Project Plan: Design and Implementation of the Dune Archive System

CMPE321 – Introduction to Database Systems
Spring 2025

1. Overview

This document presents a structured development plan for CMPE321 Project 4: *Dune Archive System*. The system will simulate a minimalist database management system implemented in Python 3. It must support type creation, record manipulation, and search/delete operations based on a file-page-record architecture.

The development plan is divided into six logical phases. Each phase includes specific tasks and deliverables. Additional clarifications are included at the end to address common omissions in previous reports.

2. Part 1 – Project Comprehension and Scope Analysis

Objective: Understand project constraints, command formats, and output expectations.

Tasks:

- Review the project description PDF.
- Examine `input.txt`, `output.txt`, and `README_input_explanation.txt`.
- Identify and document:
  - Operation formats
  - Primary key behavior
  - Failure cases
- Understand `log.csv` structure and persistence requirement.

Deliverables: None.

3. Part 2 – Design Documentation

Objective: Plan system architecture (file organization, page/record layout, metadata handling).

Tasks:

- Describe type-level file structure.
- Design slotted page format with bitmap header and fixed-length slots.
- Define record structure with a validity flag.
- Specify limits on type name and field name lengths.
- Detail the system catalog format for storing type definitions.

Deliverables:

- `report.pdf` (initial version, includes design decisions).

4. Part 3 – Core Infrastructure: Data Structures

Objective: Implement low-level classes and binary I/O.

Tasks:

- Implement `Page`, `Record`, `FileManager`, and `Catalog` classes.
- Add functions for serializing and deserializing records to/from binary format.
- Simulate page-level storage with slot status tracking (bitmap).

Deliverables:

- `archive.py` (base infrastructure implemented).

5. Part 4 – Type and Record Creation

Objective: Implement `create type` and `create record` operations.

Tasks:

- Store new type metadata in system catalog.

- Initialize type file.
- Find available slot for record insertion; check PK uniqueness.
- Log operations into `log.csv`.

Deliverables:

- `archive.py` (create logic complete).
- `log.csv` (initial version).
- Updated `report.pdf`.

6. Part 5 – Deletion and Search Operations

Objective: Complete manipulation layer by supporting delete and search.

Tasks:

- Traverse pages to locate a record by PK.
- Mark deleted records as invalid.
- Write found records to `output.txt`, log results to `log.csv`.

Deliverables:

- `archive.py` (fully functional).
- `log.csv` (expanded).
- `output.txt`.
- Final `report.pdf`.

7. Part 6 – Testing, README, and Packaging

Objective: Finalize testing and package project for submission.

Tasks:

- Create comprehensive `input.txt` for testing.
- Verify `output.txt` and `log.csv` accuracy.

- Write a clean and informative `README.md`.
- Prepare and write `StudentID_Contribution.pdf`.
- Package folder as `2019400XXX.zip` with all files.

Deliverables:

- `2019400XXX.zip` containing all project artifacts.

8. Additional Clarifications and Improvements

1. Report Submission Schedule

The project report (`report.pdf`) will be updated throughout development:

- After Part 2: Include system design (files, pages, record format).
- After Part 4: Add algorithms for create type/record.
- After Part 5: Add deletion and search logic.
- After Part 6: Include testing methodology, limitations, and possible improvements.

2. Test Plan Details

The `input.txt` test file should include:

- At least 2 types and 3–5 records each.
- Cases with:
  - Successful operations
  - Failure scenarios (e.g., duplicate PK, non-existent record)
  - Edge cases (deleting already-deleted records, very long strings)
- A deleted record being searched afterward.

3. log.csv Format Specification

The log file must contain all operations with the following format:

- UNIX timestamp (e.g., 1653412345)

- Original input command string
- Operation status: `success` or `failure`

Each log line will follow CSV format:

```
timestamp, operation string, status
```

Example:

```
1653412345, create record house Atreides Caladan Duke 8000 5000 150, success
```