FAN ENGAGEMENT PORTAL

CODEX

TEAM NAME-

- Implenation of Fan Engagement Portal By using CRUD operations
- And also by using UNITTEST method

1. VIDYASHREE N T
2. SHREYA MODI
3. Y INDUSREE
4. VIDYASHREE

ABSTRACT

Our Fan Engagement Portal (FEP) revolutionizes the way fans connect with their favorite brands, teams, and personalities. Through a seamless blend of content aggregation, interactive features, and personalized experiences, the FEP serves as a one-stop destination for fan engagement. With interactive quizzes, virtual events, gamification elements, and social integration, the FEP cultivates vibrant communities, drives loyalty, and amplifies brand reach. Join us on a journey to redefine fan engagement and unlock the full potential of your audience relationships.

PROJECT REVIEW-

The goal of this project is to create a Proof-of-Concept (POC) for a Fan Engagement Portal using Object-Oriented Programming (OOP) principles in Python. The portal will allow for the management of fan data, conducting fan surveys, and generating detailed reports on fan engagement metrics.

Step 1: Defining Data Structures and Classes:

Fan Class: This class represents individual fan details such as name, email, preferences, etc. It encapsulates the attributes and behaviors associated with a fan entity.

FanDatabase Class: This class manages the storage and CRUD (Create, Read, Update, Delete) operations for fan data objects. It provides methods to add, retrieve, update, and delete fan records efficiently..

SurveyManager Class: This class handles operations related to survey management. It includes functionalities to conduct surveys to gauge fan satisfaction and preferences.
ReportManager Class: This class is responsible for generating reports based on the current fan data. It computes and presents detailed engagement metrics and insights.

Step 2: Implementing Python Code:

The Python code will include the actual implementation of the classes and their functionalities as outlined in Step 1. This will involve writing methods for CRUD operations, survey management, report generation, and any other necessary functionalities.

Step 3: Unit Tests:

Unit tests will be developed to ensure the correctness and reliability of the implemented functionalities. These tests will verify that each component behaves as expected and handles various edge cases effectively.

- MODULE DESCRIPTION-
- Module 1: Fan Management-
- Description: This module focuses on managing fan data, including creating, retrieving, updating, and deleting fan records
- Classes:1.Fan Class:
- Description: Represents individual fan details.
- Attributes:
- Name
- Email
- Preferences (e.g., favorite team, favorite player)
- Methods:
- Constructor to initialize fan attributes
- 2.FanDatabase Class:
- Description: Manages the storage and CRUD operations for fan data objects.
- Methods:
- Add fan: Adds a new fan record to the database
- Get fan by email: Retrieves a fan record based on the email address
- Update fan: Updates an existing fan record

- Delete fan: Removes a fan record from the database
- Module 2: Survey Management-
- Description: This module handles conducting surveys to gauge fan satisfaction and preference.
- Classes:
- SurveyManager Class:
- Description: Handles operations related to survey management.
- Methods:
- Conduct survey: Initiates and conducts a survey to gather fan feedback
- Analyze survey results: Processes and analyzes the survey responses to extract insights
- Module 3: Report Generation-
- Description: This module generates detailed reports on fan engagement metrics based on the current fan data.
- **Classes:**
- **ReportManager Class:**
    - Description: Responsible for generating reports based on the fan data.
    - Methods:
- Generate engagement report: Generates a detailed report on fan engagement metrics
- Visualize data: Provides visualization of engagement metrics using charts or graphs.

Module 4: Main Application-

Description: This module serves as the main entry point for the application, orchestrating the interaction between different modules and functionalities.

Functions:

Main Functionality:

Description: Integrates the functionalities of fan management, survey management, and report generation.

Steps:

Initialize fan database

Manage fan data (CRUD operations)

