Technical Documentation

The Software is designed in Visual Studio 2013 and has been made to serve all the Travelling related needs of the IIT-Guwahati fraternity.

Pre-Requisite software for Proper Functioning

* MS Access
* And two database engines which we have bundled with this software, which need to be installed before running the software (AccessDatabaseEngine.exe, AccessDatabasefix.exe)

Database Management

* Database Used: MS Access 2016
* Connection type Used: Microsoft.ACE.OLEDB.12.0
* ALL the Database connections are done dynamically via OleDbCommands.

There is one Central Database for the entire software. It has the Following tables.

User

Table to store the data for all users (all types to users). It is accessed throughout the software while login, signup, bookings etc.

It has the following columns

* User – Username for the user (Primary key, can’t be duplicate)
* Password- Password for the user, required for login
* FullName – Full Name of the User
* PhoneNumber – Phone Number of the user, for all communication purpose
* Type – Used for identifying the type of user (Customer, ShortDistanceDriver, LongDistanceDriver, ErickshawDriver)

Drivers\_shortDist

Table used for storing details of all the drivers of Short Distance Cabs. It is accessed in Short distance bookings and recommendations for short distance.

It has the following columns:

* FirstName- Stores the Name of the Driver (Needs to be unique currently)
* PhoneNumber – Phone Number of the Driver, for all communication purpose
* CarModel – Car Model of the Driver, Helpful for the Customer to decide which cab they prefer.
* NumberPlate – Number plate of the car of the driver
* Fare – Fare charged per kilometre by the driver
* Location – Current location of the driver out of a given list of Locations
* Available – Stores if the Driver is currently available for booking or no.

Drivers\_Erickshaw

Table used for storing the details of all e-rickshaw drivers. It is accessed in the process of recommending Erickshaw drivers. It has the following columns:

* ID – Stores the unique ID of each driver
* Driver\_Name: (Needs to be unique currently)
* Phone\_No – The driver’s mobile no
* Location – The driver’s current location (one of 13 locations marked on the IITG map)
* Availability – Stores whether the driver is available for booking
* Photo – Driver’s photo

BusTimings

Table used for storing the details of each bus – its starting time, availability, pick up points etc. It has been used in the bus timings section of the software to display the bus timings given the source and destination of the user

* ID – Stores the unique ID of each bus
* Availability
* Bus\_Time – The time that particular bus starts service each day
* Starting\_Point – The starting point of the bus’s route
* Pick\_up\_points – The pick up points for the bus along its route
* To – The end point of the bus’s route

Bookings

Table used for storing all the booking details (Only short/long distance cabs can be booked; e-rickshaws can’t be)

* ID – Unique ID for each booking
* Date\_time – Date and time of the booking
* Driver – Full name of the driver
* Short/Long Distance – Whether the booking is for short or long distance cab
* Destination – Travel destination
* User – Username of the customer
* Fullname – Fullname of the customer
* UserPhNo – The user’s mobile no
* Source – Travel source
* Days – Denotes the no of days for which cab has been booked. Has value 1 by default. Only long-distance cabs can be booked for more than a day

Drivers\_LongDistance

Table used for the storing the details of long-distance drivers. It has been used in the long-distance cabs booking section of the software

* ID – Unique ID of the driver
* f\_name – The first name of the driver (Needs to be unique currently)
* l\_name – The last name of the driver
* ph\_no – Driver’s mobile no
* car\_no – The driver’s car’s number plate
* status – The driver’s availability status
* will\_go – The places the driver is willing to go
* price – The approximate price that the driver will charge for each place he is willing to go

Visual Studio

We have used windows form application with .NET Framework 4.5. There are a total of 15 Forms, divided into 6 Broad Modules, which will be hence described.

User Control and Dashboard

Home

The first window opened on the start of the software is the Homepage. IT is the base form for the entire software and remains active throughout the software runtime. It has the following Functionalities:

It has buttons to Open all the different Modules:

* Short Distance Bookings
* Long Distance Booking
* Recommendation for short distance
* Bus Timings
* Erickshaw Display
* Emergency contacts

It also has buttons for user management including:

* Login
* Logout
* Signup
* And Profile Management

It also stores the details of the current logged in user, which is used by all the forms to check if the user is logged in.

