**1. Оценка необходимого количества тестов**

Проект — PlantDoctor, веб-приложение для мониторинга растений с несколькими модулями:

* Авторизация (логин, регистрация, выход)
* Растения (добавление, просмотр, отметка полива)
* Статьи (просмотр, добавление)ы
* Напоминания (создание, редактирование)

Необходимые тесты:

Фронтенд (HTML/CSS/JS):

 Юнит-тесты (Jest, Mocha):

* Проверка работы функций в auth.js, plants.js, articles.js.
* Тесты на валидацию форм (логин, регистрация, добавление растения).
* Тесты на переключение страниц (showPage()).

 Интеграционные тесты (Cypress, Selenium):

* Проверка работы модальных окон.
* Тест авторизации → добавления растения → отметки полива.
* Проверка загрузки данных из API.

Итого:

* ~15–20 юнит-тестов (по 3–5 на каждый модуль).
* 5–10 интеграционных тестов (основные сценарии).

**2. Оценка программных средств с помощью метрик**

**Основные метрики:**

| Метрика | Значение (оценка) | Комментарий |
| --- | --- | --- |
| Cyclomatic Complexity | Низкая (2–3) | Простые функции (нет сложных ветвлений). |
| Coupling (связность) | Средняя | Модули (auth.js, plants.js) зависят от API, но не сильно связаны между собой. |
| LOC (Lines of Code) | ~300 строк (HTML) + ~200 (JS) | Оптимально. |
| Coverage | 0% (пока нет тестов) | Нужно добавить Jest/Cypress. |

Инструменты для анализа:

* ESLint — проверка стиля JS.
* Lighthouse — производительность, доступность.
* SonarQube — качество кода (если проект масштабируется).

**3. Инспекция кода на стандарты**

Найденные проблемы:

 HTML:

* Нет alt у изображений (нарушение accessibility).
* Нет семантических тегов (<article>, <nav>) — только <div>.
* Инлайн-стили (style="display: none;") — лучше вынести в CSS.

 CSS:

* Жёсткие значения вроде margin-top: 20px; — лучше использовать переменные.

**4. Диаграмма модулей**

Проект можно представить так:

A screenshot of a computer program

AI-generated content may be incorrect.

Рисунок 1 - Диаграмма модулей

Пояснение:

* Auth — вход/регистрация.
* API — общий слой для запросов к бэкенду.
* Plants/Articles/Reminders — модули для работы с растениями, статьями и напоминаниями.

**5. Инспекция модулей**

Инспекция модуля app.py:

1. Структура и ключевые компоненты

* Инициализация приложения:
  + Чёткое разделение конфигурации (SQLALCHEMY\_DATABASE\_URI, SECRET\_KEY).
  + Использование .env для чувствительных данных (но дефолтные значения есть).
* CORS: Настроен глобально (можно было бы ограничить домены).
* Blueprints: Подключены модули (auth, plants, admin и др.) с префиксами /api.

2. Потенциальные проблемы

* Безопасность:
  + Дефолтный SECRET\_KEY ('dev-secret-key') опасен для прода.
  + Нет HTTPS-редиректа (если планируется продакшен).
* Производительность:
  + debug=True в продакшене — риск.
  + Нет конфигурации для пула соединений БД.
* Организация:
  + Все Blueprints импортируются внутри app\_context() — лучше вынести в отдельный файл (routes.py).

3. Тесты для этого модуля

1. Позитивные:
   * Запрос / → возвращает index.html (200).
   * Проверка, что Blueprints зарегистрированы (через test\_client).
2. Негативные:
   * Отсутствие .env → падение при инициализации.
   * Неверный путь для UPLOAD\_FOLDER → ошибка создания папки.

4. Зависимости от других модулей

* extensions.py: Содержит db = SQLAlchemy() — инициализируется здесь.
* Все Blueprints: Зависят от app и db.

Инспекция модуля admin.py

1. Структура модуля

* Blueprint: admin\_bp с префиксом /api/admin.
* Эндпоинты:
  + POST /plants – добавление растения.
  + PUT/DELETE /plants/<id> – редактирование/удаление.
  + GET /users – список пользователей.
  + PUT /articles/featured – управление избранными статьями.
* Роли: Проверка прав через JWT (is\_admin()).

