CS/SE/CE 3354 Software Engineering

Final Project Deliverable #2

Hospital Management Project

David Fonseca

Hunter Smith

Uriel Lucio

Hiba Hussain

Marisella Medoza

Jenson Antony

Jonathan Pedraza

George Zhang

Tahseen Arnan

1. Delegation of Tasks:

- Software Process Model
  - Jenson Antony
- Functional Requirements
  - David Fonseca
- Non-Functional Requirements
  - Marisella Mendoza
- Use Case Diagram
  - Uriel Lucio
- Sequence Diagrams
  - George Zhang
  - Tahseen Arnan
- Class Diagram
  - Hunter Smith
  - Tahseen Arnan
- Architectural Design
  - Hiba Hussein
  - Johnathan Pedraza
- Deliverable #1 Editors
  - Tahseen Arnan
- GitHub
  - Jenson Antony
  - Hunter Smith
- Cost Estimations
  - David Fonseca
  - Marisella Mendoza
  - Uriel Lucio
  - George Zhang
- Test Plan
  - Hunter Smith
- Conclusion & References
  - Tahseen Arnan
  - Hiba Hussein
- Presentation Slides
  - All Members
- Deliverable #2 Editors
  - Tahseen Arnan
  - Johnathan Pedraza
  - Hunter Smith

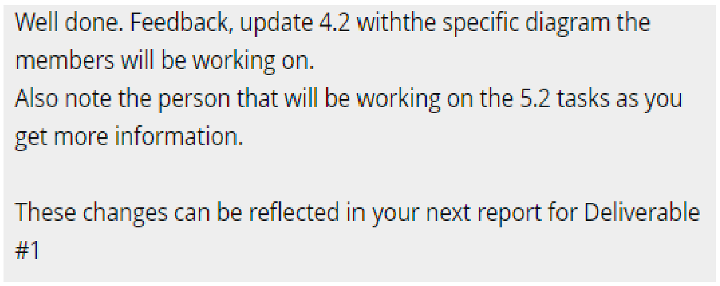
2. **Deliverable #1 Content:**

   a. **Project Goals:** Create hospital management software that displays information about departments and staff. Patients can register, login, and schedule bookings. Doctors can register, login, schedule bookings, and edit patient medical information.

   b. **Project Motivation:** We were motivated to improve on antiquated software in the healthcare industry.

   c. **Delegation of Tasks:** Tasks were delegated in the expected manner outlined by the project proposal. No students with no/poor contribution.

   d. **Proposal Feedback:**



Well done. Feedback, update 4.2 withthe specific diagram the members will be working on.
Also note the person that will be working on the 5.2 tasks as you get more information.

These changes can be reflected in your next report for Deliverable #1

- **4.2**: Tasks delegated to reflect who is working on each specific diagram.
- **5.2 task assignments:**
  - Scheduling:
    - Proposal: 9/10
    - Git Creation: 9/15
    - Deliverable #1: 9/15 - 10/06, All Team Members
    - Deliverable #2: 10/07 - 10/20, All Team Members
    - Test Plan: Jenson Antony

- **GitHub Repository:**

GitHub Link - https://github.com/3589hunter/3354-GroupPatients

- **Delegation of Tasks:**
  - Software Process Model
    - Jenson Antony
  - Functional Requirements
    - David Fonseca
  - Non-Functional Requirements
    - Marisella Mendoza
  - Use-Case Diagram
    - Uriel Lucio
  - Sequence Diagram
    - Tahseen Arnan
    - George Zhang
  - Class Diagram
    - Hunter Smith
  - Architectural Design
    - Hiba Hussein
    - Johnathan Pedraza
  - Deliverable #1 Editor
    - Johnathan Pedraza
    - Tahseen Arnan

Software Process Model:

Waterfall is the software process model involved in our project. It is due to a number of reasons mentioned below:

- **Well Defined Requirements:** If the hospital's requirements are clear and unlikely to change significantly during development, the Waterfall model provides a structured approach to document and follow these requirements throughout the project. This also provides stability to the process.
- **Structured Approach:** The linear progression through distinct phases (requirements, design, implementation, testing, deployment, and maintenance) makes it easy to track progress and manage all the tasks.
- **Easy to Manage:** The clear sequence of phases can make project management straightforward, helping project managers monitor progress and ensure that each phase is completed before moving on. By completing each phase before moving to the next, it can be easier to identify potential risks at each stage. This makes it easy to manage.
- **Clear Documentation:** Detailed documentation aids in future maintenance, training, and onboarding new staff or developers who may work on the system later. The major steps such as Requirements Analysis, System Design, Implementation, Testing, Deployment and Maintenance can easily help in proper documentation.

Software Requirements:

a. **Functional Requirements:**

   i. **Registering Page:** Patients shall have the ability to register on the website and create a personal account to keep in the system. All of their personal and medical information will be securely protected in our database.

ii. **Log-in and Account Information Retrieval:** After registering, patients shall have the ability to go to our login page and securely login with an authentication method. After logging in patients shall be able to view their accounts and information including personal and medical information.

