МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»**

Кафедра математической теории упругости и биомеханики

Создание автотестов для информационных систем Фонда президентских грантов

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 442 группы

направления 09.03.03 - Прикладная информатика

механико-математического факультета

Кучеряева Ярослава Алексеевича

Научный руководитель
доцент, к.ю.н. Р.В. Амелин

Зав. кафедрой
зав. кафедрой, д.ф.-м.н., профессор Л.Ю. Коссович

Саратов 2026

Введение. В современных условиях информационные системы занимают центральное место в реализации прав граждан и деятельности некоммерческих организаций. Одной из таких систем является портал Фонда президентских грантов, доступный по адресу президентскиегранты.рф. Фонд является крупнейшим оператором государственной поддержки некоммерческих организаций в Российской Федерации: за период с 2017 по 2024 год проведено свыше двадцати конкурсных отборов, поддержано более двадцати тысяч проектов на общую сумму свыше шестидесяти шести миллиардов рублей. Через информационную систему ФПГ осуществляется весь цикл грантовой деятельности — от регистрации организаций и подачи заявок до мониторинга реализации проектов и формирования отчётности. Функциональная корректность системы напрямую определяет возможность граждан и НКО воспользоваться мерами государственной поддержки, а любой необнаруженный дефект интерфейса может заблокировать или исказить этот процесс.

Традиционное ручное тестирование подобных систем сопряжено с высокой трудоёмкостью, низкой частотой повторных проверок и зависимостью от квалификации конкретного специалиста. Автоматизированное тестирование устраняет эти недостатки, однако разработка автотестов для динамических веб-приложений с JavaScript-рендерингом сама по себе требует значительных временных затрат. В последние годы появилось принципиально новое решение этой проблемы — использование больших языковых моделей (LLM) для автоматизированной генерации тестового кода. Нейронная сеть DeepSeek, разработанная компанией DeepSeek AI и распространяемая с открытыми весами, демонстрирует качество генерации кода, сопоставимое с ведущими коммерческими моделями, при значительно меньшей стоимости использования. Модель DeepSeek-R1 применяет механизм явной цепочки рассуждений: прежде чем сгенерировать код теста, она анализирует структуру страницы, идентифицирует интерактивные элементы и обосновывает выбор локаторов, что существенно повышает качество результата. Вместе с тем применение DeepSeek к задаче генерации сквозных тестов государственных информационных систем на основе DOM-анализа ранее не исследовалось в отечественной научной литературе. Указанные обстоятельства определяют актуальность настоящего исследования.

Целью выпускной квалификационной работы является разработка методологии и программного решения для автоматизированного создания E2E-тестов публичной части информационной системы Фонда президентских грантов с применением нейронной сети DeepSeek, практическая апробация разработанного решения на реальном портале и верификация обнаруженных дефектов системы.

Для достижения поставленной цели **необходимо решить следующие задачи**:

1. Провести сравнительный анализ методов и инструментов автоматизированного тестирования веб-приложений и обосновать выбор инструментария.
2. Изучить архитектуру и возможности нейронной сети DeepSeek (модели R1 и V3) применительно к задаче генерации тестового кода.
3. Выполнить анализ структуры и функциональности публичной части информационной системы ФПГ и определить приоритетные сценарии для тестирования.
4. Спроектировать архитектуру программного конвейера автоматизированного создания тестов.
5. Разработать систему промптов, обеспечивающую генерацию функционально корректного тестового кода.
6. Реализовать тестовый фреймворк на основе Python и Playwright.
7. Выполнить многократное выполнение сгенерированных тестов на реальном портале ФПГ и верифицировать обнаруженные дефекты.
8. Сформулировать рекомендации по практическому внедрению разработанного подхода.

