Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

(Университет ИТМО)

Факультет программной инженерии и компьютерной техники

Образовательная программа 09.04.04 ПРОГРАММНАЯ ИНЖЕНЕРИЯ

Направление подготовки (специальность) 09.04.04 Программная инженерия

О Т Ч Е Т
о научно-исследовательской работе

Тема задания:

«Проектирование онлайн тренажера
для обучения пользователей скринридеров работе с веб-интерфейсом»

Обучающийся:
Довыденков Владимир Николаевич, группа P4150

Согласовано:
Руководитель практики от университета:
Муромцев Дмитрий Ильич, доцент факультета программной инженерии и компьютерной техники, к.т.н.

Практика пройдена с оценкой: ___

Подписи членов комиссии:

_____ Ф.И.О.
(подпись)

_____ Ф.И.О.
(подпись)

_____ Ф.И.О.
(подпись)

Дата_____

Санкт-Петербург 2025

СОДЕРЖАНИЕ

Характеристика организации

Университет ИТМО (Федеральное государственное автономное образовательное учреждение высшего образования "Национальный исследовательский университет ИТМО") — один из ведущих научно-образовательных центров России, обладающий устойчивой академической репутацией и высоким уровнем международного признания. Университет был основан в 1900 году в Санкт-Петербурге и с 2009 года носит статус национального исследовательского университета.

ИТМО реализует широкий спектр образовательных программ бакалавриата, магистратуры, аспирантуры и дополнительного профессионального образования в таких областях, как информационные технологии, компьютерные науки, искусственный интеллект, фотоника, робототехника, биотехнологии, химия, урбанистика и цифровая культура. Университет уделяет особое внимание междисциплинарным подходам и интеграции науки, образования и предпринимательства.

Организационная структура включает Наблюдательный совет, Ученый совет и Ректорат. Основу образовательной и научной деятельности составляют мегафакультеты: компьютерных технологий и управления, физикотехнический, трансляционных информационных технологий, наук о жизни, а также факультет технологического менеджмента и инноваций, Высшая инженерно-техническая школа, Высшая школа цифровой культуры и Военный учебный центр. Каждый из них объединяет современные лаборатории, исследовательские центры и образовательные программы.

ИТМО активно развивает административную инфраструктуру, направленную на поддержку ключевых направлений деятельности: стратегическое развитие, сопровождение образовательной и научной работы, реализацию проектов, международное сотрудничество, информационную и правовую поддержку, кадровую и финансово-экономическую деятельность. Университет формирует инновационную экосистему, ориентированную на развитие стартапов, партнерство с бизнесом и трансфер технологий.

Университет стабильно занимает высокие позиции в международных рейтингах. В 2023 году ИТМО занял 79-е место в мире по направлению "Computer Science" и вошел в диапазон 51–70 лучших университетов мира в области "Data Science & Artificial Intelligence". Он также входит в число лидеров по качеству приёма и трудоустройству выпускников.

ИТМО находится на стадии зрелости и демонстрирует устойчивое развитие, сочетая фундаментальные академические традиции с новаторским подходом к образованию, науке и взаимодействию с обществом и индустрией. Университет активно инвестирует в будущее, развивая новый научно-образовательный центр — кампус "Хайпарк".

ВВЕДЕНИЕ.

Цель. Разработать проектную документацию программного обеспечения для обучения пользователей скринридеров работе с веб-интерфейсом.

Задачи.

1. Проанализировать существующие образовательные нормативы и специализированные обучающие инструменты.
2. Изучить современные стандарты доступности веб-интерфейсов.
3. С помощью анкетирования и проведения интервью с экспертным сообществом определить функциональные и технические требования к программному обеспечению.
4. Выявить оптимальные технологические решения для создания эффективной обучающей среды .
5. Разработать основные элементы проектной документации: архитектуру системы, сценарии использования и взаимодействия пользователя с интерфейсом.

Актуальность.

Графический пользовательский интерфейс (GUI) единообразен в своих ключевых инструментах, и взаимодействие пользователя с ним на разных платформах принципиально не отличается. Визуальное отображение веб-интерфейса не имеет существенных отличий с визуальным отображением интерфейса десктопных (нативных) приложений. Поэтому пользователь, знакомый с принципами взаимодействия с GUI, не испытывает затруднений при переходе от интерфейса нативных приложений к веб-интерфейсу и обратно. Однако представление веб-интерфейса программами экранного доступа существенно отличается от отображения этеми же программами нативных графических интерфейсов. Инструменты исследования и перемещения, предоставляемые скринридером для элементов веб-интерфейса значительно отличаются от таковых для нативных интерфейсов. Основные отличия веб-интерфейса, представляемого программой экранного доступа состоят в следующем:

- все элементы веб-интерфейса последовательно, построчно отображаются в едином фокусном пространстве, похожем на текстовый документ;
- взаимодействие с диалоговыми элементами управления (строкой редактора, выпадающим списком) возможно только в дополнительном режиме;
- перемещение по элементам или группам элементов веб-интерфейса буквенно-цифровыми клавишами. Перечисленные особенности делают веб-интерфейс значимо менее понятным и удобным для начинающих пользователей программ экранного доступа. Поскольку различные элементы управления отличаются сложностью в понимании и манипулировании с ними, представляется целесообразным разработать программное средство, поэтапно

представляющее элементы управления веб-интерфейсом и ориентированное на пользователей программ экранного доступа.

Исследование актуальности

Для исследования актуальности специального программного решения, был проведён опрос среди целевой группы незрячих и слабовидящих пользователей программ экранного доступа. Сопроводительный текст и формулировки вопросов были намеренно упрощены и выдержены в неформальном стиле, для привлечения максимально широкого круга заинтересованных пользователей.

В опросе приняло участие 17 человек, из которых 14 (82%) указали, , что у них есть группы учеников. Для исследования вопроса о дополнительной сложности веб-интерфейса по сравнению с десктопным (нативным) интерфейсом был задан вопрос: "**Что удобнее для ваших учеников: веб-интерфейс или интерфейс обычных программ?**" В таблице №1 приведены результаты опроса:

Таблица №1

Оценка	**Количество голосов**	**Процент от общего числа**
Обычный десктопный интерфейс удобнее	11	64%
Веб-интерфейс удобнее	2	12%
Нет вообще никакой разницы	4	24%
Всего голосов	**17**	**100%**

Таким образом, 64% опрошенных считают десктопный (нативный) интерфейс более удобным, по сравнению с веб-интерфейсом.

Основным сущностным вопросом исследования был вопрос: "**нужен ли специальный сайт с пояснениями для изучения веб-интерфейса?**" Результаты представлены ниже в таблице №2.

