zx

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**BELGAVI - 590014**

**A DBMS (18CSL58) MINI PROJECT REPORT ON**

**“COVID BED SLOT BOOKING DATABASE MANAGEMENT SYSTEM”**

***Submitted in the partial fulfilment of the requirements for***

***The award of Degree***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Submitted By** | |  |
|  | **TANU SHREE** |  |  |
|  | **(1KN19CS107)** |  |  |
|  | **UNDER THE GUIDANCE OF** |  |  |
|  | **Mrs B.Vidhya Banu** | |  |
|  | **Asst. Prof. , Dept of CSE** | |  |

**K.N.S INSTITUTE OF TECHNOLOGY**

**Hegde Nagar, Yelhanka Road, (Off Outer Ring Road)**

**Thirumenahalli, BANGALORE-560 064**

**2021-2022**

**K.N.S INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

This is to certify that the technical Project Work entitled **“COVID BED SLOT BOOKING DATABASE MANAGEMENT SYSTEM MANAGEMEN”** is a bonafide work carried out by **TANU SHREE (1KN19CS107)** in partial fulfilment for the award of **Bachelor of Engineering in Computer Science Engineering** under **Visvesvaraya Technological University, Belagavi,** during the year **2021-2022**. It is certified that all corrections/suggestions given for Internal Assessment have been incorporated in the report. This Project report has been approved as it satisfies the academic requirements in aspects of project work prescribed for the said degree

**Signature of Guide Signature of HOD**

**Mrs. B. Vidhya Banu Dr. Suresh H**

Asst. Professor Professor & HOD

Dept. of CSE, Dept. of CSE

KNSIT, Bangalore KNSIT, Bangalore

**INTERNAL EVALUATION**

**External Viva**

**Name of the Examiners Signature with date**

**1.**

**2.**

**ACKNOWLEDGEMENT**

The satisfaction that accompanies the success completion of this project would be incomplete without the mention of the people who made it possible, without whose constant guidance and encouragement would have made my efforts go in vain.

We consider myself privileged to express gratitude and respect towards all those who guided me through the completion of this project.

We would like to express my gratitude to **Dr. S. M. Prakash, Principal** of **KNS INSTITUTE OF TECHNOLOGY** for providing us congenial environment and surrounding to work in.

We would like to express my gratitude to **Dr Suresh H, Head of the Department, Department of Computer Science and Engineering, KNS INSTITUTE OF TECHNOLOGY** for giving me the support and encouragement that was necessary for the completion of this project.

We would also like to express my gratitude to **Mrs. B. Vidhya Banu , Assistant Professor,**

**Department of Computer Science and Engineering, KNS INSTITUTE OF TECHNOLOGY** for giving me the support and encouragement that was necessary for the completion of this project.

We would also convey my regards to all other Staff members and non-staff members for Constantly Motivating and guiding us throughout this study.

Finally, we thank all my friends for their co-operation and their guidance in Bringing out this project successful.

|  |  |
| --- | --- |
|  | **TANU SHREE (1KN19CS107)** |
|  |  |
|  |  |

**Table of Contents**

**Contents**

[**Chapter I: Introduction**](#page5) **1**

[1.1Introduction](#page5) 1

[1.2 Objective](#page5) and project aims 1

[1.3 Methodology Development Model](#page6) 2

[1.4 Tools and Technique](#page6) 3

[1.5.1hardware and software requirement](#page10)  5

1.6Specification Requirement.......................................................................................................6

Figure :Logical Database

[Figure: Schema Diagram](#page16)

[**Chapter II: Task and Activities Performed**](#page18)

[2.1 System Analysis](#page19) 10

[2.2 System Design](#page20) 10

[2.3 Implementation](#page20) 22

2.4 System testing [20](#page20)

[Screen Shot](#page20) 23

References……………………………………………………………………………… 25

**Chapter I: Introduction**

**1.1Introduction**

In this 21'st century, all is about technology.Covid Bed Slot Booking Management in this pandemic has become an issue that almost all of the families of covid patients face. But by using the latest technology this problem can be solved, The main reason for this problem is hospitals not having a proper portal for managing beds. Our solution to this problem can be used to save many lives of covid patients. Therefore we have tried to address this problem with help of our project where the focus is to make sure that every covid patient gets a bed. We have achieved this by creating two apps one for users through which they can request a bed in a hospital and another one for hospitals where they can manage beds.

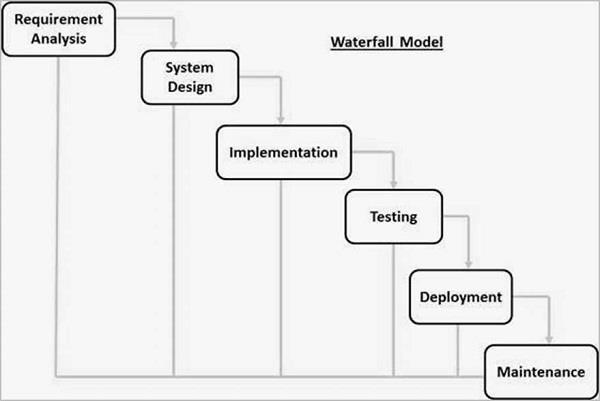
**1.2 Objective**

The project aim and objectives that will be achieved after completion of this project are

discussed in this subchapter. The aims and objectives are as follows:

* Design a system for better patientcare
* Seat Management.
* Equipment Management.
* Oxygen Management.
* Doctor Management.
* Daily Patient Routine Checkup .
* Hospital capacity, including information on ICU capacity and available ventilators

**1.3 Methodology Development Model**



The sequential phases in Waterfall model are −

**Requirement Gathering and analysis**− All possible requirements of the systemto be developed are captured in this phase and documented in a requirement specification document.

**System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

**Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

**Integration and Testing** − All the units developed in the implementation phase areintegrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

**Deployment of system** −Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

**maintenance** −There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

**1.4 Tools and Technique**

* Frontend
* HTML5
* CSS3
* JQuery
* Bootstrap4
* Backend
* Django framework
* Database
* SQLite
* Python

**HTML**

Hypertext Markup Language (HTML) is the standard [markup language](https://en.wikipedia.org/wiki/Markup_language) for creating [web](https://en.wikipedia.org/wiki/Web_page) pages and [web applications.](https://en.wikipedia.org/wiki/Web_application) With [Cascading Style Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) and [JavaScript,](https://en.wikipedia.org/wiki/JavaScript) it forms a triad of cornerstone technologies for the [World Wide Web.](https://en.wikipedia.org/wiki/World_Wide_Web)

[Web browsers](https://en.wikipedia.org/wiki/Web_browser) receive HTML documents from a [web server](https://en.wikipedia.org/wiki/Web_server) or from local storage and [render](https://en.wikipedia.org/wiki/Browser_engine) the documents into multimedia web pages. HTML describes the structure of a web page [semantically](https://en.wikipedia.org/wiki/Semantic_Web) and originally included cues for the appearance of the document.

[HTML elements](https://en.wikipedia.org/wiki/HTML_element) are the building blocks of HTML pages. With HTML constructs, [images](https://en.wikipedia.org/wiki/HTML_element#Images_and_objects) and other objects such as [interactive forms](https://en.wikipedia.org/wiki/Fieldset) may be embedded into the rendered page. HTML

provides a means to create [structured documents](https://en.wikipedia.org/wiki/Structured_document) by denoting structural [semantics](https://en.wikipedia.org/wiki/Semantics) for text such as headings, paragraphs, lists, [links,](https://en.wikipedia.org/wiki/Hyperlink) quotes and other items.

