BIRTH & DEATH REGISTRY AUTOMATION SYSTEM

BiRDS

Prepared for: Dr. Samira Sadaoui

Prepared by: Sibdow Abdul-Jalil Iddrisu & Ibrahim Alotaibi

Date: 29th March, 2018

Course Name: CS872 – Software Engineering

Table of Contents

List of Figures

1. Problem Definition

“A name and nationality are every child’s right, enshrined in the Convention on the Rights of the Child and other international treaties” [2]. Equally, death registrations are important, as countries need to know how many people die each year as well – and the main causes of their deaths – in order that they may redesign or improve their health systems to reduce those deaths caused by diseases. However, according to the WHO, two-thirds (38 million) of 56 million annual deaths are still not registered and almost half of the world’s children go unregistered [1].

Population is constantly changing sometimes quiet rapidly that, it is not economically feasible to conduct censuses every now and then. These changes in population is measured by the difference between population size at different times or dates. The two main elements known to affect population growth are births and deaths. These elements are very difficult to obtain due to numerous factors. Sometimes, the fact that individuals have to form long queues just to register throws them off. This problem may only arise when things are done manually and applicants would have to fill out forms by writing, hence delaying and prolonging the registrations. Prospective applicants may sometimes have to wait months-on-end before getting feedback from these registry departments regarding their applications, because it has to be sent to the central registry which may be miles away, hence many loose faiths in the efficiency and effectiveness in the department and are reluctant to register though prescribed by the law.

These shortfalls could be overcome by the use of ICT to develop a very effective and efficient software system, i.e. BiRDS, to help the process.

This study will also aim at integrating the birth and death registration into the health sector, where all data regarding births and death could easily be captured quickly and accurately.

BiRDS shall aim to decentralize the activities of the department and collate the activities at the various registration centers in the districts to the regional offices rather than sending them out of the region to a central registry which could easily be overwhelmed in such instances.

2. Feasibility Study

In this section, we shall briefly describe the proposed system; the benefits of having such a system proposed to solve the above related problem as opposed to the current system and deciding whether it is economically feasible to proceed with development.

2.1 Proposed System

Vital human information is now an important commodity in the world and sells very much. For such reason this information should be made available. A nation spends so much on population census because of lack of a **complete computerized registration system**. Also, applicants spend so much time and resources on registration and obtaining of certificate. With the advent of Information Communication Technology (ICT), a nation can make very good use of its resource by investing in such a system. Therefore, the BiRDS software system is being proposed.

The Birth and Death Registry Automation System (BiRDS) is a web-based system that seeks to eliminate the challenges the department currently faces in the registration of births and deaths occurring, as well as in the issuance of these certificates.

2.2 Proposed System vs Existing System

2.2.1 Existing System

The current mode of operation is a manual way of registering a child and a deceased person. With this, registration takes place at the hospital, a health center or any location within the community for example under a tree. This is always done by an official assigned to that post.

With the birth registration process, the birth of every child is to be registered in the district where the birth occurred. An informant is required to produce evidence of birth, such as a clinical weighing card. A Registration Assistant administers a questionnaire, (the Birth Report Form "A") to the informant. Information thereby collected is recorded in the center’s register of births following which a birth certificate is issued [3].

The Registration Assistant at the registration center sends the Register of Births and the Registration Forms to the District Office that the center falls under. The Registration Assistant then transfers all the entries from the Center Register into the District Register of Births. The District Officer having collected all entries from the various registration centers under him/her, then sends the District Register of Birth and the Registration Forms to the Regional Office. The Officer then transfers all the entries from his/her register into the Regional Register of Birth. The Regional Officer also scans all the registration forms from the various districts, save them on a CD-ROM or pen drive and then travel to Accra, the Central Registry Office with the CD-ROM or pen drive and the Regional Register of Birth ("Big Book") every Wednesday for verification and issuing of certificates. When birth certificates are brought from the Central Registry Office, they are sorted according to districts for the various district officers. For an applicant of a birth certificate, it takes at least 21 working days, i.e. three (3) weeks or more to get a certificate from the department.

The death registration is done almost through the same method as the birth registration, but with this a Death Report Form "B" is issued by the registration assistant.

2.2.2 Comparison between Systems

To the best of our knowledge, there was no computerized system for the department to collate these records. The manual system has been in existence for some time now and has become cumbersome and problematic in this current age. It is inefficient and ineffective to help the department achieve its goals. This proposed system, which aims to computerize the processes performed by the department shall ultimately solve the problems with the current system, which includes delay in processing, difficulty in retrieving data, etc.

2.3 Benefits of Proposed System to Stakeholders

The aims and objectives of this proposed system brings a lot of benefits to stakeholders, these include:

- To provide an economical and convenient means of getting births and deaths registered
- To provide a computerized database on all citizens in the country
- To reduce time spent on registration and in obtaining a certificate.
- To help reduce the amount of errors made by applicants and registrars alike when filling out an application form
- To reduce the stress the department goes through to provide services to applicants
- To provide means of interaction between the various communities and the registry offices
- To help expediate the retrieval of birth and death records from a centralized storage
- To help provide the government with reliable data that shall help in making decisions to enhance the health sector as well as the economy

3. Software Requirements Specifications

In this section, we introduce the actors of the system and go ahead to list their functional requirements. We then explain the use case diagrams of the system along with three (3) most important use cases of the system. Finally, we explain the quality requirements of the system.

3.1 Functional Requirements

The proposed system shall have 3 main actors, who are able to perform functions that would lead to the achievement of the overall goals of the system. These actors are:

- Regional Administrator
- District Administrator
- Center Registration Assistant (Registrar)

Regional Administrator

- The regional administrator shall be able to add new users to the system, who report directly to him/her (i.e. District Admins and Registrars)
- The system shall also enable this user to easily update system user details working under their jurisdiction, should in case the system user personal details change
- The system shall enable this user to manage all users accounts (i.e. activate/deactivate account) within their jurisdiction (i.e. working under his/her region)
- The system shall allow this user to easily change their passwords
- The system shall allow this user to easily create new district offices in their region
- The system shall allow this user to manage (edit/delete) the district offices in their region of management
- The system shall allow this user to create new center offices under the various districts in their region
- The system shall as well allow this user to manage(edit/delete) these center offices in their region

- The system shall allow this user to make the final approvals of all birth details entered into the system by the registrar and approved by the district admin
- The system shall allow this user to make the final approvals of all deceased details entered into the system by the registrar and approved by the district admin
- The system shall allow this user to easily approve and print all birth certificate request made within their region
- The system shall allow this user to easily approve and print all death certificate request made within their region
- The system shall provide user with a means of making a general search for a particular birth or death details

District Administrator

- The system shall allow this user to easily assign a registrar working within their district to a particular station (district center)
- The system shall allow for this user to easily change their passwords
- The system shall allow for user to easily approve all registration of birth details entered into system by registrar, before it is being forwarded to the regional level
- The system shall allow user to easily approve all registration of deceased details entered into system by registrar, before it is being forwarded to the regional level
- The system shall provide the means for user to easily approve birth certificate requests made within their district before being forwarded to regional admin's office for printing
- The system shall provide the means for user to easily approve death certificate requests made within their district before being forwarded to regional admin's office for printing
- The system shall provide user with a means of making a general search for a particular birth or death details

Center Registration Assistant (Registrar)

- The system shall provide a means for this user to easily change their passwords
- The system shall provide a means for the user to register births details into the system
- The system shall provide the means for the user to register deceased details into the system
- The system shall provide user with a means of making a general search for a particular birth or death details
- The system shall provide the means for user to easily place requests for a birth certificate for clients
- The system shall provide the means for user to easily place requests for a death certificate for clients

3.2 Use Cases

In this section, we provide a detailed use case diagram for all users of the system. We then give a detailed description of the three (3) most complex use cases in the system.

3.2.1 Use Case Diagrams for all User Types

Registration Center Official (Registrar):

The registrar is the user who initiates most of the processes that are executed within the system. This user is responsible of entering birth and deceased details of individuals into the system. S/he is also responsible of placing a request for a birth or death certificate for an applicant. This user is also able to make a general query on birth and death details stored in the centralized database as well as being able to amend his/her user account information.

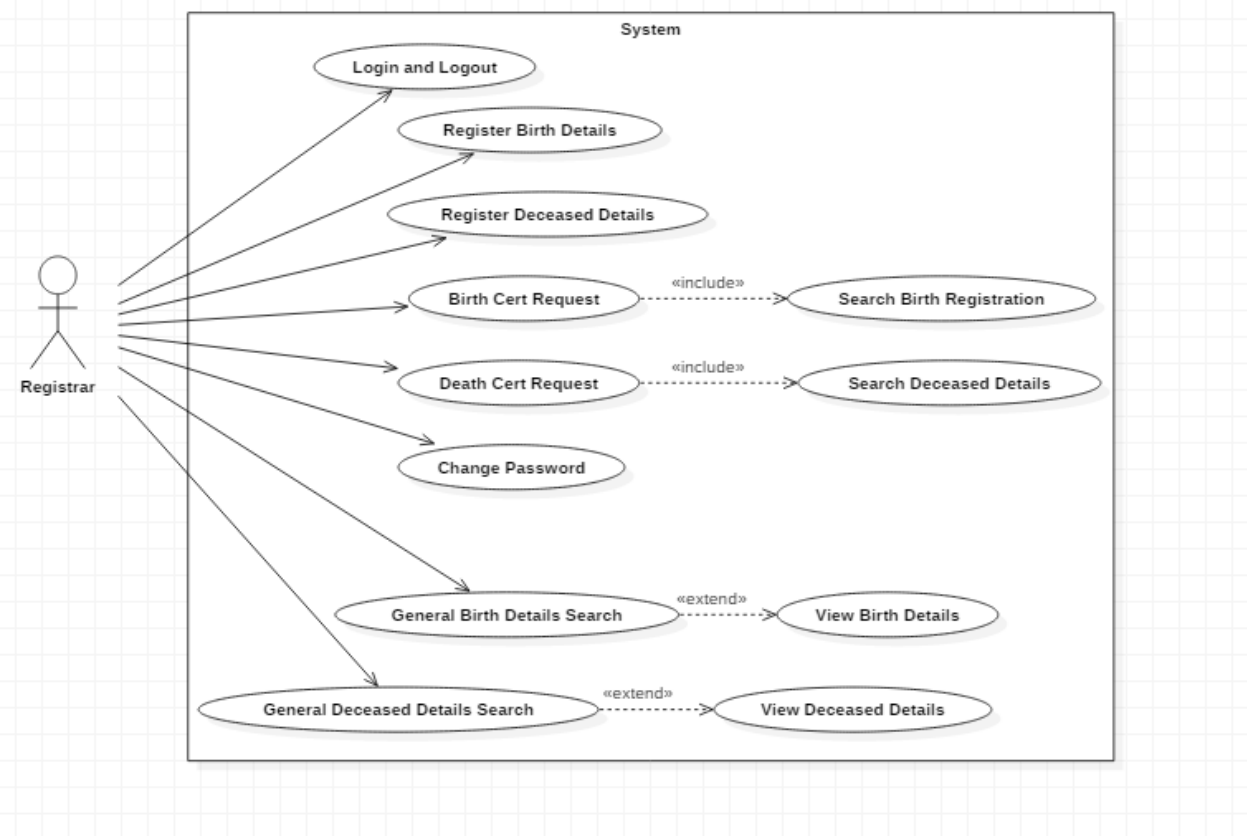


Figure 3.2.1a Use Case diagram for the Registrar Actor

District Administrator

The district administrator is the user whom the registrar reports directly to. After the registrar has entered the birth or deceased details into the system, it is forwarded to this user who confirms everything needed in the application is present before s/he forwards it to the regional level. This user also is responsible for the approval and forwarding of all birth and death certificate requests made under his/her district to the regional level. S/he is also able to assign staff (registrars) under his district to various district centers, make a general query on birth and deceased details stored in the centralized database as well as being able to amend his/her use account information.

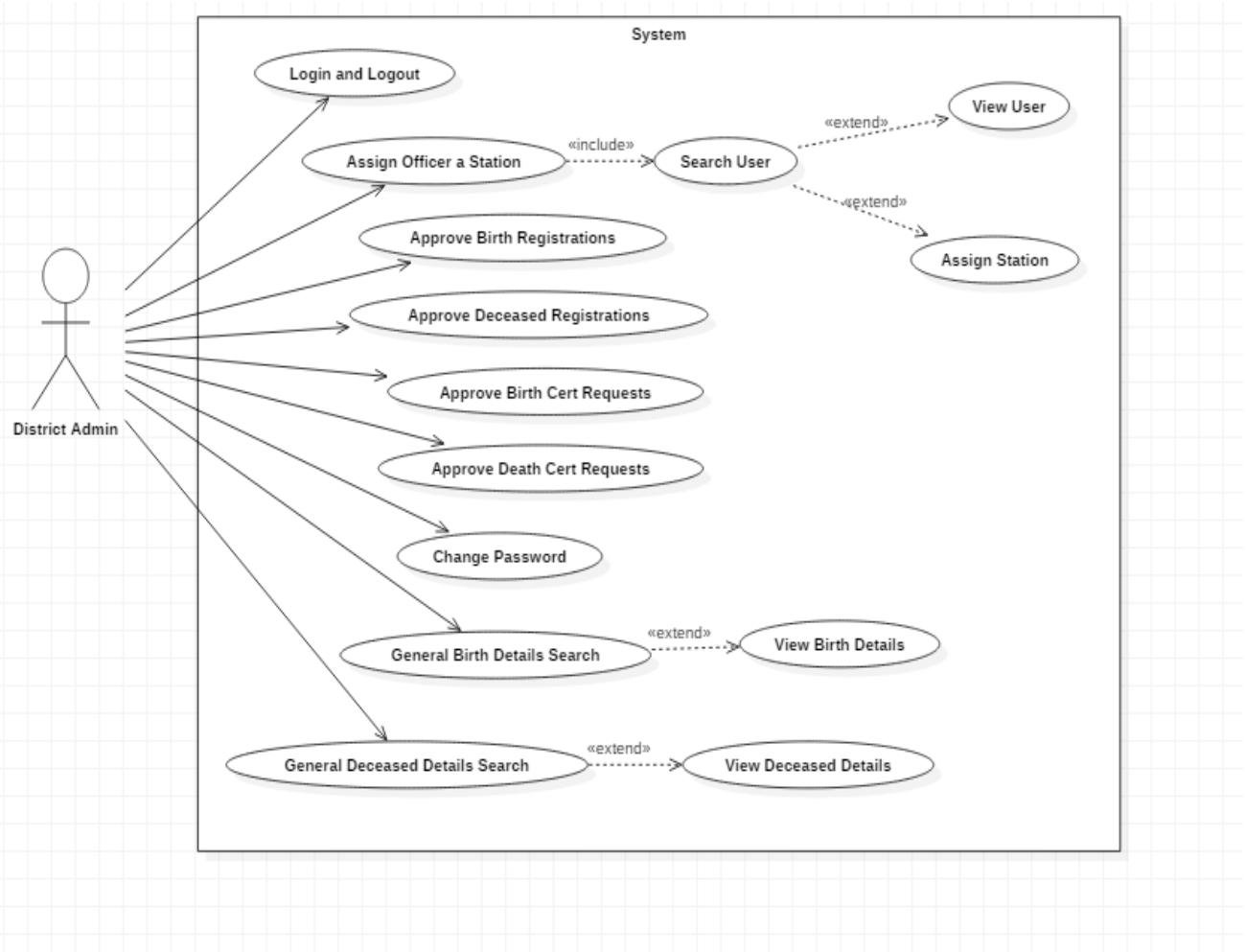


Figure 3.2.1b Use Case diagram for the District Administrator Actor

Regional Administrator

The regional administrator is the overall manager of the above 2 actors. S/he is responsible for registering new system users to serve under his/her region in the various districts, as well as manage these users. He also can register new district offices within the region and assign registration centers to a particular district. S/he also does the final approval of the birth and deceased registration details before it is permanently saved into the centralized database. The regional administrator also can print all the certificates request coming in from the various district offices under his/her region. S/he also can make a general query on birth and deceased details stored in the centralized database as well as being able to amend his/her user account information.

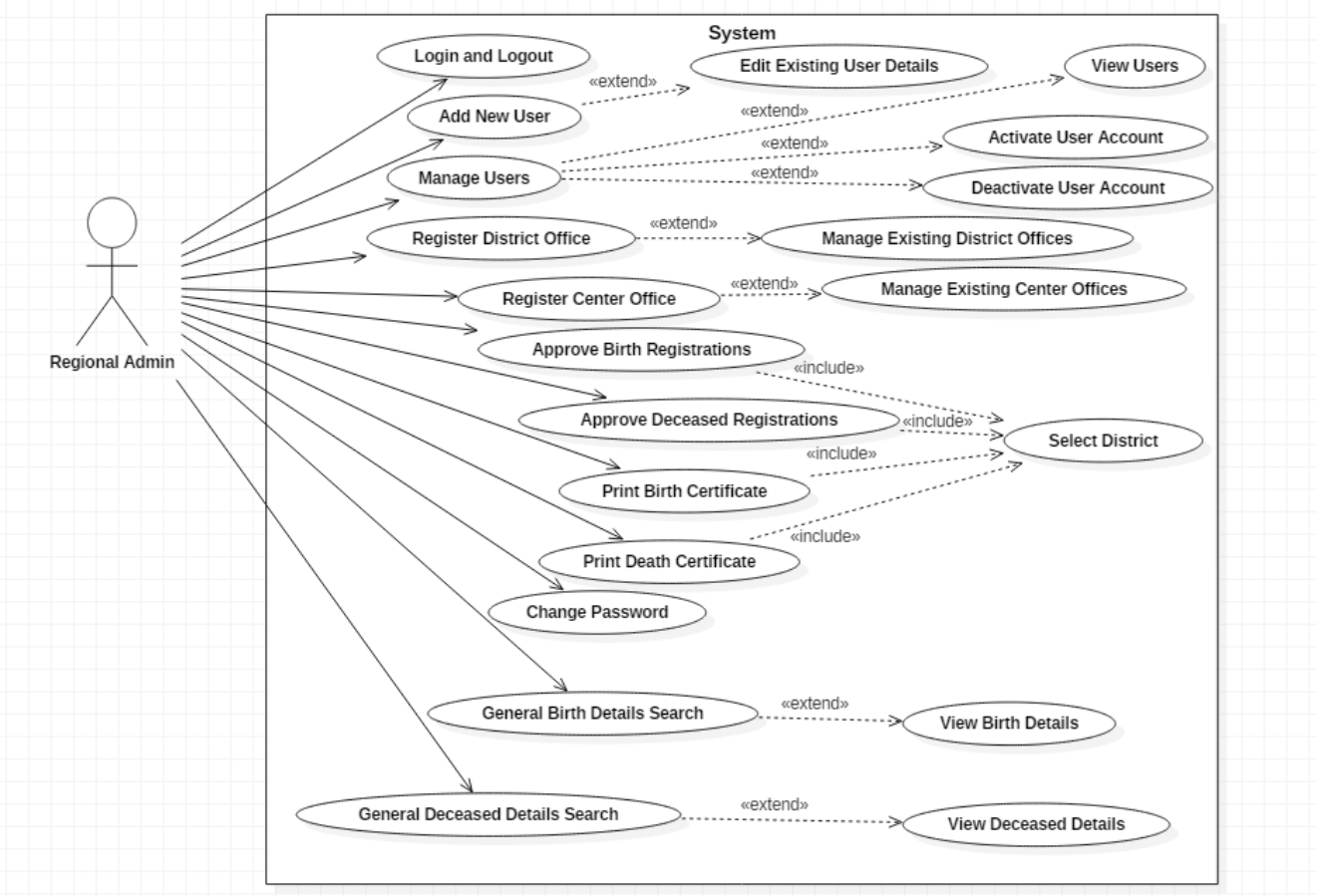


Figure 3.2.1c Use Case diagram for the Regional Administrator Actor

3.2.2 Three Complex Use Cases

The selected three (3) most complex use cases we have selected are:

- Assign Officer a Station: This allows the district administrator to assign staffs working within the district to one district registration center
- Manage User Account: This allows the regional administrator to change the accounts status (i.e. Activate or deactivate a user account) for all staff working under their region of management
- Birth Certificate Request: This allows the registration center officer (registrar) to search for an applicant's record and submit a request for a birth certificate

USE CASE: ASSIGN OFFICER A STATION

A group of registration office assistants (registrar) usually work under a certain district administrator, who is their overall boss. The district being managed has a number of center offices (termed stations) where registrars could be assigned to work. A new user just added to the district is not assigned any center as at that time, it is then the responsibility of the district administrator to do the needful and assign the registrar a station so they may start work. Also, if the registrar is already assigned a station but it becomes necessary for him/her to be reassigned to a different center office, the district administrator can do so through this use case. One important constraint though is that, the district administrator appears in the list of users, but cannot assign him/her self as the form field that enables activity would be disabled upon selecting themselves.

