UTD Event Hub, Team 16
CS 3354 – Software Engineering

1. Team Profile

- **Swarna Sre Ganesh Shankar—**Backend Development (Data Scraping, APIs, Database)
- **Ebrahim Khan**– Frontend Development (React/Flutter UI)
- **Harsha Gundavelli**– Web Scraping & Meta API Integration
- Project Management & Documentation by the team

2. Introduction

UTD students and faculty struggle to keep up with campus events due to scattered event postings across Instagram, Facebook, and other platforms. With no single source of event information, it becomes difficult to stay informed. This leads to missed opportunities and lower event engagement.

UTD Event Hub is a centralized web/mobile app that aggregates event data from these platforms and presents them in a unified, user-friendly interface. It will allow users to easily browse and search for events, ensuring they never miss out.

3. Objectives

- Provide a **one-stop platform** for UTD events.
- Automate event collection using **Facebook & Instagram scraping/API**.
- Categorize events by **tags**: “Free Food,” “Tabling,” “Activities,” and “Major-Specific.”
- Allow users to **bookmark, filter, and get event reminders**.
- Offer a **searchable event calendar** with personalized recommendations.

4. Target Customers

- **UTD Students** – Stay updated on campus events.
    - Manage schedules
    - Find relevant activities
- **UTD Faculty & Organizations** – Increase event attendance by reaching a wider audience.
    - Schedule events

5. Value Proposition

- **Saves Time**: No need to manually check multiple platforms.
- **Better Event Discovery**: Users find events based on interests (e.g., free food, major).
- **Higher Engagement**: Clubs/organizations get more visibility for their events.
- Simplified Event Management: One-stop platform for scheduling and promoting events without relying solely on social media
- Data & Insight: Dashboard to track even engagement, attendee demographics, and popular event categories

6. Application Features

1. **Event Aggregation**
    - Scrape data from Facebook/Instagram pages (Meta API or web scraping).
    - Collect details: event name, date, location, description, images, and RSVP links.
        - Additional data: UTD official websites, Google Calendar API (if available)
    - Store in a structured database.
    - Events categorized by: date, type, and host organization
    - Duplicate event detection to prevent redundancy
2. **User Dashboard & Event Calendar**
    - Display events in a searchable and filterable list/calendar view.
    - Provide view of events with RSVP options and details
    - Users can bookmark and receive event reminders.
3. **Tags & Filters**
    - Events categorized with tags: "Tabling," "Free Food," "Activities," etc.
    - Filter events by category, date, or relevance.
4. **Personalized Recommendations**
    - Suggest events based on user preferences (e.g., past events attended, major).
    - Show most attended and highly popular events to encourage participation
5. **Event Submission & Admin Features**
    - Allow organizations to submit events manually for approval.
    - Admin panel for event editing, approval, and removal
    - Flagging/reporting system for duplicate or inappropriate events

7. Tools and Resources

- **Front;**
- **end:** React.js / Next.js (Web), Flutter / React Native (Mobile).
- **Backend:** Python (Flask/FastAPI) or Node.js.
- **Database:** PostgreSQL / Firebase.
- **Scraping Tools:** Selenium, BeautifulSoup, Scrapy (if Meta API is restrictive).
- **Hosting:** AWS / Firebase / Vercel.

For the frontend, React.js is a JavaScript library designed for building fast, interactive web interface and combining it with Next.js enhances these with features like server-side rendering and static site generation. For mobile development, Flutter and React Native both allow for the building of cross-platform apps with a single codebase. Flutter uses Dart, which provides a rich UI experience, whereas React Native is built in JavaScript, which will integrate well with existing web technologies.

For data storage, PostgreSQL is a relational database that ensures structured and reliable data management, which will make it well-suited for storing event details, user interactions, and recommendations. Firebase is a cloud-based platform that provides real-time syncing, making it useful for live updates or notification features.

Since the Meta API may have restrictions that limit access to event data, data scraping will have to be an alternative option in gathering this data. In this case, Selenium, BeautifulSoup, and Scrapy can be used.

For hosting, a popular option is AWS, which is a powerful cloud platform that can handle large-scale applications. Firebase is an all-in-one solution that has built-in features like authentication and real-time updates that can simplify the development process. Vercel is designed for frontend applications, ensuring fast load times and a smooth user experience.

8. Challenges & Solutions

- **Meta API Restrictions:** One of the main obstacles are these restrictions, so research will be conducted on the API permissions for event data access. If necessary, use web scraping with anti-detection measures.
- **Data Accuracy:** Implement deduplication & verification to prevent outdated event listings using event metadata such as title, date, location, club, and any other unique identifiers.
- **User Engagement:** Introduce event reminders and personalized recommendations to boost retention. Additionally, push notifications and calendar can be implemented to remind users about upcoming events.

9. Plan of Work

Week 1-2 – Research Meta API, set up scraping prototype, finalize tech stack. This phase will start with research into the Meta API, and if necessary setting up a data scraping prototype. Simultaneously, a technology stack will be finalized to ensure that the most efficient tools are selected for development.
Week 3-4 – Develop backend API for event storage and retrieval. The focus will shift to the backend where API will be created for event storage and retrieval. API integrations for gathering

event data will also be set up.
Week 5-6 – Implement frontend UI for event browsing and filtering. These weeks will be dedicated to the frontend, where the UI will be developed for event browsing, filtering, and searching.
Week 7-8 – Integrate notifications, refine user experience, and beta testing. Additional features such as push notifications and other user engagement tools will be integrated. Beta testing will also begin with the simultaneous refinement of user experience.
Week 9 – Final testing, deployment, and launch. To finish, the app will go through final testing, ensuring there are no bugs and is in proper working order. Once this is complete, the app will be launched.