**Федеральное государственное автономное**

**образовательное учреждение высшего образования**

**«Национальный исследовательский университет**

**«Высшая школа экономики»**

**Факультет математики, информатики и компьютерных наук**

***«Разработка мобильного приложения-помощника для слабовидящих покупателей»***

***Курсовая работа***

по направлению подготовки 09.03.04 «Программная инженерия»

Студент группы 18-ПИ2

Данилов С. Д.

Научный руководитель

Старший преподаватель,

НН Базовая кафедра группы компаний "MERA"

Лейкин М.В.

Нижний Новгород, 2021

**Оглавление**

**Введение**

Рынок современных мобильных приложений стремительно растет. С каждым годом создается все больше библиотек, инновационных решений в сфере дизайна, интерфейса, функциональности. Смартфон стал незаменимой частью нашей жизни во многих сферах жизни - это и доставка еды, и вызов такси, покупка билетов. Все это явно улучшает качество жизни, делает обыденные вещи проще.

При всем многообразии решений в сфере мобильных технологий, не многие продукты адаптированы к использованию людьми с ограниченными возможностями, такими как нарушения зрения, слуха, ориентирования и так далее. Пользователю с перечисленными нарушениями важны такие вспомогательные возможности, как крупный, удобочитаемый шрифт, озвучивание текста, простой и понятный интерфейс и дополнительные улучшения, упрощающие работу с приложением Многие компании разработчики программного обеспечения просто игнорируют этот факт, что потенциальным пользователем может быть пользователь с отклонениями по здоровью, так как адаптация – это дорогой процесс, затратный по времени и требует дополнительных знаний. К счастью, многие исследовательские компании начали понимать необходимость поддержки решений для людей с ограниченными возможностями, и такой параметр, как accessibility (англ. доступность) стал неотъемлемым пунктом в оценке качества готового приложения (не только мобильного, но и веб приложения) Только если приложения, которые упрощают рутину во всех сферах жизни, станут более доступны и удобны людям с ограничениями по здоровью, жизнь таких людей станет намного приятнее, полноценнее, они смогут познавать и осуществлять деятельность наравне с обычными пользователями. Разве не в этом смысл создания мобильных приложений?

**Актуальность выбранной темы**

Проблемами доступности приложений на данный момент озабочены в большей степени IT “гиганты” – компании Google, Apple, Facebook, . На данный момент существует большое количество приложений в секции "Распознавание текста" или распознавание предметов на фото. Но не так много приложений удовлетворяют высокому уровню удобства для использования им людям со специальными возможностями. Я уверен, что устранение этого недочетаочень упростит жизнь таким пользователям. Поэтому изучение рекомендаций по дизайну и технологиям для слабовидящих и применение полученных знаний при проектировании приложений является остро актуальной темой.

**Цели и задачи**

Цель данной работы - изучить технологии создания приложений для людей с нарушениями зрения и на основе этих знаний создать Android приложение для слабовидящих покупателей. Работа состоит из нескольких частей:

* Изучение пособий и материалов по проектированию интерфейсов мобильных приложений для людей с ограниченными возможностями, в частности - для пользователей с нарушениями зрения
* Приобретение знаний и навыков разработки мобильных приложений для платформы Android
* Изучение решений в области обработки изображений, распознавания и обработки текста, облачного хранения данных пользователей
* Проектирование приложения для слабовидящих пользователей на основе полученных знаний и его реализация

**Обзор аналогов**

На текущий момент мне не удалось найти приложение с полностью сходным функционалом и идентичной предметной областью, которое было бы размещено на платформе Google Play Market. Но несмотря на это, на данном ресурсе можно найти схожие приложения, адаптированные под использование слабовидящими пользователями и предоставляющие либо функционал распознавания информации с изображения либо голосовые подсказки.

* **TapTapSee (*CloudSight*)** – бесплатно, доступно для Android и iOS

Приложение позволяет распознавать объекты, окружающие пользователя, позволяя ему легко ориентироваться в пространстве. При наведении камеры на интересующий объект и двойном нажатии на экран, приложение озвучивает информацию о распознанном объекте. Также можно голосом попросить у приложения найти определенный предмет, и когда пользователь наведет камеру на необходимый объект, приложение оповестит пользователя.

* **Blind-Droid Wallet (Blind-Droid Apps)** – бесплатно, доступно для Android

Энтузиасты команды Blind-droid apps, занимающиеся созданием приложений для людей с проблемами зрения, разработали ассистент, позволяющий распознавать номинал купюр разных валют. Этот функционал очень важен для слабовидящих пользователей при оплате наличными в магазине, так как в таком случае легко ошибиться, не разобрав номинал купюры.

* **«Искусство. Вслух» (Rambler&Co, Сбербанк)** – бесплатно, доступно для Android и iOS

Разработка компании Rambler позволяет сделать искусство доступным для людей с ограниченными возможностями. Когда пользователь приходит в театр, приложение подключается к Wi-Fi сети театра и загружает информацию о спектакле. Пользователь может «видеть» спектакль с помощью тифлокомментарий, описывающих происходящее на сцене, которые выводятся на экран (что полезно для людей с проблемами со слухом) и дополнительно озвучиваются.

Также можно отметить приложения, которые имеют высокий уровень доступности. Такие приложения, как Google, Yandex, Sber, Tinkoff имеют собственные голосовые помощники, позволяющие легко совершать действия по поиску информации, заказу такси, проведению банковских операций, просто «общаясь» с приложением. Используя различные особенности и детали реализации данных приложений, я создам свое решение в области помощи пользователям с ограниченными возможностями.

