DESIGN AND IMPLEMENTATION OF COMPUTERIZED CHILD CARE INFORMATION SYSTEM

By

Amidu Dabor

Zainab Sesay

David Maada Gegbai

SUPERVISED BY

Dr. Hamza

Mr. M. Bangura

A PROJECT RESEARCH SUBMITTED TO THE

DEPARTMENT OF PHYSICS AND COMPUTER SCIENCE, NJALA UNIVERSITY

IN PARTIAL FULFILLMENT TO THE AWARD OF BSc. (Hons) IN COMPUTER SCIENCE

DECLARATION

This is to certify that this project which was done and written by Amidu Dabor, Zainab Sesay, and David Maada Gegbai has been approved by the Department of Physics and Computer Science, Njala University, in partial fulfillment for the award of Bsc. (Hons) in Computer Science.

SIGNATURES

…………………………….	**……..……………….**
Date	**Amidu Dabor**
	STUDENT
…………………………….	**……..……………….**
Date	**Zainab Sesay**
	STUDENT
…………………………….	**……..……………….**
Date	**David Maada Gegbai**
	STUDENT
…………………………….	**……..……………….**
Date	**Dr. Hamza**
	PROJECT SUPERVISOR

DEDICATION

We dedicate this work to Almighty God, our family, friends, lecturers, youtube, stackoverflow, geekforgeeks, Mozilla developers network (MDN), and other online learning platforms.

ACKNOWLEDGEMENT

Very sincerely, we first and foremost recognize the grace of God for seeing us through to this level of our project. To the staff and to the entire administration of the Department of Physics and Computer Science, Njala University, we are truly indebted. In spite of the unimaginable challenges that we met along the way to our academic pursuits, you made it possible for us to complete this project, through mentorship and through provision of the appropriate learning platforms we needed, to help us realise our full potentials.

This project is a landmark in our academic careers. We have been fortunate to learn theories and concepts which would have been impossible if we had not extensively carried out the needed research. We are grateful to a number of people who have guided and supported us throughout the research process, and who have provided assistance to our venture. We would first like to thank our supervisors, Dr. Hamza and Mr. M. Bangura, who guided us in selecting the final theme/scope for this research. Our supervisors were there throughout our preparation of the proposal and the conceptualization of its structure. We would not have been able to do the research and achieve learning in the same manner without their help and support. Their recommendations and instructions have enabled us to assemble and finish the project write-up effectively. We would also like to thank all of our instructors, lecturers, and teachers, who, throughout our educational career, have supported and encouraged us to believe in our abilities to work in highs and lows _ as in this our final project filled with challenges _ in order to achieve our goals. They directed us through various situations, enabling us to reach this accomplishment. Finally, our families have supported and helped us along the course of this project by giving encouragement and by providing the moral and emotional support we needed to complete the project. To them, we are eternally grateful.

ABSTRACT

This project is centered on computerized child care information system. The current process of recording children's medical information at Ola Durin Children's Hospital is a manual one. As a result of this procedure, numerous problems are being encountered. The Computerized Child Care Information System was designed to solve the problems with the manual process of taking records. The problems were identified after series of interviews and examinations of documents; and after which, an analysis was made, and a computerized procedure was then recommended. This project, in addition, brings with it, suggestions on how to implement the new procedure and how to overcome the obstacle(s) that would hinder the successful implementation of the system. The new system was designed as web-based, using HTML, CSS, JAVASCRIPT, MONGODB, and NodeJS technologies. These languages were chosen because of their easy syntaxes and features for developing web-based applications.

ORGANISATION OF WORK

Chapter one:

This chapter introduces the topic of this project, what it is all about, and what is to be achieved. It points out the statement of the problem, purpose of the study, aims and objective, definition, limitations, and definition of terms.

Chapter two:

This chapter reviews the literature on what people have been writing or said about the topic.

Chapter three:

This chapter aims at describing and analyzing the existing system. It goes further to give detailed explanations on the method used in fact findings, organizational structure, identifying the objective (s) of the existing system, input process, output analysis, information, flow diagram or chart problem of the existing system, and justifications for the new system.

Chapter four:

Explains the specification design of the output and input of the new system, its file design, procedure chart, system flowchart, and system requirement.

Chapter five:

It is where the recommendation and the conclusion were made.

TABLE OF CONTENTS

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND OF THE STUDY

Children are heritage of God, and the fruit of the womb is His reward. As arrows are in the hands of the mighty man, so are children of the youth happy is the man that hath his quiver full of them (Psalm 127:3-5). The African child occupies a strategic position in the family, in the society, and in the nation; which is not an over statement and cannot be underplayed. A child is considered as the life wire of the family and the society at large, to the extent that any couple that is not blessed with a child is looked upon with pity and sometimes sympathized with. There is the need to know who is a child, and what is child care.

In traditional and even contemporary African homes, various ethnic groups have different concepts of who is a child, while some ethnic groups see one as a child in so far as she can contribute to the development of the society. Some others see it in terms of one who has not yet attained the age of initiation into the age grade.

There is not one acceptable age, which is considered as a worthy definition of the upper limit of childhood. The age at which a child can become capable in law for his or her action is the age at which childhood begins in terms of the right to vote and to be voted for and the age considered by the federal ministry of youth and culture and education as being the upper limit of childhood different. Even internationally there are divergent definitions as to who is a child. Whereas the International Labour Organisation (ILO) and the United Nations (UN) population division refers to a child as those below 18 years of age. The 1989 Convention on the right of the child states that a child means every human being below the age of 18 years. Even though there are divergent opinions and views as to what is the age limit of a child is. It is the views of the authors that a child is anybody between birth to completion of physiologically/ psychological and physical development. In order words, anybody between ages zero of his/her birth to 18 years is a child.

Child care, on the other hand, is a kind of human acts that jeopardizes the physical, psychological, and the featuring of the child either intentionally or unintentionally.

Child care is an implied act of giving basic needs, rights and deeds of a child by parents/guardians, which will pair the well-being of the child. Caring can also take the form of adequate provision of resources for the welfare of the child both now and in future.

Also, there should not be any form of encounter with problems of misplacement of vital (sensitive) information or records as a result of services not being available or being rendered when needed.

1.2 STATEMENT OF THE PROBLEM

In every child care medical center, there are numerous problems that are encountered in regards to maintaining a cost-effective/cost-efficient and a well-organised database management system. A child medical care center like Ola Durin Children Hospital _ our case study _ encounters problem of the manual system in the area of documentation which includes misplacement of vital information or cases as a result of poor child care data management services that are being used. Duplication of efforts due to inconsistency in activities and in the long time taken in search of a file or of reported cases when needed for processing have not been going so well to uphold efficiency, effectiveness, accuracy, and a well-structured approach of child care data management.

This is the problem that is discovered that promoted for the design and implementation of a computerized child care information system to enable the aforenamed hospital to work more effectively.

1.3 PROBLEMS OF THE EXISTING SYSTEM

The problems that are obtainable from the existing system are inherent mostly from the current method of operation (The manual method) which includes the following:

LACK OF OFFICE SPACE
This is due to many working materials like files, paper, cartoons, shelves etc. that must be accommodated.

MANUAL LIMITATION
Many processes demand high mental exercise and care which if not properly carried out might cause errors in the records due to complexities of the processing.

MISSING ITEM
Most time, children’s records or even files are out of place. This can be caused by not being careful, oversight, over-stressed by any of such unpredictable factors.

TIMELINESS
Despite the high staff turnover, information is not usually prepared on time. Comprehensive laboratory result sheet, statement of laboratory result reference list etc. are usually never ready at the appropriate time.

POOR SECURITY
There is not enough security to keep data from unauthorized personnel. This means that in critical cases, records can be altered without due formalities.

RECORD UPDATE IS DIFFICULT

Due to the fact that records can only be searched out by going through file shelves, it becomes more tedious as opposed to glance some data which are operations, but mishandled the following are the inputs patients Name(surname), other Names, lab. No, provisional diagnosis, investigation required, sex, Age, Date, Health Institution etc.

1.4 JUSTIFICATION OF THE NEW SYSTEM

The initial problem of the existing system as outlined above hinders efficiency in the Institution and reduces effectiveness in data processing. That notwithstanding, initial changes and other crucial analysis require lots of labour and mathematical calculations which are perhaps quite numerous and complex.

A computer is known for its capability to perform complex and routine functions satisfactorily, notably difficult for man. Computerization therefore offers the benefits of cost and labour effectiveness, for Ola Durin Children Hospital to meet up with challenges of modern data processing. This has greatly motivated the design of this new system. Objectively, the system will provide computer-based tools and designs, suitable and technically acceptable for the COMPUTERIZED CHILD CARE INFORMATION SYSTEM.

1.5 PURPOSE OF STUDY

The purpose of this study is to design and implement a computerised system that will eliminate the above stated problems, as in misplacement of relevant information and files such as reported cases, documents, records, duplication of efforts and the long time that is required Durin searching and processing of cases and files.

1.6 AIMS AND OBJECTIVES

The objective of the project is to study how Ola Durin Children Hospital operates with managing its child care records, on which through thorough research/observations, a suitable design and implementation of a computerised solution can be provided.

It aims at detecting problems that pose obstacle with a view of modifying the operations and developing a new system that will be more efficient and accurate such as:

In the area of misplacement of vital documents, a computer will be used to record, and retrieve large volume of documents which will reduce the duplication of efforts due to inconsistency in activities and in the long time taken in search of files when they are required for processing.

1.7 DELIMITATION/ SCOPE OF THE STUDY

This study is specifically concerned with the computerized child care and carrying of the Ola Durin Children's Section of Cottage Hospital. Although a lot of activities are being performed by this center, this work is therefore concerned with childcare and caring manipulation of both the care and reporter of a case.

1.8 LIMITATIONS OF THE STUDY

The key limitations of our study are lack of time and other required resources. The study focuses on a specific area concerned with tracking and keeping records of all clients and personnel and processing them using a computer.

We could not obtain much relevant data from the administration of the named children hospital to add to enhance the work of our project. This prompts us to use data samples from external data sources on different online and offline child care medical management systems.

In spite of the hinderance of time and other resource factors, we could make a good start which has now led us to a higher level of the development process.

1.9 ASSUMPTION

Apparently, before starting this work, we assumed a successful completion by division, grace of God, and the cooperation of workers of Ola Durin Children Hospital in supplying us with data/information needed for the successful completion of this project.

In terms of material resources, monetary, and personnel resources, we were unable to meet up with them. In terms of the fact, it is believed that they are valid.

1.10 DEFINITION OF TERMS

A child: A child means every human being below the age of 18 years unless under the law applicable to the child majority is attained

Childcare: That is any kind of human act that jeopardizes the physical, psychological, and the futurity of the child either intentionally or unintentionally.

Child caring: This implies the act of giving basic needs, rights, and deals of the child by parents/guardians, peers, government, and cultural community which will pair the well-being of a child. Caring can also take the form of adequate provision of resources for the welfare of the child both now in the future.

Flowchart: This is the graphical representation of the logical steps and sequences involved in a procedure or program.

Data: This can be defined as groups of non-random symbols (words value figures) which represent things that have happened.

Information: This is the resource that enables the collection, management, control, and dissemination of information throughout an organisation.

Monitoring: This is to keep watch and see who denies children their rights, and take the necessary steps to stop this.

CHAPTER TWO

2.0 LITERATURE REVIEW

Computerization is the process of building a new system upon a computer technology for input, output, processing and storing. Computer entirely replaces the manual system that is using only paper, and pencil for processing.

Computers have replaced manual technology because of its ability to process large volume of data or even handle complex work (processing capability) at a very high speed. It gives out accurate result at each time except when it is fed with incorrect data, Garbage-in-garbage-out. Hence, the need for computerization is certified.

In Hospitals, computerization helps to keep accurate patients records in which case, one can call up a patient's record to find out necessary information about the patient when needed. This also helps to reduce redundancy in collecting patient's record and also eliminate the problem of missing of some patient's files.

Zachariyah et. Al (2015), in their project report on "An Automated Database Management System For Births and Deaths in Sierra Leone", stated that"birth is the complete expulsion or extraction of a child from its mother of a product of conception, irrespective of the duration of pregnancy, which after such separation, breathes or shows any other evidence of life, such as beating of the heart, pulsation of the umbilical cord or definite movement of voluntary muscles, whether or not the umbilical cord has been cut, or the placenta is attached; each product of such, irrespective of gestational age or whether alive or dead at the time of registration, and if they die at any time following births, they should also be registered and counted as deaths"

Barba D.L.A (1979), in his contribution, says that computerization does not only involve computer technology consisting to only hardware and software but also the communication link, that is, it establishes the link for data communication devices to interact and share data as well as transferring data/information from one location to another. Besides, computers can be used for keeping records and these records are always available whenever they are needed, and the need of carrying office file from one place to another is eliminated, and in most cases, some document may get lost or be tampered with Durin transfer.

Also, French CS. (1996) states that, a file is a document stored in the computer individually by name and is organized in a particular way with a well-defined structure consisting of a collection of records each of which are made up of files.

Henry C.L. commented that a typical organisation has a large number of files, many of which may be stored on a computer device. We call these data machine readable because one can use computer to process them. Paper files on the other hand are much less accessible. A large organisation of related files are part of a database.

French C.S also defined a database as a single organised collection of structured data stored with a minimum of duplication of data items so as to provide a consistent user of the system but is independent of programs that use the data. Databases are normally set up in order to meet the information needs of major parts of an organisation. It is not possible to construct a database in a single operation; it is usually a built-up section. Durin this process, it is possible to:
Add new “files” of data.
Delete existing data
Add new or update fields to record(s) already present in the database.

Create relationships between the data.

A database requires data being stored on large capacity direct access devices. The usual medium is the magnetic disk. For security purposes a copy of the database may be held on magnetic tape or disk.

Like in some clinics and general out-patients departments, patient's data may be duplicated, it is important to realize the duplication are minimized and controlled. This is referred to as controlled redundancy.

Although to the user, the database may appear as a collection of files, data in database is organized in a more complex way than data in conventional files. Database may be classified according to the approach taken to database organisation. The classes are relational, network, hierarchical and file inversions. But this project work discusses more on non-relational database (information system) that is: it uses collections and documents.

Data description must be standardized for this reason. A data description language (D.D.L.) is provided which must be compared to the declarations and processing statement in a conventional programming language.

Since complex files are processed in the database, a complex software system called database management system is required for construct, expansion and maintenance of the database. It provides the controlled interface between the user and the data in the database. The DBMS allocated storage of data.

It maintains indices so that any required data can be retrieved and so that separate items of data in the database can be cross-referenced. The DBMS provides facilities for different types of file processing such as processing a complete file (serially or sequentially), process required records (selective sequential or random), and retrieve individual records. It has the function of providing security for the data in the database.

Most computerized systems cannot accept data in forms that are customary to human communication such as speech or hand written documents. It is necessary therefore to present data to the computer in a way that provides easy conversion into its own electronic pulse-based form. This is commonly achieved by typing the data into keyboard devices that convert it into machine sensible forms. Data finally enters storage.

There is a distinction between data and information. By using the description between information storage and retrieval rather that storage and retrieval that emphasis is firmly placed upon something meaningful to a user rather than upon the technicalities of storage. It can also be stressed that the more the meaning that was to be represented and stored, the more complex the storage organisation and structure must be. As records are stored in these systems their contents are automatically indexed by the software. Subsequently, the use may be able to find every instance of selected record very quickly.

A general conclusion drawn from this is that, the data to be processed by the computer must be collected. The process of data collection then involves getting the original data, converting it from one medium to another, and finally getting it into the computer.

Abudullahi, J.I defines data collection as the process involved in getting the data from its points of original collection. It starts at the services of raw data and ends when valid data is within the computer in a form ready for processing. The process of data collections may involve any number of the following depending on the method used which includes the following:

- Data creation.
- Transmission of data.
- Data preparation.
- Possible conversion from one medium to another.
- Input of data to the computer from validation.
- Sorting
- Control-all stages must be controlled

Also, in processing the patients' record, data control measures should be involved. The following measures are stated below:

- Manual controls
- Data collection controls
- Validation checks
- Batch controls to ensure that all data is processed, preserving the integrity of maintained data, deleted, corrected and reprocesses all errors.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.0 RESEARCH METHODOLOGY

Research is an investigation in order to discover new factors through planning and systematic collection, analysis, and interpretation of data, whereas particular task, therefore, research methodology is a detailed description of what the researcher planned and procedure adopted in gathering new facts relevant to the project work.

It is therefore an established fact that without data, there can be no analysis. This is the crux of social science research. Data can be defined simply as basic facts and figure mostly numeric in nature, resulting from business economic and social activities of man.

3.1 METHOD OF DATA COLLECTION

In the course of this project, the following outlines the method of data collection used.

Primary method

Secondary method

3.1.1 PRIMARY METHOD

The direct observation method was used.

OBSERVATION

Some facts were recorded through our observation of some activities carried out on children care record in the named hospital.

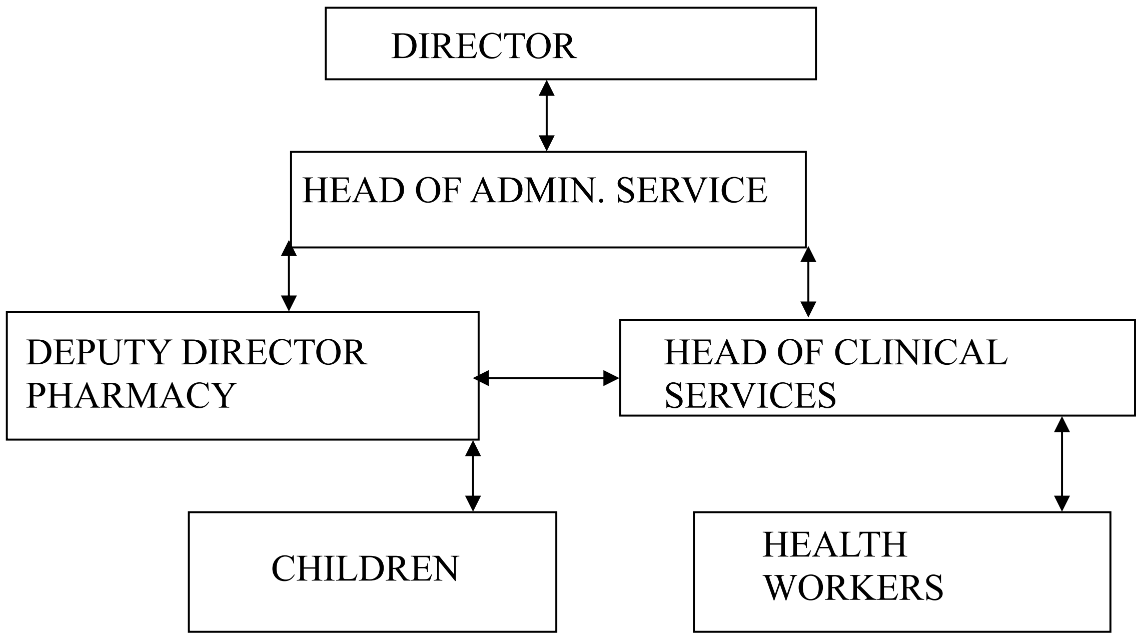
3.1.2 SECONDARY METHOD (OTHER METHOD)

Other method we used in data collection, as stated, was deductions from the children care record and hand book or cards in the hospital, online.

