# ВВЕДЕНИЕ

Данная курсовая работа посвящена разработке веб-приложения для управления персональными задачами, именуемого «Список Дел» (To-Do List).

Отличительной особенностью проекта является создание интуитивно понятной и функциональной системы для организации повседневных дел. Приложение предоставляет пользователям возможности регистрации и аутентификации, управления задачами (создание, редактирование, удаление, отметка о выполнении), а также их фильтрации и сортировки, что способствует эффективному планированию и повышению личной продуктивности.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Обзор существующих конкурирующих продуктов по теме работы

Приложения для управления задачами (To-Do List) являются популярным инструментом повышения личной и командной продуктивности. Рынок предлагает множество решений, от простых однопользовательских списков до комплексных систем управления проектами. Рассмотрим несколько известных продуктов:

**Microsoft To Do** – бесплатный кроссплатформенный менеджер задач от Microsoft, пришедший на смену Wunderlist. Он предлагает создание списков дел, установку напоминаний, сроков выполнения, добавление заметок и файлов к задачам, а также ежедневный планировщик «Мой день». Интегрирован с другими продуктами Microsoft, такими как Outlook. Ориентирован как на индивидуальное использование, так и на совместную работу над списками. Основной упор сделан на простоту и интеграцию в экосистему Microsoft.

**Todoist** – мощное и гибкое приложение для управления задачами, доступное на множестве платформ. Отличается поддержкой естественного языка для быстрого добавления задач (например, «купить молоко завтра в 5 вечера»), возможностью создавать проекты, подзадачи, метки, фильтры и устанавливать приоритеты. Предлагает функции для совместной работы, отслеживания прогресса и интеграции с большим количеством сторонних сервисов. Имеет как бесплатный, так и платные тарифы с расширенным функционалом.

**Google Tasks** – простой менеджер задач, интегрированный в экосистему Google (Gmail, Google Calendar). Позволяет создавать списки задач, подзадачи, устанавливать сроки выполнения и добавлять описания. Его сила в глубокой интеграции с другими сервисами Google, что удобно для пользователей этой экосистемы. Функционал достаточно базовый по сравнению с более специализированными приложениями, но покрывает основные потребности в управлении задачами.

**Any.do** – кроссплатформенный менеджер задач, который сочетает в себе списки дел, календарь и напоминания. Особенности включают ежедневный обзор «Момент Any.do» для планирования дня, голосовой ввод, возможность прикреплять файлы и делиться списками. Предлагает как индивидуальное, так и командное использование. Имеет бесплатную версию и премиум-подписку.

**Things 3** (для экосистемы Apple) – платный менеджер задач, известный своим продуманным дизайном и вниманием к деталям. Ориентирован на методологию GTD (Getting Things Done). Предлагает проекты, области, теги, сроки и напоминания. Работает исключительно на устройствах Apple (Mac, iPhone, iPad, Apple Watch) и обеспечивает глубокую интеграцию между ними.

## 1.2 Сравнительный анализ

Проведем сравнительный анализ рассмотренных приложений для управления задачами, выделяя их ключевые особенности, преимущества и недостатки, а также сопоставляя их с целями нашего проекта «Список Дел».

**Microsoft To Do** выделяется своей бесплатностью и тесной интеграцией с экосистемой Microsoft, особенно с Outlook, что делает его удобным выбором для пользователей этих продуктов. Приложение предлагает базовый, но достаточный для многих набор функций: списки, задачи, подзадачи, сроки и напоминания, а также уникальную функцию «Мой день» для ежедневного планирования. Поддерживается совместная работа над списками. Основным преимуществом является простота и доступность, однако для управления сложными проектами его гибкости может быть недостаточно.

**Todoist** позиционируется как более мощное и гибкое решение, доступное на широком спектре платформ. К его сильным сторонам относятся поддержка распознавания естественного языка для быстрого добавления задач, возможность создания сложной иерархии (проекты, подзадачи), использование меток и фильтров для организации, а также функции для совместной работы и отслеживания продуктивности («карма Todoist»). Todoist интегрируется с большим количеством сторонних сервисов. Бесплатная версия несколько ограничена, а полный функционал доступен по подписке, что может быть минусом для некоторых пользователей. Сложность освоения несколько выше, чем у более простых аналогов.

**Google Tasks** является самым минималистичным из рассмотренных вариантов. Его главное преимущество – глубокая интеграция в сервисы Google, такие как Gmail и Google Calendar, что делает его удобным для тех, кто активно пользуется этой экосистемой. Функционал ограничен созданием списков, задач, подзадач и установкой сроков. Совместная работа возможна, но ограничена рамками Google Workspace. Это простое и бесплатное решение для базовых нужд, но не подходит для сложного управления задачами.

**Any.do** стремится найти баланс между функциональностью и простотой, объединяя списки дел, календарь и напоминания. Среди его особенностей – ежедневный обзор «Момент Any.do», помогающий спланировать день, поддержка голосового ввода и возможность делиться списками. Приложение доступно на большинстве платформ и предлагает как бесплатную, так и премиум-версию. Некоторые пользователи отмечают, что предложения перейти на премиум-версию могут быть навязчивыми.

