My assignment in COSC 498 – Capstone Project, is to develop an enhancement when an airline has irregular operations and must cancel or delay flights, the passenger’s ticket must be reissued to reflect the new itinerary if it’s changed.

When the ticket is exchanged/reissued, the reservation system prints the receipt of the fare and the new itinerary that will be flown, but it also prints unnecessary data, such as baggage data & rules, weight & balance information, and agent information.

This project is to eliminate printing the unneeded ticket stock data.

The airline agent throws away all unnecessary ticket stock printouts from the reissue except for the receipt and itinerary, which waste several pages of ticket stock paper.

* If the ticket does not get reissued, the passenger will not be able to check-in for that flight to obtain their boarding pass.
* Only an airline agent can reissue an airline ticket; passenger cannot reissue their ticket.
* By implementing this new change, the following will improve:
  + Less time for passengers to wait for an agent to reissue their ticket.
  + Airlines would save money by using less ticket stock when a re-issuance is completed.
  + Agents would not have to tear off & throw away the unnecessary ticket stock for the pieces they do not need.

In conclusion, saving passengers' time, the agent's time and the company money would justify this implementation.

I must first design a database that has the passengers flight details, such as flight number, date of flight, class of service, city pairs, flight time of departure and arrival, and the passengers record locator in the reservation system.

For my project, I'm using a relational database MySQL to design my reservation data. This database is robust, widely used, and well-suited for handling structured data like flight details. Here’s a breakdown of the steps to design my database:

Database Design

1. **Tables**:
   * **Passengers**:
     + passenger\_id (Primary Key)
     + record\_locator
     + name
   * **Flights**:
     + flight\_id (Primary Key)
     + flight\_number
     + date\_of\_flight
     + departure\_time
     + arrival\_time
     + class\_of\_service
     + origin\_city
     + destination\_city
   * **Reservations**:
     + reservation\_id (Primary Key)
     + passenger\_id (Foreign Key)
     + flight\_id (Foreign Key)
     + ticket\_status (e.g., confirmed, reissued, canceled)
2. **Relationships**:
   * Each reservation links passengers to their flight details.
   * passenger\_id in the Reservations table is a foreign key to passenger\_id in the Passengers table.
   * flight\_id in the Reservations table is a foreign key to flight\_id in the Flights table.

**Explanation**

* **Passengers Table**:
  + passenger\_id is the primary key and will automatically increment with each new passenger.
  + record\_locator is an all-alpha code (not alphanumeric) used for reservation identification, and is in ALL CAPS.
  + name stores the passenger's name.
* **Flights Table**:
  + flight\_id is the primary key and will automatically increment with each new flight entry.
  + flight\_number, date\_of\_flight, departure\_time, arrival\_time, class\_of\_service, origin\_city, and destination\_city store flight details.
* **Reservations Table**:
  + reservation\_id is the primary key and will automatically increment with each new reservation.
  + passenger\_id serves as a foreign key referencing passenger\_id in the Passengers table.
  + flight\_id serves as a foreign key referencing flight\_id in the Flights table.
  + ticket\_status uses an ENUM data type to restrict status values to 'confirmed', 'reissued', or 'canceled'.

Other Tools

* **Database Management System (DBMS)**: I will use MySQL Workbench as a tool to manage and visualize my database.
* **Programming Language**: Python will be used to interact with the database. Libraries such as SQLAlchemy (Python) can provide ORM (Object-Relational Mapping) support, making database operations more intuitive.
* **Version Control**: I will use Git to keep track of changes in my database schema and code.
* **Testing**: I will use a testing framework like pytest (for Python) to ensure your database operations are functioning correctly.
* **Project Management Tools**: Trello or Jira can help in tracking progress and managing tasks. I haven’t decided which one to use yet.

This setup will streamline my database design process and ensure efficient management of flight details and re-issuance processes.

Discussion 2:

1. What is your product vision from the last module?
2. What are the key features to identify for this product? (3 to 5 is a good starting place)
3. What are the user requirements?
4. What is the current state of your development resources?
5. What deficiencies do you see in the current state that may need to be addressed?

The vision I have for my product is to enhance ticket reissues to save company money, customer processing time, and agent productivity. The key features of this enhancement are Passenger data items, such as name and record locator in the reservation system, and ticketing information such as ticket status, whether it’s confirmed, reissued, or canceled; flight details such as flight number, date of travel, departure and arrival times, origin and destination cities, and class of service.

User requirements include primary keys for the passenger id, flight id and the reservation id, and two foreign keys in the Reservation section for passenger id and flight id. Currently, there are enough development resources to enhance this coding.

During irregular operations (IROPS), where flights are delayed or canceled, a ticket sometimes must be reissued if the passenger must change their flight. When the ticket is reissued, the system prints data that is not necessary for the passenger to have. All the passenger wants is the confirmation of the new itinerary to show any rebooked flights, the new ticket receipt, and their boarding passes for the new itinerary. (The passenger cannot get checked in for the new flight unless the ticket is reissued.) Current programming must be changed to eliminate the coding that prints the unneeded ticket stock items. This change would save ticket stock costs, because less pages are printed, it will save agent time and also the passengers time because they do not have to wait for the unneeded pages to be torn off and thrown away by the agent.