SOFTWARE REQUIREMENTS SPECIFICATION

Project: THE HIVE

Prepared by Group: Fantastic 4
Andaç Bilgili · Gülşah Öykü Kırlı · Mustafa Efe Arslan · Yiğit Aydoğan

Date: 07/12/2025 · Course: Software Engineering

Contents

1.0 Introduction

This section provides an overview of the entire requirement document. This document describes all data, functional, and behavioral requirements for the software "The Hive".

1.1 Goals and objectives

The primary goal of The Hive is to create a simple web application that serves as a central hub for all ITU events, addressing the lack of a single campus-wide announcement platform. The specific objectives are:

- To support student engagement and participation in campus life.
- To provide a consolidated list and calendar view of all upcoming school events.
- To allow authorized clubs to manually input event data easily.

1.2 Statement of scope

A description of the software is presented. Major inputs, processing functionality, and outputs are described without regard to implementation detail.

- **Inputs:**
  1. Manual data entry by student clubs via secure forms.
  2. Automated scraping of public Instagram accounts of registered clubs to extract basic event metadata (title, date, time).
- **Processing:** Data validation, centralized database storage, review queue management for scraped events, and notification scheduling.
- **Outputs:** A responsive web interface displaying chronological lists, detailed event pages, calendar views, and reminder notifications.

1.3 Software context

The software is placed in the context of university campus life. It acts as a student-facing utility designed to aggregate fragmented information sources (social media, bulletin boards) into a single product line. The intent is for the reader to understand the big picture.

1.4 Major constraints

Any business or product-line constraints that will impact the manner in which the software is to be specified, designed, implemented, or tested are noted here.

- **Schedule:** The project has a firm total duration of 11 calendar weeks.
- **Technology:** Must use open-source stacks to ensure zero licensing costs.

- **Data Policy:** The automated component must only access publicly available data and respect platform terms of use.
- **Reliability:** The scraper must handle failures gracefully, falling back to manual entry if necessary.

2.0 Usage scenario

This section provides a usage scenario for the software. It organizes information collected during requirements elicitation into use-cases.

2.1 User profiles

The profiles of all user categories are described here.

- **Student (Viewer):** Browses the calendar, filters events, and sets reminders.
- **Club Administrator:** Enters event details via secure login.
- **System Administrator:** Maintains Instagram whitelist and reviews/approves scraped events.

2.2 Use-cases

All use-cases for the software are presented.

- **UC-01 Manual Event Creation:** Authenticated Club Admins enter event details (T6).
- **UC-02 Automated Aggregation:** System scrapes Instagram posts to create draft events (T10).
- **UC-03 Event Discovery:** Students search and filter events by category/date (T8).
- **UC-04 Reminder Setup:** Students set reminders for events (T4).

2.3 Special usage considerations

Special requirements associated with the use of the software are presented.

- **Mobile responsiveness:** Must be usable on mobile devices.
- **Scraper fallback:** If scraping fails, admins can switch to manual entry without interruption.

3.0 Data Model and Description

This section describes the information domain for the software.

3.1 Data Description

Data objects that will be managed/manipulated by the software are described in this section.

3.1.1 Data objects

The data model consists of three primary entities: **Club**, **Event**, and **Student**.

3.1.2 Relationships

Relationships among data objects are described using an ERD-like form. No attempt is made to provide detail at this stage.

- A **Club** organizes many **Events** (1:N).
- A **Student** can save/set reminders for many **Events** (M:N).
- An **Event** can be sourced manually or via **Scraper**.

3.1.3 Complete data model

An ERD for the software is developed.

