Software Engineering

Report 2 - Biometric Health Monitoring System

Group No - 3

Project Website - https://sites.google.com/site/healthmonitorinsystem/

Group members:-

Ajinkya Bilolikar
Amanbir Singh
Jagbir Singh
Siddhesh Surve
Swapnil Sarode
Yanze Zhang

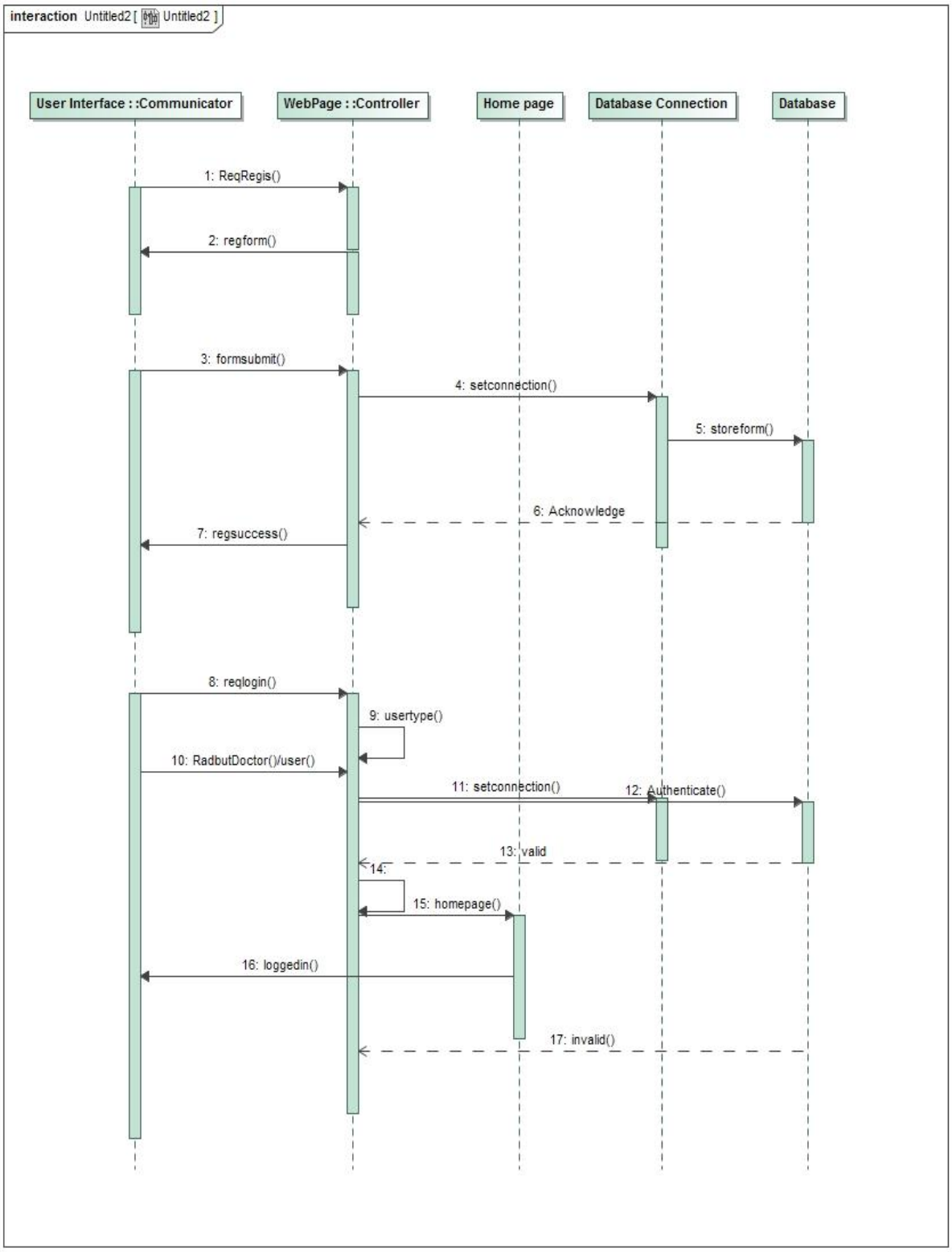
Table of Contents

1. Contribution Breakdown

“All Team members Contributed Equally”