Таблица №2

Оценка	**Количество ответов**	**Процент от общего числа**
Суперполезная штука, давно нужно (5/5)	10	59%
Хорошая идея, точно пригодится (4/5)	3	18%
Да, может быть полезным (3/5)	3	18%
Смысла нет, но пусть будет (2/5)	0	0%

Точно не нужен (1/1)	1	5%
Всего ответов	**17**	**100%**

Таким образом, 95% опрошенных так или иначе согласны с необходимостью специального программного средства для изучения веб-интерфейса.

Со строго социологической точки зрения опрос не является реливантным, и не отражает реальное распределение мнений сообщества о необходимости разработки дополнительного инструмента для изучения веб-интерфейса. Однако он демонстрирует потребность в этом инструменте у части этого сообщества. Что представляется достаточным для обоснования актуальности разработки.

1. Сбор и анализ требований

1.1. Опрос целевой аудитории

1.1.1. Формулирование целей

На начальном этапе были определены ключевые параметры и целевая аудитория: незрячие и слабовидящие пользователи, преподаватели и эксперты по доступности. Были сформулированы основные исследовательские вопросы, включая выявление приоритетных функций веб-тренажёра, наиболее востребованных сценариев использования, а также технических и организационных требований, таких как скорость работы, совместимость и объём собираемой статистики.

1.1.2. Выбор платформы и проектирование структуры анкеты

Сравнительный анализ доступных онлайн-сервисов опросов показал, что наибольшую совместимость со средствами чтения экрана обеспечивает Google Forms. На его основе была спроектирована структура анкеты, разделённая на пять логических блоков:

1. Профиль респондента
2. Функциональные требования
3. Нефункциональные требования
4. Требования преподавателей

5. Общие предложения и комментарии Каждому блоку предшествовало краткое текстовое описание его цели и инструкции по навигации для пользователей скринридеров .

1.1.3. Разработка и валидация вопросов

Сокращение и упорядочивание формулировок

В процессе подготовки текст вопросов был разделён на короткие предложения. Контекстные пояснения были вынесены в описания разделов, что позволило избежать избыточных сведений внутри вариантов ответов.

Выбор типов вопросов

- **Радиокнопки** применялись для сбора однозначных ответов (роль, опыт, требуемая совместимость).
- **Чекбоксы** использовались в разделах с множественным выбором (список поддерживаемых экранных ридеров, инструменты контроля).
- **Шкалы Лайкерта** (1–5) внедрялись в блоках функциональных и нефункциональных требований для количественной оценки.
- **Текстовые поля** предусматривались для сбора дополнительных идей и комментариев.

Для повышения заметности открытых полей была добавлена уточняющая подпись «Если вариант не представлен, укажите его здесь» непосредственно под чекбоксами.

1.1.4. Пилотное тестирование и корректировка

Проведение пилотного опроса

Пилотный запуск был организован среди трёх незрячих пользователей и одного эксперта по доступности. В ходе тестирования были выявлены следующие проблемы:

- Ошибки в типе ответов: список с единственным выбором, вместо списка со множественным выбором.
- Перегруженность длинных вариантов в списках «инструментов контроля».
- В некоторых случаях сложные, непонятные формулировки.

Внесённые исправления

- Исправлены ошибки с типами ответов.
- Варианты были разделены на две строки для удобства восприятия.
- Вопросы упрощены и переформулированы.

Полный текст финальной версии Анкеты смотрите в Приложении.

1.1.5. Проверка доступности

Комплексная проверка была выполнена с использованием программ экранного доступа NVDA, Jaws и VoiceOver. В ходе проверки подтверждена корректность озвучивания состояний чекбоксов и радиокнопок, а также последовательность чтения текста шкальных вопросов.

1.1.6. Сбор выборки респондентов

Для увеличения охвата целевой аудитории анкета была размещена в специализированных сообществах (Telegram-канал, электронная почтовая рассылка). Стимулирующий текстовый блок подчёркивал ценность вклада каждого участника, а также предлагал доступ к итоговым материалам и участие в дальнейших тестированиях. В итоге было получено 26 завершённых ответов .

1.1.7. Обработка и анализ данных

Экспорт и предварительная очистка

Ответы были экспортированы в Google Sheets. Проведена проверка на дубликаты. Текстовые комментарии были нормализованы (устранены лишние пробелы, проверена целостность e-mail адресов).

Количественный анализ

Частоты ответов по закрытым вопросам и средние оценки по шкалам рассчитаны с помощью встроенных инструментов Google Sheets. Многовариантные ответы (чекбоксы) сгруппированы в таблицы, а текстовые комментарии классифицированы вручную по тематическим категориям.

1.1.8. Основные выводы

1. **Навигация и формы** получили наивысшие средние оценки (~4.7 из 5).
2. **Запрос на примеры** графической разметки и корректного использования alt-текста.
3. **Необходимость расширенной панели преподавателя**, включая гибкий выбор метрик и систему уведомлений.
4. **Требование поддержки** ключевых скринридеров (NVDA, JAWS, TalkBack, VoiceOver) и основных браузеров.

5. **Идеи развития**: офлайн-режим, многоуровневые аккаунты, доступная CAPTCHA.

На основе полученных данных запланировано добавление модуля с примерами alt-текста, реализация интерфейса преподавателя с настраиваемыми метриками и предварительное тестирование офлайн-режима.

1.2. Анализ существующих программных решений

1.2.1. Обзор существующих решений в литературе

В ходе обзорного поиска по технической литературе не были обнаружены концептуально или функционально сходные решения предлагаемому тренажёру. Анализ публикаций и отраслевых отчётов не выявил ни отечественных, ни зарубежных материалов, описывающих интерактивный инструмент обучения веб-интерфейсу для пользователей скринридеров.

1.2.2. Поиск по зарубежным ресурсам

Для уточнения наличия готовых продуктов был выполнен поиск на ведущих зарубежных платформах и в блогах, сочетающих темы доступности, веб-технологий и образования. Были выявлены следующие инструменты:

1. **WebAIM Screen Reader Simulation** Онлайн-симулятор скринридера, позволяющий разработчикам и дизайнерам оценить, как их страницы воспринимаются средствами чтения экрана. URL: https://webaim.org/simulations/screenreader
2. **TPGi JAWS Inspect** Инструмент для анализа доступности, демонстрирующий, как скринридер JAWS произносит элементы веб-страницы. URL: https://www.tpgi.com/arc-platform/jaws-inspect/
3. **Top Web Accessibility Tools for Developers** Обзорный материал, перечисляющий утилиты для выявления проблем доступности в веб-проектах. URL: https://blog.pixelfreestudio.com/top-web-accessibility-tools-for-developers/
4. **Inclusive Design 24** Серия онлайн-курсов и вебинаров, посвящённых демонстрации работы скринридеров с различными элементами интерфейса. URL: https://inclusivedesign24.org/2025/
5. **Microsoft Accessibility Insights** Инструмент для тестирования доступности, частично моделирующий восприятие элементов страницы скринридером. URL: https://accessibilityinsights.io/

1.2.3. Выводы

Ни один из перечисленных ресурсов не представляет собой интерактивный тренажёр, обучающий навыкам работы со скринридером в контексте веб-интерфейса. Таким образом, среди существующих отечественных и зарубежных предложений не было обнаружено аналогов по сочетанию функциональности и формата предлагаемого решения.