I. Initiating Actor: District Administrator who needs to assign a registrar to a station

II. Precondition: District Administrator is logged in and registrar is working under his/her district of management

III. Scenario 1: Assigning an existing registrar to a center office (station)

- District Administrator search for registrar account details in District Officers Table
- District Administrator selects registrar row to view details
- District Administrator changes user center office
- System then updates registrar details with selected station

Scenario 2: Assign a new registrar to a center office (station)

- District Administrator search for registrar account details in District Officers Table
- District Administrator selects registrar row to view details
- District Administrator selects a center office for new registrar
- System then updates registrar details with selected station

IV. Postcondition: Registrar is assigned to a center office (station) under district

V. Benefiting Actor(s): Registrar and District Administrator

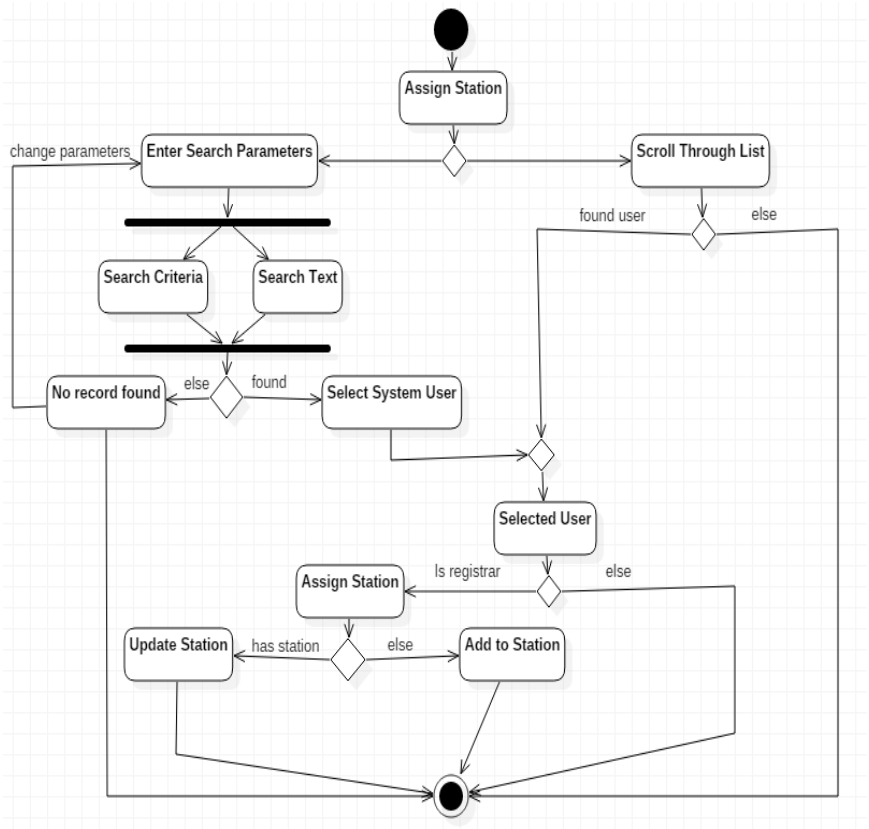


Figure 3.2.2a: Activity Diagram for Assign Officer a Station Use Case

USE CASE: MANAGE USER ACCOUNT

I. Initiating Actor: Regional Administrator needing to change a user account status

II. Precondition: Regional Administrator is logged in and user is working under his region

III. Scenario: Manage User Account

- Regional Administrator search for user account details in Regional Officers Table
- S/he then identifies the row of the user account details
- Regional Administrator then selects the 'Activate' or 'Deactivate' link depending on current state of user account status
- System then update the user account details according to the link selected

IV. Postcondition: User account status is changed to either 'Active' or 'Inactive'

V. Benefiting Actor(s): Registrar, District Administrator, Regional Administrator

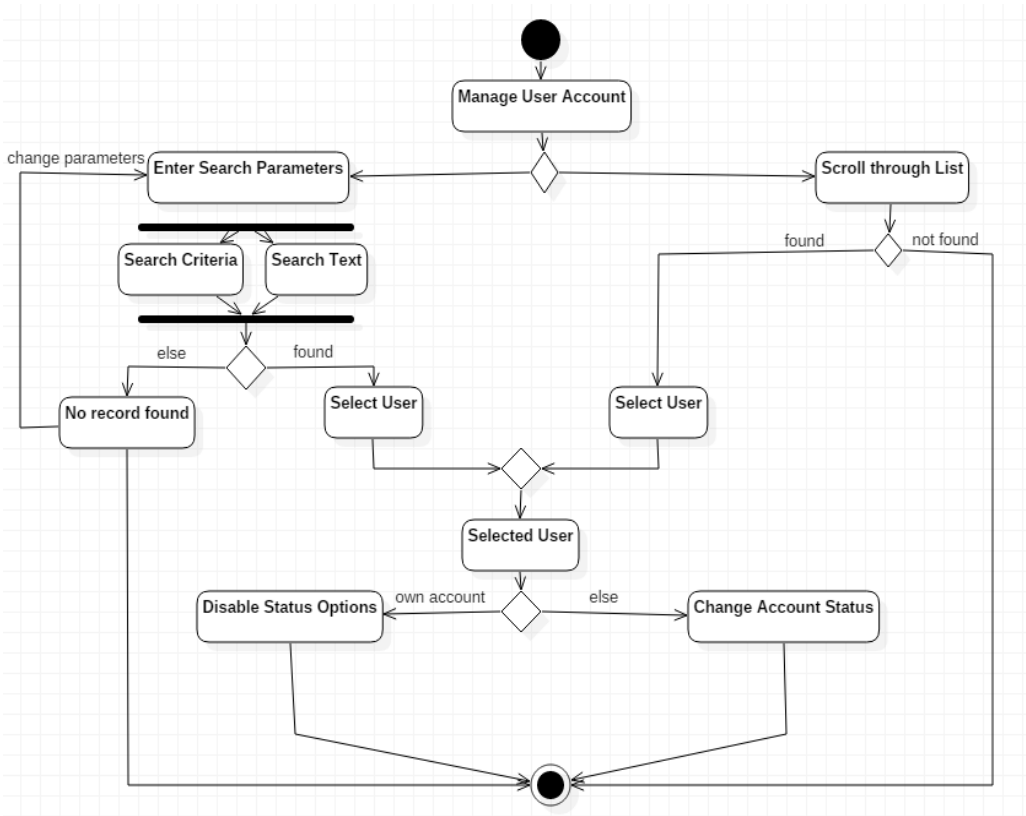


Figure 3.2.2b: Activity Diagram for Manage User Account Use Case

USE CASE: BIRTH CERTIFICATE REQUEST

I. Initiating Actor: A registrar who aims at placing a request on behalf of a client for a birth cert

II. Precondition: Registrar is logged into system and user making request has his/her birth detail already entered into the system as well and approved by the district and regional offices

III. Scenario: Birth Certificate Request

- Registrar search for user by their birth Id or full name criteria
- Registrar enters search text based on criteria chosen above
- User birth details appear on the table list
- Registrar then selects the birth detail by clicking on the “Select” link
- Birth details are viewed in detail on a separate page
- Registrar then submits request on behalf of applicant

IV. Postcondition: Request for a birth certificate has been sent for approval and printing

V. Benefiting Actor(s): The Registrar who seeks to request a birth certificate for a client

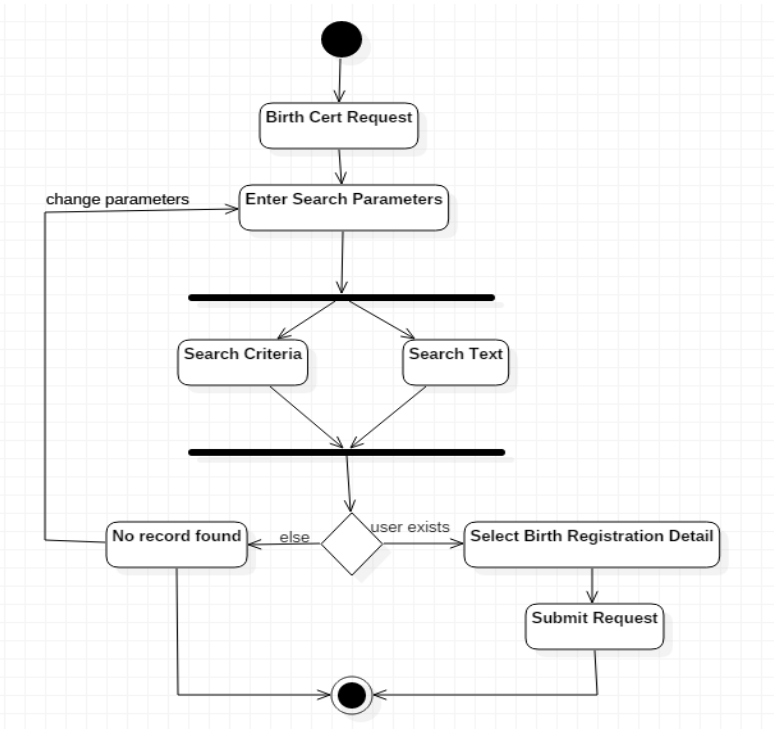


Figure 3.2.2c: Activity Diagram for Birth Cert Request Use Case

3.3 Quality Requirements

3.3.1 Security

- The system shall start with a Login page where users are required to enter correct credentials in order to be allowed access into the core functionalities. Allowing anyone access without credentials validation to just enter data into the system will have a tremendous effect on the data integrity, in that the data cannot be trusted and used.
- There shall be data protection mechanisms in place, i.e. Authorization, so that the different user groups only see data they are authorized to see. For instance, the Registrar cannot see the list of system users nor can they activate/deactivate a user details. You work within your jurisdiction.
- The system shall also provide a data encryption mechanism to encrypt all user passwords in a SHA-256 format. This encryption format generates an almost-unique, fixed-size 256-bit (32-byte) hash [8] of all user passwords and is a one-way function, so the resulting password cannot be decrypted back to the original value. So, say a hacker getting access to user account data from database cannot login with details because the password cannot be decrypted.

3.3.2 Time Efficiency

- The system shall be very responsive and take no longer than 10 seconds in registering a birth/deceased details, making a certificate request, approving of requests and even the printing of certificates. This system is proposed to cut down the time applicants and the service provider take to achieve their goals in the currently existing system. If a user has to wait about a minute or two for a process to load, then it is no different from the previous system. Hence should have a better and faster response time
- During searching the database for a particular birth/deceased details, system shall be able to retrieve record in a very quick and efficient manner for the user

3.3.3 Robustness

- Some users do not understand the implications of what they sometimes enter into a system. The system input fields should be able to accept and process all user inputs and display meaningful and appropriate error messages to them if data passed into fields are wrong, rather than come to an abrupt halt or crush
- For instance, if the user enters a number as a first name of an applicant, which should rather be a string value, the system processes this field and detects it received a number instead of a string value. An appropriate error message is then displayed to the user to change that field to a string value before processing and saving of data can be done.

3.3.4 Correctness

- This is an important software requirement, in that it ensures that the design requirements are implemented as was originally planned. When a system user should enter an input data into the system, the output information must be the correct and expected result
- A correct username and password is required for a user to have access to the functionalities provided by the system. The credentials entered is checked against access control list in the database to make sure user is authorized to use the system. A wrong input shall deny the user from entering and a prompt shall be rendered requesting for correct credentials
- A system user is likely to enter a date of birth or date of death greater than even the current date the registration is taking place, which shouldn't be the case because a date of birth/death cannot occur in the future. The system shall provide a mechanism to make sure such dates are entered correctly by providing appropriate error messages
- A search made by a user shall return the desired results rather than randomly selecting from the storage

3.3.5 User Friendliness

- To the end user, the interface is the system itself and they do not really care about what goes on within the system. Hence, having a nice graphical user interface would not only make the user happy but help him/her better achieve their goals better. As such, the system shall have very beautiful and colorful user interfaces and shall be responsive (i.e. adjust very well to the size of the screen on which it is used)
- The system shall be designed as much as possible to match with the users' mental model [4], in that their real world thinking of how the application should work wouldn't be different from how the system actually works
- The system shall provide for the user an overall positive user experience

4. Design Specification

In this section, we focus on the logical software architecture and design pattern that are used in the development of this system. We then provide the class diagrams to explain further the implementations of the design patterns used.

4.1 Logical Software Architecture

There are several logical software architectures out there that could help us realize the goals we seek to achieve with this project. But looking through these architectures, the MVC (Model View Controller) architecture stood out to be a better choice for our system. It separates a software application into three inter-connected parts. This is usually done to separate the internal representation of information from the way information is presented to and accepted from the user [5].

Components in the MVC Architecture Include:

- **The Model:** This component is responsible for the direct management of the data, logic and the 'rules of engagement' in the software application. The model is constructed without concern for its look and feel when displayed to the user. Our system would have quiet a number of models which would include: System Users, Birth Registration Details, Certificate Requests, etc.
- **The View:** This provides a graphical representation of our underlying models. A model may have multiple views attached to it, i.e., we can have several different displays for some data or model.
- **The Controller:** This component is responsible for processing and decision making. It may accept input and convert it to a command for the model or view. In our system, it shall be responsible for flow control and validation of user inputs before they are passed into the model. Framework used already comes with out-of-the-box controllers

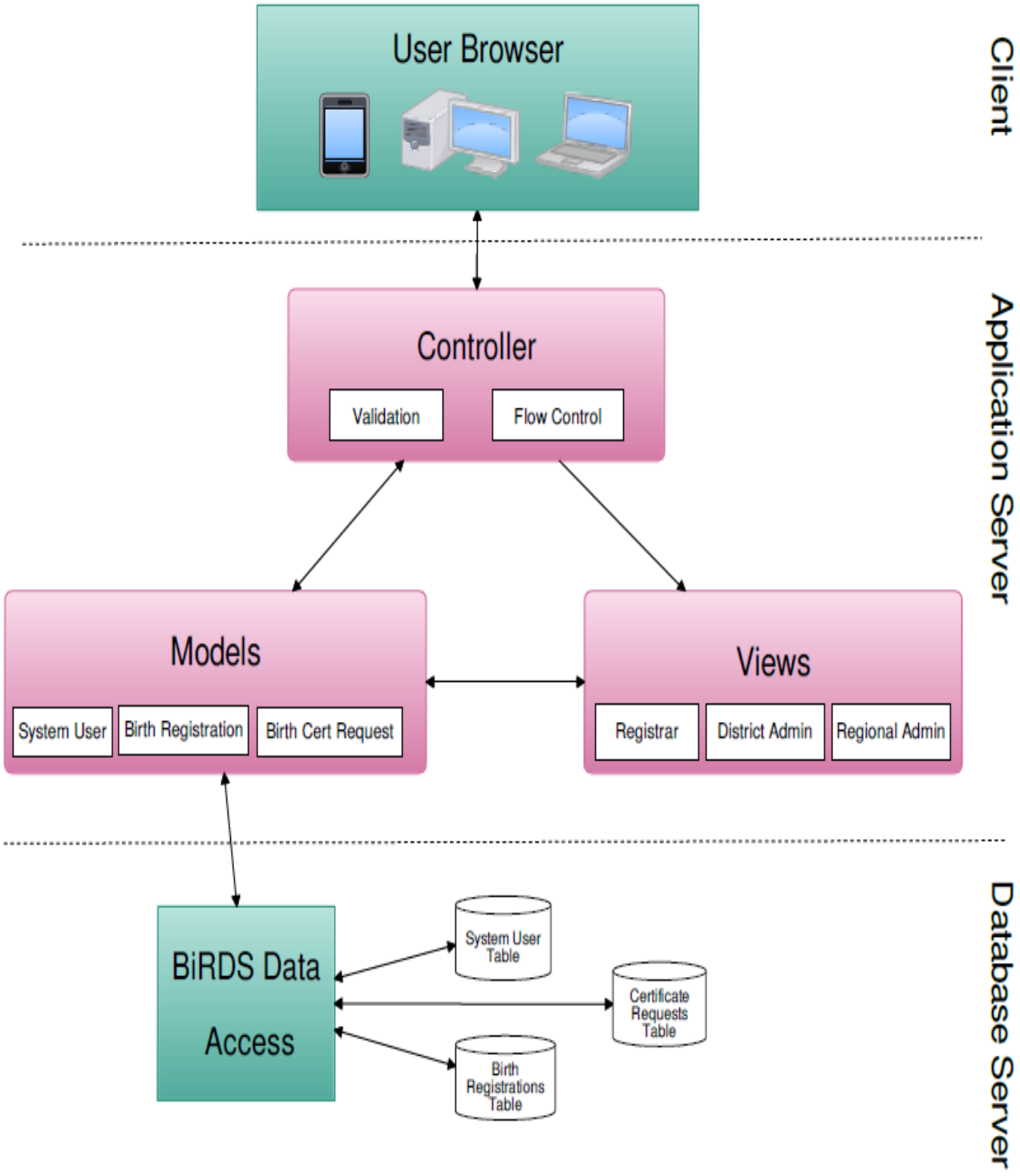


Figure 4.1a Logical Architecture of our system

4.1.1 How the Architecture Works

1. The user initiates the process by sending an HTTP request from the browser to the system

2. The system then chooses the appropriate controller for the user request, and passes control to it

3. The chosen controller then decides the appropriate actions required to complete request

4. Based on the action required, the underlying Model is initialized for the user request

5. The Model then sends a query to the database storage, and the corresponding data is returned into the Model

6. Based on the changes in the Model, the attached Views shall be notified and updated

7. The view is then sent back and rendered on the user's browser in an HTTP response

There may exist different scenarios based on the kind of user requests. For instance, if the user request does not necessitate any database interaction, then the Controller may simply call only the view without using the Model.

4.1.2 Reasons for choosing the MVC Architecture

The MVC architecture has several advantages to our system as compared to the other traditional architecture out there. The reasons why we chose this architecture over the others include:

- Provides a clean separation of concern to our system by isolating the model (business logic) from the Views (user interfaces). Business logic usually doesn't mix well with UI code, so when the two are mixed, the application becomes much harder to maintain and less scalable. MVCs separation of concern shall help our application be more loosely coupled and as a result it shall be easier to modify either the views of the application or the underlying models without one affecting the other.
- MVC assures that there is consistency between the Views and the Models of an application. This feature is a fundamental requirement of our application as an update to certain models should automatically refresh the attached views without them

explicitly being refreshed. MVC provides us with the perfect platform for such an implementation.

- It also allows us to attach as many views as possible to the models of our application.

4.2 Design Patterns

In this section, we present the details about the design patterns that we used in our system, the reasons for using such design patterns and the benefits they bring to our system in general.

There are a lot of design patterns out there to help solve specific problems that may arise during a software development. Our system requires us to use some of these patterns to help solve the problems that our system brings along. There are more than two design patterns used in our project but the two main ones we shall be discussing are:

1. Observer Pattern
2. Template Method Pattern

4.2.1 Observer Pattern

The observer design pattern is one of the behavioral patterns proposed by the Gang of Four [6]. The observer design pattern defines a one-to-many dependency between object in a system, so that when one of the objects change state, all other objects depending on it are notified and updated automatically – More like a publisher-consumer fashion.

Reasons for choosing pattern

- Our system has a form of chain of responsibility, where one user group is directly dependent on a second user group. In this case, the first user group needs to know if an action has been performed by the second and should be automatically notified of this action. These actions include a change to the underlying model, i.e. a new input, an update, etc. Users should not have to refresh pages to see these changes, and there is no better way to achieve such a feat than to use the observer design pattern
- Our system also need to display data from a model in different ways to the different user groups, and the observer design pattern shall help us achieve this goal
- With the logical software architecture chosen, the design pattern that shall help us easily implement the notification system within components of the architecture is the observer pattern

Where it is implemented

This observer pattern has been implemented in multiple instances within our application. We shall discuss in detail one of the instances it has been used within the system. When the Registration Center Officer (Registrar) makes a request for a new Birth Certificate into the system, the District Admin is immediately notified of this change and his/her view of the model is automatically updated. This allows the District Admin to see all the incoming requests in real time rather than having to refresh the view manually from time-to-time in order to see these changes.

The **CertificateRequestController.java** - more like the concrete subject, first creates and fires the event into the application container when the user clicks to save a birth certificate request. The Observer class for the birth certificate request, which is called **BirthCertObserver.java** contains within it a method called the **onBirthCertRequestCreate (@Observes @Create BirthCertRequest birthCertRequest)**, which is responsible for observing events within the application container (with the @Observes declaration). It then specifies the type of event to

observe: **Create BirthCertRequest** (which means any event fired into the application container with a **@Create** qualifier and the object/payload attached to event is the **BirthCertRequest**). This observer method then takes the event fired and then decides as to what to do with it, in our case it sends a push notification through the district admin channel, defined in the **DistrictAdminNotify.java** class, to the page displaying this model and then forces the page to update the view to accommodate for the new changes.

Path on GitHub to CertificateRequestController.java:

https://github.com/Siaj/birdsApp/blob/master/src/main/java/com/app/birds/web/controllers/CertificateRequestController.java

Path on GitHub to BirthCertObserver.java:

https://github.com/Siaj/birdsApp/blob/master/src/main/java/com/app/birds/observer/BirthCertObserver.java

Path on GitHub to DistrictAdminNotify.java:

https://github.com/Siaj/birdsApp/blob/master/src/main/java/com/app/birds/observer/notify/DistrictAdminNotify.java

Other Implementations of the Observer Pattern within the System

- On Registrar save new birth/deceased details, notify and refresh automatically the District Admins view of model as well as the general search views of model so views remain consistent with underlying model
- On District Admin Approve Birth and Deceased registrations models, notify and refresh the general search views of the model as users may need to know in real time if a particular data point/row in the model has changed.
- On Registrar place a request for a Death certificate for a client, notify and update the District Admin's view attached to the underlying models

Complete Class Diagram of the Observer Design Pattern

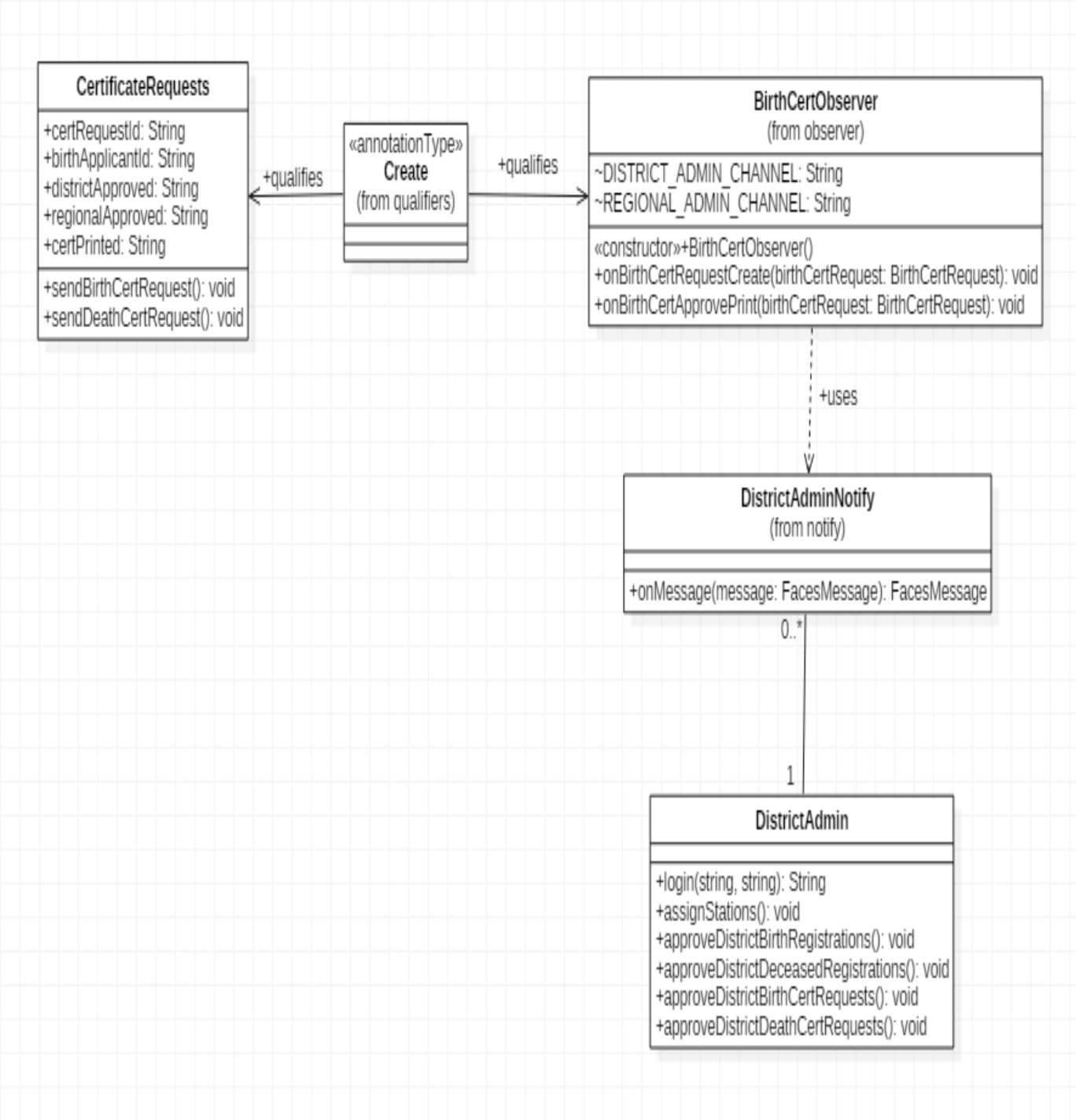


Figure 4.2.1a: Class diagram of Observer Design Pattern Implementation

Algorithms Corresponding to Observer Design Pattern

Algorithm 1: CertificateRequestController Class

```
Public class CertificateRequestController() ←Main class defined to handle certificate requests
//Declare Event to be fired when a Registrar sends a request for a Birth Certificate
@Inject @Create ←Qualifiers that makes the Event unique from other events
Event<BirthCertificateObject> eventName;
Public sendBirthCertRequest() ←function that handles sending birth certificate request
If(Birth Certificate Request Sent)
        eventName.fire(BirthCertificateObject)
Else
        Show error message to user
END ← terminate sendBirthCertRequest() function
END ← terminate main class CertificateRequestController()
```

Algorithm 2: BirthCertObserver Class

```
public class BirthCertObserver() ←class of BirthCertificate Observer
String Channel = "/districtAdmin" ← Declare Chanel to perform push notifications
// @Observes annotation automatically makes method an observer for subjects with events
declaring @Create qualifiers and with payload BirthCertificateObject
Public void onBirthCertRequestCreate(@Observes @Create BirthCertificateObject instance )
        Push notifications through Channel variable to the views
END ← terminate onBirthCertRequestCreate function
END ← terminate class BirthCertObserver()
```

Algorithm 3: DistrictAdminNotify

```
@PushEndpoint("/districtAdmin")
public class DistrictAdminNotify() ←Notification channel class – sends message to views
@OnMessage; encode message into JSON format ←lightweight for easy transmission
Return message to PushEndpoint
END ← terminate DistrictAdminNotify class
```

4.2.2 Template Method Pattern

The template method design pattern is a behavioral pattern, which implies they are mainly responsible for managing algorithms, relationships and responsibilities between the various classes and objects in a software system. This design pattern is intended to define the skeleton of an algorithm in an operation and leaving some steps to be implemented by the subclasses. They allow subclasses to redefine certain steps of an algorithm without changing its structure [6]. It is predominantly used in frameworks (large scale reuse infrastructure).

