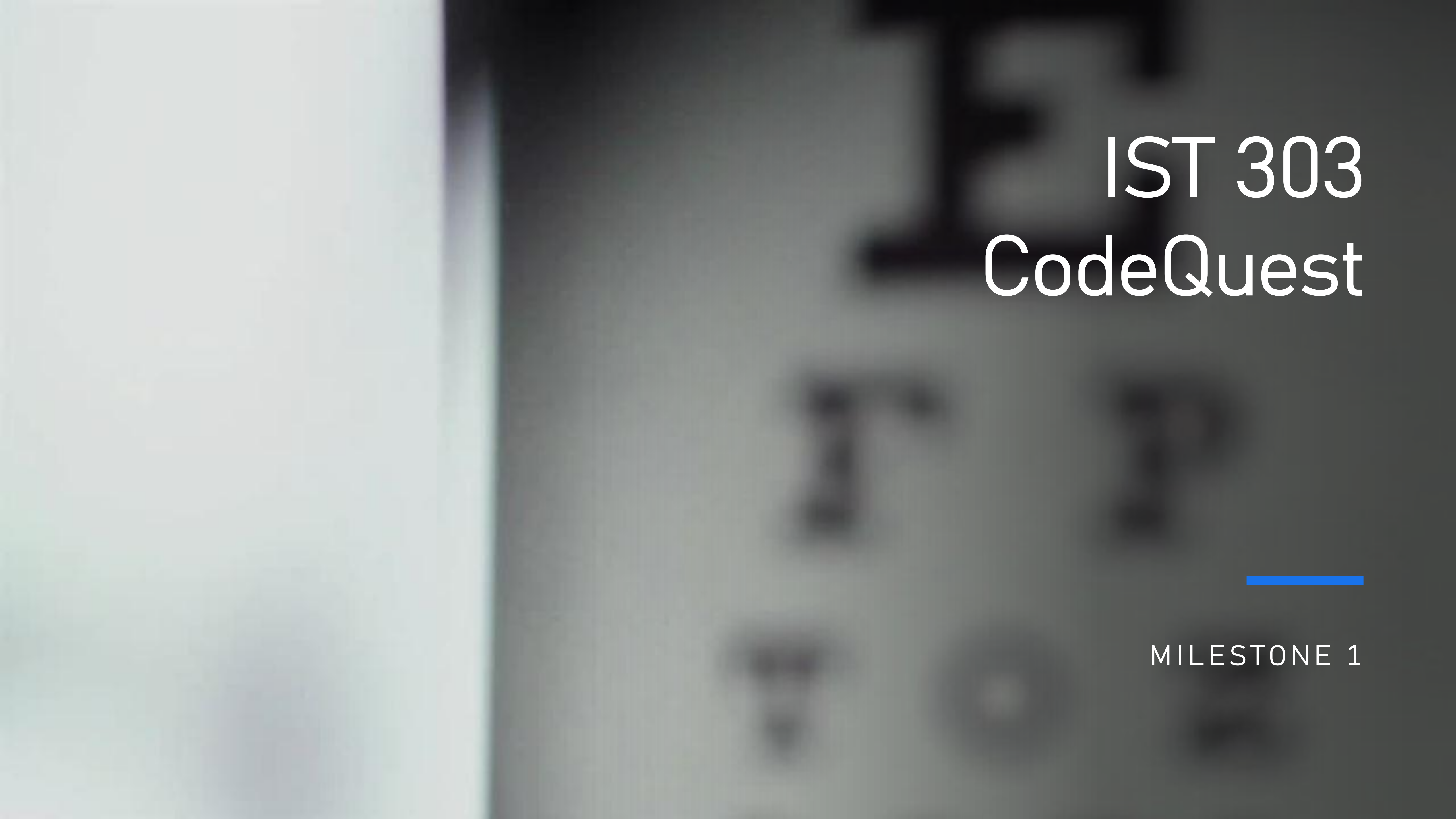


IST 303 CodeQuest

MILESTONE 1

Overview

CodeQuest – Python Mastery

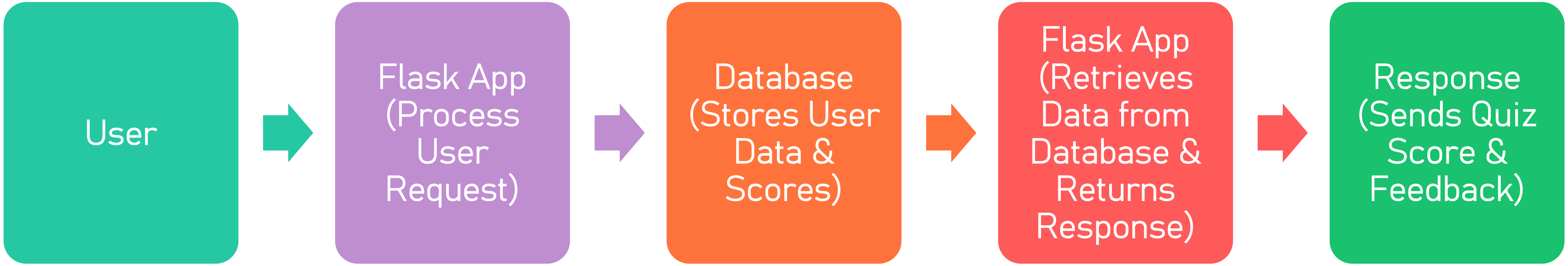
Team Members

- Emmanuel
- Hla Win Tun
- Paniz
- Rogelio

Project Description:

