**Milestone 1 – Project Specification**

**Pokémon** Database Management System

**Team Members:**

* Michal Salabura
* Bernard Joyce
* Nikita Smiichyk
* Oliver Halpenny
* Hanna Bokariuk

**1. Overview**

The Pokémon Database Management System project was developed to model and manage the world of Pokémon — including Trainers, Pokémon species, Gyms, Regions, and Towns — using MySQL and Java (JDBC). The database ensures logical relationships between entities through normalization and the use of primary and foreign keys.

Although the full database is already near completion, this submission represents the specification stage for CA1 Milestone 1, covering the database design and core schema implementation required by the brief.

**2. Progress Completed**

* **Project Description and Scenario:** Defined the purpose and scope of the Pokémon database *(detailed earlier in the Project Description document)*.
* **Entity & Attribute Identification:** Created a comprehensive list of tables and fields (Trainers, Pokémon, Gyms, Towns, Regions, etc.), including data types and constraints *(detailed earlier in the Project Description document)*.
* **Normalization:** Achieved 3NF to ensure data integrity and reduce redundancy *(detailed earlier in the Project Description document)*.
* **ERD:** Designed a diagram illustrating relationships between entities and keys *(detailed earlier in the Project Description document)*.
* **SQL Schema:** Developed and tested CREATE TABLE statements for all main entities and inserted sample data. All database tables, relationships, and constraints have been created and tested in MySQL.

The MySQL code now includes views, triggers, and sample CRUD operations, which have been validated to meet the project requirements. These elements are attached.

* **Database Triggers**

Several SQL triggers were implemented to automate processes, maintain data consistency, and improve database reliability.  
The triggers include:

**after\_trainer\_delete** – logs deleted trainers into a separate DeletedTrainers table for auditing and traceability.

**delete\_trainer\_pokemon** – removes all Pokémon belonging to a trainer before their record is deleted, preventing orphaned data.

**change\_leader\_if\_deleted** – automatically reassigns a gym leader if the current leader is deleted, ensuring every gym remains managed.

**check\_pokemon\_level** – enforces valid Pokémon levels between 1 and 100 upon insertion.

**cap\_max\_iv** – caps the Pokémon’s IV (individual values) to a maximum of 31, ensuring balanced and realistic statistics.

These triggers collectively preserve data integrity and automate essential maintenance tasks, reducing manual intervention and preventing logical inconsistencies within the Pokémon Database.

* **Indexes**

Several indexes were created to enhance database performance, especially during frequent join operations:

**index\_wildpokemon\_region** – improves joins between WildPokemon and Region, speeding up regional Pokémon queries.

**index\_wildpokemon\_pokemon** – optimizes lookups involving Pokemon and WildPokemon tables.

**idx\_pokemontypes\_pokemon\_id** and **idx\_pokemontypes\_type\_id** – enhance joins between Pokemon and Types, ensuring faster retrieval of Pokémon type information.

Indexes were verified using the EXPLAIN command, confirming their active use by the MySQL query optimizer.  
Together, these indexes contribute to faster data access and improved overall database efficiency.

* **Views**

Three key SQL views were developed to simplify data retrieval and improve the usability of complex relational data:

**TrainerPokemonStats** – calculates real-time HP, Attack, and Defense for each caught Pokémon based on its base stats, IVs, and level.

**PokemonWithTypes** – aggregates multiple Pokémon types (e.g., Water/Ice) into one row using GROUP\_CONCAT(), simplifying presentation and reporting.

**TrainerSummary** – provides an overview of each trainer, including the total number of Pokémon owned, gym leadership status, hometown, region, and gym details.

These views reduce query complexity, optimize performance, and make it easier to generate analytical and user-facing reports.

Although most of the project’s SQL implementation has been completed ahead of schedule (including CRUD queries, triggers and views), this submission focuses on the core specification required for Milestone 1 (50% progress).

**3. Work Planned for Milestone 2**

* Refinement of advanced SQL features (,stored procedures, transactions).
* Integration with Java application through JDBC for data retrieval and updates.
* Expansion of test data and CRUD operations.
* Development of presentation materials and reflective documentation.

**4. Summary**

The Pokémon Database project has surpassed the initial Milestone 1 target and currently stands at approximately 90–100% completion. This document summarizes the work completed as per the Milestone 1 requirements, including the project specification, database schema, and normalization stages. Remaining tasks are final testing, presentation, and reflective documentation to be submitted in later milestones.