Smart Class Update Agent

System Design Document

Name: **Dhruv Choudhary**
Email: dhruv.choudhary@iitgn.ac.in
University: **Indian Institute of Technology Gandhinagar**

1 Introduction

The **Smart Class Update Agent** is an AI-powered application designed to automate the process of tracking academic deadlines from Google Classroom and synchronizing them to Google Calendar. The system extracts deadlines from classroom assignments and announcements, then creates corresponding calendar events, helping students manage their academic responsibilities more efficiently.

2 Problem Statement

University students often struggle to keep track of multiple deadlines across different courses in Google Classroom. The information is scattered across assignments and announcements, making it difficult to maintain a comprehensive view of upcoming deadlines. Manual tracking is time-consuming and prone to human error. This system addresses these pain points by providing automated deadline extraction and calendar synchronization.

3 System Architecture

3.1 High-Level Architecture

The system follows a client-server architecture with:

- **Frontend Client**: React-based web application providing user interface
- **Backend Server**: FastAPI-based REST API handling business logic
- **AI Services**: Google Gemini API for natural language understanding
- **External Services**: Google Classroom API and Google Calendar API integration

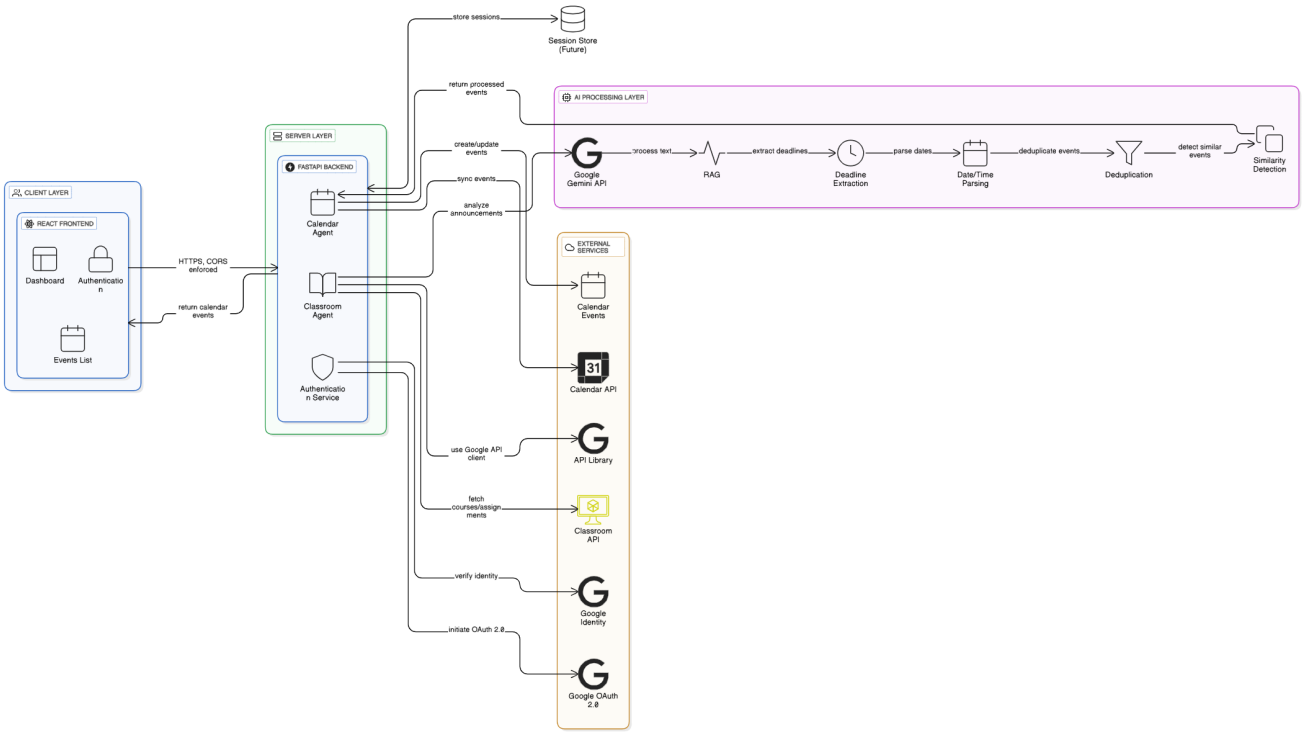

Figure 1: High-Level System Architecture for the Smart Class Update Agent.

