INTERNATIONAL UNIVERSITY OF APPLIED SCIENCES

HABIT TRACKER APP

A Python backend app to track and analyze habits

Bullet Points

- Combines **object-oriented** and **functional programming**
- **Features**: create habits, track streaks, analyze performance
- **Backend only** – CLI-based, lightweight, extensible

Date: sept. 2025
Author: Cosulean Cristian

iu
INTERNATIONAL
UNIVERSITY OF
APPLIED SCIENCES

Frameworks and Tools Used

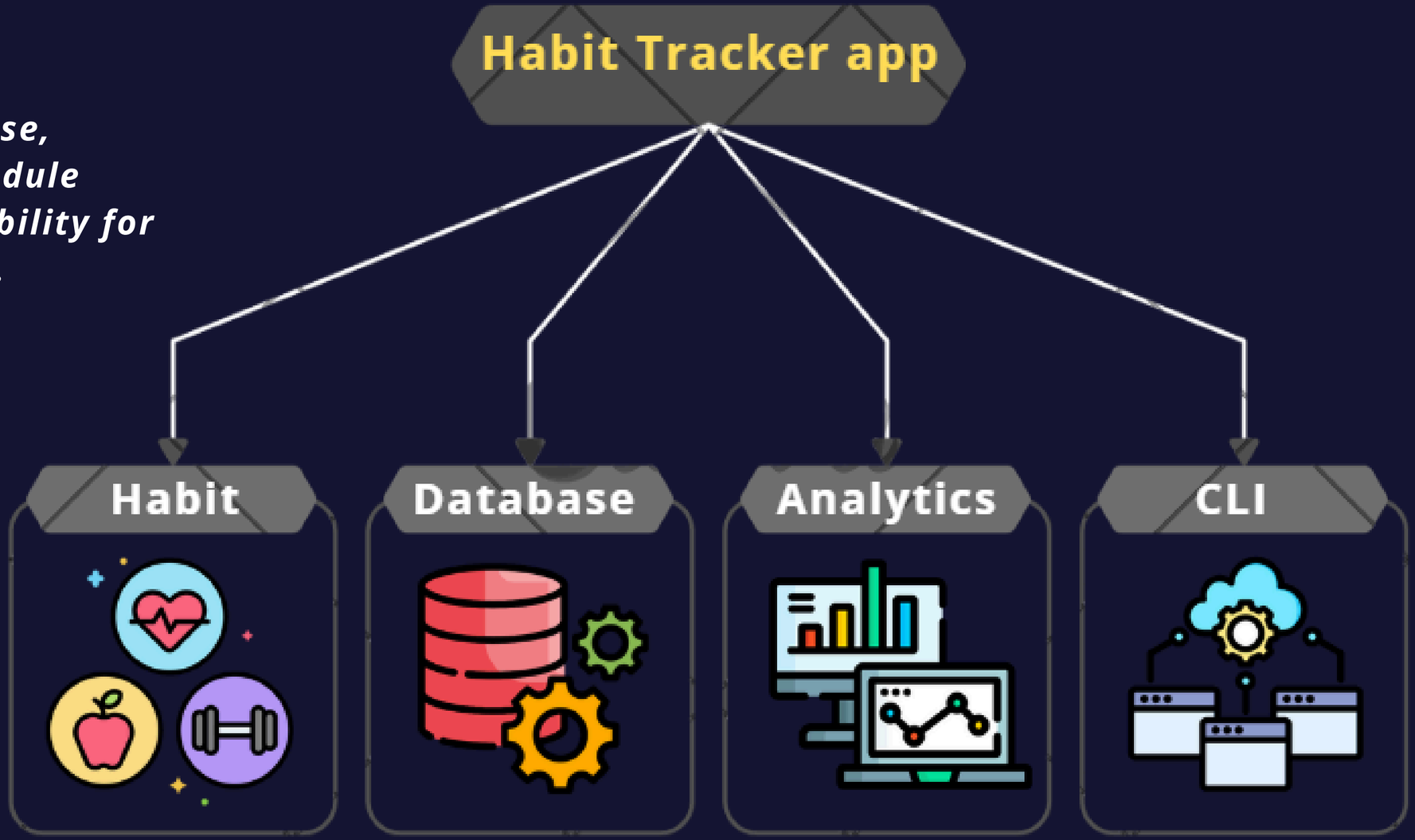
pytest

- Python 3 – core programming language
- SQLite – persistent storage of habits and completions
- Questionary – user-friendly CLI menus and input handling
- Pytest – automated testing for correctness and stability
- VS Code – development environment

