SAMSUNG INNOVATION CAMPUS

CODING AND PROGRAMMING

Project Title:

"HEALTHCARE PATIENT ADMISSION RECORDS DATA TRANSFORMATION"

Type: Using Stream lit and Pandas

Description:

- The project outlines a Healthcare Patient Admission Data Management System using Stream lit and Pandas.
- It focuses on handling hospital admission records, including patient details, admission/discharge dates, Stay durations, and diagnoses.
- The workflow mainly includes loading data, cleaning (date parsing, missing values), duplicate removal.
- It provides filtering, grouping and aggregation (admissions per department, average stay), along with visualizations like charts.
- Finally, the processed dataset is exported into a transformed/updated CSV file for hospital reporting and record management.

Member Details:

Name: Keerthana B

USN: 4GW23CI021

Email: keerthanab675@gmail.com

Department: CSE(AI&ML)

Submission date: 02/09/2025

College: GSSS Institute of Engineering and Technology for Women, Mysuru

Index:

Detailed Explanation on Tech-Stack And their Feature:

1. Stream lit (Frontend & UI Layer)

Why used?

- **To build an interactive and user-friendly web interface quickly without needing traditional web development frameworks.**

Role in project:

- **Provides an easy way to upload CSV files, add/edit/delete patient records, and display data in tables.**
- **Offers interactive widgets like forms, checkboxes, filters, and sidebar controls for exploring records.**
- **Generates real-time visualizations (charts and graphs) directly in the browser.**

---

2. Pandas (Data Processing & Transformation)

Why used?

- **For efficient handling, manipulation, and analysis of structured healthcare data.**

Role in project:

- **Reading and writing CSV files (patient admission records).**
- **Cleaning and transforming data (handling missing values, parsing dates).**
- **Creating derived fields such as Stay Duration (difference between discharge and admission dates).**
- **Performing grouping, aggregation, and filtering operations for insights (e.g., admissions per department, average stay).**
-

---

3. Matplotlib (Data Visualization)

Why used?

- **To create clear and customizable charts for data analytics.**

Role in project:

  - **Plotting bar charts for department-wise admissions.**
  - **Creating pie charts for gender distribution.**
  - **Helping healthcare staff visually interpret patterns and distributions in patient data.**

---

4. Python Standard Libraries

- **datetime & date (from datetime):** Used for handling admission and discharge dates.
- **io (Bytes IO):** Used for generating in-memory CSV files so users can download the transformed dataset.
- **OS:** Helps check if the dataset file already exists and manages auto-saving of records.

📋 End-to-End Workflow of Healthcare Patient Admission Portal:

1. Data Input (CSV Upload / New Record Entry)

- The app starts by **loading an existing patient_admissions.csv** if available.
- Alternatively, the user can **upload a CSV file** containing admission records.
- Users can also **add new patient entries manually** using the Stream lit form (Patient ID, Name, Department, Admission Date, etc.).

---

2. Data Cleaning & Preprocessing (via Pandas)

- Dates (Admission Date, Discharge Date) are converted into proper datetime format.
- If **Discharge Date is missing**, it is auto-filled with the **current date**.
- **Stay Duration** is calculated as Discharge Date – Admission Date.

- Duplicate records are removed based on **Patient ID + Admission Date**.
- Missing values in other fields are handled to ensure consistency.

---

3. Data Storage & Persistence

- The cleaned dataset is stored in **s t. session _state. d f** so that all updates remain active while the app is running.
- On each change, the data is **saved back into patient_admissions.csv** for persistence across sessions.

---

4. Data Exploration & Interaction (Stream lit UI)

- Users can **view the dataset** in a table format.
- Filtering options allow searching by **Name, Department, Diagnosis**.
- Advanced filters: e.g., patients in *Cardiology* with stay >5 days.
- Records can be **edited or deleted** interactively.

---

5. Analysis & Aggregation (via Pandas)

- Admissions are grouped by **Department** to count the number of patients.
- Average **Stay Duration** is computed department-wise.
- Other aggregations (e.g., gender distribution) are prepared for insights.

---

6. Visualization (via Matplotlib + Stream lit)

- **Bar Chart** → Department-wise admission count.
- **Pie Chart** → Gender distribution.
- These charts make it easy for healthcare staff to interpret patterns based on number of admissions.

---

7. Output & Reporting

- After transformations, the final dataset (with cleaned dates, calculated stay duration, and no duplicates) is prepared.
- Users can **download the transformed dataset** as transformed_admissions.csv.
- This file acts as a clean hospital record for reporting and future use.

Algorithm and its explanation:

Algorithm 1: Load & Initialize Data

-Start Stream lit application.

-Check if patient_admissions.csv exists:

*If yes → load into Pandas Data Frame.

*Else → create an empty Data Frame with predefined columns.

-Store the Data Frame in s t. session_ state for persistence.

-If user uploads a CSV file → replace existing Data Frame with uploaded data.

Algorithm 2: Data Cleaning & Transformation

-Parse Admission Date and Discharge Date columns into datetime objects.

-If Discharge Date is missing,

- fill it with today's date.

-Calculate Stay Duration = Discharge Date – Admission Date (in days).

-Drop duplicate records using Patient ID + Admission Date as a unique key.

-Save the transformed Data Frame back to session state.

Algorithm 3: Add New Patient Record Manually

-Accept patient details from Stream lit form (Patient ID, Name, Gender, Department, Diagnosis, Admission Date, Discharge Date).

-If Discharge Date is empty

-set it to today's date.

-Calculate Stay Duration from Admission Date and Discharge Date.

-Append new record to Data Frame.

-Save updated Data Frame to session state and CSV file.

Algorithm 4: Edit / Delete Patient Record

EDIT:

-Select a patient record from Data Frame.

-Update fields (Name, Department, Diagnosis, Dates, etc.).

-Recalculate Stay Duration.

-Replace the old row with the updated row.

-Save Data Frame to session state & CSV.

DELETE:

-Select a patient record.

-Remove it from Data Frame.

-Save updated Data Frame to session state & CSV.

Algorithm 5: Data Exploration & Filtering

-Accept user input for filtering (by Department, Diagnosis, Name, Stay Duration).

-Apply filter conditions using Pandas.

-Display filtered records in a Stream lit table.

Algorithm 6: Aggregation & Analysis

-Group records by Department.

-Count total admissions per department.

-Calculate average Stay Duration per department.

-Count patients by Gender.

-Store results in summary tables.

Algorithm 7: Visualization

-Use Matplotlib to plot:

-Bar chart for Department-wise admissions.

-Pie chart for Gender distribution.

-Render charts in Stream lit app.