**Things 3** представляет собой премиальное решение, ориентированное исключительно на пользователей экосистемы Apple. Оно славится своим продуманным дизайном, вниманием к деталям и ориентацией на методологию GTD (Getting Things Done). Функционал включает проекты, области, теги и удобное планирование. Однако высокая стоимость (разовая покупка для каждой платформы) и отсутствие поддержки других операционных систем делают его нишевым продуктом. Совместная работа не является его сильной стороной.

Наш проект «Список Дел», разрабатываемый в рамках данной курсовой работы, нацелен на реализацию основного и наиболее востребованного функционала простого менеджера задач. В отличие от коммерческих продуктов с широким набором функций и интеграций, наш проект изначально фокусируется на предоставлении базовых возможностей: регистрация и аутентификация пользователей, создание, просмотр, редактирование, удаление задач, отметка об их выполнении, а также базовая фильтрация. Платформой является веб-приложение. На данном этапе не предполагается сложной совместной работы или интеграций со сторонними сервисами. Главным преимуществом проекта является его учебная направленность, позволяющая освоить стек технологий (Flask, SQLAlchemy) и продемонстрировать понимание принципов разработки full-stack приложений. Сложность освоения для конечного пользователя предполагается низкой, так как интерфейс будет интуитивно понятным.

## 1.3 Вывод по итогам анализа существующих конкурентов

Анализ показывает, что на рынке существует большое количество приложений для управления задачами, каждое со своими сильными и слабыми сторонами. Многие из них предлагают широкий спект информации и ориентированы на различные сегменты пользователей – от тех, кому нужен простой список покупок, до тех, кто управляет сложными проектами в команде.

Несмотря на насыщенность рынка, разработка собственного приложения «Список Дел» в рамках курсовой работы имеет смысл:

Учебная цель: Проект позволяет освоить и применить на практике современные технологии веб-разработки (в нашем случае Flask для бэкенда, HTML/CSS/JS для фронтенда, SQLAlchemy для БД).

Фокус на основном функционале: Вместо попытки конкурировать по количеству функций, проект сосредоточен на качественной реализации базового набора возможностей, необходимых для эффективного управления личными задачами: создание и управление задачами, аутентификация пользователей.

Понимание архитектуры: Разработка с нуля дает глубокое понимание архитектуры подобных систем, взаимодействия между фронтендом и бэкендом, работы с базами данных.

Кастомизация: В будущем проект может быть доработан с учетом специфических потребностей или идей, которые могут отсутствовать в существующих решениях (хотя это выходит за рамки текущей курсовой).

Таким образом, хотя наш проект «Список Дел» не претендует на конкуренцию с гигантами рынка, он служит отличной платформой для изучения и демонстрации навыков full-stack разработки, реализуя востребованный и понятный функционал.

# 2 АНАЛИЗ ПРОГРАММНЫХ СРЕДСТВ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЯ «To-Do list»

# Для обеспечения стабильной работы, высокой производительности и удобства разработки, мы выбрали следующий стек технологий: Java с использованием Spring Framework для бэкенда, React для фронтенда и PostgreSQL в качестве системы управления базами данных.

## 2.1. Бэкенд: Python с использованием Flask Framework

Выбор Python и фреймворка Flask для разработки серверной части (бэкенда) обусловлен следующими факторами:

Простота и скорость разработки Python:

* Python известен своим чистым и читаемым синтаксисом, что значительно ускоряет процесс написания кода и снижает порог вхождения. Большое количество встроенных и сторонних библиотек позволяет быстро реализовывать необходимый функционал.
* Минималистичность и гибкость Flask: Flask является микрофреймворком, что означает его легковесность и отсутствие жестких рамок. Он предоставляет только самые необходимые инструменты для создания веб-приложений (маршрутизация, обработка запросов, шаблонизация), позволяя разработчику самостоятельно выбирать и интегрировать нужные компоненты и расширения (например, SQLAlchemy для работы с БД, Flask-Login для аутентификации, Flask-WTF для форм). Это идеально подходит для учебных проектов, где важно понимать каждый аспект работы приложения.
* Расширяемость: Несмотря на свою минималистичность, Flask легко расширяется с помощью множества доступных расширений (extensions), которые покрывают практически все типичные задачи веб-разработки. В данном проекте используются Flask-SQLAlchemy, Flask-Migrate, Flask-Login, Flask-WTF.
* Активное сообщество и документация: Python и Flask имеют огромное и активное сообщество разработчиков, что обеспечивает доступность большого количества учебных материалов, документации, примеров кода и поддержки при возникновении проблем.
* Подходит для прототипирования и небольших/средних проектов: Flask отлично подходит для быстрого создания прототипов и разработки небольших и средних веб-приложений, каким и является «Список Дел».

