ParkSmart

Team 2: Boolean Bros

Team Details

Nicholas Burlakov

Austin Harrison

Dillon Pikulik

Caleb Stewart

Tyler Woody

Project Overview

- ParkSmart: Where parking meets precision, making every spot count
- Our website will revolutionize parking by allowing you to:
  - View parking spot availability in real time
  - See how public events/holidays affect parking fees
  - Compare rates based on the time and day
  - Use QR codes or vehicle recognition to detect if spot is available.

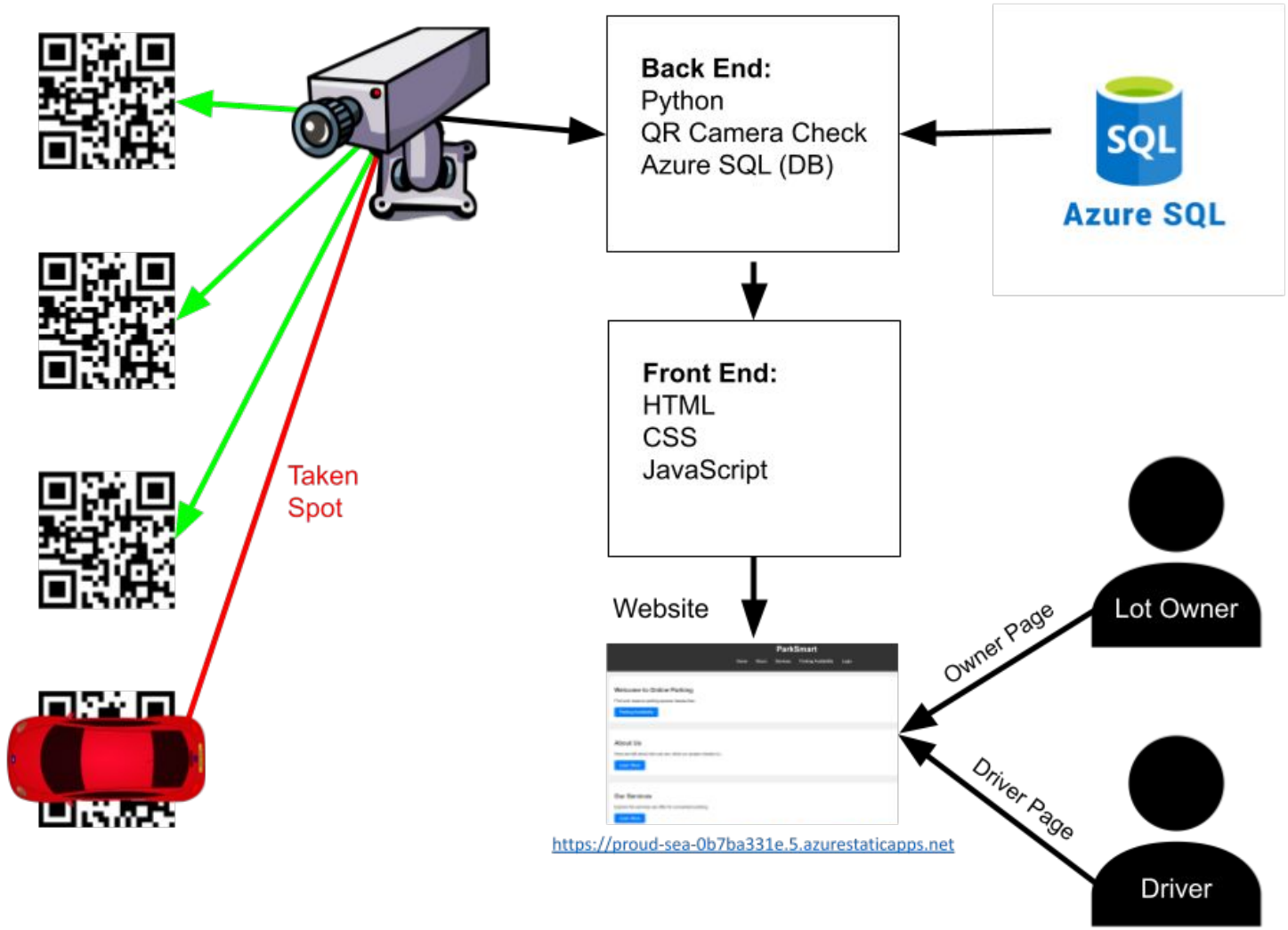
Requirements List

- R1: Constantly updating view of available parking spaces, with respective rates.
- R2: Users with parking lot owner privileges can see how full a parking lot is, and to check how much time each car has paid.
- R3: Login system to allow users to record their assets, such as cars and/or favorite parking lots.
- R4: Registered users will have a way to reserve spots in advance.
- R5: There will be a way for users to pay for their parking spot once they park or after they reserve their spot.

