DIGITAL HEALTHCARE

SMART QUEUING MANAGEMENT SYSTEM

SQMS

Overview

Overview

1. Introduction

Overview

1. Introduction
2. **Problems**

Overview

1. Introduction
2. Problems
3. **Objectives**

Overview

1. Introduction
2. Problems
3. Objectives
4. **Proposed System**

Overview

1. Introduction
2. Problems
3. Objectives
4. Proposed System
5. **Implementation**

Overview

1. Introduction
2. Problems
3. Objectives
4. Proposed System
5. Implementation
6. **Conclusion**

Overview

1. Introduction
2. Problems
3. Objectives
4. Proposed System
5. Implementation
6. Conclusion
7. **References**

Overview

1. Introduction
2. Problems
3. Objectives
4. Proposed System
5. Implementation
6. Conclusion
7. References
8. **Roadmap to future**

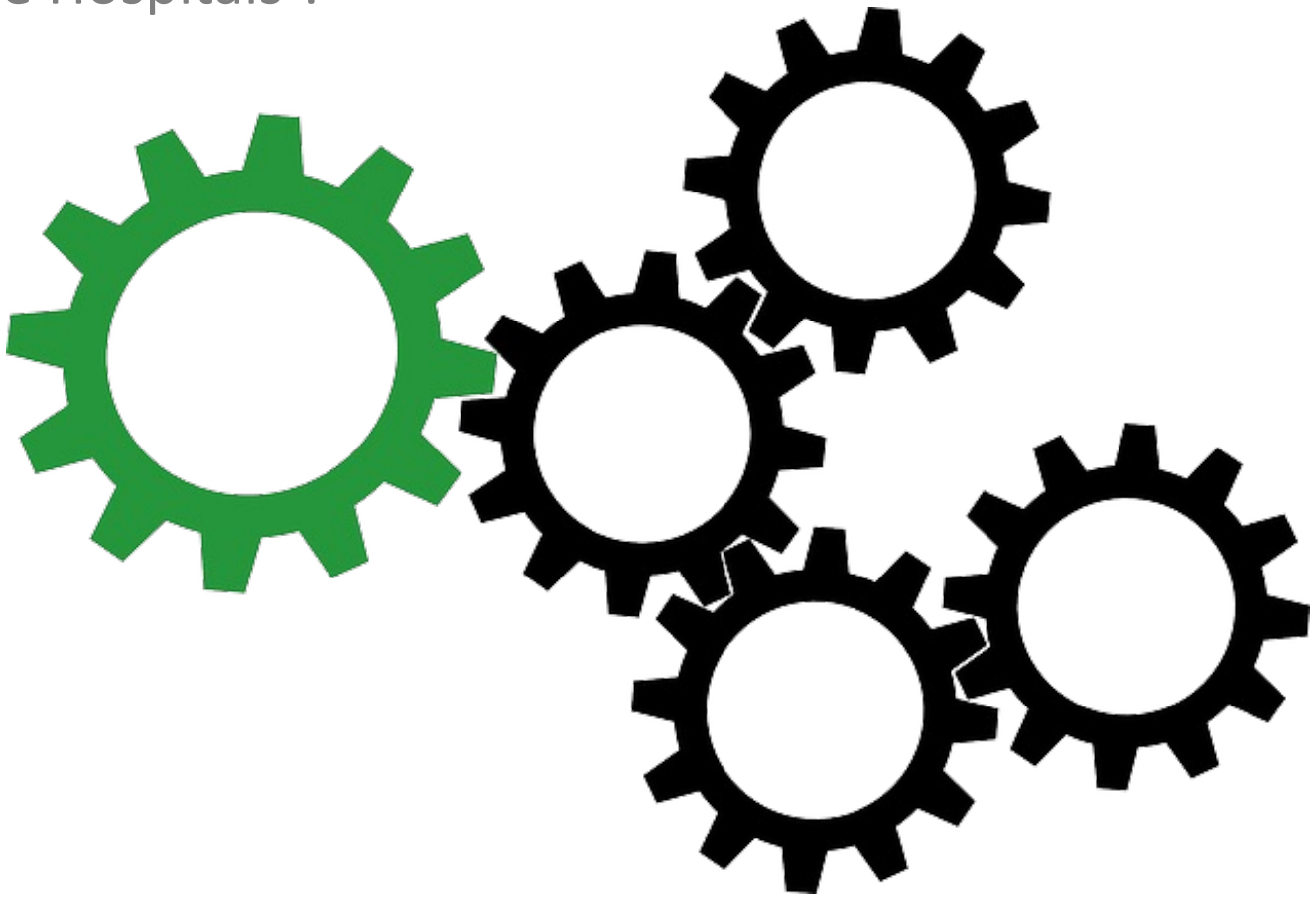
Introduction

Introduction

- **What is Queue Management Inside Hospitals ?**

Introduction

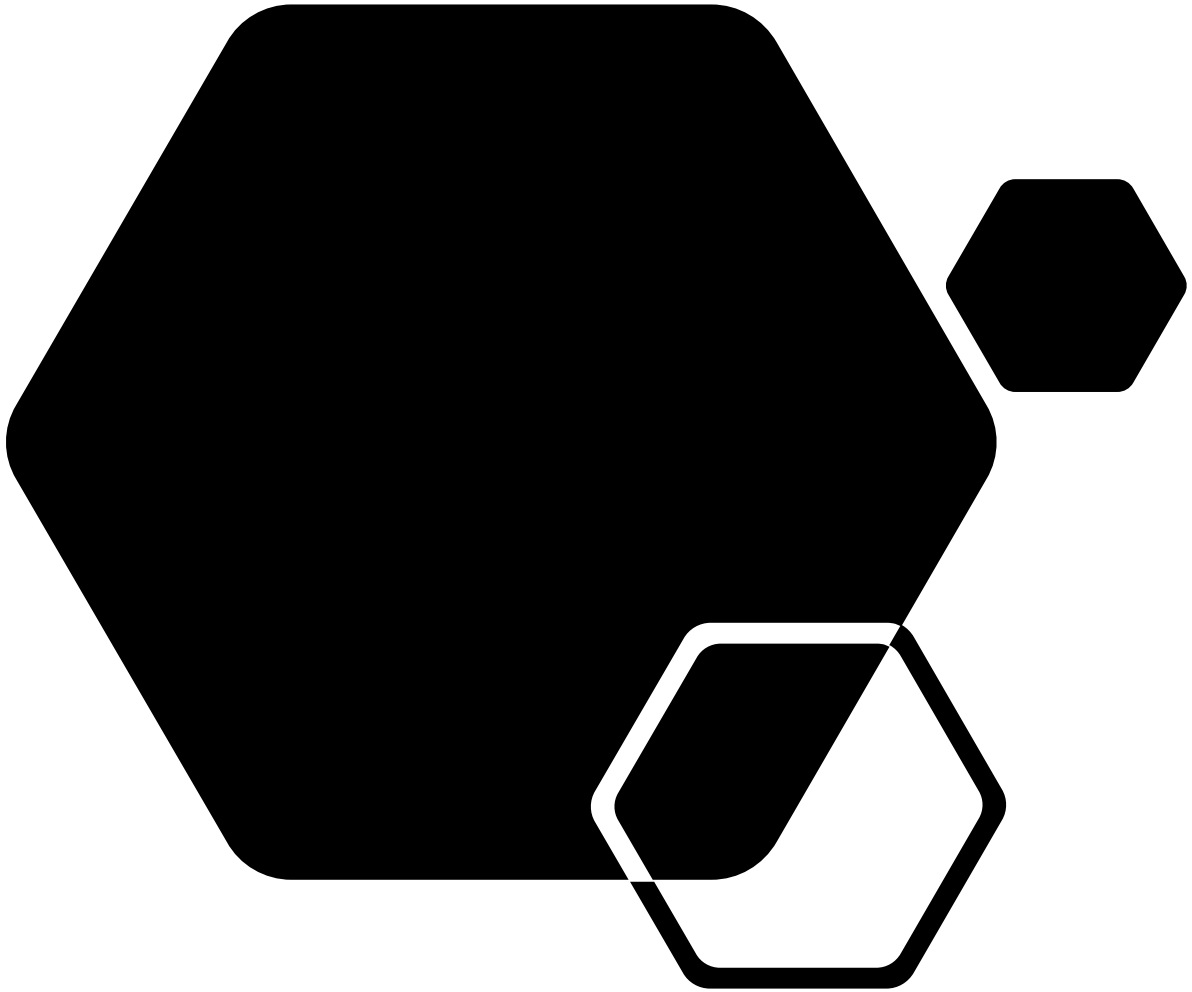
- What is Queue Management Inside Hospitals ?
- **How is queue managed ?**



Introduction

- What is Queue Management Inside Hospitals ?
- How is queue managed ?
- **Appointments and Time allocation Process**

What is Queue Management inside Hospitals?

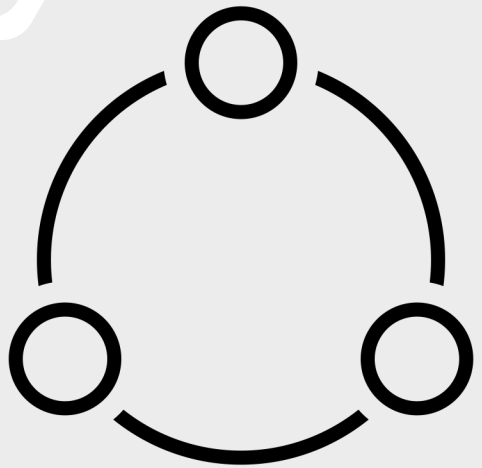


What is Queue Management inside Hospitals

- **Provides information**

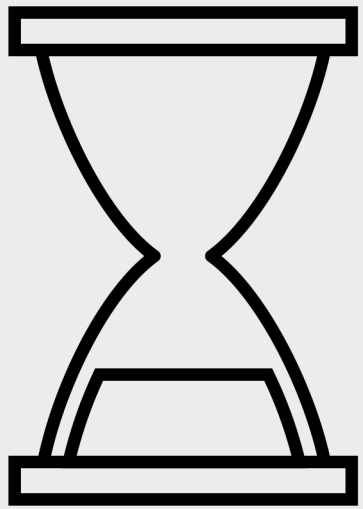
What is Queue Management inside Hospitals

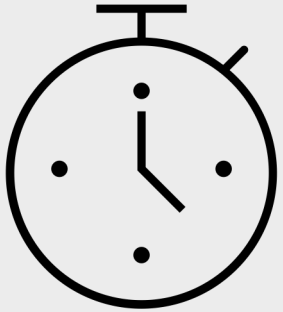
- Provides information
- **Management of patient flow**



What is Queue Management inside Hospitals

- Provides information
- Management of patient flow
- **Reduces patient wait times**





What is Queue Management inside Hospitals

- Provides information
- Management of patient flow
- Reduces patient wait times
- **Avoid chaotic situations**

What is Queue Management inside Hospitals

- Provides information
- Management of patient flow
- Reduces patient wait times
- Avoid chaotic situations
- **Simplify consultation process**

How Queue is Managed in Hospitals ?

