Лабораторная работа по дисциплине «Автоматизация научных исследований»

Соломатов Александр Денисович
Гр. 5040102/50201

December 2025

1 Цель работы

Провести эксперимент по автоматической генерации одностраничного веб-сайта (лендинга) с использованием ИИ-инструмента, оценить качество результата по функциональным и техническим критериям, а также измерить временные затраты на выполнение задачи.

2 Задание

I. Исходные данные

Для выполнения задания необходимо подготовить следующие материалы:

1. **Промпт для генерации** — полное текстовое описание лендинга, включающее:
   - Название продукта/проекта;
   - Краткое описание (цель, преимущества, целевая аудитория);
   - Требуемые блоки (заголовок, описание, функции, отзывы, форма обратной связи, футер);
   - Предпочтения по стилю (минимализм, корпоративный стиль и т.д.).
2. **Технические требования** — правила для ИИ-системы:
   - Использовать только HTML, CSS и JavaScript (без внешних зависимостей) или разрешить конкретные библиотеки;
   - Обеспечить адаптивность (корректное отображение на мобильных устройствах);
   - Соблюдать принципы модульности и читаемости кода;
   - Обеспечить кроссплатформенную совместимость.
3. **Описание ИТ-проекта** — реальный или условный проект, для которого создаётся лендинг.

II. Порядок выполнения

1. Подготовьте промпт, технические требования и описание проекта
2. Передайте эти данные выбранному ИИ-инструменту (например, Cursor, GitHub Copilot, Claude и др.).
3. Запросите генерацию полноценного лендинга в виде одного или нескольких файлов (index.html, style.css, script.js).
4. Зафиксируйте время начала и окончания генерации (в минутах).
5. Проверьте результат и оцените качество согласно критериям.

3 Критерии оценки

1. **Бинарная проверка (Да/Нет):**
   - Лендинг содержит все запрошенные блоки
   - Код валиден и не содержит синтаксических ошибок
   - Страница корректно отображается в браузере
   - Реализована адаптивная вёрстка
   - Код легко читаем и подходит для повторного использования
2. **Качественная оценка (шкала 1-5):**
   - Соответствие дизайна и содержания исходному описанию проекта
   - Техническое качество клиентского кода
   - Общая пригодность лендинга для практического применения

4 Описание IT-проекта

В рамках магистерского диплома (НИР) по теме "Разработка метода генерации интерпретаторов DSL на основе графовых моделей грамматик и LLM", для инструмента хотелось бы разработать UI. Недавно я увидел замечательный интерфейс от Microsoft `https://microsoftlearning.github.io/ai-apps/scriptbook/` за референс решил взять именного его, расширив его немного в рамках use-case инструмента.

