МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

План тестирования

по дисциплине «Системная и программная инженерия»

Выполнили:

Студенты группы ИКБО-02-22 Боков А. Д.

Краскова Е.А.

Мичурин Д. В.

Кононенко С.С.

Чиков М.Е.

Проверил:

Ассистент кафедры МОСИТ Золотухина М. А.

МОСКВА 2025 г.

Содержание

Часть 1. План тестирования

1.1 Идентификатор тестового плана

ТР-СК-01 — Тестовый план веб-приложения "Кулинарный помощник", версия 1.0

1.2 Ссылки на используемые документы

- ГОСТ 19.301–79 — Единая система программной документации. Общие положения
- ГОСТ 19.101–77 — Виды программ и программных документов
- ГОСТ 34.602–89 — Техническое задание на создание автоматизированной системы
- ГОСТ 34.603–92 — Автоматизированные системы. Требования к содержанию документов
- ГОСТ 28195–89 — Тестирование программных средств. Общие положения
- ГОСТ Р ИСО/МЭК 25010–2014 — Системы и программное обеспечение. Характеристики качества
- ГОСТ Р ИСО/МЭК 29119 (1–5) — Стандарты по тестированию ПО

1.3 Введение

Настоящий документ представляет собой план тестирования веб-приложения «Cook Cake» — интерактивной платформы для поиска, хранения и обмена кулинарными рецептами. Основная цель тестирования — обеспечение качества, стабильности и надёжности функционирования системы при различных сценариях использования.

Проект реализован с использованием современных технологий:

- Java Spring MVC — серверная логика;
- JavaScript — клиентская часть;
- PostgreSQL — система управления базами данных;

- Docker — контейнеризация и упрощённое развертывание компонентов.

План тестирования охватывает ключевые аспекты функционирования системы, включая:

- Проверку корректности регистрации и авторизации пользователей;
- Верификацию отображения и фильтрации рецептов;
- Тестирование функционала публикации, редактирования и удаления рецептов;
- Проверку возможности комментирования, проставления лайков и взаимодействия с контентом;
- Оценку производительности, безопасности и устойчивости приложения.

Тестирование будет проводиться вручную и с применением автоматизированных средств, с целью выявления возможных дефектов и проверки соответствия системы функциональным и нефункциональным требованиям. Все выявленные ошибки будут документироваться, анализироваться и направляться на устранение в рамках итерационного процесса разработки.

1.4 Тестируемые элементы

В рамках проекта «Cook Cake» предполагается тестирование следующих компонентов и модулей системы:

1. Регистрация и авторизация пользователей
   - Проверка корректной регистрации нового пользователя
   - Проверка входа с верными и неверными данными
   - Поведение системы при неправильном вводе
2. Работа с рецептами
   - Добавление нового рецепта (с изображением и описанием)
   - Отображение списка рецептов
   - Просмотр полного описания рецепта

3. Поиск рецептов
   - Поиск по названию
   - Поиск по ингредиентам
4. Взаимодействие пользователей
   - Добавление комментариев
   - Установка лайков
5. Основные элементы интерфейса
   - Проверка корректного отображения страниц на разных экранах
   - Наличие навигации между основными разделами

1.5 Проблемы риска тестирования ПП

В процессе тестирования веб-приложения «Cook Cake» возможны следующие проблемы и риски:

- Неполнота требований. Возможны изменения требований в процессе разработки, что повлияет на полноту и актуальность тестов.
- Ограниченные ресурсы. Недостаток времени, оборудования или квалифицированных специалистов может снизить качество тестирования.
- Нестабильность версий. Частые изменения кода могут привести к ошибкам регрессии и усложнённой отладке.
- Сложности с воспроизведением ошибок. Некоторые дефекты могут проявляться только при определённых условиях (например, при высоких нагрузках).
- Ограничения тестовой среды. Отличие тестовой среды от реальной может привести к неполному покрытию сценариев.
- Риски безопасности. Возможность уязвимостей при недостаточной проверке механизма авторизации, управления сессиями и обработки пользовательских данных.

