**PROJECT REPORT ON**

**BUS TIMING MANAGEMENT SYSTEM**

**¬**

**By**

**Syed Nazmul Islam**

**IT-22023**

**Session 2021-2022**

**Under the Supervision of**

**Md. Mahedi Hasan**

**Lecturer**

**Department of Information and Communication Technology**

**1. Introduction**

The **Bus Timing Management System** is a simple program designed to help users check bus schedules and view the next available bus. This system is developed using **C++** and **SQLite** to manage and store bus schedules and login information. It provides a user-friendly interface for both bus passengers and the admin panel to view, add, and remove bus timings.

**2. Objective**

The main objective of the project is to build a simple and efficient bus scheduling system that can:

* Show next bus schedule time.
* Allow the admin to add or remove bus schedules.
* Manage login for admin to do changes in schedules.

**3. System Features**

* **Bus Schedule Viewing**: Users can view the next available bus based on the current time.
* **Admin Login System**: Admin can add or remove bus schedules after login with their password.
* **Data Storage**: Bus timings and admin login information’s are stored in an SQLite database.

**4. Technologies Used**

* **C++**: Used for coding the program.
* **SQLite**: A database used to store bus timings and login information.
* **CodeBlocks**: The code editor used for programming c++.
* **SQLite3 Library**: Integrated with C++ to manage database operations.

**5. SQLite Database**

Two tables are used to store information:

1. **bus Table**:
   * + time (INT): Stores the bus departure time in HHMM format.
     + origin (VARCHAR (50)): Stores the departure location of the bus.
     + destination (VARCHAR(50): Stores the destination location of the bus.

***SQL Query****: CREATE TABLE IF NOT EXISTS bus (time INT, origin VARCHAR(50), destination VARCHAR(50));*

1. **login info Table**:
   * + username (VARCHAR (50)): Stores the username for login.
     + password (INT): Stores the password as an integer.

***SQL Query****: CREATE TABLE IF NOT EXISTS logininfo(username VARCHAR(50), password INT);*

**6. Key Functionalities**

1. **Next Bus Display**:
   * Displays the next available bus that is departing after the current time.
   * Uses current time to filter bus departing time.
   * Displays information such as the departure time, origin, and destination.
2. **Login System**:
   * Verifies user username and password from the database.
   * If the admin is not registered, the system do registration.
   * After login, the admin can access functionalities to add or remove buses.
3. **Admin Panel**:
   * **Add New Bus**: Admin can add new bus schedules by giving the time, origin, and destination.
   * **Remove Bus**: Admin can remove a bus from the schedule by specifying the bus’s departure time in HHMM format.
4. **Logout and Termination**:
   * Admin can log out of the system and return to the main menu.
   * Users can terminate the entire program from the menu.

**7. Code Structure**

* **Main Function**: Initializes the SQLite database, creates necessary tables, and continuously displays the menu for user interaction.
* **Menu Function (menubar)**: Displays the main menu and navigates the user to different functions based on their selection.
* **Bus Management Functions**:
  + **nextbus()**: Displays the next available buses from the database.
  + **addnewbus()**: Adds a new bus schedule into the bus table.
  + **removebus()**: Removes a bus schedule from the bus table based on the time provided.
* **Login and Registration Functions**:
  + **login()**:Ask for username and password, checks if they exist in the logininfo table.
  + **registerUser()**: Registers a new admin user if none exist in the system.
  + **isUserRegistered()**: Checks if there is an existing user in the logininfo table.
* **Utility Functions**:
  + **curtime()**: Gets the current time in HHMM format to compare with bus timings.

**8. Challenges and Improvements**

**Challenges**:

1. Making connection between SQLite database and c++ was challenge for me .As I was using database for first time. I had to spend a lot of time to using SQL on c++ code . Setting MySQL with CodeBlocks seems very difficult to me . so I used SQLIte database.
2. I got some difficulty while creating login functionality. As for some buffer was hitting to my code . Every time I got wrong password error while debugging.

