

Population Report generator

Checklist 1



|  |  |
| --- | --- |
| Name | ID |
| Syed Abdul Samad Zaidi | 40692317 |
| Muhammad Ali Khalid | 40692222 |
| Muhammad Taha Khalid | 40692271 |
| Hamza Shahid | 40692218 |

**Population report generator?**

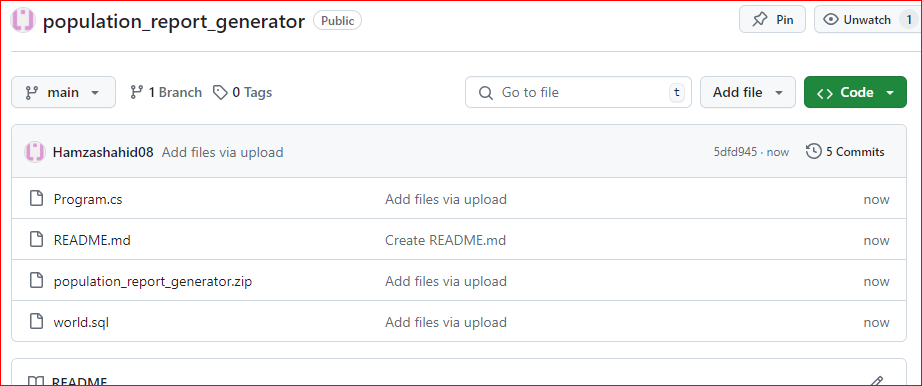
A software program called the demographic Report System creates comprehensive demographic reports. Users can examine population data at many tiers, such as nations, cities, and major cities. The system contains features that allow population data to be categorized based on size and provides extensive details regarding language speakers and population distribution. This project, which is a component of the Software Engineering Methods program, prioritizes teamwork and adheres to Scrum project management concepts.

**GITHUB:** **https://github.com/Hamzashahid08/population\_report\_generator/tree/main**

* **Setup of Repository:**

To help team members collaborate and manage version control, a project repository was set up on GitHub. The procedure for configuring the repository and uploading the initial project files is covered in this section.

1. Created a new repository on GitHub named "Population\_report\_generator".
2. Uploaded initial project files, including:
   * **Program.cs**: The main program file with placeholder functions and database connection setup.
   * **README.md**: Document providing an overview and setup instructions for the project
   * **World**.sql:data base that is connected to generate report



* **Backlog:**

1. **Setting Up and Getting Started**
   * **User Story**: As a team member, I want to set up the development environment and create the project structure.
   * **Tasks**:
     + Set up the MySQL database connection.
     + Create the initial C# project structure.
     + Initialize a GitHub repository with the basic project structure.
     + Choose project management tools (like GitHub Projects or Zube.io).
2. **Retrieving Population Data**
   * **User Story**: As a user, I want to easily retrieve population data at different levels (world, continent, region, country, district, city).
   * **Tasks**:
     + Write MySQL queries for data retrieval.
     + Design classes to handle population data.
     + Implement basic data retrieval functions in C#.
     + Set up unit testing for data retrieval.
3. **Generating Population Reports**
   * **User Story**: As a user, I want to analyze population trends through reports (e.g., countries, cities, capital cities organized by population).
   * **Tasks**:
     + Create report generation functionality.
     + Implement reports for countries, cities, and capital cities.
     + Add sorting options based on population size.
     + Include top N populated countries, cities, and capital cities.
4. **Language Speaker Statistics**
   * **User Story**: As a user, I want to understand global language demographics (e.g., Chinese, English speakers).
   * **Tasks**:
     + Design and implement language speaker statistics functionality.
     + Retrieve language data from the database.
     + Calculate and display language speaker percentages globally.
5. **Population Breakdown**
   * **User Story**: As a user, I want to explore population breakdown by continent, region, country, and city.
   * **Tasks**:
     + Implement population breakdown functionality.
     + Calculate and display total population and percentages for each geographical level.
     + Ensure accuracy and validate population data.
6. **User-Friendly Interface** 
   * **User Story**: As a user, I’d like an intuitive interface for interacting with population data.
   * **Tasks**:
     + Design GUI components for displaying reports.
     + Implement navigation and interaction features.
     + Ensure the interface is responsive and user-friendly.
7. **Documentation and Quality Assurance**
   * **User Story**: As a team member, I want thorough documentation and code quality.
   * **Tasks**:
     + Document system architecture and design decisions.
     + Write user and developer documentation.
     + Conduct code reviews and address feedback.
     + Perform unit testing and ensure good test coverage.