- A Python quiz web application designed to help users test their Python knowledge.
- Includes user authentication, a quiz interface, and score tracking.
- Built using FLASK, SQLite, and HTML/CSS for UI

Architecture Design


User
Flask App (Process User Request)
Database (Stores User Data & Scores)
Flask App (Retrieves Data from Database & Returns Response)
Response (Sends Quiz Score & Feedback)

The user interacts with the Flask App, which processes requests, stores quiz data in the database, and returns results in real time.

How It Works

1

Flask App: Handles authentication & quiz logic

2

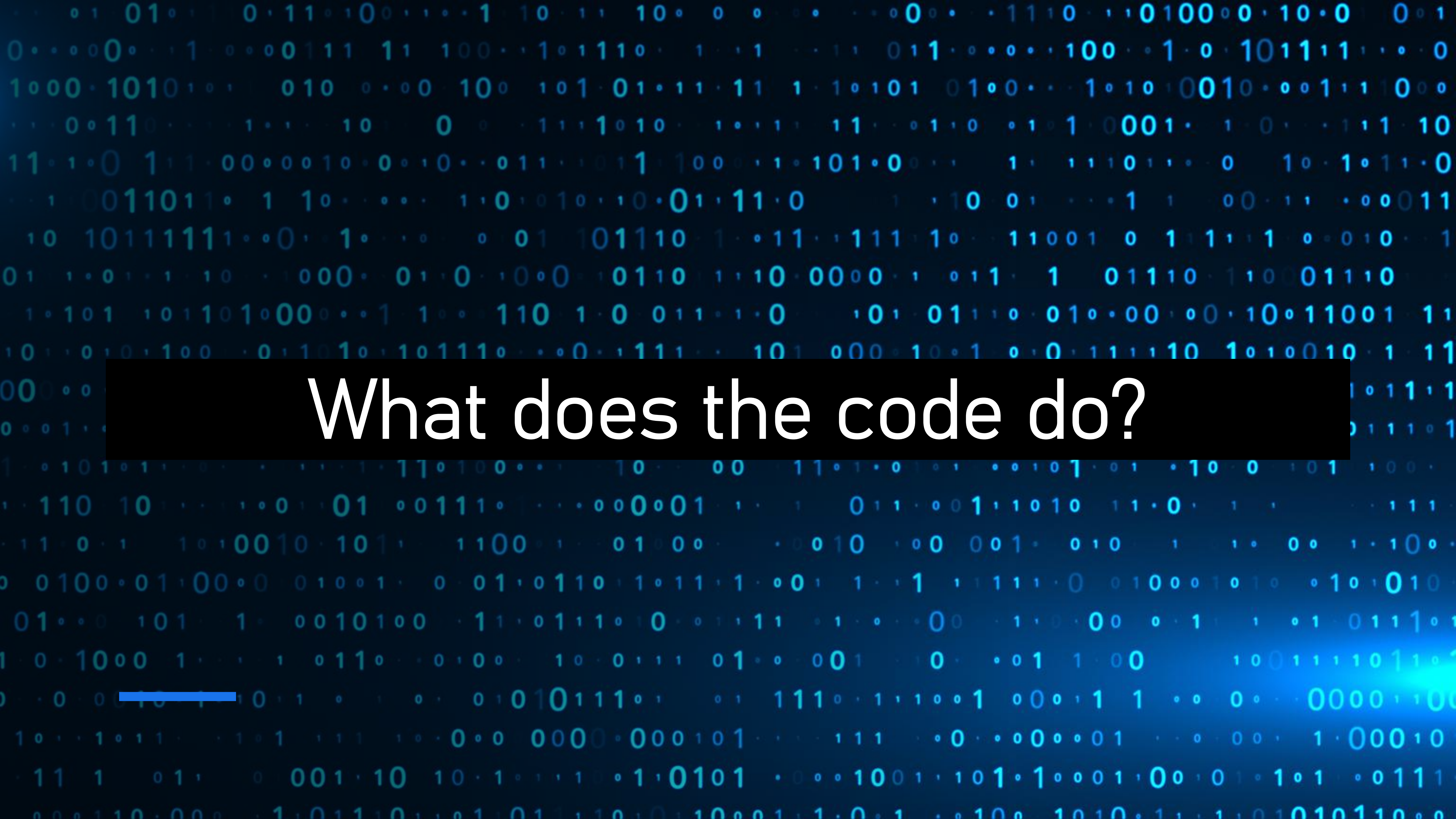
SQLite Database: Stores user credentials & scores

3

Response Handling: Sends real-time quiz results

4

Tech Stack: Flask (Backend), SQLite (Database), HTML/CSS (UI)



What does the code do?