3.2 PROCESS/INFORMATION FLOW ANALYSIS

The highest profile from which information flows and are processed is “The admission while the lowest is “the children” in between this terminal are several departments whose functions are definitely instrumental to the effective and efficient processing as well as circulations of data. As you go from down (children level) to up (admission), facts and figures are passed in form of data while as you descend from highest to lowest, facts and figures are passed on as information. Below shows the trend of process information flow analysis of children’s record in the hospital.

INFORMATION FLOWCHART



3.3 INPUT AND OUTPUT DESIGN

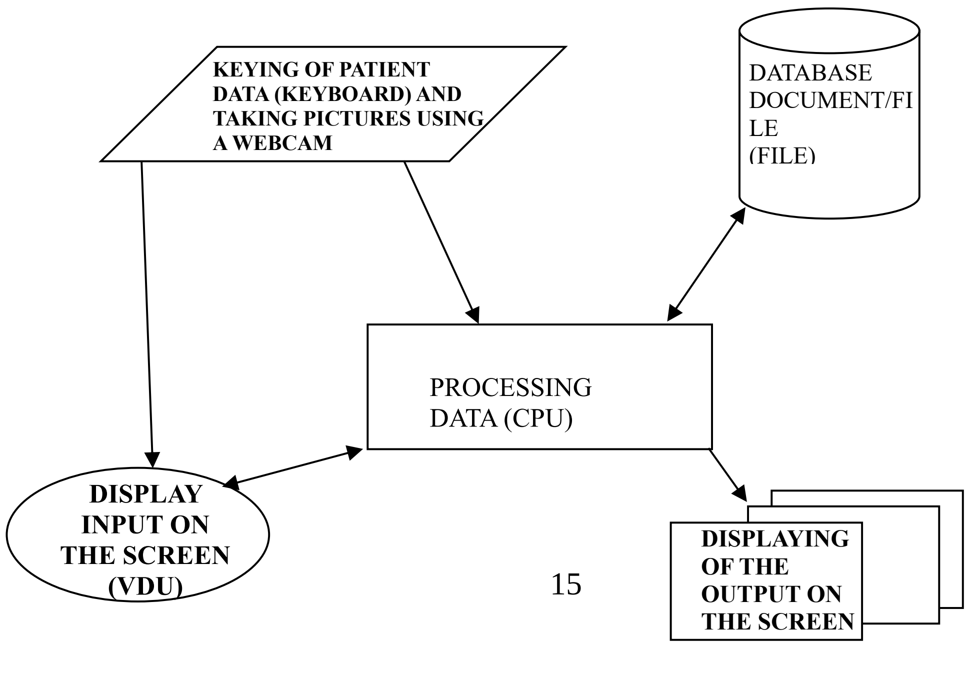
Data fed into the system talks more about the output desired for this project, the user shall input data via the keyboard, initialize command via the keyboard or with the aid of a backing storage. Then the output processed data can be accessed from the monitor, or stored in the online system database, the following describes the data design for the new system.

INPUT DATA (DESIGN)

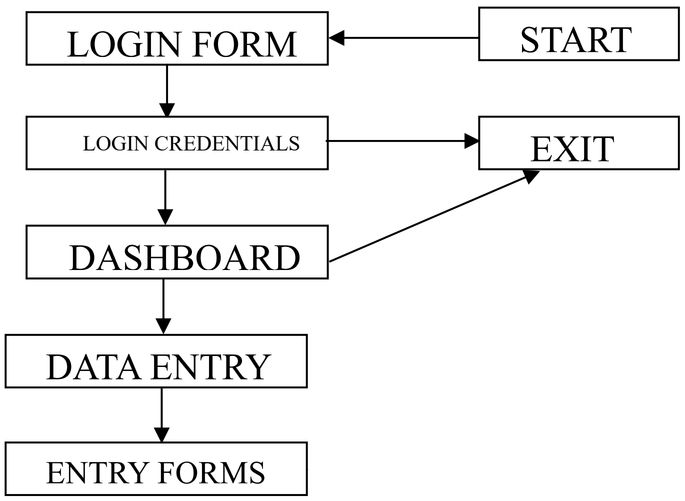
FILE DESIGN

File are sets of records which must be retained over a number of operational cycles of the system, because of the volume of information that computer holds in storage-keeping where other storage tools are limited. Filing is adopted to hold records in this case. Specially, random are used in this project.

3.4 SYSTEM FLOWCHART


KEYING OF PATIENT DATA (KEYBOARD) AND TAKING PICTURES USING A WEBCAM
DATABASE DOCUMENT/FILE (FILE)
PROCESSING DATA (CPU)
DISPLAY INPUT ON THE SCREEN (VDU)
DISPLAYING OF THE OUTPUT ON THE SCREEN

3.5 ARCHITECHURAL DESIGN



CHAPTER FOUR

PROGRAM TESTING AND IMPLEMENTATION

4.0 INTRODUCTION

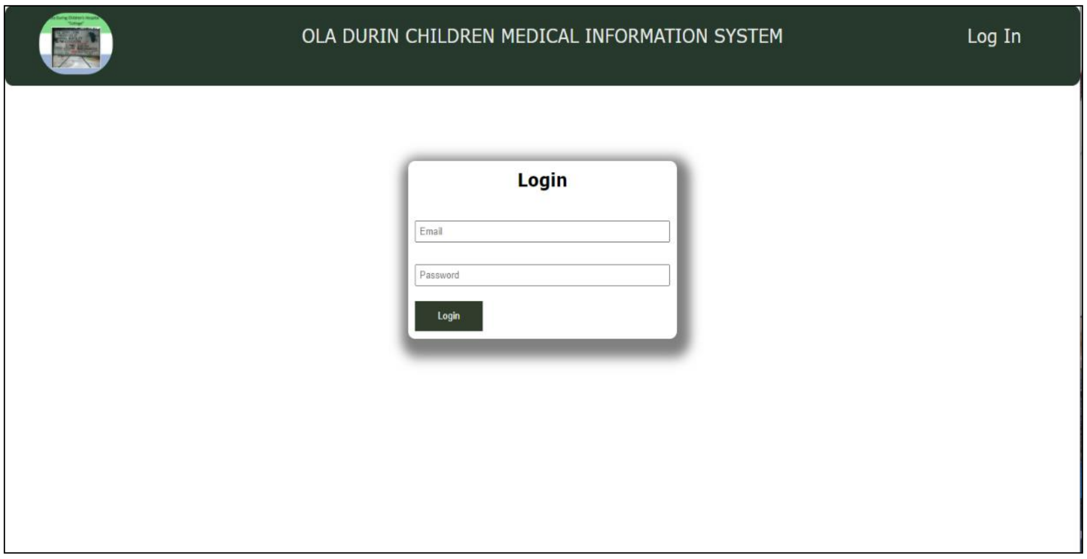
The program design was considered in the following two areas:

1) **Modular design:** Creation of modules was necessary since we realize that the system would be made of different units which would be somewhat difficult to design as one whole unit. We therefore created representative modules for the complex whole.

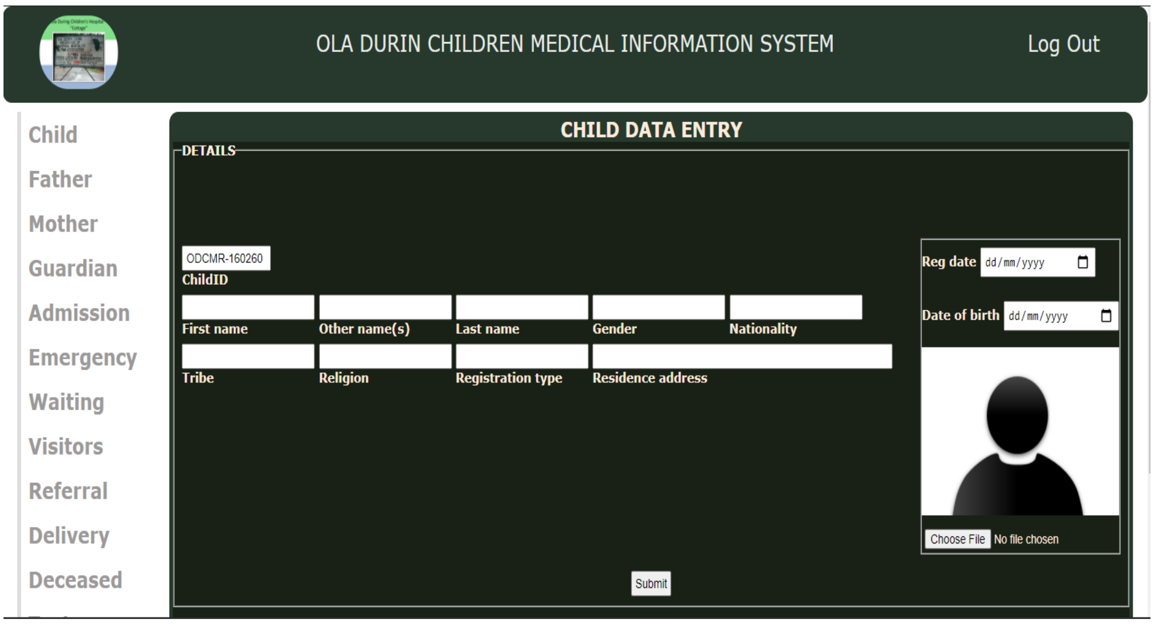
2) **Actual Design:** At this point, the individual modules so far created were transformed into actual working design. This design stage involves the creation of forms and placing of necessary objects on these forms.

4.1 PROGRAM DATA ENTRY FORMS AND SOURCE CODE

Users Login



Child



ODCMR-160260
ChildID

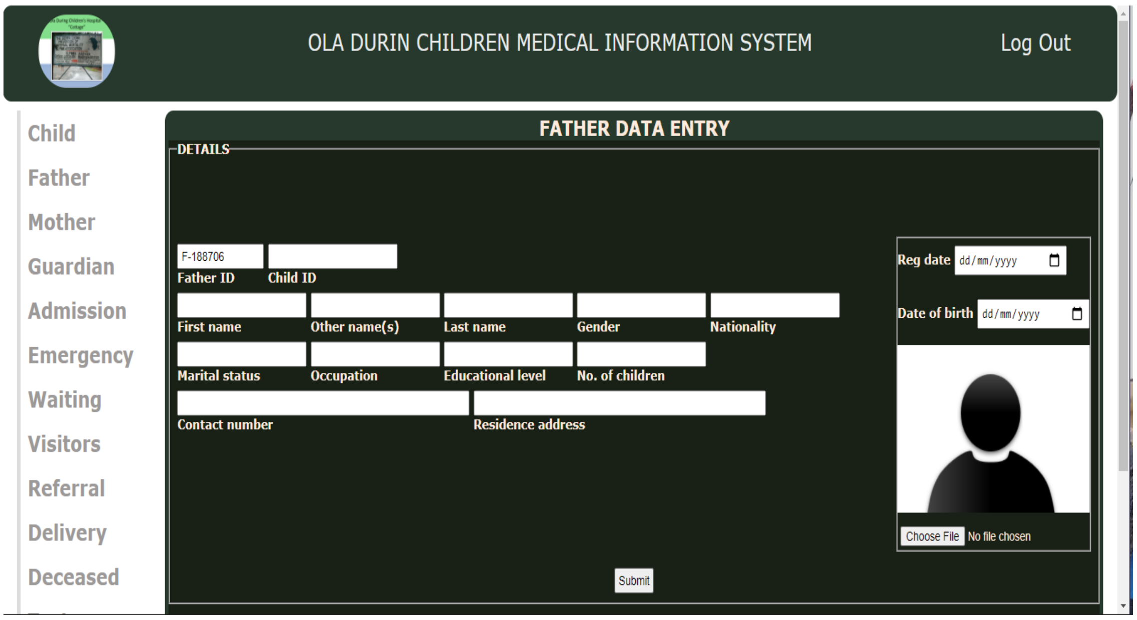
First name | Other name(s) | Last name | Gender | Nationality

Tribe | Religion | Registration type | Residence address

Reg date dd/mm/yyyy

Date of birth dd/mm/yyyy

Father



F-188706
Father ID | Child ID

First name | Other name(s) | Last name | Gender | Nationality

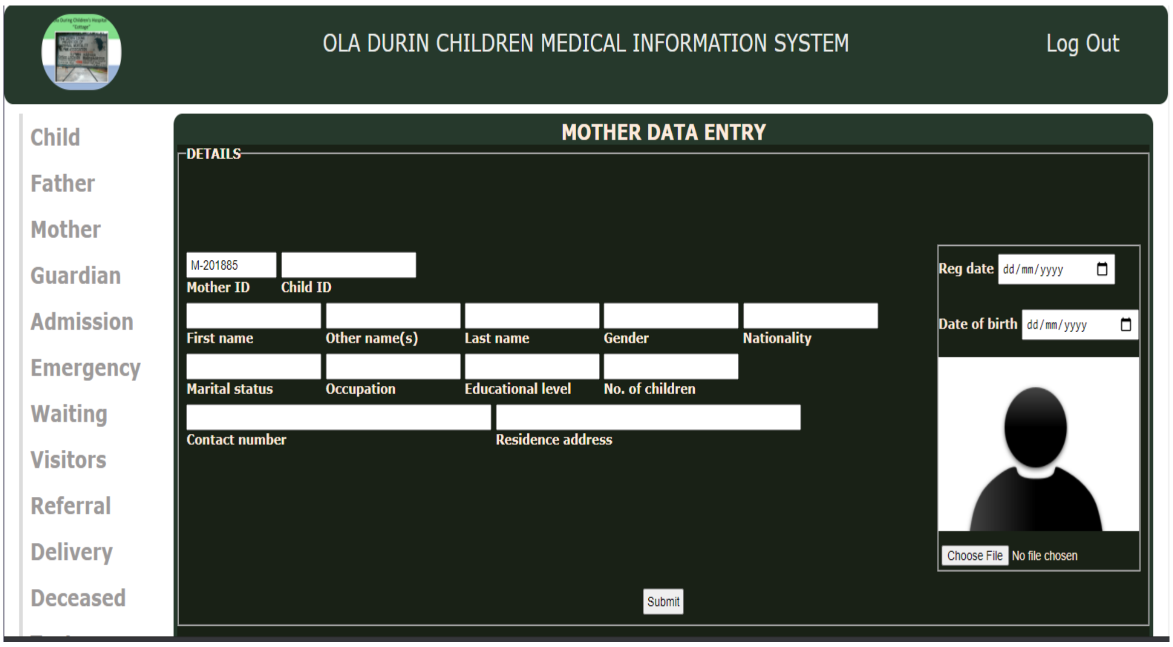
Marital status | Occupation | Educational level | No. of children

Contact number | Residence address

Reg date dd/mm/yyyy

Date of birth dd/mm/yyyy

Mother


OLA DURIN CHILDREN MEDICAL INFORMATION SYSTEM

Log Out

Child
Father
Mother
Guardian
Admission
Emergency
Waiting
Visitors
Referral
Delivery
Deceased

MOTHER DATA ENTRY

DETAILS

M-201885

Mother ID | Child ID

First name | Other name(s) | Last name | Gender | Nationality

Marital status | Occupation | Educational level | No. of children

Contact number | Residence address

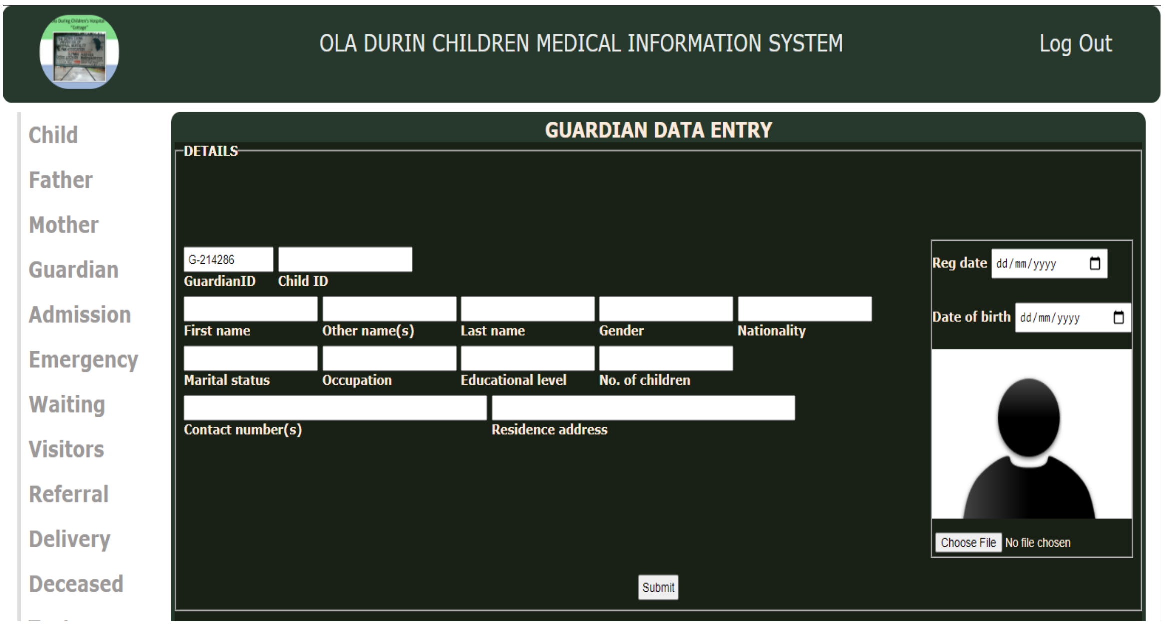
Reg date dd/mm/yyyy

Date of birth dd/mm/yyyy

Choose File No file chosen

Submit

Guardian


OLA DURIN CHILDREN MEDICAL INFORMATION SYSTEM

Log Out

Child
Father
Mother
Guardian
Admission
Emergency
Waiting
Visitors
Referral
Delivery
Deceased

