Исследовательский план: Теоретическая основа для тестирования Tornado Todd

Цель исследования: Получить необходимые теоретические знания для грамотного планирования, проведения и анализа тестирования веб-приложения на Django.

Блок 1: Методы и виды тестирования (45 минут)

Что изучить: Какие бывают проверки и для чего они нужны.
Конкретно для Tornado Todd сосредоточьтесь на:

1. **Функциональное тестирование:**
    - *Суть:* Проверка, что функции работают согласно требованиям ("При нажатии кнопки 'Добавить' задача появляется в списке").
    - **Ключевые понятия:** Тест-кейс, сценарий (use case), позитивное/негативное тестирование.
    - *Что искать:* Как составлять тест-кейсы для форм ввода, кнопок, ссылок.
2. **Тестирование пользовательского интерфейса (UI):**
    - *Суть:* Проверка, что всё отображается корректно и элементы на месте.
    - **Ключевые понятия:** Юзабилити (usability), кроссплатформенность, вёрстка.
    - *Что искать:* Быстрая проверка: отображается ли страница правильно в разных браузерах (Chrome, Firefox)?
3. **Тестирование безопасности (Security Testing):**
    - *Суть:* Проверка на уязвимости. **Критически важно для вашего приложения с аутентификацией!**
    - **Ключевые понятия:** Межсайтовая подделка запроса (CSRF), SQL-инъекция, аутентификация и авторизация.
    - *Что искать:* Как Django защищает от CSRF по умолчанию? Как проверить, что один пользователь не может получить доступ к задачам другого?
4. **Дымовое (Smoke) и регрессионное тестирование:**
    - *Суть:* Быстрая проверка основ после внесения изменений (дымовое) и проверка, что новое не сломало старое (регрессионное).
    - *Что искать:* Как составить минимальный набор тестов для быстрой проверки работоспособности?

Где искать: Статьи на Хабре по запросам "виды тестирования", "функциональное тестирование для начинающих".

Блок 2: Инструменты тестирования (30 минут)

Что изучить: Какие программы и библиотеки помогут нам провести тесты.
Для вашего проекта актуальны:

1. **Инструменты для ручного тестирования:**
    - **Браузер:** Встроенные **Developer Tools** (F12) — для просмотра ошибок в консоли, анализа сетевых запросов.
    - **Менеджер тест-кейсов:** Обычно используют **Excel/Google Таблицы** или **TestRail**. Вам хватит таблицы.
2. **Инструменты для автоматического тестирования (познакомиться поверхностно):**
    - **Для Backend (Django):** Встроенный **Django Test Framework** (на основе `unittest`). Позволяет писать тесты на Python.
    - **Для Frontend (UI): Selenium** — для автоматизации действий в браузере. Пока просто знать о его существовании.

Где искать: "Django testing tutorial", "как тестировать Django приложения".

Блок 3: Стандарты и рекомендации (30 минут)

Что изучить: Какие существуют общепринятые правила и лучшие практики в индустрии.

1. **Жизненный цикл тестирования:**
    - *Суть:* Понимание, что тестирование — это не разовое действие, а процесс.
    - **Ключевые этапы:** Планирование -> Проектирование тестов -> Выполнение -> Анализ результатов -> Заведение баг-репортов.
    - *Что искать:* Схемы и статьи про "Software Testing Life Cycle (STLC)".
2. **Классификация дефектов (багов):**
    - *Суть:* Как правильно описывать найденные ошибки.
    - **Ключевые понятия:** Серьёзность (Severity: S1-критический, S2-высокая...) и Приоритет (Priority: P1-высокий...). Вам хватит простого разделения на "Критично", "Важно", "Незначительно".
    - *Что искать:* "Как правильно оформить баг-репорт".
3. **Основы стандартов:**
    - *Суть:* Достаточно знать, что основные принципы тестирования описаны в стандарте **ISO/IEC/IEEE 29119**. Не читать его, а понять, что он существует и является основным.

Где искать: "Жизненный цикл тестирования", "как классифицировать баги", "стандарт ISO 29119".

Блок 4: Примеры из реальной жизни (15 минут)

Что изучить: Увидеть последствия плохого тестирования.

1. **Пример неудачного теста:**
    - *Что искать:* Найдите новость о сбое в каком-либо онлайн-сервисе (например, "падение Сбербанк Онлайн", "сбой в мобильном приложении Такси").
    - *Анализ:* Подумайте, какой вид тестирования (из Блока 1) мог бы предотвратить эту ситуацию? (Например, сбой такси — явный провал нагрузочного тестирования).
2. **Пример успешного тестирования:**
    - *Что искать:* Любой крупный сервис, котор**Исследовательский план: Теоретическая основа для тестирования Tornado Todd**

Цель исследования: Получить необходимые теоретические знания для грамотного планирования, проведения и анализа тестирования веб-приложения на Django.