.

|  |  |
| --- | --- |
| **Task Description** | **Priority** |
| Setup and Initialization | High Importance, High Feasibility |
| Initialize GitHub repository with basic structure | High Importance, High Feasibility |
| Set up project management tools (GitHub Projects or Zube.io) | High Importance, High Feasibility |
| Set up MySQL database connection | High Importance, Medium Feasibility |
| Implement MySQL queries for data retrieval | High Importance, Medium Feasibility |
| Implement basic data retrieval functions in C# | High Importance, Medium Feasibility |
| Set up unit testing for data retrieval functions | High Importance, Medium Feasibility |
| Design report generation functionality | High Importance, Medium Feasibility |
| Implement reports for countries, cities, and capitals | High Importance, Medium Feasibility |
| Add sorting functionality for reports based on population size | High Importance, Medium Feasibility |
| Implement top N populated countries, cities, and capitals | High Importance, Medium Feasibility |
| Design and implement language speaker statistics functionality | High Importance, Medium Feasibility |
| Retrieve language data from the database | High Importance, Medium Feasibility |
| Calculate and display percentages of language speakers globally | High Importance, Medium Feasibility |
| Implement population breakdown functionality | High Importance, Medium Feasibility |
| Calculate and display total population and percentages | High Importance, Medium Feasibility |
| Design GUI components for report display | Medium Importance, Medium Feasibility |
| Implement navigation and interaction features | Medium Importance, Medium Feasibility |
| Ensure responsiveness and usability of the interface | Medium Importance, Medium Feasibility |
| Document system architecture and design decisions | Medium Importance, Low Feasibility |
| Write user and developer documentation | Medium Importance, Low Feasibility |
| Conduct code reviews and address feedback | Medium Importance, Low Feasibility |
| Perform unit testing and ensure test coverage | Medium Importance, Low Feasibility |

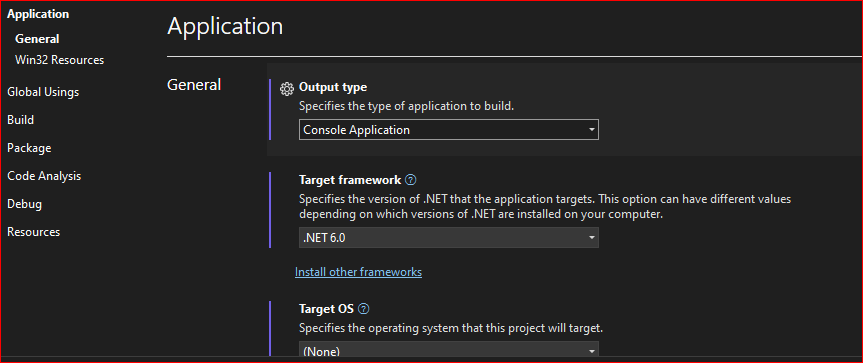
* **Configuring Project Builds with MSBuild**

In this section, we’ll walk through the steps to set up your C# project for building into either a standalone executable or a DLL using MSBuild. MSBuild is a powerful build tool integrated with Visual Studio

1. **Project Setup**

* We set Visual Studio with all the necessary source files, configurations, and dependencies.

1. **Configure Output Type**



1. **Build the Project**

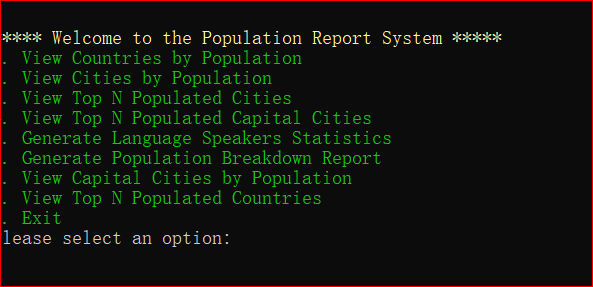
* Selecting “Build > Build Solution” from the Visual Studio menu.
* This action compiles your project into the specified output type (executable or DLL).

4. **Include MySQL Connector/C**:

* + We Downloaded and extracted MySQL Connector/C.
  + We Added the include and library directories to VC++ Directories.

**5.Verify Output**

* + After building, locate the compiled executable or DLL in the project’s bin\Debug or bin\Release directory.
  + Confirm that the output file is generated correctly.



* **Dockerfile :**

1. **Environment Setup:**

* Installed Docker Desktop.
* Created a Dockerfile for the project.

1. **Building the Image:**

* Built a Docker image named population-report-system.

1. **Container Execution:**

* Created and ran a container named population-report-container.

