Jaydon Ellick

20230443

2025

ADP631

Formative Assessment 1

# Phase 1:

## Research on existing municipality management systems:

Municipality management systems are software to help automate and streamline the functions of municipal governments. They help manage services, allows for engagement with their citizens and more important operations.

With them, managing and storing of their citizens information has never been easier and in turn the citizens can request services and allow the government to be in the know about problems. It is also useful for their own employees, allowing them to manage their staff’s shifts, assignments salary etc. With all this data they can improve their area and offer quality service.

Popular systems include Trimble CityWorks, Accela, CityView.

With CityWorks, you can track assets such as roads, water system and more with the help of Geographic information system, helping for mapping where exactly most issues or other services occur. They allow municipalities to track and resolve service requests from their inhabitants and customers. They offer dashboards so that their staff can make data-driven decisions.

**Citation:**

Trimble. (n/d.). Cityworks – Public asset and work management software. Trimble Asset Lifecycle Management. Retrieved from <https://assetlifecycle.trimble.com/en/products/software/cityworks>

---

Accela offers a powerful cloud-based platform for local governments. It assists them with overseeing services like planning, public works, code enforcement. It streamlines workflows, increases transparency and allows customers to apply, pay and track services online.

**Citation:**

Accela. (n.d.). Powering digital government. Accela. Retrieved April 4, 2025, from <https://www.accela.com>

---

Cityview reduces calls to staff but also has online access to submit services, make payments and other applications. With streamlined workflow and business processes, it allows municipalities to be more efficient, which would reduce costs and allow them to invest in increasing their population’s happiness. It is highly flexible and can offer numerous benefits to whatever municipalities need.

They can track and manage prime building places, view data on location-based land management activities with Cityview. Monitor and oversee enforcement events from the filing of a citizen complaint until the conclusion of the case. Allow easy animal licencing services for pet lovers, cemetery management and marriage or death registration.

**Citation:**

Municipal Software. (n.d.). CityView Suite. Retrieved April 1, 2025, from https://www.municipalsoftware.com/cityview-suite/

## System requirements and functionalities.

The municipality system would have a method citizens can register with their information, update their profile and view pasts services and reports they have done. They should be allowed to submit service requests, track the status of these and notifications when the status changes. Their emails should be unique.

Staff should be able to register, with their details and be able to be assigned to services/requests. They should be able to track their pay, performance and attendance.

Citizens should also be able to make reports on issues such as safety concerns and be able to track the progress of this with staff being able to manage these and resolves these issues.

The municipality should be able to manage the staff information, updating, adding and deleting staff if needed. Their emails should be unique.

The websites should be fast and can handle multiple concurrent users. Implement role-based access to staff for important features on the website.

The website should be fully responsive on mobile devices and have an easy-to-understand user interface. Most actions load under a second.

**Technical Requirements**

ASP.NET EMPTY

Razor Pages, CSS

## Entity Framework Core with SQL Server

xUnit with EF Core In-Memory DB for testing and in-memory to test without corrupting real data.

Real-time validation to ensure data integrity.

## Project Proposal

**System Objectives:**

The goal of the system is to improve the management and operation of municipal services, allowing for efficient management of service requests, reports, staff and citizens. These being:

A system to manage citizen information such as their contact info, names, addresses and more.

A system to keep track of and resolve service requests from their citizens.

A system to keep track of staff of the municipality.

A system to allow citizens to report emergencies and other issues which can be managed by staff.

**Scope:**

Included:

Adding, updating, deleting and view citizen information will be included.

Creating, updating and resolving service requests.

Managing staff information and adding, removing, updating it.

Report handling: Adding, removing, seeing its details and changing its status.

Not Included:

Generating reports on service requests, incident reports and other data.

Logging hasn’t been developed.

Payment integration isn’t included.

It is only a website for now.

GPS integration hasn’t been developed to pinpoint locations of reports and services.

**Features:**

Basic Citizen management.

Service Request Management.

Staff Management.

Reports Management.

**Why is the System Useful:**

It minimizes errors since data is validated, reducing common mistakes.

Efficiency is enhanced; staff can access date without needing to request them.

Transactions and other interactions are logged, with anyone being able to see the status updates making the new system more transparent for its citizens and other users.

It improved communication and lowers response time, customers can get auto updates and track progress. Staff can respond faster due to it being in real time.

# Phase 2:

## Key Entities

Table Citizens {

CitizenID integer [primary key]

FullName varchar [not null]

AddressID integer [not null]

PhoneNumber varchar [not null]

Email varchar [unique]

DateOfBirth datetime

RegistrationDate datetime [default: 'current date']

}

Citizens may submit many reports and service requests.

Table ServiceRequests {

RequestID integer [primary key]

CitizenID integer [ref: > Citizens.CitizenID]

ServiceType varchar [not null]

RequestDate datetime [default: 'current date']

Status varchar [default: 'Pending']

}

Citizens may submit many reports and service requests.

Table Staff {

StaffID integer [primary key]

FullName varchar [not null]

Position varchar [not null]

Department varchar [not null]

Email varchar [unique, not null]

PhoneNumber varchar [not null]

HireDate datetime [not null]

}

Table Reports {

ReportID integer [primary key]

CitizenID integer [ref: > Citizens.CitizenID]

ReportType varchar [not null]

Details varchar [not null]

SubmissionDate datetime [default: 'current date']

Status varchar [default: 'Under Review']

} Citizens may submit many reports and service requests.

## Relationships:

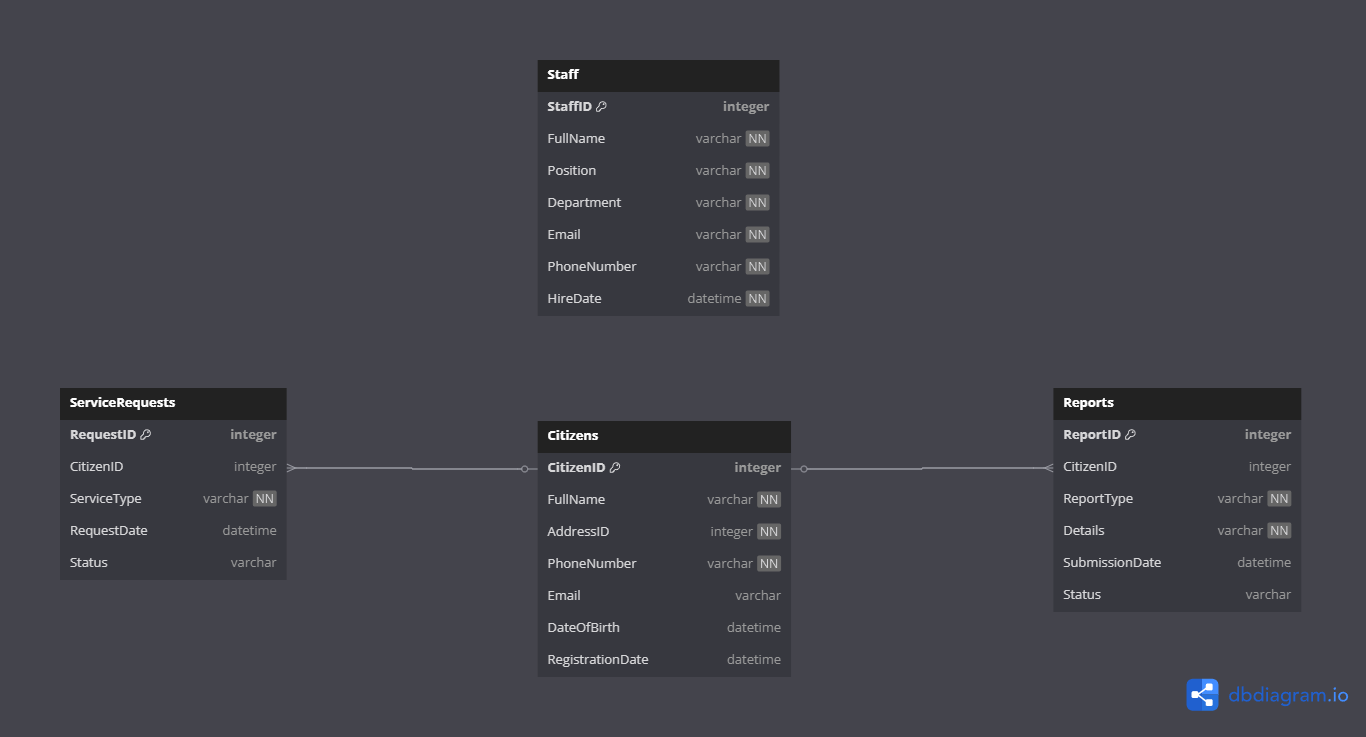
Citizens should have a one-to-many relationship with Service Request table. With the citizen being able to make multiple requests. Citizen Id being the foreign key in the request table.