3.2 Component Architecture

Frontend Components

- **Authentication Module**: Manages Google OAuth login flow
- **Dashboard**: Displays synchronization statistics and calendar events
- **Sync Interface**: Provides controls for the synchronization process
- **Calendar Events Manager**: Handles display, creation, and deletion of calendar events

Backend Components

- **API Server**: FastAPI application managing HTTP requests and responses
- **Authentication Service**: Google OAuth 2.0 integration and session management
- **Classroom Agent**: Fetches assignments and announcements from Google Classroom
- **Calendar Agent**: Creates and manages Google Calendar events
- **RAG System**: AI-powered deadline extraction from unstructured text
- **Deduplication Service**: Prevents creation of duplicate calendar events

4 Component Breakdown with File Structure Context

4.1 Frontend Components

4.1.1 Project Structure

```
frontend/
          public/
                favicon.ico
                index.html
          src/
                components/
                      CalendarEventsList.jsx
                      DashboardView.jsx
                      Header.jsx
                      LoadingSpinner.jsx
                      LoginScreen.jsx
                      StatsCard.jsx
                      SyncProgressBar.jsx
                App.jsx
                index.css
                main.jsx
          package.json
          vite.config.js
```

4.1.2 Key Component Files and Functions

App.jsx

- **Purpose**: Main application container managing authentication and routing
- **Key Functions**:
  - `checkAuthStatus()`: Verifies if the user is authenticated
  - `handleLogin()`: Redirects to the OAuth login flow
  - `handleLogout()`: Terminates user session

DashboardView.jsx

- **Purpose**: Main dashboard interface after login
- **Key Functions**:
  - `handleSync()`: Initiates the classroom-calendar synchronization
  - `handleDeleteEvent()`: Removes a calendar event
  - `handleAddEvent()`: Creates a custom calendar event
  - `getLastSyncTimeDisplay()`: Formats the last sync time

CalendarEventsList.jsx

- **Purpose**: Displays and manages calendar events
- **Key Functions**:
  - `handleSubmit()`: Processes form submission for new events
  - `adjustTimeForBackend()`: Converts local time to UTC
  - `formatEventDate()`: Formats event dates for display
  - `handleInputChange()`: Manages form input state changes

Header.jsx

- **Purpose**: Top navigation bar with app title and logout button
- **Key Functions**: Renders app title and conditional logout button

LoadingSpinner.jsx

- **Purpose**: Reusable loading indicator component
- **Key Functions**:
  - Renders different sized spinners with customizable messages
  - Supports fullscreen overlay mode

LoginScreen.jsx

- **Purpose**: Initial login page with OAuth button
- **Key Functions**:
  - Handles Google login button click
  - Displays application introduction

StatsCard.jsx

- **Purpose**: Dashboard card showing synchronization statistics
- **Key Functions**: Renders statistics with icons and formatted numbers

SyncProgressBar.jsx

- **Purpose**: Visual progress indicator during synchronization
- **Key Functions**:
  - Shows progress through multi-step process
  - Displays current synchronization action

4.2 Backend Components

4.2.1 Project Structure

```
backend-agent/
          src/
                utils/
                      google_auth.py
                      gemini_api.py
                main.py
                config.py
          client_secret.json
          requirements.txt
          README.md
```

4.2.2 Key Files and Functions

main.py

- **Purpose**: Core FastAPI application with all endpoints
- **Key Functions**:
  - `login()`: Initiates Google OAuth flow
  - `oauth2callback()`: Handles OAuth response
  - `check_auth_status()`: Verifies authentication status
  - `sync_all()`: Main synchronization endpoint
  - `create_calendar_event()`: Creates a custom event
  - `delete_calendar_event()`: Removes a calendar event
  - `get_current_credentials()`: Dependency for extracting credentials

utils/google_auth.py

- **Purpose**: Google OAuth authentication utilities
- **Key Functions**:
  - `get_flow()`: Creates OAuth flow object
  - `get_authorization_url()`: Generates OAuth consent URL
  - `fetch_token()`: Exchanges auth code for tokens

utils/gemini_api.py

- **Purpose**: Interfaces with Google Gemini API for AI processing
- **Key Functions**:
  - `extract_deadlines_from_text()`: Uses AI to find deadline information
  - `structure_deadline_data()`: Converts AI output to structured format

4.3 Detailed Component Implementation

4.3.1 Authentication Flow Components

OAuth Integration (`google_auth.py` & `main.py`)

