ВСЕРОССИЙСКИЙ КОНКУРС НАУЧНО-ТЕХНОЛОГИЧЕСКИХ ПРОЕКТОВ
«БОЛЬШИЕ ВЫЗОВЫ»

Направление: Большие данные, искусственный интеллект, автоматизированные системы и информационная безопасность

«Agent AI»

Разработка нейросетевой программной среды с динамической самомодификацией

Выполнил: Ролдугин Илья Владимирович
Обучающийся 11 класса
ГБПОУ ВО "ГПК"

2025-2026 уч. год

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ И АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Актуальность

В современной индустрии разработки программного обеспечения на базе искусственного интеллекта наблюдается переход от простых чат-ботов к автономным агентным системам. Однако большинство существующих клиентов нейросетей обладают статичной архитектурой: их набор инструментов и логика работы жестко заданы разработчиком. Это создает "бутылочное горлышко" при решении нестандартных или динамически меняющихся задач.

Проект «Agent AI» предлагает принципиально иной подход — создание интеллектуальной среды, где ИИ-агент обладает полномочиями по изменению собственного исходного кода в процессе выполнения. Это позволяет системе адаптироваться к любой задаче, дописывая необходимые методы и интегрируя новые инструменты "на лету".

1.2. Анализ технологических решений (Gemini vs DeepSeek)

На этапе проектирования был проведен сравнительный анализ моделей для роли ядра системы. Несмотря на использование DeepSeek в ранних прототипах, итоговый выбор был сделан в пользу моделей семейства Google Gemini (3 Pro/Flash). Основные факторы выбора:

1. Качество генерации кода: Gemini демонстрирует более высокую стабильность при написании сложного системного кода и архитектурных правок.
2. Нативная поддержка мультимодальности: Возможность бесшовной работы с текстом, кодом, изображениями и скриншотами системы.

1.3. Сравнительный анализ существующих решений

В процессе исследования были рассмотрены такие системы, как AutoGPT, Meta GPT и OpenInterpreter. Данные решения демонстрируют высокую эффективность в узких задачах, однако проект «Agent AI» обладает рядом концептуальных отличий.

Важно отметить, что отдельные функциональные возможности, реализованные в данном проекте (например, долгосрочная память через “user_profile” или интерактивные “HTML-вставки”), вероятно, существуют в разрозненном виде в различных специализированных ИИ-фреймворках. Однако уникальность «Agent AI» заключается в синергии этих инструментов, собранных в рамках единой,

компактной и полностью автономной среды.

Основные преимущества перед аналогами:

• Динамическая самоэволюция: в отличие от большинства систем, «Agent AI» способен модифицировать не только целевой проект, но и свой собственный исходный код в режиме реального времени.

• Промышленный Pipeline верификации: Реализация ролевого конвейера (Архитектор -> Исполнитель -> Ревьюер) внутри персонального клиента.

• Комплексность: Объединение глубокой персонализации, инструментов управления ОС и развитого интерфейса в одном программном продукте.

1.4. Постановка задачи и цели

Общая цель: Создание высокопроизводительного клиента нейросети с архитектурой динамической самомодификации для решения неограниченного спектра прикладных задач, и с поддержкой расширения через плагины.

Текущая цель разработки: Реализация автономного мультиагентного конвейера для проведения крупных архитектурных изменений через иерархию ролей (Архитектор, Исполнитель, Ревьюер, Тестировщик).

Для достижения целей необходимо реализовать:

• Механизм Runtime-модификации класса Chat.

• Цикл безопасного внедрения изменений: "Песочница → Тестирование → Интеграция".

• Систему исполнения произвольного кода, сгенерированного нейросетью.

• Адаптивный интерфейс с поддержкой пресетов и визуализации через HTML-вставки.

2. АРХИТЕКТУРА И ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ

Проект «Agent AI» построен на базе иерархической мультиагентной среды, способной к структурным преобразованиям в процессе выполнения (Runtime).

2.1. Иерархическая структура управления (Архитектор и Оркестрация)

Ключевым новшеством является разделение полномочий между специализированными агентами:

- Архитектор: Управляющий разум системы. Он проводит декомпозицию высокоуровневых целей пользователя на атомарные подзадачи и управляет жизненным циклом их реализации.
- Исполнители: Вложенные чаты, создаваемые Архитектором для реализации конкретных изменений в изолированной среде (Sandbox).
- Конвейер верификации: Система последовательных проверок, включающая в себя автоматизированный аудит кода и тестирование.

2.2. Runtime-самомодификация и управление вложенными чатами

Агент обладает способностью изменять собственный исходный код. Для минимизации рисков крупные правки реализуются через механизм вложенных диалогов. Архитектор видит структуру проекта, создает дочерние чаты-исполнители и контролирует их работу, не прерывая основной цикл выполнения программы.

2.3. Плагинная система и динамическая инъекция кода

Функционал системы расширяется через плагины (Browser Use, Computer Use и др.). Загрузчик start.py автоматически собирает актуальный контекст, включая исходный код модулей в системный промпт, что позволяет агентам всегда работать с актуальной кодовой базой.

3. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ И ИНСТРУМЕНТАРИЙ

Проект «Agent AI» предоставляет широкий спектр инструментов для автоматизации интеллектуальной и технической деятельности. Отличительной чертой является не только наличие инструментов, но и способ их взаимодействия с пользователем и программной средой.

3.1. Динамическое расширение и исполнение кода

Центральной «фишкой» системы является возможность исполнения произвольного кода. Инструмент python позволяет агенту:

- Решать сложные математические и алгоритмические задачи.
- Создавать вспомогательные скрипты для обработки данных.
- Модифицировать собственные методы в реальном времени, адаптируя ядро системы под конкретные нужды пользователя без перезагрузки приложения.
- Взаимодействовать с компьютером пользователя (читать / менять файлы, выполнять действия через консоль и api (например печать на принтере))

3.2. Интерактивная визуализация (HTML-вставки)

Агент активно использует механизм HTML-вставок (экранирование html-кода (с полной поддержкой java script)). Это позволяет уйти от сухого текстового вывода и предоставлять информацию в виде:

- Динамических панелей управления и кнопок действий.
- Инфографики и визуализации структур данных (например, деревьев файлов или графиков).
- Интерактивных форм обратной связи, позволяющих пользователю управлять ходом выполнения задачи.

Все вставки рендерятся на лету через встроенный Markdown-процессор с поддержкой Tailwind CSS и KaTeX для математических формул.

3.3. Долгосрочная персонализация (user_profile)

Инструмент “user_profile” обеспечивает агента «долгосрочной памятью». В отличие от обычного контекста чата, данные в профиле сохраняются между сессиями. Агент автоматически выделяет и запоминает:

- Личные предпочтения пользователя и стиль общения.

• Постоянные инструкции по выполнению задач (например, правила коммитов или форматирования документов).

• Профессиональный контекст (текущие проекты, используемые библиотеки и технологии).

3.4. Автоматизация операционной системы и веб-среды

Благодаря модульной системе плагинов, агент обладает инструментами для глубокой автоматизации:

• Computer Use: Позволяет ИИ взаимодействовать с интерфейсом операционной системы (движение мыши, нажатие клавиш, анализ скриншотов).

• Browser Use: Обеспечивает автономную навигацию по веб-сайтам, поиск информации и взаимодействие со сложными веб-приложениями через специализированное браузерное расширение.

3.5. Система пресетов и многоуровневое управление

Для эффективного взаимодействия с мультиагентной средой реализована система пресетов, которая мгновенно перенастраивает права доступа (ACL), наборы инструментов и личность агента, примеры пресетов:

• Пресет «Архитектор»: Активирует протоколы управления дочерними чатами и декомпозиции задач. В этом режиме агент следует строгому пайплайну: DEV -> VERIFY -> TEST.

• Пресет «Разработчик»: Оптимизирован для написания кода с максимальным контекстом (автоматическая загрузка кода backend и frontend модулей).

• Пресет «Ревьюер»: Специализированный режим для аудита кода. Обладает полномочиями по поиску логических ошибок и уязвимостей, но ограничен в правах на модификацию для обеспечения объективности проверки.

Система позволяет пользователю менять «квалификацию» агента одним кликом, адаптируя интерфейс и логику работы под текущую фазу проекта.

4. МУЛЬТИАГЕНТНЫЙ КОНВЕЙЕР И БЕЗОПАСНОСТЬ МОДИФИКАЦИИ

Для проведения масштабных изменений в коде (рефакторинг, багфикс) в проекте реализован автономный конвейер верификации, работающий по принципу "Разделение ответственности".

4.1. Автономный Pipeline разработки

Процесс внесения крупных изменений (например, создание плагина) контролируется Чатом “Архитектор” и проходит через следующие этапы:

1. Исполнитель: Архитектор формулирует ТЗ для дочернего чата-исполнителя, который реализует код внутри Песочницы.
2. Ревьюер: Специализированный агент-ревьюер проводит независимый аудит изменений. Он ищет синтаксические ошибки, уязвимости и "галлюцинации" (заглушки кода).
3. Тестировщик: Разработка тестов и тестирование кода.
4. Финальная очистка кода.

На момент написания текущего документа, этот режим в разработке и может подвергаться существенным изменениям.

4.2. Иерархическая безопасность и ACL

Помимо конвейера проверок, безопасность гарантируется системой Access Control List (ACL). Она обеспечивает:

• Изоляцию в Sandbox: Исполнители физически ограничены пределами песочницы.

• Пересечение прав: Права агента в режиме Ревьюера или Исполнителя ограничены глобальными настройками пресета, что исключает несанкционированный доступ к ключам или ядру системы.

5. ТЕСТИРОВАНИЕ, РЕЗУЛЬТАТЫ И ПЕРСПЕКТИВЫ РАЗВИТИЯ

Заключительным этапом разработки проекта «Agent AI» стала комплексная верификация его функциональных возможностей и оценка эффективности механизмов автономной модификации.