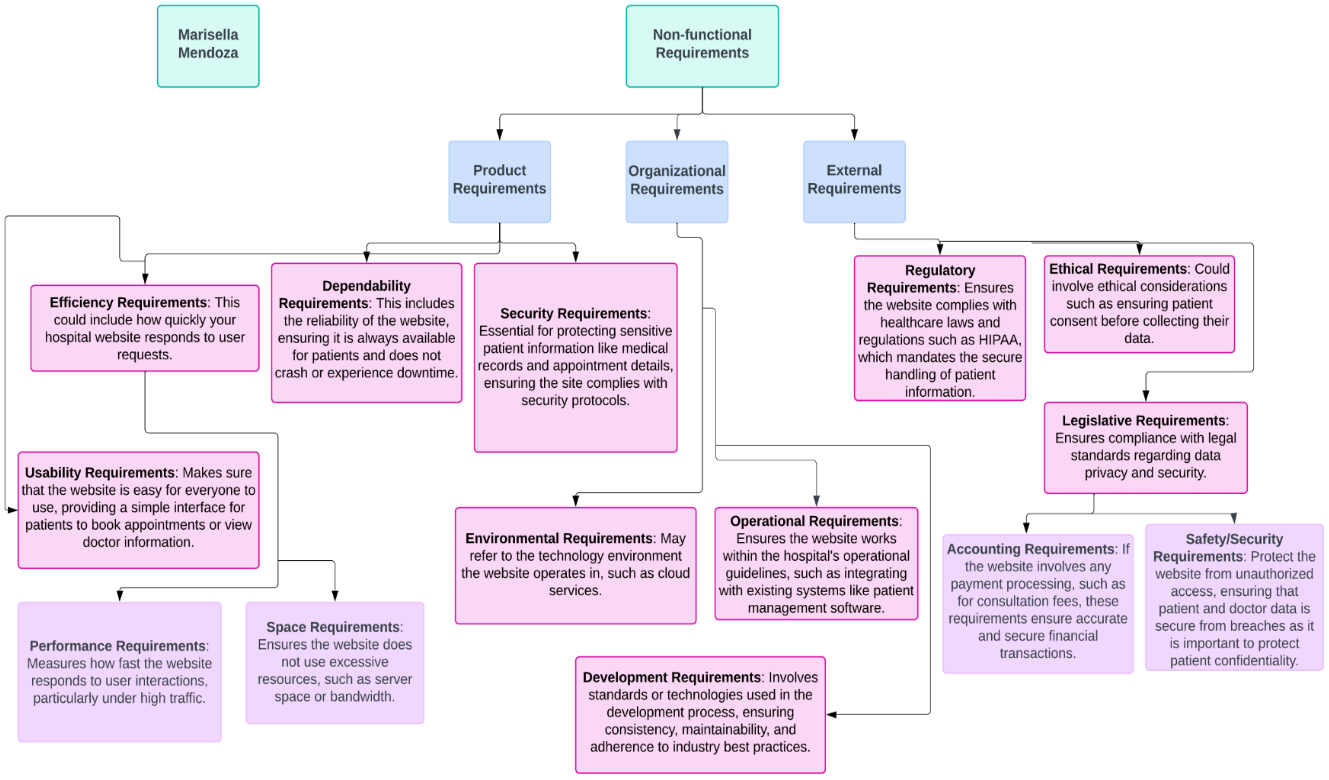
iii. **Homepage:** One of the key pages of the website will provide patients with an overview of the hospital, offering quick and easy access to all fundamental services and information. It will highlight our facilities, licenses, services, and the overall environment, creating a hospitable experience for both current and future patients.

iv. **Doctor and Department Pages:** Patients will be able to explore a directory of hospital departments and doctors, each with its own page on our website. The doctor directory will give patients detailed info, such as their licenses, specialties, qualifications, availability, and contact details. The department page will showcase all the areas our hospital specializes in, offering patients clear and specific information about each one.

v. **Appointments Page:** Our website will feature a page where patients can easily book appointments with doctors. This page will let patients check a doctor's availability, choose a preferred time for their visit, and review both past and upcoming appointments to stay on top of their schedule.

vi. **Contact Us Page:** Our website will feature a Contact Us page where patients can find contact details for hospital administration and support staff, along with the addresses of our facilities. This allows patients to reach out directly for any booking or general inquiries without needing to speak to a doctor.

Non-Functional Requirements:



Efficiency Requirements: This could include how quickly your hospital website responds to user requests.

Dependability Requirements: This includes the reliability of the website, ensuring it is always available for patients and does not crash or experience downtime.

Security Requirements: Essential for protecting sensitive patient information like medical records and appointment details, ensuring the site complies with security protocols.

Regulatory Requirements: Ensures the website complies with healthcare laws and regulations such as HIPAA, which mandates the secure handling of patient information.

Ethical Requirements: Could involve ethical considerations such as ensuring patient consent before collecting their data.

Legislative Requirements: Ensures compliance with legal standards regarding data privacy and security.

Usability Requirements: Makes sure that the website is easy for everyone to use, providing a simple interface for patients to book appointments or view doctor information.

Environmental Requirements: May refer to the technology environment the website operates in, such as cloud services.

Operational Requirements: Ensures the website works within the hospital's operational guidelines, such as integrating with existing systems like patient management software.

Accounting Requirements: If the website involves any payment processing, such as for consultation fees, these requirements ensure accurate and secure financial transactions.

Safety/Security Requirements: Protect the website from unauthorized access, ensuring that patient and doctor data is secure from breaches as it is important to protect patient confidentiality.

Performance Requirements: Measures how fast the website responds to user interactions, particularly under high traffic.

Space Requirements: Ensures the website does not use excessive resources, such as server space or bandwidth.

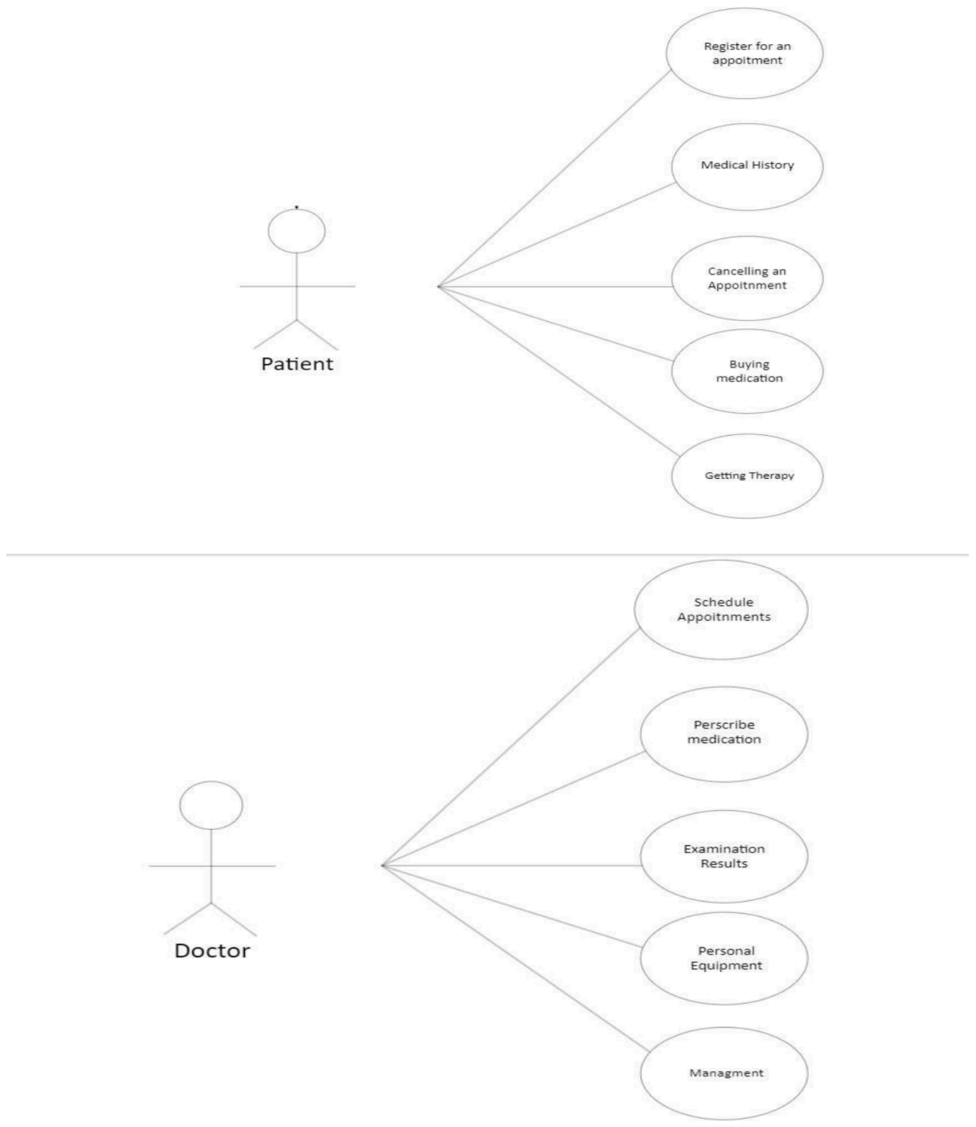
Development Requirements: Involves standards or technologies used in the development process, ensuring consistency, maintainability, and adherence to industry best practices.