How Queue is Managed in Hospitals ?

- Appointment Registration

How Queue is Managed in Hospitals ?

- Appointment Registration
- Time Slot allocation F.C.F.S

How Queue is Managed in Hospitals ?

- Appointment Registration
- Time Slot allocation F.C.F.S
- Token Generation

How Queue is Managed in Hospitals ?

- Appointment Registration
- Time Slot allocation F.C.F.S
- Token Generation
- Arrivals

How Queue is Managed in Hospitals ?

- Appointment Registration
- Time Slot allocation F.C.F.S
- Token Generation
- Arrivals
- Late Arrivals

How Queue is Managed in Hospitals ?

- Appointment Registration
- Time Slot allocation F.C.F.S
- Token Generation
- Arrivals
- Late Arrivals
- Management of Late arrivals

How Queue is Managed in Hospitals ?

- Appointment Registration
- Time Slot allocation F.C.F.S
- Token Generation
- Arrivals
- Late Arrivals
- Management of Late arrivals
  - Insert in rear

How Queue is Managed in Hospitals ?

- Appointment Registration
- Time Slot allocation F.C.F.S
- Token Generation
- Arrivals
- Late Arrivals
- Management of Late arrivals
  - Insert in rear
  - Insert in front

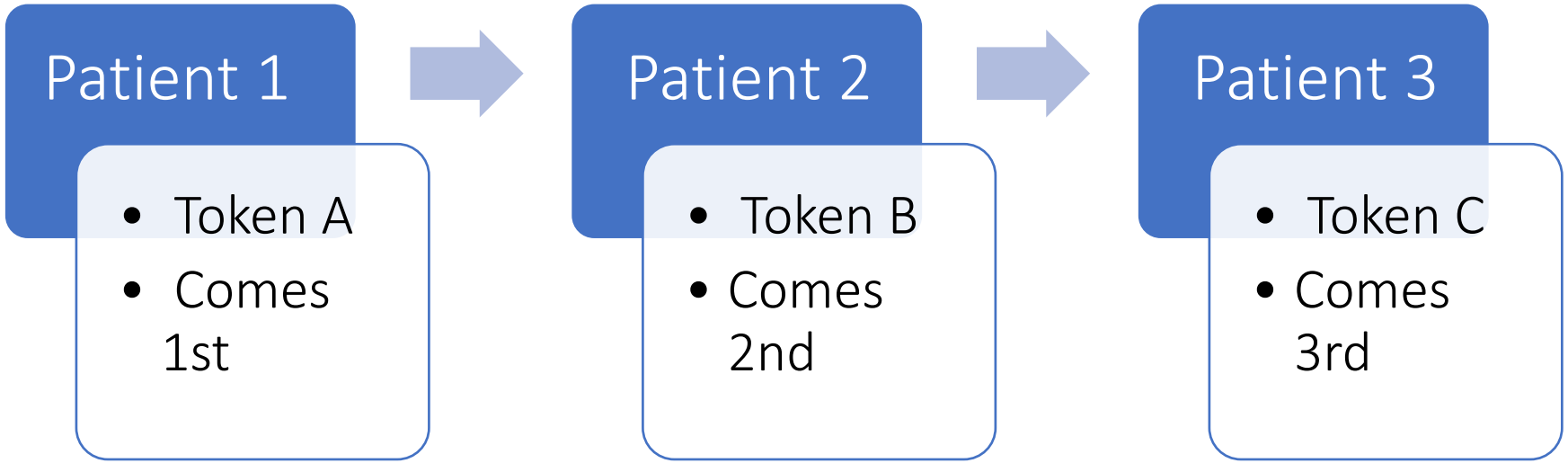
How Queue is Managed in Hospitals ?

- Appointment Registration
- Time Slot allocation F.C.F.S
- Token Generation
- Arrivals
- Late Arrivals
- Management of Late arrivals
  - Insert in rear
  - Insert in front
  - Insert randomly

Appointments and time slot allocation

Appointments and time slot allocation

- First Come First Serve (F.C.F.S)



Appointments and time slot allocation

- First Come First Serve (F.C.F.S)
  - Time Slot

Appointments and time slot allocation

- First Come First Serve (F.C.F.S)
  - Time Slot
  - Token

Problems

Problems

- **Conventional Queue management**

Problems

- **Conventional Queue management**
  - No Standard System

Problems

- **Conventional Queue management**
  - No Standard System
  - No proper organization for latecomers

Problems

- **Conventional Queue management**
  - No Standard System
  - No proper organization for latecomers
  - No Age Factor (Senior Citizen)

Problems

- **Conventional Queue management**
  - No Standard System
  - No proper organization for latecomers
  - No Age Factor (Senior Citizen)
  - Long Queues

Problems

- **Conventional Queue management**
  - No Standard System
  - No proper organization for latecomers
  - No Age Factor (Senior Citizen)
  - Long Queues
  - Waiting time

Problems

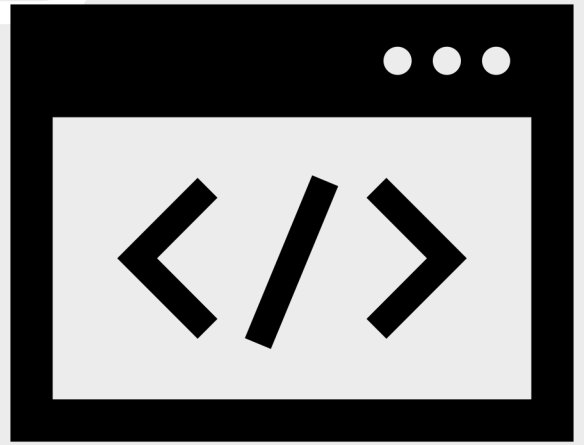
- **Conventional Queue management**
  - No Standard System
  - No proper organization for latecomers
  - No Age Factor (Senior Citizen)
  - Long Queues
  - Waiting time
  - chaos

Fakhra

Objectives

Objectives

- **Redesign the Smart Queue management**

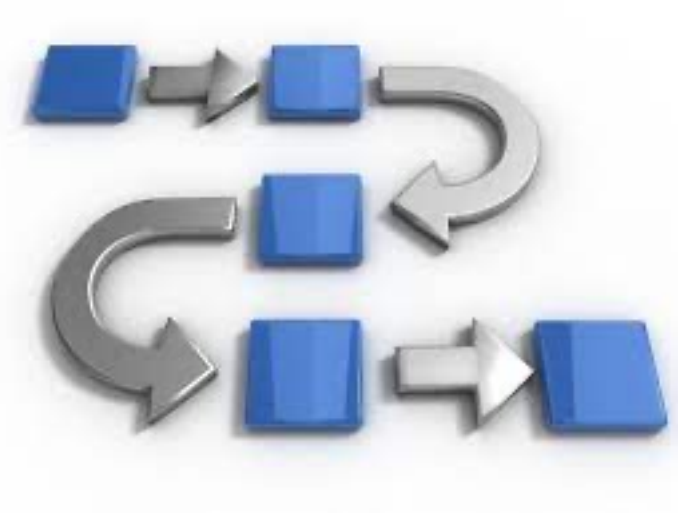


Objectives

- Redesign the Smart Queue management
- **Develop an Algorithm**

Objectives

- Redesign the Smart Queue management
- Develop an Algorithm
  - **Improve Patient flow management**



Objectives

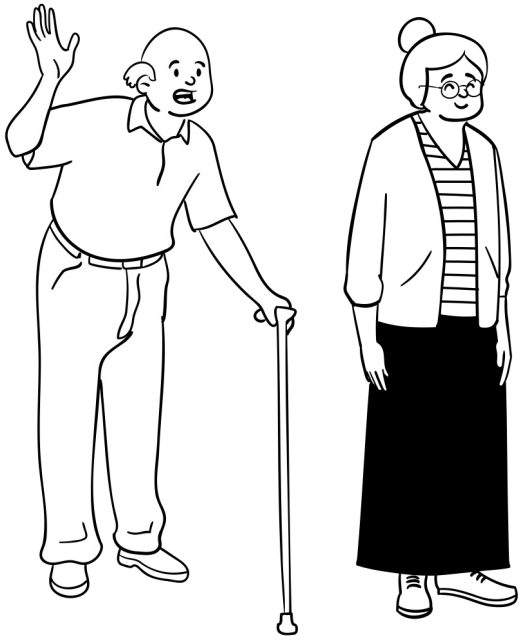
- Redesign the Smart Queue management
- Develop an Algorithm
  - Improve Patient flow management
  - **Minimize waiting time**

Objectives

- Redesign the Smart Queue management
- Develop an Algorithm
  - Improve Patient flow management
  - Minimize waiting time
  - **Organize delayed patients**

Objectives

- Redesign the Smart Queue management
- Develop an Algorithm
  - Improve Patient flow management
  - Minimize waiting time
  - Organize delayed patients
  - **Prioritize senior citizens**



Objectives

- Redesign the Smart Queue management
- Develop an Algorithm
  - Improve Patient flow management
  - Minimize waiting time
  - Organize delayed patients
  - Prioritize senior citizens
  - **Reduce queue size**

Objectives

- Redesign the Smart Queue management
- Develop an Algorithm
  - Improve Patient flow management
  - Minimize waiting time
  - Organize delayed patients
  - Prioritize senior citizens
  - Reduce queue size
- **Automate the process using web technologies.**

