Absolutely — here's a **detailed PRD** for your product based on the idea we discussed:

PRD — Student OS (International Student Edition)

Version: Draft v1.0
Owner: Ayushmaan Mehta (Product Builder)
Type: AI-enabled web app (student productivity + communication assistant)
Primary goal: Help international students in Germany manage deadlines, documents, and formal communication with less stress.

---

1) Product Summary

Student OS is a web app for international students (starting with Germany) that combines:

- **deadline tracking** (uni, rent, permits, insurance, appointments)
- **document checklist management**
- **AI email drafting** (German/English, formal tone)
- **proof/payment record logging** (rent paid, fees paid, etc.)
- **simple reminders and organization**

Why this product exists

International students deal with too many admin tasks, deadlines, and formal emails — and small mistakes cause stress (or expensive problems). Current tools are fragmented:

- notes app for lists
- calendar for reminders
- docs folder for files
- ChatGPT for email drafts
- random screenshots for proof

Student OS puts this into **one system**.

---

2) Problem Statement

International students frequently struggle with:

1. **Missed deadlines**
   - residence permit appointments
   - insurance submissions
   - uni forms
   - rent due dates
2. **Confusing formal communication**
   - writing emails to landlords, university offices, insurance, city office, etc.
   - German wording / tone issues
   - fear of sounding rude or unclear
3. **Poor document organization**
   - no clean checklist for “what’s done / what’s missing”
   - hard to track proof of payment or sent emails
4. **Mental overload**
   - too many small admin tasks
   - uncertainty about what’s urgent

---

3) Target Users

Primary User Persona

International Student (Germany)

- Age: 18–30
- New or relatively new in Germany
- Uses English (and maybe basic German)
- Needs help with bureaucracy, communication, and organization
- Often mobile-first but also uses laptop

Secondary Personas (later)

- Erasmus students
- International students in other EU countries
- Working professionals new to Germany
- Student support mentors / agencies (B2B-lite use case)

---

4) Jobs To Be Done (JTBD)

Functional jobs

- “Help me track what I need to do and by when.”
- “Help me write a correct email quickly.”
- “Help me remember which documents I already submitted.”
- “Help me keep proof of payments and admin steps.”

Emotional jobs

- “Reduce my anxiety around official tasks.”
- “Make me feel in control.”
- “Help me avoid mistakes.”

Social jobs

- “Help me look professional in emails.”
- “Help me communicate confidently with offices/landlords.”

---

5) Product Vision

Build the **default operating system for international student admin life**:

- organized
- reliable
- AI-assisted
- simple enough to use daily

---

6) Goals and Non-Goals

Goals (MVP → v1)

- Let users create and manage deadlines
- Let users generate formal emails with AI
- Let users maintain document checklists
- Let users log payment/proof records
- Deploy fast and make it usable in real life
- Build enough trust (clean UI, reliability, no messy outputs)

Non-Goals (for now)

- No direct integration with government portals
- No automatic visa/legal advice
- No OCR-heavy document parsing in MVP
- No complex calendar sync in MVP
- No native mobile app (web-first)

---

7) Success Metrics

MVP Success Metrics (first 30–60 days)

- **Activation:** % users who create first deadline + generate first email in same session
- **Retention (7-day):** % users who return within 7 days
- **Core usage:**
    - avg deadlines created/user
    - avg AI drafts/user
    - checklist completion rate
- **Quality metrics:**
    - % AI drafts copied
    - thumbs up/down on AI output
- **Business metric (if monetized):**
    - free → paid conversion
    - first 5 paying users

Initial Targets (realistic)

- 25 signups
- 10 weekly active users
- 100 AI drafts generated
- 5 users saying they'd pay
- 1–3 paid users (soft launch)

---

8) Scope and Feature Plan

MVP (Week 1–3)

1) Authentication

- Email sign-in (magic link or password)
- Secure session handling
- User-specific data only

Supabase Auth uses JWTs and is designed to work with Row Level Security (RLS), which is a good fit for user-scoped data. (Supabase)

2) Deadline Tracker

- Create / edit / delete deadlines
- Fields:
    - title
    - due date
    - category
    - status
    - notes (optional)
- Sort by due date
- Highlight overdue / upcoming