1.3. Анализ нормативных документов

1.3.1. Международный стандарт WCAG

Описание: Основным международным нормативным документом, определяющим требования к доступности цифрового контента, включая веб-страницы, является **Web Content Accessibility Guidelines (WCAG)**. Стандарт разрабатывается Рабочей группой по доступности W3C (World Wide Web Consortium). **Цель:** Обеспечить максимально широкий доступ к веб-контенту для пользователей с различными формами инвалидности. **Структура:** Рекомендации сгруппированы по четырём принципам:

- **Perceivable (Воспринимаемость)**
- **Operable (Управляемость)**
- **Understandable (Понятность)**
- **Robust (Надёжность)**

Уровни соответствия:

- **A** — минимальные требования;
- **AA** — средний, рекомендованный уровень;
- **AAA** — максимальный уровень;

Текущая версия: WCAG 2.2 (октябрь 2023). В ней уточнены критерии для мобильных устройств, сенсорного управления и динамического контента. **Ресурс:** https://www.w3.org/WAI/standards-guidelines/wcag/

1.3.2. Национальный стандарт РФ ГОСТ Р 52872-2019

Наименование: «ГОСТ Р 52872-2019. Интернет-ресурсы и другая информация, представленная в электронно-цифровой форме. Приложения для стационарных и мобильных устройств, иные пользовательские интерфейсы. Требования доступности для людей с инвалидностью и других лиц с ограничениями жизнедеятельности». **Статус:** Национальный стандарт Российской Федерации. **Утверждение и введение в действие:** Приказ Федерального

агентства по техническому регулированию и метрологии от 29 августа 2019 г. № 589-ст. **Издатель:** Стандартинформ, Москва, 2019. **Ресурс:** https://www.frcds.ru/wp-content/uploads/2020/05/GOST52872-2019-min.pdf

1.3.3. Рекомендации по применению в проекте

- Для разработки веб-тренажёра рекомендуется ориентироваться на **уровень AAA** WCAG 2.2, обеспечивая максимальную доступность всех элементов интерфейса.
- При этом следует учитывать требования национального стандарта ГОСТ Р 52872-2019, реализуя обязательные критерии соответствия для пользователей с ограничениями жизнедеятельности.
- Все тестовые страницы тренажёра должны содержать явные ARIA-атрибуты (aria-label, aria-describedby, aria-live), текстовые альтернативы для графики и корректно структурированные заголовки, списки и формы.

Таким образом, использование комбинации международных и национальных нормативных требований обеспечивает полное соответствие предлагаемого решения стандартам доступности.

1.4. Базовые требования к программному обеспечению

1.4.1. Формирование функциональных требований

Основано на результатах опроса 26 респондентов (незрячие пользователи, преподаватели, эксперты по доступности).

- **Приоритетные модули тренажёра:**
  - **Навигация по заголовкам, ссылкам, кнопкам** — средняя оценка 4,73;
  - **Работа с формами** (текстовые поля, флажки, переключатели) — 4,73;
  - **Сложные элементы** (таблицы, списки, модальные окна) — 4,62;
  - **Встроенные подсказки и справочная информация** — 4,19.
- **Дополнительные запросы** зафиксированы в открытых комментариях (1 ответ):
  - демонстрация корректной и некорректной разметки графических элементов (alt-текст vs. его отсутствие);
  - классическая форма регистрации с альтернативной аудиокапчей или математической задачей.

Проблемы при разработке и их решения:

1. **Пересечения в темах:** в первоначальной версии навигация и формы во многом дублировали друг друга. Было решено чётко разделить эти блоки и вынести пояснения по работе со скринридером в отдельные инструкции.

2. **Сохранение сессии:** 19 из 26 участников (≈ 73 %) отметили важность возобновления обучения с последнего этапа. Для этого был внедрён механизм идентификации по cookies, URL-идентификатору или IP и автоматическое восстановление состояния.

1.4.2. Проектирование навигационной логики и структуры разделов

Подход: материал подаётся «ступенчато» — каждая страница оформляется как самостоятельный этап со строгой инструкцией и единообразными контролями.

- **Базовый раздел** (Basic) содержит четыре темы: навигация, формы, сложные элементы, подсказки.
- **Опциональные разделы:** Multimedia (аудио и видео) и Math (формулы).

Техническое решение проблем:

- Для устранения пропусков контролей скринридером все интерактивные элементы снабжены атрибутами ARIA (aria-label, aria-describedby, aria-live).
- Шаблоны страниц унифицированы: одинаковые заголовки, кнопки «Далее» и индикаторы прогресса.

1.4.3. Реализация механизма отслеживания прогресса

Обоснование: 9 из 15 преподавателей (60 %) назвали «процент успешно пройденных этапов» ключевой метрикой, а 11 (73 %) — «количество ошибок».

Модель «веса» этапов:

- Каждому этапу присвоена сложностью обусловленная оценка (10–50 баллов).
- Фиксация в БД: идентификатор этапа, время старта и завершения.
- Расчёт прогресса: сумма баллов пройденных этапов делится на сумму всех баллов и переводится в проценты.

Устранение рассинхронизации:

- Для исключения двойного учёта при многократных заходах реализована запись единственной активной сессии с блок-сессией в БД.

1.4.4. Проектирование функционала преподавателя

Исходя из отзывов 15 преподавателей:

- Наиболее важные метрики:

1. Количество ошибок (11 упоминаний)
2. Процент успешных этапов (9)
3. Время прохождения (6)

- Функции управления: просмотр статистики, комментарии к этапам, создание пользовательских упражнений, настройка порядка уроков.

Решения:

1. **Идентификация через приглашения:** вместо небезопасного поиска по email реализована система уникальных ссылок-приглашений.
2. **Хранение подсказок:** комментарии преподавателя сохраняются в отдельной таблице и выводятся на соответствующей странице для ученика.
3. **Гибкая панель метрик:** преподаватель может включать/выключать отображение ошибок, процента, времени и обращений к подсказкам.
4. **Каналы уведомлений:**

- Внутри тренажёра (9 упоминаний)
- По e-mail (7)
- В Telegram (5)
- Поддержка дайджестов: 8 «да» / 7 «нет».