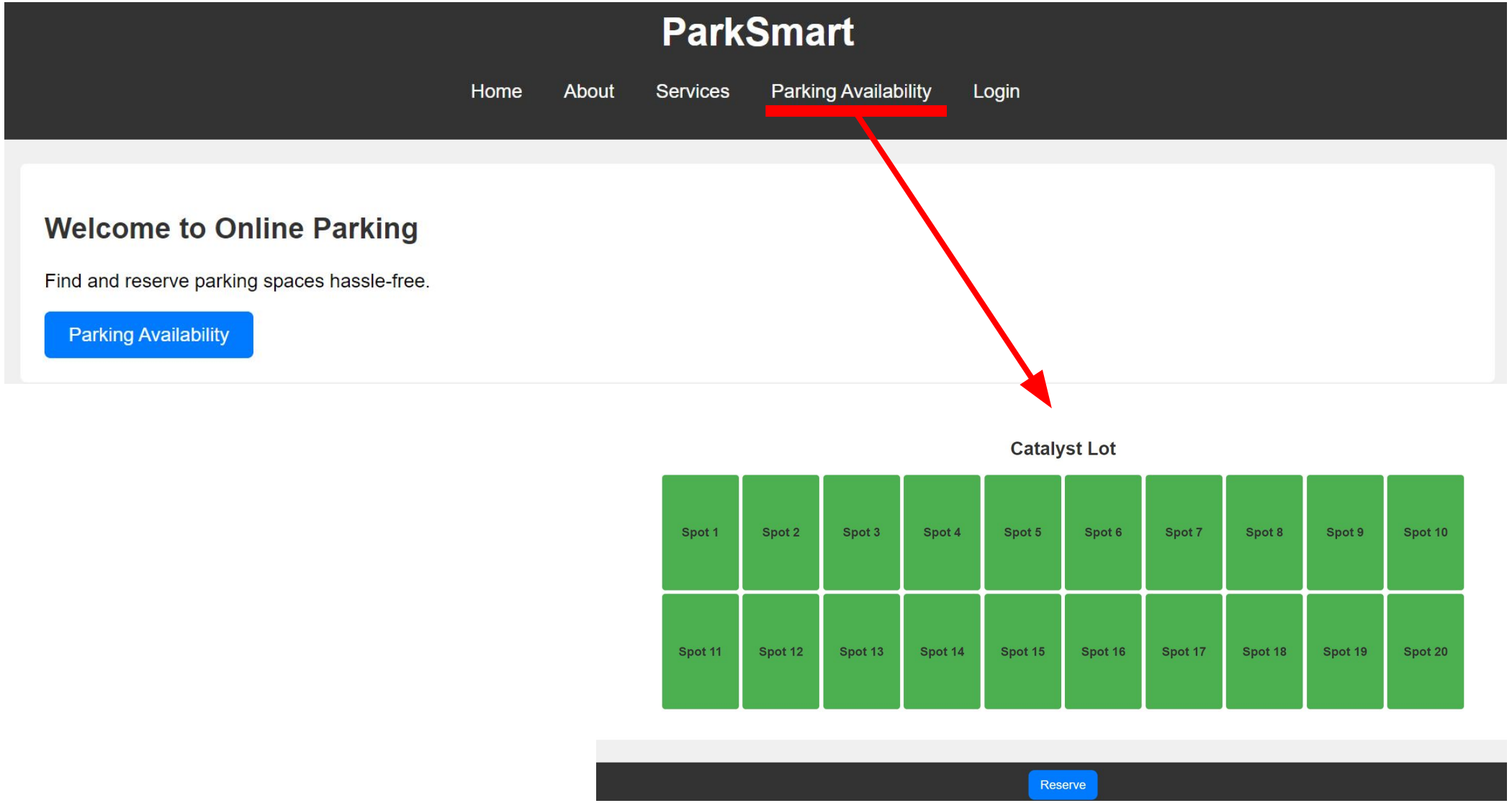
Project Solution Approach

- Our major components in our solution design?
  - Deployed to Azure as a static web app
- What are the tools, frameworks, platforms, libraries, etc. that you'll be using
  - JavaScript/HTML/CSS frontend
  - Python backend for versatile functionality
  - Azure SQL database for painless (more or less) API calls
  - QR scanner to update parking availability

