**Practical Assignment 2**

**Course name: CST8002 050**

**Professor’s Name: Todd Keuleman**

**Name: Tanaz Imrankhan Pathan**

**Student Number: 041102857**

Table of Contents

[**Evidence of learning** 3](#_Toc190586221)

[**1.** **Variables:** 3](#_Toc190586222)

[**2.** **loop structure** 4](#_Toc190586223)

[**3.** **Decision Structure** 4](#_Toc190586224)

[**4.** **File-IO reading from dataset** 5](#_Toc190586225)

[**5.** **exception handling:** 6](#_Toc190586226)

[**6.** **. API** 6](#_Toc190586227)

[**Program architecture** 8](#_Toc190586228)

[**Program Demonstration via Screen Shots** 9](#_Toc190586229)

[12](#_Toc190586230)

[**Source Code** 16](#_Toc190586231)

# **Evidence of learning**

## **Variables:**

class Facility:

    def \_\_init\_\_(self, region: str, district: str, license\_number: str, facility\_name: str,

                 facility\_type: str, facility\_address\_1: str, facility\_address\_2: str,

                 facility\_address\_3: str, max\_children: int, max\_infants: int,

                 max\_preschool: int, max\_school\_age: int, language\_of\_service: str,

                 operator\_id: str, designated\_facility: str):

        self.region = region  # Facility's region

        self.district = district  # Facility's district

        self.license\_number = license\_number  # Unique license number for the facility

        self.facility\_name = facility\_name  # Name of the facility

        self.facility\_type = facility\_type  # Type of the facility (e.g., daycare, school)

        self.facility\_address\_1 = facility\_address\_1  # First address line of the facility

        self.facility\_address\_2 = facility\_address\_2  # Second address line of the facility

        self.facility\_address\_3 = facility\_address\_3  # Third address line of the facility (optional)

        self.max\_children = max\_children  # Maximum number of children the facility can accommodate

        self.max\_infants = max\_infants  # Maximum number of infants the facility can accommodate

        self.max\_preschool = max\_preschool  # Maximum number of preschool-aged children

        self.max\_school\_age = max\_school\_age  # Maximum number of school-aged children

        self.language\_of\_service = language\_of\_service  # Language in which services are provided

        self.operator\_id = operator\_id  # ID of the operator in charge of the facility

        self.designated\_facility = designated\_facility  # Whether it is a designated facility or not

## **loop structure**

def load\_facilities(file\_path: str):

    """Loads facility data from a CSV file and returns a list of Facility objects."""

    facilities = []

    try:

        with open(file\_path, mode='r', newline='') as file:

            reader = csv.DictReader(file)

            for row in reader:

                facility = Facility(\*\*row)  # Unpacks row into the Facility constructor

                facilities.append(facility)

    except FileNotFoundError:

        print(f"File {file\_path} not found.")

    return facilities

## **Decision Structure**

def show\_menu(self):

    """Displays the menu for user interaction and handles decisions."""

    while True:

        print("Facility Management System - Main Menu")

        print("1. Add Facility")

        print("2. Edit Facility")

        print("3. Delete Facility")

        print("4. View All Facilities")

        print("5. Save Facilities")

        print("6. Exit")

        choice = input("Enter choice: ")

        if choice == "1":

            self.add\_facility()  # Calls the function to add a facility

        elif choice == "2":

            self.edit\_facility()  # Calls the function to edit an existing facility

        elif choice == "3":

            self.delete\_facility()  # Calls the function to delete a facility

        elif choice == "4":

            self.view\_facilities()  # Calls the function to view all facilities

        elif choice == "5":

            self.save\_facilities()  # Calls the function to save facility data to CSV

        elif choice == "6":

            print("Exiting program...")  # Exits the program if '6' is selected

            break

        else:

            print("Invalid choice. Please try again.")  # Handles invalid inputs

## **File-IO reading from dataset**

def read\_facility\_records(file\_path: str):