1.6 Особенности или свойства, подлежащие тестированию

Следующие характеристики подлежат обязательному тестированию:

- Функциональность: реализация всех пользовательских сценариев — регистрация, авторизация, работа с рецептами, поиск, взаимодействие.
- Удобство использования (usability): корректность отображения элементов интерфейса на различных устройствах и разрешениях экрана.
- Производительность: время отклика сервера, скорость загрузки страниц, устойчивость при увеличении числа пользователей.
- Совместимость: стабильная работа в различных браузерах (Chrome, Firefox, Safari), мобильных устройствах.
- Надёжность: способность системы к восстановлению после сбоев.

1.7 Особенности (свойства), не подлежащие тестированию

В рамках данного тестового плана не подлежат тестированию следующие аспекты:

- Процесс разработки и архитектура кода. Не проводится ревизия исходного кода, фокус — на внешнем поведении приложения.
- Маркетинговые характеристики. Удовлетворённость рынка, оценка конкурентных преимуществ не рассматриваются.
- Дизайн и креативные элементы. Эстетика интерфейса (цвета, стиль) не оцениваются, если не влияют на функциональность.
- Интеграция с внешними API, если таковые используются, будет покрыта отдельным тестированием при наличии документации.

1.8 Подход

Тестирование приложения будет проводиться с использованием комбинированного подхода, включающего:

- Функциональное тестирование по тест-кейсам на основе спецификации требований;
- Исследовательское тестирование для выявления нестандартных ошибок;
- Ручное тестирование основных пользовательских сценариев;

- Автоматизированное тестирование (на основе Selenium) для проверки регрессионных тестов и повторяющихся операций;
- Тестирование производительности с помощью инструментов вроде JMeter;
- Тестирование совместимости на популярных браузерах и мобильных устройствах.

1.9 Критерии смоук-тестирования

Смоук-тестирование проводится после каждого развертывания новой сборки. Критериями успешного прохождения смоук-тестов являются:

- Успешный запуск приложения на тестовом стенде без ошибок;
- Работоспособность основных функций:
  - регистрация и авторизация;
  - загрузка главной страницы;
  - отображение списка рецептов;
  - добавление нового рецепта;
- Отсутствие критических ошибок в логах;
- Возможность навигации между основными разделами сайта.

1.10 Критерии прохождения тестов

Тест считается успешно пройденным при соблюдении следующих условий:

- Все обязательные тест-кейсы (приоритет "высокий" и "средний") выполнены и завершены со статусом **Passed**;
- Все критические и блокирующие дефекты устранены и перепроверены;
- Доля успешно пройденных тестов составляет не менее **95%** от общего количества;
- Остаточные ошибки не влияют на ключевой функционал и задокументированы как известные дефекты.

1.11 Критерии приостановки и возобновления работ

Работы по тестированию приостанавливаются в следующих случаях:

- Обнаружение критического дефекта, блокирующего тестирование;
- Недоступность тестовой среды или системы сборки;
- Изменения в требованиях, влияющие на актуальность текущих тестов.

Возобновление тестирования осуществляется:

- После устранения блокирующих дефектов и успешного прохождения смоук-теста;
- После обновления тестовой документации в соответствии с новыми требованиями;
- После восстановления работоспособности инфраструктуры.

1.12 Тестовая документация

В процессе тестирования создаются и используются следующие документы:

- План тестирования (настоящий документ);
- Тест-кейсы с описанием шагов, входных данных и ожидаемых результатов;
- Отчёты о тестировании с указанием результатов и дефектов;
- Баг-репорты — регистрация и описание найденных ошибок;
- Чек-листы для смоук- и регрессионного тестирования;

1.13. Основные задачи тестирования

Задачи тестирования включают:

- Проверку соответствия реализованного функционала заявленным требованиям;
- Выявление и документирование дефектов;
- Оценку надёжности, производительности и безопасности приложения;

- Обеспечение стабильной работы основных пользовательских сценариев;
- Повышение уверенности в готовности продукта к релизу.

