DS Ferries System Documentation

Author: Tsagkaris Nikolaos

A. M. : E23172

Date: 28 / 01 / 2024

Table of Contents

Introduction

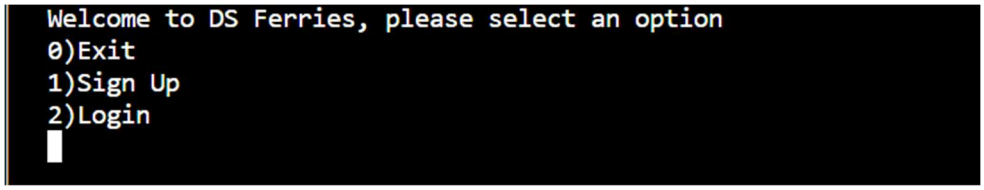
This documentation provides an overview of the DS Ferries System, a program designed for managing ferry bookings and trips. The system allows to 30 users maximum to sign up, log in, view the 13 available routes of Greek islands, book up to 50 trips per user, make payments, and view their confirmed trips. The DS Ferries System is implemented in C and utilizes various data structures for user information, routes, booked trips, and dates. The system is menu-driven with New Window feeling, providing users with a simple interface to interact with the functionality.

In this Documentation, you will have all the system explained, with screen captures and tips in order to use it properly. Also a function – variable – table that will show the type of the variable and if it is a parameter and in which function.

System Manual

Home page

Within the ambit of the Home Page, the code orchestrates the initial user interaction by presenting a navigational menu housing pivotal options such as user registration and login. The function 'HomePage' takes center stage, steering users towards their desired actions based on input parameters.



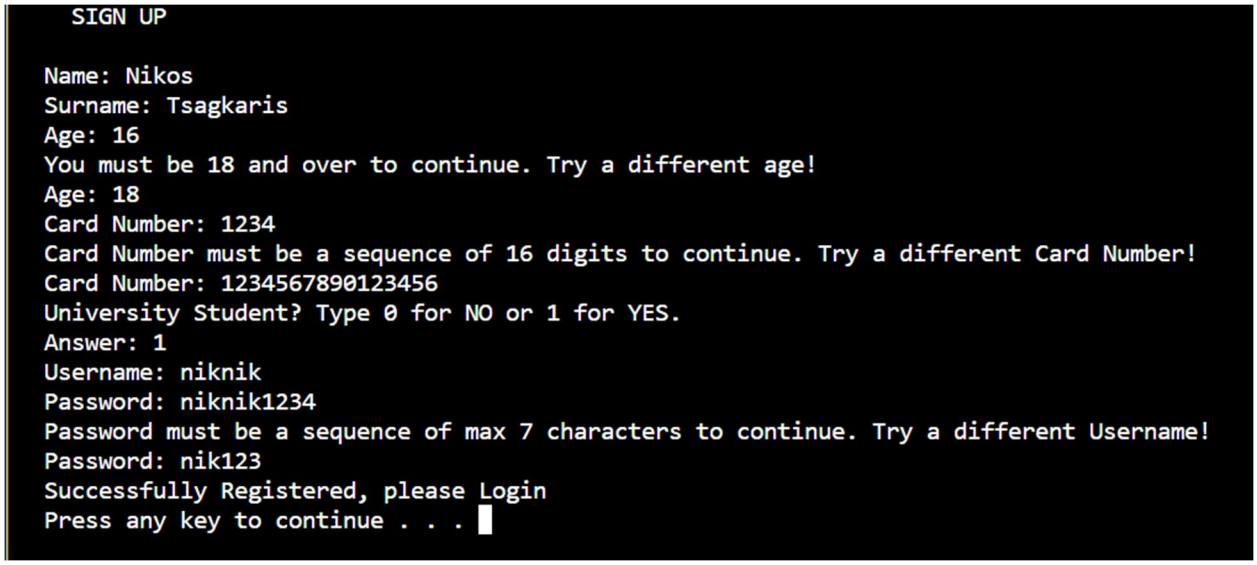
If 0 is given, the system responds:

And terminates the program.