**Cascading Style Sheets** (**CSS**)

Is a s[tyle sheet language](https://en.wikipedia.org/wiki/Style_sheet_language) used for describing the [presentation](https://en.wikipedia.org/wiki/Presentation_semantics) of a document written in

a [markup language](https://en.wikipedia.org/wiki/Markup_language) such as [HTML](https://en.wikipedia.org/wiki/HTML).CSS is a cornerstone technology of the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web),

alongside HTML and [JavaScript](https://en.wikipedia.org/wiki/JavaScript).

CSS is designed to enable the separation of presentation and content, including [layout](https://en.wikipedia.org/wiki/Page_layout), [colors](https://en.wikipedia.org/wiki/Color),

and [fonts](https://en.wikipedia.org/wiki/Typeface). This separation can improve content [accessibility](https://en.wikipedia.org/wiki/Accessibility); provide more flexibility and

control in the specification of presentation characteristics; enable multiple [web pages](https://en.wikipedia.org/wiki/Web_page) to share

formatting by specifying the relevant CSS in a separate .css file, which reduces complexity

and repetition in the structural content; and enable the .css file to be [cached](https://en.wikipedia.org/wiki/Cache_(computing)) to improve the

page load speed between the pages that share the file and its formatting.

**Bootstrap**

Bootstrap is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software) front-end framework for designing [websites](https://en.wikipedia.org/wiki/Website) and [web](https://en.wikipedia.org/wiki/Web_application) applications. It contains [HTML-](https://en.wikipedia.org/wiki/HTML) and [CSS-](https://en.wikipedia.org/wiki/CSS)based design templates for [typography,](https://en.wikipedia.org/wiki/Typography) forms, buttons, navigation and other interface components, as well as optional [JavaScript](https://en.wikipedia.org/wiki/JavaScript) extensions. Unlike many web frameworks, it concerns itself with [front-end development](https://en.wikipedia.org/wiki/Front-end_web_development) only.

**Django Framework**

Django is an extremely popular and fully featured server-side web framework, written in

Python. This module shows you why Django is one of the most popular web server

frameworks, how to set up a development environment, and how to start using it to create your

own web applications.

**SQLite**

SQLite is a C library that provides a lightweight disk-based database that doesn’t require a

separate server process and allows accessing the database using a nonstandard variant of the

SQL query language. Some applications can use SQLite for internal data storage. It’s also

possible to prototype an application using SQLite and then port the code to a larger database

such as PostgreSQL or Oracle.

PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language.

Python is designed to be highly readable. It uses English keywords frequently whereas the other

languages use punctuations. It has fewer syntactical constructions than other languages.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not

need to compile your program before executing it. This is similar to PERL and PHP.

* Python is Interactive − You can actually sit at a Python prompt and interact with the

interpreter directly to write your programs.

* Python is Object-Oriented − Python supports Object-Oriented style or technique of

programming that encapsulates code within objects.

* Python is a Beginner's Language − Python is a great language for the beginner-level

programmers and supports the development of a wide range of applications from simple

text processing to WWW browsers to games.

**1.5 Hardware And Software Requirement**

**Hardware:**

* **Python**
* **SQL**
* **Djangoframework**
* **Window 10**
* **Html &Css**

**SOFTWARE**

* **Intel i3,7 gen**
* **Harddisk 500 mb**

1**.6 Specification Requirement**

**1.6.1 External Interfaces**

* This interface will be actual interface through which the user will communication with the application and perform the desired tasks.

Admin login

**I.D:**

**Role**: Admin wishes to login to the system

**Precondition**: Username and Password

**Success end Condition**: Main option of screen display

**Failed end Condition**: User has entered incorrect Username and

Password or both

**1.6.2 Software Product**

**Features**

**Login Information System**

Description

-The system will maintain the login information of its user to enter in to the software

Validating Checks

-Administrator need to login the unique id and password. -Contact number should have maximum 10 digits.

-All the details must be fill up.

-Email address should be in the proper format.

Sequencing information

-Login information should be filled before the user allowed.

Error Handling

-If user doesn’t filled up validate information then the system display error message for user and request to enter the validate information.

