МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Вятский государственный университет»
(ФГБОУ ВПО «ВятГУ»)
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Допущено к защите
Руководитель проекта
_____ (Долженкова М. Л.)
«___» _____2023г.

«**Разработка программного модуля для сбора информации об интересах посетителей парковки»**

Пояснительная записка курсового проекта по дисциплине
«Курс знаний бакалавра»
ТПЖА 090301.387 ПЗ

Разработал студент группы ИВТ-31 _____/Жеребцов К.А./

Руководитель _____/Долженкова М. Л./

Консультант _____/Долженкова М.Л./

Работа защищена с оценкой «_____» _____
(оценка) *(дата)*

Члены комиссии _____ /_____/
(подпись)
_____ /_____/
(подпись)
_____ /_____/
(подпись)

Киров 2023

Реферат

Жеребцов К. А. «Разработка программного модуля для сбора информации об интересах посетителей парковки». ТПЖА. 090301.387 ПЗ: Курс. проект / ВятГУ, каф. ЭВМ; рук. Долженкова М. Л. - Киров, 2023. Графическая часть 7 л. – ф. А2, ПЗ 65 с, 32рис., 5 источников, 5 приложений.

Разработка программного модуля «Разработка программного модуля для сбора информации об интересах посетителей парковки».

Объект исследования и разработки – программный модуль для сбора информации об интересах посетителей парковки.

Цель курсового проекта – автоматизация процесса сбора информации об интересах посетителей парковки.

Результат работы – готовый прототип программного модуля.

Содержание

Введение

В современном мире сбор и анализ данных становятся все более важными инструментами для принятия решений в различных областях деятельности. Особенно в сфере общественных мест, таких как парковки, понимание интересов и предпочтений посетителей может существенно повлиять на управление и развитие таких мест. В рамках данного проекта был разработан модуль для анализа интересов посетителей парковки с использованием методов компьютерного зрения и обработки опросных данных.

Целью данного проекта является создание инструмента, способного автоматизировать сбор информации о посетителях парковки, их предпочтениях и интересах. Для достижения этой цели были поставлены следующие задачи:

1. Выявление наиболее оптимального метода сбора данных
2. Разработка алгоритма анализа полученных данных для определения интересов посетителей.
3. Создание скрипта для извлечения информации о модели машины.
4. Интеграция разработанных компонентов в единый функциональный модуль.

1 Анализ предметной области

В данном разделе проводится анализ предметной области, который позволит обосновать актуальность разработки модуля, приводятся ключевые требования и особенности, которые должны быть реализованы.

1.1 Сбор данных

Сбор данных - важный аспект экономики, поскольку он помогает систематически принимать обоснованные решения. Данные - бесценный инструмент для принятия взвешенных решений, который экономит как время, так и ресурсы. Для сбора данных экономисты и статистики, среди прочих, используют некоторые соответствующие методы, которые помогают им получать информацию, относящуюся к интересующим их предметам.

Сбор данных - это процесс измерения и сбора информации о желаемых переменных таким образом, чтобы можно было находить вопросы, связанные с данными, и использовать их в исследованиях различных типов. Сбор данных является общей чертой обучения по различным дисциплинам, таким как маркетинг, статистика, экономика, естественные науки и т.д. Методы сбора данных могут варьироваться в зависимости от предмета, но конечная цель исследования и честность при сборе данных имеют одинаковое значение во всех вопросах обучения.

В зависимости от характера сбора данных их можно разделить на два основных типа, а именно:

- Первичный метод сбора данных
- Вторичный метод сбора данных.

Первичные данные собираются исследователями самостоятельно и впервые в рамках исследования. Существуют различные способы сбора первичных данных, некоторые из которых следующие:

- Интервью: Интервью являются наиболее часто используемым методом первичного сбора данных. При проведении интервью для сбора данных используется анкета или исследователь может задавать вопросы непосредственно интервьюируемому. Идея заключается в поиске информации по волнующим темам из ответов респондента. Используемые анкеты можно отправить по электронной почте или уточнить подробности в ходе телефонного интервью.
- Метод Дельфи: В этом методе исследователь запрашивает информацию у группы экспертов. Исследователь может выбрать очное исследование или отправить анкеты по электронной почте. В конце применения метода Дельфи все данные собираются в соответствии с потребностями исследования.
- Проективные методы: Проективные методы используются в исследованиях, которые являются частными или конфиденциальными таким образом, что исследователь считает, что респонденты не раскроют информацию, если будут заданы прямые вопросы. Существует множество типов проективных методик, таких как тематические тесты оценки (TAT), ролевые игры, завершение мультфильма, словесная ассоциация и завершение предложения.
- Интервью в фокус-группе: Здесь собираются несколько человек, чтобы обсудить насущную проблему. В таких интервью обычно участвует от шести до двенадцати человек. Каждый участник выражает свое собственное мнение, и принимается коллективное единогласное решение.
- Метод анкетирования: Здесь используется вопросник для сбора данных от различных групп населения. Для соответствующего

исследования используется набор вопросов, и респонденты отвечают на вопросы, прямо или косвенно связанные с вопросником. Этот метод может быть, как открытым, так и закрытым.

Вторичные данные означают данные, которые исследователь собирает не самостоятельно и в первый раз. Фактически, вторичные данные уже доступны и их необходимо собирать из различных источников.

Некоторые источники данных, которые могут быть использованы для вторичного сбора данных, включают:

- Газеты
- Журналы
- Журналы
- Публичные записи
- Деловые документы
- Исторические и статистические документы и т.д.

Помимо упомянутых выше источников опубликованных данных, для вторичного сбора данных могут использоваться неопубликованные данные, полученные из писем, дневников и неопубликованных биографий.

Количественные методы первичного сбора данных могут быть различных типов, некоторые из которых следующие:

- Вероятностная выборка

Здесь в целевой совокупности осуществляется некоторая форма случайного отбора. Затем на основе данных случайной выборки, полученных от генеральной совокупности, делаются заявления о вероятности. Преимущество случайной р[вероятностной выборки заключается в том, что исследователи могут собирать данные от представителей, а не от всех популяций, и данные в основном беспристрастны по форме.

- Интервью

Интервью также являются хорошим способом сбора первичных данных. Собеседования могут быть трех типов: телефонные, очные и компьютерные. Целью интервью является быстрое и эффективное получение объективных персональных данных о респондентах.

- Опросы

Анкетирование также может быть отличным способом сбора первичных данных. Анкету можно отправить по электронной почте или через Интернет. Основная цель опросов - быстрое получение данных в поддающейся проверке форме. Это также очень простой способ получить первичные данные от населения.

- Наблюдения

Наблюдения также являются очень простым способом сбора количественных данных. В ходе наблюдений за группой населения в данной демографической зоне проводится наблюдение для получения ответов, связанных с первичным исследованием. Обычно наблюдения проводятся для выяснения, кто, что и почему отвечает на вопросы исследования.

Методы сбора качественных данных

Как упоминалось выше, существует множество методов сбора качественных данных, некоторые из которых пересекаются с методами сбора количественных данных. Например**,** интервью могут проводиться для сбора как количественных, так и качественных данных. То же самое верно и для наблюдений.

Четыре наиболее часто используемых метода сбора качественных данных - это опросы, групповые обсуждения, наблюдения и интервью. Также могут использоваться другие методы, такие как методика Дельфи и проективные методики.

Качественные данные часто являются текстовыми и используются для исследований и анализа. Следовательно, необходимо поддерживать

допустимую погрешность в качественных данных, чтобы получить правдивую картину результатов исследования. С другой стороны, количественные данные должны быть точными, поскольку они напрямую определяют результат исследования.

1.2 Актуальность темы

Сбор и анализ данных о посетителях парковки является ключевым элементом для повышения уровня сервиса и удовлетворения потребностей пользователей. Традиционные методы сбора информации часто требуют значительных временных и человеческих ресурсов для обработки и анализа. Вместе с тем, использование современных технологий, таких как компьютерное зрение и распознавание текста, предоставляет новые возможности для автоматизации процессов сбора и анализа данных на парковках.

Проект направлен на разработку инновационного модуля, способного автоматически собирать и анализировать информацию о посетителях парковки на основе их ответов в опросах и распознавания номеров автомобилей. Подобный подход позволяет существенно сократить время, затрачиваемое на сбор и обработку данных, а также предоставляет более точную и полную информацию о предпочтениях и интересах посетителей.

С учетом растущей конкуренции в области общественных мест, таких как парковки, эффективное использование собранных данных становится ключевым конкурентным преимуществом. Автоматизированный анализ интересов посетителей и их профилей поможет улучшить качество сервиса, оптимизировать распределение ресурсов и разработать персонализированные предложения для пользователей.

Таким образом, разработка модуля для анализа интересов посетителей парковки на основе современных технологий компьютерного зрения и

обработки данных представляет собой актуальное и перспективное направление в развитии сферы общественных мест и сервисной индустрии.

1.3 Постановка задачи

1.3.1 Цели и задачи разработки

Целью разработки является автоматизация процесса сбора информации об интересах посетителей парковки.

Задачи:

- Выявление наиболее оптимального метода сбора данных
- Разработка алгоритма анализа полученных данных для определения интересов посетителей.
- Создание скрипта для извлечения информации о модели машины.
- Интеграция разработанных компонентов в единый функциональный модуль.

1.3.2 Определение ключевых функций

- Сбор и обработка опросных данных:

Модуль должен быть способен автоматически собирать опросные данные с заполненных анкет посетителей парковки.

Был определен ряд вопросов, который позволяют определить некоторые основные интересы:

- Интересуетесь ли вы искусством (картины, скульптура, музыка и т.д.)?
- Занимаетесь ли вы спортом или физической активностью?
- Любите ли вы чтение книг или просмотр фильмов?
- Интересуетесь ли вы наукой и технологиями?
- Увлекаетесь ли вы путешествиями и открытием новых мест?
- Является ли для вас кулинария или готовка хобби?

– Интересуетесь ли вы политикой и общественными вопросами?

Так же на бланке присутствует поле, в котором опрашиваемый должен указать свой возраст. Структура бланка представлена на рисунке 1.

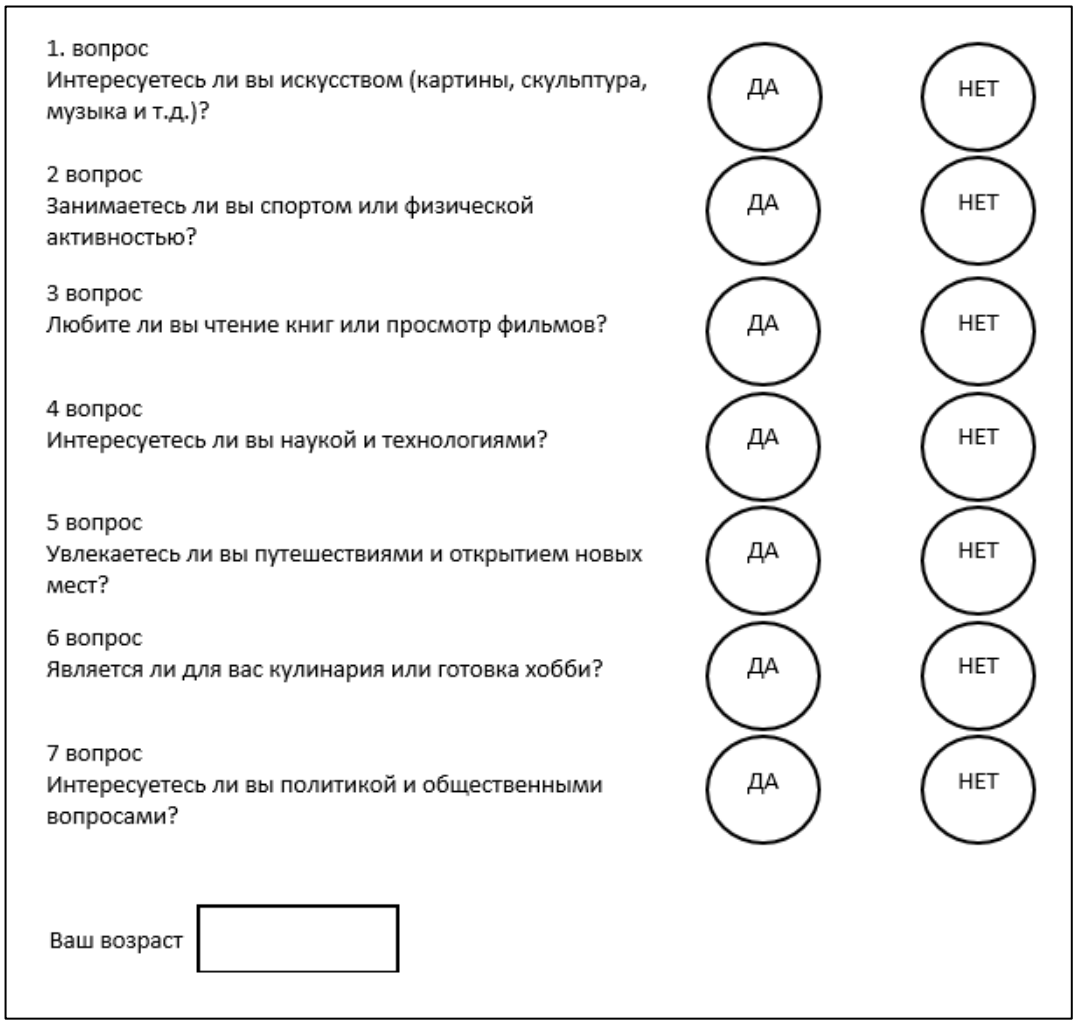

Рисунок 1 – Опрос

Опрашиваемый должен закрасить кружки, соответствующие его ответу на каждый вопрос, а также указать свой возраст.