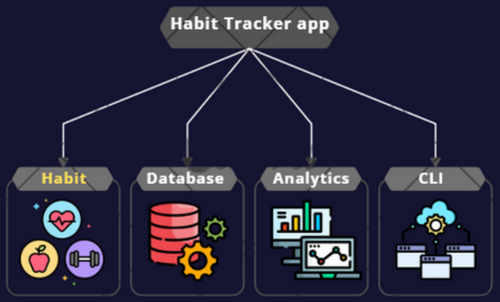
iu
INTERNATIONAL
UNIVERSITY OF
APPLIED SCIENCES

System Architecture Overview

Modular design with four components: Habit, Database, Analytics, and CLI. Each module handles a specific responsibility for maintainability and clarity.



INTERNATIONAL UNIVERSITY OF APPLIED SCIENCES



Habit Module

Represents an individual habit:

```python
class Habit:

    """
    Represents a habit that can be tracked, completed, and analyzed.
    Stores its name, frequency, periodicity, and interacts with the database.
    """
```

Key Methods:

```python
    def __init__(self, name, frequency, periodicity, db_instance=None, habit_id=None):···

    def performed(self):···

    def calculate_current_streak(self):···

    def calculate_longest_streak(self):···

    def can_mark_performed(self) -> bool:···
```

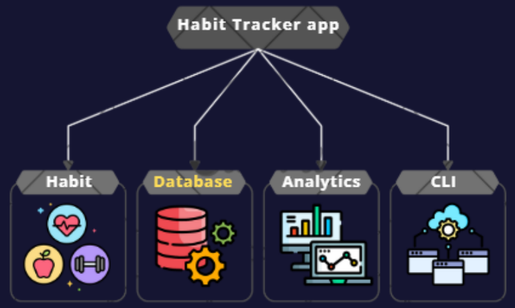
Attributes: name, frequency, periodicity, creation date, completions

```python
@classmethod
def from_db(cls, habit_id: int, db_instance=None):

    """
    Creates a Habit instance from a database record using its habit_id.
    It searches the database for a habit with the given ID and initializes an object with its attributes (name, frequency, periodicity, creation_date).
    If no matching record is found, it returns None.
    """

    db_inst = db_instance if db_instance else db
    record = next((h for h in db_inst.load_habits() if h['id'] == habit_id), None)
    if not record:
        return None
    obj = cls.__new__(cls)
    obj.name = record['name']
    obj.frequency = record['frequency']
    obj.periodicity = record['periodicity']
    obj.id = habit_id
    obj.creation_date = datetime.fromisoformat(record['creation_date'])
    obj.db = db_inst
    return obj
```

Handles creation, completion, and streak tracking for habits. Supports daily and weekly habits, periodicity, and notes.
Every action flows through the Habit class before reaching Database/Analytics.

INTERNATIONAL UNIVERSITY OF APPLIED SCIENCES

Database Module



Stores all habits and completions in SQLite. Default habits and example data allow immediate testing and usage.

Uses SQLite for persistent storage:

```python
def create_tables(self):

    """
    Creates the required tables `habits` and `completions` if they do not exist.
    - `habits`: stores id, name, frequency, periodicity, creation_date
    - `completions`: stores habit_id, completion_date, with a foreign key constraint
    Commits changes to the database.
    """

    cursor = self.conn.cursor()

    cursor.execute("""
    CREATE TABLE IF NOT EXISTS habits (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        name TEXT NOT NULL,
        frequency TEXT NOT NULL,
        periodicity INTEGER NOT NULL,
        creation_date TEXT NOT NULL
    )
    """)

    cursor.execute("""
    CREATE TABLE IF NOT EXISTS completions (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        habit_id INTEGER NOT NULL,
        completion_date TEXT NOT NULL,
        FOREIGN KEY(habit_id) REFERENCES habits(id) ON DELETE CASCADE
    )
    """)

    self.conn.commit()
```