    """Reads facility records from a CSV file and returns a list of Facility objects."""

    facilities = []

    try:

        with open(file\_path, mode='r', newline='') as file:

            reader = csv.DictReader(file)

            for row in reader:

                facilities.append(Facility(\*\*row))  # Convert each row into a Facility object

    except FileNotFoundError:

        print(f"Error: The file '{file\_path}' was not found.")

    return facilities

## **exception handling:**

try:

    with open(file\_path, mode='r', newline='') as file:

        reader = csv.DictReader(file)

        for row in reader:

            facilities.append(Facility(\*\*row))

except FileNotFoundError:

    print(f"Error: File '{file\_path}' not found.")

except PermissionError:

    print(f"Error: Permission denied to access '{file\_path}'.")

except Exception as e:

    print(f"Unexpected error occurred: {e}")

## **. API**

**Persistence Layer API**:

def write\_facility\_records(file\_path: str, facilities: list):

    """Writes a list of Facility objects to the specified CSV file."""

    with open(file\_path, mode='w', newline='') as file:

        writer = csv.DictWriter(file, fieldnames=Facility.\_\_dict\_\_.keys())

        writer.writeheader()

        for facility in facilities:

            writer.writerow(facility.\_\_dict\_\_)  # Write each facility as a row

The write\_facility\_records function acts as an API to persist facility data. It writes the list of Facility objects to the specified CSV file, ensuring that the file is updated with all records.

**Business Layer API**:

def add\_facility(facilities: list, new\_facility: Facility):

    """Adds a new facility to the list and returns the updated list."""

    facilities.append(new\_facility)

    return facilities

The add\_facility function allows the addition of new facilities to the internal list. It serves as the business logic API for adding a facility to the system, ensuring the list of facilities is updated.

**7.Array**

facilities = []  # This list will store Facility objects

# Adding a new facility to the list

new\_facility = Facility("Ontario", "Ottawa", "1234", "ABC Daycare", "Daycare", "123 Street", "", "",

                        50, 10, 20, 30, "English", "OP123", "Yes")

facilities.append(new\_facility)  # Adds a new facility

# Displaying the list of facilities

print(facilities)

# Removing a facility from the list by license number

facilities = [facility for facility in facilities if facility.license\_number != "1234"]  # Remove facility with License Number '1234'

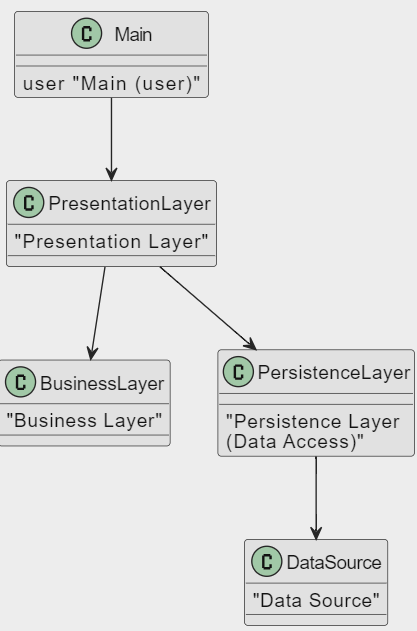
# Output the current state of facilities

print(facilities)  # Displays the updated list of facilities

The facilities list holds the Facility objects. We can add facilities to this list with append(), and we can remove a facility using a list comprehension (based on the license number). This showcases how arrays (or lists) manage collections of facility records.

In this module, I use lists to manage collections of **Facility** objects, taking advantage of Python's dynamic list functionality. The facilities list starts empty and can expand as new facility records are added using the append() method. Furthermore, facilities can be easily removed using a list comprehension (filtering by unique attributes such as the license number), providing efficient management of facility data throughout the program. This approach emphasizes the flexibility and ease of using lists for managing and manipulating collections of facility records in my implementation.