**Анализ предметной области**

Я решил проанализировать свой жизненный опыт и узнать мнение людей о том, какие именно полезные функции должно иметь мое приложение, чтобы сделать процесс приобретение товаров наиболее удобным для пользователей с нарушениями зрения. Ни для кого не секрет, что у людей с плохим зрением возникают проблемы, когда они выбирают товар на полках магазина. Им проблематично узнать информацию с этикетки, такую как состав, срок годности, компания изготовитель, так как текст, содержащий эту информацию, зачастую мелкий или неразборчивый. Также в результате опроса было выявлено, что слабовидящие пользователи хотели бы иметь приложение, считывающее информацию о продукте с этикетки и с ценников на полке, и определяющее по сканированной информации цену продукта. Также во многих существующих решениях по распознаванию текста отсутствует озвучка, что приносит мало пользы от использования приложения. Я уверен, что функционал моего приложения поможет решить вышеописанные проблемы.

**Особенности приложений для слабовидящих**

При проектировании приложения для слабовидящих пользователей необходимо на самом раннем этапе выстроить особую стратегию построения дизайна и схемы взаимодействия между пользователем и программой. Главная мысль, которой нужно придерживаться при разработке стратегии – минимализм, фокус на звук, а не изображение и использование специальных возможностей. Представленные ниже требования можно также найти в национальном стандарте РФ ГОСТ Р 52872-2019 «Интернет-ресурсы и другая информация, представленная в электронно-цифровой форме. приложения для стационарных и мобильных устройств, иные пользовательские интерфейсы. требования доступности для людей с инвалидностью и других лиц с ограничениями жизнедеятельности»

* Минимализм

Что подразумевается под минимализмом если мы говорим о мобильных приложениях? Порой при использовании приложения мы можем столкнуться с большим количеством компонент интерфейса на одном экране или диалоге, такими как всевозможные кнопки, поля ввода, меню и так далее. Такое многообразие элементов усложняет процесс использования продукта, особенно если пользователь имеет проблемы со зрением. Именно в этом случае очень помогает соблюдение разумного минимализма, соблюдая который, мы оставляем только самое необходимое из интерфейса. К этому понятию можно также отнести отсутствие анимации и сложных элементов дизайна, так как их наличие может рассеять внимание пользователя и усложнить восприятие интерфейса, особенно если используется система озвучивания интерфейса TalkBack.

* Быстрое достижение цели пользователем

Также важным показателем качества приложения для слабовидящих является то, как быстро сможет пользователь достичь своей цели использования приложения. Порой поиск нужного элемента интерфейса, его распознавание и в целом ориентирование по приложению может занимать больше времени и сил для лиц с ограниченными возможностями. Если приложение не соответствует этому критерию качества, пользователь не сможет комфортно пользоваться продуктом, поэтому важно проработать сценарии использования и построить граф состояний приложения таким образом, чтобы путь к критически важным функциям был как можно меньше.

* Четкий текст и дизайн элементов

Эти два критерия являются необходимыми к соблюдению для всех приложений, не только к продуктам для людей с ограниченными возможностями, так как качественный дизайн делает процесс использования программного продукта намного приятнее. Но что касается типа приложений, описанных в данной работе, эти два критерия обретают больший смысл. В случае, если не используется система озвучивания интерфейса, пользователь будет ориентироваться на текст, символы, кнопки и т.д. , составляющие интерфейс, и важно, чтобы эти элемента интерфейса не были расплывчатыми и сложными к восприятию у людей с ограниченными возможностями. Поможет поддержать это требование соблюдение данных конкретных рекомендаций к дизайну:

* Контраст цветов (между текстом и фоном например)
* Крупный размер кегля и удобочитаемый шрифт
* Строгий дизайн элементов (минимальный угол, монотонная заливка, минимум анимации)
* Текстовое обозначение важных элементов интерфейса
* Голосовое сопровождение системами TalkBack\VoiceOver

Незаменимым помощником для слабовидящих пользователей мобильных устройств на базе ОС Android является система озвучки интерфейса TalkBack (аналог для операционной системы iOS – VoiceOver) Он позволяет озвучивать информацию о происходящем на экране в целом или о выделенном элементе интерфейса. Так пользователь с ограниченными возможностями по зрению сможет найти нужный ему элемент интерфейса или получить результат работы приложения. Понимая важность использования этой функции пользователем, следует максимально адаптировать приложение к этой системе, чтобы озвучиваемая информация на каждом экране была максимально содержательной, и взаимодействие с интерфейсом не представляло больших проблем пользователю.

* Использование специальных возможностей

Людям с ограниченными возможностями зачастую необходимы средства, которые бы давали возможность выполнять повседневные действия, познавать мир, пользоваться мобильными приложениями наравне со всеми. К счастью, на текущий момент создано множество решений в области искусственного интеллекта, которые своим функционалом восполняют те самые утраченные способности слышать, четко видеть и так далее. Одним из самых распространённых технологий в этой сфере является голосовой ассистент. Такая система позволяет распознавать речь пользователя и выполнять переданные голосом команды, отвечать на запросы сгенерированной речью и даже поддерживать общение. В мобильных приложениях функция голосового помощника помогает быстро выполнить определенное действие, сэкономив время на поиск интересующего раздела или выполнения последовательности действий. Также важной функцией является автоматическая генерация субтитров при воспроизведении видео или аудио материала, так как она позволяет пользователям с нарушением слуха воспринимать информацию наравне с обычными пользователями. И наконец, технологии компьютерного зрения существенно упрощают повседневные вещи благодаря считыванию штрих-кодов вместо ручного ввода информации, распознавания текста, объектов, описание изображения с помощью тифлокомментариев и так далее. В нашем приложении решающую роль играет распознавание текста на изображении, поэтому остановимся на нем подробнее.