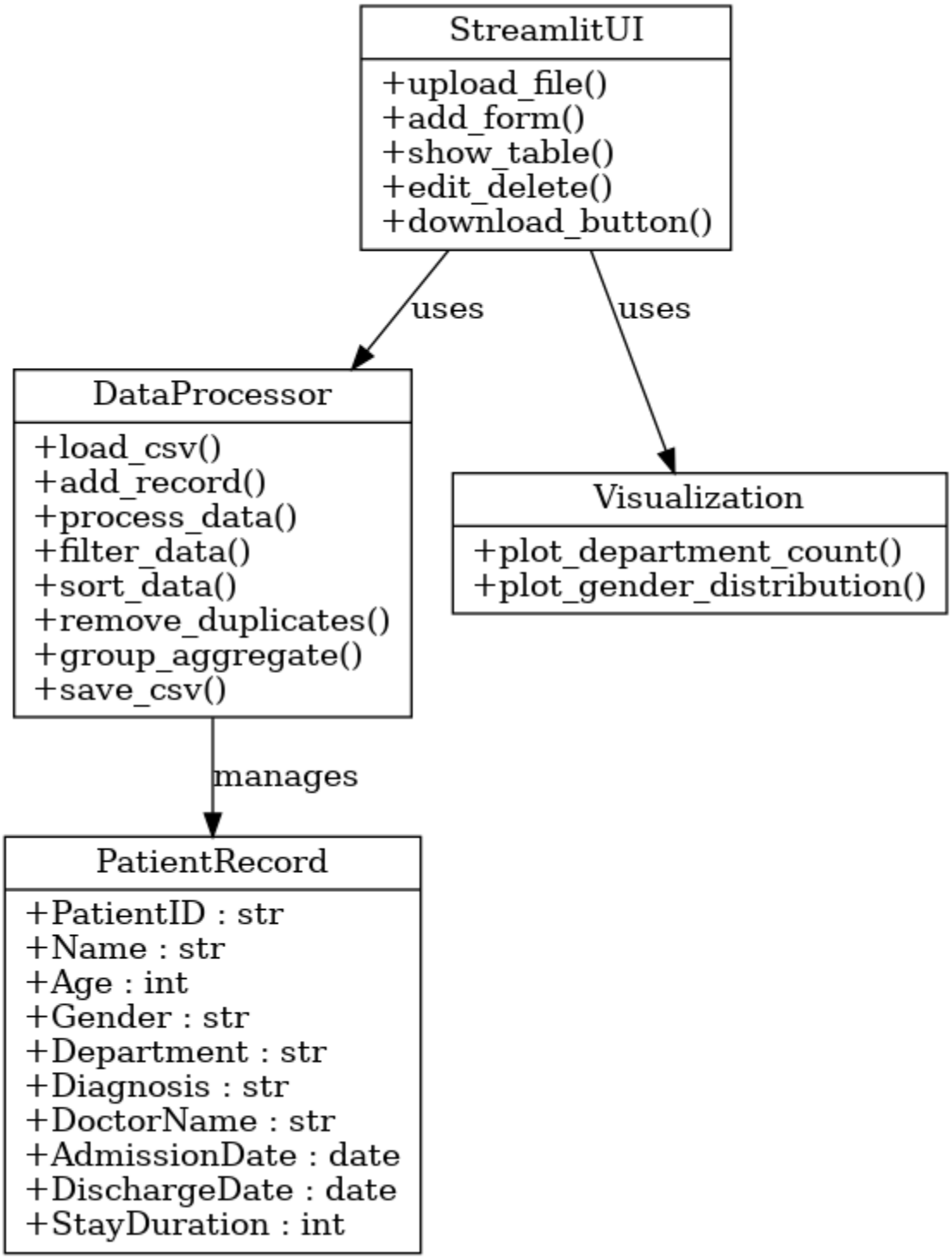
Algorithm 8: Download Transformed Dataset

-Convert final Data Frame into CSV format using Pandas.

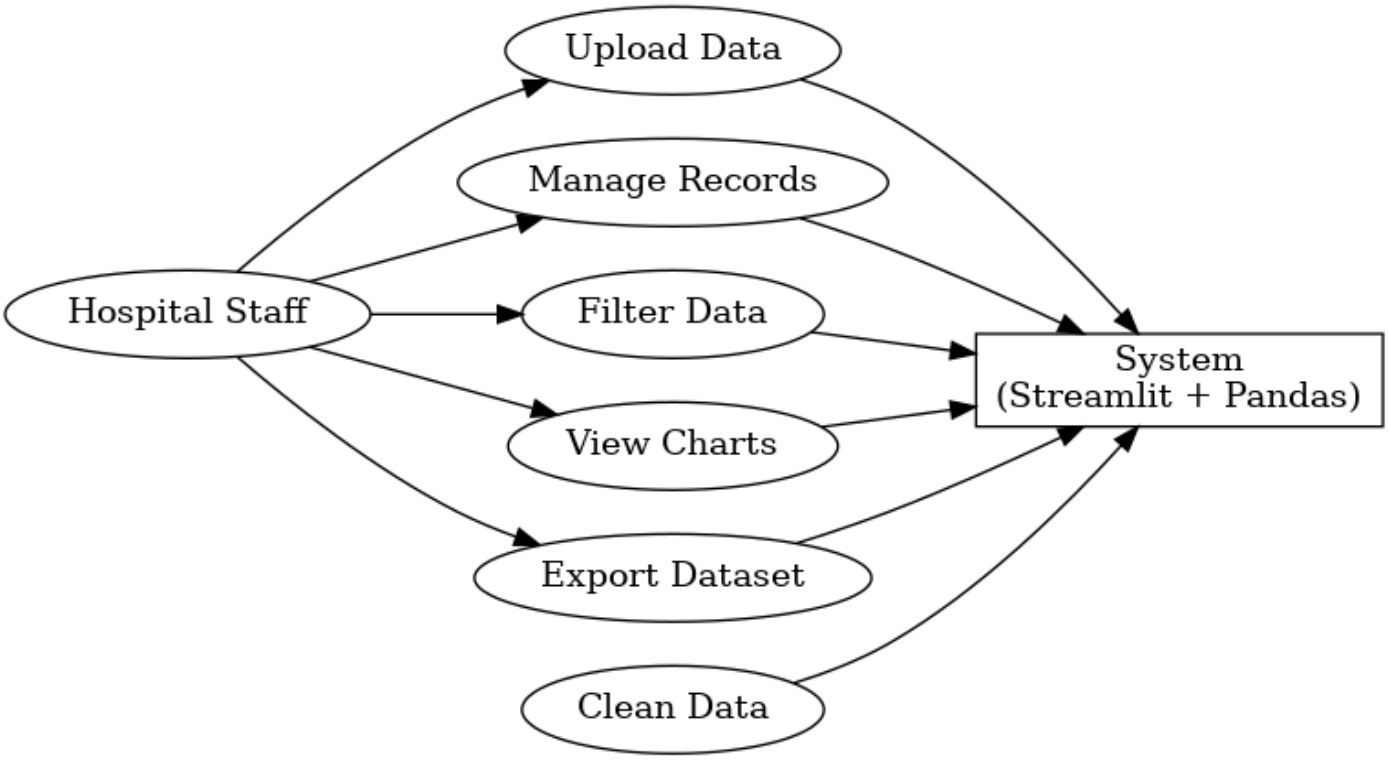
-Save into memory (Bytes IO).

-Provide a Stream lit download button to export as transformed_admissions.csv.

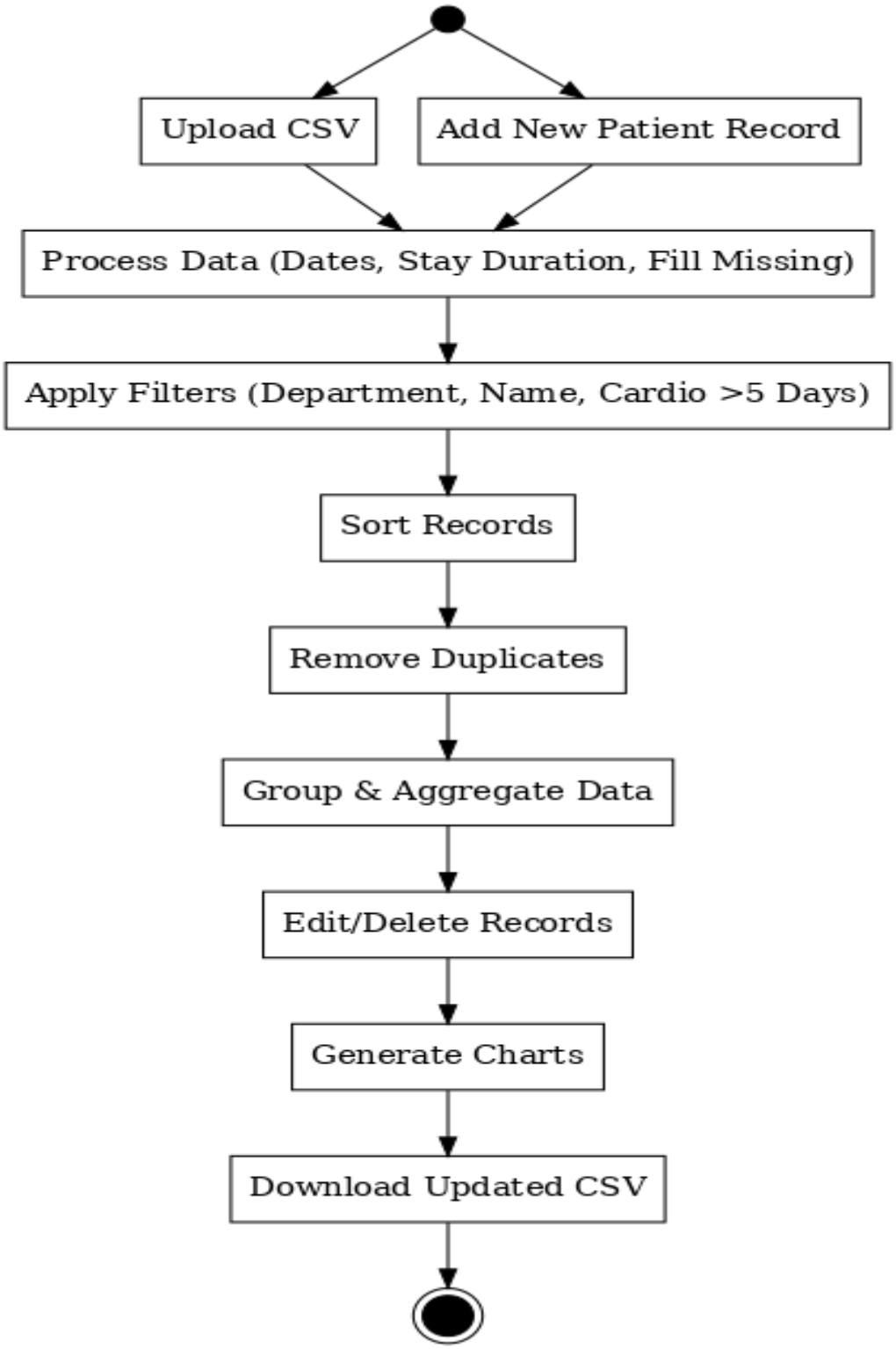
UML Class Diagram:



