Attendance Tracker Specifications

System Architecture Documentation

1. System Overview

Attendance Tracker is a web-based application that tracks attendance of a site/place. Where user can submit their attendance through scanning QR code. This allows admin to keep track and manage the attendance seamlessly.

1.1 Purpose

Digitalize attendance and replace traditional ways of keeping user attendance through hardcopy

1.2 Scope

- User must have a device with camera to scan the QR code.
- User device must be connected to the internet.
- Admin can view the attendance via the web apps with any sort of devices that is connected to the internet.
- In order to prevent fraudulent login, QR code is expired per login session. This is handled in the form of unique token (GUID)
    - Internal mutex lock is used to maintain consistency between threads request
- SignalR to perform live refresh and re-generation of QR code
    - When a single user is done with check-in or check-out, a new QR code must be generated and thus a new GUID token need to be refresh. This live updates is done by having SignalR listening to any changes in event

2. Architecture Components

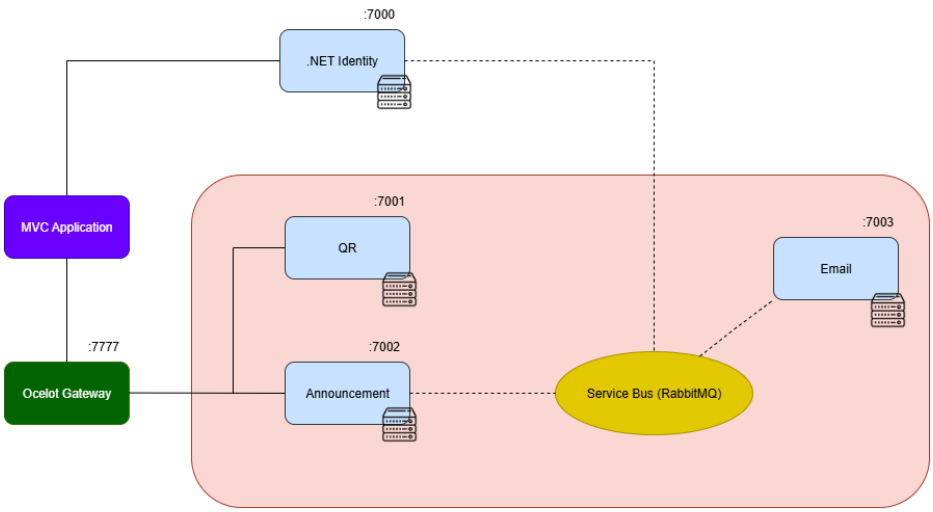
2.1 Frontend Architecture

https://www.figma.com/design/BgfDkm4683JAr7QDNk3ZmE/AttendanceTracker?node-id=1-1045&p=f&t=B8IY009NFo7L96te-0

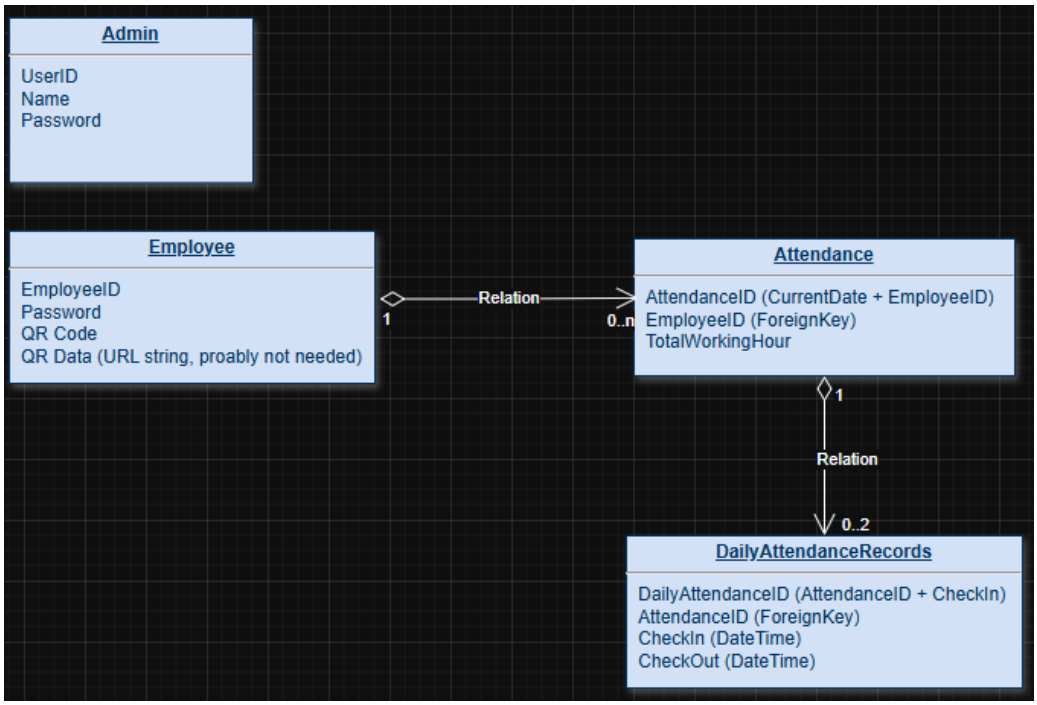
- User Interface Design
- Admin Interface Design