3) AI Email Draft Helper

- Inputs:
    - recipient type (landlord / uni / insurance / employer / misc)
    - language (EN/DE)
    - tone (formal / polite / concise)
    - context (what happened + what user needs)
- Output:
    - subject line
    - email body
- Actions:
    - copy
    - regenerate
    - simplify

Use server-side API calls only, with API keys stored securely in environment variables (not exposed client-side). OpenAI docs explicitly recommend keeping API keys secret and loading them on the server. (OpenAI Developers)

4) Document Checklist

- Create checklist templates:
    - “University”
    - “Housing”
    - “Insurance”
    - “Residence permit”
- User can mark:
    - pending / done / submitted
- Optional notes per item

5) Payment / Proof Log

- Simple manual log:
    - item (e.g., rent)
    - amount
    - date
    - paid to
    - note / reference text
- Optional “proof link” field (cloud URL or local reminder)

---

v1 (Next 4–8 weeks)

6) Templates

- Saved email templates
- Reusable checklist templates
- “Duplicate previous email draft”

7) AI Writing Quality Controls

- “More formal”
- “Shorter”
- “Translate to German”
- “Explain this email in simple English”

8) Basic Notifications (in-app)

- Upcoming deadlines banner
- Overdue tasks section

9) Monetization

- Stripe Payment Link for Pro upgrade (fastest launch path)
- Pro features:
    - unlimited AI drafts
    - saved templates
    - export history
    - priority tone presets

Stripe supports Payment Links as a no-code way to accept payments, and Payment Links can be used for products/subscriptions via a hosted payment page. (Stripe Docs)

---

v2 (Later)

- file upload + document tagging
- calendar sync (Google Calendar)
- reminder emails
- “AI admin assistant” chat across user’s deadlines/docs
- multi-country onboarding packs
- shared mode (student + parent)

---

9) User Stories (MVP)

Auth

- As a student, I want to sign in easily so my data is private and saved.
- As a user, I want my deadlines and drafts to be visible only to me.

Deadlines

- As a student, I want to add deadlines so I don’t miss important admin tasks.
- As a student, I want to see what is overdue and what is due soon.

AI Email Drafts

- As a student, I want to generate a formal email in German so I can contact offices confidently.
- As a student, I want a clear subject + body so I can send it quickly.

- As a student, I want to regenerate drafts if the wording feels wrong.

Checklist

- As a student, I want a checklist for documents so I know what’s missing.
- As a student, I want to mark items as submitted so I can track progress.

Payment Log

- As a student, I want to log payments (rent/fees) so I can remember what I paid and when.

---

10) Functional Requirements

A. Authentication & User Access

Requirements

- User can sign up/login via email
- User session persists securely
- App prevents unauthorized access to protected pages
- All data queries are scoped to the authenticated user

Notes

Use Supabase Auth + RLS policies on all user tables for defense-in-depth. Supabase’s docs specifically describe RLS as a Postgres primitive and note its “defense in depth” value. (Supabase)

B. Deadline Tracker

Required fields

- `title` (string, required)
- `due_date` (date, required)
- `category` (enum/string, required)
- `status` (pending / done / overdue)
- `notes` (optional)

Behavior

- Default status = pending
- Overdue if due_date < today and not done
- Dashboard shows:
  - overdue
  - next 7 days
  - later
- CRUD actions must be responsive (<500ms perceived with optimistic UI if possible)

C. AI Email Draft Helper

Input form

- Recipient type
- Language (EN/DE)
- Tone
- Context text box
- Optional: “Include deadline date”

Output

- `subject`
- `body`
- optional “translated version” (later)

Validation

- Input length limits (avoid abuse + cost spikes)
- Disallow empty context
- Rate limit requests/user (basic)

Guardrails

- Show disclaimer:
  - “Review before sending”
  - “This is drafting support, not legal advice”
- Prevent risky outputs (e.g., fake claims, legal instructions)
- Log prompt usage metadata only (not full content by default in MVP, unless user opts in)

D. Document Checklist

Behavior

- User can create checklist categories
- User can add items
- User can mark status
- User can sort/filter by category/status
- App shows completion progress (%)

E. Payment/Proof Log

Fields

- `type` (rent / insurance / fee / misc)
- `amount`
- `currency`
- `paid_on`
- `recipient`
- `reference_note`
- `proof_url` (optional)
- `status` (paid / pending)

Behavior

- Basic list + filter
- No automatic accounting logic
- Export later (v1+)