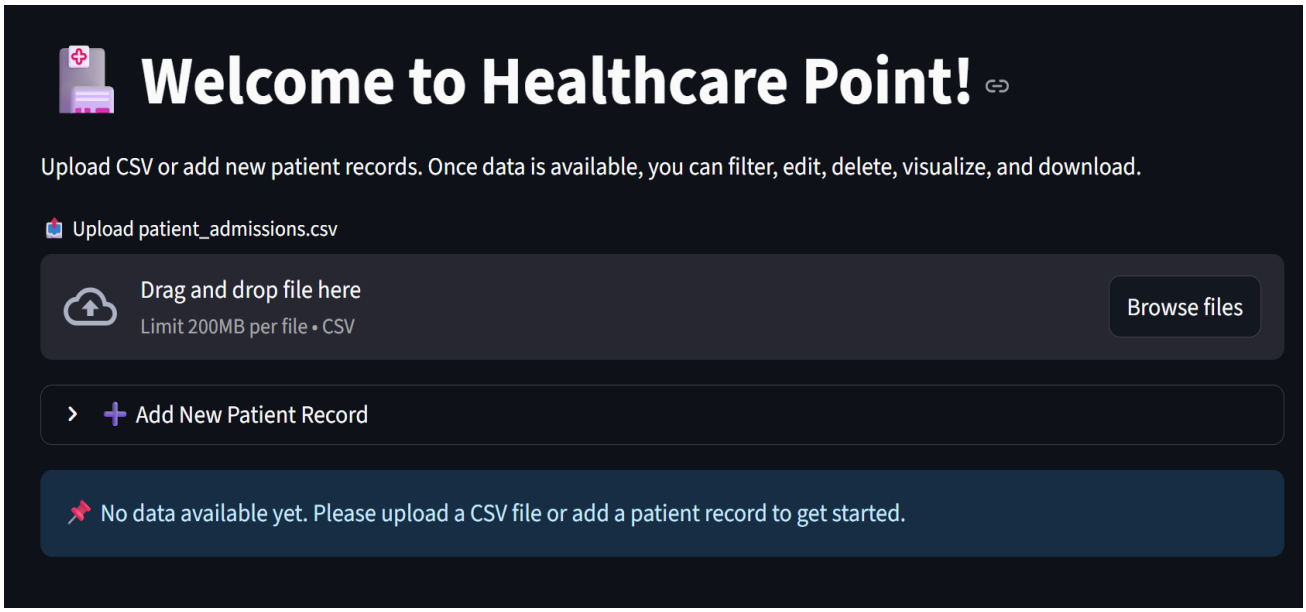
Case Diagram:


Upload Data
Manage Records
Hospital Staff
Filter Data
System
(Streamlit + Pandas)
View Charts
Export Dataset
Clean Data

Flow Chart:

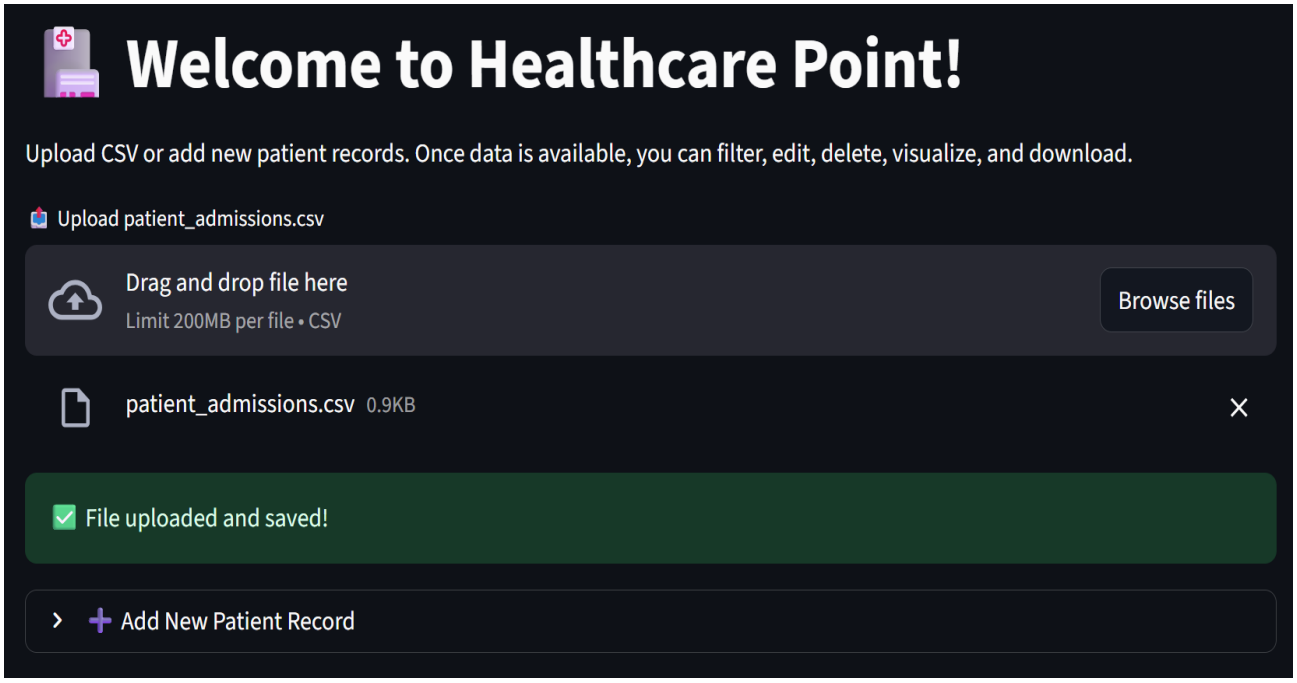

Upload CSV
Add New Patient Record
Process Data (Dates, Stay Duration, Fill Missing)
Apply Filters (Department, Name, Cardio >5 Days)
Sort Records
Remove Duplicates
Group & Aggregate Data
Edit/Delete Records
Generate Charts
Download Updated CSV

Interface:

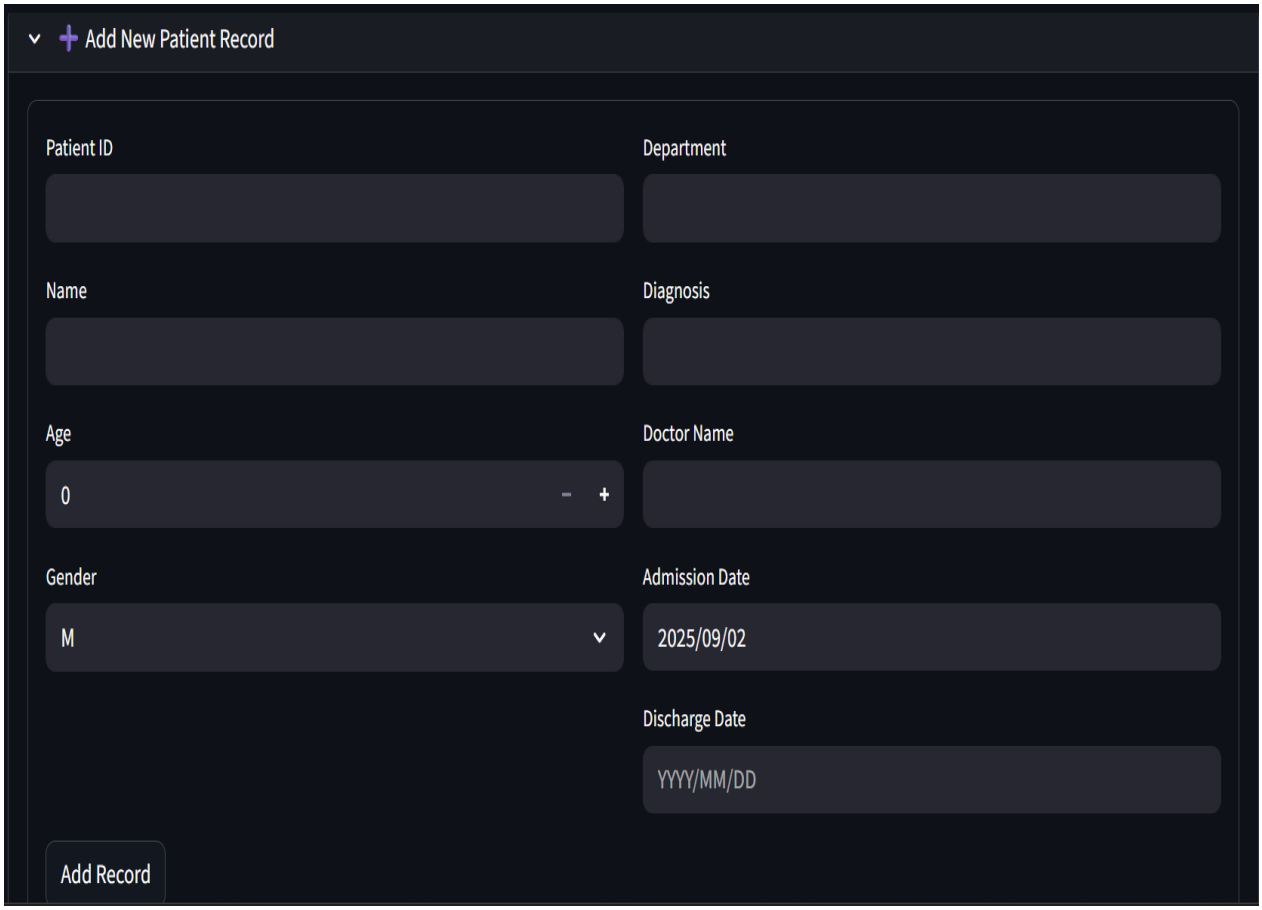


- The **Healthcare Point interface** acts as the central hub for patient admission management.
- It is designed with simplicity and accessibility in mind, enabling users to efficiently interact with healthcare data.
- The interface begins with a welcoming dashboard where users can either upload existing patient admission records in CSV format or manually add new entries through a structured form.
- This dual functionality ensures flexibility in data entry, supporting both bulk uploads and individual record creation.
- The interface also guides users clearly by displaying alerts when no data is available, prompting them to take the next step.
- Once data is provided, it becomes the foundation for further operations such as filtering, editing, deleting, and visualizing patient records.
- The interface emphasizes ease of use with drag-and-drop file upload, dedicated buttons for adding records, and a clean layout, making it intuitive for healthcare staff with minimal technical expertise.
- Ultimately, this design streamlines data handling, reduces manual errors, and provides a reliable workflow for managing admissions effectively.

Screenshots of Output with Explanation:



- When users choose to upload the .csv file then it says File uploaded successfully!
- And as per the details saved in .csv, code starts to run according to the requirements.



- When user chooses to add patient details manually then they need to update it here, all the details updated here will be added on.

Group & Aggregate Insights

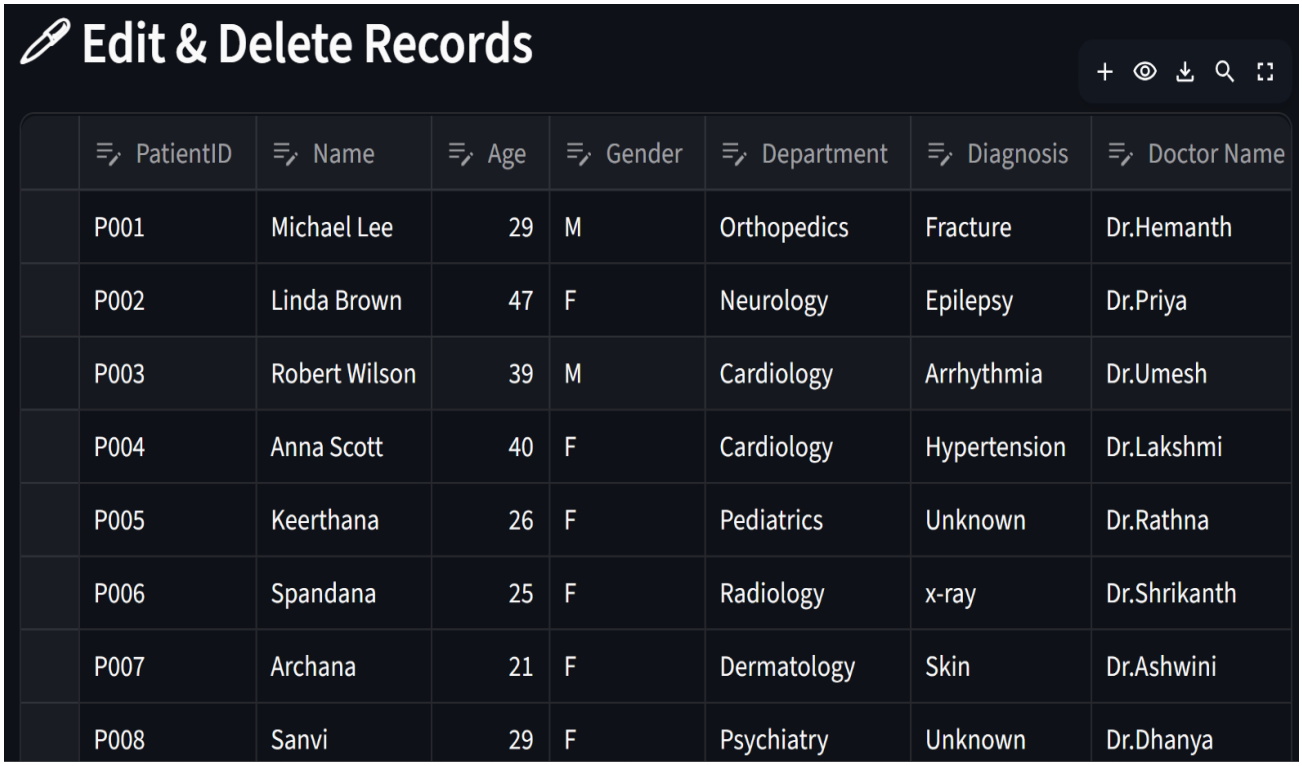
Number of Admissions per Department

	Department	Admissions
0	Cardiology	2
1	Dermatology	1
2	Neurology	1
3	Orthopedics	1
4	Pediatrics	1
5	Psychiatry	1