Conduct surveys

Generate reports

Display results to the user

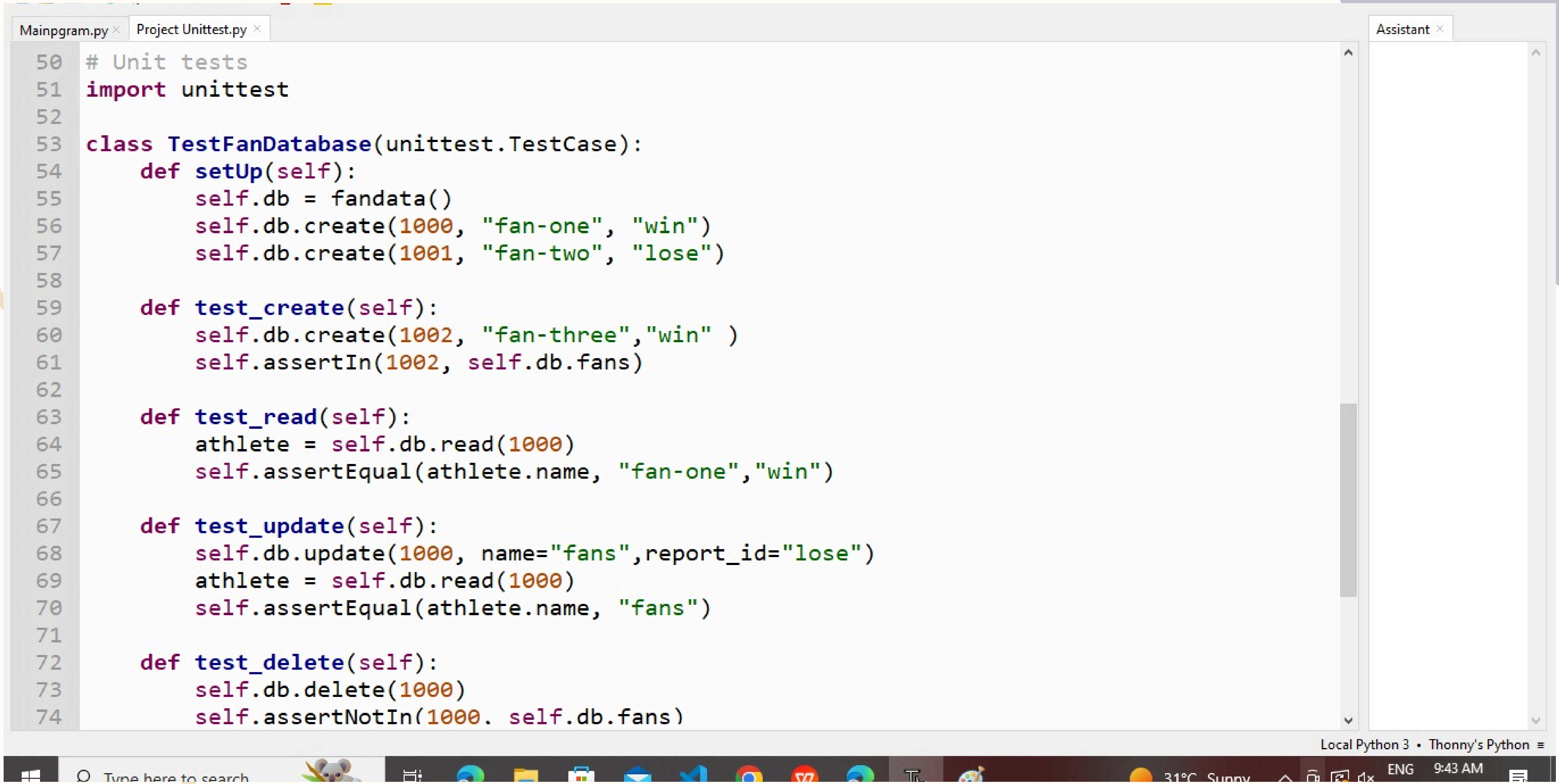
Module 5: Unit Testing-

Description: This module contains unit tests to ensure the correctness and reliability of the implemented functionalities.