**Docker file:**

# Use the official .NET SDK image as the base image

FROM mcr.microsoft.com/dotnet/sdk:latest AS build

# Set the working directory inside the container

WORKDIR /app

# Copy the necessary project files into the container

COPY . /app

# Restore the dependencies

RUN dotnet restore

# Build the project

RUN dotnet build -c Release -o /app/build

# Publish the project

RUN dotnet publish -c Release -o /app/publish

# Use the official .NET runtime image as the base image for runtime

FROM mcr.microsoft.com/dotnet/aspnet:latest AS runtime

# Set the working directory inside the container

WORKDIR /app

# Copy the published output from the build stage to the runtime stage

COPY --from=build /app/publish .

# Install MySQL client library

RUN apt-get update && apt-get install -y \

default-libmysqlclient-dev \

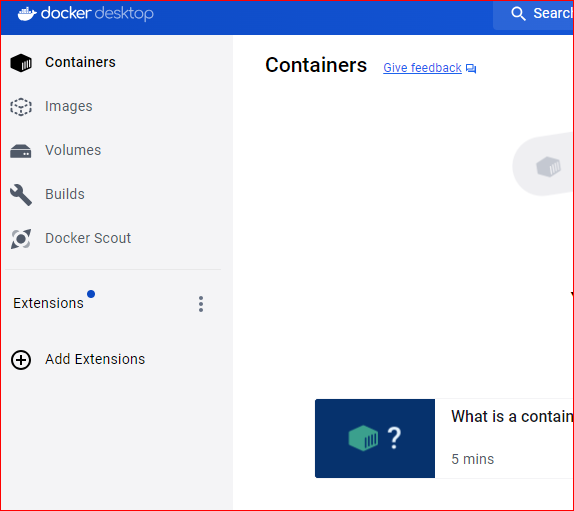
&& rm -rf /var/lib/apt/lists/\*

# Expose the port on which your application will run (if applicable)

# EXPOSE 8080

# Set the default command to run the application

CMD ["dotnet", "YourApp.dll"]



* **Setting up GitHub Actions :**

Implemented GitHub Actions workflows to automate project setup, build process, and testing for our C# project. This provides continuous integration and deployment capabilities within our GitHub repository.

**Workflow Creation:**

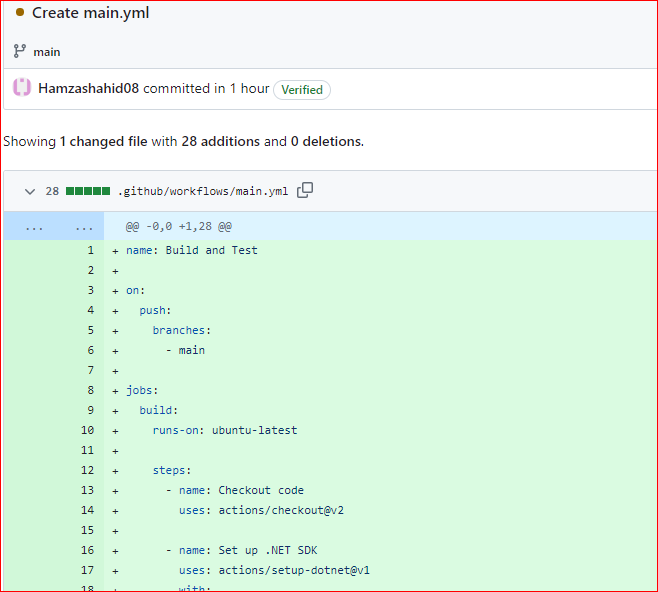
* Created a new workflow file named main.yml in .github/workflows.

**Workflow Definition:**

* Configured triggers for the workflow to execute on push to main and pull\_request to main.

**Workflow Execution:**

* Saved changes to main.yml and committed to the main branch.
* Monitored execution in the GitHub Actions tab for logs and errors.



* **GitFlow Branching :**

Implemented GitFlow branching strategy in the GitHub repository "population\_report\_generator" to manage development, releases, and hotfixes effectively.

**Steps Taken:**

* **Repository Cloning:**

Cloned the GitHub repository locally using github desktop.

* **GitFlow Initialization:**

Initialized GitFlow to set up branch structure (master, develop, feature,

* **Feature Branch Creation:**

Created feature branches for new features or enhancements.

* **Feature Branch Completion:**

Completed feature branches and merged changes into develop.

* **Release Branch Creation:**

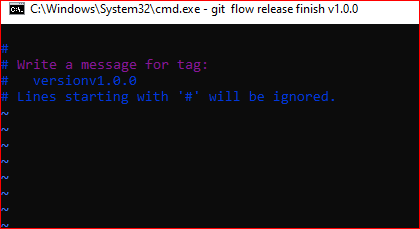
Created release branches for preparing new releases.