**Performance required**

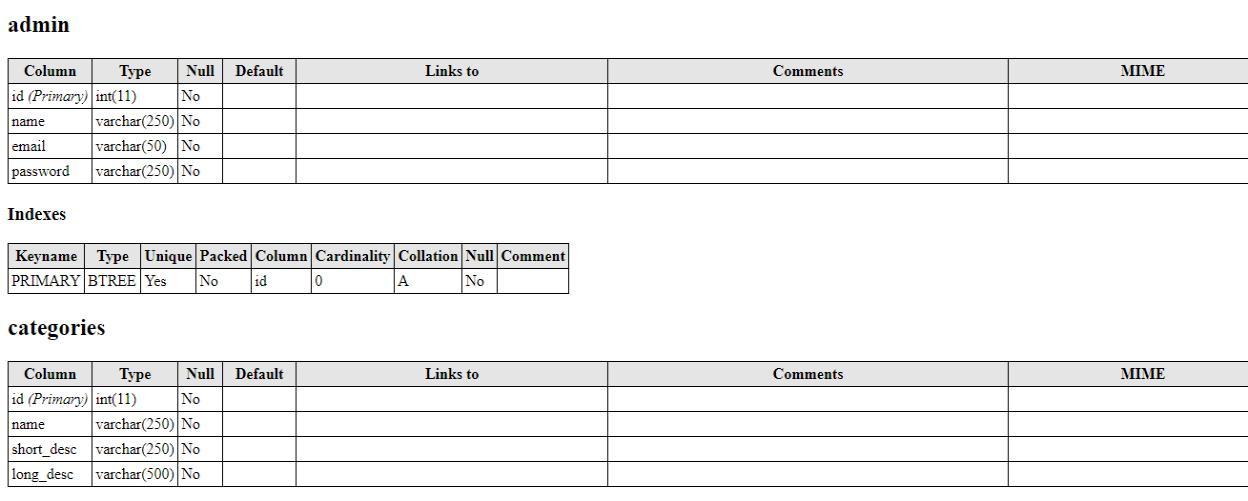
Security

-System should be Protected from unauthorized access Where the validate

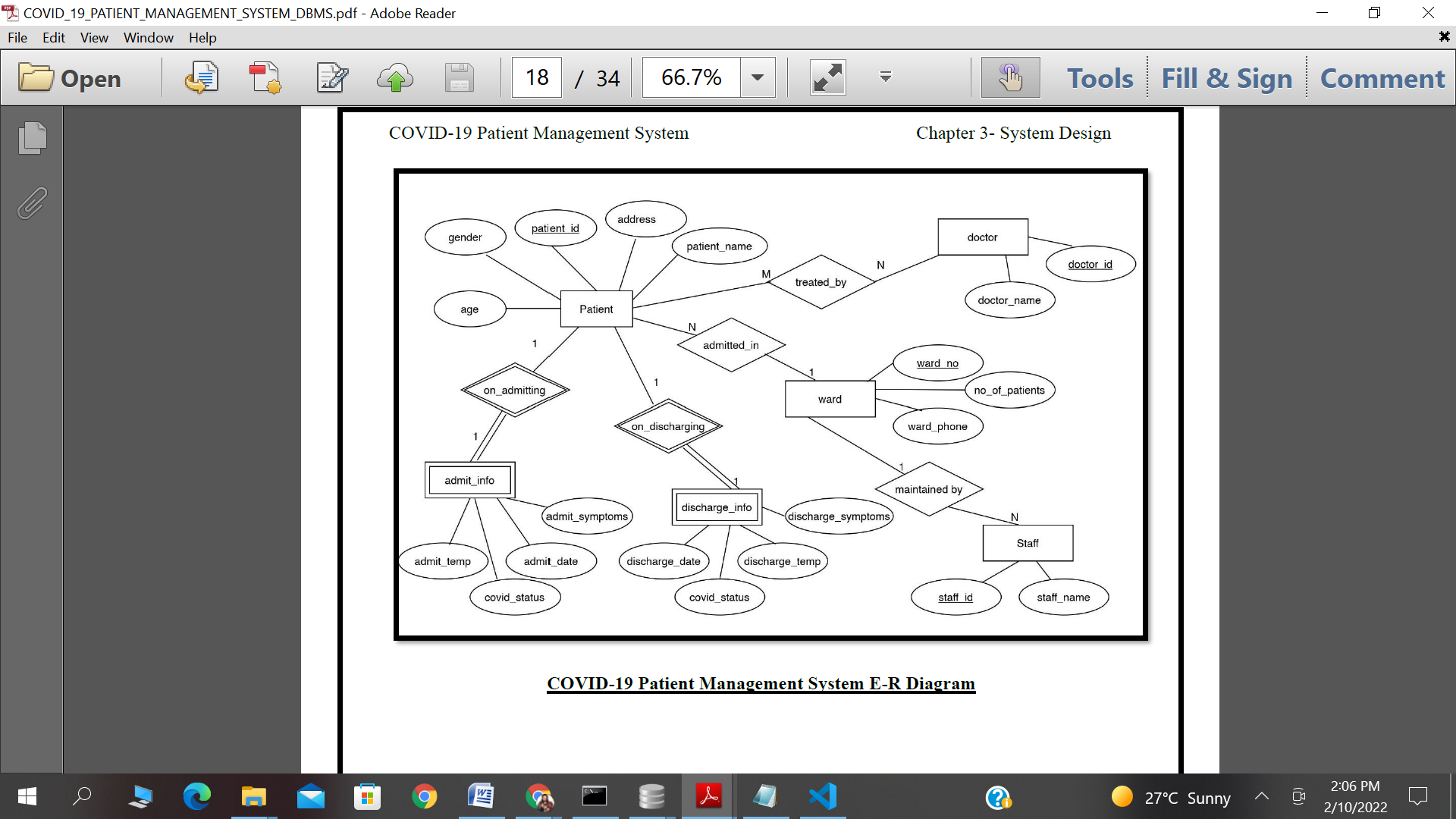
Username and Password are required so no other can access.Maintainability

-System should be design in a maintain order. So it can be easily modified

**Logical Database**



**ER Diagram**

****

**2.1 System Analysis**

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components.

System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose.

The objective of the system analysis activity is to develop structured system specification for the proposed system. The structured system specification should describe what the proposed system would do; independent of the technology, which will be used to implement these requirements. The structured system specification will be used to implement these requirements.

The essential model may itself consist of multiple models, modeling different aspect of the system. The data flow diagrams may model the data and there relationships and the state transition diagram may model time dependent behavior of the system. The essential model thus consists of the following.

Context diagram

Leveled data flow diagrams

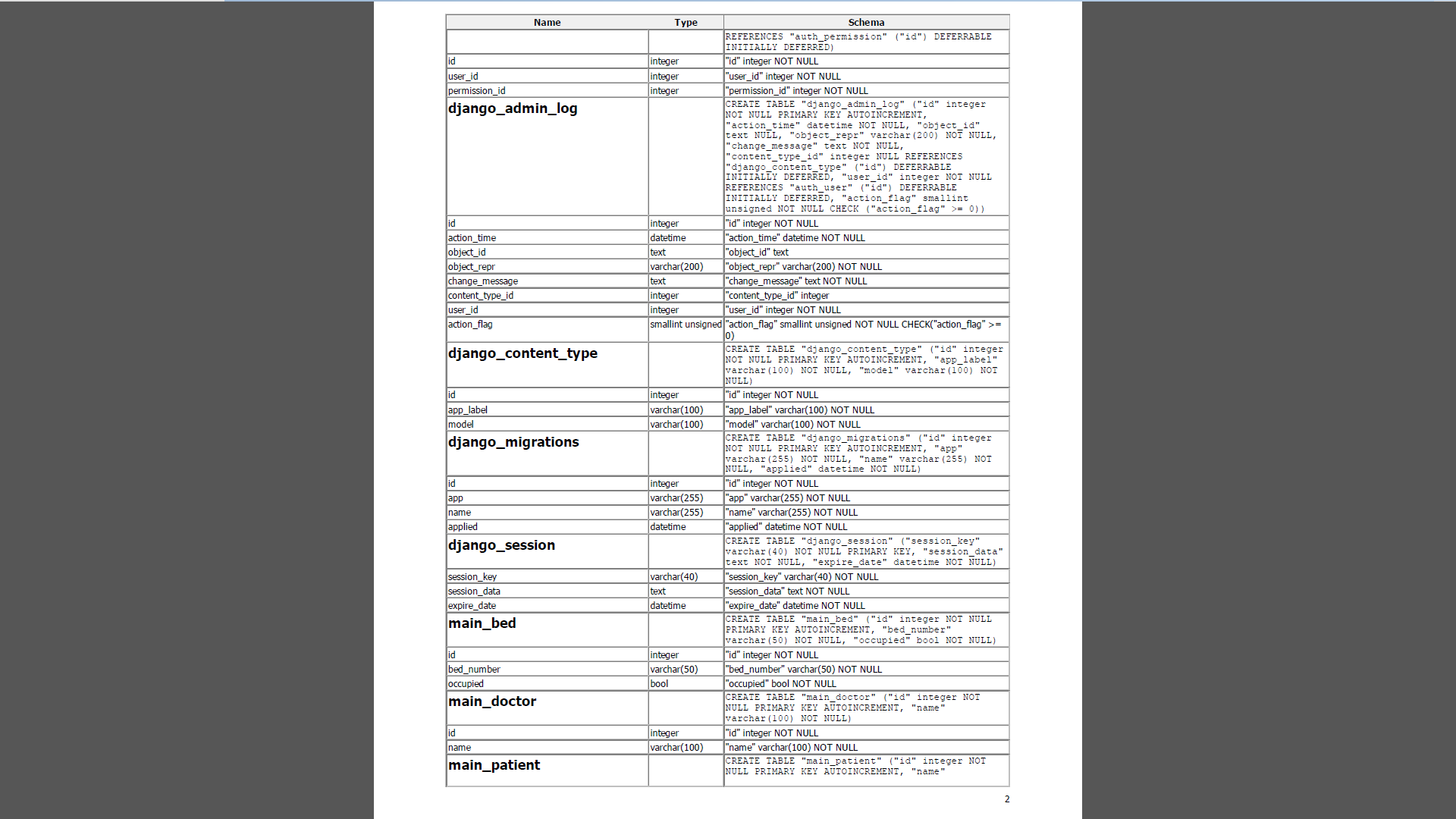
Process specification for elementary bubbles