Citizens should have a one-to-many relationship with the Reports table, with the citizen being able to make multiple reports. Citizen Id being the foreign key in the reports table.

## Define database schema using SQL Server:

In separate file named ‘Phase2\_DatabaseSchema.sql’.

## ERD Diagram:



# Phase 3:

## Steps To Create the Project:

Open Visual Studio, select Asp.Net Empty A screenshot of a computer program

AI-generated content may be incorrect.

Configure the location and name of the project.

A screenshot of a computer

AI-generated content may be incorrect.

Select .NET 9.0 as your framwork and leave the rest of the setting default.

A screenshot of a computer

AI-generated content may be incorrect.

Install the Nuget Packages, using the package manager console. Can be found in tools.A screenshot of a computer

AI-generated content may be incorrect.

Install-Package Microsoft.EntityFrameworkCore

Install-Package Microsoft.EntityFrameworkCore.SqlServer

Install-Package Microsoft.EntityFrameworkCore.Tools

Install-Package Microsoft.EntityFrameworkCore.Design

Install-Package Microsoft.VisualStudio.Web.CodeGeneration.Design

Install-Package Moq

Install-Package xUnit

Install-Package xunit.runner.visualstudio

Run these commands in the package manager.

A screenshot of a computer program

AI-generated content may be incorrect.

Create The models, views, data, controllers, wwroot folders. With the css, js and lib folders.

A screenshot of a computer

AI-generated content may be incorrect.

Create your models classes.

A screenshot of a computer

AI-generated content may be incorrect.

Create your controllers with the entity framework.

A screenshot of a computer

AI-generated content may be incorrect.

Select the model for it, give the controller a name and add the data context

A screenshot of a computer

AI-generated content may be incorrect.

Choose SQL Server, leave every thing in the views check marked.

A screenshot of a computer program

AI-generated content may be incorrect.

In The Package Manger console add these commands.

Add-Migrations ‘InitialCreate’

Update-Database

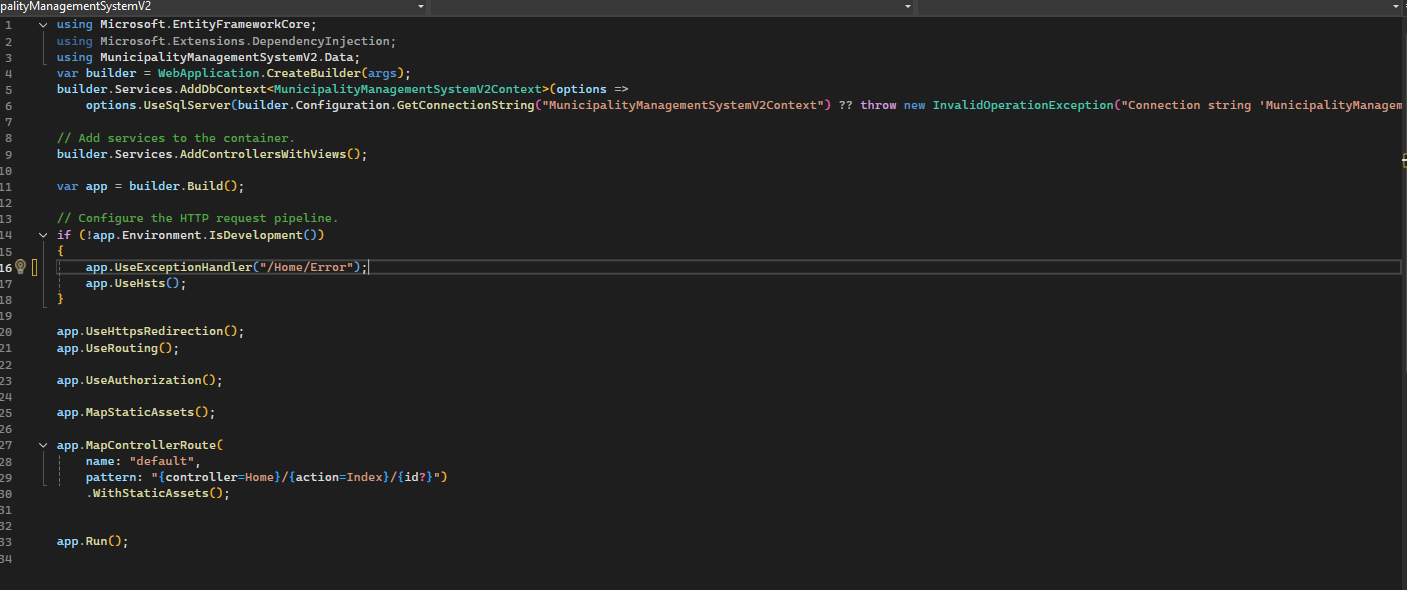
Create your layout view and viewstart cshtml files

A screenshot of a computer program

AI-generated content may be incorrect.

In the view start just specify your layout. Customize your views and create a home page with a home controller as well.

Modify the program.cs file with all the routing and specifying the connection string



Run your project and customize it.

A black screen with white text

AI-generated content may be incorrect.

## Nugget Packages Installed and Why:

Microsoft Entity Framework: To work with databases in my project.

MicrosoftEntityFrameworkCore.Design (9.0.3): To use Migrations, generate database schemas and scaffolding

Microsoft.EntityFrameworkCore.InMemory (9.0.3): Allows me to use an in-memory database to test during my unit tests.

Microsoft.EntityFrameworkCore.SqlServer (9.0.3): Allows me to use sql server as my database provider in EF Core.

Microsoft.EntityFrameworkCore.Tools (9.0.3): Essential for migration and updating the database.

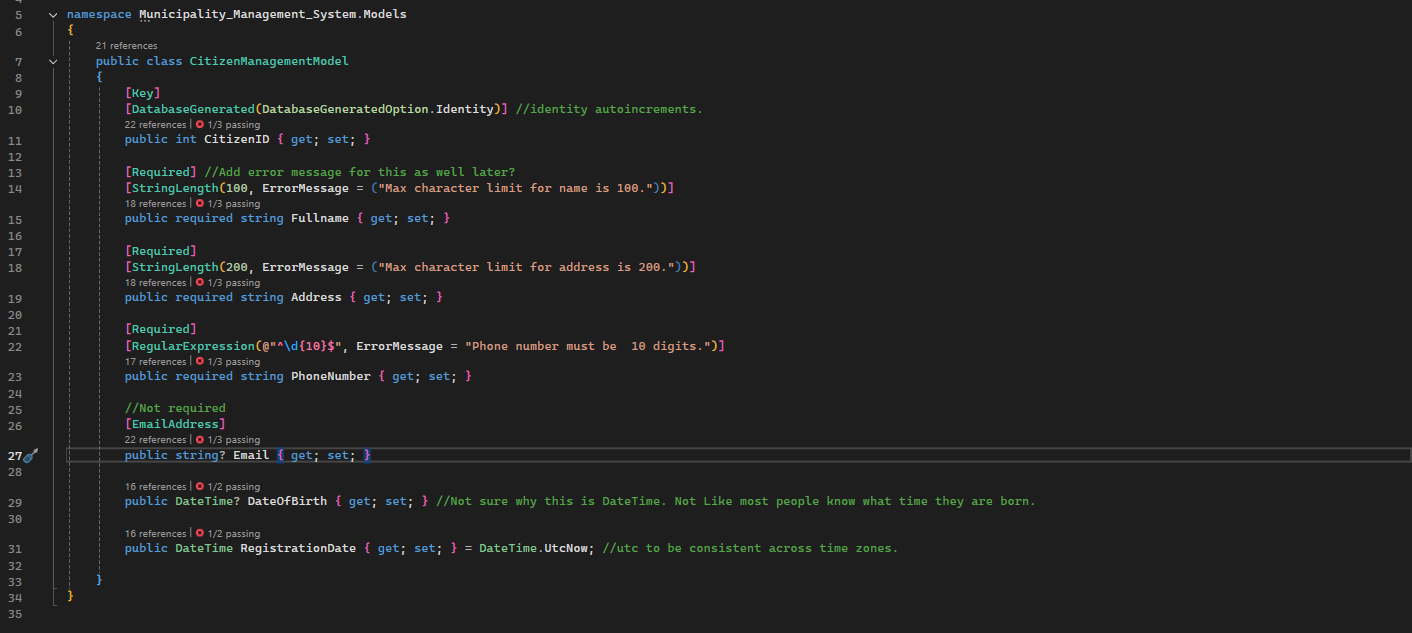
Microsoft.NET.Test.Sdk (17.13.0): It is required for using xUnit

Microsoft.VisualStudio.Web.CodeGeneration.Design (9.0.0): Helped scaffold controllers, view and models saving time.

xUnit (2.9.3): Used to write the test cases for my project.

xUnit Runner for Visual Studio (3.0.2): To run xUnit test inside Visual Studio’s Test Explorer.