Assumptions:

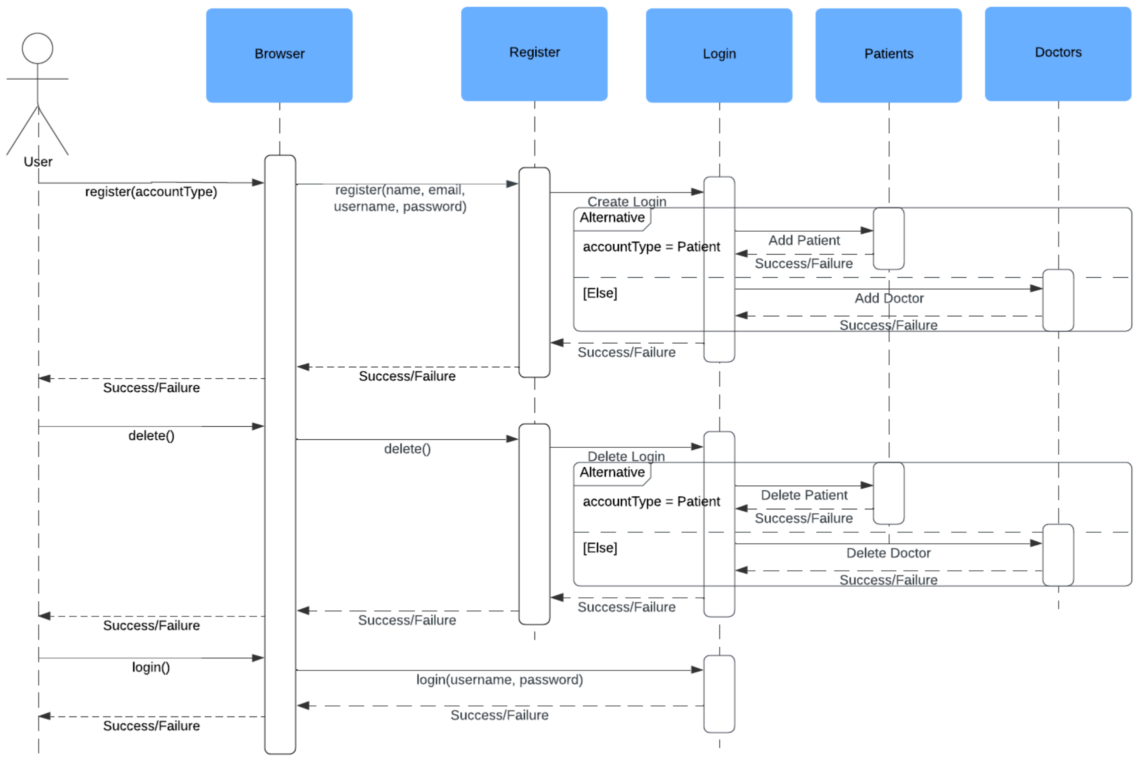
- **Patient Data Protection:** Hospitals will keep personal health information safe and secure through an end-to-end encrypted process.
- **Health Information Sharing:** The health records will only be shared with the authorized personnel. Only the authorized staff and specialists will be able to access the information.
- **Patient Safety and Quality Improvement:** Safety is the key element. Focused training will be done every day to make the environment safe.
- **Telemedicine Access:** Virtual consultations will be done every day for the convenience of customers.