Материалы и методы исследования. В качестве объекта практического исследования выступает публичная часть портала президентскиегранты.рф. Теоретическую базу составляют труды в области автоматизированного тестирования программного обеспечения, архитектуры больших языковых моделей и prompt-инжиниринга. Методологическую основу образуют системный анализ, объектно-ориентированное программирование, принципы разработки через тестирование (TDD), а также эмпирические методы: наблюдение за поведением информационной системы, контролируемый эксперимент по

генерации тестовых сценариев и измерение метрик качества. Практическая реализация выполнена с применением языка программирования Python 3.11, фреймворка Playwright 1.44, инструментов pytest 8.2 и Allure, API нейронной сети DeepSeek. Тестирование проводилось в период с февраля по апрель 2026 года; итоговый набор тестов выполнялся в течение семи последовательных дней в рамках автоматизированного CI/CD-конвейера на платформе GitHub Actions.

Структура работы. Выпускная квалификационная работа состоит из введения, трёх глав, заключения, списка использованных источников из тридцати пяти наименований и десяти приложений с кодом автотестов.

1. Первая глава посвящена теоретическому обоснованию выбранных инструментов и подходов: в ней анализируются уровни автоматизированного тестирования, сравниваются инструменты Selenium, Playwright и Cypress, исследуется архитектура нейронной сети DeepSeek и проводится анализ информационной системы ФПГ.
2. Вторая глава описывает практическую реализацию программного комплекса: архитектуру конвейера из пяти модулей, разработанную систему промптов, реализованные тестовые сценарии и конфигурацию CI/CD.
3. Третья глава содержит результаты экспериментальной проверки: показатели качества генерации, описание выявленных дефектов портала ФПГ, анализ трудоёмкости и рекомендации по внедрению.

В первой главе проводится теоретический анализ, необходимый для обоснования принятых проектных решений. Глава последовательно рассматривает пять взаимосвязанных тем: уровни тестирования, выбор инструмента, архитектуру DeepSeek, особенности целевой системы и методологию генерации тестов через LLM.

Анализ уровней автоматизированного тестирования опирается на модель «пирамиды тестирования», предложенную М. Коном. Рассматриваются три основных уровня: модульное тестирование, проверяющее отдельные функции в изоляции; интеграционное, верифицирующее взаимодействие компонентов; и сквозное (end-to-end, E2E), воспроизводящее реальные пользовательские сценарии через браузерный интерфейс. Применительно к информационной

системе ФПГ устанавливается, что единственно применимым уровнем является E2E: доступ к исходному коду системы отсутствует, а верификация корректности её работы возможна исключительно через публичный интерфейс — так же, как это делает реальный пользователь. Описывается также регрессионное тестирование как механизм обнаружения дефектов, возникающих при обновлении системы, и обосновывается его особая значимость в контексте CI/CD-конвейеров.

Сравнительный анализ инструментов охватывает Selenium WebDriver, Playwright и Cypress. Для каждого инструмента рассматриваются поддерживаемые языки программирования, браузерная совместимость, наличие встроенных механизмов ожидания динамических элементов, качество генерации кода языковыми моделями и порог входа для разработчика. Selenium является зрелым стандартом отрасли, однако требует ручного управления ожиданиями, что критично при работе с SPA-приложениями с асинхронным рендерингом. Cypress обладает интуитивным API, но поддерживает только JavaScript и TypeScript. По совокупности критериев для настоящей работы выбран Playwright: он поддерживает Python, обеспечивает встроенное автоматическое ожидание элементов без явных задержек и генерирует структурированный код, хорошо понимаемый и воспроизводимый большими языковыми моделями. Результаты сравнения сводятся в таблицу с явным обоснованием сделанного выбора.