Average Stay Duration per Department

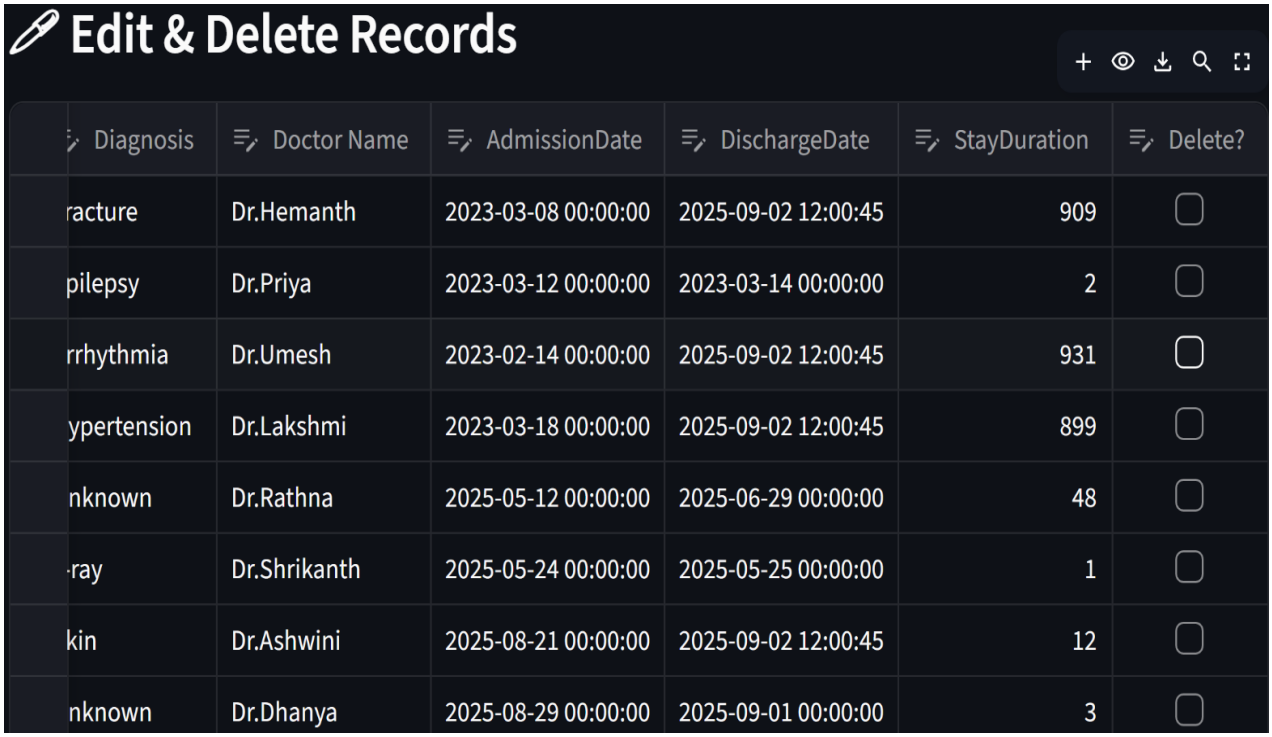
	Department	AvgStayDuration
0	Cardiology	915
1	Dermatology	12
2	Neurology	2
3	Orthopedics	909
4	Pediatrics	48
5	Psychiatry	3

- The analysis shows the **number of admissions per department**, with Cardiology recording the highest (2 admissions) and other departments having 1 admission each.
- The **average stay duration** varies significantly across departments, ranging from very short (2 days in Neurology) to extremely long (915 days in Cardiology).
- Dermatology and Psychiatry have relatively short stays (12 and 3 days), while Paediatrics shows a moderate duration of 48 days.
- Extremely high durations in Cardiology (915 days) and Orthopaedics (909 days) may highlight cases of chronic care or possible data inconsistencies.
- These insights provide hospitals with a clear view of **departmental workloads, patient turnover rates, and resource utilization**, supporting better operational planning.

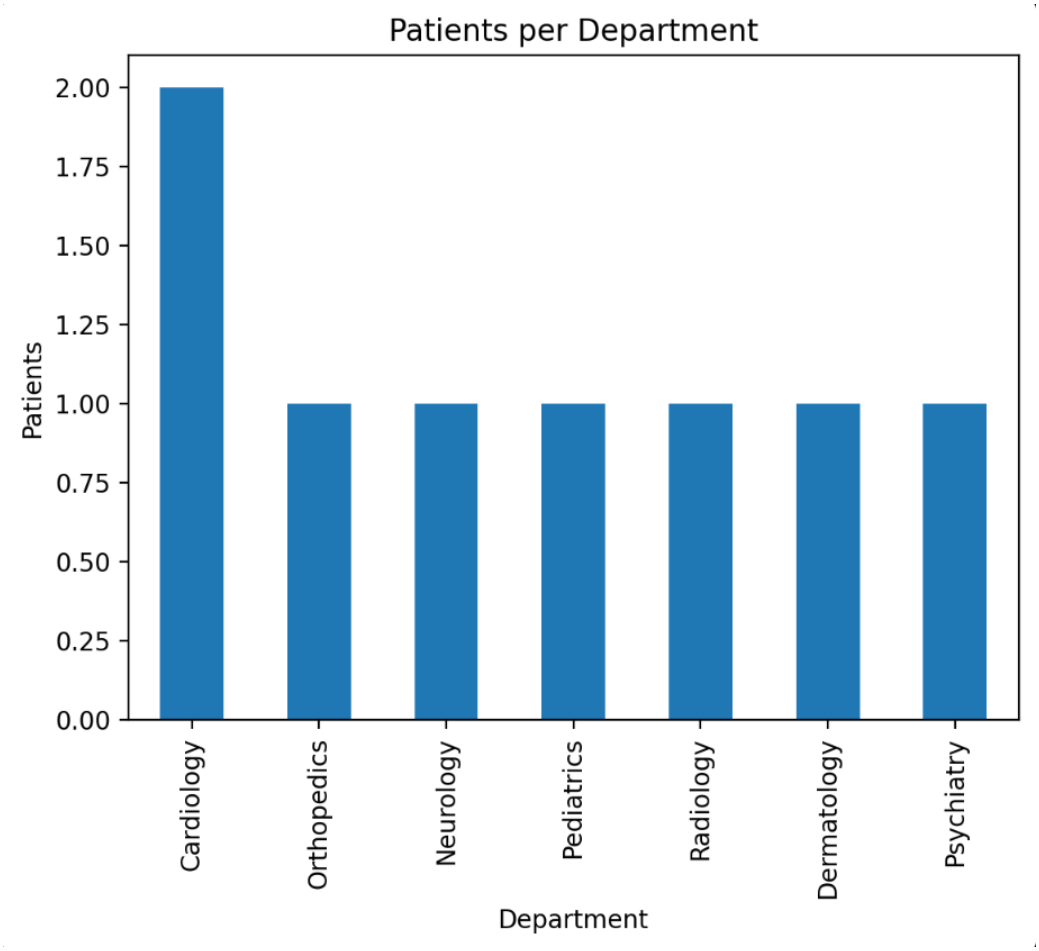


Edit & Delete Records

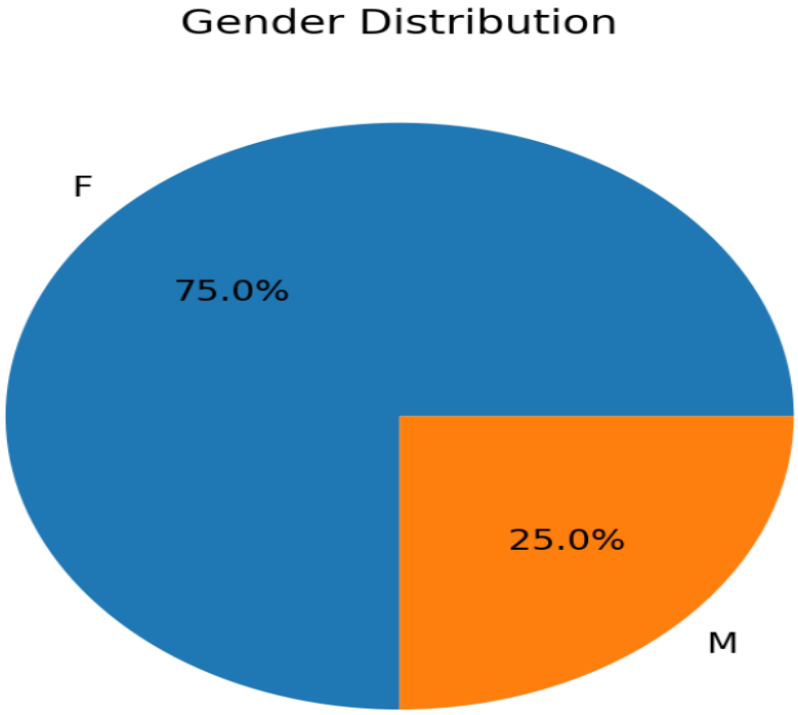
PatientID	Name	Age	Gender	Department	Diagnosis	Doctor Name
P001	Michael Lee	29	M	Orthopedics	Fracture	Dr.Hemanth
P002	Linda Brown	47	F	Neurology	Epilepsy	Dr.Priya
P003	Robert Wilson	39	M	Cardiology	Arrhythmia	Dr.Umesh
P004	Anna Scott	40	F	Cardiology	Hypertension	Dr.Lakshmi
P005	Keerthana	26	F	Pediatrics	Unknown	Dr.Rathna
P006	Spandana	25	F	Radiology	x-ray	Dr.Shrikanth
P007	Archana	21	F	Dermatology	Skin	Dr.Ashwini
P008	Sanvi	29	F	Psychiatry	Unknown	Dr.Dhanya



