

MINI PROJECT REPORT ON
AUTOMATIC VACCINATION ALERT

Submitted in partial fulfillment of requirement for the award of the degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING



Submitted by

C.LAKSHMI PRASANNA

20BD1AO50G

Under the guidance of
Dr.M ANURADHA Assistant Professor
Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

(Approved by AICTE, Affiliated to JNTUH)
Narayanaguda, Hyderabad, Telangana-29
2023-2024



KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
(An Autonomous Institute)



**(Accredited by NBA & NAAC, Approved By A.I.C.T.E., Reg by
Govt of Telangana State & Affiliated to JNTU, Hyderabad)**

CERTIFICATE

This is to certify that mini project work entitled **AUTOMATIC VACCINATION ALERT** is a bonafide work carried out in the seventh semester by **C.LAKSHMI PRASANNA 20BD1A050G** in partial fulfillment for the award of Bachelor of Technology in **COMPUTER SCIENCE & ENGINEERING** from JNTU Hyderabad during the academic year 2023 - 2024 and no part of this work has been submitted earlier for the award of any degree.

INTERNAL EXAMINER

MINI PROJECT INCHARGE

HEAD OF THE DEPARTMENT

Vision of KMIT

Producing quality graduates trained in the latest technologies and related tools and striving to make India a world leader in software and hardware products and services. To achieve academic excellence by imparting indepth knowledge to the students, facilitating research activities and catering to the fast growing and ever- changing industrial demands and societal needs.

Mission of KMIT

- To provide a learning environment that inculcates problem solving skills, professional, ethical responsibilities, lifelong learning through multi modal platforms and prepare students to become successful professionals.
- To establish industry institute Interaction to make students ready for the industry.
- To provide exposure to students on latest hardware and software tools.
- To promote research based projects/activities in the emerging areas of technology convergence.
- To encourage and enable students to not merely seek jobs from the industry but also to create new enterprises.
- To induce a spirit of nationalism which will enable the student to develop, understand India's challenges and to encourage them to develop effective solutions
- To support the faculty to accelerate their learning curve to deliver excellent service to students.

Vision & Mission of CSE

Vision of the CSE

To be among the region's premier teaching and research Computer Science and Engineering departments producing globally competent and socially responsible graduates in the most conducive academic environment.

Mission of the CSE

- To provide faculty with state of the art facilities for continuous professional development and research, both in foundational aspects and of relevance to emerging computing trends.
- To impart skills that transform students to develop technical solutions for societal needs and inculcate entrepreneurial talents.
- To inculcate an ability in students to pursue the advancement of knowledge in various specializations of Computer Science and Engineering and make them industry-ready.
- To engage in collaborative research with academia and industry and generate adequate resources for research activities for seamless transfer of knowledge resulting in sponsored projects and consultancy.
- To cultivate responsibility through sharing of knowledge and innovative computing solutions that benefit the society-at-large.
- To collaborate with academia, industry and community to set high standards in academic excellence and in fulfilling societal responsibilities enterprises.
- To induce a spirit of nationalism which will enable the student to develop, understand India's challenges and to encourage them to develop effective solutions



KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY (An Autonomous Institute)



(Accredited by NBA & NAAC, Approved By A.I.C.T.E., Reg by Govt of
Telangana State & Affiliated to JNTU, Hyderabad)

Course Outcomes and CO-PO Mapping

Course Outcomes:

- CO1.** Proficiency in the MERN Stack for Web Application Development.
CO2. Mastery of Database Management for Web Applications.
CO3. Implementing Message Alert mechanism for notifying through Web Application.
CO4. Designing User-Centric Web Applications for Engineering Modules.
CO5. Creating Engineering Solutions for Industry and Societal Needs through Web Development.

CO-PO Matrix:

| | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
|------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|
| CO1 | 3 | 2 | 3 | | 2 | 2 | | | 3 | 2 | 1 | 2 |
| CO2 | 2 | 2 | | | 2 | 2 | | | 3 | 2 | 1 | 3 |
| CO3 | 2 | 1 | 2 | 2 | 1 | 2 | | | 3 | 3 | | 2 |
| CO4 | 2 | 3 | | 2 | 2 | 3 | | | 3 | 3 | | |
| CO5 | 2 | | 2 | 1 | | 3 | | | 3 | 2 | 2 | |
| Avg | 2.2 | 1.6 | 1.4 | 1 | 1.4 | 2.4 | | | 3 | 2.4 | 0.8 | 1.4 |

CO-PSO Matrix:

| | PSO1 | PSO2 |
|----------------|------|------|
| CO1 | 2 | 2 |
| CO2 | 2 | 2 |
| CO3 | 3 | 2 |
| CO4 | 2 | 1 |
| CO5 | 2 | 2 |
| Average | 2.2 | 1.8 |

Note: L -1 M – 2 H – 3

The average of Each CO with every PO 's and PSO's should lie between 2 and 3 values.

DECLARATION

We hereby declare that the project report entitled **"AUTOMATIC CHILD VACCINATION ALERT"** is done in the partial fulfillment for the award of the Degree in Bachelor of Technology in Computer Science and Engineering affiliated to Jawaharlal Nehru Technological University, Hyderabad. This project has not been submitted anywhere else.



C.LAKSHMI PRASANNA(20BD1A050G)

ACKNOWLEDGMENT

We take this opportunity to thank all the people who have rendered their full support to our project work. We render our thanks to **Dr. Maheshwar Dutta**, B.E., M Tech., Ph.D., Principal who encouraged us to do the Project. We are grateful to **Mr. Neil Gogte**, Director for facilitating all the amenities required for carrying out this project. We express our sincere gratitude to **Mr. S. Nitin**, Director and **Professor Dodle Jaya Prakash**, Dean Academics for providing an excellent environment in the college. We are also thankful to **Mr Para Upendar**, Head of the Department for providing us with both time and amenities to make this project a success within the given schedule. We are also thankful to our guide **Dr. M Anuradha**, for her valuable guidance and encouragement given to us throughout the project work. We would like to thank the entire CSE Department faculty, who helped us directly and indirectly in the completion of the project. We sincerely thank our friends and family for their constant motivation during the project work.

C.LAKSHMI PRASANNA (20BD1A050G)



KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY



(An Autonomous Institute)

(Accredited by NBA & NAAC, Approved By A.I.C.T.E.,
Reg by Govt of Telangana State & Affiliated to
JNTU, Hyderabad)

INDEX

| Table of Contents | Page No. |
|--------------------------|---|
| S.No. | Topic |
| | Abstract.....I |
| | List of Figures II |
| 1. | Introduction.....1 |
| 1.1 | Background and Motivation3 |
| 1.2 | Problem Statement8 |
| 1.3 | Objectives... ..10 |
| 1.4 | Scope and Limitations.....14 |
| 1.5 | Organization of the Document.....27 |
| 2. | System Requirement Specifications29 |
| 2.1 | What is SRS?.....29 |
| 2.2 | Role of SRS29 |
| 2.3 | Requirements Specification Document29 |
| 2.4 | Functional Requirement Specification.....30 |
| 2.5 | Performance Requirements.....30 |
| 2.6 | Non Functional Requirements.....31 |
| 2.7 | Hardware Requirements.....33 |
| 2.8 | Software Requirements.....34 |

| | |
|--|----|
| 3.Literature Survey | 35 |
| 3.1 Relevant Research Papers and Projects..... | 37 |
| 3.2 Theoretical Framework | 39 |
| 3.3 Related Technologies and Tools | 40 |
| 4.System Design..... | 42 |
| 5. Implementation... .. | 47 |
| 5.1 Description of Project Execution | 47 |
| 5.2 Code Structure and Architecture..... | 50 |
| 5.3 Technical Challenges and Solutions | 56 |
| 6. Testing..... | 58 |
| 7. Screenshots | 62 |
| 7.1 Approach Description | 68 |
| 7.2 Presentation Of Results..... | 70 |
| 8. Conclusion And Future Work | 74 |
| 8.1 Summary of Achievements. | 74 |
| 8.2 Contribution to the Field..... | 76 |
| 8.3 Future work and recommendations..... | 80 |
| 9.Reference..... | 81 |



KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY



(An Autonomous Institute)

**(Accredited by NBA & NAAC, Approved By A.I.C.T.E.,
Reg by Govt of Telangana State & Affiliated to JNTU
,Hyderabad)**

Abstract

This abstract presents a minor project in the domain of Web Development. Generally, students have a lot of knowledge in their specified field, but do not have any experience in the professional field. Mock interviews help in this aspect to help students get more experience for their interviews. But for a candidate to experience mock interviews, it involves reaching out to professionals over various platforms, waiting for their response and then finally deciding on a time, which is very tedious. The goal of this app is to provide a portal for student-professional interaction and streamline the interview process.

ProConnect enables the student and the professional to login using their respective details and navigate through the app seamlessly. The student can set their own profile, specifying their respective fields and can seek out interviewers based on their required fields and their preferred time. The interviewers can filter interview requests based on their availability, and select suitable candidates to connect with. The details of the interview will further be notified on their mails after a successful connection



KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY



(An Autonomous Institute)

(Accredited by NBA & NAAC, Approved
By A.I.C.T.E., Reg by Govt of
Telangana State & Affiliated to
JNTU, Hyderabad)

List of Figures

| Fig. No. | Figure Name | Page No. |
|-----------------|----------------------|-----------------|
| 4.1 | System Design | 43 |
| 4.2 | Uml Diagram | 46 |
| 5.1 | Front End view | 50 |
| 5.2 | Front End Components | 51 |
| 5.3 | Front End Images | 52 |
| 5.4 | Back End View | 54 |
| 7.1 | Login Page | 62 |
| 7.2 | Student Profile Page | 63 |
| 7.3 | Schedule Interview | 65 |
| 7.4 | Professional Profile | 65 |
| 7.5 | Match Requests | 65 |
| 7.6 | Scheduled Interviews | 65 |
| 7.7 | Scheduled Requests | 66 |
| 7.8 | Review Page | 66 |
| 7.9 | Report Page | 66 |
| 7.10 | Card for Interview | 68 |

| | | |
|------|--------------------------------------|----|
| 7.11 | Meeting link in mail for Student | 69 |
| 7.12 | Meeting link in mail for Interviewer | 70 |
| 7.13 | Feedback Mail | 70 |
| 7.14 | Feedback shown in card | 71 |

1.INTRODUCTION

Childhood vaccinations are a critical component of public health, helping prevent the spread of infectious diseases and safeguarding the health and well-being of our future generations. Timely vaccinations are essential, as they protect children from potentially life-threatening illnesses. To ensure that every child receives the necessary vaccinations on schedule, the "Automatic Child Vaccination Alert Management System" is introduced.

