A Database Management System for Electronic Health Records in ICU

Dandi Chen, Jifan Gao, Yujia Shi
{dandi.chen, jgao85, shi235}@wisc.edu
Department of Biostatistics and Medical Informatics, University of Wisconsin-Madison

1 Introduction

1.1 Motivation

Patients in intensive care units (ICU) are closely monitored due to severe or life-threatening illnesses or injuries. Huge volumes of data are collected during monitoring, such as vital signs, laboratory tests, diagnosis, care plan, radiology images, medical history, medications, demographics and administrative information. With promising artificial intelligence (AI) techniques, several healthcare applications based on these electronic health records (EHRs) are becoming more flexible and intelligent [4]. Therefore, this application is aimed at developing a management application for patients and physicians on top of aforementioned monitored ICU data. The main focus is a healthcare Web-based application. The functionality of the application can be divided into management and visualization, where the former one grants users permissions to insert, delete, modify and query the data based on user types, and the latter one generates statistics plots based on users' choice.

1.2 Application Description

For this application, it is important for us to implement a database since we need to store the patient related data and offer various functionalities for users to manage the data. Patient entity has many different attributes, like lab test results, diagnosis, medications, which are essential for diagnosis and treatment. Our application can provide a platform for doctors not only to examine patients' information, but also visualize information to figure out some latent relationship between different attributes. Medical confidentiality needs to be concerned as well, and a database will offer a place to store all these data and only the person with permission can have access to them. It guarantees the privacy and security.

The application focuses on five core features:

- **Insertion**. Add new patients, ICU stay and records to the database.
- **Deletion**. Delete all the records associated with a patient or a single ICU unit stay in the database.
- **Modification**. Modify the data stored in the database based on provided patient ICU stay record ID.
- **Query**. Query patient ICU stay records in the database.
- **Visualization**. Visualize descriptive statistics for around 200,000 patients in the whole dataset.

Besides, three types of users with different level of permissions are defined to manage the data: **admin**, **physician** and **patient**. **Admin** holds the highest authority to insert, delete, query, visualize all records both in patient and disease view. It is created by default in the database. **Physician** accounts can be created by **admin** to insert, modify, query, visualize all records, and add patients or ICU stay record that are not existed in the database. But they cannot delete records. **Patients** are able to query their own ICU stay records.

1.3 Report Organization

The report is organized as follows:

Section 1 contains background, motivation, application description, organization, task assignment of the project and the software we used. It aims at a high-level understanding of the problem that we would like to explore and our target of the project.

Section 2 intends to introduce implementation details. Overall implementation framework, details of the dataset, entity–relationship (ER) diagram, relational schema, phototype and evaluation are all included. It summarizes the steps we went over during the implementation and all the technique details are covered.

Section 3 warps up what we have learned in this project.

1.4 Task Assignment

Table 1 shows our project schedule and group members' roles. We communicate via weekly face-to-face meetings.

Dates	**Expected deliverables**
7/8 - 7/14	data cleaning (Dandi) ER diagram modeling (Dandi) specific functionalities and visualization (Jifan and Yujia) specific user interface (Jifan, Yujia and Dandi)
7/15 - 7/21	relational schema (Dandi, Yujia and Jifan) SQL query level 1 (Jifan and Yujia)
7/22 - 7/28	SQL query level 2 (Jifan) SQL insert/modify/delete (Yujia) visualization (Dandi)
7/29 - 8/4	SQL query level 3 (Jifan and Yujia) user interface (Jifan and Yujia) visualization (Dandi)
8/5 - 8/11	user interface (Jifan, Yujia and Dandi) final report (Dandi, Jifan and Yujia) video recording (Dandi, Jifan and Yujia)

Table 1: Project schedule and task assignment

1.5 Software Platforms and Languages

We use SQLite, Django, Python to manage the database, and HTML, CSS, JavaScript for user interface (UI) implementation. A CSS Framework Bulma[1] is included to make UI looks nicely. A detailed demonstration is shown in Fig. 1.

[1] https://bulma.io/

2 Implementation

2.1 System Architecture

As shown in Fig. 1, our application contains front-end and back-end components. We provide an interface for users to interact with our data, which is provided and managed by the database. Corresponding programming languages are specified as well.

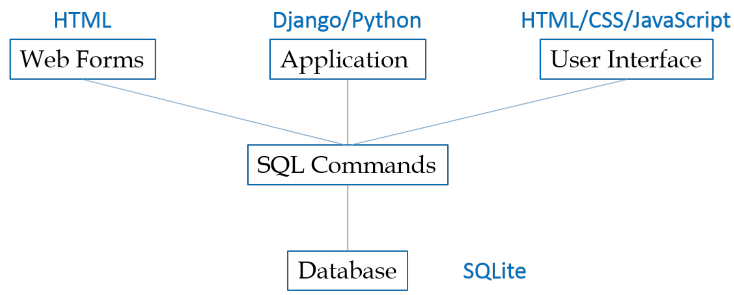


Figure 1: System architecture

A detailed functionality architecture is indicated in Fig. 2 and this is the main guide we referred to when implementing the application. As mentioned in section 1.2, five core features are realized and three types of users are defined for different level of permissions to better simulate the real world case. ICU stays are what we focus on since it conveys most information from the original dataset. Most entities are highly correlated with ICU stays and its unique identifier has been widely used as foreign keys to link most tables. More details about the dataset can be found in section 2.2.

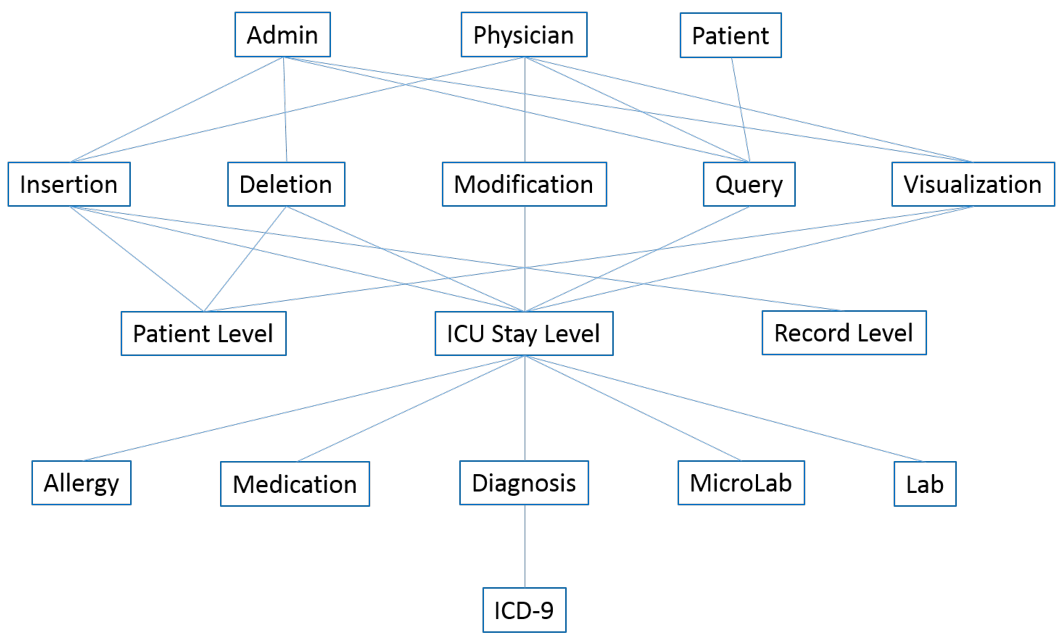


Figure 2: Functionality architecture

Since each patient may be admitted to ICU for multiple times and each ICU stay may contain multiple records from many perspectives, such as allergy, medications, diagnosis and lab tests, and corresponding time intervals might vary a lot, we built this patient–ICU stay–record hierarchical framework to manage the data efficiently. Five core features are able to operate the data via three levels as well but they do have their own focus. Insertion provides interfaces for all three levels while deletion cares about first two levels, as deleting via patient level will delete all the records associated with specific patient, and deleting via ICU stay level will delete all the records associated with specific ICU stay. Modification and query concentrates on ICU stay level as ICU stay is a proper unit for a whole medical profile for a patient. It covers a series of data that can describe the disease and relevant information completely for a given period. Patient's personal information, such as age, height, weight may change in multiple ICU stays, and those are linked directly to ICU stay for better description. Visualization pays attention to both patient level and ICU stay level for two views. The former one creates a blueprint for the patient population by defining multiple cohorts based on different filters and conditions, such as a group of people with same disease and similar demographics. The latter one groups all ICU stays by emphasizing medical records, regardless of the bias of individual differences in patients.

2.2 Dataset

The data we used is from the eICU Collaborative Research Database (eICU-CRD) [3]. It is a collection of comma separated value (CSV) files [2] produced by Philips Healthcare in partnership with the Laboratory for Computational Physiology at Massachusetts Institute of Technology (MIT), related to patients as part of the Philips eICU program [3]. Compared with other EHR databases such as Medical Information Mart for Intensive Care (MIMIC) database [2] that is collected from a single medical center, eICU-CRD is collected from a large number of hospitals located within the United States, which means it may cover more variety in patients' demographics. eICU-CRD contains 31 deidentificated tables and records of 139,367 unique patients admitted by 335 units at 208 hospitals in the United States between 2014 and 2015, to meet Health Insurance Portability and Accountability Act (HIPAA) [1]. Demographics, vital signs, laboratory measurements, medications, care plan, medical history, diagnosis and treatments are all included. Instead of setting up a database management system with the entire eICU-CRD dataset, we extracted a few necessary information from following six tables of eICU-CRD to realize the designed functionality finally.