Бланк на вход должен подаваться в виде фотографии или скан копии.

После сбора данных необходимо провести их предварительную обработку, включая распознавание текста и извлечение ответов на вопросы опроса.

– Распознавание номеров автомобилей:

Модуль должен обеспечивать распознавание номеров автомобилей с изображений, полученных с камер видеонаблюдения на парковке.

Разрешение входного изображения должно быть не ниже 1280x720. Кроме того, распознаваемый номер должен быть читаем, то есть быть чистым, не выцветшим и не закрыт посторонними предметами. Это позволит повысить точность распознавания и сократить количество ошибок.

После распознавания номеров необходимо извлечь информацию о модели автомобиля и ассоциировать её с данными о посетителе.

- Классификация данных:

Модуль должен не только собирать ответы посетителей, но уметь определять каждый ответ в соответствующую группу. Полученные ответы на вопросы должны быть классифицированы по заданным категориям, например, "интерес к продуктам питания", "интерес к развлечениям" и т.д.

Вывод

В данном разделе был проведен анализ предметной области, подчеркнута актуальность разработки модуля для анализа интересов посетителей парковки. Исходя из рассмотренных факторов, определены ключевые требования и функции, которые должны быть реализованы в разрабатываемом программном продукте.

Модуль предоставляет средство для более точного и полного анализа данных о посетителях парковки. Автоматизированный сбор и анализ информации о предпочтениях и интересах посетителей поможет оптимизировать уровень сервиса, а также разрабатывать персонализированные предложения для пользователей. Таким образом, разработка данного модуля является актуальной и перспективной в контексте современных тенденций в сфере общественных мест и сервисной индустрии.

2 Обзор аналогов

В этом разделе будут рассмотрены аналоги, который используются при анализе различный бланков ответов и опросов. Кроме того, будет выявлен принцип их работы, который будет использоваться при создании компонента для анализа опроса.

2.1 Аналоги

2.1.1 ZipGrade

Это приложение для сканирования и оценивания тестовых работ с помощью камеры смартфона или планшета. Оно позволяет преподавателям быстро сканировать и оценивать бумажные тесты, используя технологию оптического распознавания символов (OCR). ZipGrade автоматически распознает ответы и вычисляет итоговые оценки, что существенно экономит время преподавателей.

Рисунок 2– ZipGrade

Основные возможности:

1. Сканирование тестов:
   - ZipGrade позволяет преподавателям сканировать бумажные тесты, включая множественный выбор, правильные/неправильные ответы и другие форматы.
2. Оценивание:
   - После сканирования теста приложение автоматически оценивает ответы и вычисляет итоговую оценку на основе предварительно заданных правильных ответов.
3. Гибкие настройки:
   - Пользователи могут настраивать параметры теста, такие как количество вопросов, формат ответов и оценочные критерии.
4. Статистика:
   - ZipGrade предоставляет детальную статистику о производительности класса, включая средние баллы, распределение оценок и анализ наиболее часто ошибочных вопросов.
5. Экспорт результатов:
   - Результаты тестов могут быть легко экспортированы в формате CSV или PDF для дальнейшего анализа или сохранения.

Преимущества:

- Экономия времени: ZipGrade существенно сокращает время, затрачиваемое на оценивание тестов, благодаря автоматизированному процессу сканирования и оценивания.
- Точность: Приложение обеспечивает высокую точность при сканировании и оценивании ответов, что уменьшает вероятность ошибок, связанных с ручным оцениванием.

- Удобство: ZipGrade удобно использовать благодаря интуитивно понятному интерфейсу и широким возможностям настройки.

Использование:

Преподаватели могут использовать ZipGrade для оценивания различных типов тестов, в том числе для проверки знаний студентов, проведения контрольных работ и подготовки к экзаменам. Приложение также может быть полезно для учебных заведений, где требуется частое проведение тестирования.

2.1.2 Eyegrade

Eyegrade - это еще одно приложение, которое используется для сканирования и автоматического оценивания бумажных тестов. Оно предоставляет схожие функции с ZipGrade, но с некоторыми отличиями в особенностях и интерфейсе.

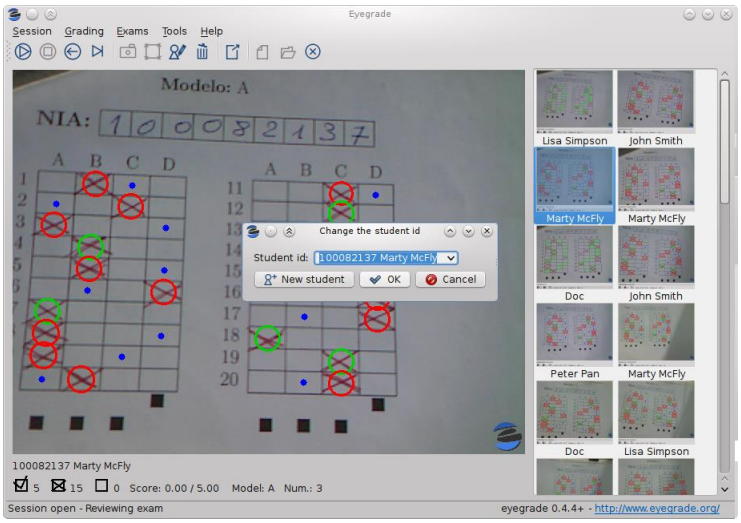

Рисунок 3 - Eyegrade

Основные возможности Eyegrade:

1. Сканирование тестов:
   - Eyegrade позволяет сканировать бумажные тесты с помощью камеры мобильного устройства.

2. Автоматическое оценивание:
    - После сканирования приложение автоматически распознает ответы студентов и вычисляет итоговую оценку.
3. Поддержка различных типов тестов:
    - Eyegrade поддерживает различные форматы тестов, включая множественный выбор, правильные/неправильные ответы и другие.
4. Экспорт результатов:
    - Пользователи могут экспортировать результаты в различные форматы для дальнейшего анализа или сохранения.
5. Удобство использования:
    - Eyegrade обладает простым и интуитивно понятным интерфейсом, что делает его удобным в использовании.

Преимущества:

- Экономия времени: Подобно ZipGrade, Eyegrade позволяет преподавателям существенно сократить время, затрачиваемое на оценивание тестов.
- Точность: При правильной настройке и использовании Eyegrade обеспечивает высокую точность при распознавании и оценивании ответов.
- Гибкость: Пользователи могут настраивать параметры тестов в соответствии с их потребностями.

Использование:

Eyegrade также может использоваться преподавателями для проведения тестирования, контрольных работ и оценивания студентов. Подобно другим приложениям подобного рода, его можно применять в различных образовательных учреждениях.

2.2 Выявление сходств и принципа работы

Оба приложения, ZipGrade и Eyegrade, представляют собой инновационные инструменты, которые значительно упрощают и ускоряют процесс оценивания бумажных тестов. Эти программы основаны на использовании передовых технологий в области компьютерного зрения и обработки изображений, что позволяет им автоматически распознавать ответы студентов и проводить оценивание с высокой точностью.

Одно из основных преимуществ использования подобных приложений состоит в экономии времени. Там, где ранее преподаватели тратили много часов на ручное оценивание бумажных тестов, сейчас им достаточно просто сфотографировать эти тесты с помощью смартфона или планшета, и приложение сделает всю работу за них, автоматически вычислив результаты.

Кроме того, использование этих приложений способствует повышению точности оценивания. Поскольку распознавание ответов происходит автоматически, это исключает возможность человеческих ошибок, которые могли бы возникнуть при ручном подсчете баллов. Это особенно важно при проведении массовых тестирований или экзаменов, где даже небольшие ошибки могут иметь серьезные последствия.

Кроме того, благодаря гибким настройкам и возможности экспорта результатов, преподаватели могут эффективно управлять данными о производительности студентов и использовать их для дальнейшего анализа и улучшения учебного процесса.

Общие черты и принцип работы ZipGrade и Eyegrade:

- Сканирование бумажных тестов: Оба приложения предоставляют возможность сканирования бумажных тестов с помощью камеры мобильного устройства. Этот процесс позволяет быстро и удобно перенести информацию с бумажного теста в цифровой формат.

- Автоматическое распознавание ответов: Обе программы используют технологию оптического распознавания символов (OCR), чтобы автоматически распознавать ответы, отмеченные на бумаге. Это позволяет сразу же перевести бумажные ответы в цифровой формат для последующей обработки.
- Оценивание и вычисление оценок: После сканирования ответов приложения автоматически оценивают тесты, основываясь на заранее заданных критериях правильных ответов. Они вычисляют итоговые оценки на основе этих критериев, что позволяет преподавателям быстро узнать результаты тестирования.
- Гибкие настройки: Оба приложения позволяют настраивать различные параметры тестов, такие как формат ответов, количество вопросов и оценочные критерии. Это обеспечивает гибкость в адаптации программы к конкретным потребностям преподавателя.
- Экспорт результатов: После оценивания тестов пользователи могут экспортировать результаты в различные форматы, такие как CSV или PDF, для последующего анализа или сохранения. Это помогает в учете и анализе данных о производительности студентов.

Исходя из схожих черт, можно определить принцип работы аналогов. Принцип работы обеих программ основан на комбинации технологий сканирования, оптического распознавания символов и алгоритмов оценивания. После сканирования бумажного теста, программа анализирует изображение, распознает отмеченные ответы с помощью OCR, сопоставляет их с правильными ответами и вычисляет итоговую оценку в соответствии с заданными критериями. Этот процесс автоматизирован и ускорен, что делает оценивание тестов более эффективным и удобным для преподавателей.

Вывод

В этом разделе были рассмотрены аналоги, который используются при анализе различный бланков ответов и опросов, а также выявлен принцип их работы. Принцип работы обеих программ основан на комбинации технологий сканирования, оптического распознавания символов и алгоритмов оценивания, что делает процесс оценивания более эффективным и удобным. В целом, использование таких приложений способствует экономии времени, повышению точности оценивания и эффективному управлению данными

3 Разработка компонентов

В этом разделе будут рассмотрены основные компоненты разрабатываемого модуля. Будет определен их функционал и особенности, а также описание их взаимодействия.

3.1 Структура модуля

Необходимо сформировать наиболее точное описание разрабатываемого программного обеспечения. Для этого было принято решение о рассмотрении функциональной диаграммы верхнего уровня.

В данном случае в качестве отображения взаимосвязей была выбрана нотация IDEF0. В качестве входных бланк опроса и изображение автомобиля. В качестве субъекта выступает пользователь и вычислительная машина. Управление задается алгоритмами анализа бланков, распознавания номера и получения модели автомобиля. К выходным данным будут относиться информация об интересах посетителя, его возраст и модель машины.

Контекстная диаграмма IDEF0 представлена на рисунке 4.

Рисунок 4 – Контекстная диаграмма IDEF0

Детализирующая функциональная диаграмма более подробно раскрывает функциональную диаграмму верхнего уровня: описывает взаимодействия и связи процессов, происходящих внутри системы. На ней можно увидеть, какие процессы взаимосвязаны и что между ними общего.

На детализирующей функциональной диаграмме показаны следующие этапы:

- Анализ бланков. Входными данными являются бланки опросов. На выход поступает информация об интересах и возраст опрашиваемого
- Распознавание номеров. Входными данными являются изображения автомобилей. На выход поступает государственный номер автомобиля.
- Определение модели. На вход поступает номер автомобиля, а на выход идет модель машины.

Детализированная контекстная диаграмма представлена на рисунке 5.

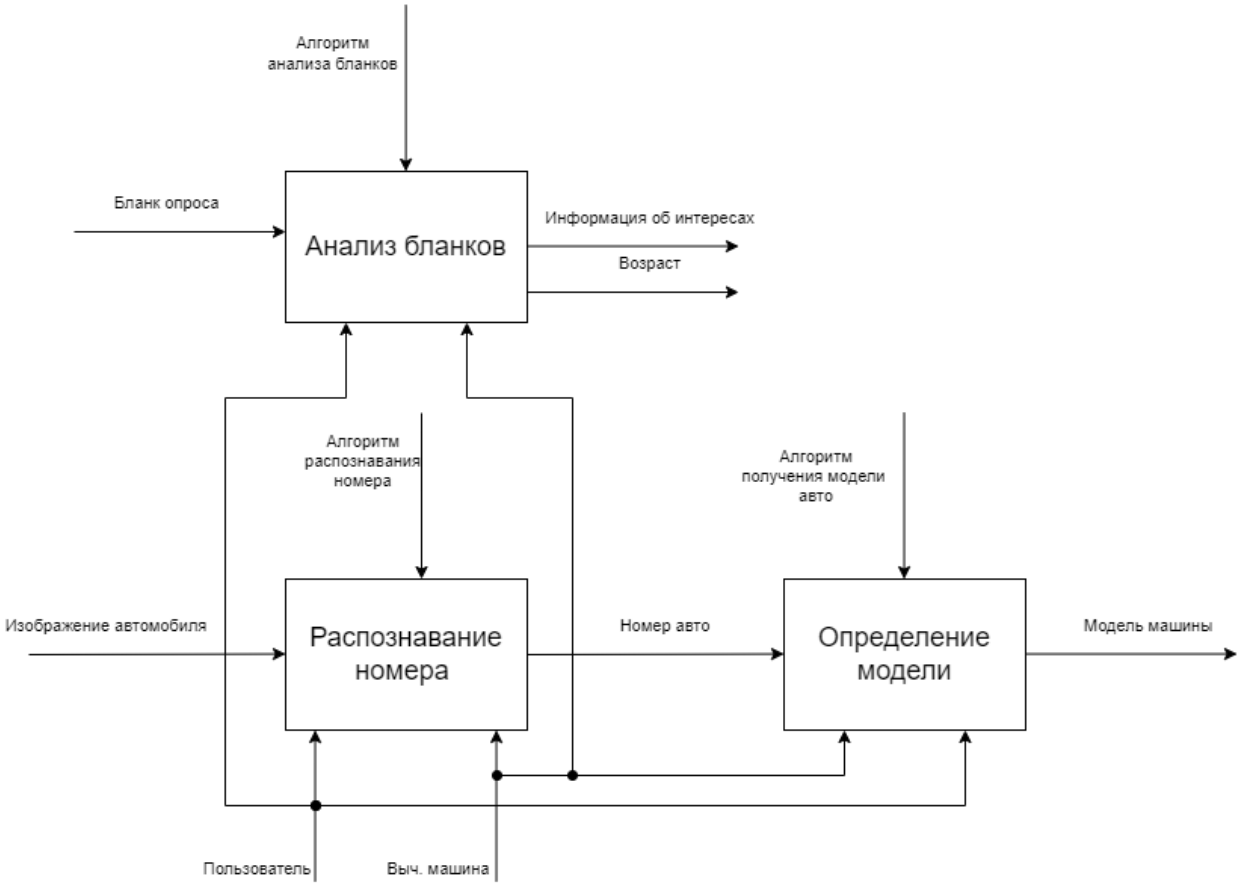

Рисунок 5 – Детализированная контекстная диаграмма IDEF0

Таким образом, разрабатываемый модуль будет состоять из трех основных компонентов: «Анализ бланков», «Определение модели», «Распознавание номера».