The Automatic Child Vaccination Alert Management System is an innovative solution designed to streamline and enhance the vaccination process for children. It leverages technology to automate and improve the management of child vaccinations, making it easier for parents, healthcare providers, and public health agencies to ensure that children receive their vaccinations on time.

Key Features:

1. **Automated Reminders:** One of the core features of this system is automated vaccination reminders. Parents and guardians will receive timely notifications via SMS, email, or mobile app alerts, reminding them of upcoming vaccinations, well-baby check-ups, and booster shots.
2. **Personalized Schedules:** The system tailors vaccination schedules based on the child's age and medical history. It accounts for variations in recommended vaccines, ensuring that each child receives a personalized immunization plan.
3. **Healthcare Provider Integration:** The system can be integrated with healthcare provider databases, enabling real-time updates on a child's vaccination history. This integration ensures that healthcare professionals can access accurate information, reducing the risk of redundant vaccinations or missed shots.
4. **Information Access:** Parents and guardians can access educational materials, vaccine information, and resources related to child immunizations through the system. This promotes awareness and helps address concerns and questions related to vaccinations.

5. Reporting and Monitoring: Public health agencies and healthcare providers can use the system to track vaccination coverage rates and identify areas where additional outreach or resources are needed to ensure community immunity.

Benefits:

- Improved Vaccination Coverage: By sending timely reminders and making information easily accessible, the system helps increase vaccination coverage rates, reducing the risk of outbreaks.
- Reduced Administrative Burden: Healthcare providers can streamline their record-keeping and appointment scheduling processes, allowing them to focus more on patient care.
- Enhanced Parental Engagement: Parents and guardians receive support in managing their child's immunization schedule, leading to better adherence to vaccination guidelines.
- Data-Driven Decision-Making: Public health agencies can utilize the data generated by the system to make informed decisions regarding vaccination programs and resource allocation.
- Prevention of Vaccine-Preventable Diseases: Ultimately, the Automatic Child Vaccination Alert Management System aims to contribute to the elimination of vaccine-preventable diseases, protecting the health and future of our children.

In conclusion, the Automatic Child Vaccination Alert Management System is a technological innovation that has the potential to transform the way we manage and monitor child vaccinations. By leveraging automation, personalization, and data-driven insights, the system ensures that children receive their vaccinations on time, contributing to the overall improvement of public health and the well-being of our youngest citizens.

1.1 Background and Motivation

Childhood vaccination is one of the most effective public health interventions for preventing the spread of infectious diseases and protecting the health of children. Vaccination programs have significantly reduced the incidence of serious illnesses, disabilities, and deaths caused by vaccine-preventable diseases like measles, polio, and whooping cough. However, despite the proven benefits of vaccination, many children around the world still miss out on timely vaccinations due to various factors, including forgetfulness, lack of awareness, or logistical challenges.

Relevance:

The automated child vaccination alert system is highly relevant due to the following reasons:

1. **Public Health Impact:** Timely and complete vaccination is essential to maintain herd immunity, prevent disease outbreaks, and reduce the burden on healthcare systems. Automated alerts can help increase vaccination rates, ultimately contributing to public health.
2. **Parental Forgetfulness:** Parents and guardians often struggle to keep track of their child's vaccination schedule, leading to missed vaccinations. Automated alerts provide a convenient solution to remind them when vaccines are due.
3. **Improved Health Outcomes:** A well-implemented system can significantly reduce the occurrence of vaccine-preventable diseases and their associated complications, leading to better health outcomes for children.

4. **Reduced Healthcare Costs:** Fewer cases of vaccine-preventable diseases result in cost savings for healthcare systems and families who would otherwise bear the expenses of treating sick children.

5. **Data Management:** Automated systems can efficiently manage and maintain vaccination records, making it easier for healthcare providers to access and update patient information.

6. **Accessibility:** The system can reach a wide range of parents, including those in remote areas or underserved communities, provided they have access to basic technology.

Real-World Problems Addressed:

1. **Vaccination Compliance:** A significant problem is low vaccination compliance, where children miss or delay their immunizations, leaving them vulnerable to preventable diseases.

2. **Data Management:** Healthcare providers often struggle with maintaining accurate and up-to-date vaccination records, which can lead to missed opportunities for vaccination.

3. **Communication Gaps:** In busy lives, parents can overlook vaccination schedules, and healthcare providers may have limited means to effectively remind them.

4. **Health Disparities:** There are disparities in vaccination rates among different demographic groups, including those with limited access to healthcare or information.

5. **Emergency Preparedness:** Rapid response to disease outbreaks or changes in vaccination recommendations can be challenging without efficient communication channels.

6. **Data Privacy and Security:** Safeguarding vaccination and personal health data is crucial to ensure patient privacy and maintain public trust in vaccination programs.

An automated child vaccination alert system addresses these real-world problems by leveraging technology to improve vaccination rates, streamline data management, enhance communication, and ensure that children receive essential vaccinations on time, thereby reducing the burden of vaccine-preventable diseases on public health systems and families.

1.2 Problem Statement

Nearly 2 Million children under the age of 5 years die every year in India .The Indian Academy of Pediatrics (IAP) estimates that over 50 percent of these are vaccine preventable. In order to prevent the infants from hazardous diseases such as small pox , hepatitis B , tetanus ,etc. we've taken this initiative .

This website is easy and convenient to use and it maintains the data of users and the main motto of our vaccination model is to provide convenience to the parents. The desired impact of this app is to ensure that the children receive all the mandatory vaccination on time.

At the time of child birth the hospital uploads the child details in this website. The vaccination alerts will be sent to the parents from three days prior to the vaccination date when the hospital clicks on a button in this website.

Vaccination model is specially designed for maintaining the health of babies. So, parents need not to worry about remembering the dates of their child's vaccines, they automatically receive reminder notifications.

The VACCINATION ALERT app provides 3 reminder alerts for the caregiver in a week that the vaccinations are due. The app requires data of both the child and the parent or caregiver. It's the responsibility of the caregiver to provide the correct information at the hospital. The website uses the information provided to schedule the reminder alert.

1.3 Objectives

1. **Improve Child Immunization Rates:** Ensure that a higher percentage of children receive timely vaccinations by sending automated alerts to parents and guardians.
2. **Timely Reminders:** Provide timely and personalized vaccination reminders to parents and guardians, helping them stay on schedule with their child's immunization needs.
3. **Reduction of Vaccine-Preventable Diseases:** Contribute to a reduction in vaccine-preventable diseases by increasing the number of children who are fully vaccinated.
4. **Efficient Data Management:** Implement a system for efficiently storing and managing vaccination records, ensuring easy access for healthcare providers and parents.
5. **Customized Alert Preferences:** Allow parents and guardians to set their preferred communication channels (e.g., SMS, email) and frequencies for vaccination reminders, making it more convenient for them.
6. **Accessibility and Inclusivity:** Ensure that the system is accessible to a wide range of users, including those with different language preferences and varying levels of technology literacy.
7. **Compliance with Health Regulations:** Ensure that the system adheres to local and national health regulations regarding vaccination schedules and data privacy.

8. Integration with Healthcare Providers: Facilitate communication and coordination between parents, healthcare providers, and public health agencies, streamlining the vaccination process.

9. Data Security and Privacy: Implement robust security measures to protect sensitive vaccination records and personal information of children and their families.

10. User Education: Provide educational resources and information about the importance of vaccination, addressing common concerns and misconceptions.

11. Monitoring and Reporting: Enable tracking and reporting features to monitor vaccination rates, compliance, and system performance, helping public health agencies make data-driven decisions.

12. Scalability and Sustainability: Design the system to accommodate an increasing number of users and changing vaccination requirements, ensuring its long-term sustainability.

13. Cost-Efficiency: Optimize the system to be cost-effective in terms of development, maintenance, and operational expenses.

14. User Satisfaction: Collect feedback from users to continuously improve the system and enhance user satisfaction.

15. **Emergency Notifications:** Include a feature to send emergency alerts and updates related to vaccines, such as disease outbreaks or changes in vaccination schedules.

16. **Cultural Sensitivity:** Ensure that the system respects and takes into account cultural and religious differences that may influence vaccination decisions.

17. **Community Engagement:** Encourage community involvement and support for vaccination programs, promoting a collective responsibility for child health.

18. **User Support and Assistance:** Provide customer support and assistance to address user inquiries, troubleshoot technical issues, and guide users through the system.

19. **Research and Analysis:** Enable the system to collect data for research and analysis of vaccination trends, contributing to public health research and policy decisions.

20. **Public Awareness:** Support public awareness campaigns about the benefits of vaccination and the role of the system in promoting child health.

1.4 Scope and Limitations

SCOPE

The scope of the automated child vaccination alert minor project encompasses the development and implementation of a user-friendly software solution designed to send automated vaccination reminders to parents and guardians. This system will allow hospitals to input the contact information, receiving timely alerts via the parents preferred communication channels (e.g., SMS, email). The project includes the design of a user interface for data input, a database for storing vaccination records, and a notification system. The scope also covers the integration of relevant healthcare guidelines and regulations, ensuring data security, privacy, and accessibility. While the project aims to improve vaccination rates and reduce vaccine-preventable diseases, it focuses on a limited geographic area and a specific user base for the purposes of this minor project.

LIMITATIONS

Certainly, here are some potential limitations of an automated child vaccination alert system for a minor project, presented as bullet points:

1. **Limited Outreach:** The system's effectiveness heavily relies on the availability of contact information for parents or guardians, which may not cover all children in the target population.
2. **Data Accuracy:** The accuracy of vaccination records is contingent upon the input of correct information by parents or healthcare providers, and errors or omissions may occur.

3. **Dependency on Technology:** The system assumes that users have access to and are comfortable using technology (e.g., smartphones, internet) to receive alerts, potentially excluding those without such access.

4. **Language and Literacy Barriers:** Parents who speak languages other than the supported languages or have limited literacy may not fully benefit from the system.

5. **Privacy Concerns:** Some parents may be apprehensive about sharing personal and vaccination data through the system, raising privacy concerns that could limit its adoption.

6. **User Engagement:** Not all parents or guardians may consistently engage with the alerts or prioritize vaccinations, which can affect vaccination compliance.

7. **System Reliability:** Technical issues, such as server outages or software glitches, can disrupt the system's ability to send timely alerts.

8. **Cost Constraints:** Developing, hosting, and maintaining the system could be cost-prohibitive for a minor project, potentially limiting its scalability and sustainability.

9. **Geographical Reach:** The system may not be effective in reaching children in remote or underserved areas where access to healthcare and technology is limited.