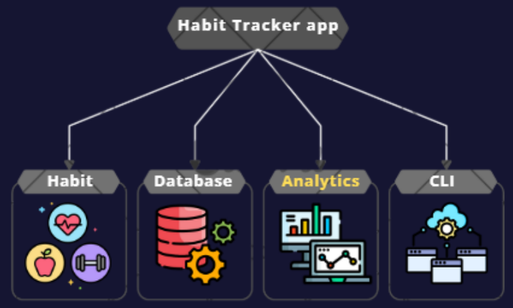
Functions:

- Add, update, delete habits
- Save completion history
- Load habits by frequency

```
31
32  >   def create_tables(self): ···
63
64
65  >   def add_habit(self, name: str, frequency: str = 'daily', periodicity: int = 1) -> int: ···
80
81
82  >   def add_completion(self, habit_id: int, completion_date: Optional[datetime] = None): ···
98
99
100 >   def load_habits(self, frequency: Optional[str] = None) -> List[Dict]: ···
116
117
118 >   def load_completions(self, habit_id: int) -> List[datetime]: ···
129
130
131 >   def delete_habit(self, habit_id: int): ···
142
143
144 >   def reset_to_default(self): ···
158
159
160 >   def add_predefined_habits(self): ···
179
180
181 >   def get_streaks(self, habit_id: int) -> Dict[str, int]: ···
254
255
256 >   def period_summary(self, period: str = "daily") -> Dict[str,int]: ···
280
281
282 >   def completions_in_current_period(self, habit_id: int) -> int: ···
310
311
312 >   def generate_fixture_data(self): ···
348
349
350 >   def reset_empty(self): ···
```

INTERNATIONAL UNIVERSITY OF APPLIED SCIENCES

Analytics Module



Provides insights such as longest streaks, period summaries, recent completions, and habit-specific analytics using functional programming

Output example:

```
PS C:\Users\user\OneDrive\Desktop\Habit Tracker> py analytics.py

📊 Welcome to Habit Analytics!

Here's an overview of your current habits:

ID  | Habit Name          | Freq   | Periodicity | Current | Longest | Recent Completions
-------------------------------------------------------------------------------------------
1   | Drink Water         | daily  | 3           | 0       | 28      | 84 completions, last: 2025-09-14
2   | Stretch             | daily  | 1           | 0       | 28      | 28 completions, last: 2025-09-14
3   | Learn Python        | daily  | 2           | 0       | 28      | 56 completions, last: 2025-09-14
4   | Grocery Shopping    | weekly | 2           | 0       | 4       | 8 completions, last: 2025-09-08
5   | Clean Room          | weekly | 2           | 0       | 4       | 8 completions, last: 2025-09-08

📈 Analytics Summary:

🏆 Longest streak overall: 'Drink Water', 'Stretch', 'Learn Python' with 28 periods.
🗓️ Today's completions: 0 out of 5 habits.
🗓️ This week's completions: 0 out of 5 habits.


ℹ️ Use this table and analytics to understand your habits better.
You can manage habits interactively using: py cli.py

? Move to: (Use arrow keys)
 » 🏠 Run main.py
   📂 Run database.py
   ▶️ Run cli.py
   ❌ Exit
```

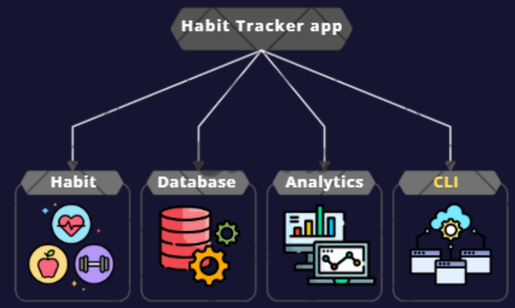
Features:

- List all habits or filter by frequency
- Find longest streak (global or per habit)
- Show completion trends
- Period summaries (completed vs. missed habits)

```
10
11  def list_habits(frequency: str = None) -> List[Dict]: ···
16
17
18  def recent_completions(habit_id: int, n: int = 5) -> List[datetime]: ···
25
26
27  def recent_completions_fp(completions: List[datetime], n: int = 5) -> List[datetime]: ···
32
33
34  def recent_completions_summary(habit_id: int) -> str: ···
43
44
45  def longest_streak_for_habit_fp(habit: Dict, completions: List[datetime]) -> int: ···
80
81
82  def longest_streak_all(): ···
100
101
102 def habit_longest_streak(habit_id: int, db_instance=None) -> int: ···
110
111
112 def period_summary(period="daily"): ···
```

