ADJUFACE TELEGRAM BOT

by Dmitrii Didenko

Table of Contents

1. Introduction

1.1. Project Overview

The Adjuface Telegram Bot is a powerful tool designed to provide users with the ability to create entertaining and creative content by swapping faces in images. This bot is integrated into the Telegram messaging platform, making it easily accessible to a wide range of users.

1.1.1. Key Features:

- **Face Swapping:** Users can upload their images and select from a menu of available images to swap faces with. This feature allows for fun and imaginative image transformations.
- **Premium User Benefits:** Premium users enjoy additional privileges, including the ability to upload their custom target images. These custom target images can later be used for face swapping with other images.

1.1.2. Target Audience:

- The primary target audience for the Adjuface Telegram Bot includes individuals who enjoy creating humorous or artistic content by manipulating images. It caters to users who seek a user-friendly and engaging platform for face swapping.

1.1.3. Project Significance:

- **Entertainment**: The bot adds a fun and creative dimension to image editing, allowing users to experiment with face swapping and generate humorous or artistic content.
- **Engagement**: By integrating with Telegram, the bot leverages the popularity of the messaging platform to engage a broad user base.
- **Customization**: Premium users have the flexibility to upload their custom target images, enhancing their ability to create unique face-swapped content.

The Adjuface Telegram Bot aims to provide an enjoyable and interactive experience for its users, making image manipulation accessible to both novice and experienced creators.

1.2. Purpose of the Document

The purpose of this document is to provide a comprehensive overview of the system design for the Adjuface Telegram Bot. It serves as a reference guide for all possible stakeholders (I am open to cooperation) who want to get involved in the development, maintenance, and enhancement of the bot. This document aims to achieve the following objectives:

1.2.1. System Understanding

- **Clarification of System Functionality**: To offer a clear and detailed understanding of how the Adjuface Telegram Bot operates, including its key features and components.
- **Scope Definition**: To define the boundaries of the system, specifying what is within and outside the scope of the bot's functionality.

1.2.2. Design Guidelines

- **Architectural Design**: To describe the system's architecture, including the components, their interactions, and data flow. This section will outline the design principles and patterns used in developing the bot.
- **User Interface Design**: To provide insights into the user interface (UI) design, including screen layouts, navigation, and user interactions.

1.2.3. Development and Maintenance

- **Development Guidelines**: To provide a glimpse into details of implementing and enhancing the Adjuface Telegram Bot by detailing coding standards, best practices, and development tools.
- **Maintenance Procedures**: To establish procedures for system maintenance, including updates, bug fixes, and scalability considerations.

1.2.4. User Experience

- **User Experience Enhancement**: To ensure that the bot delivers an optimal user experience, focusing on responsiveness, reliability, and ease of use.

1.2.5. Project Transparency

- **Transparency and Documentation**: To maintain a transparent record of the system design decisions, which can be referenced throughout the project's lifecycle.

This document is intended to serve as a valuable resource for the development team, quality assurance professionals, and project managers, ensuring a shared understanding of the Adjuface Telegram Bot's design and functionality. It will also assist in aligning development efforts with project objectives, ultimately contributing to the successful deployment and maintenance of the bot.

1.3. Scope

1.3.1. Development Ownership

- The development, maintenance, and ownership of the Adjuface Telegram Bot are solely the responsibility of Dmitrii Didenko.
- Dmitrii Didenko is responsible for the entire software development life cycle, including design, coding, testing, deployment, and ongoing maintenance.
- The ml models and training dataset are a property of deepinsight insightface team under a MIT license.

1.3.2. Bot Functionality

- The bot allows users to upload their images and select from a predefined menu of available images for face swapping.
- Premium users have the additional capability to upload custom target images, which can be used for face swapping with other images.

1.3.3. Target Audience

- The target audience for the Adjuface Telegram Bot includes Telegram users who wish to utilize the face-swapping functionality.
- Premium users represent a specific segment of the audience with enhanced features and capabilities.

1.3.4. Development Environment

- Dmitrii Didenko will maintain the development environment and choose the necessary tools, libraries, and technologies for bot development.
- The development environment is not limited to a specific platform or technology stack and can be adapted as needed.

This scope statement outlines the boundaries and responsibilities associated with the Adjuface Telegram Bot project. Dmitrii Didenko has full ownership and control over the development process and the bot's functionality allowed by the licenses of projects parts . The scope encompasses both standard and premium user features, with flexibility in the development environment.

2. Architectural Overview

2.1. Architecture Components and Modules

The Adjuface Telegram Bot is a multifunctional application with a modular architecture that encompasses FastAPI web service, AI-powered image processing, and database interaction. This section provides an overview of the primary components and their interactions.

2.1.1. Main Entry Point

- **main.py**: The central entry point of the application where the main execution begins. It orchestrates the interaction between the AI-powered bot and the database.

2.1.2. Telegram Bot

- **bot/message_handler.py**: This module is responsible for handling Telegram bot functionality using the aiogram library. It receives and processes user messages, manages user interactions, and communicates with the database.

2.1.3. AI-Powered Image Processing

- **face_carver/swapper.py**: The AI-powered image processing component handles the core functionality of the bot, which includes face swapping and image manipulation. It operates independently of the bot and communicates with the database and the FastAPI service.

2.1.2. FastAPI Web Services

- **FastAPI for face_carver/swapper.py**: Another FastAPI web service responsible for processing image-related requests and serving as an interface for the AI component. It facilitates communication between the AI model and the main application.

2.1.3. Database Interaction

- **user_database.db**: The SQLite database where user-related data is stored. The bot and AI components interact with this database using bot/db_requests.py to retrieve user details and image filenames.

2.1.4. Image Storage

- **temp Folder**: Located in the root directory, this folder contains subdirectories for storing original images, target images, and results. Images are retrieved by name from this folder based on requests from the bot and AI component.