2.2 Backend Architecture

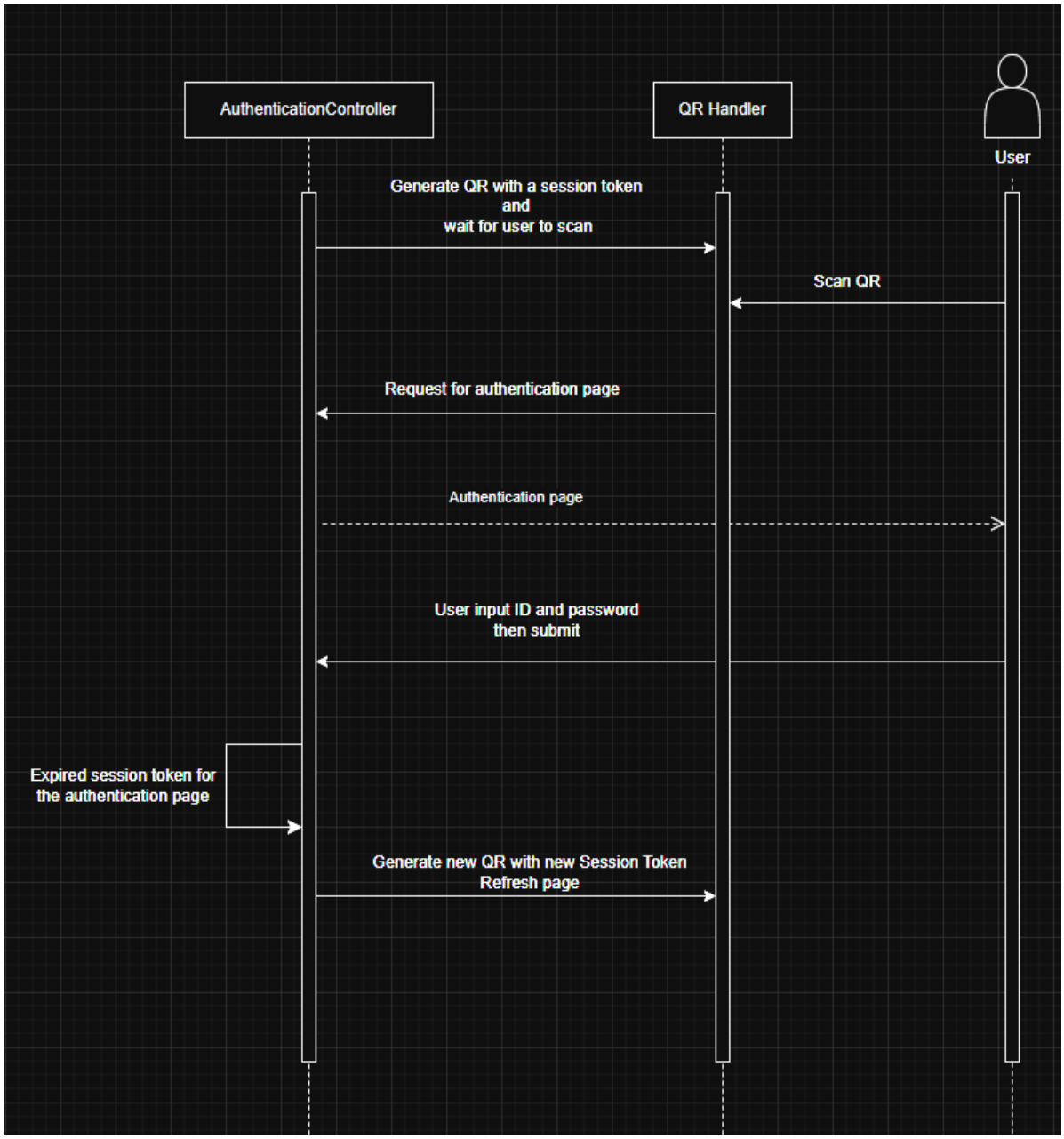
- Microservices Diagram



- Database Design



- System interaction
  - Attendance record sequence diagram



- API Architecture

2.3 Integration Points

- Third-party Services
  - Toastr to display the successful message upon
    - Recording an attendance
    - Created/Edit/Delete user