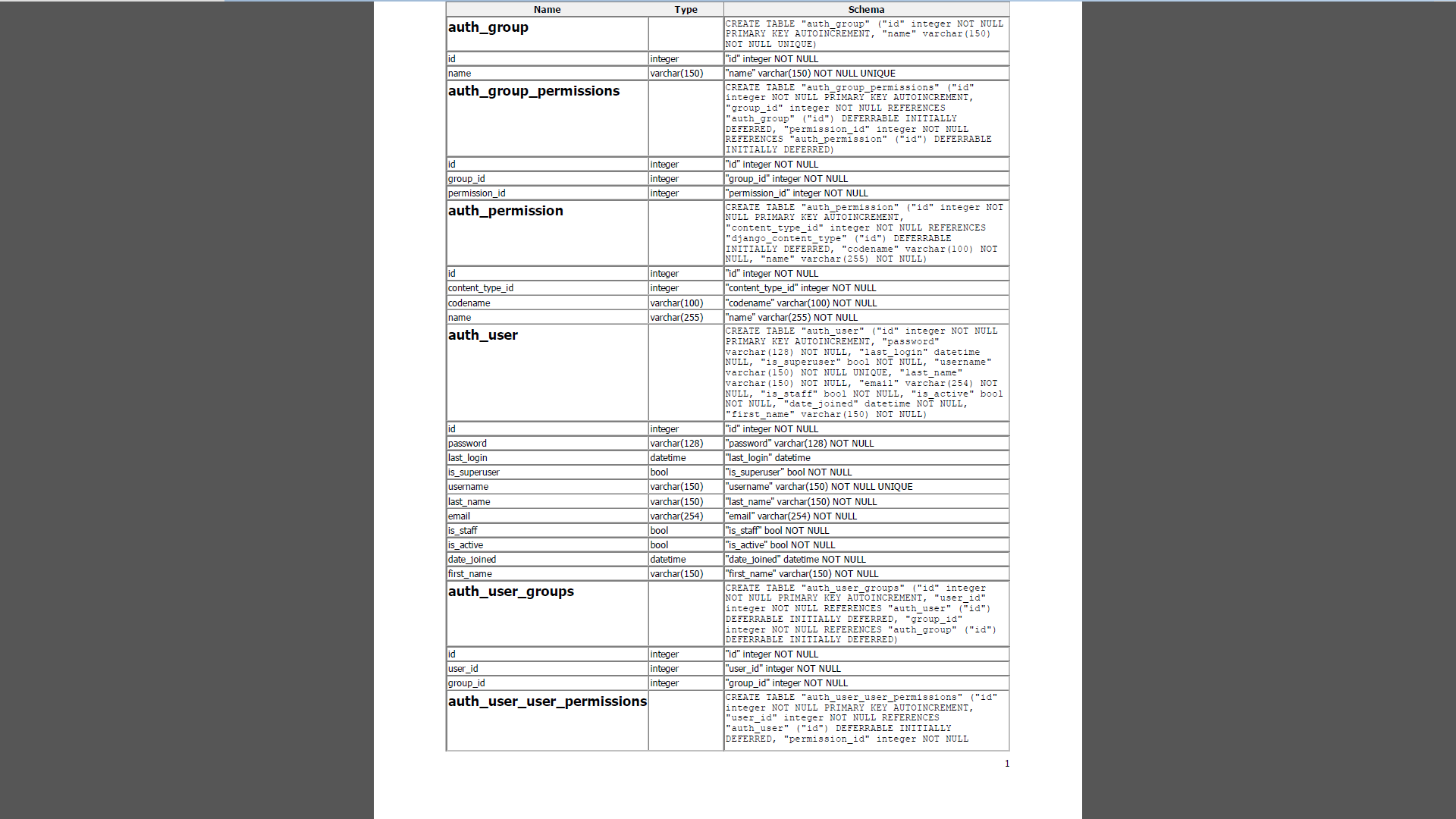
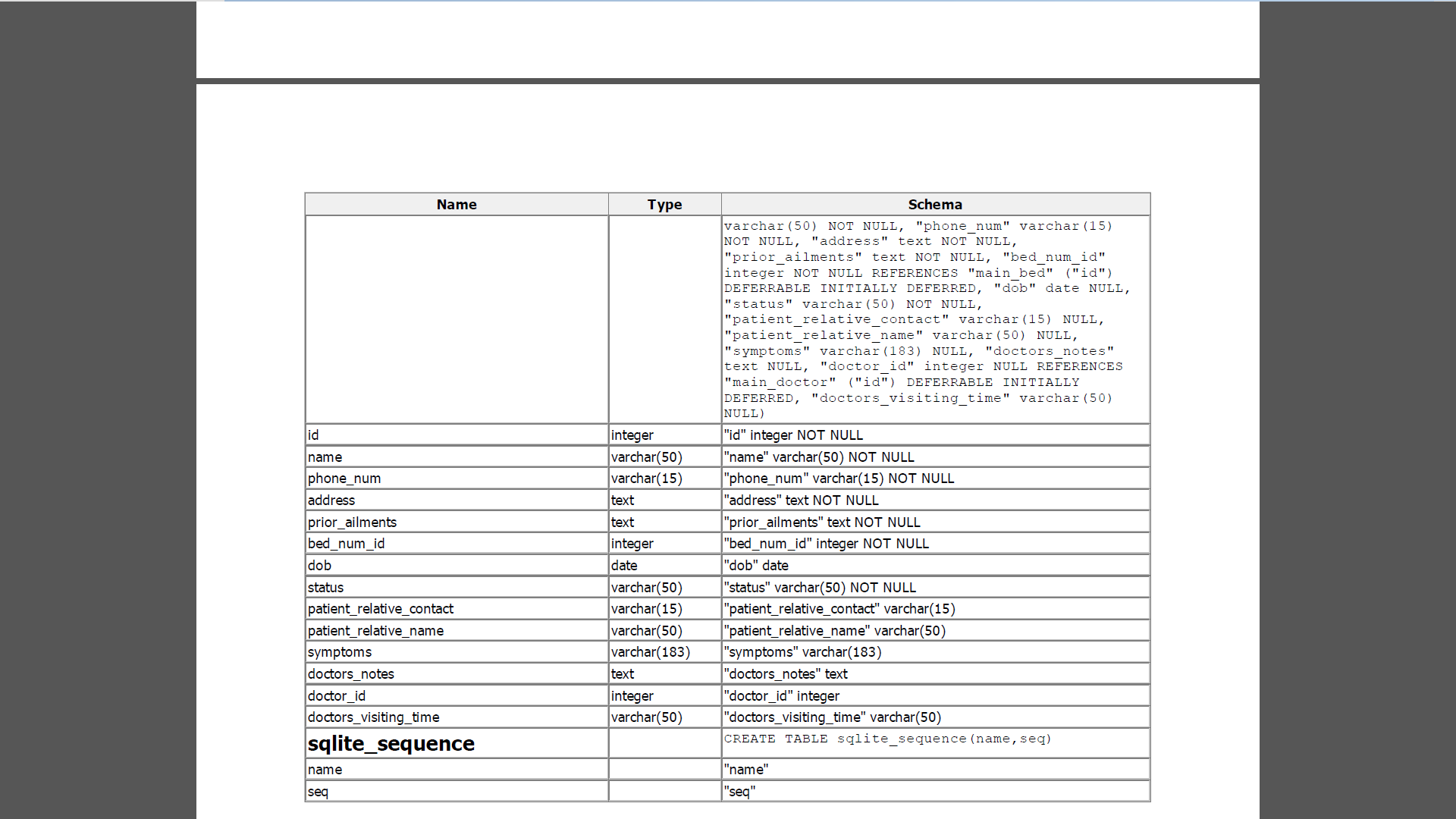
Data dictionary for the flow and stores on the DFDs.

