U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

[M.EGI017]
Enterprise Information Systems

TeamWork 2025/2026

Sprint 2

TeamWork 3

Team

Afonso Dias Fachada Ramos [202108474]
Ana Isabel Dias Cunha Amorim [202107329]
Filipa Marisa Duarte Mota [202402072]
Matheus Fernandes Vilhena Campinho [202202004]

Supervisor(s):

Prof. Telmo Manuel Sampaio Pinto de Matos

Course: M.EGI

2025/2026 - 1S

Project Overview

Project Name: GoodCollections - Collections Management Information System

Objective: Develop a web-based information system that allows collectors to manage their collections, items, and events with persistent data storage in a relational database.

Sprint 2: Building on Sprint 1's interface design, Sprint 2 implemented a dynamic system using PHP and MySQL. The result is a fully functional web application that manages collections, items, events, and users with persistent storage."

Development Environment:

- **Server:** XAMPP (Apache 2.4 + PHP 8.0.30)
- **Database:** MySQL 10.4.32 (MariaDB)
- **Client:** Modern browsers (Chrome, Safari, and Edge)
- **Version Control:** Git + GitHub repository
- **IDE/Editor:** NetBeans, and VS Code

Task Description

UML Use Case Diagram

A **Use Case Diagram** provides a high-level view of how users interact with the system. It identifies:

- **Actors** (e.g., Visitor, Collector)
- **Use cases**, representing the main actions users can perform (such as browsing collections, liking items, editing content, confirming event attendance, etc.)
- **Relationships** between actors and use cases, showing who can do what

This diagram helps clarify the system's functional scope from the user's perspective. It is especially useful during requirements definition and sprint planning, as it highlights the core interactions expected in the final system.

Figure 1 illustrates the Use Case Diagram developed for this project, offering an overview of how the platform should behave and how different users are expected to interact with its features.

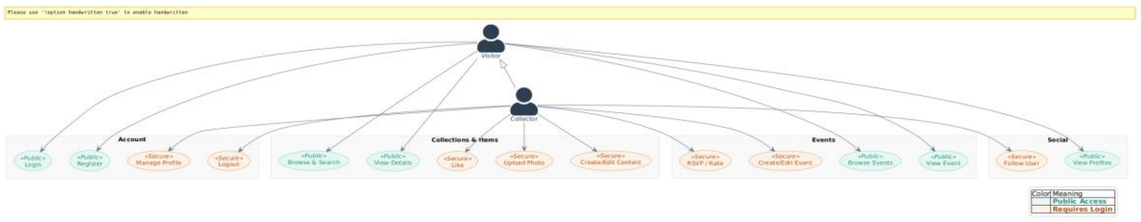

Figure 1 - UML Use Case Diagram.

MySQL Database (phpMyAdmin)

All information is now stored persistently in a relational database. A MySQL database (version 10.4.32) named **sie_db** was created to support the storage of all relevant entities, including collections, items, events, and associated users (collectors), among other data required by the system.

With this database in place, the web portal developed during the first phase of the project (Sprint 1) becomes fully dynamic. It is now possible to **view, insert, modify, and delete (CRUD)** all information stored in the database, ensuring that the platform reflects real-time updates made by users.

Server-side communication is handled using **PHP**, which manages requests, processes user actions, and interacts with the database to retrieve or update information as needed.

The conceptual **Entity-Relationship (ER) Diagram** representing the system's data model is shown in **Figure 2**.