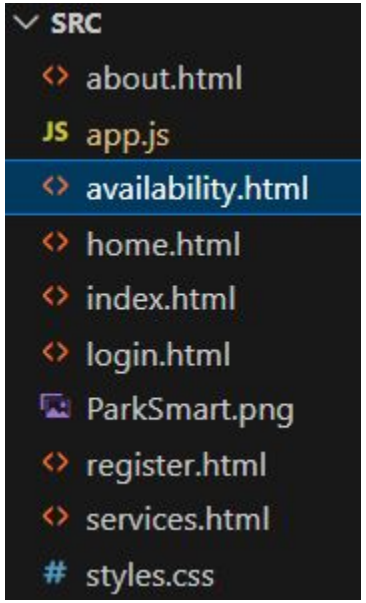
System Architecture Diagram



Screenshots (Frontend)



Screenshots (Backend)



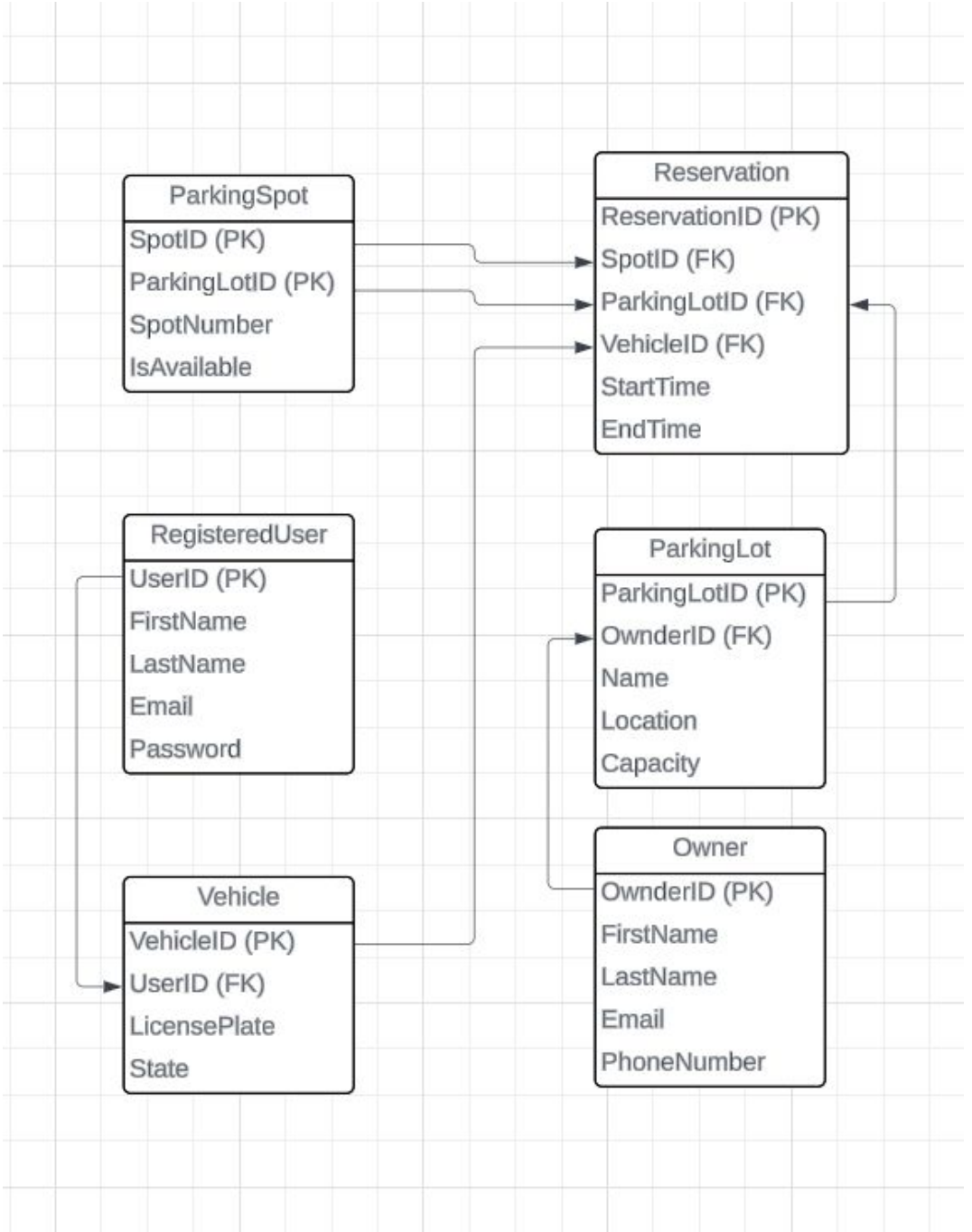
.github > workflows > azure-static-web-apps-proud-sea-0b7ba331e.yml

```yaml
name: Azure Static Web Apps CI/CD

on:
  push:
    branches:
      - main
  pull_request:
    types: [opened, synchronize, reopened, closed]
    branches:
      - main

jobs:
  build_and_deploy_job:
    if: github.event_name == 'push' || (github.event_name == 'pull_request' && github.event.action != 'closed')
    runs-on: ubuntu-latest
    name: Build and Deploy Job
    steps:
      - uses: actions/checkout@v3
        with:
          submodules: true
          lfs: false
      - name: Build And Deploy
        id: builddeploy
        uses: Azure/static-web-apps-deploy@v1
        with:
          azure_static_web_apps_api_token: ${{ secrets.AZURE_STATIC_WEB_APPS_API_TOKEN_PROUD_SEA_0B7BA331E }}
          repo_token: ${{ secrets.GITHUB_TOKEN }} # Used for Github integrations (i.e. PR comments)
          action: "upload"
          ###### Repository/Build Configurations - These values can be configured to match your app requirements. ######
          # For more information regarding Static Web App workflow configurations, please visit: https://aka.ms/swaworkflowconfig
          app_location: "./src" # App source code path
          api_location: "api" # Api source code path - optional
          output_location: "./src" # Built app content directory - optional
          ###### End of Repository/Build Configurations ######

  close_pull_request_job:
    if: github.event_name == 'pull_request' && github.event.action == 'closed'
    runs-on: ubuntu-latest
    name: Close Pull Request Job
    steps:
      - name: Close Pull Request
        id: closepullrequest
        uses: Azure/static-web-apps-deploy@v1
        with:
          azure_static_web_apps_api_token: ${{ secrets.AZURE_STATIC_WEB_APPS_API_TOKEN_PROUD_SEA_0B7BA331E }}
          action: "close"
```



