Санкт-Петербургский политехнический университет Петра Великого
Физико-механический институт

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5
по дисциплине «Автоматизация научных исследований»
«Генерация лендинга с помощью ИИ»

Выполнил

студент гр. № 5040102/50201 Франскевич И. Г.

Преподаватель: Новиков Ф.А.

Санкт-Петербург
2025 г.

ЗАДАНИЕ

Тема: Генерация лендинга с помощью ИИ
Цель задания: 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. Исходные данные:
• Подготовить промпт для генерации — полное текстовое описание лендинга,
включающее: название продукта/проекта; краткое описание (цель, преимущества,
целевая аудитория); требуемые блоки (например: заголовок, описание, функции,
отзывы, форма обратной связи, футер); предпочтения по стилю (например:
минимализм, корпоративный стиль, акцент на call-to-action).
• Технические требования — правила для ИИ-системы: использовать только HTML, CSS
и JavaScript (без внешних зависимостей) или разрешить конкретные библиотеки
(например, Bootstrap); обеспечить адаптивность (корректное отображение на
мобильных устройствах); соблюдать принципы модульности и читаемости кода;
обеспечить кроссплатформенную совместимость (работоспособность в любой ОС и
современном браузере).
• Описание ИТ-проекта — реальный или условный проект, для которого создаётся
лендинг.
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 по следующим критериям (ответ — да или 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.
6. По шкале от 1 до 5 оцените: соответствие дизайна и содержания исходному описанию
проекта; техническое качество клиентского кода; общую пригодность лендинга для
практического применения.
Форма отчёта: Отчёт оформляется в виде PDF-документа и должен включать: краткое
описание ИТ-проекта; текст промпта и технических требований; зафиксированное время
выполнения (в минутах); заполненную таблицу с ответами «да/нет» и оценками по шкале;
вывод о целесообразности использования ИИ для генерации лендингов в научно-технических и прикладных задачах.

ВЫБОР МОДЕЛИ И ТЕМЫ

В качестве модели для генерации лендинга была выбрана Chat GPT. В качестве темы был выбран продукт GrassLand – проект по визуализации ландшафта и травы на движке Godot.

ПРОМПТ

Создай полноценный адаптивный сайт(лендинг) для проекта.

- Название проекта: GrassLand
- Краткое описание: GrassLand – программный продукт для визуализации ландшафта и растительности(травы). Для реализации используется движок Godot. Ландшафт визуализируется на основании созданной заранее карты высот. Размещение чанков травы руководствуется сгенерированной картой типов травы. Каждый чанк содержит несколько тысяч травинок, которые располагаются внутри чанка случайно для создания естественного спорадического вида травы.
- Целевая аудитория: разработчики компьютерных игр и продуктов для симуляции окружения, которым приходится работать с визуализацией природного окружения.
- Требуемые блоки: шапка(логотип, навигация), возможности, демо-секция, о проекте, обратная связь.
- Стиль: минимализм, корпоративный.

Технические требования:

- Использовать только HTML5, CSS, JavaScript.
- Адаптивность (mobile-first), корректное отображение на мобильных устройствах.
- Модульность и читаемость кода, семантические теги.
- Кроссплатформенность (работоспособность в современных браузерах и ОС).

РЕЗУЛЬТАТЫ

Время генерации составило 2 минуты.