GUARDIAN DATA ENTRY

DETAILS

G-214286

GuardianID | Child ID

First name | Other name(s) | Last name | Gender | Nationality

Marital status | Occupation | Educational level | No. of children

Contact number(s) | Residence address

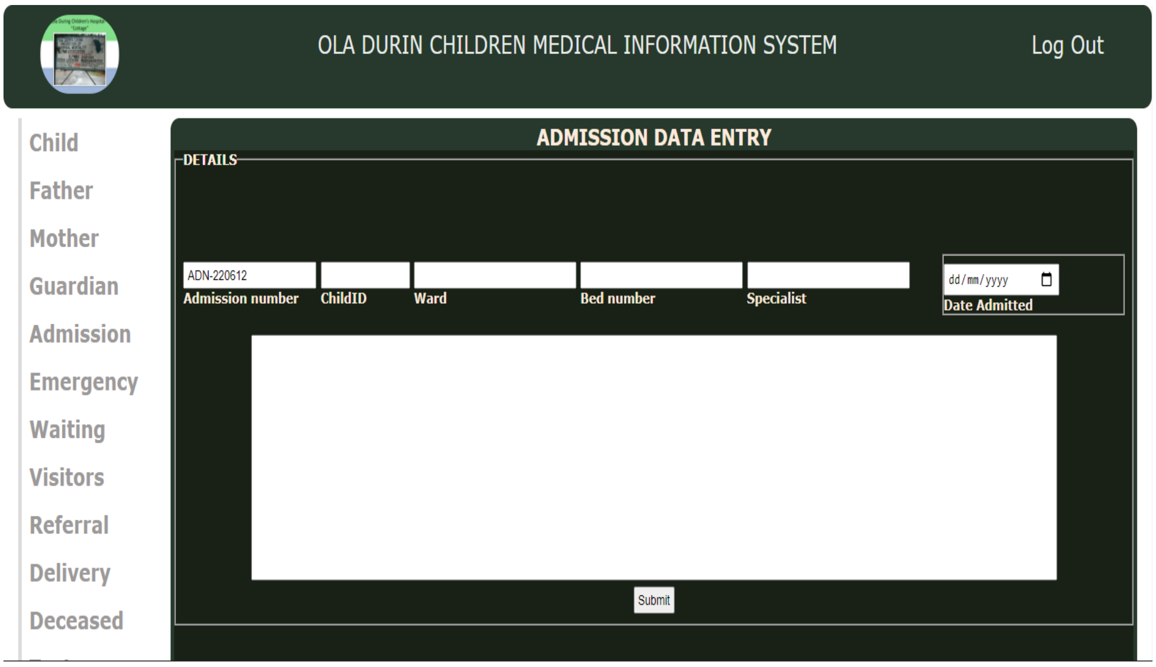
Reg date dd/mm/yyyy

Date of birth dd/mm/yyyy

Choose File No file chosen

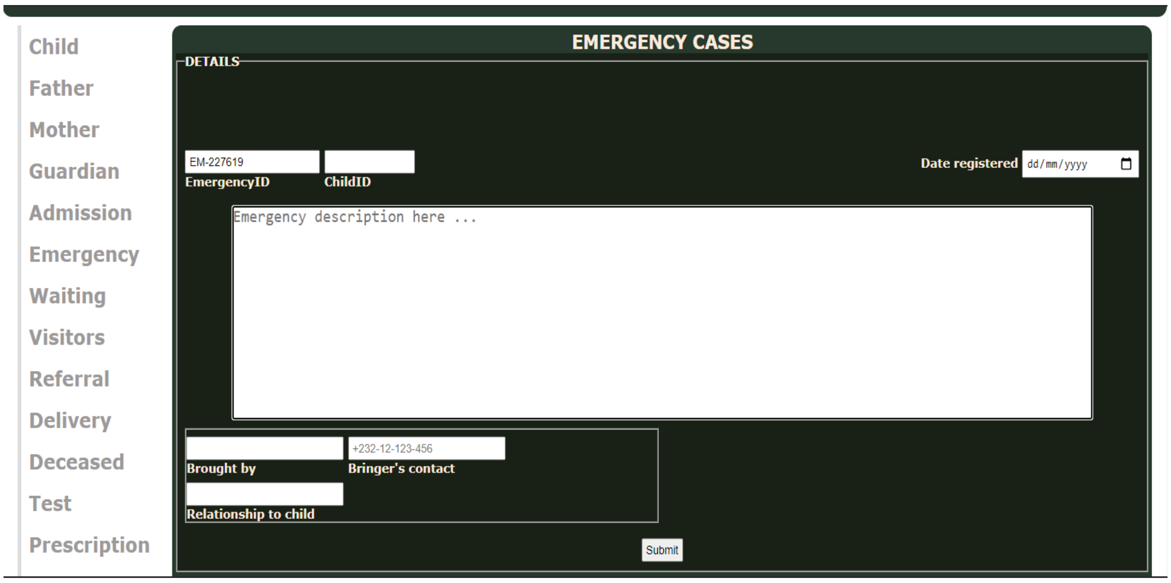
Submit

Admission



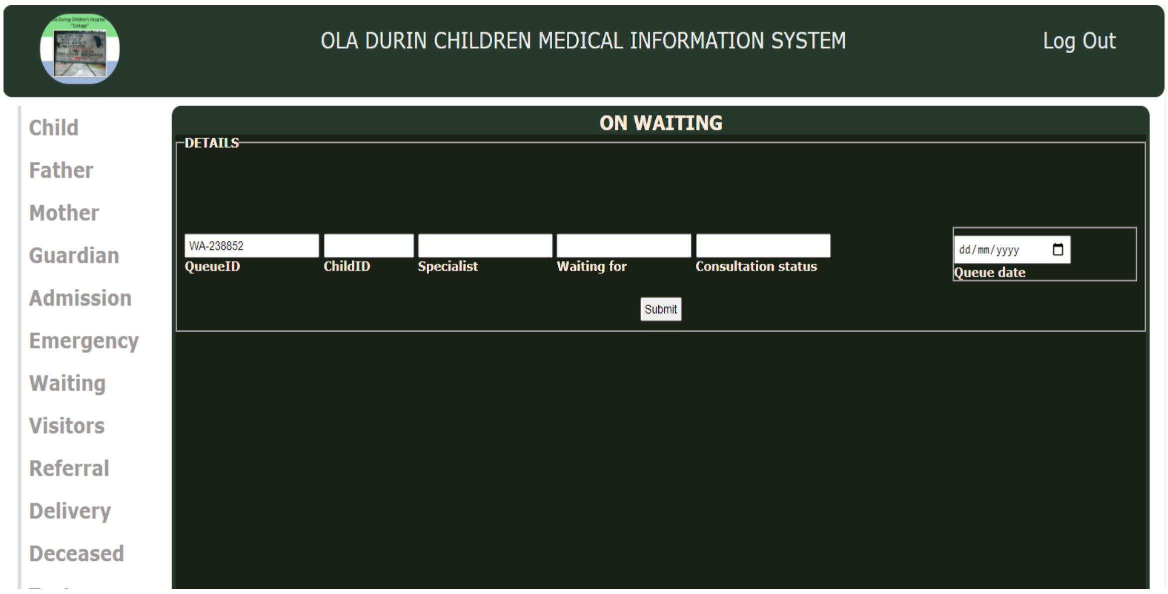
Admission number	ChildID	Ward	Bed number	Specialist	Date Admitted
ADN-220612					dd/mm/yyyy

Emergency



EmergencyID: EM-227619
ChildID:
Date registered: dd/mm/yyyy
Emergency description here ...
Brought by:
Bringer's contact: +232-12-123-456
Relationship to child:

Waiting


OLA DURIN CHILDREN MEDICAL INFORMATION SYSTEM

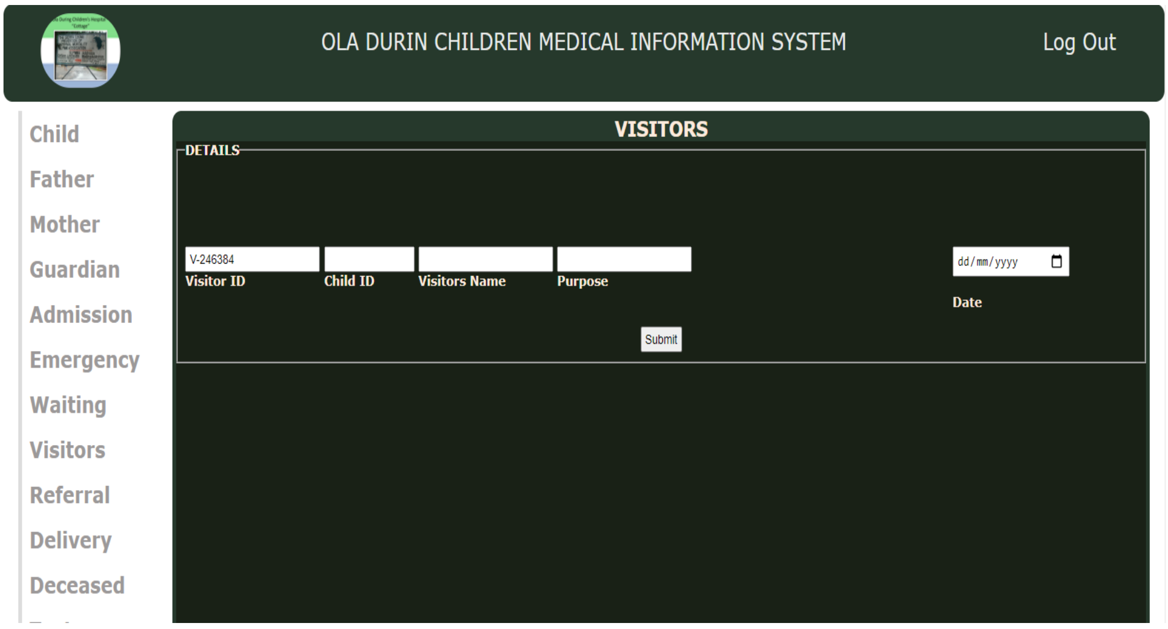
Log Out

ON WAITING

DETAILS

WA-238852					dd/mm/yyyy
QueueID	ChildID	Specialist	Waiting for	Consultation status	Queue date

Visitors


OLA DURIN CHILDREN MEDICAL INFORMATION SYSTEM

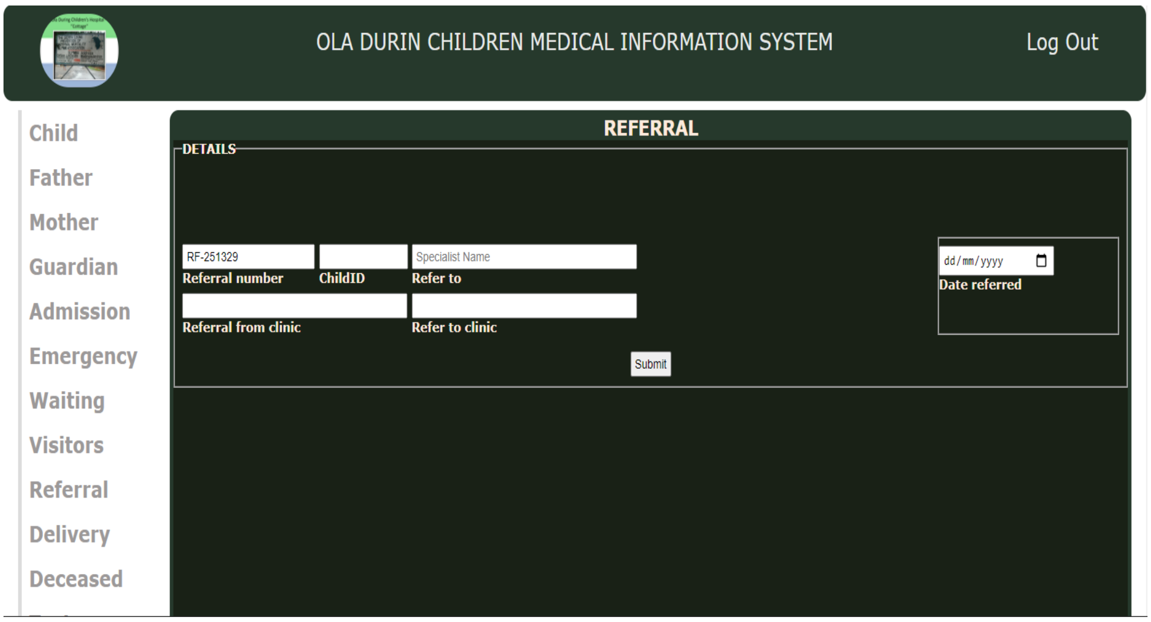
Log Out

VISITORS

DETAILS

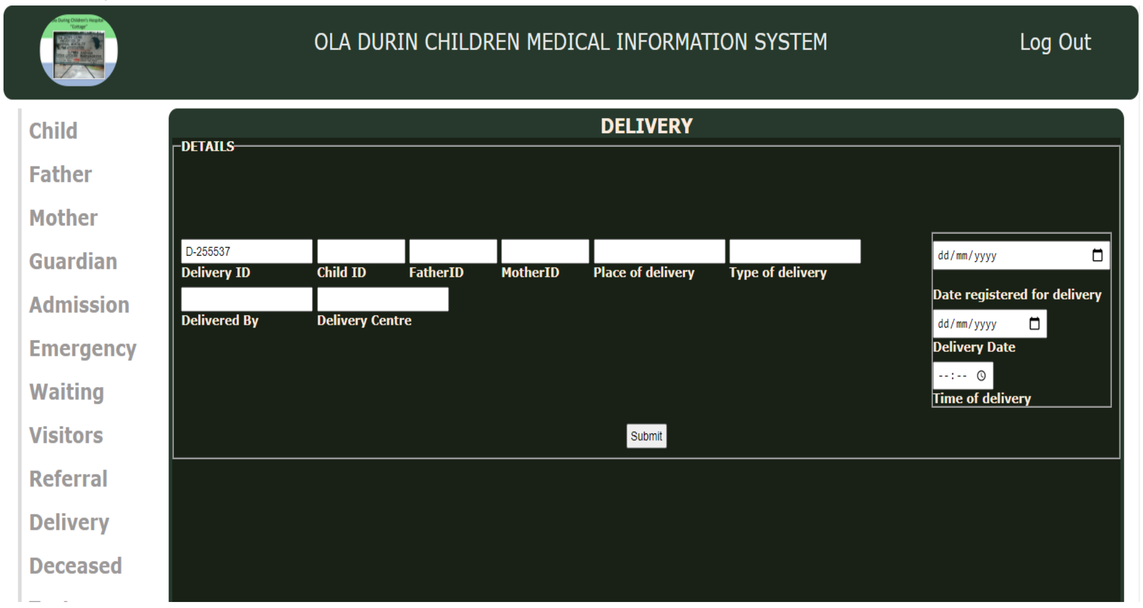
V-246384				dd/mm/yyyy
Visitor ID	Child ID	Visitors Name	Purpose	Date

Referral



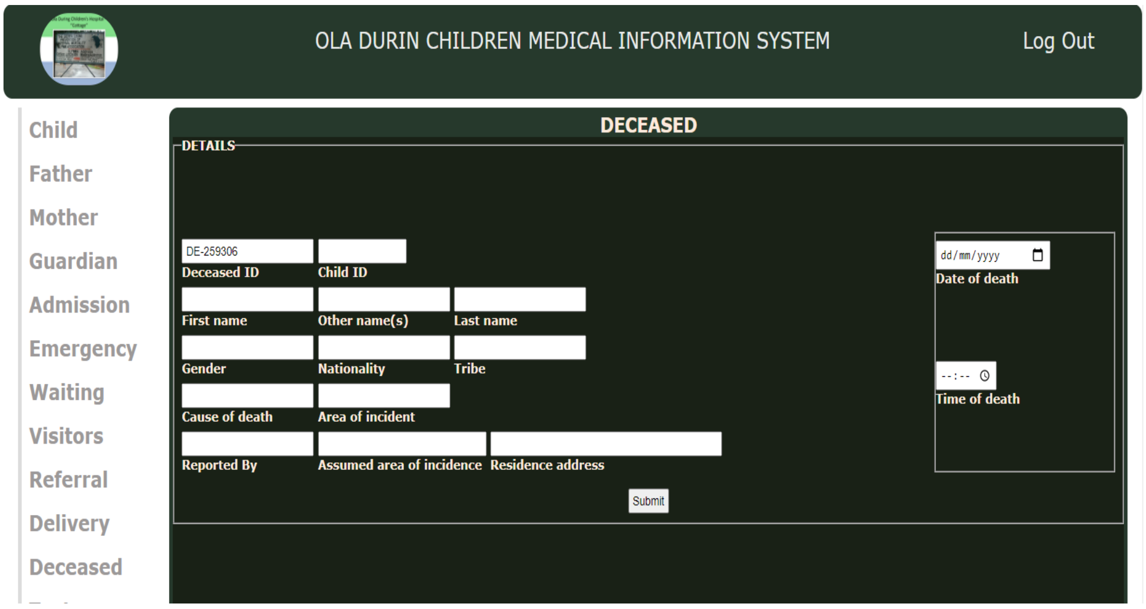
RF-251329
Referral number
ChildID
Specialist Name
Refer to
dd/mm/yyyy
Date referred
Referral from clinic
Refer to clinic

Delivery



D-255537
Delivery ID
Child ID
FatherID
MotherID
Place of delivery
Type of delivery
dd/mm/yyyy
Date registered for delivery
Delivered By
Delivery Centre
dd/mm/yyyy
Delivery Date
--:--
Time of delivery

Deceased


OLA DURIN CHILDREN MEDICAL INFORMATION SYSTEM

Log Out

Child
Father
Mother
Guardian
Admission
Emergency
Waiting
Visitors
Referral
Delivery
Deceased