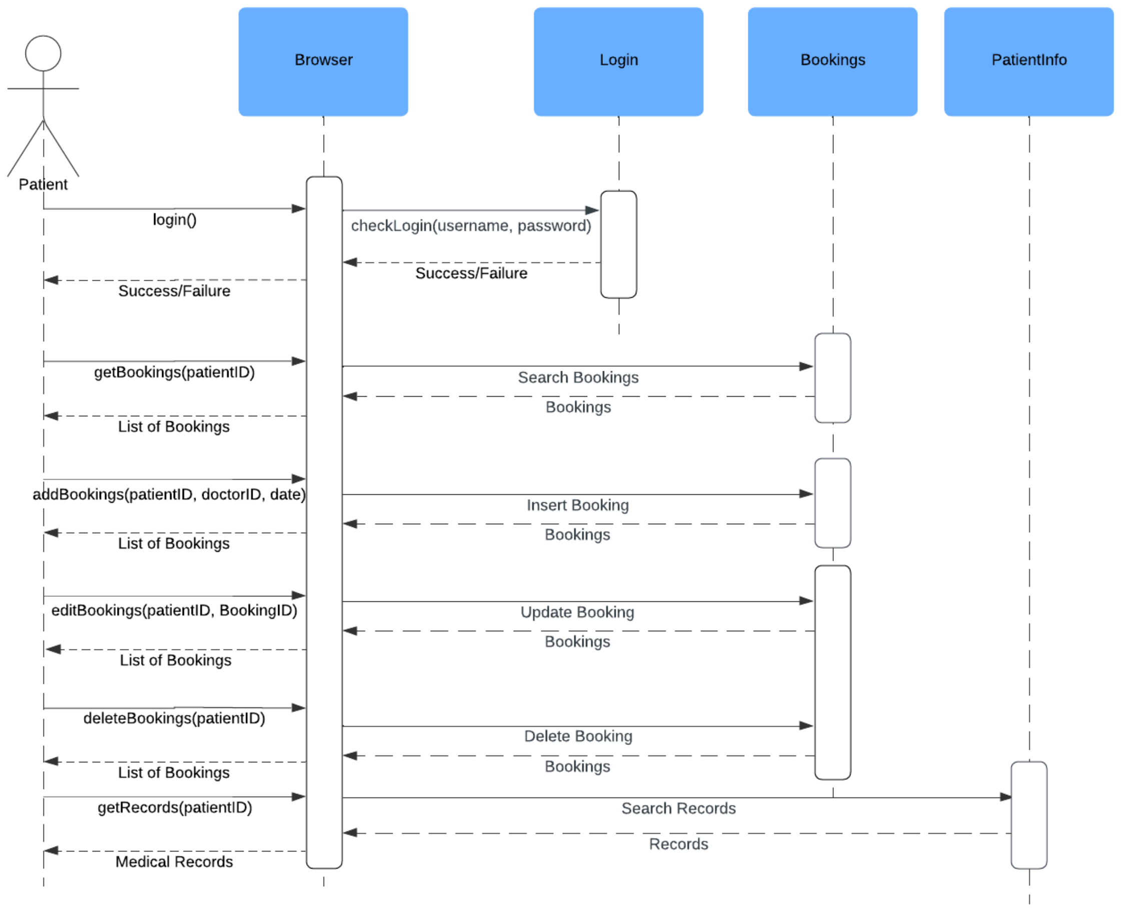
- **Emergency Preparedness:** Emergency drills for fires and other crises will be done regularly to ensure staff are aware of how to navigate through them.
- **Customer Rights:** Everybody will be treated equally and justly. Patients' advocates will be available in case there is an issue with customer rights.
- **Staff Training:** Staff will be regularly trained to keep up with the evolving technologies and practices of medicine.
- **Facility Safety:** Regular inspections will be done to make sure the facility is up to standard.

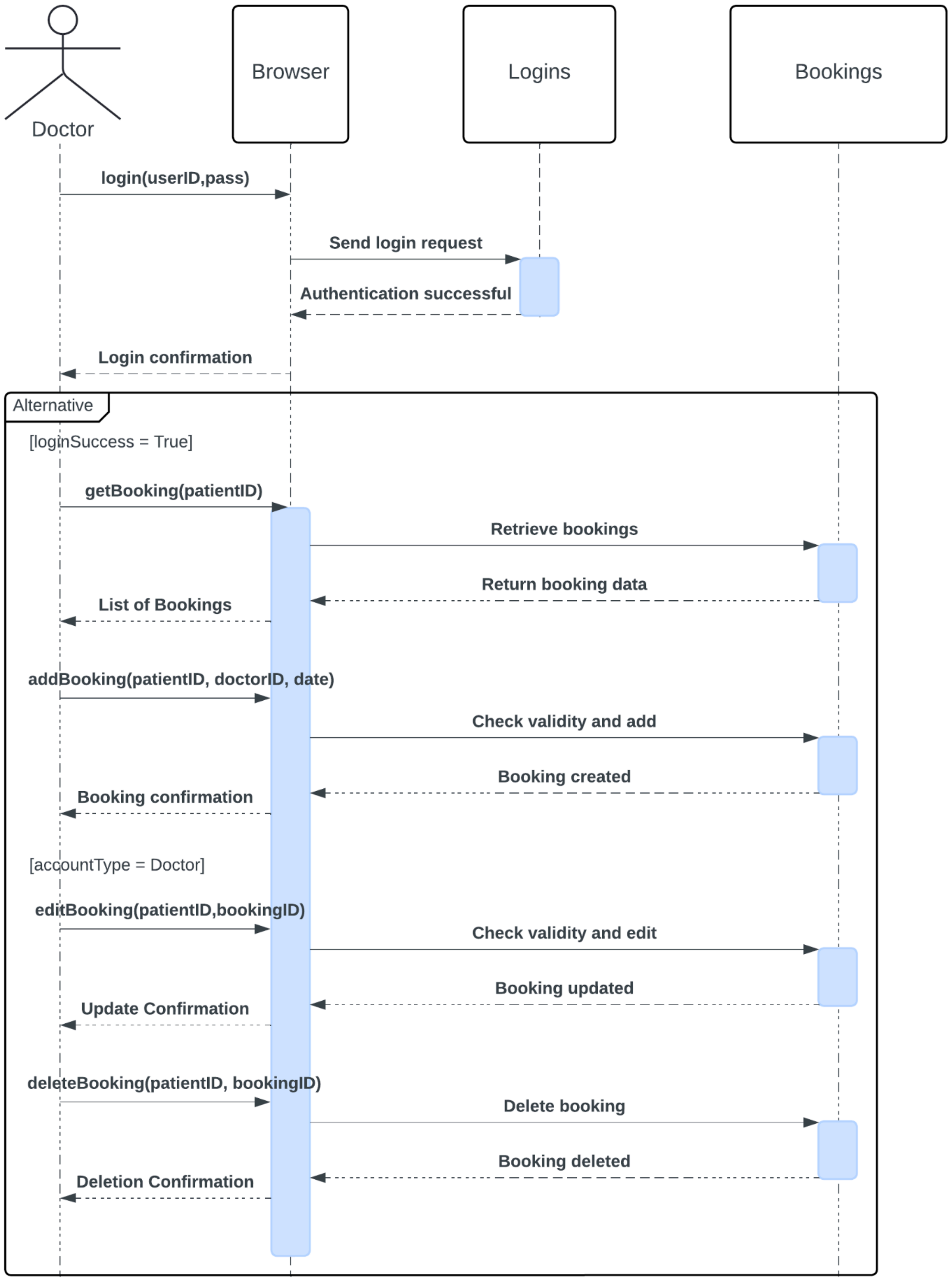
Use Case Diagram:

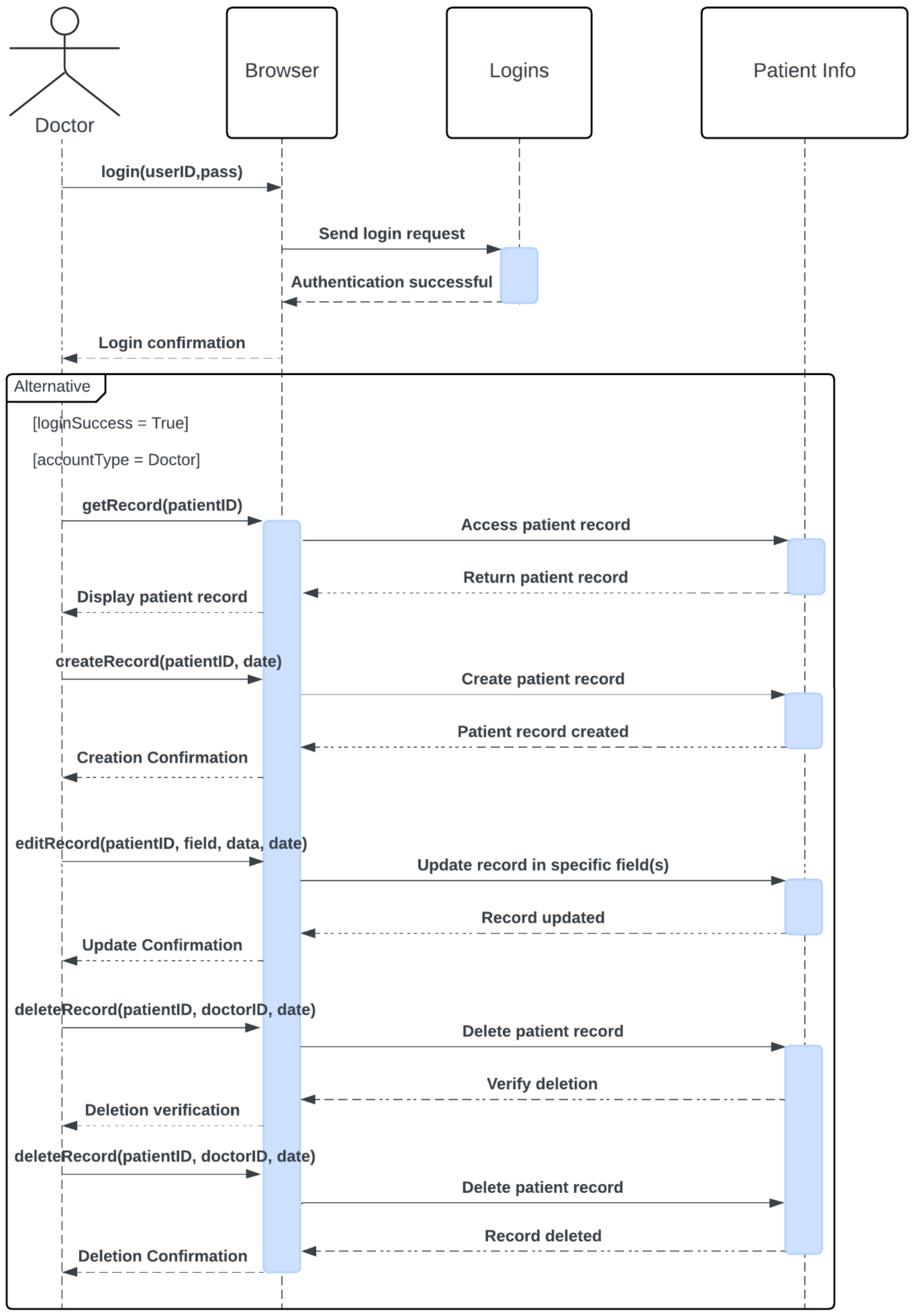


Sequence Diagrams:

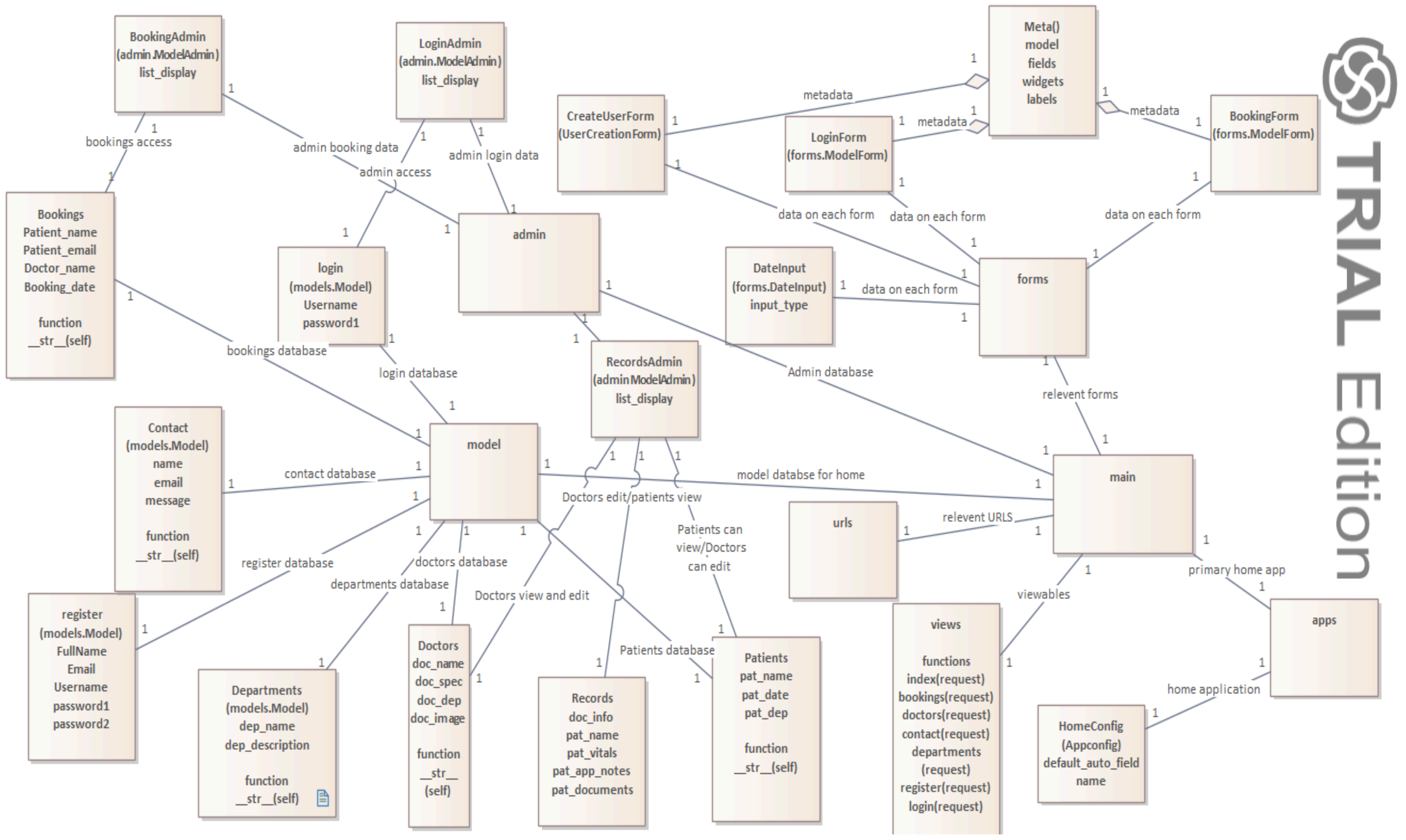




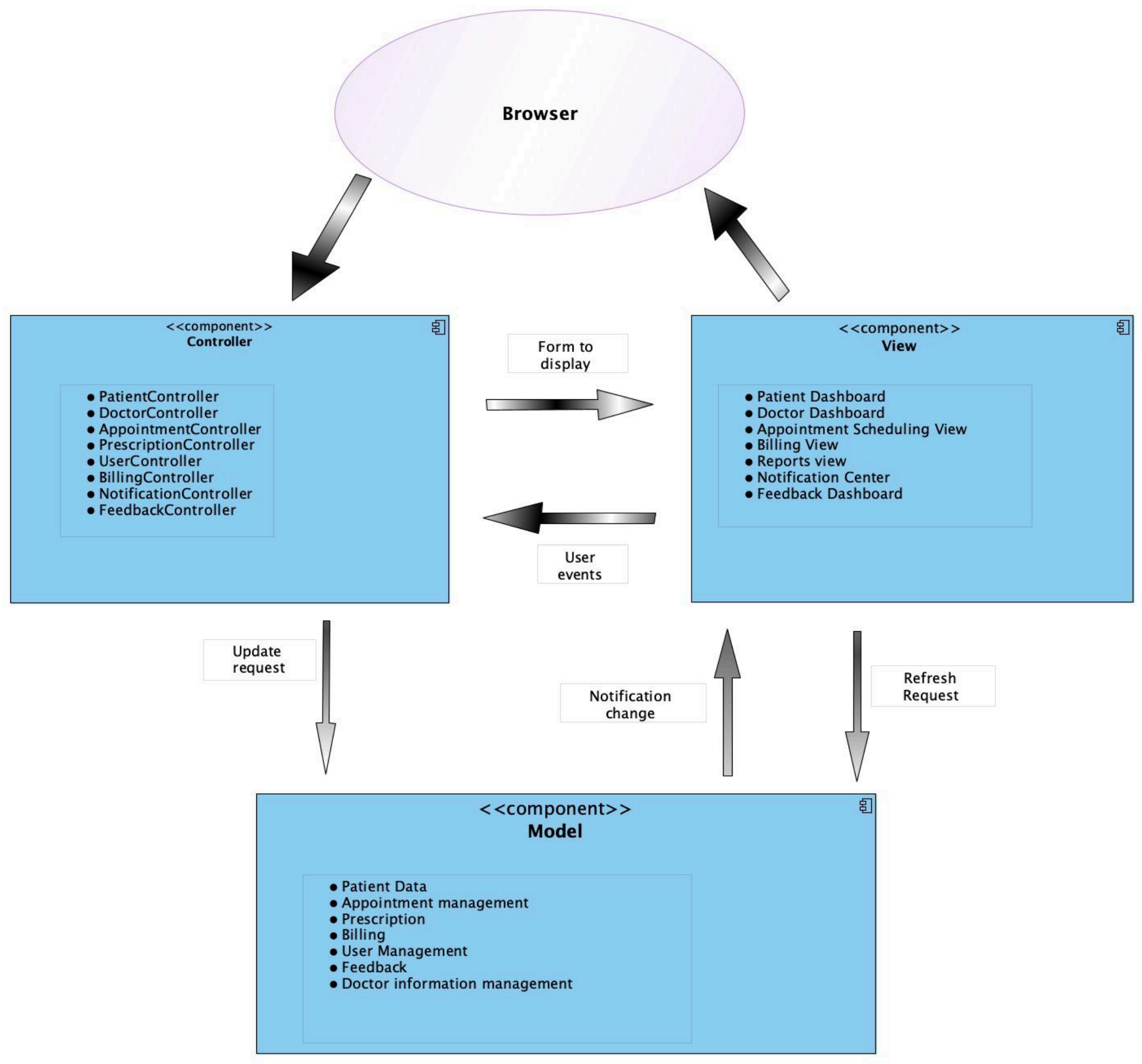

Doctor
Browser
Logins
Bookings
login(userID,pass)
Send login request
Authentication successful
Login confirmation
Alternative
[loginSuccess = True]
getBooking(patientID)
Retrieve bookings
Return booking data
List of Bookings
addBooking(patientID, doctorID, date)
Check validity and add
Booking created
Booking confirmation
[accountType = Doctor]
editBooking(patientID,bookingID)
Check validity and edit
Booking updated
Update Confirmation
deleteBooking(patientID, bookingID)
Delete booking
Booking deleted
Deletion Confirmation