2.1.5. Configuration Files

- **bot/contacts.yaml**: A YAML configuration file containing contact information, including Telegram usernames and other contact details.
- **target_images.json**: A JSON configuration file that defines modes, names, and paths to target images used for face swapping.
- **config.yaml**: A configuration file storing the Telegram bot token.

The Adjuface Telegram Bot is designed as a modular and flexible system, with clearly defined components responsible for distinct tasks. This architecture enables smooth communication between the Telegram bot, AI image processing, and the underlying database. Additionally, configuration files store essential parameters for bot operation and customization.

2.2. Data Flow

The data flow within the Adjuface Telegram Bot involves the exchange of messages, images, and user-related data between various components. This section illustrates how data moves through the system during different user interactions.

2.2.1. Message Processing

2.2.1.1. User Sends a Message:

- A user initiates a conversation with the Adjuface bot on the Telegram platform by sending a message.

2.2.1.2. Message Analysis:

- The Telegram bot component (**bot/message_handler.py**) receives the user's message.
- The type of message is analyzed to determine its content.

2.2.1.3. Content Validation:

- If the message contains anything other than an image, the bot responds with a message indicating that the data should not be processed.
- Multiple images in a single message are not allowed.

2.2.2. User Data and Limits

2.2.2.1. User Data Retrieval:

- The Telegram bot queries the database using the **bot/db_requests.py** module to retrieve user data based on their Telegram account credentials.
- User details, messages sent, and images sent and received are logged

2.2.2.2. Limit Checks:

- The bot checks if the user has run out of processing limits (e.g., requests left).
- If the user has exceeded their limits, they are informed accordingly.

2.2.2.3. Flag Interpretation:

- If the message is a target image (flag set), the image is processed as a target for face swapping.
- If the message is a source image (flag not set), it is used as a source image for face extraction.

2.2.3. Image Processing

2.2.3.1. Processing:

For Target Images:

- Only one face on the target image is allowed.
- The image is processed to prepare it as a target for face swapping.

For Source Images:

- Multiple faces are detected, and each face is processed separately.
- The AI-powered image processing component (**face_carver/swapper.py**) is responsible for image manipulation tasks.
- Face swapping and carving are performed using machine learning models.

2.2.3.2. Result Images:

- The AI component generates result images corresponding to the number of faces detected in the source image.
- Result images are temporarily stored in the temp folder under the appropriate subdirectory (e.g., "**results**").

2.2.4. Result Delivery

2.2.4.1. Sending Result Images:

- The Telegram bot sends the processed result images back to the user.

2.2.5. Logging

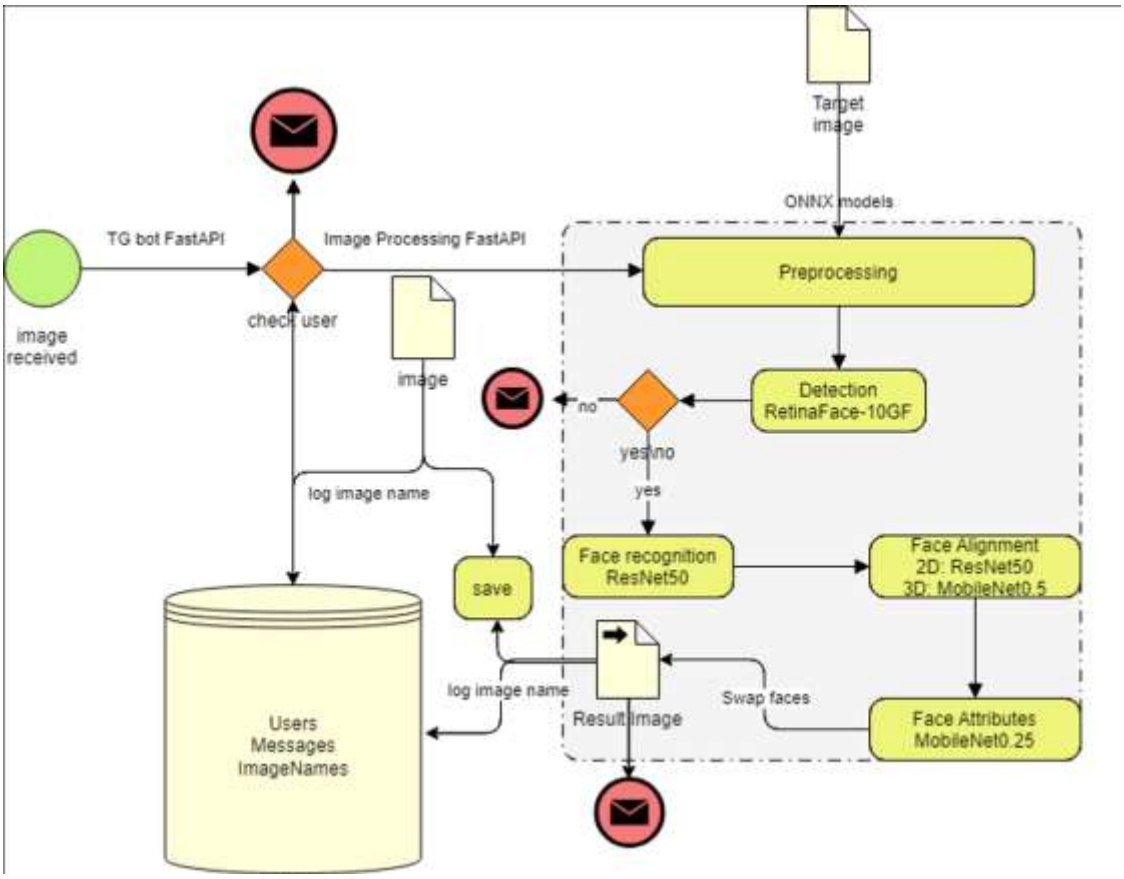
- User data, including the Telegram account credentials, messages sent, and images sent and received, is logged for future reference.

2.2.6. Updating Limits

- After processing, the bot updates the user's limits (e.g., reducing the number of requests left).