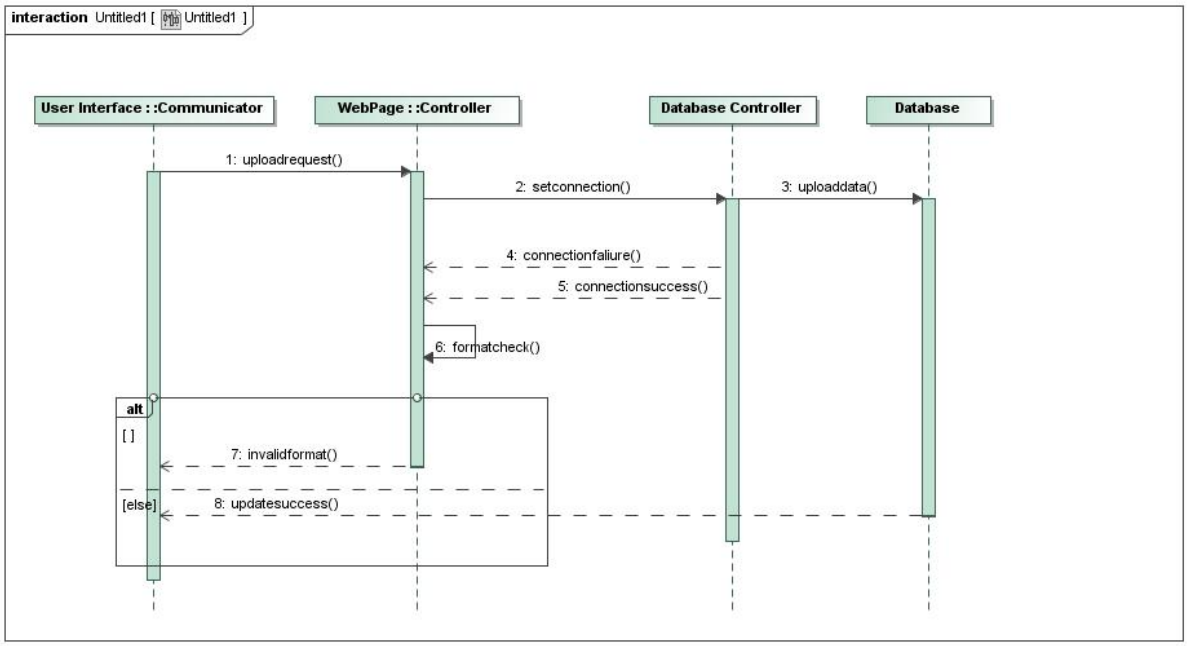
2. Interaction Diagrams:-

2.1 Use Case -1



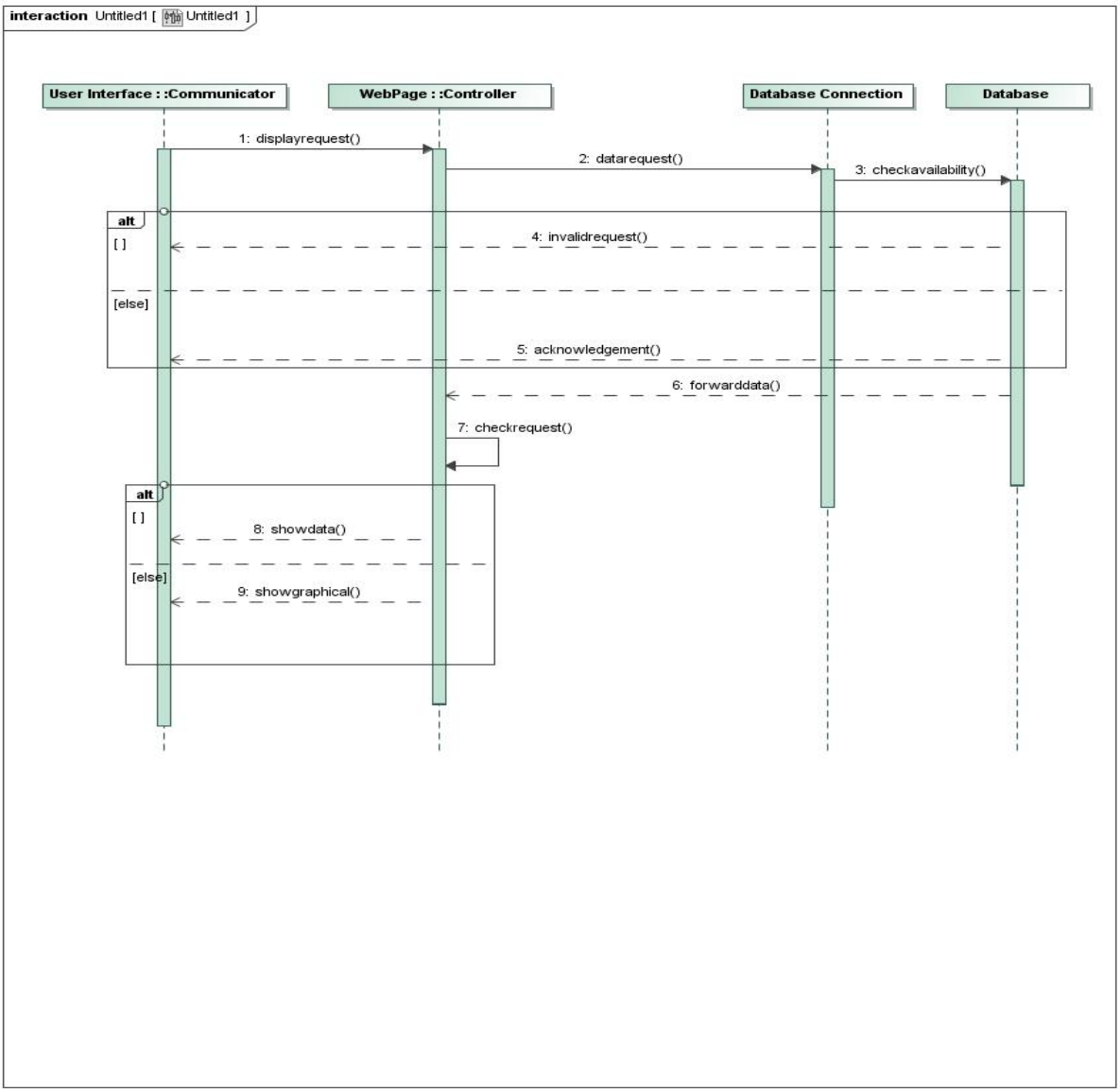
The above interaction diagram is for the first use case. The user sends a Reqregis() via the user interface to the webpage communicator. After the user requests for the first time to get registered, the web page controller sends back the form to user via regform(). Now, the user submits his form(formsubmit()) and then a connection is set up with the database. After this the form submitted by the user is stored via storeform() into the database. The user then receives an acknowledge reply from the database. If an existing user wishes to login then usertype() has to be chosen via a radiobutton. Again the setconnection() method is called and the database connection is set up. After this the database validates and sends a response to the webpage controller. If valid then it displas the home screen to the user via loggedin() method. If database doesn't validate then an invalid() method is sent to the web page controller.

2.2 Use case 2



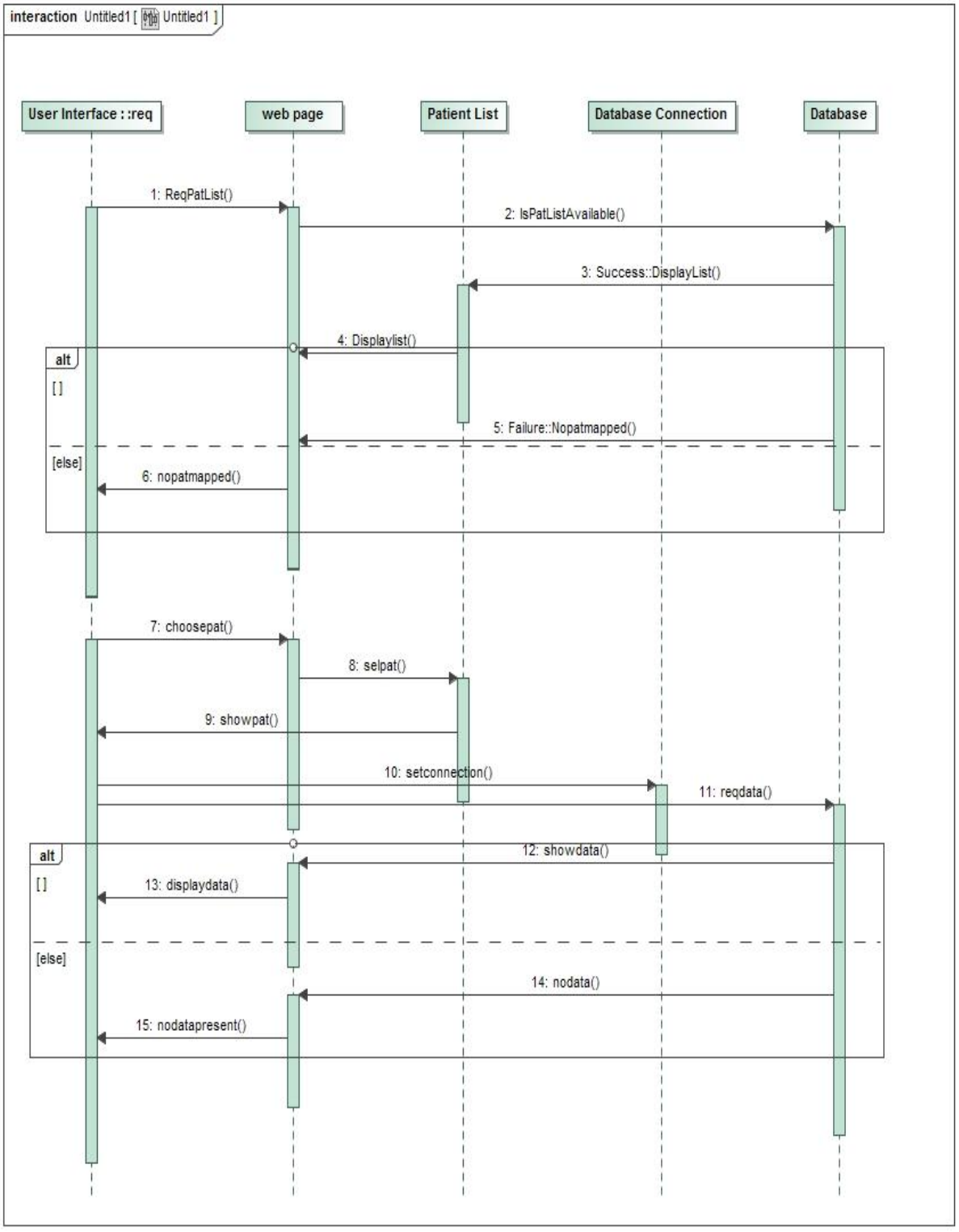
The above interaction diagram corresponds to the UC-2. The patient sends an uploadrequest() via the user interface::Communicator to the Web Page::Controller . Formatcheck() is performed here to check if the file being uploaded by the user is an excel sheet. If the file is of any other format then invalidformat() is sent back to the userinterface. If the file uploaded is an excel sheet then Setconnection() is sent to the database controller to set up the connection with the database. If the connection is unsuccessful then connectionfailure() is sent back to the web page::controller. If the connection is successful then the data is uploaded into the database. An updatesuccess() is sent back to the user interface confirming the user that the file has been uploaded.

