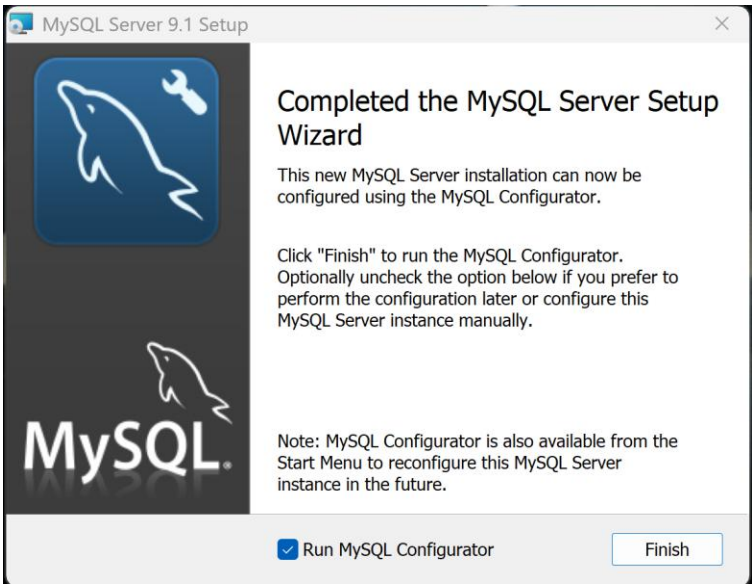
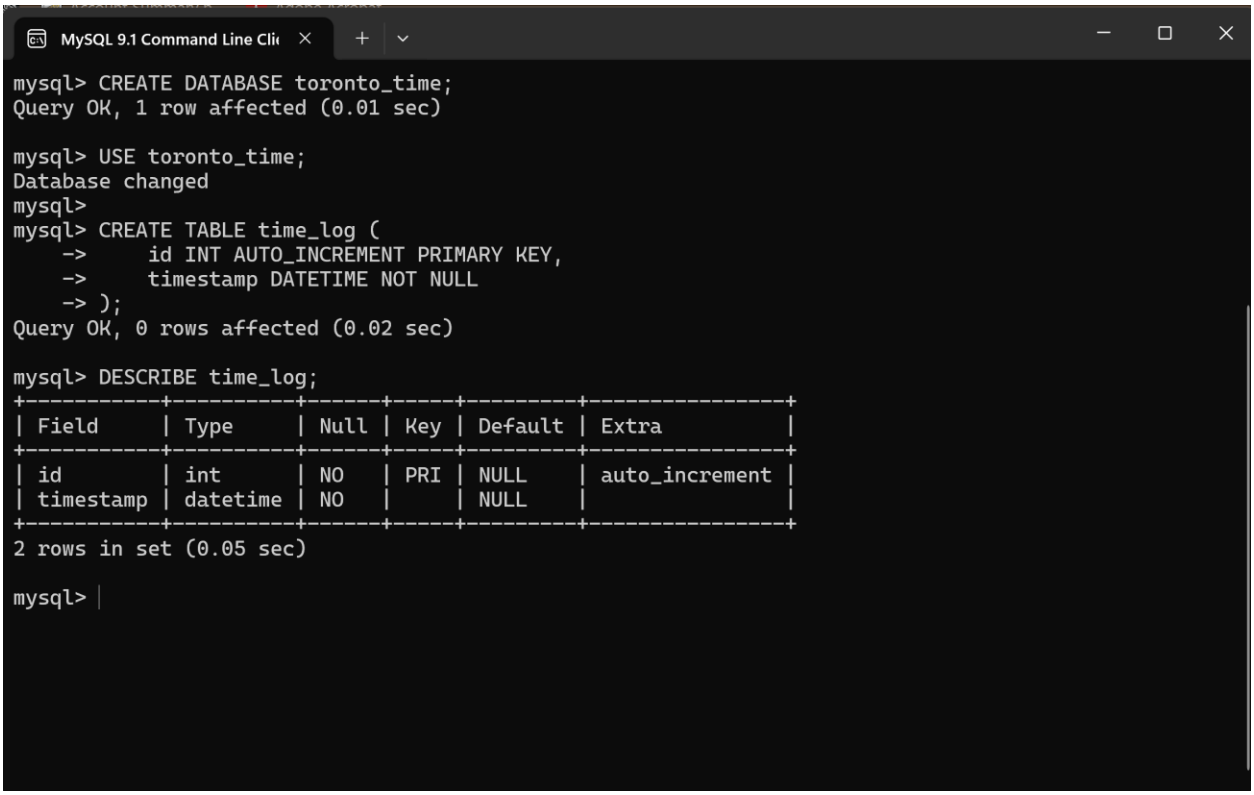
1. **Set Up MySQL Database:**