DECEASED

DETAILS

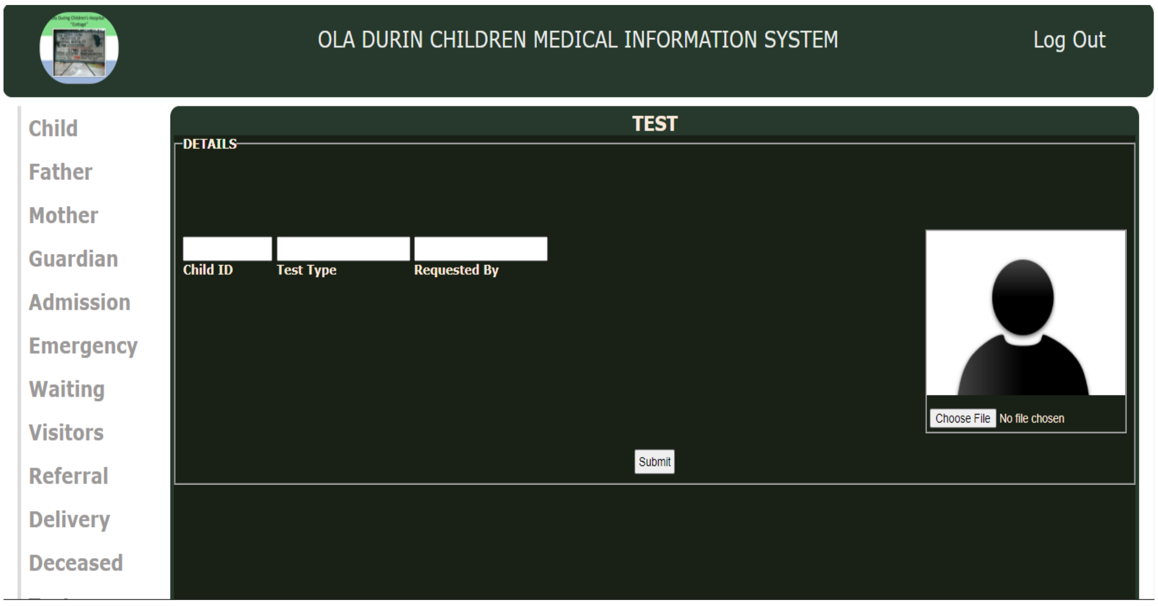
DE-259306		
Deceased ID	Child ID	
First name	Other name(s)	Last name
Gender	Nationality	Tribe
Cause of death	Area of incident	
Reported By	Assumed area of incidence	Residence address

dd/mm/yyyy
Date of death

--:--
Time of death

Submit

Test


OLA DURIN CHILDREN MEDICAL INFORMATION SYSTEM

Log Out

Child
Father
Mother
Guardian
Admission
Emergency
Waiting
Visitors
Referral
Delivery
Deceased

TEST

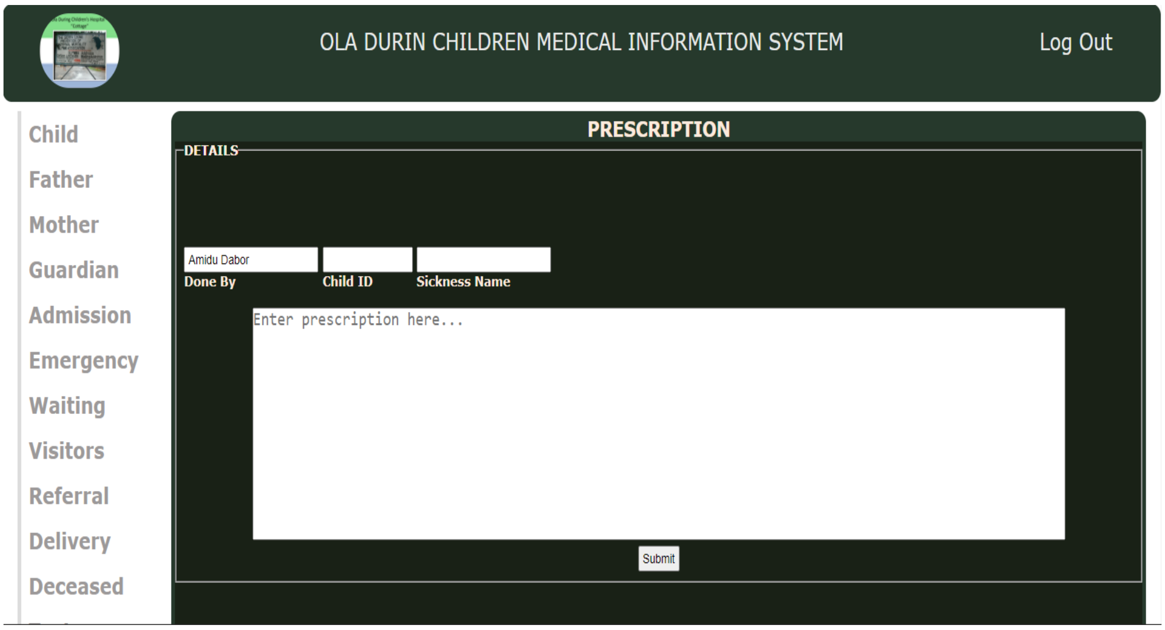
DETAILS

Child ID	Test Type	Requested By

Choose File No file chosen

Submit

Prescription



OLA DURIN CHILDREN MEDICAL INFORMATION SYSTEM

Log Out

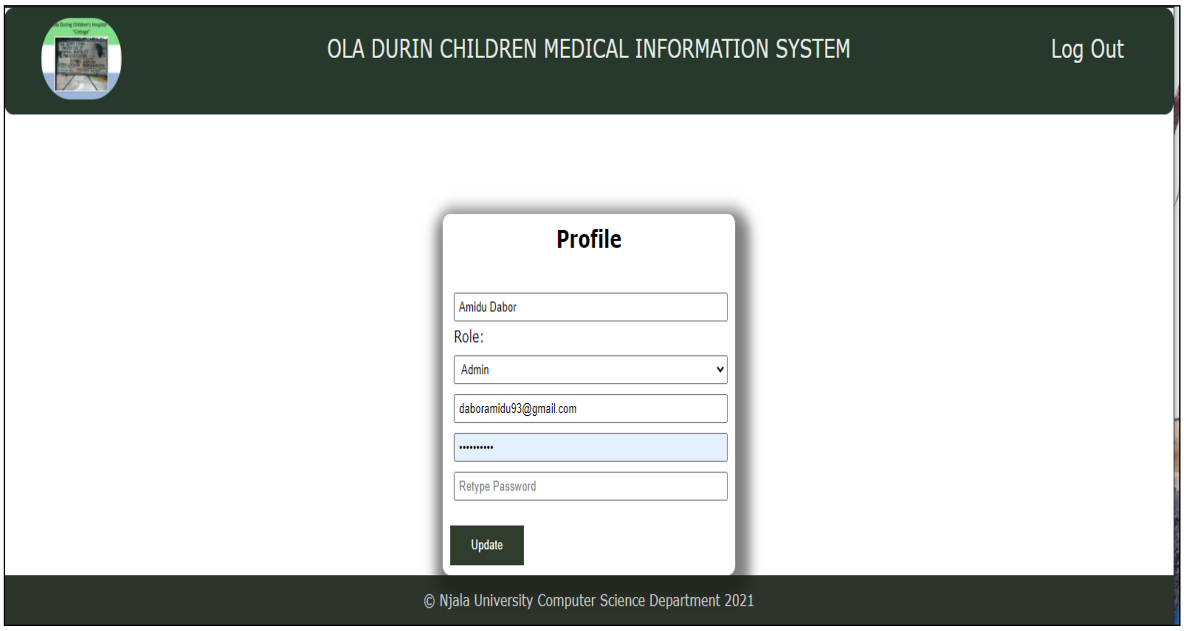
PRESCRIPTION

DETAILS

Done By	Child ID	Sickness Name
Amidu Dabor		

Enter prescription here...

User Profile



OLA DURIN CHILDREN MEDICAL INFORMATION SYSTEM

Log Out

Profile

Amidu Dabor

Role: Admin

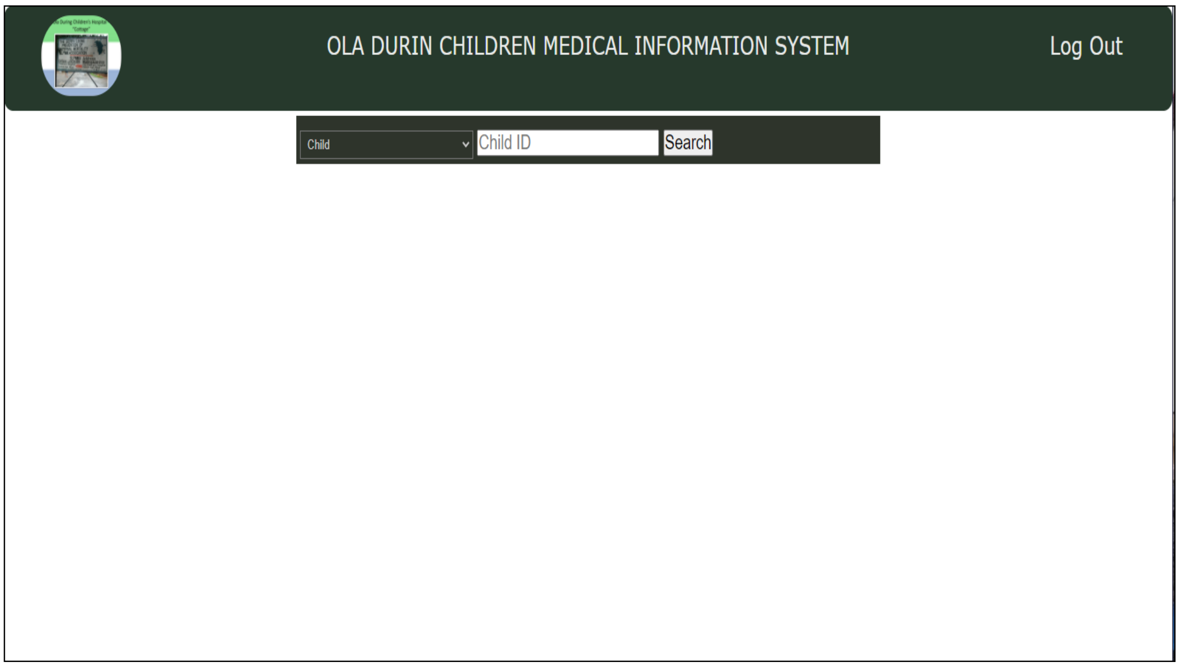
daboramidu93@gmail.com

••••••••••

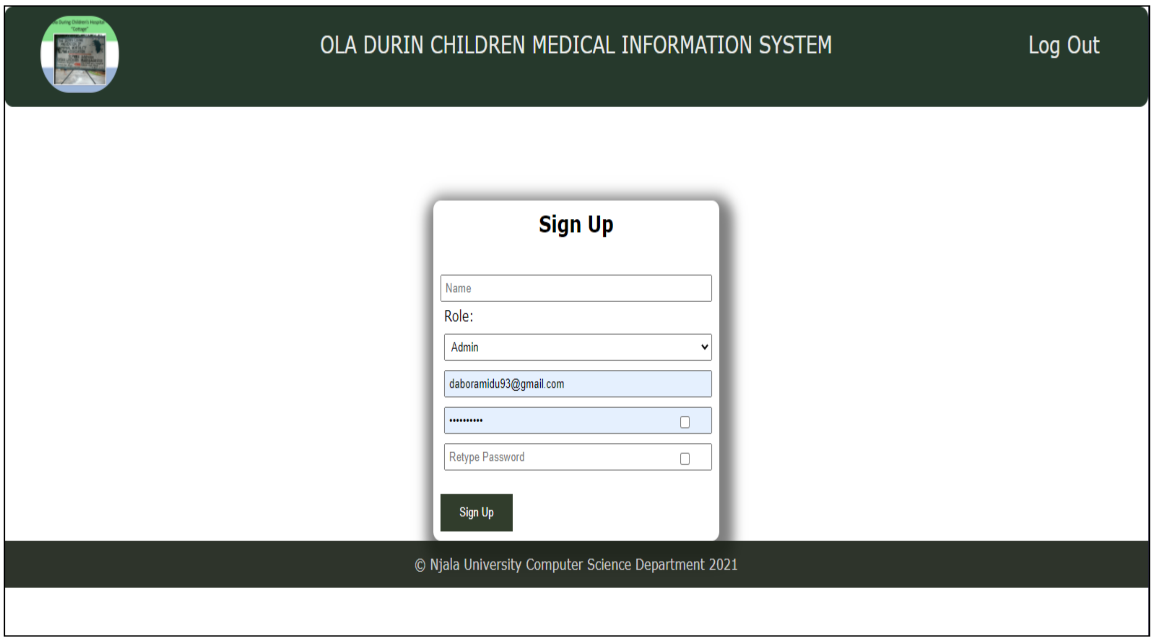
Retype Password

© Njala University Computer Science Department 2021

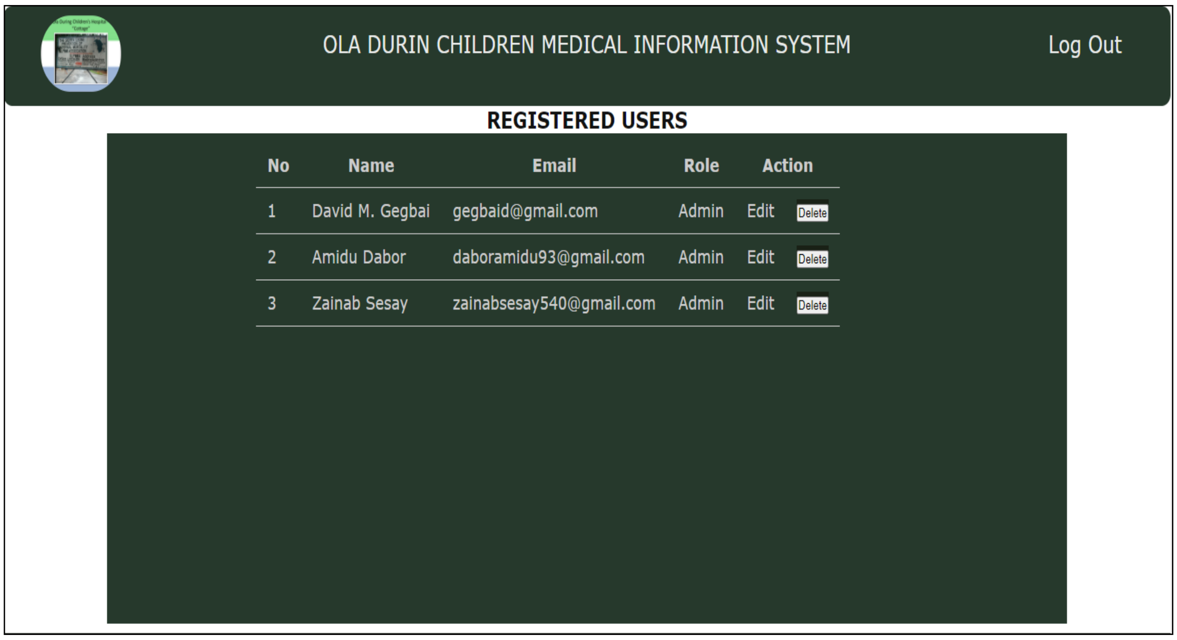
Search Form (Contains a dropdown list of form names with records to search for and display via a child's ID)



User Signup Form


OLA DURIN CHILDREN MEDICAL INFORMATION SYSTEM
Log Out
Sign Up
Name
Role:
Admin
daboramidu93@gmail.com
••••••••••
Retype Password
Sign Up
© Njala University Computer Science Department 2021