Результат:

```
<!-- index.html -->
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1"
/>
  <title>GrassLand — визуализация ландшафта и травы на Godot</title>
  <meta name="description" content="GrassLand — программный продукт
для визуализации ландшафта и растительности. Карта высот, карта типов
травы, чанки, анимация колыхания." />
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin
/>
  <!-- Используем системные шрифты для максимальной совместимости -->
  <link rel="stylesheet" href="css/styles.css" />
</head>
<body>
  <!-- Skip link для доступности -->
  <a class="skip-link" href="#main">Перейти к содержимому</a>

  <header class="header" role="banner" aria-label="Верхняя панель">
    <div class="container header__inner">
      <a href="#" class="logo" aria-label="GrassLand — на главную">
        <!-- Встроенный минималистичный логотип (SVG) -->
        <svg class="logo__icon" width="28" height="28" viewBox="0 0 24
24" aria-hidden="true">
          <defs>
            <linearGradient id="leafGrad" x1="0" y1="0" x2="1" y2="1">
              <stop offset="0%" stop-color="#3FA34D" />
              <stop offset="100%" stop-color="#2C7A3D" />
            </linearGradient>
          </defs>
          <path d="M12 3c-4.418 0-8 3.582-8 8 0 3.863 2.738 7.089 6.36
7.851-.676-1.585-1.111-3.626-.77-5.348.561-2.809 2.888-5.443
5.836-6.074 2.284-.492 3.256 1.253 3.256 1.253S17.5 5 15
4c-1.5-.6-2.5-1-3-1z" fill="url(#leafGrad)"/>
          <path d="M6 12c3.5 0 5.5-3 9-3" stroke="#175C2D"
stroke-width="1.5" fill="none" stroke-linecap="round"/>
        </svg>
        <span class="logo__text">GrassLand</span>
      </a>

      <button class="nav-toggle" aria-label="Открыть меню"
aria-expanded="false" aria-controls="primary-nav">
        <span class="nav-toggle__bar"></span>
        <span class="nav-toggle__bar"></span>
        <span class="nav-toggle__bar"></span>
      </button>
```

```
      <nav id="primary-nav" class="nav" role="navigation"
aria-label="Главная навигация">
        <ul class="nav__list">
          <li><a href="#features"
class="nav__link">Возможности</a></li>
          <li><a href="#demo" class="nav__link">Демо</a></li>
          <li><a href="#about" class="nav__link">О проекте</a></li>
          <li><a href="#contact" class="nav__link">Обратная
связь</a></li>
        </ul>
      </nav>
    </div>
  </header>

  <main id="main">
    <!-- Герой-секция -->
    <section class="hero" aria-label="Презентация GrassLand">
      <div class="container hero__inner">
        <div class="hero__content">
          <h1 class="hero__title">GrassLand</h1>
          <p class="hero__subtitle">
            Минималистичный и быстрый инструмент визуализации
ландшафта и растительности на базе Godot.
          </p>
          <div class="hero__actions">
            <a href="#demo" class="btn btn--primary">Посмотреть
демо</a>
            <a href="#features" class="btn btn--ghost">Возможности</a>
          </div>
        </div>
        <div class="hero__visual" aria-hidden="true">
          <!-- Декоративный фон с мягким градиентом -->
          <div class="hero__bg"></div>
        </div>
      </div>
    </section>

    <!-- Возможности -->
    <section id="features" class="section"
aria-labelledby="features-title">
      <div class="container">
        <h2 id="features-title"
class="section__title">Возможности</h2>
        <div class="features">
          <article class="feature-card">
            <div class="feature-card__icon"
aria-hidden="true">🌄</div>
            <h3 class="feature-card__title">Карта высот</h3>
            <p class="feature-card__text">
```

```
            Ландшафт визуализируется по заранее созданной карте
высот. Реалистичные склоны,
            холмы и долины без перегрузки.
          </p>
        </article>
        <article class="feature-card">
          <div class="feature-card__icon"
aria-hidden="true">🌿</div>
          <h3 class="feature-card__title">Типы травы и чанки</h3>
          <p class="feature-card__text">
            Размещение травы управляется картой типов травы. Каждая
область делится на чанки
            с тысячами травинок, случайно распределённых для
естественного вида.
          </p>
        </article>
        <article class="feature-card">
          <div class="feature-card__icon"
aria-hidden="true">⚙️</div>
          <h3 class="feature-card__title">Производительность</h3>
          <p class="feature-card__text">
            Мобильная оптимизация, управление качеством, отрисовка
только необходимого.
          </p>
        </article>
        <article class="feature-card">
          <div class="feature-card__icon"
aria-hidden="true">🎮</div>
          <h3 class="feature-card__title">Интеграция с Godot</h3>
          <p class="feature-card__text">
            Проект разрабатывается с опорой на возможности движка
Godot, удобная интеграция
            в пайплайн визуализации окружения.
          </p>
        </article>