Install MySQL



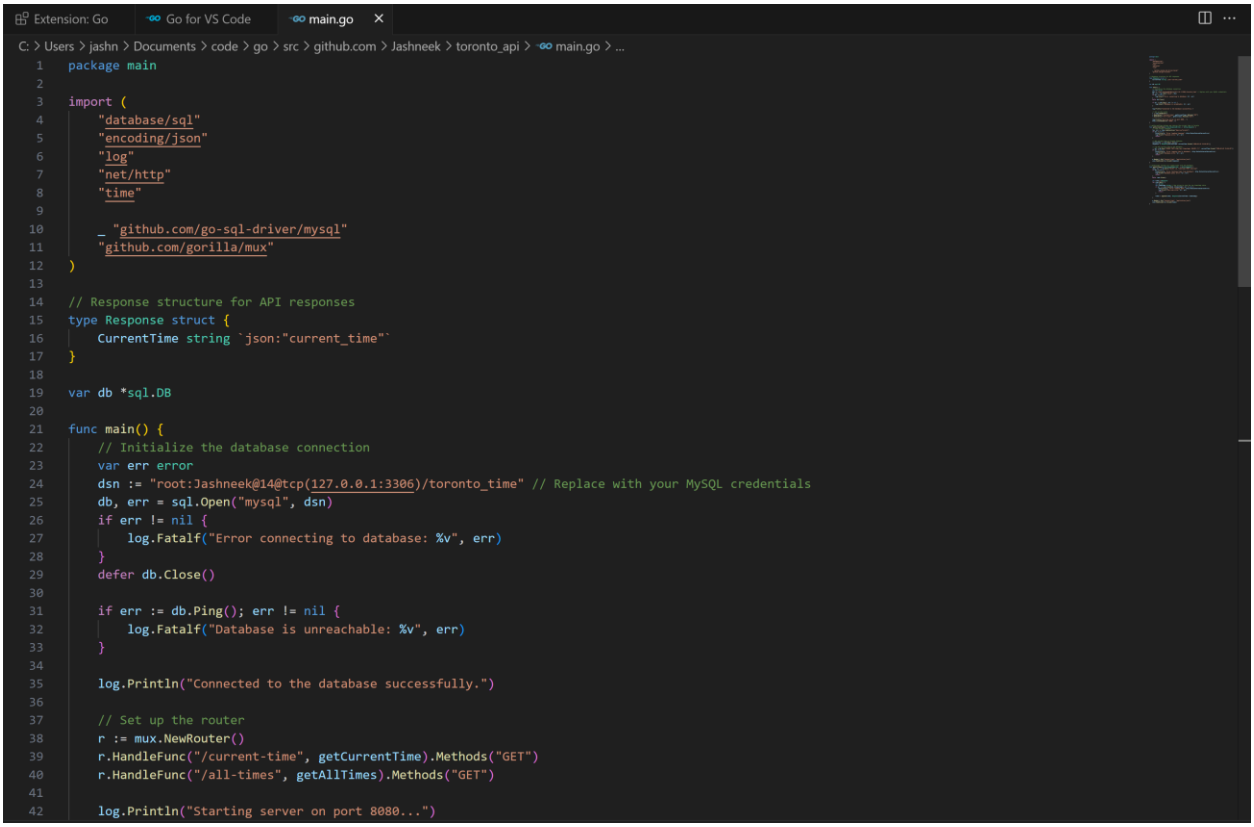

```
mysql> SELECT VERSION ();
+------------+
| VERSION () |
+------------+
| 9.1.0      |
+------------+
1 row in set (0.00 sec)

mysql>
```

Create a new database and then Create a table named time_log with at least two fields: id (primary key) and timestamp.