1) **Allergy**. 251,949 records. 120MB. Details of patient allergies, including
   - *allergyID*. Unique allergy identifier.
   - *allergyName*. Picklist of the allergy, such as penicillins, pollen and shellfish.
   - *allergyType*. Type of allergy: Drug or Non Drug.
   - *allergyOffset*. Number of minutes from unit admit time that the allergy was detected.
2) **Diagnosis**. 2,710,672 records. 270MB. Active patient diagnosis that were documented in the ICU stay, including
   - *diagnosisID*. Unique diagnosis identifier.
   - *ICD9Code*. ICD-9 code[4] for the diagnosis.
   - *diagnosisOffset*. Number of minutes from unit admit time that the diagnosis was entered.
3) **Lab**. 39,132,531 records. 2GB. Laboratory measurements for patient derived specimens, which have been mapped to a standard set of measurements, including
   - *labID*. Unique lab test identifier.
   - *labType*. The type of lab that is represented in the values: 1 for chemistry, 2 for drug level, 3 for hemo, 4 for misc, 5 for non-mapped, 6 for sensitive, 7 for ABG lab.
   - *labName*. Picklist of the lab, such as CPK, troponin-I, RBC, HCO3 and Total CO2. It is hospital specific.
   - *labResult*. Numeric value of the lab test.
   - *labMeasureNameSystem*. The measurement name of the lab, such as mm Hg, mmol/L and mEq/L.
   - *labResultOffset*. Number of minutes from unit admit time that the lab value was drawn.
4) **MicroLab**. 16,996 records. 1MB. Microbiology information from patient derived specimens, including

[2] https://eicu-crd.mit.edu/

[3] https://www.usa.philips.com/healthcare/product/HC865325ICU/eicu-program-telehealth-for-the-intensive-care-unit

[4] https://www.cdc.gov/nchs/icd/icd9.htm

- *microLabID*. Unique microbiology test identifier.
- *cultureSite*. Picklist of site name from where the culture was takensuch as Wound, Drainage Fluid, Sputum, Expectorated, Nasopharynx.
- *organism*. Picklist of organism found, such as Staphylococcus aureus, Pseudomonas aeruginosa and no growth.
- *sensitivityLevel*. Picklist of sensitivity level of antibiotic: Intermediate, Resistant, or Sensitive.
- *antibiotic*. Picklist of antibiotic used, such as ceftazidime and aztreonam.
- *cultureTakenOffset*. Number of minutes from unit admit time that the culture was taken.

5) **Medication**. 7,301,853 records. 604MB. Active medication orders for patients, including
   - *medicationID*. Unique drug identifier.
   - *routeAdmin*. Picklist of route of administration for the drug, such as IV (intravenous), IV - continuous infusion (intravenous) and PO (oral).
   - *drugName*. Name of selected drug.
   - *dosage*. The dosage of the drug, e.g.1 mcg/kg/min.
   - *antibiotic*. Picklist of antibiotic used, such as ceftazidime and aztreonam.
   - *drugOffset*. Number of minutes from unit admit time that the drug was ordered.
6) **Patient**. 200,859 records. 46MB. Demographic and administrative information regarding the patient and their unit or hospital stay, including
   - *patientUnitStayID*. Unique ICU Stay identifier.
   - *uniquePID*. Unique patient identifier.
   - *age*. In full years. Ages are grouped into '>89' if the age is larger than 89.
   - *gender*. Gender of a patient: Male, Female, Unknown, Other, NULL.
   - *height*. Admission height of the patient in centimeters.
   - *weight*. Admission weight of the patient in kilograms.
   - *ethnicity*. Picklist ethnicity of the patient: Asian, Caucasian, African American, Native American, Hispanic, Other/Unknown, NULL.
   - *unitType*. Picklist of the unit type, such as MICU, SICU and Med-Surg ICU.
   - *unitDischargeOffset*. Number of minutes from unit admit time that the patient was discharged from the unit.
   - *unitDischargeStatus*. Patient's condition upon leaving the unit: Alive, Expired, or NULL.

Six tables can be linked by identifiers such as *patientunitstayid*, which uniquely identifies a single ICU stay, and *uniquePID* which uniquely identifies a patient.

2.3 Entity–relationship Diagram

We extended above six tables into ten entity sets, including three type of user profile entity sets and other seven data related entity sets. Attributes are demonstrated in ER diagram (Fig. 3) accordingly.

Our application is aimed at offering service to patient, physician and administrator. These profile entity sets all have Email and password attributes, while the primary key of patient profile entity set is ***uniquePID***. As for physician and administrator profile entity sets, the primary keys are ***physicianUserID*** and ***adminnUserID***.

Based on original eICU-CRD dataset and our main focus, we kept **PatientStay** (from **Patient** table), **Diagnosis** (from **Diagnosis** table), **MicroLab** from **MicroLab** table), **Lab** (from **Lab** table), **Medication** (from **Medication** table), **Allergy** (from **Allergy** table), since all of them are highly related to ICU stays but not just a patient, due to their dynamics. We also monitor their *offset* to quantize the dynamics. More details about dynamics have been explained in checkpoint 3 section 1.1.

ICD-9 code has been pulled out from **Patient** table for an independent entity set, as it is possible to be diagnosed with multiple diseases in a single ICU stay or for a single patient. The foreign key of aforementioned *offset* related entity sets is ***patientUnitStayID***, except for **ICD-9** and **PatientStay**. It

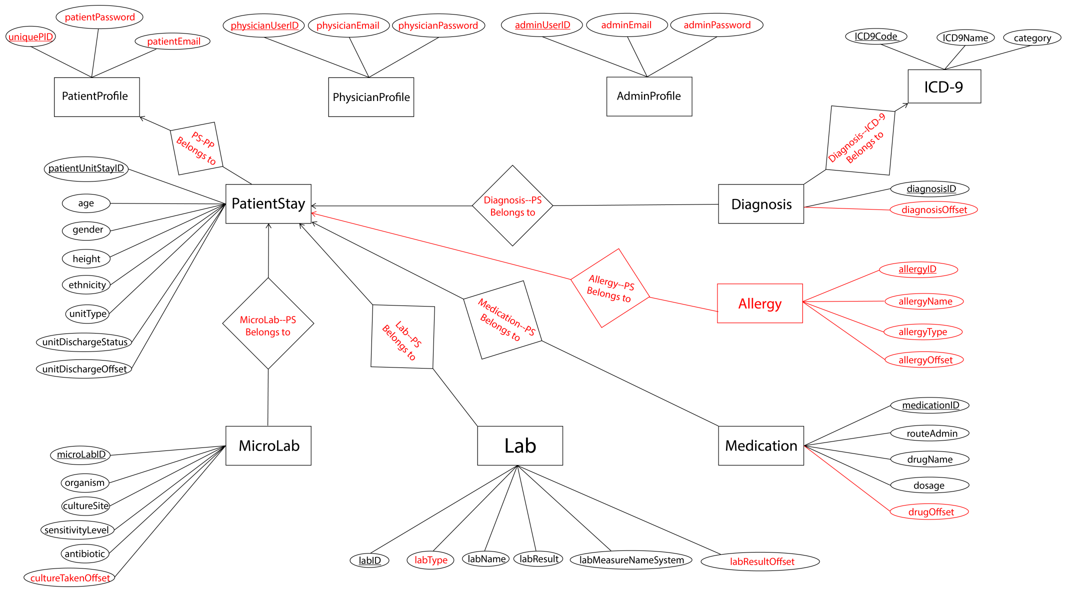


Figure 3: Entity–relationship Diagram

indicates the existence of referential integrity constraints. Furthermore, key constraints exist in our design as well, since all these entities have a unique ID attribute to identify themselves.

The design model consists of seven similar relationship sets. **Allergy**, **Diagnosis**, **Medication**, **Lab** and **MicroLab** entity sets all belong to **PatientStay** entity set, while these relationships are all many-to-one, since each patient unit stay can have multiple lab tests or be diagnosed with different diseases. The relationship between **Diagnosis** and **ICD-9** entity sets is many-to-one as well, taking the fact that different patients may be diagnosed with same disease. In addition, each patient may be admitted to ICU for multiple times and each ICU stay will be assigned a ***patientUnitStayID***. It implies the relationship between **PatientStay** entity set and **Patient profile** entity set is many-to-one.

2.4 Relational Model

PatientStay (patientUnitStayID, age, gender, height, ethnicity, unitType, unitDischargeStatus, unitDischargeOffset, uniquePID)

MicroLab (microLabID, orginism, cultureSite, sensitivityLevel, antobiotic, cultureTakenOffset, patientUnitStayID)

Lab (labID, labType, labName, labResult, labMesureNameSystem, labResultOffset, patientUnitStayID)

Medication (medicationID, routeAdmin, drugName, dosage, drugOffset, patientUnitStayID)

Diagnosis (diagnosisID, diagnosisOffset, ICD9Code, patientUnitStayID)

ICD-9 (ICD9Code, ICD9Name, category)

Allergy (allergyID, allergyName, allergyType, allergyOffset, patientUnitStayID)

PatientProfile (uniquePID, patientPassword, patientEmail)

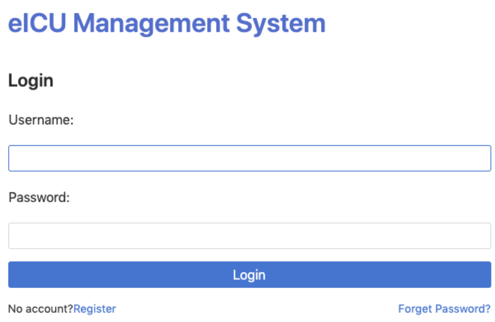
PhysicianProfile (physicianUserID, physicianPassword, physicianEmail)

AdminProfile (adminUserID, adminPassword, adminEmail)

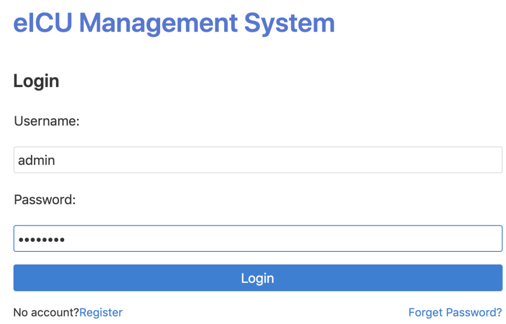
All primary keys are underlined accordingly. ***ICD9Name*** is also a key of **ICD-9**.

