### ✅ Project Description: Python Typing Automation Application

### 💡 Project Overview

The **Typing Automation Application** is designed to automate repetitive typing tasks by capturing text from specific screen areas and typing it into a browser. It includes **browser selection, screen capturing, OCR (text recognition), and automated typing** with customizable tasks such as timed writing or lesson completion.

### 🚀 Features and Workflow

1. **First-time Setup**
   * Requests **system permissions** for screen capturing and automation.
   * Prompts the user to select their **preferred browser** (Chrome, Firefox, Edge, etc.).
   * Allows the user to **select and save screen coordinates** for:
     + Verification area (for initial login authentication).
     + Typing words area (where the target text appears).
   * Saves these coordinates to a **user\_settings.json** file in the data/ folder.
2. **Browser Automation**
   * Opens the selected browser.
   * Automates the **login process** using saved verification coordinates.
   * Ensures smooth navigation and stability using browser control modules.
3. **Text Recognition (OCR)**
   * Uses OCR (e.g., pytesseract) to **recognize and extract text** from the screen.
   * Captures the displayed typing words from the specified coordinates.
4. **Typing Execution**
   * Automates typing using **keyboard emulation** (pyautogui or pynput).
   * Handles error cases (e.g., text mismatches, special characters, or delays).
5. **Task Selection**
   * **Timed Writing:** Perform typing for a specific time duration.
   * **Lesson Completion:** Execute multiple typing lessons one by one until finished.

### 🛠️ File Structure and Description

typing\_automation\_app/  
│  
├── main.py # Entry point for the application  
├── config.py # Configuration and settings management  
├── data/ # Directory for saved data  
│ └── user\_settings.json # Saved coordinates and user preferences  
│  
├── ui/ # User interface components  
│ ├── \_\_init\_\_.py  
│ ├── app\_window.py # Main application window  
│ ├── browser\_selector.py # Browser selection interface  
│ ├── coordinate\_selector.py # Coordinate selection interface  
│ ├── task\_selector.py # Task selection (timed writing/lessons)  
│ └── notification.py # Notification system  
│  
├── core/ # Core functionality  
│ ├── \_\_init\_\_.py  
│ ├── screen\_capture.py # Screen capturing functionality  
│ ├── browser\_controller.py # Browser automation  
│ ├── text\_recognition.py # OCR for identifying text from screen  
│ ├── typing\_controller.py # Controls typing simulation  
│ ├── lesson\_manager.py # Manages lesson completion tasks  
│ └── timed\_writing.py # Manages timed writing tasks  
│  
└── utils/ # Utility functions  
 ├── \_\_init\_\_.py  
 ├── permission\_handler.py # Handles system permissions  
 ├── coordinate\_manager.py # Manages saved coordinates  
 └── logger.py # Logging functionality

### 🔥 Module Breakdown

#### ✅ **1. main.py**

* The entry point of the application.
* Initializes the UI, loads saved settings, and handles the main execution loop.

#### ⚙️ **2. config.py**

* Centralized configuration settings (e.g., paths, browser choices, and permissions).
* Easy to modify and extend.

#### 💾 **3. data/user\_settings.json**

* Stores **user preferences** and saved coordinates:

{  
 "browser": "chrome",  
 "verification\_coords": [100, 200, 300, 400],  
 "typing\_coords": [500, 600, 700, 800]  
}

#### 🎯 **4. UI Components (ui/)**

* **app\_window.py:** Main GUI window using Tkinter or PyQt5.
* **browser\_selector.py:** Dropdown selection for available browsers.
* **coordinate\_selector.py:** Allows users to select screen coordinates with mouse clicks.
* **task\_selector.py:** Menu for selecting between timed writing or lesson completion.
* **notification.py:** Displays status messages and errors to the user.

#### 🔥 **5. Core Functionality (core/)**

* **screen\_capture.py:**
  + Uses Pillow and PyAutoGUI to capture screenshots of selected areas.
* **browser\_controller.py:**
  + Automates browser operations using selenium or pyppeteer.
* **text\_recognition.py:**
  + Uses pytesseract for OCR-based text recognition.
* **typing\_controller.py:**
  + Automates typing using pyautogui or pynput.
* **lesson\_manager.py:**
  + Handles lesson navigation and completion tracking.
* **timed\_writing.py:**
  + Manages timed typing sessions with a countdown timer.

#### 🛠️ **6. Utilities (utils/)**

* **permission\_handler.py:**
  + Manages system permission requests.
* **coordinate\_manager.py:**
  + Saves and retrieves screen coordinates from user\_settings.json.
* **logger.py:**
  + Provides logging functionality for debugging and monitoring.

### 🔧 Tech Stack

* **Language:** Python
* **GUI:** Tkinter or PyQt5
* **OCR:** pytesseract
* **Automation:** pyautogui, pynput, selenium
* **Data storage:** JSON for configuration and coordinate management
* **Logging:** logging module

### ✅ Next Steps

1. **Environment Setup:** Install required libraries and dependencies.
2. **Build UI:** Create the GUI for browser selection, coordinate capture, and task selection.
3. **Implement Core Modules:**
   * Screen capturing, OCR, and typing automation.
4. **Integrate Logging and Config:**
   * Add logging and user settings handling.
5. **Test and Optimize:**
   * Validate OCR accuracy and typing precision.