## Code Snippets:



Citizen Management class, the id will increment automatically with maximum string limits and error messages for the fields. Primary key is included and the specified fields are set to required or nullable as specified in the assignment. Also default values specified.

A screenshot of a computer

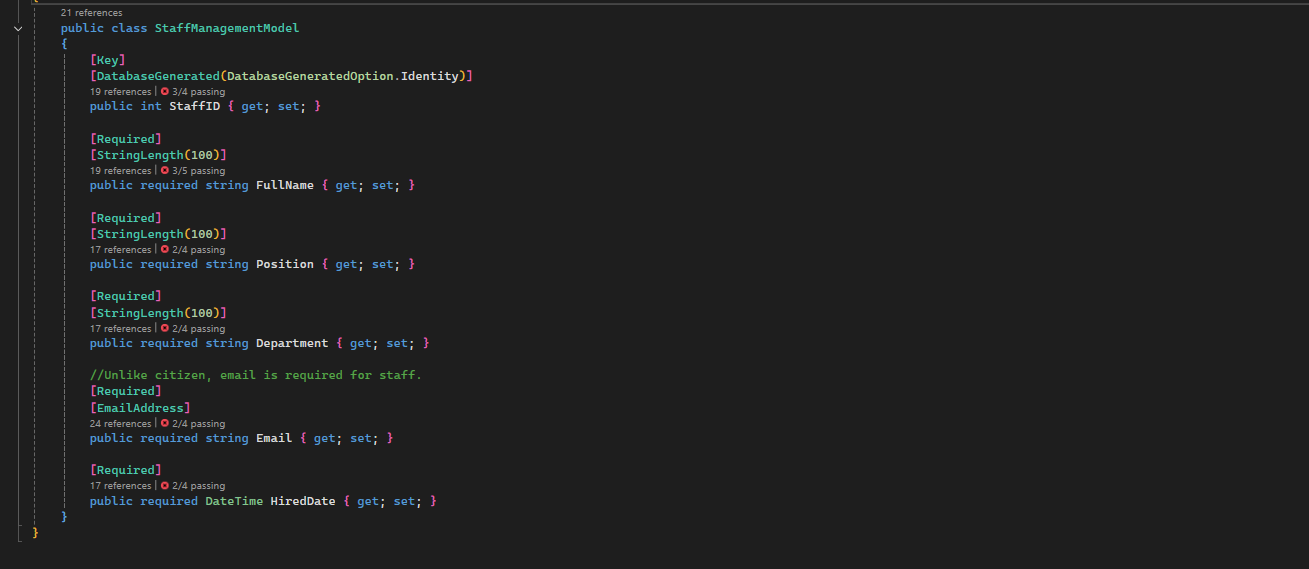
AI-generated content may be incorrect.

Similar to the citizen model, but this time included a foreign key to link the table to the citizen table.

A screenshot of a computer

AI-generated content may be incorrect.

The service request model also has a foreign key linking the citizen id to the citizen management model. With similar fields as the previous models, with autoincrementing ID, default values for specified fields and data validation.



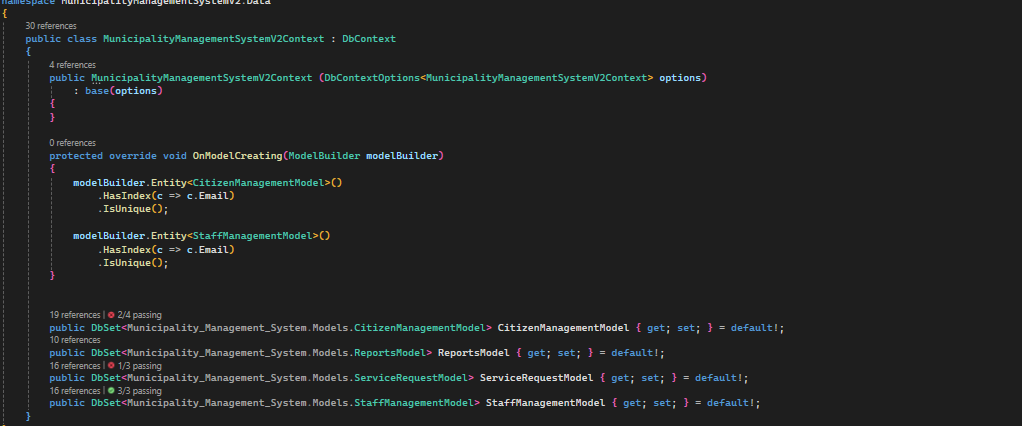
The staff model is as similar to the citizen model as it gets except this has required for the email and no phone number.

A screen shot of a computer

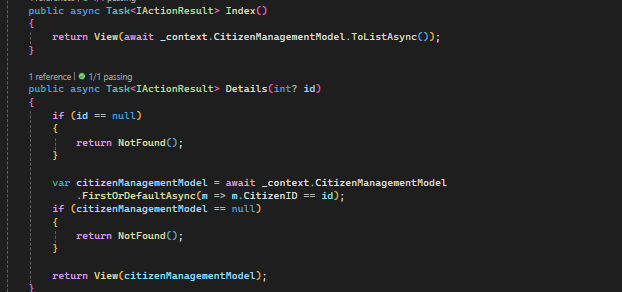
AI-generated content may be incorrect.

The project cs file configures the database context and tells the EF Core to use SQL server it fetches the connection string from the appsetting.json file.

Also enables MVC, allowing the app to handle web requests and finally compiles everything in the builder. Enforces HTTPS, and redirects to the Error page in case problems occur.



Forces the email to be unique for the citizen management model and staff management model. Defines the 4 tables in the database, created after the models in the model folder.



Index lists all of the records in the Citizen management table on the website and details lists all the details for a record matching the id.



Get and Post methods, to create the citizens which to add to the database. In the post, adds error handling for existing emails and if everything is valid saves it to the database.

A screen shot of a computer

AI-generated content may be incorrect.

A black background with many lines

AI-generated content may be incorrect.

Get and Posts method to edit an existing citizen through finding their id and even validating if the person is trying to update their email to one that another citizen already has. Binding allow them only to add fields matching that into it and saves the changes to the database.

A screenshot of a computer

AI-generated content may be incorrect.

Get and Post methods to delete a citizen who matches the ID provided.

A black screen with white text

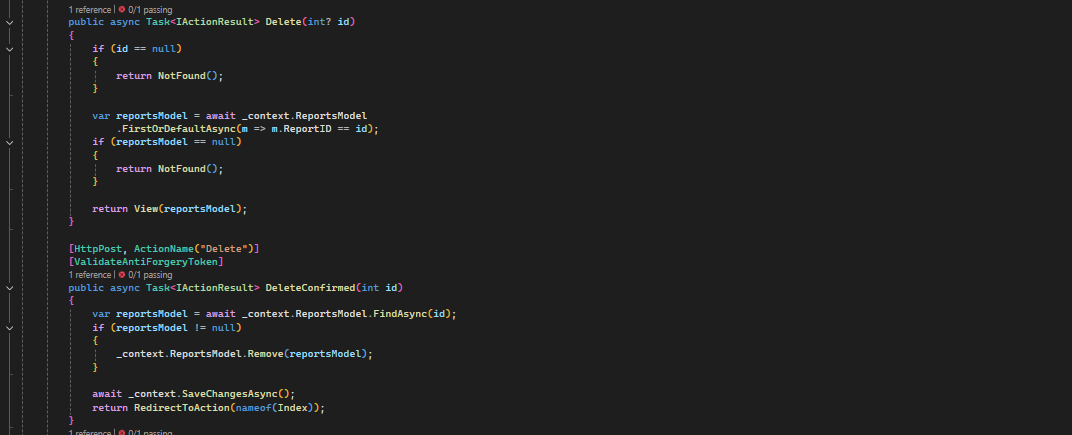
AI-generated content may be incorrect.