2.5 Prototype

The login page is default when opening our application, as shown in Fig. 4. It also provides the functionalities to reset password and register (i.e. *Forget Password* and *register* in Fig. 4).



(a) Default login page



(b) Login as administrator

Figure 4: Login page

Administrator account is created by default and physician accounts need to be created by the administrator. Django does provide its own administration site to manager users and other properties (Fig. 5). In the administration system, administrator can set authority for each user (including three groups, doctor, patient and admin), as shown in (Fig. 6).

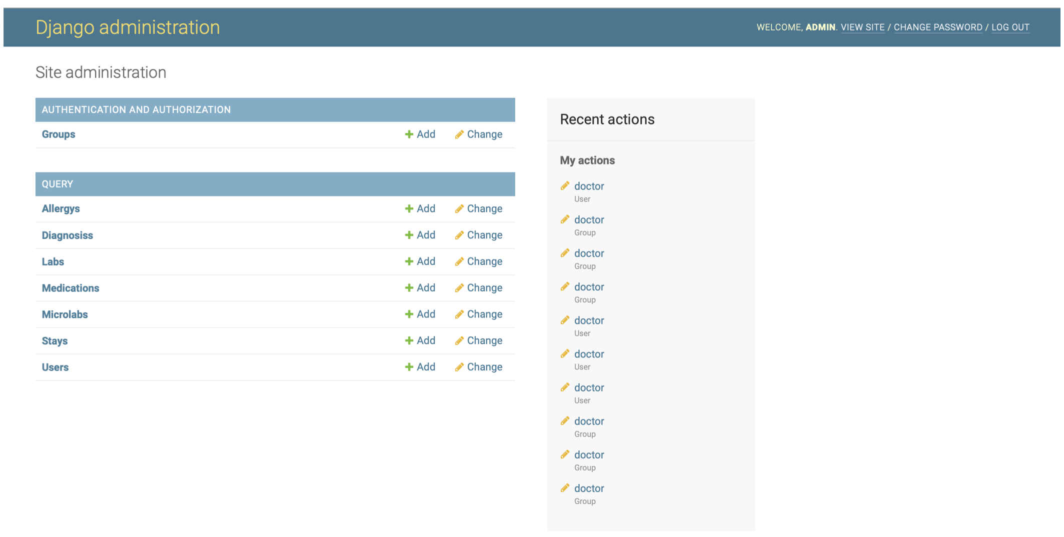


Figure 5: Django administration page

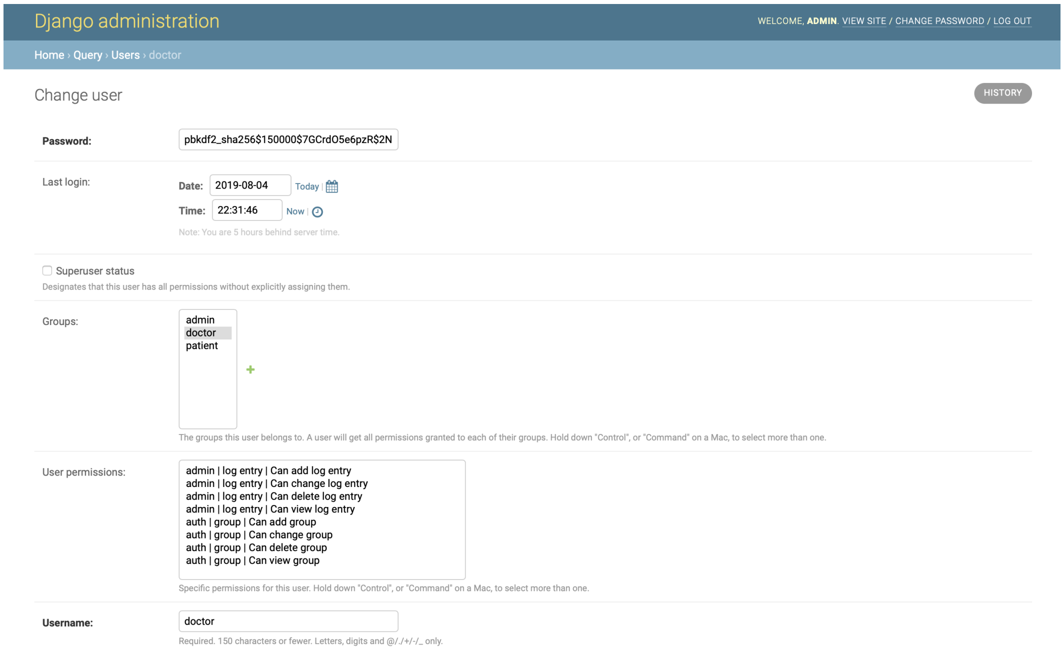


Figure 6: Create physician accounts by administrator

Patient accounts can be created by themselves with given patient ID, which are associated with ***uniquePID*** in **patient** table (Fig. 7). It will be used as username. Password of all types of accounts can be reset via email (Fig. 8).

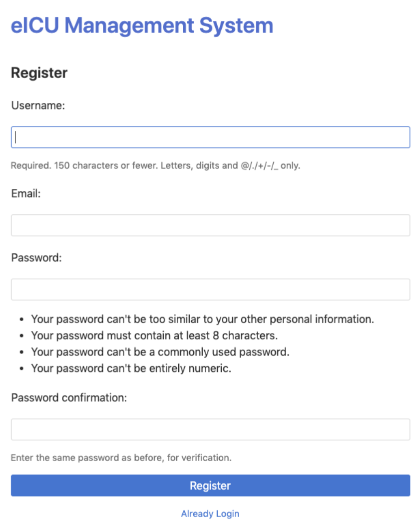


Figure 7: Register page

2.5.1 Insertion

After entering username and password of the admin, four tabs *Insertion*, *Deletion*, *Query* and *Statistics* become available. Admin can add patient, add a single ICU stay for an existing patient, or add a new record for an existing ICU stay for an existing patient, as demonstrated in Fig. 10.

By clicking the button *Add a Patient*, a new patient ID will be created and this ID is the uniquely identifier for the patient (i.e. ***uniquePID***). The ID is automatically generated and a confirmation message will pop up when it is done. Then, a new ICU stay can be created for the existing patient.

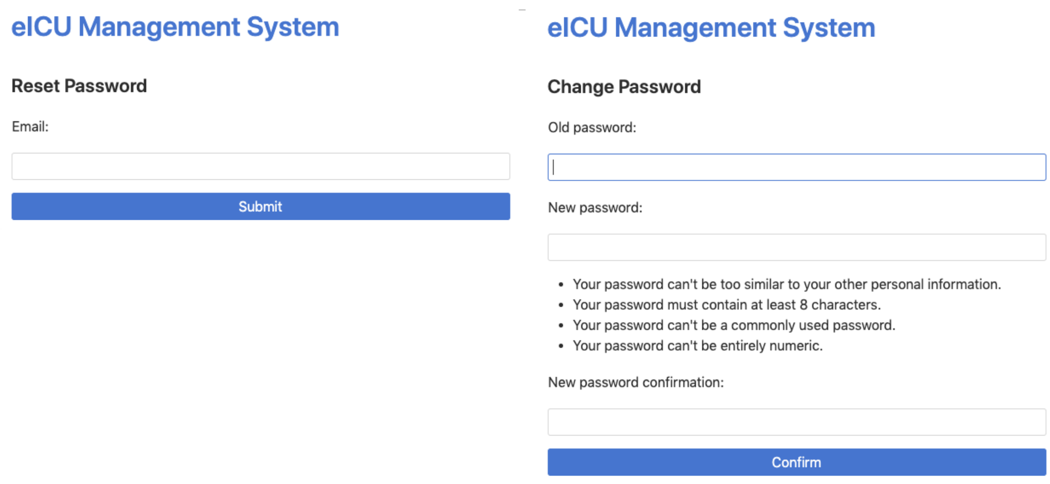


Figure 8: Reset password page

It is possible to add record to an existing ICU stay. A unique ICU stay (i.e. ***patientUnitStayID***) is required and a few data fields are able to be inserted via a drop-down list. All relevant data fields correspond to our entity sets **Allergy**, **Diagnosis**, **Lab**, **MicroLab** and **Medication**(Fig. 9). After selecting an entity set, a new page will be switched to list all corresponding attributes. If records are filled by clicking the button *Add*, a new pop-up message is shown to verify records are added successfully. If button *Return* is clicked, this insertion operation will be cancelled.

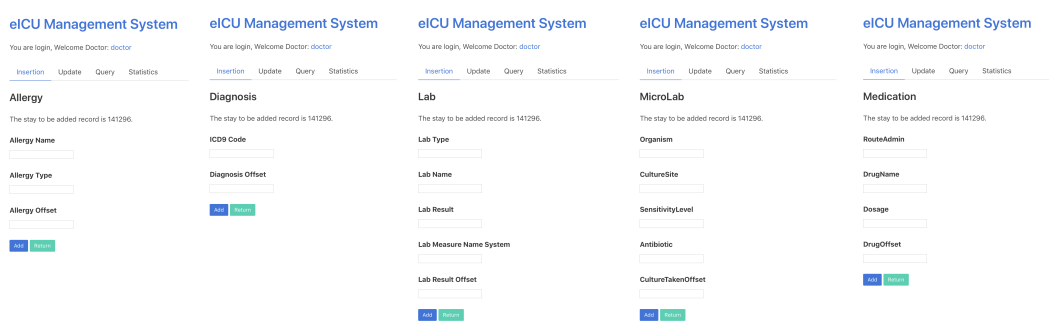


Figure 9: Add a record to an existing ICU stay

Physician account has the same insertion permission as admin. However, patient accounts cannot insert any records.