2. Потенциальные проблемы

2.1 Безопасность

* Уязвимости:
  + Нет валидации полей plant\_data (длина строк, HTML-теги).
  + Проверка токена без учёта срока жизни (exp).

2.2 Обработка ошибок

* Проблема: Нет try-catch для db.session.commit().

3. Тесты для модуля

| Метод | Сценарий | Ожидаемый результат |
| --- | --- | --- |
| POST /plants | Валидные данные | 201, растение в БД |
| POST /plants | Пустое название | 400 |
| PUT /plants/1 | Неверный токен | 403 |
| DELETE /plants/1 | Несуществующий ID | 404 |

4. Зависимости от других модулей

* Использует:
  + models.User, models.Plant, models.Article – для работы с БД.
  + auth.SECRET\_KEY – проверка JWT.
* Риски:
  + Прямой импорт SECRET\_KEY → циклическая зависимость.

Инспекция модуляarticles.py

1. Структура модуля

* Blueprint: articles\_bp с эндпоинтами:
  + GET /articles - список статей (с фильтром по featured)
  + GET /articles/<id> - детали статьи
  + GET /articles/plant/<id> - статьи по растению
  + POST /articles - создание статьи
* Логика:
  + Работа с моделями Article и Plant
  + JWT-аутентификация для создания статей
  + Формирование краткого описания (excerpt)

2. Потенциальные проблемы

2.1 Безопасность

* Уязвимости:
  + Прямой импорт SECRET\_KEY из auth.py
  + Нет санитизации HTML в content и title
  + Возможность XSS-атак через пользовательский контент

2.2 Валидация данных

* Проблемы:
  + Нет проверки длины title (3-100 символов)
  + Нет ограничения на размер content
  + Не проверяется существование plant\_id

2.3 Производительность

* N+1 проблема: Отдельные запросы для получения plant.name
* Нет пагинации: Может вернуть все статьи сразу

3. Тест-кейсы

| Метод | Сценарий | Ожидаемый результат |
| --- | --- | --- |
| POST /articles | Короткий заголовок (2 символа) | 400 |
| POST /articles | Несуществующий plant\_id | 404 |
| GET /articles | featured=true | Только избранные |
| GET /articles/plant/999 | Несуществующее растение | 404 |

4. Зависимости

* Использует:
  + models.Article, models.Plant
  + extensions.db
  + auth.SECRET\_KEY (лучше заменить)
* Влияет на:
  + Модуль растений (связь статей)
  + Главную страницу (фичированные статьи

Инспекция модуля auth.py

1. Структура модуля

* Blueprint: auth\_bp с эндпоинтами:
  + POST /register – регистрация
  + POST /login – аутентификация
  + GET/PUT /profile – управление профилем
  + POST /logout – выход (клиентская реализация)
* Дополнительно:
  + Декоратор @token\_required для проверки JWT
  + Интеграция с User моделью

2. Потенциальные проблемы

2.1 Безопасность

* Уязвимости:
  + Дефолтный SECRET\_KEY ('plantdoctor-secret-key') в коде
  + Нет ограничения попыток входа (риск брутфорса)
  + Пароли хранятся с werkzeug.security, но нет проверки сложности

2.2 Валидация данных

* Проблемы:
  + Нет проверки формата email
  + Нет ограничения длины username

3. Тесты для модуля

| Метод | Сценарий | Ожидаемый результат |
| --- | --- | --- |
| POST /register | Корректные данные | 201, токен в ответе |
| POST /register | Существующий email | 400 |
| POST /login | Неверный пароль | 401 |
| PUT /profile | Без токена | 401 |

4. Зависимости от других модулей

* Использует:
  + models.User – работа с пользователями
  + extensions.db – подключение к БД
* Риски:
  + Прямое использование SECRET\_KEY (лучше через current\_app.config)

Инспекция модуля models.py

1. Структура модуля

* Основные модели:
  + User - пользователи системы
  + Plant - справочник растений
  + UserPlant - растения пользователей
  + Article - статьи
  + Reminder - напоминания
* Связи:
  + Один-ко-многим (User ↔ Articles, User ↔ Reminders)
  + Многие-ко-многим (User ↔ Plant через UserPlant)
* Дополнительно:
  + Индексы для ускорения запросов
  + Хеширование паролей