1.4.5. Формализация нефункциональных требований

Основано на ответах 26 респондентов:

- **Скорость отклика:**
  - < 1 с: 13;
  - 1–3 с: 9;
  - 3 с: 2. → Установлено требование: **< 3 с**.
- **Доступность сервиса:**
  - «Всегда доступен» — 14;
  - «Можно отложить» — 7;
  - «Не важно» — 3. → Предел непрерывной недоступности ≤ 2 ч, суммарный простой ≤ 5 ч/мес.

- **Совместимость:**
  - **Скринридеры:** NVDA (26), JAWS (25), TalkBack (22), VoiceOver iOS (18), VoiceOver macOS (14), ORCA (12).
  - **Браузеры:** Chrome (23), Yandex (21), Firefox (18), Edge (14), Safari (12).
- **Безопасность:** риск оценён как незначительный (чувствительные данные не хранятся).
- **Резервирование:**
  - БД — ежечасно (с перезаписью предыдущей копии);
  - Полный бэкап сервиса — раз в сутки.
- **Локализация:** предусмотрена возможность подключения новых языковых пакетов.

2. Сценарии использования и ключевые функции системы

2.1. Сценарий навигации по заголовкам, ссылкам и кнопкам

Основания для разработки На основании результатов опроса 26 респондентов (незрячие пользователи, преподаватели, эксперты по доступности) было установлено, что навигация по заголовкам, ссылкам и кнопкам имеет средний балл важности **4,73** из 5. Этот высокий рейтинг позволил обосновать включение отдельного сценария, посвящённого изучению базовых приёмов управления веб-страницей при помощи скринридера.

Выделение ключевых элементов В этом разделе были выделены следующие три этапа:

1. **Ссылки** — основа гипертекстовой навигации между разделами.
2. **Кнопки** — элементы управления для запуска действий (воспроизведение звука, переход к следующему этапу).
3. **Заголовки** — как средство быстрой ориентировки в иерархии страницы.

Построение этапов обучения Каждый этап был спроектирован по единому шаблону:

- **Объяснение** функциональной роли элемента;
- **Пример** его визуального и семантического представления;
- **Задание** с заданным критерием успешного выполнения.

При этом для комфортного восприятия текст «Объяснения» был сведён к одному–двум предложениям, а «Пример» оформлен в виде конкретного фрагмента интерфейса (абзаца с ссылкой или кнопкой с коротким звуковым эффектом).

Работа с ARIA-атрибутами Для обеспечения однозначного озвучивания всеми основными скринридерами (NVDA, JAWS, VoiceOver, TalkBack) каждому интерактивному элементу необходимо добавлять ARIA-метки:

- role="heading" и aria-level для заголовков;
- role="link" и aria-label для ссылок;
- role="button" и aria-pressed (при необходимости) для кнопок.

Это решение было принято на основании открытых комментариев респондентов, указывавших на непоследовательность в озвучивании элементов без явных семантических атрибутов.

Формулировка практических заданий Задания были сформулированы так, чтобы тестировать именно навык фокусировки и активации:

1. Переместить фокус скринридера на гиперссылку «Перейти далее» и активировать её.
2. Последовательно найти и открыть три тематические ссылки («Факт», «Стих», «История»), каждая из которых содержит короткий текст.
3. Найти на странице кнопку «Звук!» и активировать её для воспроизведения звукового сигнала.

Порядок заданий выстроен в порядке возрастания сложности: от простейшего перехода по одной ссылке к работе с несколькими элементами и завершению действия кнопкой.

2.2. Работа с формами

2.2.1. Основания для разработки сценария

На этапе подготовки пользовательских сценариев был учтён высокий уровень важности работы с формами (средняя оценка 4,73 из 5), полученный в результате опроса. Дополнительно были проанализированы рекомендации WCAG и общепринятые практики доступного дизайна форм, что позволило сформировать обоснование для включения данного раздела.

2.2.2. Выделение ключевых элементов управления

Исходя из анализа реальных веб-форм и результатов анкетирования, были определены минимум 8 элемента, которые наиболее часто встречаются в практических заданиях:

- **Текстовое поле** (single-line input)
- **Многострочное текстовое поле** (textarea)

- **Флажок** (checkbox)
- **Переключатель** (radio button)
- **Выпадающий список** (select)
- **Поле загрузки файла** (file upload)
- **Выбор даты** (date picker)
- **Ползунок** (range input)

Каждый элемент был выбран с учётом того, что в ходе опроса респонденты обращали внимание на разнообразие компонентов в реальных формах и необходимость освоения разных приёмов навигации.

2.2.3. Основные затруднения и принятые решения

В процессе формулировки сценариев были выявлены следующие трудности:

1. **Разнообразие способов ввода.** Пользователи могли не понимать, как переходить между текстовыми полями и textarea. *Решение:* в каждом сценарии было чётко указано, какие клавиши (Tab, Shift+Tab, Ctrl+Arrow) задействованы для навигации внутри и между элементами ввода.
2. **Неоднозначность озвучивания состояний.** При переключении чекбоксов и радио-кнопок скринридеры иногда не сообщали текущий выбор. *Решение:* в сценариях акцент сделан на проверку озвучивания «выбран/не выбран» после каждого действия и на необходимость наличия текстовых меток.
3. **Управление сложными элементами.** Элементы типа select, date picker и range требуют специальных приёмов управления (стрелки, клики). *Решение:* для каждого из таких элементов сформулированы отдельные инструкции о комбинациях клавиш и порядке действий, что соответствует требованиям опытных пользователей (19 из 26 имеют более 10 лет практики).
4. **Консолидация контекста.** В одном задании могли смешиваться разные типы полей, что вызывало перегрузку. *Решение:* сценарии разбиты на пять последовательных этапов — от простейших текстовых полей до комбинированной формы, в которой все элементы сочетаются.

2.2.4. Структура пользовательских сценариев

Каждый элемент формы представлен в рамках единого шаблона, включающего три компонента:

1. **Объяснение** его роли и поведения при озвучивании.

2. **Пример** краткой разметки или визуального описания.
3. **Практическое задание** с чётким критерием успешного выполнения.

Этапы освоения выстроены по возрастанию сложности:

- **Этап A. Текстовое поле и textarea**: ввод короткой и многострочной информации, проверка озвучивания и перемещения курсора.
- **Этап B. Checkbox и radio buttons**: множественный и одиночный выбор, контроль состояния каждого элемента.
- **Этап C. Select и file upload**: работа со списками и скачиваемыми ресурсами, проверка названия выбранного пункта или файла.
- **Этап D. Date picker и ползунок**: установка даты и настройка диапазона значений, контроль динамического оповещения скринридера о текущем выборе.
- **Этап E. Итоговая форма**: объединение всех компонентов на одной странице и выполнение комплексного задания «заполнить — отправить — прослушать результат».