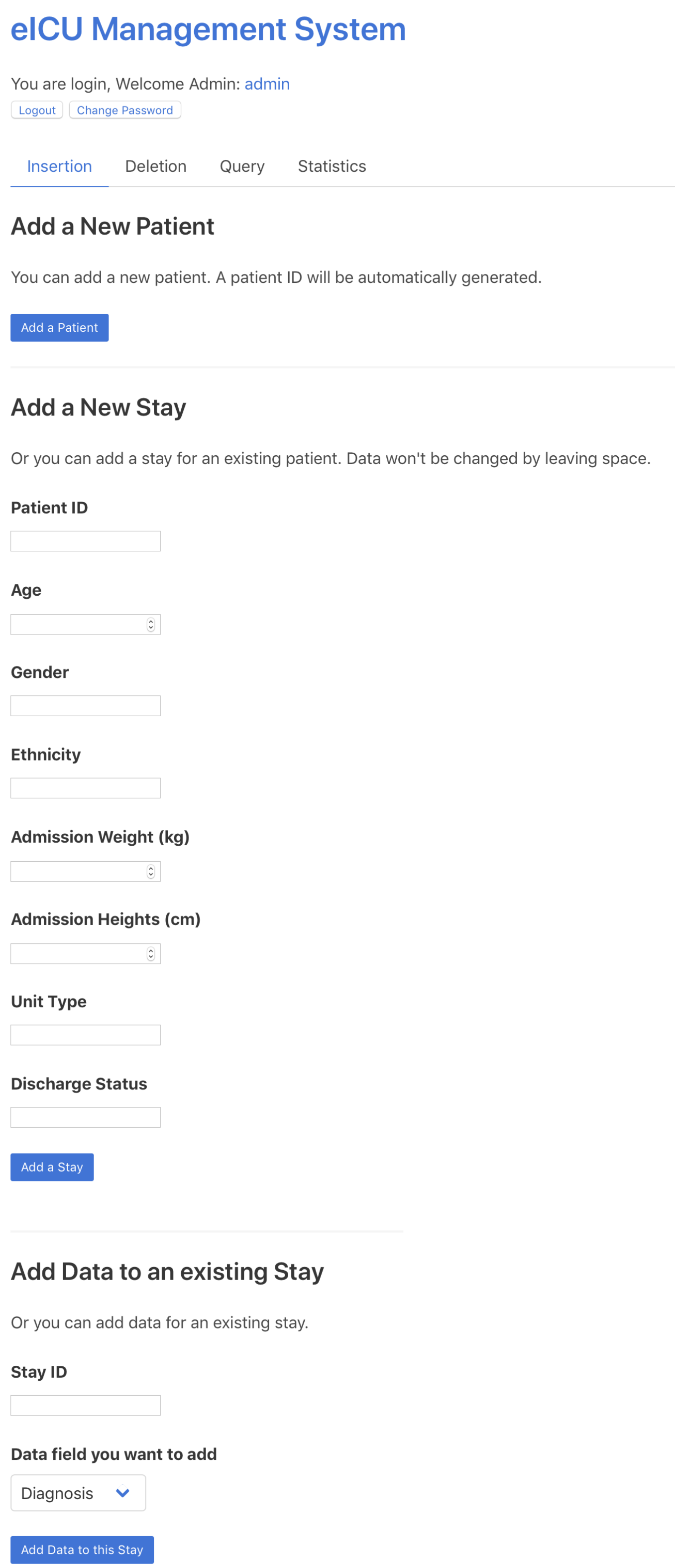


Figure 10: Insertion page

2.5.2 Modification

Physicians are able to modify a single record for an existing ICU stay. After entering unique ***patientUnitStayID***, a few data fields are included in the drop-down list to show available records that can be modified (Fig. 11).

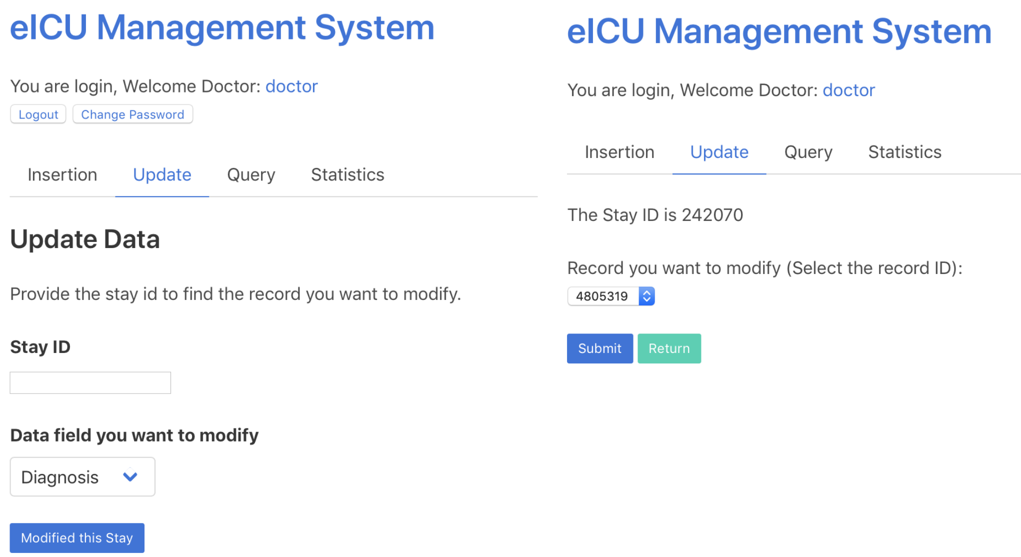


Figure 11: Modification page

Fig. 12 demonstrates the case to modify specifying records using corresponding unique identifiers ***allergyID***, ***diagnosisID***, ***LabID***, ***microLabID***, ***medicationID***. Once the specific data field is selected, all available attributes are shown for modification.

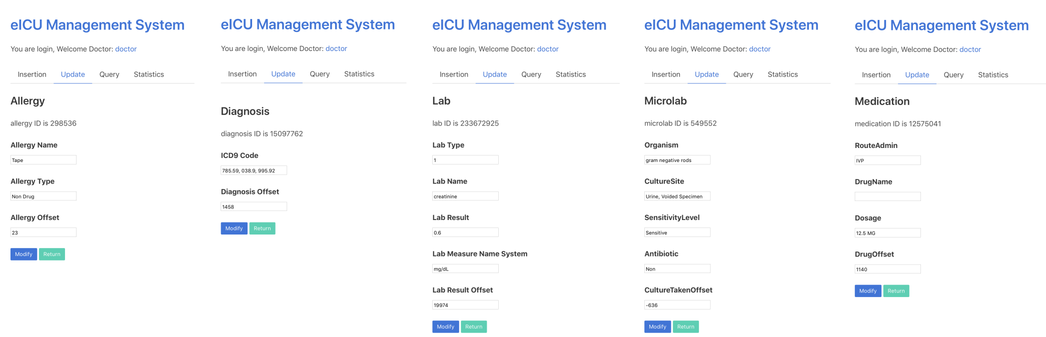
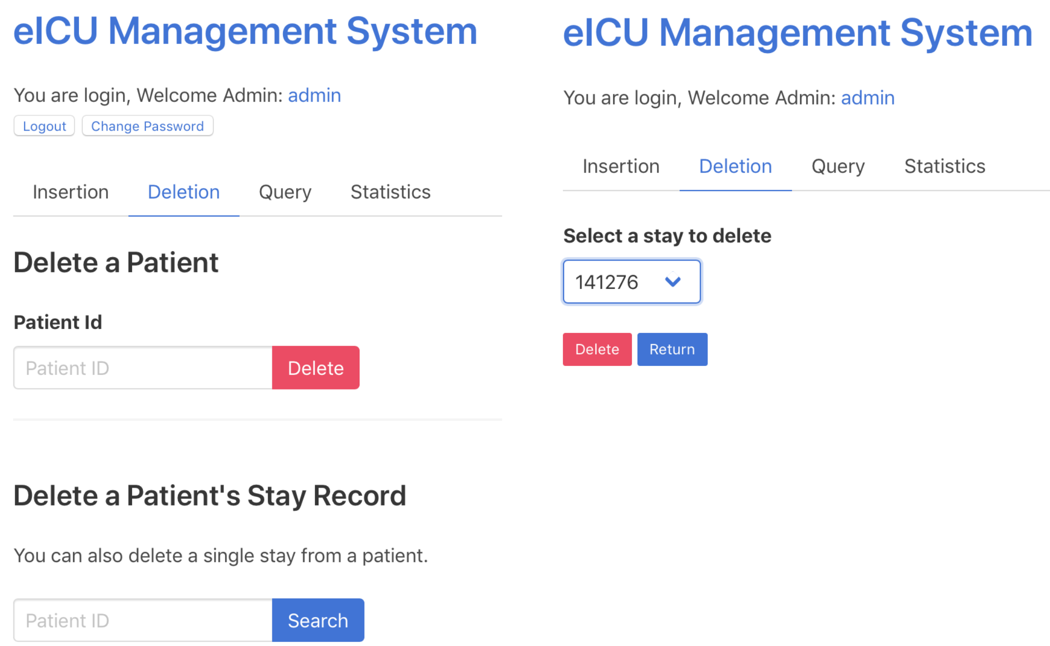


Figure 12: Specify a ICU stay to modify records

2.5.3 Deletion

Administrator can delete all the records associated with a single patient via ***uniquePID*** (Fig. 13a). It is also possible to delete a single ICU stay of a patient (Fig. 13b). Similar as *Insertion* tab, a confirmation message will be displayed to verify the deletion operation is completed.



(a) An example to delete a single patient

(b) An example to delete a single ICU stay

Figure 13: Deletion page

2.5.4 Query

Query is open to all types of account, i.e. administrator, physicians and patients. It can be realized by entering patient identifier ***uniquePID*** and ICU stay identifier ***patientUnitStayID***. Then all the records associated with it will be listed. Fig. 14 provides an example when logining as patient.