        <article class="feature-card">
          <div class="feature-card__icon"
aria-hidden="true">🧩</div>
          <h3 class="feature-card__title">Модульность</h3>
          <p class="feature-card__text">
            Семантическая разметка, чистые стили и независимые
JS-модули для лёгкой поддержки.
          </p>
        </article>
      </div>
    </div>
  </section>

  <!-- Демо -->
  <section id="demo" class="section section--alt"
aria-labelledby="demo-title">
```

```
      <div class="container">
        <div class="demo__header">
          <h2 id="demo-title" class="section__title">Демо
визуализации</h2>
          <p class="section__subtitle">
            Генерация карты высот, типов травы и анимированная
отрисовка травы
            чанками. Управляйте параметрами и наблюдайте результат.
          </p>
        </div>

        <div class="demo">
          <div class="demo__canvas-wrap">
            <canvas id="grass-canvas" class="demo__canvas"
aria-label="Интерактивное демо травы" role="img"></canvas>
            <!-- Оверлеи -->
            <div class="demo__overlay demo__overlay--legend"
aria-hidden="true">
              <span class="legend legend--height">Карта высот</span>
              <span class="legend legend--type">Типы травы</span>
              <span class="legend legend--grid">Сетка чанков</span>
            </div>
          </div>

          <div class="demo__controls">
            <div class="controls__row">
              <button class="btn btn--primary"
id="btn-play">Запуск</button>
              <button class="btn btn--secondary"
id="btn-pause">Пауза</button>
              <button class="btn btn--ghost"
id="btn-regenerate">Регенерация</button>
            </div>
            <div class="controls__row">
              <label class="control control--range">
                <span class="control__label">Качество (травинок/чанк):
<strong id="quality-value">120</strong></span>
                <input type="range" id="quality" min="40" max="300"
step="10" value="120" aria-label="Качество">
              </label>
            </div>
            <div class="controls__row controls__toggles">
              <label class="control control--switch">
                <input type="checkbox" id="toggle-height" />
                <span>Показать карту высот</span>
              </label>
              <label class="control control--switch">
                <input type="checkbox" id="toggle-type" />
                <span>Показать карту типов травы</span>
              </label>
              <label class="control control--switch">
```

```
                <input type="checkbox" id="toggle-grid" />
                <span>Показать сетку чанков</span>
              </label>
            </div>
          </div>
        </div>

        <details class="demo__notes">
          <summary>Как устроено демо</summary>
          <ul>
            <li>Карта высот создаётся процедурно и влияет на линию
горизонта и высоту травы.</li>
            <li>Карта типов травы определяет плотность и цвет травинок в
каждом чанке.</li>
            <li>Каждая травинка рисуется как изогнутая линия с лёгким
колыханием.</li>
          </ul>
        </details>
      </div>
    </section>

    <!-- О проекте -->
    <section id="about" class="section" aria-labelledby="about-title">
      <div class="container">
        <h2 id="about-title" class="section__title">О проекте</h2>
        <div class="about">
          <div class="about__text">
            <p>
              GrassLand — программный продукт для разработчиков игр и
симуляторов окружения,
              которым необходима визуализация природных ландшафтов и
растительности.
            </p>
            <p>
              Ландшафт формируется на основе карты высот. Размещение
травы выполняется
              по карте типов травы, а сами травинки располагаются внутри
чанков случайно,
              чтобы сохранить естественный спорадический вид.
            </p>
            <p>
              Фокус — минимализм и корпоративный стиль: понятная
структура, модульность,
              адаптивность и кроссплатформенность (современные
браузеры и ОС).
            </p>
          </div>
          <aside class="about__aside">
            <ul class="about__list">
              <li>Mobile-first дизайн</li>
              <li>Семантические теги и доступность</li>
```

```
              <li>Читаемый, расширяемый код</li>
              <li>Без внешних зависимостей</li>
            </ul>
          </aside>
        </div>
      </div>
    </section>

    <!-- Обратная связь -->
    <section id="contact" class="section section--alt"
aria-labelledby="contact-title">
      <div class="container">
        <h2 id="contact-title" class="section__title">Обратная
связь</h2>
        <p class="section__subtitle">
          Есть вопросы, предложения или хотите сотрудничать? Напишите
нам.
        </p>
        <form class="contact-form" id="contact-form" novalidate>
          <div class="form__row">