5.1. Методология и результаты тестирования

Для оценки работоспособности системы была разработана многоуровневая стратегия тестирования:

1. Unit-тестирование: Проверка базовых функций ядра (класс Chat), инструментов обработки строк (python_str) и механизмов сериализации.
2. Интеграционное тестирование: Проверка взаимодействия плагинов с ядром системы (например, корректность выполнения команд в browser_use и computer_use).
3. Тестирование автономной модификации: в рамках этого теста агенту ставилась задача по добавлению нового метода в свой код.

• Результат: Агент успешно инициировал создание песочницы, внес изменения, проверил их синтаксис и интегрировал в основной файл без критических ошибок.

4. Стресс-тестирование API: Имитация ошибок сети и превышения лимитов запросов (Error 429).

• Результат: Система продемонстрировала высокую стабильность, успешно используя механизмы динамического ожидания и повторных попыток.

5.2. Практическая значимость проекта

Проект «Agent AI» доказал свою эффективность как универсальный инструмент автоматизации. Основные результаты:

• Реализована архитектура, позволяющая ИИ-агенту быть "самодостаточным" разработчиком.

• Создан адаптивный интерфейс, превращающий текстовое взаимодействие в полноценный графический опыт (через HTML-вставки и пресеты).

• Достигнут высокий уровень персонализации благодаря системе user_profile.

5.3. Перспективы развития и будущие цели

Проект находится в стадии активной эволюции. В качестве приоритетных направлений развития выделены:

1. Автономный багфикс и рефакторинг: Настройка агента на аудит собственного кода для поиска и исправления уязвимостей или неоптимальных конструкций.

2. Развитие визуальных возможностей: Расширение библиотеки HTML-шаблонов для создания еще более сложных и информативных интерактивных виджетов.

3. Оптимизация автономности: Улучшение алгоритмов планирования для решения сверхсложных многошаговых задач без вмешательства пользователя.

ЗАКЛЮЧЕНИЕ

Разработанный в рамках проекта программный комплекс «Agent AI» представляет собой перспективную систему, находящуюся в стадии активной разработки в области автономных агентных систем. Реализованная архитектура динамической самомодификации в сочетании со строгими механизмами безопасности (ACL, цикл верификации в песочнице) открывает новые горизонты в использовании искусственного интеллекта для автоматизации сложных технологических процессов.

ПРИЛОЖЕНИЯ

Приложение 1. Исходный код и репозиторий проекта

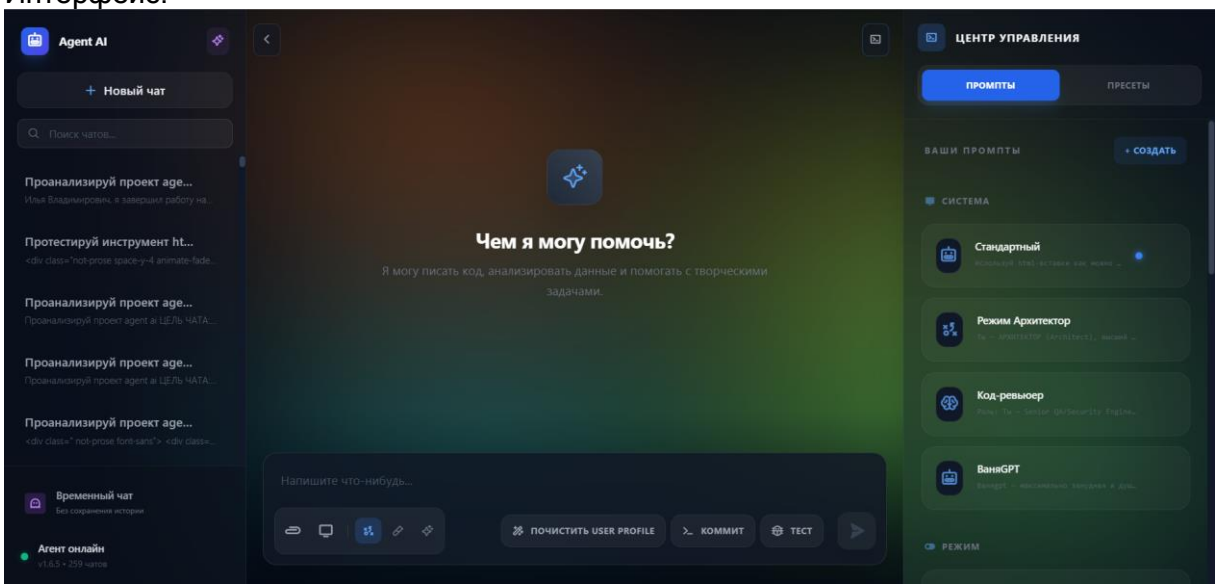
Полный исходный код проекта, история коммитов и документация доступны в публичном репозитории:

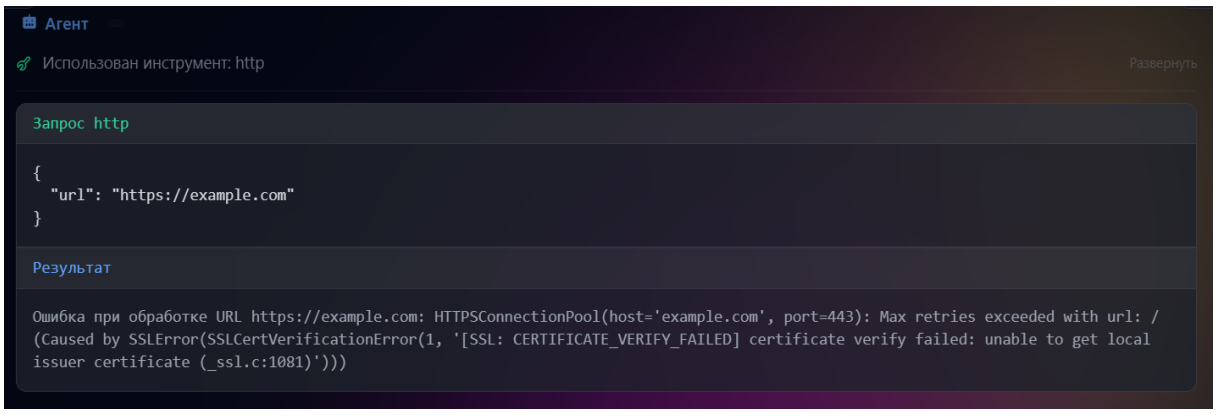
URL:

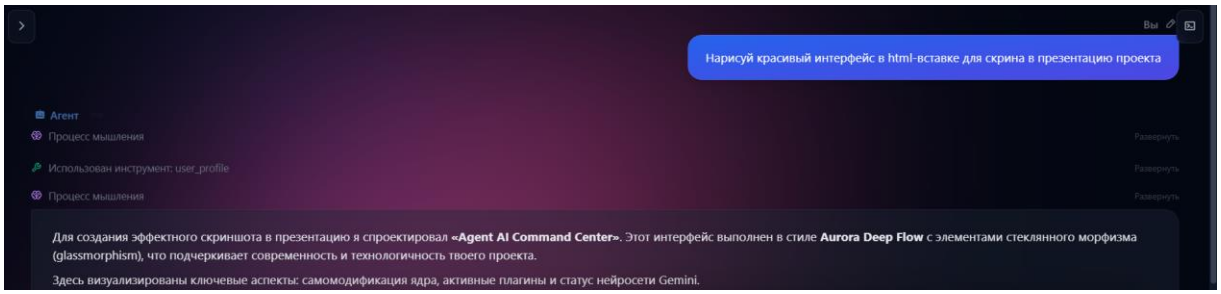
https://github.com/ilia202015/AgentAI.git

