Final Sprint Week

Mar 28 – Apr 14, 2024

- **Projects 1, 2, and 4 are to be completed in your Robot Groups – Project 3 – JavaScript will be done individually. Submission for the group projects (1, 2) should be made by one person on behalf of the group. Project 4 – Obstacle Course – will be completed on the Robot Competition Day.**

- **All projects are due by Sunday, April 14, 2024. Each project is to be attached to the assignment portal as in the past. The Robot Competition Day will be Friday, April 12, 2024. As projects are completed, post them to the portal so the instructors can get a start on the marking.**

- **There will be no classes during the final sprint. I will be at the school from 1 – 4 pm each day to allow you to work with your robots and for questions.**

- **I expect that each group will set up a Trello Board and have daily stand ups.**

Project 1 – ERD and Report Design

The context of this problem is that a notional client, Harry Mackie, and his brothers Archie and Brian, have decided to open a small taxi stand (HAB Taxi Services – what a great name) in their town and asked for your help with the technology. All the brothers are mechanics so all maintenance will be performed in-house. They have a large garage between their homes (they are neighbors too) with a small build-out on the side that will be used as an office and command center. They have 4 cars that they are using in the business and rent out to drivers, while other employees use their own vehicles and pay stand fees.

A text file called Defaults.dat includes some values that will be used by the system. Values include the Next transaction number (143), the Next driver number (1922), the Monthly stand fee ($175.00) for drivers with their own car, the Daily rental fee ($60.00) and the Weekly rental fee ($300.00) for drivers who rent one of the company cars, and the HST rate (15%).

To keep track of drivers, set up an Employees table to store their driver number, name, address and phone number, drivers license number and expiry date, insurance policy company and policy number, a field to represent if they have their own car, and a field for their balance due – based on stand fees and rentals. Note that Harry, Archie, and Brian will also be set up as employees in this table.

A table is required to record any revenues from the company in a Revenues table. This table will include a Transaction ID, the date of the transaction, a brief description of the transaction, the driver number that generated the revenue, the amount of the transaction, the HST and the Total. NOTE: On the first day of each month stand fees are automatically charged to the drivers when the program is turned on, the balance due is updated in the employee table, and the revenue will be recorded in the revenue table.

Another table is required to record company Expenses for supplies and repairs to the company cars in an Expenses table. This table will include an Invoice number, an invoice date, the driver number that generated the expense, the item number, description, Cost, the Quantity, and the item total (Cost * Quantity) – NOTE: there could be multiple items entered on the invoice. Finally, the Subtotal, the HST and the Total will be added.

As drivers rent one of the company cars, we need to record the information to a Rentals table. The table will include a Rental ID, the driver number, the start date of the rental, the car number (1, 2, 3, or 4), whether the rental is for a day (from noon to noon the next day) or a week, the number of days (if they pick a week, it will appear as 7), the rental cost, the HST and the total. NOTE: this will update the balance due in the employees table above and add a record to the Revenue table as well.

As a driver makes a payment on their balance due, record the information to a Payment table. This will include a Payment ID, the driver number, the date of the payment, the amount of the payment, the reason for the payment, and the payment method (Cash, Debit, or Visa).

Create an ER Diagram based on the case study presented. Add one other feature to the ERD. It could be some new fields, a new table, or a combination of both. Add a description of what you did at the end of your ERD.

Design each of the following Reports / Receipts that are generated by the system. Place each of the designs on separate pages (Use Ctrl + Enter to move to the next page) after the ERD.

- As a driver rents a company car or makes a payment on their balance, the system must print a receipt. Make sure it clearly indicates the reason for generating the receipt.

  Payment Table (ERD)/ #5 in Python

- A Profit Listing is a report that is generated based on a Start and End Date – could be a single day, a month, or the entire year. Include all revenues and expenses and a calculated Profit (Loss) which is the total revenues less the total expenses.

  Revenue & Expenses Table (ERD)/ #6 in Python

- A Driver Financial Listing can be generated for any driver based on a Start and End Date. It includes all entries from the Revenue table for the driver and dates specified. Include relevant totals at the end of the report.

  Employee & Revenue Table (ERD)/ #7 in Python

Project 2 – Some Python Code – I know you LOVE this.

The program will be set up to work from the following menu. There are enough options that someone can set up the menu and everyone can take a part or two from the menu options.

HAB Taxi Services
Company Services System

1. Enter a New Employee (driver).
2. Enter Company Revenues.
3. Enter Company Expenses.
4. Track Car Rentals.
5. Record Employee Payment.
6. Print Company Profit Listing.
7. Print Driver Financial Listing.
8. Your report – add description here.
9. Quit Program.

