing merit is higher. Consider improving test score.'

};

}

}

**🚀 Implementation Roadmap**

**Week 1-2: Foundation**

* Set up Next.js project with TypeScript
* Configure Clerk authentication
* Set up database (PostgreSQL on Vercel/Supabase or MongoDB Atlas)
* Create database schema & migrations
* Build basic UI components library

**Week 3-4: Core Features**

* Student profile creation & management
* Merit calculation engine
* University matching algorithm
* University detail pages
* Deadline tracker component

**Week 5: Scraping & Data**

* Implement Jina AI scraper
* Create scraping jobs for 4 universities
* Build admin panel for manual updates
* Set up cron jobs
* Manually populate initial data

**Week 6: Polish & Launch**

* Responsive design improvements
* Performance optimization
* Error handling & logging
* User testing with 20-30 students
* Soft launch & feedback collection

**📊 Success Metrics**

**MVP Success Criteria:**

1. **User Registration:** 1,000+ students in first month
2. **Engagement:** 60%+ profile completion rate
3. **Data Accuracy:** 90%+ accurate deadline information
4. **Performance:** < 2 second page load time
5. **User Satisfaction:** 4+ star rating from beta testers

**🔐 Environment Variables**

Bash

# .env.local

# Clerk Authentication

NEXT\_PUBLIC\_CLERK\_PUBLISHABLE\_KEY=pk\_test\_xxxxx

CLERK\_SECRET\_KEY=sk\_test\_xxxxx

# Database

DATABASE\_URL=postgresql://user:password@host:5432/nucap

# OR for MongoDB

MONGODB\_URI=mongodb+srv://user:password@cluster.mongodb.net/nucap

# Jina AI

JINA\_API\_KEY=jina\_c9d9d964838e48b2a99ca4eb006db329i3KwBWso30zOi6blN-AKnqd0unRt

# Cron Job Security

CRON\_SECRET=your-secret-key-here

# App

NEXT\_PUBLIC\_APP\_URL=http://localhost:3000

NODE\_ENV=development

**🛠️ Tech Stack Summary**

| **Layer** | **Technology** | **Purpose** |
| --- | --- | --- |
| **Frontend** | Next.js 14 (App Router) | React framework with SSR |
| **UI Components** | Tailwind CSS + Shadcn/ui | Rapid UI development |
| **Authentication** | Clerk | User management & auth |
| **Database** | PostgreSQL / MongoDB | Data persistence |
| **ORM** | Prisma / Mongoose | Database operations |
| **Scraping** | Jina AI API | Web data extraction |
| **Hosting** | Vercel | Frontend & API hosting |
| **Cron Jobs** | Vercel Cron / Upstash | Scheduled scraping |
| **Analytics** | Vercel Analytics | User tracking |

**📱 Sample User Flow**

text

1. Student visits NUCAP → Lands on homepage

2. Clicks "Get Started" → Redirected to Clerk signup

3. Creates account → Redirected to profile form

4. Fills academic details:

- Matric: 1000/1100 (90.9%)

- Inter: 950/1100 (86.4%)

- Expected NUST Test: 140/200

5. Submits form → System calculates merit

6. Redirected to Matches page → Shows:

- NUST CS: High chance (Merit: 82.5 vs Closing: 79.2)

- FAST CS: Good chance (Merit: 78.3 vs Closing: 78.0)

- COMSATS CS: High chance (Merit: 81.2 vs Closing: 75.5)

7. Clicks NUST → Sees:

- Application deadline: August 15, 2024 (30 days left)

- All CS departments & their merit

- How to apply button → External link

8. Adds to watchlist → Gets email notifications

**💡 Future Enhancements (Post-MVP)**

1. **Mobile App** (React Native)
2. **WhatsApp Bot** for deadline reminders
3. **AI Chatbot** for admission guidance
4. **Document Checklist** for applications
5. **University Comparison Tool**
6. **Student Forum/Community**
7. **Scholarship Database**
8. **Financial Aid Calculator**
9. **Alumni Network Integration**
10. **Predictive Analytics** for merit trends

**🎯 Next Steps to Start**

1. **Clone Next.js starter:**

Bash

npx create-next-app@latest nucap --typescript --tailwind --app

cd nucap

1. **Install dependencies:**

Bash

npm install @clerk/nextjs prisma @prisma/client axios date-fns

npm install -D @types/node

1. **Initialize Prisma:**

Bash

npx prisma init

1. **Set up Clerk:**
   * Create account at clerk.com
   * Add API keys to .env.local
2. **Create first API route:**
   * /app/api/universities/route.ts
3. **Build landing page:**
   * /app/page.tsx