SOFTWARE PROJECT PLAN: THE HIVE

Group Name: Fantastic 4
Members:
Andaç Bilgili (Team Lead/UX)
Gülşah Öykü Kırlı (DevOps & Testing)
Mustafa Efe Arslan (Frontend Developer)
Yiğit Aydoğan (Backend Developer)
Course Information: SW Engineering, Project Management

Report Name: A platform to find all ITU events in one place
Date: 14/10/2025

Contents

1.0 INTRODUCTION

The goal of The Hive is to address the lack of a single, campus-wide event announcement platform at ITU by creating a simple **web application** where students can find all upcoming school events in one place. This is intended to support student engagement and participation in campus life.

1 Project Scope

The Hive is a web application designed to be the central hub for all ITU events.

- **Major Inputs:**
  - Event data entered manually by authorized student clubs and administrators via a secure form.
  - Event data gathered **automatically** from the **public Instagram accounts of registered university clubs**, via a lightweight scraping component that extracts essential event information (title, date, time, location, link) from approved posts.
- **Processing Functionality:** Data validation, storage in a centralized database, automated ingestion of Instagram events into a review queue, and scheduling of user reminders.
- **Outputs:** A responsive web application that shows events in a chronological list view and a calendar view, detailed event pages, and a simple interface for setting event reminders. Events may come from either manual input or the automated Instagram pipeline.

2 Major Software Functions (Functional Decomposition)

The main software functions are:

1. **Data Management:** Secure club/admin login and manual event creation/editing (T6).
2. **Automated Event Aggregation (Instagram Scraper):** Periodic collection of basic event metadata from the **public Instagram accounts** of whitelisted ITU clubs using a scheduled backend worker (T10), with scraped candidates stored in the same event database for validation and publishing.
3. **Information Display:** Presentation of events in list view (T1), detail view (T2), and calendar view (T3).
4. **User Interactivity:** Search and filtering (T8), reminders (T4), and *Add to Calendar* exports (T5).
5. **Core Technology:** A responsive application that works across common devices (T9).

3 Performance/Behavior Issues

- **Performance:** The main event list (T1, T3) must behave consistently across platforms, even as new events are automatically ingested from Instagram.

- **Usability:** The interface must be intuitive for students, and the data entry process (T6) must be simple enough to encourage regular use by student clubs. Automatically scraped events should be clearly labeled and easy for authorized admins to review and confirm.

- **Data Freshness and Robustness:** The Instagram scraping component (T10) will run at a controlled frequency (for example, a few times per day for selected clubs) and must handle failures gracefully (such as temporary Instagram changes or rate limits), with a simple fallback to manual entry when needed.

- **Availability:** The application should maintain more than **90%** uptime during the academic term.

4 Management and Technical Constraints

- **Management Constraint:** The project has a firm delivery date and a total duration of **11 calendar weeks from now**. The Instagram-based automated event gathering feature is planned as a **minimal pilot implementation** (limited number of clubs, basic metadata only) so that it fits within the current schedule.

- **Technical Constraint:** The application will be developed using a modern, open-source technology stack (e.g., Python/Django or Node.js/Express for the backend, and React/Vue for the frontend) to keep licensing costs at zero and make future maintenance by the university realistic.

- **Data and Platform Constraint:** The Instagram scraping functionality will only use **publicly available data** from official club accounts, respect platform terms of use, and rely on standard open-source scraping and automation libraries (e.g., Playwright or Selenium-style tools) rather than paid APIs.

2.0 PROJECT ESTIMATES

This section summarizes cost, effort, and schedule estimates for the project.

1 Historical Data Used for Estimates

Estimates are mainly based on **analogy estimation** from previous academic software projects of similar size and complexity. The Hive is categorized as a **medium-sized** web application because it includes secure authentication, interactive front-end views (including a calendar view), and a **small automated Instagram event gathering component** integrated into the backend.

2 Estimation Techniques Applied and Results

2.2.1 Estimation Technique 1: Work Breakdown Structure (WBS) Sizing

The project is decomposed into 20 subtasks. Each subtask is assigned a size (T, S, M, L, XL), which is mapped to an estimated effort in $\text{man} \cdot \text{week}$ based on historical team productivity ($1\ \text{man} \cdot \text{week} \approx 40$ person-hours). The Instagram scraping feature (T10) is scoped as a **small-to-medium backend task** that reuses the existing infrastructure (database, API, deployment pipeline) and open-source libraries.

Size Category	Man-Weeks ($\text{man} \cdot \text{week}$)
Tiny (T)	0.1
Small (S)	0.5
Medium (M)	1.0
Large (L)	2.0
Extra-Large (XL)	3.0

2.2.2 Estimate for Technique 1: Total Effort

The sum of the estimated effort for all WBS subtasks (from the original WBS breakdown) is:

$$\text{Total Effort Estimate} = \mathbf{25.0}\ \text{man} \cdot \text{week}$$

The Instagram scraping component is included in this total by slightly reallocating effort from non-critical buffer tasks and by keeping the automation scope limited (a small set of club accounts and basic event fields).