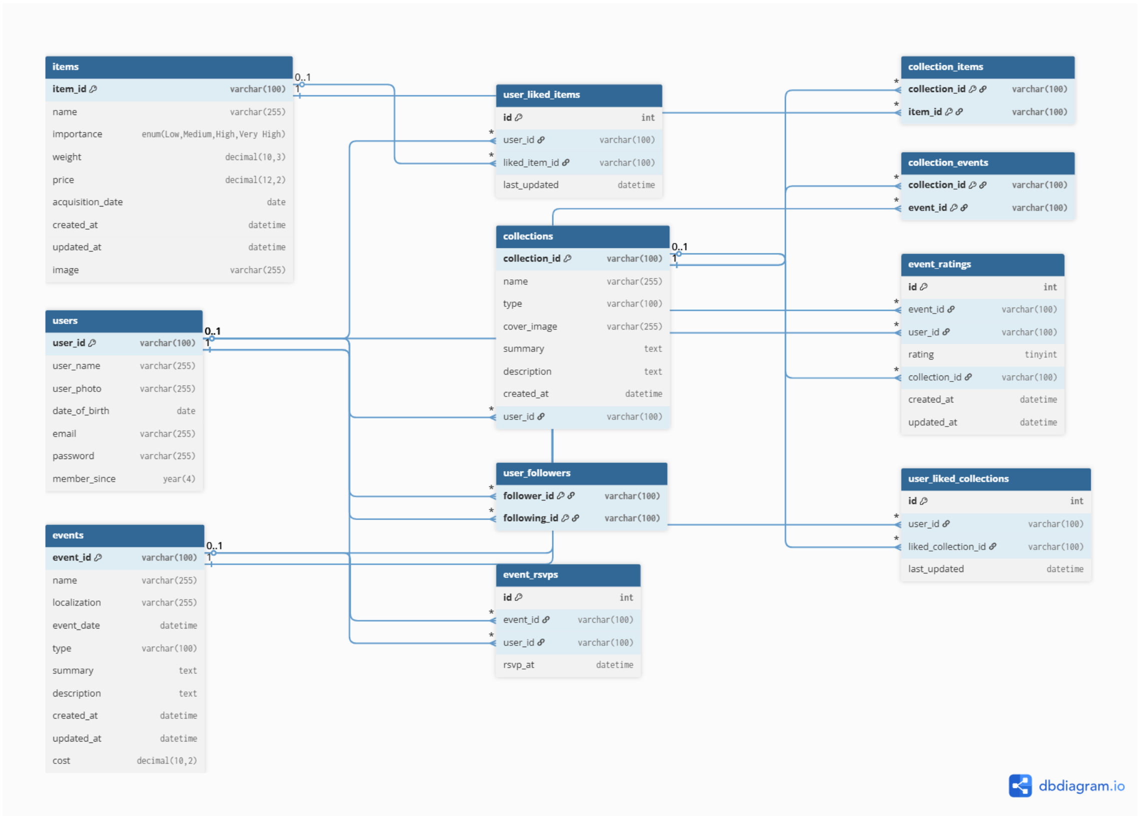

Figure 2 – ER Diagram.

Key Features Implemented

Core CRUD Operations

- **Collections:** Create, Read, Update, Delete collections
- **Items:** Create, Read, Update, Delete items; assign to multiple collections
- **Events:** Create, Read, Update, Delete events; link to collections
- **Users:** Register users, manage user information
- **Ratings:** Like/rate events

Authentication & Authorization

- **Session-Based Authentication:** Users sign in and maintain session state
- **Password Security:** Passwords hashed with bcrypt (PASSWORD_DEFAULT)
- **Ownership Verification:** Users can only modify their own data
- **Permission Checks:** Ownership checks in all CRUD endpoints before modifications
- **User Registration:** New users can create accounts through web interface

Data Relationships

- **One-to-Many (Users → Collections):** Each user owns multiple collections
- **Many-to-Many (Collections ↔ Items):** Items shared across collections without duplication
- **Many-to-Many (Collections ↔ Events):** Events linked to multiple collections
- **One-to-Many (Users → Events)**: Users host/attend events

REST & HTTP Compliance

- **Interaction model:** Browser form submissions with PHP handlers that validate input, mutate data, set flash messages, and redirect.
- **GET:** Render pages (home, collections, items, events, users) and fetch JSON only in helper endpoints like auth.php (session status/login) and events_alert.php (RSVP alerts).
- **POST:** Handle create/update/delete for users, collections, items, events, likes, follows, ratings, and RSVPs. Responses are redirects with success/error flashes, not JSON bodies.
- **Status codes:** Standard codes on the JSON helpers; form flows use redirects (303/302).
- **Security:** All DB writes use prepared statements to prevent SQL injection.

Data Access Layer (DAL)

- **Centralized Configuration:** PHP/config/db.php opens the MySQL connection used across pages and action controllers.
- **Prepared Statements:** All database writes and most reads use parameterized queries to mitigate SQL injection.
- **Separation of Concerns:** Data access logic isolated from business logic

Task Distribution

Table 1 below presents the distribution of tasks among the team members throughout the Sprint 2 for the project.

Table 1– Tasks distribution.