10. Parental Compliance: The system cannot guarantee that parents will follow through with vaccination recommendations after receiving alerts.

11. Emergency Situations: While the system can send routine reminders, it may not be equipped to handle emergency situations, such as sudden disease outbreaks or rapid vaccination schedule changes.

12. Maintenance and Updates: Regular maintenance and updates are necessary to keep the system functional and up-to-date with changing vaccination recommendations and technologies.

1.5 Organization of the Document

1. Introduction

- Background and Motivation
- Problem Statement
- Objectives
- Scope and Limitations

2. Literature Review

- Review of Relevant Research Papers and Projects
- Theoretical Framework
- Related Technologies and Tools

3. Methodology

- Description of the Approach/Method Used
- Software/Hardware

4. Implementation

- Detailed Explanation of How the Project Was Executed
- Code Structure and Architecture
- Technical Challenges and Solutions

5. Results

- Presentation of Results (Tables, Charts, Graphs)
- Analysis and Discussion of Results
- Comparison with Expected Outcomes

6. Conclusion

- Summary of Achievements

- Contributions to the Field
- Future Work and Recommendations

7. References

- List of All Cited Sources (Books, Journals, Websites, etc.)
- Follow a Citation Style (e.g., APA, IEEE)

8. Appendices

- Additional Technical Details
- Code Snippets

2.SPECIFIC REQUIREMENT SPECIFICATION

2.1 What is SRS?

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.)

The SRS phase consists of two basic activities:

Problem/Requirement Analysis:

The process is order and more nebulous of the two, deals with understand the problem, the goal and constraints.

Requirement Specification:

Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity. The Requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

2.2 Role of SRS

The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and the developers. Software Requirement Specification is the medium through which the client and user

needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties involved in the system.

2.3 Requirements Specification Document

A Software Requirements Specification (SRS) is a document that describes the nature of a project, software or application. In simple words, SRS document is a manual of a project provided it is prepared before you kick-start a project/application. This document is also known by the names SRS report, software document. A software document is primarily prepared for a project, software or any kind of application.

There are a set of guidelines to be followed while preparing the software requirement specification document. This includes the purpose, scope, functional and non functional requirements, software and hardware requirements of the project. In addition to this, it also contains the information about environmental conditions required, safety and security requirements, software quality attributes of the project etc.

The purpose of SRS (Software Requirement Specification) document is to describe the external behavior of the application developed or software. It defines the operations, performance and interfaces and quality assurance requirement of the application or software. The complete software requirements for the system are captured by the SRS. This section introduces the requirement specification document for Word Building Game using Alexa which enlists functional as well as non-functional requirements.

2.4 Functional Requirement Specification

The System after careful analysis has been identified to be present with the following modules.

A functional requirement defines a function of a system or its components. Functional requirements may be calculations , technical details, data manipulation and processing and other specific functionality that defines what a system is supposed to accomplish the functional requirement specification documents the operation and activities that a system able to perform. Functional requirements include functions performed by specific screens, outlines of work flows performed by the system, and other business compliance requirements the system must meet. This project has four modules.

2.5 Performance requirements

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system

Requirements about resources required, response time, transaction rates, throughput, benchmark specifications or anything else having to do with performance. In this project, Data publisher (or data holder, who collects data from

record owner ex. Alice and bob) and data miner or the public, called the data recipient and record owners like patients and doctors.

Modifiability

Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person- months).

Portability

The effort required to move the software to a different target platform. The measurement is most commonly person-months or % of modules that need changing.

Response time- Response time will be minimum. System should response within 0.5 seconds atmost.

Capacity-The system must support 500 people at a time.

2.6 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define the overall qualities or attributes of the resulting System Non-functional requirements place restrictions on the product being developed, the development process, and specify external constraints that the product must meet. Examples of NFR include safety, security, usability, reliability and performance Requirements. Project management issues(costs, time, and schedule) are often considered as non-functional requirements.

Reliability

Requirements about how often the software fails. The measurement is often expressed in MTBF (mean time between failures). The definition of a failure must be clear. Also, don't confuse reliability with availability which is quite a different kind of requirement. Be sure to specify the consequences of software failure, how

to protect from failure, a strategy for error detection, and a strategy for correction.

Security

One or more requirements about protection of your system and its data. The measurement can be expressed in a variety of ways (effort, skill level, time) to break into the system. Do not discuss solutions (e.g. passwords) in a requirements document.

Usability

Requirements about how difficult it will be to learn and operate the system. The requirements are often expressed in learning time or similar metrics.

Legal

There may be legal issues involving privacy of information, intellectual property rights, export of restricted technologies, etc.

2.7 HARDWARE REQUIREMENTS

Computers with 1 GB RAM is required. A Computer with an internet connection and any internet browser is required for the client to run the web application.

2.8 SOFTWARE REQUIREMENTS

Interpreter: PyCharm: It features a lightning-fast source code editor, perfect for day to day use with support of hundreds of languages. It also helps to be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box section, snippets etc. User can run the application either on LINUX or Windows with an internet connection and any internet browser.

Acceptance criteria: Before accepting, the developer must check whether the application is running properly or not and should also check whether the data is correctly sorted or not.

2.9 TECHNOLOGIES USED

2.9.1 HTML

Hypertext Markup Language(HTML), the languages of the world wide web(WWW), allows users to produces web pages that included text, graphics and pointer to other webpages .HTML is not a programming language but it is an application of ISO Standard8879,SGML(Standard Generalized Markup Language),but specialized to hypertext and adapted to the Web. We can navigate through the information based on out interest and preference. A markup language is simply a series of items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document. Html can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop. HTML provides tags(special codes) to make the document look attractive. HTML provides are not case-sensitive. Using graphics,fonts,different sizes, color, etc.. can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

Basic Html Tags:

<! >Specific Comments.

 Text is changed to bold

<Body> </Body> Contains all tags and text in the Html-document

<DD>.....</DD> Definition of a term.

</TABLE>.....</TABLE>Creates table

<Td>.....</Td> Indicates table data in a table.

<Tr>..... </Tr>Designates a table row

<Th> </Th>Creates a heading in a table.

Advantages:

✓ A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information.

✓ HTML is platform independent.

✓ HTML tags are not case-sensitive.

2.9.2 JAVA SCRIPT

JavaScript is a compact, object-based scripting language for developing client and server internet applications. Netscape Navigator 2.0 interprets JavaScript statements embedded directly in an HTML page, and Livewire enables you to create server-based applications similar to common gateway interface(cgi) programs. In a client application for Navigator, JavaScript statements embedded in an HTML page can recognize and respond to user events such as mouse clicks form Input, and page navigation. For example, you can write a JavaScript function to verify that users enter valid information into a form requesting a telephone number or zip code. Without any network transmission, an Html page with embedded Java Script can interpret the entered text and alert the user with a message dialog if the input is invalid or you can use JavaScript to perform an action (such as play an audio file, execute an applet, or communicate with a plug-in).

2.9.3 CSS

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

2.9.4 Framework

React is an open-source JavaScript library primarily used for building user interfaces in web applications. Created and maintained by Facebook, React has gained widespread popularity for its efficiency, reusability, and component-based architecture.

At its core, React allows developers to create UI components that update efficiently and predictably in response to changes in application state, making it ideal for building complex and dynamic user interfaces. React employs a virtual DOM (Document Object Model) that optimizes the rendering process by reducing direct interaction with the actual DOM, resulting in faster and more efficient updates.

React's key concepts include components, which are self-contained and reusable building blocks for user interfaces, and a unidirectional data flow, which ensures that changes in data are reflected consistently in the UI. JSX (JavaScript XML) is a syntax extension used in React to define component structures, making it easy to write and understand UI code.

React can be paired with other technologies, such as Redux for state management or React Router for handling routing in single-page applications. Its strong community and ecosystem of libraries, along with its performance and developer-friendly features, have made React a go-to choice for front-end development, powering many popular websites and web applications.

Back-end:

MongoDB is a popular, open-source NoSQL database management system designed to handle unstructured, semi-structured, or structured data. Unlike traditional relational databases, MongoDB stores data in a flexible, JSON-like format called BSON (Binary JSON), making it well-suited for handling dynamic, rapidly evolving data.

Key features of MongoDB include:

1. **Document-Oriented:** MongoDB stores data in collections of documents, each of which is a JSON-like object. This design allows for easy storage of complex data structures and hierarchical relationships.

2. Scalability: MongoDB is horizontally scalable, meaning it can distribute data across multiple servers or nodes, making it suitable for handling large datasets and high-traffic applications.
3. Flexibility: The schema-less nature of MongoDB allows for easy modification and expansion of data without the need to adhere to a fixed schema.
4. Querying: MongoDB provides powerful querying capabilities, supporting a wide range of queries and indexing for efficient data retrieval.
5. High Availability: It offers features like replica sets for data redundancy and automatic failover, ensuring data availability and reliability.
6. Geospatial Capabilities: MongoDB includes geospatial indexing and querying, making it useful for location-based applications.
7. Aggregation Framework: MongoDB's aggregation framework allows for complex data transformations and analytics operations.

MongoDB is widely used in various applications, from content management systems and e-commerce platforms to real-time analytics and mobile applications. Its flexible and scalable nature, along with an active open-source community and comprehensive documentation, make it a favored choice for modern, data-intensive projects.

Safety Requirements

If there is any damage to a wide portion of the database due to any kind of failure, such as a disk crash, the recovery method restores a past copy of the database, So that users data will not lose. Also nobody can change system's internal records except the system administrator.

Security Requirements

1. Want take the responsibility of failures due to hardware malfunctioning.
2. Warranty period of maintaining the software would be one year.
3. Additional payments will be analyzed and charged for further maintenance.
4. If any error occur due to a user's improper use. Warranty will not be allocated to it.
5. No money back returns for the software.
6. User's data must be secure.

Software system attributes

Correctness: Software should be correct in all aspects, also it should meet all the caretaker's requirements.

Completeness: Software system should be complete.

Availability: The system shall be available all the time. Means caretaker can access the software anytime.

Usability: Software can be used again and again without any distortion.

Accessibility: Administrator and many other users can access the system but the access level, vary from user to user means admin, caretaker and doctor, is controlled for each user according to their work scope.

Accuracy: The system should be accurate and reliable.

Stability: System should be stable.

The system output won't change time to time. Same output will be given always for a given input.

Maintainability and Modifiability: It's structure and style are such that if there is need to any change, that changes should easily made. System should have ability to maintain, modify information and update fix problems of the system.

