LOYALIST COLLEGE
IN TORONTO

Go API for Current Toronto Time with MySQL Database Logging

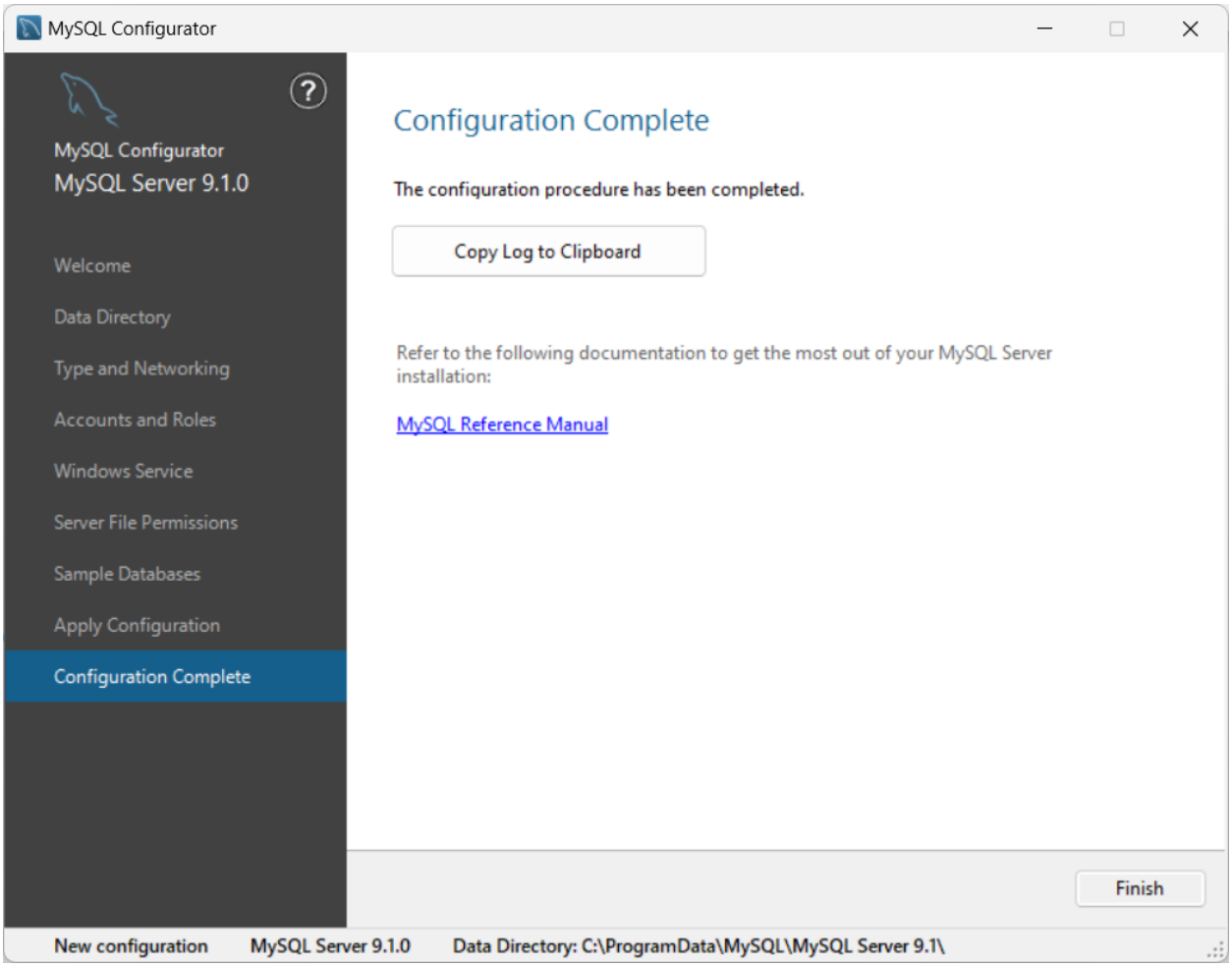
Student Name: Chandanpreet Singh
Student ID: 500221058
Course Code: WINP2000
Instructor Name: Maziar Sojoudian

Building a Go API for Current Toronto Time with MySQL Database Logging.

Tasks:

1. **Set Up MySQL Database:**

   Installing and configuring MySQL.




