EEE102 API Document

Taxi Company Management System

Table of Contents

Table of Contents 2

1 Preface 4

2 User Interface 6

2.1 Display_interfaces.h 6

2.1.1 void show_Vehicle_second(void); 6
2.1.2 void show_Wait(void); 6
2.1.3 void wrong_Search(void); 6
2.1.4 void system_Show(void); 6
2.1.5 void system_Show_client(char* tmp); 7
2.1.6 void system_Show_driver(void); 7
2.1.7 void system_Show_vehicle(void); 7
2.1.8 void vehicle_Board(void); 7
2.1.9 void driver_Board(void); 7
2.1.10 void show_Search(void); 8
2.1.11 void show_Super(void); 8
2.1.12 void show_man(void); 8

2.2 Screen.h 8

2.2.1 void get_NowCursor(void); 8
2.2.2 void gotoxy(int x, int y); 8

2.3 System.h 9

2.3.1 char* szFilePath(string txt); 9
2.3.2 int szFileSize(string sFileName); 9

3 Operation Interface 10

3.1 The System Controller 10

3.1.1 void systemController(); 10
3.1.2 void searchKey(); 10
3.1.3 void addDriver(); 10
3.1.4 void addVehicle(); 10
3.1.5 void vehicleListAll(); 11
3.1.6 void driverListAll(); 11
3.1.7 void vehicleListAvaliable(); 11
3.1.8 void driverListAvaliable(); 11
3.1.9 void modifyVehicle(); 11
3.1.10 void modifyDriver(); 12
3.1.11 void driverDelete(); 12
3.1.12 void vehicleDelete(); 12
3.1.13 void DeleteAllVehilce(); 12
3.1.14 void DeleteAllVehilce(); 12
3.1.15 void showEditingRecords(); 13
3.1.16 void deleteHistory(); 13
3.1.17 void doRewardsAndPnishiment(); 13
3.1.18 void payment(); 13
3.1.19 int the_Last_Number(); 13

3.1.20 char* readPassword(); ... 14

4 Data Base & Data Structure ... 15

4.1 File Operation ... 15

4.1.1 virtual void File_Arrange() {}; ... 15
4.1.2 virtual bool File_To_Hashtable() = 0; ... 15
4.1.3 virtual void addNew(void) {}; ... 15
4.1.4 virtual void Modify(void) {}; ... 15
4.1.5 virtual void Delete(void) {}; ... 16
4.1.6 Driver* Find(char *dn); ... 16
4.1.7 Vehicle* Find(char *lp); ... 16
4.1.8 void delete_all(); ... 16

4.2 The Data Structure (Hashtable) ... 17

4.2.1 unsigned long Hash_Key_Name(const char *name); ... 17
4.2.2 virtual void Hash_Expand() {}; ... 17
4.2.3 void Hash_Delete(Driver *obj_driver); ... 17
4.2.4 virtual void Hash_Amend() {}; ... 18
4.2.5 void Hash_Free(); ... 18
4.2.6 void Hash_Print(); ... 18
4.2.7 driver_node* Get_Node(const char *name); ... 18
4.2.8 vehicle_node* Get_Node(const char *name); ... 19

5 Basic Objects ... 20

5.1 Basic Unit: Vehicle ... 20

5.1.1 void browse_Veh(); ... 20
5.1.2 void pilot_Change(char *dn); ... 20
5.1.3 void Pilot_Delete(char *driver_Num); ... 20
5.1.4 void set_Veh(car_info *vehicle); ... 20
5.1.5 car_info* Get_info(); ... 21
5.1.6 void payment(int distance); ... 21
5.1.7 bool canAddDriver() ... 21

5.2 Basic Unit: Driver ... 22

5.2.1 void browse_Driver(); ... 22
5.2.2 void car_Change(char *lp); ... 22
5.2.3 void Car_Delete(char *license_plate); ... 22
5.2.4 void set_User(user_info *target); ... 22
5.2.5 car_info* Get_info(); ... 23
5.2.6 void reward_Driver(int num); ... 23
5.2.7 void punishment_Driver(int num); ... 23
5.2.8 bool canAddVehicle(); ... 23

1 Preface

This a taxi management system whose target users are taxi drivers and Taxi Company's administrators. This system is convenient for taxi drivers to check their up-dated information, such as rewards or punishment records and cars' information. This system also will improve the Taxi Company's management efficiency when it comes to integrating, viewing and modifying vehicle information and driver information.