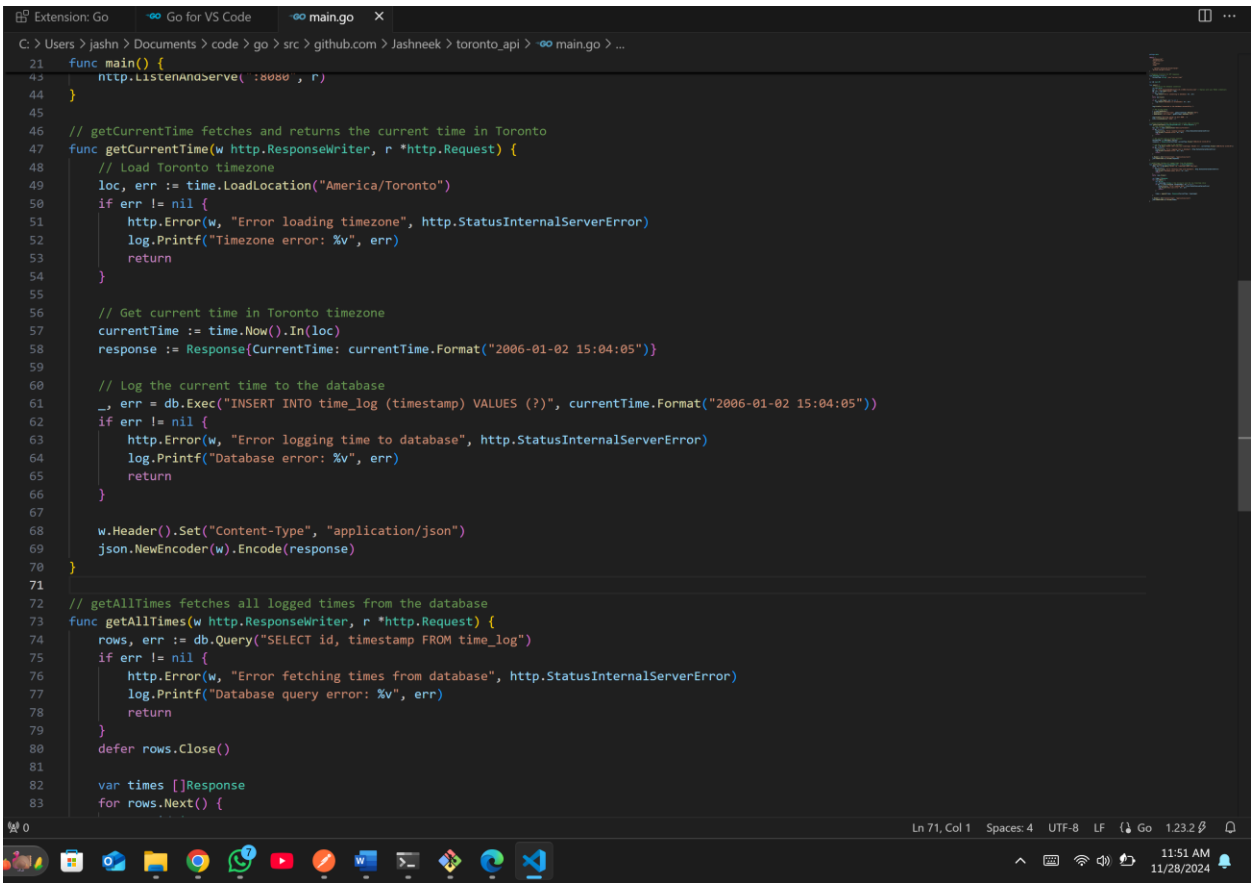
2. **API Development:**

Write a Go application with a web server.



```
Extension: Go | Go for VS Code | main.go
C: > Users > jashn > Documents > code > go > src > github.com > Jashneek > toronto_api > main.go > ...
```