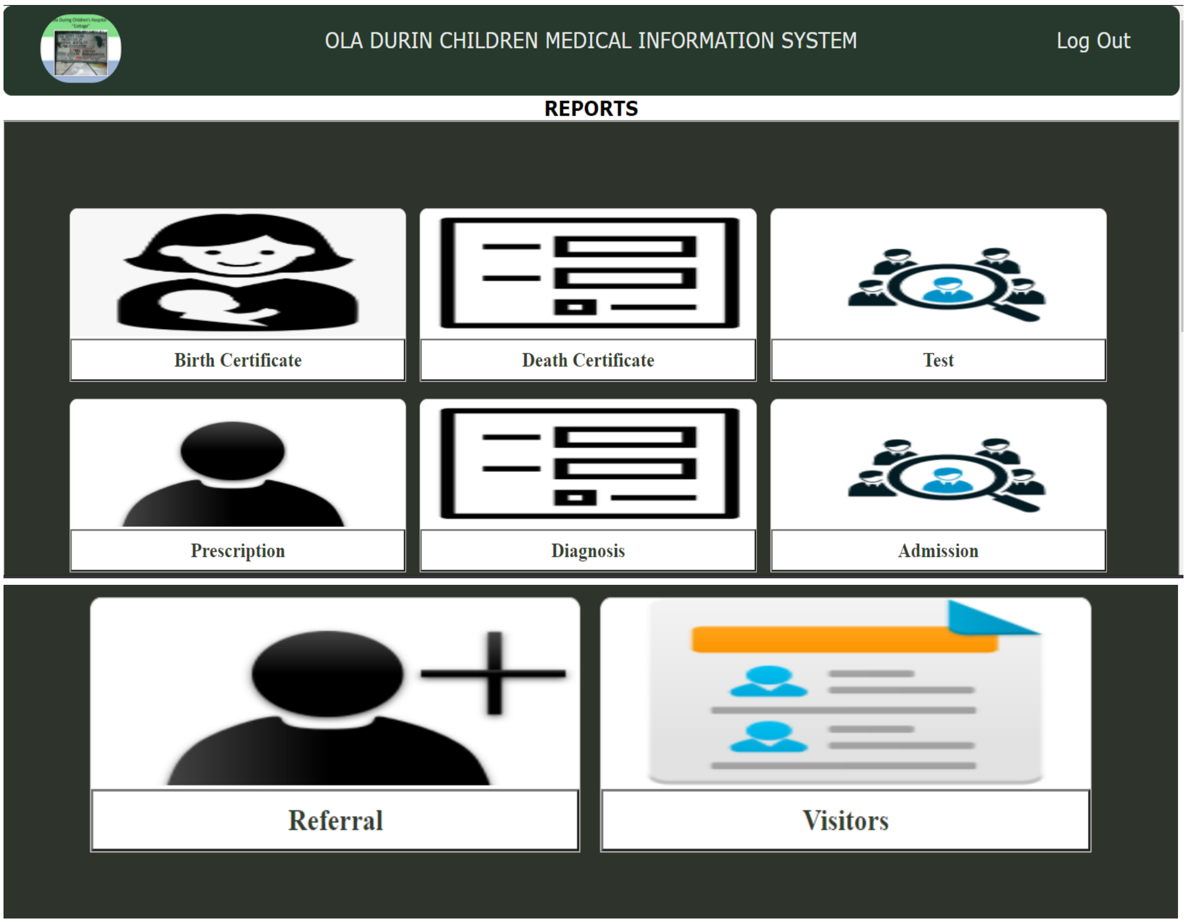
Records of Registered Users



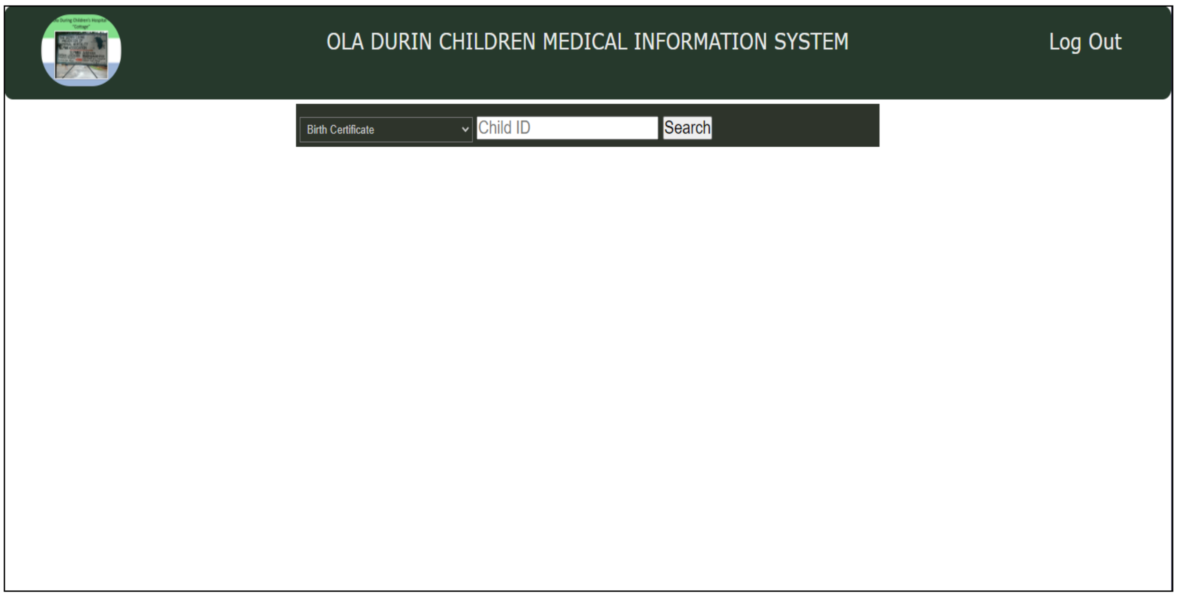
REGISTERED USERS

No	Name	Email	Role	Action	
1	David M. Gegbai	gegbaid@gmail.com	Admin	Edit	Delete
2	Amidu Dabor	daboramidu93@gmail.com	Admin	Edit	Delete
3	Zainab Sesay	zainabsesay540@gmail.com	Admin	Edit	Delete

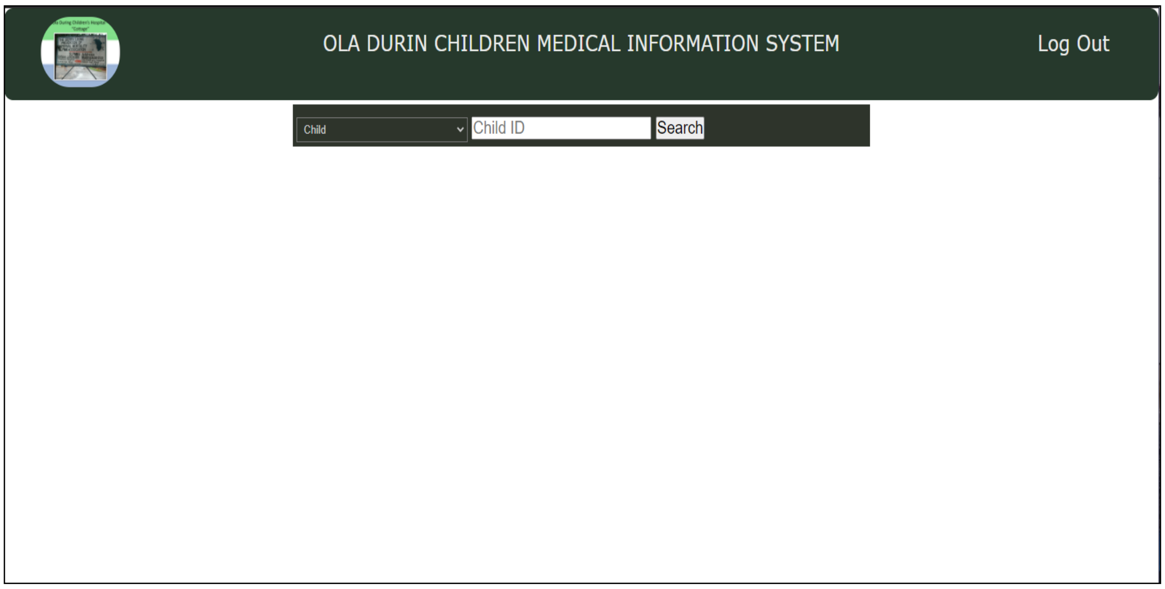
Reports



Reports (Contains a dropdown list of reports with records to make queries from via ID’s)



Deleting Records (Contains a dropdown list of forms with records to search for and delete via ID’s)



Mail Sending Credentials and Service

```
const transporter = nodemailer.createTransport({
    service : "gmail",
    auth: {
        type : "login",
        user : "childrensmedicalrecords@gmail.com",
        pass : "oladurin"
    }
});
```

//HANDLING FILES UPLOAD AND DOWNLOAD

```
const URI = process.env.MONGODB;
global.Promise = mongoose.Promise
const conn = mongoose.createConnection(URI, {
    useUnifiedTopology : true,
    useNewUrlParser : true,
    useFindAndModify: false
});
```

//GRIDFS CONFIG FOR IMAGES

```
let gfs;
conn.once('open', () => {
    gfs = Grid(conn.db, mongoose.mongo);
    gfs.collection("files");
});
```

//GRIDFS STORAGE CONFIG

```
const storage = new GridFsStorage({
    url: URI,
    options : {useUnifiedTopology : true},
    file: (req, file) => {
        return new Promise((resolve, reject) => {
            crypto.randomBytes(16, (err, buf) => {
            if (err) {
                return reject(err);
            }
            const filename = buf.toString('hex') + path.extname(file.originalname);
            const fileInfo = {
                filename: filename,
                bucketName: "files"
            };
            resolve(fileInfo);
            });
        });
    }
});
```

//MULTER CONFIG FOR IMAGES

```
const files = multer({ storage });
```

//Login Logic

```
router.post("/login", (req, res, next) => {
    passport.authenticate("local", {
        successRedirect : "/",
        successFlash : true,
        failureRedirect : "/login",
        failureFlash : true
    })(req, res, next);
});

//Logout Route
router.get("/logout", isLoggedIn, (req, res) => {
    req.logout();
    req.flash("success", "Logged out Successfully"),
    res.redirect("/login");
});



//==========================================================
=============================
//ROUTES
//Get the dashboard template
router.get("/", isLoggedIn, (req, res) => {
    res.render("dashboard", {
        title : "OLA DURIN CHILDREN'S MEDICAL RECORDS",
        description: "Ola Durin Children's Medical Records System"
    });
```

```
});

//============================================================
==============================
//STAFF ROUTES
//Registration/SignUp Route Logic
router.post("/register", (req, res) => {
    if(req.body.password === req.body.rePassword){
        bcrypt.genSalt(10)
        .then(salt => {
            bcrypt.hash(req.body.password, salt)//Encrypting the password
            .then(hash => {
                User.create({
                    name : req.body.name,
                    email : req.body.email,
                    password : hash,
                    role : req.body.role
                })
                .then(user => {
                    if(user){
                        if(req.body.role === "Admin"){
                            const mailOptions = {
                                from : "user@gmail.com",
                                to : req.body.email,
                                subject : "Login Information",
                    html : `<p>Dear ${req.body.name},</p> <p>Hope this mail finds
you well.</p><p>You have been successfully registered into the Ola Durin Childrens
Medical Record System.</p><p>Here are your login
credentials.</p><h3>Credentials:</h3><p>Email:
<strong>${req.body.email}</strong></p><p>Password:
<strong>${req.body.password}</strong></p><p>You have been registered as an
Admin, meaning you have full rights to manage the
system</p><p>Regards</p>Management</p>`
                }
                transporter.sendMail(mailOptions)
                .then(mail => {
                    if(mail){
                        req.flash("success", "LOGIN INFORMATION SENT
SUCCESSFULLY");
                    }else{
                        req.flash("error", "LOGIN INFORMATION NOT SENT");
                    }
                })
            }else{
                const mailOptions = {
                    from : "user@gmail.com",
                    to : req.body.email,
                    subject : "Login Information",
                    html : `<p>Dear ${req.body.name},</p> <p>Hope this mail finds
you well.</p><p>You have been successfully registered into the Ola Durin Childrens
```

```
Medical Record System.</p><p>Here are your login
credentials.</p><h3>Credentials:</h3><p>Email:
<strong>${req.body.email}</strong></p><p>Password:
<strong>${req.body.password}</strong></p><p>You have been registered as an
Matron, meaning you can only add data and information into the
system</p><p>Regards</p>Management</p>`
              }
              transporter.sendMail(mailOptions)
              .then(mail => {
                if(mail){
                  req.flash("success", "LOGIN INFORMATION SENT
SUCCESSFULLY");
                }else{
                  req.flash("error", "LOGIN INFORMATION NOT SENT");
                }
              })
            }
            console.log(user);
            req.flash("success", "ACCOUNT CREATED SUCCESSFULLY");
            res.redirect("back");
          }
        })
        .catch(err => {
          if(err){
            console.log(err);
```

```
                        res.redirect("back");
                    }
                })
            })
        })
    }
});
```

//Profile Update Logic Route

```
router.put("/profile/:id", (req, res) => {
    if((req.body.password !== "") && (req.body.password ===
req.body.rePassword)){
        bcrypt.genSalt(10)
        .then(salt => {
            bcrypt.hash(req.body.password, salt)
            .then(hash => {
                User.findOneAndUpdate({_id : req.params.id}, {
                    name : req.body.name,
                    email : req.body.email,
                    password : hash
                })
                .then(updated => {
                    if(updated){
                        req.flash("success", "PROFILE UPDATED SUCCESSFULLY");
                        res.redirect("back");
                    }
                })
                .catch(err => {
                    if(err){
                        console.log(err)
                        res.redirect("back");
                    }
                });
            })
        })

    }else{
        User.findOneAndUpdate({_id : req.params.id}, {
            name : req.body.name,
            email : req.body.email
        })
        .then(updated => {
            if(updated){
                console.log("PROFILE UPDATED SUCCESSFULLY");
                res.redirect("back");
            }
        })
        .catch(err => {
            if(err){
                console.log(err)
```

```
                res.redirect("back");
            }
        });
    }

});
```

//Admin Account Delete Route

```
router.delete("/user/:id", (req, res) => {
    User.findOneAndDelete({_id : req.params.id})
    .then(deletedUser => {
        if(deletedUser){
            req.flash("success", "USER ACCOUNT DELETED SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });

});
```

//Search form logic

```
router.post("/deleteSearch", (req, res) => {
    if(req.body.options === "Child"){
        Child.findOne({childID : req.body.search})
        .then(child => {
            if(child){
                res.render("deleteChild", {
                    title : "Deleting child record",
                    description : "Deleting a single child record",
                    child : child
                });
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("back");
            }
        });
    }else if(req.body.options === "Mother"){
        Mother.findOne({childId : req.body.search})
        .then(mother => {
            if(mother){
                res.render("deleteMother", {
                    title : "Deleting mothers record",
                    description : "Deleting a single mothers record",
                    mother : mother
                });
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("err");
            }
        });
    }else if(req.body.options === "Father"){
        Father.findOne({childId : req.body.search})
        .then(father => {
            if(father){
                res.render("deleteFather", {
                    title : "Deleting father record",
                    description : "Deleting a single father record",
                    father : father
                });
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("err");
            }
        });
    }else if(req.body.options === "Guardian"){
        Guardian.findOne({childId : req.body.search})
        .then(guardian => {
            if(guardian){
                res.render("deleteGuardian", {
                    title : "Deleting guardian record",
                    description : "Deleting a single guardian record",
                    guardian : guardian
                });
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("err");
            }
        });
    }else if(req.body.options === "Admission"){
        Admission.find({childID : req.body.search})
        .then(admissions => {
            if(admissions){
                res.render("deleteAdmissions", {
                    title : "Showing childs admissions",
                    description : "Deleting a single child admission record",
                    admissions : admissions
                });
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("err");
            }
        });
    }else if(req.body.options === "Emergency"){
        Emergency.find({childID : req.body.search})
        .then(emergencies => {
            if(emergencies){
                res.render("deleteEmergencies", {
                    title : "Showing childs emergency records",
                    description : "Deleting a single child emergency record",
                    emergencies : emergencies
                });
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("err");
            }
        });
    }else if(req.body.options === "Waiting"){
        Waiting.find({childID : req.body.search})
        .then(waitings => {
            if(waitings){
                res.render("deleteWaitings", {
                    title : "Showing childs waiting records",
                    description : "Deleting a single child waiting record",
                    waitings : waitings
                });
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("err");
            }
        });
    }else if(req.body.options === "Visitors"){
        Visitors.find({childID : req.body.search})
        .then(visitors => {
            if(visitors){
                res.render("deleteVisitor", {
                    title : "Showing childs visitor records",
                    description : "Deleting a single child visitor record",
                    visitors : visitors
                });
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("err");
            }
        });
    }else if(req.body.options === "Referral"){
        Referral.find({childID : req.body.search})
        .then(referrals => {
            if(referrals){
                res.render("deleteReferral", {
                    title : "Showing childs referral records",
                    description : "Deleting a single child referral record",
                    referrals : referrals
                });
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("err");
            }
        });
    }else if(req.body.options === "Test"){
        Test.find({childID : req.body.search})
        .then(tests => {
            if(tests){
                res.render("deleteTests", {
                    title : "Showing childs test records",
                    description : "Deleting a single child test record",
                    tests : tests
                });
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("err");
            }
        });
    }else if(req.body.options === "Prescription"){
        Prescription.find({childID : req.body.search})
        .then(prescriptions => {
            if(prescriptions){
                res.render("deletePrescription", {
                    title : "Showing childs prescription records",
                    description : "Deleting a single child prescription record",
                    prescriptions : prescriptions
                });
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("err");
            }
        });
    }else if(req.body.options === "Diagnosis"){
        Diagnosis.find({childID : req.body.search})
        .then(diagnosis => {
            if(diagnosis){
                res.render("deleteDiagnosis", {
                    title : "Showing childs diagnosis records",
                    description : "Deleting a single child diagnosis record",
                    diagnosis : diagnosis
                });
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("err");
            }
        });
    }else if(req.body.options === "Delivery"){
        Delivery.findOne({childID : req.body.search})
        .then(delivery => {
            if(delivery){
                res.render("deleteDelivery", {
                    title : "Showing childs delivery record",
                    description : "Deleting a single child delivery record",
                    delivery : delivery
                });
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("err");
            }
        });
    }else if(req.body.options === "Deceased"){
        Deceased.findOne({childId : req.body.search})
```

```
        .then(deceased => {
            if(deceased){
                res.render("deleteDeceased", {
                    title : "Showing childs deceased records",
                    description : "Deleting a single child deceased record",
                    deceased : deceased
                });
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("err");
            }
        });
    }
});
```

// CHILD CRUD FUNCTIONALITY

//Adding record in the child document

```
router.post("/child/add", files.single("photo"), (req, res) => {
    Child.findOne({childID : req.body.childID})
    .then(foundChild => {
        if(foundChild){
            req.flash("error", "ID ALREADY EXIST, PLEASE USE ANOTHER");
            res.redirect("back");
        }else{
            if(req.file !== "" && (req.file.mimetype === "image/png" || req.file.mimetype
=== "image/jpg" || req.file.mimetype === "image/jpeg")){
                Child.create({
                    childID : req.body.childID,
                    firstName : req.body.firstName,
                    lastName : req.body.lastName,
                    otherName : req.body.otherName,
                    dob : req.body.dob,
                    address : req.body.address,
                    photo : req.file.filename,
                    ethnicity : req.body.ethnicity,
                    religion : req.body.religion,
                    registrationType : req.body.registrationType,
                    regDate : req.body.regDate,
                    gender : req.body.gender,
                    nationality : req.body.nationality
                })
                .then(child => {
                    if(child){
                        req.flash("success", "CHILD ADDED SUCCESSFULLY");
                        res.redirect("back");
                    }
                })
                .catch(err => {
                    if(err){
                        console.log(err);
                        res.redirect("back");
                    }
                });

            }else{
                req.flash("error", "FILES MUST BE EITHER .JPG OR .PNG");
                res.redirect("back");
            }

        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    })

});
```

//Getting one child information

```
router.get("/child/:id", isLoggedIn, (req, res) => {
    Child.findById({_id : req.params.id})
    .then(child => {
        if(child){
            res.render("viewChild", {
                title : "Child Information",
                description : "Getting the information of a single child",
                child : child
            });
        }else{
            req.flash("error", "CHILD ID NOT IN SYSTEM");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "NO RECORD FOUND");
            res.redirect("back");
        }
    })
});
```

//Updating the child's information

```
router.put("/child/:id/edit", files.single("photo"), (req, res) => {
    if(req.file !== undefined && (req.file.mimetype === "image/png" ||
req.file.mimetype === "image/jpg" || req.file.mimetype === "image/jpeg")){
    Child.findOneAndUpdate({_id : req.params.id}, {
        childCode : req.body.childCode,
        childID : req.body.childID,
        firstName : req.body.firstName,
        lastName : req.body.lastName,
        otherName : req.body.otherName,
        dob : req.body.dob,
        address : req.body.address,
        photo : req.file.filename,
        ethnicity : req.body.ethnicity,
        religion : req.body.religion,
        registrationType : req.body.registrationType,
        regDate : req.body.regDate,
        gender : req.body.gender,
        nationality : req.body.nationality
    })
    .then(child => {
        if(child){
            req.flash("success", "CHILD INFORMATION UPDATED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });
  }else{
    Child.findOneAndUpdate({_id : req.params.id}, {
        childCode : req.body.childCode,
        childID : req.body.childID,
        firstName : req.body.firstName,
        lastName : req.body.lastName,
        otherName : req.body.otherName,
        dob : req.body.dob,
        address : req.body.address,
        ethnicity : req.body.ethnicity,
        religion : req.body.religion,
        registrationType : req.body.registrationType,
        regDate : req.body.regDate,
        gender : req.body.gender,
        nationality : req.body.nationality
    })
    .then(child => {
        if(child){
            req.flash("success", "CHILD INFORMATION UPDATED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });
  }

});
```

