Product Requirements Document (PRD): BITS-GPT

Version: 1.0
Status: Approved for MVP Development
Timeline: 8 Weeks (MVP)
Target: BITS Pilani (Dubai Campus Pilot -> Multi-Campus Rollout)

1. Executive Summary

BITS-GPT is a unified, RAG-based (Retrieval-Augmented Generation) AI assistant designed to eliminate information fragmentation within the university ecosystem. It serves as a central interface connecting users to two distinct data sources:

1. **Static Institutional Knowledge:** Official documents, handbooks, and policies via Vector Search.
2. **Dynamic Personal Data:** Real-time student records via secure ERP/LMS API integration.

Core Value Proposition: A single secure point of entry for "General Queries" (e.g., Library rules) and "Personal Queries" (e.g., "What are my outstanding dues?").

2. User Personas & Access Levels

Persona	Access Level	Key Capabilities
Student	Level 1	Query personal grades, fees, attendance; access general campus info.
Faculty	Level 2	Query teaching schedules, course rosters, research grants; access general info.
Admin	Level 3	Query department-level stats, generate reports; access all general info.
Guest	Public	Access public website info only (Admissions, General FAQs).

3. Functional Requirements

3.1. Core Conversation Engine

- **Context-Aware Chat:** Multi-turn conversation capability retaining context within a session.
- **Intent Classification:** Automatic routing of queries to either the "Vector Store" (General) or "ERP Client" (Personal).
- **Citations:** Every general answer must cite the source document (e.g., *"Source: Student Handbook 2024, Page 12"*).
- **Hallucination Guardrails:** If data is missing, the system must reply "I do not have information on that" rather than inventing facts.

3.2. Data Modules (The "Brain")

A. Static Knowledge Base (Vector RAG)

- **Academic Regulations:** Course catalogs, prerequisites, grading policies, add/drop rules.
- **Campus Life:** Hostel rules, mess menus/timings, club details, event calendars.
- **Library:** Rules, repository access guides, journal subscriptions.
- **Placements:** Past placement records, company eligibility criteria, PS-I/PS-II guidelines.
- **IT Support:** Wi-Fi configuration guides, VPN access, software license portals.

B. Dynamic Personalization (ERP Integration)

- **Financial:** Fetch outstanding dues, generate payment links, view transaction history.
- **Academic:** Fetch current CGPA, transcript summary, enrolled courses, attendance %.
- **Schedule:** Fetch personalized class timetable and exam dates.

3.3. Output Formats

- **Text:** Standard conversational responses.
- **Tables:** For schedules, fee breakdowns, and grade summaries.
- **Links:** Direct download links for PDFs (handouts, past papers).
- **Actionable Items:** "Pay Now" links or "Book Slot" deep links.

4. Technical Architecture

4.1. High-Level Stack

- **Frontend:** React + Vite (Lightweight, Single Page Application).
- **Backend Orchestrator:** FastAPI (Python) - Handles routing, auth, and logic.
- **Vector Database:** ChromaDB (Self-hosted MVP) or Pinecone.
- **LLM Provider:** Hosted APIs (Anthropic Claude 3 Haiku for speed/cost, or GPT-4o for complex reasoning).
- **Infrastructure:** Dockerized microservices deployed on AWS/Azure.

4.2. The 4-Layer Data Flow

1. **Presentation Layer:** React app sends User Query + JWT Token to Backend.
2. **Orchestration Layer (FastAPI):**
   - Validates JWT.
   - **Router:** Classifies intent -> IS_PERSONAL vs IS_GENERAL.
3. **Data Retrieval Layer:**
   - *If General:* Queries Vector DB for top-k chunks.
   - *If Personal:* Calls ERP_Client -> Secure API request to University ERP -> Returns JSON.
4. **Generation Layer:**
   - Constructs Prompt: System Context + Retrieved Data + User Query.
   - Sends to LLM -> Streams response to Frontend.

4.3. Critical Security Architecture (Zero Retention)

- **In-Memory Processing:** Personal data fetched from ERP is stored in volatile memory *only* for the duration of the request lifecycle.
- **No PII Logging:** Logs must sanitize inputs to remove Student IDs, Names, and Financial figures.
- **Encryption:** All internal traffic (Backend <-> ERP) via mTLS.

5. API Definitions (Internal)

5.1. POST /api/v1/chat

- **Headers:** Authorization: Bearer <token>
- **Body:** { "message": "What are my dues?", "session_id": "uuid" }
- **Response:** { "reply": "Your dues are...", "sources": [...] }

5.2. POST /api/v1/ingest (Admin Only)

- **Body:** { "document_url": "s3://...", "category": "hostel_rules" }
- **Action:** Triggers OCR/Parsing pipeline -> Embedding generation -> Vector Store update.

6. Data Integration Schema

6.1. ERP Connector Interface

The system requires a read-only API exposed by the university ERP (e.g., ERPNext, PeopleSoft).

Mock Response Structure (Student Profile):

```
{
  "student_id": "2023A7PS001U",
  "finance": {
    "outstanding_amount": 2450.00,
    "currency": "AED",
    "due_date": "2025-10-25",
    "payment_link": "[https://bits-pay.ae/](https://bits-pay.ae/)..."
  },
  "academics": {
    "cgpa": 8.4,
    "current_semester": 5,
    "enrolled_courses": ["CS F211", "MATH F111"]
  }
}
```

7. UX/UI Design Requirements

7.1. Interface

- **Minimalist Chat:** Similar to ChatGPT/Perplexity. Focus on typography and readability.
- **Mobile-First:** Optimized for smartphones (primary student device).
- **Source Pills:** Clickable citations below the answer (e.g., [1] Student Handbook).

7.2. User Flow

1. **Login:** SSO via University Email (Microsoft/Google).
2. **Onboarding:** Simple carousel: "Ask me about Fees, Grades, or Rules."
3. **Active Chat:**
    - User types.
    - "Thinking..." indicator.
    - Streaming text response.
4. **Action:** User clicks a provided link (e.g., "Download Handout").

8. Implementation Roadmap (8-Week MVP)

Phase	Duration	Key Deliverables
1. Foundation	Weeks 1-2	IaC setup, SSO integration, Vector DB setup, Ingestion of public PDFs.
2. Core Dev	Weeks 3-5	FastAPI Backend, ERP Connector (Mock/Staging), Basic RAG pipeline, React UI Skeleton.

3. Integration	Weeks 6-7	Connect Real ERP APIs, Security Audit, Prompt Engineering (reduce hallucinations), E2E Testing.
4. Deployment	Week 8	Production Deploy, Monitoring setup, Pilot launch (50 users).

9. Constraints & Edge Cases

- **ERP Downtime:** If the ERP API is unreachable, the bot must reply: "I cannot access personal records right now, but I can answer general questions."
- **Ambiguous Queries:** If a user asks "When is the exam?", the bot must ask: "Which course are you referring to?"
- **Rate Limiting:** Limit user queries to 50/hour to prevent API abuse and cost overruns.
- **Data Latency:** ERP data fetch must timeout after 5 seconds to prevent hanging the chat.

10. Future Scope (Post-MVP)

- **Voice Interface:** Speech-to-text for accessibility.
- **Proactive Alerts:** Push notifications for due dates (via ERP webhook events).
- **Multi-Language:** Support for Arabic/Hindi query processing.
- **Agentic Workflows:** "Book a room for me" (Write-access to ERP).