2.3 Use case 3 & 4



The user generated request for displaying data is converted to datarequest() from the database.The Database Controller in turn checks for availability of data in the database. In response to this if the data is present in the database an acknowledgment() or invalidrequest() is invoked accordingly and the data is forwarded to Webpage::Controller. Depending on whether the user requested data in simple format or graphical format the methods showdata() and showgraphical() are invoked by the Webpage::Controller which completes the displaying task.

2.4 Use case 5


interaction Untitled1 [Untitled1]
User Interface : :req
web page
Patient List
Database Connection
Database
1: ReqPatList()
2: IsPatListAvailable()
3: Success::DisplayList()
4: Displaylist()
alt
[ ]
5: Failure::Nopatmapped()
[else]
6: nopatmapped()
7: choosepat()
8: selpat()
9: showpat()
10: setconnection()
11: reqdata()
alt
[ ]
12: showdata()
13: displaydata()
[else]
14: nodata()
15: nodatapresent()

In this scenario, the doctor wants to see the list of patients attached/mapped to this doctor account and the choose a patient accordingly to view his/her monitored data.

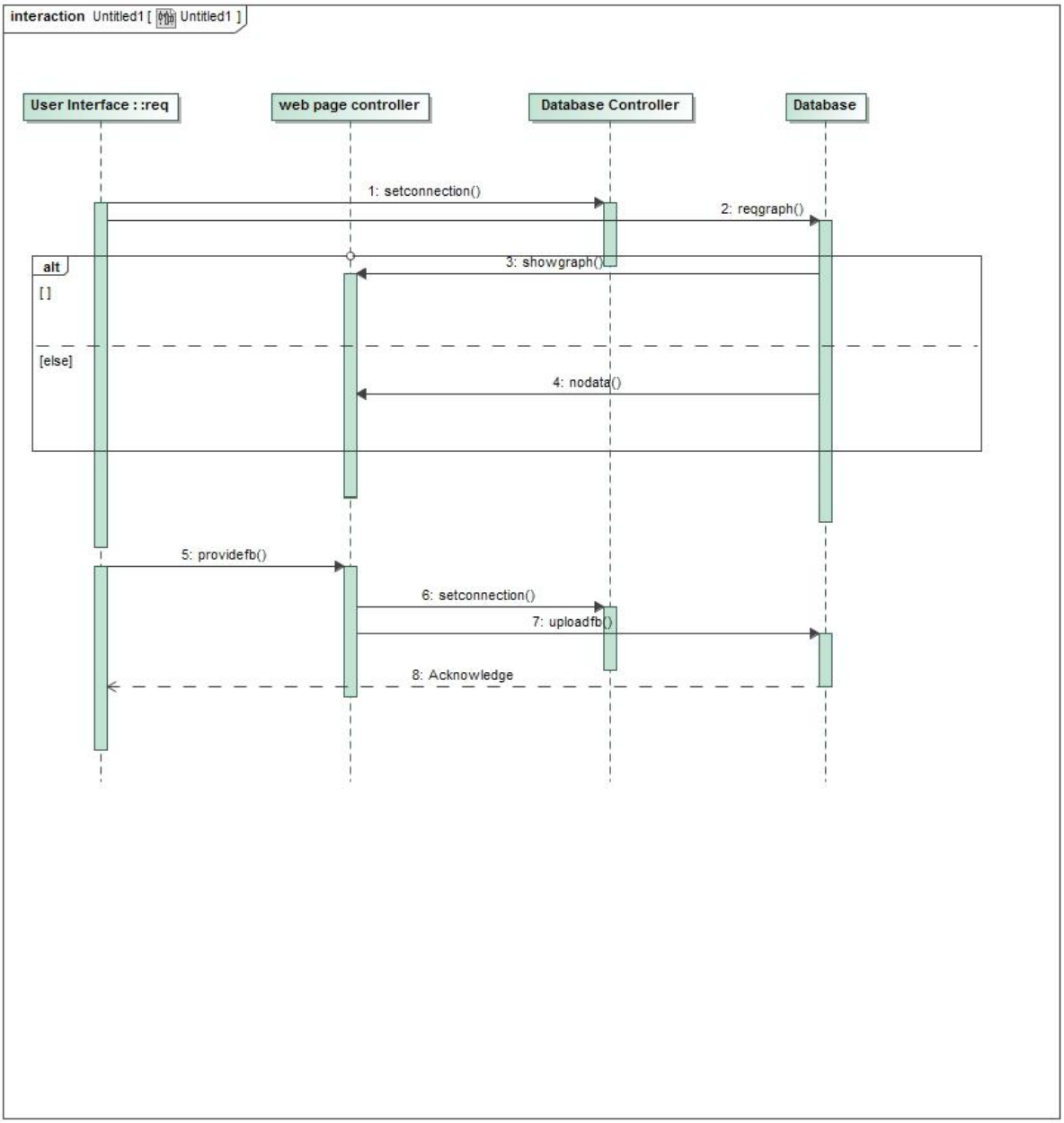
User Interface:-This object constitutes the most important part for user i.e doctor in this case. All the communications with the system are done through this object only, Doctor when wishes to see the patients mapped to his account send a request to view the patient list as shown. Also it shows the patient list system communicated with database and in case no patient mapped it displays an error message. This part is also responsible for the user sending the request for patient data and then getting the data displayed to the doctor for analysis

Webpage/controller:- this object controls all the communications ,requests and responses from user interface and is responsible for link between database and the user-interface. In case of request for patient list, webpage parses this request to the database and sends the response/acknowledgement back to the use-interface.

Database Connection:-Database connection is established whenever any data is required to be fetched from the database and presented to user or webpage for further processing. The connection when establishes allow the communication between database and webpage. It is closes as soon as the response/acknowledgement is sent until next request.

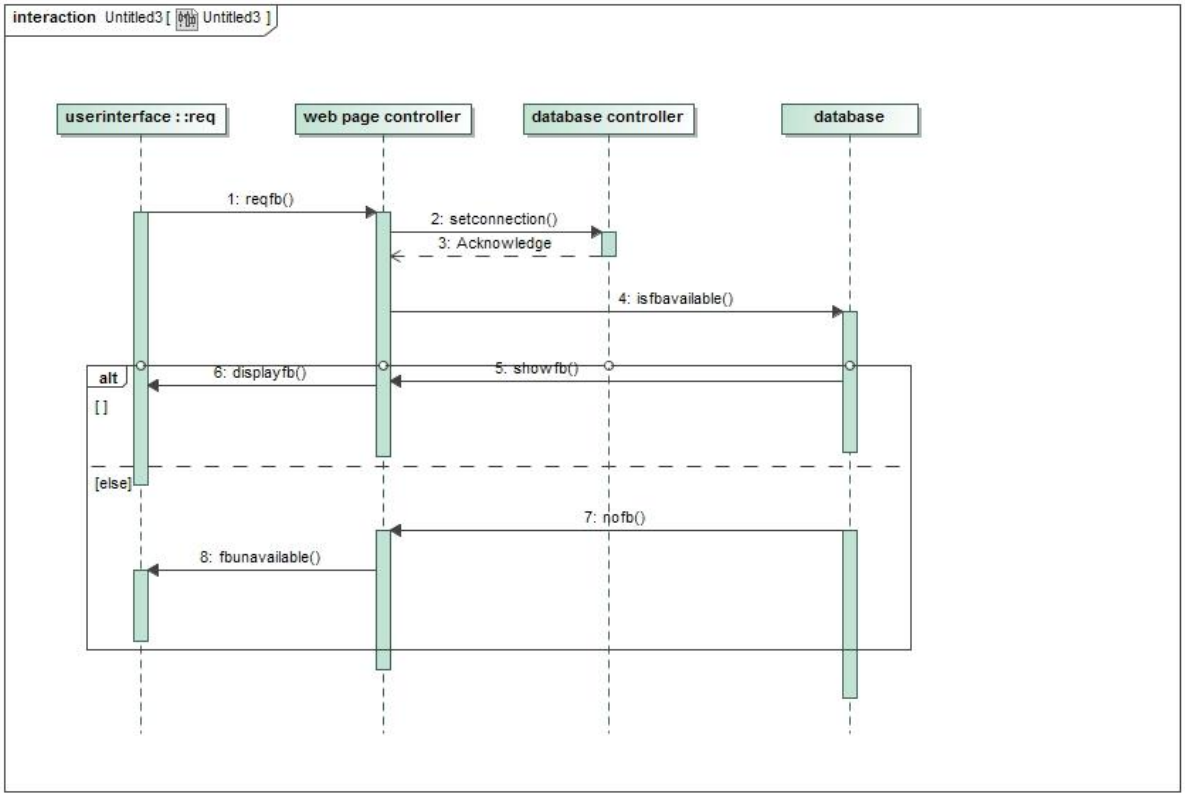
Database:- It stores all the information pertaining to which doctor is mapped to which all patients and when it receives the request for providing certain list of patients ,it fetches the desired records and sends the response accordingly to the webpage.