Enter choice (1-9):

Create the defaults file as described in the case study. At the beginning of this program, read the values from the defaults file to proper variable. Write the values back at the end of the program.

Remember how I said programmers hate the word "Automatically"? On the first day of each month stand fees are automatically charged to the drivers with their own car when the program is turned on, and the revenue will be recorded in the revenue table. A sample record in the revenue table would appear as follows. It was mentioned that this would also update the balance due in the employees table – Place a comment indicating that the Balance Due would be updated now – this update does not need to be coded – it is quite tricky with text files – BONUS mark if you can figure it out.

143, 2022-04-01, Monthly Stand Fees, 1918, 175.00, 26.25, 201.25

From the menu, complete the code for **Option 1, OR 2, OR 3** – Did you notice the word **OR** - just one is required to be coded. Based on your ERD, design, and code a text-based interface. Add some calculated fields and display the results once the inputs are complete and write the values to the appropriate files.

The data files for Options 1, 2 and 3 (including the option from the previous step) should be created. Just create the data files for the other options not created in the previous step. Add up to 5 - 6 records in the Employee table, and 10 - 20 records in the revenue and expenses tables.

Write the code for **Options 6 OR 7** based on the designs you created in the first project. At the end of the report, create some analytics data that can be used for decision making.

Design one other report based on the information in the ERD that you feel would be useful. This is to be added as option 8 in the menu.

Project 3 - JavaScript FINAL SPRINT

Overview: Given this course sections introductory focus on the JavaScript syntax this final sprint module will be exercising the basics of JavaScript. In this sprint you will create a JSON file, read the file, and then write a few functions to display the contents of the file to the browsers console.

The JavaScript final project will be done individually.

For a sample solution visit the following GitHub repository and review the code. DO NOT copy this code but use it as reference. prawstho/IntroToWebDevSprint (github.com)

Project: The project is focused on creating, reading, and displaying a JSON file you create yourself. The JSON file should contain a variety of data types. The display of the data in the JSON file should be done using functions within a .js file referenced from a html file. The project could be built using the following steps. Don't hesitate to use your own approach to developing this project if you like, please make sure you are doing this work yourself. It is ok to ask for help from your classmates, friends, teaching assistants, or lecturer.

Step 1: Create the html and JavaScript files in the visual studio environment. Link the html file to the JavaScript file using the html <SCRIPT> tag. Test this works and you are displaying information in the script file to the browser console using the console.log() statement.

Step 2: Create a simple JSON file with more than five records in the file.

Step 3: Read the file using the Fetch API available in JavaScript.

Step 4: Using a forEach loop iterate over all the records available from the JSON file and display one of the key-value pairs of data to test that your reading of the JSON file is working.

Step 5: write three functions to return strings of data describing the contents of the JSON file.

Write the results of reading the JSON file to the browser window as HTML. The JSON data should also be written to the browser console.

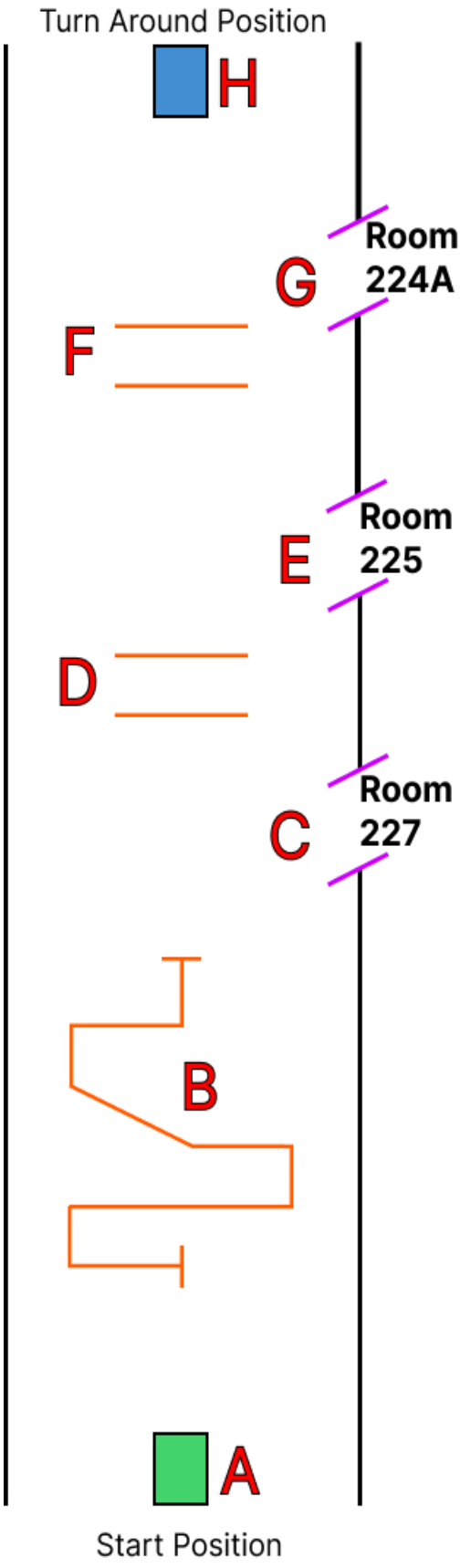
Learning Outcomes:

1. Proven ability to create a simple JSON file.
2. Proven ability to create and read a JavaScript file from an HTML page.
3. Proven ability to set up a hosting HTML page which executes an embedded <SCRIPT> file.
4. Proven ability to read a JSON file from disk and display its contents.
5. Proven ability to iterate over the records in a JSON file to the console.log.
6. Proven ability to write simple JavaScript functions that format and emit record contents read from a JSON file.

Project Deliverables:

1. Upload all the project files to your personal GitHub repository.
2. Submit the GitHub URL to the assignment portal for marking.

Project 4 – Robot Obstacle Course



You job is to position your robot in at each performance point (A – H) and perform the following. Position the robot to the start position (A). Note that at each horizontal line, it is a reset point – you can sleep for 5 seconds to allow you to reset the position of the robot.

Each door (227, 225 and 224A) will be assigned a value of 1, 2 or 3 indicating the room type. If it is a 1, there is a fire – find the marker and shoot it. A type 2 is poisonous – skip the room and do not enter. Finally, a type of 3 has a person – locate the person and take him to safety (Back to A – then return to the door)

At position B – follow the line to bring the robot to the top horizontal line. This is a Reset point.

On the way up the hall, bypass position D – you can, however, use this as a Reset Point.

At position F, there will be a marker with 1, 2, or 3 on the wall. If the marker is a 1, do something with the Chassis and Gimbal, if the marker is a 2 do something with the LED lights, if it is a 3, do both the Chassis and Gimbal and the LED lights. Again, this can change each run. This is also a reset point.

Position H is the turn around point. This is also a reset point.

On the way back to the start, at Position D, research something you can do with the robot and implement it here. This will also be the final reset point on the way back to the Start Position (A) for the end of the course. You are not required to do the zigzag again – just go straight through.

The Robot competition will be held on Friday, April 12. There will be a first run at 10:00 am, then pizza for lunch supplied by the school, and a second run for those who need it at 1:00 pm.

Obstacle Course Notes

The room types can be changed before you run the course for many different alternatives. The next line is to set the robot to free mode so the chassis and gimbal work separately.

```
def start():

    Room1Type = 1  # Fire
    Room2Type = 2  # Skip
    Room3Type = 3  # Person

    robot_ctrl.set_mode(rm_define.robot_mode_free)
```

The next lines will move through the zigzag and to the first room – use your own measurements you took. As you reach any door, turn 90 degrees so the robot is pointing to the door. Then based on the Room Type, take the appropriate action. When you leave a room, again turn 90 degrees so you are pointing back down the main hall.

Once you are front of the door, you want to determine what to do at the door you set an if statement to check for each type. If it is a 1 (Fire) – you go into the room and shoot the target. If it is a 2 (Poison) - you bypass the room. Lastly, if it is a 3 (Person) – you find the person and bring them back to the start to safety. Note that the markers (F) in each room will not change position – the only thing that will change is the room type.

```
if Room1Type == 1:
        # Enter the room, find the marker, and shoot once.
elif Room1Type == 2:
        # Skip the room.  Turn back 90 degrees and continue.
elif Room1Type == 3:
        # Enter the room, find the marker, find the person, and bring the
        # person back to the Start Position for safety.  Return to the front
        # of the same door, and continue
```

Note that this process will be repeated as you get to the front of each door. Remember that any room could be a 1, 2 or 3 for any run.

Suggested processes:

1. Take measurements in the main hall – stop at all points – turn 90 degrees when you get to a room so that you are pointing at it. Test the main hall code.
2. Take measurements from the hall going into each room to the marker. Test moving inside and position the robot approximately 3 feet in front of the marker.
3. Write the code to scan for the marker or person – test each room for all room types.