System specification

1.Base of the system:
This system is constructed based on two hash-tables, one of which is used to store drivers' information and another is to store vehicles' information. The hash-tables are set based on the database file storing information.
By applying algorithm, hash-keys are obtained. Each hash-key points to a node in the hash-table. For example, for the hash-table storing drivers' information, when a node is pointed, the node includes information such as a specific driver's vehicle information, reward and punishment record, etc.

2.User interface:
To provide uncluttered interface and simple views and control, the system simulates LINUX SHELL using C++ codes. There are two threads consisted in the system. The main thread contains function commands, such as registration and editing information. User can call function by inputting corresponding number. The second thread is about system commands, such as exiting the program, returning to previous menu and getting help. User can call system command at any time when the main thread is operating. In particular, a practical system command is designed that is to send database file to a preset e-mail address.

3.Authority of driver:
Driver can search for his own information by inputting his unique job number. Every time when the administrator adds a new driver's information, a job number is generated. This job number is related to the calculation of hash-key. Thus by inputting the job number, the corresponding node of the driver storing hash-table can be found. Then the driver's all information will be shown on the screen.

4.Authority of administrator:
- Add Vehicle or Driver information

The adding process will take the form of Questions and Answers.
When add a driver, a generated job number will be assigned to whom. Then according to the job number, a new hash-key is attained, pointing to a new node in the hash-table. The information collected through Questions and Answers will store in the node.
When add a vehicle information, the process is similar. The only difference is that this time the hash-key is generated based on car's license number.

- Modify

Administrator firstly need to input job number or car's license number to find the specific node in the hash-table. This process is quite similar with the process for drivers searching for their own information. Then the object is located and then be changed. The modified object will then take the place of the old one.

- Delete

Delete user of vehicle information. For the integrity of this function, not only a node can be deleted from the hash-table, but also the data inside the node can be deleted. More specifically, deleting all the information of a driver is called deleting a node. However, if just delete the driver’s vehicle information is not easily deleting the node.

- Registration

This system can register the payment information for vehicle and register rewards and punishments for every driver. Node in the hash-table can be found according to job name and license number and then add information to the node.

- Edit password

The administrator’s password can be changed out of practical consideration

2 User Interface

2.1 Display_interfaces.h

2.1.1 void show_Vehicle_second(void);

Function	Show the second board concerning to the vehicle operations
Mention	This function is only for displaying the choices of operations.
Definition	void show_Vehicle_second(void);
Parameter(s)	NULL
Return value	NULL

2.1.2 void show_Wait(void);

Function	Show the “Congratulations” when it comes to the specific situation
Mention	The function will call Sleep() for three times and each makes a system pause for around 0.4 second.
Definition	void show_Wait(void);
Parameter(s)	NULL
Return value	NULL

2.1.3 void wrong_Search(void);

Function	Show the wrong information when the user name entered in doesn’t correspond to the set names.
Mention	The function only display at the beginning part of the programme.
Definition	void wrong_Search(void);
Parameter(s)	NULL
Return value	NULL

2.1.4 void system_Show(void);

Function	Show the “Linux” user command system.
Mention	The function is called at many places, check the ‘\n’ operations before using it.
Definition	void system_Show(void);
Parameter(s)	NULL
Return value	NULL

2.1.5 void system_Show_client(char* tmp);

Function	Show the "Linux" user command system when the user is not super user but a common user who wants to check his/her informations.
Mention	The function should pass a char* type parameter which represents the name/id of the users in the database.
Definition	void system_Show_client(char* tmp);
Parameter(s)	char* tmp; The user name in the db.
Return value	NULL