Task	Afonso Ramos	Ana Amorim	Filipa Mota	Matheus Campinho

Database creation (phpMyAdmin)			O	O
Data Access Layer (DAL)			O	O
REST style with proper HTTP methods and status codes			O	O
User registration and profile editing	O	O		
Authentication system with password hashing			O	O
Search functionality	O	O		
HTML, CSS, and JavaScript adaptation	O	O		

Legend:

X – to be involved in the task
O – to be the main responsible for its accomplishment

Testing & Validation

Test Scenarios Completed:

- ✓ User registration and login
- ✓ Create collection
- ✓ Add items to collection
- ✓ Assign item to multiple collections
- ✓ Update item properties (importance, weight, price)
- ✓ Delete collection (owned by user)
- ✓ Create event linked to collection
- ✓ Rate/like event
- ✓ View collections of other users (read-only)
- ✓ Permission validation (users cannot delete others' collections)
- ✓ Password storage (bcrypt hashing)
- ✓ Session-based access control
- ✓ Ownership verification on write operations

Work Limitations

Feature	Status	Reason
CSV Import/Export	Not Implemented	Time constraints; prioritized core CRUD
Advanced Search Filters	Partial	Basic filters are working, but full implementation was not accomplished through all pages of the website
JavaScript separation (dedicated .js files)	Partial	Some JS stayed inside a few PHP pages. We extracted it, but time was short and it affected other pages.
Back button / navigation consistency	Partial	Back buttons don't always return to the exact previous page/state, especially after redirects or different entry points.
UI consistency across similar forms	Partial	Same CSS is used, but small differences in page-specific styles cause some minor visual inconsistencies.

Observations

Assumptions & Clarifications

- Users must be authenticated to create/modify collections, items, and events
- Non-authenticated users can view all collections and items (read-only access)
- Events are optional related entities; collections can exist without events
- Events can have ratings only after the event date has passed
- Password reset functionality not implemented
- Comments sections not implemented

How to Test the System

Setup:

- Import Database/sie_db.sql into MySQL (phpMyAdmin)
- Serve SIE/ via XAMPP/Apache (htdocs) with PHP enabled.
- Place project files in htdocs/ (XAMPP)
- Start Apache and MySQL
- Navigate to: http://localhost/EIS_2526-G03_MEGI/SIE/PHP/pages/home_page.php

Test Scenarios:

Scenario 1: User Registration & Login

1. Click "Log In" button on navigation bar

2. Click "Create account" link
3. Fill in registration form (name, email, photo, date of birth, password)
4. Click "Create Account"
5. Should redirect to login; log in with new credentials
6. Should see personalized profile page

Scenario 2: Create & Manage Collection

1. Log in with test account
2. Click "Add Collection" button
3. Fill in collection details (name, type, summary, description, image)
4. Click "Create"
5. Should appear in collections list and profile
6. Try to edit/delete collection (should succeed as owner)

Scenario 3: Multi-Collection Item

1. Create or select two collections
2. Add item to first collection
3. Edit item and assign to second collection
4. Verify item appears in both collections' item lists
5. Verify item details link to correct collection

Scenario 4: Events

1. Create event linked to collection
2. Add event name, date, location, associated collections, etc
3. Try RSVP to event (if authenticated user) to upcoming event
4. After due event date, rate event 1-5 stars
5. Verify rating persists

Scenario 5: Permission Validation

1. Log in as User A
2. Create collection and item
3. Log in as User B (or view as anonymous)
4. Verify User B can see User A's collection (read-only)
5. Verify User B cannot edit/delete User A's collection

Self and Hetero Evaluation

All team members completed the **self and hetero evaluation** form according to the Excel file provided on Moodle.

Evaluation	Afonso Ramos	Ana Amorim	Filipa Mota	Matheus Campinho	**Average**
Afonso Ramos	5	5	5	5	**5**
Ana Amorim	5	5	5	5	**5**
Filipa Mota	5	5	5	5	**5**
Matheus Campinho	5	5	5	5	**5**
Average	**5**	**5**	**5**	**5**	–

Legend:

0 – Unacceptable (0%)
1 – Unsatisfactory (25%)
2 – Acceptable (50%)
3 – Good (75%)
4 – Very Good (90%)
5 – Excellent (100%)

Appendices

Database Schema SQL:

See Database/sie_db.sql for complete database implementation

GitHub Repository:

View repository at: https://github.com/mfvc-campinho/EIS_2526-G03_MEGI