INTERNATIONAL UNIVERSITY OF APPLIED SCIENCES

Command-Line Interface (CLI)



Features:

- Create and delete habits
- Mark habits as completed
- List habits and streaks
- Display analytics (filtering, summaries)

Built with Questionary for structured menus

Provides user interaction point:

```
PS C:\Users\user\OneDrive\Desktop\Habit Tracker> py cli.py

? === HABIT TRACKER MENU === (Use arrow keys)
 » Add a new habit
   List all habits
   Mark habit as done
   Delete a habit
   Show analytics
   Reset to default habits
   Reset entire database (empty)
   Exit
```

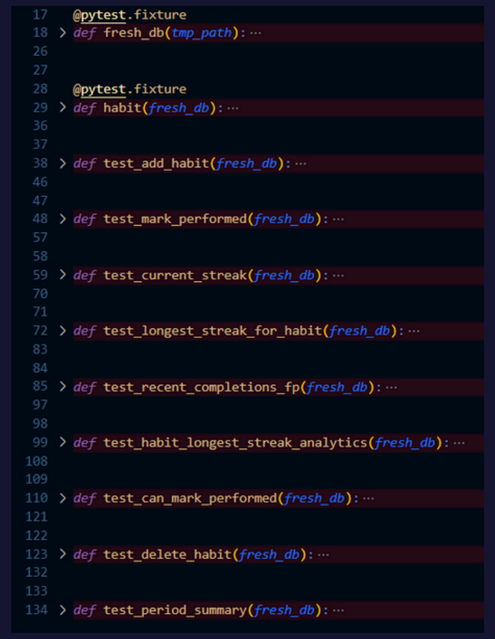
```
14  def format_recent_completions(completions):…
26
27
28  @click.group()
29  def cli():…
34
35  @cli.command()
36  @click.option('--name', prompt='Habit name', help='The name of the habit')
37  @click.option('--frequency', default='daily', type=click.Choice(['daily', 'weekly']), prompt=True)
38  @click.option('--periodicity', default=1, type=int, prompt='Times per period')
39  def add(name, frequency, periodicity):…
54
55
56  @cli.command()
57  def list_habits():…
70
71
72  @cli.command()
73  @click.option('--habit_id', type=int, prompt='Habit ID to mark as done')
74  def done(habit_id):…
93
94
95  @cli.command()
96  @click.option('--habit_id', type=int, prompt='Habit ID to delete')
97  def delete(habit_id):…
103
104
105 @cli.command()
106 def interactive_menu():…
348
349
350
351 if __name__ == "__main__":
352     interactive_menu()
```

INTERNATIONAL UNIVERSITY OF APPLIED SCIENCES

Testing & Quality Assurance

Unit tests cover:

- Habit creation & completion
- Streak calculations
- Database persistence
- Analytics correctness

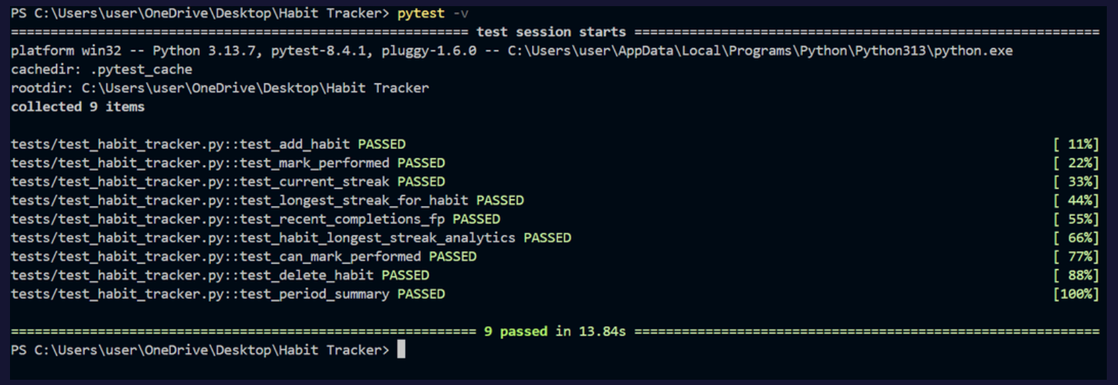


```
17  @pytest.fixture
18  def fresh_db(tmp_path): ...
26
27
28  @pytest.fixture
29  def habit(fresh_db): ...
36
37
38  def test_add_habit(fresh_db): ...
46
47
48  def test_mark_performed(fresh_db): ...
57
58
59  def test_current_streak(fresh_db): ...
70
71
72  def test_longest_streak_for_habit(fresh_db): ...
83
84
85  def test_recent_completions_fp(fresh_db): ...
97
98
99  def test_habit_longest_streak_analytics(fresh_db): ...
108
109
110 def test_can_mark_performed(fresh_db): ...
121
122
123 def test_delete_habit(fresh_db): ...
132
133
134 def test_period_summary(fresh_db): ...
```