При подготовке сценариев соблюдались принципы минималистичного дизайна и чёткого разбиения инструкций, что обеспечивает последовательное освоение навыков без лишней нагрузки на пользователя.

2.3. Взаимодействие со сложными элементами

2.3.1. Основания для разработки сценария

Включение раздела по сложным элементам (таблицы, списки, модальные окна) было обосновано оценкой важности этих навыков на уровне **4,62**. При этом акцент делался не столько на статистике, сколько на характере практических задач современных веб-интерфейсов: работа с табличными отчётами, вложенными списками и всплывающими диалогами — частые и критичные сценарии для незрячих пользователей.

2.3.2. Выделение ключевых компонентов

В сценарии включены три группы элементов:

- **Таблицы** (простые и с объединёнными ячейками)
- **Списки** (маркированные и нумерованные, с уровнями вложенности)
- **Модальные окна** (диалоги, блокирующие фон)

Выбор этих компонентов продиктован их распространённостью, а также тем, что управление ими требует разного подхода к фокусировке и навигации.

2.3.3. Основные затруднения и принятые решения

1. **Навигация по простым таблицам.** Некоторые респонденты отмечали, что в табличных отчётах скринридер не всегда сообщает заголовки строк и столбцов. *Решение:* в тексте сценария уточняется, что при перемещении по таблице необходимо прослушать сообщения о Заголовок строки и Заголовок столбца — это заложено в примерах разметки.

2. **Чтение сложных таблиц.** Объединённые ячейки (colspan, rowspan) создают путаницу при переходе. *Решение:* выделены конкретные критерии поиска (например, «занятие в среду в 14:00»), что позволяет сфокусироваться на одном пересечении и избегать «блуждания» по ячейкам.

3. **Переход между уровнями списков.** Вложенные списки часто теряются при последовательном чтении. *Решение:* в сценарии введены маркеры уровня (“уровень 1”, “уровень 2”) и практическое задание «найти второй элемент во втором уровне», что нацеливает пользователя на точную проверку глубины вложения.

4. **Управление модальными окнами.** Модал блокирует фон и требует переноса фокуса. *Решение:* в описании этапа подчёркивается необходимость активировать кнопку, дождаться автоматического перехода фокуса внутрь окна и затем вернуться к основному содержанию после закрытия.

2.3.4. Структура пользовательских сценариев

Для каждого из трёх компонентов сценарии выстроены по единому шаблону:

1. **Введение** — краткое объяснение целей раздела и ролей элементов;
2. **Пример** — визуальный макет или описанная HTML-структура с указанием заголовков, списков или диалогов;
3. **Задание** — пошаговая инструкция с конкретным критерием успешного выполнения.

Этапы освоения сгруппированы следующим образом:

- **Этап A: Преамбула** *Введение* в понятие «сложный элемент» и обзор трёх групп с кнопкой «Далее».
- **Этап B: Простая таблица** *Задание:* найти ячейку «Овощи × Цена».

- **Этап C: Сложная таблица** *Задание:* определить занятие в среду в 14:00 в таблице расписания.
- **Этап D: Списки** *Задание:* найти второй элемент второго уровня в списке покупок.
- **Этап E: Модальные окна** *Задание:* открыть диалог сообщением, прочитать текст и закрыть окно.
- **Этап F: Итоговое задание** *Комбинированная страница* с таблицей, списком и кнопкой модального окна; требуется последовательно выполнить три действия.

2.4. Встроенные подсказки и справочная информация

2.4.1. Основания для разработки сценария

При формировании сценария четвёртого раздела учитывались следующие факторы:

- Средняя оценка важности встроенных подсказок и справки по результатам опроса составила **4,19** из 5.
- Открытые комментарии респондентов подчёркивали нехватку примеров alt-текста и доступных подсказок в реальных интерфейсах.
- Рекомендации WCAG выделяют использование aria-describedby, aria-label и live-регионов как ключевые приёмы повышения доступности.

На основании этих данных был сформулирован отдельный сценарий, посвящённый освоению знакомства с невизуальными подсказками и динамическими уведомлениями.

2.4.2. Выделение ключевых элементов интерфейса

В сценарии рассматриваются четыре типа подсказок:

1. **ARIA-описания и атрибуты title** (tooltip-атмосфера)
2. **Подсказки при наведении или фокусе** (tooltips)
3. **Контекстная справка** (иконка «?» или **i** с раскрывающимся текстом)
4. **Live-регионы** (aria-live) для динамических сообщений

Каждый тип выбран исходя из частоты их применения в веб-приложениях и различных приёмов взаимодействия, требующих специфических навыков у пользователей скринридеров.

2.4.3. Основные затруднения и принятые решения

1. **Скрытость подсказок в атрибутах**

- *Проблема:* пользователи часто пропускают элементы с title или aria-describedby, если не знают о их наличии.
- *Решение:* в тексте сценария акцентируется необходимость перехода по фокусу и проверка атрибутов при помощи команд скринридера (например, «обновить фокус» или «прочитать свойство title»).

2. **Нестандартизованное поведение tooltips**
   - *Проблема:* разные реализации всплывающих подсказок требуют разных способов активации (мышь, клавиша Tab, сочетания).
   - *Решение:* прописаны инструкции для каждой ситуации: «при наведении мышью — дождаться паузы и нажать клавишу для озвучивания», «при фокусе — воспользоваться командой “прочитать текущее имя элемента”».
3. **Контекстная справка как модальное окно**
   - *Проблема:* раскрывающаяся справка может восприниматься как модальный диалог, нарушая порядок навигации.
   - *Решение:* отдельным этапом сценария показано, как при активации и закрытии справки вернуть фокус к исходному элементу, используя команды «назад» или «возврат фокуса».
4. **Динамическое обновление page content**
   - *Проблема:* live-регионы могут обновляться без явных визуальных изменений, что сбивает с толку пользователя.
   - *Решение:* сценарий содержит пример зоны с aria-live="polite" и пошаговое задание «нажать кнопку — прослушать новое сообщение», чтобы подчеркнуть необходимость внимания к динамическим уведомлениям.

2.4.4. Структура пользовательских сценариев

Каждое задание оформлено по единой схеме:

1. **Введение** — описание типа подсказок и их роли в интерфейсе.
2. **Пример** — конкретный фрагмент интерфейса с указанием используемых атрибутов.
3. **Практическое задание** — чёткие шаги и критерий успеха.

