1. Описание проекта

Цель:

Создать интерактивную обучающую веб-игру, в которой студенты MBO 4 на направлении verpleegkundige (уход/медсестринство) практикуют навыки наблюдения, коммуникации, оценки состояния пациента и первичных действий до вызова врача. Игра представляет серию ситуаций из реальной практики с акцентом на эмпатию, правильное распознавание симптомов и принятие безопасных решений.

Основная идея:

Пользователь — медсестра/медбрат, сталкивающаяся с типичными ситуациями ухода за пациентом. Игрок должен правильно проводить наблюдение, измерения, анализировать симптомы и принимать решения: проводить первичный осмотр, сопровождать пациента, оповещать врача или давать рекомендации по первичной помощи.

2. Целевая аудитория и требования к уровню сложности

Целевая аудитория:

Студенты MBO 4 направления verpleegkundige. Уровень подготовки базируется на практических навыках и эмпатии, без необходимости глубоких клинических знаний, свойственных врачам.

Уровень сложности:

- Использование упрощённой медицинской терминологии (без сложных диагнозов)

- Ограничение на базовые обследования и типичные сценарии — напр. измерение давления, пульса, базовый осмотр, анализ крови, ЭКГ, УЗИ.

- Фокус на поддержании безопасности пациента, наблюдательности, коммуникации с пациентом и корректном выполнении последовательности действий.

3. Функциональные требования

3.1 Основные сценарии игры

**Главное меню:**

- Кнопки: "Начать игру", "Продолжить", "Настройки", "Выход".

- Возможность выбора профиля (с привязкой к учетной записи студента).

**Сценарии пациента:**

- Несколько кейсов (минимум 3-5) с различными жалобами (например, боль в спине, усталость, головокружение, боли в ногах).

- Каждый кейс представляется в виде диалогового окна, где пациент излагает свою жалобу в простом языке.

- Варианты диалоговых ответов — выбор вариантов с разным эффектом (назначение обследования, ошибочная трактовка ситуации).

**Обследование/диагностика:**

- После диалогового этапа игрок выбирает действие (измерить пульс/АД, провести осмотр, назначить анализ).

- Визуальная компонента: минимально затратная анимация или статичный экран с указанием результата.

- Результаты обследования влияют на ход игры (открытие дополнительных вариантов и подсказок).

**Назначение первичной помощи:**

- Игра не должна содержать сложных алгоритмов лечения, а лишь демонстрировать пример правильного алгоритма первичного ухода.

- Возможные действия: комфортирование пациента, вызов врача, контроль жизненных показателей.

- Обратная связь от игры: краткий комментарий (например, «Вы правильно измерили давление» или «Пропущен важный симптом»).

**Этап оценки:**

- По завершении кейса выводится итоговый отчёт, в котором кратко анализируются действия игрока: какие проверки выполнены, какие моменты упущены, рекомендации по улучшению.

3.2 Варианты обследований (подробнее описаны в разделе 4 ниже)

- Физикальный осмотр (осмотр, пальпация, измерение, прослушивание)

- Лабораторные анализы (общий анализ крови, биохимия, анализ мочи)

- Инструментальные обследования (ЭКГ, УЗИ, рентген)

3.3 Дополнительные функции

**Локальные сохранения и прогресс:**

- Использование localStorage для хранения текущего прогресса и баллов.

- Вариант серверного сохранения через API (JWT-аутентификация, REST/GraphQL).

**Обратная связь от преподавателя:**

- После завершения кейса выводится текстовой комментарий с рекомендациями.

- Возможность хранения оценок для анализа преподавателем.

**Адаптивность:**

- Интерфейс должен корректно отображаться на десктопах и мобильных устройствах.

**Локализация:**

- Поддержка двух языков: нидерландский (Verpleegkunde) и русский (при необходимости).

- Возможность расширения для добавления других языков (i18n).

4. Нефункциональные требования

Производительность:

- Минимальное время загрузки (оптимизированная графика с использованием SVG и статичных изображений).

- Быстрая анимация и реакция элементов (в пределах 100–200 мс).

Удобство использования (UX):

- Интуитивный интерфейс, понятная навигация.

- Минимальное обучение: максимум 2-3 подсказки в начале каждого кейса.