Reasons for choosing pattern

- Our system has two different components that have a lot of similarities in terms of steps used to reach a particular goal. In this case, if a particular process should change, duplicate efforts would be required to implement those changes in both of the components – which may even increase in the near future as the system evolves and grows. Hence it became necessary to have these common steps defined in a template so as to be easily accessible to the subclasses and when a change is required (say, steps required to reach goals changed), it is done in the template class and subclasses do not need to worry much about the effect of such changes.
- It also affords the components (subclasses) in our system the freedom to easily override and redefine functions within the template class if it became necessary, so as to suite their needs.

Where it is implemented

The **CertificateTemplate.java** class defines the abstract class where the steps required to achieve the goal of printing a Birth or Death Certificate are defined. The **GenerateBirthCert.java** class extends this **CertificateTemplate.java** class and overrides the necessary methods so as to achieve its purpose of generating and printing a Birth Certificate. The **GenerateDeathCertificate.java** class also overrides and customizes the necessary methods in

order to generate a Death Certificate for a client. The different Certificate generation classes are then called and used in the **RegionalApprovalsController.java class** (methods on **line 221** and **line 240**)

Path on GitHub to CertificateTemplate.java:

https://github.com/Siaj/birdsApp/blob/master/src/main/java/com/app/birds/web/reports/CertificateTemplate.java

Path on GitHub to GenerateBirthCert.java:

https://github.com/Siaj/birdsApp/blob/master/src/main/java/com/app/birds/web/reports/GenerateBirthCert.java

Path on GitHub to GenerateDeathCertificate.java:

https://github.com/Siaj/birdsApp/blob/master/src/main/java/com/app/birds/web/reports/GenerateDeathCertificate.java

Path on GitHub to RegionalApprovalsController.java:

https://github.com/Siaj/birdsApp/blob/master/src/main/java/com/app/birds/web/controllers/RegionalApprovalsController.java

Complete Class Diagram of the Template Method Design Pattern

The class diagram of the template method pattern is shown below in figure 4.2.2a. The **RegionalApprovalController**, which is only invoked by the Regional Administrator user group is the client to the **CertificateTemplate** abstract class. The **'final printCertificate()'** function contains the sequential steps required to print a particular certificate and hence cannot be overridden by the subclasses but only called to perform the original function its been defined to perform. But the remain methods may be overridden and customized to achieve the desired goal.

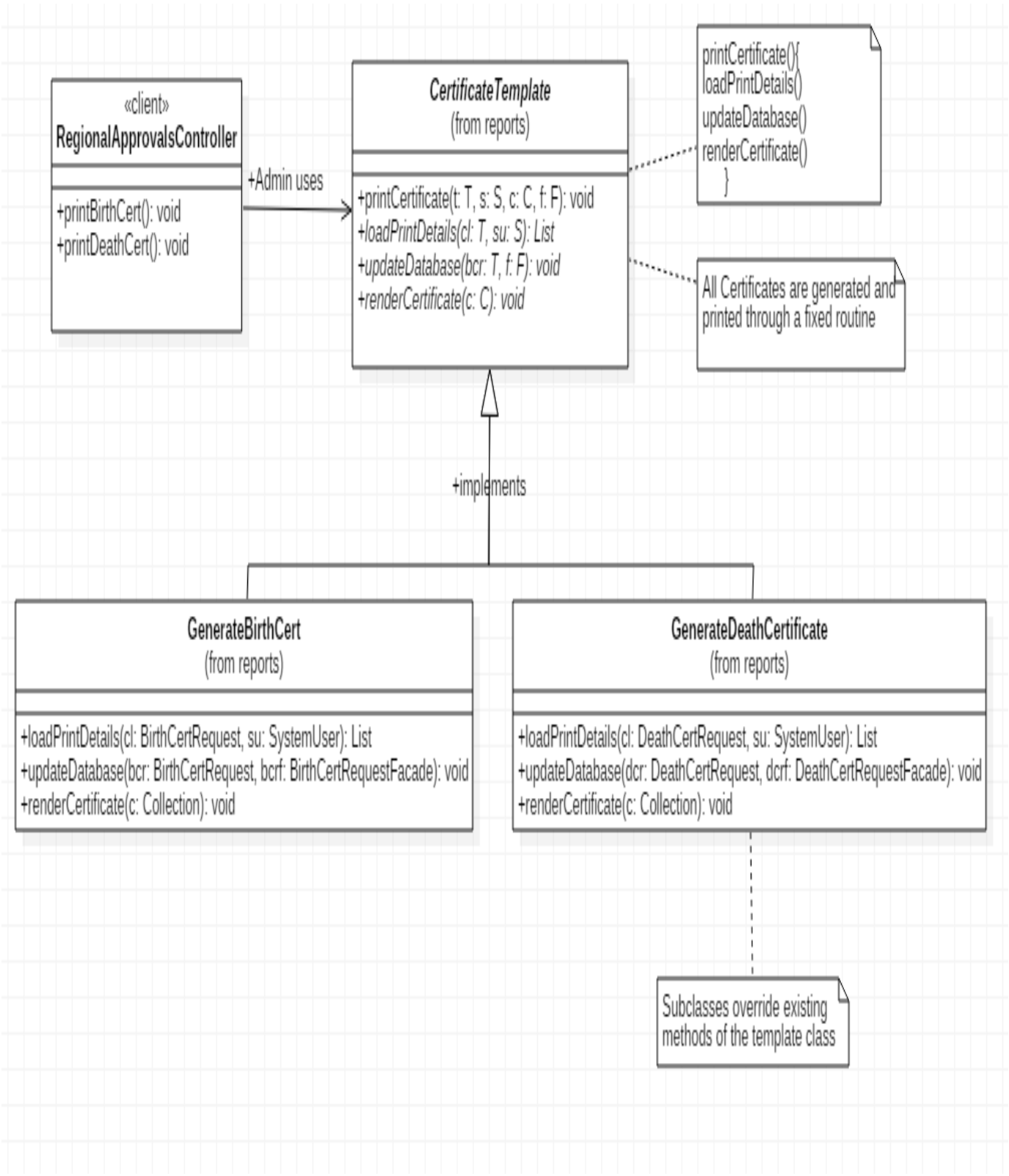


Figure 4.2.2a: Class diagram of Template Method Design Pattern

Algorithms Corresponding to Template Method Design Pattern

Algorithm 1: CertificateTemplate Class

```
abstract class CertificateTemplate () ← Main abstract class created
// Methods are defined within body of class
final printCertificate() ← Final method implemented in template class can't be overridden
        //Contains the other abstract methods in a sequence of steps to reach goal
        loadPrintDetails();
        updateDatabase();
        renderCertificate();
END ← function printCertificate Terminates here

// Abstract methods defined below to be overridden and modified by subclasses
abstract loadPrintDetails();
abstract updateDatabase();
abstract renderCertificate();
END ← class terminates here
```

Algorithm 2: GenerateBirthCertificate and GenerateDeathCertificate Class

```
class ClassName() extend CertificateTemplate ← extends the CertificateTemplate() abstract class

//Overrides abstract methods in the extended class
@Override
public List loadPrintDetails(); ← Defines own implementation of this method
public void updateDatabase(); ← Defines own implementation of this method
public void renderCertificate(); ← Defines own implementation of this method
END ← class terminates here
```

Algorithm 3: RegionalApprovalsController Class ← Uses template design pattern

```
Public printBirthCertificate() ← Method that invokes the CertificateTemplate
CertificateTemplate Obj = new  GenerateBirthCertificate() ← Certificate template of type Birth
Use Obj created to Call printCertificate() final method in CertificateTemplate() abstract class

-- The flow process defined to generate certificate is followed
-- Certificate is then rendered to user's view in pdf format and can be printed
```

4.3 Class Diagrams

The class diagram shown below in figure 4.3a shows how entities in our system interact with one another. Due to the large number of classes in our application, we are showing only the class diagrams that are involved in the workings of the design patterns used in our application.

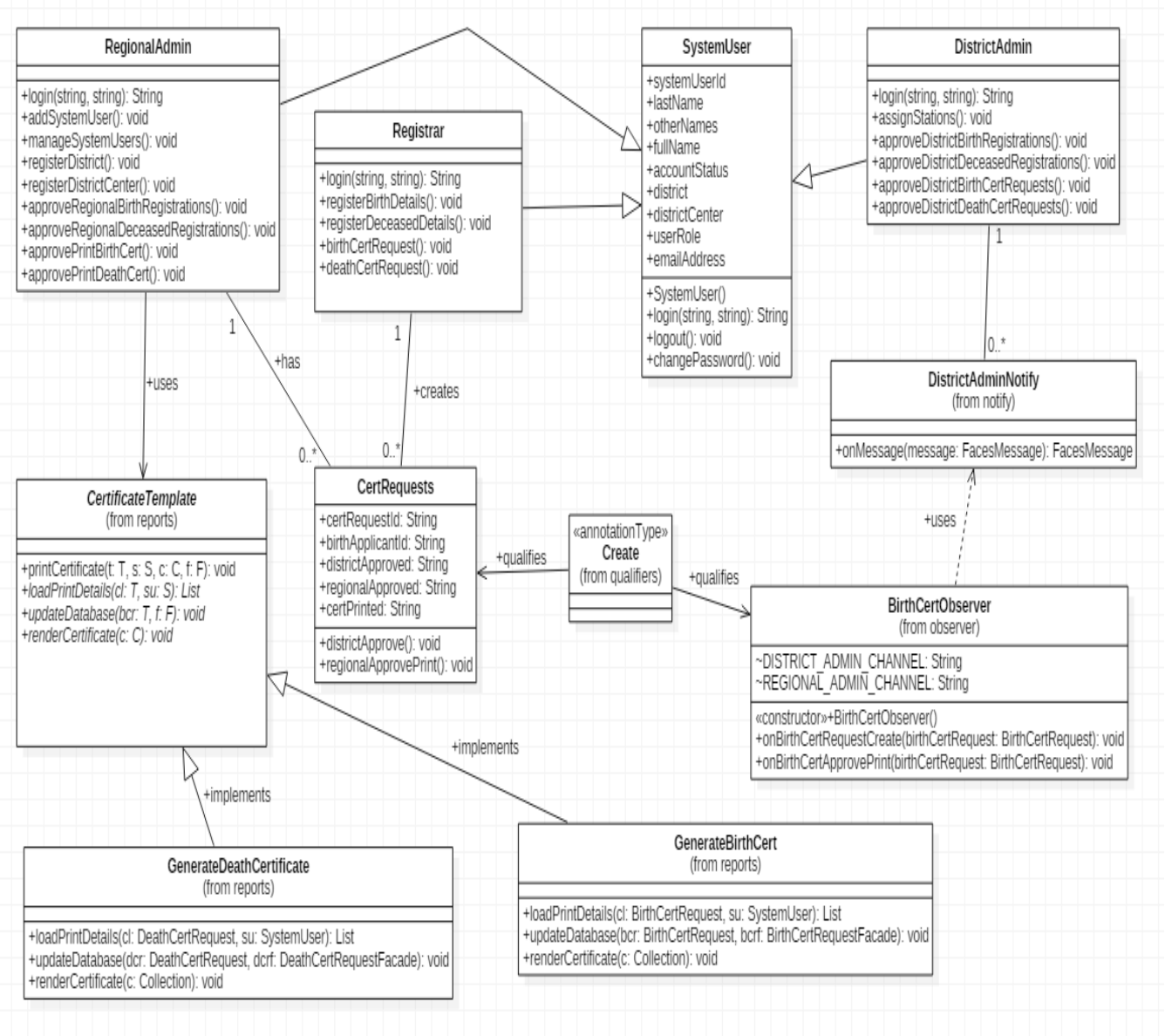


Figure 4.3a: Class diagram of the main entities of our system

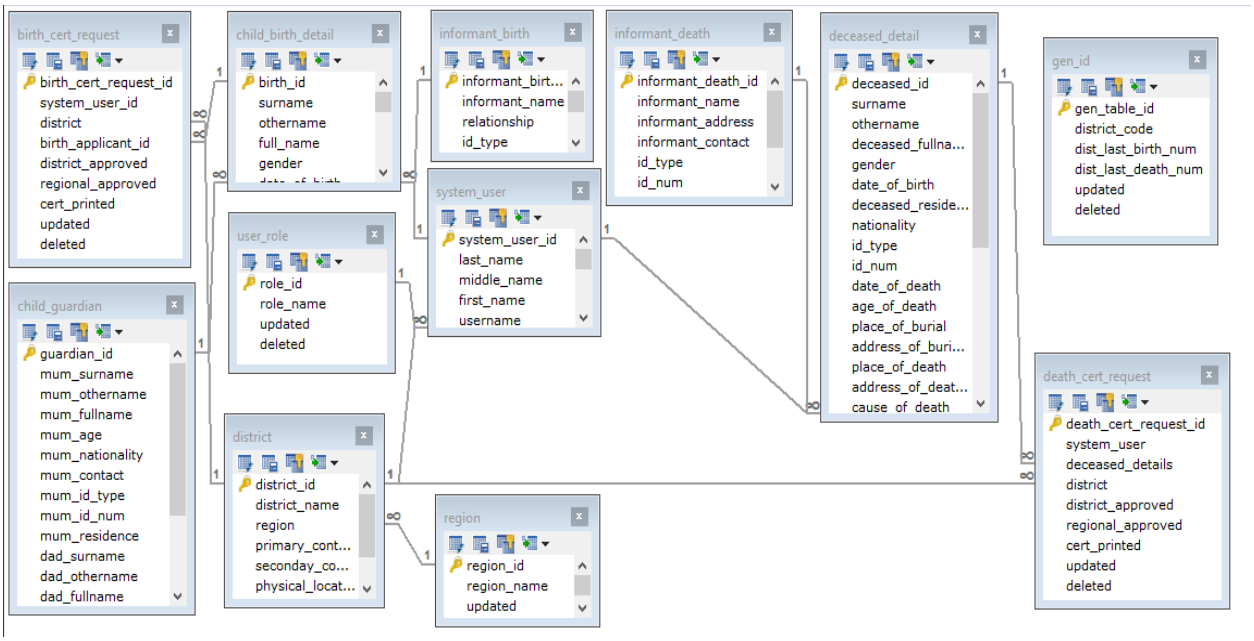


Figure 4.3b: Database schema of how the entities interact of our system

4.4 Pros and Cons of UML Tool Used

The main UML tool used for designing the UML diagrams for our project was the Star UML (version 2.8.1, unregistered version). We found some good stuff about it that helped us quickly design our UML diagram. But as the saying goes, “Nothing is perfect”, we found some other challenges using the application.

To start with, the application is very easy to use out of the box. All labels, menus and other interface entities are clear and easily understandable. It even had a feature where we could import the codes of our classes from our project and easily create our class diagrams from there, instead of starting from scratch.

Some challenges we had using it included: It does not allow the use of special character in labelling, for example, when we tried to name TCP/IP it did not allow it so we had to go with TCPIP. Also, the arrows are sometimes not directed the way we want them and redirecting them is always a big challenge.

Overall, we found Star UML to be useful and beneficial as we easily created all our UML diagrams

5. Source Code

In this section, we present the implementation details of our system which shall include the Component Diagram, Deployment Diagram, number of lines of codes written and screen captures of the database storage.

5.1 Component Diagram

The component diagram of our system in the figure below shows the main components we have within our system and the connection/interaction between these components.

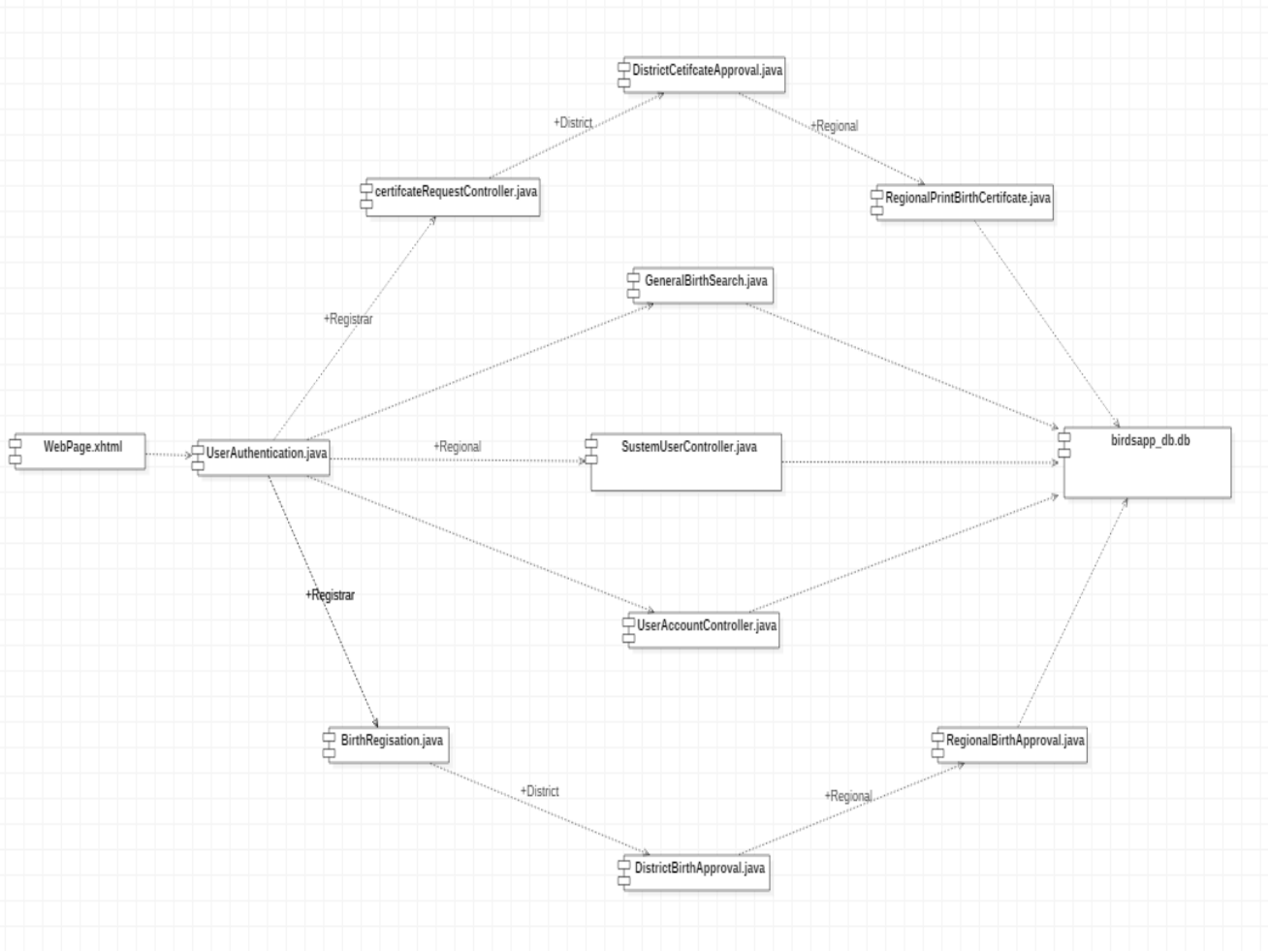


Figure 5.1a: Component Diagram showing organization of code

5.2 Deployment Diagram

Our software system is hosted on two (2) separate servers. There is a separate server for the web application, that is the web server part defined in our logical architectural design. The other server being the database server which hosts our software system's database and all its related tables. The client machine may use any web browser to access the software system. Tests have been carried out on using all the major browsers to access our system, hence users would not need to worry about the kind of browser they have on their computer system.

The software system is hosted on Jelastic (NYC) Platform-as-a-Service (paas) infrastructure.