Последовательность этапов:

- **Этап А. ARIA-описания и title** *Объяснение:* текстовое поле с пояснением в aria-describedby. *Задание:* установить фокус на поле «Введите email» и прослушать описание.

- **Этап B. Tooltip при наведении** *Объяснение:* подсказка появляется при наведении или фокусе, оформлена через title. *Задание:* навести фокус на кнопку с иконкой и определить её действие.
- **Этап C. Контекстная справка** *Объяснение:* иконка «?» открывает справочный текст. *Задание:* активировать иконку рядом с полем «Ключ API», прочитать текст и закрыть справку.
- **Этап D. ARIA-метки без визуального текста** *Объяснение:* кнопка с aria-label="Удалить элемент" без видимой подписи. *Задание:* определить действие кнопки, полагаясь лишь на скринридер.
- **Этап E. Live-регионы** *Объяснение:* зона обновляется после действия без перезагрузки. *Задание:* нажать кнопку «Отправить» и прослушать сообщение «Успешно отправлено».
- **Этап F. Итоговое задание** *Объяснение:* комбинированная страница с полем (aria-describedby), кнопкой (title), иконкой справки и live-регионем. *Задание:* последовательно выполнить четыре взаимодействия и убедиться в корректном озвучивании.

3. Разработка архитектуры программного средства

3.1. Проектирование структуры системы

3.1.1. Уровни и модули

1. **Клиентская часть (Frontend)**
   - Одностраничное приложение для последовательного отображения учебных страниц.
   - Компоненты:
     - **AuthClient** — управление идентификацией через cookies и URL-параметры.
     - **Navigator** — логика навигации по этапам тренажёра.
     - **ProgressDisplay** — индикатор процента прохождения, счётчик этапов и затраченного времени.
     - **TutorDashboard** — панель преподавателя для регистрации учеников, отслеживания их прогресса и отправки подсказок.
2. **Серверная часть (Backend)**
   - Четырёхзвенная структура:

- **REST API** — маршрутизация HTTP-запросов.
- **Контроллеры** — авторизация, базовая валидация, преобразование запросов в вызовы сервисного слоя.
- **Сервисный слой** — реализация ключевых сценариев:
- Автоматическая регистрация и идентификация пользователя.
- Фиксация завершения этапов и обновление прогресса.
- Управление связями преподаватель–ученик и подсказками.
- **DAO (Data Access Objects)** — взаимодействие с PostgreSQL через ORM, с возможностью выполнения оптимизированных запросов.

3. **База данных (PostgreSQL)**

- Логическая схема соответствует инфологической модели:
- Таблицы: Users, Tutors, Topics, Levels, User_Level, Progress, Sessions, Hints.
- Внешние ключи и индексы обеспечивают целостность и производительность.
- Триггер при вставке в User_Level автоматически обновляет агрегированные данные в таблице Progress.

4. **Инфраструктура и отказоустойчивость**

- **Контейнеризация**: все сервисы упакованы в Docker для воспроизводимой среды.
- **Реверс-прокси (Nginx)** — маршрутизация запросов и раздача статических ресурсов.
- **Резервное копирование**:
- БД — ежечасный pg_dump, хранится одна последняя копия.
- Статические файлы — суточное копирование, хранится одна последняя копия.
- **Мониторинг и логирование**: сбор метрик и логов для оперативного выявления сбоев и анализа работы системы.

3.2. Описание взаимодействия компонентов

3.2.1. Сценарий «Старт / Продолжение»

1. Клиент отправляет GET /api/session/start.
2. Контроллер проверяет наличие user_id в cookie или URL-параметре.
3. При отсутствии:
- Создаются записи в таблицах Users и Sessions.
- Клиенту возвращается URL первой страницы тренажёра.

4. При наличии:

- Из таблиц User_Level и Progress извлекаются данные о последнем пройденном этапе и проценте прохождения.
- Клиент получает URL для продолжения и текущее состояние прогресса.

3.2.2. Сценарий «Прохождение этапа**

1. Клиент запрашивает GET /api/levels/{level_id}, получая описание этапа и инструкции.
2. После выполнения задания отправляет POST /api/user_levels с { user_id, level_id, session_id }.
3. Сервисный слой проверяет отсутствие дубликатов и создаёт запись в User_Level.
4. Триггер в СУБД обновляет соответствующую запись в таблице Progress.
5. Ответ API содержит обновлённый процент прохождения и суммарное время, после чего клиент обновляет индикатор и предлагает перейти к следующему этапу.

3.2.3. Сценарий «Работа преподавателя**

1. **Регистрация преподавателя**: POST /api/tutors/register — создание записи в таблице Tutors, отправка письма с подтверждением на email.
2. **Управление учениками**:
   - **Создание**: POST /api/tutors/{tutor_id}/students — создаёт запись в Users с заполненным tutor_id.
   - **Список**: GET /api/tutors/{tutor_id}/students — возвращает досье ассоциированных учеников (user_id, name, progress).
   - **Удаление связи**: DELETE /api/tutors/{tutor_id}/students/{user_id} — обнуляет поле tutor_id у ученика.
3. **Подсказки**:
   - **Отправка**: POST /api/hints с { tutor_id, user_id, message } — создание новой записи в Hints.
   - **Чтение**: GET /api/hints/{user_id} — получение непрочитанных подсказок и сохранение времени read_at.

3.2.4. Логирование и мониторинг

- HTTP-запросы и ключевые события (регистрация, завершение этапа, отправка подсказок) фиксируются в логах с указанием session_id и user_id.
- Метрики системы (latency, error rate, active sessions) собираются инструментами мониторинга для своевременного реагирования на инциденты.

3.3. Выбор технологий

В процессе разработки архитектуры было проведено всестороннее сравнение доступных инструментов и фреймворков с целью обеспечить баланс между скоростью разработки, простотой сопровождения и качеством конечного продукта. При выборе технологий учитывались следующие факторы:

- **Фреймворк для серверной части (Flask):** Flask обладает минималистичным ядром и позволяет строить REST-сервисы с гибкой организацией слоёв. Он хорошо интегрируется с SQLAlchemy, имеет обширную экосистему расширений (для валидации, аутентификации, миграций) и прост в освоении.
- **ORM (SQLAlchemy):** Использование ORM ускоряет разработку моделей данных, упрощает миграции схемы (через Alembic) и снижает вероятность ошибок при формировании запросов. Для критичных по производительности участков предусмотрена возможность написания “сырых” SQL-запросов.
- **Одностраничное приложение (SPA) на React/Vue/Angular:** SPA обеспечивает динамичную подгрузку контента без перезагрузки страницы, что критично для пользователей скринридера, поскольку позволяет сохранять контекст и фокус интерфейса. Изучение и выбор конкретного фреймворка опирались на наличие готовых библиотек для управления состоянием и маршрутизацией.
- **PostgreSQL:** Надёжная СУБД с поддержкой транзакций, индексов на сложных типах данных и расширениями.
- **Docker для контейнеризации:** Контейнеризация обеспечивает идентичность сред разработки и продакшн, упрощает масштабирование и развёртывание сервисов.