3 Reconciled Estimate

The final estimate for the project, based on the WBS, is:

- **Effort: 25.0** $\text{man} \cdot \text{week}$ (including a minimal Instagram scraping pipeline).
- **Time (Duration): 12** weeks (fixed constraint, unchanged).
- **Cost: 0** (limited to student effort and free tools, as stated in the project charter).

4 Project Resources

Category	**Item/Tool**	**Requirement/Purpose**
People	Andaç, Efe, Öykü, Yiğit	4 dedicated team members (25.0 man · week total).
Hardware	Developer PCs/Laptops	Standard development environment for each member.
Software	Backend Framework (e.g., Python/Node.js)	Core logic and API development, including the Instagram scraping worker and ingestion endpoints.
Software	Frontend Framework (e.g., React/Vue)	UI development and responsive views.
Tools	Git/GitHub	Version control and collaborative development.
Tools	PostgreSQL / MongoDB	Persistent storage for both manually entered and automatically scraped events.
Tools	Cloud Hosting (e.g., AWS)	Production and staging deployment environment.
Tools	Web Automation/Scraping Library (e.g., Playwright or Selenium-style)	Periodic automated retrieval of event data from public Instagram posts of registered club accounts.

3.0 RISK MANAGEMENT

This section discusses project risks and how they will be handled.

1 Project Risks (CTC Format)

1. **Lack of Club Adoption/Data: Condition:** Clubs do not use the platform at first due to inertia or lack of training. **To get:** The database remains sparse. **Consequence:** The platform fails to reach the target of "90% of events listed" and loses value for students.

2. **Integration Issues (BE/FE): Condition:** The contract between the backend (Yiğit) and the frontend (Efe) endpoints is not well aligned. **To get:** Significant delays around Week 7 (WBS 4.1). **Consequence:** The testing phase (WBS 4.3) starts late, reducing final quality.

3. **Instagram Scraper Breaks / Policy Changes: Condition:** Instagram UI, limits, or terms of use change in a way that breaks or restricts the scraping component (T10). **To get:** Automated event collection partially or fully stops working. **Consequence:** The platform falls back to manual data entry only, and the automated gathering promise is weakened.

2 Risk Table

Name of Risk	Probability	Impact	Risk Score	RM3
Lack of Club Adoption/Data	4	4	16	Aversion 5
Integration Issues (BE/FE)	4	3	12	Aversion 4
Scope Creep	3	4	12	Aversion 3
Key Developer Sick/Unavailable	2	5	10	Aversion 1
Instagram Scraper Breaks / Policy Changes	3	3	9	Aversion 3

3 Overview of Risk Mitigation, Monitoring, Management (RM3)

The RM3 strategy emphasizes **Aversion** for risks with both high probability and high impact. The team will **monitor**:

- Weekly API completion rates.
- Number of clubs onboarded.
- Success rate of Instagram scraping runs (for example, the percentage of posts parsed successfully).

Mitigation actions include:

- For lack of club adoption, actively onboarding a core set of clubs and providing short training on the manual entry interface. In parallel, the Instagram scraping feature (T10) offers an **automatic baseline of events**, reducing dependence on manual input.

- For scraper-related risks, keeping the implementation small and isolated in a separate backend worker. If Instagram changes, the system can **fall back to manual event entry** without affecting the rest of the platform. Any major extension of scraping to other platforms or more complex parsing will go through the formal change request process.

Management also includes using redundancy where possible and enforcing a simple change request process to avoid uncontrolled scope growth. The Instagram scraping feature is treated as a clearly bounded in-scope feature; later extensions (for example, more clubs or additional social networks) will be handled as separate change requests.

4.0 PROJECT SCHEDULE

This section presents the overall project tasks and the output of the scheduling effort.

1 Project Task Set

The project follows an **iterative and incremental** process model. The main activities are: planning, requirement gathering, design, construction, testing, and deployment. The task set is derived from the WBS (Section 2.0). The **Instagram scraping component (T10)** is part of backend construction and DevOps tasks, and is scoped as a basic, working pipeline for a small number of club accounts so that the overall duration **remains 12 weeks**.

2 Functional Decomposition (for Scheduling)

The functional breakdown used for scheduling is described in the WBS. Tasks are decomposed to a level where assignment, estimation, and tracking are practical (for example, "build responsive list component" is an assignable subtask; "implement minimal Instagram scraping worker for 3 pilot clubs" is another).