1.14. Необходимый персонал и обучение

Для выполнения тестирования потребуется следующая команда:

- Тестировщик (QA-инженер) — подготовка и выполнение тестов, регистрация багов;
- Тимлид — организация процесса тестирования, контроль выполнения;
- Разработчики — исправление дефектов и участие в их анализе;

Обучение включает:

- Ознакомление с требованиями проекта и технической документацией;
- Инструктаж по работе с системой баг-трекинга (например, Jira или YouTrack);
- Ознакомление с архитектурой приложения и средой развертывания.

1.15. Требования к тестовой среде

Для проведения тестирования необходимо:

- Сервер с развернутыми компонентами приложения
- Интернет-браузеры: Google Chrome, Firefox, Safari;
- Мобильные устройства: Android, iOS
- Инструменты тестирования: Postman, Selenium, JMeter;
- Система контроля версий и CI/CD: Git, Jenkins или аналог;
- Баг-трекинговая система: Jira / Trello / YouTrack.

1.16. Распределение ответственности

- Руководитель проекта: согласование сроков и приоритетов, общая координация работ.
- Тестировщик: создание тест-кейсов, выполнение ручного и автоматизированного тестирования, отчётность.

- Разработчики: анализ и исправление найденных дефектов, предоставление сборок для тестирования.
- Аналитик: консультации по требованиям, помощь в интерпретации спорных случаев.

1.17. График работ (календарный план)

Таблица 1 – График работ

Тап работы	Начало	Окончание	Ответственные лица
1. Анализ требований и постановка задач	01.03.2025	14.03.2025	Аналитик
2. Подготовка тестового плана и документации	14.03.2025	28.03.2025	Тестировщик, Тех. писатель
3. Разработка тест-кейсов	28.03.2025	10.04.20255	Тестировщик
4. Подготовка тестовой среды	10.04.2025	15.04.2025	Разработчик, тестировщик
5. Проведение смоук-тестирования	15.04.2025	20.04.2025	Тестировщик
6. Проведение функционального тестирования	20.04.2024	25.05.2025	Тестировщик, разработчик
7. Проведение нефункционального тестирования	25.05.2025	30.05.2025	Тестировщик, аналитик
8. Сбор и анализ результатов, баг-репорты	30.05.2025	06.05.2025	Тестировщик, тех. писатель
9. Регрессия после исправлений	06.05.2025	15.05.2025	разработчик
10. Финальное тестирование и приёмка	15.05.2025	21.05.2025	Тестировщик, руководитель проекта
11. Подготовка итогового отчёта	21.0.2025	30.05.2025	Тестировщик, тех. писатель

1.18. Риски и непредвиденные обстоятельства

В ходе тестирования веб-приложения «Cook Cake» возможны риски, способные негативно повлиять на процесс, сроки и результат тестирования.

Для каждого риска предусмотрены меры по предотвращению или минимизации последствий.

Таблица 2 – Перечень рисков и мер предосторожности

Риск	**Вероятность**	**Влияние**	**Меры по снижению риска**
Изменения в требованиях в процессе тестирования	Средняя	Высокое	Постоянная связь с заказчиком и аналитиком
Недоступность тестовой среды	Средняя	Среднее	Наличие резервной среды
Задержки со стороны команды разработки	Средняя	Высокое	Совместное планирование спринтов
Критические баги, блокирующие тестирование	Высокая	Высокая	Проведение смоук-тестов перед началом спринта
Отказ оборудования или системных компонентов	Низкая	Среднее	Использование облачных решений и резервное копирование
Невоспроизводимые ошибки (плавающие баги)	Средняя	Среднее	Подробное логирование, анализ с разработчиками

1.19. Утверждение плана тестирования

План тестирования считается утверждённым после прохождения следующих этапов:

- Проведение внутреннего согласования с командой и техническим руководителем;
- При необходимости – согласование с заказчиком или представителем бизнеса;
- Утверждение фиксируется датой и подписями ответственных лиц.