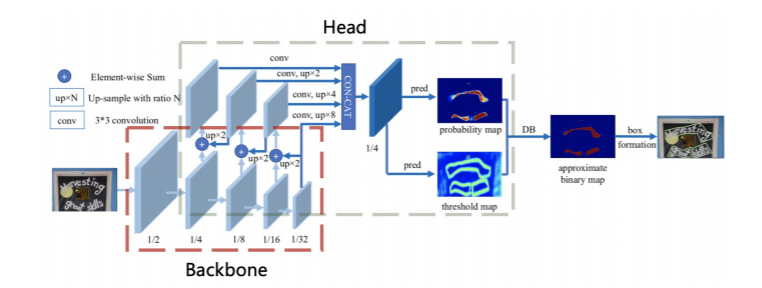
**Обзор технологий распознавания текста (OCR)**

Функционал разрабатываемого приложения включает в себя распознавание текста на изображении этикетки, поэтому следует подробно рассмотреть сам процесс обработки изображения и нахождения на нем символов.

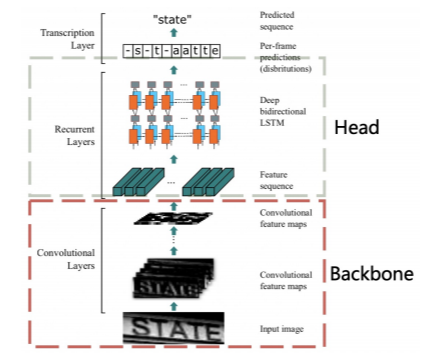
Система распознавания текста предполагает наличие на входе изображения с текстом (в формате данных графического файла). На выходе система должна выдать текст, выделенный из входных данных. Весь процесс распознавания текста состоит из нескольких задач.

Сначала изображение преобразуется из изображения RGB в оттенки серого. Это означает удаление цветов изображения и преобразование матрицы изображения из трех измерений в

только два измерения. Далее из изображения удаляются различные шумы, которые могут потенциально понизить качество распознавания, с помощью сглаживания. На отдельных участках изображения выбирается среднее между пикселями и это значение используется на этом участке. Также полезным методом повышения точности является инверсия цветов, когда значение цвета или оттенка меняется на противоположный. Таким образом достигается более четкое выделение текста. После такой обработки изображение готово для дальнейшего распознавания на нем текста.

Следующий шаг сегментации - важный процесс распознавания изображений. Используя сверточную нейронную сеть, мы определяем положение слов внутри изображения, а затем в отдельных секциях со словами разграничиваем положение слов и букв.

Остановимся более подробно на архитектуре сверточных нейронных сетей. Данные математические модели работают на основе фильтров, которые занимаются распознаванием определенных характеристик изображения. Фильтр - это обычная матрица чисел, называемых «весами», которые заранее «обучаются» на тестовых изображениях с размеченными данными. Фильтр перемещается последовательно по изображению и определяет, присутствует ли некоторая искомая характеристика в конкретной его части. Для получения ответа такого рода совершается операция свертки, которая является суммой произведений элементов фильтра и матрицы входных сигналов. Так последовательно происходит свертка массивов в конечный, где уже собирается информация о различных характеристиках всего изображения или его части. В нашем случае, как показано на рисунке, мы получаем участки, где предположительно может располагаться текст.



Далее области с текстом выделяются из исходного изображения в виде прямоугольников и передаются дальше алгоритму распознавания символов. При распознавании символов мы так же используем алгоритм свертки, но уже к выделенному участку с текстом. Получившуюся последовательность характеристик мы отдаем заранее обученной нейросети, которая имеет несколько выходов , каждый выход относится к определенному символу и описывает вероятность , что анализируемый участок это и есть тот самый символ.

Так последовательно обрабатывается каждый текстовый блок, найденный на изображении и готовый текст передается для дальнейшего использования.

Функционал распознавания текста предоставляют несколько библиотек и сервисов, которые могут работать как на устройстве, так и в «облаке», обрабатывая в последнем случае изображение на сервере компании. Рассмотрим некоторые из них:

* Библиотека Tesseract OCR

Tesseract — это движок оптического распознавания символов (OCR) с открытым исходным кодом. Плюсами данной библиотеки можно отметить обученные языковые модели (>192), разные виды распознавания (изображение как слово, блок текста, вертикальный текст), легкая настройка. Так как Tesseract OCR написан на языке C++, для других языков используются сторонние библиотеки-обертки. Минусом данного подхода является отсутствие автоматического определения языка текста на изображении, что дает существенные ошибки в результате распознавания текста.

* Google Firebase ML and ML Kit

Сервис Google Firebase ML предоставляет функционал анализа изображений и нахождении на них объектов, подбор лейблов для изображений и распознавание текста. У этого сервиса есть мощный API, адаптированный под Android, iOS, Desktop разработку и поддерживающий различные языки программирования. Плюсами данного сервиса являются высокое качество распознавания и хорошая, постоянно обновляющаяся документация. Также существует библиотека ML Kit для обработки данных на конечном устройстве, такой подход дает большую скорость обработки. Но главный ее минус в том, что она не поддерживает другие языки кроме английского, что не подходит для нашего приложения.