2.1.6 void system_Show_driver(void);

Function	Show the system sign of the driver operations.
Mention	Only be called at the second board
Definition	void system_Show_driver(void);
Parameter(s)	NULL
Return value	NULL

2.1.7 void system_Show_vehicle(void);

Function	Show the system sign of the vehicle operations.
Mention	Only be called at the second board
Definition	void system_Show_vehicle(void);
Parameter(s)	NULL
Return value	NULL

2.1.8 void vehicle_Board(void);

Function	Show the options concerning to the vehicles.
Mention	Only be called at the second board
Definition	void vehicle_Board(void);
Parameter(s)	NULL
Return value	NULL

2.1.9 void driver_Board(void);

Function	Show the options concerning to the drivers.
Mention	Only be called at the second board
Definition	void driver_Board(void);
Parameter(s)	NULL
Return value	NULL

2.1.10 void show_Search(void);

Function	Show the search mentions above the searching rectangle.
Mention	NULL
Definition	void show_Search(void);
Parameter(s)	NULL
Return value	NULL

2.1.11 void show_Super(void);

Function	Show the super information
Mention	NULL
Definition	void show_Super(void);
Parameter(s)	NULL
Return value	NULL

2.1.12 void show_man(void);

Function	Show the system commands.
Mention	Only show when the user enters the “man” command.
Definition	void show_man(void);
Parameter(s)	NULL
Return value	NULL

2.2 Screen.h

2.2.1 void get_NowCursor(void);

Function	Get the current position of the cursor.
Mention	Mention the global variables.
Definition	void get_NowCursor(void);
Parameter(s)	NULL
Return value	NULL

2.2.2 void gotoxy(int x, int y);

Function	Go to the set position
Mention	Only pass in the interger type of data.
Definition	void gotoxy(int x, int y);
Parameter(s)	int x, int y;
Return value	NULL

2.3 System.h

2.3.1 char* szFilePath(string txt);

Function	Return the file path
Mention	Only pass in the string type of data.
Definition	char* szFilePath(string txt);
Parameter(s)	String txt;
Return value	NULL

2.3.2 int szFileSize(string sFileName);

Function	Return the integer type of data which represents the file size.
Mention	Only pass in the string type of data.
Definition	int szFileSize(string sFileName);
Parameter(s)	String sFileName
Return value	NULL

3 Operation Interface

3.1 The System Controller

3.1.1 void systemController();

Function	Construction of the class.
Mention	This function can not repetition. This function should be called at the very beginning as a pointer.
Definition	void systemController();
Parameter(s)	NULL
Return value	NULL

3.1.2 void searchKey();

Function	When the system has collected a key, no mater it is a driver number or vehicle license plate, the function will print out the information of the object or not search
Mention	This function can not repetition. This function should be called at the very beginning as a pointer.
Definition	void searchKey();
Parameter(s)	NULL
Return value	NULL

3.1.3 void addDriver();

Function	Used to add a new driver and generate a sequential number started from 10000.
Mention	addDriver()
Definition	void addDriver();
Parameter(s)	NULL
Return value	NULL

3.1.4 void addVehicle();

Function	Used to add a new driver and avoid.
Mention	addVehicle();
Definition	void addVehicle(void);
Parameter(s)	NULL
Return value	NULL

3.1.5 void vehicleListAll();

Function	Used to display all the vehicles and their information in the database.
Mention	vehicleListAll();
Definition	void vehicleListAll();
Parameter(s)	NULL
Return value	NULL

3.1.6 void driverListAll();

Function	Used to display all the Drivers and their information in the database.
Mention	driverListAll();
Definition	void driverListAll();
Parameter(s)	NULL
Return value	NULL

3.1.7 void vehicleListAvaliable();

Function	Used to display all the vehicles which are able to attach new driver and their information in the database.
Mention	vehicleListAvaliable();
Definition	void vehicleListAvaliable();
Parameter(s)	NULL
Return value	NULL

