**🧠 System Ka Core Idea**

* Year-wise student batches (5 students per batch).
* Diverse course-wise grouping (at least 2 B.Tech per batch, max 2 B.Tech).
* Faculty assignment yearly (2–3 batches/year).
* Students fixed per batch (change only by admin).
* Weekly attendance & marks, not daily.
* OTP-based login (email + phone).
* Admin can send notices, see pending status.
* Heavy load expected (5k+ users).
* **🧱 System Architecture (High-Level)**

**Frontend (React/Next.js + Tailwind)**

**|**

**v**

**Backend (Node.js/Express or NestJS)**

**|**

**v**

**Database (MongoDB or PostgreSQL)**

**|**

**v**

**Cloud Services (AWS/GCP/Azure or Render/Vercel)**

**|**

**v**

**Authentication (Custom OTP or Firebase Auth)**

**|**

**v**

**Notification Service (SMS API + Email SMTP)**

**🔩 Main Modules**

**1. Authentication**

* **OTP-based login via phone.**
* **Name + Email college domain check.**
* **JWT tokens for sessions.**

**2. Admin Panel**

* **Add students manually/yearly via CSV/Excel import.**
* **Batch auto-create logic:**
  + **5 per batch.**
  + **2 B.Tech min, max 2 only.**
  + **Remaining 3 from other courses.**
* **Assign faculty (2–3 per year).**
* **Change faculty for batch.**
* **See pending assignments (students/faculty).**

**3. Student Dashboard**

* **View assigned faculty.**
* **Upload/view research documents.**
* **View weekly marks & attendance.**
* **Receive notices from admin.**

**4. Faculty Dashboard**

* **View assigned batches.**
* **Access batch-wise student data.**
* **Upload attendance/marks (weekly).**
* **Send feedback to students.**

**5. Batch Management Logic**

* **Auto assign on student addition.**
* **If group not fulfilled (5 members), waitlist.**
* **When complete, assign eligible faculty.**

**🛢️ Database Design (PostgreSQL Recommended)**

**Tables:**

* **users (id, name, email, phone, role: student/faculty/admin)**
* **batches (id, year, faculty\_id, created\_at)**
* **students (user\_id, course, year, batch\_id)**
* **faculty\_assignments (faculty\_id, year, batch\_ids[])**
* **research\_docs (student\_id, batch\_id, file\_url, remarks)**
* **attendance (batch\_id, week\_number, data)**
* **marks (student\_id, subject, week\_number, marks)**
* **notifications (title, message, to: all/faculty/students/specific\_ids)**

**🔐 Security**

* **Use HTTPS (SSL) everywhere.**
* **JWT tokens with expiration.**
* **Role-based access: Student/Faculty/Admin.**
* **Rate-limiting + throttling login requests.**
* **Sanitize file uploads (for research files).**

**☁️ Cloud Infrastructure Plan**

| **Component** | **Technology** |
| --- | --- |
| **Frontend** | **Next.js (on Vercel)** |
| **Backend** | **Node.js/NestJS (Render/AWS)** |
| **Database** | **PostgreSQL (Supabase/AWS RDS)** |
| **Storage** | **S3 or Supabase for files** |
| **Auth** | **Firebase Auth or Custom OTP** |
| **Notification** | **Twilio (SMS) + SMTP (Mailgun, etc.)** |
| **Domain** | **Buy from Namecheap/GoDaddy** |
| **SSL** | **Auto (via Vercel/Cloudflare)** |

**📊 Handling Load (5000–6000 users)**

* **Use Load Balancer on cloud.**
* **Enable Horizontal Scaling (auto-scale pods/services).**
* **Use Redis for session caching if needed.**
* **Backend rate-limiting with tools like Express Rate Limit.**

**💡 Tools/Stack Suggestion**

| **Purpose** | **Tool/Library** |
| --- | --- |
| **Frontend** | **Next.js + TailwindCSS** |
| **Backend** | **NestJS or Express.js** |
| **Database** | **PostgreSQL (Supabase)** |
| **Auth** | **Firebase/Auth0 or Custom OTP** |
| **Deployment** | **Vercel + Supabase or AWS** |
| **CI/CD** | **GitHub Actions** |
| **Mobile Responsiveness** | **React Native Web (optional)** |

