REFUGEE STATUS APPLICATION SYSTEM

TEAM X SPARK

1. Damaresh
2. Pampapathi
3. Manju Kadam
4. Pavanesh Prabhu
5. Bharath Kumar M

- This Application Management System allows users to manage applications, particularly for processing refugee status. The system implements a simple menu-driven interface that supports Create, Read, Update, Delete (CRUD) operations, as well as integrity audits on applications

Data Structures:

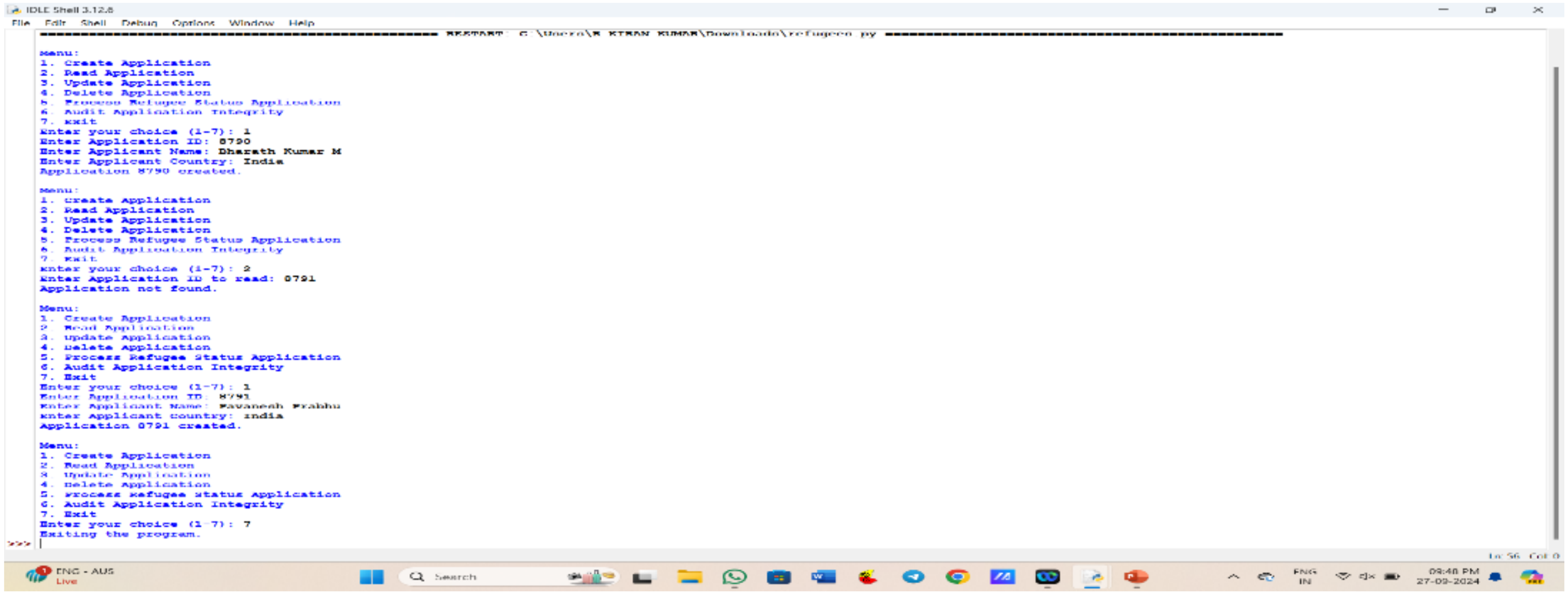
Create two dictionaries: applications to store application data and integrity_audit_log to track integrity audits.
CRUD Operations:

Create Application:
Add a new application to applications with an ID, applicant data, a status of “ pending” , and an integrity check status of False.
Read Application:
Retrieve and return the details of an application by its ID, or indicate if it doesn’ t exist.
Update Application:
Modify the applicant data of an existing application based on its ID.
Delete Application:
Remove an application from the applications dictionary by its ID.
Process Applications:

- Process Refugee Status:
  Change the status of an application to “ processed” if it exists.
  Audit Integrity:

  Audit Application Integrity:
  Check if an application has been audited:
  If not, perform a mock integrity check, update the status to “ Integrity check passed” , and log the audit.
  If already audited, notify the user.
  Menu-Driven Interface:

  Display a menu to the user with options to create, read, update, delete applications, process refugee applications, audit integrity, or exit.
  Continuously prompt the user for their choice and execute the corresponding function until they choose to exit.