3.1.8 void driverListAvaliable();

Function	Used to display all the driver which are able to attach new vehicle to them and their information in the database.
Mention	driverListAvaliable();
Definition	void driverListAvaliable();
Parameter(s)	NULL
Return value	NULL

3.1.9 void modifyVehicle();

Function	Modify the vehicle information
Mention	modifyVehicle()
Definition	void modifyVehicle();
Parameter(s)	NULL
Return value	NULL

3.1.10 void modifyDriver();

Function	Modify the vehicle information
Mention	modifyDriver()
Definition	void modifyDriver()
Parameter(s)	NULL
Return value	NULL

3.1.11 void driverDelete();

Function	Delete one driver, the function will ask the user to select the object
Mention	driverDelete()
Definition	void driverDelete();
Parameter(s)	NULL
Return value	NULL

3.1.12 void vehicleDelete();

Function	Delete one vehicle, the function will ask the user to select the object
Mention	vehicleDelete()
Definition	void vehicleDelete();
Parameter(s)	NULL
Return value	NULL

3.1.13 void DeleteAllVehilce();

Function	Delete all vehicle, the function will ask the user to select the object
Mention	DeleteAllvehicle()
Definition	void DeleteAllVehilce();
Parameter(s)	NULL
Return value	NULL

3.1.14 void DeleteAllVehilce();

Function	Delete all drivers, the function will ask the user to select the object
Mention	DeleteAllDriver()
Definition	void DeleteAllVehilce();
Parameter(s)	NULL
Return value	NULL

3.1.15 void showEditingRecords();

Function	Show all the editing records
Mention	showEditingRecords()
Definition	void showEditingRecords();
Parameter(s)	NULL
Return value	NULL

3.1.16 void deleteHistory();

Function	Delete all the recording history
Mention	deleteHistory()
Definition	void deleteHistory()
Parameter(s)	NULL
Return value	NULL

3.1.17 void doRewardsAndPnishiment();

Function	Ask the user to note the rewards or punishment of the driver.
Mention	doRewardsAndPnishiment()
Definition	void doRewardsAndPnishiment();
Parameter(s)	NULL
Return value	NULL

3.1.18 void payment();

Function	Ask the user to note the payment of the vehicle.
Mention	doRewardsAndPnishiment()
Definition	void payment();
Parameter(s)	NULL
Return value	NULL

3.1.19 int the_Last_Number();

Function	Return the current number of the driver number
Mention	int id = the_Last_Number();
Definition	int the_Last_Number()
Parameter(s)	NULL
Return value	int the current id number;

3.1.20 char* readPassword();

Function	Get the user-set password.
Mention	If there is no user-set password, default value is 1111.
Definition	char* readPassword();
Parameter(s)	NULL
Return value	char* (the user-set password)

4 Data Base & Data Structure

4.1 File Operation

4.1.1 virtual void File_Arrange() {};

Function	Arrange the sequence of each structure in the file.
Mention	This function is only called after deleting a certain structure in file.
Definition	virtual void File_Arrange() {};
Parameter(s)	NULL
Return value	NULL

4.1.2 virtual bool File_To_Hashtable() = 0;

Function	Create the Hashtable at the very beginning.
Mention	This function is only called once the program runs.
Definition	virtual bool File_To_Hashtable() = 0;
Parameter(s)	NULL
Return value	NULL

4.1.3 virtual void addNew(void) {};

Function	Add new object either in the Hashtable and corresponding file.
Mention	This function is called once a new object is created and expected to be added in the data base.

Definition	void addNew(user_info *struct_new);
	void addNew(car_info *struct_new);

Parameter(s)	The address of the struct information for new object
Return value	NULL

4.1.4 virtual void Modify(void) {};

Function	Amend existing object either in the Hashtable and corresponding file.
Mention	This function is called once the amended version of an object is created and expected to be added in the data base.