3.4. Выработка требований к производительности и безопасности

На основании нефункциональных требований, анализа результатов опроса и потенциальных рисков были сформулированы следующие ключевые критерии:

1. **Производительность и отклик интерфейса:**

- Время загрузки страницы не должно превышать 3 секунд при одновременной работе до 100 активных пользователей.
- Для уменьшения задержек предусмотрены:
  - Индексация полей user_id, topic_id, level_id и session_id на стороне БД.
  - Кеширование неизменяемых данных (список этапов, содержание уроков) на уровне CDN или в памяти приложения.
  - Пул соединений с базой данных (connection pool) для снижения накладных расходов на установку соединений.

2. **Безопасность хранения и передачи данных:**

- **Аутентификация и авторизация:** все запросы к API требуют проверки токенов или сессионных идентификаторов; права преподавателя и ученика разграничены на уровне контроллеров.
- **Хранение паролей:** используется PBKDF2/Bcrypt с солями для хэширования паролей в таблице Tutors.
- **Защита от уязвимостей:** внедрены механизмы валидации входных данных (на стороне сервера и клиента), защита от CSRF/XSS через стандартные расширения Flask и настройку заголовков безопасности (Content Security Policy).
- **Ротация резервных копий:** автоматизированные процедуры резервного копирования базы данных (ежечасно) и файлов сервиса (ежесуточно) с удалением старых копий, чтобы снизить риск утечки и обеспечить готовность к восстановлению.

Данные требования позволят обеспечить надёжность, устойчивость к нагрузкам и защиту пользовательских данных в рамках заявленных сценариев эксплуатации.

3.5. Разработка архитектурных диаграмм

Для наглядного представления структуры и поведения системы были выбраны четыре типа диаграмм: диаграммы последовательности, пользовательские сценарии, связи между компонентами системы и ER-диаграмма базы данных. Каждая из них призвана подчеркнуть определённый аспект архитектуры и обеспечить максимально полное понимание строения и функционирования программного средства.

3.5.1. Диаграммы последовательности

Диаграммы последовательности иллюстрируют динамику взаимодействия между объектами и модулями в рамках ключевых процессов:

1. **Старт и идентификация пользователя** На первой диаграмме последовательно воспроизводится сценарий запроса /api/session/start:
   - Клиент инициирует HTTP-запрос, передавая cookie или URL-параметр.
   - Контроллер проверяет наличие записи в таблице Users.
   - В случае отсутствия создаются новые записи в Users и Sessions.
   - Контроллер возвращает ответ с URL первой страницы и параметрами сессии.
2. **Прохождение этапа** Вторая последовательность отражает логику POST-запроса /api/user_levels:
   - Клиент отправляет { user_id, level_id, session_id }.
   - Сервис проверяет дубликат в User_Level.
   - При отсутствии дублирования вставляется запись, триггер обновляет Progress.
   - Сервис формирует ответ с актуальным процентом завершения и общим временем.
3. **Работа преподавателя** Третья диаграмма показывает сценарий создания ученика и отправки подсказки:
   - Преподаватель через UI инициирует POST /api/tutors/{tutor_id}/students.
   - Сервис создаёт User с полем tutor_id.
   - Далее пользователь-ученик получает ссылку и при первом входе автоматически ассоциируется.
   - Преподаватель отправляет подсказку через POST /api/hints, сервис сохраняет запись в Hints и уведомляет клиента.

При описании этих диаграмм особое внимание уделено последовательности вызовов и тому, как контроллер, сервисный слой и база данных обмениваются сообщениями. Выбор именно четырёх ключевых сценариев обусловлен их центральной ролью в пользовательском опыте и администрировании системы.

3.5.2. Пользовательские сценарии

Пользовательские сценарии детализируют шаги, которые выполняют различные роли:

1. **Сценарий гостевого пользователя**
   - Вход на главную страницу → автоматическая регистрация → отображение первой инструкции.

- Нажатие “Далее” → выбор раздела → начало прохождения этапов.

2. **Сценарий возвращающегося ученика**

- Вход через сохранённые cookie → определение последнего пройденного уровня → перенаправление на соответствующую страницу.
- Отображение прогресса и продолжение обучения с текущего шага.

3. **Сценарий преподавателя**

- Регистрация учётной записи → подтверждение email → вход в панель управления.
- Создание нового ученика: ввод имени и получение ссылочного токена.
- Мониторинг прогресса учеников: выбор пользователя → просмотр процента завершения, количества этапов и времени.
- Отправка подсказки: ввод текста → моментальное появление в интерфейсе ученика.

Каждый сценарий описан как линейная цепочка действий и реакций системы, с указанием точек принятия решений (например, “если пользователь уже зарегистрирован”) и возможных ветвлений (например, “при ошибке ввода пароля”). В тексте обозначены все варианты исходов: успешный, неуспешный и повторный.

3.5.3. Диаграммы связей между компонентами системы

Чтобы наглядно представить статическую структуру приложения, составлена диаграмма компонентов, отображающая четыре основных пакета:

1. **Frontend-приложение**

- Компоненты аутентификации, навигации и визуализации прогресса.
- Модуль TutorDashboard для управления учениками и подсказками.
- Клиентские сервисы для работы с REST-API и локальным хранилищем cookie.

2. **Backend-сервис (Flask)**

- Контроллеры, принимающие HTTP-запросы и управляющие аутентификацией, маршрутизацией и валидацией.
- Сервисный слой, отвечающий за бизнес-логику: регистрацию пользователей, обработку этапов, расчёт прогресса, управление подсказками.
- DAO-слой, взаимодействующий с PostgreSQL через SQLAlchemy, с возможностью выполнения “сырых” SQL-запросов.

3. **СУБД (PostgreSQL)**

- Таблицы Users, Tutors, Topics, Levels, User_Level, Progress, Sessions, Hints.
- Триггерная логика для обновления агрегатов.

4. **Вспомогательный контейнер**

- Задачи резервного копирования (pg_dump) и передачи бэкапов на удалённое хранилище.
- Мониторинг (Prometheus-экспортер) и лог-агент (ELK-stack или подобный).

Стрелками показаны основные интерфейсы взаимодействия: REST-вызовы между Frontend и Backend, TCP-соединение к БД, запуск фоновых задач в контейнере бэкапов. Диаграмма подчёркивает независимость модулей и их чётко определённые границы ответственности.