3. Literature Review

● Review of Relevant Research Papers and Projects

- ❖ <https://machinelearningmastery.com/>
- ❖ <https://iq.opengenus.org/>
- ❖ https://en.wikipedia.org/wiki/Generative_adversarial_network
- ❖ <https://nevonprojects.com/child-vaccination-management-system-using-python/>

● Theoretical Framework

Theoretical Framework for Automated Child Vaccination Alerts Using React:

1. User-Centered Design Principles:

- Usability: The system should adhere to usability principles, ensuring that it's easy to navigate and understand for users with varying levels of technical proficiency.
- User Feedback: Incorporate user feedback and iterative design processes to continually enhance the UI.

2. Health Behavior Theory:

- Health Belief Model: Consider integrating aspects of the Health Belief Model to understand how parents perceive the severity and susceptibility of vaccine-preventable diseases and design the UI to address these perceptions.

- Theory of Planned Behavior: Explore how parents' attitudes, subjective norms, and perceived behavioral control affect their intention to follow vaccination recommendations, and design the UI to encourage the desired behaviors.

3. Information Design and Visualization:

- Information Hierarchy: Implement a clear information hierarchy in the UI, prioritizing essential information such as upcoming vaccination dates, vaccination schedules, and contact details.

4. Personalization and Tailoring:

- User Profiles: Develop user profiles that allow parents or guardians to customize notification preferences, such as the choice of communication channels (SMS, email).

- Tailored Messaging: Implement personalized messages and reminders based on the child's vaccination history and specific needs.

5. Mobile-First Design:

- Responsive Design: Ensure that the UI is designed with a mobile-first approach to accommodate users accessing the system on various devices, such as smartphones and tablets.

6. Data Privacy and Security:

- Data Protection Framework: Adhere to data privacy regulations and establish strong security measures to protect sensitive user information and vaccination records.

7. Feedback Loop and Iteration:

- **User Feedback Mechanism:** Create a mechanism for users to provide feedback on the UI and their experience with the system, facilitating continuous improvement.

8. Health Communication Strategies:

- **Effective Messaging:** Employ best practices in health communication to ensure that the UI conveys the importance of vaccination and addresses common concerns and misconceptions.

9. Integration with Backend and Data Management:

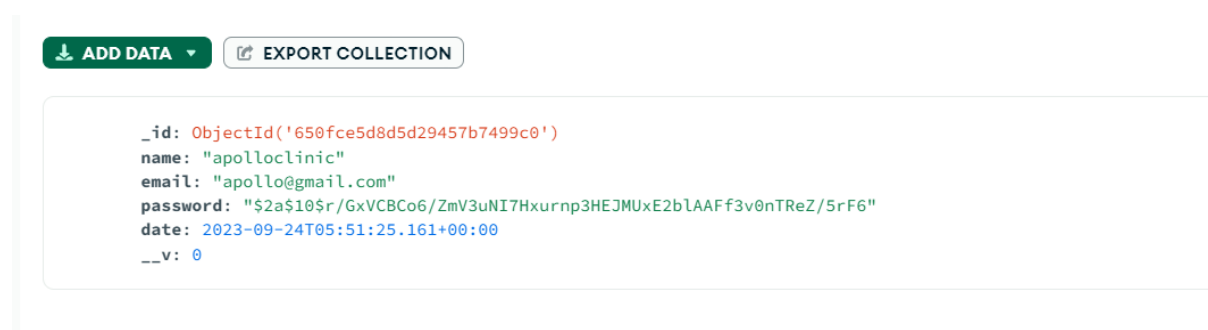
- **React as a Frontend Framework:** Recognize React as the chosen frontend framework and integrate it seamlessly with the backend, which manages vaccination data and schedules.

● Related Technologies and Tools

Technologies: React , Node , MongoDB , Express

Services : Gmail ,Twilio

The following picture gives the mongoDB snapshot of the hospital.



The following mongoDB snapshot is the details of the children which also includes the parent contact information.

```

_id: ObjectId('651708cc4a343e88db9bfb93')
parent_name: "suresh"
phone_num: "+917396675154"
birth_date: 2023-09-29T00:00:00.000+00:00
name: "apolloclinic"
bloodgroup: "o+"
weight: "12"
email: "clp05154@gmail.com"
address: "House no:19-188/3,ram mandir Street,badepally,station jadcherla, telan..."
age: 30
surname: "chilukuri"
time: "04:38"

```

MongoDB is a popular NoSQL database designed for flexibility and scalability. It's an excellent choice for storing user details. In MongoDB, data is organized into collections, each containing documents in BSON format, which is similar to JSON. User details, such as name, email, and profile information, can be stored as documents within a "users" collection. Each document represents an individual user, making it easy to retrieve and update user data. The schema flexibility allows for dynamic fields, making it adaptable to evolving user requirements. Additionally, MongoDB supports powerful querying and indexing for efficient data retrieval, making it suitable for a wide range of applications.



Twilio's messaging service is a cloud-based platform that empowers businesses and developers to send and receive SMS (Short Message Service) and MMS (Multimedia Messaging Service) messages programmatically. It offers a robust API for seamless integration into applications, enabling two-way communication with users via text messages. Twilio's global reach allows for international messaging, handling complexities like carrier regulations. It supports rich content delivery, including images and videos. Developers can track message status and delivery receipts, ensuring reliable communication. Twilio's message service is commonly used for applications like appointment reminders, notifications, customer support, and marketing campaigns, providing a versatile solution for SMS and MMS communication.



Gmail is a popular email service provided by Google. It offers users a free, web-based email platform with a user-friendly interface. Some key features of Gmail include ample storage space, efficient email organization through labels and filters, powerful search capabilities, and integration with other Google services like Google Drive and Google Calendar. It supports both personal and business accounts, and its mobile app allows access on various devices. Gmail also includes robust spam filters and security features to protect users from phishing and malware. Its widespread use and continuous updates make Gmail a reliable and versatile choice for sending, receiving, and managing emails.

4.SYSTEM DESIGN

4.1 Introduction

Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic, semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows:

1. User Model View

This view represents the system from the users' perspective. The analysis representation describes a usage scenario from the end-users' perspective.

2. Structural Model View

In this model, the data and functionality are arrived from inside the system. This model view models the static structures.

3. Behavioural Model View

It represents the dynamic of behavioural as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

4. Implementation Model View

In this view, the structural and behavioural as parts of the system are represented as they are to be built.

5. Environmental Model View

In this view, the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

4.2.1 USE CASE DIAGRAMS

To model a system, the most important aspect is to capture the dynamic behaviour. To clarify a bit in details, dynamic behaviour means the behaviour of the system when it is running/operating.

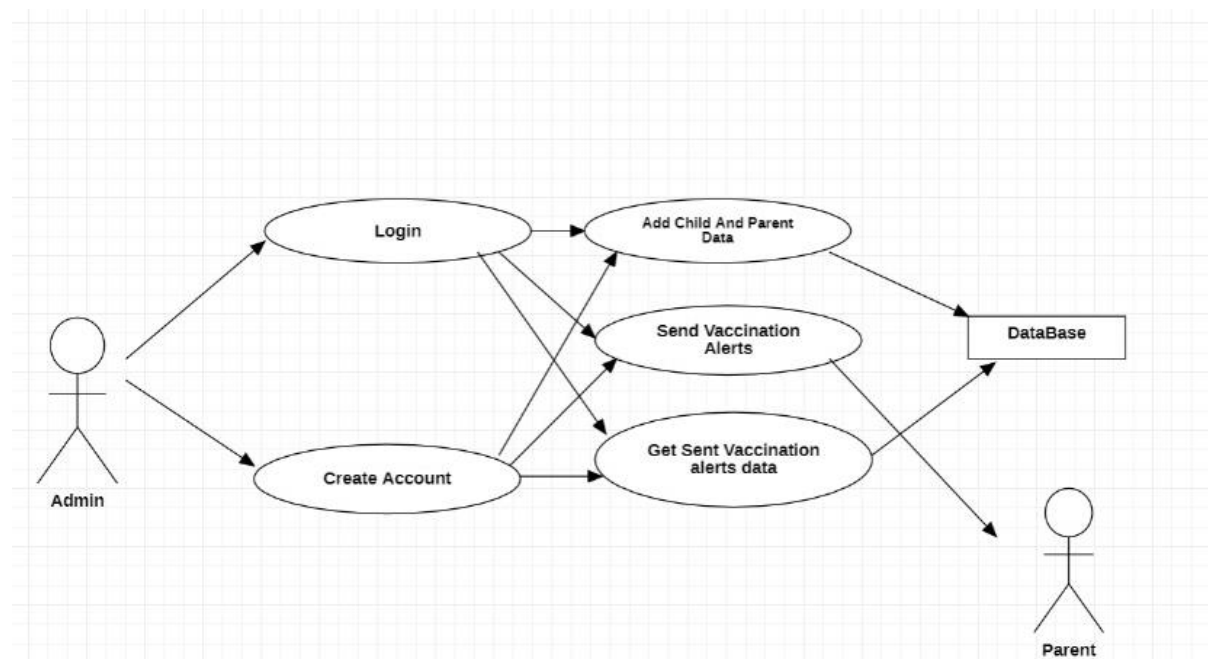
So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are

five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So, use case diagrams are consisting of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. So, to model the entire system numbers of use case diagrams are used. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So, when a system is analysed to gather its functionalities use cases are prepared and actors are identified.

In brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors.



4.2.2 SEQUENCE DIAGRAM

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modelling a new system.

The aim of a sequence diagram is to define event sequences, which would have a desired outcome. The focus is more on the order in which messages occur than on the message per se. However, the majority of sequence diagrams will communicate what messages are sent and the order in which they tend to occur.