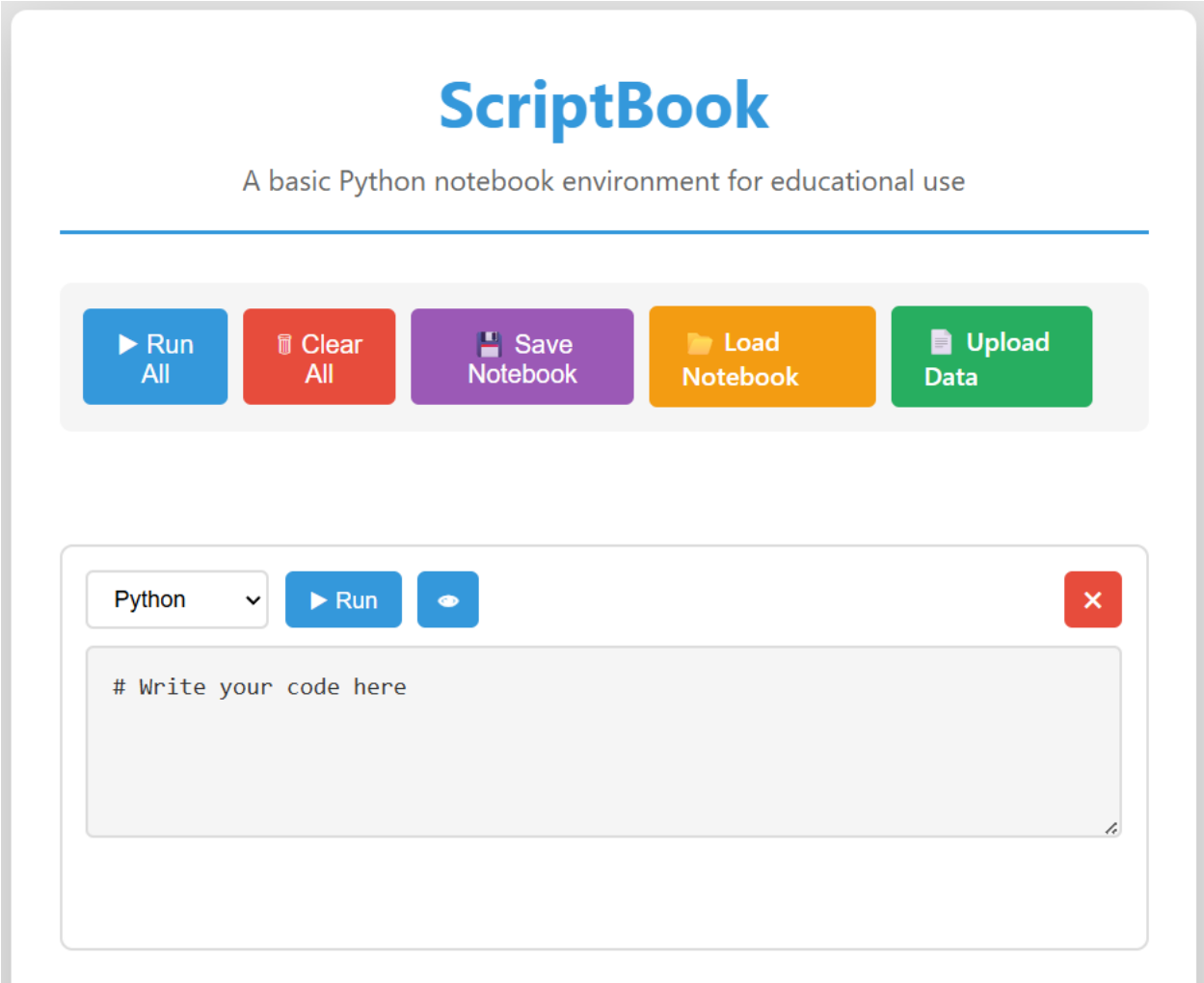

Figure 1: Референс

5 Промт

Промт 5 (Оптимизированный аналитический фреймворк)

```
# Техническое задание: WebDSLCompiler мокверсия(-)

## 1. Общее описание
Название**:** WebDSLCompiler
Цель**:** Интерактивная вебсреда- для разработки и тестирования DSL в
учебных целях по предмету Грамматики" и автоматы".

Архитектурный** подход:** Pure frontend мокверсия- с имитацией всех
операций. Все действия эмулируются на клиенте без реального бэкенда.

## 2. Пользовательские роли режимы( работы)
- Режим** Разработчик" DSL"** - работа с грамматиками и генерация парсеров
```

```
- Режим** Пользователь" DSL"** - написание и выполнение программ на
готовом DSL

**UI элемент:** Переключатель Режим‘[: Разработчик | Пользователь]‘ в
хедере

## 3. Детальные сценарии использования

### 3.1. Режим разработчика DSL
#### 3.1.1. Загрузка грамматики
- Кнопка**:** Загрузить" грамматику (.rbnf)"
- Формат**:** RBNF Расширенная( форма БэкусаНаура-)
- После** загрузки:** Открывается текстовый редактор с кодом
- Подсветка** синтаксиса моковая() необходима лишь в случае когда введены
коючевые слова:**
  - Ключевые слова блоков: ‘KEY‘, ‘TERMINAL‘, ‘NON-TERMINAL‘, ‘RULES‘, ‘
METABLOCK‘


#### 3.1.2. Генерация парсера
- Кнопка**:** Сгенерировать" парсер"
- Моковое** действие:**
- Сообщение: Парсер" сгенерирован успешно!"
- В консоли логируется структура для( демонстрации)
- Имя папки: из поля ‘name‘ в ‘METABLOCK‘ по( умолчанию "yourtool")

### 3.2. Режим пользователя DSL
#### 3.2.1. Подключение интерпретатора
- Кнопка**:** Подключить" DSLинтерпретатор-"
- Моковое** действие:** Сообщение Интерпретатор" yourtool подключен"

#### 3.2.2. Загрузка программы
- Кнопка**:** Загрузить" программу на DSL"
- Отображение**:** Текстовый редактор с загруженным кодом

