Technical Presentation Outline (10 Minutes)

1. Intended User Description (1 min)

- **User Profile & Organizational Context:**
  - **Profile:** Healthcare Analyst or Clinical Informatics Specialist within a hospital or health system's analytics department.
  - **Organizational Hierarchy (Hypothetical Example - Adjust as needed):**
    - CEO -> Chief Medical Officer -> Director of Clinical Analytics -> **Healthcare Analyst (User)**
  - **Thought Process:** The application focuses on analyzing aggregated patient data (demographics, conditions, readmissions, length of stay) and requires interpretation beyond raw data display, fitting an analyst role rather than direct patient care or high-level executive summary (though summaries are provided).
  - **Assumed Knowledge:** User understands basic healthcare terminology (conditions like Diabetes, Hypertension, COPD), metrics (readmission rates, length of stay), demographic categories, and data analysis principles. Familiarity with EHR data concepts is assumed.

2. User Decision-Making Needs (1 min)

- **Key Areas of Responsibility (Examples):**
  - Monitoring and reporting on patient population health trends.
  - Identifying factors contributing to hospital readmissions.
  - Analyzing operational metrics like average length of stay.
- **How Application Meets Needs:**
  - Provides visualizations and summaries of demographic distributions, condition prevalence, and readmission rates (`reports/page.tsx` , analysis functions in `lib/data-parser.ts`).
  - Allows filtering and exploration of patient data to investigate specific cohorts (`reports/page.tsx`).
  - Calculates key metrics like average length of stay, readmission rates, condition correlations (`getAverageMetrics` , `calculateReadmissionRate` , `getReadmissionByCondition` in

`lib/data-parser.ts`).
    - Offers AI-driven summaries for quick insights (`model/page.tsx` using `/api/summarize`).
- **Arrival at Decision Needs:**
    - Based on typical responsibilities of healthcare analyst roles and the types of analyses enabled by the data available in `patient-data.csv`.
- **Challenges Faced:**
    - [List specific challenges YOU faced, e.g., Translating high-level analytical goals into specific data queries/visualizations? Handling data granularity?]
- **Overcoming Challenges:**
    - [Explain how YOU overcame these, e.g., Developed specific data aggregation functions (`get...` functions)? Chose appropriate chart types?]
- **Relevant Coursework:**
    - [List specific courses and how they helped, e.g., Database Management, Health Informatics, Data Visualization]

3. Data Validation and Preparation (2 mins)

- **Source Data Identification (Based on `patient-data.csv` columns - Map to your actual EHR):**
    - **Hypothetical EHR Tables/Fields (Replace with actuals):**
        - `Patients` Table: `PatientID` (-> `Patient_ID`), `DOB` (-> calculated `AGE`), `Gender` (-> `Gender`), `Race` (-> `Race`), `Ethnicity` (-> `Ethnicity`)
        - `Admissions` Table: `AdmissionID`, `PatientID`, `AdmissionDate`, `DischargeDate` (-> calculated `Length_Of_Stay_DAYS`), `ReadmissionFlag` (-> `Is_Readmission`)
        - `Diagnoses` Table: `AdmissionID`, `DiagnosisCode`, `IsPrimary` (-> calculated `Total_Conditions`, `Has_Diabetes`, `Has_Hypertension`, etc. based on codes)
        - `Medications` Table: `AdmissionID`, `MedicationName` (-> calculated `Total_Medications`)
        - `Procedures` Table: `AdmissionID`, `ProcedureCode` (-> calculated `Total_Procedures`)
    - **Identification Process:** [Explain how YOU identified these tables/fields, e.g., Explored EHR schema documentation? Queried database metadata? Collaborated with DBAs?]
- **Data Validation:**
    - **Process:** Primarily performed during CSV parsing in `lib/data-parser.ts` (`parseCSVData` function).
        - Checks for expected number of columns per row.

        - Attempts numeric conversion (`parseInt` , `parseFloat`) and checks for `isNaN` , using 0 as a fallback.
        - Uses a specific helper (`parseBoolean`) for boolean fields (`TRUE` , `1` , `YES`).
        - Filters out empty lines.
    - **Basis:** Ensuring data types conform to the `PatientRecord` interface and handling common inconsistencies found in raw data exports.
- **Data Transformation/Computation:**
    - **Operations (in `lib/data-parser.ts`):**
        - Parsing strings to appropriate types (Number, Boolean).
        - Assigning default values (e.g., 0) for invalid numeric data.
        - Calculating aggregate metrics for display (e.g., `getAverageMetrics` , `calculateReadmissionRate`).
        - Grouping data into categories for analysis (e.g., age groups in `getConditionsByAge` , LOS buckets in `getLengthOfStayDistribution`).
    - **Rationale:** To convert raw data into a structured format suitable for the application's state (`PatientRecord[]`), handle potential data quality issues gracefully, and pre-compute metrics/distributions needed for UI display and analysis.