- **Files Used**: `google_auth.py`, `main.py`, `App.jsx`, `LoginScreen.jsx`
- **Implementation Details**:
  - Backend creates OAuth flow with specified scopes (Classroom and Calendar)
  - Frontend redirects to `/login` endpoint
  - Backend redirects to Google consent screen
  - After consent, Google redirects to `/oauth2callback`
  - Backend processes token and establishes session
  - Frontend redirects to dashboard on successful authentication

Session Management

- **Files Used**: `main.py` (SessionMiddleware), `App.jsx` (auth state)
- **Implementation Details**:
  - FastAPI's SessionMiddleware maintains server-side session
  - Frontend checks auth status on initial load
  - Access tokens stored securely in session
  - Token refresh handled automatically when needed

4.3.2 Classroom Data Synchronization

Course and Assignment Retrieval

- **Files Used**: `main.py` (`sync_all()` function)
- **Implementation Details**:
  - Fetches active courses using Google Classroom API
  - Retrieves course work (assignments) for each course
  - Collects course announcements using separate API calls
  - Processes assignment due dates into standardized format

Announcement Processing

- **Files Used**: `main.py`, `gemini_api.py`
- **Implementation Details**:
  - Extracts raw announcement text from Classroom API
  - Sends text to Gemini API for deadline extraction
  - Parses structured deadline information from AI response
  - Validates extracted dates and times

4.3.3 Calendar Integration

Event Creation Pipeline

- **Files Used**: `main.py` (`sync_all()` function)
- **Implementation Details**:
  - Converts assignments to calendar event format
  - Converts extracted deadlines to event format
  - Checks for duplicate events before creation
  - Creates events via Google Calendar API
  - Returns created event details to frontend

Custom Event Management

- **Files Used**: `main.py` (`create_calendar_event()`, `delete_calendar_event()`), `CalendarEventsLis`
- **Implementation Details**:
  - Frontend provides form for event details
  - Adjusts time zones to ensure correct event timing
  - Backend creates/deletes events using Calendar API
  - Updates UI state after successful operations

4.3.4 UI Components

Dashboard Statistics

- **Files Used**: `DashboardView.jsx`, `StatsCard.jsx`
- **Implementation Details**:
  - Calculates statistics from synchronization results
  - Displays counts of assignments, announcements, and events
  - Updates in real-time after synchronization

Event Management Interface

- **Files Used**: `CalendarEventsList.jsx`
- **Implementation Details**:
  - Provides filterable, sortable event list
  - Form for adding custom events with date/time picker
  - Delete confirmation and success feedback
  - Time zone conversion between display and storage

Synchronization Progress

- **Files Used**: `DashboardView.jsx`, `SyncProgressBar.jsx`
- **Implementation Details**:
  - Visual progress indicator for multi-step process
  - Shows current synchronization stage
  - Provides feedback on errors or completion
  - Prevents multiple simultaneous sync operations

4.4 Time Zone Handling Implementation

Frontend Time Zone Management

- **Files Used**: `CalendarEventsList.jsx` (`adjustTimeForBackend()`, `formatEventDate()`)
- **Implementation Details**:
  - Detects user's local time zone using browser APIs
  - Converts local input times to UTC for backend
  - Converts UTC times to local for display

Backend Time Zone Processing

- **Files Used**: `main.py` (`create_calendar_event()` function)
- **Implementation Details**:
  - Receives time data in UTC format
  - Uses user's Google Calendar time zone setting
  - Creates events with proper time zone context
  - Handles all-day vs. time-specific events differently

4.5 Data Processing Implementations

Assignment Processing

- **Files Used**: `main.py` (`sync_all()` function)
- **Implementation Details**:
  - Extracts due dates from assignment objects
  - Handles assignments with and without due times
  - Creates structured event descriptions with links
  - Sets appropriate event colors based on type

AI-Powered Deadline Extraction

- **Files Used**: `gemini_api.py`, `main.py`
- **Implementation Details**:
  - Constructs context-aware prompts for Gemini API
  - Parses natural language dates and times
  - Extracts assignment titles and details
  - Validates extracted information against rules

Event Deduplication

- **Files Used**: `main.py` (`sync_all()` function)
- **Implementation Details**:
  - Compares event titles, dates, and courses
  - Uses fuzzy matching for similar assignments
  - Prioritizes newer information when conflicts occur
  - Avoids creating duplicate events for the same deadline

5 Data Design

5.1 Data Flow Architecture

1. Authentication via Google OAuth 2.0
2. Classroom data retrieval
3. AI deadline extraction
4. Deduplication
5. Calendar sync
6. Updated events returned to frontend