#### 3.2.3. Запуск программы
- Кнопка**:** "RUN" интерактивная( анимация):
- Изначально: зелёный цвет
- При нажатии: становится серым, появляется спиннер
- Через 2 секунды: возвращается зелёный
- Сообщение: Программа" успешно выполнена! Сгенерирован файл output.json"
- Ссылка для скачивания: ‘output.json‘ моковый( JSON файл)
```

```
#### 3.2.4. Генерация и отображение AST
- Условие**:** После выполнения"" программы появляется кнопка
- Кнопка**:** Показать" AST (Abstract Syntax Tree)"
- Моковая** реализация:**
- Отправляется моковый запрос на бэкенд""
- Получаем статичное** изображение AST** заранее( подготовленный SVG/PNG)
- Отображаем изображение в модальном окне
- Важно**:** В мокверсии- НЕ делаем интерактивную визуализацию, только
картинку
- Подпись: В" реальной версии здесь будет интерактивное AST из DOT/JSON"

## 4. Технические требования фронтенд()

### 4.1. Базовые технологии
- **HTML5, CSS3, JavaScript (ES6+)**
- Текстовый** редактор:** ‘<textarea>‘ с базовой стилизацией или CodeMirror
 Lite

### 4.2. Адаптивность: Обязательна реализация Mobile-First или адаптивной
верстки с использованием
медиазапросов- (@media). Сайт должен корректно и удобно отображаться на
устройствах с
шириной экрана от 320px мобильные() до 1920px десктоп().

### 4.3. Структура проекта
webdsl-compiler/
   index.html # Главная страница
   css/
   style.css # Основные стили
    editor.css # Стили редакторов
   js/
     main.js # Основная логика
     mode-switcher.js # Переключение режимов
     mock-api.js # Имитация API
     syntax-highlighter.js # Базовая подсветка
   images/
     mock-ast.png # Пример AST для демонстрации
   mock-data/
    example.rbnf # Пример грамматики
    example-dsl.txt # Пример программы на DSL
    README.md # Описание запуска проекта
```