Так же стоит определить формат создаваемого датасета. В него будет входить вся получаемая информация. Следовательно, датасет будет иметь следующие поля:

- model: модель автомобиля
- age: возраст
- art: интерес к искусству
- sport: интерес к спорту
- book/film: интерес к фильмам и книгам
- science: интерес к науке и технологиям
- travel: интерес к путешествиям
- cooking: интерес к готовке

- politics: интерес к политике

3.2 Анализатор бланков

Основными функциями компонента являются функции для анализа анкет, сбора информации об ответах и возрасте, указанном на бланке.

Задача компонента заключается в анализе анкет. Он должен автоматически обрабатывать изображения анкет, извлекать информацию об ответах на вопросы и возрасте, указанном на бланке.

Для выполнения задачи выбраны следующие инструменты:

- OpenCV: для обработки изображений, выделения контуров и преобразований перспективы.
- imutils: для удобной работы с изображениями, сортировки контуров и других операций.
- pytesseract: для распознавания текста на изображениях с использованием технологии OCR.

На вход компоненту подается изображение анкеты. Это может быть фотография сверху или скан копия бланка. После своей работы результаты будут сохраняются в CSV-файл.

Описание алгоритма:

- Начало: Вход в компонент.
- Загрузка изображения: Компонент получает на вход изображение анкеты для анализа.
- Препроцессинг изображения:
  - Преобразование в оттенки серого.
  - Сглаживание с помощью фильтра Гаусса.
  - Применение оператора Canny для обнаружения границ.
- Поиск области анкеты:
  - Поиск контуров на изображении.

- Определение контура анкеты.

- Выделение области анкеты:
  - Применение преобразования перспективы для выделения области анкеты.
  - Получение и сохранение выделенной области.
- Обработка анкеты:
  - Применение пороговой обработки к выделенной области для получения четкого изображения.
  - Обнаружение контуров на обработанном изображении.
  - Определение контуров блоков с ответами на анкете.
- Анализ ответов:
  - Идентификация блоков с ответами на анкете.
  - Определение заполненных ответов в каждом блоке.
  - Оценка правильности ответов согласно ключу ответов.
  - Формирование списка результатов анализа.
- Извлечение возраста:
  - Определение области, содержащей возраст на анкете.
  - Применение преобразования перспективы для выделения области с возрастом.
  - Применение пороговой обработки и распознавание текста для извлечения возраста.
- Запись результатов:
  - Сохранение результатов анализа и возраста в CSV файл.
- Конец: Завершение работы компонента.

Алгоритм работы компонента представлен на рисунке 6.

Рисунок 6 – Алгоритм работы анализатора

Данный модуль в принципе является самостоятельным. Он может использоваться в различных областях, где требуется автоматизированный анализ анкет. Также при желании, его можно расширить и настроить для анализа других типов анкет.

Также не стоит забывать об обработке ошибок. При реализации данного компонента необходимо предусмотреть решение ошибочных ситуаций (неправильное распознавание, неверный формат изображения), которые могут возникать в процессе работы, чтобы обеспечить корректную работу.

3.3 Поиск модели авто

Разработку компонента стоит начать с определения основной функции: обеспечение автоматизированного доступа к информации о моделях автомобилей. Информацию о модели можно взять с различных сайтов, таких как Nomerogram.ru, Avtocod.ru и другие. Важно учесть, что работа с сайтом будет вестись программно, а значит стоит выбрать сайт не только с самой большой базой автомобилей, но и тот, у которого нет «проверки на робота». Из представленных вариантов, а именно Nomerogram.ru, Avtocod.ru, Autoteka.ru, Avtoproverka.ru, имеющих примерно одинаковую базу автомобилей, все кроме Autoteka.ru имеют так называемую «проверку на робота» (рисунок 7).

Рисунок 7 – Проверка на робота

Следовательно, работа будет происходить именно с Autoteka.ru (рисунок 8).

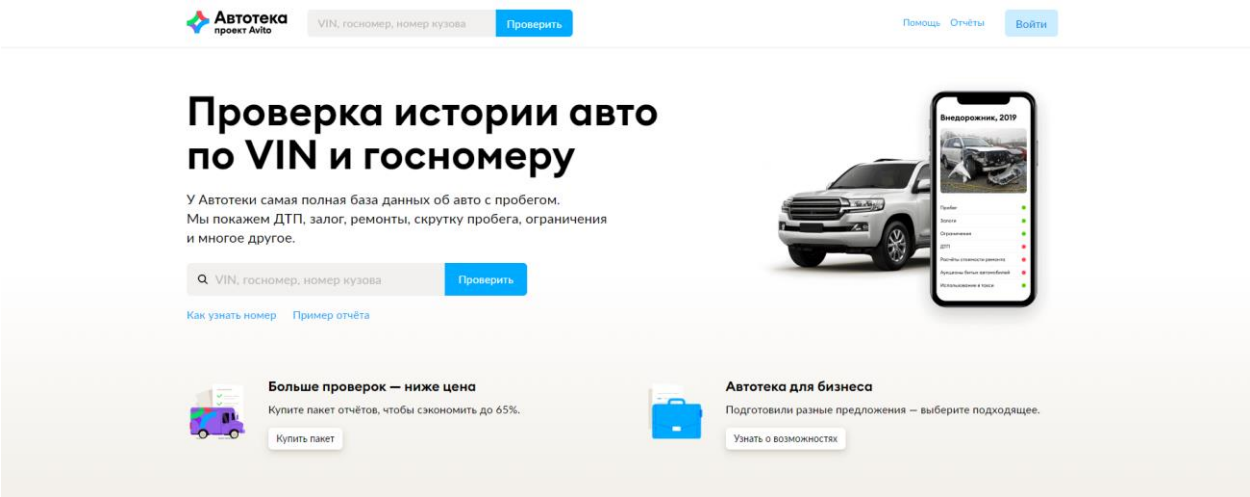

Рисунок 8 – Главный экран Autoteka.ru

Сайт autoteka.ru - это онлайн-платформа, предоставляющая информацию о транспортных средствах на территории России. Он широко используется автолюбителями, автодилерами, страховыми компаниями и другими участниками автомобильного рынка. Вот основные характеристики и особенности сайта autoteka.ru:

- Проверка истории автомобиля: autoteka.ru позволяет пользователям проверить историю конкретного автомобиля по его государственному регистрационному номеру. Эта функция позволяет узнать информацию о предыдущих владельцах, дате и результатам техосмотра, наличии ограничений и запретов на регистрацию, участии в ДТП и другие важные сведения.
- Получение технических данных: Пользователи могут получить доступ к техническим характеристикам автомобиля, таким как марка, модель, год выпуска, тип кузова, мощность двигателя и другие технические параметры. Это полезно при выборе автомобиля для покупки или продажи.
- Проверка наличия ограничений: Сайт позволяет проверить наличие ограничений на регистрацию автомобиля, таких как залоги, аресты или запреты на регистрацию по различным причинам. Это важно для покупателей, которые хотят удостовериться в чистоте транспортного средства перед сделкой.
- Получение отчетов по автомобилю: Пользователи могут заказать подробные отчеты об истории автомобиля, включая информацию о ранее совершенных сделках, числе владельцев, технических особенностях, участии в ДТП и других сведениях. Это помогает сделать информированный выбор при покупке подержанного автомобиля.
- Онлайн-консультации: Кроме того, avtoteka.ru предоставляет возможность онлайн-консультаций и поддержки пользователей в вопросах, связанных с автомобильной историей, юридическими аспектами регистрации транспортных средств и другими темами.

Сайт avtoteka.ru является надежным источником информации о транспортных средствах, предлагая широкий спектр услуг и функций для удовлетворения потребностей различных категорий пользователей на автомобильном рынке. Всю информацию можно получить по VIN, госномеру, номеру кузова. Алгоритм работы прост. Необходимо ввести имеющуюся информацию и нажать «Поиск». после чего появится страница с информацией об автомобиле (рисунок 9).

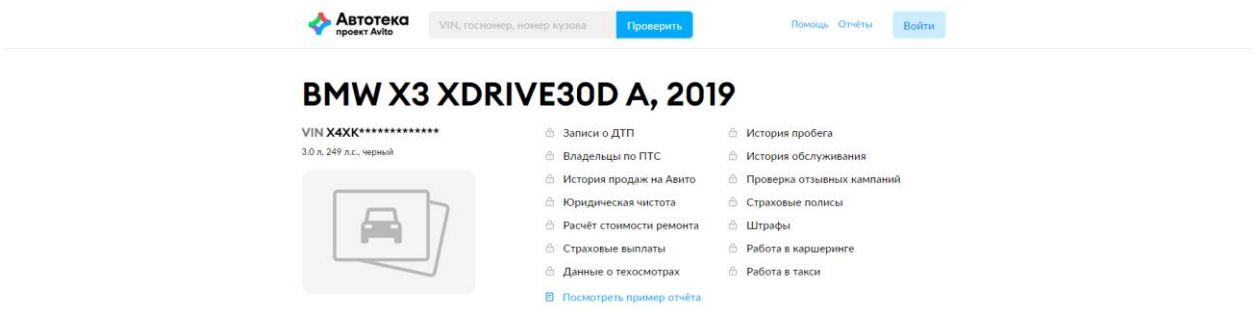

Рисунок 9 – Информация об автомобиле

Полная информация предоставляется по платной подписке или за разовый платеж, но для выполнения поставленной задачи необходимо получить только модель, что можно сделать бесплатно.

Функциональные требования: компонент должен обеспечивать ввод номера, отправку запроса на сайт Autoteka, получение результатов и извлечение необходимой информации.

Выбор инструментов и технологий:

- Для автоматизации взаимодействия с веб-страницами можно использовать библиотеку Selenium для Python, так как она обладает мощными возможностями веб-автоматизации.
- Для парсинга полученных данных удобно применять инструменты для работы с текстом и регулярные выражения, например, встроенные функции Python.

Составление алгоритма работы:

- Компонент должен открывать веб-браузер, переходить на сайт Autoteka и вводить указанный регистрационный номер.
- После получения результата, необходимо обработать страницу с результатами и извлечь нужные данные, такие как модель автомобиля.
- Важно учесть возможные сценарии ошибок и способы их обработки, например, отсутствие данных по указанному номеру или проблемы с доступом к сайту Autoteka.

Алгоритм работы представлен на рисунке 10.

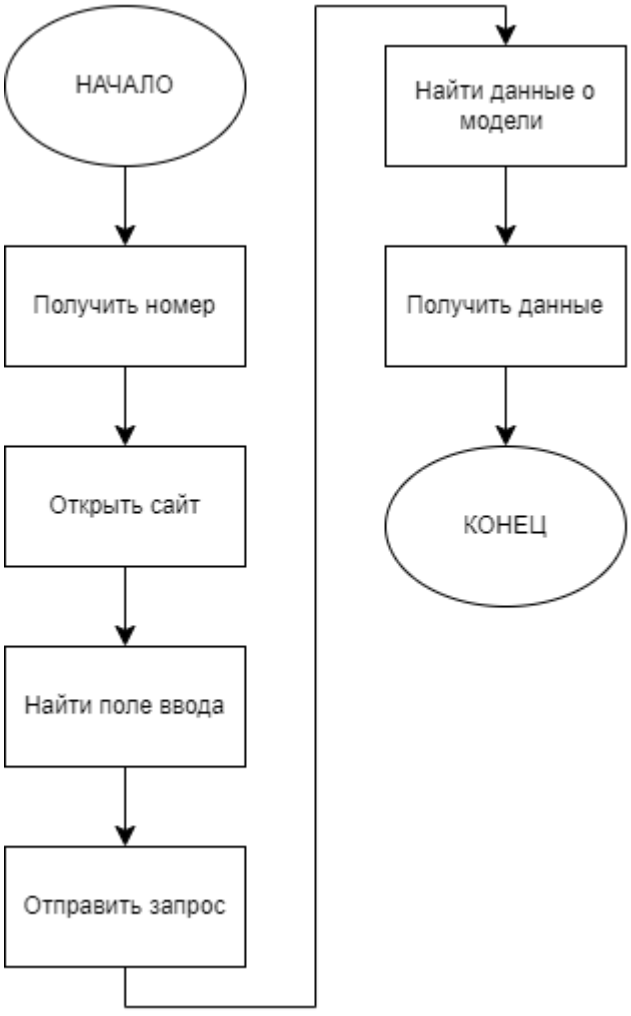

Рисунок 10 – Алгоритм работы

3.4 Распознавание номера

Данный компонент был разработан при выполнении предыдущего курсового проекта. Компонент служит для того, чтобы распознать гос. номер автомобиля.