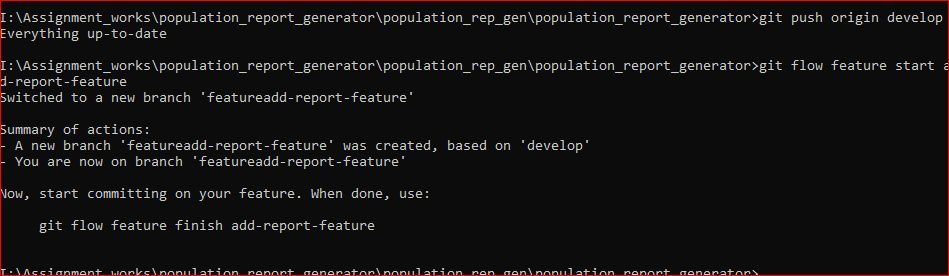
* **Hotfix Handling:**

Managed hotfixes for urgent production fixes.

* **Workflow Tracking:**

Tracked workflow using git flow status to monitor ongoing development, releases, and hotfixes.

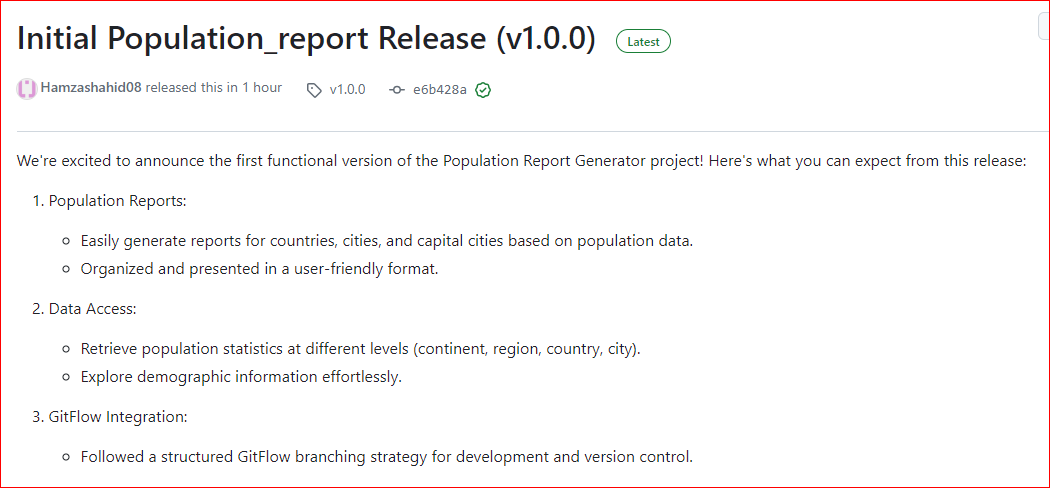




* **First Release:**

Created the initial release (v1.0.0) of the Population Report Generator project on GitHub, marking the first functional version ready for evaluation.

1. **Project Readiness:**
   * Ensured that the project represents the initial functional copy ready for evaluation.
2. **Versioning Update:**
   * Updated versioning to v1.0.0 in the project.
3. **Commit and Push Changes:**
   * Committed all final changes and pushed to the develop branch on GitHub.
4. **Tag Creation:**
   * Created a release tag (v1.0.0) to mark the initial release.
5. **Drafted Release on GitHub:**
   * Drafted a release on GitHub, providing a description of the release contents and relevant information.



* **Code of Conduct:**

1. **Respectful Communication:**
   * Treat all team members with respect, appreciating their diverse perspectives and contributions.
   * Use inclusive language and avoid any discriminatory or hurtful remarks based on race, gender, sexual orientation, religion, or other characteristics.
2. **Collaboration and Teamwork:**
   * Foster an open and collaborative atmosphere.
   * Encourage constructive feedback and support each other’s success.
   * Work together effectively to achieve project goals.
3. **Professionalism:**
   * Maintain integrity, honesty, and accountability in all interactions.
   * Safeguard sensitive project information and discussions.
4. **Conflict Resolution:**
   * Address conflicts promptly and respectfully.
   * Seek understanding and mutually beneficial solutions.
   * Involve project leads or mediators if necessary.
5. **Adherence to Guidelines:**
   * Follow established coding standards, version control practices, and project management processes (like GitFlow or Agile).
   * Strive for consistency and quality in your work.
6. **Inclusivity and Diversity:**
   * Celebrate differences in background, perspectives, and experiences.
   * Create an inclusive environment where everyone feels valued and has equal opportunities to contribute.