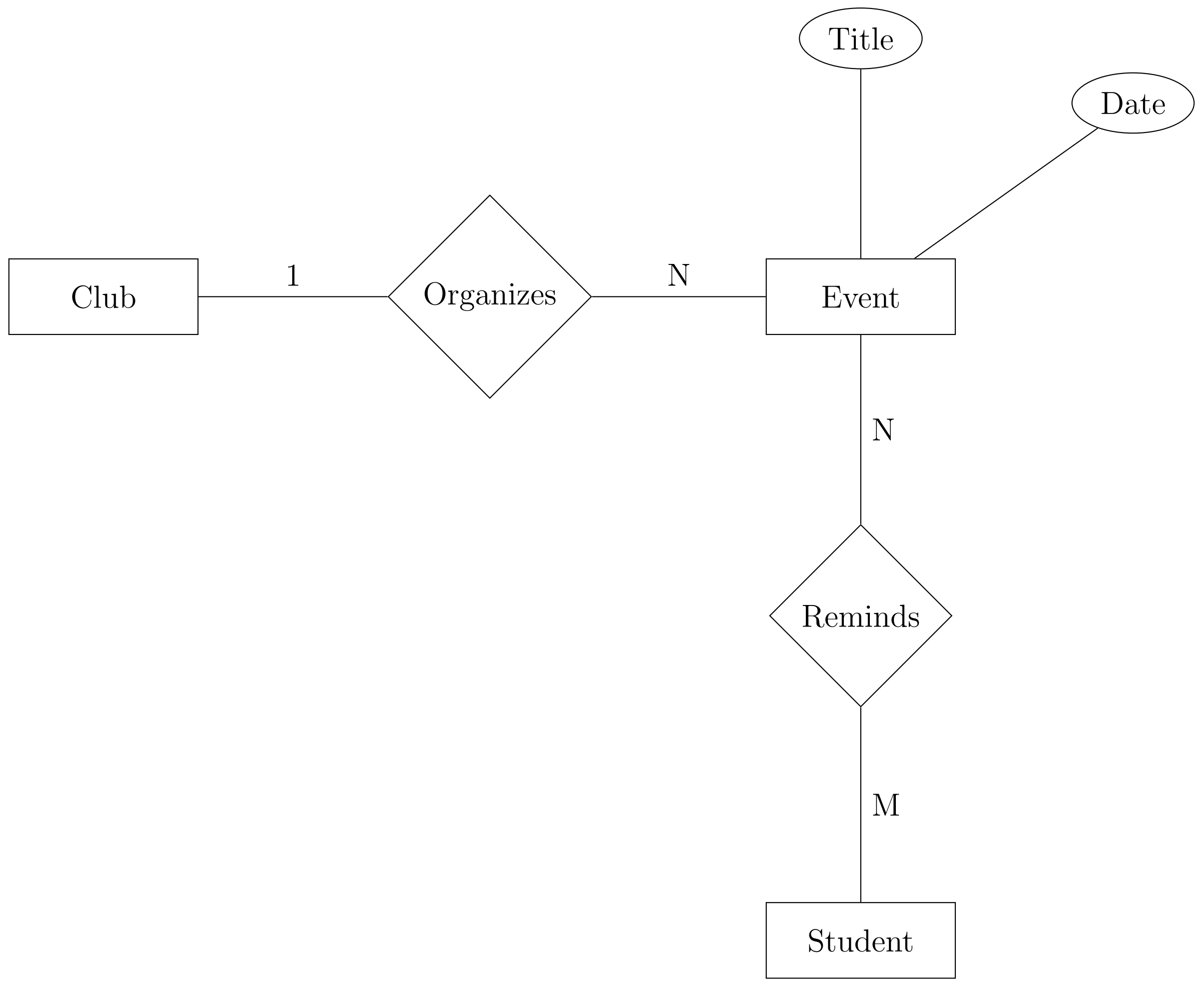

Figure 1: Entity Relationship Diagram (ERD)

3.1.4 Data dictionary

A reference to the data dictionary is provided. The dictionary is maintained in electronic form.

Data Item	**Description**	**Format**
Event_ID	Unique identifier for an event	Integer (PK)
Event_Source	Origin of the data	[Manual - Scraped]
Event_Status	Publishing status	[Draft - Review - Published]
Club_Instagram	Public URL for scraping	String (URL)

4.0 Functional Model and Description

A description of each major software function, along with data flow or class hierarchy (OO) is presented.

4.1 Description for Function 1: Event Discovery

This function allows the primary user (Student) to view, search, and filter the aggregated list of events. It acts as the primary query interface for the system.

4.1.1 Processing narrative (PSPEC) for function 1

The student initiates a search request by visiting the home page. The system retrieves the list of "Published" events from the database. The student may refine this list by applying filters (Category, Date) or entering a search keyword. The system validates the search string, queries the database for matching records, sorts them chronologically, and returns the list to the user interface.

4.1.2 Function 1 flow diagram

A diagram showing the flow of information through the function is presented.

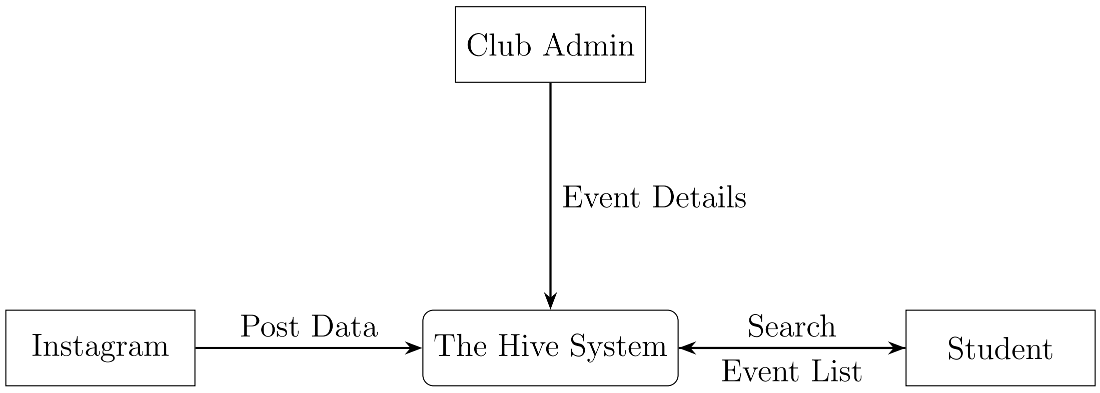

Figure 2: Level-0 Context DFD

4.1.3 Function 1 interface description

- **Input:** Search String (Alphanumeric), Date Range (Start Date, End Date), Category ID.
- **Output:** JSON array of Event objects containing Title, Location, Date, Time, and Club Name.