Application Server Public IP: 185.44.65.95:4848

Database Server Public IP: 185.44.66.235

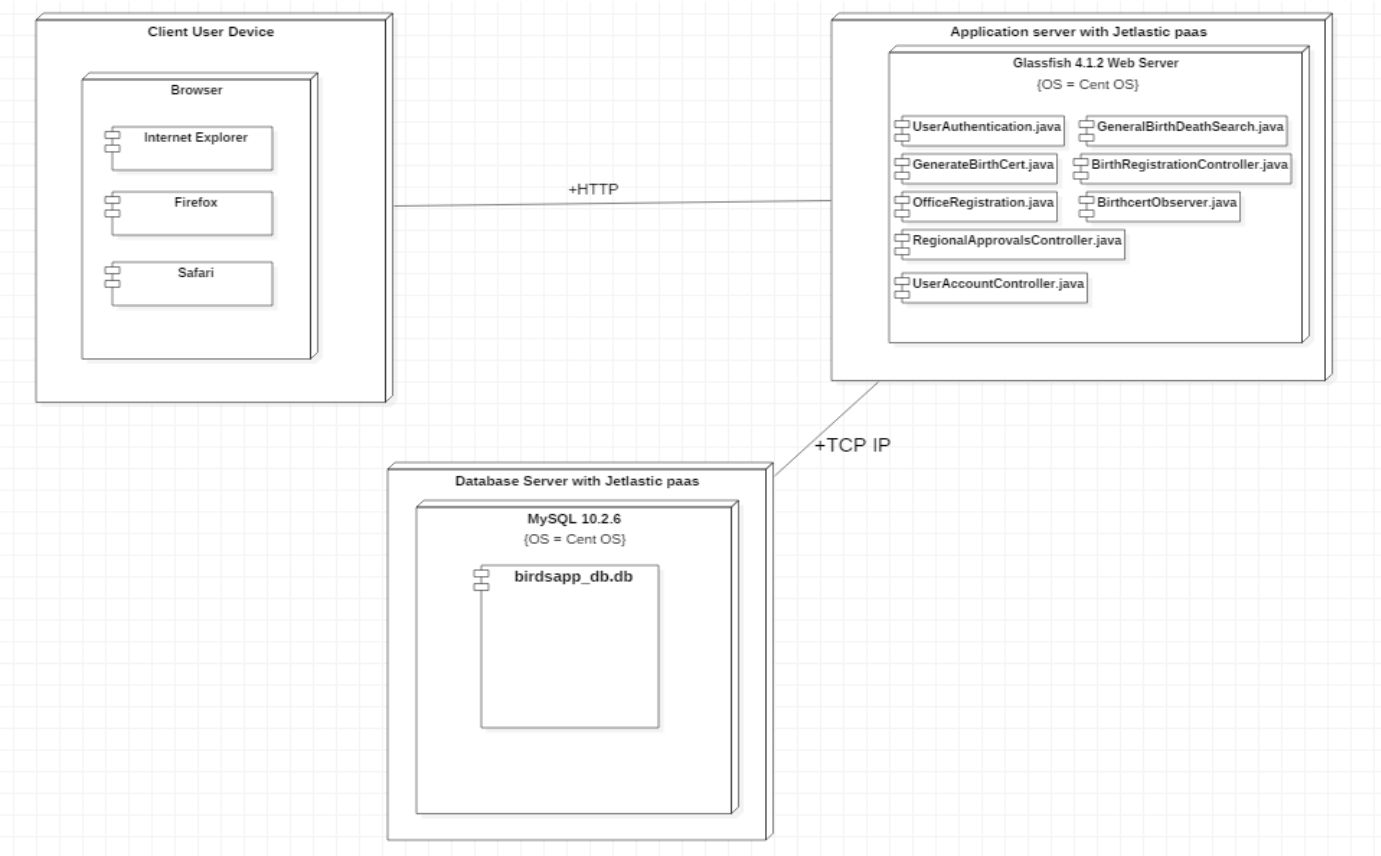


Figure 5.2a: Deployment Diagram regarding hardware configuration of source code

5.3 List of Classes and functions

In this section, we shall provide the class and function names implemented by the team in this project. The project is quite huge, so we shall be focusing on the main functional parts of the program only.

- **User Authentication**

Class responsible for authenticating a system user before allowing access into the system. It also checks if a user has an active session before allowing access to certain URLs. Finally, it handles system user logout and clearing of all sessions.

public String authenticateUser()
public void isLoggedIn()
public String logOutUser()

- **User Account Controller**

Class responsible for handling all user management functionalities.

public void searchAccount()
public void activateUserAccount()
public void deActivateUserAccount()
public void changePassword()
public void resetPasswordChageFields()

- **System User Controller**

Class responsible for adding a new system user as well as amending user details

public void searchUser()
public void saveNewUser()
public void updateUser()
public void rowSelectData()
public void resetButton()

- **Birth Registration Controller**

Class responsible for handling all the registration of births into the system

public String generateBirthRegistryId()
public void saveBirthRegistryDetails()
public void checkBeforeSave()
public void saveBirthRegistrationBtn()
public void resetButton()
public void setViewOnInformantRelation(ValueChangeEvent event) throws ParseException

- **Deceased Registration Controller**

Class responsible for handling all the registration of deaths into the system

public String generateDeceasedRegistryId()
public void deceasedRegistration()
public void checkBeforeSave()
public void saveButton()
public void resetButton()
public void setViewOnDeceasedBurialStatus(ValueChangeEvent event) throws ParseException

- **Certificate Request Controller**

Class that helps Registrar to make a request for a birth or death certificate for clients

public void searchBirthDetails()
public void searchDeathDetails()
public void resetEntry()
public void sendBirthCertRequest()
public void cancelBirthCertRequest()
public void sendDeathCertRequest()
public void cancelDeathCertRequest()
public void fetchBirthDetails()
public void fetchDeathDetails()

- **Regional Approvals Controller**

Class that holds all the functionalities to assist Regional Admin make approvals to all requests

public void approveBirthDetails()
public void approveDeathDetails()
public void fetchBirthDetails()
public void fetchBirthCertRequestDetails()
public void fetchDeathDetails()
public void fetchDeathCertRequestDetails()
public void cancelRequest()
public int noOfRegBirths()
public int noOfRegDeaths()
public int noOfRegBirthCertRequests()

public int noOfRegDeathCertRequests()
public void loadBirths()
public void loadDeceased()
public void loadBirthCerts()
public void loadDeathCerts()
public void printBirthCert()
public void printDeathCert()
resetDistrictSelected()

- **District Approvals Controller**

Class that holds all the functionalities to assist District Admin make approvals to all requests

public SystemUser getSystemUser()
public void approveBirthDetails()
public void approveBirthCertRequest()
public void approveDeathDetails()
public void approveDeathCertRequest()
public void fetchBirthDetails()
public void fetchBirthCertRequestDetails()
public void fetchDeathDetails()
public void fetchDeathCertRequestDetails()
public void cancelRequest()

- **General Search**

Class holding the functionalities to allow system user to make a general search in the database

public void searchBirthDetails()
public void resetBirthSearch()
public void closeBirthSearch()
public void fetchBirthRowData()
public void ammendBirthDetails()
public void searchDeceasedDetails()
public void resetDeceasedSearch()
public void closeDeceasedSearch()
public void fetchDeceasedRowData()

- **Birth Registration Observer**

Class responsible for holding the observers for the birth registration model in system

public void onBirthDetailsCreate(@Observes @Create ChildBirthDetail birthDetail)
public void onRegistrarBirthDetailsUpdate(@Observes @Update @Registrar ChildBirthDetail birthDetail)
public void onDistrictAdminBirthRegApproval(@Observes @Update @DistAdmin ChildBirthDetail birthDetail)
public void onRegionalAdminBirthRegApproval(@Observes @Update @RegAdmin ChildBirthDetail birthDetail)

- **Deceased Registration Observer**

Class responsible for holding the observers for the deceased registration model in system

public void onDeceasedDetailsCreate(@Observes @Create DeceasedDetail deceasedDetail)
public void onDistrictAdminBirthRegApproval(@Observes @Update @DistAdmin DeceasedDetail deceasedDetail)
public void onRegionalAdminBirthRegApproval(@Observes @Update @RegAdmin DeceasedDetail deceasedDetail)

- **Birth Certificate Observer**

Class responsible for holding the observers for the birth cert requests model in system

public void onBirthCertRequestCreate(@Observes @Create BirthCertRequest birthCertRequest)
public void onBirthCertApprovePrint(@Observes @Update @RegAdmin BirthCertRequest birthCertRequest)

- **Death Certificate Observer**

Class responsible for holding the observers for the deceased registration model in system

public void onDeathCertRequestCreate(@Observes @Create DeathCertRequest deathCertRequest)
public void onDeathCertApprovePrint(@Observes @Update @RegAdmin DeathCertRequest deathCertRequest)

- **Certificate Template (abstract class)**

Defines the framework that other classes shall use to generate a birth or death certificate

public final void printCertificate(T t, S s, C c, F f)
public abstract List loadPrintDetails(T cl, S su)
public abstract void updateDatabase(T bcr, F f)
public abstract void renderCertificate(C c)

- **Generate Birth Cert**

Class that extends the certificate template to generate a birth certificate for clients

public List loadPrintDetails(BirthCertRequest cl, SystemUser su)
public void updateDatabase(BirthCertRequest bcr, BirthCertRequestFacade bcrf)
public void renderCertificate(Collection c)

- **Generate Death cert**

Class that extends the certificate template to generate a death certificate for clients

public List loadPrintDetails(DeathCertRequest cl, SystemUser su)
public void updateDatabase(DeathCertRequest dcr, DeathCertRequestFacade dcrf)
public void renderCertificate(Collection c)

5.4 Screenshots of all tables

Below are the screenshots of all tables used in our software system. In all, we have 13 active database tables used within the system though there are more. All our databases are created and managed within the SQLyog Ultimate edition GUI application.

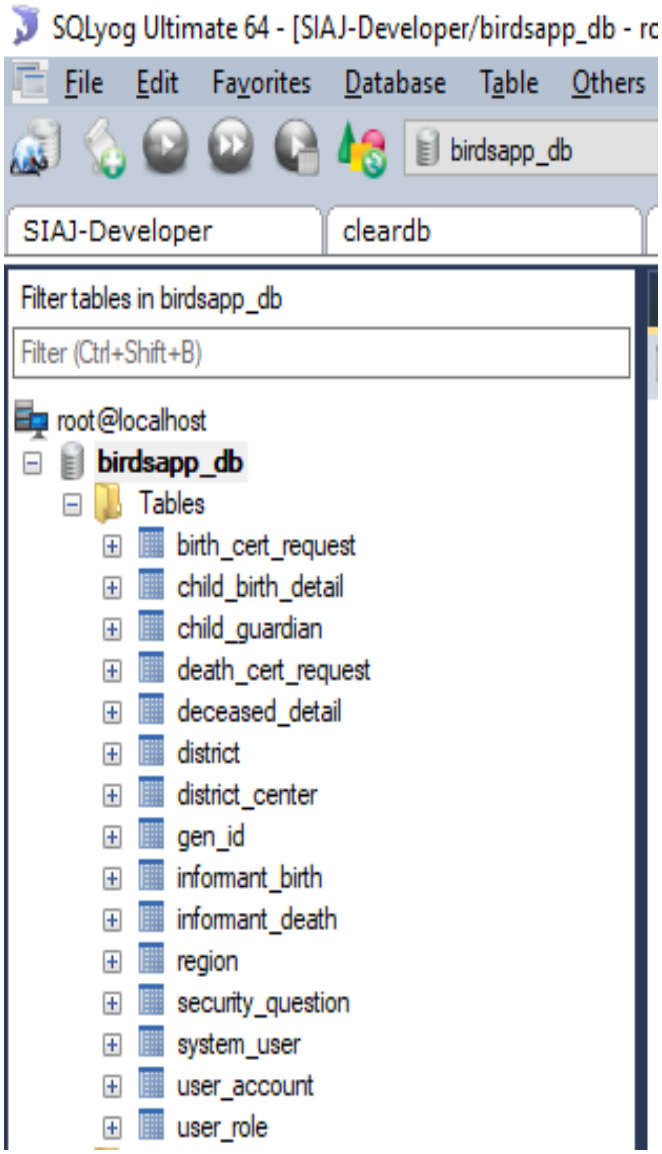


Figure 5.4a: All database tables shown within SQLyog GUI application

Screenshots of key tables with system data

Table: System_User

Purpose: To keep records of the various users in the system

#	system_user_id	last_name	middle_name	first_name	username	password	acco
1	06c16473-e690-4500-a2f9-c8b6efa26fae	Ibrahim	Iddrisu Alhassan	IBRAHIM Iddrisu Alhassan	iaibrahim	edf8bb84bdc6f0adfc2a17c74bf5923fd53f7...	Active
2	0df4d41b-2d87-4cbb-9180-f6f04bfa0a22	Iddrisu	Amal Wunzooya	IDDRISU Amal Wunzooya	awiddrisu	2c0a6aec5c8d5bf3b79602d449f18a26497...	Active
3	b3bb4001-0f06-4730-9199-abf2225e9bc6	Peter	Jamie Wunam	PETER Jamie Wunam	jpeter	2654358fea5bb0ac5e3b9b15064ec6f3bed...	Active
4	c81f9331-d8c1-479f-884b-b6006c19f081	Ahmed	Sibawaihi Shani	AHMED Siba Shani	ssahmed	f4237ace363afd6a24c719e6969fbc95472f...	Active
5	first#siaj#name143	Iddrisu	Abdul-Jalil	Sibdow	sibdow	897479897c736c41b31bf587644ca8d8fa4...	Active

Table: User_Role

Purpose: Holds the data for the different user groups we have in the system

role_id	role_name	updated	deleted
01	Regional Administrator	NO	NO
02	District Administrator	NO	NO
03	Registrar	NO	NO
(NULL)	*(NULL)*	NO	NO

Table: Region

Purpose: For storing the various Regions within the service area

region_id	region_name	updated	deleted
030	Greater Accra	NO	NO
031	Western	NO	NO
032	Ashanti	NO	NO
033	Central	NO	NO
034	Eastern	NO	NO
035	Brong Ahafo	NO	NO
036	Volta	NO	NO
037	Northern	NO	NO
038	Upper East	NO	NO
039	Upper West	NO	NO

Table: District

Purpose: Holds the various districts located within the various regions

district_id	district_name	region	primary_contact	seconday_contact	physical_location	updated	deleted
0324098	Test12345	032	3063212547	1234567890	Kpamberu Estates, Tamale	NO	NO
0328099	Sibdow	032	3063212547	1234567890	ABCDEFGHIJ	NO	NO
050001	Regional Office	032	0302584022	0274580235	Behind the Tech Hospital	NO	NO
053310	Amansaman	032	0278564520	0245213021	Opposite the District Cou...	NO	NO
055928	Knust	032	0325454552	0245102365	Inside the Tech Hospital,...	NO	NO

Table: District Center

Purpose: For storing the various offices located within the districts

center_id	center_name	center_type	center_location	regio...	distr...	pr...	seco...	updated	deleted
1209	Afiya Agyei Center	Health Center	Opposite the Main	032	053310	02785	*(NULL)*	NO	NO
1681	Bomso	Mortuary	Behind the Ghanam	032	053310	03256	*(NULL)*	NO	NO
3922	Ayigya Center	Others	Close to the Para	032	053310	03265	*(NULL)*	NO	NO
4258	Ibrahim M	Health Center	S4S 0A2, UoR	032	055928	30678	*(NULL)*	NO	NO
4899	Ayeduase Center	Health Center	Opposite the Ayed	032	053310	03254	*(NULL)*	NO	NO
5250	Aprukusu Center	Others	Close to the main	032	053310	02456	*(NULL)*	NO	NO
8908	Doodo	Health Center	Kpamberu Estates,	*(NULL)*	050001	30678	*(NULL)*	NO	NO

Table: Gen_Id

Purpose: For keeping track of the birth and death numbers so as to assign unique Ids

gen_table_id	district_code	dist_last_birth_num	dist_last_death_num	updated	deleted
0a6ad254-159c-4d4a-a585-a966b4a1ea40	0324098	0	0	NO	NO
71d1ff61-d989-4703-8d4d-f7cc2c417ed7	0321039	0	0	NO	NO
812ccb72-5f85-427b-875c-cae02b7e2d76	0325086	0	0	NO	NO
819d7961-c05a-44a9-853d-81d7fdc2b8bc	0324757	0	0	NO	NO
84d3777a-bb3d-40d4-bfa7-514f80f913bd	0328099	0	0	NO	NO
acd5448a-5d8e-4d46-81cf-da8d33hsgf5e	053310	11	12	NO	NO
adfsr43e-6ytg-09jk-675t-jkhbg6ggfy81	055928	1	1	NO	NO

Table: Child_Birth_Detail

Purpose: Table for storing the registrations of births

birth_id	surname	othername	full_name	gender	date_of_birth	place_of_birth	town_delivered	guardian
053310-1-2013	Abdul-Hanan	Rayan S.	ABDUL-HANAN Rayan S.	Female	2013-04-09	Hospital	Tamale	45a2b301-40
053310-10-2018	Siba	Muhammad Bashar	SIBA Muhammad Bashar	Male	2018-01-23	Hospital	Tamale	e4362558-63
053310-11-2018	Iddrisu	Sibdow Abdul-Jalil	IDDRISU Sibdow Abdul-Jalil	Male	2018-01-23	Hospital	Tamale	a4cc00fd-df
053310-3-2013	Sudais	Abubakar	SUDAIS Abubakar	Male	2013-04-17	Maternity Home	Kumasi	b531e692-fd
053310-4-2013	Farouk	Humu Salma	FAROUK Humu Salma	Female	2013-04-17	House	Kumasi	c1eb6939-4f
053310-5-2013	Zakaria	Jamie	ZAKARIA Jamie	Female	2012-06-20	Clinic	Tamale	189b70b2-cc
053310-6-2013	Fadil	Rahimatu	FADIL Rahimatu	Female	2013-04-18	Hospital	Kumasi	a93ef670-c8
053310-7-2013	Abdul-Mumin	Sibdow	ABDUL-MUMIN Sibdow	Male	2009-10-09	Hospital	Tamale	a515c20a-e1
053310-8-2013	Mumuni	Abdul-Rahim	MUMUNI Abdul-Rahim	Male	2003-12-18	Hospital	Accra	f9a86a5f-f7
053310-9-2013	Alhassan	Kemi	ALHASSAN Kemi	Female	2013-04-03	Hospital	Accra	6a984897-98
055928-1-2013	Yamusah	Abubakar	YAMUSAH Abubakar	Male	1988-02-10	House	Tamale	f93e82e8-50

Table: Child_Guardian

Purpose: Table for storing the guardian/parent information of the birth registration details

birth_id	surname	othername	full_name	gender	date_of_birth	place_of_birth	town_delivered	guardian
053310-1-2013	Abdul-Hanan	Rayan S.	ABDUL-HANAN Rayan S.	Female	2013-04-09	Hospital	Tamale	45a2b301-40
053310-10-2018	Siba	Muhammad Bashar	SIBA Muhammad Bashar	Male	2018-01-23	Hospital	Tamale	e4362558-63
053310-11-2018	Iddrisu	Sibdow Abdul-Jalil	IDDRISU Sibdow Abdul-Jalil	Male	2018-01-23	Hospital	Tamale	a4cc00fd-df
053310-3-2013	Sudais	Abubakar	SUDAIS Abubakar	Male	2013-04-17	Maternity Home	Kumasi	b531e692-fd
053310-4-2013	Farouk	Humu Salma	FAROUK Humu Salma	Female	2013-04-17	House	Kumasi	c1eb6939-4f
053310-5-2013	Zakaria	Jamie	ZAKARIA Jamie	Female	2012-06-20	Clinic	Tamale	189b70b2-cc
053310-6-2013	Fadil	Rahimatu	FADIL Rahimatu	Female	2013-04-18	Hospital	Kumasi	a93ef670-c8
053310-7-2013	Abdul-Mumin	Sibdow	ABDUL-MUMIN Sibdow	Male	2009-10-09	Hospital	Tamale	a515c20a-e1
053310-8-2013	Mumuni	Abdul-Rahim	MUMUNI Abdul-Rahim	Male	2003-12-18	Hospital	Accra	f9a86a5f-f7
053310-9-2013	Alhassan	Kemi	ALHASSAN Kemi	Female	2013-04-03	Hospital	Accra	6a984897-98
055928-1-2013	Yamusah	Abubakar	YAMUSAH Abubakar	Male	1988-02-10	House	Tamale	f93e82e8-50

Table: Informant_Birth

Purpose: Storing the informant details of a birth registration

informant_birth_id	informant_name	relationship	id_type	id_num	residence	updated	deleted
043e3ff0-0eec-458f-a73e-188d63845d5a	AHMED Siba Wunnam	Father	*(NULL)*	*(NULL)*	*(NULL)*	NO	NO
29ba3bb0-c4bb-4603-9fab-b4ecb04f90fd	*(NULL)*	Father	*(NULL)*	*(NULL)*	*(NULL)*	NO	NO
2ad68d84-ce2f-4931-8479-5319614ffae0	*(NULL)*	Father	*(NULL)*	*(NULL)*	*(NULL)*	NO	NO
52cc7dd4-ec04-4459-8ea4-5d3db23ab85b	AHMED Siba Wunnam	Father	*(NULL)*	*(NULL)*	*(NULL)*	NO	NO
5d03c12f-3393-468b-a9d3-a446e2d4c0ca	*(NULL)*	Self	*(NULL)*	*(NULL)*	*(NULL)*	NO	NO
5dfa0ead-68d2-4815-b284-458723le2aa7	Iddrisu Sibdow Abdul-Jalil	Other	Voters' Id	HUYT590	Tishigu,Tamale	NO	NO
95500463-605a-4ac0-8b6a-c489876f9c7d	Labaran Adamu Liman	Other	Voters' Id	YTRF768	Accra	NO	NO
a4fafa86-4a87-41f2-8924-459e7e9b5e01	ABDUL-MUMUNI Ibrahim	Father	*(NULL)*	*(NULL)*	*(NULL)*	NO	NO
a5e67826-7364-47db-ab0e-c48e57ff0395	YAKUBU Memunatu	Mother	*(NULL)*	*(NULL)*	*(NULL)*	NO	NO
adc8c56c-1ca8-451d-8649-a9678cb5ea42	*(NULL)*	Mother	*(NULL)*	*(NULL)*	*(NULL)*	NO	NO
ce06cd99-6a97-48d6-ac49-a012ea6d4636	PETER Jamie	Self	*(NULL)*	*(NULL)*	*(NULL)*	NO	NO
f8675a7a-5863-4de0-b3bc-b5d9ce0975c3	MOHAMMED Rahamatulai	Mother	*(NULL)*	*(NULL)*	*(NULL)*	NO	NO

Table: Deceased_Details

Purpose: Table for storing the registrations of deceased persons

deceased_id	surname	othername	deceased_fullname	gender	d...	deceased_residence	nationality	id_type	id_num	date_of_death	age_o
053310-1-2013	Atambiri	Atampoga	ATAMBIRI Atampoga	Female	*(NUL*	Manhyia 7B	Ghanaian	Voters' Id	H3TY90G	2013-04-03	
053310-11-2018	Ibrahim	Abubakar	IBRAHIM Abubakar	Male	*(NUL*	House No.... 31B	Ghanaian	National Id	GH42534	2017-12-05	
053310-12-2018	Try	Test	TRY Test	Female	*(NUL*	Hhdhhhs 7B	Cho	NHIS Card	Hqbgs	1254-08-12	
053310-2-2013	Pamboba	Abubakar	PAMBOBA Abubakar	Male	*(NUL*	Walawale 8B	Ghanaian	Voters' Id	WD456TR	2010-10-20	
055928-1-2013	Alhassan	Ibrahim	ALHASSAN Ibrahim	Male	*(NUL*	Tishigu,T... 14B	Ghanaian	Voters' Id	GH76547	1990-04-09	

Table: Informant_Death

Purpose: Storing the informant details of a deceased registration

informant_death_id	informant_name	informant_address	informant_con...	id_type	id_num	relationship	updated	deleted
29de1eb7-9395-4a0d-ac10-92295f	Hhvvg	Hhshdveh	085755886	National Id	Bggqy	Vvhuh	NO	NO
2c4b6737-15af-4af8-b79a-c9fb7b	Ibrahim Iddrisu	Tishigu,Tamale	0244748913	National Id	GH587TL	Son	NO	NO
5aa013dc-59e8-444d-bd60-bb03e3	Atambire Abi	Ayigya,Kumasi	0248563012	National Id	567GH69	Inlaw	NO	NO
cf883c62-0b64-4528-a9a0-f436ad	Ibrahim	House No. B188, Tish	0265452103	NHIS Card	H263463	Abass	NO	NO
ecb9f98b-40fd-472d-80b3-bacc47	Pamboba Abdul-Mumin	Tamale	0203247851	National Id	GH765TL	Brother	NO	NO