Milestone 1.0 Goals

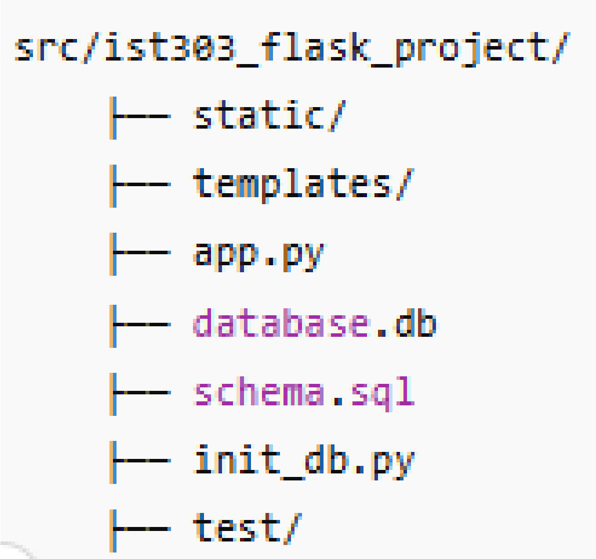
What we aimed to achieve:

- **Core Features:** User registration, login, quiz functionality, score tracking
- **Agile Development:** Sprint planning, stand-up meetings, burndown tracking
- **Deliverable:** A functional **Minimum Viable Product (MVP)**

Why this is important?

- Sets the foundation for **future enhancements** in Milestone 2.0

GitHub Repository Overview



```
src/ist303_flask_project/
    ├── static/
    ├── templates/
    ├── app.py
    ├── database.db
    ├── schema.sql
    ├── init_db.py
    ├── test/
```

Folder Structure:

- What's inside?
  - app.py: Main Flask application logic
  - database.db: SQLite database
  - schema.sql: Database schema
  - test/: Contains test_quiz.py for testing

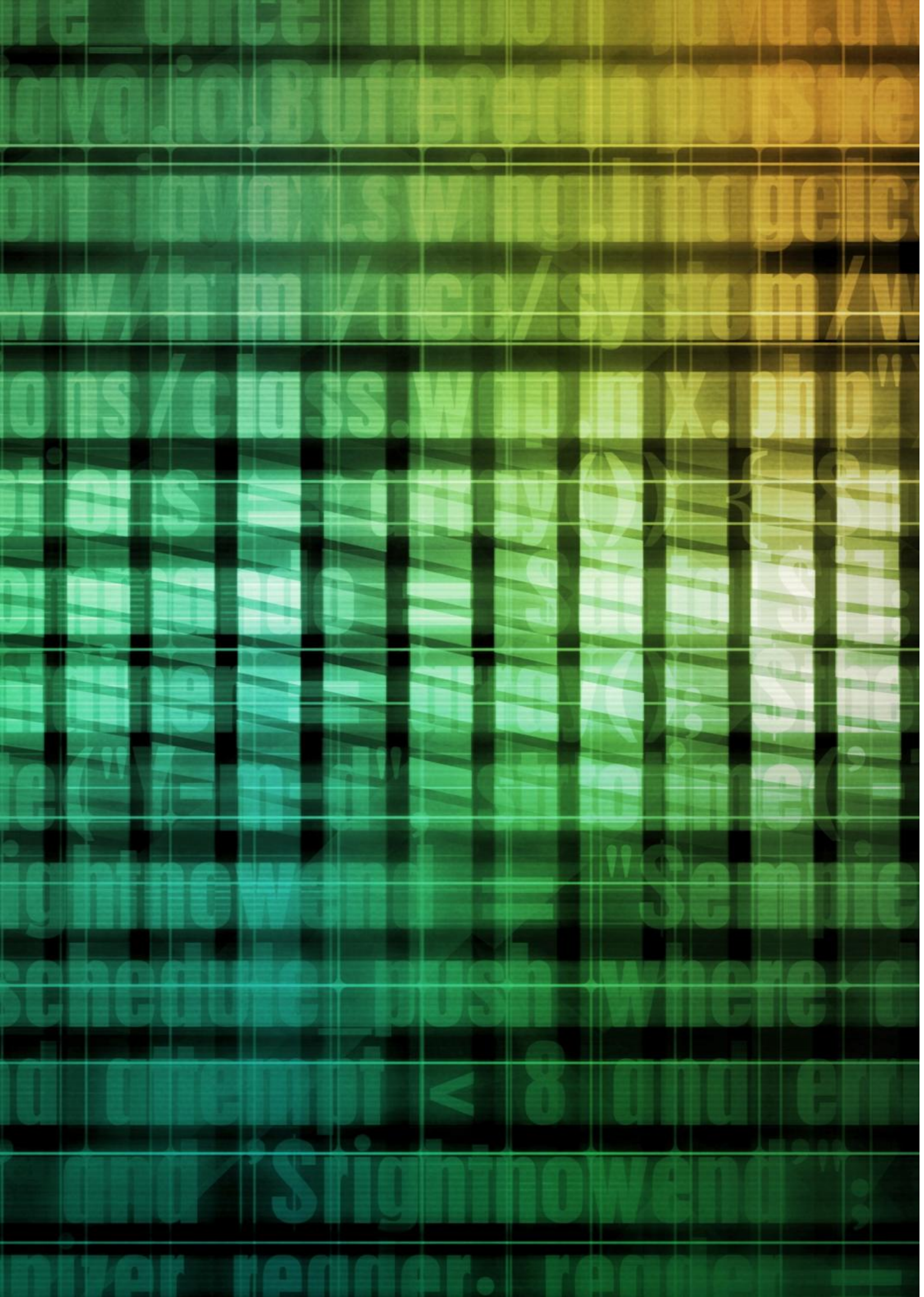
Code Breakdown (Frontend – HTML & CSS)