4.1.4 Function 1 transforms

The search function relies on three logical transforms:

1. **Input Parsing:** Cleaning the user search string.
2. **Query Execution:** Fetching events where status is 'Published' AND matches filters.
3. **Formatting:** Converting database rows into front-end consumable JSON.

4.1.5 Performance issues

The search query must return results in under 2 seconds for a database size of up to 10,000 events. The system must maintain consistency even while the Instagram scraper is ingesting new data.

4.1.6 Design constraints

The UI must be responsive to support mobile browsers, as 80% of traffic is expected from smartphones.

4.2 Software Interface Description

The software interfaces to the outside world are described.

4.2.1 External machine interfaces

The software resides on a cloud server (e.g., AWS/DigitalOcean) and interfaces with client devices (Mobile/Desktop) via standard HTTP/HTTPS protocols over TCP/IP (Hypothetically, we arent going to do this FYI).

4.2.2 External system interfaces

- **Instagram (Target):** The scraper interacts with public Instagram web pages.
- **Database Server:** The application connects to a PostgreSQL instance for persistent storage.

4.2.3 Human interface

The human interface is a web-based GUI. It features a navigation bar, a central card-based feed for events, and a secure dashboard for Club Admins. The design prioritizes scannability and ease of use for students on the go.

4.3 Control flow description

The system acts as a transaction-based system where user requests (Search, Login, Submit) trigger specific processes. Detailed control logic regarding event lifecycle management is described in Section 5.0.

5.0 Behavioral Model and Description

A description of the behavior of the software is presented.

5.1 Description for software behavior

The software behavior is driven by the lifecycle of an "Event" object. The system behavior changes based on the user's role (Admin vs. Student) and the state of the data.

5.1.1 Events

The following control items trigger behavioral changes:

- **Submit Event:** Triggered by Manual Entry or Scraper Success.

- **Approve Event:** Triggered by System Admin or an LLM.
- **Reject Event:** Triggered by System Admin.
- **Date Passed:** Triggered by System Clock.

5.1.2 States

- **Draft/Scraped:** Initial state for new data.
- **Pending Review:** Events waiting for admin validation.
- **Published:** Events visible to Students.
- **Archived:** Past events, hidden from the main feed.

5.2 State Transition Diagrams

Depict the overall behavior of the system.

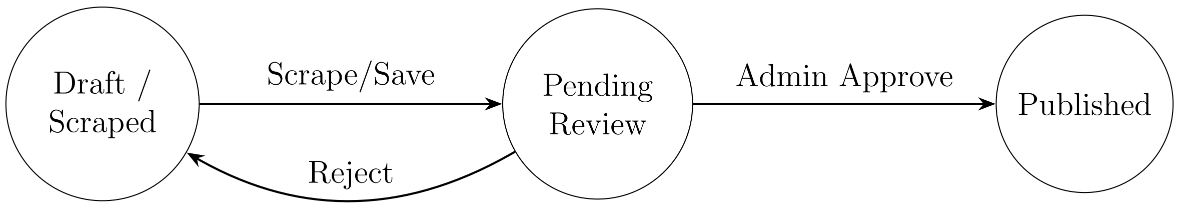

Figure 3: State Transition Diagram for Event Object

5.3 Control specification (CSPEC)

The control logic dictates that only "System Admins" and/or an LLM with admin permissions can transition an event from "Pending Review" to "Published." The Scraper automates the transition from "Null" to "Draft/Scraped." A background worker automates the transition from "Published" to "Archived" once the event end-time is reached.

6.0 Restrictions, Limitations, and Constraints

Special issues which impact the specification, design, or implementation of the software are noted here.

- **Time Constraint:** The project must be completed within 11 weeks.
- **Platform Dependency:** The automated feature is dependent on Instagram's HTML structure remaining consistent; changes may require maintenance patches.
- **Budget:** Zero financial budget; relies entirely on free tier cloud services and open-source libraries.

7.0 Validation Criteria

The approach to software validation is described.

7.1 Classes of tests

- **Unit Testing:** For backend scraping logic and data parsing.
- **Integration Testing:** Ensuring the Frontend correctly displays data fetched from the API.
- **User Acceptance Testing (UAT):** A pilot group of 20 students will verify if they can find events easily.

7.2 Expected software response

The software must handle invalid inputs (e.g., end date before start date) with clear error messages. Scraper failures must log errors without crashing the web server.

7.3 Performance bounds

- **Uptime:** 90% availability during the academic term.
- **Concurrent Users:** Support at least 50 simultaneous users without degradation.

8.0 Appendices

Presents information that supplements the Requirements Specification.

8.1 Product Strategies

The product follows an "MVP First" strategy, prioritizing the manual entry and display features before refining the automated Instagram scraper.

8.2 Analysis metrics to be used

- **Function Points:** Estimated size of the application.
- **Defect Density:** Number of bugs found per KLOC during testing phases.

8.3 Supplementary information (as required)

Assumption: Student clubs will agree to make their Instagram profiles public for the scraper to function.