Algorithm

First Come First Serve Modified F.C.F.S.M [7]

1. Dynamic Priority

1. Score (senior citizens)

2. Tribonacci sequence (Latecomer Organization)

3. Threshold (Avoid Long queues)

Algorithm

- **Handling Late Patients**

  1. **Tribonacci series [6]**

     Sequence of integer such that

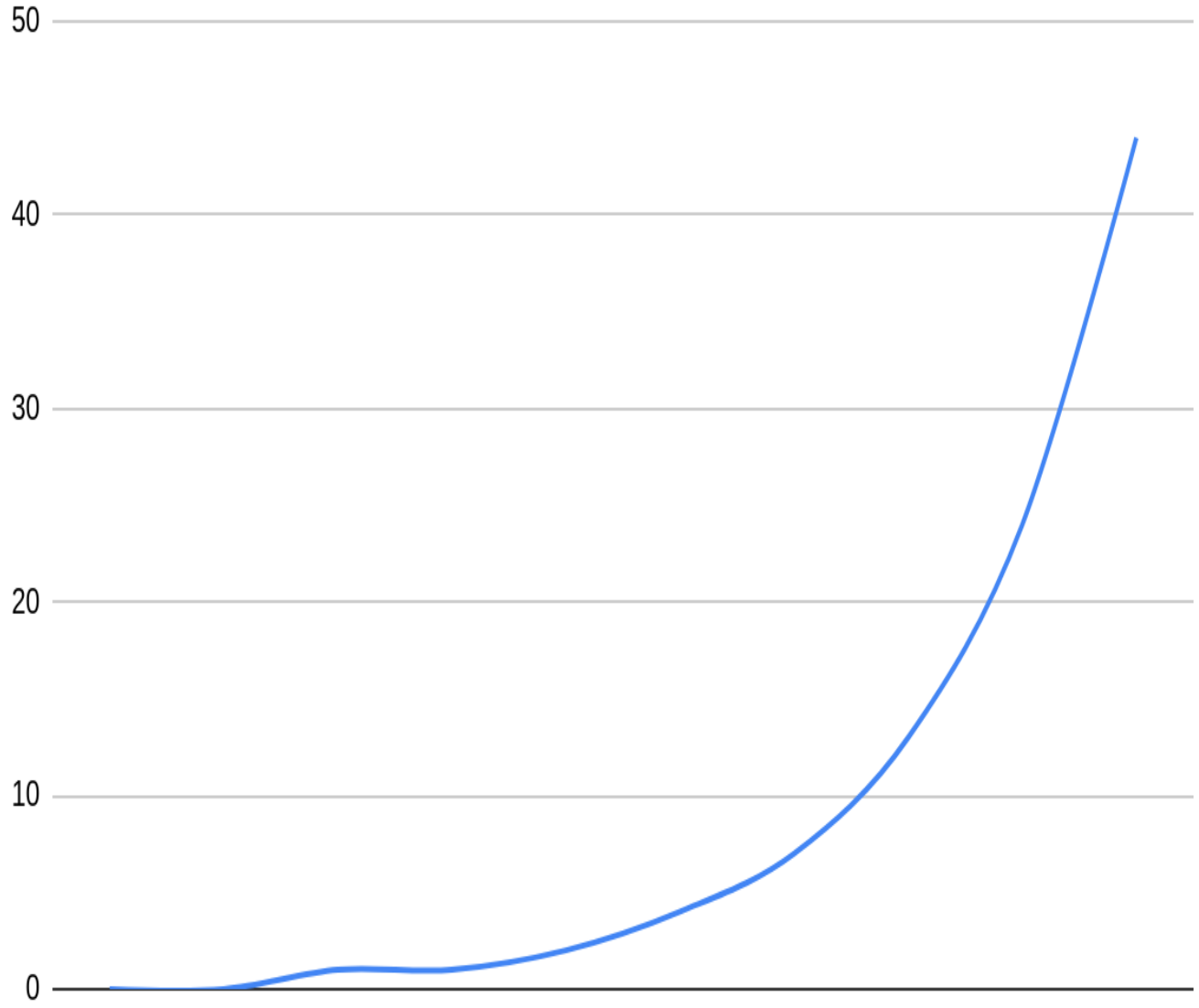
     T0 = 2
     T1 = 3
     T2 = 4

     Tn = Tn-1 + Tn-2 + Tn-3 (n >= 3)

Algorithm

- **Handling Late Patients**

    1. **Tribonacci series**



Algorithm

- **Handling Late Patients**

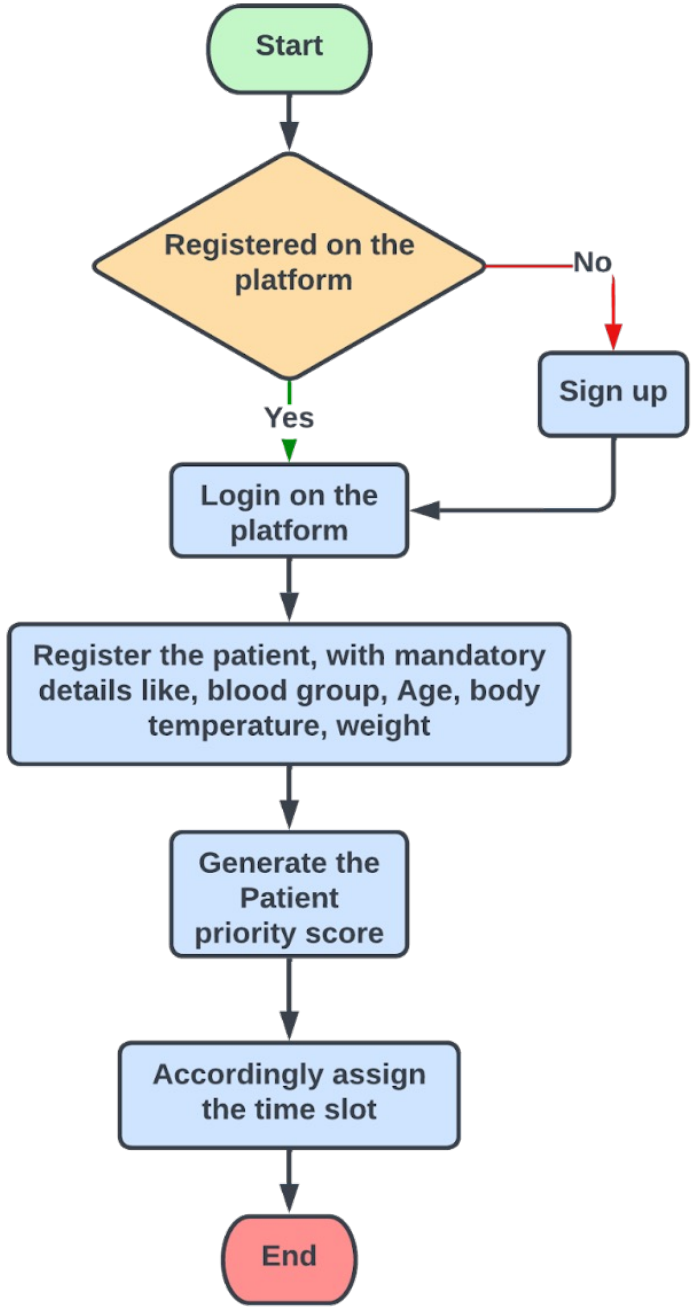
  2. Dynamic / Changing Priority | Aging[7]

$$P_n = P_{n-1} + T_n + score$$

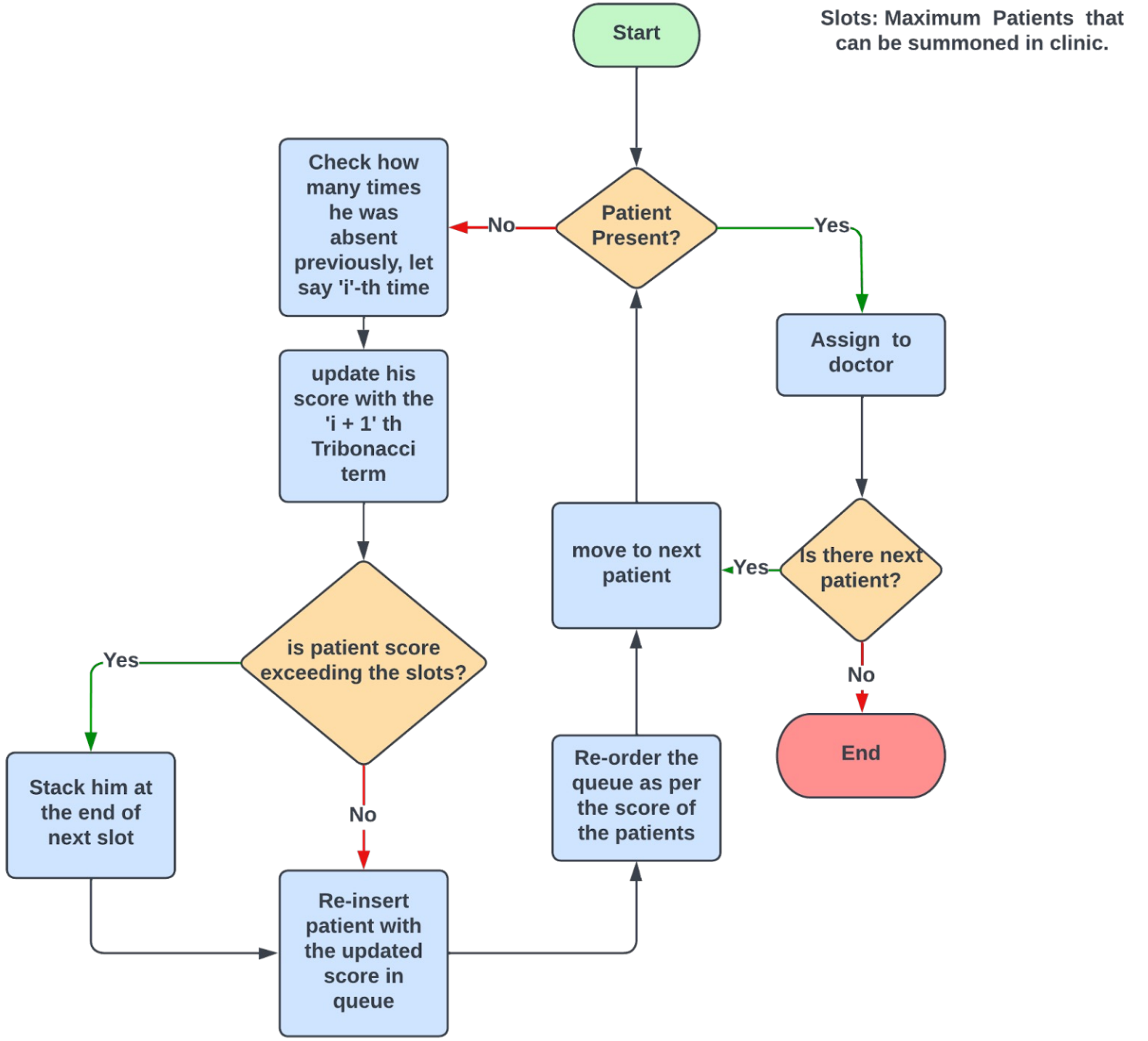
P_n : Priority of current patient in queue (nth absence)
T_n : Tribonacci sequence after n turns (nth absence)