Templates:

- templates/login.html
- templates/quiz.html
- templates/result.html

Styling:

- static/style.css for UI improvements



Code Breakdown (app.py)

- Key Functions:
  - **User Authentication**: login(), register()
  - **Quiz Logic**: start_quiz(), evaluate_answer()
  - **Score Tracking**: update_score()

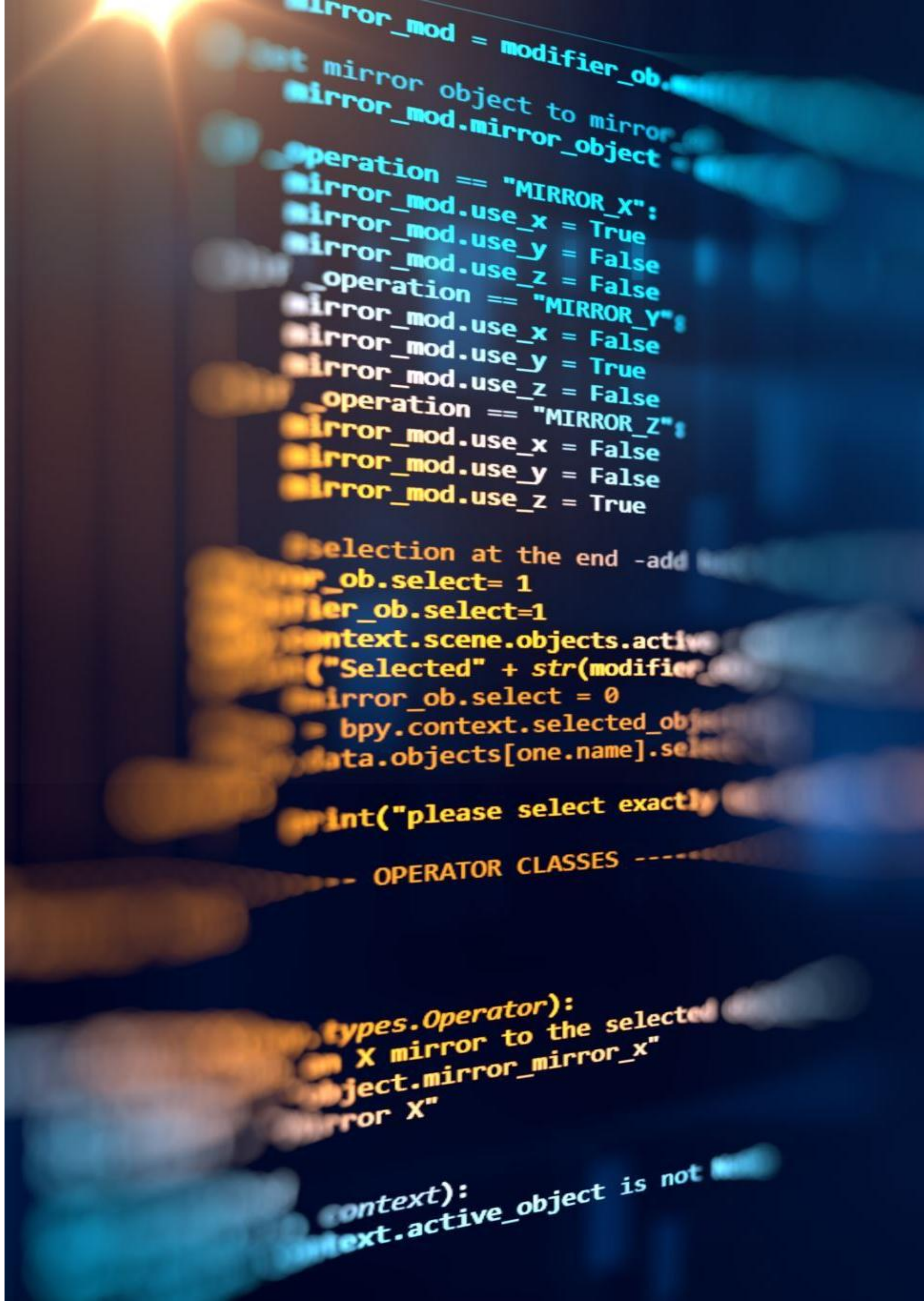
Code Breakdown

Database Schema (schema.sql):

