

**Programming Applications and Frameworks (IT3030)**

**3rd Year, 1st Semester**

**2020**

**Hospital Management System**

**Feedback Report**

Submitted by: **Group: S1142.1**

* IT18162974 - Lakshan P.A.D
* IT18112474 - Rodrigo M.N.D
* IT18185126 - Mallawarachchi S.N
* IT18136234 - Sathsarani B.G.K
* IT18167160 - Watthuhewa M.P

**Table of Contents**

Methodology 1

Version Control Mechanism---------------------------------------------------------------------1.1

Group & work distribution 2

Requirements 3

Stakeholder analysis------------------------------------------------------------------------------ 3.1

Requirement analysis------------------------------------------------------------------------------ 3.2

**Functional ----------------------------------------------------------------------------------------------3.2.1**

**Non-functional----------------------------------------------------------------------------------------3.2.2**

**Technical-------------------------------------------------------------------------------------------------3.2.3**

Requirement modeling-----------------------------------------------------------------------------3.3

System Design 4

Overall Architecture------------------------------------------------------------------------------4.1

Database design-------------------------------------------------------------------------------------4.2

Workflow design-----------------------------------------------------------------------------------4.3

Functions 5

Patient management ------------------------------------------------------------------------------5.1

Hospital management-----------------------------------------------------------------------------5.2

Doctor management ------------------------------------------------------------------------------5.3

Appointment management -----------------------------------------------------------------------5.4

Payment management -----------------------------------------------------------------------------5.5

Gantt chart 6

References 7

Appendix 8

1. Methodology

n E healthcare system will be implemented by the project. It is a hospital management system which allows three user levels in the system. Patient, doctor and hospital will be the three user levels above mentioned.

To use the currently implementing system, user must be registered to the system. Reports will be generated by the system, according the requirements of the user levels. Registered users can make appointments with the registered doctors who visit the registered hospitals. The users can even make the payments for the appointments online.

Version Control

2.Group and Work Distribution Details

|  |  |  |  |
| --- | --- | --- | --- |
| Name | IT Number | Function | Description |
| Lakshan P.A.D. | IT18162974 | Hospital | * Add schedule. * update schedule. * delete schedule. * Update profile. * Delete profile. * View the hospital details. |
| Rodrigo M.N.D | IT18112474 | Appointment | * Add the appointment details. * Edit the appointment. * Cancel appointments. * View the appointment details. |
| Watthuhewa M. P. | IT18167160 | Patient | * Add personal details * Update personal details * Delete the profile * Add appointment * View the patient details. |
| Sathsarani B.G.K | IT18136234 | Payment | * Add payment details. * Update the payment details. * Delete the payment records. * View the payment profile. |
| Mallawarachchi S.N | IT18185126 | Doctor | * Add doctor details. * Update doctor details. * Add patient   reports details**.**   * Update patient reports**.** * View the doctor details. |

3. Requirements

3.1 Stakeholder Analysis

**UI**

**DB**

**Testing**

* **Patient API**
* **Doctor API**
* **Hospital API**
* **Appointment API**
* **Payment API**
* **Patient Management**
* **Doctor**

**Management**

* **Hospital Management**
* **Appointment Management**
* **Payment Management**

4.2 Requirement Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Functionality | Functional Requirements | Non-Functional  Requirements | Technical Requirements |
| Patient Management | * Should be able to register as a new user. * Can receive a schedule of doctors and hospitals. * Can register an appointment and make payments online. * Allow to edit their profile. * Should be able to provide a feedback about the system. * Should be able to have printouts according to the requirements. | * Security * Usability * Privacy * Reliability * Efficiency | * Patient registering to the system as a new user. * After validating the account of the user, update, delete and view patient details. * Connecting with appointment class to register an appointment. * Connecting with payment to make a payment. |
| Doctor Management | * Should be able to register as a new user. * Can receive a list of schedules for the required time period. * Can receive a list of patients. * Should be able to have printouts according to the requirements. | * Security * Usability * Privacy * Reliability * Efficiency | * Doctor registering to the system as a new user. * After validating the account of the user, update, delete and view doctor details. * Connecting with appointment class to receive a schedule. * Connecting with hospital class to create a list of hospital with doctor IDs. * Connecting with payment class to finalize the doctor charges. |
| Hospital Management | * Should be able to register as a new user. * Can receive a list of schedules for the required time period. * Can receive a list of patients and doctors. * Should be able to have printouts according to the requirements. | * Security * Usability * Privacy * Reliability * Efficiency | * Hospital registering to the system as a new user. * After validating the account of the user, update, delete and view doctor details. * Connecting with hospital class to create a list of hospital with doctor IDs. * Connecting with payment class to finalize the hospital charges. |
| Appointment Management | * Patient should be able to schedule an appointment after registering to the system with valid username and password. * Patient should be able to receive a schedule about appointments. | * Security * Usability * Privacy * Reliability * Efficiency | * Patient will register an appointment after signup. * Admin will manage appointments and connect with patient, doctor and payment to complete the task of registering an appointment. * After validating the account of the user, update, delete and view doctor details |
| Payment Management | * A registered patient should be able to make an online payment. * User should be able to receive a bill as a printout. | * Security * Usability * Privacy * Reliability * Efficiency | * Connect with patient, doctor, appointment, and hospital to complete calculating the total fee. * Provide payment methods to use. * After validating the admin account of the user, update, delete and view doctor details. |