Basic Sequence Diagram Notations

Class Roles or Participants

Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Activation or Execution Occurrence

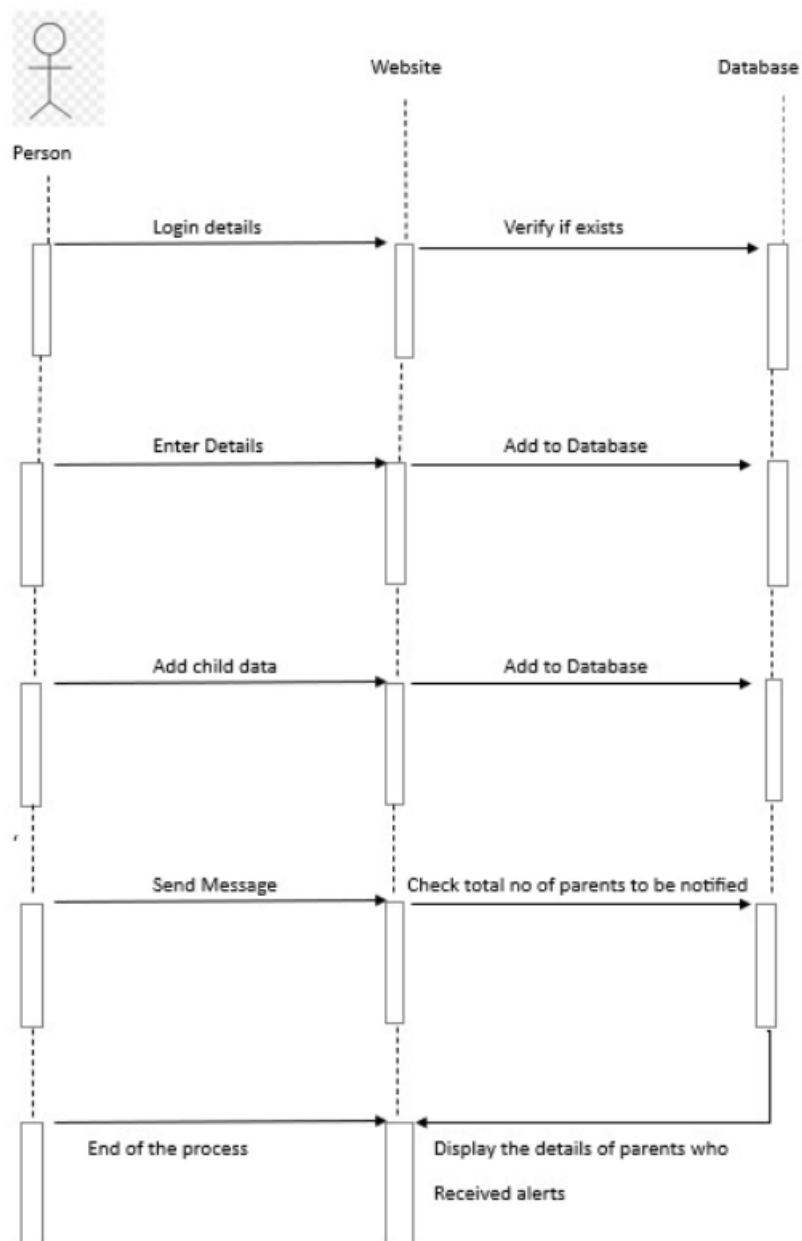
Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin grey rectangle placed vertically on its lifeline.

Messages

Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.

Lifelines

Lifelines are vertical dashed lines that indicate the object's presence over time.



Destroying Objects

Objects can be terminated early using an arrow labelled "<< destroy >>" that points to an X. This object is removed from memory. When that object's lifeline ends, you can place an X at the end of its lifeline to denote a destruction occurrence.

Loops

A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets []. When modelling object interactions, there will be times when a condition must be met for a message to be sent to an object. Guards are conditions that need to be used throughout UML diagrams to control flow.

4.2.3 ACTIVITY DIAGRAM

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination. It is also suitable for modelling how a collection of use cases coordinates to represent business workflows.

- Identify candidate use cases, through the examination of business workflows
- Identify pre- and post-conditions (the context) for use cases
- Model workflows between/within use cases
- Model complex workflows in operations on objects
- Model in detail complex activities in a high-level activity Diagram

4.2.4 CLASS DIAGRAM

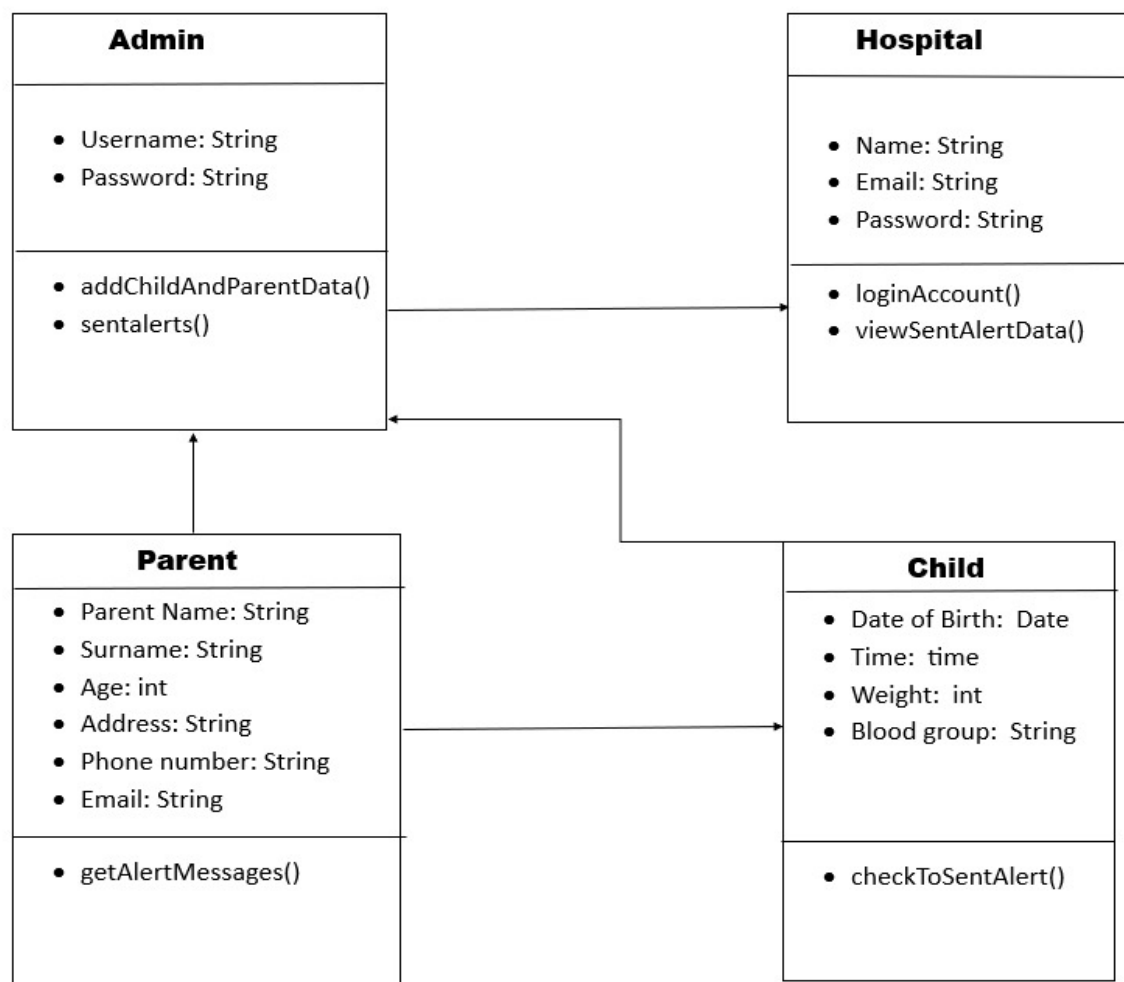
Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object- oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

Purpose of Class Diagrams

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction. UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.



● **Description of the Approach/Method Used**

Vaccination Alert attempts to find a solution for the parents specially for those who forget about their babies vaccines. This product increases the vaccination rate also.

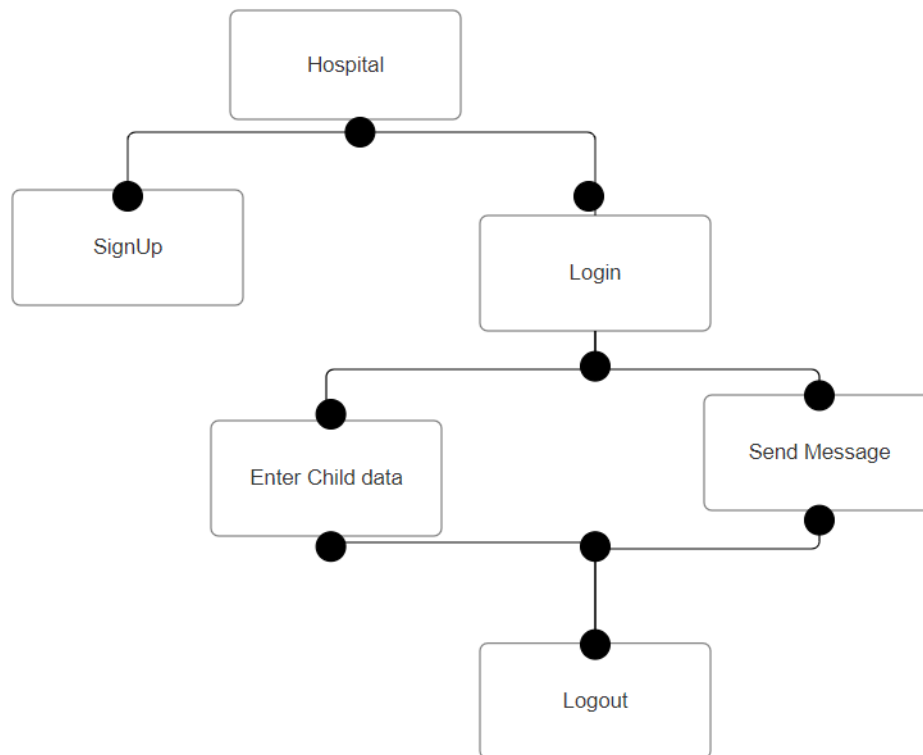
The main functionalities of this application is:

- Register to the software.
- Login/Logout.
- Send Vaccination Alert to the users.
- Send message for updating infants records.
- Provide Doctors.

Initially every hospital has to register by providing the credentials. When the child is born the details have to be uploaded in the website by logging in with the registered details.

When a registered hospital logs they can undertake two type of actions.

- 1.Entering the details of the new born child and saving them.
- 2.Sending the vaccination alert. It can be done just by clicking a button . The details of the parents who had been notified will be displayed.
- 3.when the parent have a registered email id the alert is being done through email else it will be done using the message.



● Software/Hardware Used (if applicable)

This system is designed to be transparent to its users and hence all the complexity is hidden from the user, i.e., user has no need to take care about the internal working. The user will interact with system using the GUI.

➤ *User Interfaces*

- This section provides a detailed description of all inputs into and outputs from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.
- SMTP protocol is used for sending mails.

➤ **Hardware Interfaces**

No specific hardware is required. The website needs just a software compatible hardware on which website can run.

- **Operating System:** We have chosen windows operating system for its best support, performance and user friendliness.
- **Database:** To save the records of the users and their details, SQL database is used.

5. Implementation

● Detailed Explanation of How the Project Was Executed

It is a react application connected to the mongoDB database to store the details of the hospital and the child details along with the contact details.

The user interface is built using the components by the integration with the bootstrap in order to enhance the responsiveness.

When we open the editor and we run the react application the website will be running in the port 3000.

