Documentation for Advanced Firewall Implementation project

ADVANCED FIREWALL PROJECT

This is a web application designed to manage incoming and outgoing network traffic using predefined rules.

FUNCTIONALITY:

1. **Login Authentication:**

Provides secure access to the application.

2. **Whitelists and Blacklists Management:**

Allows users to manage lists of allowed and blocked entities.

3. **Rule Management:**

Facilitates adding, moving, and removing rules between whitelists and blacklists.

4. **Packet Validation:**

Checks the validity of network packets based on configured rules.

OBJECTIVES:

1. **Design Patterns**

- **Strategy, Factory, and Singleton Patterns:**

  Implemented for efficient rule handling and scalability.

2. **Network Traffic Filtering**

- **IP Addresses, Ports, and Protocols:**

  Implements strategies for filtering traffic based on these parameters.

3. **User Interface**

- **Rule Management Interface:**

  Provides a user-friendly web interface for setting rules and monitoring network traffic.

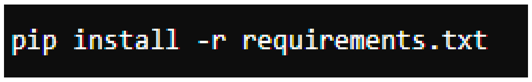
SOFTWARE REQUIREMENTS:

- Python 3.12 or later: Core programming language.

- Flask: Web framework for building the application.

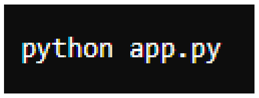
INSTALLATION:

Step 1: First, we must install the dependencies:



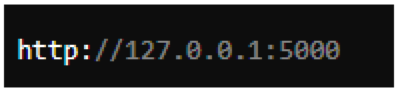
```
pip install -r requirements.txt
```

Step 2: we must run the application:



```
python app.py
```

Step 3: Usage, we must access this on



```
http://127.0.0.1:5000
```

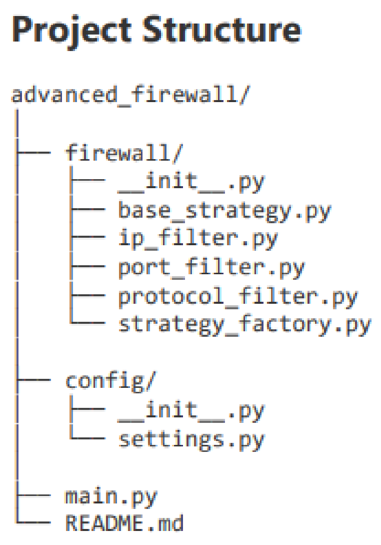
Logging In:

1. Use the default admin credentials:
   - **Username:** admin
   - **Password:** password

Navigating the Application:

2. Use the home page to manage rules, display lists, and check packets.

Project Structure:



Project Structure

```
advanced_firewall/
│
├── firewall/
│   ├── __init__.py
│   ├── base_strategy.py
│   ├── ip_filter.py
│   ├── port_filter.py
│   ├── protocol_filter.py
│   └── strategy_factory.py
│
├── config/
│   ├── __init__.py
│   └── settings.py
│
├── main.py
└── README.md
```

- **Firewall/**: Contains core application files.
  - `__init__.py`: Initialization file.
  - `strategy_factory.py`: Implements strategy and factory patterns.
  - `ip_filter.py`: Handles IP filtering logic.
  - `settings.py`: Configuration settings.
- **templates/**: HTML templates for the web interface.
  - `index.html`: Main page template.
  - `home.html`, `set_rules.html`, `display_lists.html`, `check_packet.html`: Specific page templates.
- **static/**: Static files like CSS for styling.
  - `style.css`: CSS stylesheets.
- **app.py**: Entry point for running the Flask application.
- **requirements.txt**: Lists all dependencies required for the project.