Безопасность:

- Если реализуется серверная часть — использовать шифрование данных, JWT для аутентификации и авторизации.

- Обработка персональных данных (если применимо) в соответствии с GDPR.

Масштабируемость:

- Легкость добавления новых сценариев через изменение JSON-файлов.

- Модульная архитектура, позволяющая расширять функционал без глобальных переделок.

Тестируемость:

- Наличие модульных тестов (Jest для frontend, соответствующий фреймворк для backend).

- Интеграционные тесты (например, Cypress) для проверки ключевых сценариев.

5. Архитектура системы

5.1 Фронтенд

**Стек технологий:**

- JavaScript/TypeScript: React или Vue для создания компонентов, взаимодействующих с пользователем.

- UI-библиотеки: Radix UI или shadcn/ui для компонентов типа модальных окон, кнопок, карточек.

- Стилизация: TailwindCSS или SCSS для адаптивного дизайна.

- Анимации: Framer Motion для плавных переходов (например, анимация сканирования или появления результатов обследования).

- Роутинг: React Router (или аналог для Vue) для организации навигации между экранами (главное меню, кейс, итоговый экран).

**Компоненты UI:**

- Header и Footer: статичные элементы, отображающие название игры, текущий уровень и настройки.

- Карточки пациента: отображение информации о пациенте, текст диалога и варианты действий.

- Окна обследования: простые статичные или с минимальной анимацией экраны, демонстрирующие результаты (например, график пульса, таблица анализа крови).

5.2 Бэкенд

**Стек технологий:**

- Node.js (Express) или Django (Python) для создания API.

- База данных: PostgreSQL для хранения профилей пользователей, сценариев, прогресса и оценок.

- Аутентификация: JWT для безопасной авторизации.

- API: REST или GraphQL для взаимодействия с фронтендом.

**Основные маршруты API:**

- POST `/login` — аутентификация пользователя.

- GET `/scenarios` — получение списка кейсов.

- POST `/scenario/:id/result` — сохранение результатов кейса и комментариев преподавателя.

- GET `/progress` — получение истории выполнения кейсов и текущих оценок.

5.3 Локальное сохранение

- Использование localStorage для хранения промежуточных данных:

- Ключи: `currentScenario`, `userProgress`, `settings`.

- Механизм автосохранения при каждом изменении состояния (например, выбор варианта ответа).

6. Структура данных и JSON-сценарии

6.1 Формат JSON-сценария

Каждый кейс хранится как отдельный JSON-объект.

7. UI/UX Дизайн

7.1 Основной интерфейс

**Главное меню:**

- Отображает логотип приложения, название и краткое описание.

- Кнопки и ссылки должны быть крупными и разборчивыми с учетом целевой аудитории.

- Анимационные эффекты (например, легкая пульсация кнопок) для привлечения внимания.

**Экран пациента:**

- Разделён на область с изображением пациента (стилизованный портрет с возможностью анимации эмоций) и область диалога.

- Слева – информация о пациенте: имя, возраст, история болезни (коротко).

- Справа – диалоговое окно с текстом и кнопками с вариантами ответов.

- Использовать цветовую кодировку для положительной/отрицательной обратной связи (зелёный/красный).

**Экран обследования:**

- Минимальная анимация (напр., затемнение экрана с эффектом «сканера»; а затем появление карточки с результатами).

- Карточки, отображающие результаты в виде текста, графиков или простых изображений (SVG, PNG).

- Пример: для ЭКГ – статичная анимированная линия, для анализа крови – таблица с показателями.

7.2 Адаптивный дизайн

- Использовать гибкую верстку (Flexbox, Grid) для поддержки экранов от смартфонов до десктопов.

- Простота навигации за счёт крупной типографики и интуитивных иконок.

7.3 Указания по анимации

- Лёгкие анимации для переходов между сценами:

- Анимация сканирующего света для эффекта проведения обследования (CSS-анимации или Framer Motion).

- Плавное появление/исчезновение окон с результатами.

8. Минимизация затрат энергии при визуализации обследований

8.1 Использование статичных изображений и SVG

**SVG-иллюстрации:**

- Применять векторную графику для иконок, диаграмм, линий ЭКГ. Это минимизирует нагрузку на процессор и уменьшает размер файлов.