Table: Birth_Cert_Request

Purpose: Used for storing all the request made for a birth certificate

birth_cert_request_id	system_user_id	district	birth_applicant_id	district_approved	regional_approved
1061861f-a71d-41f8-8acd-2bde336dc90c	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310	053310-4-2013	YES	YES
1c663856-4360-4316-8a1a-7854617a8b61	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310	053310-4-2013	YES	YES
1d2136f8-6706-4a98-89c7-d7d90af02cd5	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310	053310-10-2018	YES	YES
21c153d8-03df-4ae7-8f99-674ac551c6d2	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310	053310-1-2013	YES	YES
5400d7d2-d79d-49b9-8376-7730cb50b552	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310	053310-10-2018	NO	NO
5d93c06a-e946-406a-acf6-e01e35d4220b	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310	053310-5-2013	NO	NO
662ba826-f61d-4c0f-99ac-c6255731402a	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310	053310-9-2013	NO	NO
6cbebbc3-93a5-43f5-8ea5-16b61e64ca97	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310	053310-7-2013	YES	NO
6eb1af6d-f112-496a-a0e6-272b3afacdb5	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310	053310-1-2013	YES	YES
7d0be6d9-9463-4afd-896d-a92bbd2fc9f2	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310	053310-8-2013	NO	NO
b7a4fc7d-f4aa-4929-8756-63bfa303a37b	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310	053310-6-2013	YES	YES
bbb1582a-41e7-427d-9274-837b51bff1ca	0df4d41b-2d87-4cbb-9180-f6f04bfa0a22	055928	055928-1-2013	YES	YES

Table: Death_Cert_Request

Purpose: Used for storing all the request made for a death certificate

death_cert_request_id	system_user	deceased_details	district	district_approved	regional_approved	cert_
13c5cfc8-1612-4753-a3b4-6d22194e3035	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310-1-2013	053310	NO	NO	NO
3b5f7412-83e3-4c5e-b453-fc55c1318b2e	0df4d41b-2d87-4cbb-9180-f6f04bfa0a22	055928-1-2013	055928	YES	NO	NO
3f75c7d5-848e-465e-bc86-4f3418445609	0df4d41b-2d87-4cbb-9180-f6f04bfa0a22	055928-1-2013	055928	NO	NO	NO
c96acc49-d8aa-4e18-85c6-dec560f51381	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310-1-2013	053310	YES	YES	YES
d241aacb-5fcd-4977-a453-7bad823760a5	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310-2-2013	053310	YES	NO	NO
d6602ba2-2374-4dd2-a675-8a3faa8e3b22	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310-1-2013	053310	YES	YES	YES
d938f647-8450-4867-bf8f-69d1c7a91763	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310-1-2013	053310	YES	YES	YES
fced437f-9b18-4c0b-9d1b-5031bacb58a2	b3bb4001-0f06-4730-9199-abf2225e9bc6	053310-2-2013	053310	NO	NO	NO

5.5 Link to web application

Complete Source code: https://github.com/Siaj/birdsApp/

Running application: http://node13526-env-6959028.ny-1.paas.massivegrid.net:8080/birds-app

6. Technical Documentation

In this section, we shall provide the technical details about how the software system is designed. We shall state here the programming languages used, reused algorithms and programs and finally we shall discuss the software tools and environments which were used during works on the software system.

6.1 Programming Languages

There were several programming languages used to realize our project. They include:

- **Java**: The java programming language was the main language used for the majority of our software system

Source: https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html

- **JavaScript/jQuery**: These languages was used extensively to help make our user interfaces very responsive and more interactive

Sources:

1. https://www.javascript.com/
2. https://jquery.com/

- **Java Persistent Query Language (JPQL)**: This language was used to perform all the queries to our database. It is more like the popular SQL language we all know

Source: https://docs.oracle.com/html/E13946_01/ejb3_langref.html

6.2 Reused Algorithms and Programs

Our system was so extensive that we could not implement all of its functionalities within the time period that we had, so we reused a lot of algorithms and libraries to help us.

- **Light bootstrap**: An Admin Dashboard template that we used to design the main template for our system.

Source: https://www.creative-tim.com/product/light-bootstrap-dashboard

- **Java Server Faces (JSF 2.2) MVC Framework**: Mainly used to build our server-side user interfaces.

Source: http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

- **JasperReports 6.1.0**: Our reporting system was designed with the help of this library.

Source: https://community.jaspersoft.com/project/jasperreports-library

- **Guava 16.0.1:** This google library was used to realize the encryption we used to secure the user password in the database

Source: https://opensource.google.com/projects/guava

- **PrimeFaces 6.1:** A user interface framework for Java EE that was used for all the data tables in our system as well as the display of messages, either as a normal message or a growl message

Source: https://www.primefaces.org/

- **Atmosphere Runtime 2.4.6:** Client-Server framework for push notifications

Source: https://github.com/Atmosphere/atmosphere

- **GitHub**: A web-based Git repository hosting service. It was used to version control our system as well as provided us with an easy source code management functionality.

URL: https://github.com/

6.3 Software tools and Environment

- **NetBeans IDE 8.2:** The Integrated Development Environment (IDE) that was used to write all the codes required by our software system
- **Glassfish Sever 4.1.1:** The underlying web application server for running application
- **MySQL Database Server:** This is a relational database management system we using to maintain our database
- **iReport:** Used in the design of the look and feel of our reports/certificates
- **SQLyog Ultimate:** The GUI application that works with the MySQL database server. We are using it to model and manage our MySQL databases from a GUI
- **StarUML:** We used this tool for the design of the majority of our UML diagram which includes: Use Cases, Activity, Component, Deployment and Class Diagrams
- **Cacoo:** A cloud-based system we used to model the logical architectural design of our system

7. Acceptance Testing

In this section, we show the results for the functional, robustness and time-efficiency testing for our system.

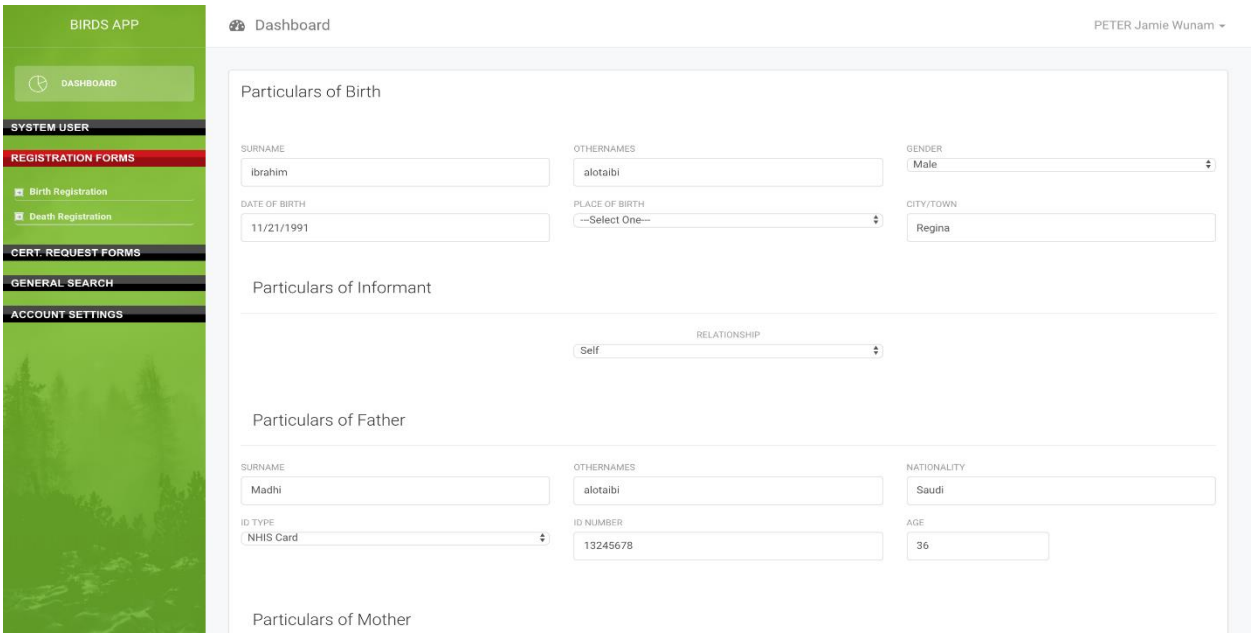
7.1 Functional Testing

We are conducting these test cases to very that our software system performs all the functions correctly as defined in the design specifications.

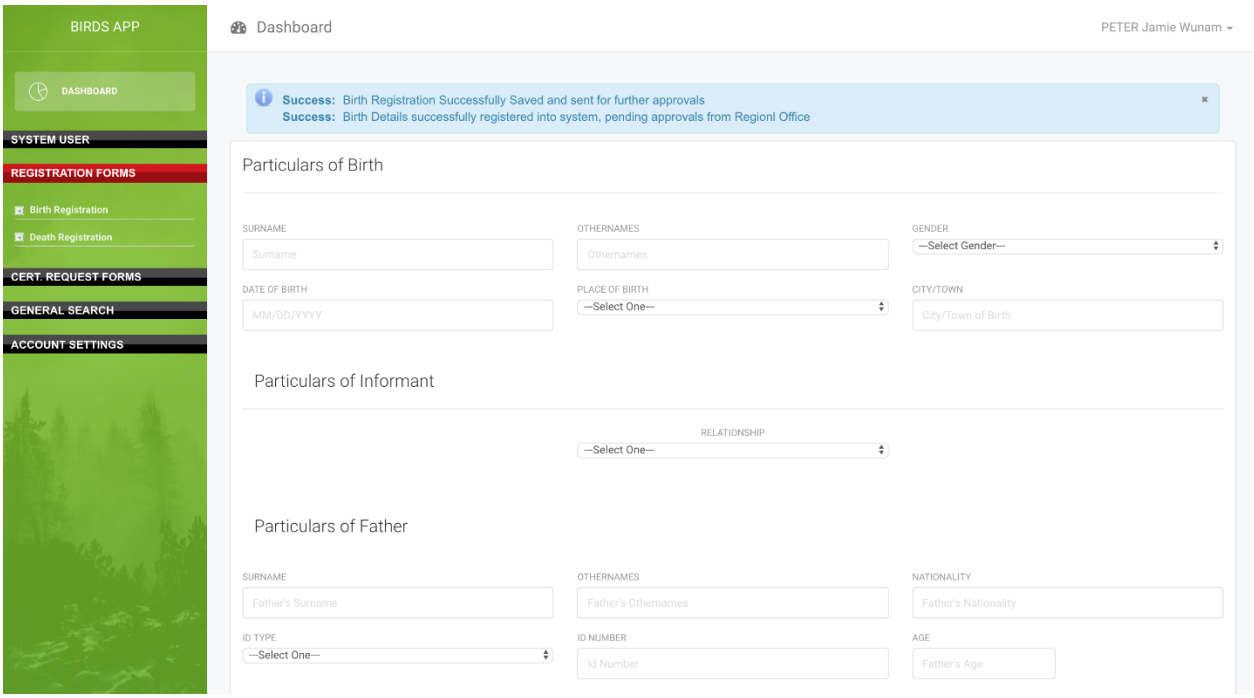
1) Birth Details Registration

Test Case ID	F01 - **Birth Details Registration**
Description	Registrar registering birth details into system
Pre-Condition	Registrar has successfully logged into his/her dashboard
Task Steps	• Registrar clicks on 'Birth Registration Link' on dashboard • Fills out all the required fields • And click on the 'Save' button
Expected Results	Birth details will be saved in the system for further approvals

Input Screenshot: Registrar enters details into registration forms



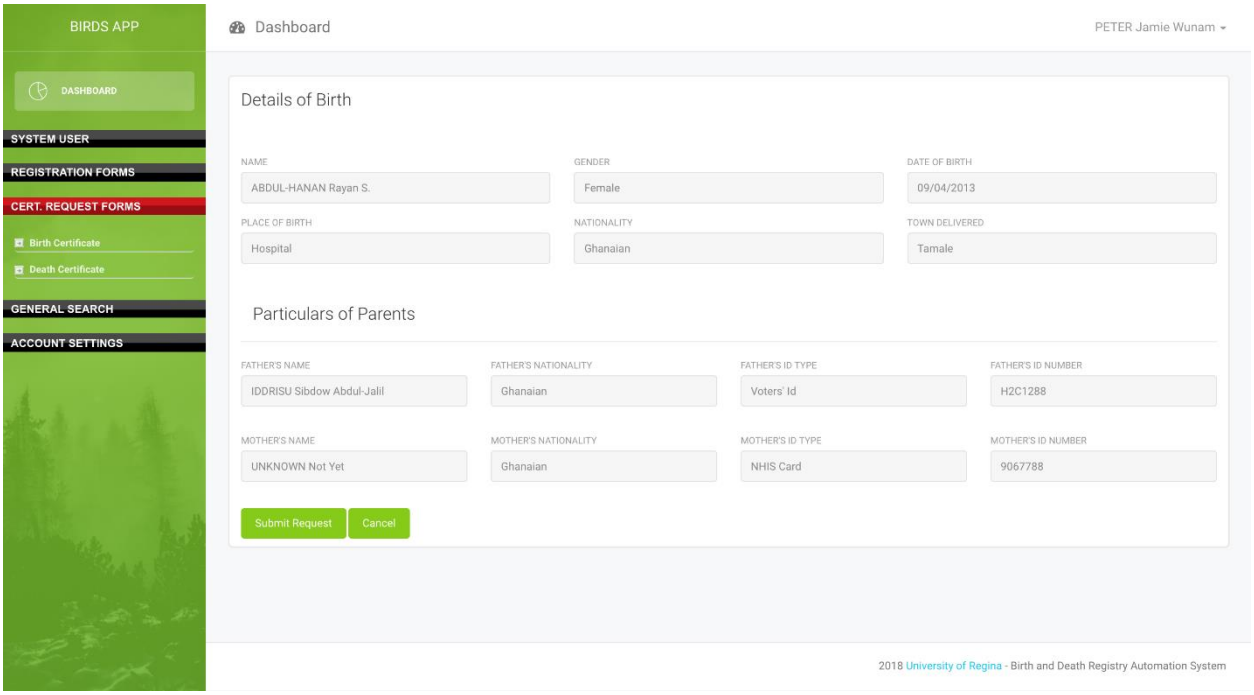
Output Screenshot: Registrar gets a notification for confirmation of submission



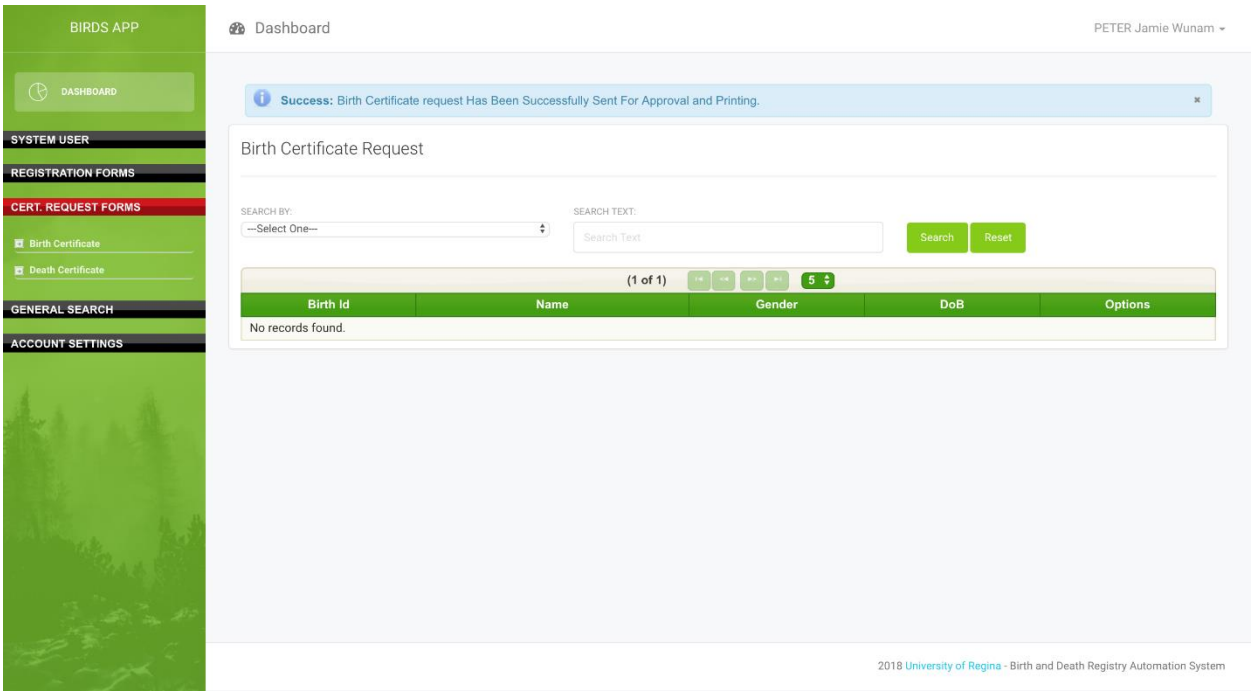
2) Birth Certificate Request

Test Case ID	F02 - **Birth Certificate Request**
Description	Registrar makes a request for a birth certificate for a client
Pre-Condition	Registrar is logged in and client already has birth details in system
Task Steps	• Registrar searches for birth details in the system • Selects that particular birth details • And then click on the 'Submit Request' button
Expected Results	Birth cert request will be saved in the system for further approvals

Input Screenshot: Registrar searches and selects birth registration details, then submits request



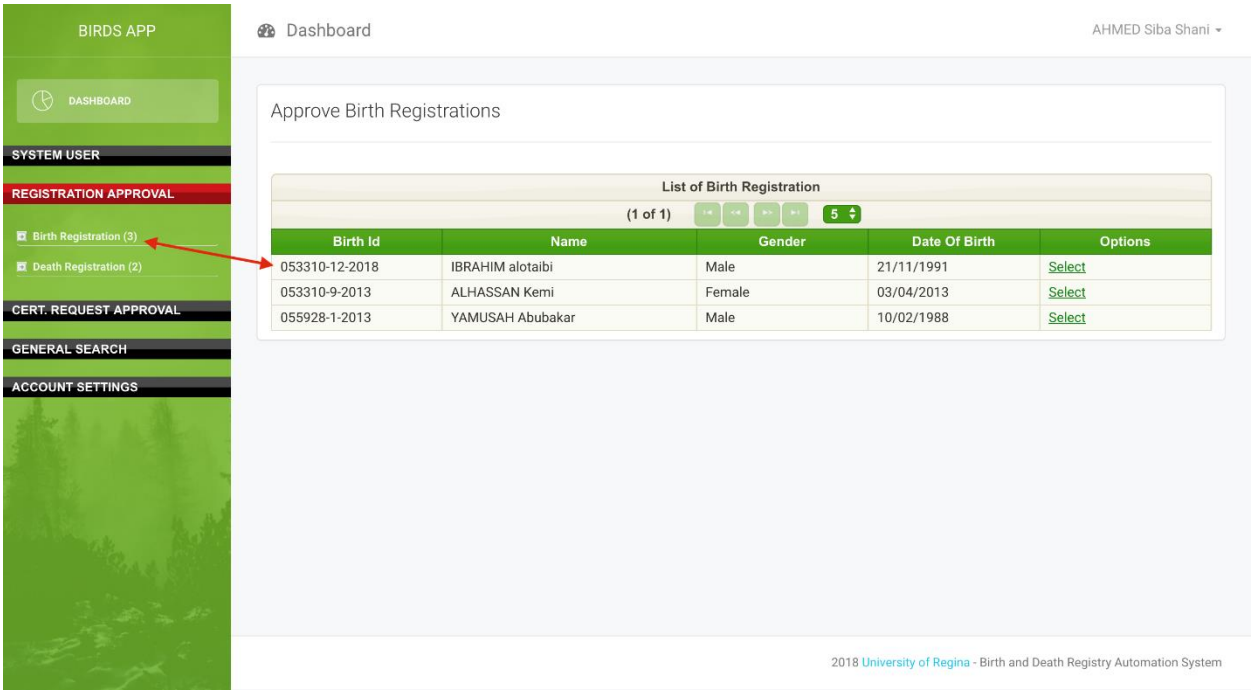
Output Screenshot: Registrar gets a notification for confirmation of submission