1.20. Глоссарий

Тест-кейс	—	формализованный сценарий тестирования, содержащий входные данные, шаги выполнения и ожидаемый результат.
Баг / Дефект	—	ошибка в программном обеспечении, приводящая к отклонению от ожидаемого поведения.
Регрессионное тестирование	—	повторное тестирование существующего функционала с целью проверки, что изменения не вызвали новых ошибок.
Смоук-тестирование	—	поверхностная проверка критически важного функционала, позволяющая определить, пригодна ли сборка для дальнейшего тестирования.
Тестовая среда	—	специальная конфигурация аппаратного и программного обеспечения, предназначенная для проведения тестирования.
Юзабилити	—	удобство использования интерфейса, простота навигации и интуитивность взаимодействия пользователя с системой.
Баг-репорт	—	документ, описывающий обнаруженный дефект, условия его воспроизведения и технические детали.

Часть 2. Автоматизация тестирования

2.1 Продукт для тестирования

Для выполнения данного шага был выбран программный продукт Selenium, поскольку он является одним из самых популярных и проверенных инструментов для автоматизации веб-приложений. Selenium поддерживает множество языков программирования, таких как Python, Java и C#, и обеспечивает широкую совместимость с различными браузерами, что делает его универсальным решением для тестирования и автоматизации. Кроме того, Selenium предоставляет гибкий функционал для взаимодействия с элементами страниц, эмуляции действий пользователя и выполнения сложных сценариев, что делает его оптимальным выбором для эффективной разработки скриптов.

2.2 Скрипты для тестирования

В листингах ниже приведены скрипты, разработанные для тестирования приложения.

Листинг 1 – Тест-скрипт 1

```
class TestTest1():


def setup_method(self, method):
  self.driver = webdriver.Chrome()
  self.vars = {}

def teardown_method(self, method):
  self.driver.quit()

  def test_test1(self):
    self.driver.get("http://localhost:8089/")
    self.driver.find_element(By.LINK_TEXT, "Сделать пост").click()
    self.driver.find_element(By.LINK_TEXT, "Вернуться на главную").click()
```

Данный скрипт проверяет возможность перехода на страницу создания поста и обратного возврата на главную страницу через соответствующие ссылки.

Листинг 2 – Тест-скрипт 2

```
class TestTest2():


def setup_method(self, method):
  self.driver = webdriver.Chrome()
  self.vars = {}

def teardown_method(self, method):
  self.driver.quit()

  def test_test2(self):
    self.driver.get("http://localhost:8089/")
    self.driver.find_element(By.ID, "view-recipe-btn").click()
    self.driver.get("http://localhost:8089/post/53")
    self.driver.find_element(By.CSS_SELECTOR, ".container:nth-child(1) > p").click()
    self.driver.find_element(By.CSS_SELECTOR, ".container:nth-child(2) > p").click()
    self.driver.find_element(By.CSS_SELECTOR, "h2:nth-child(2)").click()
```

Данный скрипт проверяет базовую навигацию от главной страницы до конкретного поста и взаимодействие с элементами контента внутри поста.

Листинг 3 – Тест-скрипт 3

```
class TestTest3():


def setup_method(self, method):
  self.driver = webdriver.Chrome()
```

```
  self.vars = {}

def teardown_method(self, method):
  self.driver.quit()

  def test_test3(self):
    self.driver.get("http://localhost:8089/")
    self.driver.find_element(By.LINK_TEXT, "Регистрация/Войти").click()
    self.driver.find_element(By.ID, "switch2").click()
    self.driver.find_element(By.ID, "username").click()
    self.driver.find_element(By.ID,
"username").send_keys("m.chikov04@mail.ru")
    self.driver.find_element(By.ID, "password").send_keys("qwerty")
    self.driver.find_element(By.CSS_SELECTOR, ".login > button").click()
```

Тест моделирует сценарий стандартной авторизации пользователя через ввод логина и пароля и нажатие кнопки входа.

2.3 Результаты тестирования

В данной части приведены результаты тестирования на изображениях 1-3. Как можно увидеть по результатам, все тесты были пройдены успешно.

