Projects Asignments for 2 week Internship Recruitement

🗓️ Challenge Timeline

- **Start Date:** July 24, 2025 (00:00 PKT)
- **End Date:** July 31, 2025 (23:59 PKT)

Students can begin submitting their projects from **July 26, 2025** onward. All final submissions must be uploaded at: 🔗 **https://swiftdynamics.io/python-ai-verge-projects**

📊 Evaluation Criteria

Each project will be judged based on:

- **Code Functionality** – Does the program work correctly and follow the original project goal?
- **Code Structure** – Is the code clean, organized with functions, and easy to read?
- **File Usage** – Does the program read/write to files properly and use them meaningfully?

🎓 Internship Selection Criteria

For the 2-week remote internship, we'll look at:

- **Creativity** – Did the student add anything extra or go beyond the basic instructions?
- **Problem Solving** – Did they handle errors, edge cases, or unexpected input well?
- **Consistency** – Did they stay active, follow the plan, and submit on time?

🧠 Challenge Solution Guidelines

To complete the project successfully, students should:

- Stick to the assigned topic and complete all listed features.
- Use functions, loops, conditionals, and file handling properly.
- Make the project easy to use, with a clear menu or steps.
- Save their final `.py` file(s) and any `.txt` or `.csv` used.
- Submit only their own work — no copy-pasting from others.

📤 How to Submit

- You must enter:
    - **Your full name**
    - **Valid email address**
    - **Public GitHub link** to your project
- Certificates will be sent to the email you provide.

🏆 Internship Selection

> Only 3 or 4 students will be slected....

Top students will be chosen based on:

- Creativity or extra features
- Good error handling
- On-time and complete submission

---

1. Affan Owais – "Quiz Builder and Score Tracker"

Your task is to build a Python-based **Quiz Builder**. This project allows users to **take a quiz**, see their **score**, and **save their attempts**. You'll write questions and answers into a file, then build a program that loads the quiz, asks the questions, scores the answers, and saves the result to another file.

- Plan the quiz – decide on 5–10 questions. Learn how to write them to a file.
- Learn how to read from the file and ask one question at a time.
- Store the user's answers and calculate the score.
- Add functions to organize your code better (e.g., `ask_question()`, `check_answer()`).
- Learn file handling to save the player's score with their name.
- Add retry or exit option.
- Polish the interface and handle edge cases like empty input.

2. Abdul Bari – "Digital Diary with Search Feature"

Create a **digital diary** app that lets a user write daily entries, view them later, and **search by keyword**.

- Create a welcome message and a simple menu system.
- Build the feature to write diary entries and save them to a file with the date.
- Build a feature to view past entries.
- Add a search function that looks for a word or phrase in previous entries.
- Organize your code using functions like `write_entry()` and `search_entry()`.
- Add checks to make sure the file is not empty and display results nicely.
- Make it user-friendly with colors or headings (using `print()` creatively).

---

3. Ayaan Farrukh Memon – "Python Flashcards Game"

Build a **flashcards learning game**. Users will be shown a question (like a math problem or a definition), and they must type the answer. The game will check the answer and keep score.

- Design how the flashcards will work — use a dictionary or file to store questions/answers.
- Create a function that asks questions randomly.
- Count correct/incorrect answers and display the result.
- Save the session's score in a file.
- Allow the user to review flashcards where they answered incorrectly.
- Make it loop until the user quits.
- Polish and organize your code.

---

4. Muhammad Hamza – "Simple Banking System"

Build a console-based banking system where a user can create an account, deposit money, withdraw funds, and check balance. All data should be stored in a file.

- Use functions like `create_account()`, `deposit()`, `withdraw()`, and `check_balance()`.
- Save account details to a file using the account holder's name or ID.
- Read from the file every time a transaction is performed.
- Validate entries like ensuring withdrawal amount doesn't exceed balance.
- Display transaction history at the end of each session.
- Use loops and conditionals for smooth user navigation.
- Focus on clear messages and user prompts.

---

5. Jazil – "Personal Expense Tracker"

Design a Python program where users can enter daily expenses and track their weekly spending.

- Let the user input date, item name, and amount spent.
- Save all entries to a file (e.g., `expenses.txt`).
- Show a summary of spending by day or item.
- Create a menu to view total, highest, or average spending.
- Add functions like `add_expense()`, `view_summary()`, and `search_expense()`.
- Use loops to enter multiple expenses without restarting.
- Include input validation and a nice summary at the end.

---

6. Abdul Rehman – "Library Book Manager"

Make a system to manage books in a small library. Users can add, view, lend, or return books.

- Store book info (title, author, status) in a file.
- Use functions for each task: `add_book()`, `lend_book()`, `return_book()`, `list_books()`.
- Check for duplicates or unavailable books.
- Loop through the menu until the user exits.
- Maintain a log file to track when books were borrowed or returned.
- Practice file handling and string formatting for clean outputs.

---

7. Ibrahim Yousuf – "Math Puzzle Generator and Solver"

Create a puzzle generator that gives users math challenges like equations, number patterns, or small recursive puzzles.

- Randomly generate puzzles using `random` (addition, multiplication, or patterns).
- Take user input and verify their answer.
- Use functions like `generate_puzzle()` and `check_solution()`.
- Keep track of the score and time taken per puzzle.
- Save puzzles and user responses to a file.
- Let the user retry wrong answers and view their past attempts.
- Recursion can be used for pattern puzzles (e.g., factorial).