3.5.4. ER-диаграмма базы данных

ER-диаграмма описывает сущности и их связи:

- **Users** 1—* **User_Level** *—1 **Levels**
- **Users** 1—* **Sessions**
- **Users** *—1 **Tutors** (через поле tutor_id)
- **Users** 1—* **Hints** *—1 **Tutors**
- **Topics** 1—* **Levels**
- **Users** — **Topics** (через таблицу Progress)

Для каждой связи указаны кардинальности и направление внешних ключей. Диаграмма содержит основные атрибуты сущностей: первичные ключи, поля времени (registered_at, completion_time, read_at) и служебные идентификаторы. На схеме обозначены обязательные и опциональные атрибуты, а также индексы, влияющие на производительность выборок.

Таким образом, предложенные четыре диаграммы формируют целостное представление архитектуры системы: от описания последовательности взаимодействий и пользовательских сценариев до статических связей между компонентами и моделью данных.

ЗАКЛЮЧЕНИЕ

В ходе практики был проведен всесторонний анализ информации, касающейся разработки программного средства. Проведен Сбор и анализ требований, включая результаты опросов целевой аудитории. Особое внимание было уделено изучению и применению нормативных документов, таких как Международный стандарт WCAG и Национальный стандарт РФ ГОСТ R 52872-2019, что обеспечивает соответствие разработанного решения современным стандартам доступности.

Систематизированы и обоснованы все этапы разработки программного средства. Отражены результаты анализа предметной области, проектирования пользовательского взаимодействия и формирования технической архитектуры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. WebAIM Screen Reader Simulation, [https://webaim.org/simulations/screenreader].
2. TPGi JAWS Inspect, [https://www.tpgi.com/arc-platform/jaws-inspect/].
3. Top Web Accessibility Tools for Developers, [https://blog.pixelfreestudio.com/top-web-accessibility-tools-for-developers/].
4. Inclusive Design 24, [https://inclusivedesign24.org/2025/].
5. Microsoft Accessibility Insights, [https://accessibilityinsights.io/].
6. Web Content Accessibility Guidelines (WCAG), [https://www.w3.org/WAI/standards-guidelines/wcag/].
7. ГОСТ Р 52872-2019. Интернет-ресурсы и другая информация, представленная в электронно-цифровой форме. Приложения для стационарных и мобильных устройств, иные пользовательские интерфейсы. Требования доступности для людей с инвалидностью и других лиц с ограничениями жизнедеятельности, [https://www.frcds.ru/wp-content/uploads/2020/05/GOST52872-2019-min.pdf].
8. The World Wide Web Consortium (W3C), [https://www.w3.org/]

ПРИЛОЖЕНИЕ

АНКЕТА

«Онлайн-тренажёр для незрячих и слабовидящих: сбор требований»

Мы разрабатываем онлайн-тренажёр для обучения незрячих и слабовидящих пользователей работе с веб-интерфейсами.

Веб-тренажёр — это онлайн-сервис, который помогает изучать веб-интерфейс тем, кто использует программы экранного доступа.Веб-сайт, на котором поэтапно, страница за страницей, появляются всё более сложные интерфейсные элементы. Каждая страница содержит описание, как эти элементы найти на веб-странице и как с ними взаимодействовать при помощи скринридера. Существует дополнительная функциональность для преподавателей компьютерных курсов: после регистрации преподавателю доступно подключение и отслеживание прогресса связанных с ним учеников.

Пожалуйста, ответьте на вопросы анкеты — это займёт около 10 минут. Анкета анонимна.

Раздел 1

1. К какой группе вы относитесь?
   - Незрячий/слабовидящий пользователь
   - Преподаватель
   - Эксперт по доступности веб-контента
   - Другое: _____
2. Какой у вас опыт работы со скринридером?
   - Менее 1 года
   - 1 – 5 лет
   - 5 – 10 лет
   - Более 10 лет

Раздел 2

Функциональные требования

1. Оцените важность функций тренажёра по шкале от 1 (неважно) до 5 (очень важно).

Функция	1	2	3	4	5
Навигация по заголовкам, ссылкам, кнопкам при помощи скринридера					
Работа с формами (текстовые поля, флажки, переключатели)					

Взаимодействие со сложными элементами (таблицы, списки, модальные окна)					
Понимание ARIA-ориентиров и семантических тегов					
Практика поиска и исправления ошибок доступности					
Встроенные подсказки и справочная информация					

2. Есть ли другие важные функции?

Раздел 3

Нефункциональные требования

1. Какую скорость отклика тренажёра вы считаете приемлемой?
   - Меньше 1 сек (мгновенная реакция)
   - 1 – 3 сек
   - Больше 3 сек (скорость реакции значения почти не имеет)
2. Насколько для вас важна бесперебойная работа тренажёра?
   - Крайне важно — будет использоваться на уроках, всегда должен быть доступен
   - Важно, но не критично — прохождение можно отложить
   - Не важно — пройти тренажёр можно в любое другое время
3. Какие скринридеры необходимо обязательно поддерживать?
   - NVDA (Windows)
   - JAWS (Windows)
   - ORCA (Linux)
   - VoiceOver (macOS)
   - TalkBack (Android)
   - VoiceOver (iOS)
   - Другое: _____
4. Какие браузеры необходимо обязательно поддерживать?
   - Google Chrome
   - Mozilla Firefox
   - Microsoft Edge
   - Яндекс.Браузер
   - Safari
   - Другое: _____

Раздел 4

Дополнительные функции для преподавателя

Пропустить, если не используете преподавательский функционал.

1. Какие инструменты контроля учеников вам необходимы? (можно выбрать несколько)

- o Отслеживание прогресса по урокам
- o Просмотр статистики ошибок и успешных прохождений
- o Настройка уровня сложности
- o Настройка порядка прохождения разделов
- o Обратная связь с учеником (комментарии в интерфейсе)
- o Создание собственных упражнений
- o Другое: _____

2. Как вы предпочитаете получать уведомления об успехах учеников?

- o По email
- o В Телеграм
- o Внутри тренажёра (преподавательский аккаунт)
- o Другое: _____

3. Нужны ли вам еженедельные или ежемесячные сводки (дайджесты) отчётов?

- o Да
- o Нет
- o Другое: _____

4. Какие метрики успешности учеников для вас важны?

- o Процент успешно пройденных этапов
- o Количество ошибок ученика
- o Время прохождения этапов
- o Количество обращений к подсказкам
- o Другое: _____

Раздел 5

Общие предложения

1. Что, по вашему мнению, важно добавить или изменить в концепции тренажёра?

_____

2. Если хотите участвовать в тестировании, укажите email или Телеграм:

_____