DSCI 551 – Spring 2026

Homework 1: Firebase and ODM (100 points)

Due: 11:59pm, January 30, 2025, Friday

NO Late Submissions will be accepted!

1. **Background**: Object–Document Mappers (ODMs) provide an abstraction that maps application-level objects to records stored in NoSQL databases. While convenient, ODMs must be designed carefully to avoid hidden I/O, incorrect semantics, and misleading abstractions.

   In this assignment, you will complete a small ODM for Firebase Realtime Database. The design emphasizes:

   - explicit persistence (via save())
   - local mutation vs remote updates
   - faithful modeling of database constraints (e.g., type checking)
   - controlled use of Python descriptors

   You are given a partially implemented system and will complete the missing components.

2. **Provided Code**: You are given the following components (see the attached hw1_template.ipynb):

   - ***Field***: a descriptor that enforces types and tracks local modifications
   - ***Expr*** and CmpExpr: expression objects for query construction
   - ***Document***: a base class for persistence and querying
   - ***User***: a concrete document model

   You may not change the public interfaces or method signatures unless explicitly allowed.

3. **Learning Objectives**: By completing this assignment, you will:

- Understand how Python descriptors implement ODM field behavior
- Distinguish **local mutation** from **database persistence**
- Implement explicit persistence, e.g., using requests.patch()
- Translate high-level query expressions into Firebase REST queries
- Respect the limitations of Firebase Realtime Database querying

- Reason about abstraction boundaries in database systems

4. **Assignment Tasks**:
   - **Task 1 — Comparison Operators in Field (10 points):**

     Complete the Field descriptor to support '<' and '>' operators. These operators must return appropriate CmpExpr objects. Expressions must not be usable in boolean contexts.

   - **Task 2 — Persisting Modified Fields (save) (30 points)**

     Complete the Document.save() instance method. Requirements:

     - Only fields marked as dirty (_dirty) may be sent to the server
     - Use requests.patch() to update the document
     - Clear _dirty only after a successful update
     - Do nothing if no fields are dirty

     Note: you should not be writing to the database during assignment (u.name = "john"), nor writing during attribute access (e.g., u.age).

   - **Task 3 — Fetching a Document (fetch) (25 points)**

     Complete the Document.fetch() class method.

     Requirements:

     - Retrieve the document from Firebase using requests.get()
     - Return a fully initialized object of the calling class
     - Return None if the document does not exist

     Hints: use the class constructor with keyword arguments to populate fields.

   - **Task 4 — Querying the Database (query) (35 points)**

     Complete the Document.query() class method.

     Requirements:
     - Accept exactly **one** comparison expression
     - Translate the expression into valid Firebase RTDB query parameters
     - Support the following operators:
       - ==
       - >=
       - <=
       - >
       - <

- Fill in the code for supporting > and < (10 points)
- Fill in the code to retrieve matching documents and return a list of objects (25 points).

Notes:

- Compound filters are **not supported**
- orderBy="$key" and orderBy="$value" are **not supported**
- limitToFirst and limitToLast are **not supported**

See example usages in the template ipynb file for testing.

5. **Submission:**
    - Rename hw1_template.ipynb to hw1.ipynb and submit it with completed codes.
    - Note: your hw1.ipynb should include cells containing test case and output for each of the four tasks. Clearly add comments, indicating which cell is for testing which case.