Specifies the database I’m working with and like previously, index lists all the records for the reports table and details lists the detail of a single report.

A black background with colorful lines

AI-generated content may be incorrect.

Returns the view for the reports model, and creates the report on post. Also making sure a valid citizen id has been provided before saving the record to the database.



Returns the view to the page matching the id of the report to delete it. The post deletes the report matching said id and return the index page if successful.

A black rectangular object with white text

AI-generated content may be incorrect.

Goes to the view of the report, where it will allow you to see the details of the review and change its status.

A screen shot of a computer

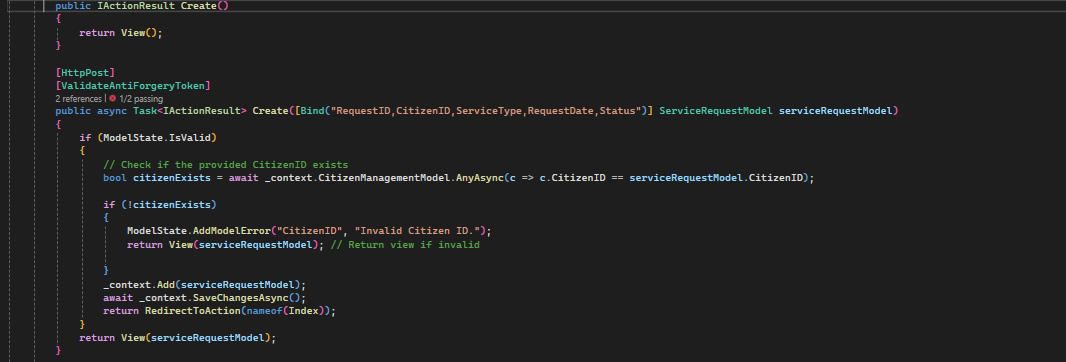
AI-generated content may be incorrect.

Updates only the status field for the report and saves the changes to the database.

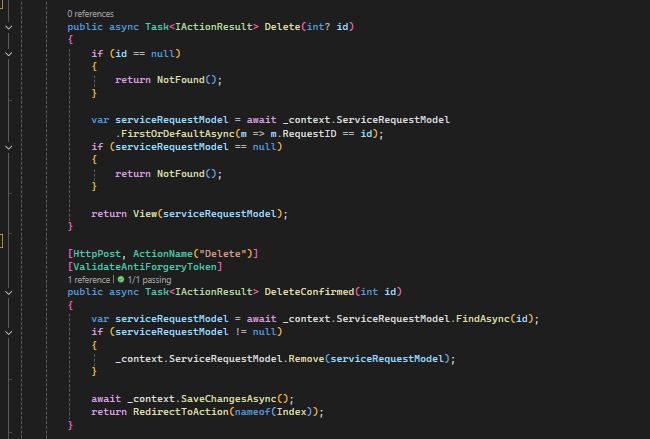
A screenshot of a computer

AI-generated content may be incorrect.

Lists all services requests from the database on the webpage and details retrieves the specific request matching the ID to the view.



Takes the user to the view to create the service request and the post adds the given information matching the fields in the parameters to the model. Adding and saving them to the database if successful and giving an error page if not.



Gives the view to the page to delete the specified service request matching the id. And on post, allows the deletion of it. Saving the changes to the database in the end.

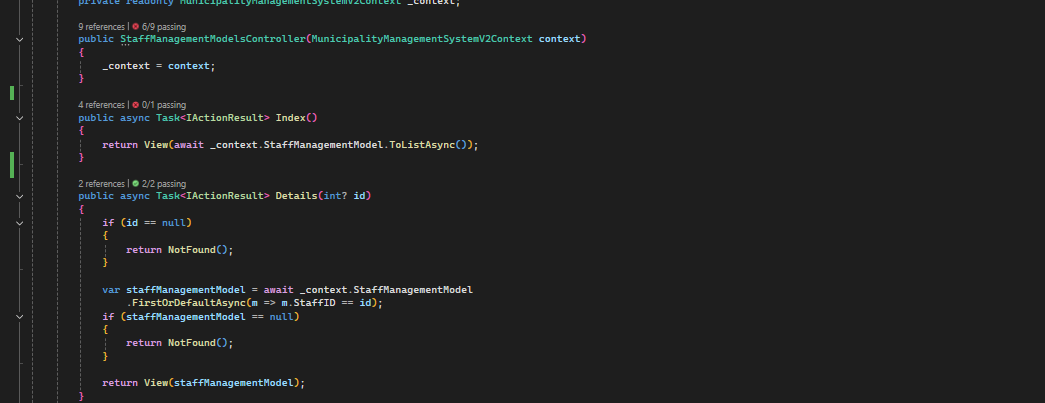
A screen shot of a computer program

AI-generated content may be incorrect.

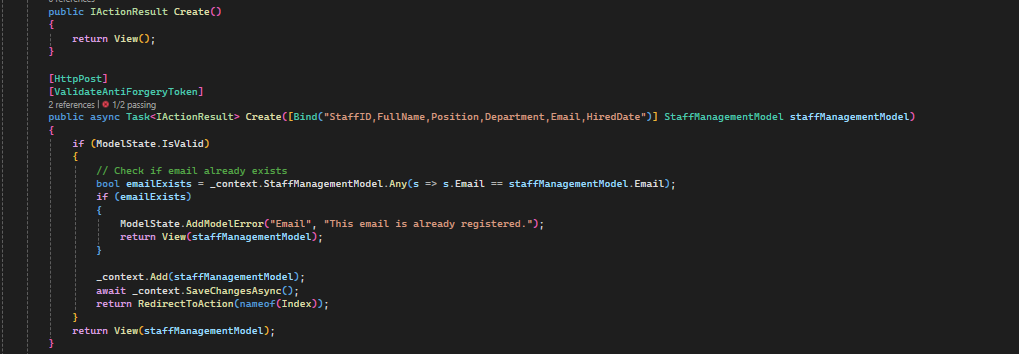
A screen shot of a computer program

AI-generated content may be incorrect.

Gives the details in a view matching the specified service request where the staff can change the status of it.



The Index gives the view where all the records for the staff will be listed, the details method will list all of the details for the specified staff member matching the ID.



Allows adding a new staff member, listing the view to add the data and on post saves it to the database if the data matches the fields in the parameters. Also enforces the unique of emails, so the program doesn’t combust because of multiple similar emails.



This method gives the view of the staff member that you wish to edit, along with their details and allow you to edit it.

A screen shot of a computer

AI-generated content may be incorrect.This post method adds the changes to the databases if the new data values matches the fields in the parameters and if the email address if unique for the staff.

A black screen with white text

AI-generated content may be incorrect.

Specifies which database to the connect to, in this case just a local SQL database.s

## Working Features Screenshots:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Date of birth isn’t required so no error and registration date would give today date. The model name, in this case ‘CitizenManagementModel’, has been changed to just Citizen and the same for the other models. I’m just not willing to screenshot it again for something so small.

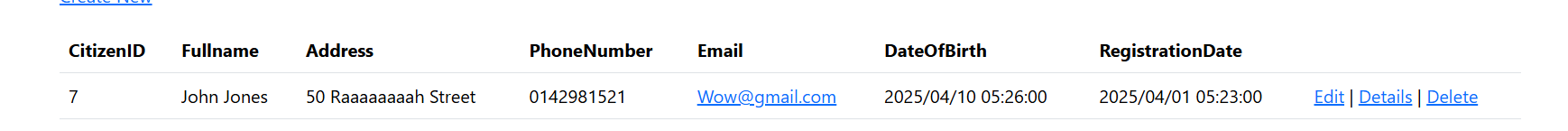
A close-up of a white background

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Required fields give me error.



The phone number, street address and date of birth was edited.

A screenshot of a computer

AI-generated content may be incorrect.

Details page

A screenshot of a computer

AI-generated content may be incorrect.

Delete Page

A screenshot of a computer

AI-generated content may be incorrect. Also the email has to be unique.

A group of black and orange text

AI-generated content may be incorrect.

The results of the delete, all records are gone.

A white background with black text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect. Invalid since the person got deleted and no one has the id of 1.

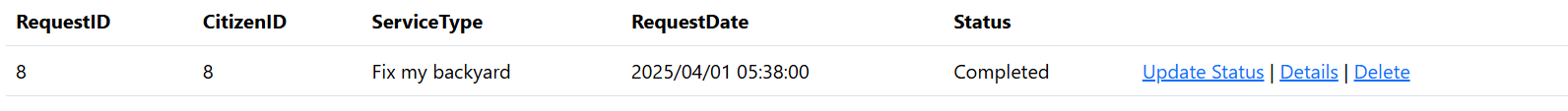
A close-up of a computer screen

AI-generated content may be incorrect.