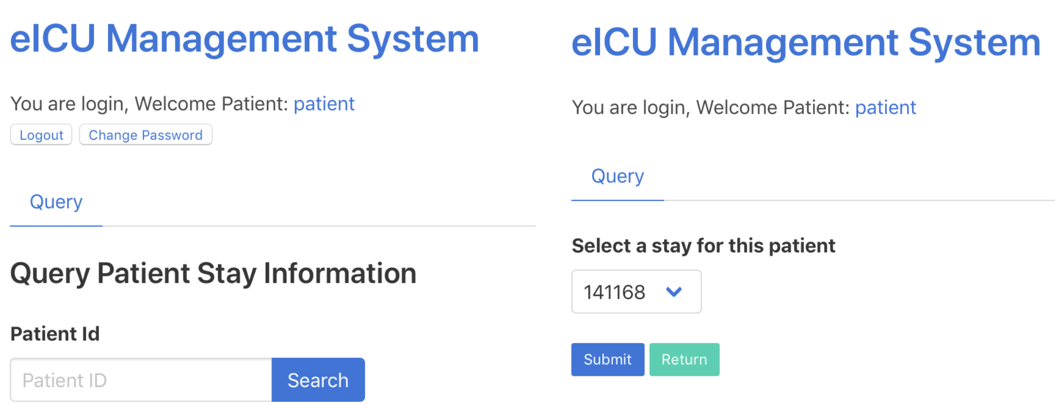
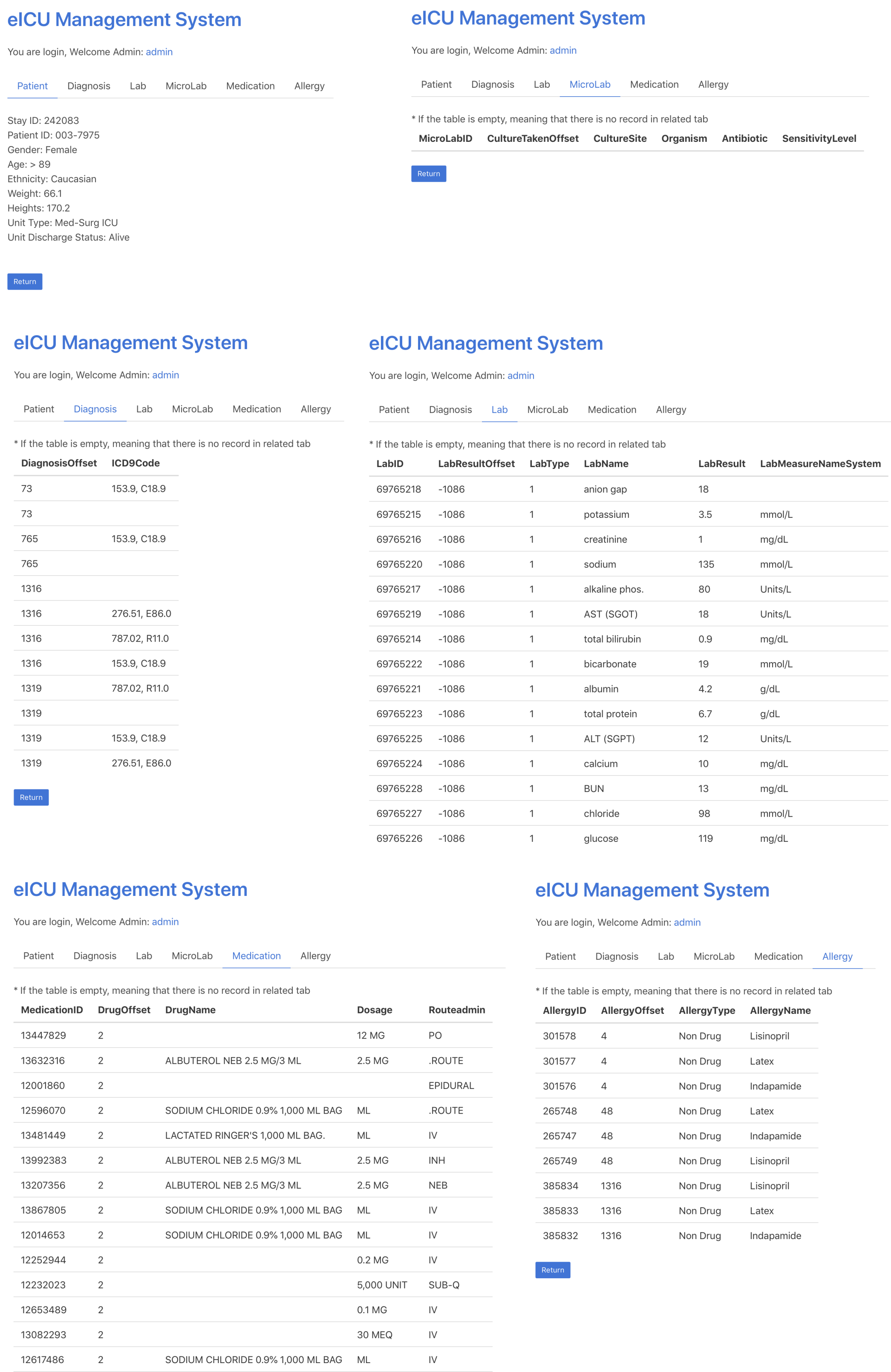


Figure 14: Query page

When loging as admin or doctor, patient's personal information as well as specifying records are able to be queried using corresponding unique identifiers, just like modification feature. Switching to the proper tab will show all the relevant information associated with the single ICU stay, as demonstrated in Fig. 15.



eICU Management System

You are login, Welcome Admin: admin

Patient | Diagnosis | Lab | MicroLab | Medication | Allergy

Stay ID: 242083
Patient ID: 003-7975
Gender: Female
Age: > 89
Ethnicity: Caucasian
Weight: 66.1
Heights: 170.2
Unit Type: Med-Surg ICU
Unit Discharge Status: Alive

Return

eICU Management System

You are login, Welcome Admin: admin

Patient | Diagnosis | Lab | MicroLab | Medication | Allergy

* If the table is empty, meaning that there is no record in related tab

MicroLabID	CultureTakenOffset	CultureSite	Organism	Antibiotic	SensitivityLevel

Return

eICU Management System

You are login, Welcome Admin: admin

Patient | Diagnosis | Lab | MicroLab | Medication | Allergy

* If the table is empty, meaning that there is no record in related tab

DiagnosisOffset	ICD9Code
73	153.9, C18.9
73	
765	153.9, C18.9
765	
1316	
1316	276.51, E86.0
1316	787.02, R11.0
1316	153.9, C18.9
1319	787.02, R11.0
1319	
1319	153.9, C18.9
1319	276.51, E86.0

Return

eICU Management System

You are login, Welcome Admin: admin

Patient | Diagnosis | Lab | MicroLab | Medication | Allergy

* If the table is empty, meaning that there is no record in related tab

LabID	LabResultOffset	LabType	LabName	LabResult	LabMeasureNameSystem
69765218	-1086	1	anion gap	18	
69765215	-1086	1	potassium	3.5	mmol/L
69765216	-1086	1	creatinine	1	mg/dL
69765220	-1086	1	sodium	135	mmol/L
69765217	-1086	1	alkaline phos.	80	Units/L
69765219	-1086	1	AST (SGOT)	18	Units/L
69765214	-1086	1	total bilirubin	0.9	mg/dL
69765222	-1086	1	bicarbonate	19	mmol/L
69765221	-1086	1	albumin	4.2	g/dL
69765223	-1086	1	total protein	6.7	g/dL
69765225	-1086	1	ALT (SGPT)	12	Units/L
69765224	-1086	1	calcium	10	mg/dL
69765228	-1086	1	BUN	13	mg/dL
69765227	-1086	1	chloride	98	mmol/L
69765226	-1086	1	glucose	119	mg/dL

eICU Management System

You are login, Welcome Admin: admin

Patient | Diagnosis | Lab | MicroLab | Medication | Allergy

* If the table is empty, meaning that there is no record in related tab

MedicationID	DrugOffset	DrugName	Dosage	Routeadmin
13447829	2		12 MG	PO
13632316	2	ALBUTEROL NEB 2.5 MG/3 ML	2.5 MG	.ROUTE
12001860	2			EPIDURAL
12596070	2	SODIUM CHLORIDE 0.9% 1,000 ML BAG	ML	.ROUTE
13481449	2	LACTATED RINGER'S 1,000 ML BAG.	ML	IV
13992383	2	ALBUTEROL NEB 2.5 MG/3 ML	2.5 MG	INH
13207356	2	ALBUTEROL NEB 2.5 MG/3 ML	2.5 MG	NEB
13867805	2	SODIUM CHLORIDE 0.9% 1,000 ML BAG	ML	IV
12014653	2	SODIUM CHLORIDE 0.9% 1,000 ML BAG	ML	IV
12252944	2		0.2 MG	IV
12232023	2		5,000 UNIT	SUB-Q
12653489	2		0.1 MG	IV
13082293	2		30 MEQ	IV
12617486	2	SODIUM CHLORIDE 0.9% 1,000 ML BAG	ML	IV

eICU Management System

You are login, Welcome Admin: admin

Patient | Diagnosis | Lab | MicroLab | Medication | Allergy

* If the table is empty, meaning that there is no record in related tab

AllergyID	AllergyOffset	AllergyType	AllergyName
301578	4	Non Drug	Lisinopril
301577	4	Non Drug	Latex
301576	4	Non Drug	Indapamide
265748	48	Non Drug	Latex
265747	48	Non Drug	Indapamide
265749	48	Non Drug	Lisinopril
385834	1316	Non Drug	Lisinopril
385833	1316	Non Drug	Latex
385832	1316	Non Drug	Indapamide

Return

Figure 15: Specify a ICU stay to query records

2.5.5 Visualization

The application offers an interface to concentrate on disease analysis based on demongraphics filters, i.e. gender, age and ethnicity, as indicated in Fig. 16.