Score is patient dependent variable ex. Age, appointment time

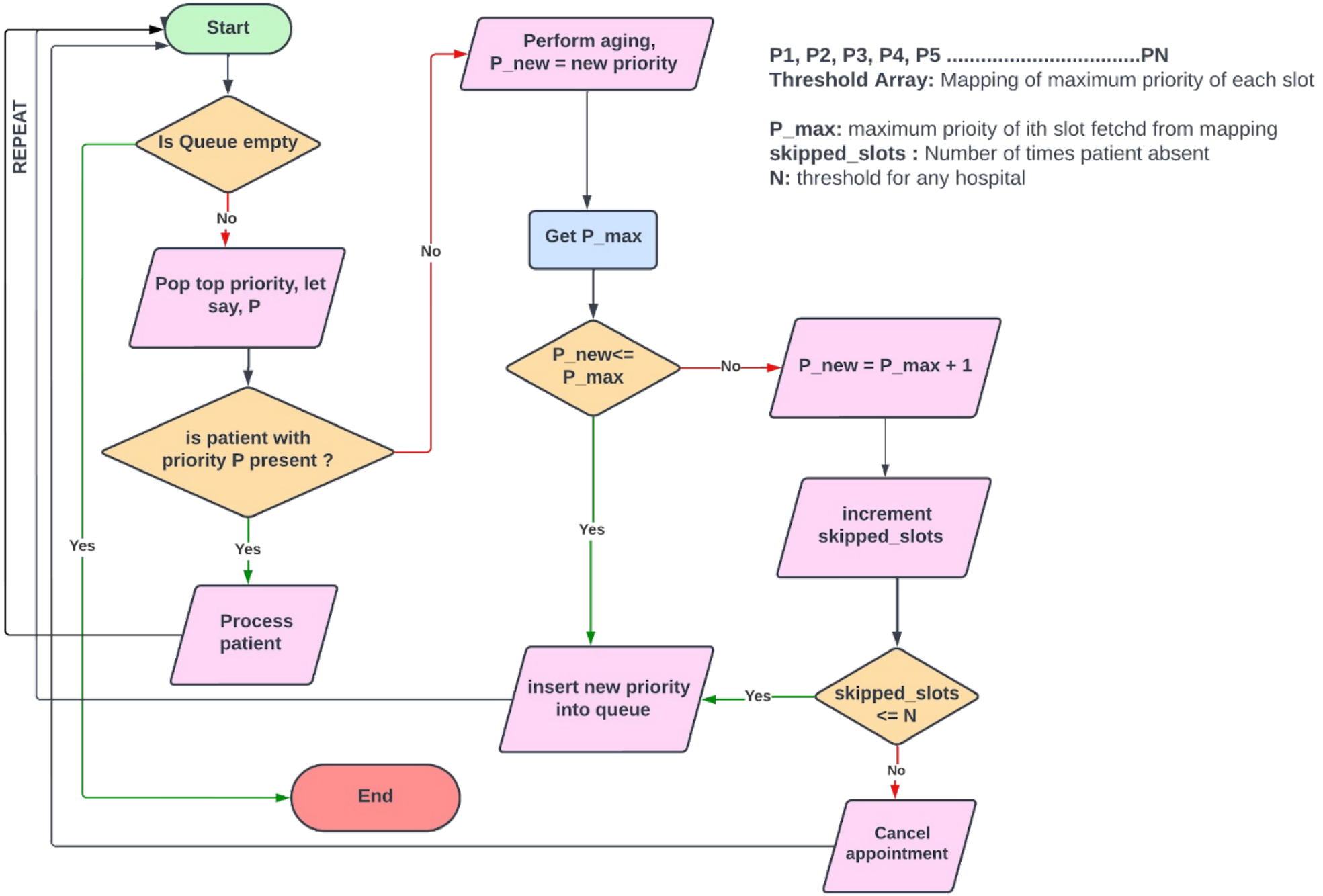
APPOINTMENT BOOKING



PATIENT DELAY MANAGEMENT



ALGORITHM WORKING

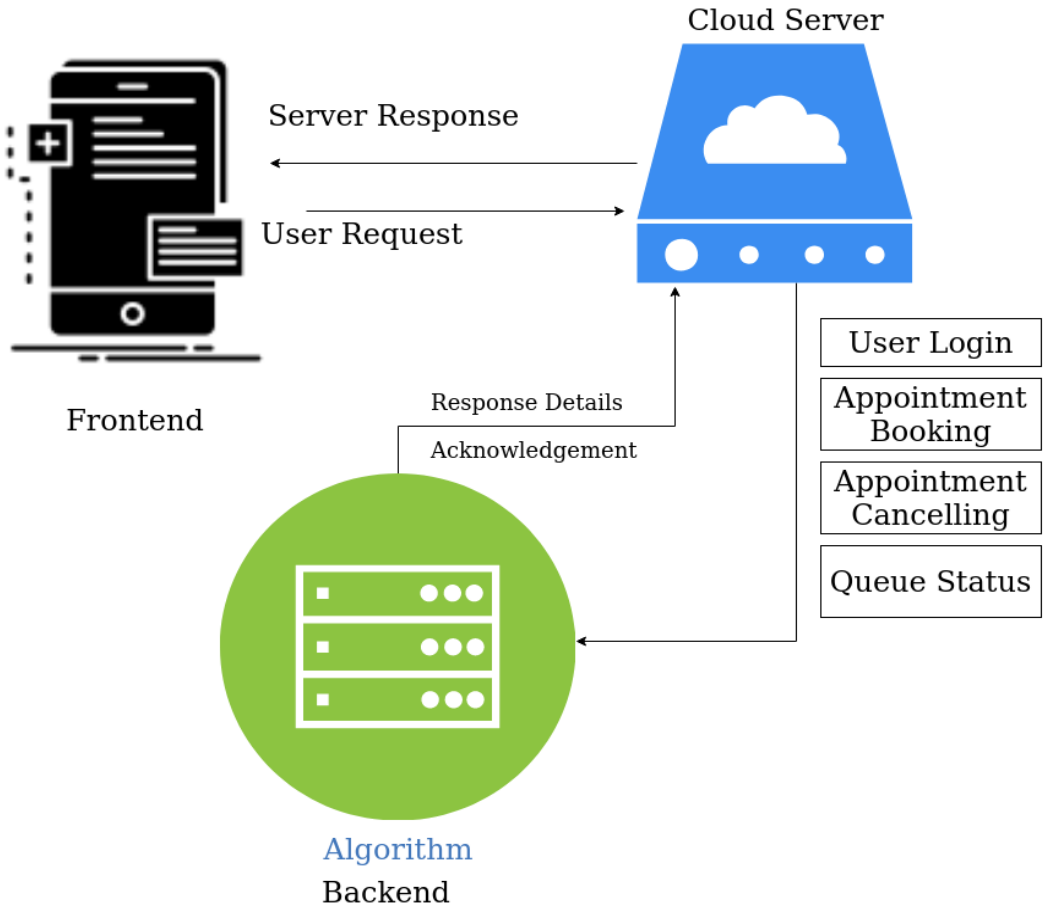


Mayank

Progress

System Architecture

Block Diagram

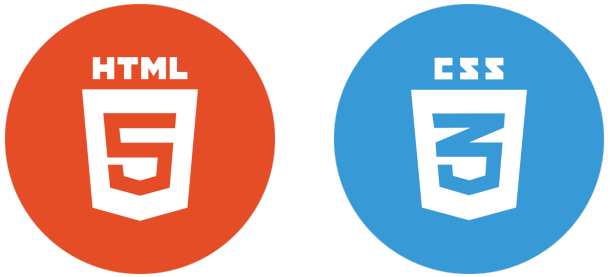


System Architecture

Components

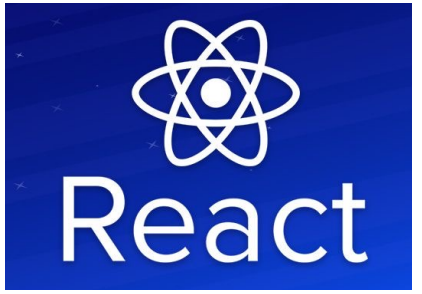
- Frontend
  - HTML/CSS
  - JS
  - ReactJS

For providing the basic structure and the styling



For Providing the functionality to HTML/CSS

Build on the top of the JS, to provide the production level environment along with rich features like



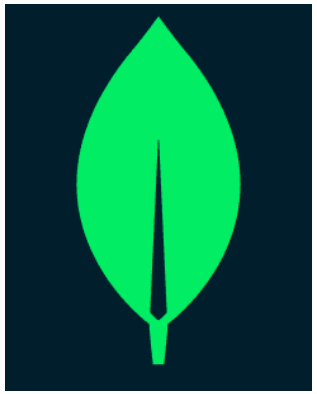
System Architecture

Components

- Backend
  - Node JS
  - Express JS
- Database
  - Mongo DB

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on a JavaScript Engine and executes JavaScript code outside a web browser, which was designed to build scalable network applications

MongoDB is a source-available cross-platform document-oriented database program. MongoDB uses JSON-like documents with optional schemas.