Sign up

The Sign-Up module intricately navigates users through the labyrinth of creating a new account. It meticulously validates user input, scrutinizing details ranging from personal particulars like name and age to critical information such as card number and password, ensuring the integrity and accuracy of the entered data.

Now if 1 is given, the system ask the user critical information line by line:

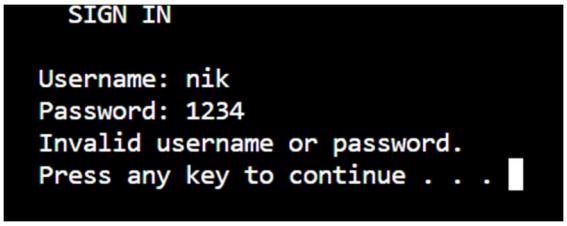


In this example the user after many mistakes and system error messages has finally successfully registered. And then the system suggest us to sign in.

Sign in

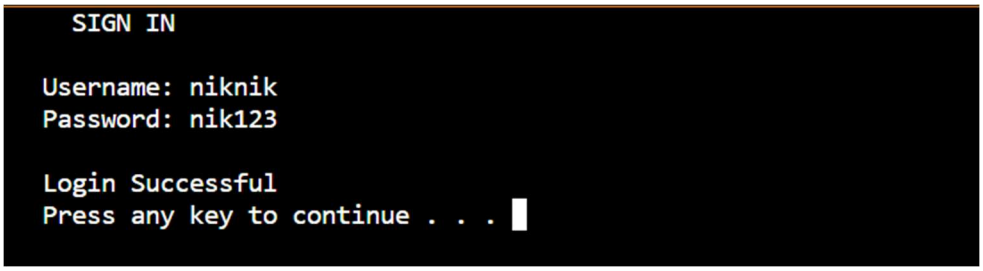
The Sign-In functionality facilitates the seamless access of existing users, scrutinizing their credentials against the stored user data for verification. Upon successful authentication, users are seamlessly transitioned into the User Dashboard, the nerve center of their interaction with the ferry reservation system.

After we go back to Home Page again, we press 2 and the system responds:



It is important to mention that even if there wasn't any user registered the respond would be the same as the example, that we have inserted wrong credentials.

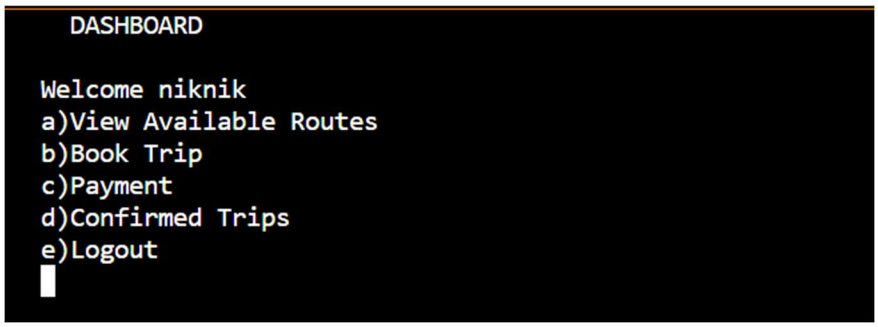
After this repetition of us trying to log in to the system, we actually made it:



And the user Dashboard is the screen that comes up next!