Блок 1: Методы и виды тестирования (45 минут)

Что изучить: Какие бывают проверки и для чего они нужны.
Конкретно для Tornado Todd сосредоточьтесь на:

1. **Функциональное тестирование:**
    - *Суть:* Проверка, что функции работают согласно требованиям ("При нажатии кнопки 'Добавить' задача появляется в списке").
    - **Ключевые понятия:** Тест-кейс, сценарий (use case), позитивное/негативное тестирование.
    - *Что искать:* Как составлять тест-кейсы для форм ввода, кнопок, ссылок.
2. **Тестирование пользовательского интерфейса (UI):**
    - *Суть:* Проверка, что всё отображается корректно и элементы на месте.
    - **Ключевые понятия:** Юзабилити (usability), кроссплатформенность, вёрстка.
    - *Что искать:* Быстрая проверка: отображается ли страница правильно в разных браузерах (Chrome, Firefox)?
3. **Тестирование безопасности (Security Testing):**
    - *Суть:* Проверка на уязвимости. **Критически важно для вашего приложения с аутентификацией!**
    - **Ключевые понятия:** Межсайтовая подделка запроса (CSRF), SQL-инъекция, аутентификация и авторизация.
    - *Что искать:* Как Django защищает от CSRF по умолчанию? Как проверить, что один пользователь не может получить доступ к задачам другого?

4. **Дымовое (Smoke) и регрессионное тестирование:**
    - *Суть:* Быстрая проверка основ после внесения изменений (дымовое) и проверка, что новое не сломало старое (регрессионное).
    - *Что искать:* Как составить минимальный набор тестов для быстрой проверки работоспособности?

Где искать: Статьи на Хабре по запросам "виды тестирования", "функциональное тестирование для начинающих".

Блок 2: Инструменты тестирования (30 минут)

Что изучить: Какие программы и библиотеки помогут нам провести тесты.
Для вашего проекта актуальны:

1. **Инструменты для ручного тестирования:**
    - **Браузер:** Встроенные **Developer Tools** (F12) — для просмотра ошибок в консоли, анализа сетевых запросов.
    - **Менеджер тест-кейсов:** Обычно используют **Excel/Google Таблицы** или **TestRail**. Вам хватит таблицы.
2. **Инструменты для автоматического тестирования (познакомиться поверхностно):**
    - **Для Backend (Django):** Встроенный **Django Test Framework** (на основе `unittest`). Позволяет писать тесты на Python.
    - **Для Frontend (UI): Selenium** — для автоматизации действий в браузере. Пока просто знать о его существовании.

Где искать: "Django testing tutorial", "как тестировать Django приложения".

Блок 3: Стандарты и рекомендации (30 минут)

Что изучить: Какие существуют общепринятые правила и лучшие практики в индустрии.

1. **Жизненный цикл тестирования:**
    - *Суть:* Понимание, что тестирование — это не разовое действие, а процесс.
    - **Ключевые этапы:** Планирование -> Проектирование тестов -> Выполнение -> Анализ результатов -> Заведение баг-репортов.
    - *Что искать:* Схемы и статьи про "Software Testing Life Cycle (STLC)".
2. **Классификация дефектов (багов):**
    - *Суть:* Как правильно описывать найденные ошибки.
    - **Ключевые понятия:** Серьёзность (Severity: S1-критический, S2-высокая...) и Приоритет (Priority: P1-высокий...). Вам хватит простого разделения на "Критично", "Важно", "Незначительно".

- *Что искать:* "Как правильно оформить баг-репорт".

3. **Основы стандартов:**
    - *Суть:* Достаточно знать, что основные принципы тестирования описаны в стандарте **ISO/IEC/IEEE 29119**. Не читать его, а понять, что он существует и является основным.

Где искать: "Жизненный цикл тестирования", "как классифицировать баги", "стандарт ISO 29119".

Блок 4: Примеры из реальной жизни (15 минут)

Что изучить: Увидеть последствия плохого тестирования.

1. **Пример неудачного теста:**
    - *Что искать:* Найдите новость о сбое в каком-либо онлайн-сервисе (например, "падение Сбербанк Онлайн", "сбой в мобильном приложении Такси").
    - *Анализ:* Подумайте, какой вид тестирования (из Блока 1) мог бы предотвратить эту ситуацию? (Например, сбой такси — явный провал нагрузочного тестирования).
2. **Пример успешного тестирования:**
    - *Что искать:* Любой крупный сервис, который работает стабильно (YouTube, Google Поиск). За ними стоят тысячи автоматических тестов, которые запускаются при каждом обновлении.

Где искать: Новостные сайты, tech-блоги.

Итоговый результат вашего исследования:

После изучения теории вы должны будете able to (быть в состоянии):

1. **Объяснить,** какие виды тестирования вы планируете применить к своему приложению и почему.
2. **Составить** чек-лист (список пунктов) для ручного тестирования Tornado Todd.
3. **Знать,** на какие уязвимости безопасности нужно обратить внимание в веб-приложении.
4. **Понимать,** как правильно оформить найденный баг.

Этот план сфокусирует ваше изучение теории на практике вашего же проекта, что сделает его максимально полезным и эффективным. Удачи в погружении!

- ый работает стабильно (YouTube, Google Поиск). За ними стоят тысячи автоматических тестов, которые запускаются при каждом обновлении.

Где искать: Новостные сайты, tech-блоги.

Итоговый результат вашего исследования:

После изучения теории вы должны будете able to (быть в состоянии):

1. **Объяснить,** какие виды тестирования вы планируете применить к своему приложению и почему.
2. **Составить** чек-лист (список пунктов) для ручного тестирования Tornado Todd.
3. **Знать,** на какие уязвимости безопасности нужно обратить внимание в веб-приложении.
4. **Понимать,** как правильно оформить найденный баг.

Этот план сфокусирует ваше изучение теории на практике вашего же проекта, что сделает его максимально полезным и эффективным. Удачи в погружении!