4. Design of User Interface (3 min)

- **(Walkthrough each relevant page/component)**
- **`/model` Page (`app/model/page.tsx`):**
    - **Information Presented:** CSV Upload area, processing status (success/error messages), preview of first 5 parsed records (ID, Age, Gender, Conditions, Readmission), AI-generated data summary (if successful), Download button for processed data.
    - **Interactions:** Upload CSV file via drag-and-drop or browse (`CSVUploader`), trigger AI summary generation upon successful parse, download the processed data as a new CSV (`handleDownload`).
    - **Meeting User Needs:** Allows users to ingest their own datasets, get immediate validation feedback, preview data structure, obtain a high-level AI summary, and export cleaned data.
- **`/reports` Page (`app/reports/page.tsx`):**
    - **Information Presented:** Paginated table displaying patient records (ID, Age, Gender, Length of Stay, Total Conditions, Readmission status), search bar, filters for Gender and Condition, total patient count, current page/total pages.

    - **Interactions:** Search by Patient ID/Age/Gender (`SearchBar`), filter data by Gender (`Select`), filter data by specific Condition (`Select`), clear filters, navigate through pages (`Button` with pagination logic).
    - **Meeting User Needs:** Enables detailed exploration and filtering of the patient cohort, allowing users to drill down into specific segments based on demographics or medical conditions.
- **Shared Components (e.g., `components/ui/`):**
    - **Information Presented:** Consistent visual elements (Buttons, Tables, Cards, Selects) based on `shadcn/ui` .
    - **Interactions:** Standard UI interactions (button clicks, dropdown selections, text input).
    - **Meeting User Needs:** Provides a consistent, modern, and usable interface across the application.

5. Validation and Verification (3 min)

- **Rationale for Satisfying Needs:**
    - The application is verified against the decision-making needs outlined in the traceability matrix by examining its implemented features.
- **Verification using Traceability Matrix:**
    - **Requirement:** Enable clinicians/analysts to interact with and explore readmission data (Matrix Rows 4, 7).
        - **Information Needed:** Patient demographics, diagnosis, LOS, readmissions, time periods.
        - **Method of Provision:** The `/reports` page (`app/reports/page.tsx`) provides an interactive UI with a paginated table, search functionality, and filter options for Gender and Condition.
        - **Validation:** This directly meets the need for data exploration. User testing would confirm the accuracy and usability of filters and data display.
    - **Requirement:** Visualize readmission trends and related factors (Matrix Rows 3, 7).
        - **Information Needed:** Aggregated trends, demographic correlations, condition correlations.
        - **Method of Provision:** While specific time-series charts aren't explicitly coded in `/reports` or `/model` , the underlying data aggregation functions exist in `lib/data-parser.ts` (e.g., `getReadmissionByCondition` , `getDemographicDistribution` , `getConditionsByAge`). The `/reports` page allows

filtering that enables viewing subsets relevant to these trends. The AI summary on `/model` provides high-level insights.
        - **Validation:** The provision of filterable data and aggregated metrics partially meets this need. Full validation would require implementing specific visualizations (like line/bar charts) and comparing them against known data or stakeholder feedback.
    - **Requirement:** Provide insights for healthcare professionals (Matrix Row 2).
        - **Information Needed:** Risk factor contributions, patient-level details.
        - **Method of Provision:** The application provides patient-level detail exploration via the `/reports` page. The AI summary on `/model` aims to provide high-level insights. Functions like `getReadmissionByCondition` in `data-parser.ts` calculate correlations relevant to risk factors.
        - **Validation:** Exploration features are present. However, specific risk factor contribution charts (e.g., Feature Importance, SHAP values) are not implemented in the reviewed code. Validation would require domain expert review of the available correlations and summaries.
    - **Requirement:** Predict patients at risk / Optimize resources / Achieve specific AUC-ROC / Track progress / Display confidence (Matrix Rows 1, 3, 5, 6, 8).
        - **Information Needed:** ML Risk score, model performance metrics, prediction confidence, historical trends.
        - **Method of Provision:** These requirements relate to a predictive ML model. The reviewed codebase (`lib/data-parser.ts` , `app/model/page.tsx` , `app/reports/page.tsx`) focuses on data parsing, display, filtering, and descriptive analysis/summarization. It does not currently contain the implementation or display of an ML prediction model, risk scores, AUC-ROC metrics, or confidence indicators.
        - **Validation:** These specific requirements are not met by the current front-end/data-parsing implementation. Validation would require building, integrating, and evaluating an ML model separately.
- **Overall Verification:** The application successfully provides features for data ingestion, cleaning (via parsing logic), exploration, and basic summarization, addressing the descriptive and interactive analysis needs. The predictive modeling requirements outlined in the matrix would require further development and integration of an ML component.