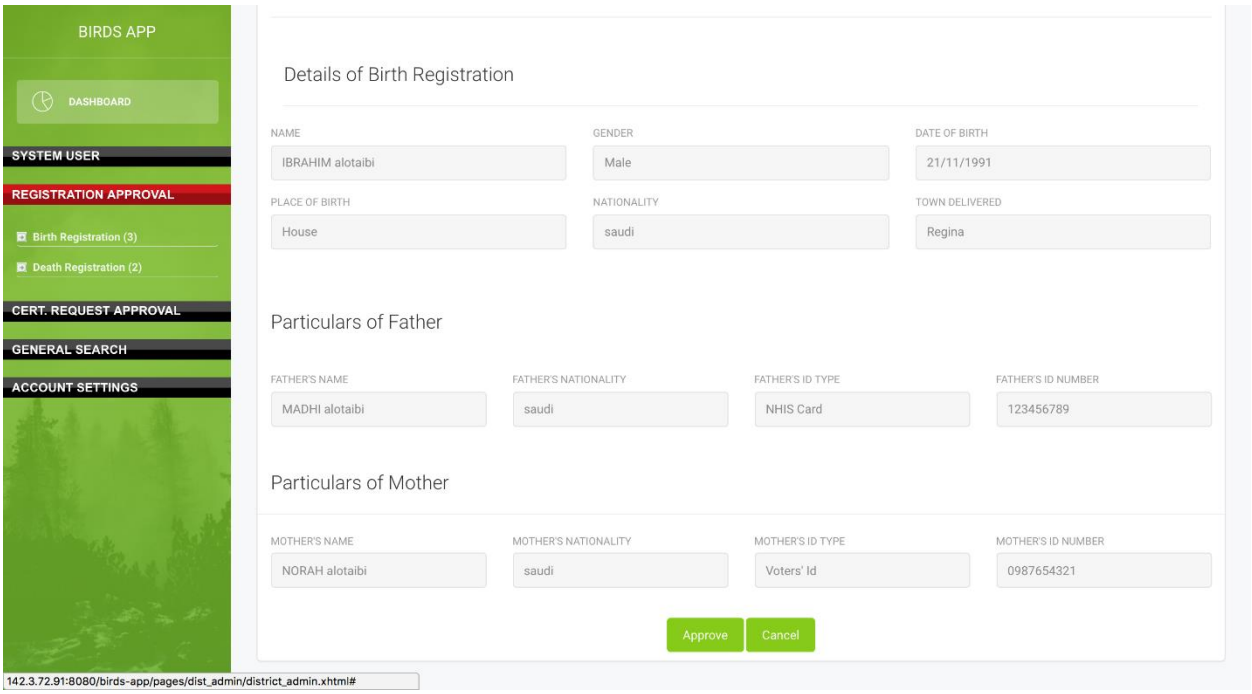


3) District Admin Birth Registration Approval

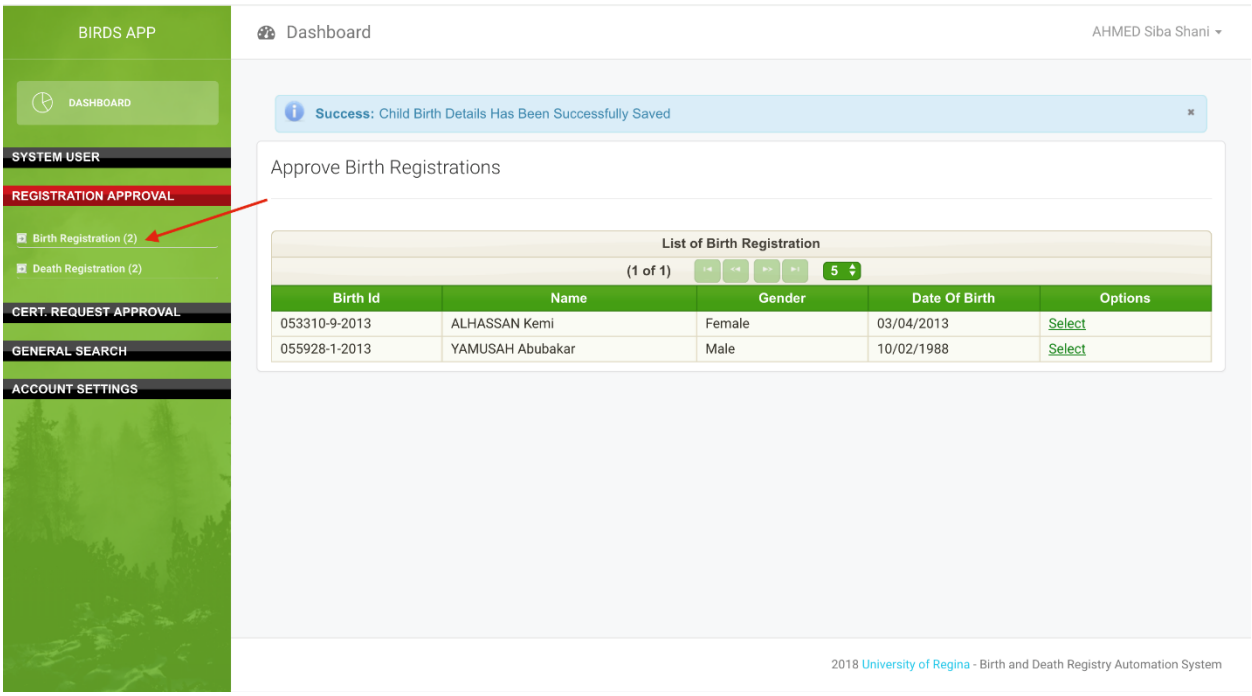
Test Case ID	F03 - **District Admin Birth Registration Approval**
Description	District Admin approves and confirms birth details registration
Pre-Condition	District Admin is logged into dashboard
Task Steps	• District Admin clicks on 'Birth Registration' link under Registration Approval Sub-Category • Birth details is selected • Approve link is clicked
Expected Results	Birth registration details approved and confirmation message shown

Input Screenshot: District Admin searches and selects birth registration details, then approves





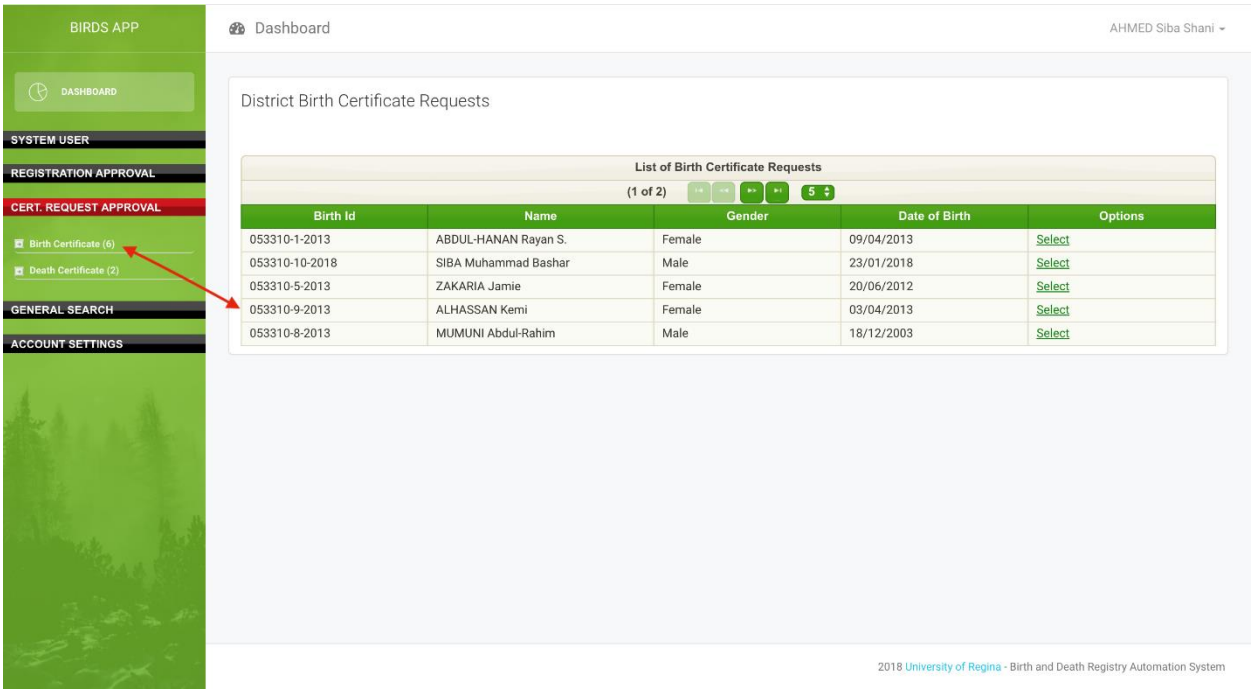
Output Screenshot: District Admin gets a notification for confirmation of submission

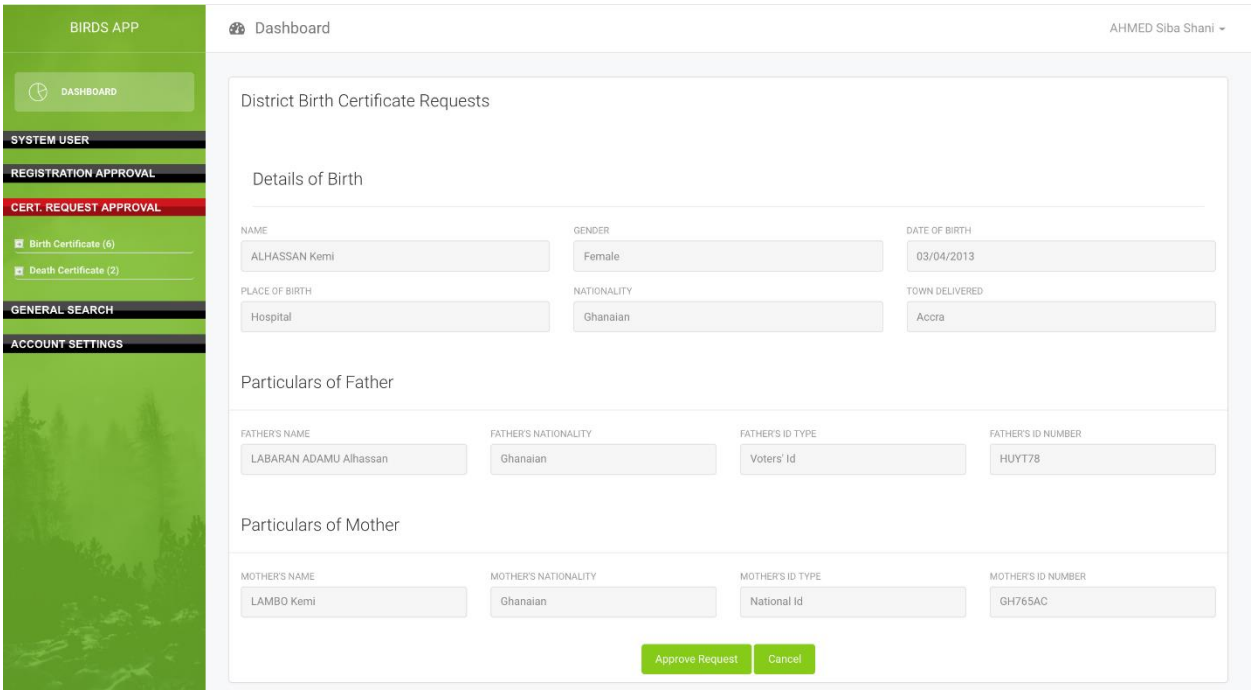


4) Birth Certificate Request Approval

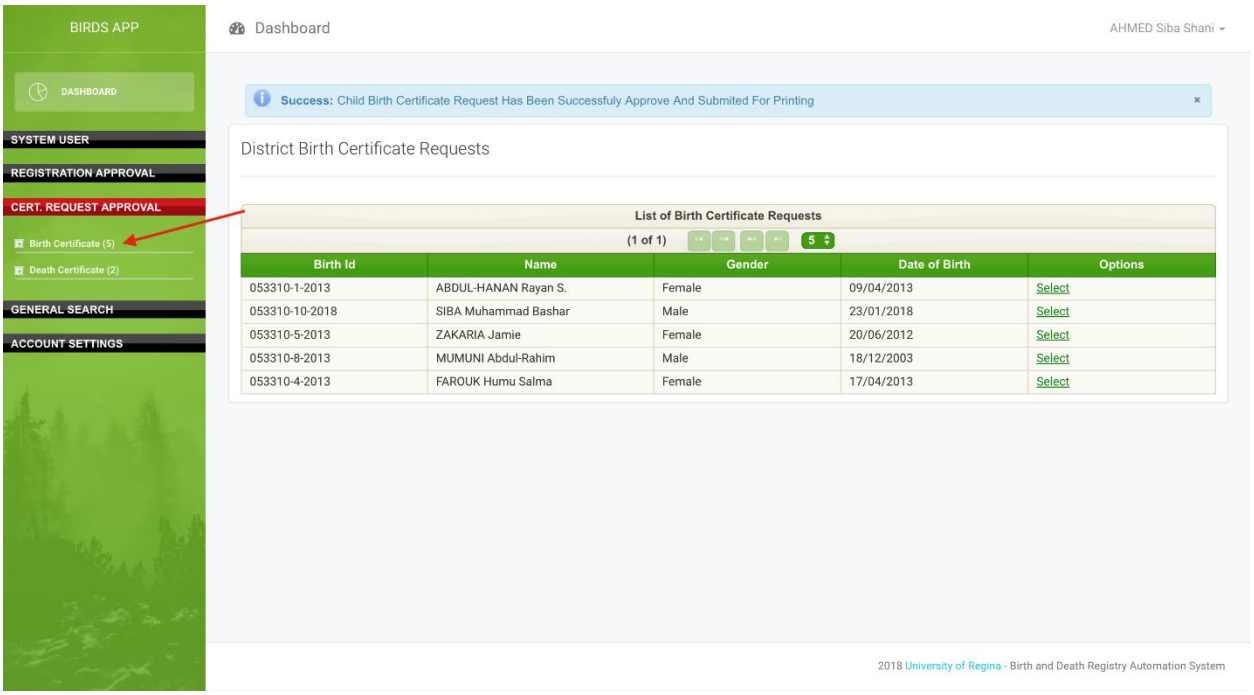
Test Case ID	F04 – **District Admin Birth Certificate Request Approval**
Description	District Admin approves a request for a birth certificate
Pre-Condition	District Admin is logged into dashboard
Task Steps	• District Admin clicks 'Birth Certificate' link under Cert. Request Approval Sub-Category • Certificate request details is selected from list • 'Approve Request' link is clicked
Expected Results	Birth cert request has been confirmed and approved for printing

Input Screenshot: District Admin views list of requests and selects a request





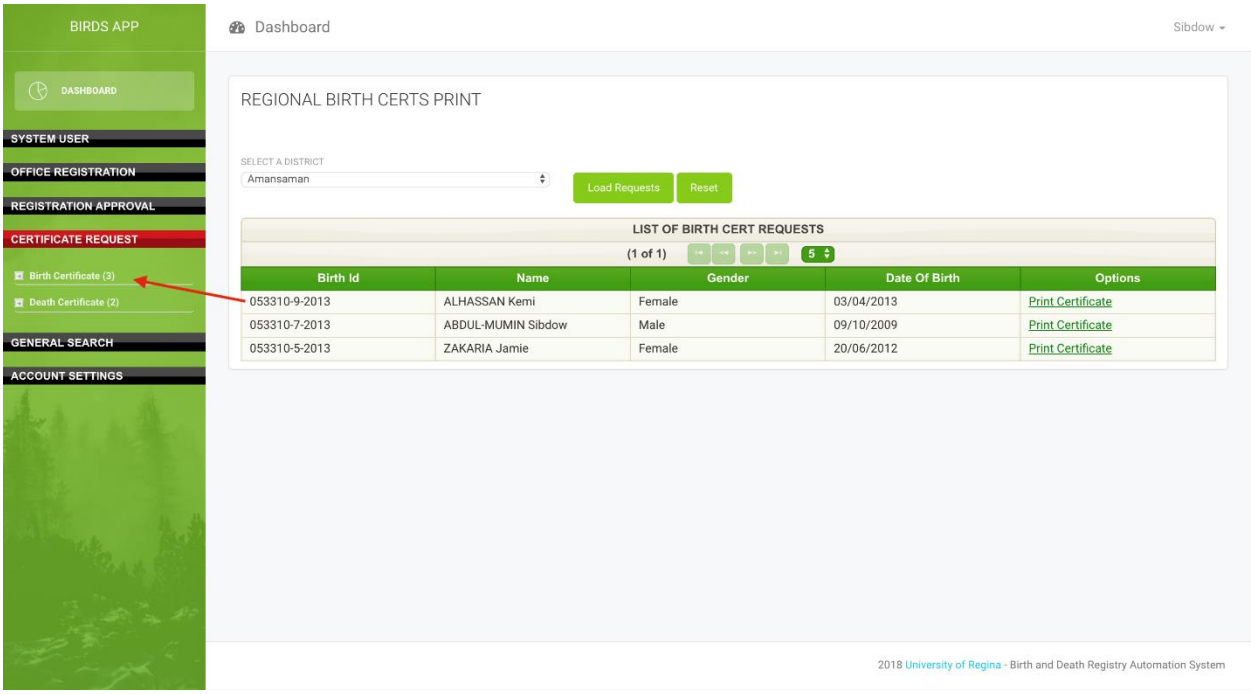
Output Screenshot: District Admin gets a notification for confirmation of approval



5) Regional Admin Birth Certificate approve and print

Test Case ID	F05 - **Regional Admin Birth Certificate approve and print**
Description	Regional Admin approves and generates a birth certificate request
Pre-Condition	Regional Admin is logged into dashboard
Task Steps	• Regional Admin clicks on the 'Birth Certificate' link under the Certificate Request Sub-Category • A particular district is then selected to retrieve requests • Regional Admin clicks on 'Print Certificate' link
Expected Results	Request is approved and a birth certificate is generated in pdf format

Input Screenshot: Regional Admin loads request by district selected



Output Screenshot: A birth certificate is generated with the details of the requester

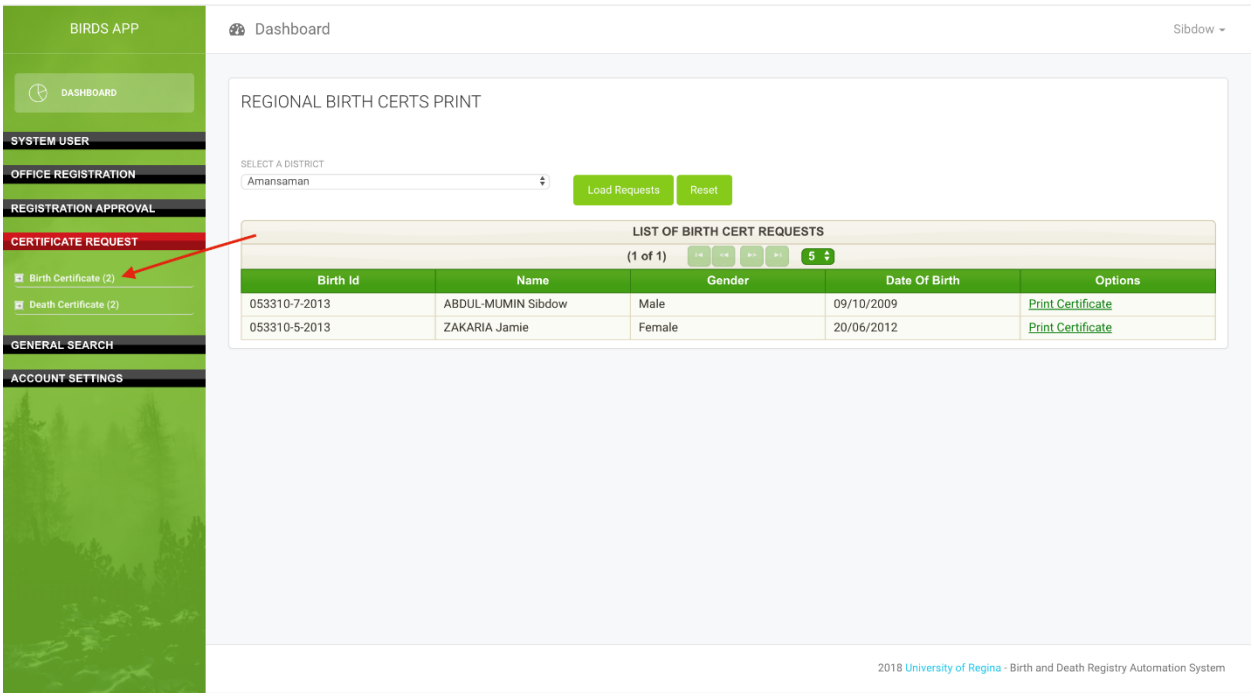
Birth Id: 053310-9-2013

CERTIFIED COPY OF ENTRY IN THE REGISTER OF BIRTHS

Entry Number: 9 Registry: Afiya Agyei Center

CHILD'S NAME (Name in full.Surname first)	ALHASSAN Kemi		Sex: Female
FATHER	Name:	LABARAN ADAMU Alhassan	
	Nationality:	Ghanaian	
MOTHER	Name:	LAMBO Kemi	
	Nationality:	Ghanaian	
Date of Birth	03 April 2013		
Where Born (Write Address in full)	Accra		
Name of Informant	Labaran Adamu Liman		Relationship to the child Other
Date of Registration	27 April 2013		
Sign of Registration	PETER Jamie Wunam		
Margin			

I Iddrisu Sibdow ,Registrar of the Births and Deaths for Ghana do hereby certify that the foregoing is a true copy of the entry NUMBER 9 in the Register of Births for Afiya Agyei Center in the Amansaman Registration District in Ghana, and the Register is now legally in my custody.



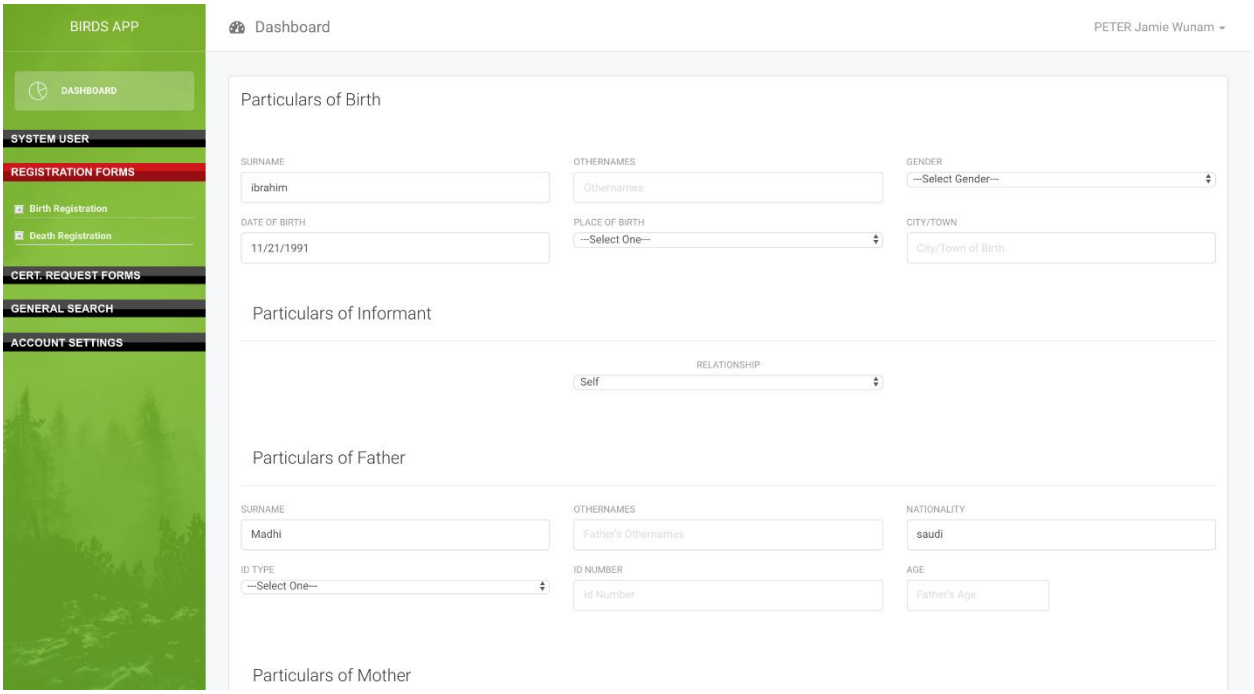
7.2 Robustness Testing

We performed robustness testing to help us detect the vulnerabilities of our software system under an unexpected input or in a stressful environment.