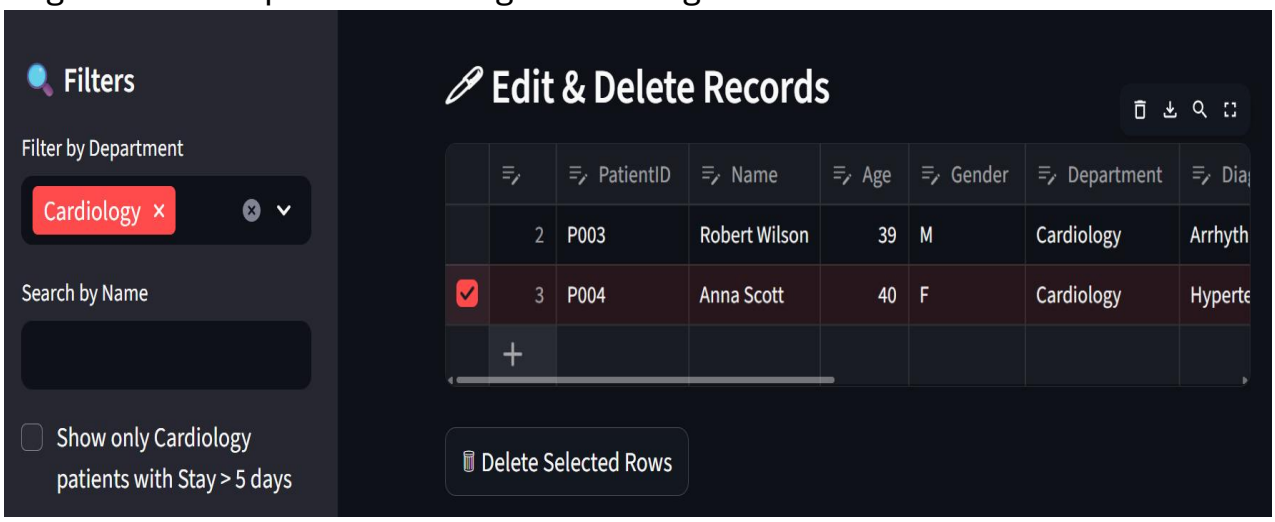
- You can also Edit/ Delete the rows that you needed to be edited and deleted if not required, respectively.
- You can see the whole data that has been uploaded with department, diagnosis, doctor name, admission date, discharge date and with stay duration.
- If Discharge date is not mentioned inside the data, then it takes Today's date and time.
- And Stay Duration is also calculated through Discharge Date – Admission Date which gives Stay Duration of patients.



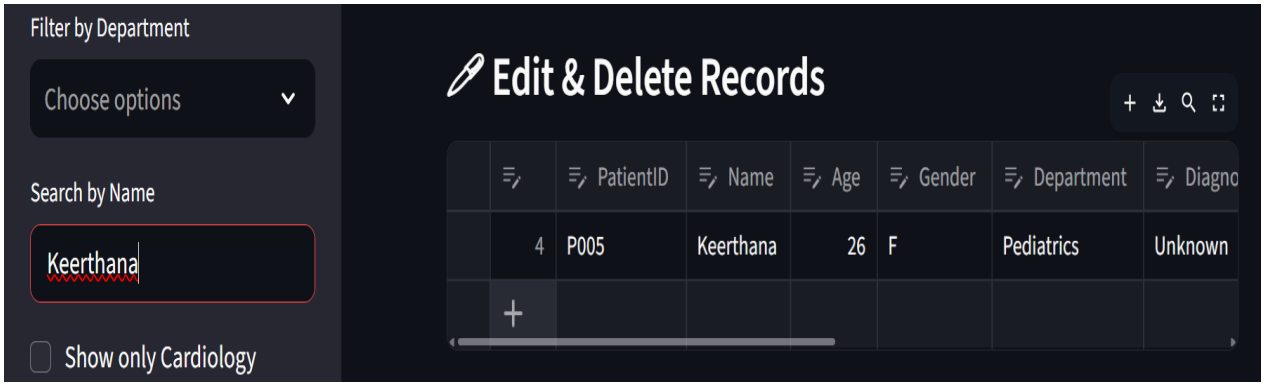
- This above Analytics and Visualizations shows of each patient with their department where large number of patients belongs to.



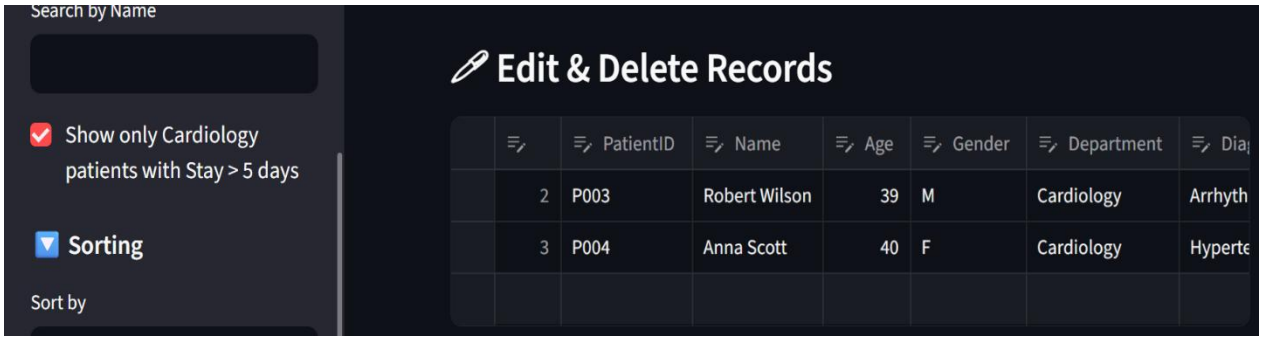
- This above Analytics and Visualizations shows of each patient Gender where large number of patients belongs to which gender.



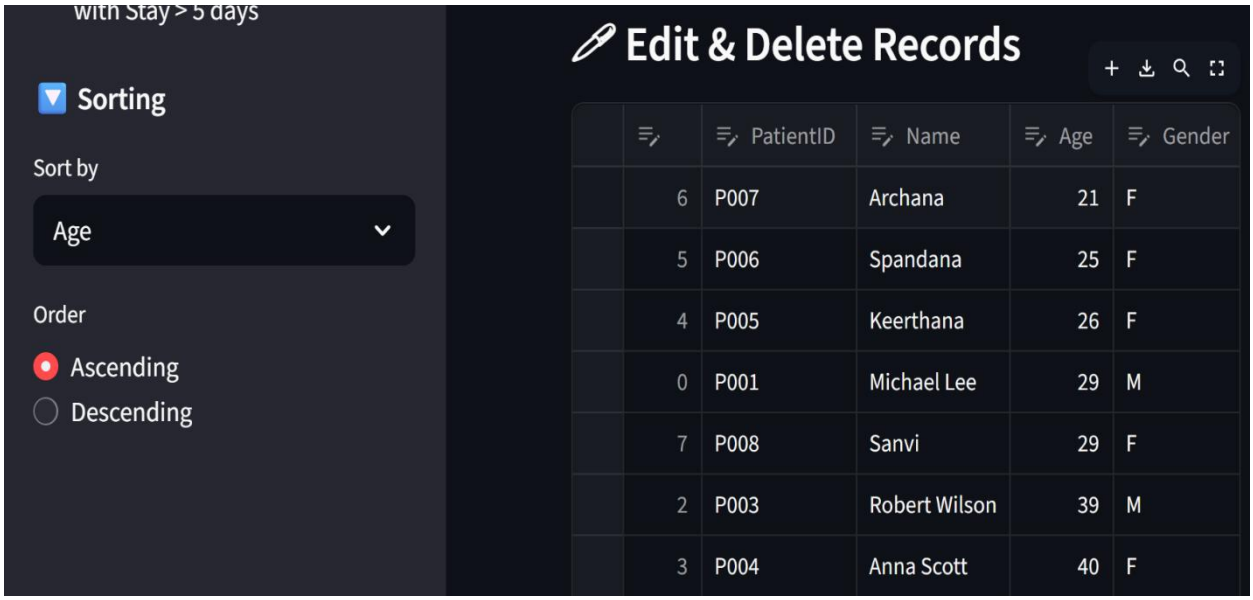
- Here, the staff members can filter the data based on the department they choose.
- So that there will no other confusions that at what department which patients are admitted.