STEPS TO COMPLETE PROJECT:

Step 1: Set Up Configuration Settings (Singleton Pattern)

- **Purpose:** Ensure only one instance of the settings exists.
- **File:** config/settings.py

Step 2: Define the Base Strategy (Strategy Pattern)

- **Purpose:** Create an abstract base class for filtering network traffic.
- **File:** `firewall/base_strategy.py`

Step 3: Implement IP Filter Strategy

- **Purpose:** Filter network traffic based on IP addresses.
- **File:** `firewall/ip_filter.py`

Step 4: Implement Port Filter Strategy

- **Purpose:** Filter network traffic based on ports.
- **File:** `firewall/port_filter.py`

Step 5: Implement Protocol Filter Strategy

- **Purpose:** Filter network traffic based on protocols.
- **File:** `firewall/protocol_filter.py`

Step 6: Create the Strategy Factory (Factory Pattern)

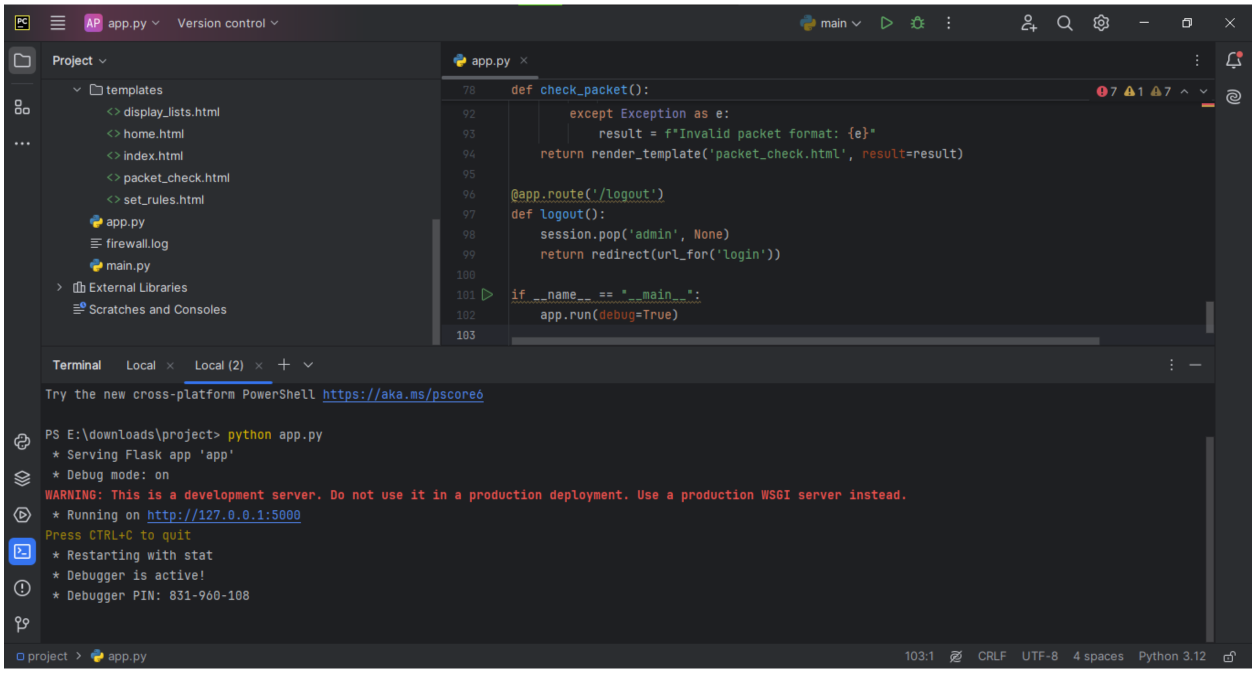
- **Purpose:** Create instances of different firewall strategies.
- **File:** `firewall/strategy_factory.py`

Step 7: Combine Everything in the Main File

- **Purpose:** Use the strategies for filtering network traffic.
- **File:** `main.py`