**Improvements**:

1. The process of storing password is insecure. In future, password hashing should be implemented for better security.

2. The interface could be expanded to support more user-friendly features, such as searching buses by origin and destination.

3.In future we can develop a GUI interface for better user experience.

**9. Conclusion**

The **Bus Timing Management System** does its job in small environments like ours one work for **MBSTU** Campus. Using **SQLite** database make it more useful. In farther it can be developed by GUI Interface. It successfully does login for changing schedule information.

**10. Future Scope**

There is a huge scope for developing this program. We can use hashing technique for user authentication. It will secure the system more efficiently. We can develop a mobile app using an android system. We may add some search functionality to develop user experience. We may give a GUI interface for better user experience.

**11. Appendix**

**Tools**:

1. C++ Compiler (G++/MinGW)

2. SQLite3 Library

3. CodeBlocks

Source code:

#include <bits/stdc++.h>

#include <cstdio>

#include <ctime>

#include <thread>

#include <windows.h>

#include "sqlite3.h"

#define intx int64\_t

using namespace std;

sqlite3 \*db;

sqlite3\_stmt \*stmt;

int result;

string query;

void menubar();

void nextbus();

void login();

void logout();

void terminatex();

int curtime();

void addnewbus();

void removebus();

int logincheck(string username, int password);

int isUserRegistered();

void registerUser();

int main() {

cout<<endl;

cout<<" {\_\_BUS TIMING MANAGEMENT MBSTU\_\_}"<<endl<<endl;

result = sqlite3\_open("bus\_timing\_info.db", &db);

if (result != SQLITE\_OK) {

cout << "Hmm1..We got some glitch at ->" << sqlite3\_errmsg(db) << endl;

return 1;

}

query = "CREATE TABLE IF NOT EXISTS bus(time INT, origin VARCHAR(50), destination VARCHAR(50));";

result = sqlite3\_prepare\_v2(db, query.c\_str(), -1, &stmt, NULL);

if (result != SQLITE\_OK) {

cout << "Hmm2..We got some glitch at ->" << sqlite3\_errmsg(db) << endl;

}

sqlite3\_step(stmt);

sqlite3\_finalize(stmt);

query = "CREATE TABLE IF NOT EXISTS logininfo(username VARCHAR(50), password INT);";

result = sqlite3\_prepare\_v2(db, query.c\_str(), -1, &stmt, NULL);

if (result != SQLITE\_OK) {

cout << "Hmm4..We got some glitch at ->" << sqlite3\_errmsg(db) << endl;

}

sqlite3\_step(stmt);

sqlite3\_finalize(stmt);

while (1) {

menubar();

}

}

void menubar() {

cout << " ---------------------------" << endl;

cout << " \033[42m| MENU |\033[0m" << endl;

cout << " ---------------------------" << endl;

cout << " | Select Operation-> |" << endl;

cout << " | [1] Watch Next Bus |" << endl;

cout << " | [2] Login |" << endl;

cout << " | [3] Logout |" << endl;

cout << " | [4] Terminate Program |" << endl;

cout << " ---------------------------" << endl;

int operation;

cin >> operation;

getchar();

switch (operation) {

case 1:

nextbus();

break;

case 2:

login();

break;

case 3:

logout();

break;

case 4:

terminatex();

break;

default:

for (int i = 0; i < 15; i++) {

cout << "\033[41m. \033[0m";

Sleep(200);

}

getchar();

cout << "\n\033[31mInvalid choice...Try again.\033[0m" << endl;

menubar();

break;

}

}

void nextbus() {

int now = curtime();

const char\* query = "SELECT \* FROM bus WHERE time >= ?;";

sqlite3\_stmt\* stmt;

int result = sqlite3\_prepare\_v2(db, query, -1, &stmt, NULL);

if (result != SQLITE\_OK) {

cout << "Hmm..We got some glitch at -> " << sqlite3\_errmsg(db) << endl;

return;