F. Pro Upgrade (v1)

Free tier (default)

- 10 AI drafts / month
- 2 checklist templates
- basic deadlines

Pro tier

- unlimited drafts
- saved templates
- advanced tone controls
- history & export

Payments

- Start with Stripe Payment Links
- Upgrade button opens hosted Stripe page
- App updates plan status via webhook after payment completion

Stripe recommends webhooks for async payment events, and `checkout.session.completed` is a key event for successful checkout completion. (Stripe Docs)

---

11) Non-Functional Requirements

Performance

- Initial page load < 3s on normal connection
- AI generation response target: 3–8s
- Dashboard interactions feel instant

Reliability

- No silent failures
- Friendly error messages for:
  - auth issues
  - AI timeouts
  - network failures

Security

- Secrets only on server (OpenAI, Stripe)
- Input validation on all API routes
- RLS on all tables
- Rate limiting on AI endpoints
- Basic audit logs (later)

- Follow OWASP Top 10 awareness during development (current release is OWASP Top 10 2025). (OWASP Foundation)

Testing

- E2E smoke tests for:
    - login
    - add deadline
    - generate email draft
- Run in CI on each push/PR

Playwright supports cross-browser E2E testing and CI use, and GitHub Actions supports running build/test workflows automatically. (Playwright)

Observability

- Error logging (Sentry or equivalent)
- Track failed AI requests
- Track API latency

Sentry provides a dedicated Next.js integration guide for capturing errors/logs/traces. (Sentry Documentation)

---

12) UX Requirements

Design Principles

- **Minimal**
- **Trustworthy**
- **No clutter**
- **Fast to act**

Core UI surfaces

1. **Dashboard**
    - upcoming deadlines
    - overdue items
    - quick actions (New Deadline / Draft Email)
2. **Email Draft Page**

    - input panel
    - output panel
    - copy/regenerate buttons
3. **Checklist Page**
    - categories + progress bars
4. **Payments/Proof Page**
    - simple records list

UX quality bar

- clear labels (especially for non-native English speakers)
- no jargon
- obvious empty states
- mobile responsive

---

13) Data Model (Initial)

Tables

`users` (handled by Supabase Auth)

- managed by auth system

`deadlines`

- `id` (uuid)
- `user_id` (uuid)
- `title` (text)
- `due_date` (date)
- `category` (text)
- `status` (text)
- `notes` (text, nullable)
- `created_at` (timestamp)
- `updated_at` (timestamp)

`checklist_lists`

- `id`

- `user_id`
- `title`
- `category`
- `created_at`

`checklist_items`

- `id`
- `list_id`
- `user_id`
- `label`
- `status`
- `notes`
- `created_at`

`email_drafts`

- `id`
- `user_id`
- `recipient_type`
- `language`
- `tone`
- `input_summary` (optional shortened version)
- `subject`
- `body`
- `created_at`

`payment_logs`

- `id`
- `user_id`
- `type`
- `amount`
- `currency`
- `paid_on`
- `recipient`
- `reference_note`
- `proof_url`
- `status`

- `created_at`

`subscriptions` (v1)

- `id`
- `user_id`
- `plan`
- `status`
- `stripe_customer_id`
- `stripe_subscription_id` (nullable if one-time)
- `current_period_end`

---

14) Permissions & Authorization

Principle

Every row in user-owned tables must be scoped to `auth.uid()` (Supabase RLS policy style).

Example policy intent (not SQL here)

- Users can only read/write rows where `user_id == current user`
- No public table access
- Server-side admin operations limited to service role (backend only)

Supabase Auth + JWT + RLS is a strong fit for this pattern, and Supabase documents how JWT claims support RLS authorization. (Supabase)

---

15) AI System Design (MVP)

AI Feature

Email Draft Generator

Input → Output contract

Input

- context text
- recipient type
- tone
- language
- optional date/details

Output (structured)

- subject
- body
- notes (optional internal quality hint)

Prompting strategy

- Use fixed system prompt for:
    - tone consistency
    - no legal advice
    - no hallucinated facts
    - concise professional style
- Use templates per recipient type (landlord, uni, insurance)

Safety rules

- Never invent legal rights/laws
- If user asks legal interpretation, suggest contacting official office/lawyer
- Encourage review before sending
- No hidden data retention

---

16) Privacy, Data Protection, and Compliance (EU-first)

Because your users are in Germany/EU, privacy matters a lot.