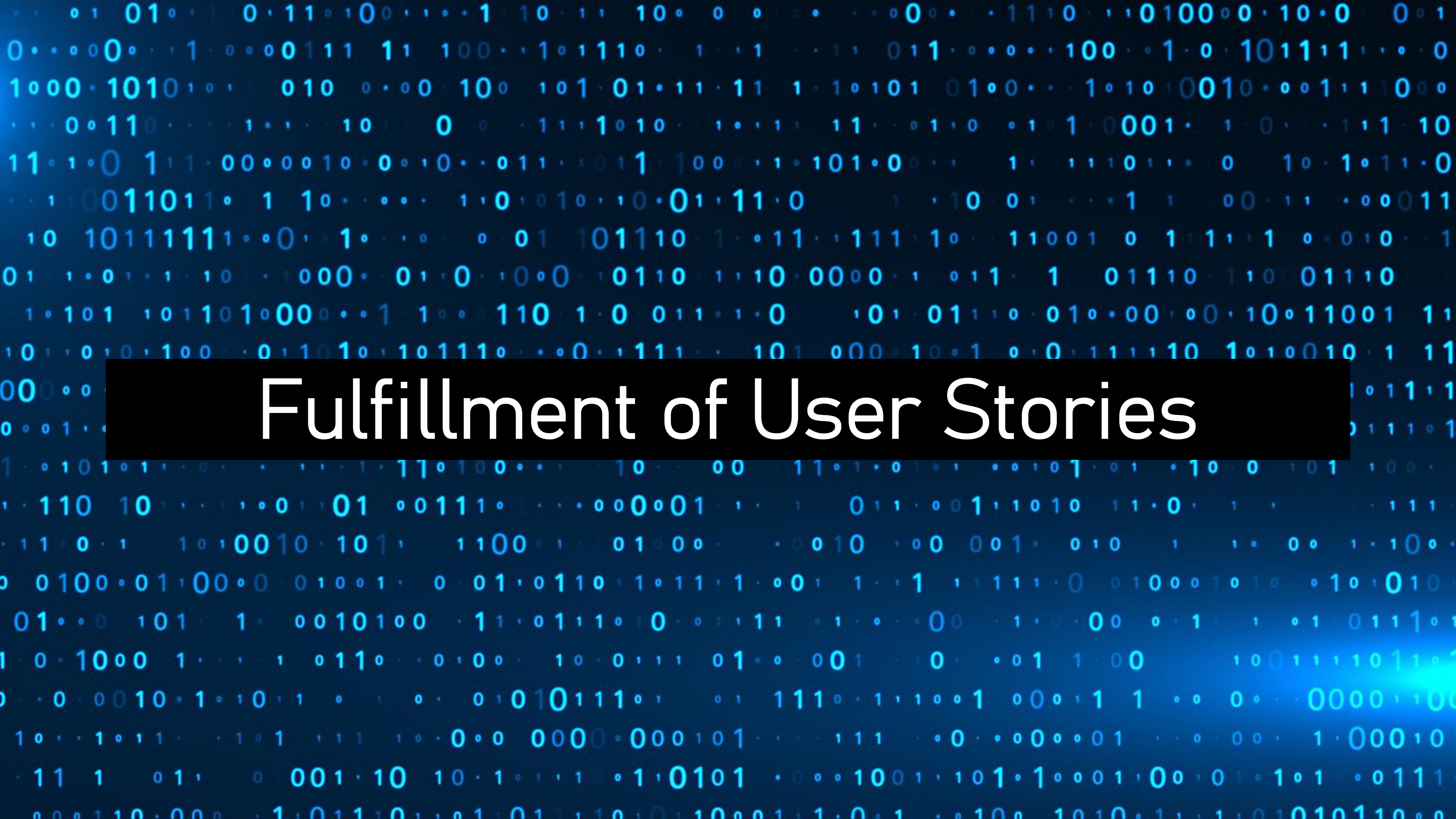
```
CREATE TABLE users (
    id INTEGER PRIMARY KEY,
    username TEXT NOT NULL UNIQUE,
    password TEXT NOT NULL
);
CREATE TABLE quiz_questions (
    id INTEGER PRIMARY KEY,
    question TEXT NOT NULL,
    answer TEXT NOT NULL
);
```

How is data stored & retrieved?

- Uses **SQLite**
- init_db.py initializes the database



Demo



Fulfillment of User Stories

User Authentication (Register & Login)