4.3 Requirement Modeling

* Use Case Diagram

A picture containing text, map

Description automatically generated

5.System Design

5.1 Overall Architecture

A screenshot of a cell phone

Description automatically generated

5.2 Database Design

A close up of a map

Description automatically generated

5.3 Workflow Diagram

6.Function Description

6.1 Patient Management: IT18167160 ( Watthuhewa M.P)

Service design

Through this patient service among five more major parts, in the API system, can be able to add, update, delete and view the patient details. There have two parts mainly model and service where model describes how the insert, update, delete and view functions work and service class describe how to call those methods to function. There have few HTTP requests where API s run on. PUT method to insert data/resources, POST method to updates resources, GET method to view data/resources and DELETE method to delete resources were used in implementation.

Service development and testing

* Dependency management tools : **MVC patterns**
* Version Control : <https://github.com/PAF-00/S1142.1.git>
* IDE : Eclipse IDE Enterprise Java Developers-2019-12
* Database : MySql (phpMyAdmin) <http://localhost/phpmyadmin/sql.php?db=healthcare&table=patient&pos=0>
* Back End : Java – JAX-RS(Jersy)
* Testing tools. : Postman.Ink. <http://localhost:8089/Health/PatientService/Patients> (testing tool)
* Reason to choose Postman : Since “Postman ” is an app which interacts with HTTP API s ,we were able to verify that our services functions well(That our insertions,deletions,retrieves and deletions function well)

<http://localhost:8089/Health/PatientService/Patients> (code)

* Code quality checking tools : Selanium
* Testing methodology and results : <https://drive.google.com/open?id=1EwEmXaAJPytP5dtVDP4BaeyatEbxolRl>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Test Description | Test Input(s) | Expected Output(s) | Actual  Output(s) | Result  (Pass/Fail) |
| 01 | Insert Patient details. | Name : Maliesha  Address : Moratuwa  Age : 21  NIC : 99877655V  Phone : 0712345345 | * Maliesha * Moratuwa * 21 * 998776555V * 0712345345 | * Maliesha * Moratuwa * 21 * 998776555V * 0712345345 | pass |
| 02 | Update Patient Details for a selected user. | Selecting update option in ID=1 and changing the Name attribute into Pramodya from Maliesha . | -Updated success message.  -Name change to Pramodya from Maliesha | -Updated success message.  -Name change to Pramodya from Maliesha | pass |
| 03 | Delete a Patient | Selecting delete option in ID=1 and delete the entry. | Deleted profile in ID=1 | Deleted profile in ID=1 | pass |
| 04 | View User Patient | Selecting view for ID=1 | * Maliesha * Moratuwa * 21 * 998776555V * 0712345345 | * Maliesha * Moratuwa * 21 * 998776555V * 0712345345 | pass |

6.2 Hospital Management: IT18162974 (Lakshan P.A.D)

Service design

In this Healthcare system there are five major parts. Hospital service is also included into that part. Through this API system can add, remove, update and view the hospitals in the system. In this API there are two parts they are called model and service. Model represent how those insert update delete work and the service class include how to call those methods.