5.2 Key Data Structures

```
# Assignment Data Structure
{
    "id": "string",
    "title": "string",
    "courseId": "string",
    "courseName": "string",
    "description": "string",
    "dueDate": {
        "year": int,
```

```
        "month": int,
        "day": int,
        "hours": int,
        "minutes": int
    },
    "alternateLink": "string",
    "workType": "string"
}
```

Listing 1: Assignment Data Structure

```
# Extracted Deadline Structure
{
    "title": "string",
    "dueDate": {
        "year": int,
        "month": int,
        "day": int,
        "hours": int,
        "minutes": int
    },
    "courseName": "string",
    "courseId": "string",
    "description": "string",
    "source": "announcement|assignment",
    "confidence": float  # AI confidence score
}
```

Listing 2: Extracted Deadline Structure

```
# Calendar Event Structure
{
    "id": "string",
    "title": "string",
    "description": "string",
    "start": {
        "dateTime": "string",
        "timeZone": "string"
    },
    "end": {
        "dateTime": "string",
        "timeZone": "string"
    },
    "htmlLink": "string",
    "type": "assignment|announcement|custom",
    "courseName": "string",
    "courseId": "string",
    "status": "created|existing",
    "colorId": "string"  # Color identifier used for different event
       types
}
```

Listing 3: Calendar Event Structure

6 Technology Stack

6.1 Frontend Technologies

- React: Selected for its component-based architecture enabling modular UI development
- Tailwind CSS: Utility-first CSS framework chosen for rapid UI development
- Axios: Promise-based HTTP client for API communication
- React-Toastify: Provides user feedback through toast notifications

6.2 Backend Technologies

- FastAPI: High-performance web framework chosen for its automatic API documentation and strong typing support
- Google API Client Libraries: Official Python libraries for integration with Google APIs
- Google Auth Library: Handles OAuth 2.0 authentication flow
- Uvicorn: ASGI server for hosting the FastAPI application

6.3 AI and ML Components

- Google Gemini API: Used for natural language understanding to extract deadlines from announcement text
- RAG System: Custom implementation for analyzing classroom announcements and extracting structured deadline information

7 API Design

7.1 Authentication Endpoints

```
GET /login
- Initiates Google OAuth 2.0 authentication flow
- Returns: Redirect URL to Google consent screen

GET /oauth2callback
- Handles OAuth callback with authorization code
- Parameters: code, state
- Returns: Authentication status and user session

GET /check-auth-status
- Verifies current user authentication status
- Returns: Boolean authentication state

GET /logout
- Terminates user session and clears tokens
```

7.2 Core Functionality Endpoints

```
GET /sync-all
- Main orchestration endpoint for complete synchronization
- Process: Fetch     Extract     Deduplicate     Create Events
- Returns: Synchronization summary and created events

POST /calendar-event
- Creates custom calendar event
- Body: Event details (title, date, description, time)
- Returns: Created event data

DELETE /calendar-event/{event_id}
- Removes specific calendar event
- Parameters: event_id
- Returns: Deletion confirmation
```

8 Security Implementation

8.1 Authentication Security

- OAuth 2.0: Secure authentication without storing user credentials
- Token Management: Secure storage and management of access tokens
- Session Security: Server-side session storage with secure cookies

8.2 API Security

- CORS Configuration: Restricts API access to authorized frontend domains
- Scope Limitations: Requests minimal necessary Google API permissions
- Input Validation: Ensures type safety and data validation

9 Future Enhancements

1. Priority Classification: AI-powered deadline importance scoring
2. Smart Notifications: Context-aware reminder scheduling
3. Database Support: Local storage with sync when connection restored
4. WhatsApp Support: Extracts deadline-related information from WhatsApp messages (via Business API) and synchronizes it with Google Calendar.

10 Development Process

The project followed an iterative development approach:

- Feature development in small, testable increments
- API-First design with backend implementation preceding frontend
- AI-Assisted development using ChatGPT and Copilot for code generation and debugging
- Continuous testing and refinement based on user feedback

11 Conclusion

The Smart Class Update Agent successfully bridges the gap between Google Classroom's information storage and Google Calendar's scheduling capabilities, creating a seamless workflow that reduces manual effort and improves academic organization. The AI-powered deadline extraction capability distinguishes this solution from simple calendar integration tools, providing intelligent processing of unstructured academic content.

The implementation delivers a solution that helps students manage their academic responsibilities more effectively through automated deadline tracking and calendar synchronization.