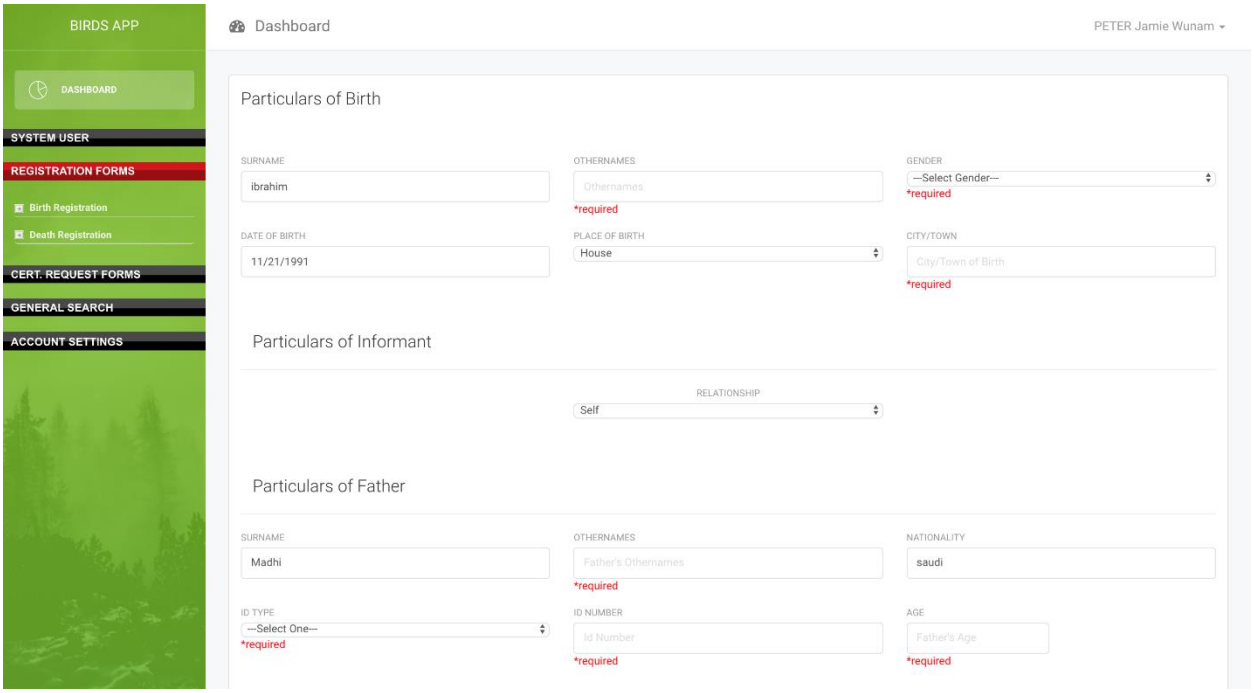
1) Birth Registration with empty fields

Test Case ID	R01 - **Birth Registration with empty fields**
Description	Registrar tries to submit a birth registration with incomplete fields
Pre-Condition	Registrar is logged into dashboard
Task Steps	• Registrar clicks on the 'Birth Registration' link • Enters birth details with some fields incomplete/empty • Then clicks on the 'Save' button
Expected Results	Registrar would be shown error messages on portions of form that are incomplete

Input Screenshot: Registrar fills out form with incomplete information



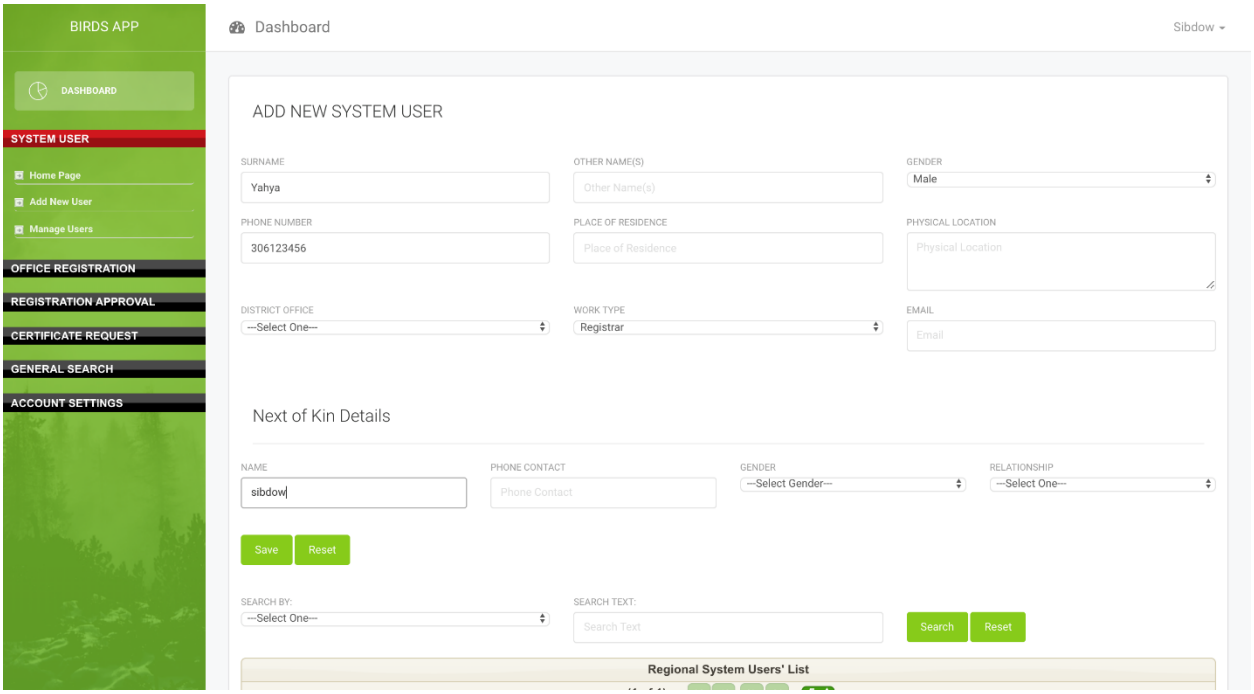
Output Screenshot: Appropriate error messages are shown at the exact places with incomplete information



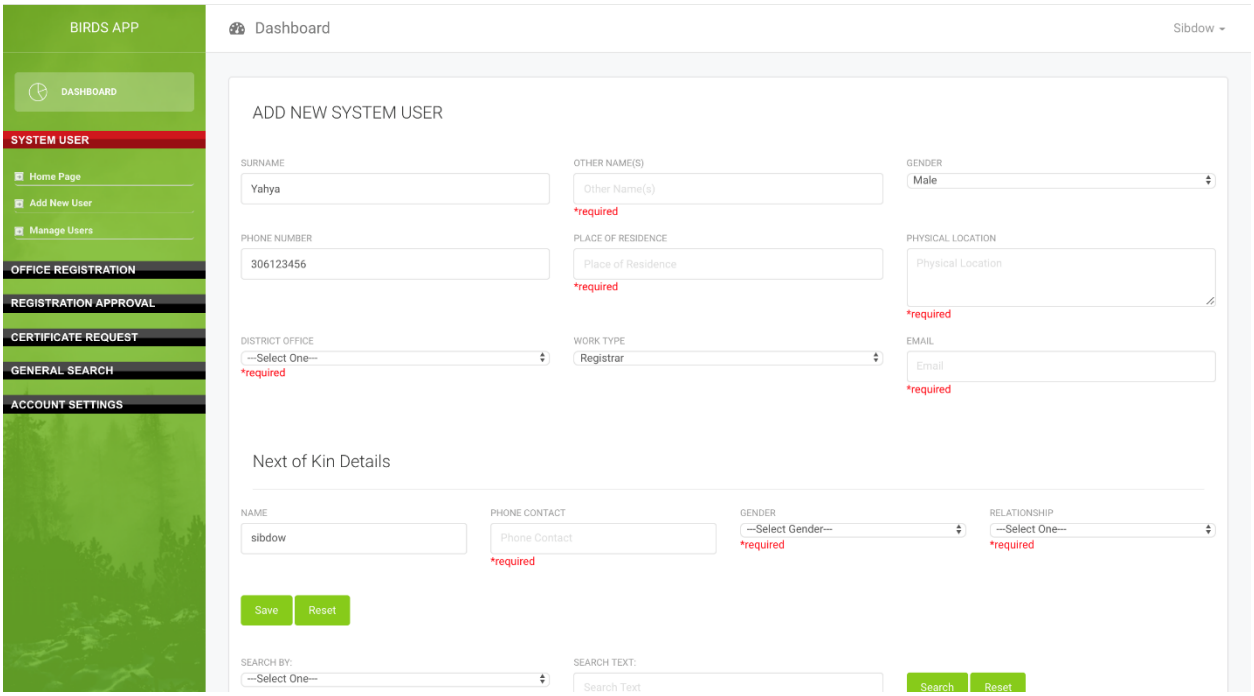
2) Add New System User with empty fields

Test Case ID	R02 - **Add New System User with empty fields**
Description	Regional Admin tries to add a new system user with incomplete fields
Pre-Condition	Regional Admin is logged into dashboard
Task Steps	• Regional Admin clicks on the 'Add New User' link • Enter new user details leaving some form fields incomplete • Then clicks on the 'Save' button
Expected Results	Regional Admin would be shown error messages on portions of form that are incomplete

Input Screenshot: Regional Admin fills out form with incomplete information



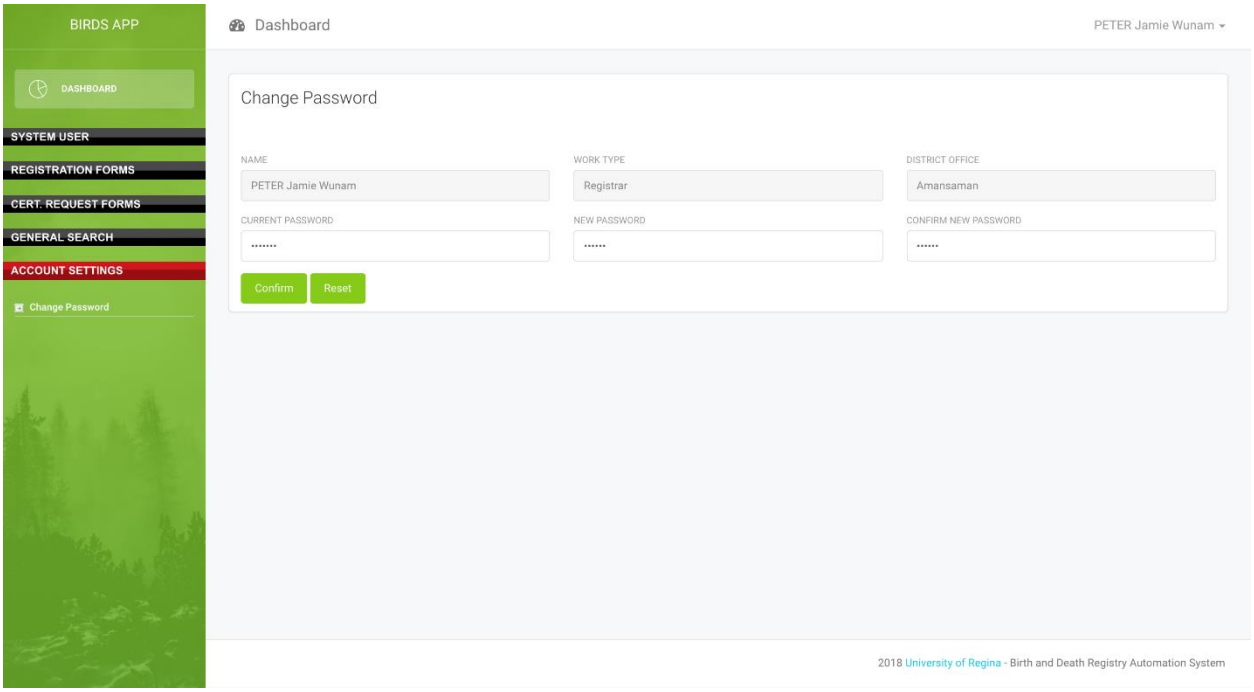
Output Screenshot: Appropriate error messages are shown at the exact places with incomplete information



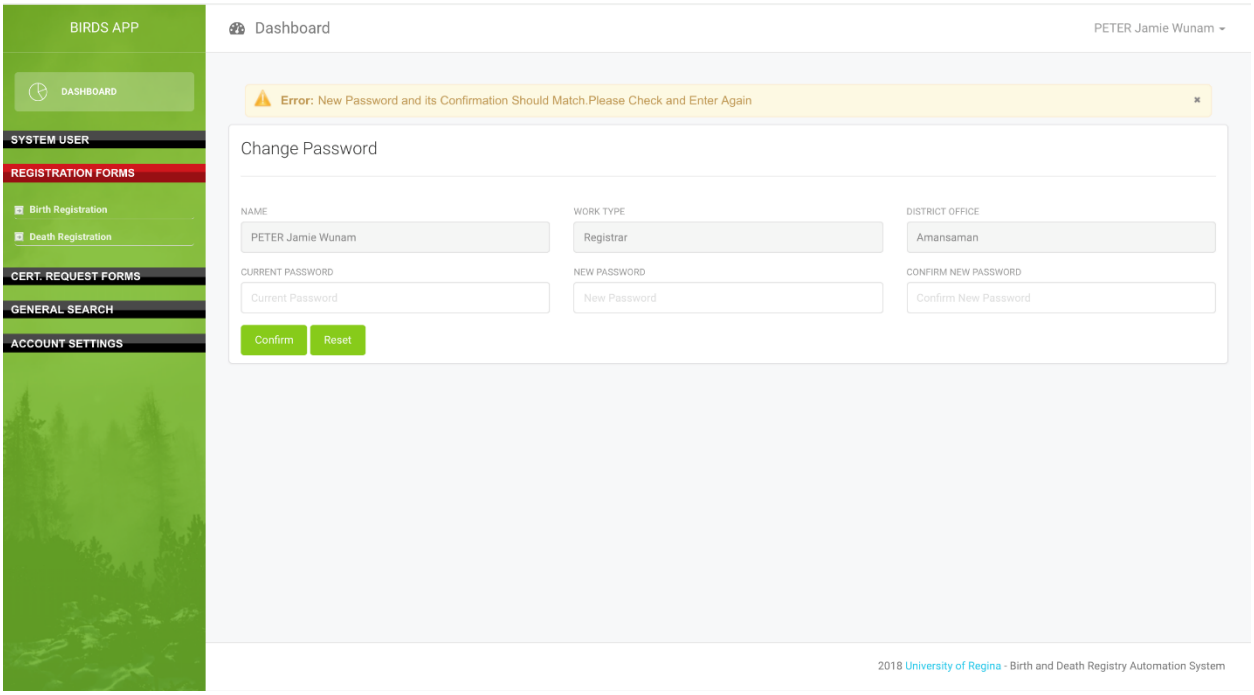
3) Change Password

Test Case ID	R03 - **Change Password**
Description	System user wishes to change current login password details
Pre-Condition	System user is currently logged into dashboard
Task Steps	• User clicks on the 'Change Password' link on the page • Password details are entered with unmatched new passwords • User clicks on the 'Confirm' button
Expected Results	User would be shown an appropriate error message

Input Screenshot: User enters current login details, then unmatched new passwords



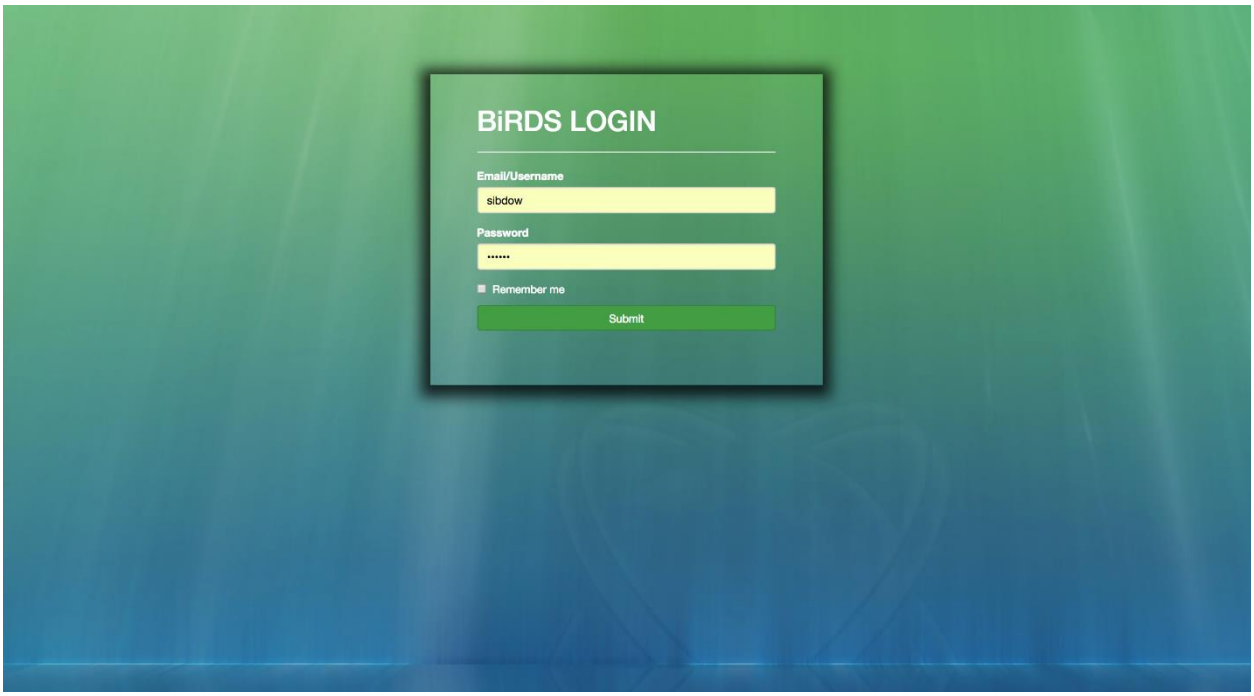
Output Screenshot: Appropriate error messages displayed alerting user of exact mistake made



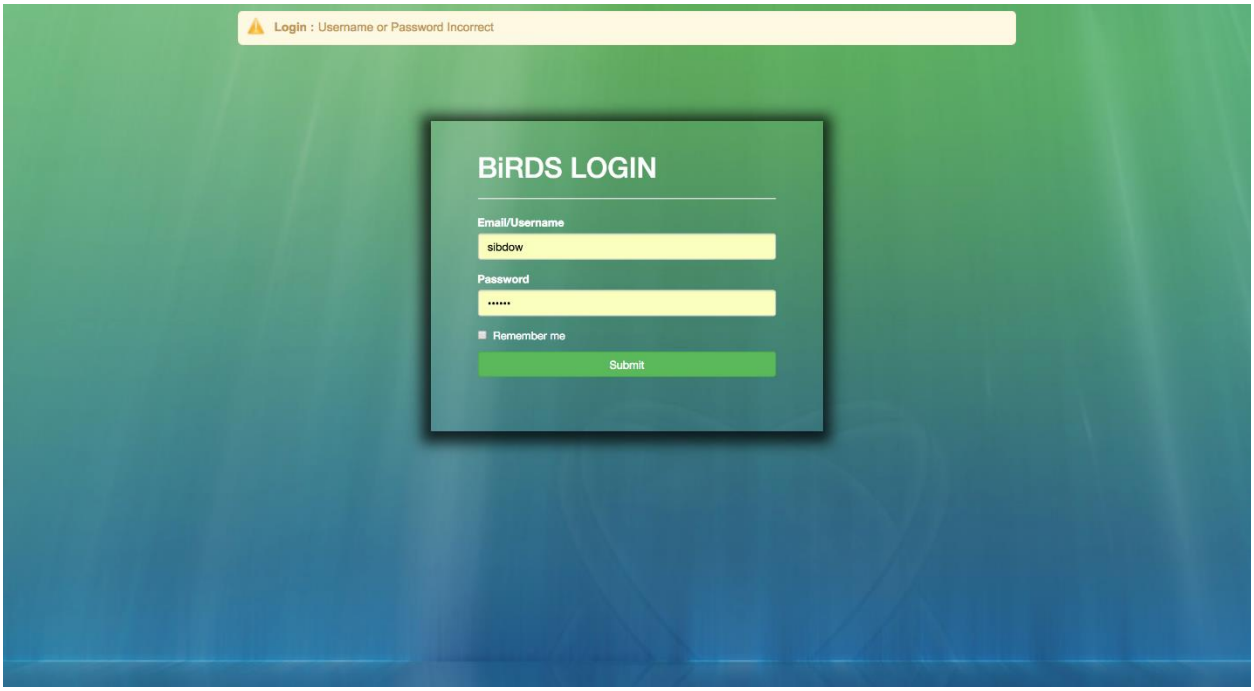
4) Login with wrong credentials

Test Case ID	R04 - **Login with wrong credentials**
Description	User tries to log into system with wrong username and password
Pre-Condition	User already has an account.
Task Steps	• Enters any username and password in login fields • Click on the 'Submit' button
Expected Results	User would be shown error messages

Input Screenshot: User enters wrong credentials into login form



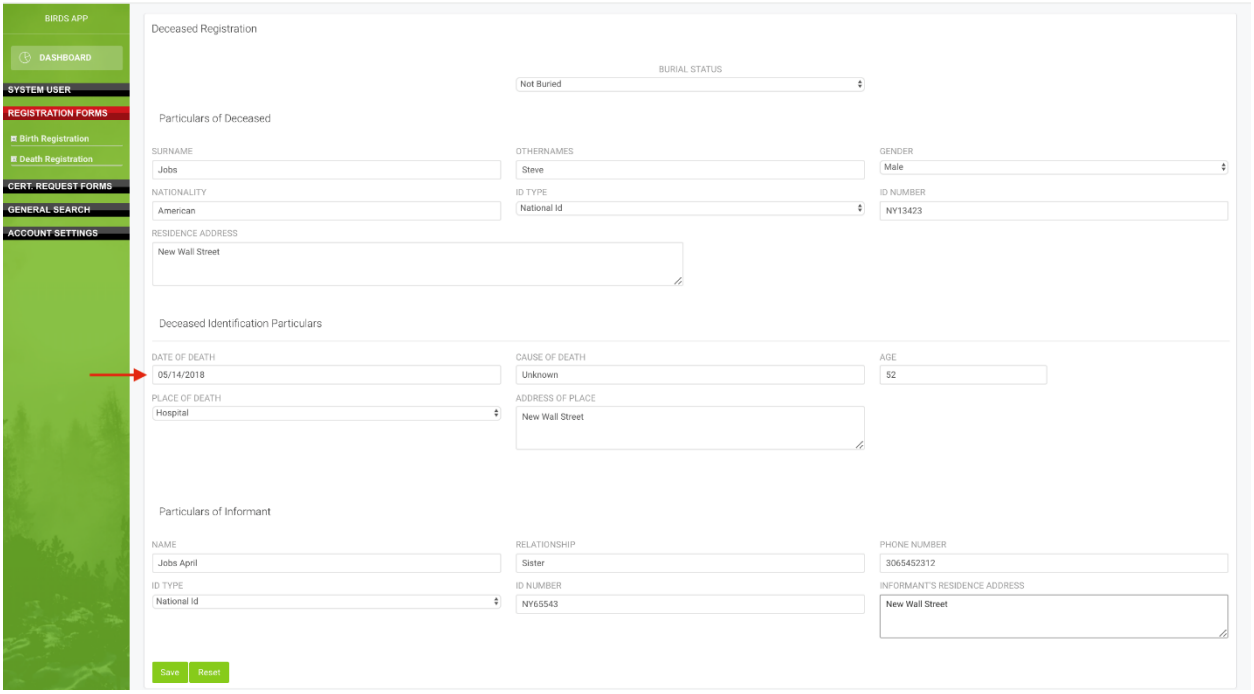
Output Screenshot: Appropriate error messages are displayed to user