2.5 Use Case 6&7



The diagram above shows is the interaction sequence diagram for UC 6. The Doctor initiates the action by creating a request by sending the view graph request function. This also sets the database controller. The database replies with show graph message to the web page controller. In alternative scenario the database replies with no data to the web page controller. After getting the data and graph the doctor sends the feedback to the web page controller. The web page controller sets the database controller and uploads the feedback on the database. After uploading the feedback the database gives an Acknowledgement of the success of the event.

2.6 Use Case 8



User-Interface:- It allows the feedback provided by the doctor to be viewed by the patient .Patient request the same by clicking a button “View feedback. If feedback is present user is able to see that otherwise a message props up displaying that no feedback comments are available.

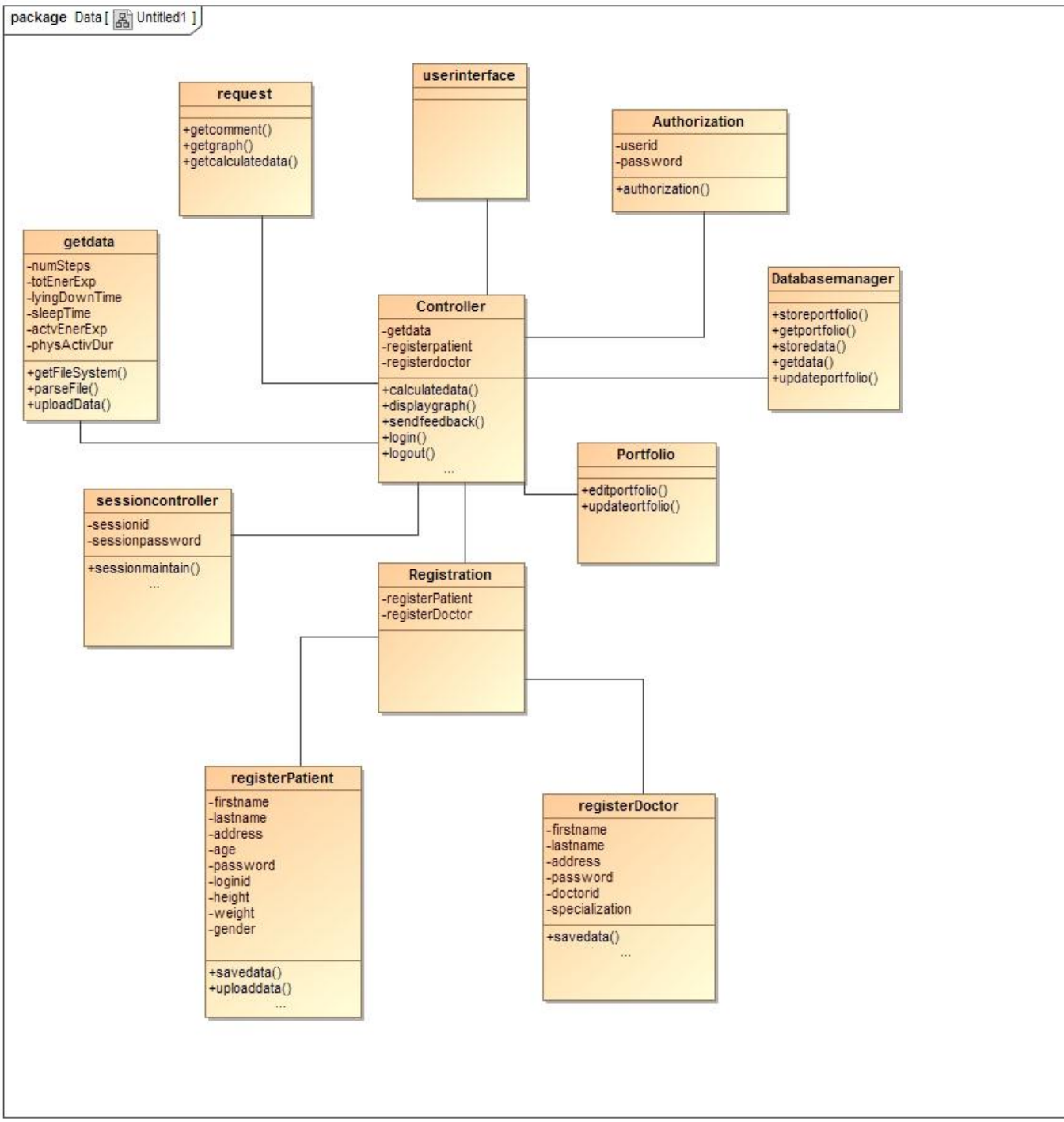
Web-Page Controller:- webpage controller gets the request form the user for viewing the feedback and parses this request to database. It is also responsible for requesting a connection to the database and proceeding when successful. The connection closing part is also handled at this stage.

Database connection:- Request from web-page controller for a database connection is checked for correct credential. This part makes sure that only the intended webpage with credentials of database is allowed to set up a connection and retrieve the records. If successful result in the database connection for communication between user and the database.

Database:- stores the feedback if available in the form of text. when receives the request it checks against the particular doctor-patient record and see if the feedback is presents, sends the feedback and connection is closed thereafter by controller.

3. Class Diagram and Interface Specification

a. Class Diagram:-



b. Data Types and Operation Signature

1. Controller:-

- **Operations:-**

- +calculateData():-Once the data is uploaded in system, this operation calculates the desired values to be displayed to the user interface.
- +displayGraph():-It displays the data calculated above in graphical form.
- +sendFeedback():-Sends the feedback to display it to patient.
- +login() :- Allows user(doctor/patient) to login.
- +logout():-User logs out through this operation.

2. registerPatient:-

- **Attributes:-**
  -string firstname
  -string lastname
  -string address
  -int age
  -string loginid
  -string password
  -float height
  -double weight
  -string gender

- **Operation:-**
- +uploaddata():- Upload the data from the user interface and send it to the controller for saving the data in database.
- +savedata():- Store the data in database and commit.

3. registerDoctor:-

- **Attributes:-**

  -string firstname
  -string lastname
  -string address
  -string password
  -string Doctorid
  -string specialization

- **Operation:-**
- +savedata():- Store the data in database and commit.

4. getData:-

- **Attributes:-**

  -string numOfSteps
  -string totEnterExpenditure
  - string lyingDownTime
  - string sleepTime
  - string actvEnerExp
  - string physActivDur

- **Operations:-**

- +getFileSystem():-Gets the raw data file from PC/laptop into the system.
- +parseFile():-Parse the data from raw file into the controller after interaction.
- +uploadData():-Uploads the extracted data and saves it into the database.