**2.2 System Design**

**TABLE DESIGN**

**VARIOUS TABLES TO MAINTAIN INFORMATION**

****

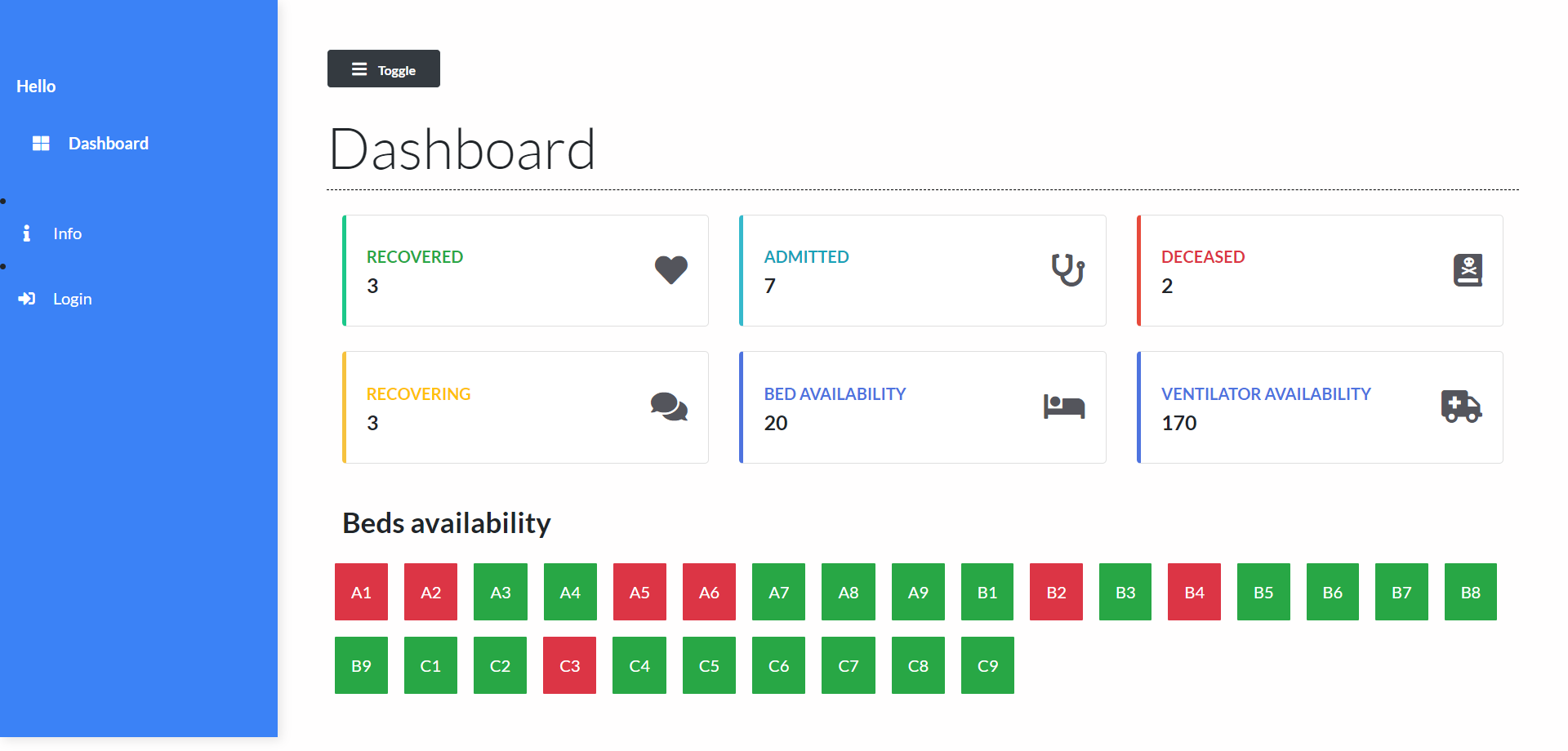
****

**2.3 SYSTEM IMPLEMENTATION**

This activity includes programming, testing and integration of modules into a progressively more complete system. Implementation is the process of collect all the required parts and assembles them into a major product.

**MODULE DESCRIPTION**

**Screenshot of homepage**

****

**SOURCE CODE**

CREATE TABLE "auth\_group"

("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,

"name" varchar(150) NOT NULL UNIQUE)

CREATE TABLE "auth\_group\_permissions"

("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,

"group\_id" integer NOT NULL REFERENCES,

"auth\_group" ("id") DEFERRABLE INITIALLY DEFERRED,

"permission\_id" integer NOT NULL REFERENCES,

"auth\_permission" ("id") DEFERRABLE INITIALLY DEFERRED)

CREATE TABLE "auth\_user"

("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,

"password" varchar(128) NOT NULL,

"last\_login" datetime NULL,

"is\_superuser" bool NOT NULL,

"username" varchar(150) NOT NULL UNIQUE,

"last\_name" varchar(150) NOT NULL,

"email" varchar(254) NOT NULL,

"is\_staff" bool NOT NULL,

"is\_active" bool NOT NULL,

"date\_joined" datetime NOT NULL,

"first\_name" varchar(150) NOT NULL)

CREATE TABLE "django\_admin\_log"

("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,

"action\_time" datetime NOT NULL,

"object\_id" text NULL,

"object\_repr" varchar(200) NOT NULL,

"change\_message" text NOT NULL,

"content\_type\_id" integer NULL REFERENCES "django\_content\_type" ("id") DEFERRABLE INITIALLY DEFERRED,

"user\_id" integer NOT NULL REFERENCES "auth\_user" ("id") DEFERRABLE INITIALLY DEFERRED,

"action\_flag" smallint unsigned NOT NULL CHECK ("action\_flag" >= 0))

CREATE TABLE "main\_bed"

("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,

"bed\_number" varchar(50) NOT NULL,

"occupied" bool NOT NULL)

CREATE TABLE "main\_doctor"

("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,

"name" varchar(100) NOT NULL)

CREATE TABLE "main\_patient"

("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,

"name" varchar(50) NOT NULL,

"phone\_num" varchar(15) NOT NULL,

"address" text NOT NULL,

"prior\_ailments" text NOT NULL,

"bed\_num\_id" integer NOT NULL REFERENCES "main\_bed" ("id") DEFERRABLE INITIALLY DEFERRED,

"dob" date NULL, "status" varchar(50) NOT NULL,

"patient\_relative\_contact" varchar(15) NULL,

"patient\_relative\_name" varchar(50) NULL,

"symptoms" varchar(183) NULL,

"doctors\_notes" text NULL,

"doctor\_id" integer NULL REFERENCES "main\_doctor" ("id") DEFERRABLE INITIALLY DEFERRED,

"doctors\_visiting\_time" varchar(50) NULL)

**FOR DATABASE CONNECTION**

DATABASES =

{ 'default':

{ 'ENGINE': 'django.db.backends.oracle',

'NAME': 'localhost:1800/XE',

'USER': 'ADMIN',

'PASSWORD': '',

#'ADMIN': '',

#'PORT': '', } }

**SOURCE CODE FOR MAIN METHOD**

#!/usr/bin/env python

"""Django's command-line utility for administrative tasks."""

import os

import sys

def main():

"""Run administrative tasks."""