4.4. Визуализация AST упрощённая()
В мокверсии- - просто показываем статичное изображение AST с пояснением, что в реальной версии будет интерактивная визуализация.

4.5. Качество кода:
Модульность: CSS должен быть хорошо структурирован. Использовать семантические классы
например(, BEMметодология-) или организовать стили по секциям.
Читаемость: Код должен быть аккуратно отформатирован, с правильными отступами. Использовать
семантические HTMLтеги- (<header>, <section>, <article>, <footer>).
Кроссплатформенность-:
Код должен быть валидным соответствовать(стандартам W3C) и гарантированно работать в
последних версиях браузеров Chrome, Firefox, Safari, Edge на любой ОС (Windows, macOS,
Linux). Не использовать специфичные для браузера функции без fallback.

5. UI/UX дизайн

5.1. Макет страницы

[DSLTools WEBBook ver. Design By Gemini] Режим[:] ← Header

Загрузить[.rbnf] Сгенерировать[парсер] ← Панель действий режим(разработчика)
ИЛИ
 Подкл[. интерпретатор] Загрузить[прог.] [RUN] ← Панель действий режим(пользователя)

 // Текстовый редактор с кодом ← Основная рабочая область
 // Подсветка синтаксиса

 Кнопка[Показать" AST"] ← Появляется после RUN
 [output.json] ← Ссылка для скачивания

5.2. Цветовая схема

```
- Основной** фон:** ‘#f8f9fa‘
- Хедер**:** ‘#343a40‘ с белым текстом
- Кнопки**:**
- Основные: ‘#007bff‘ синий()
- RUN: ‘#28a745‘ зелёный() → ‘#6c757d‘ серый( при выполнении)
- Редактор**:** Белый фон, ‘#333‘ текст

### 5.3. Состояния интерфейса
- Загрузка** файлов:** Анимация спиннера (2 сек)
- Выполнение** программы:** Кнопка RUN меняет цвет + спиннер
- Уведомления**:** Всплывающие сообщения в правом верхнем углу
- **AST просмотр:** Модальное окно с изображением

### 5.4 Типографика
Четкие, читаемые шрифты без засечек например(, Roboto, Open Sans, Arial).
Иерархия: крупный шрифт для заголовков, средний для основного текста. Call-
to-action


## 6. Моковые данные

### 6.1. Пример .rbnf файла
METABLOCK {
  name = "SimpleMathDSL"
  version = "1.0"
}

KEY
  "func", "return", "if", "else"


TERMINAL
  NUMBER: /[0-9]+(\.[0-9]+)?/,
  ID: /[a-zA-Z_][a-zA-Z0-9_]*/,
  STRING: /"[^"]*"/


NON-TERMINAL
  Program, Function, Statement, Expression


RULES
```

```
  Program = Function*;
  Function = "func" ID "(" ")" "{" Statement* "}";
  Statement = "return" Expression ";";
  Expression = NUMBER | ID;

7. Футер

    WebDSLCompiler v1.0 учебный( проект)

    Разработчик: Имя[]
  GitHub: https://github.com/AlexSoulJk

  Для предмета Грамматики" и автоматы"
  Мокверсия-, 2025

8. ФИНАЛЬНЫЕ ТРЕБОВАНИЯ К ВЫВОДУ:
Сгенерируй готовые к использованию файлы: согласно приведенному дереву из
4.3. Структура проекта.
Код должен:
Содержать ВСЕ перечисленные блоки с предоставленным контентом.
Быть валидным и не содержать синтаксических ошибок.
Обеспечивать корректное отображение в браузере.
Иметь полностью реализованную адаптивную верстку
Быть легко читаемым и подходить для повторного использованиямодификации/
```