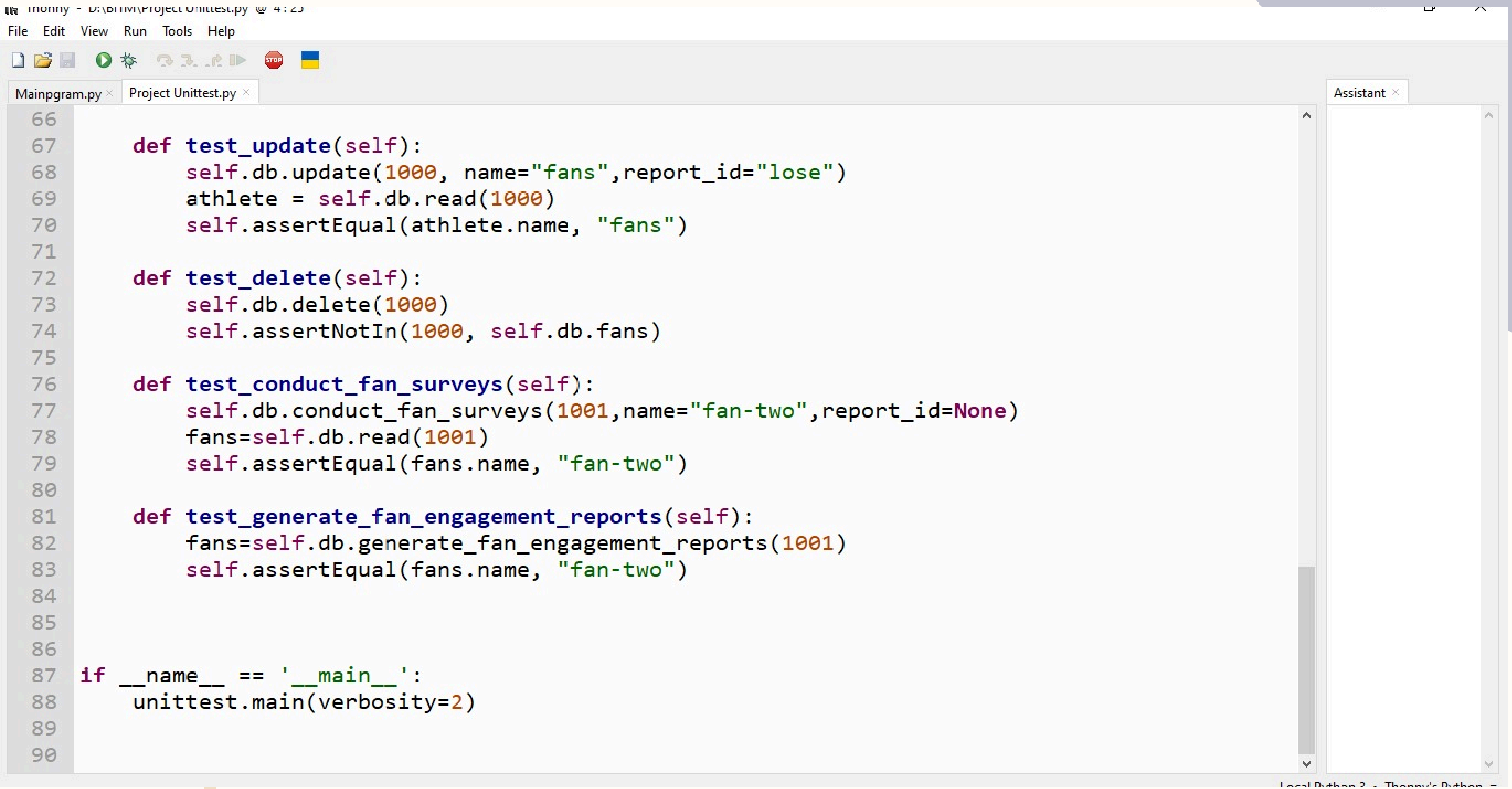
Tests:Fan Management Tests ,Survey Management Tests,