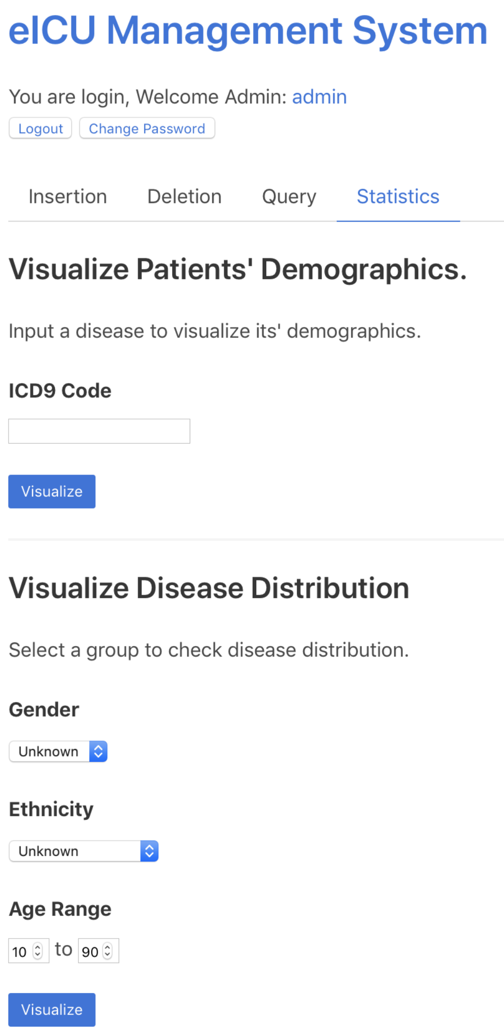


Figure 16: Visualization page

We implemented interactive visualization figures via Google Charts[5]. Fig. 17 shows an example about analyzing *malignant neoplasm of stomach*. Intuitively, an older Caucasian male patient is more likely to have this disease based on simple filters of gender, age and ethnicity. We can also notice that the incidence of 40 to 50 years old has increased rapidly. Therefore, timely and relevant physical examination of this population will accelerate the diagnosis and treatment of this disease to some extent. Top 10 drugs are also available under visualization, which might become foundations of further treatment, or even guide the medication production and medical insurance. Top 10 complicated infections summarized the infections among total 178 ICU stays and these information provides doctors with a reference for treating similar patients, saving time for complex infections and indirectly prolonging the lives of patients.

Fig. 18 verifies the consistency with the statistics provided by American Cancer Society [6] by checking the melanoma, which is one of the deadliest form of skin cancer. Melanoma is response for over

[5]https://developers.google.com/chart/
[6]https://cancerstatisticscenter.cancer.org/!/

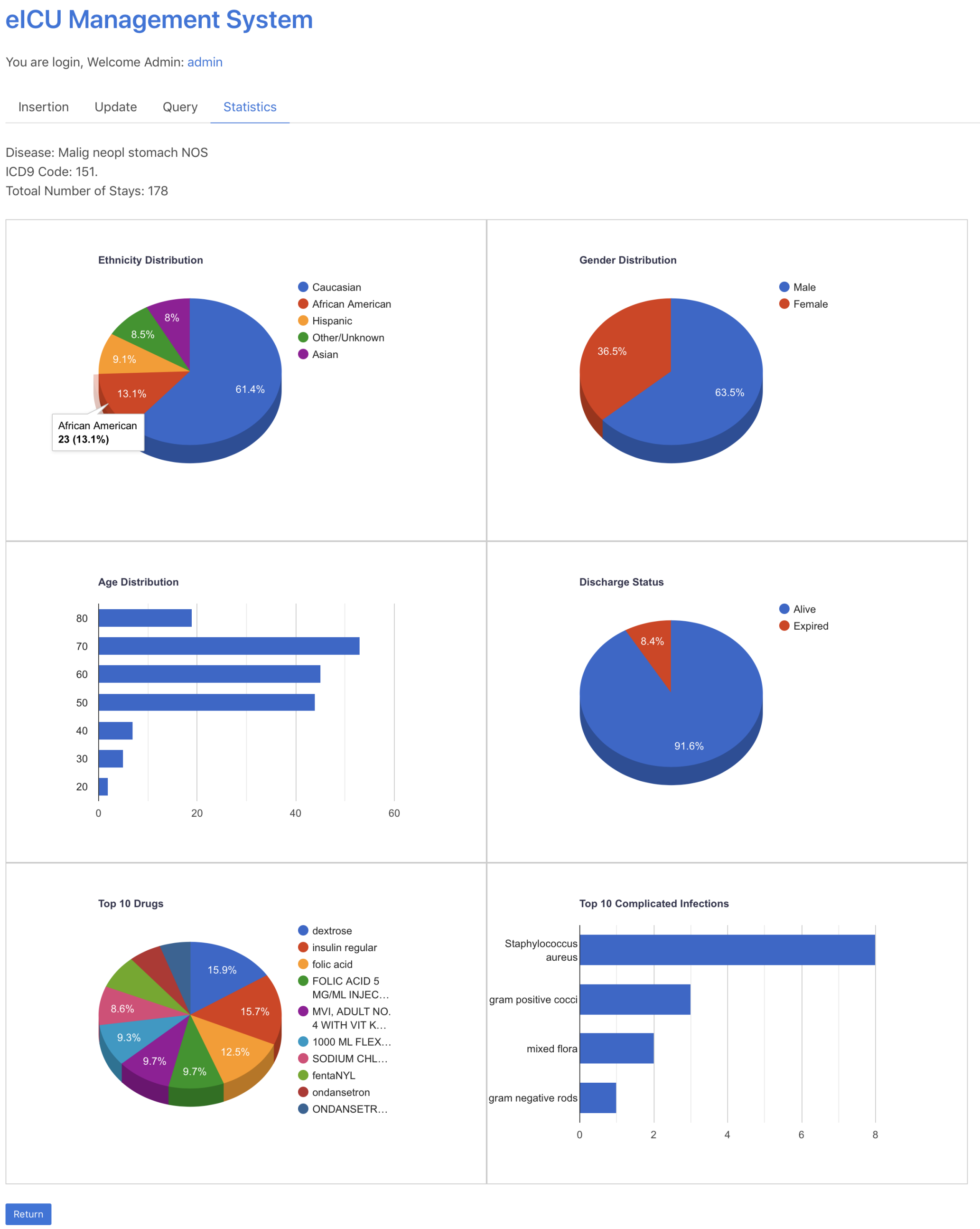


Figure 17: An example to visualize a specific disease

100,000 new cases and over 9,000 deaths each year in the United States. Based on the statistics reported by American Cancer Society(Fig. 19 and Fig. 20), number of older non-Hispanic white male patients is higher, which verifies the statistics in our database. And most of the patients are survived for 5 years.

This comparison does have some limitations and is not accurate as a totally fair comparison. Ours target on EHR data in ICU in 2014-2015, and American Cancer Society reports the statistics in 2011-2015 for incidence rates, 2013-2015 for probability of developing cancer, and 2008-2014 for 5-year relative survival. The classification criterion can be different, such as the intervals of the age and defined ethnicity groups. And EHR data is basically a subset of all Melanoma cases since it is not widely used or has not been collected by eICU-CRD. Also, not all the Melanoma patients were

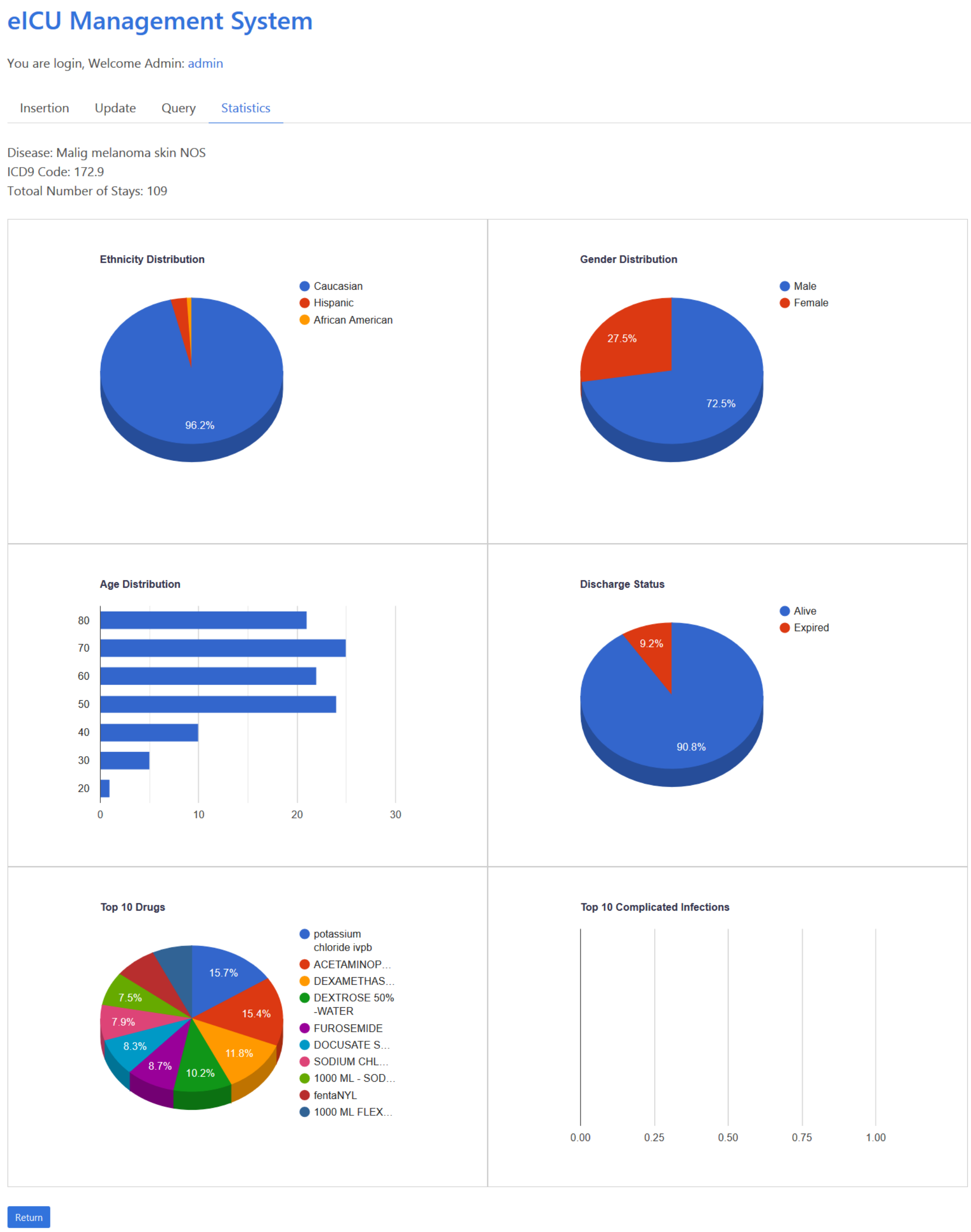


Figure 18: Stastistics of Melanoma

admitted to ICU, especially for a long period statistics. But this comparison still verifies the trends of Melanoma and can be used as an evaluation of our implementation.