IDLE Shell 3.12.6
File Edit Shell Debug Options Window Help
========== RESTART: C:\Users\R KIRAN KUMAR\Downloads\refugee.py ==========
Menu:
1. Create Application
2. Read Application
3. Update Application
4. Delete Application
5. Process Refugee Status Application
6. Audit Application Integrity
7. Exit
Enter your choice (1-7): 1
Enter Application ID: 8790
Enter Applicant Name: Dharath Kumar M
Enter Applicant Country: India
Application 8790 created.
Menu:
1. Create Application
2. Read Application
3. Update Application
4. Delete Application
5. Process Refugee Status Application
6. Audit Application Integrity
7. Exit
Enter your choice (1-7): 2
Enter Application ID to read: 0791
Application not found.
Menu:
1. Create Application
2. Read Application
3. Update Application
4. Delete Application
5. Process Refugee Status Application
6. Audit Application Integrity
7. Exit
Enter your choice (1-7): 1
Enter Application ID: 8791
Enter Applicant Name: Pavaneah Prabhu
Enter Applicant Country: india
Application 8791 created.
Menu:
1. Create Application
2. Read Application
3. Update Application
4. Delete Application
5. Process Refugee Status Application
6. Audit Application Integrity
7. Exit
Enter your choice (1-7): 7
Exiting the program.
>>>
Ln: 56 Col: 0
ENG - AUS
Live
Search
ENG
IN
09:48 PM
27-09-2024

1.Database Integration:

- Currently, the data is stored in dictionaries (applications, integrity_audit_log) within memory. For large-scale applications, integrating with a database (e.g., MySQL, MongoDB) would enable persistent storage, improved performance, and scaling.
- Implementing an ORM (e.g., SQLAlchemy for Python) to handle database interactions could further improve code maintainability.

2.Authentication and Authorization:

- Implement user roles and permissions, especially for sensitive tasks like processing and auditing applications.
- Different roles like "Applicant", "Admin", or "Integrity Auditor" can have different access levels to data and operations.

3.Input Validation and Error Handling:

- Add input validation to ensure the correctness of data (e.g., validate country names, ensure application IDs are unique).
- Implement proper error handling to manage exceptions (e.g., when an application ID doesn’ t exist).

4.Logging and Monitoring:

- Add logging functionality to monitor actions like creation, updates, and deletion of applications. This would be useful for audit trails and debugging.
- A monitoring system could track the health of the application and identify any performance bottlenecks.

5.Concurrency and Multi-User Support:

- As the application grows, multiple users might try to access or modify the same application simultaneously. To handle concurrency, implement locking mechanisms or database transactions to ensure data consistency.

- **Web Interface or API:**
- Convert this command-line interface into a web-based application using frameworks like Flask or Django.
- Alternatively, develop a RESTful API to enable interaction with other services, making it easier to integrate with external systems.
- **Advanced Search and Filtering:**
- Implement search functionality with advanced filters to retrieve applications based on status, country, applicant name, or audit status. This would enhance usability for large datasets.
- **Notification System:**
- Add a notification or email system to notify applicants about the status of their applications (e.g., pending, processed, audited).
- **Data Encryption and Security:**
- Secure sensitive information (such as applicant data) with encryption techniques to ensure data privacy and security.
- Implement HTTPS for any web-based or API interactions to safeguard communication.
- **Reporting and Analytics:**
- Add reporting functionality to generate summaries, charts, or metrics, such as the number of applications processed, applications from a specific country, etc.
- Integrate analytics to assess trends, improve decision-making, and track performance over time.