Doctor
Browser
Logins
Patient Info
login(userID,pass)
Send login request
Authentication successful
Login confirmation
Alternative
[loginSuccess = True]
[accountType = Doctor]
getRecord(patientID)
Access patient record
Return patient record
Display patient record
createRecord(patientID, date)
Create patient record
Patient record created
Creation Confirmation
editRecord(patientID, field, data, date)
Update record in specific field(s)
Record updated
Update Confirmation
deleteRecord(patientID, doctorID, date)
Delete patient record
Verify deletion
Deletion verification
deleteRecord(patientID, doctorID, date)
Delete patient record
Record deleted
Deletion Confirmation

Class Diagram:


BookingAdmin (admin.ModelAdmin) list_display
LoginAdmin (admin.ModelAdmin) list_display
CreateUserForm (UserCreationForm)
LoginForm (forms.ModelForm)
Meta() model fields widgets labels
BookingForm (forms.ModelForm)
Bookings Patient_name Patient_email Doctor_name Booking_date function __str__(self)
login (models.Model) Username password1
admin
DateInput (forms.DateInput) input_type
forms
RecordsAdmin (admin.ModelAdmin) list_display
Contact (models.Model) name email message function __str__(self)
model
main
urls
register (models.Model) FullName Email Username password1 password2
Departments (models.Model) dep_name dep_description function __str__(self)
Doctors doc_name doc_spec doc_dep doc_image function __str__ (self)
Records doc_info pat_name pat_vitals pat_app_notes pat_documents
Patients pat_name pat_date pat_dep function __str__(self)
views functions index(request) bookings(request) doctors(request) contact(request) departments (request) register(request) login(request)
apps
HomeConfig (Appconfig) default_auto_field name
bookings access
admin booking data
admin access
admin login data
metadata
data on each form
bookings database
login database
Admin database
relevent forms
contact database
model databse for home
Doctors edit/patients view
Patients can view/Doctors can edit
relevent URLS
register database
doctors database
departments database
Doctors view and edit
Patients database
viewables
primary home app
home application
1

Architectural Design:



3. **Project Scheduling, Cost, Effort, and Pricing Estimation**
   a. **Project Scheduling**
      i. **Start Date:** September 10, 2024.
      Justification: As noted in the update for #5.2 in the delivery #1 report, we finished our project proposal on 09/10/24, which meant that we could then immediately start working on designing our project.

      ii. **Schedule:** Weekdays 9 AM - 5 PM: 8 working hours per weekday
      On Fridays at 3pm the team would meet to discuss project progress and project duties.

      iii. **End Date:** October 16, 2024 Justification: As calculated in the cost section, this project would take around 1.2 months.
   b. **Cost, Effort, and Pricing Estimation**
      i. We will be using **Application Composition** to estimate the amount of effort needed for this project [1]. Application Composition uses application points to estimate the amount of person-months needed for a project.

      ii. **Estimates of application points:**
      User Authentication: 1 login screen, 1 report containing usernames and passwords, all simple = 3

      Profile and Personal Information: 2 screens to view and edit information, 1 report containing user information, all simple = 4

      Scheduling: 2 screens, one for visit details, one for the calendar view. 2 reports, one to handle the schedule and one to handle doctor availability. All simple = 6

      Patient Medical Information and Doctor Notes: 2 screens for viewing and editing information, 2 reports for patient information and notes = 6

Patient/Doctor communication: 1 screen for messages, 1 report for messages = 3

Billing and Payment: 1 screen for billing, 1 report for billing information = 3

Management: 3 screens to see information on scheduling, doctor and patient accounts, and website management = 3

Security: 1 report = 2

iii. **Total NAP: 3+4+6+6+3+3+3+2 = 30** [1]

Estimated number of developers: 5, very low experience. PROD=4
Assumed reuse%: 20%.

Thus, PM = NAP*(1 – reuse%/100)/PROD = 30*(4/5)/4 = 6
Person-Months 6/5 = **1.2 Months** [1]

c. **Estimated Cost of Hardware Products**

i. **Server costs:**
Estimated size of data center: 500 square feet [2]
Costs for a small data center: $800,000-$2,000,000 [2]

Server Racks: $1,500 per 42U rack [2]
Servers: $3,000 per server unit [2]

Servers per rack * server individual cost * number of racks:
42 * 3000 * 10 = **$1,260,000** [2]

ii. **Wiring costs:**
Electrical Installment: $28,000-$70,000 [3]
Backup Generator: $15,000 [3]

Network cable/fiber optic cost: $500-$1,000 per data point [4]

iii. **Ongoing costs:**
Labor costs: $50-$150 per hour [2]
Electricity: $150 per kW/month [3]
Internet: $80 per/month [4]

iv. **Additional Costs:**
Additional cost for hardware failure: 10% failure/year [2]
Permits, inspections, and troubleshooting: $1,000-$5,000 [2]

d. **Estimated Cost of Software Products**

i. **Core Software Costs:**
Software Package (Licensed): $50,000 - $100,000 [5]
Comprehensive Software with Customization: $100,000 - $250,000 (includes more extensive features like EHR, analytics, and integrations) [6]

ii. **Additional Costs**
Implementation and Setup: $20,000 - $50,000 (initial training, data migration, configuration, etc.) [5]
Annual Maintenance & Support: 15-20% of the software cost per year (around $7,500 - $50,000, depending on initial software investment) [5]
User Licenses (for subscription-based models): $30 - $100 per user per month [6]

iii. **Infrastructure (if on-premises)**
Cloud Hosting (if SaaS): $1,000 - $3,000 per month (scales based on usage) [6]

iv. **Estimated Total Cost for a moderately sized hospital:**
Initial Investment (first year): $100,000 - $250,000 [5]
Annual Operating Cost: $10,000 - $50,000 (for support, maintenance, and possible cloud hosting fees) [5]

e. **Estimated Cost of Personnel**

   i. **Number of Developers:** The recommendation is a team of 5 developers based on projects of this scope and complexity.
      Frontend Developer(s): Mreates user interface and experience.
      Backend Developers(s): Maintains server, database, and application logic. They make sure that the data is secure within the hospital systems.
      DevOps Engineer(s): Handle the maintenance, scaling, and deployment of the website.
      Cost of Developers: Based on location and experience salaires can vary. In the U.S, the average annual salaries are:
      Frontend Developer: $112,039 [7]
      Backend Developer: $158,331 [7]
      DevOps Engineer(s): $125,908 [7]

   ii. **Maintenance (Six-month duration):**
      Calculated at 12% of development costs.
      Estimated at $39,627.80 [2]

   iii. **Training Cost (Six-month duration):**
      A one time cost of $5,000 [2]

   iv. **Project Management (Six-month duration):**
      Estimated at $112,039 [7]

   v. **Total Costs for Five Developers over Six Months:**
      Estimated at $817,130.15 [7]

