InCollege Project: Log In, Part 1

Objective: This week, your team will lay the foundational stone for InCollege by analyzing a complex computing problem by implementing the core user authentication system. You are to apply principles of computing in creating your solution. This includes user registration and login functionality, as well as handling initial navigation for the application's alpha version. All program input will be read from a file, all output will be displayed on the screen, and then that same output will also be written to a file in order to preserve the program's output.

Focus Areas:

1. **User Interface (Simulated) & I/O Requirements:** The InCollege application will present a user with an initial screen that provides them with two different options. The user will be presented with the option of logging in using an existing InCollege account or creating a new InCollege account. While we won't be building a graphical user interface in COBOL, your program should simulate this interaction through clear console prompts and input/output.
    - **Input:** All user input (e.g., username, password, menu selections) will be read from a predefined input file.
    - **Output Display:** All program output (e.g., prompts, confirmation messages, error messages) must be displayed on the screen (standard output).
    - **Output Preservation:** *The exact same output displayed on the screen must also be written to a separate output file* for testing and record-keeping purposes. **Note that this means that all input from the user ALSO has to be written to the output file.**
2. **Account Management:**
    - **New Account Creation:** Support for up to 5 unique student accounts (unique user name and secure password) will be provided.
        - **Password Requirements:** Passwords must have a minimum of 8 characters, a maximum of 12 characters, at least one capital letter, one digit, and one special character. Your COBOL program needs to validate these criteria during registration.
        - **Persistence:** Any student accounts that are created will be saved and will be read back into the system the next time the application is started up. For this alpha version, a simple sequential file or similar COBOL-manageable data structure is acceptable for persistence.
        - **Account Limit Message:** The 6th attempt to create a student account will result in the message "All permitted accounts have been created, please come back later" to be displayed to the user.
    - **Login Functionality:**
        - **Successful Login:** If the student successfully logs in by entering a recognized username / password, the system will tell them "You have successfully logged in".

        - **Failed Login:** If they enter an incorrect username/password, the system will tell them "Incorrect username/password, please try again" and will allow the student to attempt to log in again.
        - **Unlimited Attempts:** An unlimited number of log in attempts will be permitted.
3. **Initial Post-Login Navigation (Simulated):**
    - Once logged in, additional options will be provided that will allow the user to search for a job, find someone that they know, or learn a new skill.
    - **"Under Construction" Messages:** The job search/internship option will result in an "under construction" message. "Find someone you know" will result in an "under construction" message.
    - **"Learn a New Skill" Feature:** "Learn a new skill" will present the student with a list of skills that they can learn - you make up this list of 5 skills. Selecting any of these skills will result in an "under construction" message.
    - **Return Option:** The user will be also presented with an option to not select a skill and to return to the previous level (top level).
    - **Note:** The top level menu will provide the user with a "Logout" option that will terminate the program.

COBOL Implementation Details (For Programmers):

- **Modular Design:** Begin thinking about how to break down your COBOL code into logical, reusable modules (e.g., separate sections or subprograms for user registration, login validation, and data persistence).
- **Input File Handling:** Implement COBOL READ statements to read user input from an external file.
- **Output File Handling:** Implement COBOL WRITE statements to direct all console output to an external output file in addition to the screen. You'll need to carefully manage what is displayed versus what is written to ensure they are identical.
- **Data Structures:** Define appropriate COBOL data structures (e.g., OCCURS clauses, PIC clauses) to store user information (username, hashed password, etc.).
- **Persistence File I/O:** Implement COBOL file operations to read and write user account data to a separate sequential file for persistence.
- **Input Validation:** Ensure robust input validation for password requirements during registration.

Testing Responsibilities (For Testers):

- **Test Case Development:** Create comprehensive test cases in Jira for all Week 1 functionalities, including:
    - **Positive Test Cases:** Scenarios where features are expected to work correctly (e.g., successful new account creation, successful login with valid credentials).

    - **Negative Test Cases:** Scenarios where incorrect input or conditions are provided (e.g., attempt to create account with too short/long password, password missing capital letter/digit/special character, attempt to create 6th account, login with incorrect username/password).
    - **Edge Cases:** Test boundaries and limits (e.g., password exactly 8 or 12 characters).
- **Test Execution:** Execute all developed test cases using the specified input file.
- **Bug Reporting:** For every issue or discrepancy found, no matter how small, create a detailed bug ticket in Jira. Include steps to reproduce, actual results, and expected results.
- **Output Verification:** Meticulously compare the program's console output against the generated output file to ensure they are absolutely identical. This is a crucial step for verifying test results.
- **Collaboration:** Work closely with the programmers to help them understand and reproduce bugs.