//Deleting child information

```
router.delete("/child/:id", (req, res) => {
    Child.findByIdAndRemove({_id : req.params.id})
    .then(child => {
        if(child){
            gfs.files.findOne({filename : child.photo}, (err, file) => {
                if(file){
                    gfs.files.remove({filename : file.filename, root : "files"}, (err) => {
                        if(err){
                            console.log(err);
                        }else{
                            req.flash("success", "CHILD INFORMATION DELETED
SUCCESSFULLY")
```

```
                    res.redirect("back");
                  }
                });
              }else{
                console.log(err);
              }
            });
          }
        })
      });
```

//FATHER CRUD FUNCTIONALITY

// Adding record in the father document

```
router.post("/father/add", files.single("photo"), (req, res) => {
  if(req.file !== "" && (req.file.mimetype === "image/png" || req.file.mimetype ===
"image/jpg" || req.file.mimetype === "image/jpeg")){
    if(req.body.childID !== ""){
      Child.findOne({childID : req.body.childID})
      .then(child => {
        if(child){
          Father.create({
            fatherID : req.body.fatherID,
            firstName : req.body.firstName,
            lastName : req.body.lastName,
            otherName : req.body.otherName,
            gender : req.body.gender,
            dob : req.body.dob,
            address : req.body.address,
            contact : req.body.contact,
            nationality : req.body.nationality,
            placeOfBirth : req.body.pob,
            maritalStatus : req.body.maritalStatus,
            photo : req.file.filename,
            childId : req.body.childID,
            occupation : req.body.occupation,
            educationalLevel : req.body.educationalLevel,
            noOfChildren : req.body.noOfChildren,
            regDate : req.body.regDate
          })
          .then(father => {
            if(father){
                child.father = father._id;
                child.save();
                req.flash("success", "FATHER INFO SAVED SUCCESSFULLY");
                res.redirect("back");
            }
          })
          .catch(err => {
            if(err){
                console.log(err);
```

```
                    res.redirect("back");
                }
            });
        }else{
            req.flash("error", "CHILD ID IS NOT IN THE SYSTEM");
            res.redirect("back");
        }
    })
}else{
    Father.create({
        fatherID : req.body.fatherID,
        firstName : req.body.firstName,
        lastName : req.body.lastName,
        otherName : req.body.otherName,
        gender : req.body.gender,
        dob : req.body.dob,
        address : req.body.address,
        contact : req.body.contact,
        nationality : req.body.nationality,
        placeOfBirth : req.body.pob,
        maritalStatus : req.body.maritalStatus,
        photo : req.file.filename,
        occupation : req.body.occupation,
        educationalLevel : req.body.educationalLevel,
        noOfChildren : req.body.noOfChildren,
            regDate : req.body.regDate
         })
         .then(father => {
            if(father){
               req.flash("success", "FATHER INFO SAVED SUCCESSFULLY");
               res.redirect("back");
            }
         })
         .catch(err => {
            if(err){
               console.log(err);
               res.redirect("back");
            }
         });
      }
   }else{
      req.flash("error", "FATHER'S PHOTO IS MANDATORY AND MUST BE
EITHER JPG OR PNG");
      res.redirect("back");
   }
});
```

//Getting one father information

```
router.get("/father/:id", isLoggedIn, (req, res) => {
   Father.findOne({_id : req.params.id})
```

```
    .then(father => {
        if(father){
            res.render("viewFather", {
                title : "Fathers Information",
                description : "Showing fathers information",
                father : father
            });
        }else{
            req.flash("error", "CHILD ID NOT IN SYSTEM");
            res.redirect("back");
        }

    })
    .catch(err => {
        if(err){
            req.flash("error", "NO RECORD FOUND");
            res.redirect("back");
        }
    });

});
```

// Updating record in the father document

```
router.put("/father/:id/edit", files.single("photo"), (req, res) => {
    const type = req.file
    if(type !== undefined && (req.file.mimetype === "image/png" || req.file.mimetype
=== "image/jpg" || req.file.mimetype === "image/jpeg")){
        Father.findOneAndUpdate({_id : req.params.id}, {
            fatherID : req.body.fatherID,
            firstName : req.body.firstName,
            lastName : req.body.lastName,
            otherName : req.body.otherName,
            gender : req.body.gender,
            dob : req.body.dob,
            address : req.body.address,
            contact : req.body.contact,
            nationality : req.body.nationality,
            placeOfBirth : req.body.pob,
            maritalStatus : req.body.maritalStatus,
            photo : req.file.filename,
            childId : req.body.childID,
            occupation : req.body.occupation,
            educationalLevel : req.body.educationalLevel,
            noOfChildren : req.body.noOfChildren,
            regDate : req.body.regDate
        })
        .then(father => {
            if(father){
                req.flash("success", "FATHER INFO UPDATED SUCCESSFULLY");
                res.redirect("back");
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("back");
            }
        });

    }else{
        Father.findOneAndUpdate({_id : req.params.id}, {
            fatherID : req.body.fatherID,
            firstName : req.body.firstName,
            lastName : req.body.lastName,
            otherName : req.body.otherName,
            gender : req.body.gender,
            dob : req.body.dob,
            address : req.body.address,
            contact : req.body.contact,
            nationality : req.body.nationality,
            placeOfBirth : req.body.pob,
            maritalStatus : req.body.maritalStatus,
            childId : req.body.childID,
            occupation : req.body.occupation,
            educationalLevel : req.body.educationalLevel,
            noOfChildren : req.body.noOfChildren,
            regDate : req.body.regDate
        })
        .then(father => {
            if(father){
                req.flash("success", "FATHER INFO UPDATED SUCCESSFULLY");
                res.redirect("back");
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("back");
            }
        });
    }

});
```

// Deleting father information

```
router.delete("/father/:id", (req, res) => {
    Father.findById({_id : req.params.id})
    .then(father => {
        if(father){
            gfs.files.findOne({filename : father.photo}, (err, file) => {
                if(file){
                    gfs.files.deleteOne({filename : file.filename}, (err) => {
                        if(err){
                            console.log(err);
                        }else{
                            Father.findOneAndRemove({_id : req.params.id})
                            .then(deleted => {
                                if(deleted){
                                    req.flash("success", "FATHER INFORMATION DELETED
SUCCESSFULLY");
                                    res.redirect("back");
                                }
                            })
                        }
                    });
                }else{
                    console.log(err);
                }
            });
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });


});
```

//MOTHER CRUD FUNCTIONALITY

//Adding record in the mother document

```
router.post("/mother/add", files.single("photo"), (req, res) => {
    if(req.file !== "" && (req.file.mimetype === "image/png" || req.file.mimetype ===
"image/jpg" || req.file.mimetype === "image/jpeg")){
        if(req.body.childID !== ""){
            Child.findOne({childID : req.body.childID})
            .then(child => {
                if(child){
                    Mother.create({
                        motherID : req.body.motherID,
                        firstName : req.body.firstName,
                        lastName : req.body.lastName,
                        otherName : req.body.otherName,
                        gender : req.body.gender,
                        dob : req.body.dob,
                        address : req.body.address,
                        contact : req.body.contact,
                        nationality : req.body.nationality,
                        placeOfBirth : req.body.pob,
                        maritalStatus : req.body.maritalStatus,
                        photo : req.file.filename,
                        childId : req.body.childID,
                        occupation : req.body.occupation,
                        educationalLevel : req.body.educationalLevel,
                        noOfChildren : req.body.noOfChildren,
                        regDate : req.body.regDate
                    })
                    .then(mother => {
                        if(mother){
                            child.mother = mother._id;
                            child.save();
                            req.flash("success", "MOTHER INFORMATION ADDED
SUCCESSFULLY");
                            res.redirect("back");
                        }
                    })
                    .catch(err => {
                        if(err){
                            console.log(err);
                            res.redirect("back");
                        }
                    });
                }else{
                    req.flash("error", "CHILD IS NOT IN THE SYSTEM");
```

```
            res.redirect("back");
        }
    })
}else{
    Mother.create({
        motherID : req.body.motherID,
        firstName : req.body.firstName,
        lastName : req.body.lastName,
        otherName : req.body.otherName,
        gender : req.body.gender,
        dob : req.body.dob,
        address : req.body.address,
        contact : req.body.contact,
        nationality : req.body.nationality,
        placeOfBirth : req.body.pob,
        maritalStatus : req.body.maritalStatus,
        photo : req.file.filename,
        occupation : req.body.occupation,
        educationalLevel : req.body.educationalLevel,
        noOfChildren : req.body.noOfChildren,
        regDate : req.body.regDate
    })
    .then(mother => {
        if(mother){
            child.mother = mother._id;
```

```
                child.save();
                console.log("MOTHER INFO ADDED SUCCESSFULLY");
                res.redirect("back");
              }
            })
            .catch(err => {
              if(err){
                console.log(err);
                res.redirect("back");
              }
            });

          }

    }else{
        console.log("IMAGES MUST BE EITHER JPG OR PNG");
        res.redirect("back");
    }

});
```

//Getting one mother information

```
router.get("/mother/:id", isLoggedIn, (req, res) => {
    Mother.findOne({_id : req.params.id})
    .then(mother => {
        if(mother){
            res.render("viewMother", {
                title : "Viewing childs mother information",
                description : "Viewing a single mothers information",
                mother : mother
            });
        }else{
            req.flash("error", "CHILD ID NOT IN SYSTEM");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "NO RECORD FOUND");
            res.redirect("back");
        }
    });

});
```

// Updating record in the mother document

```
router.put("/mother/:id/edit", files.single("photo"), (req, res) => {
    const type = req.file
    if(type !== undefined){
        if(req.file.mimetype === "image/jpg" || req.file.mimetype === "image/png"  ||
req.file.mimetype === "image/jpeg"){
```

```
Mother.findOneAndUpdate({_id : req.params.id}, {
   motherID : req.body.motherID,
   firstName : req.body.firstName,
   lastName : req.body.lastName,
   otherName : req.body.otherName,
   gender : req.body.gender,
   dob : req.body.dob,
   address : req.body.address,
   contact : req.body.contact,
   nationality : req.body.nationality,
   placeOfBirth : req.body.pob,
   maritalStatus : req.body.maritalStatus,
   photo : req.file.filename,
   childId : req.body.childID,
   occupation : req.body.occupation,
   educationalLevel : req.body.educationalLevel,
   noOfChildren : req.body.noOfChildren,
   regDate : req.body.regDate
})
.then(mother => {
   if(mother){
      req.flash("success", "MOTHER INFORMATION UPDATED
SUCCESSFULLY");
      res.redirect("back");
   }
```

```
            })
            .catch(err => {
                if(err){
                    console.log(err);
                    res.redirect("back");
                }
            });


        }
    }else{
        Mother.findOneAndUpdate({_id : req.params.id}, {
            motherID : req.body.motherID,
            firstName : req.body.firstName,
            lastName : req.body.lastName,
            otherName : req.body.otherName,
            gender : req.body.gender,
            dob : req.body.dob,
            address : req.body.address,
            contact : req.body.contact,
            nationality : req.body.nationality,
            placeOfBirth : req.body.pob,
            maritalStatus : req.body.maritalStatus,
            childId : req.body.childID,
            occupation : req.body.occupation,
            educationalLevel : req.body.educationalLevel,
            noOfChildren : req.body.noOfChildren,
            regDate : req.body.regDate
        })
        .then(mother => {
            if(mother){
                req.flash("success", "MOTHER INFORMATION UPDATED
SUCCESSFULLY");
                res.redirect("back");
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("back");
            }
        });
    }

});
```

// Deleting mother information

```
router.delete("/mother/:id", (req, res) => {
    Mother.findById({_id : req.params.id})
    .then(mother => {
        if(mother){
          gfs.files.findOne({filename : mother.photo}, (err, file) => {
            if(file){
              gfs.files.deleteOne({filename : file.filename, root : "files"}, (err) => {
                if(err){
                  console.log(err);
                }else{
                  Mother.findByIdAndDelete({_id : mother._id})
                  .then(deleted => {
                    if(deleted){
                      req.flash("success", "MOTHER INFORMATION DELETED
SUCCESSFULLY");
                      console.log("FILE DELETED");
                      res.redirect("back");


                    }
                  })
                }
              });
            }else{
              console.log(err);
            }
          });
        }
      })
      .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });
});
```

//GUARDIAN CRUD FUNCTIONALITY

//Adding record in the guardian document

```
router.post("/guardian/add", files.single("photo"), (req, res) => {
    if(req.file !== "" && (req.file.mimetype === "image/png" || req.file.mimetype ===
"image/png" || req.file.mimetype === "image/jpeg")){
        if(req.body.childID !== ""){
            Child.findOne({childID : req.body.childID})
            .then(child => {
                if(child){
                    Guardian.create({
                        guardianID : req.body.guardianID,
                        firstName : req.body.firstName,
                        lastName : req.body.lastName,
                        otherName : req.body.otherName,
                        gender : req.body.gender,
                        dob : req.body.dob,
                        address : req.body.address,
                        contact : req.body.contact,
                        nationality : req.body.nationality,
                        placeOfBirth : req.body.pob,
                        maritalStatus : req.body.maritalStatus,
                        photo : req.file.filename,
                        childId : req.body.childID,
                        occupation : req.body.occupation,
                        educationalLevel : req.body.educationalLevel,
                        noOfChildren : req.body.noOfChildren,
                        regDate : req.body.regDate
                    })
                    .then(guardian => {
                        if(guardian){
                            child.guardian = guardian._id;
                            req.flash("success", "GUARDIAN INFORMATION ADDED
SUCCESSFULLY");
                            res.redirect("back");
                        }
                    })
                    .catch(err => {
                        if(err){
                            console.log(err);
                            res.redirect("back");
                        }
                    });
```

```
            }else{
                req.flash("error", "CHILD IS NOT IN THE SYSTEM");
                res.redirect("back");
            }
        })
    }
  }else{
    req.flash("error", "IMAGES MUST BE EITHER JPG OR PNG");
    res.redirect("back");
  }

});
```

//Getting one guardian information

```
router.get("/guardian/:id", isLoggedIn, (req, res) => {
  Guardian.findOne({_id : req.params.id})
  .then(guardian => {
    if(guardian){
      res.render("viewGuardian", {
        title : "Viewing childs gaurdian information",
        description : "Viewing a single gaurdian information",
        guardian : guardian
      });
    }
    })
    .catch(err => {
        if(err){
            req.flash("error", "NO RECORD FOUND");
            res.redirect("back");
        }
    });

});
```

// Updating record in the guardian document

```
router.put("/guardian/:id/edit", files.single("photo"), (req, res) => {
    if(req.file !== undefined && (req.file.mimetype === "image/png" ||
req.file.mimetype === "image/png" || req.file.mimetype === "image/jpeg")){
        Guardian.findByIdAndUpdate({_id : req.params.id}, {
            guardianID : req.body.gaurdianID,
            firstName : req.body.firstName,
            lastName : req.body.lastName,
            otherName : req.body.otherName,
            gender : req.body.gender,
            dob : req.body.dob,
            address : req.body.address,
            contact : req.body.contact,
            nationality : req.body.nationality,
            placeOfBirth : req.body.pob,
            maritalStatus : req.body.maritalStatus,
            photo : req.file.filename,
            childId : req.body.childID,
            occupation : req.body.occupation,
            educationalLevel : req.body.educationalLevel,
            noOfChildren : req.body.noOfChildren,
            regDate : req.body.regDate
        })
        .then(guardian => {
            if(guardian){
                req.flash("success", "GUARDIAN INFO UPDATED SUCCESSFULLY");
                res.redirect("back");
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("back");
            }
        });

    }else{
        Guardian.findOneAndUpdate({_id : req.params.id}, {
            gaurdianID : req.body.gaurdianID,
            firstName : req.body.firstName,
            lastName : req.body.lastName,
            otherName : req.body.otherName,
            gender : req.body.gender,
            dob : req.body.dob,
            address : req.body.address,
            contact : req.body.contact,
            nationality : req.body.nationality,
            placeOfBirth : req.body.pob,
            maritalStatus : req.body.maritalStatus,
            childId : req.body.childID,
            occupation : req.body.occupation,
            educationalLevel : req.body.educationalLevel,
            noOfChildren : req.body.noOfChildren,
            regDate : req.body.regDate
        })
        .then(guardian => {
            if(guardian){
                req.flash("success", "GUARDIAN INFO UPDATED SUCCESSFULLY");
                res.redirect("back");
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("back");
```

```
            }
         });
     }


});
```

// Deleting guardian information

```
router.delete("/guardian/:id", (req, res) => {
    Guardian.findByIdAndDelete({_id : req.params.id})
    .then(guardian => {
        if(guardian){
            gfs.files.findOne({filename : gaurdian.photo}, (err, file) => {
                if(file){
                    gfs.files.remove({filename : file.filename, root : "files"}, (err) => {
                        if(err){
                            console.log(err);
                        }
                    });
                }else{
                    console.log(err);
                }
            });
            req.flash("success", "GUARDIAN INFORMATION DELETED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });
});
```

// END OF GUARDIAN FUNCTIONALITY

//ADMISSION CRUD FUNCTIONALITY

//Adding record in the admission document

```
router.post("/admission/add", (req, res) => {
    if(req.body.childID !== ""){
        Child.findOne({childID : req.body.childID})
        .then(child => {
            if(child){
                Admission.create({
                    admissionID : req.body.admissionID,
                    ward : req.body.ward,
                    bedNumber : req.body.bedNumber,
                    admittedDate : req.body.admittedDate,
                    childID : req.body.childID,
                    admissionDetail : req.body.admissionDetail,
                    specialist : req.body.specialist,
                })
                .then(admission => {
                    if(admission){
                        req.flash("success", "CHILD ADMITTED SUCCESSFULLY");
                        res.redirect("back");
                    }
                })
                .catch(err => {
                    if(err){
                        console.log(err);
                        res.redirect("back");
                    }
                });

            }
        })
    }else{
        req.flash("error", "CHILD MUST HAVE AN ID");
        res.redirect("back");
    }

});
```