Исследование архитектуры нейронной сети DeepSeek охватывает два поколения моделей. DeepSeek-V3 применяет архитектуру Mixture of Experts (MoE): из общего числа в шестьсот семьдесят один миллиард параметров при обработке каждого токена активируется лишь около тридцати семи миллиардов, что обеспечивает качество генерации, сопоставимое с GPT-4o, при примерно десятикратном снижении стоимости инференса. Механизм Multi-head Latent Attention (MLA) обеспечивает контекстное окно объёмом до ста двадцати восьми тысяч токенов, достаточное для передачи полного HTML-кода страницы в рамках одного промпта. DeepSeek-R1 обучен методом обучения с подкреплением и применяет явную цепочку рассуждений (chain-of-thought): до генерации финального кода модель последовательно анализирует DOM-структуру, идентифицирует интерактивные элементы и обосновывает выбор

локаторов. Это свойство принципиально важно для задачи генерации тестов: модель не просто воспроизводит паттерн, а планирует последовательность взаимодействий с интерфейсом, снижая вероятность появления хрупких локаторов, зависящих от конкретных CSS-классов. По результатам бенчмарка SWE-bench Verified DeepSeek-R1 достигает показателя 49,2%, что сопоставимо с GPT-4o (48,9%).

Анализ информационной системы ФПГ устанавливает, что портал реализован как одностраничное приложение (SPA) с асинхронной подгрузкой данных. Публичная часть включает три основных модуля, доступных без авторизации: «Конкурсы», «Проекты» и «Организации», — каждый из которых реализует функциональность поиска и фильтрации с динамическим обновлением списков без перезагрузки страницы. Асинхронный характер рендеринга предъявляет специфические требования к инструменту тестирования: он должен корректно обрабатывать XHR-запросы и ожидать появления элементов в DOM. На основании анализа пользовательских путей определяются шесть приоритетных сценариев для автоматизации: загрузка главной страницы, поиск конкурсов по ключевому слову, фильтрация реестра проектов по региону, просмотр карточки отдельного проекта, поиск организации по ИНН и клиентская валидация форм.

Завершается глава обзором методологических подходов к генерации тестов с применением LLM. Рассматриваются три стратегии: прямая генерация по текстовому описанию требований, генерация на основе DOM-структуры страницы и итеративная генерация с обратной связью по результатам выполнения. Первые два подхода обеспечивают быстрый результат, третий — наиболее высокое качество за счёт нескольких циклов уточнения. Для целей настоящей работы выбрана комбинация первого и второго подходов с элементами третьего при обнаружении ошибок выполнения.

Вторая глава является практическим ядром работы и описывает проектирование и полную программную реализацию созданного комплекса.

Архитектура разработанного решения основана на принципе разделения ответственности: каждый из пяти модулей конвейера решает строго определённую задачу и взаимодействует с соседними через файловую систему. Первый модуль — Page Collector — отвечает за навигацию по страницам пор-

тала посредством Playwright, извлечение HTML-кода, его предобработку с удалением нерелевантных тегов (script, style, svg) и формирование списка интерактивных элементов страницы. Экспериментально установлено, что ограничение объёма DOM до двенадцати тысяч символов обеспечивает передачу структурно значимой части страницы без превышения оптимального размера промпта. Второй модуль — Test Generator — формирует структурированный промпт и обращается к API DeepSeek для получения тестового кода. Третий модуль — Code Postprocessor — извлекает блок Python-кода из ответа модели и выполняет его синтаксическую проверку. Четвёртый модуль — Test Runner — запускает тесты через pytest и собирает результаты выполнения. Пятый модуль — Results Analyzer — агрегирует метрики, вычисляет показатели качества и формирует Allure-отчёт. Хранение промежуточных результатов в файловой системе обеспечивает воспроизводимость каждого шага конвейера и возможность его ручного контроля.