There will be a dashboard containing the login button and signup button.

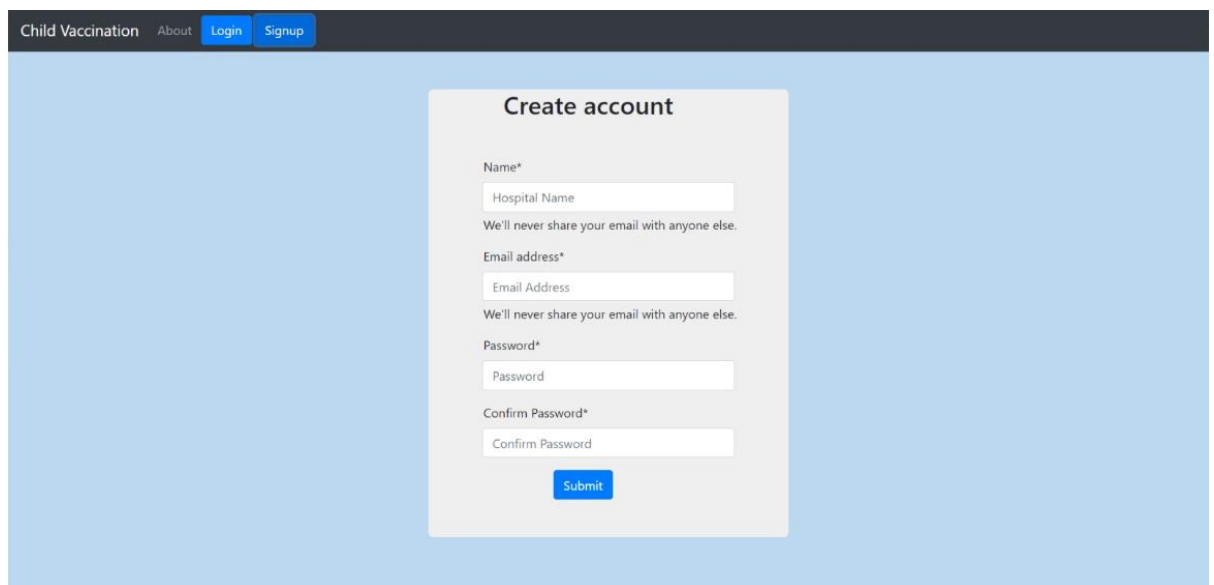
Signup button functionality is similar to register and the registered hospitals can use the login button.

The dashboard will be as follows.



Hospital Functionality:-

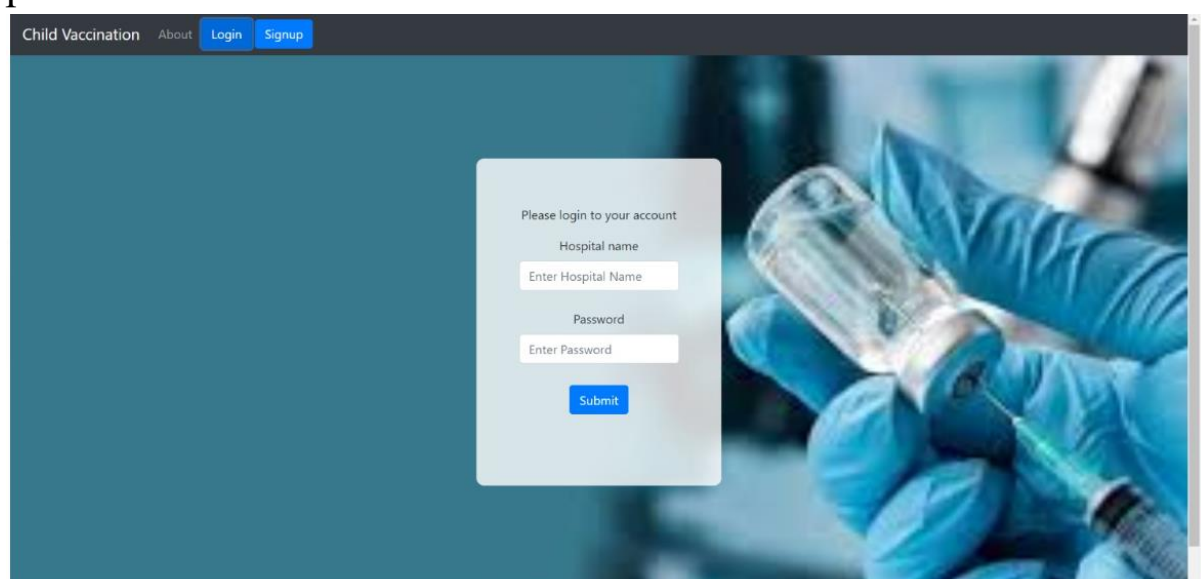
- **Login/Logout** –Hospital has authority to login/logout to the system and manage the software.
- **Generate** - Hospital can generate infants record.
- Sign up page will be as follows
- Hospital can create account by entering the credentials.



The screenshot shows a web browser window with the title 'Child Vaccination'. The navigation bar includes 'About', 'Login', and 'Signup' buttons. The main content area features a 'Create account' form with the following fields and labels:

- Name***: Input field with placeholder text 'Hospital Name'.
- We'll never share your email with anyone else.
- Email address***: Input field with placeholder text 'Email Address'.
- We'll never share your email with anyone else.
- Password***: Input field with placeholder text 'Password'.
- Confirm Password***: Input field with placeholder text 'Confirm Password'.
- Submit**: A blue button at the bottom of the form.

- **View infants record** – To check infants growth, hospital can view infants record.
- Hospital has to use the specified button to send message to parents.



The screenshot shows a web browser window with the title 'Child Vaccination'. The navigation bar includes 'About', 'Login', and 'Signup' buttons. The main content area features a login form with the following fields and labels:

- Please login to your account**: Header text above the input fields.
- Hospital name**: Input field with placeholder text 'Enter Hospital Name'.
- Password**: Input field with placeholder text 'Enter Password'.
- Submit**: A blue button at the bottom of the form.

The background of the page shows a close-up of a hand in a blue glove holding a syringe and a vial.

When a hospital already registered then to upload the child data the hospital can provide the name and password in order to login.

Once the hospital logs in then they can perform two types of operations.

They are uploading the child information and send the message to the parents by clicking a button.

While sending the message we are using two types of services.

1.Twilio

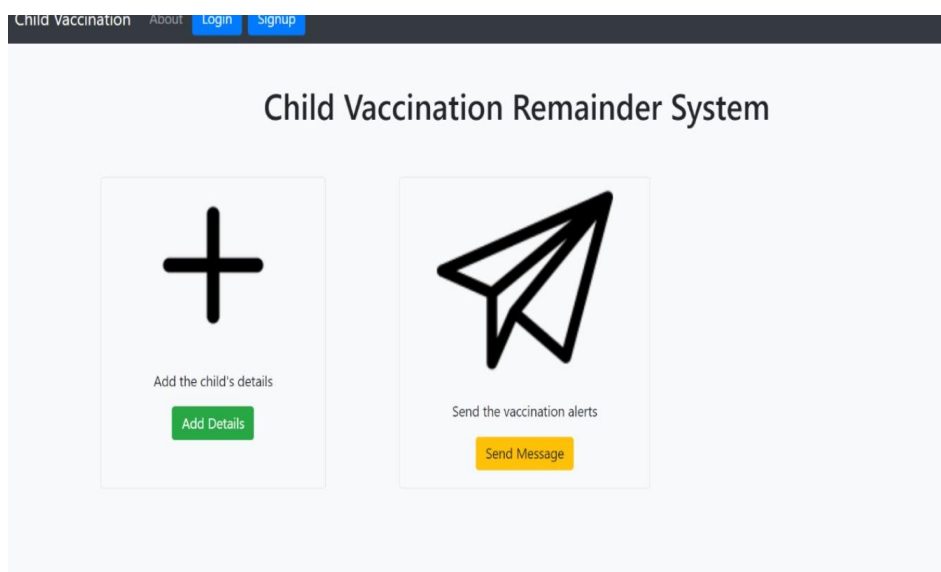
2.Gmail

The parents who have registered mail id the notification will be sent using the gmail service.

And those who don't have mail id will receive the message alert.

Message contains the information regarding the child vaccination date and the type of vaccine.

These are the options that are visible to the logged in hospital user.



Caretaker's Functionality:-

- **Provide information** – Caretaker will give all the details of their child like age, weight, height, any disabilities etc.
- Provide the contact information to receive the alerts.
- **Receive Notifications**
- The caretaker have to provide the details in the registration form regarding the contact information and the child details.

Registration Form

HospitalName:

Parent Information

Surname

Parent Name

Age

Address

Phone Number

Email

Child Information

Child Date Of Birth

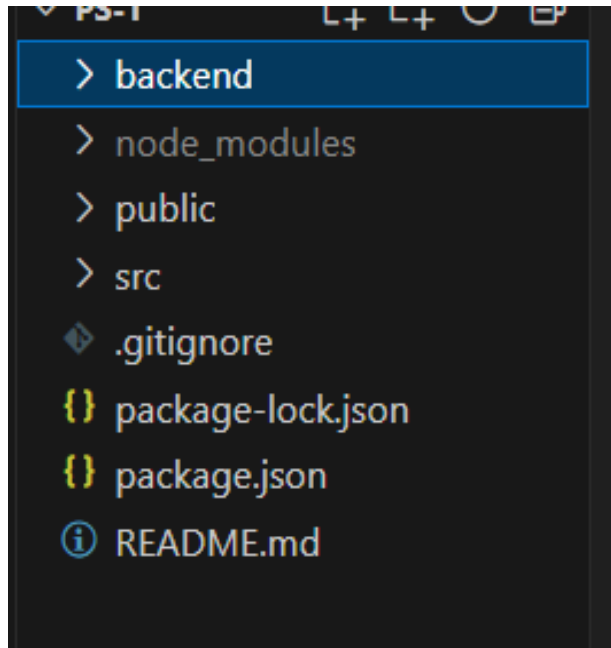
Child Weight

Blood Group

Time

back
Submit

● Code Structure and Architecture



The components in the application are placed in the components folder inside the src folder.

All the routes and database connectivity is present in the backend folder.

Package.json contains information regarding all the dependencies and installations in the project.

● Technical Challenges and Solutions

The major challenge which we had faced is the user authentication which we had overcome using the jsonwebtoken. It is also helpful in order to store the session data.

The validation of the fields is being handled while storing the data in the database.

6. TESTING

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of Validation and Verification

1. SOFTWARE VALIDATION

Validation is process of examining whether or not the software satisfies the user requirements. It is carried out at the end of the SDLC. If the software matches requirements for which it was made, it is validated. Validation ensures the product under development is as per the user requirements. Validation emphasizes on user requirements.