    - DataTable to display
        - Employee Table
        - Attendance Table
- External APIs
- System Interfaces

3. Technical Specifications

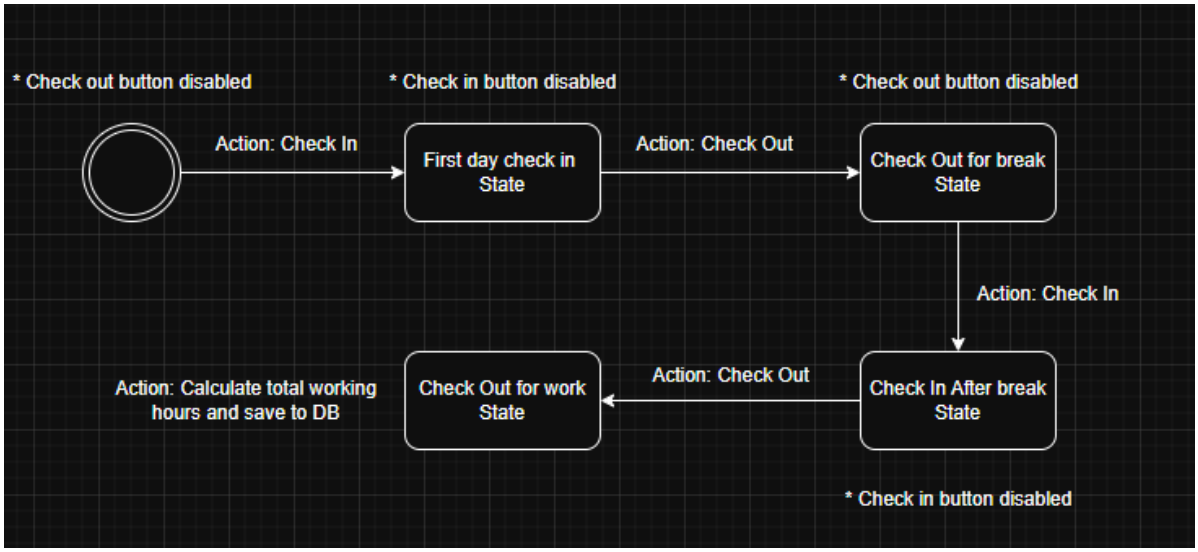
3.1 Technology Stack

- Backend
    - ASP.NET Core MVC
- Frontend
    - cshtml, ASP.NET MVC app
- Database
    - Microsoft SQL server, using Code-First Approach in ASP.NET Core
- Others
    - SignalR to add a refresh hub listener, this is used to refresh a new QR code on admin side when user logged in with the QR code.
    - External libraries:
        - DataTable is used to display the data in table form.
        - Toastr notification is used to display successful or error message on the screen.
        - Bootstrap and Bootswatch for the app theme

3.2 Data Flow

- State of data flow of check in and check out process
    - New session token is generated and embed in the QR code

