AI PAYMENT REMINDER AGENT

(Expanded Documentation — Problem → Solution → Tech Stack)

WORKFLOW

WEB FORM SUBMIT

|

WEBHOOK TRIGGER

|

STORE DATA IN GOOGLE SHEETS

|

APPLY BUSINESS LOGIC

|

EMAIL REMINDER

|

SMS REMINDER

|

STORE LOG INFORMATION

1. Problem Statement

1.1 The Real-World Challenge

In educational institutions, small businesses, and personal finance management, payment racking remains a persistent challenge.In the case of college fees, consequences of missed payments include:

* Late fee penalties
* Service restrictions (e.g., holds on results, blocked access to facilities)
* Administrative overhead for both students and the accounts department

Most students rely on manual reminders in calendars, sticky notes, or verbal alerts from peers or parents. However, manual systems are prone to:

* Human error (forgetting deadlines)
* No real-time update mechanism
* Lack of integration between deadlines and actionable payment workflows (such as direct payment portals)

1.2 Manual Reminder Limitations

From a Student's Perspective

- No central place to store all upcoming fee deadlines.
- Often reliant on emails from the administration, which can be missed.
- Difficulty tracking multiple instalments (semester fees, hostel charges, library fines).

From an Institution's Perspective

- Manual follow-up requires staff time and effort.
- Communication delays can cause a spike in late payments.
- Inability to easily filter and notify specific sets of students (e.g., only hostel residents).

1.3 Why Automation is Needed

An **automated payment reminder** system:

- Sends reminders without human intervention.
- Integrates directly with stored data (e.g., Google Sheets of due dates).
- Uses **multi-channel notifications** (Email + SMS).
- Reduces late payments by keeping students informed with precise and timely alerts.
- Enhances **cash flow predictability** for institutions.

This **Problem** is not unique to colleges — similar issues occur for:

- Subscription-based services.
- Customer invoice payments.
- Loan EMI reminders.

But in the college fees scenario, the **impact is more critical** because it affects academic access and student services.

✅ **Summary of the Problem:**

- **Key Issue:** Missed deadlines due to lack of timely communication and tracking.
- **Impact:** Financial penalties, academic disruption, administrative burden.
- **Current Solutions:** Manual reminders & mailers → **inefficient & error-prone**.
- **Need:** Automated, data-driven, multi-channel notification platform.

2. Solution Description

(~ 2 pages detailed)

2.1 Overview of the Proposed Solution

An **AI-powered, automated payment reminder** system where:

- A **simple web form** collects student payment-related information.
- Data is **securely transferred** to an automation tool.
- **Google Sheets** stores submissions for record-keeping.
- **Conditional Logic** (via custom Python scripts) validates and routes payments.
- The system **automatically sends reminders** via **email** and/or **SMS**.
- Institution staff only update due-date data when needed — the rest is handled by the system.

2.2 How It Works for College Fee Reminders

1. **Student Submission Step:**
    - Student opens a mobile-friendly form.
    - Inputs: Name, Email, Phone Number, Due Date, Fee Amount.
    - Submission triggers an **instant webhook event** to the automation platform.
2. **Data Capture & Validation:**
    - The webhook routes the data to:
        - **Google Sheets** for logging and reference.
        - **Python code** (optional) for:
            - Checking if the due date is valid.
            - Adding calculated "reminder dates" (e.g., 7 days before, 3 days before).
3. **Automated Notifications:**
    - A **router module** decides channels:
        - Path 1 → **Email via Brevo**
        - Path 2 → **SMS via Brevo**
    - Each message is personalized:
      "Dear [Name], your college fee of ₹[Amount] is due on [Date]. Please make payment before the deadline to avoid penalties."
4. **Administration Dashboard (Google Sheets):**
    - Staff see all submissions in one central sheet.
    - They can sort/filter based on due dates for extra communication if needed.

2.3 Key Features

- **Automated Trigger:** No manual reminder sending needed.
- **Multi-Platform Reach:** Email + SMS ensures maximum delivery.
- **Data Safety:** Student data stored securely.
- **Scalability:** Works for hundreds or thousands of students without performance drop.
- **Flexibility:** Can be extended to handle other payments (e.g., library fines).

2.4 Why This Solution Works

- Students are constantly on their phones → SMS has the **highest open rate** (98%+).
- Email provides a formal notification channel, useful for record purposes.
- Automation eliminates clerical errors and delays.
- Cloud tools (Google Sheets, Brevo, Make/Integromat) reduce deployment cost — no expensive hardware needed.

3. Tech Stack Breakdown

(~ 2 pages detailed)

The **technology stack** combines web technologies, cloud automation platforms, and communication tools.

Component	Technology	Role in the System	Why Chosen?
Frontend Web Form	HTML, CSS, JavaScript	Student-facing input form	Simple, responsive, widely supported
Trigger	Make (Integromat) Webhook	Captures form submissions instantly	Real-time processing without manual refresh
Storage	Google Sheets	Stores all reminders data	Free, easy sharing, cloud-based
Business Logic Engine	OCodeKit (Python)	Applies date logic, custom rules	Flexible, powerful, easily updated
Notification Service	Brevo (Sendinblue)	Sends email and SMS	Integrated, affordable, multi-channel
Flow Control	Router Module (Make)	Routes data to Email or SMS based on conditions	Reduces unnecessary processing

3.1 Why This Stack is Ideal

1. **Cost-Effective:** Uses free/low-cost tools for maximum ROI.
2. **Cloud-Based:** No dependence on physical servers.
3. **Modular:** Each part (form, webhook, logic, notifications) can be swapped or upgraded without breaking the system.
4. **Non-Technical Friendly:** College staff can use Google Sheets without technical training.

3.2 Example Data Flow

1. Student fills in a form with:
    - Name: *Rohan Sharma*
    - Email: *rohan@example.com*
    - Due Date: *15 August 2025*
    - Fee Amount: *₹50,000*
2. Form POSTs data to a webhook in Make.
3. Google Sheets row is created:

Name	Email	Phone	Due Date	Amount
Rohan Sharma	rohan@example.com	9876543210	15-Aug-2025	₹50,000

4. Router sends:
    - Email to *rohan@example.com*
    - SMS to +91-9876543210 with a friendly, clear reminder.