- Карточки с результатами:

- Статичный дизайн карточек, в которых результаты преподносятся в виде текста и простых графиков без динамической визуализации, если не требуется анимация.

8.2 Анимация с минимальным потреблением

**CSS-анимации:**

- Применять только лёгкие CSS-анимации (например, переход прозрачности, плавное увеличение размера), чтобы избежать сложных вычислений.

- Оптимизированный JavaScript:

- Минимизировать использование сложных вычислений или WebGL для визуализации. Отказ от 3D-графики.

8.3 Кэширование и ленивые загрузки

- Использовать кэширование изображений и ресурсов.

- Лениво загружать элементы обследования, которые не видны сразу, чтобы не нагружать устройство пользователя.

9. Интеграция аудио и обратной связи

Аудиоэффекты:

- Небольшие звуковые эффекты при выборе действий (нажатие кнопок, "звук сканера" при обследовании).

- Голосовые подсказки (опционально):

- Если позволит бюджет, можно добавить короткие аудио подсказки (озвучка пациента или комментарии преподавателя).

- Обратная связь:

- После каждого кейса показывать текстовый комментарий с рекомендациями и объяснением правильности действий.

10. Тестирование и качество

10.1 Тестирование кода

**Frontend-тестирование:**

- Использовать Jest и React Testing Library для модульных тестов основных компонентов.

**Интеграционные тесты:**

- Cypress для эмуляции пользовательских сценариев (нажатие кнопок, прохождение кейса, сохранение прогресса).

- Backend-тестирование:

- Тесты на API (Postman/Newman или аналогичные инструменты).

10.2 Тестирование UX

- Проведение тестов с реальными пользователями (студентами MBO 4) для выявления трудностей в навигации и понимании сценариев.

- Проведение A/B-тестирования различных вариантов оформления и уровня обратной связи.

10.3 Логирование и аналитика

- Встроить систему логирования действий пользователя для анализа ошибок и коррекции сценариев.

- Собирать данные о времени прохождения кейсов и ошибках для оптимизации образовательного процесса.

11. Безопасность и конфиденциальность

Аутентификация:

- Если используется серверный компонент, реализовать JWT-аутентификацию.

**Хранение данных:**

- Данные о пользователях и их оценках должны храниться с учётом конфиденциальности (шифрование, доступ только по ролям).

**Защита API:**

- Реализовать защиту от CSRF, XSS и других распространённых угроз.

12. Развертывание и инфраструктура

12.1 Хостинг

**Frontend:**

- Платформы: Vercel или Netlify для быстрого деплоя статичных файлов.

**Backend:**

- Платформы: Heroku, Render или Railway для Node.js/Django.

**База данных:**

- PostgreSQL на платформе Supabase или любом другом надёжном сервисе.

12.2 CI/CD

- Наладить процесс непрерывной интеграции (GitHub Actions, GitLab CI) для автоматического тестирования и деплоя при обновлениях кода.

13. Документация и поддержка

Документация к API:

- Составить подробное описание всех REST/GraphQL маршрутов (Swagger/OpenAPI).

**Кодовые комментарии и Wiki:**

- Разработчики должны вести внутреннюю документацию для упрощения поддержки и расширения функционала.

- Обучающие материалы для преподавателей:

- Интегрированный раздел с пояснениями и инструкциями по использованию оценок и анализа прогресса студентов.

14. План реализации (Roadmap)

1. Этап 1 – Проектирование и дизайн:

- Сбор требований, разработка wireframes, утверждение JSON-сценариев и начальных UI-мокапов.

2. Этап 2 – Прототипирование:

- Разработка базового прототипа на выбранном фреймворке (React/Vue) с минимальными сценариями.

3. Этап 3 – Интеграция функционала:

- Реализация диалоговой системы, визуализации обследований, автосохранения и обратной связи.

4. Этап 4 – Интеграция бекенда:

- Разработка API, подключение аутентификации и хранения прогресса (если требуется).

5. Этап 5 – Тестирование:

- Проведение unit-, интеграционных тестов, UX-тестирований с участием целевой аудитории.

6. Этап 6 – Деплой и сбор обратной связи:

- Развертывание на хостинге, сбор статистики, внесение улучшений.