Code Description

This project consists of a password generator with both a command-line interface (CLI) and a web interface. The code is organized into three main directories: `cmd`, `internal`, and `web`.

`internal/pwd`

This directory contains the core logic for password generation and formatting.

- **`pwd.go`**:
  - `GeneratePassword(length int, includeSymbols bool) (string, error)`: This function generates a random password of the specified length. It includes letters and numbers by default, and optionally includes symbols if `includeSymbols` is true. It uses `crypto/rand` to generate random bytes.
  - `FormatPassword(password string) (string, error)`: This function takes a password string and formats it by adding spaces every 3 characters.
- **`pwd_test.go`**:
  - Contains unit tests for the `GeneratePassword` and `FormatPassword` functions. It tests various scenarios, including different lengths, symbol inclusion, and formatting.

`cmd/cli`

This directory contains the code for the command-line interface.

- **`main.go`**:
  - Parses command-line flags using the `flag` package to configure password length, count, and symbol inclusion.
  - Validates the minimum password length.
  - Uses the `pwd.GeneratePassword` and `pwd.FormatPassword` functions from the `internal/pwd` package to generate and format passwords.
  - Prints the generated passwords to the console, both raw and formatted.
- **`Makefile`**:
  - Defines build targets for different operating systems (Linux, macOS, Windows).
  - Uses `go build` to compile the CLI application.
  - Sets build flags to include version and build time information.
  - Provides targets for cleaning the build directory and running the application.

`web`

This directory contains the code for the web interface.

- **`main.go`**:
  - Uses the `embed` package to embed static files (HTML, CSS, JavaScript) into the binary.
  - Sets up HTTP handlers to serve static files, the index page, and the password generation endpoint.

    - Parses query parameters from the `/generate` endpoint to configure password length, count, and symbol inclusion.
    - Uses the `pwd.GeneratePassword` and `pwd.FormatPassword` functions from the `internal/pwd` package to generate and format passwords.
    - Returns the generated passwords as a JSON response.
- **`static/index.html`**:
    - Defines the HTML structure for the password generator web page.
    - Includes input fields for password length, count, and symbol inclusion.
    - Includes a button to trigger password generation.
    - Displays the generated passwords in a results section.
- **`static/script.js`**:
    - Contains JavaScript code to handle user interaction with the web page.
    - Fetches password data from the `/generate` endpoint.
    - Dynamically updates the results section with the generated passwords.
- **`static/styles.css`**:
    - Contains CSS styles for the web page.
    - Provides basic styling for the layout, input fields, buttons, and password display.
- **`Makefile`**:
    - Similar to the CLI Makefile, defines build targets for different operating systems.
    - Uses `go build` to compile the web application.
    - Sets build flags to include version and build time information.
    - Provides targets for cleaning the build directory and running the application.

Overall Structure

The project follows a clear separation of concerns:

- The `internal/pwd` package provides the core password generation logic, which is used by both the CLI and web interfaces.
- The `cmd/cli` package provides a command-line interface for generating passwords.
- The `web` package provides a web interface for generating passwords.

This structure makes the code modular, testable, and maintainable.