System Architecture

Components

- Cloud Server
  - Git Hub
  - Heroku
- Software
  - Vim Editor WebStorm by JetBrains

Heroku is a cloud platform as a service supporting several programming languages. One of the first cloud platforms,

GitHub, Inc., is an Internet hosting service for software development and version control using Git.
It provides the distributed version control of bug tracking, software feature requests, task management, continuous integration.

WebStorm is an integrated development environment for coding in JavaScript and its related technologies,

Algorithmic Implementation

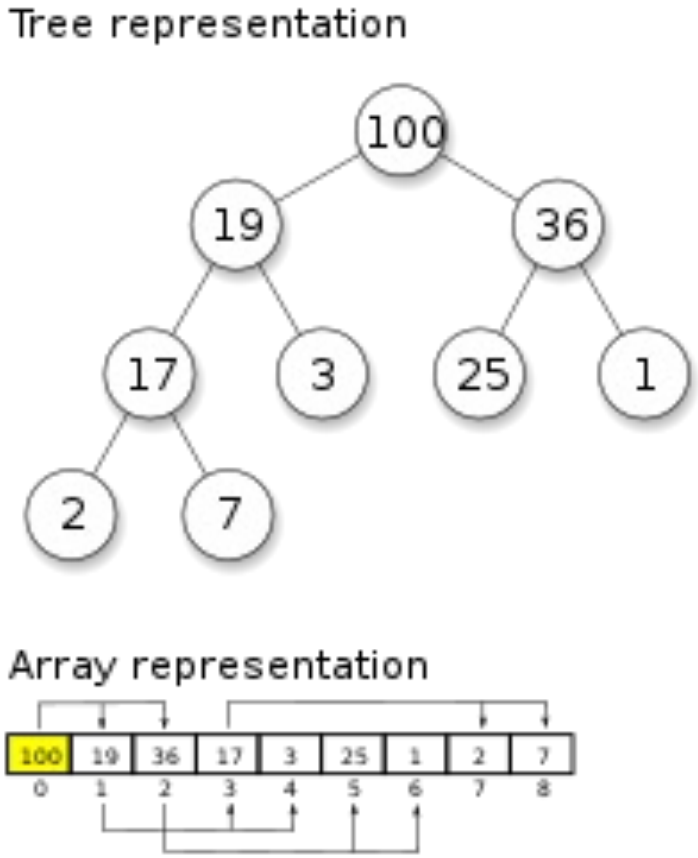
- Data structure
- Pseudo code

Heap **Data Structure**

- A data structure is a data organization, management, and storage format that is usually chosen for efficient access to data. [8]

- A heap is a tree-based data structure in which all the nodes of the tree are in a specific order. For example, if is the parent node of , then the value of follows a specific order with respect to the value of and the same order will be followed across the tree. [8]

- for building Heap and O(n log n)
  to remove each node O(log n)
  to add a node O(log n). [8]



Pseudo Code

```
MAX-HEAPIFY(A, i)
    // Input: A: an array where the left and right children of i root heaps (but i may not), i: an array index
    // Output: A modified so that i roots a heap
    // Running Time: O(log n) where n = heap-size[A] − i
 1  l ← LEFT(i)
 2  r ← RIGHT(i)
 3  if l ≤ heap-size[A] and A[l] > A[i]
 4        largest ← l
 5  else largest ← i
 6  if r ≤ heap-size[A] and A[r] < A[largest]
 7        largest ← r
 8  if largest ≠ i
 9        exchange A[i] and A[largest]
10        MAX-HEAPIFY(A, LARGEST)
```

```
BUILD-MAX-HEAP(A)
    // Input: A: an (unsorted) array
    // Output: A modified to represent a heap.
    // Running Time: O(n) where n = length[A]
1  heap-size[A] ← length[A]
2  for i ← ⌊length[A]/2⌋ downto 1
3        MAX-HEAPIFY(A, i)
```

Pseudo Code

HEAP-INCREASE-KEY(A, i, key)

```
// Input: A: an array representing a heap, i: an array index, key: a new key greater than A[i]
// Output: A still representing a heap where the key of A[i] was increased to key
// Running Time: O(log n) where n =heap-size[A]
1  if key < A[i]
2       error("New key must be larger than current key")
3  A[i] ← key
4  while i > 1 and A[PARENT(i)] < A[i]
5       exchange A[i] and A[PARENT(i)]
6       i ← PARENT(i)
```

HEAP-SORT(A)

```
// Input: A: an (unsorted) array
// Output: A modified to be sorted from smallest to largest
// Running Time: O(n log n) where n = length[A]
1  BUILD-MAX-HEAP(A)
2  for i = length[A] downto 2
3       exchange A[1] and A[i]
4       heap-size[A] ← heap-size[A] − 1
5       MAX-HEAPIFY(A, 1)
```

HEAP-EXTRACT-MAX(A)

```
// Input: A: an array representing a heap
// Output: The maximum element of A and A as a heap with this element removed
// Running Time: O(log n) where n =heap-size[A]
1  max ← A[1]
2  A[1] ← A[heap-size[A]]
3  heap-size[A] ←heap-size[A] − 1
4  MAX-HEAPIFY(A, 1)
5  return max
```

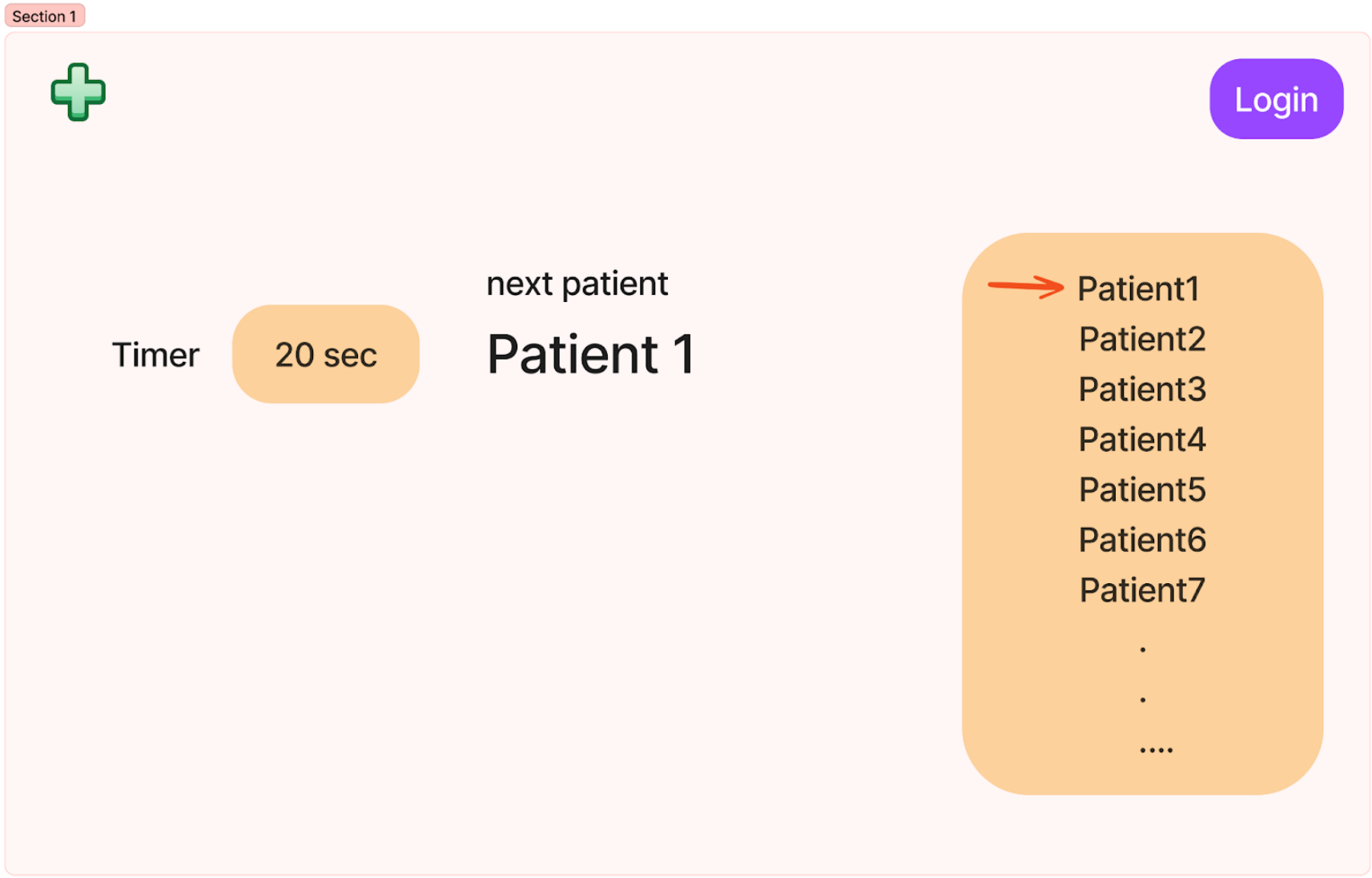
MAX-HEAP-INSERT(A, key)

```
// Input: A: an array representing a heap, key: a key to insert
// Output: A modified to include key
// Running Time: O(log n) where n =heap-size[A]
1  heap-size[A] ←heap-size[A] + 1
2  A[heap-size[A]] ← −∞
3  HEAP-INCREASE-KEY(A[heap-size[A]], key)
```