Сравнив эти решения в области компьютерного зрения, я решил использовать облачный сервис по распознаванию текста на изображении, входящий в состав Google Firebase ML.

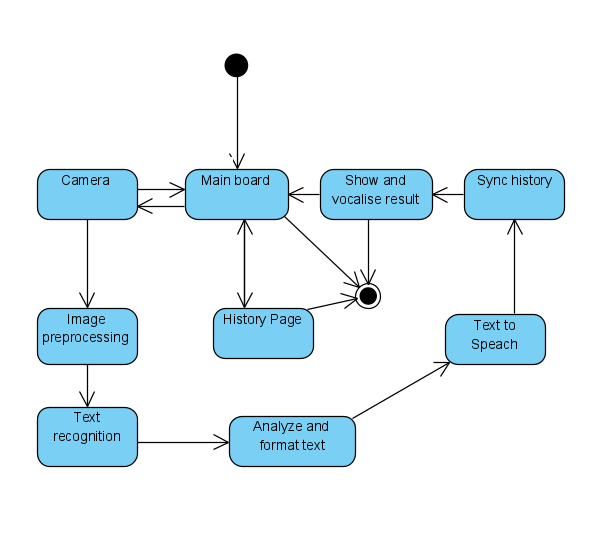
**Средства озвучивания текста**

Впервые технологии озвучивания текста были внедрены в ПК компанией Apple в 1984 г. С тех пор качество синтезации речи заметно улучшилось, сейчас практически невозможно отличить речь обычного человека и голосового помощника Google. Это заметно улучшило процесс использования гаджетов, теперь многие команды можно давать приложению голосом, не тратя время на взаимодействие с интерфейсом. В особенности стало удобнее лицам с ограниченными возможностями – пользователям с нарушением зрения не обязательно видеть то, что находится на экране приложения, во всем им помогает система VoiceOver (TalkBack в случае ОС Андроид), которая озвучивает текст и выбранные элементы интерфейса. Рассмотрим подробнее, как работает эта система.

Сначала проходит процесс пред-обработки, на котором удаляются лишние символы, которые могут мешать восприятию синтезированной речи. Далее текст разбивается на слова, а слова на фонемы. Между фонемами и звуками есть небольшие отличия. В выбранном языке может быть несколько вариантов звучания одного и того же звука, в зависимости от того, рядом с какими звуками стоит символ или в каком месте слова он находится, эти варианты и называется фонемами. Существует два способа озвучивания последовательности фонем – с помощью заранее записанного реального голоса, где различные фонемы извлекаются из речи, либо с помощью программной генерации звука. Дополнительно можно выбрать пол говорящего, тембр, скорость, частоту. На операционной системе Android функцию синтезации речи выполняет библиотека TTS (Text-to-Speech), именно ее мы и будем использовать. У объекта, осуществляющего доступ к TTS, вызывается метод speak, в аргументы которого передается текст для озвучивания. Этот метод озвучивает текст в отдельном потоке, не блокируя основной функционал приложения. В любой момент можно остановить процесс воспроизведения аудио командой stop(). TTS поддерживает русский язык, поэтому он превосходно подойдет к нашему приложению.

**Проектирование приложения**

Основной функционал приложения включает в себя распознавание текста с этикетки продукта, его обработка и озвучивание. Ниже представлена машина состояний приложения.



Сначала пользователь попадает на главный экран, где ему предлагается сделать новое фото. По нажатии кнопки «Сделать фото» вызывается диалог камеры (Camera), где пользователь делает снимок этикетки продукта. Далее созданный снимок пересылается в модуль предобработки для применении фильтров. После приложение пересылается в облачный сервис Google Firebase ML для распознавания текста или в результате ошибки приложение возвращается на главный экран и показывает диалог об ошибке. Распознанный текст обрабатывается и избавляется от мусорных символов и ошибочных слов. На основе готового текста генерируется аудиофайл, которой позже будет озвучиваться. Результаты распознавания далее сохраняются на устройстве в базе данных в виде такой модели:

|  |
| --- |
| Id |
| Дата |
| Путь до изображения |
| Распознанный текст |
| Цена (может отсутствовать) |

Готовая распознанная информация в итоге отображается на главном экране. Дополнительно пользователь может отсканировать ценники, чтобы найти цену распознанного продукта. Если часть текста ценника совпадает с текстом на этикетке, то в информацию о продукте добавляется цена с ценника. Для распознавания ценника используется все та же последовательность действий.

И наконец, пользователь может просмотреть историю сканирований, нажав на главном экране кнопку «Посмотреть историю сканирований». Из базы данных тогда будут выгружаться результаты последних сканирований, которые будут отображаться на экране и озвучиваться.

**Практическая часть**

Приступим к реализации приложения, опираясь на подготовленный проект.