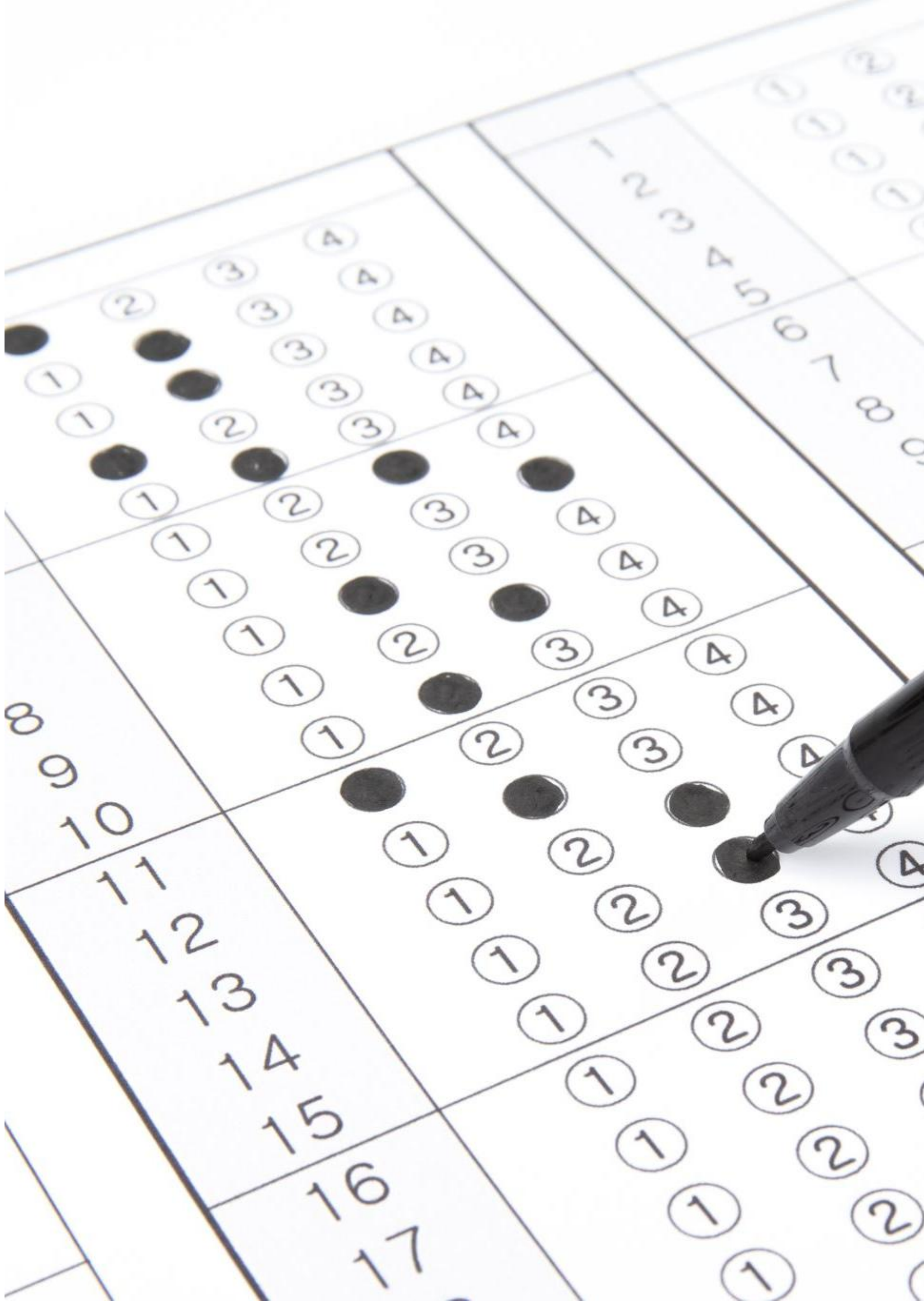
- Feature:
  - Users can **register** with a username & password.
  - Secure **login system** using session management.
- Code Mapping:
  - app.py -> (login(), register())
- User Story Mapping:
  - "As a user, I want to register and log in so that I can save my quiz progress."

Quiz Functionality

- Feature:
  - Users can **start a quiz** and answer Python-related questions.
  - Answers are validated in real-time.
- Code Mapping:
  - app.py (quiz logic)
  - templates/quiz.html
- User Story Mapping:
  - "As a user, I want to answer quiz questions and get feedback."

Score Tracking

- Feature:
  - Users **see their final score** after completing the quiz.
- Code Mapping:
  - app.py (score tracking)
  - templates/result.html
- User Story Mapping:
  - "As a user, I want to see my final score after completing the quiz."



UI Improvements

- Feature:
  - **Color-coded responses** for correct/incorrect answers.
  - **Navigation buttons** added for usability.
- Code Mapping:
  - style.css (UI styling)
- User Story Mapping:
  - "As a user, I want an intuitive quiz interface that highlights correct/incorrect answers."