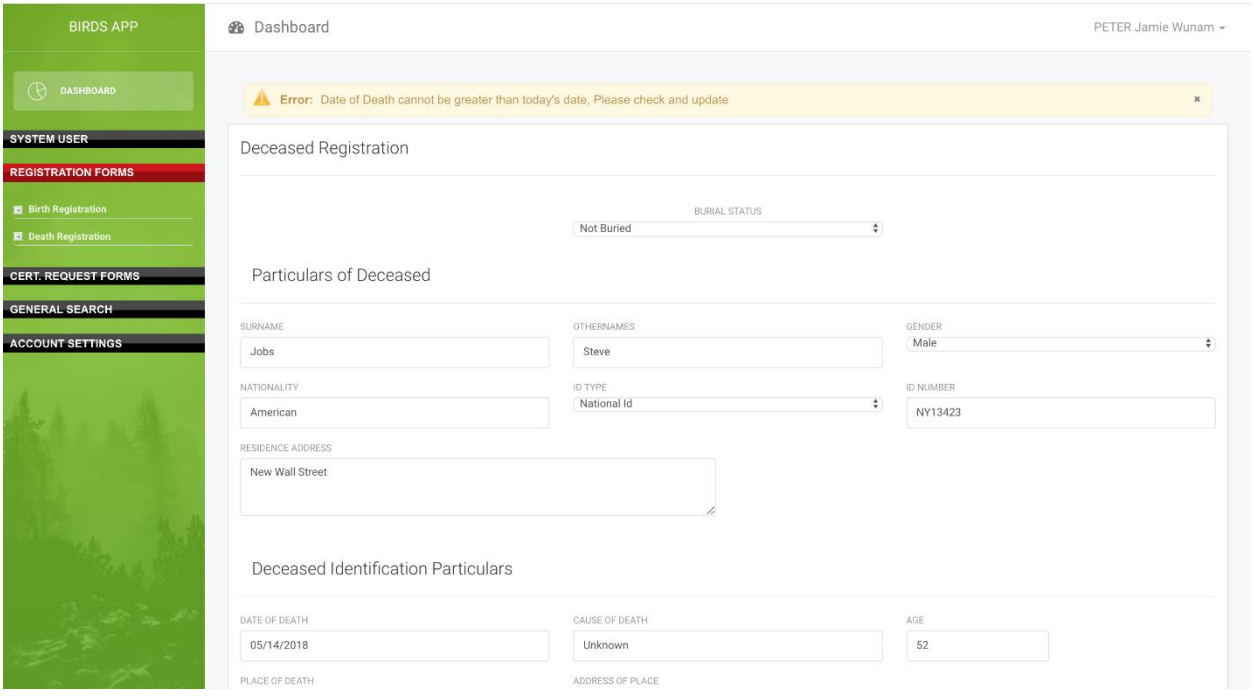
5) Deceased Registration with Inappropriate date

Test Case ID	R05 - **Deceased Registration with Inappropriate date**
Description	Registrar tries to submit a deceased registration details with date of deceased greater than the even the current day of the registration
Pre-Condition	Registrar is logged into system
Task Steps	• Registrar clicks on the 'Death Registration' link on dashboard • All details are entered but with a date of death greater than current date • Registrar then clicks on the 'Save' button
Expected Results	Registrar would be shown an appropriate error messages warning him/her about wrong date input

Input Screenshot: Registrar fills out deceased registration form with wrong date of death



Output Screenshot: Registrar is shown an appropriate error message, warning him/her that date of death cannot be in the future



7.3 Time-Efficiency Testing

We conducted some time-efficiency test on our software system with various tools online. This was to make sure that the system was fast and responsive enough to help user achieve their goals in a more efficient way rather than frustrating them.

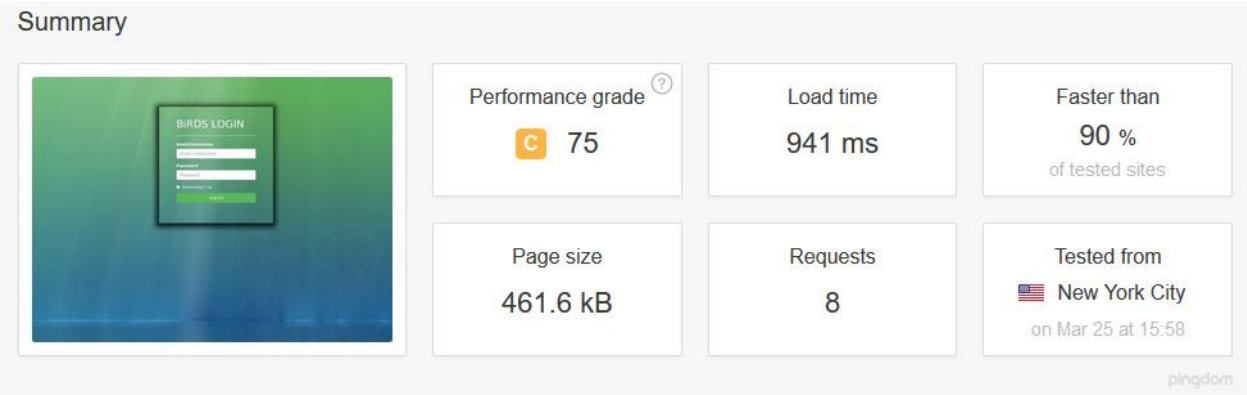
There are multiple online tools for these test, we decided to leverage the services of:

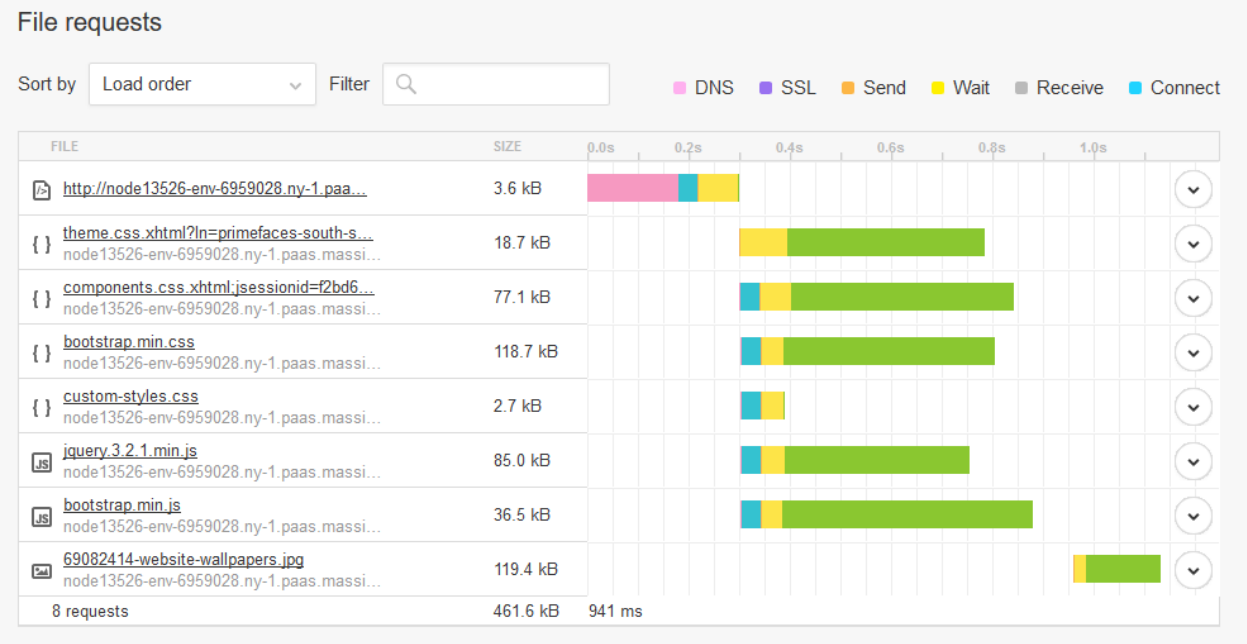
- **Pingdom**
- **GtMatrix**

Pingdom

This is a tool that helped us test the load time of the main and in demand pages of our system. The good thing is that you could choose the location to test from so as to determine how fast your system loads with distance

1) **Landing Page**



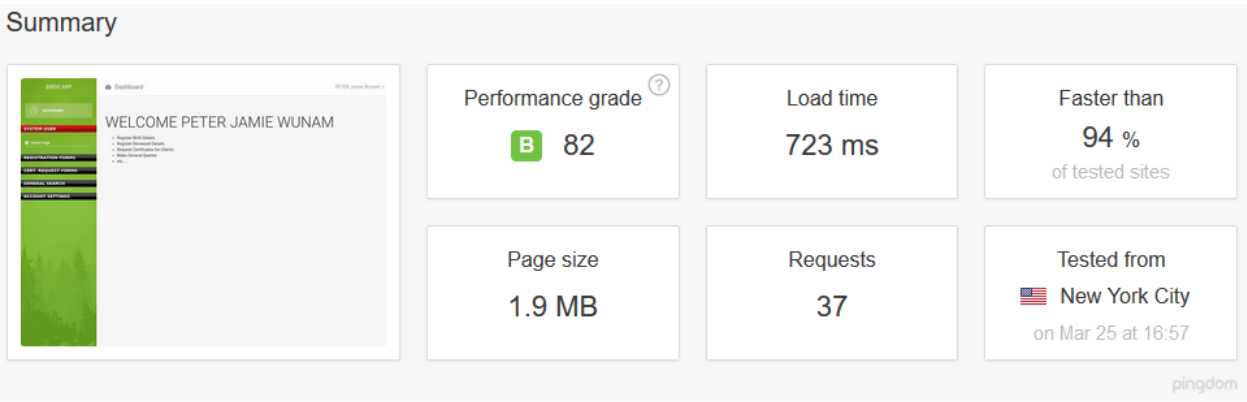


Report Summary

We could see from the above report that our software system has an overall performance of **75%** and loads faster than **90%** of other sites tested with this platform. The file request and overall load time was **941ms** from the New York City test location.

Full Report Link: https://tools.pingdom.com/#!/bSASao/http://node13526-env-6959028.ny-1.paas.massivegrid.net:8080/birds-app/

2) Registrar Main Page



Performance insights

GRADE		SUGGESTION
F	16	Leverage browser caching
C	71	Specify a Vary: Accept-Encoding header
B	85	Remove query strings from static resources
B	86	Avoid bad requests
A	97	Serve static content from a cookieless domain
A	100	Minimize redirects
A	100	Minimize request size
A	100	Specify a cache validator

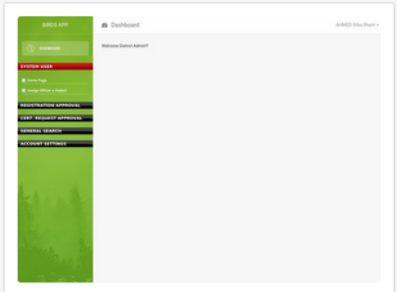
Report Summary

We could see from the above report that our software system has an overall performance of **82%** and loads faster than **94%** of other sites tested with this platform. The file request and overall load time was **723ms** from the New York City test location.

Full Report Link: https://tools.pingdom.com/#!/bKA5b1/http://node13526-env-6959028.ny-1.paas.massivegrid.net:8080/birds-app/pages/registrar/registrar.xhtml;jsessionid=f5f6fc9cf0d22dcabde5a29f5b14

3) District Administrator Main Page

Summary



Performance grade	Load time	Faster than
B 80	3.04 s	52 % of tested sites

Page size	Requests	Tested from
1.9 MB	37	Stockholm on Mar 25 at 16:36

pingdom

Performance insights

GRADE		SUGGESTION
F	14	Leverage browser caching
D	62	Specify a Vary: Accept-Encoding header
C	75	Remove query strings from static resources
B	86	Avoid bad requests
A	100	Minimize redirects
A	100	Minimize request size
A	100	Serve static content from a cookieless domain
A	100	Specify a cache validator

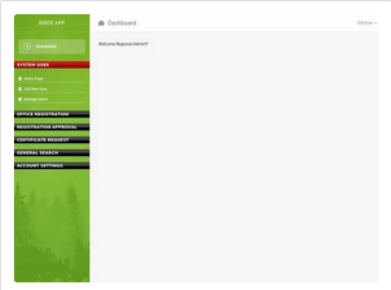
Report Summary

We could see from the above report that the district admin's main page has an overall performance of **80%** and loads faster than **52%** of other sites tested with this platform. The file request, which is quite huge compared to previous ones, and overall load time was **3.04s** from the Stockholm test location in Sweden.

Full Report Link: https://tools.pingdom.com/#!/c6Z3F9/http://node13526-env-6959028.ny-1.paas.massivegrid.net:8080/birds-app/pages/dist_admin/district_admin.xhtml;jsessionid=f4767dc8494fd17a0887a0ea972d

4) Regional Administrator Main Page

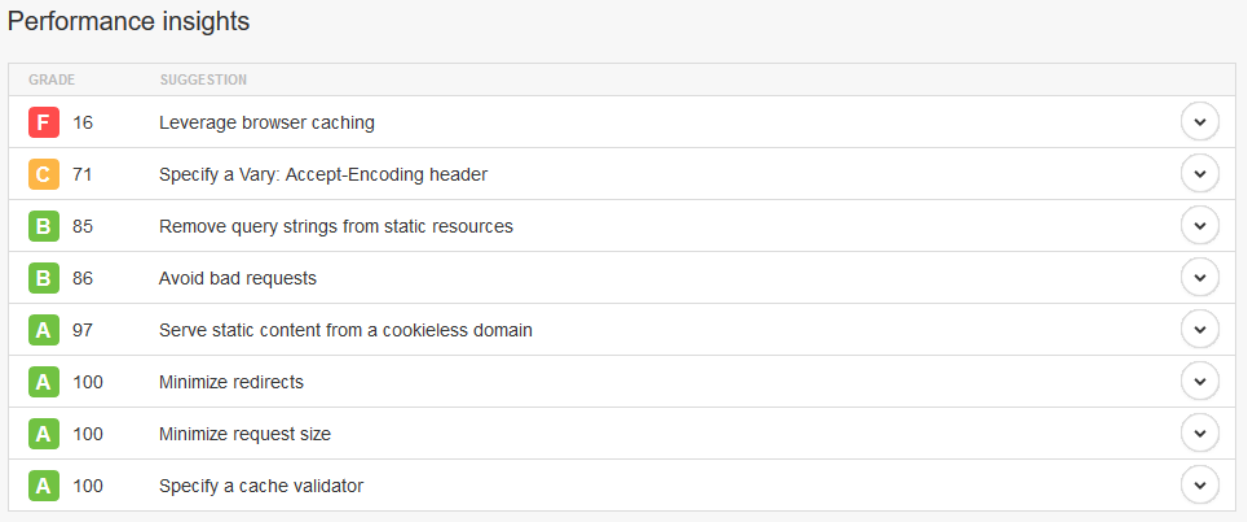
Summary



Performance grade	Load time	Faster than
B 82	1.08 s	88 % of tested sites

Page size	Requests	Tested from
1.9 MB	37	New York City on Mar 25 at 16:49

pingdom



Report Summary

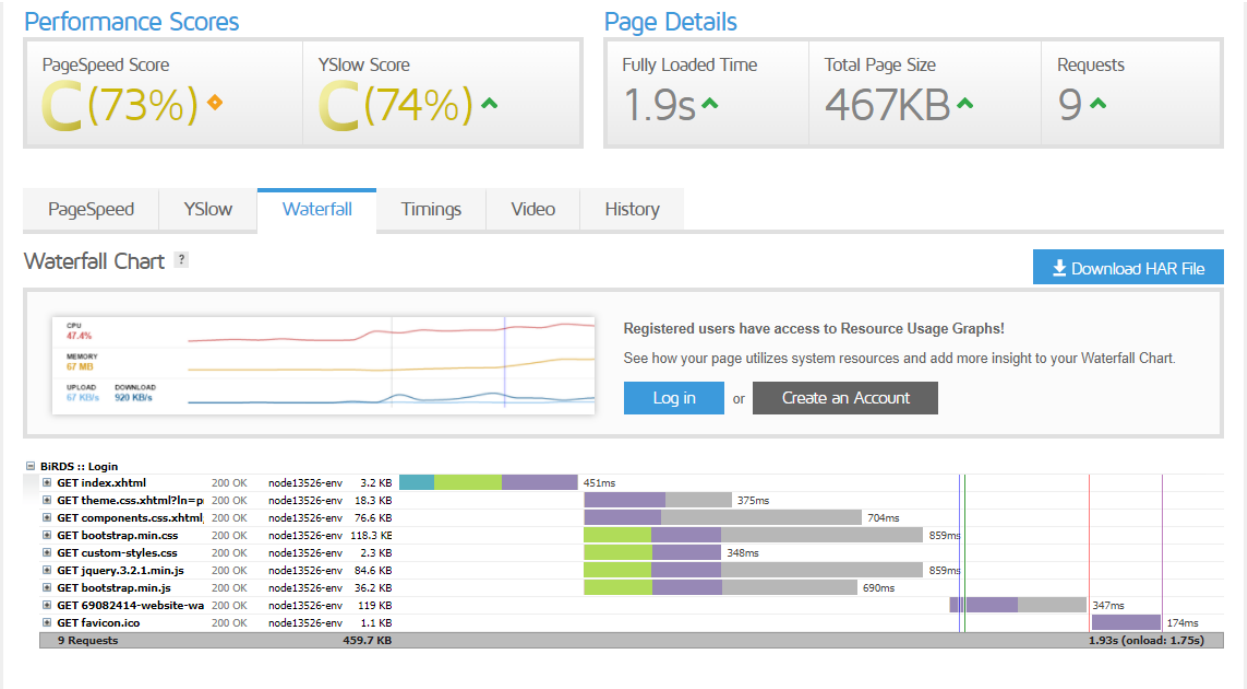
We could see from the above report that the regional admin's main page has an overall performance of **82%** and loads faster than **88%** of other sites tested with this platform. The file request, which is quite huge compared to previous ones, had an overall load time was **1.08s** from the New York City test location.

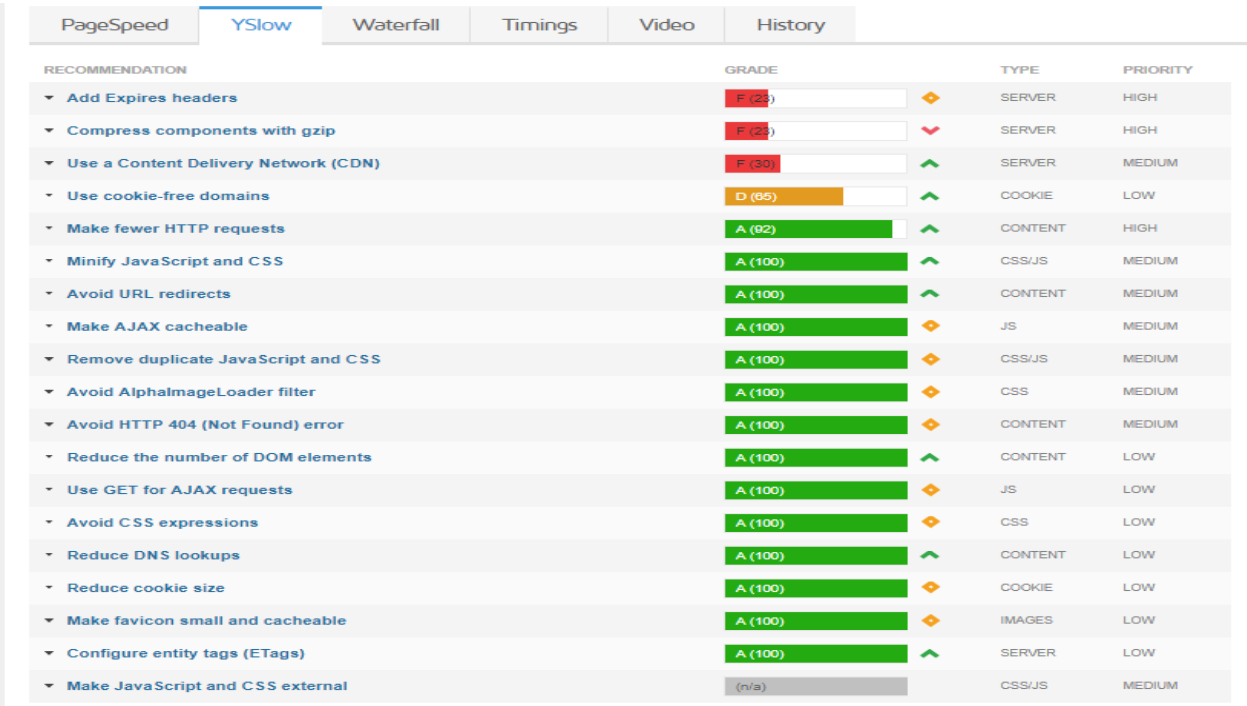
Full Report Link: https://tools.pingdom.com/#!/hqATS/http://node13526-env-6959028.ny-1.paas.massivegrid.net:8080/birds-app/pages/reg_admin/regional_admin.xhtml;jsessionid=f59f2b23353eb567cd8066299ad9

GtMetrix

This is another tool we used that gave us an insight on how well our site loads as well as provided us with actionable recommendations as to how to optimize it

5) Site test with GtMetrix





Another process whose time was calculated but could not be captured by these online tools is the Certificate Generation. Per our calculation, it took on average **5seconds** to generate each report – which is a very good time for such a process.

8. Contribution of Each Team Member

During the development of BiRDS, we followed a methodology that allowed us to work on all the project phases together side-by-side to reach our goals. We used a Software Version Control (SVC) system called GitHub where all the code of the projects was always available and each member knew what was changed at any time. This allowed each member the freedom to make changes and add ideas without having to always meet to do so. Documentations were also brainstormed and worked on together during our scheduled meetings.

At the end, the shared hard work and dedication by each member paved the way to successfully complete all the phases of our system.

References

[1]"Civil registration: why counting births and deaths is important", *World Health Organization*, 2018. [Online]. Available: http://www.who.int/mediacentre/factsheets/fs324/en/. [Accessed: 27- Feb- 2018].

[2]"Birth registration - UNICEF DATA", *UNICEF DATA*, 2018. [Online]. Available: https://data.unicef.org/topic/child-protection/birth-registration/. [Accessed: 27- Feb- 2018].

[3]"Newborn Care Programme | Division | Ghana Health Service", *Ghanahealthservice.org*, 2018. [Online]. Available: http://ghanahealthservice.org/newborn/programme-scat.php?ghspid=3&ghsscid=98. [Accessed: 28- Feb- 2018].

[4]"Mental Models and User Experience Design", *Nielsen Norman Group*, 2018. [Online]. Available: https://www.nngroup.com/articles/mental-models/. [Accessed: 03- Mar- 2018].

[5]"Model–view–controller", *En.wikipedia.org*, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller. [Accessed: 17- Mar- 2018].

[6]E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design patterns*. Boston, Mass.: Addison-Wesley, 2016.

[7]"JSON", *Json.org*, 2018. [Online]. Available: https://www.json.org/. [Accessed: 22- Mar- 2018].

[8]"SHA-256 Hashing in Java | Baeldung", *Baeldung*, 2018. [Online]. Available: http://www.baeldung.com/sha-256-hashing-java. [Accessed: 24- Feb- 2018].