# **Program architecture**



The **Facility Management System** follows a layered architecture with four main components: **Main (USER)**, **Presentation Layer**, **Business Layer**, and **Persistence Layer**. The **Main Layer** serves as the user interface, allowing users to interact with the system. The **Presentation Layer** is responsible for managing user input and displaying the output, while the **Business Layer** handles the core logic and processes the facility data. The **Persistence Layer** manages the storage of data by interacting with CSV files to read and write facility records. This architecture improves maintainability and efficiency by clearly separating concerns, ensuring that each layer has its own responsibility in managing facility records.

# **Program Demonstration via Screen Shots**A screenshot of a computer AI-generated content may be incorrect.

This option allows you to load facility records from an external file (CSV). You will be prompted to provide the file path, and the system will read the data and load it into the system.

A computer screen shot of a black screen

AI-generated content may be incorrect.

This option displays all the facility records that have been loaded into the system. If there are no records, it will let you know.

A screenshot of a computer

AI-generated content may be incorrect.

This option allows you to search for and view a specific facility by its **License Number**. You’ll be asked to enter the License Number, and the system will show the relevant facility's details if it’s found.

A screenshot of a computer program

AI-generated content may be incorrect.

This option lets you add a new facility. You'll be prompted to input details such as region, district, license number, facility name, etc. The new facility will be added to the system and saved

# A screenshot of a computer program AI-generated content may be incorrect.

This option enables you to update an existing facility’s information. You’ll search for the facility by its License Number and then be asked to enter the updated details. After editing, the system will replace the old record with the new one.

A screenshot of a computer program

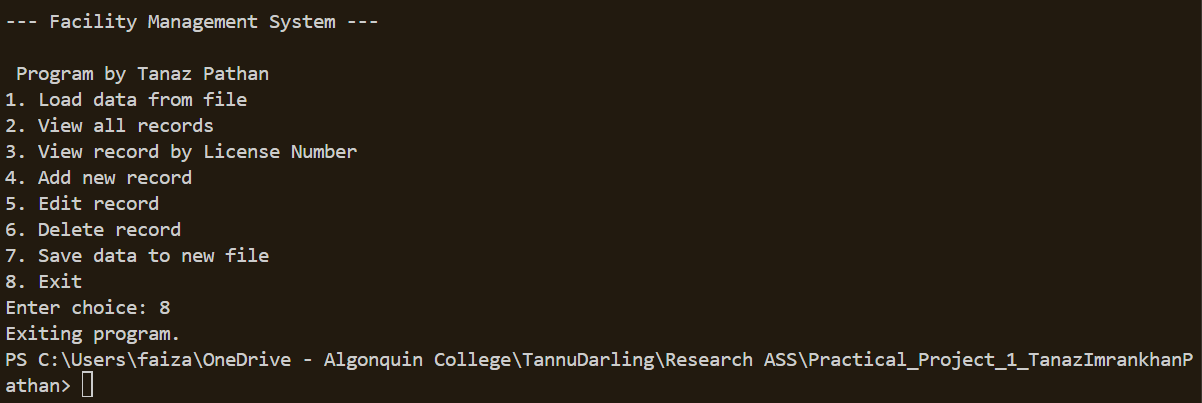
AI-generated content may be incorrect.

This option allows you to delete a facility record. You’ll be prompted to provide the License Number of the facility you wish to delete, and once found, the record will be removed from the system.

A screen shot of a computer

AI-generated content may be incorrect.

After adding, editing, or deleting records, this option allows you to save the current facility records to a new file. The system generates a new filename (UUID-based) and writes the updated data to a CSV file.



This option exits the system, ending the program. It allows you to safely close the application.