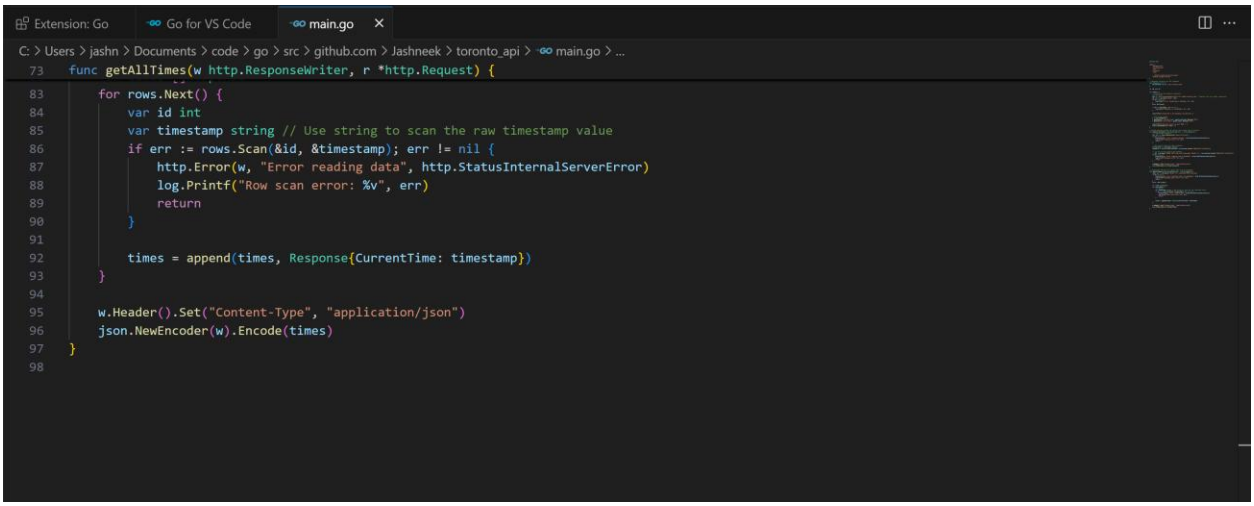
```go
package main

import (
    "database/sql"
    "encoding/json"
    "log"
    "net/http"
    "time"

    _ "github.com/go-sql-driver/mysql"
    "github.com/gorilla/mux"
)

// Response structure for API responses
type Response struct {
    CurrentTime string `json:"current_time"`
}

var db *sql.DB

func main() {
    // Initialize the database connection
    var err error
    dsn := "root:Jashneek@14@tcp(127.0.0.1:3306)/toronto_time" // Replace with your MySQL credentials
    db, err = sql.Open("mysql", dsn)
    if err != nil {
        log.Fatalf("Error connecting to database: %v", err)
    }
    defer db.Close()

    if err := db.Ping(); err != nil {
        log.Fatalf("Database is unreachable: %v", err)
    }

    log.Println("Connected to the database successfully.")

    // Set up the router
    r := mux.NewRouter()
    r.HandleFunc("/current-time", getCurrentTime).Methods("GET")
    r.HandleFunc("/all-times", getAllTimes).Methods("GET")

    log.Println("Starting server on port 8080...")
```



```
Extension: Go | Go for VS Code | main.go
C: > Users > jashn > Documents > code > go > src > github.com > Jashneek > toronto_api > main.go > ...
```

```go
    log.Println("Starting server on port 8080...")
    http.ListenAndServe(":8080", r)
}

// getCurrentTime fetches and returns the current time in Toronto
func getCurrentTime(w http.ResponseWriter, r *http.Request) {
    // Load Toronto timezone
    loc, err := time.LoadLocation("America/Toronto")
    if err != nil {
        http.Error(w, "Error loading timezone", http.StatusInternalServerError)
        log.Printf("Timezone error: %v", err)
        return
    }

    // Get current time in Toronto timezone
    currentTime := time.Now().In(loc)
    response := Response{CurrentTime: currentTime.Format("2006-01-02 15:04:05")}

    // Log the current time to the database
    _, err = db.Exec("INSERT INTO time_log (timestamp) VALUES (?)", currentTime.Format("2006-01-02 15:04:05"))
    if err != nil {
        http.Error(w, "Error logging time to database", http.StatusInternalServerError)
        log.Printf("Database error: %v", err)
        return
    }

    w.Header().Set("Content-Type", "application/json")
    json.NewEncoder(w).Encode(response)
}

// getAllTimes fetches all logged times from the database
func getAllTimes(w http.ResponseWriter, r *http.Request) {
    rows, err := db.Query("SELECT id, timestamp FROM time_log")
    if err != nil {
        http.Error(w, "Error fetching times from database", http.StatusInternalServerError)
        log.Printf("Database query error: %v", err)
        return
    }
    defer rows.Close()

    var times []Response
    for rows.Next() {
```

```
Ln 71, Col 1  Spaces: 4  UTF-8  LF  Go  1.23.2
11:51 AM
11/28/2024
```