* В качестве среды разработки была использована IDE Android Studio 4 от компании Google, она содержит весь необходимый функционал для построения интерфейса и функциональной части приложения на платформе Android.
* Языком разработки был выбран Java.
* Тестирование приложения я проводил на устройстве Huawei Honor на базе операционной системы Android Oreo.
* В качестве системы контроля версий мной был использован сервис GitHub, на базе этого сервиса я также настроил систему CI для постоянного контроля работоспособности приложения с помощью GitHub Actions.
* Для построения дизайна приложения я использовал графический редактор Figma
* В качестве провайдера облачных сервисов была использована платформа Google Cloud Services

1. Создадим проект на Google Firebase

Нашему приложению понадобится доступ к облачным сервисам компании Google. Я выбрал именно эту компанию в качестве провайдера облачных технологий и решений в системе искусственного интеллекта по нескольким причинам – во-первых, корпорация предлагает очень низкую цену предоставления услуг и предоставляет полностью прозрачную систему начисления платежей. Также важным фактором является большая популярность применения облачных ИИ технологий и баз данных в разработке IT продуктов по всему миру, таких как Telegram, VK, . Конкретно нашему приложению понадобятся – распознавание текста с изображения, перевод со стороны искусственного интеллекта и доступ к базе данных для хранения информации пользователей.

1. Опишем view модель главного экрана

Руководствуясь правилами дизайна, описанными в теоретической части, создадим макет главного экрана. Интерфейс приложения будет состоять из таких элементов, как:

* Кнопка создания фото
* Кнопка перехода к истории сканирований
* Поле с изображением сканированной этикетки
* Поле с текстом

Кнопку с созданием нового фото я специально поместил в нижней части экрана, чтобы этот элемент можно было использовать без поиска по экрану, машинально. Кнопку с переходом к истории сканирований поместим повыше, так как она будет менее часто использоваться пользователем. Далее, поместим в центральной части приложения поле с текстом. Там будут размещаться либо распознанный текст, либо подсказки пользователю. Наконец, в верхней части экрана будет высвечиваться фото, созданное пользователем, которое больше будет иметь смысл для пользователей со слабыми нарушениями со зрением и не будет особо отвлекать внимание для людей с серьезными заболеваниями органов зрения.

Данный макет был спроектирован в приложении Figma, где я разместил вышеперечисленные элементы пользовательского интерфейса, настроил внешний вид элементов и описал получившийся дизайн в XML формате.

1. Опишем переход к диалогу камеры

Создадим обработчик нажатия кнопки сканирования этикетки. Подключим к кнопке обработчик событий, который будет инициировать переход к новому экрану с диалогом камеры при нажатии. У меня было два пути, как реализовать диалог камеры – либо спроектировать собственный интерфейс, либо воспользоваться системным интерфейсом Android. Первый подход намного сложнее в реализации, но зато он позволил бы адаптировать интерфейс под особенности приложений со специальными возможностями. Второй же подход легче и лаконичнее в реализации, плюс если пользователь уже работал с камерой, то ему будет легче разобраться в интерфейсе и совершить необходимое действе, но в таком случае невозможно переопределить элементы диалога, а это грозит тем, что важные элементы интерфейса будут малые по размеру или нечеткие для пользователя с нарушениями со зрением. Я выбрал второй вариант, но дополнительно вставил голосовые подсказки, которые помогут сориентироваться пользователю в случае, если будут затруднения.

1. Опишем класс обработки изображения

Для наиболее точного распознавания следует дополнительно обработать изображение. Для этого мы применим несколько алгоритмов – черно-белый фильтр, повышение контраста, … .

Изображения представлены в проекте как объекты класса *Bitmap*, содержащие матрицу пикселей. Фильтры, применяемые к изображению, являются небольшими матрицами, последовательно применяемые к участкам исходной матрицы изображения. Мы будем использовать черно-белый фильтр, чтобы убрать не имеющую особой важности при распознавании текста информацию о цвете пикселя, важен только оттенок черного или белого. Далее мы инвертируем изображение, меняя цвет или яркость на противоположный. Так мы сможем добиться высокого контраста изображения, более четкого выделения границ и, соответственно, качества распознавания символов. И наконец применим метод, нейтрализующий шумы с фотографии. Так мы уберем те выбросы в данных, которые могут сильно повлиять в худшую сторону при распознавании символов. Применив данные фильтры, мы отдаем обработанное изображение этикетки далее модулю распознавания текста.

1. Опишем класс распознавания текста

Для распознавания текста с этикетки мы будем использовать облачный сервис Google Firebase ML, алгоритм распознавания описан в теоретической части работы. Для обращения к сервису будем использовать Firebase API, на основе которого создадим метод передачи изображения на облачный сервер и последующее ожидание результата. После успешного подключения к сервисам Firebase наше приложение пересылает изображение на облачный сервер, где методами ИИ извлекаются объекты FirebaseTextBlock, которые содержат координаты области с текстом и сам распознанный текст. В случае ошибки обработки (нет стабильного соединения с интернетом, не возможно подключиться к сервисам Firebase) выводится оповещение на экран.

1. Оптимизация решения

К сожалению, решение с использованием объекта FirebaseVisionImage, построенного напрямую с объекта Bitmap, не дает должного качества распознавания, так как при таком подходе изображение сжимается перед отправкой, следовательно теряется качество исходного изображения. Вместо этого можно сохранять обработанное изображение в файле JPG в памяти устройства и передавать файл на сервер целиком. Такой подход заметно увеличивает качество распознавания, но вместе с этим увеличивает продолжительность передачи файла и в целом ожидания пользователя. Я выбрал решение с передачей целостного файла, так как качество распознавание текста намного важнее, чем скорость обработки. Если приложение отдаст пользователю в итоге неразборчивый, бессмысленный текст, но за короткий промежуток времени, тогда приложение не будет выполнять поставленной цели.

