Google Form Automation with Selenium

Objective:
Automate the process of filling out a Google Form using Selenium WebDriver.

Overview:
This Python script automates the submission of a Google Form with predefined inputs using Selenium WebDriver. Each form field is populated programmatically with the specified data.

Key Features:

- **Form Automation:** Automates the entry and submission of data to a Google Form.
- **Selenium WebDriver:** Uses Selenium's Python bindings for browser automation.
- **Input Fields:** Populates various form fields including name, contact number, email address, address, pin code, date of birth, gender, and verification code.
- **Form Submission:** Submits the form after filling out all required fields.

Instructions for Use:

1. Ensure Python and Selenium WebDriver are installed in your environment.
2. Update the script with your Google Form URL and data inputs.
3. Run the Python script (`main.py`) to initiate the form filling process.
4. Monitor the browser as it automates the form submission.

Files Included:

- **main.py:** Python script containing the Selenium automation code.

Example Usage:

1. Open the Google Form URL specified in the script.
2. Wait for each form field to load and populate it with the corresponding data.
3. Submit the form and verify successful submission.

Future Enhancements:

- Implement error handling for field validation and submission errors.
- Add logging to track script execution and form submission status.

Sending Email with Attachment using Django

Objective:
Implement a Django application to send emails with file attachments programmatically.

Overview:
This Django project demonstrates how to send emails with attachments using Django's built-in email functionality. The application allows users to trigger an email sending action through a web interface.

Key Features:

- **Email Sending:** Utilizes Django's `EmailMessage` class to compose and send emails.
- **Attachment Support:** Demonstrates attaching files (e.g., screenshots, documents) to outgoing emails.
- **Integration with Gmail:** Configures Django settings to use Gmail SMTP for sending emails.
- **Web Interface:** Provides a simple web interface to trigger the email sending action.
- **Error Handling:** Includes basic error handling for email sending failures.

Instructions for Use:

1. **Set up Django Project:**
   - Configure `settings.py` with Gmail SMTP settings (`EMAIL_HOST`, `EMAIL_PORT`, `EMAIL_USE_TLS`, `EMAIL_HOST_USER`, `EMAIL_HOST_PASSWORD`).
2. **Implement Email Sending Logic:**
   - Create a Django view (`views.py`) to handle email composition and sending using `EmailMessage`.
   - Use Selenium to automate filling a Google Form or other web interactions.
3. **Design Web Interface:**
   - Develop a basic HTML template (`index.html`) with a button to trigger the email sending view (`send_test_email` function).
4. **Run the Development Server:**
   - Start the Django development server (`python manage.py runserver`) and navigate to `http://127.0.0.1:8000`.
5. **Send Email with Attachment:**
   - Click the "Send Email" button on the web interface to send an email with an attachment.
   - Monitor the terminal for email sending status and any error logs.

Files Included:

- **settings.py:** Django configuration file containing SMTP settings and other project configurations.
- **urls.py:** Django URL configuration file defining application routes.
- **views.py:** Django view file containing logic to send emails with attachments (`send_test_email` function).

- **index.html:** HTML template file providing a simple web interface to trigger email sending.
- **attachments/screenshot.png:** Example file to be attached to outgoing emails.

Example Usage:

1. Open the Django application URL (`http://127.0.0.1:8000`).
2. Click the "Send Email" button on the web interface.
3. Monitor the Django development server terminal for email sending status and any error messages.
4. Verify the recipient's email inbox for the sent email with attachment.

Future Enhancements:

- Implement advanced email functionalities such as HTML content, CC/BCC recipients, and email templates.
- Enhance user interface with feedback messages and loading indicators.
- Integrate file upload functionality to allow users to attach files dynamically from the web interface.