5. request:-

- **Operations:-**

- +getcomment():-Carries the request to the doctor to get feedback.
- +getgraph():-Carries the request to get the user data in graphical form.
- +getcalculatedata():-Carries the request to calculate the data/parameter values in textual form.

6. authorization:-

- **Attributes:-**
  -string userid
  -string password

- **Operations:-**

- +authorization():-Validate the credentials entered by the user.

7. portfolio:-

- **Operations:-**

- +editprofile():-Allows the user to edit the profile.
- +updatePortfolio():-Saves the updated profile in database.

8. sessioncontroller:-

- **Attributes:-**
  -string sessionid
  -string sessionpassword

- **Operations:-**

- +sessionmaintain():-Maintains the session in general and keeps the user logged in for specific period of time without inactivity, after that period, user is logged out.

9. Databasemanager:-

- **Operations:-**

- +storeportfolio():-Stores the user profile in the database.
- +getportfolio():- Gets the user portfolio to display on the user interface.
- +storedata():-Stores the data/parameters in database.
- +getdata():-Displays the data to the user interface.
- +updateProfile():-Carries the request, to update the user profile in database.

c. Traceability Matrix

Since our system is not implementing the above class structure exactly, we are using HTML/JSP to control the implementation logic. The above design is the nearest approximation of this system when it is developed using pure object oriented class structure. The classes explained above were derived from the traceability matrix. All the domain concepts were merged with the use cases in order to see how the system will function. After this merger, the classes started evolving with the features and user interface that would be displayed depending upon the type of user, whether he/she is a doctor or a patient.

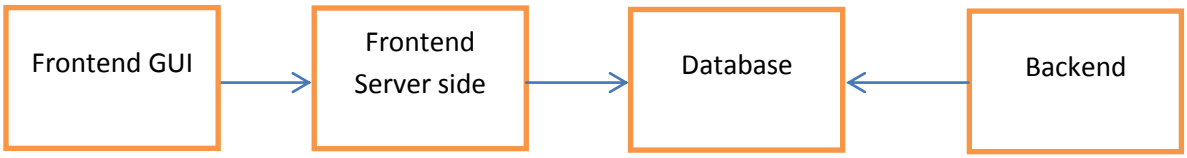
All the features vital to the system, be it the user data, portfolio or controller were included as mentioned above. The class 'databasemanager' was centred on the database operations as database holds a key spot in the system by saving all the important data related to user profile and user health monitoring data, as well as the appropriate error messages that should be displayed. Rest of the classes included the operations and attributes related to getting the raw data file and processing that data to display the user friendly data and allowing the doctor to view the data and provide the feedback.

4. System Architecture and System Design

a. Architectural Styles

Before speaking of the kinds of architectural styles, it is useful to provide a definition of an architectural style. “An architectural style …defines a family of systems in terms of a pattern of structural organization. More specifically, an architectural style determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of constraints on how they can be combined. Web applications usually combine a mix of architectural styles. The frontend design uses the Model-View-Controller architecture, which is considered to be a Separated-Presentation architectural style. The separated-presentation architectural style describes the separation of presentation code from internal logic code. It is also using the client/server architectural style. All of the data to run the application is stored centrally on the BHM server, but many clients can access the web application from different places around the world through many different Internet browsers. Both the frontend and the backend are using the component-based architectural style by using design and development languages that allows them to run, independent of the platform they are on. In this way, the code gets great reusability and allows for growth and scalability. Sometimes, the Representational State Transfer (REST) is described as an architectural style when it is thought of as being comprised of a uniform interface and a layered architectural style.

b. Identifying Subsystems



The frontend GUI provides the interface by which a user can interact with the system and view the data from the system. The front end server side retrieves data from the database and presents it to the user via the frontend GUI. It also stores new data of patient in the database. The database holds all the persistent information such as user info, portfolios, the data history. The backend handles doctor notifications of provided feedback and updates user information and data histories appropriately.

c. Mapping Subsystems to Hardware

While the server is contained to one machine, the system as a whole is spread across different machines. The system is effectively split into two separate and fairly independent sections, a frontend and a backend, with the database (residing on the server) acting as the intermediary between the two. The frontend is further subdivided into a GUI component which runs within a web browser on a client computer (or realistically, many clients' computers) is providing a rich interactive experience and the server side portion runs within the web server process on the server. The server side frontend handles interaction between the GUI and the database, such as retrieving profile and ensuring data is properly and legitimately entered into the database. The backend process runs on the server alongside the database and the server side half of the frontend.

d. Persistent Data Storage

A MySQL relational database is used for persistent data storage. There are five tables within the MySQL database. These tables are as follows:

datainfo,

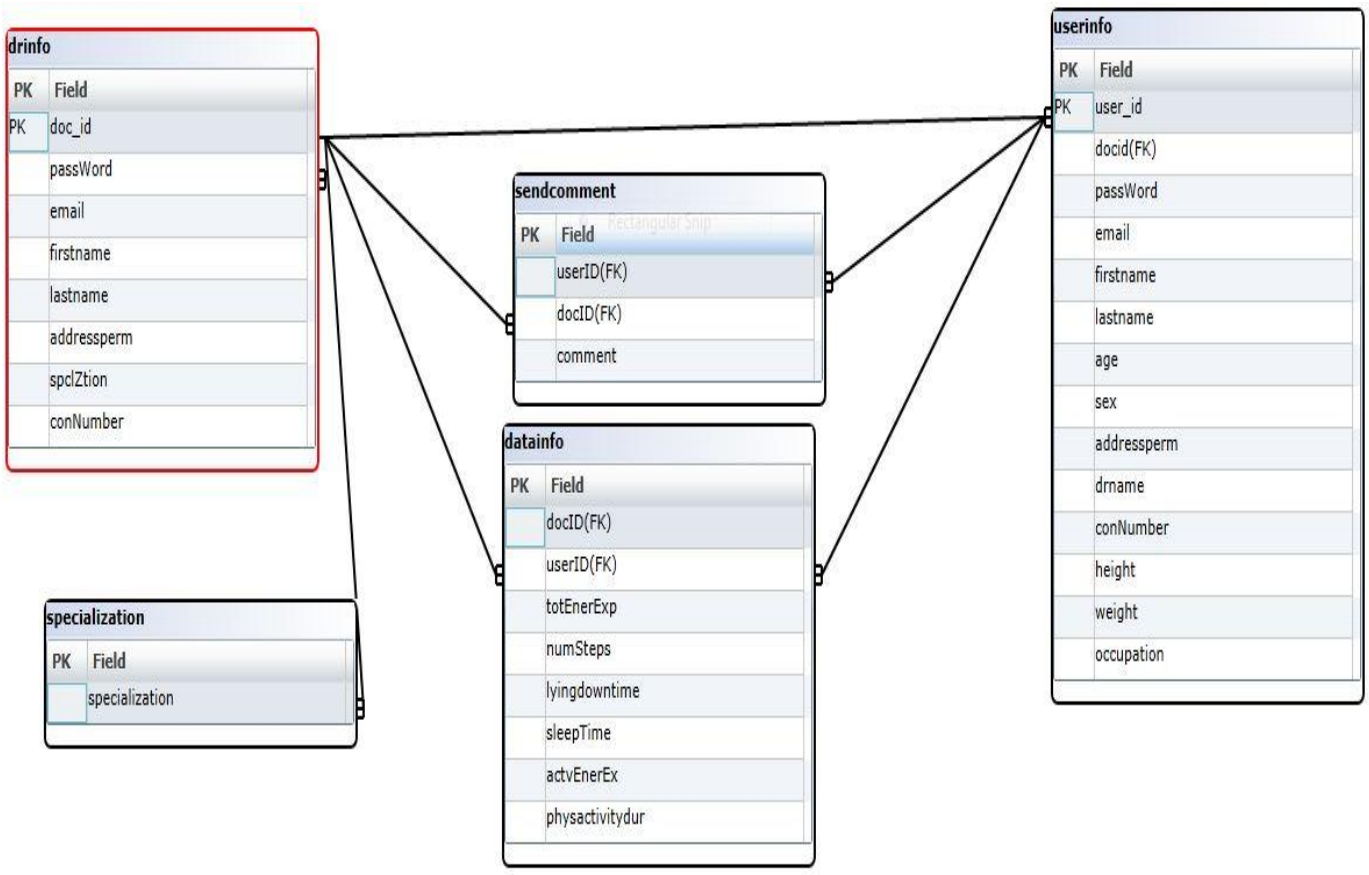
drinfo,

sendcomment,

specialization and

userinfo .