1. Опишем класс обработки и перевода текста

Так как точность распознавания текста не может быть 100%, в конечном результате анализа могут находиться лишние символы, слова либо с ошибками, либо не несущие никакого смысла выражения. Нужно попытаться максимально избавиться от этих элементов текста, чтобы на выходе получить максимально полезную для пользователя информацию о сканируемом приложением продукте. Для одиночных служебных символов, таких как «\», «(», можно применить регулярное выражение, используя которое обработчик будет проходить по тексту и удалять символы, удовлетворяющие заданному правилу. Для проверки орфографии наиболее подходящим будет использование сервиса \*\*\*. После отправки интересующего нас слова в API запросе, сервис предоставляет результат в виде более подходящих по написанию слов, либо ничего не возвращает если не возможно подобрать исправленный вариант.   
Наконец, для большего удобства подключим перевод распознанного текста на русский язык. На некоторых этикетках информация о продукте может быть только на иностранном языке, или в описании продукта могут встречаться термины на английском, непонятные пользователю. Для перевода используем сервис Google Cloud Translate, которому мы через API будем передавать распознанный текст с этикетки и запрос на перевод на русский язык.   
Применяя вышеперечисленные фильтры к тексту, в итоге приложение предоставляет информативный текст, который уже можно предоставить пользователю. Этот текст мы отображаем в текстовом поле на главном экране вместе с фотографией этикетки.

1. Добавим класс озвучки текста

Одного вывода текста на экран не достаточно для приложения в сегменте продуктов для слабовидящих. Необходимо также озвучить распознанный и обработанный приложением текст, чтобы информация с этикетки была понятна и удобна для восприятия пользователю с нарушениями со зрением. В реализации озвучивания самым распространённым инструментом является библиотека Text To Speech, входящая в стандартный Android API версии 4 и выше. На этапе инициализации приложения создается объект класса TextToSpeech, который в последствии будет озвучивать женским голосом на русском языке передаваемый ему текст. Как только текст будет распознан и к нему применены фильтры и перевод, он сразу же передается на озвучку. Средство TTS идеально справляется с поставленной задачей озвучивания текста, предоставляя плавное, слитное озвучивание предложений, что делает процесс использования приложения слабовидящим пользователям намного удобнее.

1. Добавим сохранение результата сканирования

Для пользователя может быть также полезно просматривать историю сканирований, чтобы вспомнить информацию о продукте или в других случаях. Хранить информацию мы будем на устройстве пользователя при помощи базы данных SQLite, API к которому включен в стандартную библиотеку Android. Создадим инстанс подключения к базе данных, если нет файла создадим. Далее после успешных попыток распознавания текста будем сохранять записи в базе данных в формате {ID, Date: Дата и время сьемки, Text: распознанный текст, Photo path: путь до фото, Price: цена товара}. Сохранённые результаты можно использовать далее в экране истории сканирований.

1. Добавим экран истории сканирований

Создадим экран просмотра истории сканирований. Перед началом выгружаем из вышеописанной базы данных последние результаты для вывода их на экран. На данном активити по аналогии с главным экраном сделаем поле с фотографией, поле с текстом с этикетки, поле с ценой (если цена была распознана). Внизу сделаем кнопки «назад» «вперед» по бокам и по центру кнопку «на главную», по нажатии которой приложение опять возвращается на главный экран. Теперь разместим информацию из базы данных о самом первом продукте на экране, либо покажем предупреждение что история пуста. Далее при нажатии кнопки «вперед» будем показывать информацию о следующем продукте, аналогично реализуем и с кнопкой «назад». Аналогично экрану с результатом сканирования этикетки, текст должен озвучиваться при появлении, и озвучка должна прекращаться при переключении на следующий продукт.

11. Добавим аутентификацию через отпечаток пальца

Важно обеспечить безопасность при хранении данных о результатах сканирования, контролировать к ним доступ. Естественно, для нашего приложения бессмысленно реализовывать регистрацию, логин с помощью почты или пароля. Ввод пароля с клавиатуры невозможен если у человека плохое зрение, то же самое может понадобиться и при авторизации через сервисы Google, Facebook, VK и другие. Поэтому решено было добавить аутентификацию через отпечаток пальца как самый доступный для слабовидящих пользователей вариант. Создадим объект аутентификации BiometricPrompt , в котором опишем переход на экран истории сканирований в случае успешного сканирования отпечатка. Для него создадим объект, содержащий информацию о диалоге – заголовок, текст, надписи для кнопок. Теперь при нажатии кнопки «Посмотреть историю сканирований» мы будем вызывать сначала этот диалог. Если аутентификация прошла успешно, пользователь переходит на экран истории сканирований.

12. Добавим распознавание ценников от продукта **(в процессе)**

Код приложения с реализацией каждого из вышеперечисленных методов можно посмотреть на репозитории ***github.com/geniusserg/BuyVision.*** Далее приложение необходимо протестировать, чтобы найти важные проблемы и проверить соответствие приложения заданным требованиям доступности.

**Тестирование приложения**

CI тестирование

Для постоянного тестирования работоспособности приложения использовалась техника Continuous Integration, реализованная на платформе GitHub Actions. Также были произведено ручное тестирование на устройстве Huawei Honor под управлением ОС Andoid.