2.2.7. BPMN Diagram

The data flow of the Adjuface Telegram Bot is designed to ensure efficient processing of user requests while maintaining appropriate limits and data tracking. User interactions, image processing, and result delivery are all orchestrated to provide a seamless experience.



3. System Requirements

The Adjuface Telegram Bot is a complex system with various components and dependencies. It's essential to define both functional and non-functional requirements to ensure its proper operation and performance.

3.1. Functional Requirements

Functional requirements specify what the system should do. Here are the updated functional requirements for the Adjuface Telegram Bot:

3.1.1. User Interaction:

- The bot must accept messages and images from users on the Telegram platform.
- Users can send both source and target images for face swapping.

3.1.2. Message Analysis:

- The bot should be able to analyze incoming messages to determine their content, particularly whether they contain images.

3.1.3. Image Processing:

- The bot should process source images to detect faces.
- Face swapping and carving should be performed using lightweight machine learning models (**MobileNet**s and **ResNet**).

3.1.4. Result Delivery:

- Processed result images must be sent back to the user.

3.1.5. User Data Logging:

- The bot should log user data, including Telegram account details, messages sent, and images sent and received.

3.1.6. Limit Enforcement:

- Limits on the number of requests and custom targets for users must be enforced.
- Users can request information about their remaining limits.

3.1.7. Donation Support:

- Users can support the developer by using the **/donate** command to receive donation instructions, including a Bitcoin (BTC) address for contributions.

3.1.8. Premium Account (To Be Implemented):

- Users can pay for a premium account via Telegram (future implementation), granting access to custom targets and increased image limits.

3.1.9. Time Limit Between Images:

- Users are subject to a time limit of approximately 30 seconds between sending images for processing.

3.2. Non-Functional Requirements

Non-functional requirements define the qualities and characteristics of the system. Here are the updated non-functional requirements for the Adjuface Telegram Bot:

3.2.1. Performance:

- The system should provide responsive and timely image processing, typically taking **5-20 seconds** per image **on CPU**.
- Lightweight machine learning models ensure fast processing even on CPU.

3.2.2. Scalability:

- The architecture should allow for horizontal scalability to handle increased user traffic.

3.2.3. Reliability:

- The bot should be available and reliable, minimizing downtime.

3.2.4. Security:

- User data, including Telegram credentials, must be securely stored and protected.
- Measures should be in place to prevent misuse or abuse of the system.

3.2.5. Privacy:

- User privacy should be maintained, and processed images should not be stored or shared without user consent.

3.2.6. Error Handling:

- The bot should gracefully handle errors, providing informative error messages to users.

3.2.7. Logging and Monitoring:

- Comprehensive logging and monitoring should be implemented to track system behavior and performance.

3.2.8. Maintenance and Updates:

- The system should be maintainable, allowing for updates, bug fixes, and improvements.

3.3. Use Cases

Use cases represent specific interactions or scenarios that users can perform with the system. Here are some key use cases for the Adjuface Telegram Bot:

3.3.1. User Registration:

- Users can register with the bot by sending any message or pressing /start – user data will be logged into the database.

3.3.2. Sending Source Images:

- Users can send source images to the bot for face extraction and swapping.

3.3.3. Sending Target Images:

- Premium users can upload custom target images for face swapping.

3.3.4. Requesting Face Swapping:

- Users can request face swapping using source and target images.

3.3.5. Viewing Limits:

- Users can inquire about their remaining requests, custom target limits, and the time limit between images.

3.3.6. Support Request:

- Users can request support from the administrator.

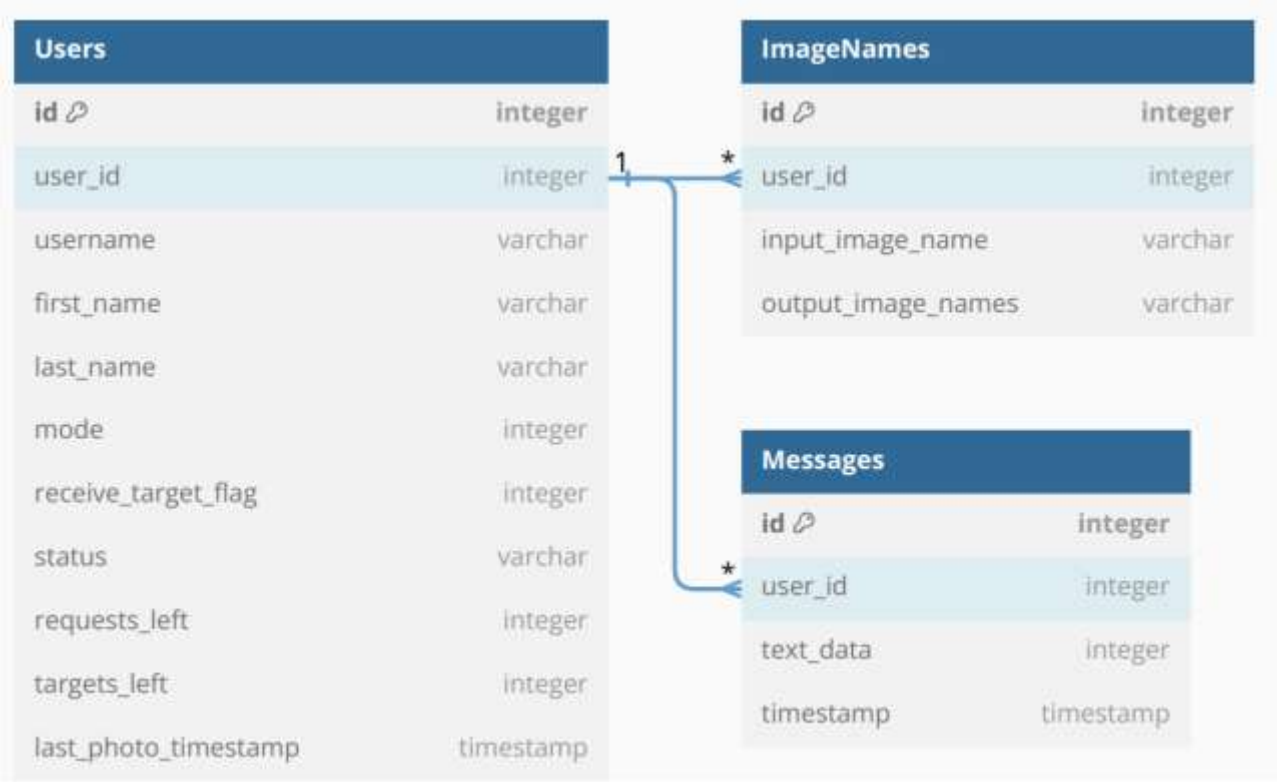
3.3.7. Contacting the Administrator:

- Users can view the administrator's contact information.