Report Generation Tests

Unittest

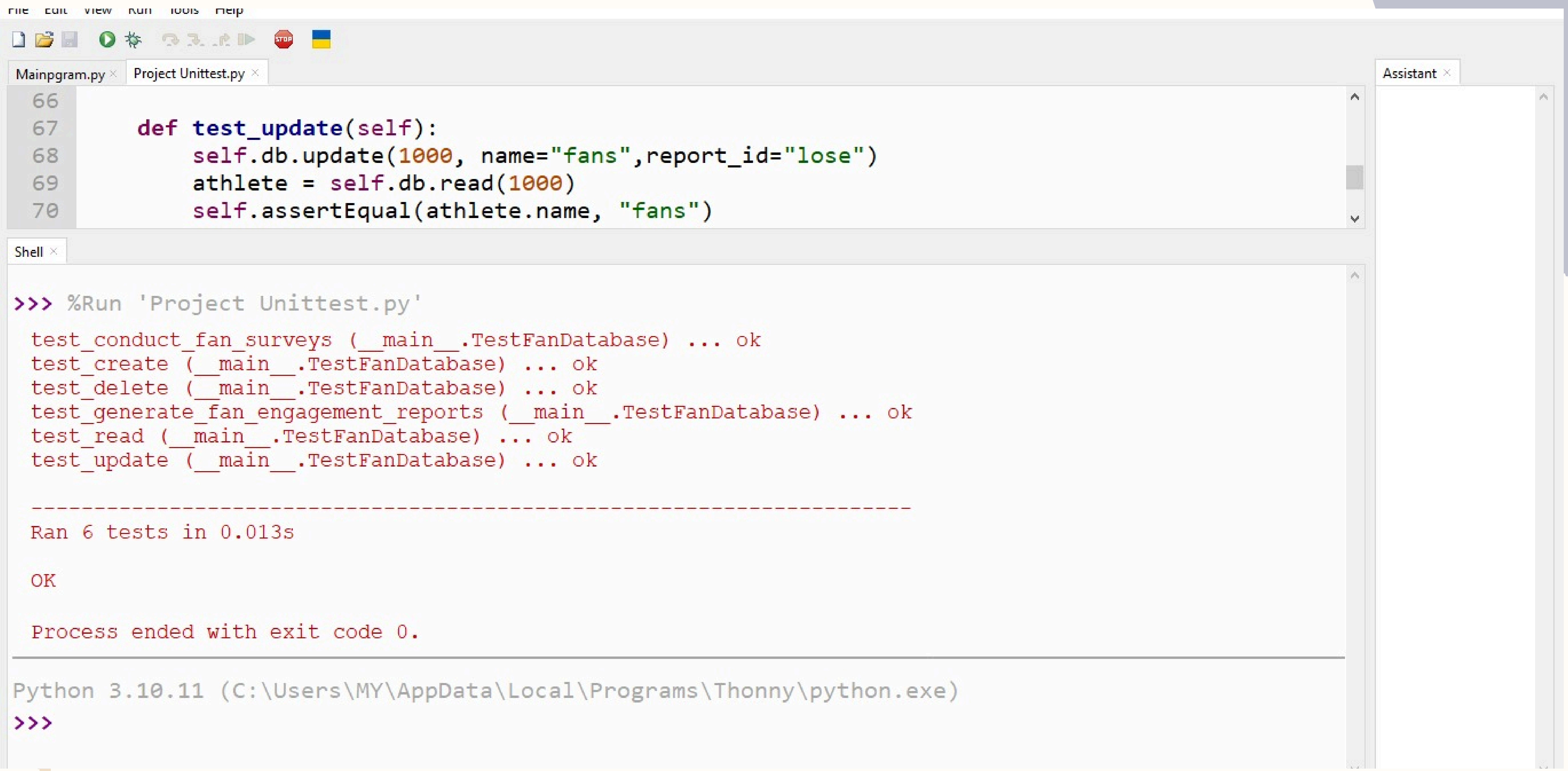


```python
# Unit tests
import unittest

class TestFanDatabase(unittest.TestCase):
    def setUp(self):
        self.db = fandata()
        self.db.create(1000, "fan-one", "win")
        self.db.create(1001, "fan-two", "lose")

    def test_create(self):
        self.db.create(1002, "fan-three","win" )
        self.assertIn(1002, self.db.fans)

    def test_read(self):
        athlete = self.db.read(1000)
        self.assertEqual(athlete.name, "fan-one","win")

    def test_update(self):
        self.db.update(1000, name="fans",report_id="lose")
        athlete = self.db.read(1000)
        self.assertEqual(athlete.name, "fans")

    def test_delete(self):
        self.db.delete(1000)
        self.assertNotIn(1000, self.db.fans)
```


Thonny - D:\BTM\Project Unittest.py @ 4 : 25
File
Edit
View
Run
Tools
Help
Mainpgram.py
Project Unittest.py
Assistant
Local Python 3 • Thonny's Python

```python

    def test_update(self):
        self.db.update(1000, name="fans",report_id="lose")
        athlete = self.db.read(1000)
        self.assertEqual(athlete.name, "fans")

    def test_delete(self):
        self.db.delete(1000)
        self.assertNotIn(1000, self.db.fans)

    def test_conduct_fan_surveys(self):
        self.db.conduct_fan_surveys(1001,name="fan-two",report_id=None)
        fans=self.db.read(1001)
        self.assertEqual(fans.name, "fan-two")

    def test_generate_fan_engagement_reports(self):
        fans=self.db.generate_fan_engagement_reports(1001)
        self.assertEqual(fans.name, "fan-two")



if __name__ == '__main__':
    unittest.main(verbosity=2)