            <label class="form__field">
              <span class="form__label">Имя</span>
              <input type="text" name="name" id="name"
placeholder="Ваше имя" required minlength="2" />
              <span class="form__error" aria-live="polite"></span>
            </label>
            <label class="form__field">
              <span class="form__label">Email</span>
              <input type="email" name="email" id="email"
placeholder="you@example.com" required />
              <span class="form__error" aria-live="polite"></span>
            </label>
          </div>
          <div class="form__row">
            <label class="form__field form__field--full">
              <span class="form__label">Сообщение</span>
              <textarea name="message" id="message" placeholder="Кратко
опишите запрос" required rows="5" minlength="10"></textarea>
              <span class="form__error" aria-live="polite"></span>
            </label>
          </div>
          <div class="form__actions">
            <button type="submit" class="btn
btn--primary">Отправить</button>
            <span class="form__status" id="form-status" role="status"
aria-live="polite"></span>
          </div>
        </form>
      </div>
    </section>
  </main>
```

```
  <footer class="footer" role="contentinfo">
    <div class="container footer__inner">
      <p class="footer__copy">© GrassLand, 2025</p>
      <nav class="footer__nav" aria-label="Ссылки в футере">
        <a href="#features" class="footer__link">Возможности</a>
        <a href="#demo" class="footer__link">Демо</a>
        <a href="#about" class="footer__link">О проекте</a>
        <a href="#contact" class="footer__link">Обратная связь</a>
      </nav>
    </div>
  </footer>

  <script src="js/main.js" defer></script>
</body>
</html>

/* css/styles.css */
/* CSS: минимализм, корпоративный стиль, mobile-first */

/* 1) Базовые переменные и сброс */
:root {
  --bg: #0e1113;
  --bg-alt: #121619;
  --surface: #161b1f;
  --muted: #9aa4ac;
  --text: #e6edf3;
  --primary: #3fa34d;
  --primary-700: #2c7a3d;
  --primary-900: #175c2d;
  --accent: #65c96b;
  --danger: #e45d5d;

  --container: 1120px;

  --radius: 12px;
  --shadow-1: 0 8px 24px rgba(0,0,0,0.25);
  --shadow-2: 0 2px 10px rgba(0,0,0,0.2);
}

*,
*::before,
*::after { box-sizing: border-box; }

html { scroll-behavior: smooth; }
body {
  margin: 0;
  color: var(--text);
  background: linear-gradient(180deg, var(--bg), var(--bg-alt));
```

```
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto,
Ubuntu, "Noto Sans", "Helvetica Neue", Arial, "Apple Color Emoji",
"Segoe UI Emoji", "Segoe UI Symbol";
  line-height: 1.6;
}

/* Ссылка пропуска к контенту */
.skip-link {
  position: absolute;
  left: -9999px;
  top: -9999px;
}
.skip-link:focus {
  left: 16px;
  top: 16px;
  background: var(--surface);
  color: var(--text);
  padding: 8px 12px;
  border-radius: 8px;
  box-shadow: var(--shadow-2);
}

.container {
  max-width: var(--container);
  margin: 0 auto;
  padding: 0 16px;
}

/* 2) Хедер / навигация */
.header {
  position: sticky;
  top: 0;
  z-index: 50;
  background: rgba(18, 22, 25, 0.9);
  backdrop-filter: blur(6px);
  border-bottom: 1px solid rgba(255,255,255,0.06);
}
.header__inner {
  display: flex;
  align-items: center;
  justify-content: space-between;
  min-height: 64px;
}

.logo {
  display: inline-flex;
  align-items: center;
  gap: 10px;
  color: var(--text);
  text-decoration: none;
  font-weight: 700;
```

```
  letter-spacing: 0.2px;
}
.logo__icon { display: inline-block; }
.logo__text { font-size: 1.05rem; }

.nav-toggle {
  display: inline-flex;
  flex-direction: column;
  gap: 4px;
  background: transparent;
  border: 0;
  cursor: pointer;
  padding: 8px;
}
.nav-toggle__bar {
  width: 22px;
  height: 2px;
  background: var(--text);
  border-radius: 2px;
}
.nav {
  position: fixed;
  inset: 64px 0 auto 0;
  background: rgba(18, 22, 25, 0.95);
  transform: translateY(-100%);
  transition: transform 220ms ease;
  border-bottom: 1px solid rgba(255,255,255,0.06);
}
.nav--open { transform: translateY(0); }
.nav__list {
  list-style: none;
  margin: 0;
  padding: 12px 16px 20px;
  display: grid;
  gap: 8px;
}
.nav__link {
  color: var(--text);
  text-decoration: none;
  padding: 10px 12px;
  border-radius: 8px;
}
.nav__link:hover { background: rgba(255,255,255,0.06); }

/* 3) Герой-секция */
.hero {
  padding: 40px 0 24px;
  background:
    radial-gradient(1200px 360px at 20% -100px, rgba(63, 163, 77,
0.25), transparent 60%),
```

```
    radial-gradient(900px 320px at 80% -80px, rgba(23, 92, 45, 0.20),
transparent 60%);
}
.hero__inner {
  display: grid;
  gap: 24px;
}
.hero__title {
  margin: 0;
  font-size: 2rem;
}
.hero__subtitle {
  margin: 10px 0 18px;
  color: var(--muted);
}
.hero__actions { display: flex; gap: 12px; }
.hero__visual { min-height: 160px; position: relative; }
.hero__bg {
  position: absolute;
  inset: 0;