6 Результаты

- Для генерации сайта с помощью ИИ была выбрана модель DeepSeek.
- Было сгененрировано 11 файлов согласно ТЗ промта
- Файлы были сгененрированы в общей сложности за 5 минут. Модель была в режиме рассуждений - рассуждала 1 минуту 12 сек.

6.1 После первого промта

Главная страница

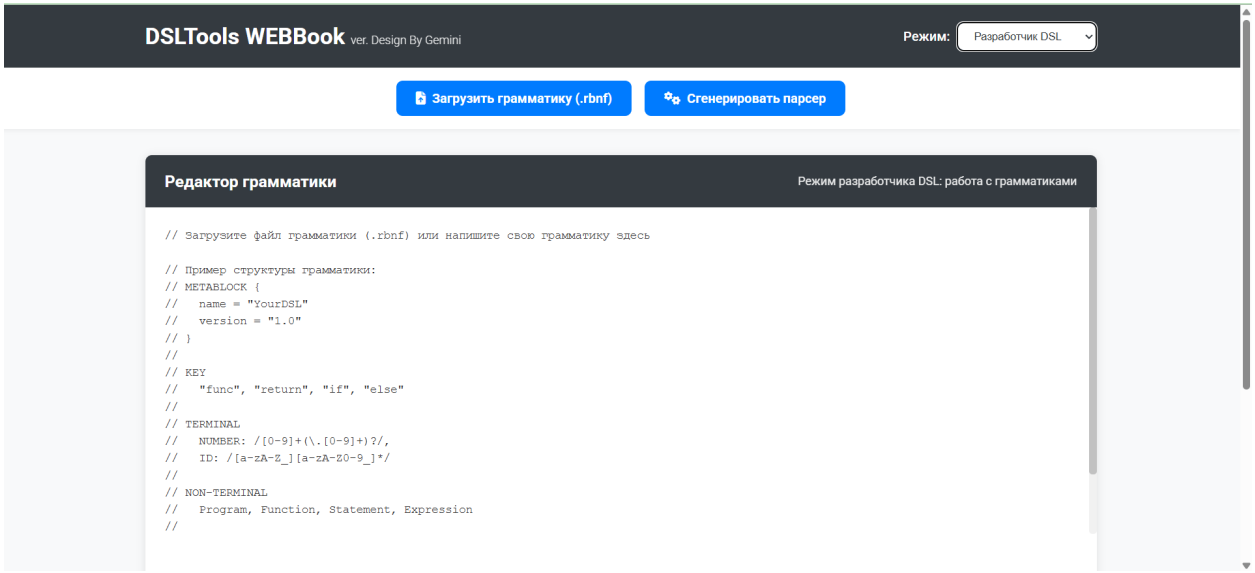

Figure 2: Страница разработчика

Страница Пользователь DSL

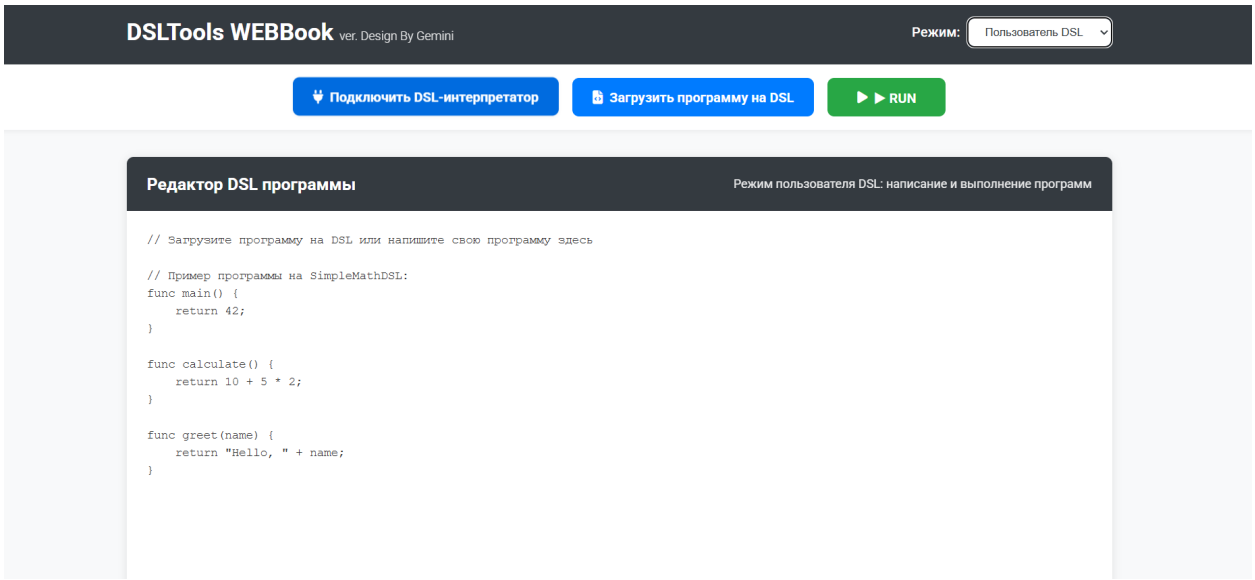

Figure 3

Хедер

Figure 4: Хедер

Футер

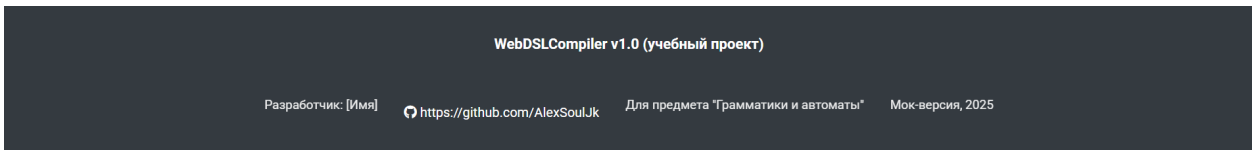

Figure 5: Футер

Пример отображения на телефоне

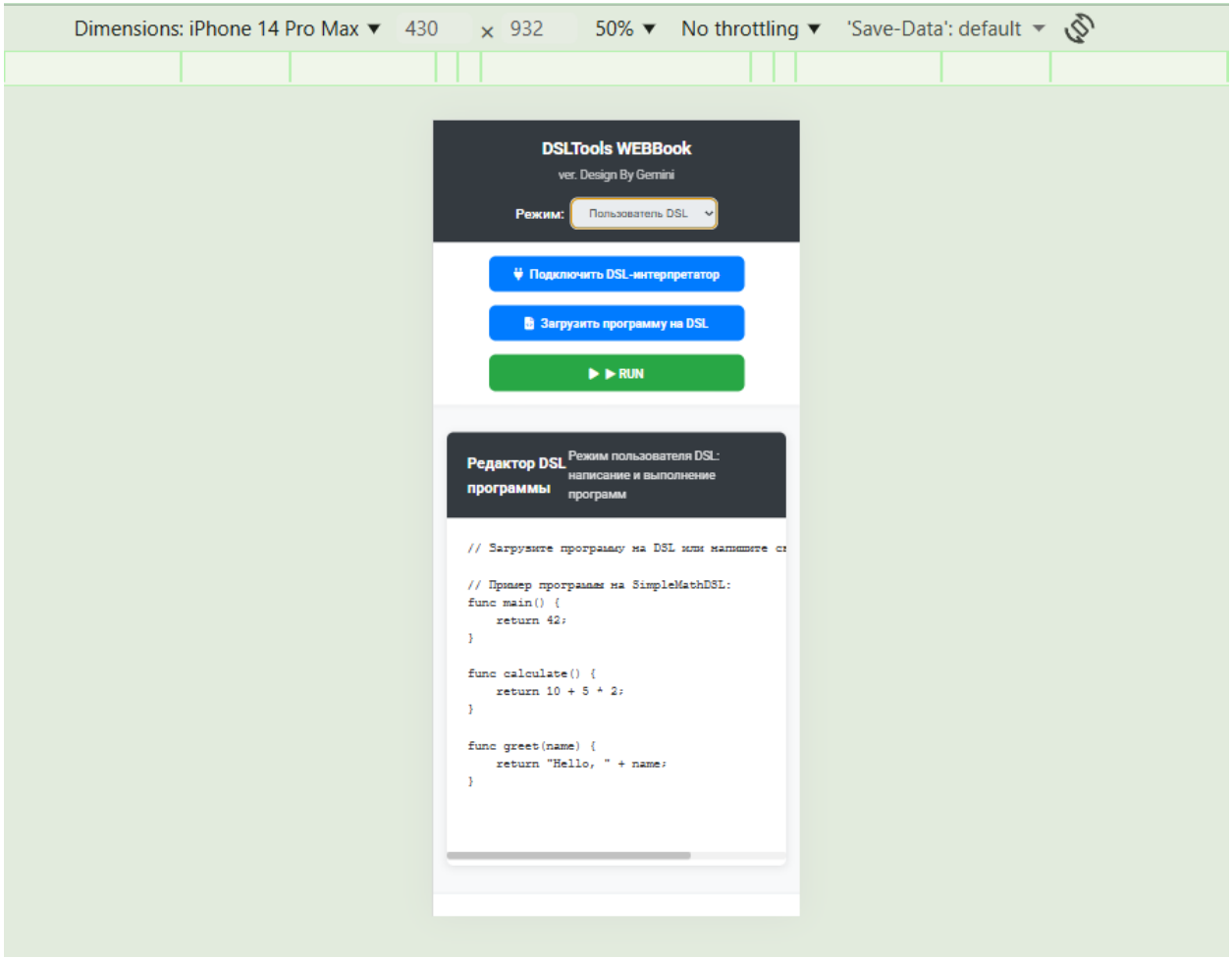

Figure 6: Отображения на телефоне

6.2 Отображение в браузере после уточняющего промпта

Текст уточнения:

Промт 5 (Оптимизированный аналитический фреймворк)

```
Вопервых-, нужно чтобы при нажатии на генерацию youtools генерировался zip
файл который можно было скачать - добавь отдельную кнопку скачать и( тут
пишем название zip файла) при нажатии скачиваем его.

Про Редактор:
```

При нажатии и выборе rbnf файла сделать отображение текста в(основном поле), который находится в файле.
Кроме того - Подсветка** синтаксиса моковая() необходима лишь в случае когда введены коючевые слова:**
- Ключевые слова блоков: ‘KEY‘, ‘TERMINAL‘, ‘NON-TERMINAL‘, ‘RULES‘, ‘METABLOCK‘
То есть их явно парсим простым регулярным выражением и подсвечиваем другим цветом отличным от базового например(желтым)
Кроме того хочется чтобы сделан был микро линтер для ключевых слов если(слово из названий ключевых слов то мы подчеркиваем его при неправильном написании)