4. Data Model

4.1. Entity-Relationship Diagram

Here is the Entity-Relationship Diagram (ERD) representing the data model of the Adjuface Telegram Bot:



4.2. Database Schema

The database schema for the Adjuface Telegram Bot is defined using the following tables:

4.2.1. Users

- **id** (**Primary Key**): Unique identifier for each user.
- **user_id**: Telegram user ID, also unique.
- **username**: Telegram username.
- **first_name**: First name of the user.
- **last_name**: Last name of the user.
- **mode**: User's mode (**default**: 1).
- **receive_target_flag**: Flag to indicate if the user can receive target images (**default**: 0).
- **status**: User's status (default: 'free').
- **requests_left**: Number of requests left for the user (**default**: 10).
- **targets_left**: Number of custom targets left for the user (**default**: 0).
- **last_photo_timestamp**: Timestamp of the last photo sent (**default**: current timestamp).

4.2.2. Messages

- **id** (**Primary Key**): Unique identifier for each message.
- **user_id** (**Foreign Key**): References the user_id in the Users table.
- **text_data**: Text data associated with the message.
- **timestamp**: Timestamp of when the message was sent (default: current timestamp).

4.2.3. ImageNames

- **id** (**Primary Key**): Unique identifier for each image name entry.
- **user_id** (**Foreign Key**): References the user_id in the Users table.
- **input_image_name**: Name of the input image.
- **output_image_names**: Comma-separated list of names of output images.

4.3. Data Flow

The data flow in the Adjuface Telegram Bot can be summarized as follows:

1) Users interact with the Telegram bot named "Adjuface" by sending messages and images.
2) Messages and images sent by users are received by the bot's message handler.
3) The message handler analyzes the type of message and performs appropriate actions:
   - If the message is not an image or contains multiple images, a response is sent indicating that the data should not be processed.
   - If the message is an image, the bot checks if the user has exceeded limits and if it's a target image or source image.
   - If it's a target image, only one face can be placed on it.
   - If it's a source image, the number of result images generated is equal to the number of detected faces.
4) The bot logs user data, including Telegram account credentials, messages sent, images sent, received, and output, as well as usage limits.
5) Users can support the bot by using the **/donate** command to receive a donation link for Bitcoin (BTC).
6) Users can also pay for a premium account via Telegram (feature to be implemented) to increase image limits and use custom target images.
7) There is a time limit of approximately 30 seconds between processing images.
8) This data flow ensures that users can interact with the bot to perform face swapping and image processing while maintaining data security and usage limits.

5. Infrastructure and Deployment

5.1. Hardware Requirements

The Adjuface Telegram Bot is designed to be resource-efficient and can be run on a variety of hardware configurations:

- Development Environment (RedmiBook Laptop):
  - Operating System: Windows, Linux, or macOS.
  - CPU: Any modern multi-core CPU (e.g., Intel Core i5 or equivalent).
  - RAM: 8GB or more.
  - Internet Connectivity: Stable and reliable internet connectivity for development and testing purposes.
- Production Environment (Old Laptop with GPU):
  - Operating System: Linux is recommended for server deployments.
  - CPU: Any multi-core CPU.
  - GPU (Optional): An old laptop with a GPU that supports CUDA (e.g., NVIDIA GeForce) can significantly accelerate image processing tasks using CUDA-compatible machine learning libraries.
  - RAM: The recommended RAM depends on the number of concurrent users and the complexity of image processing. A minimum of 8GB is suggested for small-scale deployments.
  - Internet Connectivity: Stable and reliable internet connectivity for communication with Telegram servers and external services.

5.2. Software Requirements

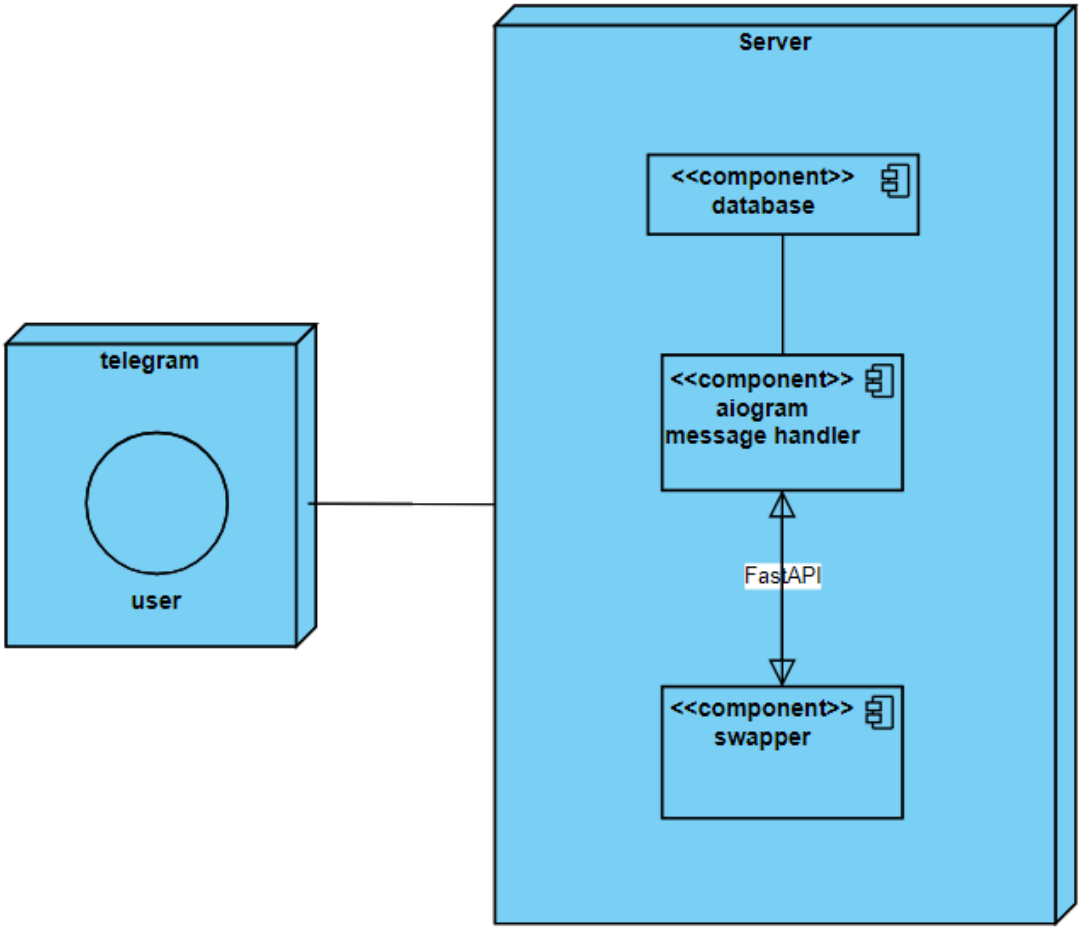
The software requirements for the Adjuface Telegram Bot include:

- Operating System:
  - For development, the bot can run on Windows, Linux, or macOS.
  - For production, Linux is recommended for server deployments due to stability and resource optimization.
- Python:
  - Python 3.x is required to execute the bot's code.
- Required Python Libraries:

    - The bot relies on several Python libraries, including aiogram, SQLAlchemy, FastAPI, and others. These libraries should be installed using pip.
- SQLite Database:
    - The bot uses an SQLite database (**user_database.db**) for data storage.

5.3. Deployment Diagram

The deployment of the Adjuface Telegram Bot involves two main components: the Aiogram Bot and the FastAPI Server.



5.3.1. Alogram Bot Component

- The Alogram Bot component handles user interactions, including receiving messages, processing images, and sending responses.
- It communicates with the SQLite database to store and retrieve user data.
- The bot is deployed on a development machine (RedmiBook Laptop) during the development and testing phase.

5.3.2. FastAPI Server Component

- The FastAPI Server component is responsible for serving the machine learning models used for image processing.
- It communicates with the Alogram Bot to receive image requests and return processed images.
- For production deployment, the server can be hosted on an old laptop with GPU support to accelerate image processing tasks using CUDA.

5.4. Scalability Considerations

5.4.1. Overall Scalability

Scalability for the Adjuface Telegram Bot can be achieved through the following considerations:

1. **Load Balancing**: To handle increased user load, multiple instances of the Alogram Bot and FastAPI Server can be deployed behind a load balancer to distribute incoming requests evenly.
2. **Vertical Scaling**: The server hosting the bot can be vertically scaled by upgrading the hardware, increasing RAM, and optimizing code for better performance.
3. **Caching**: Implementing caching mechanisms for frequently used data or processed images can reduce the load on the server and improve response times.
4. **Optimized Image Processing**: Continuously optimizing image processing algorithms and using lightweight machine learning models can reduce processing time and resource requirements.
5. **Database Scaling**: As user data grows, the database may require scaling solutions, such as migrating to a more robust database system or using database sharding.

It's important to monitor bot usage and performance regularly to identify scalability needs and ensure a smooth user experience.

The Adjuface Telegram Bot is designed to be lightweight and efficient, making it suitable for both small-scale deployments and potential future scalability.

5.4.2. Database Optimization

Optimizing the database plays a crucial role in improving the performance and responsiveness of the Adjuface Telegram Bot. Currently, the bot uses SQLite with async engine aiosqlite as the database system, which can be suboptimal for production-scale applications. Transitioning to a more robust database system, such as MySQL with asynchronous support, can significantly enhance database performance.

5.4.2.1. MySQL Database

MySQL is a widely used and reliable relational database management system. When combined with asynchronous support, it can efficiently handle concurrent requests from the bot, leading to improved responsiveness. Below are key aspects of optimizing the database:

- **Asynchronous Support**: By utilizing an asynchronous MySQL driver, the bot can execute multiple database queries concurrently without blocking the event loop. This leads to faster response times and better scalability.
- **Connection Pooling**: Implementing connection pooling allows the bot to reuse existing database connections, reducing the overhead of establishing a new connection for each query. This helps in managing database connections efficiently, especially during periods of high traffic.
- **Indexing**: Properly indexing database tables can significantly speed up query execution. Analyze the most frequently used queries and create appropriate indexes on columns used in WHERE clauses and JOIN operations.
- **Normalization**: Ensuring the database schema is properly normalized can improve data integrity and reduce redundancy. This can lead to more efficient storage and faster query performance. Currently Sqlite is close to **3NF**.
- **Query Optimization**: Carefully review and optimize SQL queries. Avoid using complex or unoptimized queries that can cause a significant performance bottleneck.

5.4.3. Potential Optimizations

Here are some potential optimizations specific to the Adjuface Telegram Bot's database usage:

- **Batch Inserts and Updates**: When inserting or updating multiple records, consider using batch operations to minimize the number of individual database queries.
- **Caching**: Implement caching mechanisms for frequently accessed data, such as user profiles, to reduce the number of database queries.
- **Database Sharding**: If the bot experiences rapid growth in user data, consider database sharding to distribute the data across multiple database instances, improving scalability.

- **Regular Maintenance**: Schedule regular database maintenance tasks, such as vacuuming and indexing, to keep the database in optimal condition.
- **Database Monitoring**: Utilize database monitoring tools to identify slow queries and bottlenecks, allowing for timely optimization.