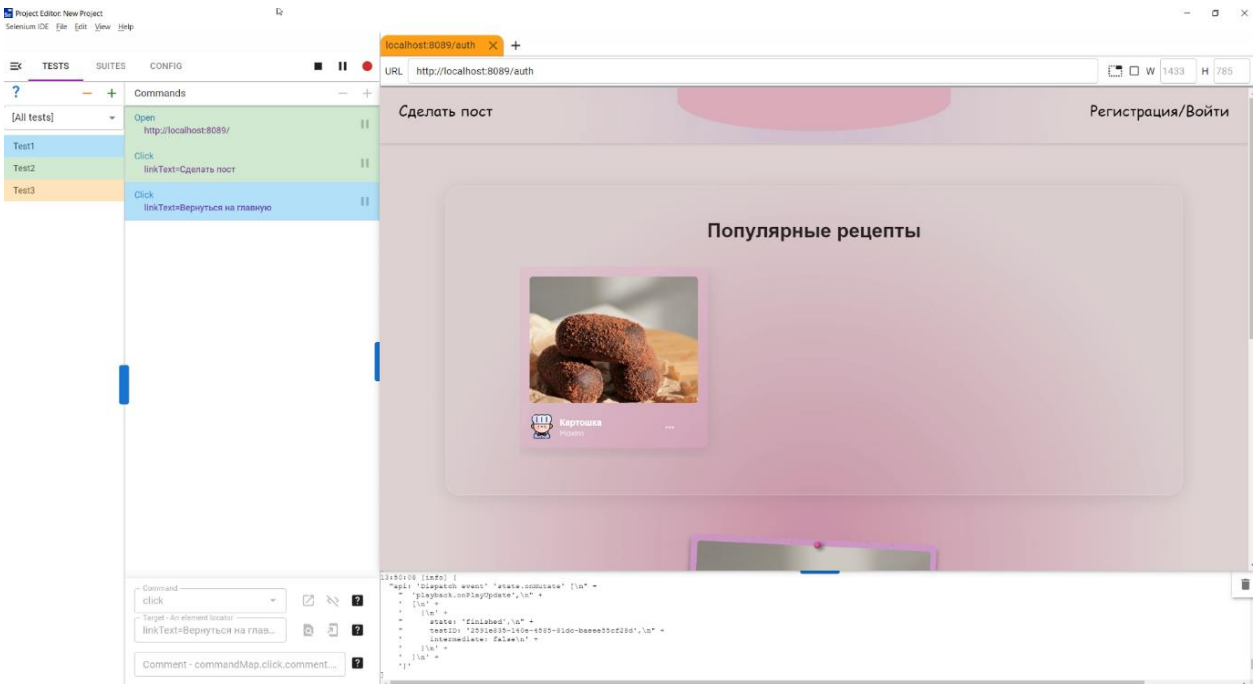

Рисунок 1 – Результат прохождения 1 теста

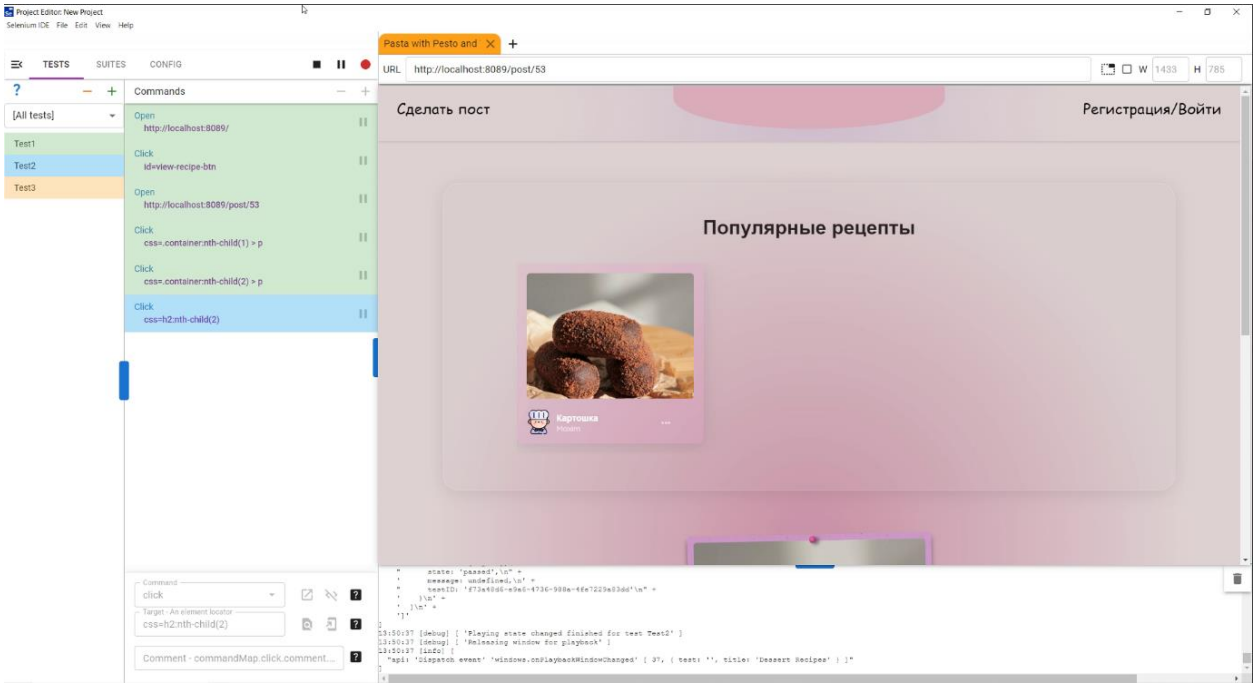