Для обнаружения положения номерного знака используется предобученная модель model_resnet.tflite с архитектурой ResNet-50. Она предназначена для выявления границ номерного знака и выделение его области. ResNet-50 выбрана благодаря ее способности эффективно извлекать признаки из изображений и обучаться на больших объемах данных.

50-уровневая архитектура ResNet включает в себя следующие элементы:

- **Свертка ядра 7 × 7** наряду с 64 другими ядрами с шагом в 2 размера.
- **Максимальный уровень объединения** с шагом в 2 размера.
- **9 слоев**- свертка ядра размером 3 × 3,64, другой с 1 × 1,64 ядрами и третий с 1 × 1256 ядрами. Эти 3 слоя повторяются 3 раза.
- **12 слоев** с 1 × 1128 ядрами, 3 × 3128 ядрами и 1 × 1512 ядрами, повторяется 4 раза.
- **18 слоев** с ядрами 1 × 1256 и 2 ядрами 3 × 3,256 и 1 × 1,1024, повторяется 6 раз.
- **9 слоев** с ядрами 1 × 1512, 3 × 3512 и 1 × 12048 повторяются 3 раза.
- **Создание среднего пула**, за которым следует полностью подключенный уровень с 1000 узлами, с использованием функции активации softmax.

Для определения номера автомобиля используется предобученная модель model_nomer.tflite. Данная модель предназначена для определения текста на номерном знаке. фрагмент изображения с номерным знаком поступает от выше описанной модели. Модель представляет собой комбинацю сверточной нейронной сети (CNN), рекуррентной нейронной сети долгой краткосрочной памяти (LSTM) и функции свертки по времени (CTC) для

распознавания текста на изображениях. CNN используется для извлечения признаков из изображения номерного знака, LSTM преобразует эти признаки в последовательность символов, а CTC используется для выравнивания последовательности символов с фактическим текстом номера.

Структура модели представлена на рисунке 11.

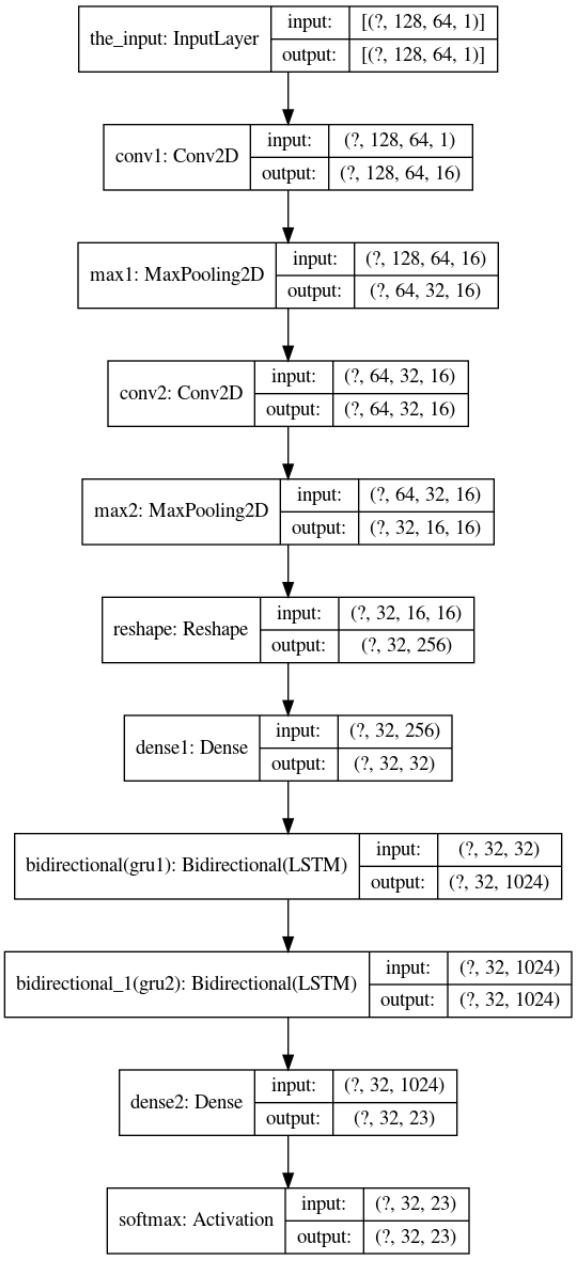

Рисунок 11 – Структура модели

К функциям компонента относятся следующие пункты:

- Загрузка и разбиение входного видео на кадры
- Выбор отдельного кадра
- Распознавание номера автомобиля на получившимся изображении

Алгоритм работы представлен на рисунке 12.

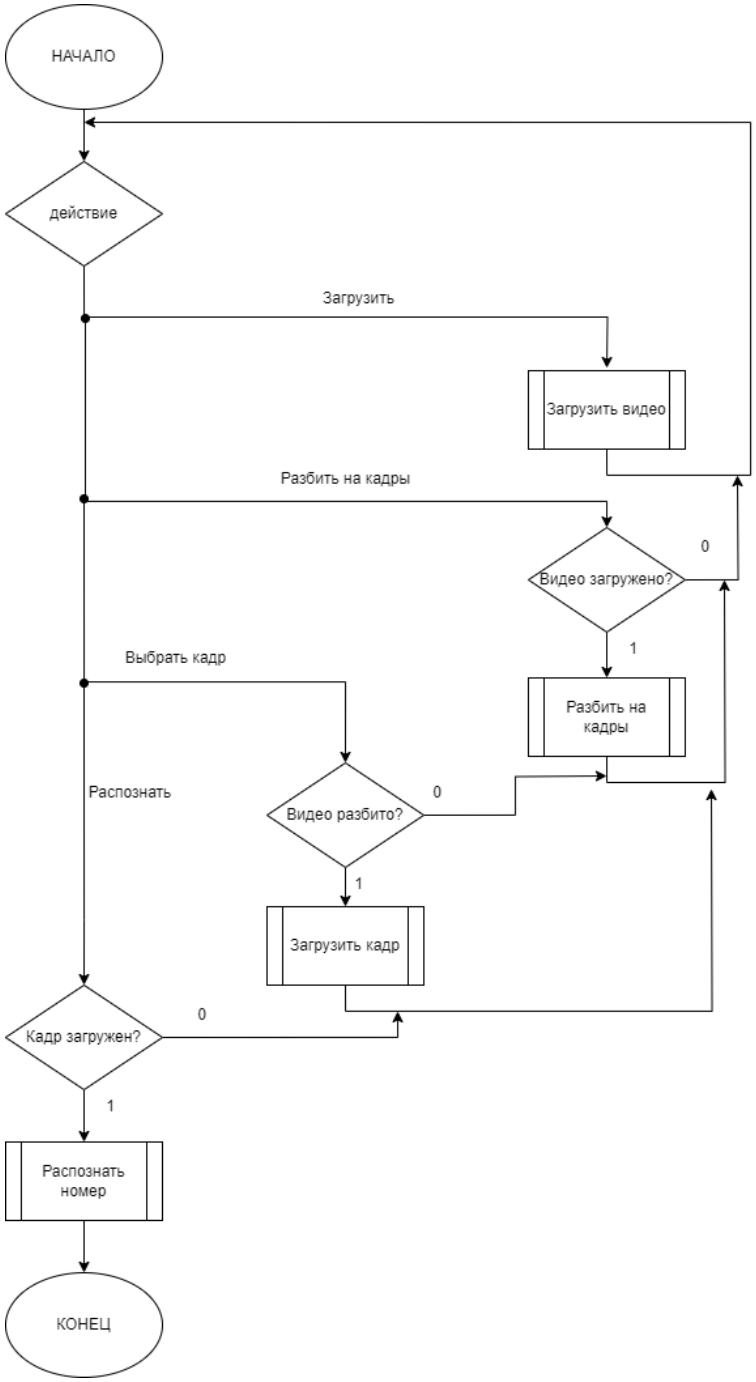

Рисунок 12 – Схема алгоритмов

Вывод

В этом разделе был рассмотрен процесс разработки компонентов, входящих в модуль. Были определены их функции и особенности.

4 Программная реализация

В данном разделе будут рассмотрены средства, методы и библиотеки, которые использовались при программной реализации.

4.1 Анализ бланков

В коде компонента используются библиотеки «cv2» (OpenCV) для обработки изображений, «numpy» для работы с массивами, «argparse» для обработки аргументов командной строки, «imutils» для удобных функций обработки изображений, «csv» для записи результатов в CSV-файл и «pytesseract» для распознавания текста на изображении.

Сам компонент имеет следующие основные функции:

- process_exam(image_path):

Функция принимает путь к изображению экзаменационного листа.

Используя библиотеку «cv2», выполняет следующие шаги обработки изображения:

- Преобразует изображение в оттенки серого, размывает его и применяет пороговое преобразование для выделения контуров.
- Находит контуры на изображении и выбирает наибольший контур, который предположительно является контуром экзаменационного листа.
- Используя функцию «four_point_transform» из «imutils», выполняет перспективное преобразование изображения, чтобы получить прямоугольный экзаменационный лист.

- Применяет пороговое преобразование к преобразованному изображению и находит контуры вопросов на экзаменационном листе.
- Для каждого вопроса находит наибольший контур, который предположительно является контуром выбранного ответа.
- Определяет выбранные ответы и возвращает результаты в виде списка.

- get_age(image_path):

Функция принимает путь к изображению документа, содержащего возраст.

Используя библиотеку «cv2», выполняет следующие шаги обработки изображения:

- Находит контуры на изображении и выбирает наибольший контур, который предположительно является контуром документа.
- Используя функцию «four_point_transform» из «imutils», выполняет перспективное преобразование изображения, чтобы получить прямоугольный документ.
- Применяет пороговое преобразование к преобразованному изображению и находит контуры чисел на документе.
- Используя библиотеку «pytesseract», выполняет распознавание текста на изображении и находит числа, представляющие возраст.
- Возвращает возраст в виде строки.

- «write_to_csv(filename, results)»:
  - Функция принимает путь к CSV-файлу и результаты обработки.
  - Используя библиотеку «csv», открывает файл в режиме добавления и записывает результаты в виде строки в CSV-файл.
- «blank(path_in, path_out, model)»:
  - Функция принимает путь к входному изображению, путь к выходному CSV-файлу и модель (номер экзаменационного листа).

- Вызывает функции «process_exam» и «get_age» для обработки изображения и определения возраста.
- Добавляет модель и возраст в начало списка результатов.
- Вызывает функцию «write_to_csv» для записи результатов в CSV-файл.

– «main()»:
- Основная функция, которая принимает путь к входному изображению и путь к выходному CSV-файлу с помощью аргументов командной строки.
- Вызывает функции «process_exam» и «get_age» для обработки изображения и определения возраста.
- Вызывает функцию «write_to_csv» для записи результатов в CSV-файл.

4.2 Поиск модели авто

При программной реализации данного компонента использовалась библиотеку Selenium для автоматизации веб-браузера. Она включает в себя следующие функции и шаги:

– Импортируются необходимые модули: selenium, time, и другие модули, необходимые для работы с Selenium.
– Указывается путь к драйверу браузера (chromedriver.exe) в переменной «driver_path».
– Указывается URL-адрес веб-сайта, на котором будет выполняться поиск, в переменной «website_url».
– Указывается номер, который будет использоваться в качестве вводного параметра, в переменной «num».
– Создается экземпляр браузера Chrome с использованием указанного драйвера в переменной «driver».

- Определяется функция «site», которая принимает в качестве аргумента номер и выполняет следующие действия:
    - Загружается веб-страница с указанным URL-адресом.
    - На странице вводится номер в поле с именем "identifier".
    - Номер отправляется в поле ввода с помощью метода «send_keys» и клавиши Enter.
    - Ожидается изменение URL-адреса на веб-странице с использованием «WebDriverWait».
    - Ожидается 3 секунды для загрузки страницы.
    - На странице находится элемент с классом "pit4K", который содержит текст, связанный с номером.
    - Текст из элемента извлекается и разбивается на строки с помощью «split('\n')».
    - Результирующий текст объединяется в две строки с помощью «join(lines[:2])».
    - Возвращается первая строка текста, разделенная запятой и приводится к нижнему регистру.
    - Завершается работа браузера с помощью «driver.quit()».
- В блоке «if __name__ == "__main__":» вызывается функция «site» с передачей номера в качестве аргумента и выводится результат в консоль.

4.3 Распознавание номера

Для работы модуля используются следующие библиотеки и модули:

- PyQt6: для создания графического интерфейса пользователя и обработки событий.
- OpenCV: для обработки изображений и видео, включая разделение видео на кадры и обнаружение номерных знаков.

- TensorFlow Lite: для загрузки и использования моделей машинного обучения для распознавания номеров автомобилей.
- NumPy: для работы с массивами и математических операций.
- Matplotlib: для визуализации результатов обработки изображений.
- SciPy: для преобразования изображений и обнаружения линий.
- autoteka и test_blank: пользовательские модули для работы с сайтом и создания пустых бланков.