It also has a function called DistanceCalc, which calculates the distance between two points from the 8 fixed points on the Guwahati map. It contains a 2D matrix which stores the distance between any 2 points and a dictionary which converts the destination from string to the corresponding number

Login

Basic login form for user of any type, just asks for username and password, has a button to reveal or hide the password written

Also has a button to go to signup page if user doesn’t have an account

Major Function: LoginButton\_Click - Accesses the user database and checks if the given combination exists, if yes then sets the details in the home form

Signup

Takes the following details from the user:

* UserName
* Password
* PhoneNumber
* FullName
* User Type
* Captcha

Data validation for them is as follows:

* Confirms Captcha is correct
* PhoneNumber is 10 digits
* Username is alphanumeric or ‘\_’ and nonempty
* Password is non-empty
* Full Name is Alphabetic and non-empty

And makes a user profile with the given details

The Main function is Button1\_Click which corresponds to pressing the signup button, it validates the data and then inserts a row into the users table and also a row into the corresponding table depending on the usertype taken if the user is a driver.

CAUTION: The Driver Needs to update all his details by going to the profile page before being able to get any bookings

Another Sub BtnRefresh\_Click corresponds to generating a captcha, it is called evertime the signup button is clicked or refresh captcha is clicked, it chooses 6 random alphanumeric characters and creates a bitimage. and stores the string into Str for validating the captcha.

UserDetails

It is the form to update the details of the user (of all types). It has Tab Control, which enables it to have 4 different tabs, only one of which is accessible to the user depending on the usertype of the user.

It has 4 different functions for loading the details of the user, depending on the usertype:

* CustLoad
* ShortDriverLoad
* LongDriverLoad
* ErickshawLoad

They first access the User table and read the details of the user and display the details, then they access the respective tables in the database and display those details. The User table is searched by the username and the respective drivers table is searched using the Full Name.

The form has a selection function which selects which tab to load and which loading function to call depending on the usertype.

After Loading the User can edit the details. And then Update his details by pressing the respective Update button in each tab. There is also a bookings button which directs the user to the Bookings form, which displays all the current bookings for the current user.

While Updating, first the details in the Users table is updated, following which the functions accesses the respective drivers table and update the data there.

The Major Functionality is that Drivers can update their location and availability by logging in, and both users and drivers can view their current bookings.

# ReccoHome

ReccoHome works as the Home page for recommending the optimal route for Travelling between two points. And allows the user to book a cab if it is a part of the route.

Algorithm Used

As this software is centred around the IITG fraternity, it aims to maximise the use of IITG bus services along the route. The basic idea behind the algorithm is that if either the source or destination lie on a bus route within the next hour then it recommends using that bus.

To achieve this, the algorithm first checks if the source is on a bus route and there is a stop on the route such that the distance between that stop and destination is a less than the original distance by a factor. It then checks if there is a cab available in the vicinity of that location, it does this for all the stops on the route, and then takes the minimum and recommends that. If the Destination is also on the bus route then, then it recommends directly using the bus and avoiding cabs.

Now if the source is not on the route but the destination is, then it repeats the same algorithm as above, first checks if there are cabs from the current location then, finds the closest bus route such that a bus departs there at least 15 minutes later and within an hour .

It also shows normal cab option from source to destination so that the user doesn’t have to change form to book a direct cab.

It contains the following buttons:

* Open Map
* Find cabs
* Confirm cab
* Go back

Contains the following major functions:

* OuttxtAdder-

To dynamically allocate output RIchtextboxes and radioButtons

* Clr –

to clear all dynamically allocated controls and reset variables.

* Button1\_click(Handles btnFindCabs.Click) –

to access DataBase and find Cabs and Busses using SQL

* printSourceAndDest –

to Print data input by user in map into textboxes.

* Button\_Click\_1(Handles btnConfirm.Click)-

Accesses DB again to fetch more details about the driver

Along the entire flow of control, DB is accessed, specifically table of ‘Driver\_shortDist’ and ‘BusTimmings’.

ShortHome

ShortHome form (dubbed as Form1.vb) works as the Home page for Selecting Short distance cabs by customer, sorted in order of price and checked for availability

It contains the following buttons:

* Open Map
* Find cabs
* Confirm cab
* Go back

Contains the following major functions:

* OuttxtAdder-

To dynamically allocate output RIchtextboxes and radioButtons

* Clr –

to clear all dynamically allocated controls and reset variables.