Definition	void Modify(Driver *obj_origin, Driver &obj_new);
	void Modify(Vehicle *obj_origin, Vehicle &obj_new);

Parameter(s)	The address of "old" object, the new object
Return value	NULL

4.1.5 virtual void Delete(void) {};

Function	Delete an existing object either in the Hashtable and corresponding file.
Mention	This function is called once an existing object is expected to be delete. All corresponding information revolved around this object is deleted.

Definition	void Delete(Driver *obj_origin);
	void Delete(Vehicle *obj_origin);

Parameter(s)	The address of "old" object
Return value	NULL

4.1.6 Driver* Find(char *dn);

Function	Find a certain driver in the Hashtable bases on the diver number.
Mention	1. Link a vehicle to its driver (one of). 2. Search a certain driver bases on specific conditions. This function is declared in class: 'Driver_File'.
Definition	Driver* Find(char *dn);
Parameter(s)	char *dn (driver number)
Return value	The target's address (Driver*).

4.1.7 Vehicle* Find(char *lp);

Function	Find a certain vehicle in the Hashtable bases on the license plate.
Mention	3. Link a driver to its vehicle 4. Search a certain vehicle bases on specific conditions. This function is declared in class: 'Vehicle_File'.
Definition	Vehicle* Find(char *lp);
Parameter(s)	char *lp (license plate)
Return value	The target's address (Vehicle*).

4.1.8 void delete_all();

Function	Delete all information in the data base.
Mention	Everything either in the Hashtable and corresponding file is deleted.
Definition	void delete_all();
Parameter(s)	NULL
Return value	NULL

4.2 The Data Structure (Hashtable)

4.2.1 unsigned long Hash_Key_Name(const char *name);

Function	Calculate the hash key through an effective algorithm
Mention	This function is based on a given typr-‘char’ string. Algorithm applied on the “driver number” and “license plate” are similar.
Definition	unsigned long Hash_Key_Name(const char *name);
Parameter(s)	const char *name
Return value	NULL

4.2.2 virtual void Hash_Expand() {};

Function	Add a new object in the Hashtable.
Mention	New object is added into data structure. The object is merely added in the data structure, no operation revolved around file in this function.

Definition	void Hash_Expand(Driver &obj_driver);
	void Hash_Expand(Vehicle &obj_vehicle);

Parameter(s)	The real object (Driver / Vehicle).
Return value	NULL

4.2.3 void Hash_Delete(Driver *obj_driver);

Function	Add a new object in the Hashtable.
Mention	Delete an object in existence is added into data structure. The object is not only deleted in the Hashtable, but also all corresponding information revolved around this car is deleted (someone who owns this car).

Definition	void Hash_Delete(Driver *obj_driver);
	void Hash_Delete(Vehicle *obj_vehicle);

Parameter(s)	The pointer of the object (Driver / Vehicle).
Return value	NULL

4.2.4 virtual void Hash_Amend() {};

Function	Modify the object in the Hashtable (combination of delete and add).
Mention	The first step: Delete the original object (node). The second step: Add (form) the new object (node).

Definition	void Hash_Amend(Driver *obj_origin, Driver &obj_new);
	void Hash_Amend(Vehicle *obj_origin, Vehicle &obj_new);

Parameter(s)	The pointer of the object (Driver / Vehicle).
Return value	NULL

4.2.5 void Hash_Free();

Function	Free the Hashtable (all allocated memory is released).
Mention	Free the linked-list from the first node to the last node (in each bucket).
Definition	void Hash_Free();
Parameter(s)	NULL
Return value	NULL

4.2.6 void Hash_Print();

Function	Display all content in the Hashtable.
Mention	Print out in the sequence: from the first bucket to the last one.
Definition	void Hash_Print();
Parameter(s)	NULL
Return value	NULL

4.2.7 driver_node* Get_Node(const char *name);

Function	Fetch out a certain node in Hashtable (Driver).
Mention	This function is used to get information precisely. Link two Hashtable through special interface.
Definition	driver_node* Get_Node(const char *name);
Parameter(s)	const char *name
Return value	The correct node of the target driver.