**Pytest**

import unittest

from business.business\_logic import add\_facility, edit\_facility, delete\_facility, view\_facilities

from model.facility import Facility

class TestBusinessLogic(unittest.TestCase):

    def setUp(self):

        self.facility1 = Facility("Region1", "District1", "LN001", "Facility1", "Type1",

                                  "Address1", "Address2", "Address3", 100, 10, 20, 30,

                                  "English", "Operator1", "Yes")

        self.facility2 = Facility("Region2", "District2", "LN002", "Facility2", "Type2",

                                  "Address1", "Address2", "Address3", 50, 5, 10, 15,

                                  "Spanish", "Operator2", "No")

        self.facilities = [self.facility1, self.facility2]

    def test\_add\_facility(self):

        new\_facility = Facility("Region3", "District3", "LN003", "Facility3", "Type3",

                                "Address1", "Address2", "Address3", 75, 7, 12, 18,

                                "French", "Operator3", "Yes")

        updated\_facilities = add\_facility(self.facilities, new\_facility)

        self.assertEqual(len(updated\_facilities), 3)

    def test\_edit\_facility(self):

        updated\_facility = Facility("Region1", "District1", "LN001", "Updated Facility1", "Type1",

                                    "Updated Address1", "Address2", "Address3", 150, 20, 30, 40,

                                    "English", "Operator1", "Yes")

        updated\_facilities = edit\_facility(self.facilities, "LN001", updated\_facility)

        self.assertEqual(updated\_facilities[0].facility\_name, "Updated Facility1")

    def test\_delete\_facility(self):

        updated\_facilities = delete\_facility(self.facilities, "LN001")

        self.assertEqual(len(updated\_facilities), 1)

    def test\_view\_facilities(self):

        view\_facilities(self.facilities)

)

**A screen shot of a computer program

AI-generated content may be incorrect.**

# **Source Code**

from typing import List

from model.facility import Facility

def add\_facility(facilities: List[Facility], facility: Facility) -> List[Facility]:

    facilities.append(facility)

    return facilities

def edit\_facility(facilities: List[Facility], license\_number: str, updated\_facility: Facility) -> List[Facility]:

    for i, facility in enumerate(facilities):

        if facility.license\_number == license\_number:

            facilities[i] = updated\_facility  # Update the record

            return facilities

    print(f"Facility with License Number {license\_number} not found.")

    return facilities

def delete\_facility(facilities: List[Facility], license\_number: str) -> List[Facility]:

    facilities = [facility for facility in facilities if facility.license\_number != license\_number]

    return facilities

def view\_facilities(facilities: List[Facility], license\_number: str = None):

    if license\_number:

        for facility in facilities:

            if facility.license\_number == license\_number:

                print(facility)

                return

        print(f"Facility with License Number {license\_number} not found.")

    else:

        for facility in facilities:

            print(facility)

class Facility:

    def \_\_init\_\_(self, region, district, license\_number, facility\_name, facility\_type,

                 facility\_address\_1, facility\_address\_2, facility\_address\_3,

                 max\_children, max\_infants, max\_preschool, max\_school\_age,

                 language\_of\_service, operator\_id, designated\_facility):

        self.region = region

        self.district = district

        self.license\_number = license\_number

        self.facility\_name = facility\_name

        self.facility\_type = facility\_type

        self.facility\_address\_1 = facility\_address\_1

        self.facility\_address\_2 = facility\_address\_2

        self.facility\_address\_3 = facility\_address\_3

        self.max\_children = max\_children

        self.max\_infants = max\_infants

        self.max\_preschool = max\_preschool

        self.max\_school\_age = max\_school\_age

        self.language\_of\_service = language\_of\_service

        self.operator\_id = operator\_id

        self.designated\_facility = designated\_facility

    def \_\_str\_\_(self):

        return f"Facility {self.facility\_name} ({self.license\_number}) - {self.region}, {self.district}"

import csv

import uuid

from typing import List

from model.facility import Facility

def read\_facility\_records(file\_path: str) -> List[Facility]:

    facilities = []

    try:

        with open(file\_path, mode='r') as file:

            csv\_reader = csv.reader(file)

            next(csv\_reader)  # Skip header row

            for i, row in enumerate(csv\_reader):

                if len(row) == 15:  # Ensure there are 15 columns

                    facility = Facility(\*row)  # Unpack row data into a Facility object

                    facilities.append(facility)

                if i == 99:  # Stop after 100 records

                    break

    except FileNotFoundError:

        print(f"Error: File '{file\_path}' not found.")