**✅ 1. ER Diagram (Entities and Relationships)**

**[USERS]**

**- id (PK)**

**- name**

**- email**

**- phone**

**- role (ENUM: student, faculty, admin)**

**|**

**| 1:1 (if student)**

**v**

**[STUDENTS]**

**- user\_enrollment\_id (PK, FK to USERS.id)**

**- name**

**- email\_id**

**- phone**

**- course**

**- year**

**- batch\_id (FK to BATCHES.id)**

**|**

**| M:1**

**v**

**[BATCHES]**

**- id (PK)**

**- year**

**- faculty\_id (FK to USERS.id) ❗optional — batch\_faculty table se bhi ho sakta**

**- created\_at**

**|**

**| 1:N (yearly)**

**v**

**[FACULTY\_ASSIGNMENTS]**

**- id (PK)**

**- faculty\_id (FK to USERS.id)**

**- year**

**- batch\_id (FK to BATCHES.id)**

**- branch**

**- name**

**- designation**

**- mobile\_no**

**- email**

**|**

**| 1:N (one faculty -> many research docs via batch)**

**v**

**[RESEARCH\_DOCS]**

**- id (PK)**

**- student\_id (FK to STUDENTS.user\_enrollment\_id)**

**- batch\_id (FK to BATCHES.id)**

**- file\_url**

**- remarks**

**|**

**| M:1**

**v**

**[ATTENDANCE]**

**- id (PK)**

**- batch\_id (FK to BATCHES.id)**

**- week\_number**

**- data (JSON of {student\_id: present/absent})**

**|**

**| M:1**

**v**

**[MARKS]**

**- id (PK)**

**- student\_id (FK to STUDENTS.user\_enrollment\_id)**

**- subject**

**- week\_number**

**- marks**

**|**

**| M:N (to: list of user\_ids or role)**

**v**

**[NOTIFICATIONS]**

**- id (PK)**

**- title**

**- message**

**- to\_type (ENUM: all, faculty, students, specific\_ids)**

**- recipient\_ids (ARRAY of FK to USERS.id or NULL if all)**

**- timestamp**

**🧮 2. Batch Creation Logic Algorithm (Student Based)**

**✅ Constraints:**

* **One batch = 5 students**
* **At least 2 B.Tech students**
* **Max 2 B.Tech students**
* **Other 3 = mix of other courses**
* **Student can only be in one batch of one year**
* **Faculty assigned automatically: 2, admin manually: +1 (optional)**

**🔁 Algorithm (Step-by-Step)**

**pseudocode**

**CopyEdit**

**Input: New list of students for current academic year**

**1. Initialize empty list: pending\_batch\_students = []**

**2. For each student in new\_students:**

**IF student already has batch in current year:**

**Skip**

**ELSE:**

**Add student to pending\_batch\_students**

**3. Group pending\_batch\_students by course:**

**btech\_students = filter where course == "B.Tech"**

**other\_students = all others**

**4. While len(btech\_students) >= 2 AND len(other\_students) >= 3:**

**Create new batch (generate batch\_id, year)**

**Pick 2 from btech\_students**

**Pick 3 from other\_students**

**Assign students to batch\_id**

**Save batch to DB**

**Remove assigned students from btech\_students and other\_students**

**5. If any leftover students:**

**Save them as "Unassigned Students"**

**Admin will manually assign or wait for more students to join**

**6. Auto assign available faculty:**

**For current year, check faculty who have <2 batches**

**Assign batch to eligible faculty**

**Update faculty\_assignments table**

**7. Log all assignments and send notification to students & faculty**

**✨ Special Notes**

* **All relational integrity must be maintained via foreign keys.**
* **Use unique constraints on (student\_id, year) in students table to avoid duplicate year-wise batch assignment.**
* **Admin can override assignment manually from dashboard if needed.**
* **Use queues or async jobs for assigning batches to faculty if many students are added in bulk.**