4.2.8 vehicle_node* Get_Node(const char *name);

Function	Fetch out a certain node in Hashtable (Vehicle).
Mention	This function is used to get information precisely. Link two Hashtable through special interface.
Definition	vehicle_node* Get_Node(const char *name);
Parameter(s)	const char *name
Return value	The correct node of the target vehicle.

5 Basic Objects

5.1 Basic Unit: Vehicle

5.1.1 void browse_Veh();

Function	Browse all vehicles' information
Mention	Belong to Vehicle class. This function can be used many times.
Definition	void browse_Veh();
Parameter(s)	NULL
Return value	NULL

5.1.2 void pilot_Change(char *dn);

Function	Change related pilot
Mention	Belong to Vehicle class. This function can be used many times.
Definition	void pilot_Change(char *dn);
Parameter(s)	char *dn
Return value	NULL

5.1.3 void Pilot_Delete(char *driver_Num);

Function	Delete related pilot
Mention	Belong to Vehicle class. This function can be used many times.
Definition	void Pilot_Delete(char *driver_Num);
Parameter(s)	char *driver_Num
Return value	NULL

5.1.4 void set_Veh(car_info *vehicle);

Function	Set vehicle
Mention	Belong to Vehicle class. This function can be used many times and the input parameter is a structure variable.
Definition	void set_Veh(car_info *vehicle);
Parameter(s)	car_info *vehicle
Return value	NULL

5.1.5 car_info* Get_info();

Function	Get all information about this car
Mention	Belong to Vehicle class. This function can be used many times.
Definition	car_info* Get_info();
Parameter(s)	NULL
Return value	Return structure variable

5.1.6 void payment(int distance);

Function	Give the payment to this vehicle
Mention	Belong to Vehicle class. This function can be used many times and input an integer number
Definition	void payment(int distance);
Parameter(s)	int distance
Return value	NULL

5.1.7 bool canAddDriver()

Function	To check if this vehicle reaches the maximum driver.
Mention	Belong to Vehicle class This function can be used many times.
Definition	bool canAddDriver();
Parameter(s)	NULL
Return value	Return a boolean value

5.2 Basic Unit: Driver

5.2.1 void browse_Driver();

Function	Browse all drivers' information.
Mention	Belong to Driver class. This function can be used many times.
Definition	void browse_Driver();
Parameter(s)	NULL
Return value	NULL

5.2.2 void car_Change(char *lp);

Function	Change related vehicle
Mention	Belong to Driver class. This function can be used many times.
Definition	void car_Change(char *lp);
Parameter(s)	char *lp
Return value	NULL

5.2.3 void Car_Delete(char *license_plate);

Function	Delete related car.
Mention	Belong to Driver class. This function can be used many times.
Definition	void Car_Delete(char *license_plate);
Parameter(s)	char *license_plate
Return value	NULL

5.2.4 void set_User(user_info *target);

Function	Set user information bases on a struct.
Mention	Belong to Driver class. This function can be used many times and the input parameter is a structure variable.
Definition	void set_User(user_info *target);
Parameter(s)	user_info *target
Return value	NULL

5.2.5 car_info* Get_info();

Function	Get all information about this driver.
Mention	Belong to Driver class. This function can be used many times.
Definition	car_info* Get_info();
Parameter(s)	NULL
Return value	Return structure variable.

5.2.6 void reward_Driver(int num);

Function	Give the reward to this driver
Mention	Belong to Driver class. This function can be used many times and input an integer number
Definition	void reward_Driver(int num);
Parameter(s)	int num;
Return value	NULL

5.2.7 void punishment_Driver(int num);

Function	Give the punishment to this driver
Mention	Belong to Driver class. This function can be used many times and input an integer number
Definition	void punishment_Driver(int num);
Parameter(s)	int num;
Return value	NULL

5.2.8 bool canAddVehicle();

Function	To check if this driver reaches the maximum vehicle.
Mention	Belong to Driver class. This function can be used many times.
Definition	bool canAddVehicle();
Parameter(s)	NULL
Return value	Return a boolean value.