И делаем подсветку на редактор - например красным красим фон, при нажатии на сгенерировать интерпретатор, если отсутсвют блоки.
Внутри редактора грамматики отображаем номер строки

Плюс добавить на страничку разработчика допом лог копиляции с(выводом сейчас ошибки осутсвия блок = ключевого слова в мок версии)
Изменить текст в кнопке загрузить сгенерировать парсер - на сгенерировать интерпретатор
Изменить в хедере Design By Gemini на Design by DeepSeek
Относительно пользователя DSL:
Когда нажимаем на подключить интерпретатор - пытаемся подгрузить папку с интерпретатором - по завершении загрузки - выводим сообщение о том, что все успешно - валидации на наличие базовой файловой структуры интерпретатора
На кнопке RUN сейчас отображается 2 треугольника - оставить только один
При нажатии на RUN предусмотреть отображение лога интерпретации в мок версии отображаем(его пока не произойдет обновление или не изменится режим)
Пишем начинаем компилировать, обращаемся к интерпретатору, вывод - return 0
В случае если программа - пустая RUN - серая некликабельная, если не подгружен интерпретатор - серая некликабельная.
Внутри редакторе программы отображаем номер строки

Разработчик - ИМЯ - Александр Соломатов
Ссылка на гитхаб отображается ниже слов в футере, спрятать в фразу: Ссылка на дистрибутив туда положить ту ссылку которая отображается. Добавить к этой фразе слева значек гитхаба

Работал примерно 6 минут, поправил часть ошибок, некоторые все же остались + добавились новые:

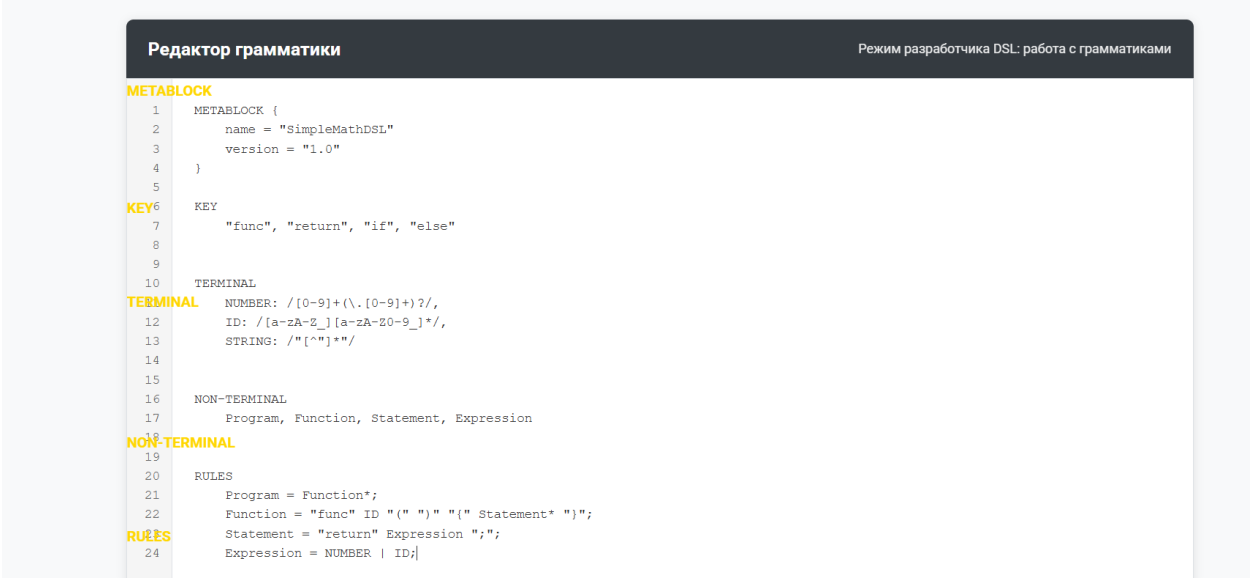

Figure 7: Редактор грамматики

Как видим, желтые слова появились - в моке это выглядит прекрасно, если бы слова прям в тексте выделялись. Модель этого не учла(

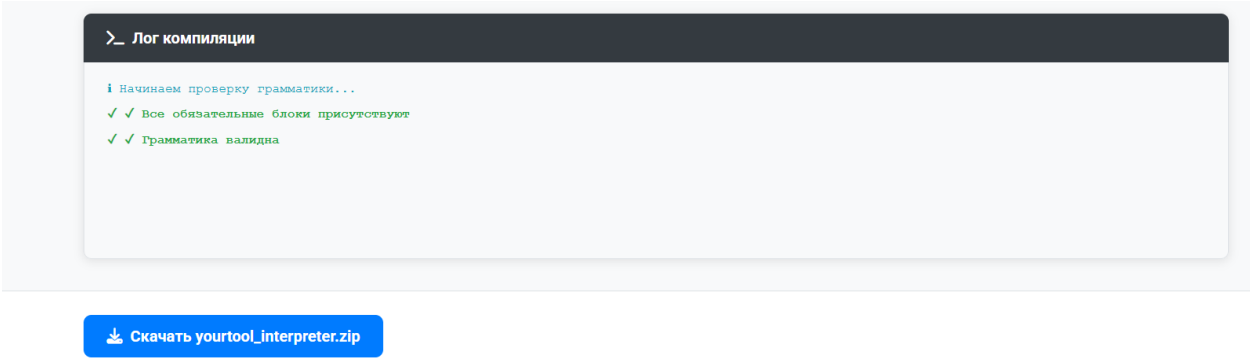

Figure 8: Окно лога + кнопка скачивания

Окно с логом добавилось, в нем отображаются все шаги, однако при открытии скачанного zip происходит ошибочка - zip не zip (

Figure 9

Как видно произошла ошибочка и текст с предыдущего режима почему то сюда переносится. На самом деле как и состояние редактора разработчика.

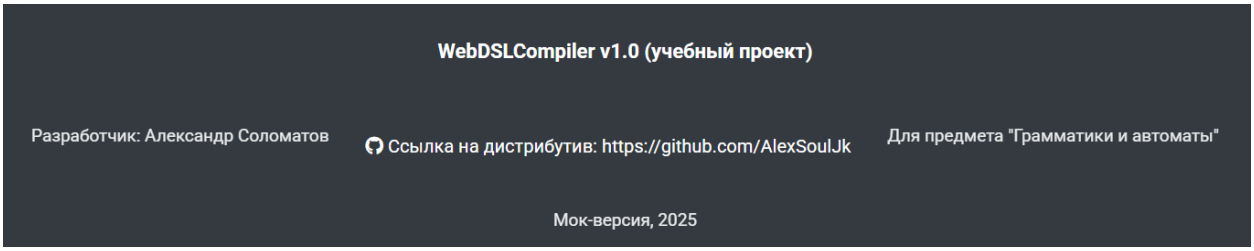

Figure 10

Футер почти правильно изменился, модель не поняла что я хотел сделать гиппер ссылку.