}

sqlite3\_bind\_int(stmt, 1, now);

cout << " ---------------------------" << endl;

cout << " \033[46m";

cout << " NEXT BUSES \033[0m" << endl;

while ((result = sqlite3\_step(stmt)) == SQLITE\_ROW) {

int x = sqlite3\_column\_int(stmt, 0);

string s = reinterpret\_cast<const char\*>(sqlite3\_column\_text(stmt, 1));

string ss = reinterpret\_cast<const char\*>(sqlite3\_column\_text(stmt, 2));

int hour = x / 100;

int minute = x % 100;

string period = "AM";

if (hour >= 12) {

period = "PM";

if (hour > 12) hour -= 12;

} else if (hour == 0) {

hour = 12;

}

cout << "\033[92m ---------------------------\033[0m" << endl;

cout << " TIME : " << hour << ":" << setw(2) << setfill('0') << minute << " " << period << endl;

cout << " DEPARTURE : " << s << endl;

cout << " DESTINATION: " << ss << endl;

cout << "\033[92m ---------------------------\033[0m" << endl << endl;

Sleep(100);

}

if (result != SQLITE\_DONE) {

cout << "Hmm..We got some glitch at -> " << sqlite3\_errmsg(db) << endl;

}

sqlite3\_finalize(stmt);

}

void login() {

if (!isUserRegistered()) {

registerUser();

}

cout << " " << "\033[42m Login to Admin \033[0m" << endl;

string username;

int password;

cout << " Enter username ->\033[33m";

//cin.ignore();

getline(cin, username); cout << "\033[0m";

cout << " Enter Password -> \033[33m";

cin >> password; cout << "\033[0m";

if (logincheck(username, password)) {

cout << "\n ";

for (int i = 0; i < 16; i++) {

cout << "\033[100m ";

Sleep(100);

}cout << "\033[0m";

cout << endl;// cout << "\033[100m";//cout << "\033[100m";

cout << "\033[103m\033[30m -------------------------------\033[0m\033[0m" << endl;

cout << "\033[103m\033[30m Admin Panel \033[0m\033[0m" << endl;

cout << "\033[103m\033[30m -------------------------------\033[0m\033[0m" << endl;

cout << "\033[103m\033[30m | [1] Add new schedule |\033[0m\033[0m" << endl;

cout << "\033[103m\033[30m | [2] Remove a schedule |\033[0m\033[0m" << endl;

cout << "\033[103m\033[30m | [3] EXIT Login |\033[0m\033[0m" << endl;

cout << "\033[103m\033[30m -------------------------------\033[0m\033[0m\n" << endl;

int choice;

cin >> choice;

cin.ignore(); // Clear the buffer after input

switch (choice) {

case 1:

addnewbus();

break;

case 2:

removebus();

break;

case 3:

cout << "\033[91m EXITING LOGIN.....\033[0m" << endl;

cout << " ";

for (int i = 0; i < 15; i++) {

cout << "\033[104m \033[0m";

Sleep(200);

}

cout << endl;

menubar();

break;

default:

cout << "Invalid choice...Try again." << endl;

menubar();

}

} else {

cout << "Username or Password is wrong" << endl;

menubar();

}

}

void logout() {

string user;

int pass;

cout<<"Enter your current Username->"<<endl;

cin>>user;

cout<<"Enter your current password->"<<endl;

cin>>pass;

getchar();

query="DELETE from logininfo";

result = sqlite3\_prepare\_v2(db, query.c\_str(), -1, &stmt, NULL);

sqlite3\_step(stmt);

sqlite3\_finalize(stmt);

//Update needs here

menubar();

}

void terminatex() {

cout << "\nProgram is Exiting........" << endl;

for (int i = 0; i < 15; i++) {

cout << "\033[41m. \033[0m";

Sleep(200);

}

cout << "\n...........TERMINATED........." << endl;

exit(0);