* In order to work word from these word from these be used HTTPs request as put post get delete

PUT-To create resources

POST-update resources

GET- get resources or list of resources

DELETE- to delete resources

* And, there are some attributes to represent the hospital. They are,

Hospital ID as HosId (int-primary key),

Hospital name as HosName (String),

Hospital City HosCity (String),

Number of Rooms as Rooms (int),

Hospital address as Address (String) and Hospital contact number as ContactNum (int)

Service development and testing

* Dependency management tools - **MVC patterns**
* Version Control : <https://github.com/PAF-00/S1142.1.git>
* IDE : **Eclipse**
* Database : **MySql (phpMyAdmin**
* Back End : **Java - JAX-RS (Jersy)**
* Testing Tools : **Postman**
* Postman was chosen to do the testing part of the function. It is easy to create and save simple HTTP/s requests. The test results are very clear and simple presented by postman.
* Code quality checking tools : **Sonarcube**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Test Description/ Test Steps** | **Test**  **Input(s)** | **Expected Output(s)** | **Actual**  **Output(s)** | **Result**  **(Pass/Fail)** |
| 01 | Insert Hospital details. | * HosId-1 * HosName -Asiri * HosCity-Narahenpita * Rooms-500 * Address-Kirimandala Mw,Narahenpita * ContactNum-011234234 | * 1 * Asiri * Narahenpita * 500 * Kirimandala Mw,Narahenpita * 011234234 | * 1 * Asiri * Narahenpita * 500 * Kirimandala Mw,Narahenpita * 011234234 | pass |
| 02 | Update Hospital Details for a selected user. | Click Update in HosID=1’s profile  Updating Number of rooms in HosID 1 INTO 600 | Updated Messageand change 500 to 600 | Updated Messageand change 500 to 600 | pass |
| 03 | Delete a Hospital | Click Delete in HosID=1’s profile | Deleted Profile HosID=1 | Deleted Profile HosID=1 | pass |
| 04 | View Hospital Details | Click View for HosID=1 | * 1 * Asiri * Narahenpita * 500 * Kirimandala Mw,Narahenpita * 011234234 | * 1 * Asiri * Narahenpita * 500 * Kirimandala Mw,Narahenpita * 011234234 | pass |

6.3 Doctor Management: IT18185126 (Mallawarachchi S.N)

Service design

Doctor is one of the key users in hospital management system. By implementing doctor management service, we will be able to produce functional requirements of the system like, scheduling appointments, receiving lists of patients and doctors, etc. Service was implemented by using JSON as web scripting language, java as a back-end language, and for database handling jQuery was used. We chose JSON because of it’s faster than other scripting languages and it’s known to be light weighted. Because of our experience with java and jQuery, it was easier to work with those languages. We used ‘phpMyAdmin’ administrative tool along with one of the popular open source cross-platform ‘xampp’ to build the database.

User has to login into the system with a valid account and then, user can see the appointment and check the patient details. After checking the patient, doctor can enter the prescription to the system. When the patient visits next time, doctor can update the report of the patient after checking the patient. A doctor can update the profile of his/her account. When the doctor resign admin can delete, his/her profile.

Service development and testing

* Dependency management tools :**MVC patterns**
* IDE : **Eclipse**
* Database : **MySQL**
* Back End : **java jax-rs jersey**
* Testing Tools :**Postman**
* Postman was chosen to do the testing part of the function. It is easy to create and save simple HTTP/s requests. The test results are very clear and simple presented by postman.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Test Description/ Test Steps** | **Test**  **Input(s)** | **Expected Output(s)** | **Actual**  **Output(s)** | **Result**  **(Pass/Fail)** |
| 01 | Insert doctor details. | * DocName- Savindu Lakshan * DocNIC - 994576181V * Gender- Male * ReqNo-4865 * Specialized- Hair * Email- savindu.l@gmail.com * DocCharges- 500000 | * Savindu Lakshan * 994576181V * Male * 4865 * Hair * savindu.l@gmail.com * 500000 | * Savindu Lakshan * 994576181V * Male * 4865 * Hair * savindu.l@gmail.com * 500000 | pass |
| 02 | Update Hospital Details for a selected user. | Click Update in DocID=1’s profile  Updating Email in DocID =1, into ‘namal1.gmail.com’ | Updated Successfully | Updated Sucessfully | pass |
| 03 | Delete a Docotr | Click Delete in DocID=1’s profile | Deleted Successfully | Deleted Successfully | pass |
| 04 | View Doctor  Appointment Details | Click View of Schedule | * 2 * 1 * 77642050 | * 2 * 1 * 77642050 | pass |

6.4 Appointment Management

Service design

Appointment management is the key of the implemented system. To register an appointment and make the payment through an online platform is the basic idea of this assessment.By implementing appointment management service, we will be able create an API for managing services connected to appointment management. Such as, registering, updating appointment details, remove an appointment and receive a schedule of appointments.

Service was implemented by using JSON as web scripting language, java as a back-end language, and for database handling jQuery was used. We chose JSON because of it is compiling easily and faster than other scripting languages. Because of our experience with java and jQuery, it was easier to work with those languages. We used ‘phpMyAdmin’ administrative tool along with one of the popular open source cross-platform ‘xampp’ to build the database.