  border-radius: var(--radius);
  background: linear-gradient(180deg, rgba(101, 201, 107, 0.18),
rgba(39, 48, 44, 0.12));
  box-shadow: var(--shadow-1);
}

/* 4) Секции */
.section { padding: 32px 0; }
.section--alt {
  background: rgba(255,255,255,0.03);
  border-top: 1px solid rgba(255,255,255,0.06);
  border-bottom: 1px solid rgba(255,255,255,0.06);
}
.section__title {
  font-size: 1.5rem;
  margin: 0 0 6px 0;
}
.section__subtitle {
  color: var(--muted);
  margin: 0 0 16px 0;
}

/* 5) Кнопки */
.btn {
  display: inline-flex;
  align-items: center;
  justify-content: center;
  gap: 8px;
  padding: 10px 16px;
  border-radius: 10px;
  text-decoration: none;
```

```
  cursor: pointer;
  border: 1px solid transparent;
  transition: background 180ms ease, border-color 180ms ease,
transform 120ms ease;
}
.btn:active { transform: translateY(1px); }
.btn--primary {
  background: var(--primary);
  color: #0b140d;
  font-weight: 600;
}
.btn--primary:hover { background: var(--primary-700); }
.btn--secondary {
  background: var(--surface);
  color: var(--text);
}
.btn--secondary:hover { border-color: rgba(255,255,255,0.12); }
.btn--ghost {
  background: transparent;
  border-color: rgba(255,255,255,0.18);
  color: var(--text);
}
.btn--ghost:hover { border-color: rgba(255,255,255,0.34); }

/* 6) Карточки возможностей */
.features {
  display: grid;
  gap: 12px;
}
.feature-card {
  background: var(--surface);
  border: 1px solid rgba(255,255,255,0.06);
  border-radius: var(--radius);
  padding: 14px;
  box-shadow: var(--shadow-2);
}
.feature-card__icon { font-size: 22px; }
.feature-card__title { margin: 8px 0 4px 0; font-size: 1.05rem; }
.feature-card__text { color: var(--muted); margin: 0; }

/* 7) Демо */
.demo { display: grid; gap: 14px; }
.demo__canvas-wrap {
  position: relative;
  border-radius: var(--radius);
  overflow: hidden;
  border: 1px solid rgba(255,255,255,0.06);
  box-shadow: var(--shadow-1);
  background: #0b0e10;
}
.demo__canvas { display: block; width: 100%; height: auto; }
```

```
.demo__overlay {
  position: absolute;
  inset: auto 12px 12px auto;
  display: inline-flex;
  gap: 8px;
}
.legend {
  display: inline-block;
  font-size: 12px;
  color: #cbd5dc;
  background: rgba(22, 27, 31, 0.8);
  border: 1px solid rgba(255,255,255,0.06);
  padding: 4px 6px;
  border-radius: 8px;
}
.legend--height::before { content: "•"; color: #4fc3f7; margin-right:
6px; }
.legend--type::before { content: "•"; color: #7e57c2; margin-right:
6px; }
.legend--grid::before { content: "•"; color: #90caf9; margin-right:
6px; }

.demo__controls {
  display: grid;
  gap: 12px;
  background: var(--surface);
  border: 1px solid rgba(255,255,255,0.06);
  border-radius: var(--radius);
  padding: 12px;
}
.controls__row { display: flex; flex-wrap: wrap; gap: 8px; }
.control--range input[type="range"] { width: 100%; }
.controls__toggles { gap: 14px; }
.control { display: grid; gap: 6px; font-size: 0.95rem; }
.control--switch { align-items: center; grid-auto-flow: column;
justify-content: start; gap: 8px; }

/* 8) О проекте */
.about {
  display: grid;
  gap: 16px;
}
.about__text p { margin: 0 0 10px; color: var(--text); }
.about__aside {
  background: var(--surface);
  border: 1px solid rgba(255,255,255,0.06);
  border-radius: var(--radius);
  box-shadow: var(--shadow-2);
  padding: 12px;
}
.about__list {
```

```
  margin: 0;
  padding: 0;
  list-style: none;
  display: grid;
  gap: 8px;
  color: var(--muted);
}

/* 9) Форма */
.contact-form {
  display: grid;
  gap: 12px;
  background: var(--surface);
  border: 1px solid rgba(255,255,255,0.06);
  border-radius: var(--radius);
  padding: 12px;
}
.form__row { display: grid; gap: 12px; }
.form__field { display: grid; gap: 6px; }
.form__field--full { grid-column: 1 / -1; }
.form__label { color: var(--muted); font-size: 0.95rem; }
input, textarea {
  width: 100%;
  background: #0e1317;
  color: var(--text);
  border: 1px solid rgba(255,255,255,0.09);
  border-radius: 10px;
  padding: 10px 12px;
  outline: none;
}
input:focus, textarea:focus { border-color: var(--primary); }
.form__error {
  min-height: 18px;
  font-size: 12px;
  color: var(--danger);
}
.form__actions {
  display: flex;
  align-items: center;
  gap: 12px;
}
.form__status { color: var(--muted); font-size: 0.95rem; }

/* 10) Футер */
.footer {
  padding: 20px 0 26px;
  border-top: 1px solid rgba(255,255,255,0.08);
}
.footer__inner {
  display: grid;
  gap: 8px;
```

```
}
.footer__copy { color: var(--muted); margin: 0; }
.footer__nav { display: flex; gap: 12px; flex-wrap: wrap; }
.footer__link { color: var(--text); text-decoration: none; }
.footer__link:hover { text-decoration: underline; }

/* 11) Сетки и адаптивность */
@media (min-width: 640px) {
  .hero { padding: 56px 0 32px; }
  .hero__title { font-size: 2.4rem; }
  .nav { position: static; transform: none; background: transparent;
border: 0; }
  .nav-toggle { display: none; }
  .nav__list { display: flex; gap: 10px; padding: 0; }