Besides, we make use of the hierarchy structure of ICD-9 code, and the visualization of query supports all levels of disease, from coarse to fine. Above example *malignant neoplasm of stomach* (ICD-9 code: 151) is a group of 10 subtypes:

- (ICD-9 code: 151.0) *Malignant neoplasm of cardia*
- (ICD-9 code: 151.1) *Malignant neoplasm of pylorus*
- (ICD-9 code: 151.2) *Malignant neoplasm of pyloric antrum*

rates.png

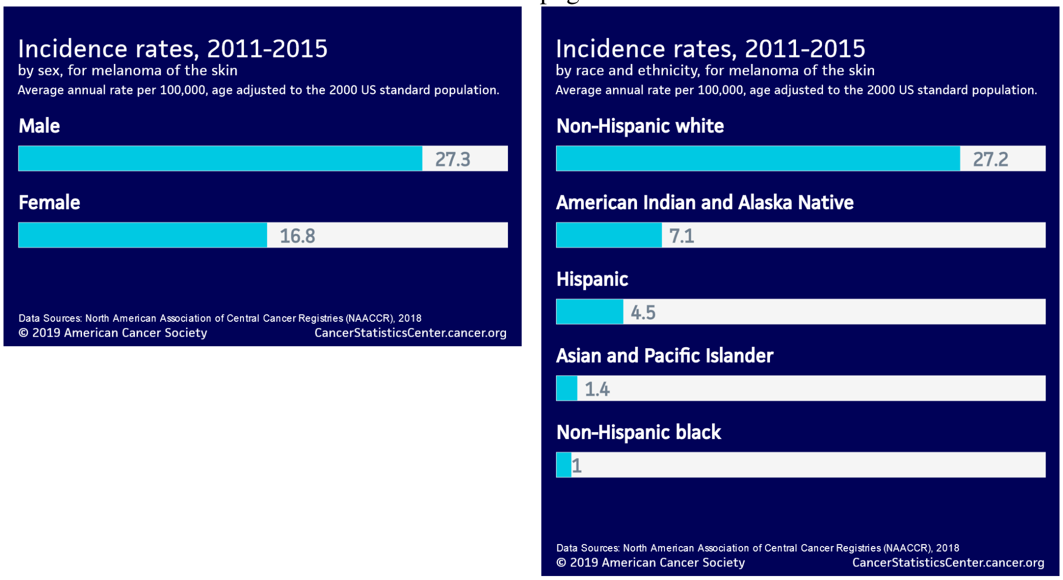


Figure 19: Incidence rates of Melanoma (2011-2015)

develops.png

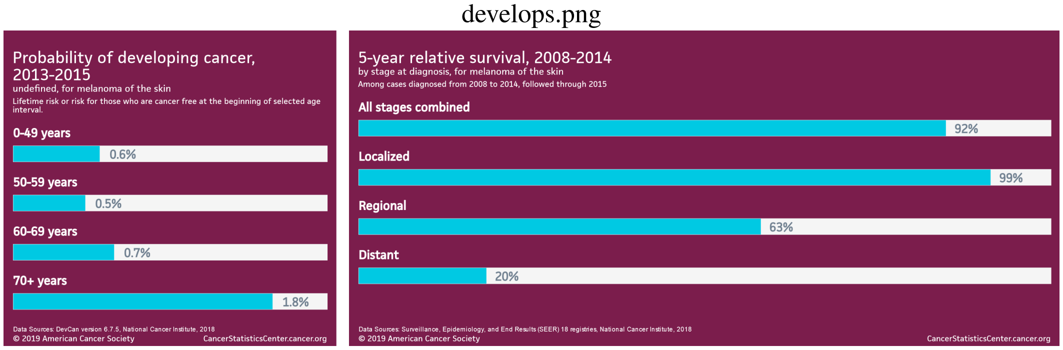


Figure 20: Probability of occurrence and survival of Melanoma

- (ICD-9 code: 151.3) *Malignant neoplasm of fundus of stomach*
- (ICD-9 code: 151.4) *Malignant neoplasm of body of stomach*
- (ICD-9 code: 151.5) *Malignant neoplasm of lesser curvature of stomach, unspecified*
- (ICD-9 code: 151.6) *Malignant neoplasm of greater curvature of stomach, unspecified*
- (ICD-9 code: 151.8) *Malignant neoplasm of other specified sites of stomach*
- (ICD-9 code: 151.9) *Malignant neoplasm of stomach, unspecified site*

Therefore, it is possible to analysis a series of diseases based on our visualization functionality, instead of treating them independently.

Fig. 21 shows a disease distribution of certain population including Male Caucasian age from 20 to 30. This visualization is necessary since we can know disease distribution of certain group, which is meaningful in public health research.

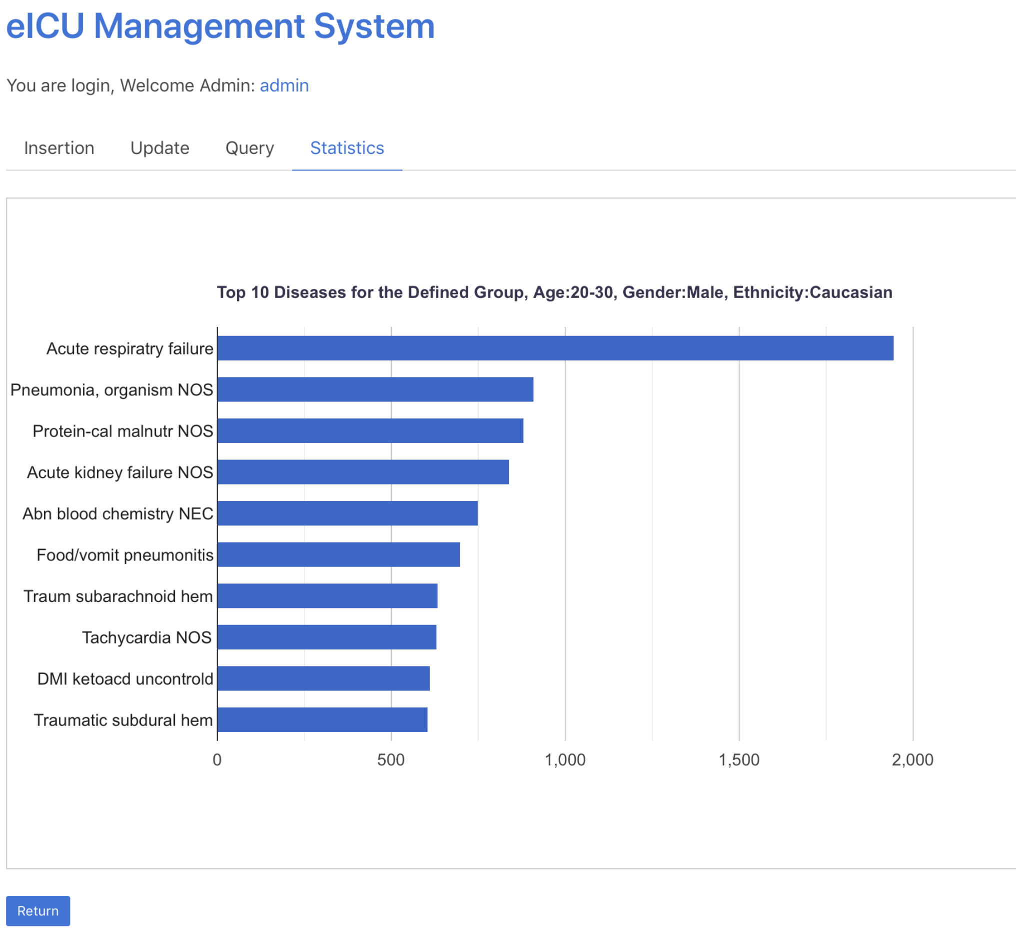


Figure 21: An example to visualize disease distribution of certain population

2.6 Evaluation

Our target is a database implementing insertion, modification, deletion, query and visualization via user interface. We expect that selected data is identical to the inserted, modified, queried, visualized data when inserting, modifying, deleting, querying, respectively. Deleted data is absent by selection is what we expect for deletion operation.

We evaluate insertion, deletion and modification feature by querying. For instance, we first check a non-exist patient, ICU stay or a record via query and then query after inserting a corresponding patient account, ICU stay or a record in our database. Similar operations can be applied to confirm the correctness of deletion and modification. A step-by-step evaluation has been demonstrated as a recorded video [7].

There are two views that we concentrate on, i.e. patient and disease. After data cleaning, all the records have been linked to each patient by patient identifiers ***uniquePID*** and ICU stay identifier ***patientUnitStayID***. A different view is to link all relevant records to a single disease by ICD-9 code, which has been verified in section 2.5.5. Functionalities like insertion, modification, deletion, query and visualization support both patient view and disease view.

Our user interface has realized aforementioned functionalities by interacting with the user interface. Visualization for diseases contains a set of tables and plots with demographics, lab tests, vital signs and medications. And variables shown in generated plots match the descriptive statistics of stored data. For example, the number of total patient stay displayed above the graphs is expected to be equal

[7] https://drive.google.com/open?id=1sKqtTRrN4MpVg5WJXvZN77MH0OcyIkDk

with the sum of the numbers in each part of a pie chart. We also built toy-dataset to check whether the outputs of our queries match eyeballing results (Fig. 22). For individuals, insertion, modification, deletion, query and visualization can filter and display relevant descriptions as expected.