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'core.settings')

try:

from django.core.management import execute\_from\_command\_line

except ImportError as exc:

raise ImportError(

"Couldn't import Django. Are you sure it's installed and "

"available on your PYTHONPATH environment variable? Did you "

"forget to activate a virtual environment?"

) from exc

execute\_from\_command\_line(sys.argv)

if \_\_name\_\_ == '\_\_main\_\_':

main()

from django.shortcuts import render, redirect

from django.http import HttpResponse, JsonResponse

from .models import \*

from .filters import PatientFilter

from django.contrib.auth.models import User, auth

from django.contrib import messages

from django.contrib.auth.decorators import login\_required

# PatientFilter = OrderFilter

# Create your view

def login(request):

if request.user.is\_authenticated:

return redirect('/')

else:

if request.method == 'POST':

username = request.POST['username']

password = request.POST['password']

user = auth.authenticate(username=username, password=password)

if user is not None:

auth.login(request, user)

return redirect('/')

else:

messages.error(request, 'Invalid username or password')

return redirect('login')

else:

return render(request, 'main/login.html')

@login\_required(login\_url='login')

def logout(request):

auth.logout(request)

return redirect('/')

def dashboard(request):

patients = Patient.objects.all()

patient\_count = patients.count()

patients\_recovered = Patient.objects.filter(status="Recovered")

patients\_deceased = Patient.objects.filter(status="Deceased")

deceased\_count = patients\_deceased.count()

recovered\_count = patients\_recovered.count()

beds = Bed.objects.all()

beds\_available = Bed.objects.filter(occupied=False).count()

context = {

'patient\_count': patient\_count,

'recovered\_count': recovered\_count,

'beds\_available': beds\_available,

'deceased\_count':deceased\_count,

'beds':beds

}

print(patient\_count)

return render(request, 'main/dashboard.html', context)

def add\_patient(request):

beds = Bed.objects.filter(occupied=False)

doctors = Doctor.objects.all()

if request.method == "POST":

name = request.POST['name']

phone\_num = request.POST['phone\_num']

patient\_relative\_name = request.POST['patient\_relative\_name']

patient\_relative\_contact = request.POST['patient\_relative\_contact']

address = request.POST['address']

symptoms = request.POST['symptoms']

prior\_ailments = request.POST['prior\_ailments']

bed\_num\_sent = request.POST['bed\_num']

bed\_num = Bed.objects.get(bed\_number=bed\_num\_sent)

dob = request.POST['dob']

status = request.POST['status']

doctor = request.POST['doctor']

doctor = Doctor.objects.get(name=doctor)

print(request.POST)

patient = Patient.objects.create(

name = name,

phone\_num = phone\_num,

patient\_relative\_name = patient\_relative\_name,

patient\_relative\_contact = patient\_relative\_contact,

address = address,

symptoms = symptoms,

prior\_ailments = prior\_ailments,

bed\_num = bed\_num,

dob = dob,

doctor=doctor,

status = status

)

patient.save()

bed = Bed.objects.get(bed\_number=bed\_num\_sent)

bed.occupied = True

bed.save()

id = patient.id

return redirect(f"/patient/{id}")

context = {

'beds': beds,

'doctors': doctors

}

return render(request, 'main/add\_patient.html', context)

def patient(request, pk):

patient = Patient.objects.get(id=pk)

if request.method == "POST":

doctor = request.POST['doctor']

doctor\_time = request.POST['doctor\_time']

doctor\_notes = request.POST['doctor\_notes']

mobile = request.POST['mobile']

mobile2 = request.POST['mobile2']

relativeName = request.POST['relativeName']

address = request.POST['location']

print(doctor\_time)

print(doctor\_notes)

status = request.POST['status']

doctor = Doctor.objects.get(name=doctor)

print(doctor)

patient.phone\_num = mobile

patient.patient\_relative\_contact = mobile2

patient.patient\_relative\_name = relativeName

patient.address = address

patient.doctor = doctor

patient.doctors\_visiting\_time = doctor\_time

patient.doctors\_notes = doctor\_notes

print(patient.doctors\_visiting\_time)

print(patient.doctors\_notes)

patient.status = status

patient.save()

context = {

'patient': patient

}

return render(request, 'main/patient.html', context)

def patient\_list(request):

patients = Patient.objects.all()

# filtering

myFilter = PatientFilter(request.GET, queryset=patients)

patients = myFilter.qs

context = {

'patients': patients,

'myFilter': myFilter

}

return render(request, 'main/patient\_list.html', context)

'''

def autocomplete(request):

if patient in request.GET:

name = Patient.objects.filter(name\_\_icontains=request.GET.get(patient))

name = ['js', 'python']

names = list()

names.append('Shyren')

print(names)

for patient\_name in name:

names.append(patient\_name.name)

return JsonResponse(names, safe=False)

return render (request, 'main/patient\_list.html')

'''

def autosuggest(request):

query\_original = request.GET.get('term')

queryset = Patient.objects.filter(name\_\_icontains=query\_original)

mylist = []

mylist += [x.name for x in queryset]

return JsonResponse(mylist, safe=False)

def autodoctor(request):

query\_original = request.GET.get('term')

queryset = Doctor.objects.filter(name\_\_icontains=query\_original)

mylist = []

mylist += [x.name for x in queryset]

return JsonResponse(mylist, safe=False)

def info(request):

return render(request, "main/info.html")

**2.4 SYSTEM TESTING**

Testing plays a vital role in the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. Once program code has been developed, testing begins. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested, and on the functional externals, that is conducted tests to uncover errors and ensure that defined input will produce actual results that agree with required results. Broadly speaking, there are at least three levels of testing: unit testing, integration testing, and system testing

**Types of Testing**

**1 Unit testing**

Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other. Unit testing is a software development process that involves a synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Unit testing aims to eliminate construction errors before code is promoted to additional testing; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development process.

**2 Integration Testing**

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system. 5.2.3 System Testing System testing tests a completely integrated system to verify that the system meets its requirements .For example, a system test might involve testing a logon interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logoff..

**Acceptance Testing**

Acceptance Testing is the final level of software testing. The main aim of this testing is to

determine the working process of the system by satisfying the required specifications and it is

acceptable for delivery. It is also known as End-User Testing. It also works under the Black Box

Testing Method

**Alpha Testing**

This is a type of validation testing. It is a type of *acceptance testing*which is done before the product is

released to customers. It is typically done by QA people.