Frontend Design

Components

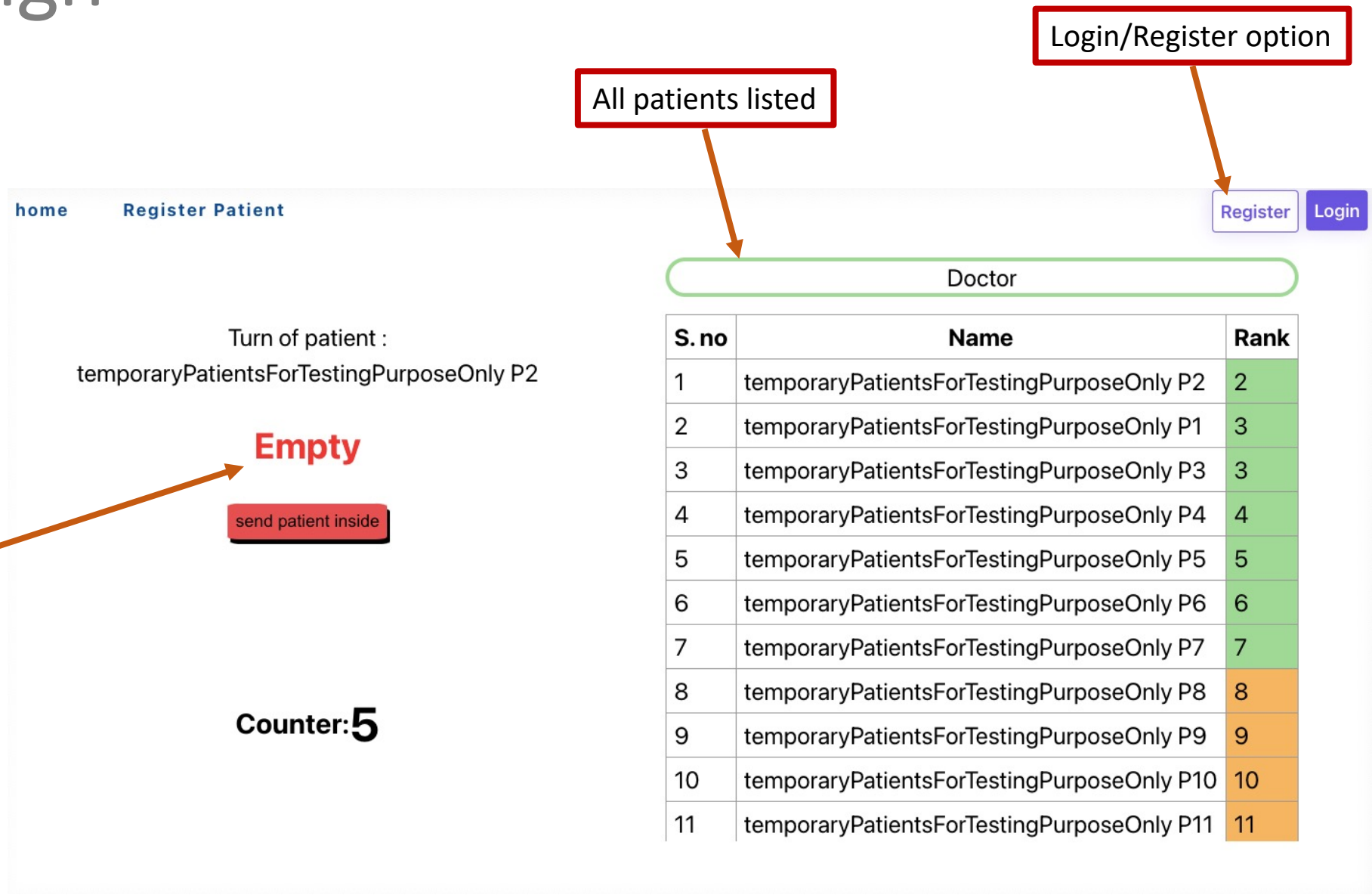
- Frontend



Frontend Design

Components

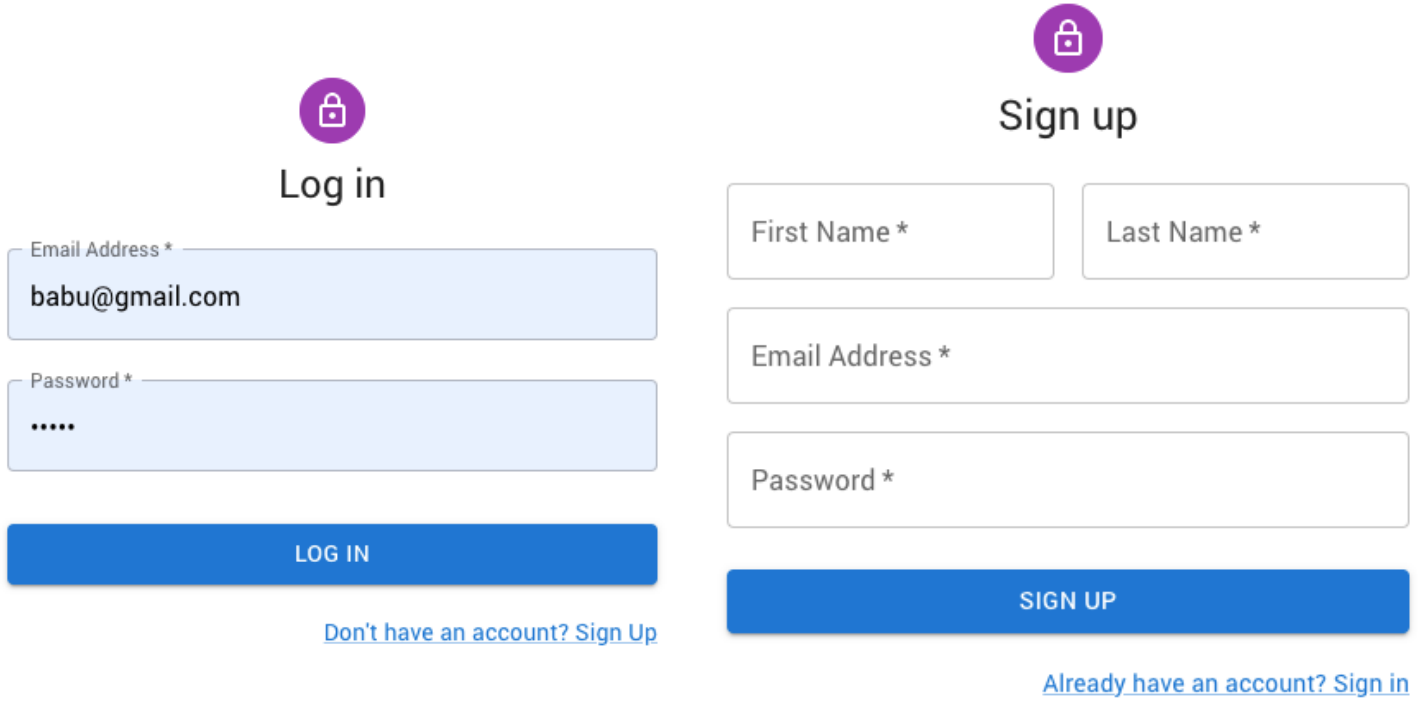
- Frontend



Frontend Design

Components

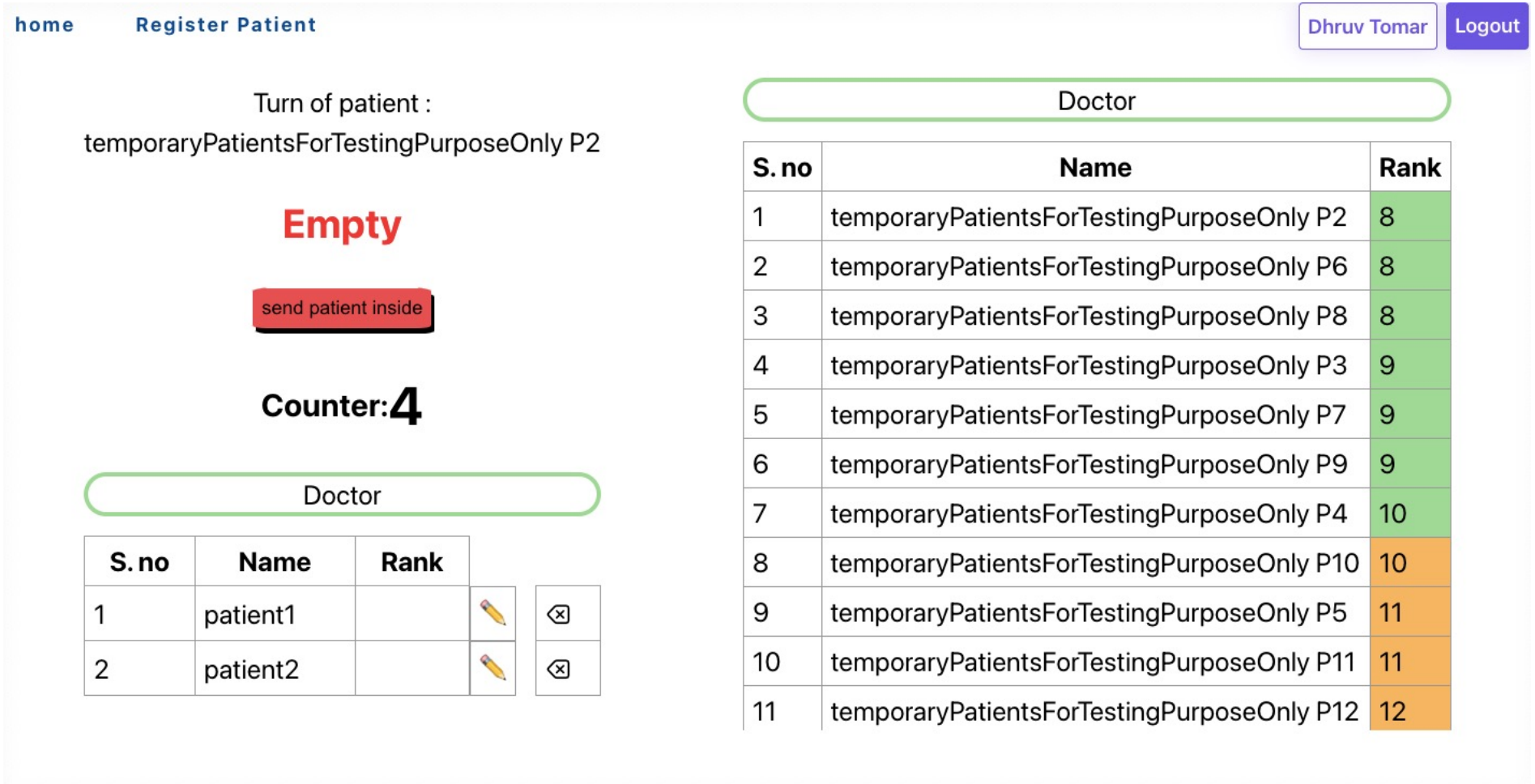
- Authentication



Frontend Design

Components

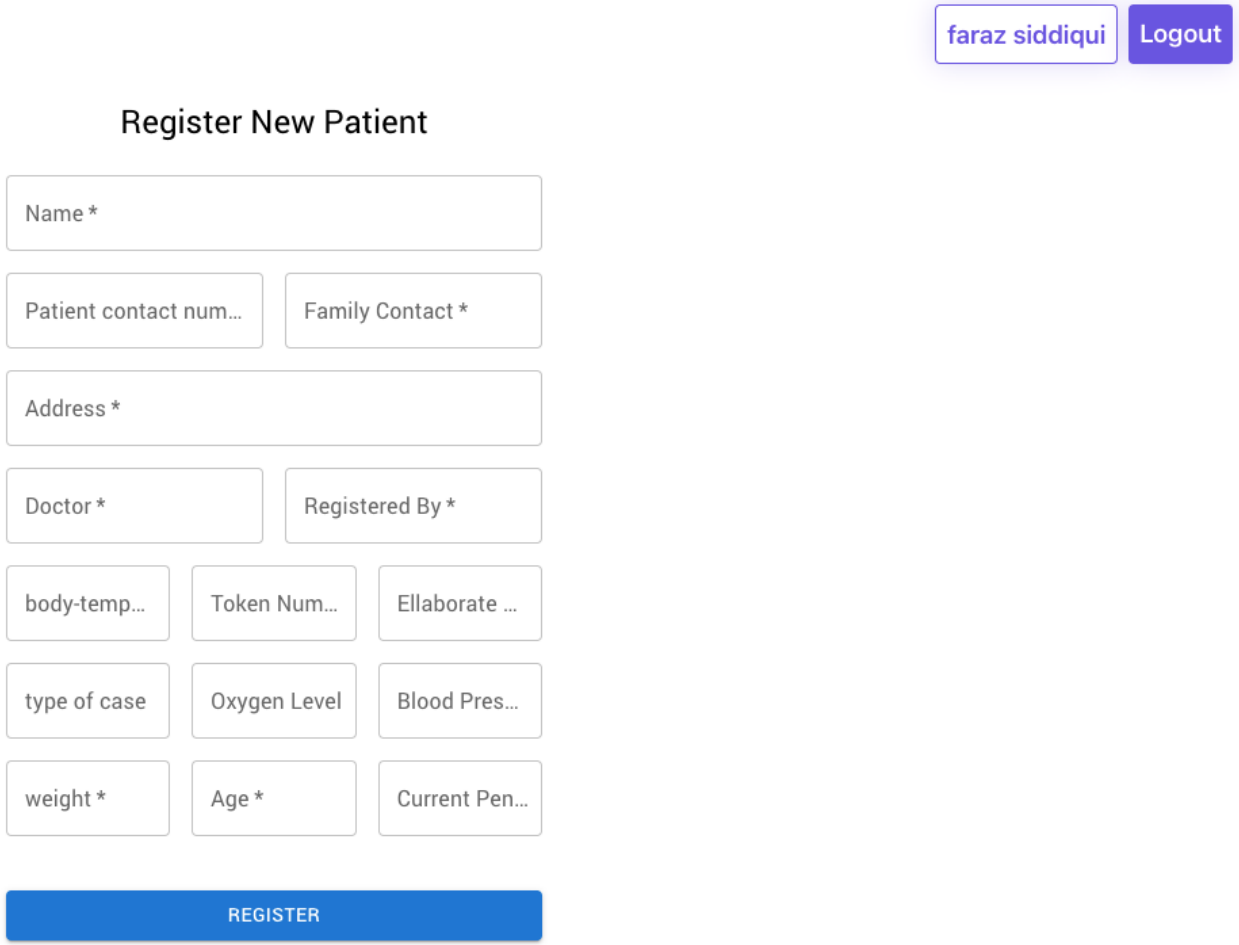
- Authenticated User



Frontend Design

Components

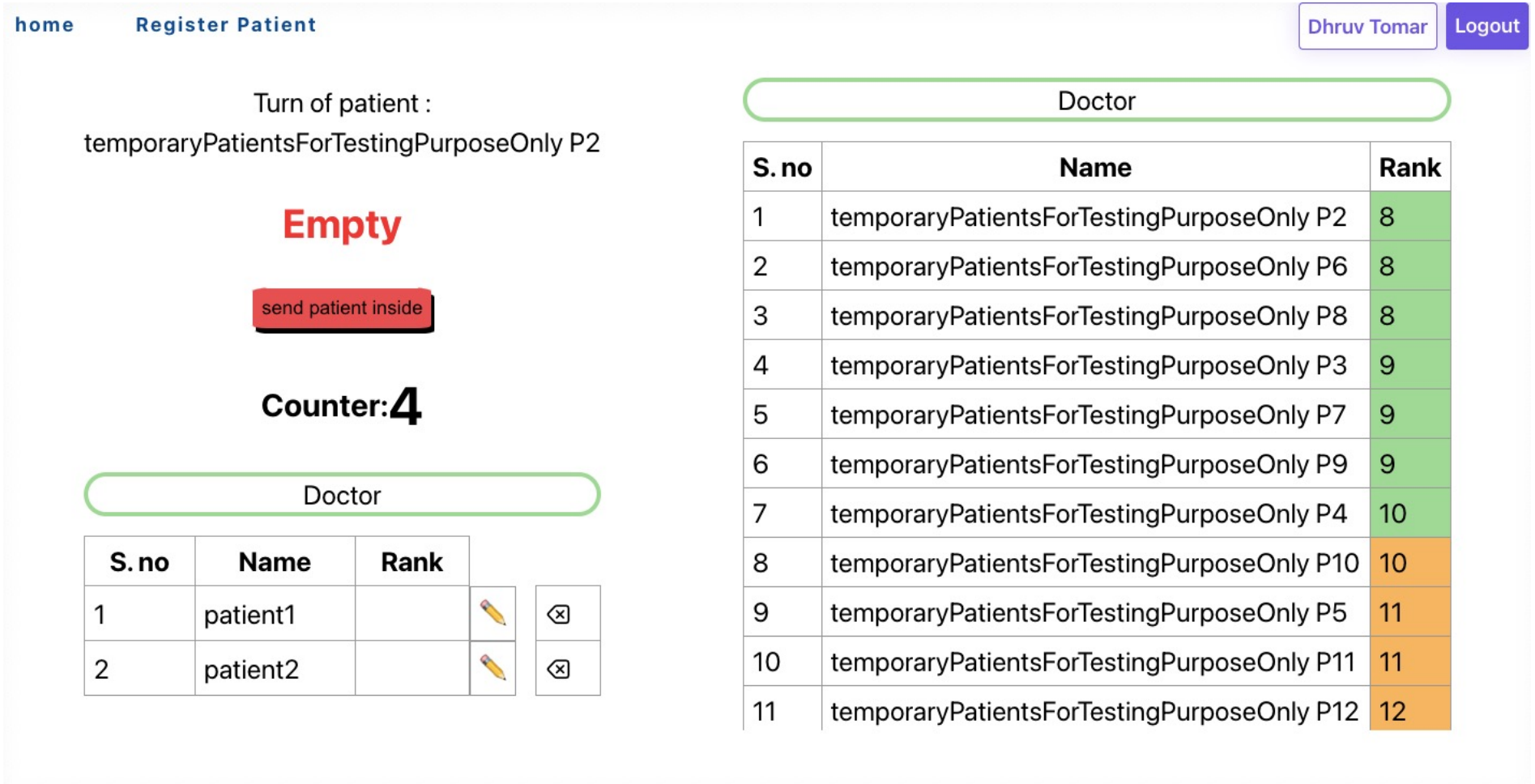
- Patient Registration



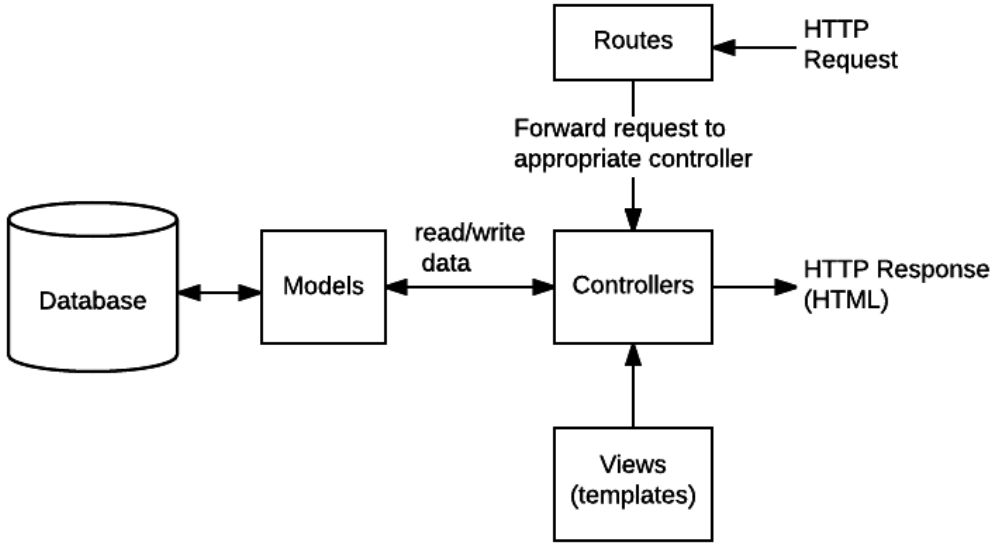
Frontend Design

Components

- Authenticated User

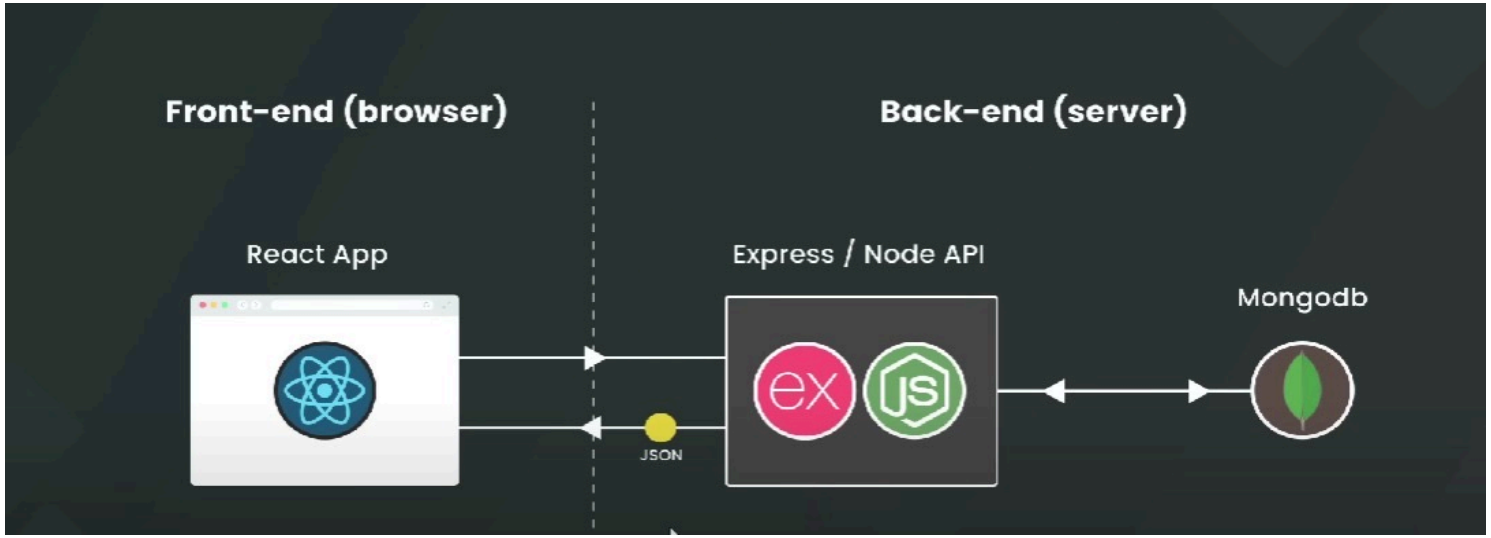


Backend Design



Components

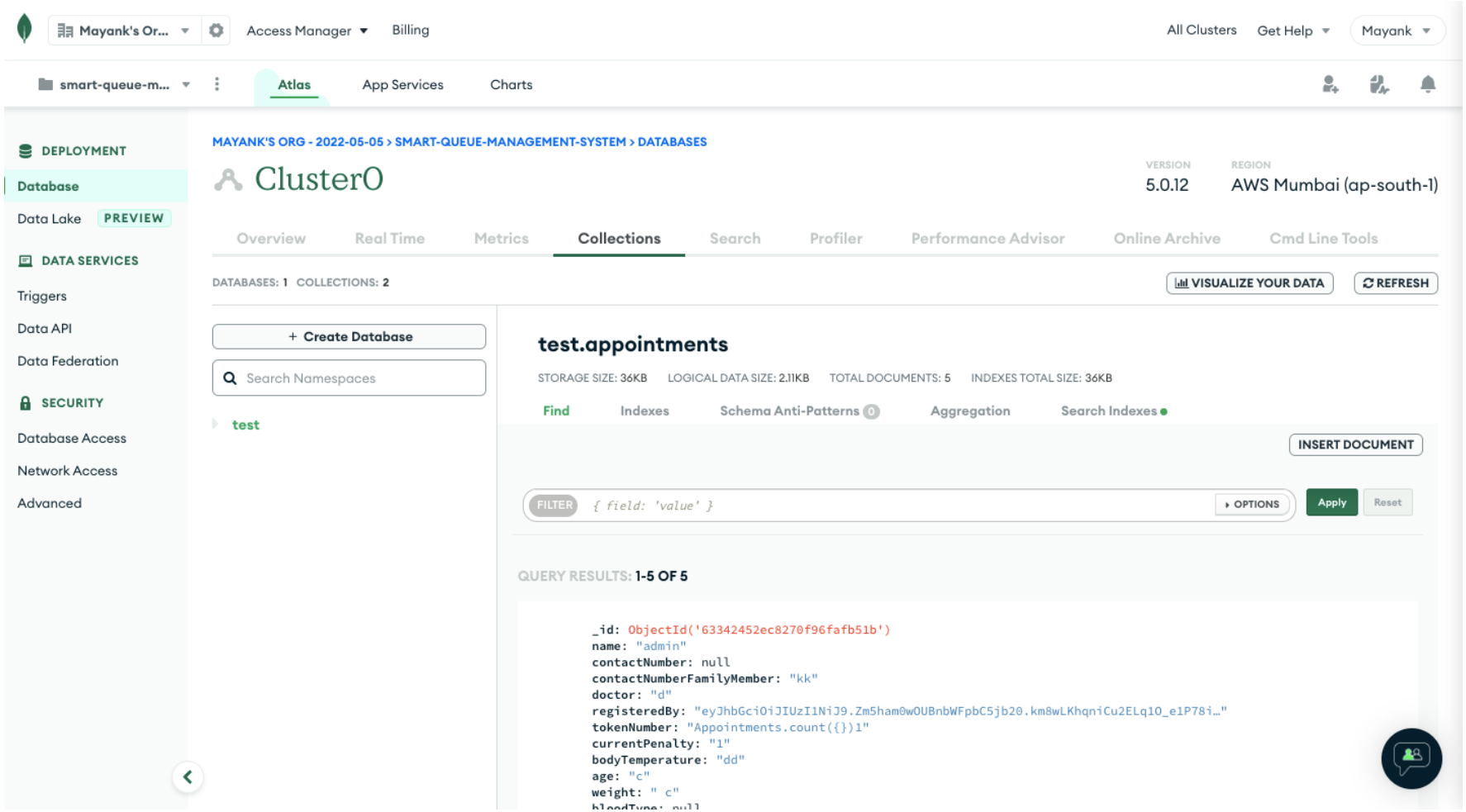
- Backend routes.
  - User Login/Signup Route
  - Patient Register/Deletion/Editing Route



Backend Design

Components

- Database
  - Mongo dB
  - 3 collections
  - Doctors, Patients, Registered User



Conclusion

Conclusion

- **Additional priority for senior citizens**

Conclusion

- Additional priority for senior citizens
- **Reduce waiting time**

Conclusion

- Additional priority for senior citizens
- Reduce waiting time
- **Reduce response time**

Conclusion

- Additional priority for senior citizens
- Reduce waiting time
- Reduce response time
- **Short queues**

Conclusion

- Additional priority for senior citizens
- Reduce waiting time
- Reduce response time
- Short queues
- **Proper organization for latecomers**

Conclusion

- Additional priority for senior citizens
- Reduce waiting time
- Reduce response time
- Short queues