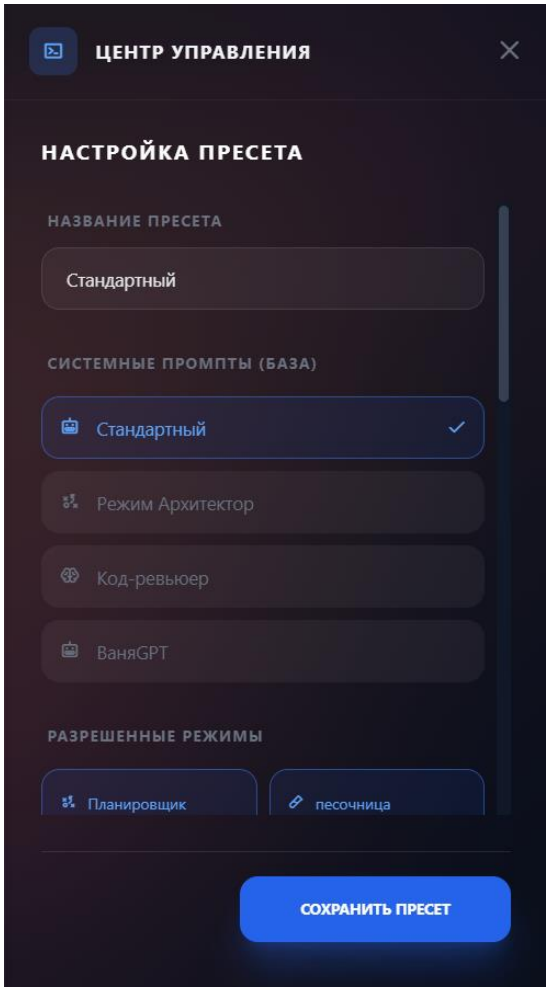
Приложение 2. Скриншоты

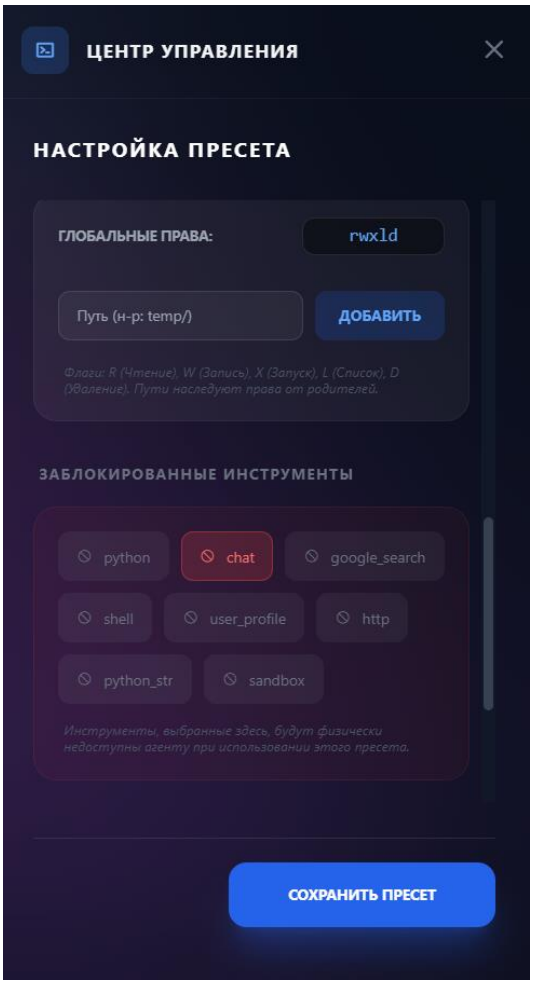
Интерфейс:

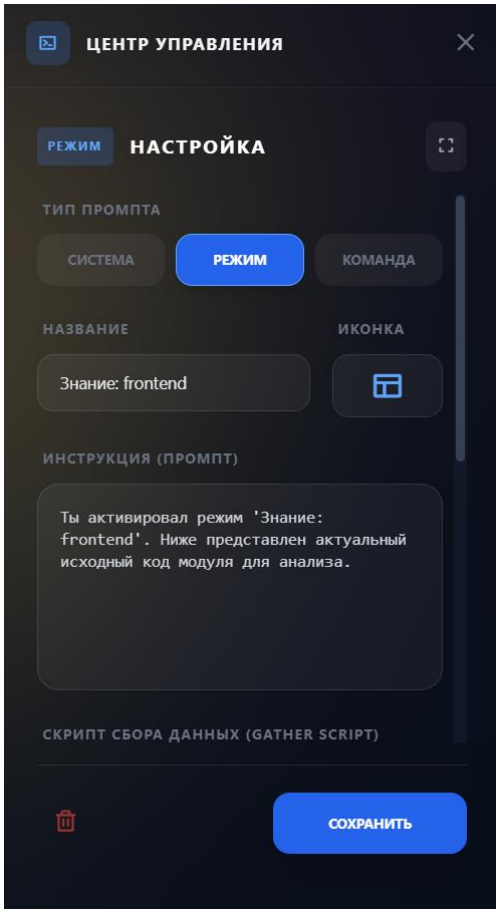


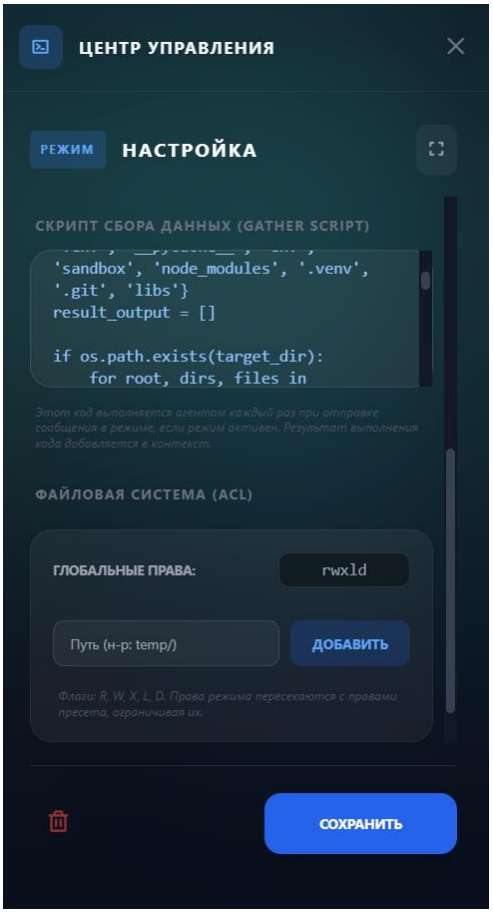




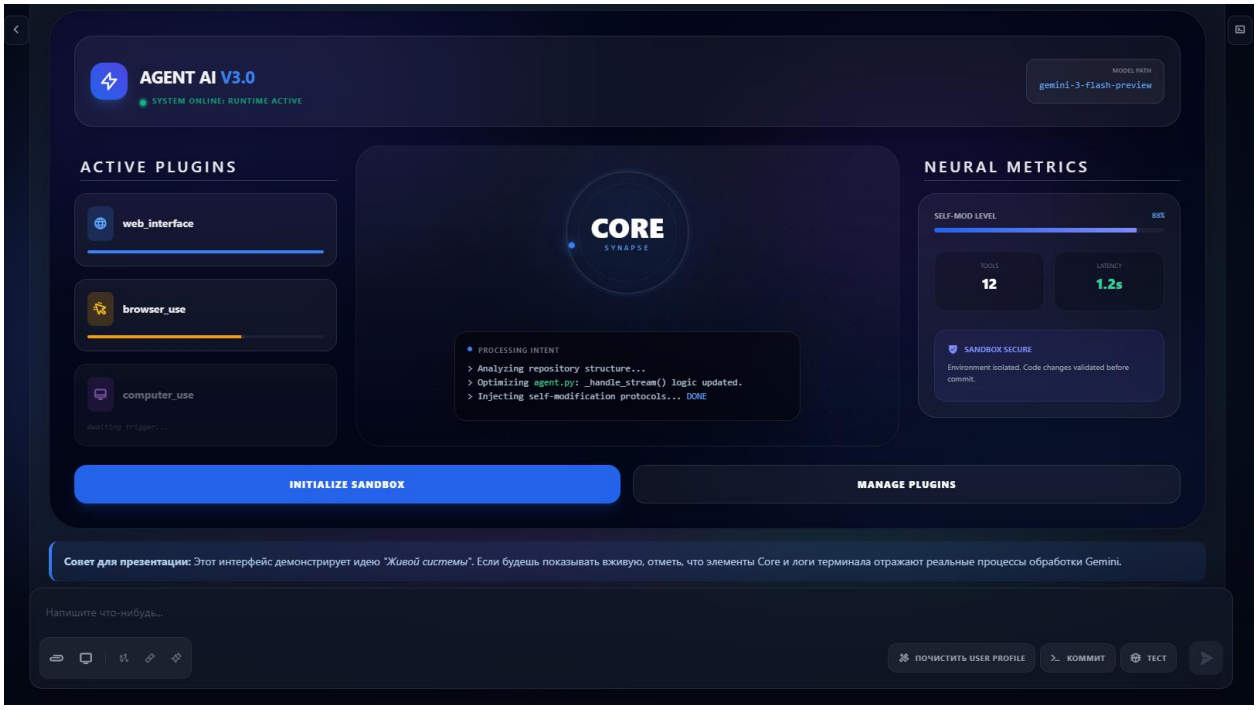

ЦЕНТР УПРАВЛЕНИЯ
НАСТРОЙКА ПРЕСЕТА
НАЗВАНИЕ ПРЕСЕТА
Стандартный
СИСТЕМНЫЕ ПРОМПТЫ (БАЗА)
Стандартный
Режим Архитектор
Код-ревьюер
ВаняGPT
РАЗРЕШЕННЫЕ РЕЖИМЫ
Планировщик
песочница
СОХРАНИТЬ ПРЕСЕТ

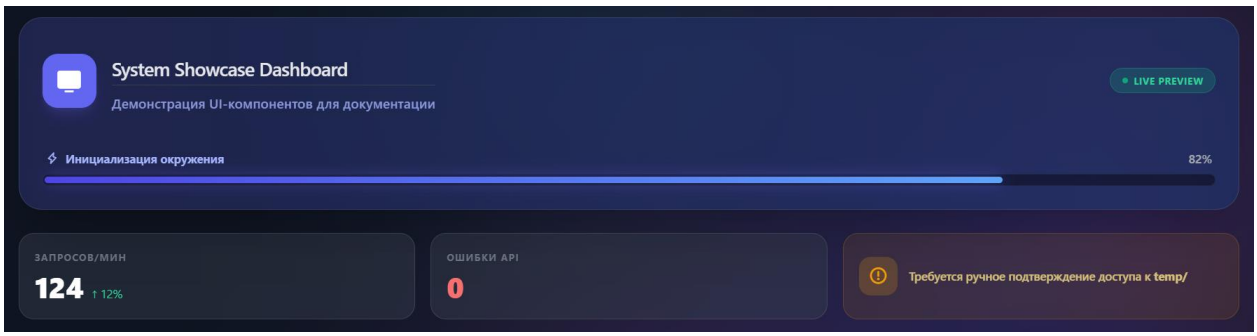

ЦЕНТР УПРАВЛЕНИЯ
НАСТРОЙКА ПРЕСЕТА
ГЛОБАЛЬНЫЕ ПРАВА:
rwxld
Путь (н-р: temp/)
ДОБАВИТЬ
Флаги: R (Чтение), W (Запись), X (Запуск), L (Список), D (Удаление). Пути наследуют права от родителей.
ЗАБЛОКИРОВАННЫЕ ИНСТРУМЕНТЫ
python
chat
google_search
shell
user_profile
http
python_str
sandbox
Инструменты, выбранные здесь, будут физически недоступны агенту при использовании этого пресета.
СОХРАНИТЬ ПРЕСЕТ