8. Ismail Yousuf – "Guess the Secret Word Game (with Hints)"

Create a guessing game where a random word is picked, and the user guesses letters one by one.

- Include a list of words stored in a file.
- Display dashes (`_ _ _`) and reveal letters when guessed correctly.
- Allow a fixed number of attempts.
- Add a hint for each word from a file (stored as word-hint pairs).
- Organize your game using functions like `load_words()`, `guess_letter()`, `show_progress()`.
- Add a replay feature and score tracking.

---

9. Hafiz Muhammad Mursaleen – "Weather Logger Simulator"

Build a program that simulates weather logging for a week.

- Ask the user to input daily weather info (temperature, condition).
- Store each day's data in a file.
- Provide summary statistics: average temperature, hottest/coldest day.
- Functions like `log_weather()`, `view_summary()`, `search_day_weather()` help organize the code.
- Use string formatting to print results in a table.
- Allow editing entries if mistakes are made.
- Explore conditions and loops for data checking.

---

10. M. Huzaifa – "Contact Book Application"

Develop a digital contact book to store, search, and update contacts.

- Ask for name, phone number, and email.
- Store data in a file in a readable format (CSV or plain text).
- Add search feature by name or number.
- Allow updating or deleting a contact.
- Use functions to manage operations.
- Validate phone numbers and emails using string methods.
- Practice file reading/writing with record searching.

---

11. Wahib – "Digital Clock with Alarm Simulation"

Create a text-based digital clock and alarm simulator.

- Show current time (manually or using `time` module).
- Let users set alarms for specific times.
- Simulate an alert using a message or sound (optional).
- Use loops to keep the program running.
- Store alarm history in a file.
- Functions: `set_alarm()`, `check_alarm()`, `show_clock()`.
- Learn how time works in Python and simulate scheduling.

---

12. Ayesha Kashif – "Student Marks Report Generator"

Create a marks report card system for multiple students.

- Ask for student name, subjects, and marks.
- Save each record in a file.
- Generate a report with percentage, grade, and remarks.
- Functions: `add_student()`, `generate_report()`, `view_all_reports()`.
- Use loops to enter data for multiple students.
- Include conditionals to assign grades (A, B, C, etc.).
- Summarize results at the end and format the output nicely.

---

13. Ibrahim Talha – "Simple Calendar Maker"

Build a Python tool to create a calendar for any month/year.

- Use Python's `calendar` module.
- Take user input for month and year.
- Display calendar in a clean format.
- Allow users to write notes/reminders for specific dates.
- Store notes in a file linked to the date.
- Functions: `show_calendar()`, `add_note()`, `view_notes()`.
- Encourage use of dictionaries or file structures for date-note mapping.

---

14. M. Anas – "Treasure Hunt Text Game"

Design a text-based adventure game where the player searches for treasure.

- Create rooms or locations using a map-like structure.
- Use loops to move player from room to room.
- Allow actions like pick up item, fight enemy, or solve puzzle.
- Use functions like `move_player()`, `check_item()`, `solve_puzzle()`.
- Save game state in a file so user can continue later.
- Add random events using `random`.
- Encourage use of nested functions or recursion for puzzle solving.

---

15. M. Ameen – "Recipe Book and Grocery List Manager"

Create an app that stores recipes and builds grocery lists.

- Let users enter recipes with ingredients and steps.
- Save each recipe in a file.
- Build a grocery list by selecting recipes.
- Combine ingredients from multiple recipes into one shopping list.
- Functions: `add_recipe()`, `view_recipe()`, `create_list()`.
- Use file reading to search and combine data.
- Explore string manipulation and list operations.

---

16. Sarah Essa – "Animal Facts Encyclopedia"

Build a simple program that gives facts about animals.

- Store animal names and facts in a file.
- Let users search for an animal and display a fun fact.
- Use functions like `search_animal()`, `add_fact()`, `list_animals()`.
- Organize data in dictionary or file format.
- Allow the user to add new animals and facts.
- Practice file handling, search, and formatting outputs.
- Encourage curiosity and structured input.

---

17. Baria Imtiaz – "Story Generator with Recursion"

Make a creative story generator that builds random stories using templates and recursion.

- Store characters, locations, actions in separate files.
- Randomly choose elements and generate sentences.
- Use recursion to build a story paragraph by paragraph.
- Functions: `generate_sentence()`, `build_story()`.
- Allow saving generated stories.
- Use conditionals to vary structure.
- Encourage creativity with structure and loops.

---

18. M. Yahya – "Fitness Tracker and Water Reminder"

Build a tracker for daily fitness activity and water intake.

- Let user input daily exercise time and glasses of water.
- Show charts (text-based) of weekly progress.
- Save logs in a file.
- Functions: `add_entry()`, `view_summary()`, `compare_days()`.
- Give feedback like "Great Job!" or "Try more tomorrow".
- Use loops, conditionals, and data averaging.
- Make the tracker fun and motivational.

---

19. Amaan Sumair – "Simple File Explorer and Organizer"

Create a tool that reads a folder and categorizes files (by type, date, size).

- Simulate a folder (you can just use text data for practice).
- List all files and group them by extension.
- Move or rename files based on rules.
- Functions: `list_files()`, `sort_by_type()`, `rename_file()`.
- Store logs of changes in a file.
- Use string and list operations.
- Reinforce loops, conditionals, and file naming concepts.
