CS:4400 Testing Plan

Team #5

To make sure that our project is high quality software, we need to make sure that we test every possible case and component of the database. First, we will focus on testing the structure of the database. We will create all tables designed in the ER diagram with constraints and parameters. After we will try to test each field if its correctly filtering data and giving correct errors. We have to check:

- Validation of the length and naming convention of the database fields and columns
- Validation of the presence of any unused/unmapped database tables/columns
- Validation of the compatibility of the data type field lengths
- Whether the database fields allow the user to provide desired user inputs as required by the project.

After when we will have a fully connected and tested database design we will try to test some queries we will need in the project. We will put some simple data into our model and we will try to write queries that produce expected results.

- Check connectivity between primary and foreign keys.
- Check whether the references for foreign keys are valid.
- Check whether the data type of the primary key and the corresponding foreign keys are same in the two tables.
- Check whether the required naming conventions have been followed for all the keys and indexes.
- Write some queries and check if the output is what we want

We need to make sure that a transaction either fails or passes, that a database is consistent, that if there are multiple transactions and they are executed all at once, they go in an order. We plan to use White Box testing since Black Box can result in some errors to be undetected. In other words, coding errors can be detected in white-box testing, so internal bugs in the database can be eliminated. We know that White Box testing does not cover SQL statements. We will account for that.

For our front end we will code the website and connect it through Python. We will build the website using many various fields and we will test it as a final user. We will try to be mean and see if the website along with the database is handling wrong inputs correctly. We will display appropriate error to the user with detailed feedback what was wrong with his action or procedure.

Overall, our testing will involve checking stored procedures, views, schemas in database, tables, indexes, keys, triggers, data validations and data consistence checks. We will check all the functionality which is happening on every action performed in the application, such as create, read, update, or save options (CRUD). Then, we will connect and build the website accounting for error handling and displaying feedback to the user.

Type being tested	Name	Test Condition	Expected	Observed
Screen	Round trip/ One way/ Multi-city	Given “from” and “to”	Make sure they aren’t the same, if they are the same give an error message. Select flights and display them for that particular route.	Implemented, tested, and passed
Screen	Round trip	Given departure date and return date	Make sure that return date is after departure date, if it isn’t, give an error message.	Implemented, tested, and passed
Screen	Your Trip (No Login Required)	Given user details and date	Make sure that user exists with that name and that they have a booking for that date. If not, the give an error message. Show the trip information.	Implemented, tested, and passed
Screen	Flight Status (Date)	Given “from”, “to, and date	Make sure they “From” and “To” aren’t the same, if they are the same give an error message. Make sure that there is a flight in that route. If not, the give an error message. Show flight information and status.	Implemented, tested, and passed
Screen	Flight Status (FlightID)	Given FlightID and date	Make sure that a flight exists with that FlightID on that particular date. If not, the give an error message. Show flight information and status.	Implemented, tested, and passed
Screen	Login	Given username/UserID and a password	Make sure that the user exists with that username and it doesn’t contain any suspicious characters that could indicate an SQL injection. If not, the give an error message. Load user’s dashboard.	Implemented, tested, and passed
Screen	Registration	Given username	Make sure this username isn’t already taken, if it is taken give an error message. Register the user.	Implemented, tested, and passed

Screen	Registration	Given email	Make sure this email isn't already taken, if it is taken give an error message. Register the user.	Implemented, tested, and passed
Screen	Forgot Username	Given an email	If it is in the database send the username to that email, otherwise give an error message.	Implemented, tested, and passed
Screen	Forgot Password	Given an email	If it is in the database send the reset link to that email, otherwise give an error message.	Implemented, tested, and passed
Screen	Contact form	Given email	Make sure there is an account with that email, if there isn't, give an error message. Send contact email to our email support.	Implemented, tested, and passed
Screen	Flights	Given departure date	Make sure that the date is between today and 16 weeks from today, if it isn't, give an error message.	Implemented, tested, and passed
Screen	Multicity	Given many flights	Make sure that the number of flights is between 2 and 6. If it isn't, give an error message. Query all the flights given and display multiple results.	Implemented, tested, and passed
Screen	Passengers	Given passport number	Make sure there aren't multiple passengers on the same flight with the same passport number, give an error message if there are multiple passengers with the same passport number.	Implemented, tested, and passed
Screen	Payment	Given credit card details	Make sure that the credit card details are real. The credit card number has to be correct and be unique. If it isn't, give an error message. Complete the payment and charge the user.	Implemented, tested, and passed

Screen	Flight Change	Given a user wants to change his or hers flight	Make sure that if the user wants to change a flight, it has to be 2 days before the departure. If it isn't, give an error message. Let the user pick a new flight and update the flight for the user.	Implemented, tested, and passed
Screen	Flight Cancellation	Give a user wants to cancel his or hers flight	Make sure that if the user wants to cancel the flight, it has to be 2 days before the departure. If it isn't, give an error message. Change the flight for the user.	Implemented, tested, and passed
Screen	Booking	Given a successful flight purchase	Make sure that a generated booking number is 6 letters and is unique. If it isn't, give an error message. Process the payment and create a booking for that user.	Implemented, tested, and passed