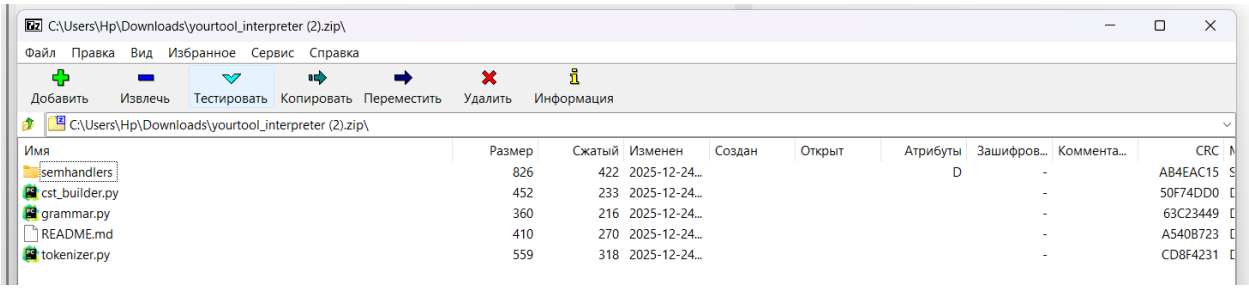

Figure 11: Zip sample

В последствии производил еще пару микро-фиксов через промт, zip генерируется! Хотя проблема с простым наложением не решилась. Модель не поняла, что нужно красить текст, а не накладывать такой же на текст)

7 Выводы

7.1 Таблицы результатов

Table 1: Проверка результата генерации лендинга до уточнения

Критерий проверки	**Результат (Да/Нет)**
Лендинг содержит все запрошенные блоки	Да
Код валиден и не содержит синтаксических ошибок	Да
Страница корректно отображается в браузере	Да
Реализована адаптивная вёрстка	Да
Код легко читаем и подходит для повторного использования	Да

Table 2: Оценка качества лендинга (по шкале от 1 до 5) до уточнения

Критерий оценки	**Балл (1–5)**
Соответствие дизайна и содержания исходному описанию проекта	5
Техническое качество клиентского кода	3
Общая пригодность лендинга для практического применения	5

Table 3: Проверка результата генерации лендинга после уточнения

Критерий проверки	**Результат (Да/Нет)**
Лендинг содержит все запрошенные блоки	Да
Код валиден и не содержит синтаксических ошибок	Нет появились баги и часть фун
Страница корректно отображается в браузере	Нет, накладывается текст
Реализована адаптивная вёрстка	Да
Код легко читаем и подходит для повторного использования	Да

1. Дизайн и структура — сильная сторона ИИ

- **Соответствие требованиям:** Обе версии лендинга (до и после уточнения) получили высокие балы (4/5) по соответствию дизайна и содержания исходному описанию.
- **Полнота реализации:** Все запрошенные блоки присутствуют в правильной логической последовательности.
- **Профессиональный визуал:** Цветовая палитра, типографика и компоновка полностью соответствуют запрошенному стилю.

2. Работоспособность кода — проблемная зона

- **Простая версия:** Первоначальный вариант с базовой функциональностью получил средние оценки по техническому качеству кода, из-за присутсвия багов.

Table 4: Оценка качества лендинга (по шкале от 1 до 5) после уточнения

Критерий оценки	**Балл (1–5)**
Соответствие дизайна и содержания исходному описанию проекта	4
Техническое качество клиентского кода	3
Общая пригодность лендинга для практического применения	4

- **Усложнённая версия:** После добавления дополнительных требований и исправлений, появились новые проблемы, хотя часть функционала была исправлена (скачанный zip стал работать).

3. Практическая пригодность

- **Базовая функциональность:** Версия первая получила высший балл за практическую пригодность.
- **Расширенная версия:** Несмотря на большее дополнения в практической плоскости, часть из них все же выглядят сыровато и частично ломают интерфейс.

ИИ-система демонстрирует высокую эффективность в создании дизайна, компоновки и вёрстки, однако испытывает трудности при реализации сложной интерактивной функциональнос Это подтверждает известный паттерн: генеративные модели хорошо справляются со статичными или простыми динамическими интерфейсами, но их возможности в создании сложной, безошибочной логики на JavaScript остаются ограниченными.

Заключение

Генерация лендингов с помощью ИИ является мощным инструментом, эффективность которого снижается с ростом сложности технической реализации. Система успешно создаёт профессиональное визуальное представление и структуру, однако для готовых решений с комплексной интерактивностью требуется дополнительная ручная доработка или упрощение технических требований.

Для задач, подобных созданию учебных инструментов, оптимальной стратегией является генерация базового варианта с помощью ИИ с последующей минимальной доработкой функциональности, что обеспечивает баланс между скоростью разработки и качеством конечного продукта.