Appointment function will be used by patient and will send data to the user level doctor. The user should have a valid account to use this function. Once the user entered relevant details to the appointment interface, system will send the data to the central database.

Service development and testing

* Dependency management tools : **MVC patterns**
* IDE : **Eclipse IDE Enterprise Java Developers-2019-12**
* Database : **php**
* Back End : **Java - JAX-RS (Jersy)**
* Testing Tools : **Postman**
* Postman was chosen to do the testing part of the function. It is easy to create and save simple HTTP/s requests. The test results are very clear and simple presented by postman.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Test Description/ Test Steps** | **Test**  **Input(s)** | **Expected Output(s)** | **Actual**  **Output(s)** | **Result**  **(Pass/Fail)** |
| 01 | Insert Appointment  details. | * PName – Malisha Wathuhewa * DName – Nilukshika   Mallawarachchi   * RoomNo – 07 * Date – 19/04/2020 * Time – 20.00 * ID - 4 | * Malisha Wathuhewa * Nilukshika   Mallawarachchi   * 077 * 19/04/2020 * 20.00 * 4 | * Malisha Wathuhewa * Nilukshika   Mallawarachchi   * 077 * 19/04/2020 * 20.00 * 4 | pass |
| 02 | Update appointment details for a selected user. | Click Update in AID=1’s RoomNo= 14 to RoomNo= 07 | Updated Successfully | Updated Successfully | pass |
| 03 | Delete an appointment | Click Delete in AID=4  record | Delete Successfully | Delete Successfully” | pass |
| 04 | View Hospital Details | Click View for Apoointment | * Rusiru Kalhara * Nipunika Rodrigo * 14 * 15/04/2020 * 14.00 * 1 | * Rusiru Kalhara * Nipunika Rodrigo * 14 * 15/04/2020 * 14.00 * 1 | pass |

6.5 Payment Management

Payment management is the one function that connect with every other function in this implementation. Which is one of the main and essential features of the system. Because of security of the data only admin can operate this function. With a valid account to create an invoice, after connecting to appointment, and getting details to create invoice, function will retrieve data from doctor function and hospital functions. We chose JSON because of it is compiling easily and faster than other scripting languages. Because of our experience with java and jQuery, it was easier to work with those languages. We used ‘phpMyAdmin’ administrative tool along with one of the popular open source cross-platform ‘xampp’ to build the database. User have to fill visa or master card details (card type, card no, card expiry date and card cvc number) according to the payment process then, user can make payment successfully after providing valid information to the system. PUT method to insert data/resources, POST method to updates resources, GET method to view data/resources and DELETE method to delete resources were used in implementation.

Service development and testing

* Dependency management tools : **MVC patterns**
* Version Control : <https://github.com/PAF-00/S1142.1>
* IDE : **Eclipse IDE Enterprise Java Developers-2019-12**
* Database : **phpMyAdmin**
* Back End : **Java – JAX-RS(Jersy)**
* Testing tools. : **Postman.Ink.**
* Postman was chosen to do the testing part of the function. It is easy to create and save simple HTTP/s requests. The test results are very clear and simple presented by postman.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Test Description/ Test Steps** | **Test**  **Input(s)** | **Expected Output(s)** | **Actual**  **Output(s)** | **Result**  **(Pass/Fail)** |
| 01 | Insert payment details. | DocCharge = 1200  HosCharge =1100  AppoCharge =1000  PayType=credit  CardNo=1234432109877890  CardExpiryDate=2024-09-21  Card\_CVNo=289  AID = 24  DocID = 56 | “Inserted Successfully” |  | pass |
| 02 | Update payment Details for a selected user. | DocCharge = 1400  HosCharge =0  AppoCharge =1200  PayType=debit  CardNo=7654456709877890  CardExpiryDate=2023-05-12  Card\_CVNo=654  AID = 15  DocID = 90 | “Hospital Charge cannot be equal or less than zero” | “Hospital Charge cannot be equal or less than zero” | pass |
| 03 | Delete a Payment | PayID = 3 | “Delete Successfully” | “Delete Successfully” | pass |

6. Gantt Chart



7. References

8. Appendix

* As an integration tool, we used git hub.

A screenshot of a cell phone

Description automatically generated

* Git hub repository link:

[https://github.com/PAF-00/S1142.1](https://github.com/NipunikaRodrigo/PAF-00/S1142.1)

A close up of text on a white background

Description automatically generated

A close up of a map

Description automatically generated

![A picture containing drawing

Description automatically generated]()A close up of a logo

Description automatically generated

A close up of a map

Description automatically generated