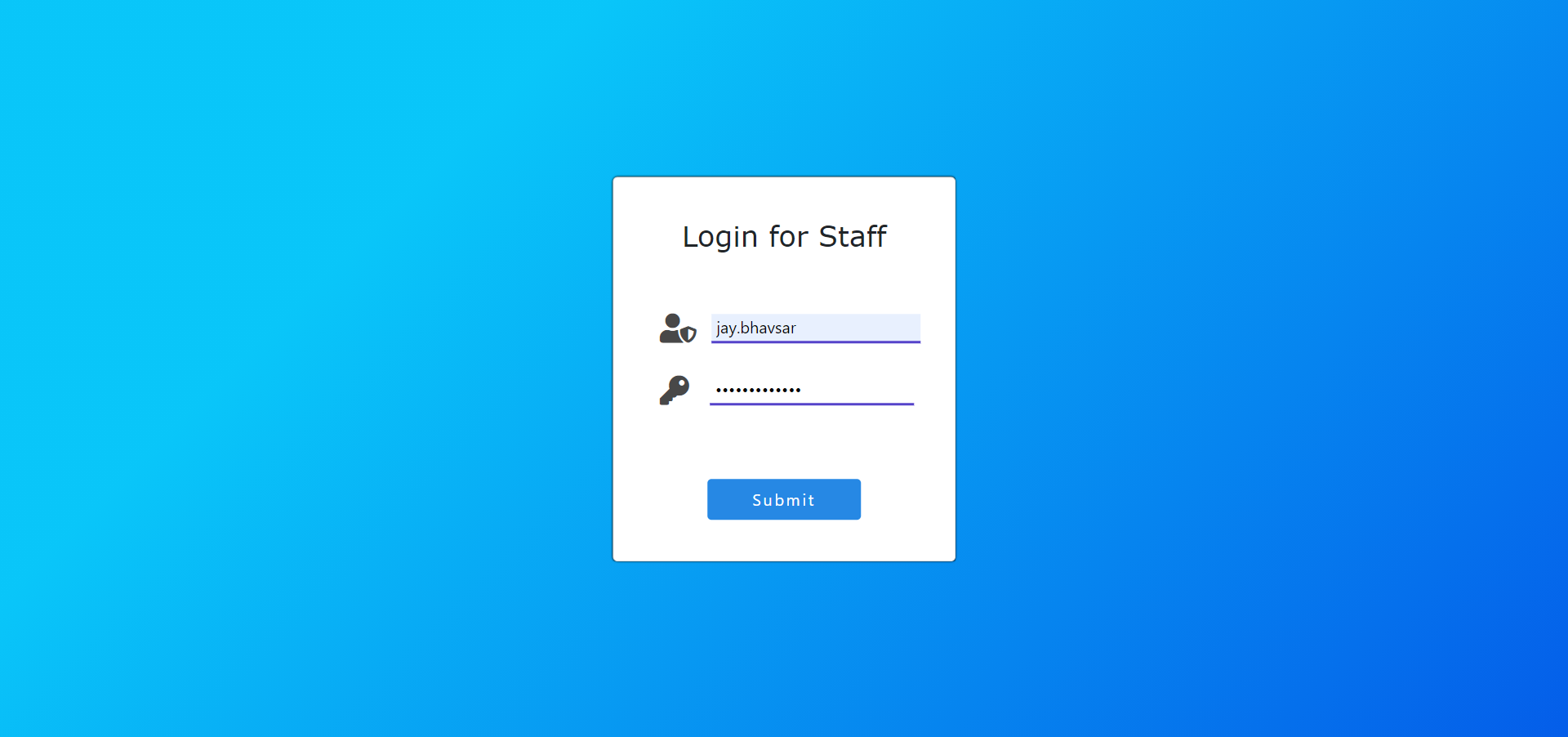
**Beta Testing**

The beta test is conducted at one or more customer sites by the end-user of the software. This

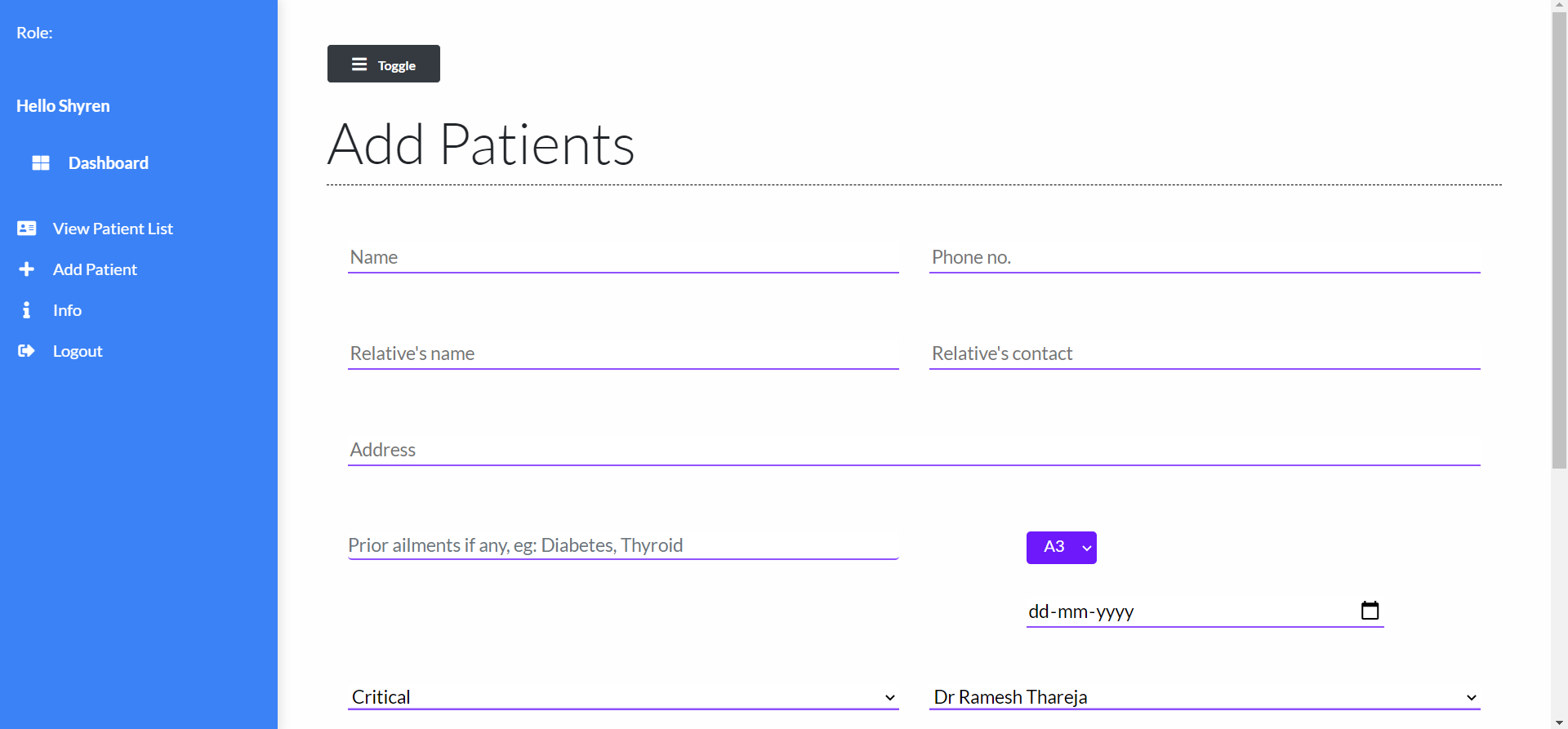
version is released for a limited number of users for testing in a real-time environment