    return facilities

def write\_facility\_records(file\_path: str, records: List[Facility]) -> None:

    try:

        with open(file\_path, mode='w', newline='') as file:

            csv\_writer = csv.writer(file)

            csv\_writer.writerow(['Region', 'District', 'License-Number', 'Facility-Name', 'Facility-Type',

                                 'Facility-Address-1', 'Facility-Address-2', 'Facility-Address-3',

                                 'Max-Number-of-Children', 'Max-Number-of-Infants', 'Max-Number-of-Preschool-Aged-Children',

                                 'Max-Number-of-School-Age-Children', 'Language-of-Service', 'Operator-Id', 'Designated-Facility'])

            for record in records:

                csv\_writer.writerow([record.region, record.district, record.license\_number, record.facility\_name,

                                     record.facility\_type, record.facility\_address\_1, record.facility\_address\_2,

                                     record.facility\_address\_3, record.max\_children, record.max\_infants,

                                     record.max\_preschool, record.max\_school\_age, record.language\_of\_service,

                                     record.operator\_id, record.designated\_facility])

    except Exception as e:

        print(f"Error: Unable to write to file: {e}")

def generate\_uuid\_filename() -> str:

    return str(uuid.uuid4()) + ".csv"

import os

from business.business\_logic import add\_facility, edit\_facility, delete\_facility, view\_facilities

from persistence.file\_io import read\_facility\_records, write\_facility\_records, generate\_uuid\_filename

from model.facility import Facility

class UserInterface:

    def \_\_init\_\_(self):

        self.facilities = []

    def show\_menu(self):

        while True:

            print("\n--- Facility Management System ---")

            print("\n Program by Tanaz Pathan")

            print("1. Load data from file")

            print("2. View all records")

            print("3. View record by License Number")

            print("4. Add new record")

            print("5. Edit record")

            print("6. Delete record")

            print("7. Save data to new file")

            print("8. Exit")

            choice = input("Enter choice: ")

            if choice == '1':

                self.load\_data()

            elif choice == '2':

                self.view\_all\_records()

            elif choice == '3':

                self.view\_record\_by\_license\_number()

            elif choice == '4':

                self.add\_record()

            elif choice == '5':

                self.edit\_record()

            elif choice == '6':

                self.delete\_record()

            elif choice == '7':

                self.save\_data()

            elif choice == '8':

                print("Exiting program.")

                break

            else:

                print("Invalid choice, please try again.")

    def load\_data(self):

        file\_path = input("Enter file path to load data: ")

        if os.path.exists(file\_path):

            self.facilities = read\_facility\_records(file\_path)

            print(f"Loaded {len(self.facilities)} records.")

        else:

            print(f"Error: File '{file\_path}' not found.")

    def view\_all\_records(self):

        if not self.facilities:

            print("No records to display.")

        else:

            for facility in self.facilities:

                print(facility)

    def view\_record\_by\_license\_number(self):

        license\_number = input("Enter License Number to search: ")

        view\_facilities(self.facilities, license\_number)

    def add\_record(self):

        print("Enter details for the new facility:")

        region = input("Region: ")

        district = input("District: ")

        license\_number = input("License Number: ")

        facility\_name = input("Facility Name: ")

        facility\_type = input("Facility Type: ")

        facility\_address\_1 = input("Facility Address Line 1: ")

        facility\_address\_2 = input("Facility Address Line 2: ")

        facility\_address\_3 = input("Facility Address Line 3: ")

        max\_children = input("Max Number of Children: ")

        max\_infants = input("Max Number of Infants: ")

        max\_preschool = input("Max Number of Preschool-Aged Children: ")

        max\_school\_age = input("Max Number of School-Age Children: ")

        language\_of\_service = input("Language of Service: ")

        operator\_id = input("Operator ID: ")

        designated\_facility = input("Designated Facility (Yes/No): ")

        new\_facility = Facility(region, district, license\_number, facility\_name, facility\_type,