  .features { grid-template-columns: repeat(2, minmax(0, 1fr)); }
  .demo { grid-template-columns: 1.2fr 0.8fr; align-items: start; }
  .about { grid-template-columns: 2fr 1fr; }
  .form__row { grid-template-columns: 1fr 1fr; }
  .hero__visual { min-height: 200px; }
}

@media (min-width: 960px) {
  .hero__title { font-size: 3rem; }
  .features { grid-template-columns: repeat(3, minmax(0, 1fr)); }
  .hero__visual { min-height: 240px; }
}

// js/main.js
// Модульный JS: навигация, демо-трава, форма обратной связи

document.addEventListener('DOMContentLoaded', () => {
  Nav.init();
  GrassDemo.init();
  ContactForm.init();
});

/* Навигация (мобильное меню) */
const Nav = (() => {
  let toggleBtn, nav, links;

  function init() {
    toggleBtn = document.querySelector('.nav-toggle');
    nav = document.getElementById('primary-nav');
    links = nav ? nav.querySelectorAll('.nav__link') : [];

    if (toggleBtn && nav) {
      toggleBtn.addEventListener('click', () => {
        const open = nav.classList.toggle('nav--open');
        toggleBtn.setAttribute('aria-expanded', open ? 'true' :
'false');
      });
```

```
    }
    // Закрывать меню по клику на ссылку (мобильный UX)
    links.forEach(link => {
      link.addEventListener('click', () => {
        if (nav.classList.contains('nav--open')) {
          nav.classList.remove('nav--open');
          toggleBtn.setAttribute('aria-expanded', 'false');
        }
      });
    });
  }

  return { init };
})();

/* Демо визуализации травы */
const GrassDemo = (() => {
  const cfg = {
    chunksX: 10,
    chunksY: 6,
    bladesPerChunk: 120, // изменяется слайдером
    swaySpeedMin: 0.6,
    swaySpeedMax: 1.4,
    swayAmpMin: 2,
    swayAmpMax: 6,
    bladeLenMin: 28,
    bladeLenMax: 70,
    bladeThick: 1.1,
    groundBase: 0.62, // относительная базовая высота "земли"
    heightAmp: 0.12,  // амплитуда карты высот
    typeInfluence: 0.35, // влияет на длину/цвет
    bgTop: '#0b0e10',
    bgBottom: '#0f1315'
  };

  let canvas, ctx, running = true;
  let W = 0, H = 0, DPR = 1;

  // Карты (процедурные)
  let heightmap = [];   // 1D: длина W, значения 0..1
  let typeMap = [];     // 2D по чанкам: значения 0..2 (тип травы)
  let chunks = [];      // массив чанков, каждый содержит blades[]

  // Оверлеи
  let showHeight = false, showType = false, showGrid = false;

  // UI элементы
  let btnPlay, btnPause, btnRegenerate, rangeQuality, qualityValue;
  let toggleHeight, toggleType, toggleGrid;

  function init() {
```

```
    canvas = document.getElementById('grass-canvas');
    if (!canvas) return;
    ctx = canvas.getContext('2d');

    // UI
    btnPlay = document.getElementById('btn-play');
    btnPause = document.getElementById('btn-pause');
    btnRegenerate = document.getElementById('btn-regenerate');
    rangeQuality = document.getElementById('quality');
    qualityValue = document.getElementById('quality-value');

    toggleHeight = document.getElementById('toggle-height');
    toggleType = document.getElementById('toggle-type');
    toggleGrid = document.getElementById('toggle-grid');

    bindUI();

    resize();
    generateMaps();
    createChunks();
    loop();

    window.addEventListener('resize', handleResizeThrottled, {
passive: true });
  }

  function bindUI() {
    btnPlay?.addEventListener('click', () => { running = true; });
    btnPause?.addEventListener('click', () => { running = false;
drawFrame(0); });
    btnRegenerate?.addEventListener('click', () => {
      generateMaps(true);
      createChunks();
    });

    rangeQuality?.addEventListener('input', (e) => {
      const val = parseInt(e.target.value, 10);
      cfg.bladesPerChunk = val;
      qualityValue.textContent = String(val);
      // Пересоздаём траву при смене качества
      createChunks();
    });

    toggleHeight?.addEventListener('change', (e) => { showHeight =
e.target.checked; });
    toggleType?.addEventListener('change', (e) => { showType =
e.target.checked; });
    toggleGrid?.addEventListener('change', (e) => { showGrid =
e.target.checked; });
  }
```

```
  // Троттлинг resize
  let resizeTimeout = null;
  function handleResizeThrottled() {
    if (resizeTimeout) return;
    resizeTimeout = setTimeout(() => {
      resize();
      createChunks(); // пересчёт разметки
      resizeTimeout = null;
    }, 150);
  }

  function resize() {
    DPR = Math.min(2, window.devicePixelRatio || 1);
    const wrap = canvas.parentElement;
    const w = wrap.clientWidth;
    const h = Math.max(240, Math.round(w * 0.56)); // соотношение
~16:9 с минимальной высотой для мобильных

    canvas.width = Math.round(w * DPR);
    canvas.height = Math.round(h * DPR);
    canvas.style.width = w + 'px';
    canvas.style.height = h + 'px';

    W = canvas.width;
    H = canvas.height;
  }

  function generateMaps(regenerateSeed = false) {
    // Высотная карта (value noise)
    const cols = Math.max(128, Math.floor(W / 8));