Diagnosis in toy_db

	dgid	stayid	icd9
0	1	141168	991
1	2	141168	991
2	3	141168	151
3	4	141169	151
4	5	141170	151

```
conn = sqlite3.connect('toy_db.sqlite3')
query = """
        SELECT d.icd9, COUNT(DISTINCT t.pid) FROM
        (SELECT DISTINCT stayid, pid FROM stay
        ) AS t, diagnosis d
        WHERE t.stayid = d.stayid
        GROUP BY d.icd9
        HAVING d.icd9 <> ''
        ORDER BY COUNT(*) DESC
        LIMIT 10
        """
c = conn.cursor()
c.execute(query)
print(c.fetchall())
conn.commit()
conn.close()
```

```
[(151, 2), (991, 1)]
```

Figure 22: Use a toy database for evaluation

In addition, three types of users with different level of permissions have been defined to manage the data. As demonstrated in section 2.5, **admin** holds the highest authority to insert, delete, query, visualize all records both in patient and disease view. Their accounts will be created by default in the database. **Physicians** can manage the data by inserting, modifying, querying, visualizing all records, and adding patients or diseases that are not existed in the database. But they cannot delete records. **Patients** are able to query their own records by logging into their own accounts associated with their identifiers ***uniquePID***. Their accounts are created when they first log in using patient identifiers. The expectation has been achieved that different users can only use the permitted functions as described above and functions without permission will not be allowed in practice.

3 Conclusion

We are glad to implement almost everything we scheduled in the project proposal (i.e. checkpoint 1). Along the process, we have a deeper understanding about the database itself and some other relevant topics, which encourages us to explore domain knowledge and new techniques. We also found meaningful insights based on the EHR dataset and it will work as a great foundation for future analysis.

3.1 Implementation Details

3.1.1 Indexing

We monitor the running time for insertion, modification, deletion, query and visualization, specially for this large amounts of data. It was slow due to table size.

As introduced in lecture, indexing could improve the speed of data retrieval operations. It quickly locates data without having to search each record in the database. For example, we added index to the *Stay* table on *patientunitstayid* using

```
CREATE INDEX PATIENT_idx01 ON stay (patientunitstayid)
```

It is quite beneficial in our case especially for the table with enormous records, such as **Diagnosis** (2,710,672 records), **Medication** (7,301,853 records) and **Lab** (39,132,531 records). All queries can be done within one second after adopting indexing, which needed more than 30 seconds originally, as shown in Fig. 23.

```
%%time
# Run query after adding index
conn = sqlite3.connect('C:\Users\Jifan Gao\CS564Project\proj_final_version\db.sqlite3')
query = """
        SELECT DISTINCT m.drugname, COUNT(*)
        FROM (
              SELECT DISTINCT patientunitstayid FROM diagnosis
              WHERE icd9code LIKE '151.%') AS t1, medication AS m
        WHERE t1.patientunitstayid = m.patientunitstayid
              AND m.drugname <> ''
        GROUP BY m.drugname
        ORDER BY COUNT(*) DESC
        LIMIT 10
        """
c = conn.cursor()
c.execute(query)
c.fetchall()
conn.commit()
conn.close()
```

```
Wall time: 443 ms
```

Figure 23: Running query in Python after applying index

3.1.2 Necessity of ER Diagram

One lesson we learned from this project is always keeping the ER diagram in mind. By looking at ER diagram, the types of relationships are clarified. For example, we ignored the fact that the relationship between **Stay** and **Diagnosis** is one-to-many at the beginning and implemented a query which find top 10 most frequently used drug for a disease as

```
SELECT DISTINCT m.drugname, COUNT(*)
FROM (
      SELECT patientunitstayid FROM diagnosis
      WHERE icd9code LIKE '995.9%') AS t1, medication AS m
WHERE t1.patientunitstayid = m.patientunitstayid
```

```
      AND m.drugname <> ''
GROUP BY m.drugname
ORDER BY COUNT(*) DESC
LIMIT 10
```

This query ignores that there are in fact some duplicated *patientunitstayid* in **Diagnosis** and thus raises incorrect results during evaluation. Considering that the relationship between **Stay** and **Diagnosis** is one-to-many, the correct query should be

```
SELECT DISTINCT m.drugname, COUNT(*)
FROM (
      SELECT DISTINCT patientunitstayid FROM diagnosis
      WHERE icd9code LIKE '995.9%') AS t1, medication AS m
WHERE t1.patientunitstayid = m.patientunitstayid
      AND m.drugname <> ''
GROUP BY m.drugname
ORDER BY COUNT(*) DESC
LIMIT 10
```

ER Diagram also guided us to implement the user interface and combine all the components in the back-end. It is clear and concise.

3.1.3 Front-end and Framework

We spent a lot of efforts in front-end and the whole framework at first, which might good for a product but not so great as a course project to apply what we have learned in class. Then we gave up the idea to write CSS from scratch but switched to Bulma as a CSS template. It saves our labor and time to focus more on data analysis itself and it is also flexible.

Django also provides a mature web-framwork, to link all essential components like front-end and back-end. It accelerates custom web application development. Django is compatible with various operating systems and databases, which would be easy to extend and scale in the future. Its robust security feature ensures the privacy of our EHR data. And activity logs, to record the timestamps and user behaviors (e.g.insertion, deletion, modification), can be embeded together for security.

3.2 Clinical Analysis

As mentioned in section 2.6, we gained some clinical insights from our built database. It is more intuitive and interactive as a foundation for future analysis. Adopting post-processing techniques (e.g. natural language processing and machine learning) on top of our application, such as disease prediction and symptom classification is natural as clinical decision support. It can be beneficial for other relevant tasks, such as adjusting medical insurance rates.

Two essential tables **nurseCharting** and **vitalPeriodic** are not involved in our database due to our time limitation but they provides tons of information regarding time series events especially for vital signs records, including respiratory rate, body temperature, intracranial pressure, central venous pressure, Invasive blood pressure (systolic and diastolic). Those monitor series events for individuals and can be grouped for a group of patients. Statistical tests, such as t-test, could be applied as well to extract potential relations among data.

Appendix A Code

We share our code via Google Drive [8] due to the space limitation of Canvas. It contains all running code and data, which has been converted to SQLite database (3.3 GB) format. All SQL queries can be found in files named as **views.py* in *query, insert, delete, stats* folders.

To run the code, Python 3.5+ and Django 2.0.8+ are required. We recommend Anaconda[9] virtual environment to setup the environment. Please refer to README file within the project folder to run the code.

[8] https://drive.google.com/open?id=1UcXjieYZOSPXjflXYykW4EbAjH-tMo2A

[9] https://www.anaconda.com/distribution/

The directory structure of our code is as follows

```
.
|   db.sqlite3
|   manage.py
|   README.md
|
+---delete
|   |   admin.py
|   |   apps.py
|   |   delete_views.py
|   |   tests.py
|   |   __init__.py
|   |
|   \---migrations
|           __init__.py
|
+---eicu_v0
|       settings.py
|       urls.py
|       wsgi.py
|       __init__.py
|
+---home
|   |   admin.py
|   |   apps.py
|   |   home_views.py
|   |   models.py
|   |   tests.py
|   |   __init__.py
|   |
|   \---migrations
|           __init__.py
|
+---insert
|   |   admin.py
|   |   apps.py
|   |   insert_views.py
|   |   tests.py
|   |   __init__.py
|   |
|   \---migrations
|           __init__.py
|
+---query
|   |   admin.py
|   |   apps.py
|   |   models.py
|   |   query_views.py
|   |   tests.py
|   |   __init__.py
|   |
|   \---migrations
|           0001_initial.py
|           __init__.py
|
+---stats
|   |   admin.py
|   |   apps.py
```

```
|   |   models.py
|   |   stats_views.py
|   |   tests.py
|   |   __init__.py
|   |
|   \---migrations
|           __init__.py
|
+---templates
|   +---delete
|   |       delete.html
|   |       delete_pt.html
|   |       delete_stay.html
|   |       delete_success.html
|   |
|   +---home
|   |       home.html
|   |       home2.html
|   |
|   +---insert
|   |       data_added.html
|   |       insert.html
|   |       new_data.html
|   |       new_modified_data.html
|   |       new_modified_id_selection.html
|   |       new_patient.html
|   |       new_stay.html
|   |
|   +---query
|   |       pt_result.html
|   |       query.html
|   |       stayinfo.html
|   |
|   +---registration
|   |       login.html
|   |       password_change_done.html
|   |       password_change_form.html
|   |       password_reset_complete.html
|   |       password_reset_confirm.html
|   |       password_reset_done.html
|   |       password_reset_form.html
|   |
|   +---statistics
|   |       icd_demograph.html
|   |       patients_disease.html
|   |
|   \---users
|           register.html
|
\---users
    |   admin.py
    |   apps.py
    |   forms.py
    |   models.py
    |   tests.py
    |   urls.py
    |   views.py
    |   __init__.py
    |
```

```
\---migrations
        __init__.py
```

Appendix B Recorded Video

As mentioned in 2.6, we recorded a video[10] to cover the detailed steps when running our application. It demonstrates the functionalities, evaluation and user interface.

References

[1] Accountability Act. Health insurance portability and accountability act of 1996. *Public law*, 104:191, 1996.

[2] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[3] Tom J Pollard, Alistair EW Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. The eicu collaborative research database, a freely available multi-center database for critical care research. *Scientific data*, 5, 2018.

[4] Kun-Hsing Yu, Andrew L Beam, and Isaac S Kohane. Artificial intelligence in healthcare. *Nature biomedical engineering*, 2(10):719, 2018.

[10]https://drive.google.com/open?id=1sKqtTRrN4MpVg5WJXvZN77MH0OcyIkDk