The figure on the next page shows the database schema mapping including field names and types.



The docinfo and userinfo table stores the user's and doctors' information such as username, password (encrypted), email, address, age, sex, specialization, height, weight etc. These tables also holds the primary key based on user and doctor which are use_id and doc_id respectively. Each user has a unique 'id' associated with their account. This id is the primary key. The datainfo table stores the docid,userid and other data of user like, number of steps, lying down time, sleep time, energy expenditure, physical activity duration, etc.

e. Network Protocol

The BHM is self-contained to one server. Running on the localhost, there is a MySQL database server and an Apache web server. The client is comprised of html, JSP and will interact with user having these services. There are different external connections needed for the server. First, the website is accessed via incoming HTTP requests on port 80, this is an Internet standard, all user interactions happen over the frontend web interface.

Both the backend and frontend make outgoing HTTP requests to gather the data and calculate the required information to be provided. Internally, there are different connections needed for application communication. The frontend relies on JSP's internal drivers to connect to a MYSQL database for access to the persistent data and JAVA for core calculation. The backend uses the JDBC driver to connect to the MySQL database

f. Global Control Flow

Our system is event-driven in that it waits for users' input and acts according to it. User interaction is required in frontend features like logging in, placing a data, etc. The system does not log anyone in until the initial event is driven by an external request. Also, we have a timer run in our backend program to fetch user Information and Data. For time dependency, the backend of our system periodically processes pending data. The system has multiple threads to support multiple users operating at the same time and upload data whenever user wants.

g. Hardware Requirements

Server-Side

Note that a complete scalability analysis has not been performed, so the server- side hardware requirements are based on the needs of the current website.

A WLAN connection is required so necessary hardware includes a network card. This hardware profile is everything needed for server maintenance and administration on the current server.

Client-Side

On the client side, the client needs to have a monitor to view the GUI Display. In addition, since a web browser is required, the other hardware that comprises a computer is required such as CPU, Graphics Card, RAM, keyboard, mouse, etc. A web connection is also required, so the computer must contain a network card (whether wireless or Ethernet).

5. Algorithms and Data Structures

Algorithms

Most of the functions of our project take user inputs and return outputs with minimal data manipulation other than page rendering. As such, there are really no noteworthy algorithms to discuss apart from the model used to send and calculate data. ADD NEW MODEL a relational database will be used for persistent data storage. Most of the data in the system will be entered into the database and so algorithms for manipulating the data, such as sorting and searching, are handled by the database. Thus search and sorting algorithms are not in the scope of the system and will not be discussed.

Data Structures

The main data structure in our system is the Database. A database table is used to maintain the user login information such as email address and password and user profile details like height, weight, etc. based on which doctor provides his/her feedback. The following data types are used to store the variables in the table.

Email address: string

User ID: string

Doctor ID: string

Password: string

First name: string

Last name: string

Date of birth: date

Sex: string

Weight: float

Height: float

Occupation: string

A database table is used to maintain the data information regarding various patients' health parameters parsed from the file uploaded by the user. The following data types are used to store the variables in the table.

Doc id: string

User id: string

Total energy expenditure: float

Number of steps: int

Lying down time: float

Sleep time: float

Active energy expenditure: float

Physical activity duration: float

A database table is used to maintain the feedback information sent by the doctor to the patients. The following data types are used to store the variables in the table.

Comment: string

Doc id: string

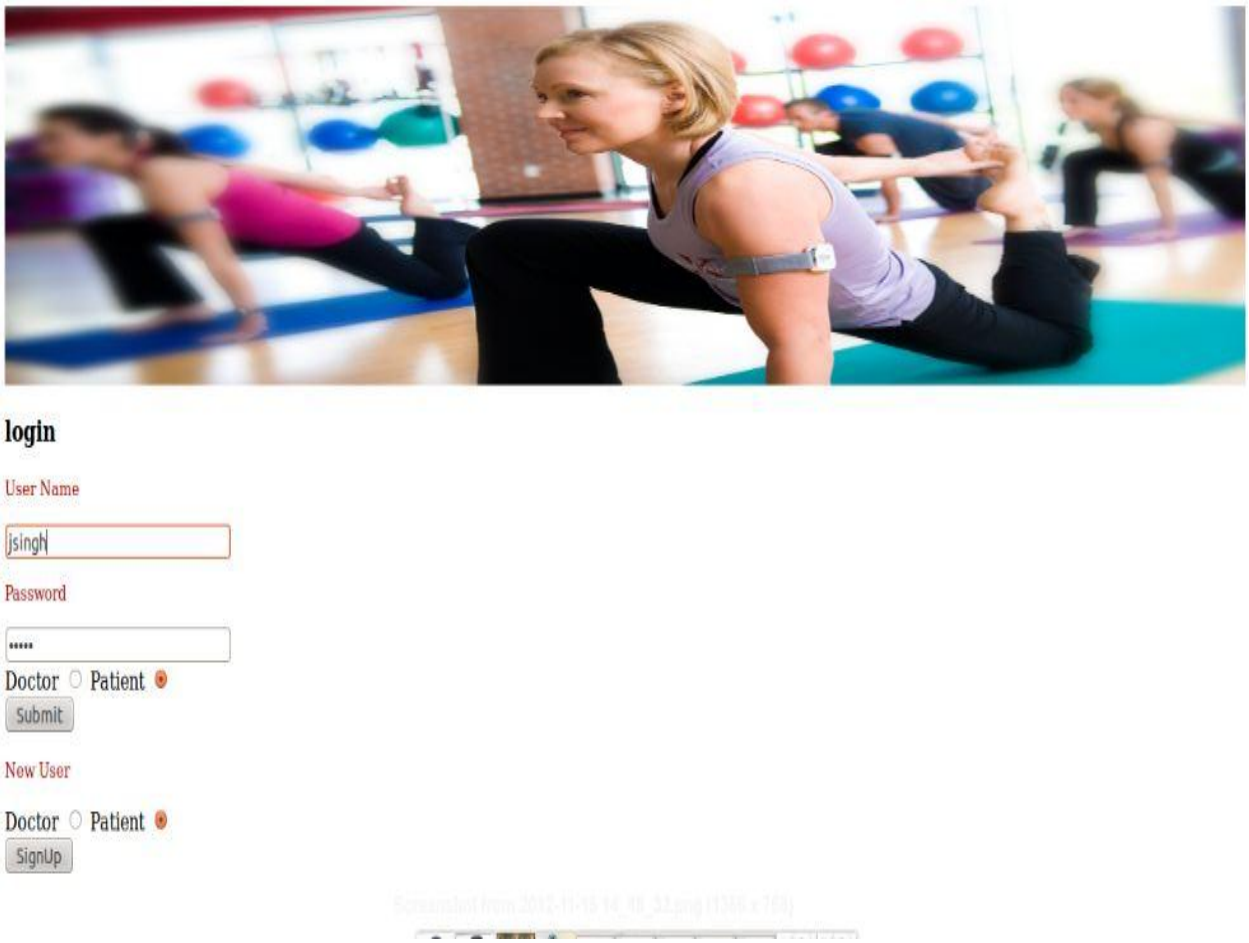
User id: string

Date: date

Other than the data structures stated above our system doesn't use any other complex data structures.