* Button1\_click(Handles btnFindCabs.Click) –

to access DataBase and find Cabs using SQL querying (SELECT cabs that match the user’s location).

* printSourceAndDest –

to Print data input by user in map into textboxes.

* Button\_Click\_1(Handles btnConfirm.Click)-

Accesses DB again to fetch more details about the driver

Along the entire flow of control, DB is accessed, specifically table of ‘Driver\_shortDist’. First access if done with respect to Driver’s location, and output is sorted in order of Price from low to high. The total estimated fare is found by Multiplying Distance between two locations to Fare per km of each driver.

The Second access is wrt selected Driver’s Name to fetch other details about the driver.

DriverConfirmation\_short

Public variables declared to transfer data between different forms:

* Fare
* driverName
* otherDetails
* locationTo
* locationFrom

Main subs and their function –

* Load sub
* To initialise labels with driver details
* Check if user is logged in
* Get user’s name and phone number to display on textboxes
* btnConfirm
* Inserting details into bookings table in DB

Buttons used –

* Confirm button
* Back button

The databse is accessed twice in the form, once to get user details upon loading and secondly to Store Booking confirmation into Bookings table in DB.

Map

The map.vb form outlines the map of Guwahati city with 8 hotspots for the user to select his/her location and destination.

This form is used both for Short Distance cabs and Recommended cabs

Buttons used –

* My Location
* My destination
* Clear choices?
* Confirm

txtboxCurrent stores the current radio button selection of User.

Main function-

* btnConfirm
* Sets the chosen locations to public variables in ShortHome/ReccoHome to transferdata to those forms

Erickshaw

Dist Array

Drivers Erickshaw Table

Access (DBControl Objeect)

Data Grid View

Top Level Working

The “Erickshaw Home” form is used to get the source and destination of the user

It then passes this data to the “E\_rick\_driver\_reco” form which is used to recommend the appropriate driver for the user

Recommendation Logic

The driver which is closest to the user’s current location, among the list of available drivers is recommended to the user. All the other available users are displayed in the “Other Available Drivers” list

Computing the distance of the drivers

For the purposes of distance computation, the 13 IITG locations are divided into groups of 8 where each group has locations that are close to each other. Then the distance between each pair of groups is calculated using Google maps, and stored in the “dist” array.