В рамках CI я тестировал успешную сборку приложения из исходного кода, чтобы качество кода было высоким на всем протяжении разработки. В качестве автоматического тестирования, я добавил проверку доступности сервиса Google Firebase, где вызывал команду распознавания текста на изображении с небольшим словом и оценивал корректность полученного результата. Так удалось на раннем этапе выявить проблему с ключами доступа к API.

Тестирование распознавания текста

Для тестирования работоспособности приложения я использовал реальные упаковки продуктов с текстовой информацией на ней.

Изначально при выбранном решении передавать объект класса Bitmap напрямую в метод processImage класса FirebaseVisionTextRecognizer было выявлено низкое качество распознавания. Связанно это с тем, то при таком подходе изображение сжимается, и алгоритму сложнее различить текст. Было решено сохранять изображение на устройстве и передавать этот файл на обработку. Это увеличило время распознавания текста за счет долгой загрузки на сервер, чтобы пользователю было комфортнее в таком случае пользоваться приложением, было добавлено сопровождающее сообщение «Загрузка».

Также была выявлена проблема с при низком разрешении сьемки – не распознавался текст. Решением стало рекомендация использовать приложение с камерой, разрешение которой составляет не ниже 13 мегапикселей.

И наконец, приложение не смогло распознать текст нелинейной формы (например закругленный текст). Этот недостаток можно исправить в будущем заменой движка распознавания или самостоятельным выделением зон с текстом и их выпрямлением.

UI тестирование

Первая проблема, найденная при работе с приложением, возникала при большом объёме распознанного текста. Приложение озвучивало его полностью, и это могло занимать минуты. Было решено озвучивать только 150 символов текста, а если пользователь хочет прослушать информацию дальше, он может просто провести пальцем снизу вверх и приложение продолжит озвучивать текст.

Далее, при повороте экрана ломалось внешнее представление интерфейса. Было решено сделать приложение полностью горизонтальным.

Метод «слепого» тестирования

Наконец, нужно было протестировать приложения в условиях максимально приближенных к реальности. Для этого я использовал метод «слепого» тестирования, где я симулировал плохое зрение либо вообще пробовал пользоваться приложением с завязанными глазами.

Первая проблема нашлась еще на этапе запуска приложения – ярлык приложения озвучивается системой как «буи вижн». Было решено переименовать ярлык в «Помощник для покупателей».

Далее была выявлена проблема с использованием диалога камеры – после того, как пользователь сделал фото, он попадает на экран подтверждения. Кнопки отменить и подтвердить очень маленькие и находятся сверху, плюс TalkBack не озвучивает их по умолчанию. Было решено добавить голосовые подсказки, как сделать фото для сканирование. В будущем нужно создать свой диалог камеры, который бы решал найденную проблему.

Далее была найдена проблема с открытием истории сканирований, используя систему TalkBack. Невозможно было вызвать диалог аутентификации двойным нажатием по кнопке, действие удавалось сделать только одновременным нажатием двух пальцев на кнопку. Было решено объяснить пользователям в голосовой справке при первом использовании, что кнопки лучше нажимать одновременным нажатием двух пальцев на кнопку.

**Вывод**

Мной была поставлена цель разработать приложение для слабовидящих покупателей. Эта тема очень актуальна на текущий момент, так как создание или адаптация готовых приложений под нужды слабовидящих пользователей не часто применяется при разработке, а иногда. В ходе реализации заданной цели мною были выполнены изучение материалов по разработке интерфейсов для людей с ограниченными возможностями и разработке приложений для операционной системы Android. Руководствуясь полученными знаниями, я реализовал функциональную часть приложения и спроектировал дизайн. На каждом этапе создания продукта я строго следовал советам и рекомендациям по проектированию приложений для слабовидящих, чтобы в итоге получить качественный продукт. Полученное решение я протестировал, используя метод «слепого» тестирования, чтобы максимально приблизиться к ситуации реального использования продукта. Разработанное приложение имеет свои недостатки на текущем этапе и еще не может называться полноценным помощником для слабовидящих покупателей. Но функционал продукта может пополняться такими полезными возможностями, как предоставление советов по подбору продукта, сравнение цен в магазинах и так далее. Я уверен, что если разработчики мобильных приложений будут проявлять активность в сфере поддержки пользователей с нарушениями зрения, они не только не потеряют в прибыли, но приобретут станут известными как компании, заботящиеся о своих пользователях и стремящихся сделать жизнь более полноценной и удобной. «Доступность – хорошая метрика компании».