Функции компонента:

- **`load()`**:
  - Функция загружает видеофайл из диалогового окна выбора файлов.
  - Использует метод **`QFileDialog.getOpenFileName()`** для открытия диалогового окна выбора файлов.
  - Путь к выбранному видеофайлу сохраняется в переменной **`video_file`** в формате **`Path(video_file[0])`**.
- **`split()`**:
  - Функция разделяет видеофайл на отдельные кадры.
  - Создает папку с именем, соответствующим имени исходного видеофайла, если такой папки еще нет.
  - Использует библиотеку **`moviepy`** для разделения видео на кадры с заданной частотой кадров (SAVING_FRAMES_PER_SECOND).
  - Каждый кадр сохраняется в папке с именем, соответствующим времени его появления в видео.
- **`choose()`**:
  - Функция загружает изображение с номерным знаком из диалогового окна выбора файлов.
  - Использует метод **`QFileDialog.getOpenFileName()`** для открытия диалогового окна выбора файлов.

- Путь к выбранному изображению сохраняется в переменной **`image_file`**.
- Изображение отображается на метке **`label_6`** с помощью метода **`setPixmap()`**.

– **`carplate_text()`**:
- Функция распознает номер автомобиля на изображении.
- Использует модели машинного обучения для обнаружения номерного знака и его распознавания.
- Производит предварительную обработку изображения, включая изменение размера, преобразование в оттенки серого, применение фильтра Canny для обнаружения границ и преобразование изображения в бинарное.
- Использует модель TensorFlow Lite для распознавания символов на номерном знаке.
- Возвращает распознанный номер автомобиля в виде строки.

– **`recognize()`**:
- Функция вызывается при нажатии на кнопку "Распознать".
- Вызывает функцию **`carplate_text()`** для распознавания номера автомобиля на выбранном изображении.
- Отображает результат распознавания на метке **`label_7`**.
- Обновляет статистику по количеству распознанных номеров и их вероятности.
- Проверяет формат распознанного номера и выводит сообщение об ошибке, если формат неверный.
- Если формат верный, сохраняет результат в массив **`arr`** и обновляет статистику на метках **`label_2`** и **`label_5`**.
- Передает полученный номер в компонент определения модели машины

4.4 Результат

На рисунке 13 представлено окно программы.

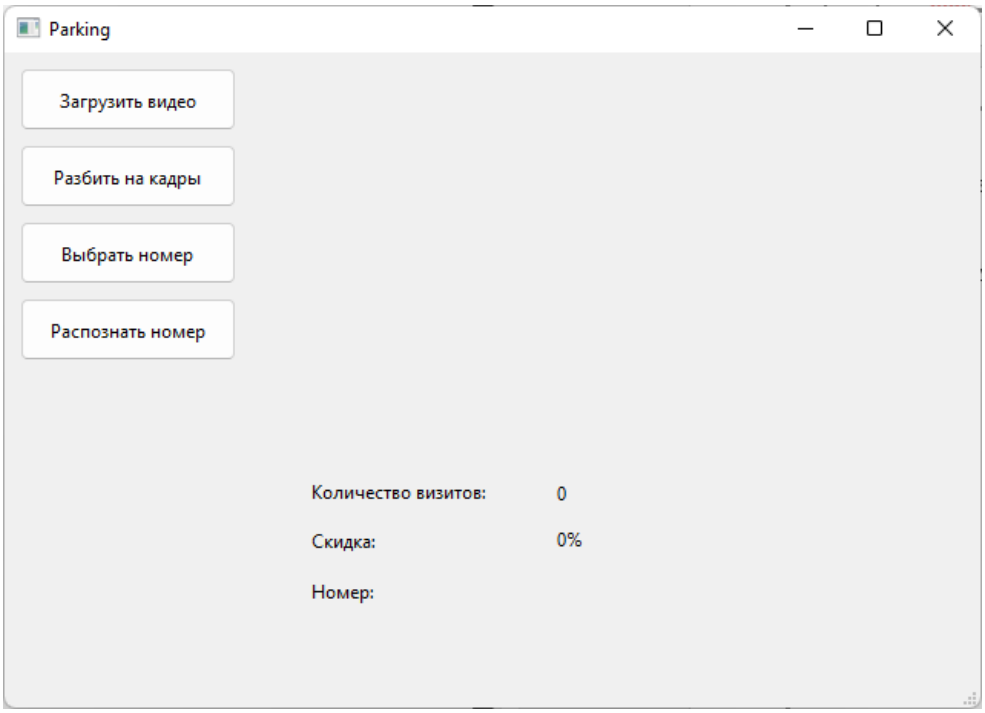

Рисунок 13 – Окно программы

Выбор изображения представлен на рисунке 14.

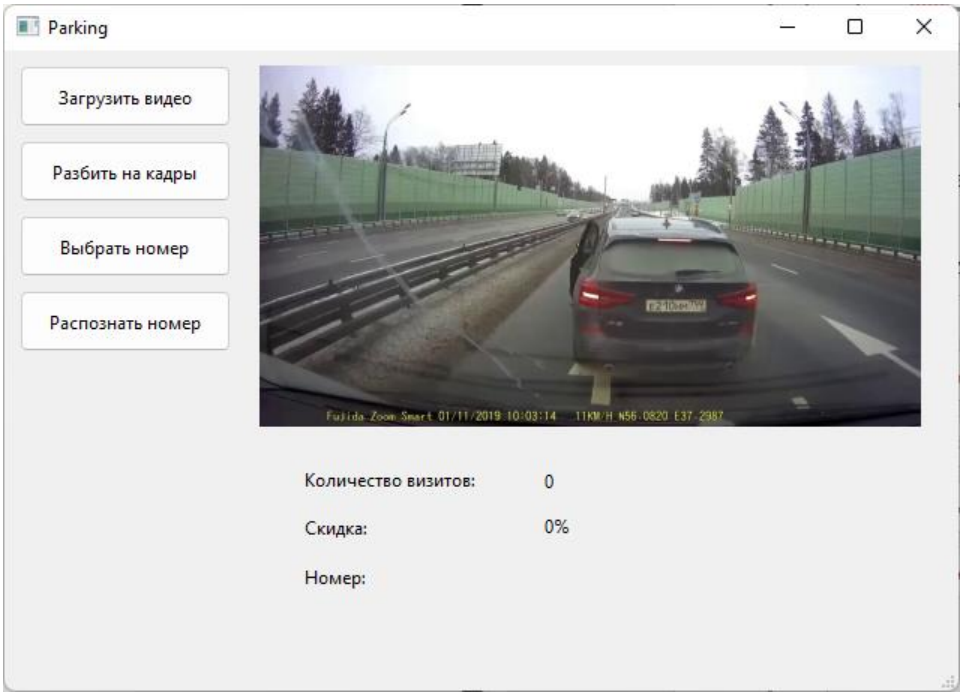

Рисунок 14 – Выбор изображения

Распознавание номера представлено на рисунке 15.

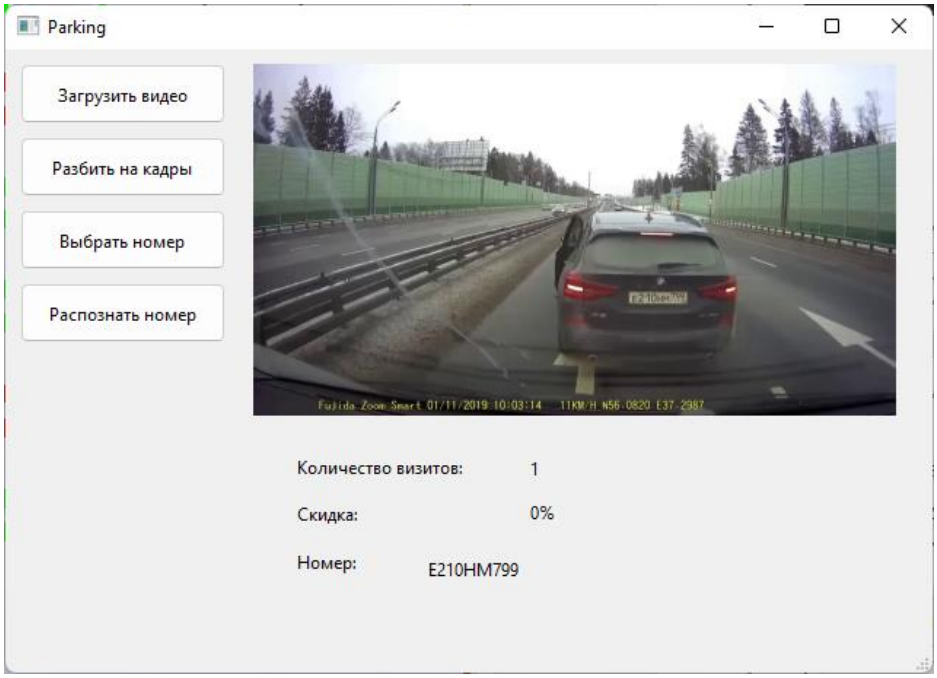

Рисунок 15 – Распознавание номера

Дальше происходит переход на сайт для определения модели автомобиля по распознанному номеру (рисунок 16).

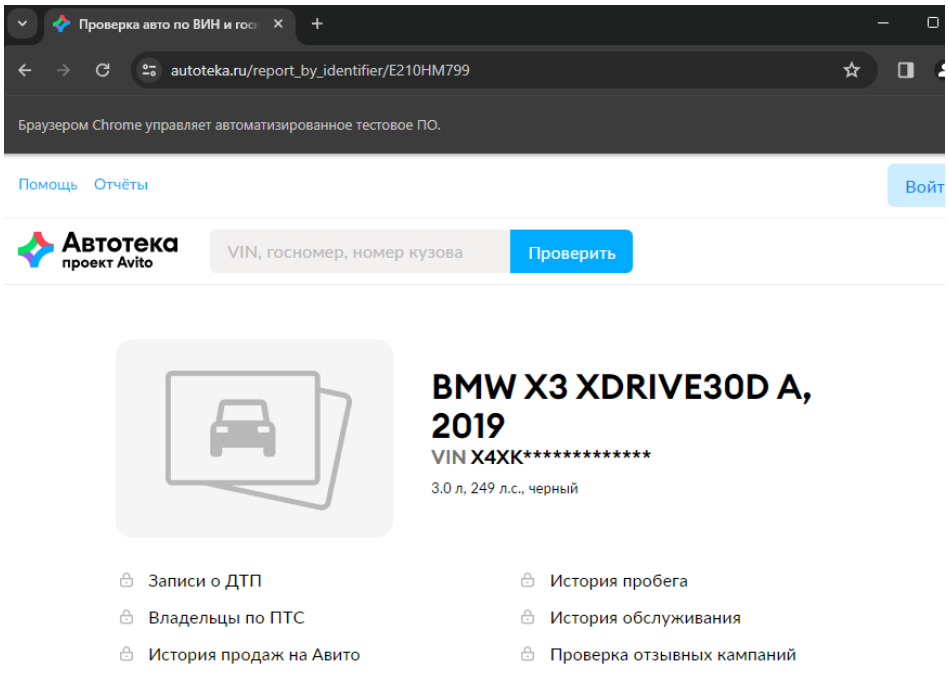

Рисунок 15 – Определение модели авто

После этого предлагается выбрать изображение с бланком (рисунок 17).

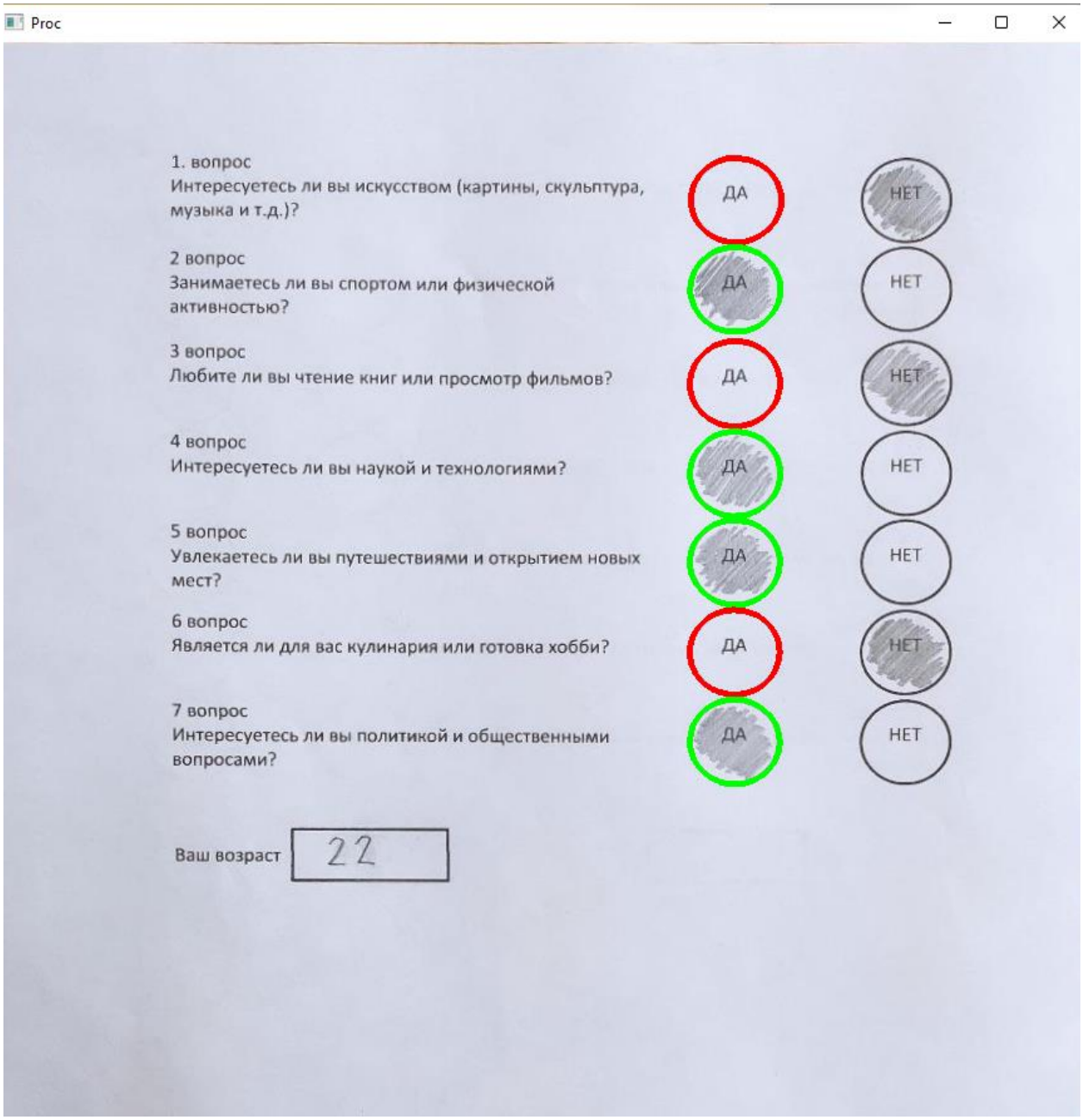

Рисунок 17 – Бланк опроса

На бланке происходит обводка ответов. красным обводится в случае ответа НЕТ, а зеленым – в случае ответа ДА. Так же происходит считывание возраста, указанного в специальном поле.