Core privacy approach

- Collect only necessary data (**data minimization**)
- Store data only as long as needed (**storage limitation**)
- Be clear about what data is used and why (**purpose clarity**)
- Give users deletion/export controls (v1+)

The European Commission’s GDPR guidance pages emphasize principles around how much data can be collected and how long it can be kept, and the EU’s data protection framework is centered on GDPR. (European Commission)

Data categories handled

- account email
- deadlines
- checklist items
- email draft content (potentially sensitive admin information)
- payment/proof notes

Important compliance notes

- Add Privacy Policy before public launch
- Add Terms of Use
- Add AI disclaimer (“drafting assistant, not legal advice”)
- If using processors outside EU, document transfer safeguards (SCCs etc. may apply depending on provider setup) per EU guidance on international data transfers. (European Commission)

---

17) Tech Stack (Recommended)

Frontend / App

- **Next.js (App Router)** — good for modern React features and full-stack web app patterns. Next.js documents the App Router as file-system based and built around newer React features. (nextjs.org)
- Tailwind CSS

Backend / Data / Auth

- **Supabase** (Postgres + Auth + RLS)
- Server actions / API routes for AI + payments

AI

- **OpenAI API** (server-side only)

- API key in environment variables only (never client-side) per OpenAI docs. (OpenAI Developers)

Deployment

- **Vercel**
- Use preview deployments on branch pushes/PRs for fast testing and feedback. Vercel supports automatic preview deployments and unique URLs. (Vercel)

Payments

- **Stripe Payment Links** for initial monetization
- Add webhooks for plan sync

Quality

- **Playwright** (E2E)
- **GitHub Actions** (CI)
- **Sentry** (error monitoring)

---

18) Release Plan

Milestone 1 — MVP Alpha (Week 1)

- Auth
- Deadline CRUD
- AI Email Draft MVP
- Vercel deploy

Milestone 2 — MVP Beta (Week 2)

- Document checklist
- Payment/proof log
- Basic UI polish
- E2E smoke tests + CI

Milestone 3 — Soft Launch (Week 3)

- Invite 10–20 students
- Collect feedback
- Add usage analytics
- Fix rough edges

Milestone 4 — Monetization Test (Week 4–5)

- Stripe Payment Link
- Pro gating
- 1 paid plan
- Basic webhook subscription status sync

---

19) Risks and Mitigations

Risk 1 — AI output is awkward/wrong

Mitigation

- strict output templates
- regenerate button
- feedback buttons
- “review before sending” disclaimer

Risk 2 — Users don’t stick

Mitigation

- make dashboard useful on day 1
- focus on 1 killer feature (email drafts + deadlines)
- reduce setup friction with starter templates

Risk 3 — Too many features too early

Mitigation

- stick to MVP scope
- no file uploads/OCR initially
- ship weekly

Risk 4 — Security/privacy mistakes

Mitigation

- server-only secrets
- RLS everywhere
- basic security review using OWASP Top 10 checklist mindset
- logging and monitoring

---

20) MVP Acceptance Criteria

The MVP is “done” when:

- User can sign in and only see their own data
- User can create/edit/delete deadlines
- User can generate AI email drafts (subject + body)
- User can manage at least one checklist
- User can log one payment/proof entry
- App is deployed and usable on Vercel
- Basic error handling is in place
- At least 1 E2E test runs in CI
- 5 real users can test it without your help

---

21) Future Extensions (Good for your portfolio + revenue)

Once MVP works, these are powerful upgrades:

A) AI Inbox Assistant (paste official email → explain + draft reply)

Huge value for international students.

B) Smart Deadline Extraction

Paste an email/letter, auto-suggest deadline and task.

C) “Admin Packs”

Prebuilt templates:

- landlord pack

- uni office pack
- insurance pack
- residence permit pack

D) Team/Parent Mode

Share checklist progress with parent or mentor (read-only)

---

22) Positioning Statement

> **Student OS** helps international students in Germany stay on top of deadlines, documents, and formal communication with an AI-powered workspace built for student admin life.

If you want, I can also give you the **next step PRD pack**:

- exact **database schema SQL**
- **API routes list** (endpoint-by-endpoint)
- **UI wireframe layout** for each page
- **Copilot prompts** for building MVP in VS Code quickly