<u>WORKING:</u>

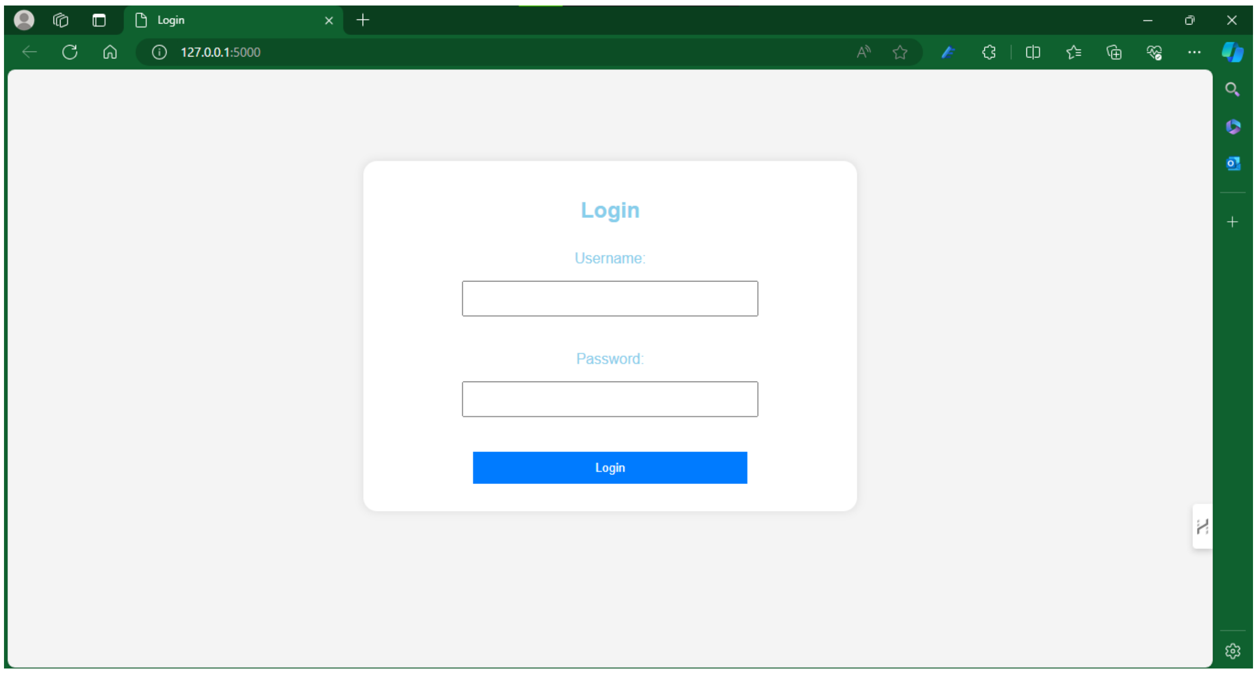
1. Firstly run the app.py file



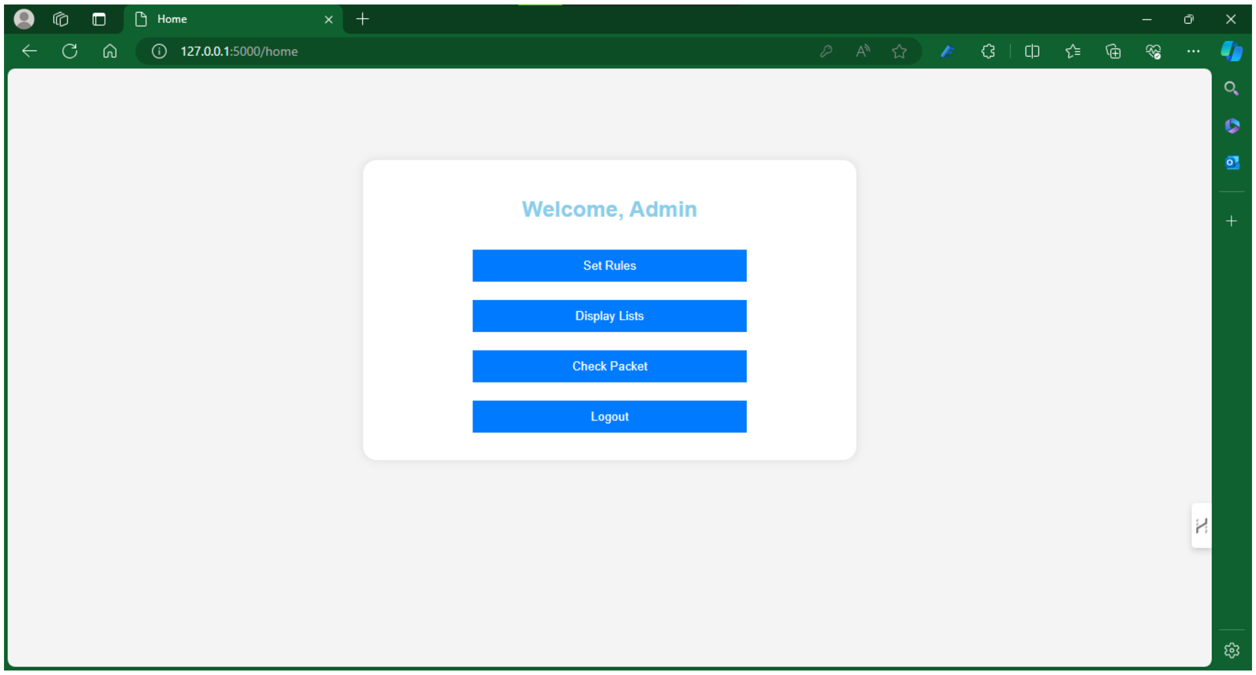
```
def check_packet():
        except Exception as e:
            result = f"Invalid packet format: {e}"
    return render_template('packet_check.html', result=result)

@app.route('/logout')
def logout():
    session.pop('admin', None)
    return redirect(url_for('login'))

if __name__ == "__main__":
    app.run(debug=True)
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\downloads\project> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 831-960-108
```