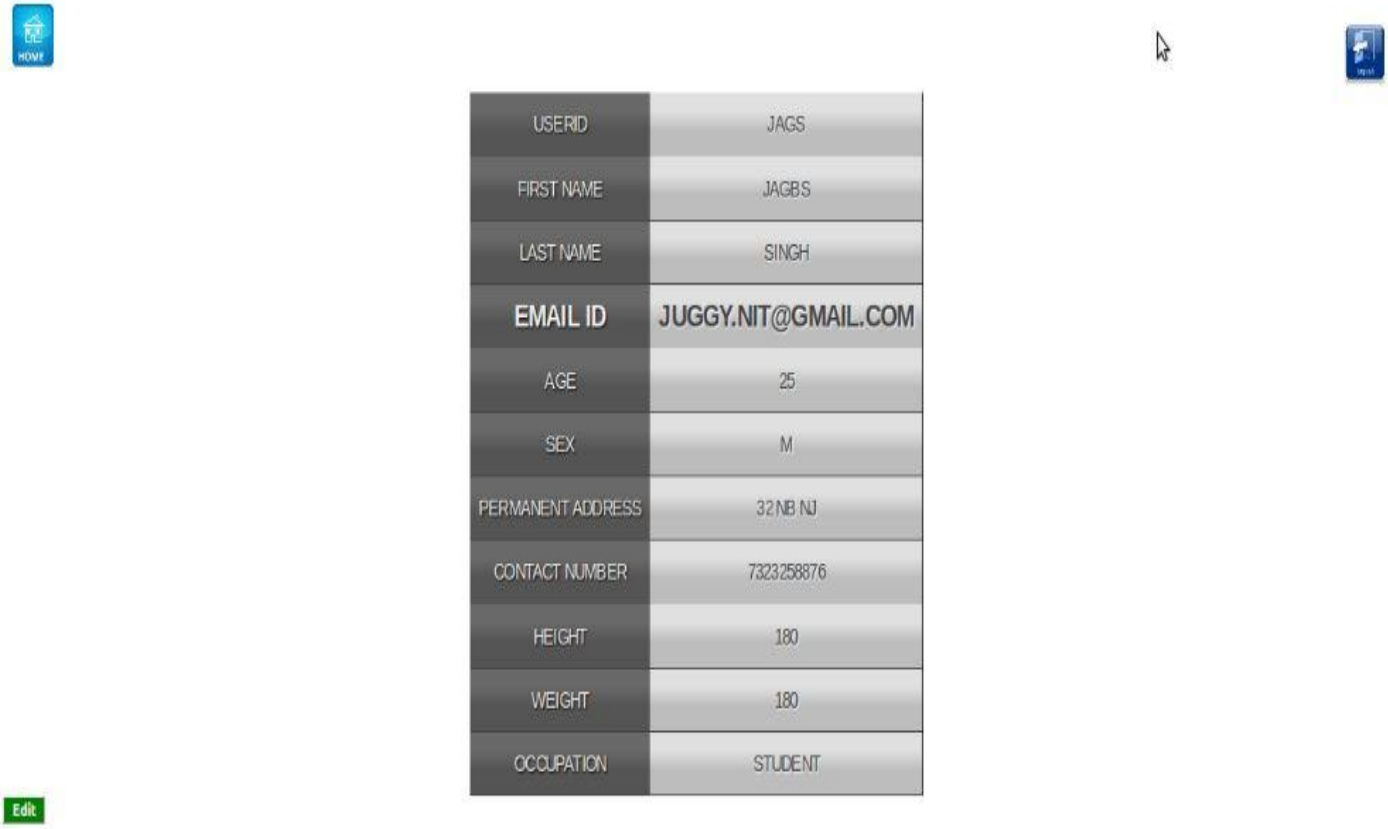
6. User Interface Design and Implementation

Login Page:



The login screen has not changed significantly except for few Graphics has been added to this page like radio buttons for doctor and patient.

Registration:



Compared to the initial user interface mock-up, the edit account page has been completely changed. In the plan that was drawn, this page would have many areas to optionally enter the information that required change. In the current design, the entire My Account page has been dedicated to manage the user account—including email address, user id, contact number and permanent address.

Home page:

The initially proposed home screen was little dull and lacked a logo and picture. The tabs were also altered and categorized. Each of the “tabs” in the above screen is a hover-style menu, unlike the initial design. JSP and html were used to create these menus.

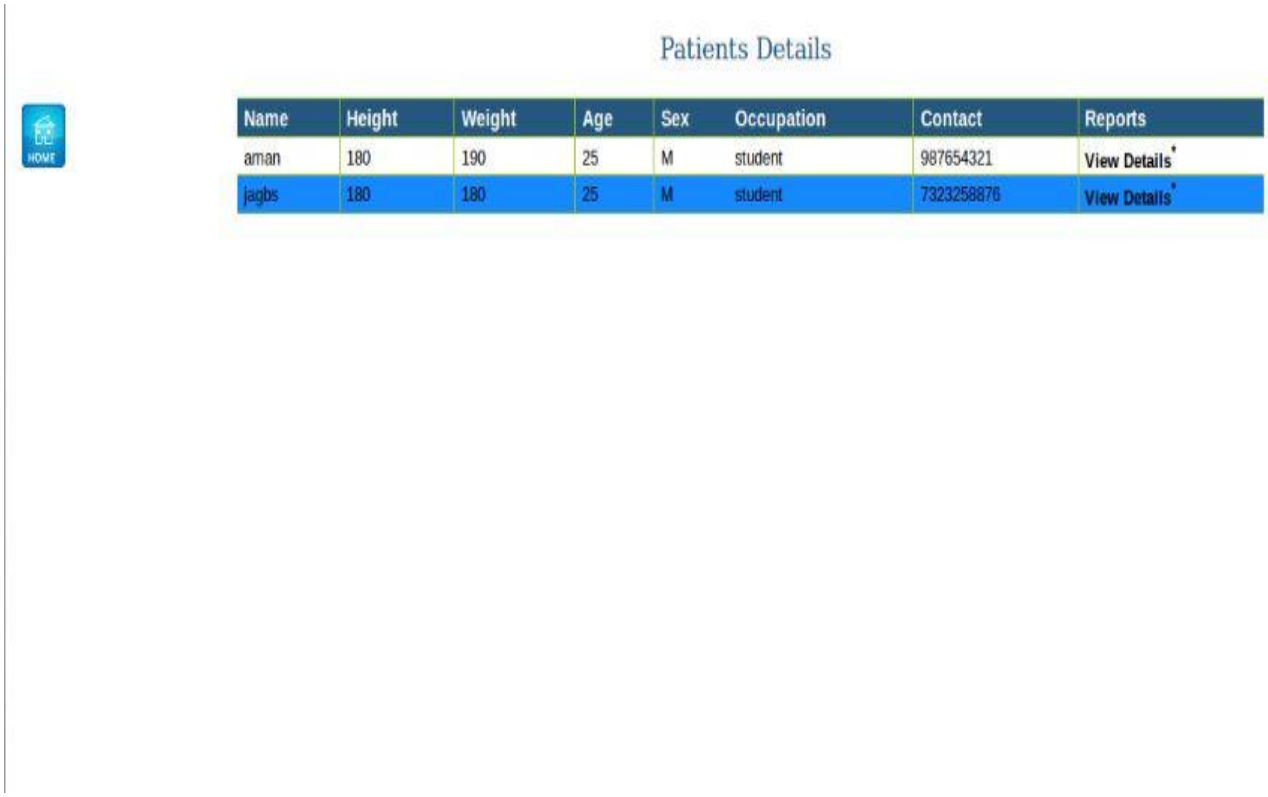
User home page:

To enhance the user experience the control of all the user functionality has been moved to tabs. The new menus provide multiple tabs on the screen. Managing user account, uploading data, calculate, contact information and logout can be found in tab on the home screen.

