Project Report

Name: Abhimanyu Singh
Reg. No.: 25BCE10956
Course Name: Introduction to Problem Solving and Programming
Course Code: CSE1021
Slot: A11+A12+A13
Faculty: Pavithra Mam
Class No.: BL2025260100797

---

1. Introduction

This project is a simple, interactive **Complaint Box Management System** that runs in the command line. The main goal was to create an efficient way for students to submit complaints and for administrators to manage them without needing complex databases or heavy software. It focuses on being lightweight, easy to use, and easy to understand.

---

2. Architectural Approach

The project is built using a straightforward, functional programming approach in Python. It doesn't use complex classes or objects to keep the logic simple and easy to follow.

2.1 In-Memory Data Storage

Instead of using file storage or a database, this project uses a simple Python list to store complaints during program execution.

- **Concept:** A global list (`complaints = []`) holds all complaint data as dictionaries
- **Format:** Each complaint contains `id`, `msg` (message), `cat` (category), and `status` fields
- **Advantage:** Very fast and simple to implement, with no file I/O overhead

2.2 Functional Design

The code is broken down into small, reusable functions rather than one big block of code.

- **Core Functions:** `submit()`, `view_all()`, `by_category()`, `update()`, `delete()` handle data operations
- **Menu Functions:** `student_menu()` and `admin_menu()` handle user interactions
- **main():** Controls the overall program flow with a main loop

2.3 Error Handling Strategy

The application uses defensive programming. It assumes users might make mistakes (like typing letters instead of numbers) and uses input validation to prevent crashes.

---

3. Implementation Details

3.1 Data Structure Logic

Each complaint is stored as a dictionary with four fields: ID (sequential number), message (complaint text), category (predefined type), and status (default: 'pending'). The ID is generated using `len(complaints) + 1`, ensuring each complaint gets a unique identifier.

3.2 User Interaction

Student Interface:

- Displays 6 categories: Academic, Facilities, Faculty, Staff, Admin, Other
- Validates category selection (1-6)
- Accepts complaint message and calls `submit()` to store it
- Provides instant confirmation with complaint ID

Admin Interface:

- Runs in a loop with 5 operations: View All, By Category, Update Status, Delete, Exit
- Uses `.isdigit()` validation for ID inputs to prevent errors
- Displays formatted output with visual separators for readability

3.3 Display Functions

view_all(): Iterates through the complaints list and displays each complaint's details with proper formatting using "=" and "-" separators.

by_category(): Creates a temporary dictionary to group complaints by category, counts them, and displays results in sorted order.

3.4 Modification Functions

update(): Searches for a complaint by ID using linear search, modifies the status field if found.

delete(): Uses enumeration to find and remove a complaint from the list using `pop()`.

---

4. Results

4.1 Functional Capabilities

- **Topic Selection:** Students can choose from 6 predefined complaint categories
- **Submission:** Simple two-step process with immediate feedback and ID assignment
- **Management:** Admins can view, filter by category, update status, and delete complaints
- **Display:** Clear, formatted output with proper visual organization

4.2 Performance Observations

- **Speed:** Operations are instant since data is stored in memory. No file reading delays.
- **Reliability:** Input validation prevents crashes from incorrect user inputs. Empty list checks prevent errors when no complaints exist.

---

5. Analysis and Limitations

5.1 Limitations

- **Data Persistence:** All complaints are lost when the program closes (no file storage)
- **ID Management:** After deletions, ID sequence has gaps (IDs aren't reused)
- **Single Session:** Cannot handle multiple users simultaneously
- **No Authentication:** No password protection for admin access

5.2 Future Enhancements

- **File Storage:** Add JSON or text file storage to save complaints permanently
- **Authentication:** Implement password protection for admin access
- **Search Feature:** Add ability to search complaints by keyword
- **Timestamps:** Record submission and update times for each complaint
- **Status Options:** Expand from just 'pending' to include 'in-progress' and 'resolved'
- **GUI:** Use Tkinter to create a graphical interface for better user experience

---

Note: The source code demonstrates core Python concepts including lists, dictionaries, functions, loops, and input validation in a practical, real-world application scenario.