}

int curtime() {

time\_t current\_time = time(0);

struct tm\* local\_time = localtime(&current\_time);

int current\_time\_int = (local\_time->tm\_hour \* 100) + local\_time->tm\_min;

return current\_time\_int;

}

void addnewbus() {

int time;

string origin;

string destination;

cout << "Enter Departure Time-> ";

cin >> time;

getchar();

cout << "Enter Departure Location-> ";

getline(cin, origin);

cout << "Enter Destination Location-> ";

getline(cin, destination);

query = "INSERT INTO bus(time, origin, destination) VALUES(?, ?, ?)";

result = sqlite3\_prepare\_v2(db, query.c\_str(), -1, &stmt, NULL);

if (result != SQLITE\_OK) {

cout << "Hmm3..We got some glitch at ->" << sqlite3\_errmsg(db) << endl;

}

sqlite3\_bind\_int(stmt, 1, time);

sqlite3\_bind\_text(stmt, 2, origin.c\_str(), origin.length(), SQLITE\_TRANSIENT);

sqlite3\_bind\_text(stmt, 3, destination.c\_str(), destination.length(), SQLITE\_TRANSIENT);

sqlite3\_step(stmt);

sqlite3\_finalize(stmt);

cout << "\033[93mNew bus added successfully!\033[0m" << endl;

}

void removebus() {

int time;

cout << "Enter Departure Time to remove-> ";

cin >> time;

getchar();

query = "DELETE FROM bus WHERE time = ?";

result = sqlite3\_prepare\_v2(db, query.c\_str(), -1, &stmt, NULL);

if (result != SQLITE\_OK) {

cout << "Hmm3..We got some glitch at ->" << sqlite3\_errmsg(db) << endl;

}

sqlite3\_bind\_int(stmt, 1, time);

sqlite3\_step(stmt);

sqlite3\_finalize(stmt);

cout << "\033[93mBus removed successfully!\033[0m" << endl;

}

int logincheck(string username, int password) {

query = "SELECT \* FROM logininfo WHERE username = ? AND password = ?";

result = sqlite3\_prepare\_v2(db, query.c\_str(), -1, &stmt, NULL);

if (result != SQLITE\_OK) {

cout << "Hmm6..We got some glitch at ->" << sqlite3\_errmsg(db) << endl;

}

sqlite3\_bind\_text(stmt, 1, username.c\_str(), username.length(), SQLITE\_STATIC);

sqlite3\_bind\_int(stmt, 2, password);

int check = 0;

while ((result = sqlite3\_step(stmt)) == SQLITE\_ROW) {

check = 1;

}

sqlite3\_finalize(stmt);

return check;

}

int isUserRegistered() {

query = "SELECT \* FROM logininfo LIMIT 1";

result = sqlite3\_prepare\_v2(db, query.c\_str(), -1, &stmt, NULL);

if (result != SQLITE\_OK) {

cout << "Hmm5..We got some glitch at ->" << sqlite3\_errmsg(db) << endl;

}

int registered = 0;

if (sqlite3\_step(stmt) == SQLITE\_ROW) {

registered = 1;

}

sqlite3\_finalize(stmt);

return registered;

}

void registerUser() {

string username;

int password;

cout << "Create an admin username -> ";

getline(cin, username);

cout << "Create a password -> ";

cin >> password;

getchar();

query = "INSERT INTO logininfo(username, password) VALUES(?, ?)";

result = sqlite3\_prepare\_v2(db, query.c\_str(), -1, &stmt, NULL);

if (result != SQLITE\_OK) {

cout << "Hmm5..We got some glitch at ->" << sqlite3\_errmsg(db) << endl;

}

sqlite3\_bind\_text(stmt, 1, username.c\_str(), username.length(), SQLITE\_STATIC);

sqlite3\_bind\_int(stmt, 2, password);

sqlite3\_step(stmt);

sqlite3\_finalize(stmt);

cout << "Admin created successfully." << endl;

}