- Proper organization for latecomers
- **Simplification of consultation process**

Conclusion

- Additional priority for senior citizens
- Reduce waiting time
- Reduce response time
- Short queues
- Proper organization for latecomers
- Simplification of consultation process
- **Digitalized transparent platform**

References

1. Sumit Soman; Sudeep Rai; Priyesh Ranjan; Amarjeet Singh Cheema; Praveen K Srivastava, International Journal **“Mobile-Augmented Smart Queue Management System for Hospitals”**, International Symposium on Computer-Based Medical Systems (CBMS), September 2020
2. R. Thirupathieswaran; Suria Prakash C.R.T; R. Santhana Krishnan; K. Lakshmi Narayanan; M. Ashok Kumar; Y. Harold Robinson**" Zero Queue Maintenance System using Smart Medi Care Application for Covid-19 Pandemic Situation"**, Third International Conference on Intelligent Communication Technologies and Virtual Mobile Network, IEEE Xplore Part Number: CFP21ONG-ART; 978-0-7381-1183-4 ICICV 2021
3. M. Ngorsed and P. Suesaowaluk, “**Hospital servic**Supriya Burungale, Komal Kurane, Sakshe Mhatre, “**Patient Queue Management System**”,International Journal of Engineering Science Invention, volume 7, issue 2, February 2018
4. Prof. D. V. Chandran, Divya Patil, Pooja Galande, Arati Ghutukade, “ **Multiple Queue Management With Real Time Tracking For OPD Scheduling In Hospitals**”, International Journal for Research in Engineering Application & Management, Volume 3, isuue 2, April 2017
