InCollege Project: Week 5 Deliverable - Accepting/Rejecting Connections & Network Display

Objective: This week, your team will finalize the core connection feature by enabling users to manage their pending connection requests (accept or reject them) and to view their complete network of established connections. All program input will continue to be read from a file, all output will be displayed on the screen, and that same output will also be written to a file.

Focus Areas:

1. **Managing Pending Connection Requests:**
   - When a user views their "Pending Connection Requests" (from Week 4), they should be presented with each request and given the option to either "Accept" or "Reject" it.
   - **Accepting a Request:**
     - If a request is accepted, the system must establish a permanent connection between the two users.
     - The accepted request should be removed from the list of pending requests.
     - Both users involved in the connection should now consider each other part of their network.
   - **Rejecting a Request:**
     - If a request is rejected, it should simply be removed from the list of pending requests. No connection is established.
   - In both cases (accept/reject), the user performing the action should receive clear confirmation of the outcome.
2. **Displaying Established Network:**
   - A new option should be available from the main post-login menu (e.g., "View My Network").
   - When a user selects this option, the system should display a list of all users they are currently connected with.
   - For each connection, display at least the connected user's full name. Optionally, you can display their University/Major to provide more context.
3. **I/O Requirements:**
   - **Input:** All user input (e.g., menu selections, accept/reject choices) will be read from a predefined input file.
   - **Output Display:** All program output (e.g., prompts, confirmation messages, displayed lists of pending requests, network lists) 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.

COBOL Implementation Details (For Programmers):

- **Connection Data Structure Update:** You'll need to modify or create a new persistent data structure to store *established* connections. This is distinct from the pending requests. A common approach is a file that stores pairs of connected usernames.
- **Request Processing Modules:** Develop COBOL modules to:
  - Read through pending requests.
  - Prompt the user to accept or reject each request.
  - If accepted:
    - Add an entry to the established connections data file.
    - Remove the request from the pending requests file.
  - If rejected:
    - Remove the request from the pending requests file.
- **Network Display Module:** Implement COBOL routines to read from your established connections data file and display the list of connected users for the logged-in user. You'll likely need to cross-reference with your main user profile data to get names and other details.
- **File Handling:** Ensure robust file handling for adding, deleting, and updating records across multiple files (user profiles, pending requests, established connections).
- **Input File Handling:** Continue to implement COBOL READ statements to read user input from the designated input file for all menu selections and accept/reject choices.
- **Output File Handling:** Ensure all program output, including prompts and displayed lists, is written to your dedicated output file *identically* to what is displayed on the screen.

Testing Responsibilities (For Testers):

- **Test Case Development:** Create comprehensive test cases in Jira for all Week 5 functionalities, including:
  - **Accepting Requests:** Scenarios for successfully accepting a single request, and accepting multiple requests.
  - **Rejecting Requests:** Scenarios for successfully rejecting a single request, and rejecting multiple requests.
  - **Mixed Scenarios:** Test a sequence of accepting some requests and rejecting others.
  - **Network Display:** Verify the network list accurately reflects accepted connections and doesn't show rejected or pending requests. Test users with no connections, one connection, and multiple connections.
  - **Persistence:** Ensure that accepted connections and the removal of pending requests persist across program restarts.
- **Test Execution:** Execute all developed test cases using the specified input file.
- **Bug Reporting:** For every issue or discrepancy found, 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 for all connection management and network display scenarios.
- **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 5 should expand upon **Epic #3: User Search & Connections (Part 1)**, or if you created a dedicated "Connection Management" Epic, continue to build on that.

- **User Stories for Managing Requests:**
  - "As a user with pending connection requests, I want to accept a request so I can expand my network."
  - "As a user with pending connection requests, I want to reject a request so I can manage who I connect with."
  - "As a user, I want to be notified when a connection request is successfully accepted or rejected."
  - "As a user, I want accepted/rejected requests to be removed from my pending list."
- **User Stories for Network Display:**
  - "As a logged-in user, I want to view a list of all users I am connected with."
  - "As a user, I want to see a clear display of my connections, including their names."
  - "As a user, I want an option in the main menu to view my established network."
- **New User Story (for Testing):** "As a tester, I want the program to read all user inputs for managing and viewing connections 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 related to connection management and network display 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:* "Develop COBOL module for accepting a connection," "Implement COBOL logic to remove rejected requests," "Define COBOL file structure for established connections," "Create COBOL routine to display network list," "Update I/O for connection management features to use file input/output." *Testers:* "Develop test cases for accepting/rejecting connections," "Execute network viewing tests," "Log bugs related to connection management," "Verify I/O consistency for connection management features").
- **Bug Tickets:** Log any issues found during development and testing.

GitHub Requirements:

- **New Modules/Files:** Commit any new COBOL modules or updated existing files for handling connection acceptance/rejection and network display.
- **Branching:** Continue to follow your team's established branching strategy for new features.
- **Regular Commits:** Ensure consistent, descriptive commits throughout the week. Testers should commit their test files.
- **README.md Update:** Update your README.md to reflect the new functionality for managing and viewing connections, explicitly detailing how to prepare input for these features and and where to find the corresponding output.

Deliverables for End of Week 5:

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 allows users to:
   a. View pending connection requests.
   b. Accept or reject individual pending requests.
   c. View a list of their established connections.
   d. This must seamlessly integrate with all previous weeks' functionality. All inputs must be read from a file, all outputs displayed on the screen, and the exact same outputs written to a separate file.
3. **InCollege-Input.txt: Sample Input File:** A sample text file demonstrating the format of input your program expects for Week 5's functionality (e.g., menu choices, accept/reject decisions).
4. **InCollege-Output.txt:** A sample text file showing the expected output for a typical run of your program, demonstrating management of pending requests and viewing the network.

```
--- SAMPLE_OUTPUT_WEEK5.TXT ---
Welcome to InCollege!
1. Log In
2. Create New Account
Enter your choice:
Please enter your username:
Please enter your password:
You have successfully logged in.
Welcome, TestUser!
1. View My Profile
2. Search for User
3. Learn a New Skill
4. View My Pending Connection Requests
5. View My Network
Enter your choice:
--- Pending Connection Requests ---
Request from: OtherUser
```

```
1. Accept
2. Reject
Enter your choice for OtherUser:
Connection request from OtherUser accepted!
-----------------------------------
1. View My Profile
2. Search for User
3. Learn a New Skill
4. View My Pending Connection Requests
5. View My Network
Enter your choice:
--- Your Network ---
Connected with: OtherUser (University: Another U, Major: Marketing)
Connected with: FriendB (University: Big State, Major: Engineering)
--------------------
1. View My Profile
2. Search for User
3. Learn a New Skill
4. View My Pending Connection Requests
5. View My Network
Enter your choice:
--- END_OF_PROGRAM_EXECUTION ---
```

5. **Epic5-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. **Epic5-Storyx-Test-Output.zip: Actual Test Output Files:** The exact output generated by running your program with the Epic5-Storyx-Input input Files, submitted for review.
7. **Jira.jpg: Updated Jira Board:** All relevant User Stories, tasks, and bugs (with their status) for Week 3 should be updated in Jira.
8. **GitHub.jpg:** Go to the repository's main page. Click the "Commits" link (next to the green "Code" button). Show a chronological list of all commits with messages, authors, and timestamps.

Your testers will be vital this week, ensuring the acceptance and rejection processes work flawlessly, that connections are correctly established (and not established when rejected), and that the network display accurately reflects the connections. They will also meticulously compare console output with the generated output file for consistency. The scrum master will, as always, be key to keeping the team on track and facilitating communication.