2. Потенциальные проблемы

2.1 Безопасность

* Уязвимости:
  + Нет валидации длины полей при вводе (username, email и др.)
  + Отсутствует ограничение на частые запросы к БД

2.2 Производительность

* Проблемы:
  + Нет индекса для часто запрашиваемых полей (например, User.username)
  + Каскадное удаление (cascade='all, delete-orphan') может быть медленным

3. Тесты для модуля

| Тест | Сценарий | Ожидаемый результат |
| --- | --- | --- |
| Создание User | Валидные данные | Успешное сохранение |
| Создание User | Дубликат email | Ошибка integrity |
| Связь User-Article | Создание статьи | Автоматическая привязка автора |
| Каскадное удаление | Удаление User | Удаление связанных Articles |

4. Зависимости от других модулей

* Использует:
  + extensions.db - подключение к БД
  + werkzeug.security - хеширование паролей
* Влияет на:
  + Все модули, работающие с БД (auth.py, articles.py и др.)

Инспекция модуля plants.py

1. Структура модуля

* Blueprint: plants\_bp с эндпоинтами:
  + GET /plants - список всех растений
  + GET /plants/<id> - детали растения
  + GET /user/plants - растения пользователя
  + POST /user/plants - добавить растение
  + POST /user/plants/<id>/water - отметить полив
  + DELETE /user/plants/<id> - удалить растение
* Логика:
  + Работа с моделями Plant и UserPlant
  + JWT-аутентификация
  + Обновление дат полива

2. Потенциальные проблемы

2.1 Безопасность

* Уязвимости:
  + Прямой импорт SECRET\_KEY из auth.py
  + Нет ограничения частоты запросов на полив
  + Возможна XSS-атака через notes в UserPlant

2.2 Валидация данных

* Проблемы:
  + Нет проверки формата image\_url
  + Нет ограничений на длину notes
  + Не проверяется существование растения перед добавлением

2.3 Производительность

* N+1 проблема:
  + В get\_user\_plants отдельные запросы для каждого Plant
  + Нет пагинации для /plants

3. Тест-кейсы

| Метод | Сценарий | Ожидаемый результат |
| --- | --- | --- |
| POST /user/plants | Несуществующий plant\_id | 404 |
| POST /user/plants | Повторное добавление | 400 |
| POST /water | Неправильный user\_plant\_id | 404 |
| GET /plants | Большое кол-во растений | Пагинация |

4. Зависимости

* Использует:
  + models.Plant, models.UserPlant
  + extensions.db
  + auth.SECRET\_KEY (лучше заменить на current\_app.config)
* Влияет на:
  + Фронтенд (отображение растений)
  + Модуль напоминаний

Инспекция модуля reminders.py

1. Структура модуля

* Blueprint: reminders\_bp с префиксом /api/reminders
* Эндпоинты:
  + GET / - список напоминаний пользователя
  + POST / - создание напоминания
  + PUT /<id> - обновление напоминания
  + DELETE /<id> - удаление напоминания
* Логика:
  + Привязка к растениям пользователя (опционально)
  + Валидация JWT токена
  + Работа с датами и статусами выполнения

2. Потенциальные проблемы

2.1 Безопасность

* Уязвимости:
  + Прямой импорт SECRET\_KEY из auth.py (риск циклических импортов)
  + Нет санитизации HTML в поле notes
  + Возможность подмены user\_plant\_id при создании напоминания

2.2 Валидация данных

* Проблемы:
  + Нет проверки корректности типа напоминания
  + Не проверяется, что due\_date в будущем
  + Нет ограничения длины для notes

2.3 Производительность

* N+1 проблема: Отдельные запросы для каждого UserPlant
* Нет пагинации: Может вернуть все напоминания сразу

2.4 Документация

* Отсутствуют docstrings для функций
* Нет примеров запросов/ответов

3. Тест-кейсы

| Метод | Сценарий | Ожидаемый результат |
| --- | --- | --- |
| POST / | Дата в прошлом | 400 |
| POST / | Неверный тип | 400 |
| PUT /1 | Чужое напоминание | 404 |
| GET / | Большое кол-во | Проверить производительность |

4. Зависимости

* Использует:
  + models.Reminder, models.UserPlant
  + extensions.db
  + auth.SECRET\_KEY (лучше заменить)
* Влияет на:
  + Модуль уведомлений
  + Календарь событий