Case Study: MyCab – Cab Booking System

Group 1

Team member Emp ID	Team member Name	Team Member Email Id
46283029	Deeksha Maurya	deeksha.maurya@capgemini.com
46292701	Soubhik Banerjee	soubhik.banerjee@capgemini.com
46289235	Madhurita Shekhawat	madhurita.shekhawat@capgemini.com
46290768	sandeep kumar	SANDEEP.AN.KUMAR@CAPGEMINI.COM
46292697	Ankit Kumar	ANKIT.AR.KUMAR@CAPGEMINI.COM
46283032	Prama Sahu	prama.sahu@capgemini.com

PROBLEM STATEMENT

1.2 OBJECTIVE

To create an online cab booking system like like Uber where users can book a cab for their destination. The application is to be develope as Executable file compiled on Linux. There are 2 entities User and Cab driver.

1.3 ABSTRACT OF THE PROJECT

1. User and cab driver should be able to login into the app.
2. User should be able to enter source and destination.
3. User should be able to choose Car size and model.
4. Predicated fare should be calculated and displayed to the user.
5. Once booked, a message should be send to the cab driver.
6. User should be able to see cab driver information and cab details.
7. Cab driver should be able to accept and reject the request.
8. User should be able to cancel the ride.
9. Once the journey is complete (simulate it randomly), show the final fare.
10. Record all the history of rides of customer.
11. Handle data and errors properly. Show appropriate messages to user.
12. Display good input, output messages and reports in proper format.
13. Security features should be implemented whereever possible. For example user passwords can be stored in encrytped format.

1.4 FUNCTIONAL COMPONENTS OF THE PROJECT

Following is a list of functionalities of the system. Wherever, the description of functionality is not adequate; you can make appropriate assumptions and proceed.

When MyCab starts it displays Following Screen -

----------Login Screen----------
1. Register new user
2. Register new cab driver
3. Login as user
4. Login as cab driver
0. Quit

System should maintain comma separated users.txt where each line stores username, password and all other information about registered user.

System should maintain comma separated cabDrivers.txt where each line stores username, password and all other information about registered cab drivers.

Whenever “Login as user” is entered, system authenticates it with entry in “users.txt” file.

- If match is found then “User Screen” is displayed.
- If match is not found then message “Invalid User or password” is displayed and system exits.

Whenever “Login as cab driver” is entered, system authenticates it with entry in “cabDrivers .txt” file.

- If match is found and “Cab Driver Screen” is displayed.
- If match is not found then message “Invalid Cab Driver or password” is displayed and system exits.

2. When user login MyCab displays “User Screen”

----------User Screen----------

1. Schedule Trip
2. Book Trip
3. Check Cab Driver Details
4. Check Cab Details
5. Check Bill
6. Make Payment
0. Quit

Enter your option : <option>

option = 1 (Schedule Trip)
MyTrip asks source and destination of the trip, number of seats required
All availble car sizes and car models are displayed
User selects car size and car model

All these details are stored in “ScheduledTrips.txt” along with username and date. This is comma seperated file.

option = 2 (Book Trip)
All scheduled trips of user whose fare is provided by cab driver are displayed.
User selects trip to book from the list.

All details are stored in “BookedTrips.txt” along with username and date. This is comma seperated file.
Corresponding entry of trip is deleted from “ScheduledTrips.txt”.

option = 3 (Check cab driver details)
With this option user can check cab driver details of the trip he/she booked.

option = 4 (Check cab details)
With this option user can check cab details of the trip he/she booked.

option = 5 (Check Bill)
With this option user can check bill of the trip he/she booked. Bill Amount = Fare entered by cab driver * number of seats required.

Option = 6 (Make Payment)
Bill is displayed to user for completed trip. Payment details such as mode of payment, credit card number, ..., bill amount, status of payment will be stored in "payments.txt" file.

2. When cab driver logins in, MyCab displays "CabDriver Screen"

---------- CabDriver Screen----------

1. Update Profile
2. Manage Car Details
3. Check Scheduled Trips
4. Check Booked Trips
5. Mark Completed Trips
0. Quit

Enter your option : <option>

option = 1 (Update Profile)

Details entered by cab drivers during registration can be modified using this option. "cabDrivers.txt" will be updated with new details.

option = 2 (Manage Car Details)

Cab driver will modify car details with this option. Details are stored in CabDetails.txt along with cab driver ID.

option = 3 (Check Scheduled Trips)

Display all scheduled trips whose date = today and time >= current time. Cab driver will select the trip to book and will provide fare. Cab drivers ID, Car ID and fare details will be updated in "BookedTrips.txt".

option = 4 (Check Booked Trips)

This option will display status of booked trip. If it is taken by user then driver is proceed to user location.

option = 5 (Mark Completed Trips)

Booked trip will be marked as completed. The trip will be removed from BookedTrip.txt and will be added to CompletedTrip.txt along with user ID, Cab driver ID, Payment ID, amount etc..

Assumptions: <Write assumptions made>

Technical Requirements -

1) C programming language
2) Use file input/output operations to read and write data.
3) Use multiple Linked Lists to read data from files at the beginning and write updated data to files before applicaton ends.
4) Use dynamic memory allocation.

Non Functional Requirements

1) Multi-file multi-directory solution is expected. Modular and maintainable code (comments) and all coding standards should be followed.
2) makefile to build application. Two-step compilation process - .o and then executable should be generated.
3) Use valgrind tool on application executable to detect memory leak. Final valgrind report to be submitted in "reports" directory.
4) Level 0 DFD (context diagram), Level 1 DFD, Flow diagram and pseudocode for 2 complex functions logic.
5) SRS in pdf format, RTM, Plan, Presentation. MOMs
6) HLD_LLD Document (Optional)
7) Unit test cases and Integration test cases in UT_IT document. Both types of test cases i.e. sunny and rainy should be present in this document

Set Up Checklist for Project

Software Requirement:

Vi Editor, ctags, splint, valgrind, gcc, make, git account

Minimum System / Hardware Requirements:

Laptop with access to internet and Linux OS