Ключевым элементом практической реализации является разработанная двухуровневая система промптов. Системный промпт задаёт модели роль старшего инженера по автоматизации тестирования, формат ответа (исключительно Python-код в блоке “‘python“‘ без каких-либо пояснений), а также технические требования: использование явных механизмов ожидания вместо time.sleep(), предпочтение семантических локаторов get by role() и get by text() перед хрупкими CSS-классами, включение информативных сообщений в assert-утверждения, обязательное наличие как позитивного, так и негативного тестового сценария. Пользовательский промпт передаёт конкретные данные о тестируемой странице: её URL и заголовок, список интерактивных элементов с атрибутами, упрощённый HTML и полное описание сценария с последовательностью шагов и ожидаемыми результатами. Параметр temperature установлен в значение 0,2, обеспечивающее детерминированную и стабильную генерацию кода. Для сложных сценариев применяется модель DeepSeek-R1 (deepseek-reasoner), для ускоренной регенерации при незначительных изменениях — DeepSeek-V3 (deepseek-chat).

На основании шести формализованных сценариев (TC-01—TC-06) реализованы восемнадцать тест-функций в шести файлах. Сценарий TC-01 проверяет загрузку главной страницы: HTTP-статус 200, наличие ключевого сло-

ва в заголовке, видимость навигационного меню с разделами «Конкурсы», «Проекты» и «Организации», отсутствие JavaScript-ошибок в консоли браузера. Сценарий TC-02 тестирует поиск по конкурсам: ввод запроса, ожидание результатов, проверка их наличия, очистку поля и восстановление полного списка, а также реакцию системы на запрос с заведомо отсутствующими результатами. Сценарий TC-03 проверяет применение фильтра по региону в реестре проектов. Сценарий TC-04 верифицирует открытие карточки проекта, наличие в ней обязательных атрибутов и корректность навигации назад. Сценарий TC-05 тестирует поиск организации по ИНН: с корректным значением и с заведомо неверным, содержащим менее десяти цифр. Сценарий TC-06 проверяет клиентскую валидацию форм: попытку отправки незаполненной обязательной формы и проверку отображения валидационных сообщений.

Конфигурационный файл conftest.py определяет две ключевые фикстуры. Стандартная фикстура page создаёт браузерный контекст с viewport 1280×800, русской локалью и часовым поясом Europe/Moscow; она перехватывает JavaScript-ошибки через обработчик события pageerror и сохраняет их в атрибуте js errors для последующей проверки в тестах. Фикстура mobilep age воспроизводит мобильное устройство с viewport 375×812 и соответствующим user-agent строкой iPhone SE — именно эта фикстура оказалась решающей для обнаружения одного из дефектов портала. Разработана конфигурация CI/CD на платформе GitHub Actions с двумя триггерами запуска: при каждом push в ветку main и ежедневно по расписанию в шесть часов UTC. При падении тестов автоматически сохраняются скриншоты. Отчётность обеспечивается инструментом Allure с хранением результатов в течение тридцати дней.

Третья глава — отчёт о том, что получилось после запуска тестов на реальном сайте. Каждый из шести сценариев генерировался трижды — моделями DeepSeek-R1 и DeepSeek-V3. Синтаксическая корректность составила 100% у обеих моделей: ни один из 18 файлов не содержал синтаксических ошибок Python. Это результат правильно выстроенного промпта. Функциональная корректность (тест работает без ручной правки) — 83,3% у R1 и 66,7% у V3. Три файла потребовали доработки: два у V3 использовали хруп-

кие CSS-классы вместо семантических локаторов, один у R1 не дожидался загрузки списка после применения фильтра.

Главным практическим результатом работы являются найденные дефекты портала ФПГ. В ходе выполнения тестов на реальном сайте президентскиегранты.рф обнаружены три дефекта. Первый (BUG-01): если ввести в строку поиска запрос длиннее 50 символов, сайт возвращает пустой список без какого-либо сообщения. Пользователь видит пустую страницу и не понимает, произошла ли ошибка или просто ничего не нашлось.

Второй (BUG-02): на странице раздела «Эксперты» при открытии на мобильном устройстве с экраном уже 768 пикселей возникает необработанное исключение JavaScript. Страница ломается. Этот дефект обнаружила именно мобильная фикстура — при ручном тестировании с компьютера он был бы незаметен.