6. User Interface Design

6.1. User Interface Mockups

The user interface of the Adjuface Telegram Bot is designed to be intuitive and user-friendly. It primarily operates within the Telegram app's chat interface, making it accessible to a wide range of users. Below are mockups of the user interface to illustrate its key features:

6.1.1. Category Selection

- When users select a *category*, a list of image names within that category is displayed as buttons.
- Users can choose an image name or return to the main menu.

6.1.2. Collage Display

- After selecting an image name, a *collage* of available images within that category is sent to the user.
- Users can then choose a specific image for further processing.

6.2. User Experience (UX) Considerations

6.2.1. Intuitive Commands

- The bot's commands are designed to be self-explanatory and easy to use. Users can access various functions by simply typing **"/help**" in the chat or pressing a button "**Menu**" in the bottom left corner of chat.

6.2.2. Clear Instructions

- The bot provides clear instructions to users, ensuring they understand its capabilities and how to interact with it effectively.

6.2.3. Dynamic Menu

- The **menu buttons** to choose targets dynamically changes based on user interactions, making it responsive and user-centric.

6.2.4. Error Handling

- The bot includes error handling to guide users when they input incorrect commands or send unsupported file types.

6.2.5. Premium Features

- Users are informed about premium features, such as adding custom target images, and how to access them through commands.

6.2.6. Support and Donation

- The bot offers options for users to seek support or make donations, enhancing user engagement and satisfaction.

The user interface of the Adjuface Telegram Bot is designed to provide a seamless and enjoyable experience for users, whether they are selecting images, processing photos, or accessing additional features. It aims to prioritize clarity, simplicity, and responsiveness to meet user expectations.

7. API Documentation

7.1. API Endpoints

The Adjuface project includes two main components with their respective API endpoints:

7.1.1. Telegram Bot API

Endpoint: Telegram Messenger

- **Description**: Users interact with the bot using Telegram's chat interface. The bot receives user messages, processes images, and responds with results and messages.
- **Authentication**: Telegram user authentication is used for each interaction with the bot.

7.1.2. FastAPI (Face Swap) API

- **Endpoint**: /swapper
- **Description**: This API is responsible for performing face swaps on images. It utilizes FastAPI and is hosted on a separate endpoint.
- **Authentication**: No specific authentication is required for using this API.

7.2. Request and Response Formats

7.2.1. Telegram Bot API

- **Requests**: Users send messages and images to the bot. Message content and image files are included in the requests.
- **Responses**: The bot responds with messages and images. Images are typically in JPEG or PNG format.

7.2.2. FastAPI (Face Swap) API

- **Requests**: Users can send POST requests to the /swapper endpoint with an image file for processing.
- **Responses**: The API responds with the processed image in JPEG or PNG format.

7.3. Authentication and Authorization

7.3.1. Telegram Bot API

- **Authentication**: Telegram user authentication is managed by the Telegram platform itself. Users must have a Telegram account and interact with the bot through the Telegram app.
- **Authorization**: The bot authorizes users to perform actions based on their account status (free or premium) and usage limits.

7.4. FastAPI (Face Swap) API

- **Authentication**: Currently, no specific authentication mechanism is implemented for the FastAPI API. It is intended to be accessible for general use.
- **Authorization**: Access to the /swapper endpoint does not require user-specific authorization. The endpoint is designed for face swap processing and is open for use.

Users can run the Adjuface project on their local machines by providing their own ONNX model, bot token in **config.yaml**, their own **target_images.json**, and **contacts.yaml.** The FastAPI component for face swapping can be accessed locally using uvicorn. There is no specific authentication required for local usage.

8. Testing and Quality Assurance

8.1. Testing Plan

The Adjuface project employs a comprehensive testing plan to ensure the reliability and correctness of its components. The testing plan covers both unit testing and manual testing for the Telegram bot component.

8.1.1. Unit Testing

db_requests.py

- **Objective**: To test the database interaction functions.
- **Test Cases**: Includes tests for inserting, updating, and retrieving user data from the database.
- **Testing Framework**: **unittest** and **pytest**.

8.1.2. Manual Testing (Telegram Bot)

- **Objective**: To test the functionality and user experience of the Telegram bot.
- **Test Cases**: Various test cases are manually executed, covering bot commands, image processing, and edge cases.
- **Testing Approach**: Manual testing is performed by interacting with the bot on the Telegram platform.
- **Test Environment**: Telegram Messenger app.

8.2. Test Cases

8.2.1. Autotests

test_db_requests.py

1) Test inserting user data into the database.
2) Test updating user data (e.g., mode, status, requests left) in the database.
3) Test retrieving user data from the database.
4) Test edge cases for database interactions (e.g., non-existing users).

Telegram Bot (Manual Testing)

1) Test /start command to initiate bot interaction.

2) Test /help command to display the help message.
3) Test /status command to check account limits.
4) Test /select command to select a category of pictures.
5) Test /reset_user (debug) command to reset the user's status.
6) Test /buy_premium command to add premium features.
7) Test /custom_target command to add custom target images.
8) Test /contacts command to display the contacts list.
9) Test /support command to send a support request.
10) Test /donate command to support the developer.
11) Test sending an image to the bot for processing.
12) Test sending non-image content and receiving an error message.
13) Test image processing for various scenarios (e.g., valid image, multiple faces, no faces).
14) Test edge cases for account limits (e.g., exceeding image limits).

8.2.2. Telegram Bot - Manual Testing

8.2.2.1. Batch Processing of Images

- Test Batch Image Processing: Send a batch of images (more than one) to the bot and verify that each image is processed separately. Check if the correct number of result images is returned.
- Test Batch Image Limits: Send a batch of images exceeding the user's limits and verify that the bot correctly stops processing after reaching the limit.