8 is a valid Id so the service request went through.

A screenshot of a login page

AI-generated content may be incorrect.



Status went from pending to completed.

A white background with black text

AI-generated content may be incorrect.

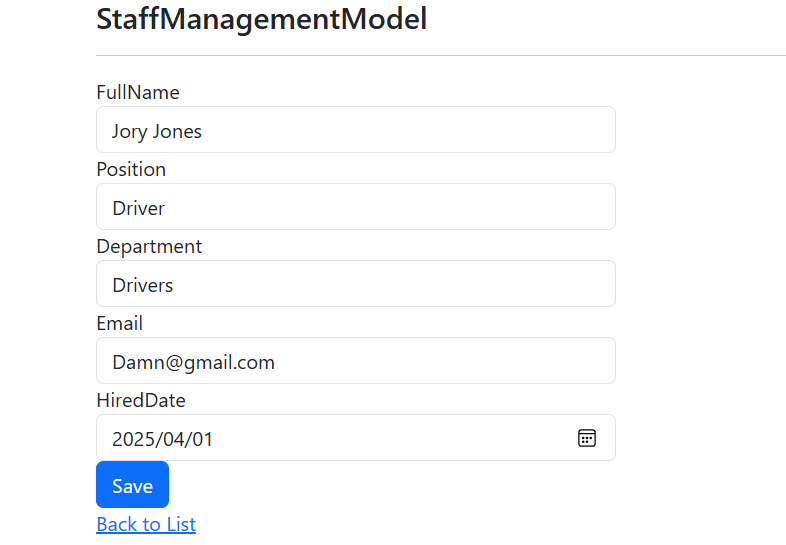
A screenshot of a browser

AI-generated content may be incorrect.

A close up of a screen

AI-generated content may be incorrect.

Review status went from Rejected to Resolved.



A screenshot of a computer

AI-generated content may be incorrect.

Staff added successfully.’ A screenshot of a computer

AI-generated content may be incorrect.

A close-up of a white background

AI-generated content may be incorrect.

Edit works successfully. Went from just a driver to Manager.

# Phase 4:

## Test Cases and Plan:

### Plan:

The testing type used was manual and Unit Testing through xUnit. The citizens management, staff management, services requests and report management was tested.

I used Visual Studio, Entity Framework Core In-Memory Database and my Firefox’ browser developer tools.

### Citizen

Test Case ID: TC001

Title: View All Citizens

Test Steps: Navigate to /CitizenManagementModels

Test Data (Input): N/A

Expected Output: List of all citizens appears

Status: Pass

A screenshot of a computer

AI-generated content may be incorrect.

---

Test Case ID: TC002

Title: View Citizen Details (Valid ID)

Test Steps: Click Details for citizen with ID = 8

Test Data (Input): CitizenID = 8

Expected Output: Full details of citizen #1 shown

Status: Pass

A screenshot of a computer

AI-generated content may be incorrect.

---

Test Case ID: TC003

Title: View Citizen Details (Invalid ID)

Test Steps: Navigate to /Details/999

Test Data (Input): CitizenID = 999

Expected Output: NotFound or error page

Status: Pass

A black screen with white text

AI-generated content may be incorrect.

----

Test Case ID: TC004

Title: Create Citizen (Valid Input)

Test Steps: Fill form and submit

Test Data (Input): Fullname: "Amy Strawson", Email: "amy.strawson@gmail.com", etc.

Expected Output: Citizen created and redirected to Index

Status: Pass



---

Test Case ID: TC005

Title: Create Citizen (Duplicate Email)

Test Steps: Submit with an existing email

Test Data (Input): Email: "boom@gmail.com"

Expected Output: Duplicate email error

Status: Pass

A close up of a text

AI-generated content may be incorrect.

---

Test Case ID: TC006

Title: Edit Citizen (Valid Update)

Test Steps: Change name and submit

Test Data (Input): New Name: "Amy Strawson The Second" etc

Expected Output: Record updated, redirected to Index

Status: Pass



---

Test Case ID: TC007

Title: Delete Citizen (Valid ID)

Test Steps: Confirm delete on citizen ID = 11

Test Data (Input): CitizenID = 11

Expected Output: Citizen deleted and redirected

Status: Pass



### Reports:

Test Case ID: TC008

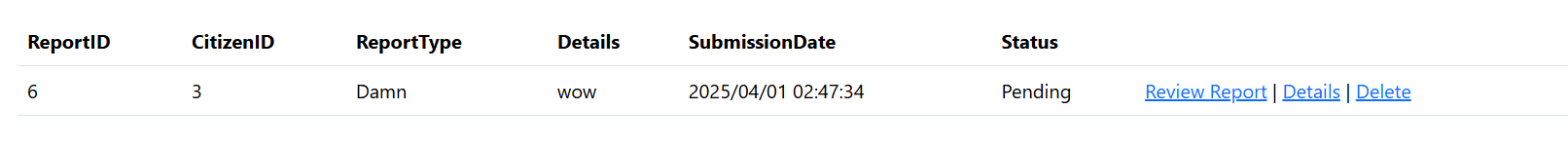
Title: View All Reports

Test Steps: Navigate to /ReportsModels

Test Data (Input): N/A

Expected Output: List of all reports appears

Status: Pass



---

Test Case ID: TC009

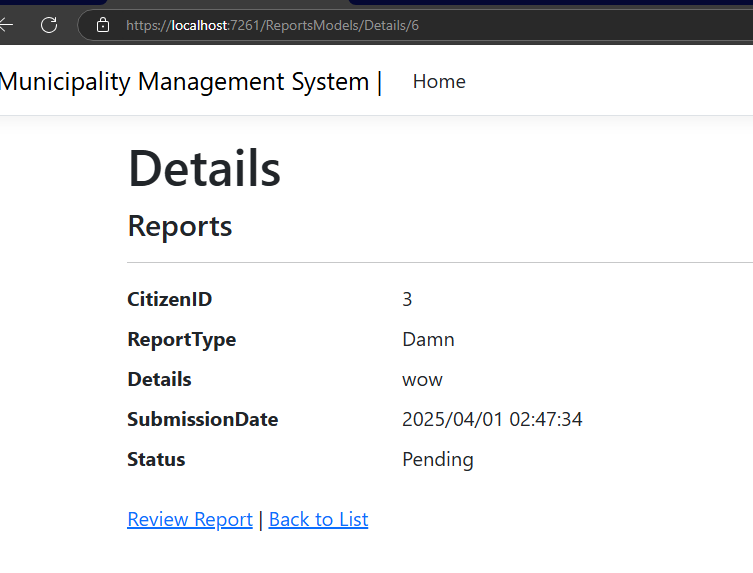
Title: View Report Details (Valid ID)

Test Steps: Click Details for report with ID = 6

Test Data (Input): ReportID = 6

Expected Output: Full details of report #1 shown

Status: Pass



----

Test Case ID: TC010

Title: Create Report (Valid Input)

Test Steps: Fill form with valid details and submit

Test Data (Input): CitizinID: 8, ReportType: "Incident", Details: "Ahh no", Date: Today etc

Expected Output: Report created and redirected to Index

Status: Pass



Test Case ID: TC011

Title: Create Report (Invalid ID ☹)

Test Steps: Submit form with invalid id

Test Data (Input): ReportType: "Incident", Details: "Save me." ID=999

Expected Output: Validation error messages appear

Status: Pass

A screenshot of a computer

AI-generated content may be incorrect.

---

Test Case ID: TC012

Title: Delete Report (Valid ID)

Test Steps: Navigate to Delete action with a valid report ID such 8

Test Data (Input): ReportID = 8

Expected Output: The report is fetched and displayed for confirmation.

Status: Pass

A screenshot of a computer

AI-generated content may be incorrect.