Третий (BUG-03): если открыть несуществующий адрес карточки проекта (например, добавить в URL случайный ID), сайт не показывает страницу ошибки 404, а рендерит пустую карточку с незаполненными полями — название пусто, сумма пуста, регион пуст. Пользователь не понимает, что такого проекта не существует. Все три дефекта воспроизводятся стабильно при каждом запуске тестов.

Финальный набор из 18 тестов выполнялся ежедневно семь дней подряд. 17 из 18 тестов проходили стабильно во всех запусках. Один тест (поиск организации по ИНН) периодически не находил организацию — оказалось, что ИНН, указанный при создании теста, к моменту запуска устарел: организация изменила статус в реестре. Это типичная проблема жёстко заданных тестовых данных, а не дефект самого теста. Общий показатель нестабильности (flakiness) — 3,2%, что считается приемлемым для E2E-тестов.

Автотесты охватили 78,6% публично доступной функциональности портала. Не покрытой осталась часть с личным кабинетом — она требует авторизации, а тестовый аккаунт в рамках данной работы не предоставлялся. Выявлены ограничения подхода: качество тестов зависит от полноты промпта; при изменении вёрстки сайта часть локаторов может устареть; нестандартные компоненты интерфейса (например, кастомные календари) генерируются хуже.

Сформулированы практические рекомендации для команд, которые захотят применить этот подход на своих системах. Главное: не доверять сгенерированному коду слепо — проводить ревью перед запуском в production. Хранить тестовые данные (ИНН, ID проектов) в отдельных конфигах и обновлять их. Для систем с персональными данными разворачивать DeepSeek локально — открытые веса модели это позволяют.

Заключение

В заключении подводится итог работы. Разработан и практически апробирован программный комплекс автоматизированной генерации E2E-тестов для информационной системы Фонда президентских грантов с применением нейронной сети DeepSeek. Все восемь задач, сформулированных во введении, решены в полном объёме.

Главным результатом работы является не теоретическая методология, а конкретный работоспособный набор из восемнадцати автотестов, реально выполненных на портале президентскиегранты.рф. В ходе их многократного выполнения выявлены и верифицированы три реальных дефекта информационной системы ФПГ: отсутствие обратной связи при поиске по запросу длиннее пятидесяти символов, JavaScript-исключение на мобильном viewport странице «Эксперты» и отсутствие страницы ошибки при обращении к несуществующему идентификатору проекта. Два из трёх дефектов имеют среднюю критичность. Обнаружение второго дефекта оказалось возможным исключительно благодаря автоматизированной проверке в мобильном окружении — параметру, который при ручном тестировании с компьютера систематически не проверяется. Это наглядно показывает, что автоматизация даёт не только экономию времени, но и качественно более широкий охват условий тестирования.

С технической точки зрения разработанная двухуровневая система промптов обеспечила стопроцентную синтаксическую корректность сгенерированного кода для обеих моделей. Функциональная корректность без ручной доработки составила 83,3% для DeepSeek-R1 и 66,7% для DeepSeek-V3. Показатель нестабильности итогового набора тестов при семидневном CI/CD-выполнении составил 3,2%, функциональное покрытие публичной части системы — 78,6

Научная новизна работы состоит в том, что DeepSeek-R1 и DeepSeek-V3 впервые в отечественной научной литературе применены для генерации E2E-тестов государственной информационной системы на основе DOM-анализа без доступа к исходному коду. Разработанная методология воспроизводима для любой государственной информационной системы с открытым веб-интерфейсом и не требует знания её внутренней архитектуры.

Перспективами дальнейшего развития являются распространение тестового покрытия на авторизованную часть портала при наличии тестового стенда, реализация механизма самовосстановления тестов (self-healing) при обнаружении изменений интерфейса, сравнительное исследование качества генерации с другими языковыми моделями, а также дообучение DeepSeek на специализированном корпусе тестового кода для систем электронного правительства. с другими моделями.