A separate dictionary stores the group no of each of the 13 locations

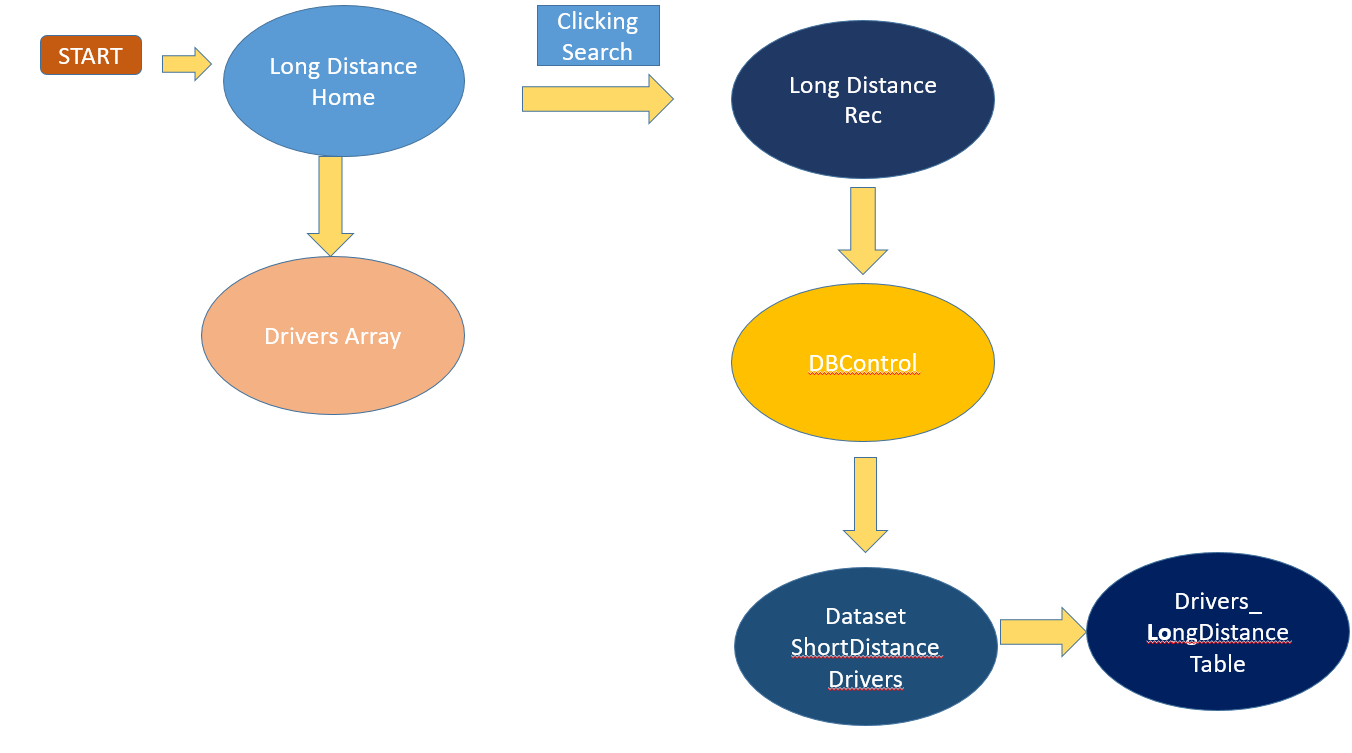
Interacting with the database

A separate object “Access” of the “DBControl” class has been created to facilitate the querying of our access database. It has primarily been used to extract the sub table of available drivers from the “Drivers\_Erickshaw” table

Long Distance Recommendation Form

* Database: We have used same data table as used in the long-distance home form.
* **Algorithm**
* In the long-Distance home form, we store the drivers’ name that had appeared in the output boxes using a list. When user select a RadioButton to select a driver, with its index we get the selected driver's name from the driver's list and we will send the driver name to long distance recommendation form.
* In the long-distance recommendation form we have declared a dictionary to store the places that are in Drivers\_LongDistance table and with respect to every place we have also stored an integer which is the number of maximum days that driver will stay in a place.
* After getting the selected driver name from the long-distance home form we will use SQL command to search the driver's name and place name that selected driver would go in the data table Drivers\_LongDistance.
* Since we have the places that selected driver would go to, now with the help of the place name we can the get maximum number of days with this number we will generate RadioButton as equal to maximum number of days.
* Once a RadioButton is selected the program will calculate the total price just simply multiplying the price per day with the selected RadioButton's index.
* If no RadioButton is selected than program will through message box asking user to select one.
* At start we start from Long Distance Home, by clicking the search button we go to Long Distance Rec. Here table Drivers Long Distance is accessed of Datasets Drivers
* The accessed Drivers are stored in Drivers Array for manipulation.
* By manipulating the this array we get the drivers for destination that we desire
* **Working of different buttons**
* **confirm?** After selecting a RadioButton confirm button will calculate the price and it will show the price in label.
* **cancel**: It will hide form long distance recommendation.
* **OK**: For final decision.

Flowchart:



* At start we start from Long Distance Home,by clicking the search button we go to Long Distance Rec. Here table Drivers\_Long Distance is accesed of DataSet Drivers
* The accessed Drivers are stored in Drivers Array for manuplation.

By manuplating the this array we get the drivers for destination that we desir

Bus Timmings

FIND

Clr() function is called initially to clear the output that is to clear the content which is displayed in the listview to help the user to give the input again.

Dictionary is used to store the full names of the locations since in the database locations are specified in their shortforms. This will help the user to select the location easily.

SetupListView() function is called to set the properties of it and to add the headings of each column.

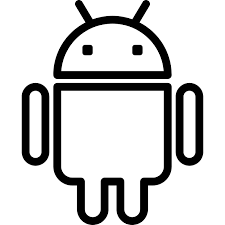
Whenever the substring of the column Pick\_up\_points of the database ShortDist\_drivers.accdb matches both the locations given as input in the comboboxes then those particular rows are displayed in the listview.

Check the Bus timings of those particular rows and if the time difference is less than or equal to thirty minutes than those rows are highlighted.

# Future Scope of the App



* Use of real-time maps (such as google maps) into the program.



* Moving to a more android based framework for common use of all IIT fraternity.

# Limitations of the App

* Actual booking not implemented, only bookings record is updated.
* Distances within and outside campus was hardcoded.