**SCREENSHOT OF VARIOUS OPERATION**

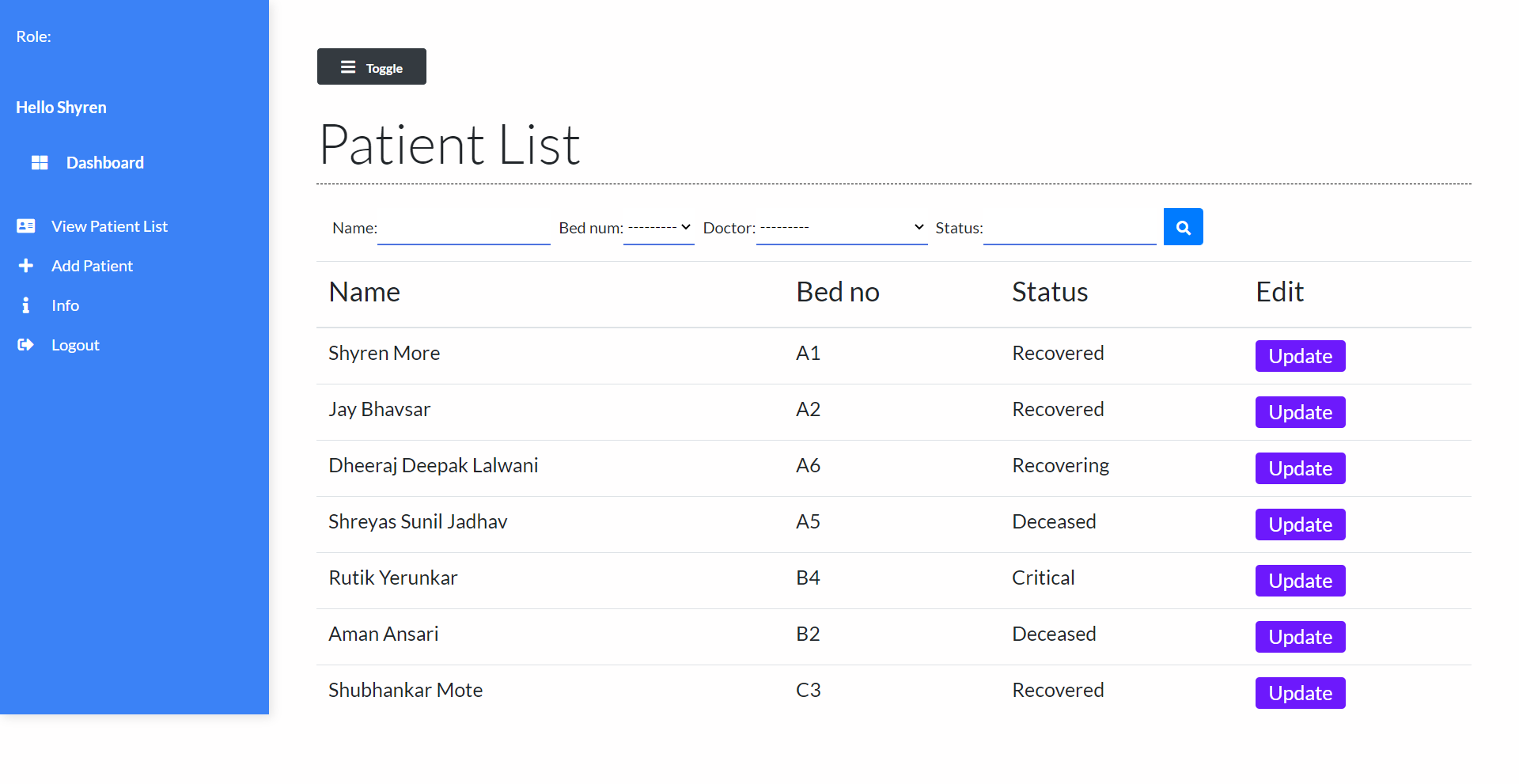
**Login page**

****

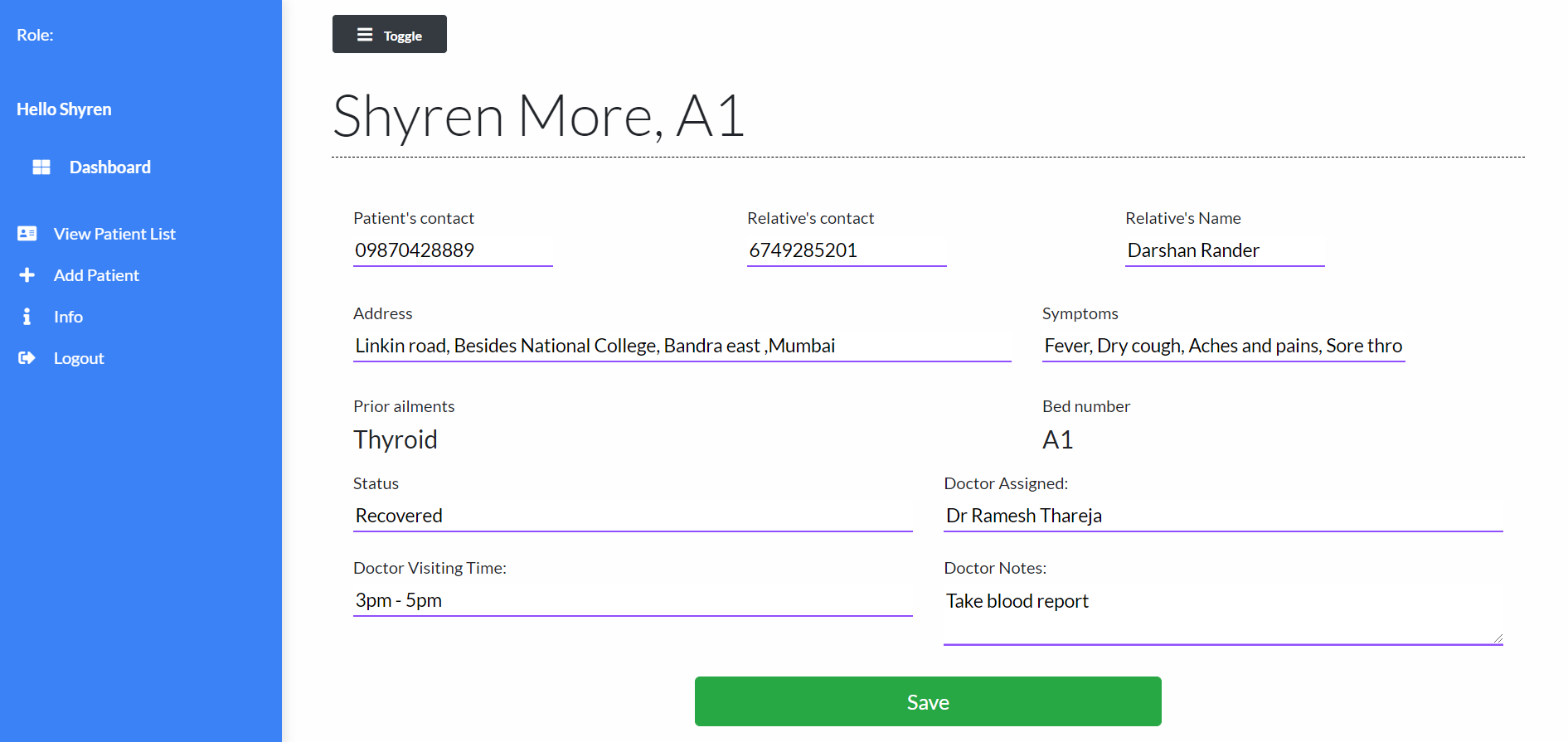
**Add Patient And Select Bed**



**Patient list**

****

**Update**

****

**REFERENCES:**

1. Fundamentals of Database Systems, RamezElmasri and Shamkant B. Navathe, 7thEdition , 2017, PEARSON
2. Database Management System , Ramakrishnan, and Gehrke ,3rd Edition, 2014, McGRAW HILL
3. Django for APi By Like Welling , PLATINUM EDITION
4. Html & Css by JON DUCKETT,2011
5. Python by Erics matthes ,2011,McGRAW HILL

**Web References:**

https://www.tutorialspoint.com/index.htm https://www.javatpoint.com

https://www.w3schools.com https://html.com