Рисунок 2 – Результат прохождения 2 теста

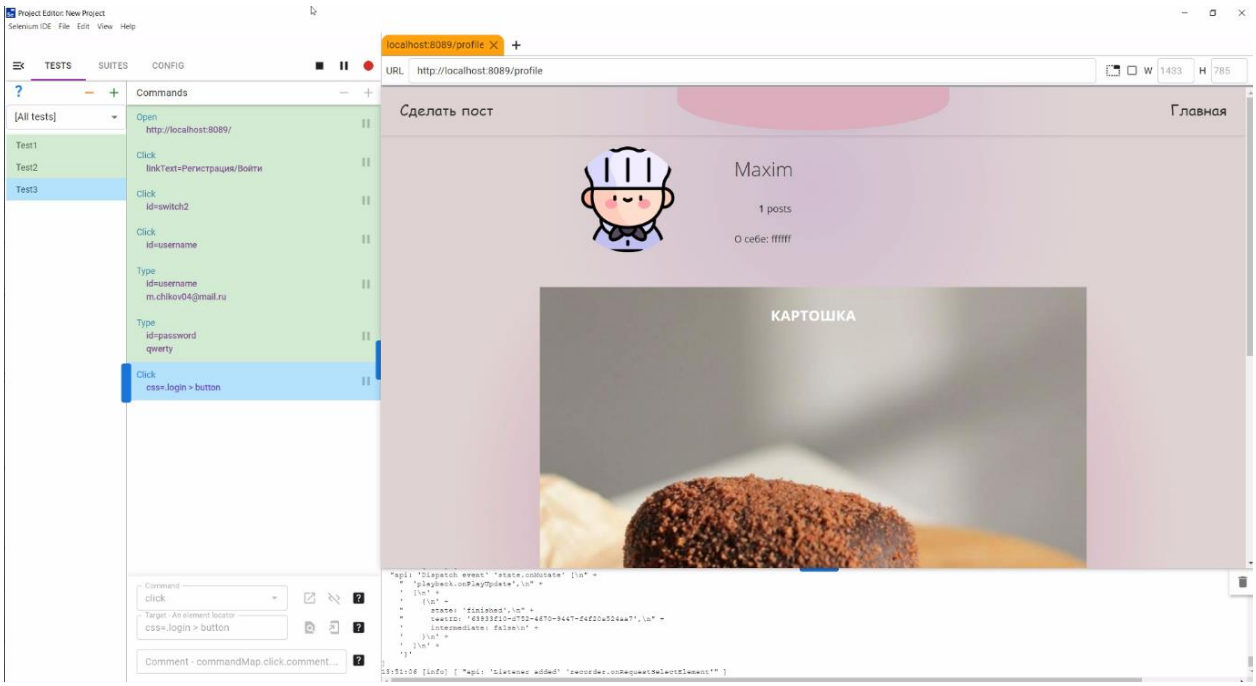

Рисунок 3 – Результат прохождения 3 теста