```

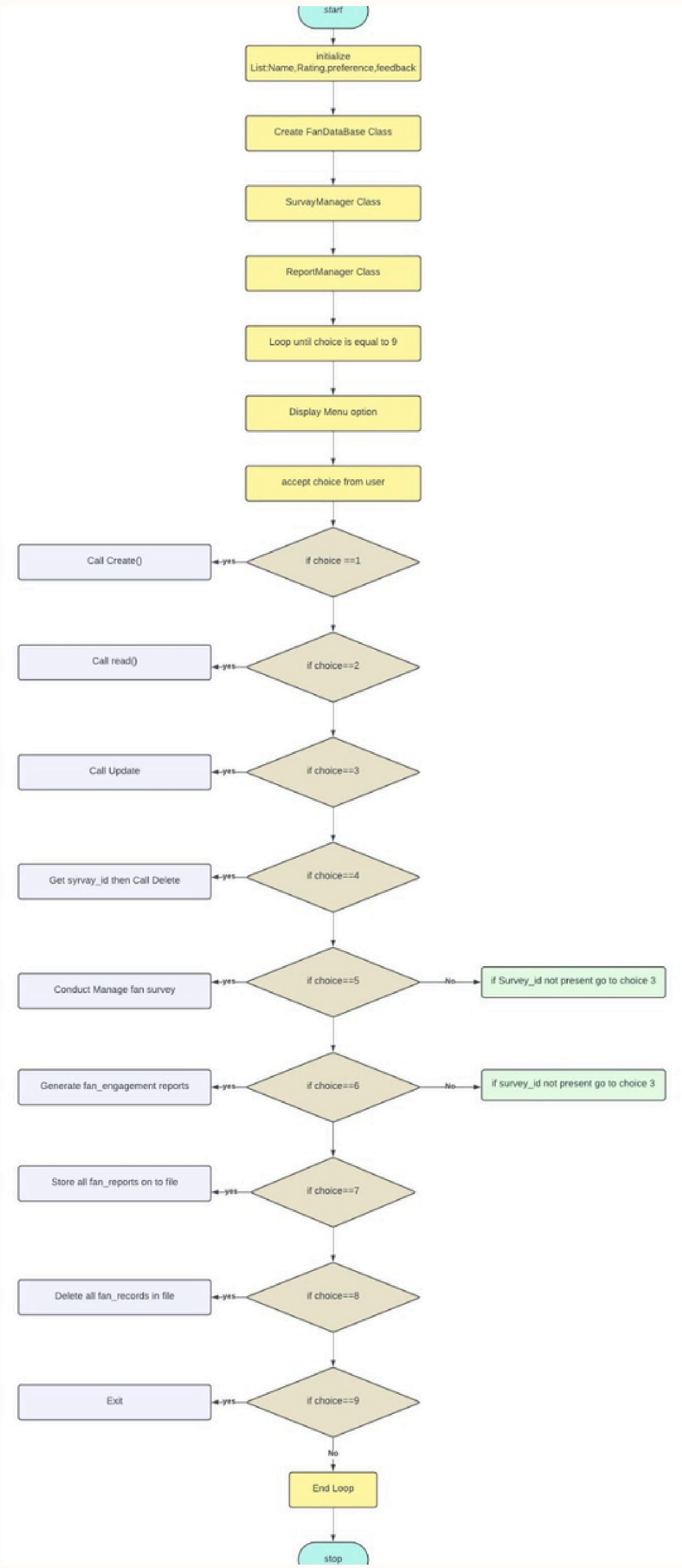
UNITTEST OP



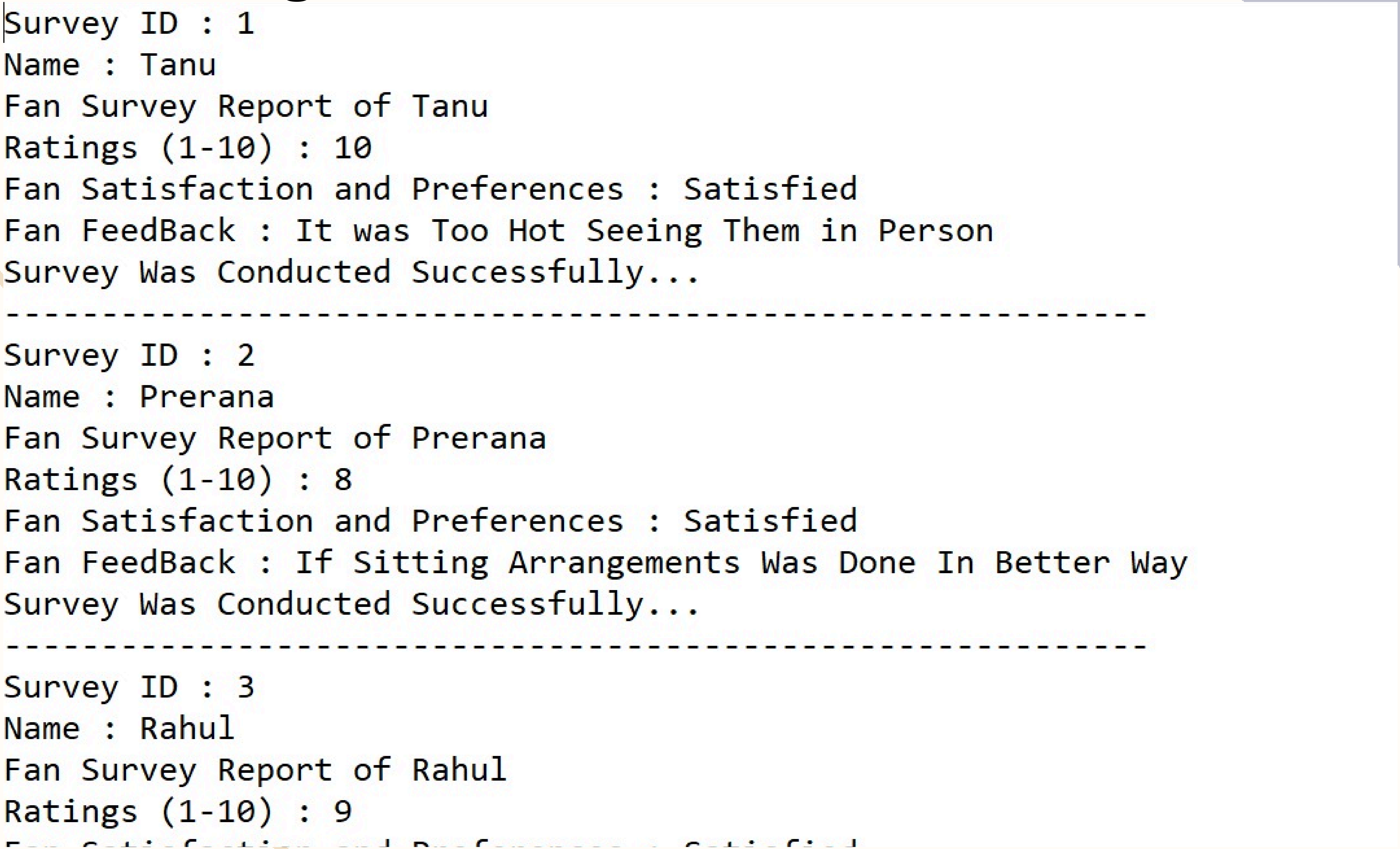
```python

    def test_update(self):
        self.db.update(1000, name="fans",report_id="lose")
        athlete = self.db.read(1000)
        self.assertEqual(athlete.name, "fans")
```

```
>>> %Run 'Project Unittest.py'
 test_conduct_fan_surveys (__main__.TestFanDatabase) ... ok
 test_create (__main__.TestFanDatabase) ... ok
 test_delete (__main__.TestFanDatabase) ... ok
 test_generate_fan_engagement_reports (__main__.TestFanDatabase) ... ok
 test_read (__main__.TestFanDatabase) ... ok
 test_update (__main__.TestFanDatabase) ... ok

 ----------------------------------------------------------------------
 Ran 6 tests in 0.013s

 OK

 Process ended with exit code 0.

Python 3.10.11 (C:\Users\MY\AppData\Local\Programs\Thonny\python.exe)
>>>
```