Для удобства отладки происходит вывод в консоль (рисунок 18).

```
BMW X3 XDRIVE30D A
recognize
[0, 1, 0, 1, 1, 0, 1]
22
['BMW X3 XDRIVE30D A', 22, 0, 1, 0, 1, 1, 0, 1]
```

Рисунок 18 – Работа программы

Как можно увидеть по рисунку 18, была определена модель автомобиля, так же выведен массив, содержащий ответы на вопросы. «0» означает ответ НЕТ, а «1» - ответ ДА. Кроме того, можно увидеть распознанный возраст «22». В конце все данные объединяются в единое целое, для записи в датасет – файл формата CSV. Результат записи представлен на рисунке 19.

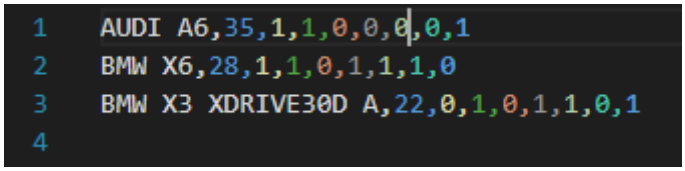

Рисунок 19 – Датасет

Вывод

В данном разделе была описана программная реализация разработанного модуля и процесс его работы.

Заключение

В ходе выполнения проекта был разработан программный модуль, который позволяет собирать информацию об интересах посетителей парковки. Для этого были использованы методы компьютерного зрения и обработки опросных данных.

Основной целью проекта была автоматизация процесса сбора информации об интересах посетителей парковки. Для достижения этой цели были поставлены и успешно выполнены следующие задачи:

- Выявление наиболее оптимального метода сбора данных.
- Разработка алгоритма анализа полученных данных для определения интересов посетителей.
- Создание скрипта для извлечения информации о модели автомобиля.
- Интеграция разработанных компонентов в единый функциональный модуль.

В процессе работы был проведен анализ предметной области, выявлены актуальность темы и ключевые требования. Были рассмотрены аналоги существующих программных продуктов, выявлены их принципы работы.

Для реализации функционала модуля были разработаны и реализованы следующие компоненты: анализатор бланков, поиск модели автомобиля и распознавание номера.

Анализатор бланков позволяет автоматически обрабатывать изображения анкет, извлекать информацию об ответах на вопросы и возрасте. Для этого используются технологии обработки изображений и распознавания текста.

Компонент поиска модели автомобиля обеспечивает автоматизированный доступ к информации о моделях автомобилей. Для этого используется взаимодействие с сайтом, предоставляющим данные о транспортных средствах.

Компонент распознавания номера автомобиля служит для определения номера на изображении. Для этого используется предобученная модель машинного обучения.

В результате работы был разработан готовый прототип программного модуля, который может быть использован для сбора информации об интересах посетителей парковки.

Таким образом, проект "Разработка программного модуля для сбора информации об интересах посетителей парковки" был успешно выполнен. Разработанный модуль предоставляет возможность автоматизировать процесс сбора и анализа данных, что позволяет повысить эффективность работы парковки и улучшить уровень сервиса.

Список литературы

1. PyImageSearch Bubble sheet multiple choice [Электронный ресурс]: https://pyimagesearch.com/2016/10/03/bubble-sheet-multiple-choice/

2.tutorialspoint Сбор данных [Электронный ресурс]: https://www.tutorialspoint.com/data-collection

3. studopedia Методы сбора данных [Электронный ресурс]: https://studopedia.su/3_37859_metodi-sbora-dannih.html

4. Habar Парсинг на Python [Электронный ресурс]: https://habr.com/ru/companies/selectel/articles/754674/

5. vc.ru Освоение OpenCV с помощью Python: Полное руководство по обработке изображений и компьютерному зрению [Электронный ресурс]:https://vc.ru/u/1389654-machine-learning/661520-osvoenie-opencv-s-pomoshchyu-python-polnoe-rukovodstvo-po-obrabotke-izobrazheniy-i-kompyuternomu-zreniyu

Приложение А

(Обязательное)

Схемы алгоритмов

Рисунок 20 – ГСА анализатора

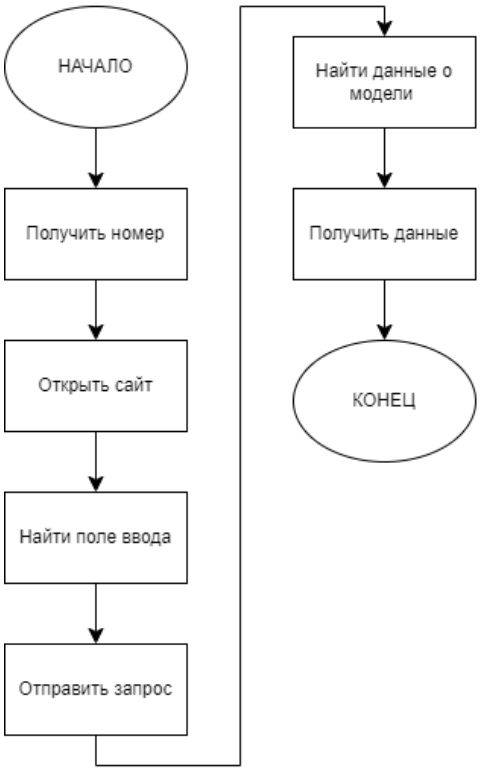

Рисунок 21 – ГСА поиска модели

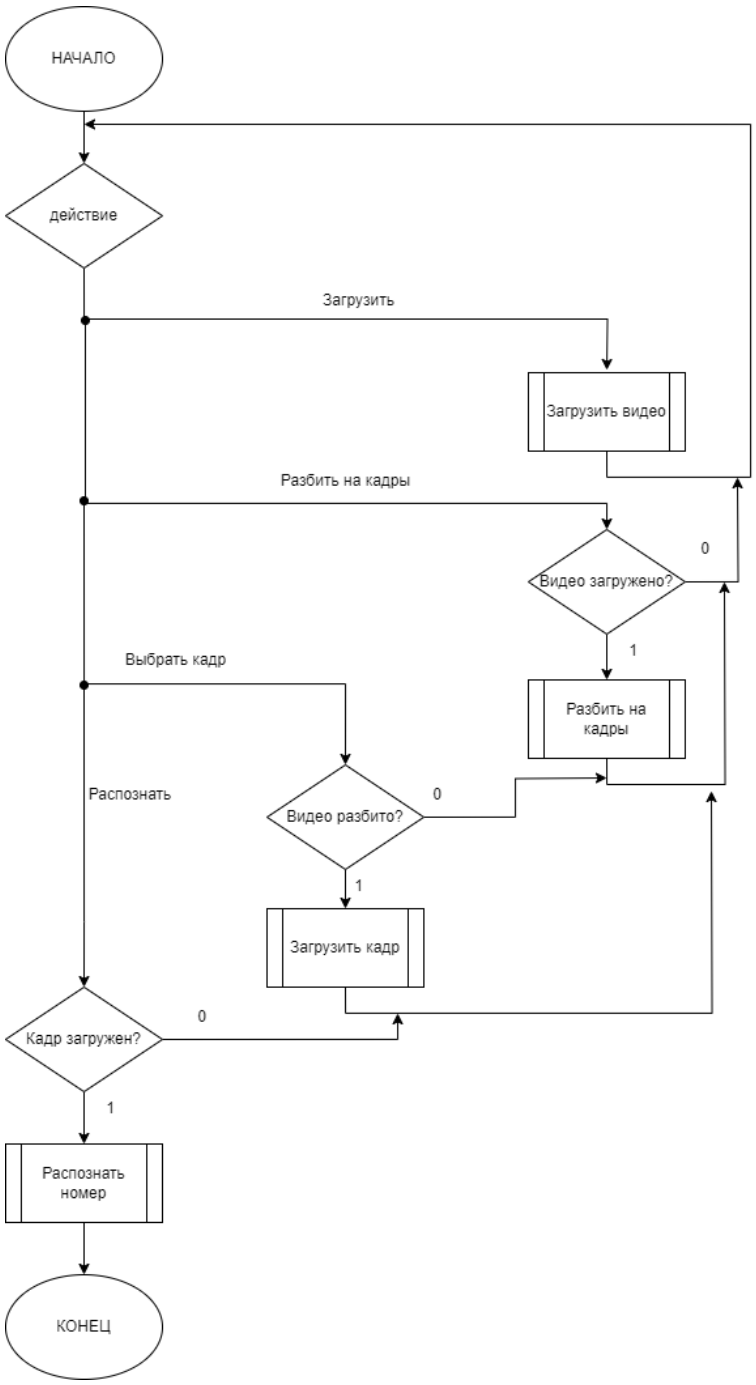

Рисунок 22 – ГСА распознавателя номеров

Приложение Б

(Обязательное)