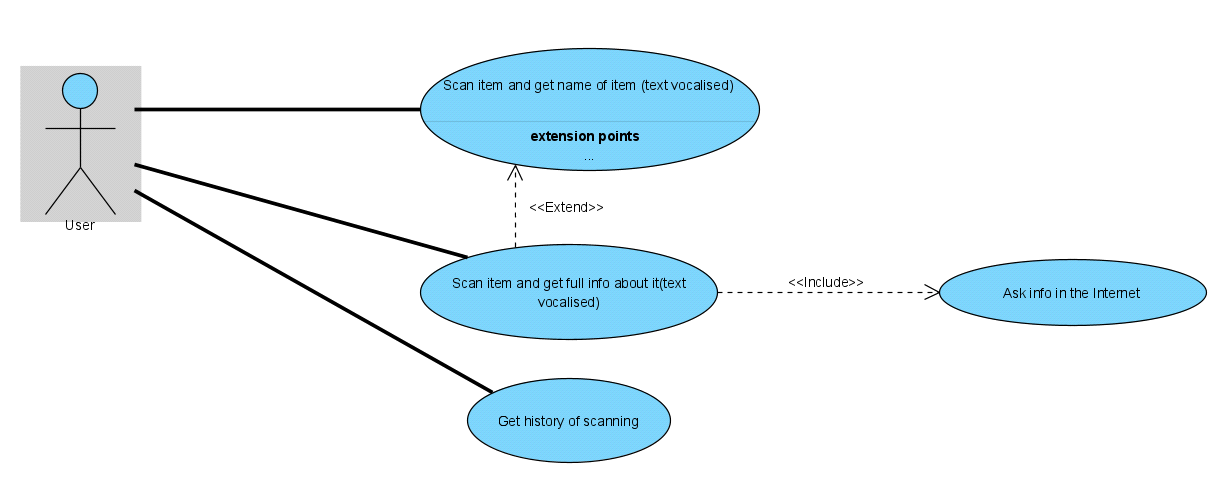
**Использованные источники**

1. Михаил Рубанов «Про доступность iOS» 2021г.
2. «PP-OCR: A Practical Ultra Lightweight OCR System» Yuning Du, Chenxia Li, Ruoyu Guo, Xiaoting Yin, Weiwei Liu,Jun Zhou, Yifan Bai, Zilin Yu, Yehua Yang, Qingqing Dang, Haoshuang Wang. Baidu Inc. 2020г.
3. «Text detection in natural images using convolutional neural networks». Marco Marten Grond 2017г.
4. Национальный стандарт РФ ГОСТ Р 52872— 2019 «Интернет-ресурсы и другая информация, представленная в электронно-цифровой форме. приложения для стационарных и мобильных устройств, иные пользовательские интерфейсы требования доступности для людей с инвалидностью и других лиц с ограничениями жизнедеятельности». Федеральное агентство по техническому регулированию и метрологии. 2019г.
5. <https://specialviewportal.ru> Портал для разработчиков и дизайнеров решений для слабовидящих людей. 2021г.
6. <https://habr.com/ru/company/russian_rehab_industry/blog/513976/> Топ самых важных функций Android для слепых 2021г.
7. <https://developer.android.com/> Руководство для разработчиков Android 2021г.

***Приложение***

**Техническое задание**

Техническое задание по созданию приложения для слабовидящих покупателей

* **Введение**
* **Название – «**Мобильное приложение для слабовидящих покупателей» (BuyVision)
* **Назначение и область применения**

Приложение позволяет сканировать этикетку\упаковку товаров и озвучивать наименование продукта, информацию с этикетки (состав, срок годности и т.д.) Мобильное приложение предназначено для людей с нарушениями зрения, которым тяжело разглядеть информацию на упаковке.

Функционал приложения включает в себя:

* Сканирование этикетки с помощью камеры
* Извлечение из изображения названия товара, описание, дополнительной информации
* Озвучивание полученной информации
* Получение дополнительной информации в сети Интернет
* Сохранение отсканированных товаров в облаке
* **Требования к приложению**
* Приложение должно быть адаптированно для людей с ограниченными возможностями: иметь минималистичный интерфейс, в дизайне должны преобладать крупные значки, шрифты и контрастные цвета, большая часть текстовой информации должна озвучиваться.
* Приложение должно делать снимки товара и передавать их на облачный сервис Firebase ML для распознавания текста или обрабатывать на устройстве с помощью библиотеки ML Kit. В случае успешного распознавания, полученный текст должен анализироваться специальным алгоритмом для выявления названия товара, описания, срока годности, состава, дополнительной информации.
* Приложение должно озвучивать наименование товара и, при желании пользователя, остальную информацию с помощью класса стандартной библиотеки Android «TextToSpeech» (возможно использование облачного сервиса Yandex SpeechKit).
* История сканирований и информации о продукте сохраняется на устройстве. Также она может быть сохранена и в базе данных Firebase Firestore, но для этого потребуется авторизация. Пользователь может создать аккаунт с помощью почты и пароля, или воспользоваться аккаунтом Google для авторизации. После успешной авторизации история сканирования синхронизируется с облаком на всех устройствах пользователя.
* Системные требования: ОС Android 6 и выше, камера 10 Мегапикселей и выше, подключение к Интернету.

**Стадии разработки**

* Разработка проектной документации
* Проектирование дизайна и интерфейса приложения
* Проектирование архитектуры приложения
* Проектирование функционала распознавания текста
* Создание алгоритма анализа полученной информации
* Проектирование системы озвучивания
* Разработка экранов приложения и интерфейса
* Добавление авторизации
* Добавление функционала работы с облачной базой данных

d. Тестирование готового продукта, испраление ошибок

e. Загрузка готового приложения на репозиторий Google Play

Начало разработки проекта – 1 января 2021 г.

Проект должен быть разработан – 1 июня 2021 г.

**Требования к дизайну**

Главный упор в дизайне должен быть на удобство использование слабовидящими пользователями. Следуя указаниям нормативов качества мобильных приложений (***источник***) – главные особенности такого приложения:

* Минималистичекий дизайн (меньше элементов управления)
* Контраст цветов ( палитра желтый\черный - #000000 #212121 #424242 #FFEA00 #c7fea0
* Крупные значки, шрифт (лучше динамический)
* Надписи к значкам – для озвучивания TalkBack
* Поддержка TalkBack, если отключен – озвучивание интерфейса
* Возможно управление с помощью кнопок громкости (возможно)
* Озвучивание текста (информация о продукте)