Setup Instructions and Dependencies

The source code of the bot is available at this GitHub repository:
https://github.com/Whyylek/Car-insurance-bot.git

Start by cloning the repository using Git. Open a terminal or command prompt and run the following command:
git clone https://github.com/Whyylek/Car-insurance-bot.git
Then navigate into the project directory by typing:
cd Car-insurance-bot

Make sure you have Python version 3.8 or higher installed. You can download it from the official website if needed. Also ensure that pip is installed, as it will be used to manage dependencies.

It is recommended to use a virtual environment to keep the dependencies for this project separate. Create one by running:
python -m venv venv
Then activate the virtual environment with:
On Windows: venv\Scripts\activate
On macOS or Linux: source venv/bin/activate

Install all required packages using the following command:
pip install -r requirements.txt

Create a new file named .env in the root directory of the project. This file will store your API keys and other sensitive information. Add the following lines to it:

TELEGRAM_BOT_TOKEN should be set to the token you received when creating your bot through BotFather on Telegram.

MINDEE_API_KEY should contain your Mindee API key which you can find in your Mindee account dashboard.

OPENAI_API_KEY should be your OpenAI API key, available in your OpenAI account settings.

Once the environment variables are set, you can start the bot by running the main Python script, usually named bot.py.

The bot will connect to Telegram and begin listening for incoming messages from users.

The bot is already deployed on Heroku and can be used directly without local setup. If you want to run your own version or make changes, you can clone the repository and follow the setup instructions to deploy it separately. When deploying your own instance, ensure that all API keys are kept secure and not shared publicly.

Detailed description of the bot workflow

The Telegram bot guides users through a structured process to help them complete the car insurance purchase. Here’s how it works step by step.

When a user starts the bot by sending the /start command or clicking the “Start” button, they receive a welcome message asking them to upload a photo of their passport. At this point, the bot sets the user's state to "awaiting_passport" to track where they are in the process.

Once the user uploads a valid photo of their passport, the bot sends the image to the Mindee API for data extraction. If successful, the system retrieves personal information such as the user's name and date of birth. The extracted details are shown to the user, along with options to confirm or re-upload the document.

If the user confirms the passport data, the bot moves on to the vehicle document stage. It asks the user to upload a clear photo of the front page of their vehicle registration certificate showing the license plate number. The system changes the user’s state to "awaiting_vehicle_doc_license_plate".

After receiving the first vehicle document, the bot again uses the Mindee API to extract vehicle data. Then, it asks the user to upload another photo that includes the VIN code and make of the vehicle. The user is now in the "awaiting_vehicle_doc_vin" state.

Once both vehicle documents are processed, the extracted data — including the license plate, VIN, make, and model — is presented to the user for confirmation. If confirmed, the bot proceeds to the price confirmation step.

At this point, the user sees a fixed insurance price of $100 and is asked to accept or decline it. If declined, the user is reminded that the price is fixed and asked again for confirmation. Once accepted, the system generates the final insurance policy document using OpenAI and creates a downloadable PDF file.

Finally, the PDF is sent to the user, and the process is complete. A message is displayed confirming that the insurance policy has been issued and can be reviewed in the attached file.

Throughout the interaction, the bot manages user states to ensure the correct flow of information and only allows valid actions at each step. This helps prevent errors and provides a smooth experience for the user.

Ось чистий текст для пункту **• Examples of interaction flows with the bot**, без вставок коду, зрозумілий і структурований:

Examples of interaction flows with the bot

Here is how a typical user interacts with the Telegram bot from start to finish.

1: Successful completion of the process

1. The user starts the bot by clicking the “Start” button.
2. The bot responds with a welcome message and asks the user to upload a photo of their passport.
3. The user sends a clear photo of their passport.
4. The bot processes the image and extracts personal data such as name and date of birth.
5. The user is asked to confirm that the extracted information is correct by clicking “Yes”.
6. After confirmation, the bot asks for a photo of the vehicle’s license plate.
7. The user uploads the photo and the bot extracts the vehicle registration number.
8. Next, the bot requests a photo showing the VIN code and make of the vehicle.
9. Once received, the bot gathers all the data and shows a summary of the vehicle details for confirmation.
10. The user confirms the details and is presented with the fixed insurance price of $100.
11. After accepting the price, the bot generates the insurance policy document.
12. The PDF file is sent to the user, completing the process.

2: User uploads incorrect or unclear document

1. The user starts the bot and uploads a blurry or incorrect photo instead of a passport.
2. The bot tries to extract data but fails due to poor image quality.
3. A message is sent back asking the user to re-upload a clearer photo of the required document.
4. The user then provides a better-quality image, and the process continues normally.

3: User rejects the extracted data

1. After uploading a document, the user sees the extracted data and clicks “No” to reject it.
2. The bot resets the current step and asks the user to re-upload the same type of document.
3. The user uploads a new photo, and the system processes it again.

4: User decides not to proceed after seeing the price

1. The user completes all previous steps and reaches the price confirmation stage.
2. They click “No” when asked to accept the $100 price.
3. The bot reminds them that the price is fixed and asks again if they would like to proceed.
4. If the user declines again, no further action is taken and the conversation remains paused unless the user responds again.

5: User sends text instead of a photo at the document upload step

1. The bot asks the user to upload a photo of their passport.
2. Instead of sending a photo, the user types and sends a message like “I’m ready” or sends a document file.
3. The bot detects that the received message is not a photo and ignores it.

4. An automatic reminder is sent to the user, asking them again to upload a clear photo of the required document.
5. The user then sends the correct photo, and the process continues as expected.