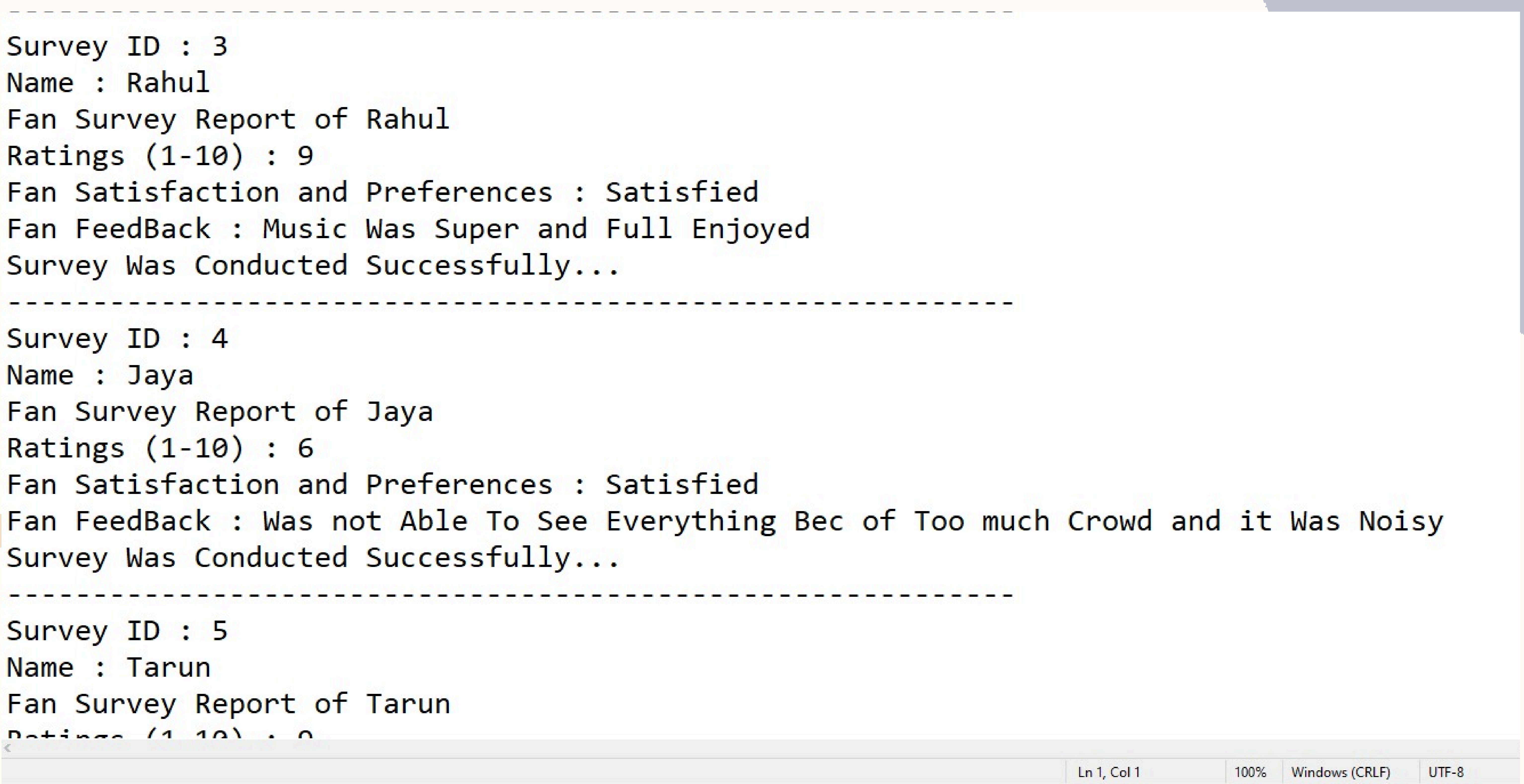
FLOW CHART


start
initialize
List:Name,Rating,preference,feedback
Create FanDataBase Class
SurvayManager Class
ReportManager Class
Loop until choice is equal to 9
Display Menu option
accept choice from user
if choice ==1
yes
Call Create()
if choice==2
yes
Call read()
if choice==3
yes
Call Update
if choice==4
yes
Get syrvay_id then Call Delete
if choice==5
yes
Conduct Manage fan survey
No
if Survey_id not present go to choice 3
if choice==6
yes
Generate fan_engagement reports
No
if survey_id not present go to choice 3
if choice==7
yes
Store all fan_reports on to file
if choice==8
yes
Delete all fan_records in file
if choice==9
yes
Exit
No
End Loop
stop