User Dashboard

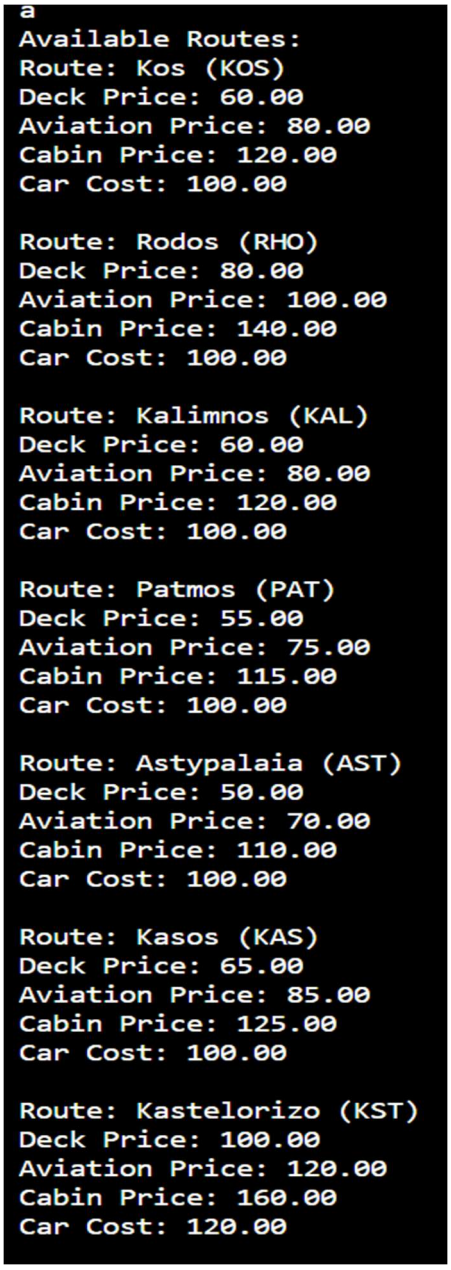
The User Dashboard serves as the epicenter of user engagement, offering a multifaceted interface for logged-in users. The 'Dashboard' function takes charge of orchestrating user interactions within this domain, encompassing key features such as viewing available routes, booking trips, managing payments, and perusing confirmed trip details.

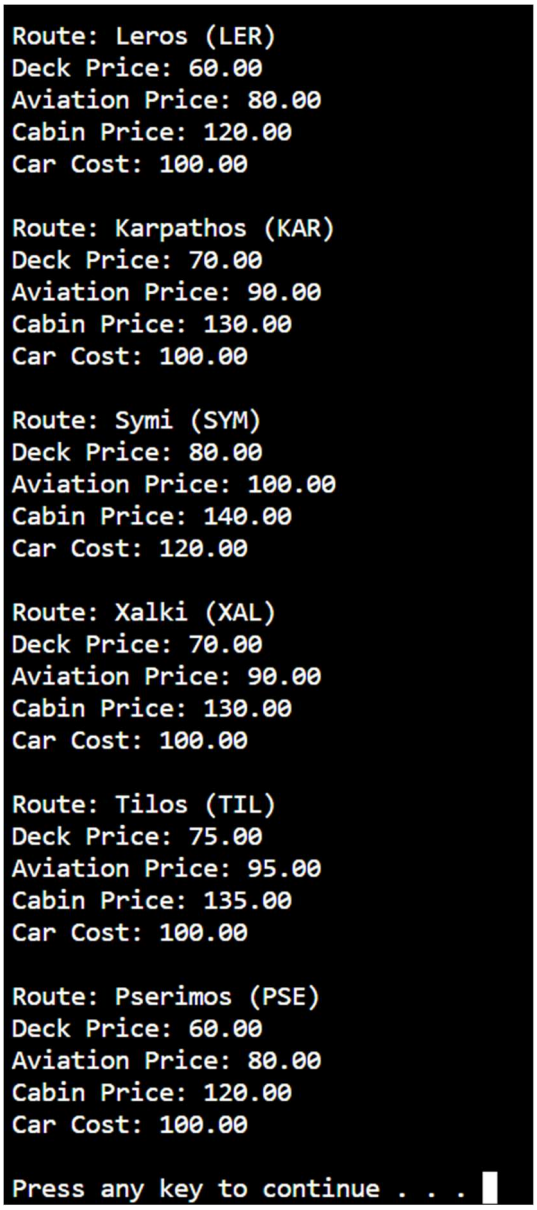


Depending on what the answer will every time be (‘a’ or ‘b’ or ‘c’ or ‘e’) the action is the following:

View Available Routes

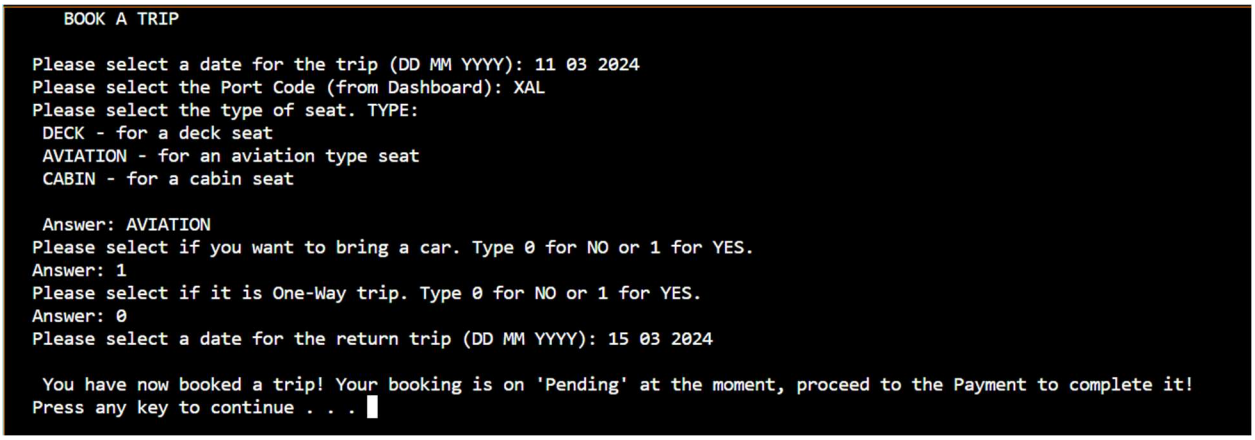
The View Available Routes feature unfolds a comprehensive panorama of information concerning different ferry routes. This includes intricate details such as port names, port codes, and associated pricing structures for diverse accommodation options, including deck, air, cabin, and car selections.





Book Trip

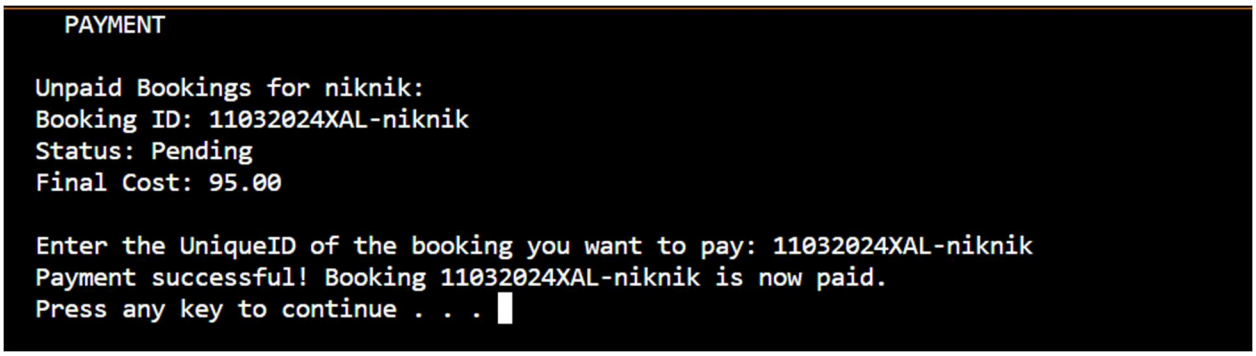
The Book Trip module artfully guides users through the intricacies of reserving a ferry journey. It encapsulates the user input process for vital trip details, including date, port code, seat preferences, car options, and return trip specifications. The culmination involves dynamic cost computation based on user choices, culminating in the storage of booking information within the 'booked_trips' array.