5. Ngorsed, M., Suesaowaluk, P. (2016). **Hospital Service Queue Management System with Wireless Approach.** In: Hung, J., Yen, N., Li, KC. (eds) Frontier Computing. Lecture Notes in Electrical Engineering, vol 375. Springer, Singapore. https://doi.org/10.1007/978-981-10-0539-8_61

6. **"A study on the determinants of some Hessenberg Toeplitz Bohemians Jishe Feng 1 , Hongtao Fan School of Mathematics and Statistics"**, Longdong University, Qingyang, Gansu, 745000, PR China College of Science, Northwest A&University, Yangling, Shaanxi 712100, PR China School of Mathematics and Statistics, Lanzhou University, Lanzhou, Gansu 730000, PR China [link].
7. Gideon Dadik Bibu1 ; Gloria Chizoba Nwankwo2 1,2Department of Computer Science, University of Jos, Nigeria 1; "**COMPARATIVE ANALYSIS BETWEEN FIRST-COME-FIRST-SERVE (FCFS) AND SHORTEST-JOB-FIRST (SJF) SCHEDULING ALGORITHMS",** International Journal of Computer Science and Mobile Computing A Monthly Journal of Computer Science and Information Technology ISSN 2320–088X IMPACT FACTOR: 6.199 IJCSMC, Vol. 8, Issue. 5, May 2019, pg.176 – 181, dadikg@unijos.edu.ng, link
8. https://en.wikipedia.org/wiki/Heap_(data_structure

Any Questions ?

Roadmap to future
Major Project Overview

Roadmap to future Major Project Overview

- **IOT Integration**

Roadmap to future
Major Project Overview

- IOT Integration
- **Smart patients' presence recorder**

Roadmap to future Major Project Overview

- IOT Integration
- Smart patients' presence recorder
- **Smart Doctor Availability recorder**


Thank you