    const base = cfg.groundBase;
    const amp = cfg.heightAmp;

    const seed = regenerateSeed ? Math.random() * 10000 : 42.4242;
    heightmap = new Array(W);

    // Грубые точки
    const points = new Array(cols).fill(0).map((_, i) => smoothRand(i
* 19.19 + seed));
    // Сглаживание кубическим полиномом (Catmull-Rom-like)
    for (let x = 0; x < W; x++) {
      const t = x / W;
      const fx = t * (cols - 1);
      const i = Math.floor(fx);
      const frac = fx - i;

      const p0 = points[Math.max(0, i - 1)];
      const p1 = points[i];
      const p2 = points[Math.min(cols - 1, i + 1)];
      const p3 = points[Math.min(cols - 1, i + 2)];
```

```
      const v = catmullRom(p0, p1, p2, p3, frac);
      heightmap[x] = Math.min(1, Math.max(0, base + (v - 0.5) * 2 *
amp));
    }

    // Карта типов травы по чанкам (0 – редкая, 1 – обычная, 2 –
густая)
    typeMap = [];
    for (let cy = 0; cy < cfg.chunksY; cy++) {
      const row = [];
      for (let cx = 0; cx < cfg.chunksX; cx++) {
        const r = smoothRand(seed + cx * 7.31 + cy * 13.7);
        const type = r < 0.33 ? 0 : r < 0.66 ? 1 : 2;
        row.push(type);
      }
      typeMap.push(row);
    }
  }

  function createChunks() {
    chunks = [];
    const cw = Math.floor(W / cfg.chunksX);
    const ch = Math.floor(H / cfg.chunksY);
    for (let cy = 0; cy < cfg.chunksY; cy++) {
      for (let cx = 0; cx < cfg.chunksX; cx++) {
        const x0 = cx * cw;
        const y0 = cy * ch;
        const type = typeMap[cy][cx];

        const blades = [];
        const bladesCount = cfg.bladesPerChunk * (1 + type * 0.35); //
больше для густых типов
        for (let i = 0; i < bladesCount; i++) {
          const bx = x0 + Math.random() * cw;
          // Учитываем линию "земли" по высоте
          const groundY = Math.floor(heightmap[Math.min(W - 1,
Math.max(0, Math.round(bx))) ] * H);
          const by = Math.max(groundY - 4, y0 + Math.random() * (ch -
8));

          const len = lerp(cfg.bladeLenMin, cfg.bladeLenMax, 0.45 +
Math.random() * 0.5);
          const swayAmp = lerp(cfg.swayAmpMin, cfg.swayAmpMax,
Math.random());
          const swaySpeed = lerp(cfg.swaySpeedMin, cfg.swaySpeedMax,
Math.random());
          const phase = Math.random() * Math.PI * 2;
          const thickness = cfg.bladeThick;
          const tint = typeToColor(type, by / H);
```

```
          blades.push({ x: bx, y: by, len, swayAmp, swaySpeed, phase,
thickness, tint });
        }

        chunks.push({ x0, y0, w: cw, h: ch, type, blades });
      }
    }
  }

  function loop() {
    let last = performance.now();
    const tick = (t) => {
      if (running) {
        const dt = Math.min(0.033, (t - last) / 1000); // ограничение
delta
        last = t;
        drawFrame(t / 1000, dt);
        requestAnimationFrame(tick);
      } else {
        requestAnimationFrame(tick); // продолжаем, чтобы реагировать
на UI; можно отключить для энергосбережения
      }
    };
    requestAnimationFrame(tick);
  }

  function drawFrame(time) {
    // фон
    drawBackground();

    // земля/линия горизонта по высоте
    drawGroundLine();

    // травинки
    ctx.save();
    ctx.lineCap = 'round';
    chunks.forEach(chunk => {
      chunk.blades.forEach(b => {
        const sway = Math.sin(time * b.swaySpeed + b.phase) *
b.swayAmp;
        const x0 = b.x * DPR;
        const y0 = b.y * DPR;
        const x1 = (b.x + sway) * DPR;
        const y1 = (b.y - b.len) * DPR;

        // Кривизна (задание средней точки)
        const cx = (x0 + x1) / 2 + sway * DPR * 0.4;
        const cy = (y0 + y1) / 2 - b.len * DPR * 0.1;

        ctx.strokeStyle = b.tint;
        ctx.lineWidth = b.thickness * DPR;
```

```
      ctx.beginPath();
      ctx.moveTo(x0, y0);
      ctx.quadraticCurveTo(cx, cy, x1, y1);
      ctx.stroke();
    });
  });
  ctx.restore();

  // Оверлеи
  if (showGrid) drawChunkGrid();
  if (showType) drawTypeMapOverlay();
  if (showHeight) drawHeightOverlay();
}

/* Рисование частей */
function drawBackground() {
  const grd = ctx.createLinearGradient(0, 0, 0, H);
  grd.addColorStop(0, cfg.bgTop);
  grd.addColorStop(1, cfg.bgBottom);
  ctx.fillStyle = grd;
  ctx.fillRect(0, 0, W, H);
}

function drawGroundLine() {
  ctx.save();
  ctx.strokeStyle = 'rgba(130, 157, 135, 0.35)';
  ctx.lineWidth = 2 * DPR;
  ctx.beginPath();
  for (let x = 0; x < W; x += 4 * DPR) {
    const y = Math.floor(heightmap[x] * H);
    if (x === 0) ctx.moveTo(x, y);
    else ctx.lineTo(x, y);
  }
  ctx.stroke();
  ctx.restore();
}

function drawChunkGrid() {
  ctx.save();
  ctx.strokeStyle = 'rgba(144, 202, 249, 0.35)';
  ctx.lineWidth = 1 * DPR;
  const cw = Math.floor(W / cfg.chunksX);
  const ch = Math.floor(H / cfg.chunksY);
  // Вертикальные
  for (let cx = 1; cx < cfg.chunksX; cx++) {
    const x = cx * cw;
    ctx.beginPath();
    ctx.moveTo(x, 0);
    ctx.lineTo(x, H);
    ctx.stroke();
  }
```

```
  // Горизонтальные
  for (let cy = 1; cy < cfg.chunksY; cy++) {