Creates a fresh database for each test

```python
@pytest.fixture
def fresh_db(tmp_path):

    """Create a fresh database for each test."""

    db_path = tmp_path / "test_habits.db"
    db = Database(str(db_path))
    db.reset_empty()
    return db
```

Automatic testing supports maintainability
Pytest ensures reliability



```
PS C:\Users\user\OneDrive\Desktop\Habit Tracker> pytest -v
============================= test session starts =============================
platform win32 -- Python 3.13.7, pytest-8.4.1, pluggy-1.6.0 -- C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\user\OneDrive\Desktop\Habit Tracker
collected 9 items

tests/test_habit_tracker.py::test_add_habit PASSED                       [ 11%]
tests/test_habit_tracker.py::test_mark_performed PASSED                  [ 22%]
tests/test_habit_tracker.py::test_current_streak PASSED                  [ 33%]
tests/test_habit_tracker.py::test_longest_streak_for_habit PASSED        [ 44%]
tests/test_habit_tracker.py::test_recent_completions_fp PASSED           [ 55%]
tests/test_habit_tracker.py::test_habit_longest_streak_analytics PASSED  [ 66%]
tests/test_habit_tracker.py::test_can_mark_performed PASSED              [ 77%]
tests/test_habit_tracker.py::test_delete_habit PASSED                    [ 88%]
tests/test_habit_tracker.py::test_period_summary PASSED                  [100%]

============================== 9 passed in 13.84s ==============================
PS C:\Users\user\OneDrive\Desktop\Habit Tracker>
```

INTERNATIONAL UNIVERSITY OF APPLIED SCIENCES

Extra Functionality

Users can view per-habit statistics, reset database to default habits, or clear it completely. Offers flexible data management and testing support.

Detailed analytics per habit:

```
📈 Analytics Summary:

🏆 Longest streak overall: 'Drink Water', 'Stretch', 'Learn Python' with 28 periods.
🗓️ Today's completions: 0 out of 5 habits.
🗓️ This week's completions: 0 out of 5 habits.

📊 Daily Summary: 0 completed, 5 missed
📊 Weekly Summary: 0 completed, 5 missed

--- Per Habit Analytics ---

Habit: Drink Water
   ✅ 84 completions, last: 2025-09-14
   🔥 Longest streak: 28
   🕒 Last completions: 2025-09-14, 2025-09-13, 2025-09-12, 2025-09-11, 2025-09-10
----------------------------------------

Habit: Stretch
   ✅ 28 completions, last: 2025-09-14
   🔥 Longest streak: 28
   🕒 Last completions: 2025-09-14, 2025-09-13, 2025-09-12, 2025-09-11, 2025-09-10
----------------------------------------

Habit: Learn Python
   ✅ 56 completions, last: 2025-09-14
   🔥 Longest streak: 28
   🕒 Last completions: 2025-09-14, 2025-09-13, 2025-09-12, 2025-09-11, 2025-09-10
----------------------------------------

Habit: Grocery Shopping
   ✅ 8 completions, last: 2025-09-08
   🔥 Longest streak: 4
   🕒 Last completions: 2025-09-08, 2025-09-01, 2025-08-25, 2025-08-18
----------------------------------------

Habit: Clean Room
   ✅ 8 completions, last: 2025-09-08
   🔥 Longest streak: 4
   🕒 Last completions: 2025-09-08, 2025-09-01, 2025-08-25, 2025-08-18
----------------------------------------

ℹ️ Use this table and analytics to understand your habits better.
```