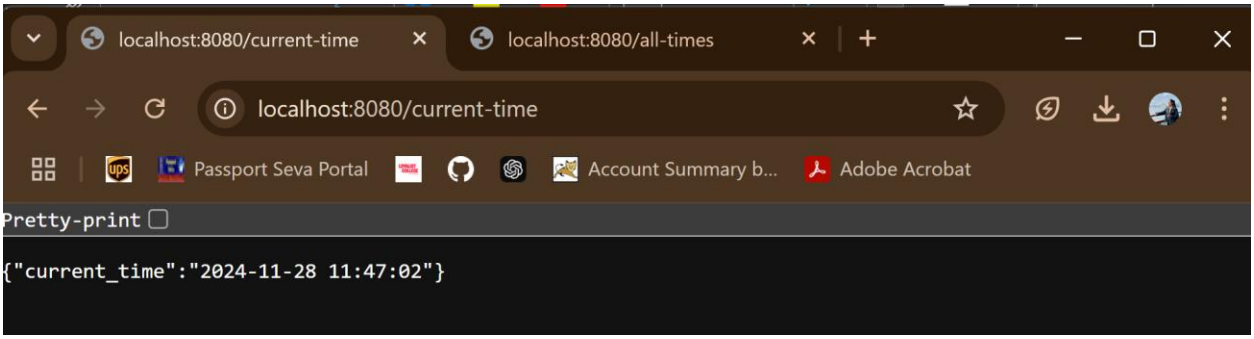
Create an API endpoint /current-time that returns the current time in Toronto. **(Created Alltimes as well)**

```
    // Set up the router
    r := mux.NewRouter()
    r.HandleFunc("/current-time", getCurrentTime).Methods("GET")
    r.HandleFunc("/all-times", getAllTimes).Methods("GET")

    log.Println("Starting server on port 8080...")
    http.ListenAndServe(":8080", r)
}
```

3. **Time Zone Conversion:**

Use Go's time package to handle the time zone conversion to Toronto's local time.

```
    // Get current time in Toronto timezone
    currentTime := time.Now().In(loc)
    response := Response{CurrentTime: currentTime.Format("2006-01-02 15:04:05")}
```



4. **Database Connection:**

Connect to your MySQL database from your Go application.

```go
var db *sql.DB

func main() {
    // Initialize the database connection
    var err error
    dsn := "root:Jashneek@14@tcp(127.0.0.1:3306)/toronto_time" // Replace with your MySQL credentials
    db, err = sql.Open("mysql", dsn)
    if err != nil {
        log.Fatalf("Error connecting to database: %v", err)
    }
    defer db.Close()

    if err := db.Ping(); err != nil {
        log.Fatalf("Database is unreachable: %v", err)
    }

    log.Println("Connected to the database successfully.")
```