//Getting one admission details

```
router.get("/admission/:id", isLoggedIn, (req, res) => {
   Admission.findById({_id : req.params.id})
   .then(admission => {
      if(admission){
         res.render("viewAdmission", {
            title : "Child Admission Info",
            description : "Showing a childs admission information",
            admission : admission
         });
      }else{
         console.log("NO RECORD FOUND");
         res.redirect("back");
      }
   })
   .catch(err => {
      if(err){
         console.log(err);
         res.redirect("back");
      }
   });

});
```

//Updating record in the admission document

```
router.put("/admission/:id/edit", (req, res) => {
    Admission.findByIdAndUpdate({_id : req.params.id}, {
        admissionID : req.body.admissionID,
        ward : req.body.ward,
        bedNumber : req.body.bedNumber,
        admittedDate : req.body.admittedDate,
        childID : req.body.childID,
        admissionDetail : req.body.admissionDetail,
        specialist : req.body.specialist,
    })
    .then(admission => {
        if(admission){
            req.flash("success", "CHILD INFORMATION UPDATED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            req.flash("error", "CHILD INFORMATION UPDATE ERROR");
            res.redirect("back");
        }
    });
});


//Deleting record in the admission document
router.delete("/admission/:id", (req, res) => {
  Admission.findOneAndDelete({_id : req.params.id})
  .then(admission => {
    if(admission){
      req.flash("success", "ADMISSION INFORMATION DELETED
SUCCESSFULLY");
      res.redirect("back");
    }
  })
  .catch(err => {
    if(err){
      console.log(err);
      req.flash("error", "ADMISSION INFORMATION DELETE
UNSUCCESSFUL");
      res.redirect("back");
    }
  });


});


// END OF ADMISSION FUNCTIONALITY
```

//EMERGENCY CRUD FUNCTIONALITY

//Adding record in the emergency document

```
router.post("/emergency/add", (req, res) => {
    if(req.body.childID !== ""){
        Child.findOne({childID : req.body.childID})
        .then(child => {
            if(child){
                Emergency.create({
                    emergencyID : req.body.emergencyID,
                    childID : req.body.childID,
                    regDate : req.body.regDate,
                    emergencyDescription : req.body.emergencyDescription,
                    broughtBy : req.body.broughtBy,
                    bringersContact : req.body.bringersContact,
                    relationship : req.body.relationship
                })
                .then(emergency => {
                    if(emergency){
                        req.flash("success", "EMERGENCY INFORMATION ADDED
SUCCESSFULLY");
                        res.redirect("back");
                    }
                })
                .catch(err => {
                    if(err){
```

```
                    console.log(err);
                    res.redirect("back");
                }
            });

        }
    })
  }else{
    Emergency.create({
        emergencyID : req.body.emergencyID,
        regDate : req.body.regDate,
        emergencyDescription : req.body.emergencyDescription,
        broughtBy : req.body.broughtBy,
        bringersContact : req.body.bringersContact,
        relationship : req.body.relationship
    })
    .then(emergency => {
        if(emergency){
            req.flash("success", "EMERGENCY INFORMATION ADDED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });

  }
});
```

//Getting one emergency info

```
router.get("/emergency/:id", isLoggedIn, (req, res) => {
  Emergency.findOne({_id : req.params.id})
  .then(emergency => {
    if(emergency){
        res.render("viewEmergency", {
            title : "Child emergency information",
            description : "Childs emergency information page",
            emergency : emergency
        });
    }
  })
  .catch(err => {
    if(err){
        req.flash("success", "NO RECORD FOUND");
        res.redirect("back");
    }
    });
});
```

//Updating record in the emergency document

```
router.put("/emergency/:id/edit", (req, res) => {
    Emergency.findByIdAndUpdate({_id : req.params.id}, {
        emergencyID : req.body.emergencyID,
        childID : req.body.childID,
        regDate : req.body.regDate,
        emergencyDescription : req.body.emergencyDescription,
        broughtBy : req.body.broughtBy,
        bringersContact : req.body.bringersContact,
        relationship : req.body.relationship
    })
    .then(emergency => {
        if(emergency){
            req.flash("success", "CHILD INFORMATION UPDATED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "ERROR UPDATING CHILD INFORMATION");
            res.redirect("back");
        }
    });

});
```

//Deleting record in the emergency document

```
router.delete("/emergency/:id", (req, res) => {
    Emergency.findOneAndDelete({_id : req.params.id})
    .then(emergency => {
        if(emergency){
            req.flash("success", "CHILD EMERGENCY INFORMATION DELETED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "ERROR DELETING INFORMATION");
            res.redirect("back");
        }
    });

});
```

// END OF EMERGENCY CRUD FUNCTIONALITY

//WAITING CRUD FUNCTIONALITY

//Adding record in the waiting document

```
router.post("/waiting/add", (req, res) => {
   Waiting.create({
      waitingID : req.body.waitingID,
      childID : req.body.childID,
      specialist : req.body.specialist,
      status : req.body.status,
      waitingFor : req.body.waitingFor,
      waitingDate : req.body.waitingDate
   })
   .then(waiting => {
      if(waiting){
         req.flash("success", "CHILD ADDED TO WAITING LIST
SUCCESSFULLY");
         res.redirect("back");
      }
   })
   .catch(err => {
      if(err){
         req.flash("error", "ERROR ADDING CHILD TO WAITING LIST");
         res.redirect("back");
      }
   });
});
```

//Getting one waiting info

```
router.get("/waiting/:id", isLoggedIn, (req, res) => {
    Waiting.findOne({_id : req.params.id})
    .then(waiting => {
        if(waiting){
            res.render("viewWaiting", {
                title : "Child waiting information",
                description : "Childs waiting information page",
                waiting : waiting
            });
        }else{
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "NO RECORD FOUND");
            res.redirect("back");
        }
    });
});
```

//Updating record in the waiting document

```
router.put("/waiting/:id/edit", (req, res) => {
   Waiting.findByIdAndUpdate({_id : req.params.id}, {
      waitingID : req.body.waitingID,
      childID : req.body.childID,
      specialist : req.body.specialist,
      status : req.body.status,
      waitingFor : req.body.waitingFor,
      waitingDate : req.body.waitingDate
   })
   .then(waiting => {
      if(waiting){
         req.flash("success", "CHILD INFORMATION UPDATED
SUCCESSFULLY");
         res.redirect("back");
      }
   })
   .catch(err => {
      if(err){
         req.flash("error", "ERROR UPDATING CHILD INFORMATION");
         res.redirect("back");
      }
   });

});
```

//Deleting record in the waiting document

```
router.delete("/waiting/:id", (req, res) => {
   Waiting.findOneAndDelete({_id : req.params.id})
   .then(waiting => {
      if(waiting){
         req.flash("success", "CHILD WAITING INFORMATION DELETED
SUCCESSFULLY");
         res.redirect("back");
      }
   })
   .catch(err => {
      if(err){
         req.flash("error", "ERROR DELETING INFORMATION");
         res.redirect("back");
      }
   });

});
```

//END OF WAITING CRUD FUNCTIONALITY

//VISITORS CRUD FUNCTIONALITY

//Adding record in the visitors document

```
router.post("/visitors/add", (req, res) => {
    Visitors.create({
        visitorsID : req.body.visitorsID,
        childID : req.body.childID,
        name : req.body.name,
        purpose : req.body.purpose,
        date : req.body.visitDate
    })
    .then(visitors => {
        if(visitors){
            req.flash("success", "VISITORS INFORMATION ADDED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "ERROR ADDING VISITORS INFORMATION");
            res.redirect("back");
        }
    });

});
```

//Getting all the visitors

```
router.get("/visitor/:childID", isLoggedIn, (req, res) => {
    Visitors.find({childID : req.params.childID})
    .then(visitors => {
        if(visitors){
            res.render("viewVisitor", {
                title : "Showing all visitors",
                description : "Showing all the childs visitors",
                visitors : visitors
            })
        }
    })
    .catch(err=> {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    })

});
```

//Getting all the visitors

```
router.get("/visitor/:childID/editVisitors", isLoggedIn, (req, res) => {
    Visitors.find({childID : req.params.childID})
    .then(visitors => {
        if(visitors){
            res.render("editVisitor", {
```

```
            title : "Showing all visitors",
            description : "Showing all the childs visitors",
            visitors : visitors
          })
        }
    })
    .catch(err=> {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    })

});
```

//Getting one visitors info

```
router.get("/visitors/:id", isLoggedIn, (req, res) => {
    Visitors.findOne({_id : req.params.id})
    .then(visitors => {
        if(visitors){
            res.render("viewVisitors", {
                title : "Child visitors information",
                description : "Childs visitors information page",
                visitors : visitors
            });
        }else{
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("success", "NO RECORD FOUND");
            res.redirect("back");
        }
    });
});
```

//Updating record in the visitors document

```
router.put("/visitors/:id/edit", (req, res) => {
    Visitors.findByIdAndUpdate({_id : req.params.id}, {
        visitorsID : req.body.visitorsID,
        childID : req.body.childID,
        name : req.body.name,
        purpose : req.body.purpose,
        date : req.body.visitDate
    })
    .then(visitors => {
        if(visitors){
            req.flash("success", "CHILD VISITORS INFORMATION UPDATED
SUCCESSFULLY");
```

```
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "ERROR UPDATING INFORMATION");
            res.redirect("back");
        }
    });

});
```

//Deleting record in the visitors document

```
router.delete("/visitors/:id", (req, res) => {
    Visitors.findOneAndDelete({_id : req.params.id})
    .then(visitors => {
        if(visitors){
            req.flash("success", "CHILD VISITORS INFO DELETED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });

});
```

//END OF VISITORS CRUD FUNCTIONALITY

//REFERRAL CRUD FUNCTIONALITY

//Adding record in the referral document

```
router.post("/referral/add", (req, res) => {
    Referral.create({
        referralID : req.body.referralID,
        childID : req.body.childID,
        referredTo : req.body.referredTo,
        fromClinic : req.body.fromClinic,
        toClinic : req.body.toClinic,
        date : req.body.date,

    })
    .then(referral => {
        if(referral){
            req.flash("success", "REFERRAL INFO ADDED SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "ERROR ADDING REFERRAL");
            res.redirect("back");
        }
    });

});
```

//Getting one referral info

```
router.get("/referral/:id", isLoggedIn, (req, res) => {
    Referral.findOne({_id : req.params.id})
    .then(referral => {
        if(referral){
            res.render("viewReferral", {
                title : "Child referral information",
                description : "Childs referral information page",
                referral : referral
            });
        }else{
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "NO RECORD FOUND");
            res.redirect("back");
        }
    });
});
```

//Updating record in the referral document

```
router.put("/referral/:id/edit", (req, res) => {
    Referral.findByIdAndUpdate({_id : req.params.id}, {
        referralID : req.body.referralID,
        childID : req.body.childID,
        referredTo : req.body.referredTo,
        fromClinic : req.body.fromClinic,
        toClinic : req.body.toClinic,
        date : req.body.date
    })
    .then(referral => {
        if(referral){
            req.flash("success", "CHILD REFERRAL INFO UPDATED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("success", "ERROR UPDATING REFERRAL INFORMATION");
            res.redirect("back");
        }
    });

});
```

//Deleting record in the referral document

```
router.delete("/referral/:id", (req, res) => {
    Referral.findOneAndDelete({_id : req.params.id})
    .then(referral => {
        if(referral){
            req.flash("success", "CHILD REFERRAL INFO DELETED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "ERROR DELETING INFORMATION");
            res.redirect("back");
        }
    });
```

```
});

//END OF REFERRAL CRUD FUNCTIONALITY

//DELIVERY CRUD FUNCTIONALITY
//Adding record in the delivery document
router.post("/delivery/add", (req, res) => {
  if(req.body.childID !== undefined){
    Child.findOne({childID : req.body.childID})
    .then(child => {
      if(child){
        Delivery.create({
          deliveryID : req.body.deliveryID,
          fatherID : req.body.fatherID,
          motherID : req.body.motherID,
          placeOfDelivery : req.body.placeOfDelivery,
          typeOfDelivery : req.body.typeOfDelivery,
          childID : req.body.childID,
          deliveryRegDate : req.body.deliveryRegDate,
          dateDelivered : req.body.dateDelivered,
          timeOfDelivery : req.body.timeOfDelivery,
          deliveredBy : req.body.deliveredBy,
          registrationCentre : req.body.registrationCentre
        })
        .then(delivery => {
                if(delivery){
                    req.flash("success", "DELIVERY INFO ADDED
SUCCESSFULLY");
                    res.redirect("back");
                }
            })
            .catch(err => {
                if(err){
                    req.flash("error", "ERROR ADDING DELIVERY INFORMATION");
                    res.redirect("back");
                }
            });

        }
    })
  }else{
    Delivery.create({
        deliveryID : req.body.deliveryID,
        fatherID : req.body.fatherID,
        motherID : req.body.motherID,
        placeOfDelivery : req.body.placeOfDelivery,
        typeOfDelivery : req.body.typeOfDelivery,
        deliveryRegDate : req.body.deliveryRegDate,
        dateDelivered : req.body.dateDelivered,
        timeOfDelivery : req.body.timeOfDelivery,
        deliveredBy : req.body.deliveredBy,
        registrationCentre : req.body.registrationCentre
    })
    .then(delivery => {
        if(delivery){
            console.log("DELIVERY INFO ADDED SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });

  }
});
```

//Getting one delivery info

```
router.get("/delivery/:id", isLoggedIn, (req, res) => {
    Delivery.findOne({_id : req.params.id})
    .then(delivery => {
        if(delivery){
            res.render("viewDelivery", {
```

```
            title : "Child delivery information",
            description : "Childs delivery information page",
            delivery : delivery
         });
      }else{
         res.redirect("back");
      }
   })
   .catch(err => {
      if(err){
         req.flash("error", "NO RECORD FOUND");
         res.redirect("back");
      }
   });
});
```

//Updating record in the delivery document

```
router.put("/delivery/:id/edit", (req, res) => {
   Delivery.findByIdAndUpdate({_id : req.params.id}, {
      deliveryID : req.body.deliveryID,
      fatherID : req.body.fatherID,
      motherID : req.body.motherID,
      placeOfDelivery : req.body.placeOfDelivery,
      typeOfDelivery : req.body.typeOfDelivery,
      childID : req.body.childID,
        deliveryRegDate : req.body.deliveryRegDate,
        dateDelivered : req.body.dateDelivered,
        timeOfDelivery : req.body.timeOfDelivery,
        deliveredBy : req.body.deliveredBy,
        registrationCentre : req.body.registrationCentre
    })
    .then(delivery => {
        if(delivery){
            req.flash("success", "CHILD DELIVERY INFO UPDATED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "ERROR UPDATING INFORMATION");
            res.redirect("back");
        }
    });

});
```

//Deleting record in the delivery document

```
router.delete("/delivery/:id", (req, res) => {
    Delivery.findOneAndDelete({_id : req.params.id})
    .then(delivery => {
        if(delivery){
            req.flash("success", "CHILD DELIVERY INFO DELETED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "ERROR DELETING INFORMATION");
            res.redirect("back");
        }
    });

});
```

//END OF DELIVERY CRUD FUNCTIONALITY

//DECEASED CRUD FUNCTIONALITY

//Adding record in the deceased document

```
router.post("/deceased/add", (req, res) => {
    if(req.body.childId !== undefined){
        Child.findOne({childID : req.body.childId})
        .then(child => {
          if(child){
             Deceased.create({
                deceasedID : req.body.deceasedID,
                timeOfDeath : req.body.timeOfDeath,
                date : req.body.date,
                reportedBy : req.body.reportedBy,
                causeOfDeath : req.body.causeOfDeath,
                childId : req.body.childId,
                residenceAddress : req.body.residenceAddress,
                assumedAreaOfIncident : req.body.assumedAreaOfIncident,
                tribe : req.body.tribe,
                nationality : req.body.nationality,
                gender : req.body.gender,
                lastName : req.body.lastName,
                otherName : req.body.otherName,
                firstName : req.body.firstName
             })
             .then(deceased => {
                if(deceased){
                   req.flash("success", "DECEASED INFO ADDED
SUCCESSFULLY");
                   res.redirect("back");
                }
             })
             .catch(err => {
                if(err){
                    req.flash("error", "ERROR ADDING DECEASED
INFORMATION");
                    res.redirect("back");
                }
            });


        }
    })
  }else{
      Deceased.create({
          deceasedID : req.body.deceasedID,
          timeOfDeath : req.body.timeOfDeath,
          date : req.body.date,
          reportedBy : req.body.reportedBy,
          causeOfDeath : req.body.causeOfDeath,
          residenceAddress : req.body.residenceAddress,
          assumedAreaOfIncident : req.body.assumedAreaOfIncident,
          tribe : req.body.tribe,
          nationality : req.body.nationality,
          gender : req.body.gender,
          lastName : req.body.lastName,
          otherName : req.body.otherName,
          firstName : req.body.firstName
      })
```

```
        .then(deceased => {
            if(deceased){
                req.flash("success", "DECEASED INFO ADDED SUCCESSFULLY");
                res.redirect("back");
            }
        })
        .catch(err => {
            if(err){
                req.flash("error", "ERROR DELETING DECEASED INFORMATION");
                res.redirect("back");
            }
        });

    }

});
```

//Getting one deceased info

```
router.get("/deceased/:id", isLoggedIn, (req, res) => {
    Deceased.findOne({_id : req.params.id})
    .then(deceased => {
        if(deceased){
            res.render("viewDeceased", {
                title : "Child deceased information",
                description : "Childs deceased information page",
                deceased : deceased
            });
        }else{
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "NO RECORD FOUND");
            res.redirect("back");
        }
    });
});
```

//Updating record in the deceased document

```
router.put("/deceased/:id", (req, res) => {
    Deceased.findByIdAndUpdate({_id : req.params.id}, {
        deceasedID : req.body.deceasedID,
        timeOfDeath : req.body.timeOfDeath,
        date : req.body.date,
        reportedBy : req.body.reportedBy,
        causeOfDeath : req.body.causeOfDeath,
        childId : req.body.childId,
        residenceAddress : req.body.residenceAddress,
        assumedAreaOfIncident : req.body.assumedAreaOfIncident,
        tribe : req.body.tribe,
        nationality : req.body.nationality,
        gender : req.body.gender,
        lastName : req.body.lastName,
        otherName : req.body.otherName,
        firstName : req.body.firstName
    })
    .then(deceased => {
        if(deceased){
            req.flash("success", "CHILD DECEASED INFO UPDATED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "ERROR UPDATING INFORMATION");
            res.redirect("back");
        }
    });

});
```

