Build Plan: Frontend User Interface (React.js)

This document outlines a phased development plan for the React-based Frontend User Interface. The approach is designed to deliver a functional and intuitive user experience incrementally, aligning with the backend services' development.

Phase 1: Foundation & Core Study Management

Goal: Establish the application's structure, set up user authentication, and build the core interfaces for managing research studies.

Key Objectives & Milestones	Technologies & Libraries
Project Setup & Architecture: - Initialize React project with TypeScript (e.g., using Vite). - Integrate a UI component library (e.g., Material-UI, Shadcn/UI). - Set up routing, global state management (e.g., Zustand, Redux Toolkit), and API client (e.g., Axios). - Create a main application layout (sidebar, header, content area).	- **Framework:** React.js, TypeScript - **UI Library:** Material-UI or Ant Design - **State Management:** Zustand or Redux Toolkit
Authentication & Dashboards: - Build login/logout functionality, connecting to the Central Backend's auth endpoints. - Implement protected routes for authenticated users. - Create a shell for the main Dashboard, including basic widgets and alert components.	- **Routing:** React Router
Study Management Interface: - Develop UI for listing, creating, and viewing research studies. - Create a form for defining study objectives and criteria. - Connect all components to the corresponding APIs on the Central Backend.	- **Forms:** React Hook Form

Deliverable: A secure, single-page application where authorized users can log in, view a dashboard, and manage the lifecycle of research studies.

Phase 2: Data Interaction & Visualization (Weeks 4-6)

Goal: Implement the dynamic form system for data entry and build interfaces for visualizing matched patients and key metrics.

Key Objectives & Milestones	Technologies & Libraries

	Dynamic Form Rendering: - Develop a component that dynamically renders a form based on a JSON Schema fetched from the backend. - Implement the standardized data entry view, clearly differentiating pre-populated and manually entered data.	- **JSON Schema:** `rjsf` (React JSON Schema Form) or custom renderer.
	Case Matching Visualization: - Create the UI to display lists of potentially eligible patients returned by the AI service. - Design components for researchers to review patient details, validate the match, and confirm eligibility.	- UI Component Library
	Dashboard Enhancement & Basic Stats: - Populate dashboard widgets with real data (e.g., enrollment progress). - Integrate a charting library to display simple trend graphs. - Build the interface for basic descriptive statistics (mean, median, etc.).	- **Charts:** Chart.js or D3.js (via a React wrapper like Recharts)

Deliverable: An interactive interface allowing users to enter data via dynamic forms, review AI-suggested patient matches, and monitor study progress on a visual dashboard.

Phase 3: Generative AI Integration & Data Export (Weeks 7-8)

Goal: Weave the LLM's capabilities directly into the user workflow to provide intelligent assistance and enable easy data export.

	Key Objectives & Milestones	Technologies & Libraries
	Direct GenAI Interaction: - **Criteria Augmentation:** Enhance the study criteria form to call the GenAI service, showing suggestions in real-time. - **Note Summarization:** Add a "Summarize" button/icon next to long text fields that opens a pop-up with a concise summary from the LLM.	- API Client (Axios, Fetch)
	One-Click Data Export: - Implement UI elements (e.g., buttons) to trigger the data export process on the Central Backend. - Handle the file download securely and provide user feedback during the export process.	- UI Component Library

Deliverable: A "smart" UI where users can leverage generative AI for assistance and easily export curated datasets with a single click.

Phase 4: Advanced Features, Refinement & Deployment (Weeks 9-10)

Goal: Integrate the final Q&A feature, conduct thorough testing, and prepare the application for production deployment.

	Key Objectives & Milestones	Technologies & Libraries
	Contextual Q&A (RAG) Interface: - Build a chat-style component for the contextual Q&A feature. - The interface will	- UI Component Library

	send user questions to the GenAI service and display the source-backed answers.	
	Final Testing, UX Refinement & Deployment Prep: - Conduct end-to-end testing across the entire platform. - Focus on responsiveness, accessibility (a11y), loading states, and consistent error handling. - Finalize build configurations and create a `Dockerfile` for easy deployment.	- **Testing:** Jest, React Testing Library

Deliverable: A polished, fully-featured, and containerized frontend application ready for production deployment and user acceptance testing.