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

Creating Database ‘go_api’

```
mysql> create database go_api;
Query OK, 1 row affected (0.01 sec)

mysql> use go_api;
Database changed
```

Creating a table named time_log with two fields: id (primary key) and timestamp.

```
mysql> create table time_log (
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> timestamp DATETIME NOT NULL
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> |
```

```
mysql> DESCRIBE time_log;
+-----------+----------+------+-----+---------+----------------+
| Field     | Type     | Null | Key | Default | Extra          |
+-----------+----------+------+-----+---------+----------------+
| id        | int      | NO   | PRI | NULL    | auto_increment |
| timestamp | datetime | NO   |     | NULL    |                |
+-----------+----------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
```

2. API Development:

Write a Go application with a web server and creating an API endpoint /current time that returns the current time in Toronto.

```go
package main

import (
    "encoding/json"
    "log"
    "net/http"
    "time"
)

type TimeResponse struct {
    CurrentTime string `json:"current_time"`
}

func main() {
    http.HandleFunc("/current-time", func(w http.ResponseWriter, r *http.Request) {
        // Load the Toronto timezone
        loc, err := time.LoadLocation("America/Toronto")
        if err != nil {
            http.Error(w, "Failed to load timezone", http.StatusInternalServerError)
            log.Printf("Error loading timezone: %v", err)
            return
        }

        // Get the current time in Toronto and format it
        currentTime := time.Now().In(loc).Format("2006-01-02 15:04:05")

        // Create a response with the formatted time
        response := TimeResponse{CurrentTime: currentTime}

        // Set the content type to JSON and send the response
        w.Header().Set("Content-Type", "application/json")
        if err := json.NewEncoder(w).Encode(response); err != nil {
            http.Error(w, "Failed to encode JSON", http.StatusInternalServerError)
            log.Printf("Error encoding JSON: %v", err)
        }
    })

    // Start the web server
    log.Println("Server is running on http://localhost:8080")
    log.Fatal(http.ListenAndServe(":8080", nil))
}
```

3. Time Zone Conversion:

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

```go
// Get the current time in Toronto and format it
currentTime := time.Now().In(loc).Format("2006-01-02 15:04:05")
```

```
{"current_time":"2024-11-28 11:30:09"}
```

4. Database Connection:

Connect to your MySQL database from your Go application.

```go
func init() {
    // MySQL credentials
    dsn := "root:#Mysql@123@tcp(localhost:3306)/go_api"
    var err error
    // Open a connection to the database
    db, err = sql.Open("mysql", dsn)
    if err != nil {
        log.Fatal("Error connecting to the database: ", err)
    }
    // Check if the connection is successful
    if err := db.Ping(); err != nil {
        log.Fatal("Error pinging the database: ", err)
    }
}
```

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

```go
    // Insert the current time into the time_log table
    _, err = db.Exec("INSERT INTO time_log (timestamp) VALUES (?)", currentTime)
    if err != nil {
        http.Error(w, "Failed to insert time into database", http.StatusInternalServerError)
        log.Printf("Error inserting time into database: %v", err)
        return
    }
```

5. Return Time in JSON:

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

```go
type TimeResponse struct {
    CurrentTime string `json:"current_time"`
}
```

```go
    // Create a response with the formatted time
    response := TimeResponse{CurrentTime: currentTime}

    // Set the content type to JSON and send the response
    w.Header().Set("Content-Type", "application/json")
    if err := json.NewEncoder(w).Encode(response); err != nil {
        http.Error(w, "Failed to encode JSON", http.StatusInternalServerError)
        log.Printf("Error encoding JSON: %v", err)
    }
}
```

6. Error Handling:

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

```go
func logCurrentTime() error {
    // Get the current time for logging
    currentTime := time.Now().Format("2006-01-02 15:04:05")

    // Insert the current time into the time_log table
    _, err := db.Exec("INSERT INTO time_log (logged_time) VALUES (?)", currentTime)
    if err != nil {
        return fmt.Errorf("error inserting time into database: %v", err)
    }

    return nil
}
```

Bonus Challenges

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

```
mysql> use go_api;
Database changed
mysql> select * from time_log;
+----+---------------------+
| id | timestamp           |
+----+---------------------+
|  1 | 2024-11-28 11:30:09 |
+----+---------------------+
1 row in set (0.00 sec)
```

```
mysql> select * from time_log;
+----+---------------------+
| id | timestamp           |
+----+---------------------+
|  1 | 2024-11-28 11:30:09 |
|  2 | 2024-11-28 11:33:21 |
+----+---------------------+
2 rows in set (0.00 sec)
```

```
mysql> select * from time_log;
+----+---------------------+
| id | timestamp           |
+----+---------------------+
|  1 | 2024-11-28 11:30:09 |
|  2 | 2024-11-28 11:33:21 |
|  3 | 2024-11-28 11:35:52 |
|  4 | 2024-11-28 11:35:54 |
+----+---------------------+
4 rows in set (0.00 sec)
```

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

```go
// Fetch all logged times from the database
func getAllLoggedTimes() ([]LoggedTime, error) {
    var times []LoggedTime

    // Query to fetch all logged times
    rows, err := db.Query("SELECT id, logged_time FROM time_log")
    if err != nil {
        return nil, fmt.Errorf("error querying database: %v", err)
    }
    defer rows.Close()
```

```go
// New endpoint to retrieve all logged times
http.HandleFunc("/all-times", func(w http.ResponseWriter, r *http.Request) {
    // Fetch all logged times from the database
    loggedTimes, err := getAllLoggedTimes()
    if err != nil {
        http.Error(w, "Failed to fetch logged times", http.StatusInternalServerError)
        log.Printf("Error fetching logged times: %v", err)
        return
    }
```