Jira Requirements (For Scrum Master, Programmers, & Testers):

Your team's Jira board for Week 1 should include:

- **Epic #1: Log In, Part 1:** This epic should be clearly defined, mirroring the project specification.
- **User Stories:**
    - "As a new user, I want to create an InCollege account so I can access the platform."
    - "As a returning user, I want to log in to my existing InCollege account so I can access the platform."
    - "As a user, I want my account information to be saved so I don't have to re-register each time."
    - "As a user, I want to be informed when I've attempted to create too many accounts."
    - "As a user, I want to be notified if my login attempt is unsuccessful."
    - "As a logged-in user, I want to see initial navigation options (Job Search, Find Someone, Learn Skill)."
    - "As a logged-in user, I want to see 'under construction' messages for undeveloped features."
    - "As a logged-in user, I want to be able to select from a list of skills or return to the main menu."
    - **New User Story (for Testing):** "As a tester, I want the program to read all user inputs from a file so I can automate testing."
    - **New User Story (for Testing):** "As a tester, I want the program to write all screen output to a file so I can easily verify results."
- **Tasks:** Break down each User Story into granular tasks that individual team members can work on. (e.g., *Programmers:* "Implement password hashing logic," "Develop COBOL

module for user file write," "Configure COBOL program to read from input file," "Implement dual output to screen and file." *Testers:* "Develop test cases for user registration," "Execute login tests with valid/invalid data," "Log bugs for Week 1 features," "Verify consistency of screen and file output").

- **Bug Tickets:** As you develop and test, *all* issues found must be logged as detailed bug tickets in Jira by the testers. Programmers will prioritize fixing these.

GitHub Requirements:

- **Repository Setup:** Ensure your team's GitHub repository is set up and accessible.
- **Initial Commit:** Commit your initial COBOL program structure and any foundational files.
- **Branching Strategy:** Begin using a sensible branching strategy (e.g., feature branches for each major piece of functionality, dedicated branches for bug fixes).
- **Regular Commits:** Programmers should be making regular, descriptive commits throughout the week as they work on their assigned tasks. Testers should commit their test files.
- **README.md:** Include a basic README.md file explaining how to compile and run your COBOL program, explicitly detailing how to prepare the input file and where to find the output file.

Deliverables for End of Week (in one Zip file):

1. **Roles.txt**: List of team members and the roles that they played this week.
2. **InCollege.cob: Working COBOL Program:** A console-based COBOL application that demonstrates the specified login, registration, and initial navigation functionality. It *must* read inputs from a file, display all outputs on the screen, and write the *exact same outputs* to a separate file.
3. **InCollege-Input.txt: Sample Input File:** A sample text file demonstrating the format of input your program expects for standard execution.
4. **InCollege-Output.txt:** A sample text file showing the expected output for a typical run of your program.

```
--- SAMPLE_OUTPUT_WEEK1.TXT ---
Welcome to InCollege!
Log In
Create New Account
Enter your choice:
Please enter your username:
Please enter your password:
You have successfully logged in.
Welcome, [Username]!
1. Search for a job
2. Find someone you know
3. Learn a new skill
Enter your choice:
Job search/internship is under construction.
```

```
1. Search for a job
2. Find someone you know
3. Learn a new skill
Enter your choice:
Find someone you know is under construction.
1. Search for a job
2. Find someone you know
3. Learn a new skill
Enter your choice:
Learn a New Skill:
Skill 1
Skill 2
Skill 3
Skill 4
Skill 5
Go Back
Enter your choice:
This skill is under construction.
Learn a New Skill:
Skill 1
Skill 2
Skill 3
Skill 4
Skill 5
Go Back
Enter your choice:
--- END_OF_PROGRAM_EXECUTION ---
```

5. **Epic1-Storyx-Test-Input.zip: Test Input Files:** A set of test input files used by the testers, covering positive, negative, and edge cases for each of this week's stories.
6. **Epic1-Storyx-Test-Output.zip: Actual Test Output Files:** The exact output generated by running your program with the Epic1-Storyx-Input input Files, submitted for review.
7. **Jira.jpg: Updated Jira Board:** All relevant User Stories, tasks, and any identified bugs (with their status) should be updated in Jira.

Remember, the testers on your team will be crucial this week in identifying issues with the registration and login flows, ensuring the password requirements are correctly enforced, and that all messages are displayed accurately and *identically* in both the console and the output file. The scrum master will facilitate the sprint planning and daily stand-ups to keep everyone aligned.