Agile Methods

Agile Progress

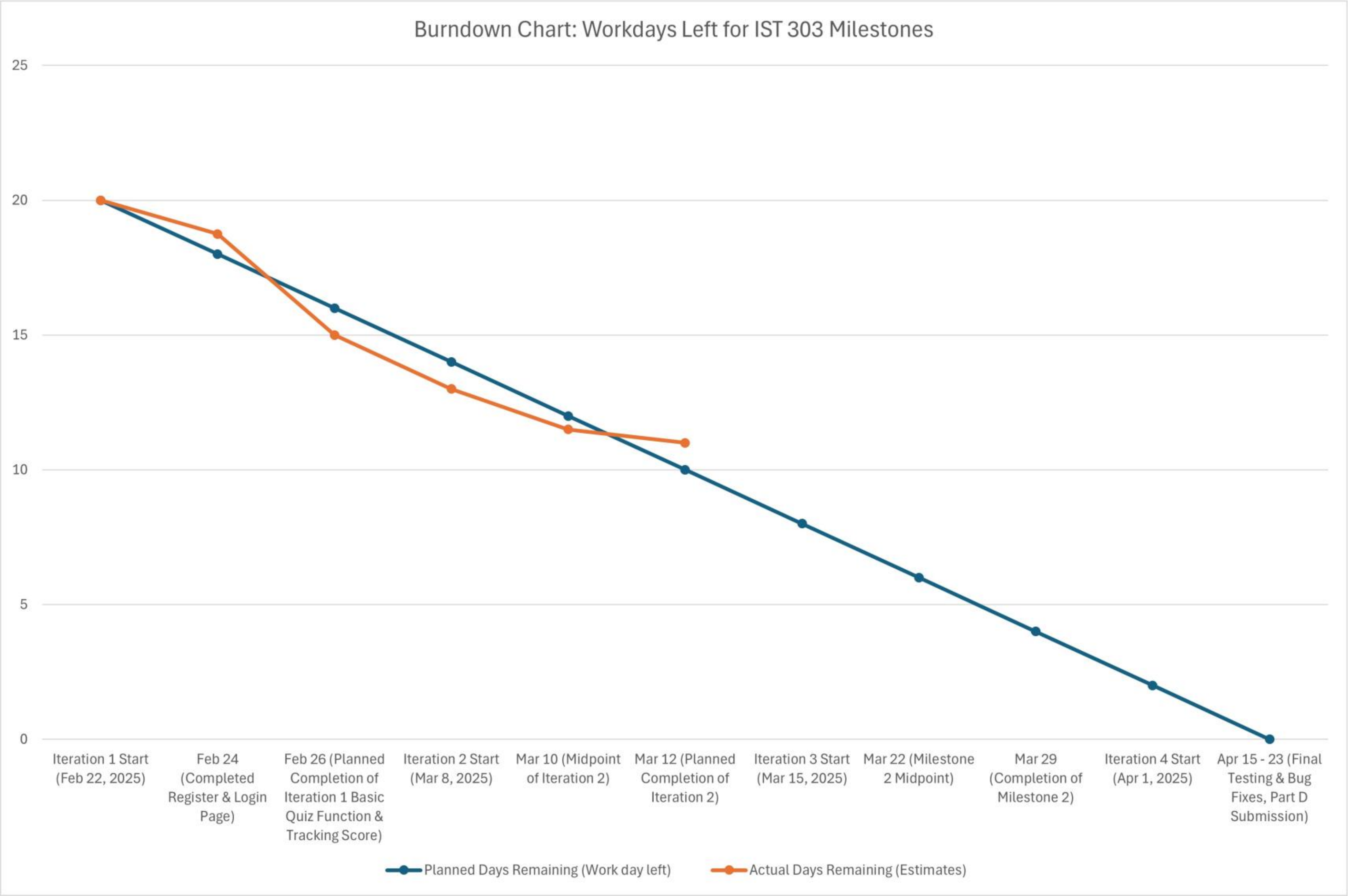
Feature:

- Visual progress tracking using a **burndown chart**
- Documented **team meetings, sprints, and task breakdowns**

Burndown Chart Highlights:

- **Planned vs. actual progress** tracking
- **Agile methods used:** iterations, regular sync up (minimum 2 meeting/week, sprint inspiration)

Our Burndown Chart & Rate



Iteration Breakdown

Iteration	Duration	Tasks Completed	Hours Spent	Velocity (Story Points)
Iteration 1	Feb 22 – Mar 7	Core Functionality & UI	40 hours	20 points
Iteration 2	Mar 8 – Mar 22	Enhancements & Testing	40 hours	14 points

Steady sprint velocity ensured core features were completed on time.

Iteration 1 focused on **core quiz functionality** (~20 points).

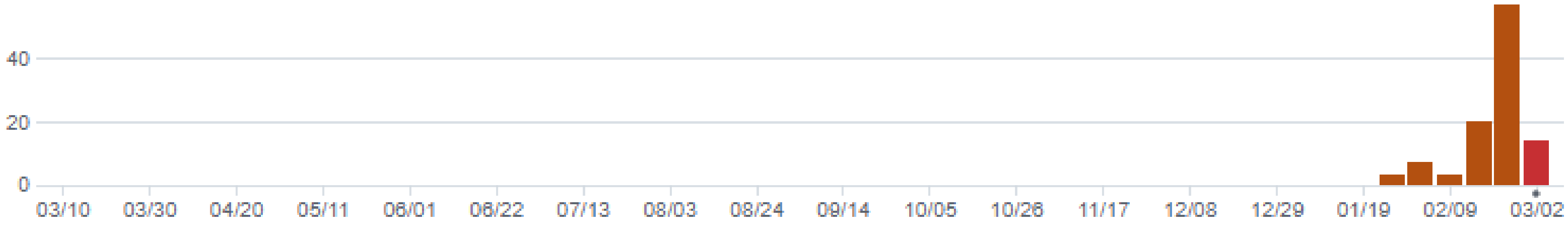
Iteration 2 focused on **enhancements & testing** (~14 points).