On each API call, insert the current time into the time_log table.

```
mysql> SELECT * FROM time_log;
+----+---------------------+
| id | timestamp           |
+----+---------------------+
|  1 | 2024-11-28 16:17:20 |
|  2 | 2024-11-28 16:19:42 |
|  3 | 2024-11-28 16:19:43 |
|  4 | 2024-11-28 16:30:37 |
|  5 | 2024-11-28 11:43:55 |
|  6 | 2024-11-28 11:47:02 |
+----+---------------------+
6 rows in set (0.00 sec)

mysql>
```

5. **Return Time in JSON:**

Format the response from the /current-time endpoint in JSON.

```go
// Response structure for API responses
type Response struct {
    CurrentTime string `json:"current_time"`
}
```

```go
    w.Header().Set("Content-Type", "application/json")
    json.NewEncoder(w).Encode(response)
}
```

6. **Error Handling:**

Implement proper error handling for database operations and time zone conversions.

```go
    // Log the current time to the database
    _, err = db.Exec("INSERT INTO time_log (timestamp) VALUES (?)", currentTime.Format("2006-01-02 15:04:05"))
    if err != nil {
        http.Error(w, "Error logging time to database", http.StatusInternalServerError)
        log.Printf("Database error: %v", err)
        return
    }

    w.Header().Set("Content-Type", "application/json")
    json.NewEncoder(w).Encode(response)
}
```

Bonus:

Implement logging in your Go application to log events and errors.

```
mysql> USE toronto_time;
Database changed
mysql> SHOW TABLES;
+------------------------+
| Tables_in_toronto_time |
+------------------------+
| time_log               |
+------------------------+
1 row in set (0.01 sec)

mysql> SELECT * FROM time_log;
+----+---------------------+
| id | timestamp           |
+----+---------------------+
|  1 | 2024-11-28 16:17:20 |
|  2 | 2024-11-28 16:19:42 |
|  3 | 2024-11-28 16:19:43 |
|  4 | 2024-11-28 16:30:37 |
+----+---------------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM time_log;
+----+---------------------+
| id | timestamp           |
+----+---------------------+
|  1 | 2024-11-28 16:17:20 |
|  2 | 2024-11-28 16:19:42 |
|  3 | 2024-11-28 16:19:43 |
|  4 | 2024-11-28 16:30:37 |
|  5 | 2024-11-28 11:43:55 |
+----+---------------------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM time_log;
+----+---------------------+
| id | timestamp           |
+----+---------------------+
|  1 | 2024-11-28 16:17:20 |
|  2 | 2024-11-28 16:19:42 |
|  3 | 2024-11-28 16:19:43 |
|  4 | 2024-11-28 16:30:37 |
|  5 | 2024-11-28 11:43:55 |
+----+---------------------+
5 rows in set (0.00 sec)
```

Create an additional endpoint to retrieve all logged times from the database.

```go
// getAllTimes fetches all logged times from the database
func getAllTimes(w http.ResponseWriter, r *http.Request) {
    rows, err := db.Query("SELECT id, timestamp FROM time_log")
    if err != nil {
        http.Error(w, "Error fetching times from database", http.StatusInternalServerError)
        log.Printf("Database query error: %v", err)
        return
    }
    defer rows.Close()

    var times []Response
    for rows.Next() {
        var id int
        var timestamp string // Use string to scan the raw timestamp value
        if err := rows.Scan(&id, &timestamp); err != nil {
            http.Error(w, "Error reading data", http.StatusInternalServerError)
            log.Printf("Row scan error: %v", err)
            return
        }

        times = append(times, Response{CurrentTime: timestamp})
    }

    w.Header().Set("Content-Type", "application/json")
    json.NewEncoder(w).Encode(times)
}

```