4. **Test Plan for Project:**
   a. Because our application does not have any incredibly simple testable pieces of code. Simply because what the application does at its core is just make database calls. Therefore, what we decided to test was one of those database calls and make sure the call succeeded. The language we used is python and the testing software we decided to use is pytest. In order to test this database call we needed to create a mocker and test this mocker call to see if we pulled the correct information.

   b. **Code:**

```
import pytest
from .forms import Bookingform
from django.test import TestCase
from django.test import Client
from django.urls import reverse
from unittest.mock import patch
# Create your tests here.
class testClass:
   def test_my_function(mocker):
   # Create a mock object for the class containing is_valid()
      mocker.patch('path.to.module.is_valid', return_value=True)
   # Mock the requests.post function
      mock_post = mocker.patch('requests.post')
      mock_post.return_value.status_code = 200
     mock_post.return_value.json.return_value = {'success': True}
   # Use the mock object in your test
     result = Bookingform(mock_post)
   # Assert the expected behavior based on the mocked is_valid()
      assert result == "Valid"
```

   c. The result of this test demonstrated that when we make the database call through Bookingform we do get a valid result. Therefore, demonstrating the Bookingform call works properly.

5. **Comparison of Our work with similar designs:**
    a. A similar design to our hospital management system is '**Oracle Health**' (previously known as Cerner Millennium). Similarly to our project, Oracle Health is a platform used for managing health information. They also have tools for patient registration, scheduling, with a focus on data protection and the ability for the software to exchange information between departments; which our project can also do [8].

        Oracle Health uses a three-tier architecture: The Database tier, Application Tier, and Client tier [9]. So it shares some similarities to our project since our project employs the MVC pattern. In both Oracle Health and in our project, there is a layer that is responsible for handling the data (Model/Database), and there is also a separate layer used for the user interface (View/Client).

        The difference is that Oracle Health's 'Application' tier focuses more on handling business logic as opposed to our project's 'Controller' pattern in which it focuses more on managing the flow between the Model and the View [9].

    b. Another similar design to our project is **Epic Systems**' healthcare management platform. Epic also allows patients to log in, save their personal/medical data, and their functionality allows them to schedule appointments directly [10]. In terms of architecture however, Epic relies on a client-server architecture and their highly integrated backend allows them to store patient information/data across various departments, which is different to our project which uses the MVC pattern. Both architectural patterns do share some similarities however.

6. Conclusion:

The creation of the hospital management system website with the use of Python has successfully demonstrated its ability to streamline and optimize the day-to-day tasks of healthcare administrators. This website features functionality for managing patient records, appointments, staff details, and billing processes, all in a secure and efficient manner.

Throughout this project, we focused on feature implementation in a way that would ensure improved accuracy and data management, all the while decreasing paperwork for healthcare staff. These characteristics in conjunction ensure that our website leads to higher staff productivity and better patient outcomes overall. The integration of the Python libraries within Django allowed for our healthcare management system to be strong, stable, and easily scalable, making sure the website can be easily adapted to the needs of different healthcare facilities.

One of the major changes that we made to Deliverable #1 was the class diagram. In the first class diagram we did not include a records or patient class and we did not include associations between the databases and their admin access. Therefore, in the final deliverable we added the correct associations and included both a patient and records database that can be edited by the doctors and viewed by the patients. We made these changes as we wanted the Bookings data and Medical Records data to be handled in a similar way for ease of use.

Going forward, further development could include the implementation of data analytics capability, real-time staff and patient notifications, and more robust security protocols. In conclusion, our hospital management system website provides a solid foundation for a variety of modern healthcare facilities, and shows the importance of technological development in a field where software is often antiquated and difficult to use.

7. References:

[1] “Designing the architecture of your outsystems applications”, *Outsystems*, May 8, 2024. https://success.outsystems.com/documentation/best_practices/architecture/designing_the_architecture_of_your_outsystems_applications/application_composition/ (accessed Nov. 10, 2024).

[2] N. Coulter, “How much does it cost to build a data center? | Eziblank,” *Eziblank*, Nov. 29, 2022. https://www.eziblank.com/how-much-does-it-cost-to-build-a-data-center/ (accessed Nov. 10, 2024).

[3] Estimator53, “Data Center Electrical Wiring Cost Estimator - Estimate Florida Consulting,” *Estimate Florida Consulting*, Feb. 05, 2024. https://estimatorflorida.com/data-center-electrical-wiring-cost-estimator/ (accessed Nov. 10, 2024).

[4] “Data Cabling for Office: A Comprehensive Guide to Understanding Costs,” *Amorserv.*, 2023. https://amorserv.com/insights/data-cabling-for-office-a-comprehensive-guide-to-understanding-costs (accessed Nov. 10, 2024).

[5] InfoStride Tech Hub, “Hospital Management Software Guide: Benefits & Development,” *InfoStride*, Jun. 18, 2024. https://infostride.com/hospital-management-software-guide/ (accessed Nov. 10, 2024).

[6] A3Logics, “Hospital Management Software in USA: 15 Top Solutions for 2024,” *A3Logics*, Mar. 27, 2024. https://www.a3logics.com/blog/top-hospital-management-software-in-usa/ (accessed Nov. 10, 2024).

[7] “Front End Developer salary in the United States,” *Indeed,* https://www.indeed.com/career/front-end-developer/salaries (accessed Nov. 10, 2024).

[8] “Patient Access | Oracle Health,” *Oracle*, 2023. https://www.oracle.com/health/revenue-cycle/patient-access/ (accessed Nov. 10, 2024).

[9] “Architecture,” *Oracle Help Center*, 2022. https://docs.oracle.com/en/industries/health-sciences/clinical-remote-capture/5.4/upgrade-guide/architecture.html (accessed Nov. 10, 2024).

[10] J. Chishtie *et al.*, “Use of Epic Electronic Health Record System for Health Care Research: Scoping Review,” *Journal of Medical Internet Research*, vol. 25, no. 1, Dec. 2023, doi: https://doi.org/10.2196/51003 (accessed Nov. 10, 2024).