Minor UI delays in Iteration 1 were adjusted in Iteration 2.

Final MVP was completed before the March 12 deadline.

GitHub Commits & Agile Workflow

- Proof of Agile workflow:
  - Frequent **commits**
  - **Branching strategy:** Feature branches merged into main
  - Team members actively committing & reviewing code



Agile Documentation Summary

- Key Artifacts:
  - Sprint Planning Documents
  - Meeting Notes Folder (meeting_notes/)
  - README.md
  - PART_B.md
  - PART_C.md

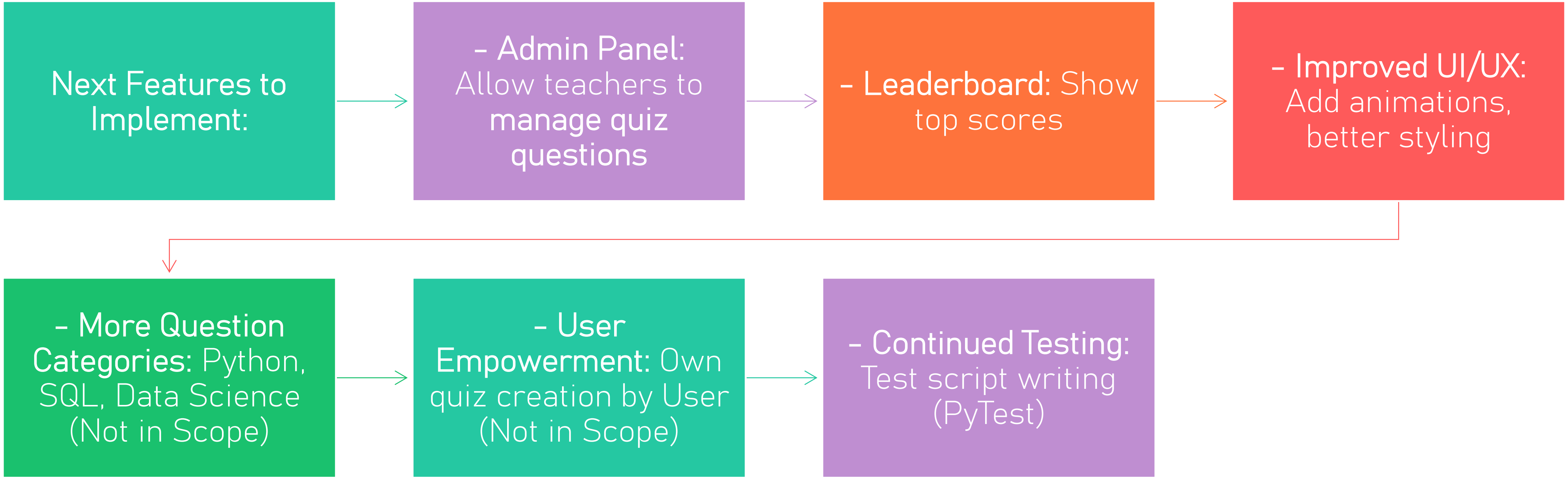
Code Testing

Code Testing Testing Methodology

- Manual Testing:
  - Tested login, quiz, and score tracking
  - Multiple iterations with adjustments
- Unit Tests (test/test_quiz.py)
- Edge Cases Handled:
  - Invalid login attempts
  - Empty inputs in quiz answers
  - Correct vs. incorrect answer validation

What's Next? (Milestone 2.0)

Milestone 2.0 Goals



Next Features to Implement:

- Admin Panel: Allow teachers to **manage quiz questions**

- Leaderboard: Show top scores

- Improved UI/UX: Add animations, better styling

- More Question Categories: Python, SQL, Data Science (Not in Scope)

- User Empowerment: Own quiz creation by User (Not in Scope)

- Continued Testing: Test script writing (PyTest)

Next steps & Final Timeline

Remaining Workload:

Task	Planned Completion
Finalize Admin Panel	March 20, 2024
Implement Leaderboard	March 27, 2025
UI Enhancement	April 5, 2025
Final Testing	April 15, 2025
Part D	April 23, 2025

Thank You!

Breakdown by Iteration
Iteration 1: Core Quiz Functionality & UI (Feb 22 – Mar 7)

Task	Hours	Story Points
Create register page	5	2
Create login page	5	2
Implement question display	5	2
Capture user input for answers	5	2
Validate correct/incorrect answers	5	2
Format quiz UI	6	2
Store quiz questions in JSON	7	3
Error handling & input validation	7	3
Basic score tracking	5	2

Total for Iteration 1: 20 story points

Iteration 2: Enhancements & Testing (Mar 8 - Mar 12)

Task	Hours	Story Points
Improve UI with color-coded response	5	2
Implement retry logic	5	2
Develop admin panel quiz management	15	5
Conduct internal testing & bug fixes	10	3
Prepare for Part C presentation	5	2

Total for Iteration 2: 14 story points