---

Test Case ID: TC013

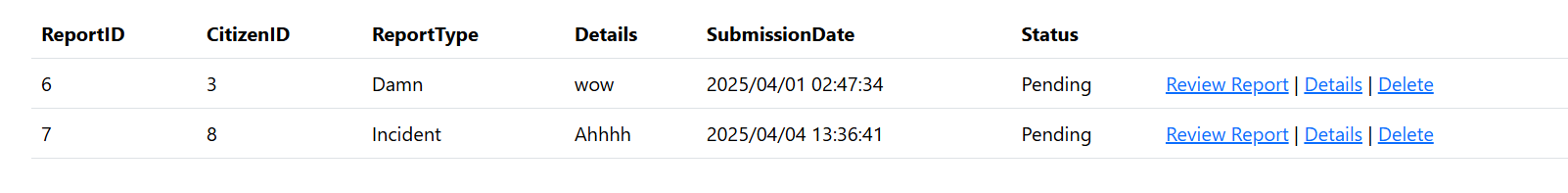
Title: Delete Confirmed Report (Valid ID)

Test Steps: Confirm the deletion of the report with a valid ReportID , in this case 8

Test Data (Input): ReportID = 8

Expected Output: Report is deleted successfully, and the user is redirected to the Index page.

Status: Pass



My boy is gone.

### Services:

Test Case ID: TC014

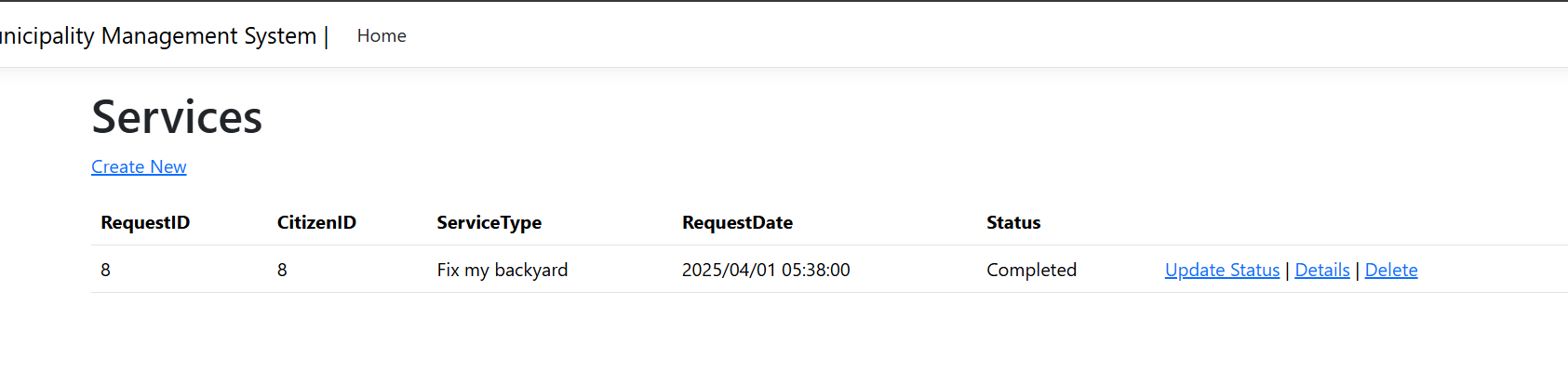
Title: View All Service Requests

Test Steps: Navigate to the Index /ServiceRequestModels

Test Data (Input): N/A

Expected Output: The list of service requests is displayed.

Status: Pass



----

Test Case ID: TC015

Title: View Service Request Details (Valid ID)

Test Steps: Navigate to the Details/8 in the web bar.

Test Data (Input): RequestID = 8

Expected Output: The details of the service request with the given RequestID are displayed.

Status: Pass

A screenshot of a computer

AI-generated content may be incorrect.

---

Test Case ID: TC016

Title: View Service Request Details (Invalid ID)

Test Steps: Navigate to the Details/99 in the web bar on top

Test Data (Input): RequestID = 99

Expected Output: A "Not Found" page is displayed.

Status: Pass

A black screen with white text

AI-generated content may be incorrect.

---

Test Case ID: TC017

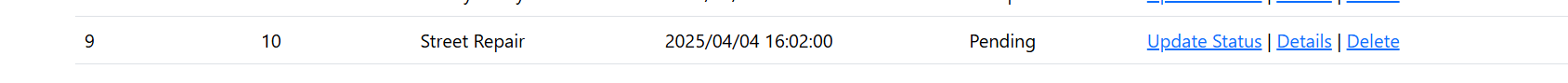
Title: Create Service Request (Valid CitizenID)

Test Steps: Submit a new service request with a valid CitizenID.

Test Data (Input): CitizenID = 10, ServiceType = "Street Repair", Status = "Pending"

Expected Output: The service request is created and the user is redirected to the Index page.

Status: Pass



---

Test Case ID: TC018

Title: Create Service Request (Invalid CitizenID)

Test Steps: Submit a new service request with an invalid CitizenID.

Test Data (Input): CitizenID = 89999, ServiceType = "Street Repair", Status = "Pending"

Expected Output: A validation error is shown, indicating the CitizenID is invalid.

Status: Pass

A screenshot of a computer

AI-generated content may be incorrect.

---

Test Case ID: TC019

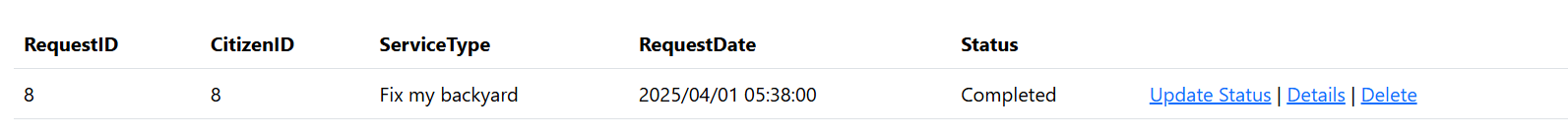
Title: Delete Service Request (Valid ID)

Test Steps: Delete a service request with a valid RequestID.

Test Data (Input): RequestID = 10

Expected Output: The service request is deleted and the user is redirected to the Index page.

Status: Pass



---

Test Case ID: TC020

Title: Update Service Request Status (Valid ID)

Test Steps: Update the status of a service request with valid ID in this case RequestID 8.

Test Data (Input): RequestID = 8, New Status = "Pending"

Expected Output: The service request status is updated, and the user is redirected to the Index page.

Status: Pass

### 

Went from completed to pending.

### Staff:

Test Case ID: TC021

Title: Index (Returns View With Staff List)

Test Steps: Call the Index method on the StaffManagementModelsController.

Test Data (Input): None

Expected Output: The action returns a view with a list containing 1 staff member.

Status: Pass

A white background with black text

AI-generated content may be incorrect.

---

Test Case ID: TC022

Title: Details (Valid ID Returns View With Staff Details)

Test Steps: Call the Details method with a valid StaffID, in this case 5

Test Data (Input): StaffID = 5

Expected Output: The action returns a view with the staff details for the given ID.

Status: Pass

A white rectangular object with a black line

AI-generated content may be incorrect.

---

Test Case ID: TC023

Title: Details (Invalid ID Returns Not Found)

Test Steps: Call the Details method with an invalid StaffID .

Test Data (Input): StaffID = 99

Expected Output: The action returns a NotFoundResult indicating the staff ID doesn't exist.

Status: Pass

A screenshot of a computer error

AI-generated content may be incorrect.

---

Test Case ID: TC024

Title: Create (Valid Model Redirects To Index)

Test Steps: Call the Create method with a valid StaffManagementModel.

Test Data (Input): FullName = "John Dorris", Position = "Developer", Department = "IT", Email = "john.dorris @gmail.com", HiredDate = DateTime.Now

Expected Output: The action redirects to the Index page, and the staff member is added to the database.

Status: Pass

A close-up of a computer screen

AI-generated content may be incorrect.

---

Test Case ID: TC025

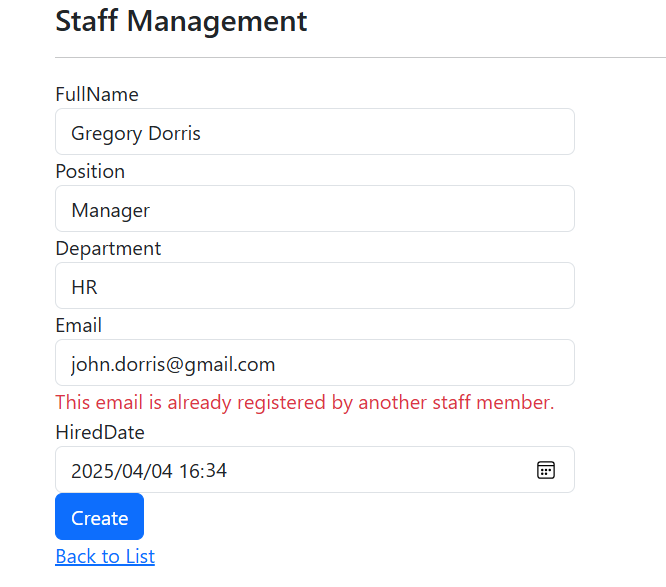
Title: Create (Existing Email Returns View With Error)

Test Steps: Call the Create method with a StaffManagementModel containing an email that already exists in the database.