//Deleting record in the deceased document

```
router.delete("/deceased/:id", (req, res) => {
   Deceased.findOneAndDelete({_id : req.params.id})
   .then(deceased => {
      if(deceased){
         req.flash("success", "CHILD DECEASED INFO DELETED
SUCCESSFULLY");
         res.redirect("back");
      }
   })
   .catch(err => {
      if(err){
         req.flash("error", "ERROR DELETING DECEASED INFORMATION");
         res.redirect("back");
      }
   });

});
```

//END OF DECEASED CRUD FUNCTIONALITY

//TEST CRUD FUNCTIONALITY

// Adding record in the father document

```
router.post("/test/add", files.single("labResult"), (req, res) => {
    if(req.file !== "" && (req.file.mimetype === "application/pdf")){
        if(req.body.childID !== ""){
            Child.findOne({childID : req.body.childID})
            .then(child => {
                if(child){
                    Test.create({
                        type : req.body.type,
                        requestedBy : req.body.requestedBy,
                        labResult : req.file.filename,
                        childID : req.body.childID
                    })
                    .then(test => {
                        if(test){
                            req.flash("success", "TEST INFORMATION SAVED
SUCCESSFULLY");
                            res.redirect("back");
                        }
                    })
                    .catch(err => {
                        if(err){
                            console.log(err);
                            res.redirect("back");
                        }
                    });
                }else{
                    req.flash("error", "CHILD ID IS NOT IN THE SYSTEM");
                    res.redirect("back");
                }
            })
        }else{
            Test.create({
                type : req.body.type,
                requestedBy : req.body.requestedBy,
                labResult : req.file.filename,
                childID : req.body.childID
            })
            .then(test => {
                if(test){
                    req.flash("success", "TEST INFO SAVED SUCCESSFULLY");
                    res.redirect("back");
                }
            })
            .catch(err => {
                if(err){
                    console.log(err);
                    res.redirect("back");
                }
            });
        }
```

```
    }else{
        req.flash("error", "TEST FILE IS MANDATORY AND MUST BE A PDF
FILE");
        res.redirect("back");
    }
});
```

//Getting one test information

```
router.get("/test/:id", isLoggedIn, (req, res) => {
    Test.findOne({_id : req.params.id})
    .then(test => {
        if(test){
            res.render("viewTest", {
                title : "Test Information",
                description : "Showing test information",
                test : test
            });
        }else{
            req.flash("error", "CHILD ID NOT IN SYSTEM");
            res.redirect("back");
        }

    })
    .catch(err => {
        if(err){
            req.flash("error", "NO RECORD FOUND");
            res.redirect("back");
        }
    });

});
```

// Updating record in the father document

```
router.put("/test/:id/edit", files.single("labResult"), (req, res) => {
    const type = req.file
    if(type !== undefined && (req.file.mimetype === "application/pdf")){
        Test.findOneAndUpdate({_id : req.params.id}, {
            type : req.body.type,
            requestedBy : req.body.requestedBy,
            labResult : req.file.filename,
            childID : req.body.childID
        })
        .then(test => {
            if(test){
                req.flash("success", "TEST INFORMATION UPDATED
SUCCESSFULLY");
                res.redirect("back");
            }
        })
        .catch(err => {
            if(err){
                console.log(err);
                res.redirect("back");
            }
        });

    }else{
        Test.findOneAndUpdate({_id : req.params.id}, {
            type : req.body.type,
            requestedBy : req.body.requestedBy,
            childID : req.body.childID
        })
        .then(test => {
            if(test){
                req.flash("success", "TEST INFORMATION UPDATED
SUCCESSFULLY");
                res.redirect("back");
            }
        })
        .catch(err => {
            if(err){
                req.flash("error", "TEST FILE IS REQUIRED AND MUST BE A PDF");
                res.redirect("back");
            }
        });
    }

});
```

// Deleting test information

```
router.delete("/test/:id", (req, res) => {
    Test.findById({_id : req.params.id})
    .then(test => {
        if(test){
            gfs.files.findOne({filename : test.labResult}, (err, file) => {
                if(file){
                    gfs.files.deleteOne({filename : file.filename}, (err) => {
                        if(err){
                            console.log(err);
                        }else{
                            Test.findOneAndRemove({_id : req.params.id})
                            .then(deleted => {
                                if(deleted){
                                    req.flash("success", "TEST INFORMATION DELETED
SUCCESSFULLY");
                                    res.redirect("back");
                                }
                            })
                        }
                    });
                }else{
                    console.log(err);
                }
            });
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });

});
```

//END OF TEST CRUD FUNCTIONALITY

//PRESCRIPTION CRUD FUNCTIONALITY

// Adding record in the prescription document

```
router.post("/prescription/add", (req, res) => {
    if(req.body.childID !== ""){
        Child.findOne({childID : req.body.childID})
        .then(child => {
            if(child){
                Prescription.create({
```

```
                doneBy : req.body.doneBy,
                prescription : req.body.prescription,
                childID : req.body.childID,
                sickness : req.body.sickness
            })
            .then(prescription => {
                if(prescription){
                    req.flash("success", "PRESCRIPTION INFO SAVED
SUCCESSFULLY");
                    res.redirect("back");
                }
            })
            .catch(err => {
                if(err){
                    console.log(err);
                    res.redirect("back");
                }
            });
        }else{
            req.flash("error", "CHILD ID IS NOT IN THE SYSTEM");
            res.redirect("back");
        }
    })
  }else{
    req.flash("success", "CHILD ID MUST NOT BE BLANK");
```

```
        res.redirect("back");
    }


});
```

//Getting one prescription information

```
router.get("/prescription/:id", isLoggedIn, (req, res) => {
    Prescription.findOne({_id : req.params.id})
    .then(prescription => {
        if(prescription){
            res.render("viewPrescription", {
                title : "Prescription Information",
                description : "Showing prescription information",
                prescription : prescription
            });
        }else{
            req.flash("error", "CHILD ID NOT IN SYSTEM");
            res.redirect("back");
        }


    })
    .catch(err => {
        if(err){
            req.flash("error", "NO RECORD FOUND");
            res.redirect("back");
        }
    });


});
```

// Updating record in the prescription document

```
router.put("/prescription/:id/edit", (req, res) => {
    Prescription.findOneAndUpdate({_id : req.params.id}, {
        doneBy : req.body.doneBy,
        prescription : req.body.prescription,
        childID : req.body.childID,
        sickness : req.body.sickness
    })
    .then(prescription => {
        if(prescription){
            req.flash("success", "PRESCRIPTION INFO UPDATED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });
});


// Deleting prescription information
router.delete("/prescription/:id", (req, res) => {
   Prescription.findOneAndRemove({_id : req.params.id})
   .then(deleted => {
      if(deleted){
         req.flash("success", "PRESCRIPTION INFORMATION DELETED
SUCCESSFULLY");
         res.redirect("back");
      }
   })
   .catch(err => {
      if(err){
         console.log(err);
         res.redirect("back");
      }
   });

});

//END OF PRESCRIPTION CRUD FUNCTIONALITY


//WARD CRUD FUNCTIONALITY
```

// Adding record in the ward document

```
router.post("/ward/add", (req, res) => {
   Ward.create({
      name : req.body.name,
      description : req.body.description,
      hod : req.body.hod,
      noOfBeds : req.body.noOfBeds
   })
   .then(ward => {
      if(ward){
         req.flash("success", "WARD INFO SAVED SUCCESSFULLY");
         res.redirect("back");
      }
   })
   .catch(err => {
      if(err){
         req.flash("error", "CHILD ID IS NOT IN THE SYSTEM");
         res.redirect("back");
      }
   });

});
```

//Getting one ward information

```
router.get("/ward/:id", isLoggedIn, (req, res) => {
   Ward.findOne({_id : req.params.id})
```

```
    .then(ward => {
        if(ward){
            res.render("viewWard", {
                title : "Ward Information",
                description : "Showing ward information",
                ward : ward
            });
        }else{
            req.flash("error", "CHILD ID NOT IN SYSTEM");
            res.redirect("back");
        }

    })
    .catch(err => {
        if(err){
            req.flash("error", "NO RECORD FOUND");
            res.redirect("back");
        }
    });

});
```

// Updating record in the ward document

```
router.put("/ward/:id/edit", (req, res) => {
    Ward.findOneAndUpdate({_id : req.params.id}, {
        name : req.body.name,
        description : req.body.description,
        hod : req.body.hod,
        noOfBeds : req.body.noOfBeds
    })
    .then(ward => {
        if(ward){
            req.flash("success", "WARD INFO UPDATED SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });

});
```

// Deleting ward information

```
router.delete("/ward/:id", (req, res) => {
    Ward.findOneAndRemove({_id : req.params.id})
    .then(deleted => {
        if(deleted){
            req.flash("success", "WARD INFORMATION DELETED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });

});
```

//END OF WARD CRUD FUNCTIONALITY

//DIAGNOSIS CRUD FUNCTIONALITY

// Adding record in the diagnosis document

```
router.post("/diagnosis/add", (req, res) => {
    Diagnosis.create({
        diagnosedBy : req.body.diagnosedBy,
        diagnosis : req.body.diagnosis,
        childID : req.body.childID
    })
    .then(diagnosis => {
        if(diagnosis){
            req.flash("success", "DIAGNOSIS INFO SAVED SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "CHILD ID IS NOT IN THE SYSTEM");
            res.redirect("back");
        }
    });

});
```

//Getting one diagnosis information

```
router.get("/diagnosis/:id", isLoggedIn, (req, res) => {
    Diagnosis.findOne({_id : req.params.id})
    .then(diagnosis => {
        if(diagnosis){
            res.render("viewWard", {
                title : "Diagnosis Information",
                description : "Showing diagnosis information",
                diagnosis : diagnosis
            });
        }else{
            req.flash("error", "CHILD ID NOT IN SYSTEM");
            res.redirect("back");
        }


    })
    .catch(err => {
        if(err){
            req.flash("error", "NO RECORD FOUND");
            res.redirect("back");
        }
    });


});
```

// Updating record in the diagnosis document

```
router.put("/diagnosis/:id/edit", (req, res) => {
    Diagnosis.findOneAndUpdate({_id : req.params.id}, {
        name : req.body.name,
        description : req.body.description,
        hod : req.body.hod,
        noOfBeds : req.body.noOfBeds
    })
    .then(ward => {
        if(ward){
            req.flash("success", "DIAGNOSIS INFO UPDATED SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });

});
```

// Deleting diagnosis information

```
router.delete("/diagnosis/:id", (req, res) => {
    Diagnosis.findOneAndRemove({_id : re q.params.id})
    .then(deleted => {
        if(deleted){
            req.flash("success", "DIAGNOSIS INFORMATION DELETED
SUCCESSFULLY");
            res.redirect("back");
        }
    })
    .catch(err => {
        if(err){
            console.log(err);
            res.redirect("back");
        }
    });

});
```

//END OF DIAGNOSIS CRUD FUNCTIONALITY

//Birth Certificate Search Logic

```
router.post("/birth_certificate/:childID", (req, res) => {
    Child.findById({_id : req.params.childID})
    .then(child => {
        if(child){
            res.redirect(`/birth_certificate/${child._id}`);
            res.render("birthCertificate", {
                title : "Child's Birth Certificate",
                description : "Child's Birth Certificate Information"
            });
        }
    })
});
```

//Birth Certificate Search Info

```
router.get("/birth_certificate/:id", isLoggedIn, (req, res) => {
    Delivery.findById({_id : req.params.id})
```

```
.then(delivery => {
return  Child.findOne({childID : delivery.childID})
.then(child => {
return Father.findOne({_id : child.father})
.then(father => {
return Mother.findOne({_id : child.mother})
.then(mother => {
   if(mother){
      res.render("birthCertificate", {
         title : "Child's Birth Certificate",
         description : "Showing child's birth certificate",
         child : child,
         mother : mother,
         father : father,
         delivery : delivery
      });
   }else{
      res.render("birthCertificate", {
         title : "Child's Birth Certificate",
         description : "Showing child's birth certificate",
         child : child,
         father : father,
         delivery : delivery
      });
   }
```

```
    })
    })
    })
    })
    .catch(err => {
        if(err){
            req.flash("error", "CHILD ID NOT IN SYSTEM");
            res.redirect("back");

        }
    });

});
```

//Death Certificate Search Logic

```
router.get("/death_certificate/:childID", (req, res) => {
    Deceased.findOne({childId : req.params.childID})
    .then(deceased => {
        if(deceased){
            res.render("deathCertificate", {
                title : "Child's Death Certificate",
                description : "Child's Death Certificate Information",
                deceased : deceased
            });
        }
    })
    .catch(err => {
        if(err){
            req.flash("error", "CHILD ID NOT IN THE SYSTEM");
            res.redirect("back");
        }
    });
});
```

// END OF REPORTS SECTION

//GETTING THE FILES

```
router.get("/files/:filename", (req, res) => {
    gfs.files.findOne({filename : req.params.filename}, (err, foundFile) => {
        if(foundFile){
            const readstream = gfs.createReadStream(foundFile.filename);
            readstream.pipe(res);
        }else{
            console.log(err);
        }
    });
});
```

4.2 PROGRAM IMPLEMENTATION

To affect a changeover into the new system, the health institution should adopt (chapter 3) steps as was discussed, to prepare the ground for the implementation and other factors which includes:

STAFF TRAINING

With the help of the program flowchart (item 4.2) the algorithm (4.1) and system chart (3.9) staff can be wonderfully equipped to use this application.

PROGRAMMING

Though this application has been exhaustively tested to meet the user requirements, we, the developers are ready to render directives in the course of implementation.

SYSTEM TESTING

The system has been tested on the basis of program flow and procedure flow. The information from the indexes proves the success so far. Files have also been converted to meet the user requirement as in the former system.

CHANGE OVER

We recommend parallel conversion for this system. This is because, since the old system has been a manual-based, this new one needs to be gradually absorbed before phasing out to avoid uncertainties.

DOCUMENTATION/USER GUIDE

This system is developed as a web-based application. In this project, both the elementary and advanced features of Web Development Technologies are extensively utilized to achieve the goal of the system. Web Development Technologies make it possible to enter data in user readable form. After processing, the user can get the output either in form of softcopy or hardcopy.

Hardware and software requirement

Made simple enough, the program has been designed in order to enable the user execute on any machine with minimum hardware requirement. This is evident by the fact that it can be accessed using any modern up to date web browser. The program, though it is designed with Web Development tools, does not require these tools to be installed on the user's machine.

USER GUIDE

The user can run application as

- Power on the computer system (booting)
- After a successful booting, open a web browser
- Ensure the computer is connected to the internet
- Enter the URL of the web application. This will open the application in the user's web browser, and request for login credentials
- The user is to enter his/her credentials, and if the user has an account in the system, then the user will be logged in.

CHAPTER FIVE

SUMMARY, CONCLUTION AND RECOMMENDATIONS

5.0 SUMMARY

According to our project topic, it is justified that a computer can be adopted to process data related to CHILD CARE RECORD.

The health institution, "Ola Durin Children Hospital", facing several approaches, needs a system that can automatically prepare:

- test result
- Comprehensive lab. Result sheet or list
- patient doctors report and as well as offer accurate/effective:
  - Health care services or treatment system
  - Maintaining security system.
  - Filling /access system
  - Updating/maintaining system for children's medical record (deletion, update and insertion operation)
- Records outputs in the form of soft copies and hard copies. With respect to achieving these outlines above, the project narrows concentration down to the patients.

5.1 CONCLUSION

Digitizing data at large is much more ideal and effective towards solving data processing problems with indent analyses of the vast activities of computerization covered in this project earlier. The prime goal to replace the manual system of recording medical data of children with the named computerized system can serve in a greater capacity of capturing all necessary medical records that can be highly secured by the system.

Having gone through tough challenges in collecting the required data and other requirements needed, we made a breakthrough and was able to accomplish the development of the system. For easy and on-the-go accessibility, it was designed to be a web-based application with high and effective security protocols defined against unauthentic access to records, and against cyber-attacks.

In spite of the time that may be required for users or operators to become familiar with the system, the institution (for which it has been developed) can find the operations of the new system ideal as easy tasks to process children's records.

Defined in this documentation are guides that you will find useful during implementation of the system.

5.2 RECOMMENDATIONS

Because of the high level of accuracy, efficiency, effectiveness, and security computerizing data brings with it, it should be used in running the daily activities of data processing (as per children's records) in Ola Durin Children Hospital, Cottage Hospital Section. In this order, we recommend this new system to Ola Durin Children Hospital to be used in keeping tracks of children's medical records.

With the addition of a webcam as an input device, a registered Avast anti-virus is to be installed on all devices that are to be used to access this system. This is to ensure that they cannot be hacked, infected with malware, used as a spying tool, or used as a back door into the system.

Regular update of the installed anti-virus is advised, as new malwares, viruses, and other nefarious tools, software, and techniques of intrusion are devised almost daily. On that note, a steady 24/7 internet connection is advised to be used by these systems, and ensuring auto update of the anti-virus software is enabled.

REFRENCES

Abdullah, J.I (2004) Introduction to the computer, A management tool: Victory publisher Nigeria, No. 2 odor street Owerri.

Feingold C. (1997) Introduction to data processing 2nd Edition USA, W.M.C. Brown Company publishers.

French, C.S. (2002) Computer Science, Book Power publisher London.

Loudon, K.C. and Loudon, J.P (1991) Business Information Systems A PROBLEM SOLVING APPROACH USA THE Dryden Press.

Henry C. L. (1978) Information Technology for Management (sixth Edition) New York University, McGraw-Hill companies, New York.

FEDERAL MEDICAL COMMISSIONING PROJECT BOOKLET 2009
To 2011 federal medical centre owerri.

Oparah, C.C and Oguike, O.E (2006) Management Information System, shack publisher Nigeria Owerri.

Orilla, L.S (1979) Introduction to Business Data processing New York, McGraw- Hill.

Barba D.L.A (1979) https://www.academia.edu/40401417/DESIGN_AND_IMPLEMENTATION_OF_A_COMPUTERIZED_STADIUM_MANAGEMENT_INFORMATION_SYSTEM

Zachariyah, et al., 2015. An Automated Database Management System For Births And Deaths Sierra Leone

WEBSITES

Bible Verse https://www.bible.com/bible/compare/PSA.127.3-5

W3C https://www.w3c.0rg

NodeJS https://nodejs.org/

Mongodb: https://mongodb.com/

https://www.developer.mozilla.org/en-US/

Legal age of a child https://www.ohchr.org/EN/ProfessionalInterest/Pages/CRC.aspx (PART I, Article 1)

Avast security: https://www.avast.com/c-webcam-security#:~:text=Hackers%20can%20gain%20webcam%20access,hacking%20malware%20on%20your%20device.