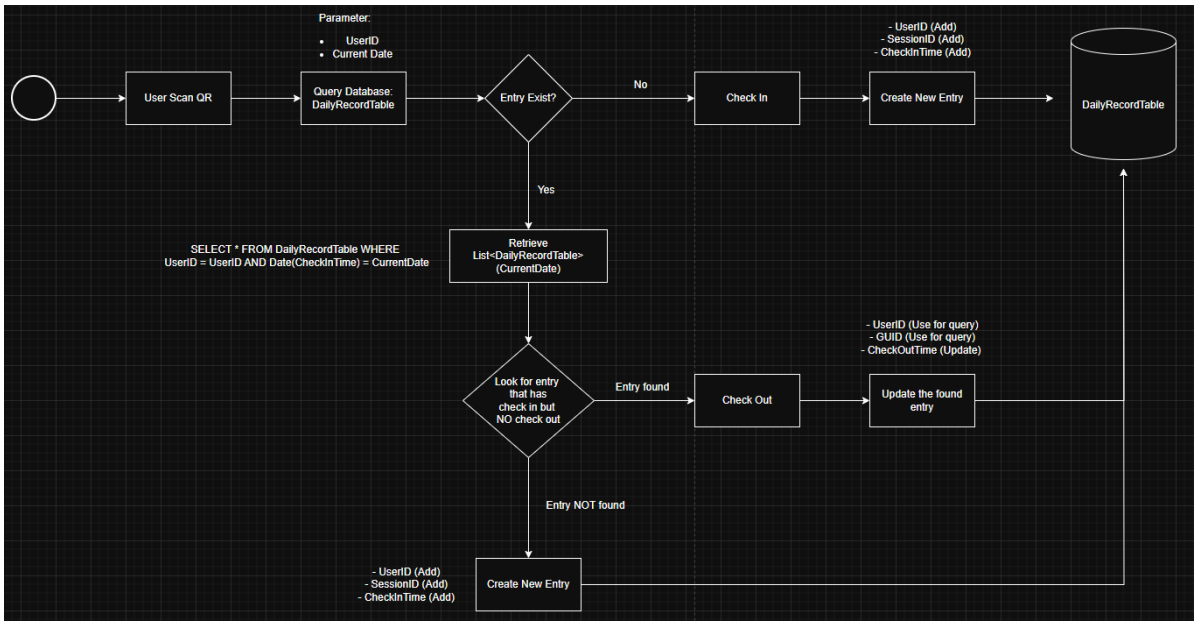
- User scan the QR code from admin page and login
- Check out button is disabled initially, user perform check in
- New QR code is refreshed in the admin web page. Existing token is expired and replace with the new one
- User scans the new QR code again and login
- Check in button is now disabled, since the user is in check out state. User perform check out



- Algorithm to determine when user should check in, or check out
  - Query the daily attendance record for current logged in user
  - If no entry exist, then this is a newly reported employee, should perform check in
  - If entry exist, iterate through all the entries from daily attendance records
    - If check in entry is found, and there are no check out, then this is a pending check out, user should perform a checkout.
    - Otherwise, this is a new check in entry.
  - NOTE: user can have multiple records of check-in and check-out on the same day

- Example: On 31/05/2025, Alice can have more than one check-in and check-out entries

Date	Check In	Check Out
31/05/2025	08:00 AM	10:00 AM
31/05/2025	11:00 AM	12:00 PM
31/05/2025	1:00 PM	05:00 PM



4. Security Architecture

- Authentication & Authorization
  - Inherited from ASP.NET Identity library
- Authorization
  - QR code to record attendance
    - Only users with admin privilege can display the QR code to be scanned by user
    - When a user is logged in by scanning the QR, the page is refresh with a new QR code
  - Employee Registration

        - Only admin
    - Modify employee attendance
        - Only admin

5. Deployment Architecture

5.1 Infrastructure

This app is deployed on premise with raspberry pi web server

5.2 Scalability

Describe scaling strategies and mechanisms.

5.3 Steps to locally host the app

Since this app requires two different devices to test:

- One with user logged in as admin to present QR code
- Another one will act as a user (usually mobile phone) to scan the QR code displayed by admin

Therefore, traditional `dotnet run` would not work here as we needed to share the resources across devices. To do that, follow the following steps:

- In your command prompt or shell terminal, retrieves the ip address
    - Windows command prompt: `ipconfig`
    - Linux terminal shell: `ifconfig`
- Copy the IPv4 address returned above
- Run following command:
    - `dotnet run --urls "<IP Address>:<PORT>"`
    - Port is a logical endpoint used to direct network traffic to the correct service running on your machine.

- In this case, i am using port 5000. This tells the machine that port 5000 is used to host the app we are running.
- Example: `dotnet run --urls " https://123.456.789.100:5000 "`

6. Monitoring and Maintenance

- Monitoring Tools
- Logging Strategy
- Backup Procedures

7. Performance Considerations

- Performance Requirements
- Optimization Strategies
- Benchmarks

8. Disaster Recovery

Document the disaster recovery plan and procedures.

> Note: This template should be customized based on your specific system requirements and organizational needs.

9. Tech Debt

Attendance record is updated everytime when admin login and view the record.

This introduced performance overhead because we need to query database everytime. To reduce this overhead, initiate a background task in the system to synchronise the database at the end of the day.

```
public IActionResult WorkingRecord()
{
    // FIXME: This is a performance overhead.
    // Use a background service to update the database automatically at the end of the day instead.
    // FIXME: End of the fixme comments.
    UpdateDatabase();
    return View();
}
```