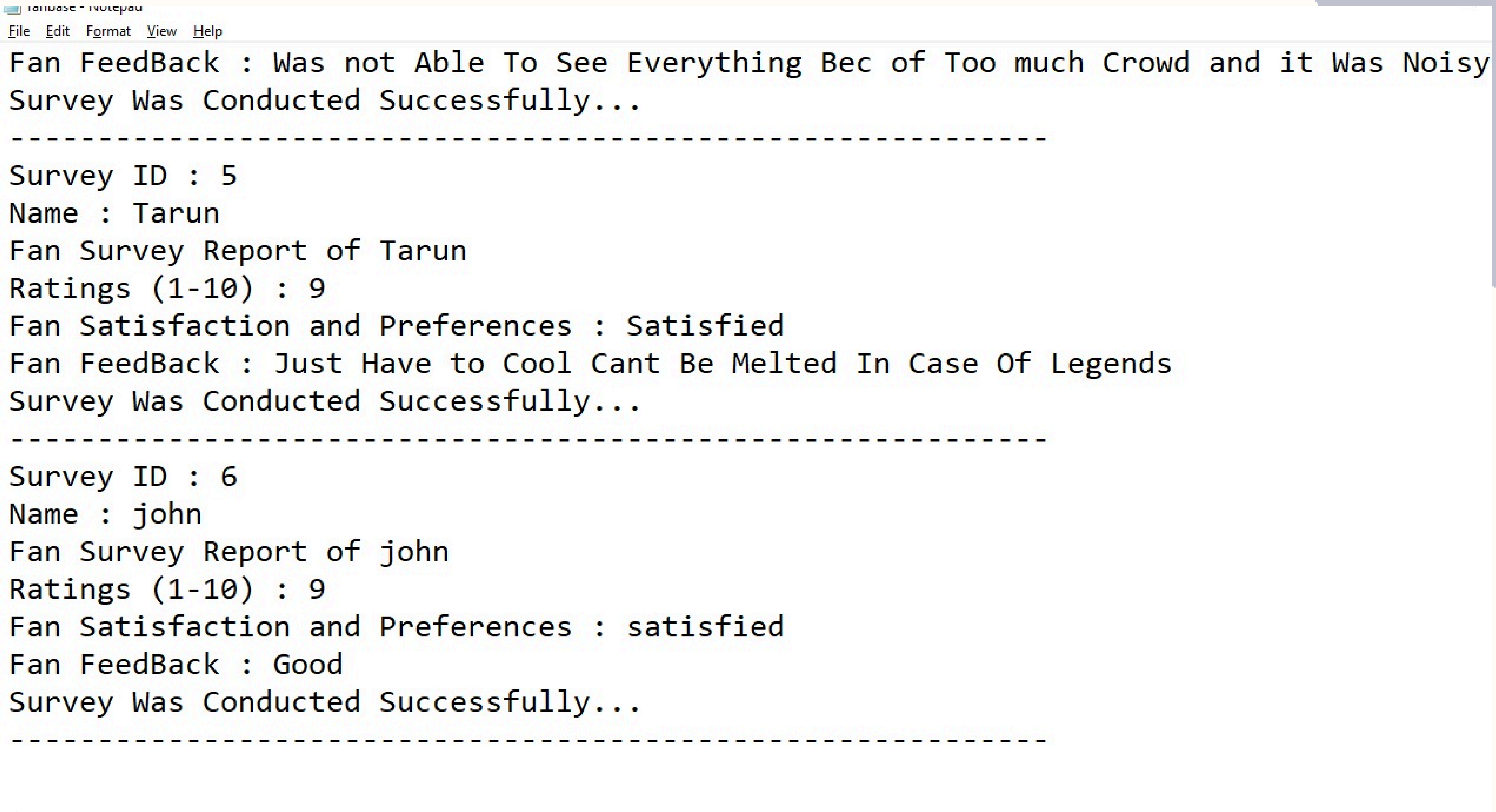
File handling OP



```
Survey ID : 1
Name : Tanu
Fan Survey Report of Tanu
Ratings (1-10) : 10
Fan Satisfaction and Preferences : Satisfied
Fan FeedBack : It was Too Hot Seeing Them in Person
Survey Was Conducted Successfully...
-----------------------------------------------------------
Survey ID : 2
Name : Prerana
Fan Survey Report of Prerana
Ratings (1-10) : 8
Fan Satisfaction and Preferences : Satisfied
Fan FeedBack : If Sitting Arrangements Was Done In Better Way
Survey Was Conducted Successfully...
-----------------------------------------------------------
Survey ID : 3
Name : Rahul
Fan Survey Report of Rahul
Ratings (1-10) : 9
Fan Satisfaction and Preferences : Satisfied
```



```
-----------------------------------------------------------
Survey ID : 3
Name : Rahul
Fan Survey Report of Rahul
Ratings (1-10) : 9
Fan Satisfaction and Preferences : Satisfied
Fan FeedBack : Music Was Super and Full Enjoyed
Survey Was Conducted Successfully...
-----------------------------------------------------------
Survey ID : 4
Name : Jaya
Fan Survey Report of Jaya
Ratings (1-10) : 6
Fan Satisfaction and Preferences : Satisfied
Fan FeedBack : Was not Able To See Everything Bec of Too much Crowd and it Was Noisy
Survey Was Conducted Successfully...
-----------------------------------------------------------
Survey ID : 5
Name : Tarun
Fan Survey Report of Tarun
Ratings (1-10) : 9
```



```
Fan FeedBack : Was not Able To See Everything Bec of Too much Crowd and it Was Noisy
Survey Was Conducted Successfully...
-----------------------------------------------------------
Survey ID : 5
Name : Tarun
Fan Survey Report of Tarun
Ratings (1-10) : 9
Fan Satisfaction and Preferences : Satisfied
Fan FeedBack : Just Have to Cool Cant Be Melted In Case Of Legends
Survey Was Conducted Successfully...
-----------------------------------------------------------
Survey ID : 6
Name : john
Fan Survey Report of john
Ratings (1-10) : 9
Fan Satisfaction and Preferences : satisfied
Fan FeedBack : Good
Survey Was Conducted Successfully...
-----------------------------------------------------------
```

BIBLIOGRAPHY-

google,class notebook
https://chat.openai.com/
https://github.com/YIndusree
https://github.com/VidyashreeAngadi
https://github.com/shreyamodi7/BITMS
https://github.com/vidyashreent14/BITMS

THANK YOU!!!