Flexible database reset options:

```
Reset to default habits
Reset entire database (empty)
```

```
? === HABIT TRACKER MENU === Reset to default habits
? This will delete ALL habits and completions. What do you want to do? Yes, reset to defaults
✅ All habits reset to defaults with fixture data for 4 weeks.

ID  | Name            | Freq   | Periodicity | Current | Longest | Recent completions
-------------------------------------------------------------------------------------------
1   | Drink Water     | daily  | 3           | 28      | 28      | 84 completions, last: 2025-09-19
2   | Stretch         | daily  | 1           | 28      | 28      | 28 completions, last: 2025-09-19
3   | Learn Python    | daily  | 2           | 28      | 28      | 56 completions, last: 2025-09-19
4   | Grocery Shopping| weekly | 2           | 4       | 4       | 8 completions, last: 2025-09-15
5   | Clean Room      | weekly | 2           | 4       | 4       | 8 completions, last: 2025-09-15
```

```
? === HABIT TRACKER MENU === Reset to default habits
? This will delete ALL habits and completions. What do you want to do? (Use arrow keys)
 » Yes, reset to defaults
   No, cancel
   View default habits
```

```
? === HABIT TRACKER MENU === Reset entire database (empty)
? This will delete ALL habits and completions and leave the database empty. Continue? (Use arrow keys)
 » Yes
   No
```

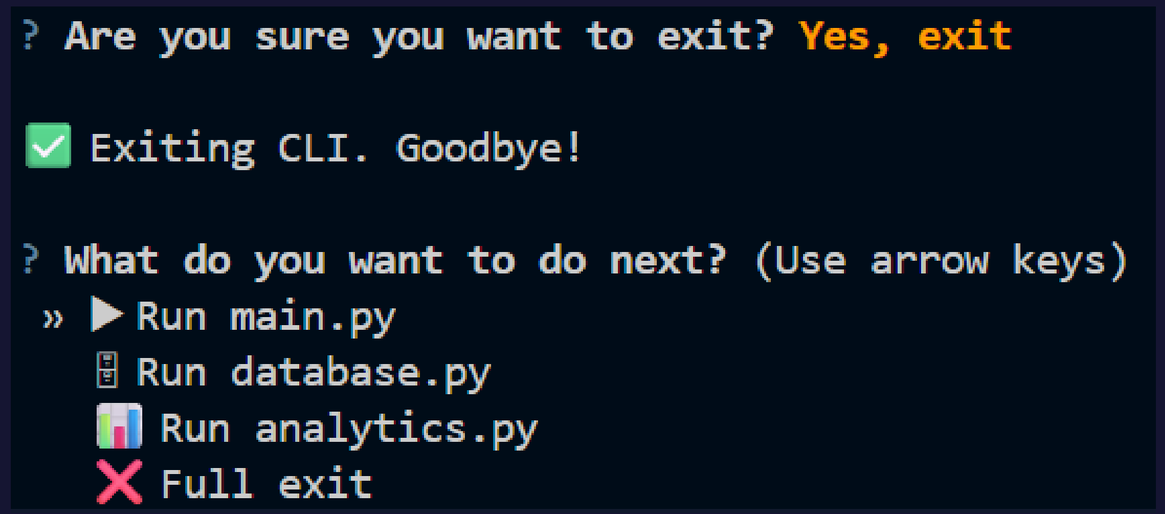
```
? === HABIT TRACKER MENU === Reset entire database (empty)
? This will delete ALL habits and completions and leave the database empty. Continue? Yes
✅ Database completely emptied.

? === HABIT TRACKER MENU === List all habits
ID  | Name            | Freq   | Periodicity | Current | Longest | Recent completions
```

iu
INTERNATIONAL
UNIVERSITY OF
APPLIED SCIENCES

Extra Functionality

Post-Exit Integration Menu:

Running any module—main.py, cli.py, analytics.py, or database.py—shows default habits first. Changes made via the CLI, like adding, completing, or deleting habits, are reflected across all modules. At the end of each run, a menu lets users navigate seamlessly between modules, keeping habit data consistent and up-to-date.



iu
INTERNATIONAL
UNIVERSITY OF
APPLIED SCIENCES

Solution Overview

The Habit Tracker is built with a modular architecture combining object-oriented programming for the Habit class and functional programming for analytics. Data is stored persistently in SQLite, ensuring reliability across sessions. Users interact through a simple command-line interface to create, delete, and complete habits, while analytics provide insights such as streaks and filtered views. The solution meets the main requirements: daily and weekly habits, predefined examples with four weeks of data, persistent storage, and automated testing with pytest. This results in a lightweight, reliable, and extensible backend for habit tracking.