Экранные формы

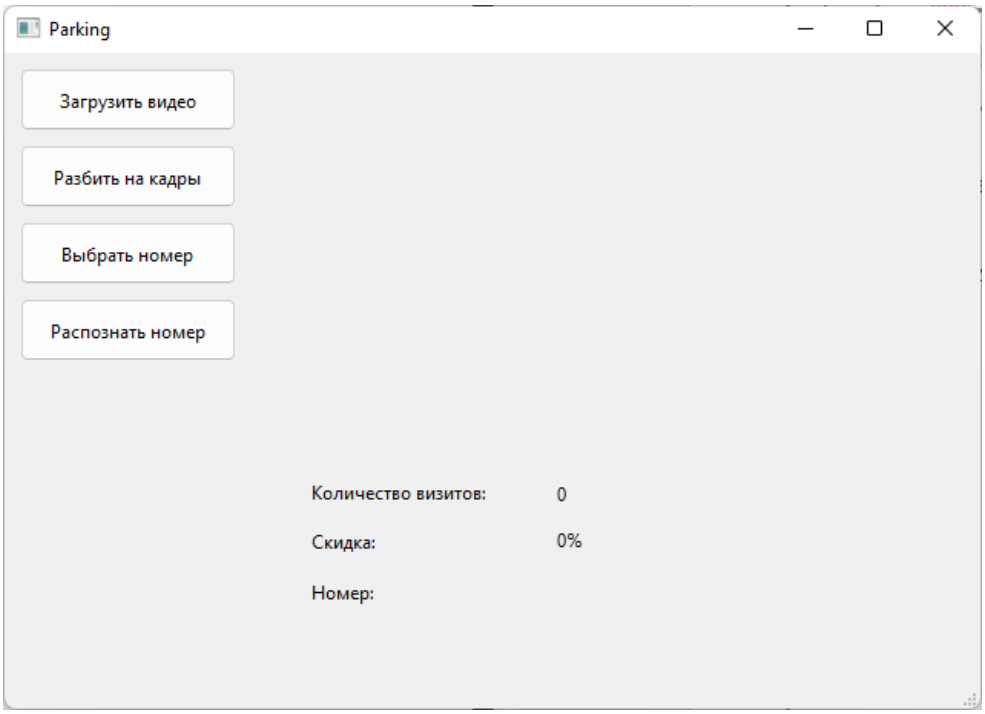

Рисунок 23 – Окно программы

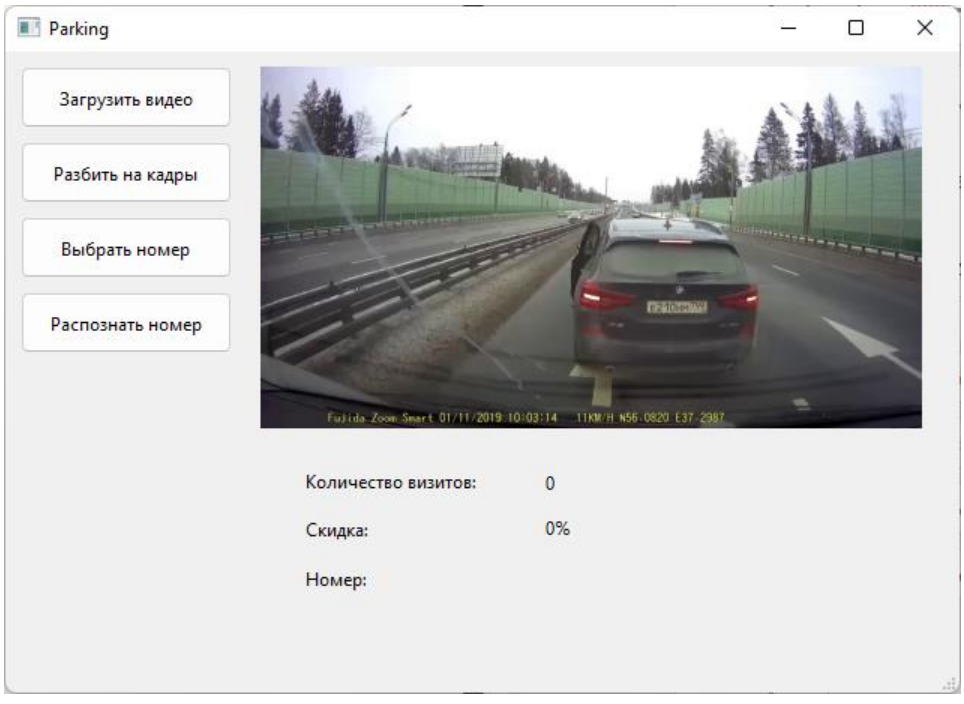

Рисунок 24 – Выбор изображения

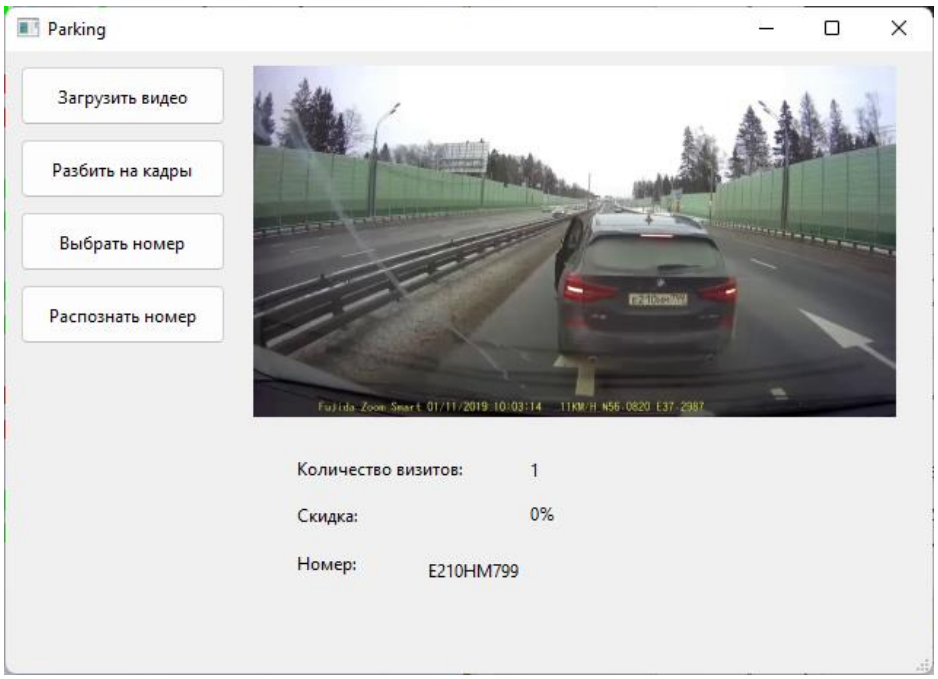

Рисунок 25 – Распознавание номера

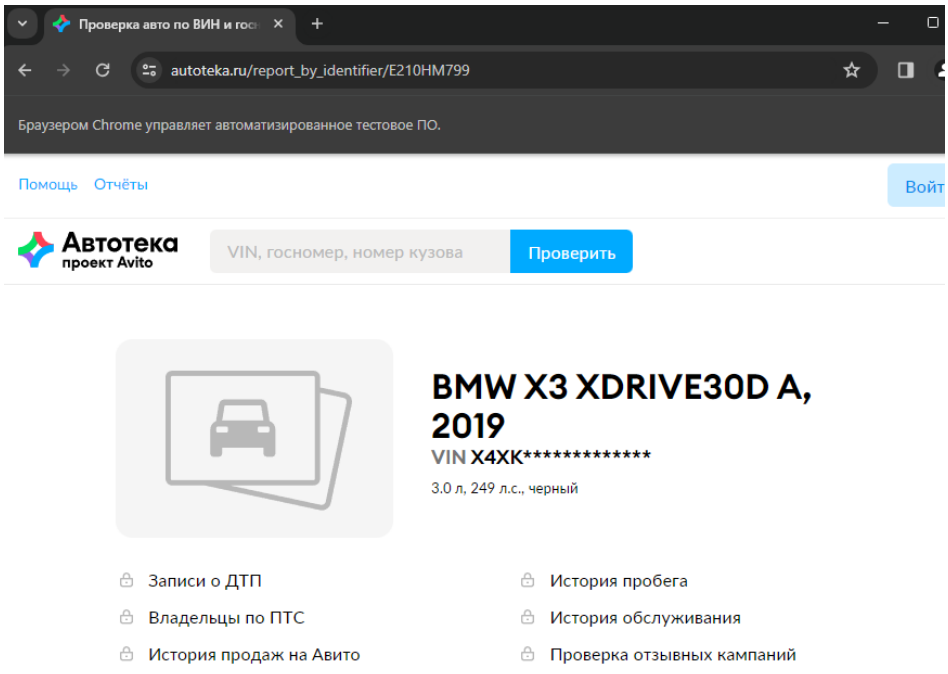

Рисунок 26 – Определение модели авто

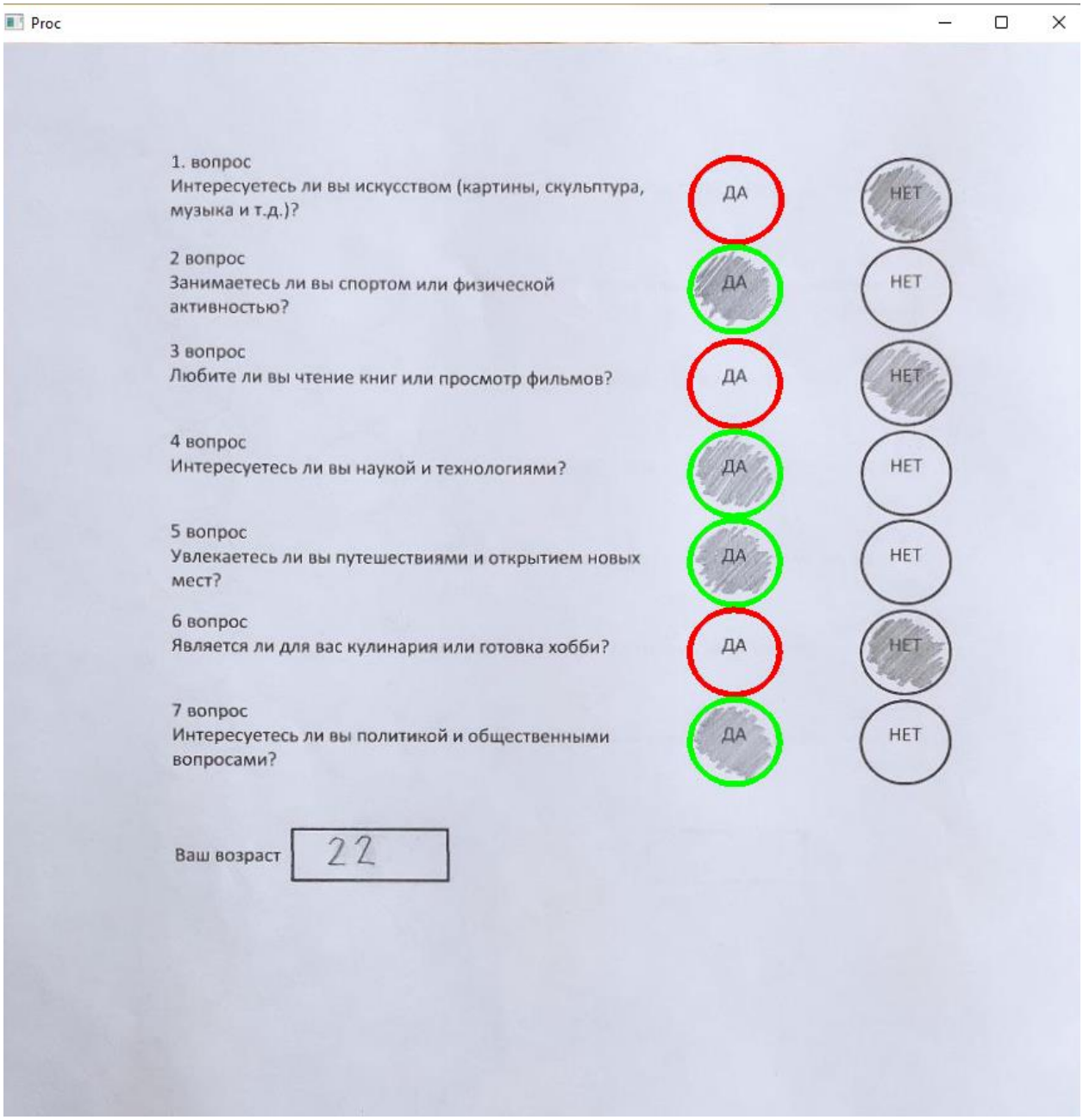

Рисунок 27 – Бланк опроса

```
BMW X3 XDRIVE30D A
recognize
[0, 1, 0, 1, 1, 0, 1]
22
['BMW X3 XDRIVE30D A', 22, 0, 1, 0, 1, 1, 0, 1]
```

Рисунок 28 – Работа программы

Приложение В

(Обязательное)

Структура программы

Рисунок 29 – Контекстная диаграмма IDEF0

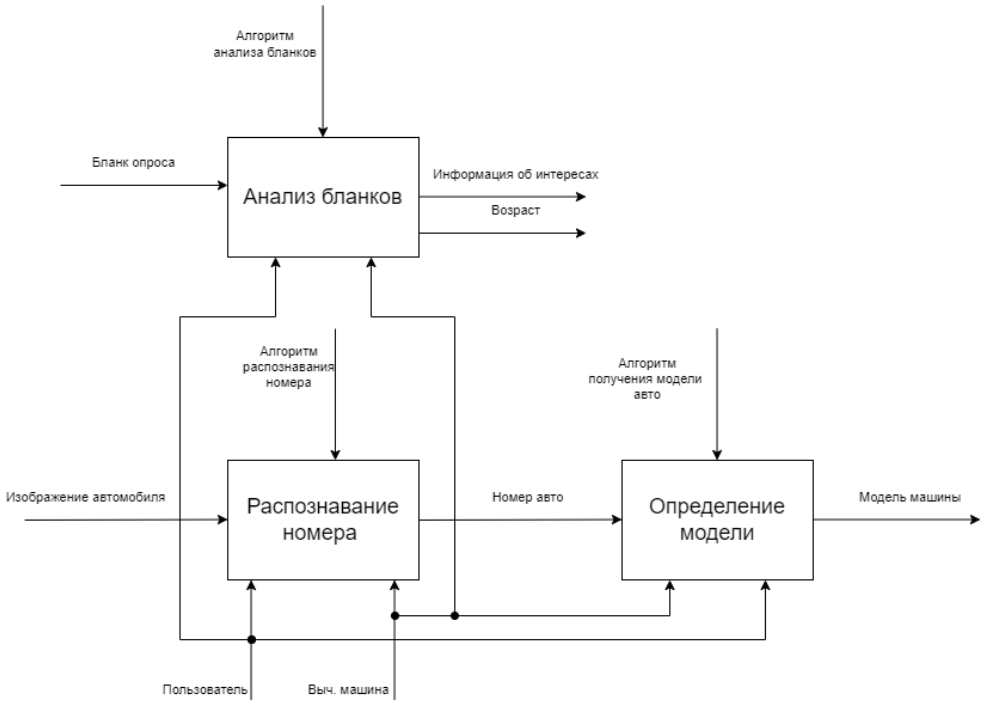

Рисунок 30 – Детализированная контекстная диаграмма IDEF0

Приложение Г

(Обязательное)

Структура датасета

model	age	art	sport	book/film	science	travel	cooking

Рисунок 31 – Структура датасета

```
AUDI A6,35,1,1,0,0,0,0,1
BMW X6,28,1,1,0,1,1,1,0
BMW X3 XDRIVE30D A,22,0,1,0,1,1,0,1

```

Рисунок 32 – Пример заполнения

Приложение Д

(Обязательное)