8.2.2.2. Handling Forwarded Images

- Test Forwarded Images with Hidden Sender Name: Simulate a forwarded image where the sender's name is hidden, making it pass the media group check. Verify that the bot correctly identifies such cases and processes only one image.
- Test Forwarded Images with Multiple Faces: Forward an image with multiple faces to the bot and check that it processes each face separately and returns the correct number of result images.

8.2.2.3. Multiple Faces Processing Logic

- Test Multiple Faces within User Limits: Send an image with multiple faces within the user's image limits and verify that all faces are processed, and the correct number of result images is returned.
- Test Multiple Faces Exceeding User Limits: Send an image with a large number of faces exceeding the user's limits and verify that the bot stops processing after reaching the limit.
- Test Limits Check Before Sending: Ensure that the bot checks the user's limits before sending any processed images. If a user has run out of limits during processing, no images should be sent.

These test cases cover scenarios related to batch processing, handling forwarded images, and the logic for processing multiple faces while ensuring users stay within their defined limits.

8.3. Quality Assurance Measures

- **Code Reviews**: All code changes are reviewed by the project owner (Dmitrii Didenko) for quality and adherence to coding standards since it's his own code he's reviewing after all.
- **Documentation**: The project includes thorough code documentation and system design documentation to enhance code quality and maintainability.
- **Logging**: Logging is implemented to track errors and events for debugging and monitoring.
- **Security Audits**: Regular security audits are conducted to identify and address vulnerabilities.
- **User Feedback**: User feedback and bug reports are actively collected and addressed to improve the user experience.
- **Performance Optimization**: Continuous optimization efforts are made to improve response times and resource utilization.
- **Error Handling**: Comprehensive error handling is implemented to provide informative error messages and handle exceptions gracefully.

The combination of unit testing, manual testing, and quality assurance measures ensures the Adjuface project maintains a high level of reliability and quality.

9. Performance Optimization

9.1. Performance Metrics

To optimize the performance of the Adjuface Telegram Bot, we need to consider the following performance metrics:

1) **Request Response Time**: Measure the time taken for the bot to respond to user requests, including image processing time.
2) **Database Query Execution Time**: Monitor the execution time of database queries to identify slow queries.
3) **Server CPU and Memory Usage**: Track CPU and memory utilization to ensure efficient resource allocation.
4) **Error Rate**: Measure the rate of errors and exceptions in the application.

9.2. Bottlenecks and Optimization Strategies

9.2.1. Database Query Optimization

- **Async Database Queries**: Implement asynchronous database queries using SQLAlchemy's async capabilities to improve query concurrency.
- **Query Batching**: Batch multiple queries into a single database transaction to reduce the overhead of multiple transactions.
- **Indexing**: Ensure appropriate indexing on database tables to speed up query execution.
- **Caching:** Implement caching mechanisms to store frequently accessed data in memory and reduce database load.

9.2.2. Logging Optimization

- **Log Level Configuration**: Correct the log level configuration to ensure that only relevant log messages are displayed, reducing unnecessary console clutter.
- **Error Logging:** Implement error logging to capture and log exceptions and errors for debugging and monitoring purposes.

9.2.3. Asynchronous Processing

- **Async Image Processing**: Optimize image processing functions to run asynchronously, reducing the time taken for image transformations.
- **Parallelism**: Utilize parallel processing techniques to handle multiple requests concurrently, improving overall system throughput.

9.3. Caching and Load Balancing

- **Result Caching**: Implement result caching for processed images to avoid reprocessing images that have been requested recently.
- **Load Balancing:** If deploying on multiple servers, use load balancing to distribute incoming requests evenly across servers, ensuring efficient resource utilization.

By addressing these optimization strategies, we aim to enhance the performance and responsiveness of the Adjuface Telegram Bot while minimizing resource usage and improving the user experience.

10. Security

10.1. Threat Analysis

The Adjuface Telegram Bot is designed to process user-submitted images and interact with external APIs. While the bot's architecture is relatively simple, there are some potential security considerations:

1) **Malicious Content**: The bot should be protected against malicious content in images or messages, which could include malware or inappropriate content.
2) **Rate Limiting**: To prevent abuse, rate limiting should be implemented to restrict the number of requests a user can make in a given time period.
3) **Authentication**: Ensure that only authorized users can access premium features or upload custom target images.

10.2. Authentication and Authorization

- **User Authentication**: Users are identified and authenticated using their Telegram accounts. No additional authentication is required.
- **Authorization:** Authorization is used to differentiate between free and premium users. Premium users have access to additional features, such as custom target images and increased image limits.

10.3. Encryption and Data Security

- **Data Transmission**: Communication between users and the bot is encrypted using Telegram's secure protocol.
- **Database Security**: Ensure that the database is properly secured, and sensitive user data, such as Telegram IDs, is stored securely.

10.4. Compliance with Security Standards

- **Licensing**: If considering commercial use, ensure compliance with licensing guidelines, especially if using third-party libraries.
- **Business Registration**: When transitioning to a commercial venture, adhere to local business registration requirements and regulations.
- **Financial Data Security**: If handling financial transactions (e.g., donations or premium subscriptions), implement secure payment gateways and adhere to relevant data security standards.
- **Tax Compliance**: Comply with tax regulations when receiving donations or generating revenue from premium subscriptions. Consider registering with tax authorities as required.

  By addressing these security considerations, the Adjuface Telegram Bot can provide a safe and secure experience for its users while complying with relevant regulations and standards when transitioning to a commercial venture.

11. Monitoring and Logging

11.1. Logging Strategy

The Adjuface Telegram Bot utilizes a logging strategy to capture and record various events and activities within the application. Here are the key aspects of the logging strategy:

- **Logging Levels**: The bot employs different logging levels, including INFO and WARN. However, there is an issue where the database logging sometimes sends INFO-level logs even when set to WARN. This issue needs to be resolved to ensure that logs are consistent and appropriate.
- **Activity Logs**: User activities, such as image processing requests and premium feature usage, are logged for monitoring and auditing purposes. These logs help track user interactions and ensure compliance with usage limits.
- **Error Handling**: The bot currently lacks comprehensive error logging. Implementing error logging is crucial for identifying and addressing issues promptly. Errors should be logged with detailed information to facilitate troubleshooting.