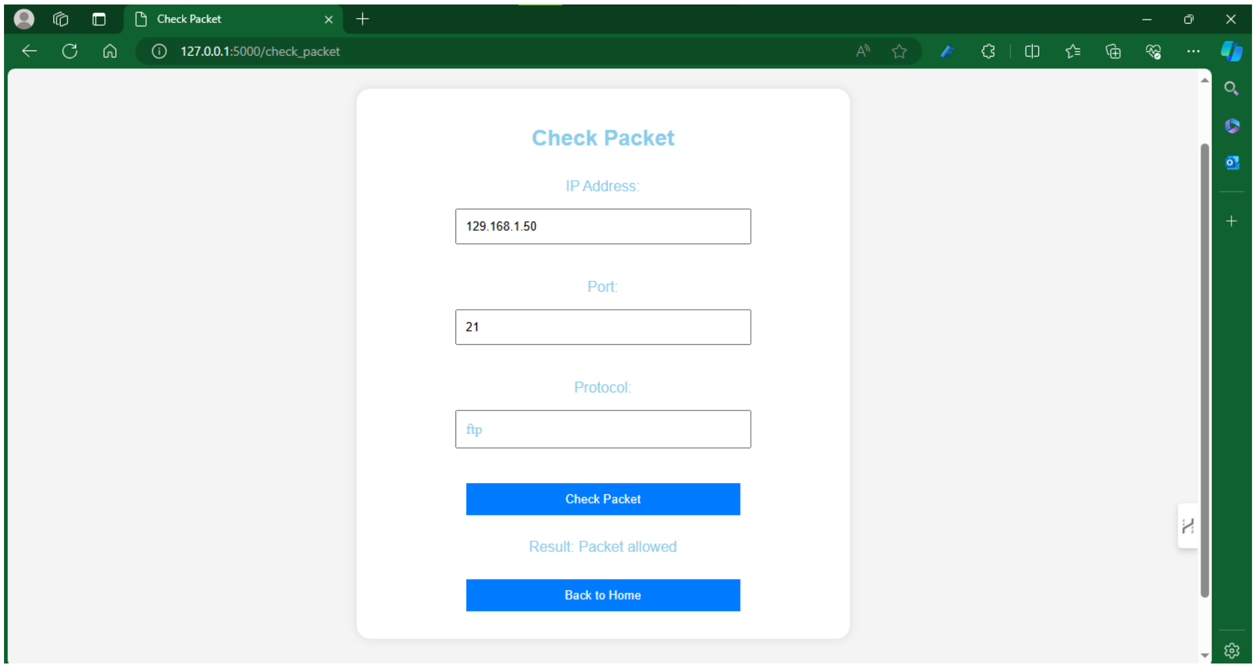
Step 2 click on Running on http://127.0.0.1:5000



Step 3 Login: username – admin password- password (you can do following things mentioned there)



Step 4: after setting rules and IP's we will check for the packets:



IP Address: 129.168.1.50

Port: 21

Protocol: ftp

Result: Packet allowed

Whitelisted packets will be allowed but blacklisted Ip's will not be allowed.