## 2.2 Фронтенд: HTML, CSS, JavaScript

Для разработки клиентской части (фронтенда) были выбраны стандартные веб-технологии, интегрированные с возможностями Flask:

* HTML (HyperText Markup Language): Используется для создания структуры и семантической разметки веб-страниц приложения.
* CSS (Cascading Style Sheets): Применяется для визуального оформления и стилизации HTML-элементов, обеспечивая привлекательный и удобный пользовательский интерфейс.
* JavaScript (клиентский): Используется для добавления интерактивности на стороне клиента, такой как подтверждение действий пользователя (например, удаление задачи) или, в более сложных случаях, для динамического обновления частей страницы без перезагрузки (хотя в текущей реализации проекта акцент сделан на серверный рендеринг).
* Шаблонизатор Jinja2: Flask по умолчанию интегрирован с мощным шаблонизатором Jinja2. Он позволяет создавать динамические HTML-страницы на стороне сервера, встраивая Python-подобный код (условия, циклы, наследование шаблонов, вызов переменных) непосредственно в HTML-разметку. Это значительно упрощает разработку фронтенда для приложений с серверным рендерингом, каким является наш «Список Дел». Использование блоков и наследования шаблонов (например, base.html) позволяет избежать дублирования кода и поддерживать консистентность интерфейса.

## 2.3 База данных: SQLite

В качестве системы управления базами данных (СУБД) для проекта «Список Дел» на начальном этапе выбрана SQLite.

Обоснование выбора:

* Простота настройки и использования: SQLite не требует отдельной установки и настройки серверного процесса. База данных представляет собой один файл на диске, что делает ее чрезвычайно удобной для разработки, тестирования и небольших приложений.
* Встроенная поддержка в Python: Модуль для работы с SQLite (sqlite3) входит в стандартную библиотеку Python, что устраняет необходимость установки дополнительных драйверов для базового доступа.
* Поддержка SQLAlchemy: Расширение Flask-SQLAlchemy отлично работает с SQLite, предоставляя удобный ORM-слой для взаимодействия с базой данных.
* Достаточно для учебного проекта: Для целей курсовой работы и функционала приложения «Список Дел» возможностей SQLite (транзакции, поддержка SQL) вполне достаточно.
* Легкость миграции (при необходимости): Хотя SQLite выбрана для простоты разработки, архитектура приложения с использованием SQLAlchemy позволяет относительно легко перейти на более мощные СУБД, такие как PostgreSQL или MySQL, в будущем, если потребуется масштабирование или более сложный функционал. Для этого достаточно будет изменить строку подключения SQLALCHEMY\_DATABASE\_URI и убедиться в совместимости специфичных для СУБД SQL-конструкций (хотя ORM минимизирует такие проблемы). Например, PostgreSQL, как и в примере одногруппника, является отличным выбором для более нагруженных и функционально богатых приложений благодаря своей надежности, расширяемости и поддержке сложных запросов.

# 3 ЦЕЛЬ И ЗАДАЧИ

Основной целью данной курсовой работы является проектирование и разработка прототипа веб-приложения «Список дел» (To-Do List), демонстрирующего ключевой функционал создания, организации и управления персональными задачами, а также обеспечивающего безопасное хранение данных и стабильное взаимодействие для каждого пользователя. При этом будет использован современный стек технологий, включающий Python с фреймворком Flask для бэкенда, стандартные веб-технологии (HTML, CSS, JavaScript) для фронтенда и SQLite (с возможностью перехода на PostgreSQL) в качестве системы управления базами данных, с акцентом на обоснование выбора и эффективное применение данных инструментов.

# Задачи:

# Провести анализ существующих решений.

# Определить целевую аудиторию

# Анализ требований и проектирование архитектуры.

# Проектирование и реализация базы данных. Разработать схему базы данных (например, SQLite, с учетом возможности миграции на PostgreSQL) для эффективного хранения информации о пользователях и их задачах (название, описание, статус выполнения, срок, привязка к пользователю). Реализовать механизмы взаимодействия бэкенда с базой данных, используя SQLAlchemy ORM и Flask-Migrate для управления миграциями.

# Разработка серверной части (бэкенд). Реализовать RESTful API (или набор маршрутов) с использованием Python и Flask для управления пользователями и задачами. Разработать логику CRUD-операций (Create, Read, Update, Delete) для задач. Интегрировать механизмы аутентификации и авторизации пользователей с использованием Flask-Login для обеспечения безопасности данных и персонализации контента. Реализовать валидацию данных, поступающих от клиента, с помощью Flask-WTF.

# Разработка клиентской части (фронтенд). Создать пользовательский интерфейс веб-приложения с использованием HTML, CSS и, при необходимости, JavaScript, обеспечивающий интуитивно понятное взаимодействие. Реализовать функционал отображения списка задач, форм для их создания и редактирования, а также элементов для управления задачами (отметка о выполнении, фильтрация). Обеспечить базовый адаптивный дизайн для корректного отображения приложения на различных устройствах.

# Тестирование и отладка. Провести ручное тестирование основных компонентов и функционала приложения для выявления и устранения ошибок.