11.2. Monitoring Tools

Monitoring the bot's performance and user interactions is essential for ensuring its smooth operation. While there may not be specific monitoring tools integrated into the application, monitoring can be achieved through the following methods:

- **Manual Inspection**: The bot owner inspects user interactions, images processed, and usage patterns by directly interacting with the bot through Telegram.
- **Performance Metrics**: Performance metrics, such as response times and image processing times, can be collected and analyzed to identify performance bottlenecks or issues.

11.3. Alerts and Notifications

To maintain the bot's integrity and address potential issues in a timely manner, the following alerts and notifications mechanisms are in place:

- **Support Requests**: Users can send support requests using the "/support" command. These requests are sent as notifications to bot owner's Telegram account, allowing for quick response to user queries or issues.
- **Inappropriate Content:** TI manually inspect images sent by users and take action if inappropriate content is detected. This may involve banning users who violate content guidelines.
- **Usage Limits**: The bot enforces usage limits, and users are notified when they reach these limits. Notifications inform users about their remaining limits and encourage them to consider premium options if they require more extensive usage.

By improving error handling and addressing the INFO-level logging issue, the Adjuface Telegram Bot can enhance its monitoring and logging capabilities to ensure smoother operation and better user support.

12. Maintenance and Support

12.1. Maintenance Plan

The Adjuface Telegram Bot is designed to be a lightweight and low-maintenance application. Maintenance primarily involves monitoring the bot's activities, addressing any reported issues, and ensuring that it continues to operate smoothly. Here is an overview of the maintenance plan:

- **Regular Monitoring**: The bot owner regularly monitor user interactions, logs, and system performance to identify any anomalies or issues. This includes checking for inappropriate content and ensuring users adhere to usage limits.
- **Error Handling**: The bot's error handling mechanisms are continuously improved to capture and log errors effectively. This helps in identifying and addressing issues promptly.
- **Database Maintenance**: The database, where user data and logs are stored, may require occasional maintenance to optimize queries and ensure efficient data retrieval.
- **Security Updates**: Security updates and patches are applied as needed to protect the bot from potential vulnerabilities.
- **Bot Updates**: The bot's functionality and features may be updated or expanded based on user feedback and requirements. These updates aim to enhance user experience and provide new capabilities.

12.2. Bug Tracking and Resolution

To track and resolve bugs and issues, the following approach is followed:

- **Manual Inspection**: The bot owner manually inspect user interactions and logs to identify any issues or irregularities.
- **User Reporting**: Users can report issues or seek support by using the **"/support**" command. These support requests are sent as notifications to bot owner's Telegram account, allowing for direct communication with users to resolve problems.
- **Bug Tracking**: Reported issues are documented and tracked to ensure that they are addressed in a timely manner. A record of reported bugs and their resolution status is maintained.
- **Testing**: Updates and changes to the bot's code are thoroughly tested to prevent the introduction of new issues. Both unit tests and user interaction tests are conducted.

12.3. User Support Channels

Users can seek support and assistance through the following channels:

- **Support Requests**: Users can send support requests using the "**/support**" command within the Telegram bot. These requests are received as notifications by the bot owner and are typically addressed promptly.
- **Contact Information**: The bot provides users with contact information through the "**/contacts**" command, allowing them to reach out for assistance or inquiries.
- **In-App Messages**: Users can send direct messages to the bot owner via the Telegram platform for support or questions related to the bot's functionality. Also the messages sent to bot are also logged.
- **Documentation**: While not a direct support channel, users can refer to the bot's documentation for information on commands and usage guidelines at **github.com/Dimildizio**.

The Adjuface Telegram Bot aims to provide a seamless user experience, and maintenance and support efforts are focused on achieving this goal by promptly addressing issues and ensuring users have access to assistance when needed.

13. Appendices

13.1. Glossary

- **Adjuface**: The name of the Telegram Bot designed for face swapping.
- **Bot Owner**: The individual named Dmitrii Didenko who developed, owns, and maintains the Adjuface Telegram Bot.
- **Face Swap**: The process of replacing a person's face in an image with another person's face.
- **Telegram Bot**: A bot (software application) that operates within the Telegram messaging platform and interacts with users through text and media messages.
- **Premium User**: A user who has opted for the premium account, allowing additional features and customization.

13.2. References

The development and operation of the Adjuface Telegram Bot have been influenced by various resources and technologies. Some key references include:

- **Telegram Bot API Documentation**: The official documentation for the Telegram Bot API, which provides essential information for bot development.
- **Insightface Documentation**: Documentation for the insightface framework, used for face swapping.
- **FastAPI Documentation**: Documentation for the FastAPI framework, used to create the API for face swapping.
- **SQLAlchemy Documentation**: Documentation for the SQLAlchemy library, used for database interaction.

13.3. Acknowledgments

The development of the Adjuface Telegram Bot was made possible by the contributions of various individuals and resources. The bot owner would like to acknowledge and express gratitude to the following:

- **OpenAI**: For their cutting-edge technology, including GPT-3 and 4, which significantly enhanced the code owner's documentation-writing abilities.
- **DeepInsight**: For their awesome open-source code which was utilized in the project.
- **Telegram**: For providing a platform that allows the development of interactive and engaging bots.
- **FastAPI**: For simplifying the process of creating the API for face swapping.
- **SQLAlchemy**: For the database management capabilities that support user data storage and retrieval.
- **Bot Users**: For their engagement and feedback, which helps improve the bot's functionality and user experience.
- **DeepLearning School**: For their great course and virtual museum graduation project that gave birth to Adjuface idea.
- **Community**: For sharing knowledge, resources, and support in the field of bot development.
- **Dungeons and Dungeons**: D&D group of it-crew geeks for their support.
- **TG ODS/DLS/RS GANG**: for heavy support and inspiration.

This bot is a lot of effort, and the bot owner extends their appreciation to all those who have contributed even indirectly to its development and operation.