2. SOFTWARE VERIFICATION

Verification is the process of confirming if the software is meeting the business requirements and is developed adhering to the proper specifications and methodologies. Verification ensures the product being developed is according to design specifications. Verification answers the question– "Are we developing this product by firmly following Verifications concentrate on the design and system specifications.

3. TARGET OF THE TEST ARE

- Errors -These are actual coding mistakes made by developers. In addition, there is a difference in output of software and desired output, considered as an error.
- Fault - When error exists fault occurs. A fault, also known as a bug, is a result of an error which can cause system to fail.
- Failure - failure is said to be the inability of the system to perform the desired task. Failure occurs when fault exists in the system.

4.BLACK-BOX TESTING

It is carried out to test functionality of the program. It is also called ‘Behavioral’ testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested ‘ok’, and problematic otherwise.

Black-Box Testing Techniques

- Equivalence class - The input is divided into similar classes. If one element of a class passes the test, it is assumed that all the class is passed.
- Boundary values - The input is divided into higher and lower end values. If these values pass the test, it is assumed that all values in between may pass too.
- Cause-effect graphing - In both previous methods, only one input value at a time is tested. Cause (input) – Effect (output) is a testing technique where combinations of input values are tested in a systematic way.
- Pair-wise Testing - The behavior of software depends on multiple parameters. In pair wise testing, the multiple parameters are tested pair-wise for their different values.
- State-based testing - The system changes state on provision of input. These systems are tested based on their states and input. WHITE-BOX TESTING It is conducted to test program and its implementation, in order to improve code efficiency or structure. It is also known as ‘Structural’ testing In this testing method, the design and structure of the code are known to the tester. The following are some White box testing techniques
- Control-flow testing - The purpose of the control-flow testing to set up a test case which covers all statements and branch conditions. The branch conditions are tested for both being true and false, so that all statements can be covered.
- Data-flow testing - This testing technique emphasis to cover all the data variables included in the program. It tests where the variables were declared and defined and where they were used or changed.

TESTING LEVELS

Testing itself may be defined at various levels of SDLC. The testing process runs parallel to software development. Before jumping on the next stage, a stage is tested, validated and verified. Testing separately is done just to make sure that there are no hidden bugs or issues left in the software.

Unit Testing

While coding, the programmer performs some tests on that unit of program to know if it is error free. Testing is performed under white-box testing approach.

Unit testing helps developers decide that individual units of the program are working as per requirement and are error free. Unit testing helps developers decide that individual units of the program are working as per requirement and are error free.

Integration Testing

Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passes and data updating etc.

System Testing

The software is compiled as product and then it is tested as a whole. This can be accomplished using one or more of the following tests:

- **Functionality testing** - Tests all functionalities of the software against requirement.
- **Performance testing** - This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environment conditions.
- **Security & Portability** - These tests are done when the software is meant to work on various platforms and accessed by number of persons.

Acceptance Testing

When the software is ready to hand over to the customer it has to go through last phase of testing where it is tested for user-interaction and response. This is important because even if the software matches all user requirements and if user does not like the way it or works, it may be rejected.

- **Alpha testing** - The team of developer themselves perform alpha testing by using the system as if it is being used in work environment. They try to find out how user would react to some action in software and how the system should respond to inputs.
- **Beta testing** - After the software is tested internally, it is handed over to the users to use it under their production environment only for testing purpose. This is not

as yet the delivered product. Developers expect that users at this stage will bring minute 65 problems, which were skipped to attend.

Regression Testing

Whenever a software product is updated with new code, feature or functionality, it is tested thoroughly to detect if there is any negative impact of the added code.

End-to-End Testing

End-to-End testing is a type of Software testing that not only validates the software system under test but also check its integration with external interfaces. It uses actual production like data and test environment to simulate real-time settings. End-to-End testing is also called Chain testing. End-to-End design framework consists of three parts: Build User functions, Build conditions, Build test cases.

7.SCREENSHOTS



This screenshot shows the 'Create account' form on the 'Child Vaccination' website. The header is dark grey with the text 'Child Vaccination' and links for 'About', 'Login', and 'Signup'. The form is centered on a light blue background. It has a title 'Create account' and five input fields: 'Name*' (with a placeholder 'Hospital Name'), 'Email address*' (with a placeholder 'Email Address'), 'Password*', and 'Confirm Password*'. Each input field is followed by the text 'We'll never share your email with anyone else.' Below the input fields is a blue 'Submit' button.

Child Vaccination

About

Login


Signup

Please login to your account

Hospital name

Password

Submit




Child Vaccination

About

Login


Signup

Child Vaccination Remainder System



Add the child's details

Add Details



Send the vaccination alerts

Send Message

Registration Form

HospitalName:

Parent Information

Surname

Please Enter Surname

Parent Name

Please Enter Parent Name

Age

Age

Address

Address

Phone Number

Email

Email

Child Information

Child Date Of Birth

dd-mm-yyyy

Child Weight

Weight

Blood Group

Blood Group

Time

--:--

back

Submit

Child Vaccination About Login Signup

Newborn Vaccination Schedule

Ensuring your newborn receives the recommended vaccinations is crucial for their health and well-being

Hepatitis B Vaccine

Schedule: Given at birth.

Description: Hepatitis B vaccine is available for all age groups. The hepatitis B vaccine is recommended for all infants, all children or adolescents younger than 19 years of age who have not been vaccinated, all adults age 19 through 59 years, and adults age 60 years or older with risk factors for hepatitis B infection. Adults who are 60 years or older without known risk factors for hepatitis B may also receive hepatitis B vaccine.

DTaP Vaccine (Diphtheria, Tetanus, and Pertussis)

Schedule: Given at 2 months, 4 months, 6 months, 15-18 months, and 4-6 years.

Description: DTaP may be given as a stand-alone vaccine, or as part of a combination vaccine (a type of vaccine that combines more than one vaccine together into one shot). DTaP is only for children younger than 7 years old. Different vaccines against tetanus, diphtheria, and pertussis (Tdap and Td) are available for older children, adolescents, and adults.

Hib Vaccine (Haemophilus Influenzae type b)

Schedule: Given at 2 months, 4 months, 6 months, and 12-15 months.

Description: Vaccines are available that can help prevent Haemophilus influenzae type b or Hib disease. These vaccines do not

23°C Partly cloudy Search 22:33 27-10-2023

8.FUTURE SCOPE

Creating an automated child vaccination alert system is a crucial step in ensuring that children receive timely vaccinations, which are vital for their health and the prevention of disease outbreaks. Here are some future work and recommendations for improving such a system:

1. Integration with Electronic Health Records (EHRs):

- Collaborate with healthcare providers and institutions to integrate the alert system with electronic health records. This will enable real-time access to a child's vaccination history, making the alerts more accurate and tailored to individual needs.

2. Customization and Personalization:

- Develop the system to consider the specific vaccination schedule for each child based on factors like age, medical history, and geographical location. This customization can help in providing more relevant alerts.

3. AI and Machine Learning:

- Implement AI and machine learning algorithms to predict the ideal vaccination schedule for a child based on historical data, health conditions, and demographic information. This can reduce the likelihood of missed vaccinations.

4.Phone call:-Instead of just a notification alert, it will be advanced in case if a call is made and machine learning model will explain about the vaccination schedule.

5. Multilingual Support:

- Ensure the system supports multiple languages to cater to diverse populations, especially in multicultural areas.

6. Geolocation Services:

- Utilize geolocation services to determine the nearest vaccination centers and provide directions to parents or caregivers. This feature can be particularly useful for remote or underserved areas.

7. Integration with Immunization Registries:

- Work with national and regional immunization registries to automatically update vaccination records and confirm completed vaccinations. This can reduce discrepancies and ensure the most up-to-date information.

8. Feedback Mechanism:

- Include a feedback mechanism that allows parents and healthcare providers to report issues or inaccuracies, fostering continuous improvement.

9. Data Security and Privacy:

- Prioritize data security and privacy by implementing encryption, access controls, and compliance with relevant regulations such as GDPR (General Data Protection Regulation).

10. Community Engagement and Awareness:

- Develop educational materials and campaigns to raise awareness about the importance of child vaccination and the use of the automated alert system.

11. Monitoring and Evaluation:

- Continuously monitor the effectiveness of the system in increasing vaccination rates and reducing vaccine-preventable diseases. Make adjustments based on the results of this evaluation.

12. Healthcare Provider Collaboration:

- Foster collaboration with healthcare providers to ensure they are aware of and engaged with the system. This can enhance the accuracy of data and provide a holistic approach to child healthcare.

By implementing these recommendations, an automated child vaccination alert system can become a powerful tool in promoting child health and reducing the incidence of vaccine-preventable diseases. It is essential to adapt and improve the system continuously to ensure its long-term success and effectiveness.

9. Conclusion

• Summary of Achievements

An automated child vaccination alert system has the potential to achieve several significant milestones and benefits, including:

1. **Improved Immunization Rates:** The system can significantly boost child immunization rates by sending timely reminders to parents and caregivers about upcoming vaccinations. This ensures that children receive their shots on schedule, reducing the risk of preventable diseases.
2. **Timely Notifications:** The system is designed to send automated alerts, either via SMS, email, making it easy for parents to remember and act on vaccination appointments.
3. **Reduced Missed Vaccinations:** By providing continuous reminders and tracking vaccination records, the system helps reduce the number of missed vaccinations, contributing to better community immunity.
4. **Healthier Children:** By ensuring children are up to date with their vaccinations, the system contributes to overall child health, preventing potentially life-threatening illnesses.

5. **Lower Healthcare Costs:** Timely vaccinations result in fewer hospitalizations and medical expenses, saving healthcare costs for both families and governments.

6. **Data Management:** The system can help maintain accurate vaccination records for children, making it easier for healthcare providers to track their immunization history and plan future vaccinations.

7. **Real-time Updates:** The system can incorporate real-time updates on vaccines and any changes in immunization schedules, ensuring that children are always receiving the most current and effective vaccines.

8. **Personalized Reminders:** The system can be tailored to send personalized reminders, accommodating specific vaccine schedules and individual child needs.

9. **Public Health Impact:** By promoting widespread vaccination, the system contributes to community immunity, helping protect vulnerable populations, such as infants and those with compromised immune systems.

10. **Research and Reporting:** The system can also aid in data collection and reporting for public health agencies, facilitating research and analysis of vaccination trends and coverage rates.

In summary, an automated child vaccination alert system can achieve several essential milestones, from improving immunization rates and child health to reducing healthcare costs and aiding public health efforts. It plays a crucial role in promoting preventive healthcare and protecting children from vaccine-preventable diseases.