Программный код

```
from imutils.perspective import four_point_transform
from imutils import contours
import numpy as np
import argparse
import imutils
import cv2
import csv
import re
import pytesseract
ANSWER_KEY = {0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0}
pytesseract.pytesseract.tesseract_cmd = r'C:\\Program Files\\Tesseract-OCR\\tesseract.exe'
custom_config = r'--oem 3 --psm 6'
def process_exam(image_path):
    image = cv2.imread(image_path)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)
    edged = cv2.Canny(blurred, 75, 200)

    cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    docCnt = None

    if len(cnts) > 0:
        cnts = sorted(cnts, key=cv2.contourArea, reverse=True)
        for c in cnts:
            peri = cv2.arcLength(c, True)
            approx = cv2.approxPolyDP(c, 0.02 * peri, True)
            if len(approx) == 4:
                docCnt = approx
                break

    paper = four_point_transform(image, docCnt.reshape(4, 2))
    warped = four_point_transform(gray, docCnt.reshape(4, 2))
```

```
        thresh = cv2.threshold(warped, 0, 255, cv2.THRESH_BINARY_INV |
cv2.THRESH_OTSU)[1]

        cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        questionCnts = []

        for c in cnts:
            (x, y, w, h) = cv2.boundingRect(c)
            ar = w / float(h)
            #if (w >= 20) and (h>=20) and (ar >= 0.9) and (ar <= 1.1):
            if (w >= 50) and (h>=50) and (ar >= 0.9) and (ar <= 1.1):
                questionCnts.append(c)

        questionCnts = contours.sort_contours(questionCnts, method="top-to-bottom")[0]
        results = []

        for (q, i) in enumerate(np.arange(0, len(questionCnts), 2)):
            cnts = contours.sort_contours(questionCnts[i:i + 2])[0]
            bubbled = None
            question_result = 0

            for (j, c) in enumerate(cnts):
                mask = np.zeros(thresh.shape, dtype="uint8")
                cv2.drawContours(mask, [c], -1, 255, -1)

                mask = cv2.bitwise_and(thresh, thresh, mask = mask)
                total = cv2.countNonZero(mask)
                if bubbled is None or total > bubbled[0]:
                    bubbled = (total, j)

            color = (0, 0, 255)
            k = ANSWER_KEY[q]

            if k == bubbled[1]:
                question_result = 1
```

```
                color = (0, 255, 0)

            cv2.drawContours(paper, [cnts[k]], -1, color, 3)
            results.append(question_result)

        print(results)

        cv2. imshow("Proc", paper)
        cv2.waitKey(0)

        return results

    def get_age(image_path):
        image = cv2. imread(image_path)
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        blurred = cv2.GaussianBlur(gray, (5, 5), 0)
        edged = cv2.Canny(blurred, 75, 200)

        cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        docCnt = None

        if len(cnts) > 0:
            cnts = sorted(cnts, key=cv2.contourArea, reverse=True)
            for c in cnts:
                peri = cv2.arcLength(c, True)
                approx = cv2.approxPolyDP(c, 0.02 * peri, True)
                if len(approx) == 4:
                    docCnt = approx
                    break

        paper = four_point_transform(image, docCnt.reshape(4, 2))

        gray = cv2.cvtColor(paper, cv2.COLOR_BGR2GRAY)
        blurred = cv2.GaussianBlur(gray, (5, 5), 0)
        edged = cv2.Canny(blurred, 75, 200)
```

```
        cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        docCnt = None

        if len(cnts) > 0:
            cnts = sorted(cnts, key=cv2.contourArea, reverse=True)
            for c in cnts:
                peri = cv2.arcLength(c, True)
                approx = cv2.approxPolyDP(c, 0.02 * peri, True)
                if len(approx) == 4:
                    docCnt = approx
                    break

        paper = four_point_transform(paper, docCnt.reshape(4, 2))

        gray = cv2.cvtColor(paper, cv2.COLOR_BGR2GRAY)
        thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
        #cv2.imshow("1", thresh)

        #cv2.waitKey(0)
        text = pytesseract.image_to_string(thresh, config=custom_config)
        number = re.findall(r'\b\d+\b', text)
        two_digit_numbers = ' '.join(num for num in number if len(num) == 2)
        print(two_digit_numbers)
        return two_digit_numbers


    def write_to_csv(filename, results):
        with open(filename, mode='a', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(results)

    def blank(path_in, path_out, model):
        results = process_exam(path_in)
        age = get_age(path_in)
        results.insert(0, int(age))
        results.insert(0, model)
```

```
    print(results)
    write_to_csv(path_out, results)
    print("done")


def main():
    ap = argparse.ArgumentParser()
    ap.add_argument("-i", "--image", required=True, help="path to the input image")
    ap.add_argument("-o", "--output", required=True, help="path to save the output CSV file")
    args = vars(ap.parse_args())
    results = process_exam(args["image"])
    age = get_age(args["image"])
    results.insert(0, int(age))
    print(results)
    write_to_csv(args["output"], results)

if __name__ == "__main__":
    main()


import os
from datetime import timedelta
from pathlib import Path
import matplotlib.pyplot as plt
import cv2
import numpy as np
import re
import imutils
import pytesseract
import pytesseract as tess
tess.pytesseract.tesseract_cmd = r'Tesseract-OCR\tesseract.exe'
from PyQt6 import uic
from PyQt6.QtGui import QPixmap
from PyQt6.QtWidgets import QApplication, QFileDialog, QMessageBox
from imutils import contours
from moviepy.editor import VideoFileClip
import json
import sqlite3
```

```
import itertools
import tensorflow as tf
from skimage.feature import canny
from skimage.transform import hough_line, hough_line_peaks
from skimage.transform import rotate
from skimage.color import rgb2gray

import matplotlib.gridspec as gridspec
import cv2

import autoteka
import test_blank




SAVING_FRAMES_PER_SECOND = 1

Form, Window = uic.loadUiType("Parking.ui")

app = QApplication([])
window = Window()
form = Form()
form.setupUi(window)
window.show()

video_file = ''
image_file = ''
result = ''
arr=[]

def format_timedelta(td):
    result = str(td)
    try:
        result, ms = result.split(".")
    except ValueError:
        return result + ".00".replace(":", "-")
```

```
            ms = round(int(ms) / 10000)
            return f"{result}.{ms:02}".replace(":", "-")


        def load():
            global video_file
            video_file= QFileDialog.getOpenFileName()
            path=Path(video_file[0])
            video_file=path.name
            print("load")

        def split():
            video_clip = VideoFileClip(video_file)
            filename, _ = os.path.splitext(video_file)

            if not os.path.isdir(filename):
                os.mkdir(filename)

            saving_frames_per_second = min(video_clip.fps, SAVING_FRAMES_PER_SECOND)
            step = 1 / video_clip.fps if saving_frames_per_second == 0 else 1 / saving_frames_per_second

            for current_duration in np.arange(0, video_clip.duration, step):
                frame_duration_formatted =
format_timedelta(timedelta(seconds=current_duration)).replace(":", "-")
                frame_filename = os.path.join(filename, f"frame{frame_duration_formatted}.jpg")

                video_clip.save_frame(frame_filename, current_duration)
            print("split")


        def check_format(variable):
            pattern = r'^[A-Z]\d{3}[A-Z]{2}\d{3}$'
            pattern2 = r'^[A-Z]\d{3}[A-Z]{2}\d{2}$'
            if re.match(pattern, variable) or re.match(pattern2, variable):
                return True
            else:
                return False
```

```
def choose():
    global image_file
    image_file = QFileDialog.getOpenFileName()
    image_file = image_file[0]
    form.label_6.setPixmap(QPixmap(image_file))
    form.label_6.setScaledContents(True)
    print("choose")


def carplate_text():
    global image_file

    image0 = cv2.imread(image_file)



    image_height, image_width, _ = image0.shape
    image = cv2.resize(image0, (1024, 1024))
    image = image.astype(np.float32)
    paths = './model_resnet.tflite'
    interpreter = tf.lite.Interpreter(model_path=paths)
    interpreter.allocate_tensors()
    input_details = interpreter.get_input_details()
    output_details = interpreter.get_output_details()
    X_data1 = np.float32(image.reshape(1, 1024, 1024, 3))

    interpreter.set_tensor(input_details[0]['index'], X_data1)

    interpreter.invoke()
    detection = interpreter.get_tensor(output_details[0]['index'])
    net_out_value2 = interpreter.get_tensor(output_details[1]['index'])
    net_out_value3 = interpreter.get_tensor(output_details[2]['index'])
    net_out_value4 = interpreter.get_tensor(output_details[3]['index'])

    img = image0
    razmer = img.shape

    img2 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
img3 = img[:, :, :]

number = 0
while number < len(detection[0][number]) and detection[0, number, 0] > 0.9:
    number = number + 1

box_x = int(detection[0, number, 0] * image_height)
box_y = int(detection[0, number, 1] * image_width)
box_width = int(detection[0, number, 2] * image_height)
box_height = int(detection[0, number, 3] * image_width)

cv2.rectangle(img2, (box_y, box_x), (box_height, box_width), (230, 230, 21), thickness=5)

# plt.imshow(img2)
# plt.xticks([]), plt.yticks([])  # Hides the graph ticks and x / y axis
# plt.show()

net_out_value3

image = image0[box_x:box_width, box_y:box_height, :]
img2 = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

grayscale = rgb2gray(image)

edges = canny(grayscale, sigma=3.0)

out, angles, distances = hough_line(edges)
h, theta, d = out, angles, distances
angle_step = 0.5 * np.diff(theta).mean()
d_step = 0.5 * np.diff(d).mean()
bounds = [np.rad2deg(theta[0] - angle_step),
          np.rad2deg(theta[-1] + angle_step),
          d[-1] + d_step, d[0] - d_step]

_, angles_peaks, _ = hough_line_peaks(out, angles, distances, num_peaks=20)
angle = np.mean(np.rad2deg(angles_peaks))
angle
```

```
if 0 <= angle <= 90:
    rot_angle = angle - 90
elif -45 <= angle < 0:
    rot_angle = angle - 90
elif -90 <= angle < -45:
    rot_angle = 90 + angle
if abs(rot_angle) > 20:
    rot_angle = 0

rotated = rotate(image, rot_angle, resize=True) * 255
rotated = rotated.astype(np.uint8)

rotated1 = rotated[:, :, :]
if rotated.shape[1] / rotated.shape[0] < 2:
    minus = np.abs(int(np.sin(np.radians(rot_angle)) * rotated.shape[0]))
    rotated1 = rotated[minus:-minus, :, :]
    print(minus)

lab = cv2.cvtColor(rotated1, cv2.COLOR_BGR2LAB)
l, a, b = cv2.split(lab)
clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))
cl = clahe.apply(l)
limg = cv2.merge((cl, a, b))
final = cv2.cvtColor(limg, cv2.COLOR_LAB2BGR)

letters = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'E', 'H', 'K', 'M', 'O', 'P', 'T', 'X',
           'Y']

def decode_batch(out):
    ret = []
    for j in range(out.shape[0]):
        out_best = list(np.argmax(out[j, 2:], 1))
        out_best = [k for k, g in itertools.groupby(out_best)]
        outstr = ''
        for c in out_best:
            if c < len(letters):
                outstr += letters[c]
        ret.append(outstr)
```

```
    return ret

paths = './model1_nomer.tflite'
interpreter = tf.lite.Interpreter(model_path=paths)
interpreter.allocate_tensors()

input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
img = final
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img = cv2.resize(img, (128, 64))
img = img.astype(np.float32)
img /= 255

img1 = img.T
img1.shape
X_data1 = np.float32(img1.reshape(1, 128, 64, 1))
input_index = (interpreter.get_input_details()[0]['index'])
interpreter.set_tensor(input_details[0]['index'], X_data1)

interpreter.invoke()

net_out_value = interpreter.get_tensor(output_details[0]['index'])
pred_texts = decode_batch(net_out_value)
pred_texts

fig = plt.figure(figsize=(10, 10))
outer = gridspec.GridSpec(2, 1, wspace=10, hspace=0.1)
ax1 = plt.Subplot(fig, outer[0])
fig.add_subplot(ax1)
ax2 = plt.Subplot(fig, outer[1])
fig.add_subplot(ax2)
return pred_texts[0]

def recognize():
    global result
    result=carplate_text().upper()
    print(result)
```

```
if result==":
    QMessageBox.information(None, "Ошибка распознания", "НЕ РАСПОЗНАНО")
elif check_format(result)!=True:
    QMessageBox.information(None, "Ошибка формата", "Результат:"+result)
else:
    form.label_7.setText(result)

    count = 0
    for i in range(len(arr)):
        if arr[i][0] == result:
            count += 1

    if count == 0:
        arr.append([result, 1])
    else:
        for i in range(len(arr)):
            if arr[i][0] == result:
                arr[i][1] += 1

    for i in range(len(arr)):
        if arr[i][0] == result:
            form.label_2.setText(str(arr[i][1]))
            if (arr[i][1]<5):
                form.label_5.setText('0%')
            elif (arr[i][1]>=5 and arr[i][1]<10):
                form.label_5.setText('5%')
            elif (arr[i][1]>=10 and arr[i][1]<20):
                form.label_5.setText('10%')
            else:
                form.label_5.setText('15%')

model = autoteka.site(result)
print(model)
print("recognize")

file_path, _ = QFileDialog.getOpenFileName()
test_blank.blank(file_path, 'data/data.csv', model)
```

```
        form.pushButton.clicked.connect(load)
        form.pushButton_2.clicked.connect(split)
        form.pushButton_3.clicked.connect(choose)
        form.pushButton_4.clicked.connect(recognize)


        app.exec()


from selenium import webdriver
        from selenium.webdriver.common.keys import Keys
        from selenium.webdriver.common.by import By
        from selenium.webdriver.support.ui import WebDriverWait  # Импортируем WebDriverWait
        from selenium.webdriver.support import expected_conditions as EC
        import time

        driver_path = 'drv/chromedriver-win64/chromedriver.exe'

        website_url = 'https://autoteka.ru/'

        num = 'E210HM799'


        chrome_service = webdriver.chrome.service.Service(driver_path)

        driver = webdriver.Chrome(service=chrome_service)


        def site(number):
            input = number
            driver.get(website_url)

            input_field = driver.find_element(By.NAME, 'identifier')
            input_field.send_keys(input)
            input_field.send_keys(Keys.ENTER)
```

```
    WebDriverWait(driver, 15).until(EC.url_changes(website_url))

    time.sleep(3)

    text_element = driver.find_element(By.CLASS_NAME, 'pit4K')
    text = text_element.text

    time.sleep(3)

    lines = text.split('\n')
    result = ' '.join(lines[:2])
    #print(result.split(',')[0].strip())



    #driver.quit()

    return result.split(',')[0].strip()

if __name__ == "__main__":
    print(site(num))
```