3 Task Network (Pert Chart Overview)

Key dependencies:

- Planning → Design → Core Construction
- Database Design (1.4) → Core API Setup (2.1)
- Core API Setup (2.1) → Instagram Scraping Worker (2.2/T10) (the scraper writes into the same event storage)
- UI/UX Design (3.1) → All Frontend Construction (3.2, 3.3, 3.4, 3.5)
- All Construction Tasks (2.4, 2.2/T10, 3.3, 3.5) → Integration (4.1)
- Integration (4.1) → Unit and Integration Tests (4.3) → Bug Fixing (5.1)

Critical Path (Longest Path): 1.4 (DB Design) → 2.1 (Core API) → 4.1 (Integration) → 4.3 (Testing) → 5.1 (Bug Fixing) → 5.3 (Deployment).

The Instagram scraping worker (T10) is designed to stay **off the critical path** by relying on already existing APIs and by being completed within the backend construction buffer. **Total duration: 12 weeks (unchanged).**

4 Timeline Chart (Gantt Chart Overview)

The project is structured into 12 weeks. Workload is leveled so that no team member is assigned more than 1.0 man · week of effort per calendar week.

Phase	**Duration**	**Key Deliverables (Mapped to Templates)**
Planning & Design	Week 1–3	1 - Project Plan Template.doc
MVP Construction	Week 3–6	Initial features integrated (core event CRUD, basic l
Feature Construction	Week 6–9	3 - Design Specification Template.doc; improved filte
Integration & Testing	Week 9–10	4 - Test Specification Template.doc, including tests f
Stabilization & Deployment	Week 11–12	2 - Requirements Specification Template.doc (finaliz

5.0 STAFF ORGANIZATION

This section explains how the staff are organized and how reporting works.

1 Team Structure

The team follows a **Chief Programmer Team (CPT)** structure with clearly defined roles:

- **Andaç Bilgili (Team Lead/UX):** Chief programmer and manager. Responsible for overall project cohesion, requirements, and user experience/design. Also coordinates which clubs are included in the Instagram scraping pilot.
- **Yiğit Aydoğan (Backend Developer):** Technical expert for backend, database, and API. Responsible for implementing the Instagram scraping worker (T10), event ingestion endpoints, and related backend logic.
- **Mustafa Efe Arslan (Frontend Developer):** Technical expert for UI implementation and client-side logic, including clearly indicating auto-scraped versus manually entered events in the UI when needed.
- **Gülşah Öykü Kırlı (DevOps & Testing):** Quality assurance and tools engineer. Responsible for testing, deployment pipeline, scheduling the scraping process (for example with cron/CI), and monitoring scraper health.

2 Management Reporting and Communication

- **Progress Reporting:** Weekly status meetings (30 minutes) held every Wednesday morning after the ISE308 lecture. Each member reports on the WBS tasks completed in the previous week and planned for the current week, including any risks or blockers. Progress on the Instagram scraping feature (T10) is discussed briefly during the construction phase.
- **Management/Customer Reporting:** Andaç (Team Lead) is the main point of contact for the course instructor and the "customer" (ITU administration/student body), and is responsible for delivering the formal reports (Templates 1, 2, 3, and 4).

6.0 TRACKING AND CONTROL MECHANISMS

This section describes how the project will be tracked and controlled.

1 Quality Assurance and Control

The main SQA mechanisms are **peer review** and **dedicated testing** (WBS 4.3, 4.4).

- **Code Review:** Every commit merged into the main development branch requires review by at least one other team member. This also applies to changes in the Instagram scraping worker and ingestion pipeline to avoid brittle or hard-to-maintain scraping logic.
- **Testing:** Öykü (DevOps/Testing) ensures that all major features have matching unit and integration tests before deployment. This includes:
  - Manual event creation, editing, and display.
  - A basic end-to-end flow for auto-scraped Instagram events: scraping sample posts from a test account, ingestion, storage, and display.

  A formal user acceptance test (UAT) with 20 students will act as the final quality gate. Results will be documented in the **Test Specification Template**. UAT will explicitly check whether "events from selected club Instagram accounts appear correctly and on time".

2 Change Management and Control

The project uses a simple **change management** process led by the Team Lead (Andaç).

1. **Change Request (CR):** Any request for a feature that is not in the current scope (Section 1.1) must be submitted as a change request. This includes extending scraping to other social platforms (e.g., Twitter/X, Facebook) or adding complex NLP-based event extraction.
2. **Impact Analysis:** The team (Andaç, Efe, Yiğit) estimates the extra effort and the impact on the 12-week schedule.
3. **Approval/Rejection:** The change is approved only if a feature of equal or greater effort is removed from scope, or if the change is minor (less than one day of effort) and approved unanimously by the team.

The current plan already includes a **minimal Instagram-based automated event gathering feature**, and this is treated as in-scope and compatible with the existing 12-week timeline and 25.0 man · week effort estimate.