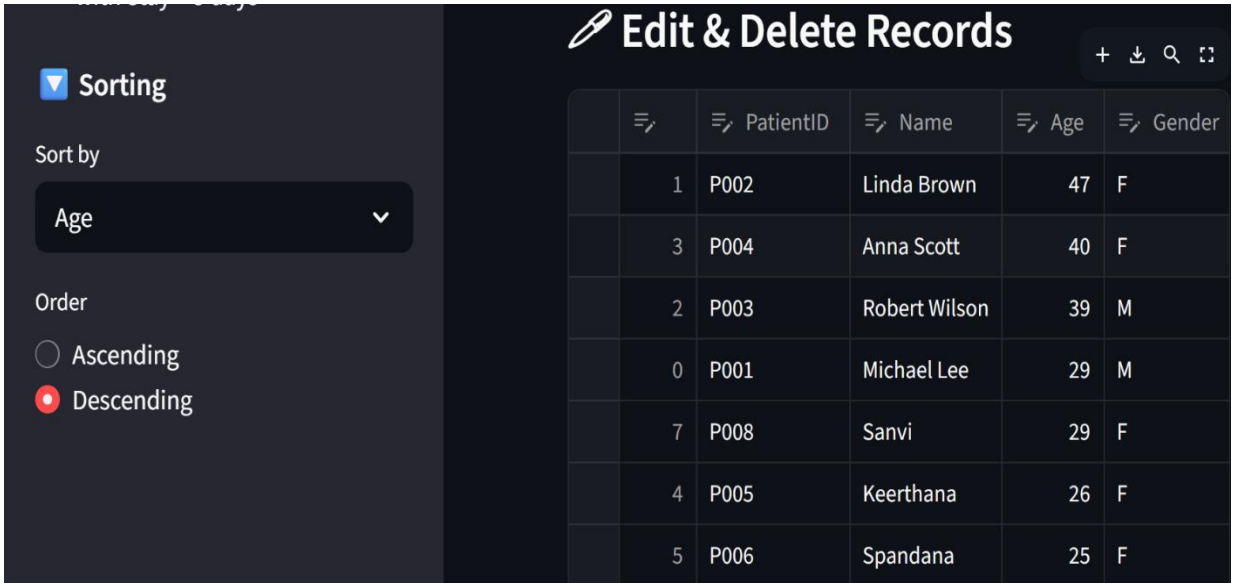


- Here, the staff members can filter the data based on their Name search button and they can easily find the patient and their details.

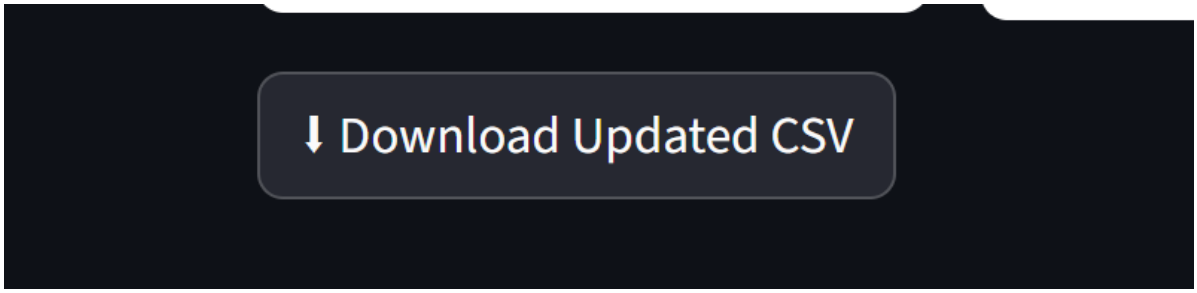


- This checks where the cardiology department patients are staying more than 5 days with their complete medical history.





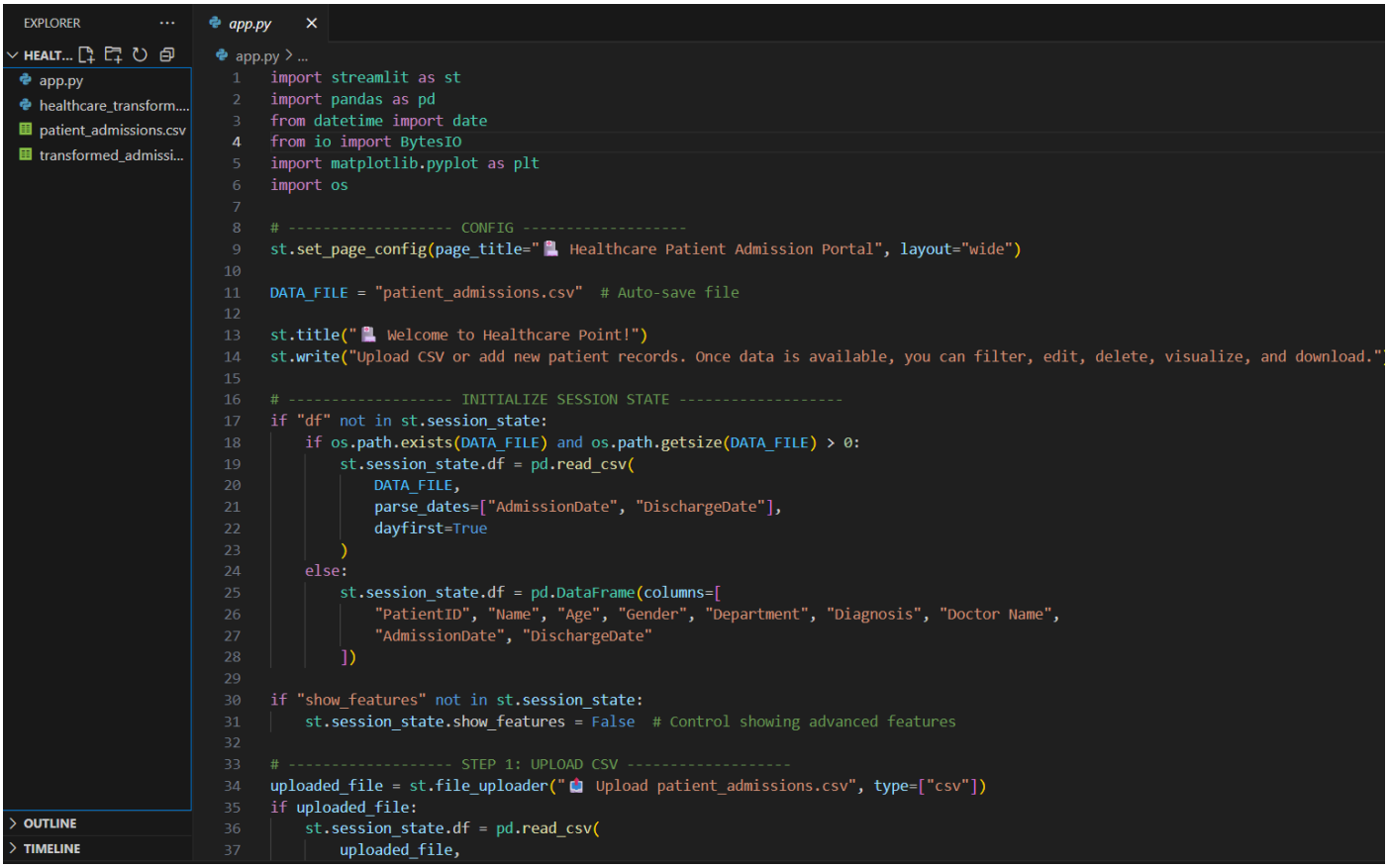
- In the above pictorial, the staff members can sort the patients based on their details.
- Here there is an example, where the data are sorted through Age.



- Finally, you can download the updated .csv file with all the necessary requirements are added.