Test Data (Input): FullName = "Gregory Dorris", Position = "Manager", Department = "HR", Email = "john.dorris@gmail.com", HiredDate = DateTime.Now

Expected Output: The action returns the view with an error indicating that the email is already registered.

Status: Pass



---

Test Case ID: TC026

Title: Edit (Valid ID Returns View With Staff Details)

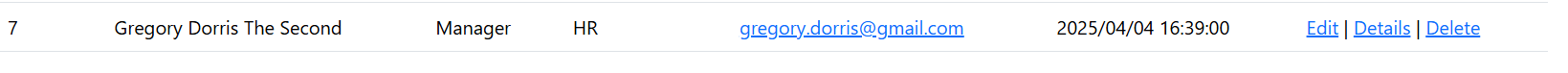
Test Steps: Call the Edit method with a valid StaffID in this case 5 and updated staff data.

Test Data (Input): StaffID = 5, FullName = "Jory Jones The Second"

Expected Output: The action redirects to the Index page, and the staff details are updated.

Status: Pass





---

Test Case ID: TC027

Title: Edit (Invalid Email Returns View With Error)

Test Steps: Call the Edit method with a valid StaffID and a new email that already exists in the database.

Test Data (Input): StaffID = 2, Email = "john.dorris@gmail.com"

Expected Output: The action returns the view with an error message indicating the email is already registered by another staff member.

Status: Pass

A screenshot of a computer

AI-generated content may be incorrect.

---

Test Case ID: TC028

Title: Delete (Valid ID Redirects To Index)

Test Steps: Call the DeleteConfirmed method with a valid StaffID to delete a staff member.

Test Data (Input): StaffID = 7

Expected Output: The action redirects to the Index page, and the staff member is deleted from the database.

Status: Pass

A close-up of a computer screen

AI-generated content may be incorrect.

My goat Gregory Is gone.

----

Test Case ID: TC029

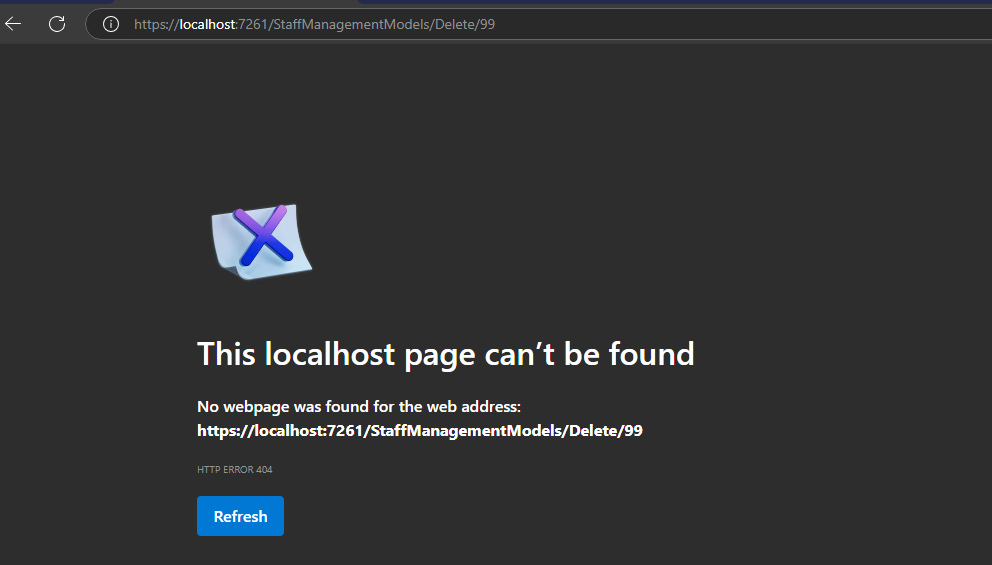
Title: Delete (Invalid ID Returns Not Found)

Test Steps: Call the Delete method with an invalid StaffID.

Test Data (Input): StaffID = 99

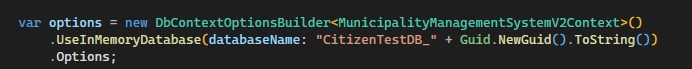
Expected Output: The action returns a NotFoundResult indicating the staff member was not found.

Status: Pass



## Bug Reports and Resolutions:

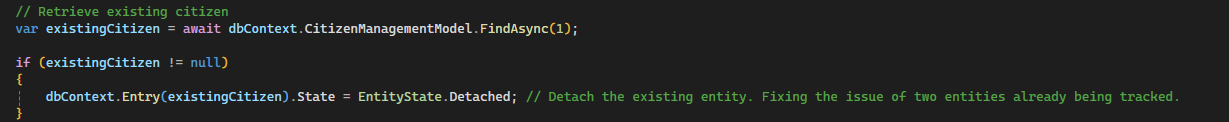
The seeded database data carried over between tests, resulting in test failures.



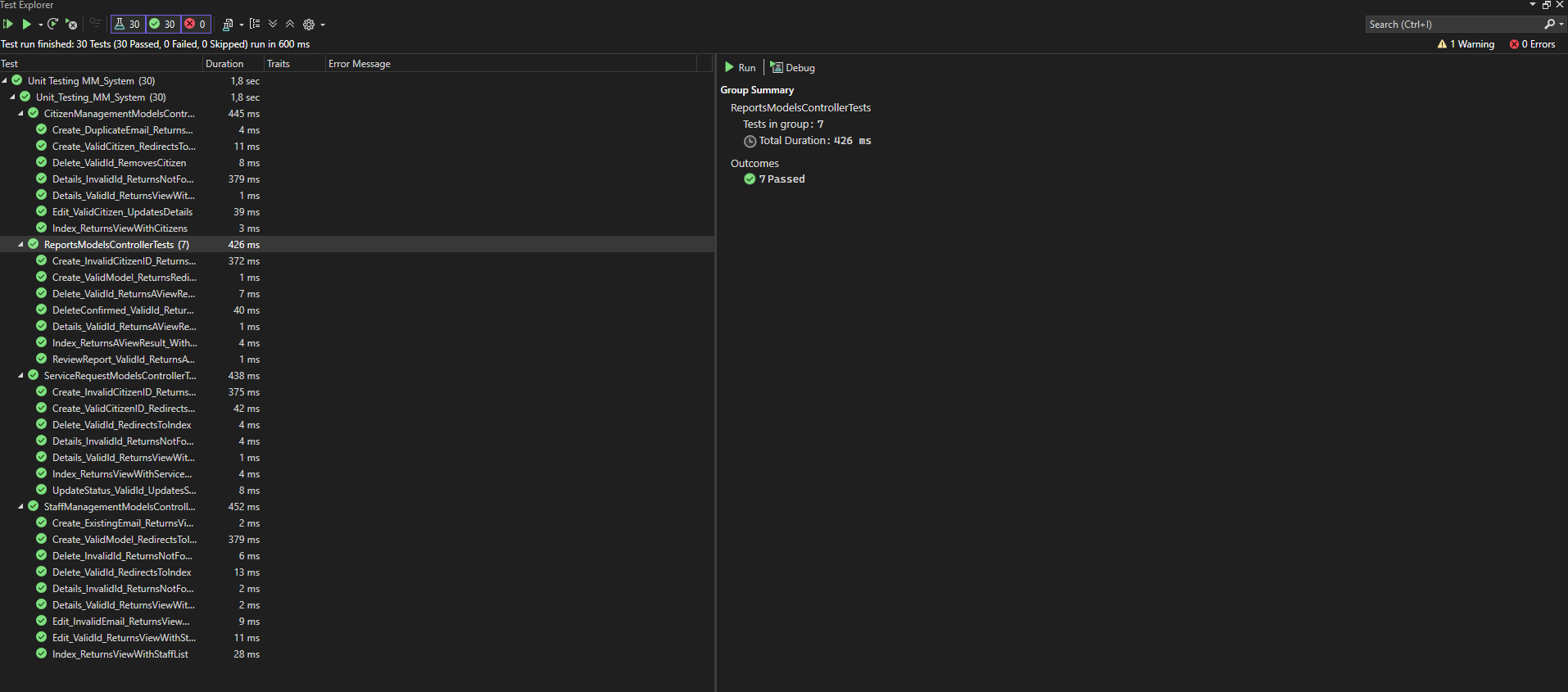
The Guid.NewGuid generated a unique database name each time, so every test run was fresh with no leftover data from others test.