• Contributions to the Field

Automated child vaccination alert systems play a crucial role in ensuring that children receive timely vaccinations, which are essential for their health and well-being. Several contributions have been made to this field to enhance the efficiency and effectiveness of such systems:

1. **Electronic Health Records (EHR) Integration:** Integrating automated child vaccination alert systems with electronic health records helps healthcare providers keep track of a child's vaccination schedule. This ensures that healthcare professionals have up-to-date information on a child's vaccination status, allowing for timely alerts and reminders.
3. **Text Messages and Phone Calls:** Automated systems can send text messages or make automated phone calls to parents or guardians to remind them about upcoming vaccination appointments. This approach is particularly effective in regions with limited smartphone access.
4. **Machine Learning and AI:** Machine learning and artificial intelligence can help predict when a child is due for their next vaccination based on their vaccination history and age. These

systems can also analyze vaccination coverage and identify areas with low immunization rates.

5. **Geospatial Data and Mapping:** Integrating geospatial data and mapping tools can help identify under-vaccinated areas and target vaccination campaigns more effectively. This is especially valuable for ensuring equitable access to vaccines.

6. **Multilingual Support:** In areas with diverse populations, providing vaccination alerts in multiple languages can help reach a broader audience and ensure that language barriers do not impede vaccination efforts.

7. **Immunization Registries:** Many regions have established immunization registries to maintain comprehensive records of vaccinations. Automated alert systems can access and use data from these registries to send timely reminders to parents and guardians.

8. **Data Analytics and Reporting:** Automated systems can provide healthcare professionals and public health authorities with data analytics and reporting tools. These tools can help in monitoring vaccination coverage rates and identifying areas that require targeted interventions.

9. **Integration with Public Health Initiatives:** Some automated alert systems are integrated with broader public health initiatives. For example, they can be linked to campaigns for distributing vaccines,

conducting outreach programs, and providing educational resources to parents and communities.

10. **Behavioral Insights:** Research in behavioral science has contributed to the design of more effective alert messages. Understanding how people make decisions about vaccinations can help create messages that are more likely to motivate parents to get their children vaccinated.

11. **Emergency Alert Systems:** In times of outbreaks or emergencies, automated alert systems can send urgent notifications to parents and healthcare providers, encouraging immediate vaccination to prevent the spread of diseases.

12. **Data Privacy and Security:** Efforts have been made to ensure the privacy and security of the data within these systems, as they often contain sensitive healthcare information. Compliance with data protection regulations is crucial.

Automated child vaccination alert systems continue to evolve with advancements in technology, data analysis, and healthcare practices. Their contributions to the field are critical for achieving high vaccination coverage rates and reducing the incidence of vaccine-preventable diseases in children.

10. REFERENCES

- ❖ <https://machinelearningmastery.com/>
- ❖ <https://iq.opengenus.org/>
- ❖ https://en.wikipedia.org/wiki/Generative_adversarial_network
- ❖ <https://nevonprojects.com/child-vaccination-management-system-using-python/>

• Code Snippets

```
const twilio = require('twilio');
const nodemailer = require('nodemailer');
const Mail = require('nodemailer/lib/mailer');

const accountSid = 'AC14c3738989716815241cede315cdea0d';
const authToken = '9702cb99cfc9623818b50406d647920a';
const twilioNumber = '+16053706224';
const client = twilio(accountSid, authToken);
```

1. ``const twilio = require('twilio');``: This line imports the Twilio Node.js module, which allows for programmatic interaction with Twilio's communication services.
2. ``const accountSid`` and ``authToken``: These variables store your Twilio account's unique identifier (SID) and authentication token, which are used to authenticate your application with Twilio's services.
3. ``const twilioNumber``: This variable holds the Twilio phone number that will be used as the sender for outgoing SMS messages and phone calls.
4. ``const client = twilio(accountSid, authToken);``: This line initializes a Twilio client by passing in your account SID and authentication token. The client is then used to interact with Twilio's services, such as sending SMS messages or making phone calls.

With these settings in place, you can use the ``client`` to send SMS messages or initiate phone calls programmatically, which is often used in applications for purposes like two-factor authentication, notifications, and more.

```

backend > JS db.js > ...
1  const mongoose=require("mongoose");
2  const mongoURI="mongodb://0.0.0.0:27017/project"
3  const connectDB = async () => {
4      try {
5          mongoose.set('strictQuery', false)
6          mongoose.connect(mongoURI)
7          console.log('Mongo connected')
8      } catch(error) {
9          console.log(error)
10         process.exit()
11     }
12 }
13
14 connectDB()

```

1. ``const mongoose = require("mongoose");``: This line imports the Mongoose library, which is a popular Node.js library for MongoDB object modeling and interaction.

2. ``const mongoURI = "mongodb://0.0.0.0:27017/project";``: This line defines the URI (Uniform Resource Identifier) for the MongoDB database you want to connect to. It specifies the database server's address (0.0.0.0) and port (27017), as well as the name of the database ("project").

3. ``const connectDB = async () => { ... }``: This code defines an asynchronous function named ``connectDB`` that will attempt to connect to the MongoDB database.

4. Inside the ``connectDB`` function:

- ``mongoose.set('strictQuery', false)``: This line disables the strict mode for queries. In Mongoose, strict mode is a setting that enforces a

schema structure. Setting it to `false` allows for more flexible data handling.

- `mongoose.connect(mongoURI)`: This line attempts to connect to the MongoDB database using the specified URI.

- `console.log('Mongo connected')`: If the connection is successful, it logs "Mongo connected" to the console.

- `try { ... } catch (error) { ... }`: This code uses a try-catch block to handle potential errors during the database connection process.

- If there is an error during the connection attempt, the error message is logged to the console using `console.log(error)`.

- `process.exit()`: If an error occurs, this line terminates the Node.js process. This is often used in development to exit the application when a critical error is encountered.

5. Finally, `connectDB()`: This line calls the `connectDB` function, which initiates the database connection when the script is executed.

Hospital SignUp schema

```

const mongoose = require('mongoose');
const { Schema } = mongoose;
const UserSchema = new Schema({
  name: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true
  },
  date: {
    type: Date,
    default: Date.now
  }
});
const User = mongoose.model('user', UserSchema);
module.exports = User;

```

Child Registration form Schema

```

const mongoose = require('mongoose');
const { Schema } = mongoose;
const ChildSchema = new Schema({
  parent_name: {type: String, required: true},
  phone_num: {type: String, required: true},
  birth_date: {type: Date, required: true},
  name: {type: String, required: true},
  bloodgroup: {type: String, required: true},
  weight: {type: String, required: true},
  email: {type: String},
  address: {type: String, required: true},
  age: {type: Number, required: true},
  surname: {type: String, required: true},
  time: {type: String, required: true}
}, {versionKey: false});
const myModel = mongoose.model("NEWCOL", ChildSchema);
module.exports = myModel;

```

```

"dependencies": {
  "@emailjs/browser": "^3.11.0",
  "@testing-library/jest-dom": "^5.16.5",
  "@testing-library/react": "^13.4.0",
  "@testing-library/user-event": "^13.5.0",
  "axios": "^1.4.0",
  "bcrypt": "^5.1.1",
  "bcryptjs": "^2.4.3",
  "concurrently": "^8.2.0",
  "cors": "^2.8.5",
  "express": "^4.18.2",
  "express-validator": "^7.0.1",
  "mongoose": "^7.4.1",
  "nodemailer": "^6.9.4",
  "nodemon": "^3.0.1",
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "react-router-dom": "^6.16.0",
  "react-scripts": "5.0.1",
  "twilio": "^4.14.0",
  "web-vitals": "^2.1.4"
},

```

- **List of dependencies**

The code snippet specified above are list of dependencies project, typically managed using a package manager like npm (Node Package Manager). Each dependency listed has a name and a version number.

Here's a breakdown of the dependencies listed in `package.json` file:

1. **@emailjs/browser (Version 3.11.0):** A library for sending email directly from the browser using JavaScript. It is often used for client-side email functionality.
2. **@testing-library/jest-dom (Version 5.16.5):** Part of the Testing Library ecosystem, this library provides custom Jest matchers for making assertions in your tests.

3. `@testing-library/react` (Version 13.4.0): Part of the Testing Library ecosystem, this library provides utilities for testing React components.

4. `@testing-library/user-event` (Version 13.5.0): Another part of the Testing Library ecosystem, this library provides utilities for simulating user events like clicks and keyboard input in your tests.

5. `axios` (Version 1.4.0): A popular JavaScript library for making HTTP requests from the client-side. It simplifies the process of interacting with APIs.

6. `bcrypt` (Version 5.1.1): A library for hashing passwords and verifying hashed passwords. It's commonly used for user authentication and password security.

7. `bcryptjs` (Version 2.4.3): A JavaScript implementation of the `bcrypt` hashing algorithm, primarily used for password hashing and validation.

8. `concurrently` (Version 8.2.0): A utility for running multiple commands concurrently in a single terminal window. This is often used in development scripts to run both server and client-side code at the same time.

9. `cors` (Version 2.8.5): A Node.js package for handling Cross-Origin Resource Sharing (CORS). It's used to configure which domains are allowed to access resources on your server.

10. `express` (Version 4.18.2): A popular Node.js web application framework used for building server-side applications and APIs.

11. `express-validator` (Version 7.0.1): A set of Express.js middlewares for data validation. It's often used for validating user input on the server.

12. `mongoose` (Version 7.4.1): A Node.js library for MongoDB, which simplifies working with MongoDB databases and provides an object data modeling (ODM) layer for Node.js.

13. `nodemailer` (Version 6.9.4): A module for sending email from Node.js applications. It's commonly used for email notifications and communication.

14. `nodemon` (Version 3.0.1): A utility that monitors changes in your Node.js application and automatically restarts the server when changes are detected, making it easier for development.

15. `react` (Version 18.2.0): The core library for building user interfaces in React applications.

16. `react-dom` (Version 18.2.0): The package that provides the integration between React and the Document Object Model (DOM). It's used for rendering React components in web applications.

17. `react-router-dom` (Version 6.16.0): A popular routing library for React applications, used for managing navigation and rendering different components based on the URL.

18. `react-scripts` (Version 5.0.1): A set of scripts and configuration used for managing and building React applications. It includes scripts for development, building, and testing.

19. **twilio** (Version 4.14.0): The Twilio Node.js module for working with the Twilio API. It allows you to send SMS messages, make phone calls, and perform other communication-related tasks using Twilio's services.

20. **web-vitals** (Version 2.1.4): A library for measuring web performance, specifically focusing on key user-centric performance metrics.