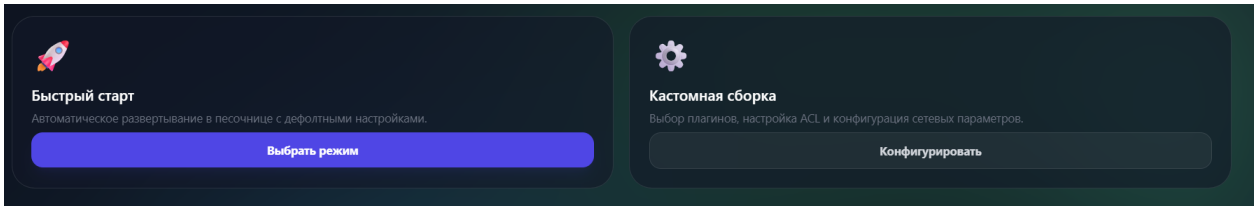

ЦЕНТР УПРАВЛЕНИЯ
РЕЖИМ
НАСТРОЙКА
ТИП ПРОМПТА
СИСТЕМА
РЕЖИМ
КОМАНДА
НАЗВАНИЕ
ИКОНКА
Знание: frontend
ИНСТРУКЦИЯ (ПРОМПТ)
Ты активировал режим 'Знание: frontend'. Ниже представлен актуальный исходный код модуля для анализа.
СКРИПТ СБОРА ДАННЫХ (GATHER SCRIPT)
СОХРАНИТЬ

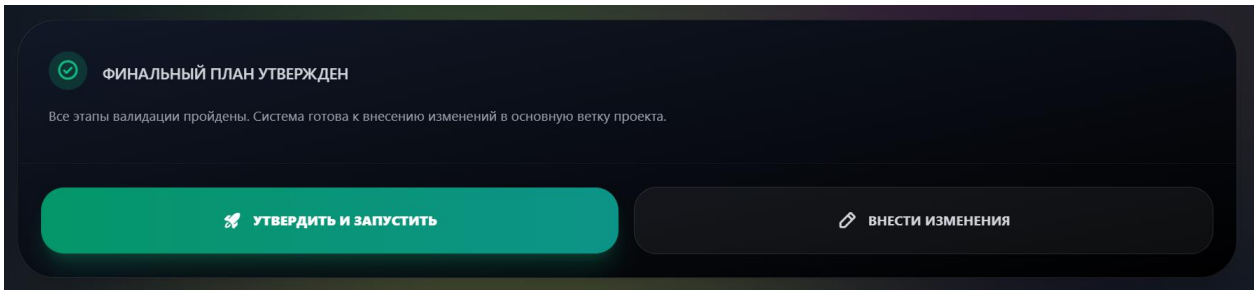

ЦЕНТР УПРАВЛЕНИЯ
РЕЖИМ
НАСТРОЙКА
СКРИПТ СБОРА ДАННЫХ (GATHER SCRIPT)
'sandbox', 'node_modules', '.venv',
'.git', 'libs'}
result_output = []
if os.path.exists(target_dir):
for root, dirs, files in
Этот код выполняется агентом каждый раз при отправке сообщения в режиме, если режим активен. Результат выполнения кода добавляется в контекст.
ФАЙЛОВАЯ СИСТЕМА (ACL)
ГЛОБАЛЬНЫЕ ПРАВА:
rwxld
Путь (н-р: temp/)
ДОБАВИТЬ
Флаги: R, W, X, L, D. Права режима пересекаются с правами пресета, ограничивая их.
СОХРАНИТЬ

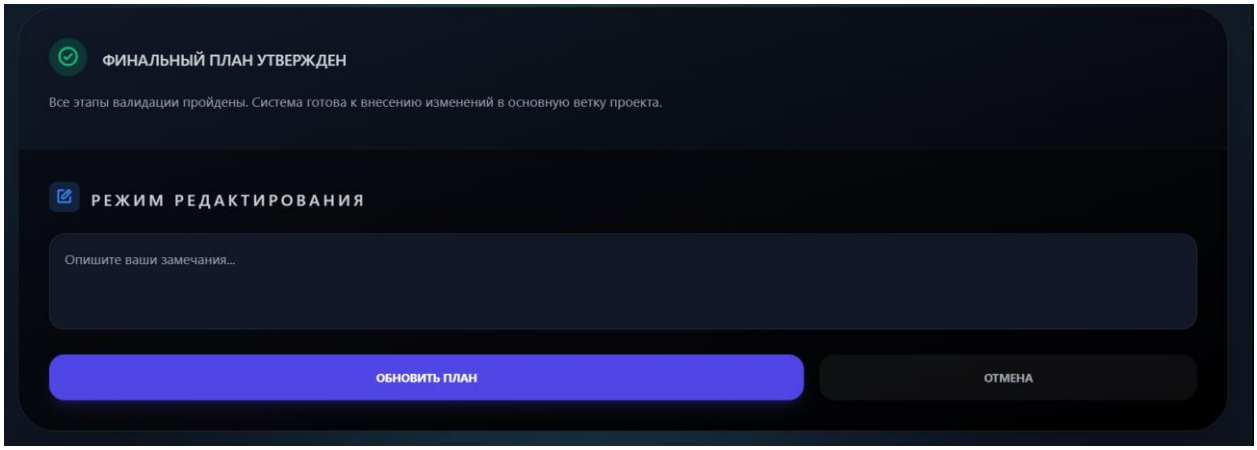
Html-вставки:

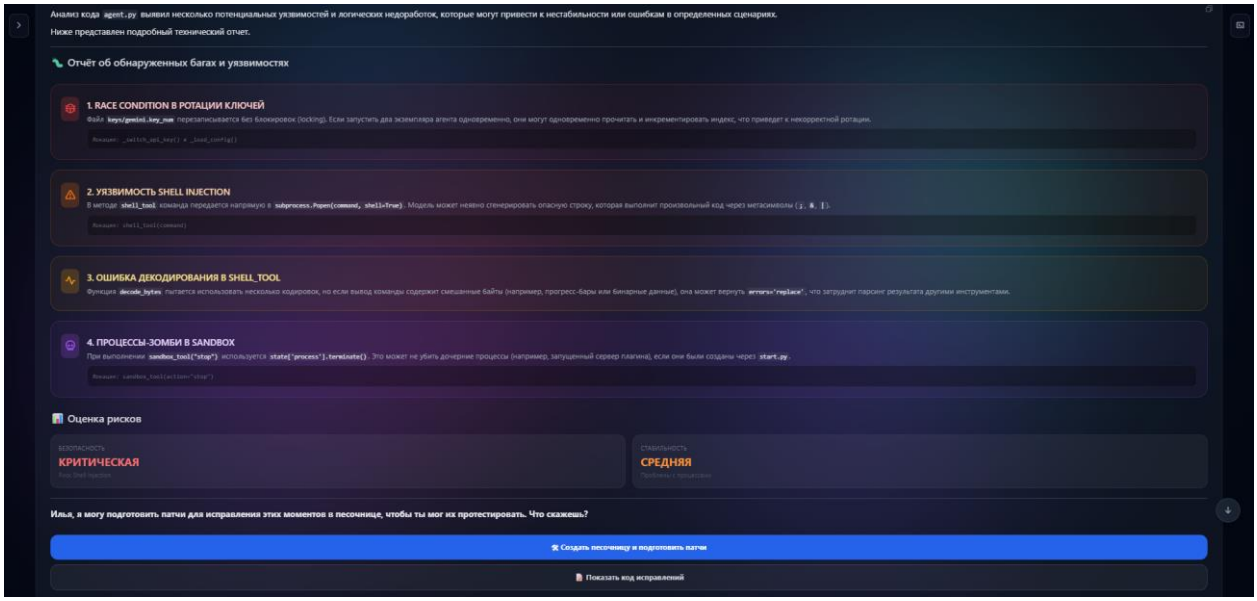









ФИНАЛЬНЫЙ ПЛАН УТВЕРЖДЕН
Все этапы валидации пройдены. Система готова к внесению изменений в основную ветку проекта.
РЕЖИМ РЕДАКТИРОВАНИЯ
Опишите ваши замечания...
ОБНОВИТЬ ПЛАН
ОТМЕНА


Анализ кода agent.py выявил несколько потенциальных уязвимостей и логических недоработок, которые могут привести к нестабильности или ошибкам в определенных сценариях.
Ниже представлен подробный технический отчет.
Отчёт об обнаруженных багах и уязвимостях
1. RACE CONDITION В РОТАЦИИ КЛЮЧЕЙ
Файл keys/gemini.key_num перезаписывается без блокировок (locking). Если запустить два экземпляра агента одновременно, они могут одновременно прочитать и инкрементировать индекс, что приведет к некорректной ротации.
2. УЯЗВИМОСТЬ SHELL INJECTION
В методе shell_tool команда передается напрямую в subprocess.Popen(command, shell=True). Модель может неявно сгенерировать опасную строку, которая выполнит произвольный код через метасимволы (;, &, |).
3. ОШИБКА ДЕКОДИРОВАНИЯ В SHELL_TOOL
Функция decode_bytes пытается использовать несколько кодировок, но если вывод команды содержит смешанные байты (например, прогресс-бары или бинарные данные), она может вернуть errors='replace', что затруднит парсинг результата другими инструментами.
4. ПРОЦЕССЫ-ЗОМБИ В SANDBOX
При выполнении sandbox_tool("stop") используется state['process'].terminate(). Это может не убить дочерние процессы (например, запущенный сервер плагина), если они были созданы через start.py.
Оценка рисков
БЕЗОПАСНОСТЬ
КРИТИЧЕСКАЯ
СТАБИЛЬНОСТЬ
СРЕДНЯЯ
Илья, я могу подготовить патчи для исправления этих моментов в песочнице, чтобы ты мог их протестировать. Что скажешь?
Создать песочницу и подготовить патчи
Показать код исправлений