Code Explanation:

app.py



- First, it prepares the app page layout and assigns a default CSV file that will store patient records.
- It shows a title and a short description so users know what the app does (upload or manage patient data).
- Next, it checks whether patient data already exists:
- If a saved CSV file is found, it loads the data into memory so the app can immediately display and work with it.
- If no file exists, it creates an empty data table with all the required patient details (like ID, name, age, department, admission/discharge dates, etc.).
- A flag is also created to control whether advanced features (like extra analytics or filtering) should be shown.
- Finally, it allows the user to upload a new CSV file. If a file is uploaded, the data inside it is read and stored in memory, replacing any old data.

app.py > ...

```python
    # Ensure datetime conversion
    df["AdmissionDate"] = pd.to_datetime(df["AdmissionDate"], errors="coerce", dayfirst=True)
    df["DischargeDate"] = pd.to_datetime(df["DischargeDate"], errors="coerce", dayfirst=True)

    # Fill missing diagnosis
    df["Diagnosis"].fillna("Unknown", inplace=True)

    # Only fill missing DischargeDate with today (don't overwrite valid ones)
    df["DischargeDate"] = df["DischargeDate"].fillna(pd.Timestamp.today())

    # Create StayDuration only if AdmissionDate is valid
    df["StayDuration"] = (df["DischargeDate"] - df["AdmissionDate"]).dt.days
    df["StayDuration"] = df["StayDuration"].fillna(0).astype(int)

    st.markdown("---")
    st.subheader("📑 Patient Admission Records & Features")

    # ------------------- STEP 6: FILTERS -------------------
    st.sidebar.header("🔍 Filters")
    dept_filter = st.sidebar.multiselect("Filter by Department", df["Department"].dropna().unique())
    name_filter = st.sidebar.text_input("Search by Name")
    cardio_filter = st.sidebar.checkbox("Show only Cardiology patients with Stay > 5 days")

    filtered_df = df.copy()
    if dept_filter:
        filtered_df = filtered_df[filtered_df["Department"].isin(dept_filter)]
    if name_filter:
        filtered_df = filtered_df[filtered_df["Name"].str.contains(name_filter, case=False, na=False)]
    if cardio_filter:
        filtered_df = filtered_df[(filtered_df["Department"] == "Cardiology") & (filtered_df["StayDuration"] > 5)]

    # ------------------- STEP 8: SORTING -------------------
    st.sidebar.subheader("🔽 Sorting")
    sort_column = st.sidebar.selectbox("Sort by", filtered_df.columns)
    sort_order = st.sidebar.radio("Order", ["Ascending", "Descending"])
    filtered_df = filtered_df.sort_values(by=sort_column, ascending=(sort_order == "Ascending"))
```

-Datetime Conversion

- Converts Admission Date and Discharge Date columns into proper datetime format.
- Ensures dates are valid and interprets them in day-first format.

-Fill Missing Diagnosis

- Replaces any empty values in the diagnosis column with "Unknown".

-Handle Missing Discharge Dates

- If a Discharge Date is missing, it is filled with today's date.
- Existing valid values are not overwritten.

-Calculate Stay Duration

- Creates a new column Stay Duration = Discharge Date – Admission Date (in days).
- Missing values are replaced with 0.

- Ensures values are stored as integers.

-Add Section Header in Stream lit

- Displays a section sub header “Patient Admission Records & Features” in the UI.

-Sidebar Filters

- Adds an option to filter by Department (multi-select).
- Adds a Name search box.
- Adds a Cardiology checkbox to filter patients in Cardiology with stay duration > 5 days.

-Apply Filters on Data

- Makes a copy of the dataset.
- If department filter is used → keeps only selected departments.
- If name filter is used → keeps rows where Name contains the input.
- If cardiology filter is used → keeps only cardiology patients with stay > 5.

-Sorting Options

- Adds a dropdown to choose which column to sort by.
- Adds a radio button to choose order (Ascending / Descending).
- Sorts the filtered dataset accordingly.

```
filtered_df = filtered_df.sort_values(by=sort_column, ascending=(sort_order == "Ascending"))

# ------------------- STEP 9: REMOVE DUPLICATES -------------------
before_rows = len(filtered_df)
filtered_df = filtered_df.drop_duplicates(subset=["PatientID", "AdmissionDate"], keep="first")
after_rows = len(filtered_df)
if before_rows != after_rows:
    st.warning(f"⚠ Removed {before_rows - after_rows} duplicate records (based on PatientID & AdmissionDate)

# ------------------- STEP 7: GROUP & AGGREGATE -------------------
st.subheader("📊 Group & Aggregate Insights")
col1, col2 = st.columns(2)
with col1:
    st.write("**Number of Admissions per Department**")
    dept_count = df.groupby("Department")["PatientID"].count().reset_index(name="Admissions")
    st.dataframe(dept_count, use_container_width=True)
with col2:
    st.write("**Average Stay Duration per Department**")
    avg_stay = df.groupby("Department")["StayDuration"].mean().reset_index(name="AvgStayDuration")
    st.dataframe(avg_stay, use_container_width=True)

# ------------------- STEP 7 (continued): EDIT & DELETE -------------------
st.subheader("✏ Edit & Delete Records")
filtered_df["Delete?"] = False
edited_df = st.data_editor(filtered_df, num_rows="dynamic", use_container_width=True, key="editor")

if st.button("🗑 Delete Selected Rows"):
    ids_to_delete = edited_df[edited_df["Delete?"]]["PatientID"].tolist()
    st.session_state.df = st.session_state.df[~st.session_state.df["PatientID"].isin(ids_to_delete)]
    st.session_state.df.to_csv(DATA_FILE, index=False)
    st.success("✅ Selected rows deleted & saved!")
else:
    updated_rows = edited_df.drop(columns=["Delete?"])
    st.session_state.df.update(updated_rows)
    st.session_state.df.to_csv(DATA_FILE, index=False)
```

-Normalize date columns

-Convert Admission Date and Discharge Date to real datetimes (day first=True, errors="coerce") so they’re safe for math and filtering.

- Fill missing diagnosis
- Replace blank Diagnosis values with "Unknown" to avoid nulls in the UI and summaries.

-Impute missing discharge dates

-If Discharge Date is missing, fill it with today’s date (simulating an ongoing admission) without touching already-valid dates.

-Compute length of stay

Create Stay Duration = (Discharge Date – Admission Date). days. Any None become 0, stored as integers for sorting/aggregation.

-Show section heading
Render the sub header “Patient Admission Records & Features” to mark the start of the interactive area.

-Provide sidebar filters

- Department multiselect
- Name text search (case-insensitive)
- Cardiology + stay > 5 days quick filter (checkbox)

-Apply filtering logic

-Work on a copy (filtered) and progressively subset it based on the chosen department(s), name search, and the cardiology +stay rule.

-Offer sorting controls
-Let the user pick any column to sort by and choose Ascending/Descending; sort the filtered dataset accordingly.

-Remove duplicate admissions
-Drop duplicate rows using ["Patient ID”, “Admission Date"] as the key (keep the first). If anything was removed, show a warning with the count.

-Department-level insights (two panels)

- Admissions per Department: group by("Department") ["Patient ID"]. count ()
- Average Stay per Department: group by("Department”) ["Stay Duration"]. mean ()
  Both are displayed in side-by-side data frames.

-Delete selected rows (if triggered)
-When the “Delete Selected Rows” button is pressed, collect Patient IDs where ---- Delete == True, remove those rows from s t. session _ state. d f, persist to CSV, and show a success message.

-Result: The dataset is cleaned, enriched (with Stay Duration), filterable, sortable, de-duplicated, provides aggregated insights, and supports inline edit/delete with all changes saved back to the patient_admissions.csv file.

📖 Bibliography

1. Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55
2. McKinney, W. (2010). *Data structures for statistical computing in Python*. Proceedings of the 9th Python in Science Conference, 56–61.
3. Stream lit Inc. (2025). *Stream lit Documentation*. Retrieved from https://docs.streamlit.io
4. The Pandas Development Team. (2025). *Pandas Documentation*. Retrieved from https://pandas.pydata.org/docs/
5. Python Software Foundation. (2025). *Python 3.11 Documentation*. Retrieved from https://docs.python.org/3/
6. World Health Organization. (2024). *Global report on health data systems and capacity, 2024*. WHO Press.
7. IBM. (2023). *Data Cleaning in Healthcare: Best Practices and Methods*. IBM Knowledge Centre.