---

Some entities couldn’t be tracked because another instance with the same key was already tracked.

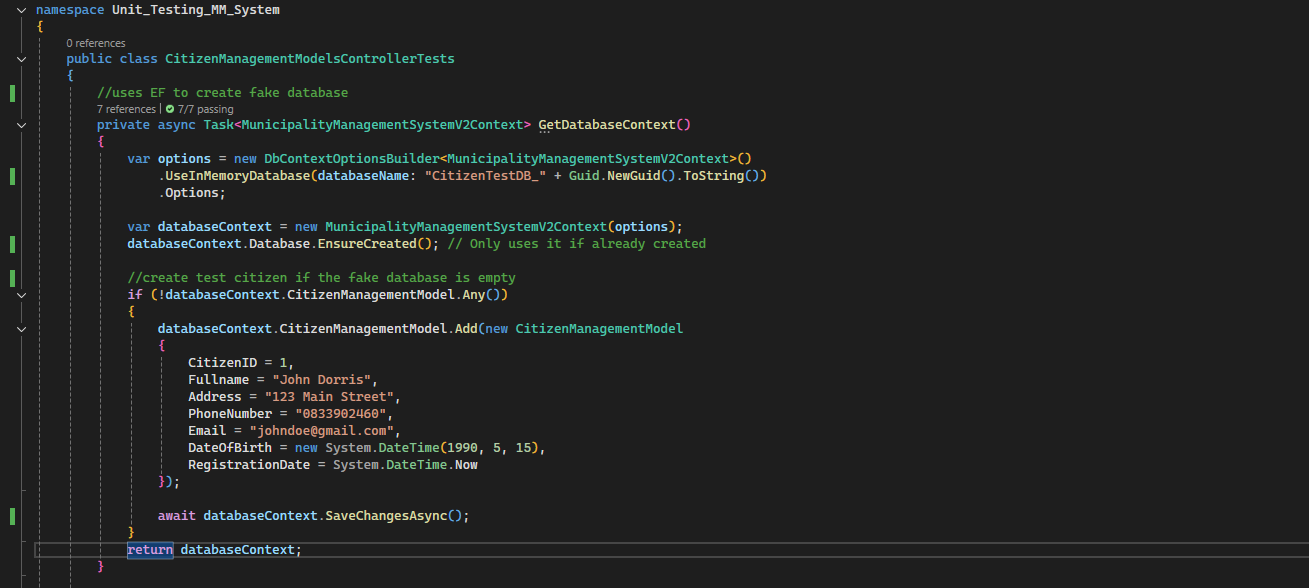


## Screenshots of unit test results:



Explanations in comments.

Citizen Tests:



A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

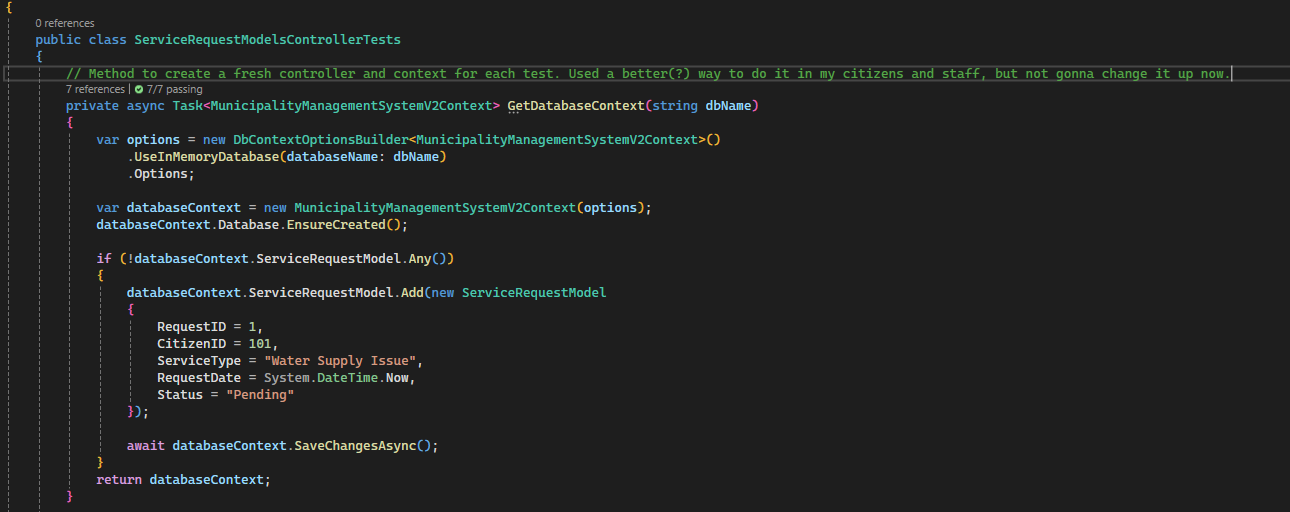
A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

Service Requests:



A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

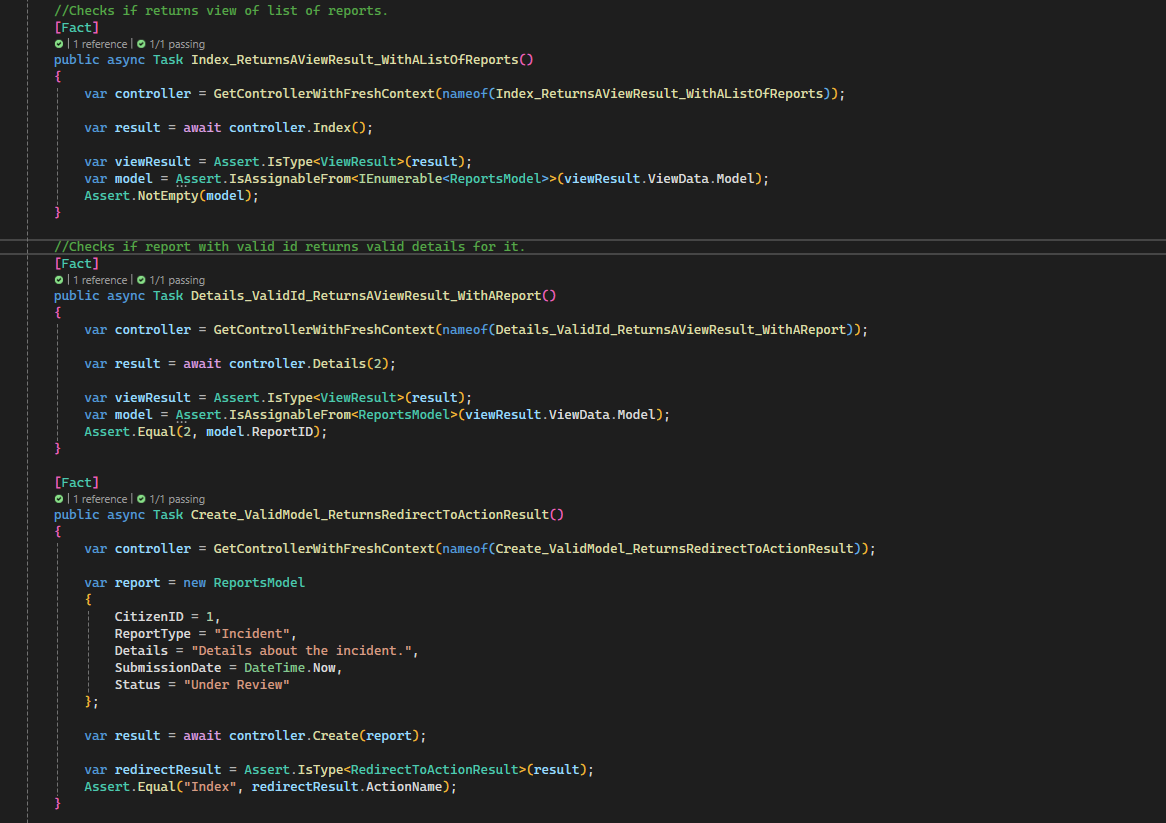
A screen shot of a computer code

AI-generated content may be incorrect.

Reports:

A screenshot of a computer program

AI-generated content may be incorrect.



A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

# Phase 5:

Power Point in separate file called ‘Phase5\_Presentation’

GITHUB LINK: https://github.com/JaydonEllick/ADP631-FA1-Municipality-System