                                facility\_address\_1, facility\_address\_2, facility\_address\_3,

                                max\_children, max\_infants, max\_preschool, max\_school\_age,

                                language\_of\_service, operator\_id, designated\_facility)

        self.facilities = add\_facility(self.facilities, new\_facility)

        print("New facility added.")

    def edit\_record(self):

        license\_number = input("Enter License Number of the facility to edit: ")

        # Find the facility by license number

        facility = next((f for f in self.facilities if f.license\_number == license\_number), None)

        if facility:

            print(f"Editing facility: {facility.facility\_name} ({facility.license\_number})")

            # Prompt the user for the updated fields

            region = input(f"Enter Region (current: {facility.region}): ") or facility.region

            district = input(f"Enter District (current: {facility.district}): ") or facility.district

            facility\_name = input(f"Enter Facility Name (current: {facility.facility\_name}): ") or facility.facility\_name

            facility\_type = input(f"Enter Facility Type (current: {facility.facility\_type}): ") or facility.facility\_type

            facility\_address\_1 = input(f"Enter Facility Address 1 (current: {facility.facility\_address\_1}): ") or facility.facility\_address\_1

            facility\_address\_2 = input(f"Enter Facility Address 2 (current: {facility.facility\_address\_2}): ") or facility.facility\_address\_2

            facility\_address\_3 = input(f"Enter Facility Address 3 (current: {facility.facility\_address\_3}): ") or facility.facility\_address\_3

            max\_children = input(f"Enter Max Number of Children (current: {facility.max\_children}): ") or facility.max\_children

            max\_infants = input(f"Enter Max Number of Infants (current: {facility.max\_infants}): ") or facility.max\_infants

            max\_preschool = input(f"Enter Max Number of Preschool-Aged Children (current: {facility.max\_preschool}): ") or facility.max\_preschool

            max\_school\_age = input(f"Enter Max Number of School-Aged Children (current: {facility.max\_school\_age}): ") or facility.max\_school\_age

            language\_of\_service = input(f"Enter Language of Service (current: {facility.language\_of\_service}): ") or facility.language\_of\_service

            operator\_id = input(f"Enter Operator ID (current: {facility.operator\_id}): ") or facility.operator\_id

            designated\_facility = input(f"Enter Designated Facility (current: {facility.designated\_facility}): ") or facility.designated\_facility

            # Create a new Facility object with updated values

            updated\_facility = Facility(region, district, license\_number, facility\_name, facility\_type,

                                        facility\_address\_1, facility\_address\_2, facility\_address\_3,

                                        max\_children, max\_infants, max\_preschool, max\_school\_age,

                                        language\_of\_service, operator\_id, designated\_facility)

            # Replace the old facility with the updated one

            self.facilities[self.facilities.index(facility)] = updated\_facility

            print("Facility record updated successfully.")

        else:

            print("Facility not found.")

    def delete\_record(self):

        license\_number = input("Enter License Number of the facility to delete: ")

        self.facilities = delete\_facility(self.facilities, license\_number)

        print(f"Facility with License Number {license\_number} deleted.")

    def save\_data(self):

        file\_name = generate\_uuid\_filename()

        write\_facility\_records(file\_name, self.facilities)

        print(f"Data saved to {file\_name}.")

from presentation.user\_interface import UserInterface

if \_\_name\_\_ == "\_\_main\_\_":

    print ("Program by Tanaz Pathan")

    ui = UserInterface()

    ui.show\_menu()