Screenshots (Code)


```
Lot A Spot 1: True, Lot A Spot 2: True
Lot A Spot 1: True, Lot A Spot 2: True
Lot A Spot 1: True, Lot A Spot 2: True
```


```
Lot A Spot 1: True, Lot A Spot 2: False
Lot A Spot 1: True, Lot A Spot 2: False
```

Lightning Mcqueen is parked in Lot A Spot 2

```python
def checkForQRCode(lotName: str, numOfSpots: int, data: []):
    # Checks if any of the QR codes in lotName in range of numOfSpots are in frame
    for i in range(1, numOfSpots + 1):
        spotName = f"{lotName}_{i}"
        # Just prints the results
        print(f"Lot {lotName} Spot {i}:\t{spotName in data}")
#End checkForQRCode()
```

```python
def captureVideo():
    detector = cv2.QRCodeDetector()
    # Create video capture feed
    cap = cv2.VideoCapture(0)
    cap.set( propId: 3,  value: 640)
    cap.set( propId: 4,  value: 480)

    while True:
        # Capture each frame of video
        _, frame = cap.read()
        # Show each frame of video
        cv2.imshow( winname: 'Video', frame)

        # Detect qr code
        # data is an array with the stored QR codes decoded
        _, data, _, _ = detector.detectAndDecodeMulti(frame)

        #Check if the QR code spots are found in each frame of video
        checkForQRCode( lotName: "B",  numOfSpots: 10, data)
```

Demo

https://proud-sea-0b7ba331e.5.azurestaticapps.net

What’s Next??

- Continue learning about Azure databases.
- Populate database.
- Create database queries as built in API calls.
- Implement a realistic parking lot.
- Implement the camera/QR scanning into database and .web page
- Add functionality to the login/register page (Link it to our database)
- Create an owner page

References

- Build static web app from scratch: https://learn.microsoft.com/en-us/azure/static-web-apps/getting-started?tabs=vanilla-javascript
- How to add API/run API locally https://learn.microsoft.com/en-us/azure/static-web-apps/add-api?tabs=vanilla-javascript
- Setting up Azure SQL Database https://learn.microsoft.com/en-us/azure/static-web-apps/database-azure-sql?tabs=bash&pivots=static-web-apps-rest