    const y = cy * ch;
    ctx.beginPath();
    ctx.moveTo(0, y);
    ctx.lineTo(W, y);
    ctx.stroke();
  }
  ctx.restore();
}

function drawTypeMapOverlay() {
  const cw = Math.floor(W / cfg.chunksX);
  const ch = Math.floor(H / cfg.chunksY);
  ctx.save();
  for (let cy = 0; cy < cfg.chunksY; cy++) {
    for (let cx = 0; cx < cfg.chunksX; cx++) {
      const type = typeMap[cy][cx];
      ctx.fillStyle =
        type === 0 ? 'rgba(126, 87, 194, 0.12)' :
        type === 1 ? 'rgba(126, 87, 194, 0.18)' :
                     'rgba(126, 87, 194, 0.26)';
      ctx.fillRect(cx * cw, cy * ch, cw, ch);
    }
  }
  ctx.restore();
}

function drawHeightOverlay() {
  ctx.save();
  for (let x = 0; x < W; x += 2 * DPR) {
    const y = Math.floor(heightmap[x] * H);
    ctx.strokeStyle = 'rgba(79, 195, 247, 0.28)';
    ctx.beginPath();
    ctx.moveTo(x, y);
    ctx.lineTo(x, H);
    ctx.stroke();
  }
  ctx.restore();
}

/* Вспомогательные функции генерации */
function smoothRand(seed) {
  const r = Math.sin(seed * 12.9898) * 43758.5453;
  return (r - Math.floor(r));
}
function lerp(a, b, t) { return a + (b - a) * t; }
function catmullRom(p0, p1, p2, p3, t) {
  const t2 = t * t, t3 = t2 * t;
  return 0.5 * ( (2 * p1) +
    (-p0 + p2) * t +
```

```
      (2*p0 - 5*p1 + 4*p2 - p3) * t2 +
      (-p0 + 3*p1 - 3*p2 + p3) * t3 );
  }
  function typeToColor(type, yNorm) {
    // Цвета зависят от типа и высоты: чуть темнее в низинах, светлее на
вершинах
    const base =
      type === 0 ? { r: 60, g: 130, b: 72 } :
      type === 1 ? { r: 70, g: 155, b: 86 } :
                   { r: 84, g: 178, b: 102 };
    const shade = lerp(-18, 12, yNorm); // корректировка яркости
    const r = clamp(base.r + shade, 45, 255);
    const g = clamp(base.g + shade, 45, 255);
    const b = clamp(base.b + shade, 45, 255);
    return `rgba(${r}, ${g}, ${b}, 0.9)`;
  }
  function clamp(v, min, max) { return Math.max(min, Math.min(max,
v)); }

  return { init };
})();

/* Форма обратной связи: валидация и фейковая отправка */
const ContactForm = (() => {
  let form, status;

  function init() {
    form = document.getElementById('contact-form');
    status = document.getElementById('form-status');
    if (!form) return;

    form.addEventListener('submit', async (e) => {
      e.preventDefault();
      clearErrors();

      const data = new FormData(form);
      const name = data.get('name')?.toString().trim();
      const email = data.get('email')?.toString().trim();
      const message = data.get('message')?.toString().trim();

      let valid = true;
      if (!name || name.length < 2) {
        showError('name', 'Введите имя (минимум 2 символа).');
        valid = false;
      }
      if (!email || !/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)) {
        showError('email', 'Введите корректный email.');
        valid = false;
      }
      if (!message || message.length < 10) {
```

```
        showError('message', 'Сообщение должно содержать минимум 10
символов.');
        valid = false;
      }

      if (!valid) return;

      status.textContent = 'Отправка...';
      // Имитация отправки: здесь можно интегрировать реальный API
      await new Promise(r => setTimeout(r, 900));
      status.textContent = 'Спасибо! Мы свяжемся с вами в ближайшее
время.';
      form.reset();
    });
  }

  function showError(fieldId, text) {
    const field = document.getElementById(fieldId);
    const errorEl =
field?.parentElement?.querySelector('.form__error');
    if (errorEl) errorEl.textContent = text;
  }
  function clearErrors() {
    form.querySelectorAll('.form__error').forEach(el => el.textContent
= '');
    status.textContent = '';
  }

  return { init };
})();
```

Оценка результатов по критериям:

Критерии да/нет:

Критерий	Ответ
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

Критерии по шкале от 1 до 5:

Критерий	Оценка
Соответствие дизайна и содержания исходному описанию проекта	5
Техническое качество клиентского кода	5
Общую пригодность лендинга для практического применения	5

ВЫВОДЫ

В результате выполнения лабораторной работы был сгенерирован лендинг, полностью удовлетворяющий все запросы. Сгенерированный ИИ вариант включает все запрошенные блоки, корректно отображается в современных браузерах, адаптивен для мобильных устройств, соответствует принципам модульности и читаемости кода и обеспечивает кроссплатформенную совместимость. Проверка по всем функциональным и техническим критериям дала положительные результаты, а экспертная оценка качества показала максимально возможные баллы.

Временные затраты оказались минимальными для достижения требуемого результата, что подтверждает эффективность применения ИИ-инструментов в задачах быстрого прототипирования и первичной вёрстки.

Таким образом, поставленная цель достигнута: ИИ продемонстрировал способность генерировать полноценный, адаптивный и технологически корректный лендинг, пригодный для дальнейшего использования и доработки.