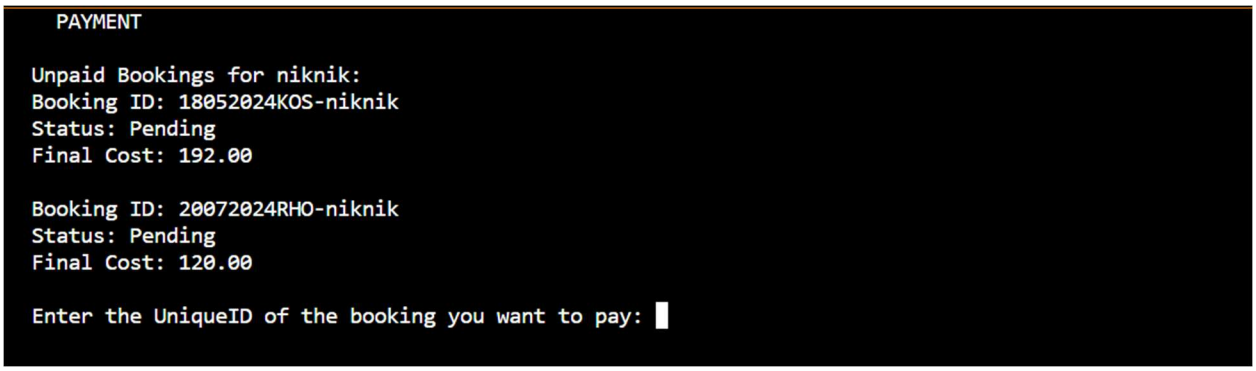
But in order to change the booking status from ‘Pending’ to ‘Paid’, you must proceed to the payment.

Payment

The Payment section assumes responsibility for the financial transactions associated with booked trips. It systematically presents users with pending payment obligations, prompting them to input the Unique ID of the booking they wish to settle. Successful payment transactions result in the instantaneous updating of the booking status to "Paid."



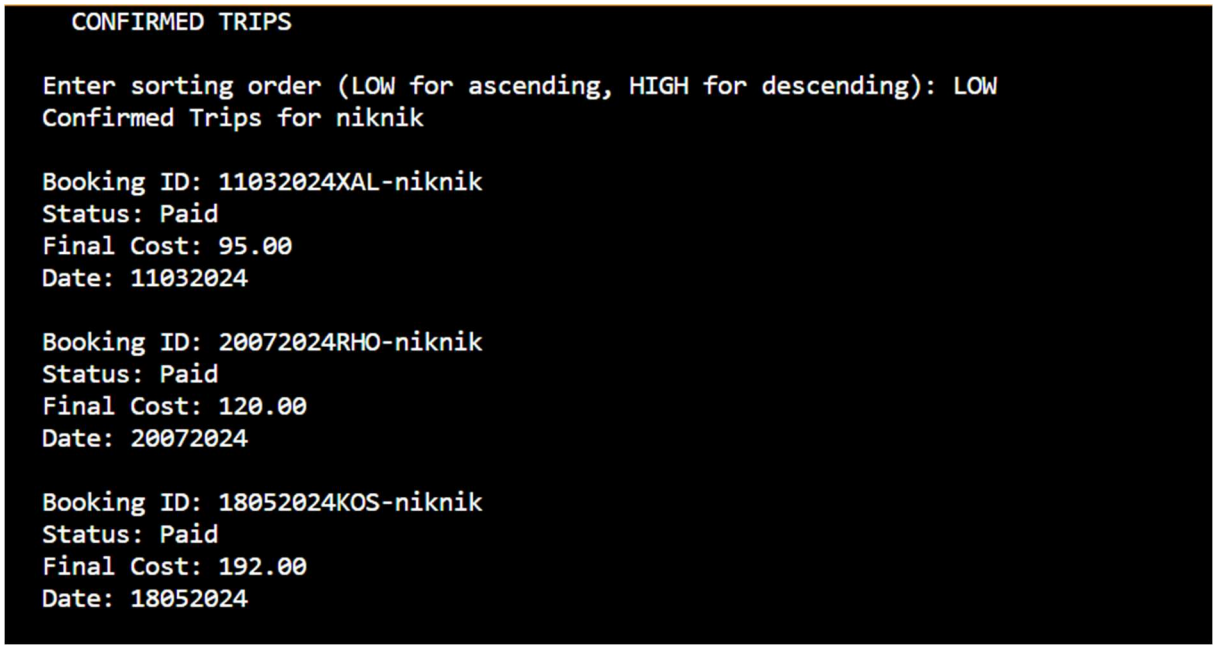
For the matter of demonstration I have added two more bookings, and paid for them:



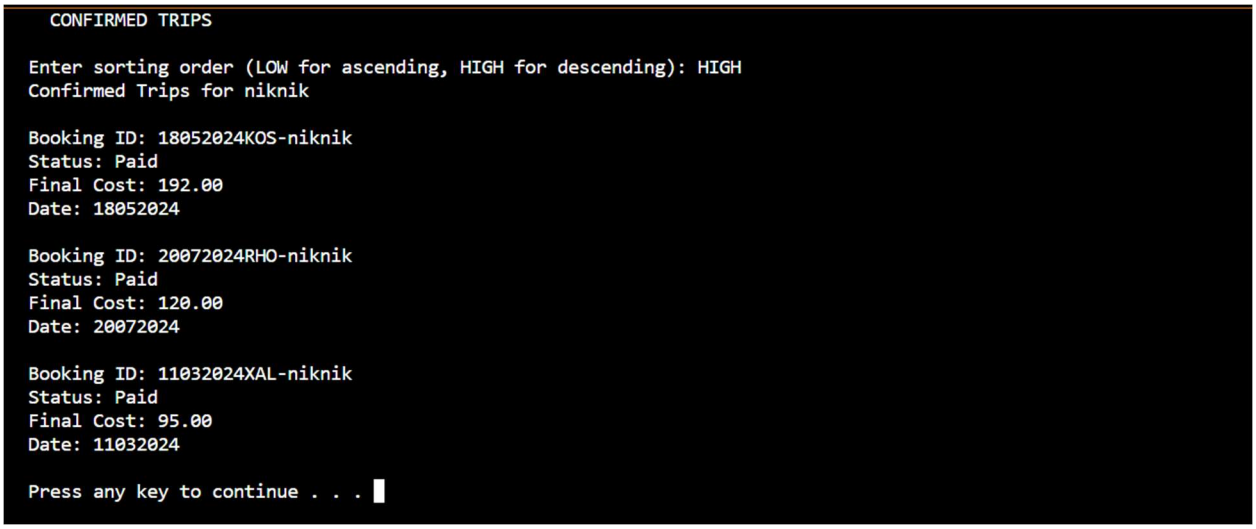
View Confirmed Trips

The View Confirmed Trips feature provides users with an immersive insight into their confirmed (paid) journeys. It retrieves and elegantly displays pertinent information such as booking ID, status, and final cost, affording users the flexibility to toggle between ascending and descending sorting orders for enhanced readability. Now that's why in the previous step I added two more bookings and paid them…because the ascending and descending sorting order would not have a meaning at all!

In this capture I have given LOW and the Confirmed trips are in an ascending sorting order:



In this capture I have given HIGH and the Confirmed trips are in a descending sorting order:



The last option is **Logout** that moves the user to the home screen again and he can choose his next action.

Table of Functions

Function Name	Parameters	Function Type
main	None	void
Homepage	int option, User users[], int *userCount)	void
SignUp	User users[], int *userCount	void
Login	User users[], int userCount	void
ViewAvailableRoutes	None	void
DashBoard	char currentUser[20]	void
Book_trip	char currentUser[20]	void
compareDates	Date date1, Date date2	int
MakePayment	char currentUser[20]	void
ConfirmedTrips	char currentUser[20]	void

What would I do Differently in the Future?

This project was not difficult in terms of its implementation. The difficult point was that the discussions were chaotic but I believe that the DS Ferries System could not be more functional. I also think that one of the most important issues of such a system is that all possible responses from the user are covered, so that there are no endless repetitions and crashes. This is done and I am very pleased. Maybe it should have been a little more user friendly and have more explanatory instructions within the system itself, a manual that you could reach with the push of a button. However, I am very satisfied with my implementation of the project.