The following take-home project measures the candidate's ability in all the spectrums of a full stack position: Front-End, Back-end and database design/query.

Upon completion, please publish the source-code in a GitHub repo and send the link. Any instructions to run/setup the webapp need to be included in a readme.md file.

Any questions please feel free to email me.

Constraints:

- Backend technology:
  - WebApp: .net core
    - The pages must be server-side-rendered. You can utilize MVC or razor pages type projects.
    - Preferably, do not use Entity framework.
  - Database: SQL server
    - Don't forget to include the database script (create table...) in your GitHub repo.
- Frontend:
  - **plain** JavaScript and/or typescript.
    - Any JavaScript ES version can be used as long it's supported by the latest production version of the chromium engine.
    - **No** Front-End JavaScript frameworks
      - You can use javascript's Fetch API for ajax requests.
  - any CSS library can be used (bootstrap, tailwind…)

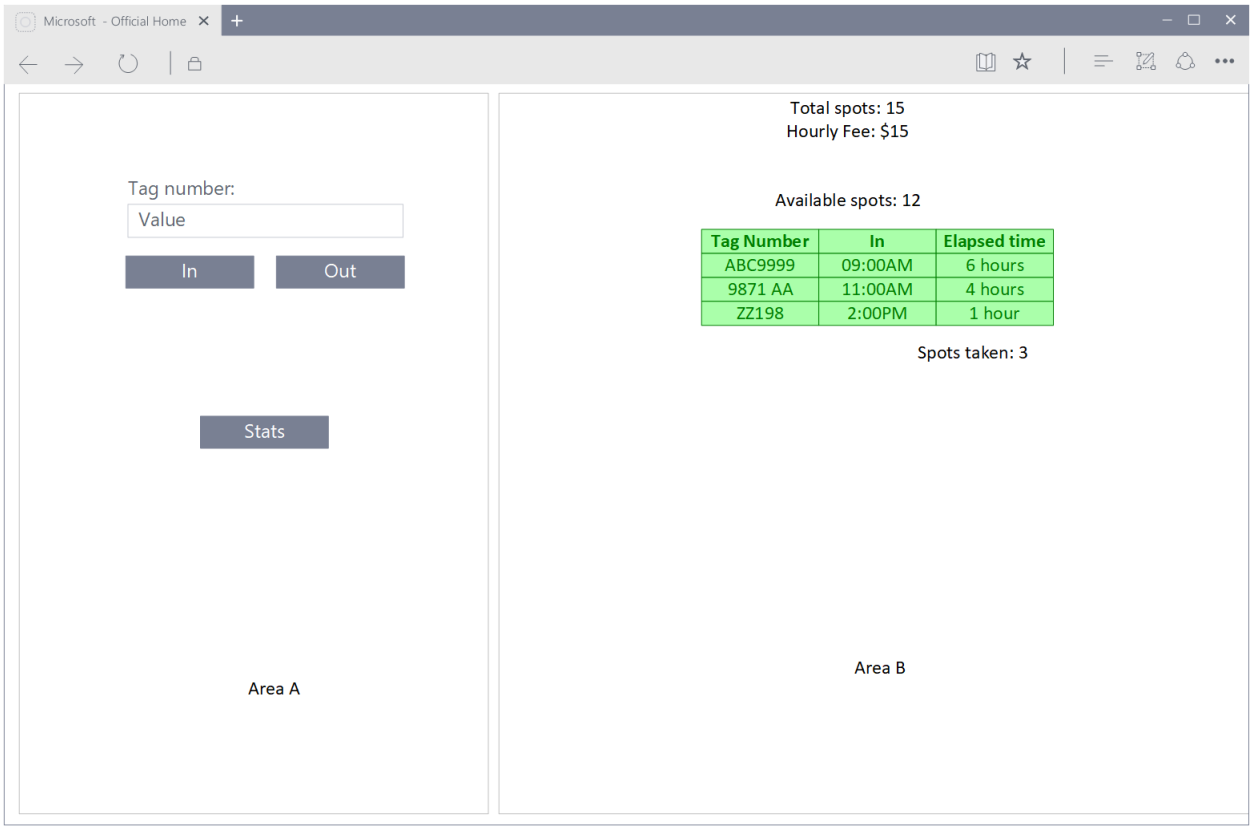
Nice to have if time permits:

- Tests (App layer using any test framework like XUnit and Front-end layer using Selenium or Playwright)

Project: Parking lot management

A web app that manages a parking lot and the flow of cars coming in/out. The web app view is split into 2 areas. "Area A" has the input controls the operator uses to check-in/out a car. "Area B" has a snapshot of the parking garage at that moment. It shows all cars currently parked. Whenever a new car arrives or leaves, "Area B" needs to be updated to show only the cars currently parked.

The web app is an enhanced-MPA (multiple page application) where page transitions and refreshes are done via ajax requests.



The total number of spots and hourly fee need to be defined in the appsettings.json file.

Cars coming in:

Whenever a new car arrives, the operator types the tag number and clicks 'In'. The app needs to validate the following.

* Any spots available?

* Is the car already in the parking lot?

Whenever the validation fails, the operator needs to be given an error message.

Whenever the validation passes, 'Area B' needs to be updated via ajax request. (no full page reload)

Cars coming out:

Whenever a car leaves the parking lot, the operator types the tag number and clicks 'Out'. The app needs to validate the following.

* Is the car registered in the parking lot

Whenever the validation fails, the operator needs to be given an error message.

Whenever the validation passes:

* The operator needs to be given the total amount that needs to be charged.

** The total amount is calculated by verifying how long the car stayed in the garage and applying the hourly fee.

E.g.:

hourly fee: $15 (defined in the appsettings.json)

Example A: 3 hours => total $45

Example B: 2 minutes => total $15

Example C: 60 minutes => total $15

Example D: 61 minutes => total $30

* Upon completion, 'Area B' needs to be updated via ajax request (no full page reload)

Stats

It opens a modal with the following information.

- Number of spots available as of now
- Today's revenue as of now
- Average number of cars per day (for the past 30 days)
- Average revenue per day (for the past 30 days)