Doctor home page:

The doctor's screen contains an additional patients tab to view the list of patients mapped to that doctor.

Patient details:



The patient details page has been simplified more by displaying all the patients' information that are mapped to the doctor. Also the copies of the patient reports are shown in this page so as to view the patient data more easily along with contact information.

7. Design of Tests:-

These are test cases for determining the correctness of implemented structures in the program:

1. **Test case id: Login_TC**

Unit to test: User/Doctor Login

Assumptions: The program has displayed the input Login screen and is waiting for user

Action

Input values	Expected Results	Pass/Fail	Comments
Valid userid, valid password	Login Succesfully	Pass if result meets the expected result.	This test case shows that system is allowing valid/existing users to log in.
Invalid UserID/Password	Display error	Pass if result meets the expected result.	This test case shows that system displays invalid user message for unregistered users or registered users entering incorrect log in information.

2. **Test case id: Register_TC**

Unit to test: User/Doctor Register

Assumptions: The program has displayed the input Login/Register screen and is waiting for user

Action

Input values	Expected Results	Pass/Fail	Comments
User information with an existing user-id field	Display error in registration.	Pass if result meets the expected result.	This test case shows that user doesn't allow same user-id to be used twice.
User information with an unique user-id field	Register Successful	Pass if result meets the expected result.	This test case shows that new user is successfully registered.

3. **Test case id: Upload_TC**

Unit to test: file type

Assumptions: The program has displayed the user homepage and user wants to upload his data

Action

Input values	Expected Results	Result	Comments
User uploads a file with .xls extension with desired data format.	The file should be successfully uploaded	Pass if result meets the expected result.	This test case shows that user sends a proper.xls file to upload.
User uploads a file with .xls extension with random data format.	Proper error should be displayed.	Pass if result meets the expected result.	This test case shows that .xls file with improper format is will not be allowed to upload.
User tries to upload a file other than .xls extension.	Error should be displayed that file format is not proper	Pass if result meets the expected result.	Only .xls files are allowed to be uploaded.

4. **Test case id: Mapping-TC**

Unit to test: Database and User-profile consistency

Assumptions: The correct .xls has been uploaded and data has been extracted from the file and stored in the database.

Input values	Expected Results	Result	Comments
Extracted data from the .xls file.	The extracted data is mapped to the correct user.	Pass	This test shows that the consistency between database and User records is maintained.

5. **Test case id: UserReview_TC**

Unit to test: User request

Assumptions: The User has logged in and needs a review of his profile from doctor.

Input values	Expected Results	Result	Comments
User saves the data shown above to the doctor.	Appropriate Doctor is able to see that patients data and profile.	Pass if result meets the expected result.	This test ensures correct mapping between users and doctor.

6. **Test case id: EditProfile_TC**

Unit to test: Profile information

Assumptions: User already has created a profile and has a valid user id.

Input values	Expected Results	Result	Comments
Update new valid profile information.	Profile update successful.	Pass if result meets the expected result.	This test shows that user has entered expected format for the profile information.
Updates new information but with incorrect format.	Profile update unsuccessful.	Pass if result meets the expected result.	This shows that profile information format validity is ensured.

7. **Test case id: ViewInfo_TC**

Unit to test: Information in database

Assumptions: User has uploaded valid .xls file and data has been extracted and stored in the database.

Input values	Expected Results	Result	Comments
User enters valid date and time.	Data pertaining to that period is displayed.	Pass if result meets the expected result.	This shows that user has the option to view the data of particular day and particular time.
User enters invalid date and time.	Display appropriate error information.	Pass if result meets the expected result.	This test shows that user cannot request information that is not present in the database.

8. **Test case id: Viewformat-TC**

Unit to test: User view format

Assumptions: User has requested information for valid duration.

Input values	Expected Results	Result	Comments
User requests information in tabular format.	Data pertaining to that request is displayed in tabular format.	Pass if result meets the expected result.	This shows that user has the option to view the data in tabular format.
User requests information in graphical format.	Data pertaining to that request is displayed in graphical format.	Pass if result meets the expected result.	This shows that user has the option to view the data in graphical format.

9. Test case id: DoctorFeedback-TC

Unit to test: Feedback

Assumptions: Doctor has reviewed user profile and provided comment.

Input values	Expected Results	Result	Comments
Doctor enters feedback.	Feedback is stored in the database and mapped to the particular user.	Pass if result meets the expected result.	This test case shows that user gets correct feedback.

10. Test case id: ViewFeedback-TC

Unit to test: Feedback

Assumptions: Doctor has given feedback.

Input values	Expected Results	Result	Comments
User (patient) requests to view feedback.	Appropriate Feedback is displayed to the user.	Pass if result meets the expected result.	This test case shows that user gets correct feedback.
User (patient) requests to view feedback while it is not provided by the doctor.	Displays the latest feedback available i.e. feedback upto the last calculated values.	Pass if result meets the expected result.	This test case shows that when doctor has not sent the feedback, the latest available feedback is present.

These above tests cover all of the high level/user testing that can be done. Other testing such as determining correctness of every single line of code will be carried out/has already been carried out by each software developer as they are writing each section of code.
As far as integration testing goes, as we combine the modules and separate code of each developer, we will make sure that any discrepancies that arise are flattened out in an orderly manner. Commenting our code excessively around places where other people's implementations fit in is what will help the process of combining everything together go much more smoothly than if we just handed each other pure code.
Each section of code will be double checked for correctness of implementation and also correctness of the actual algorithms being employed. If any vague or nonstandard implementations are used, they must first be justified by an explanation in comments in order to pass the correctness test each person writing code will perform. Vague or nonstandard means structures that don't show their purpose or function in a manner that is obvious enough to a proficient user of the programming language in which we are working.

8. Project Management and Plan of Work

Merging the Contributions from Individual Team Members One of the first problems we encountered was that when we had done our specified parts, everyone had their own part on a separate Microsoft word document for example. To combat this headache, one person suggested that we use Google documents. Only a few of the group members had experience with Google Docs, so the rest of us had to learn how to share documents, a specific part for example with the rest of the group. A problem we are still having is that Google Docs is still not as good for formatting etc. and missing some specific actions we constantly rely on in Microsoft word. For example, the ability to work with tables in Google Docs is fairly limited in manipulating them. To handle this, we still rely on importing the finished Report to Microsoft Word to add the finishing touches and do what we cannot in Google Docs before submitting our work. Another reason we rely on Google Docs is version control. Here on the web everyone has access to the same document and can edit at will. Not that this also caused some problems where editing set us back when something was edited in the wrong way, but it has done more right than wrong. Finally, a really good point about using Google Docs for this semester long project is that everyone can keep track of each other's progress. At the start of the next deliverable, the Co-leaders begin by formatting an empty document with all the sections that need to be completed when the deliverable is due. By being able to access and view the same document that we all should be working on, a missing section is pretty self-evident. This allows us to catch this simple yet destructive problem early on to motivate our team member to do his part and has been vital to our success so far!

Project Coordination and Progress Report

As of the due date of this Report 2, we have the majority of UC-1: Initialize completed. We have completed the working on the integration between the GUI and actual initializations that occur when inputs from the user are given to set up the simulation as well as working with the GUI's for the final use case of UC-6: View Graphs.

In addition, we have simple pieces of the other use cases completed. We have fully integrated the fetching of the data by parsing the raw file and getting the useful data and then calculating the information to be displayed. All the values are successfully stored in the database.

Jagbir and Amanbir have been managing the group as a whole to make sure everyone is on the same page and focusing their efforts in the right direction. As a group we are pretty open to each other about discrepancies and so far any that arose were resolved fairly

quickly. Since all of our group members are in the same classes, it has not been really that hard to coordinate meeting times except in the case of random obligations which are given to occur in every ones life at some point. Even then we have still been able to logistically coordinate at least one group meeting a week since the beginning of the semester to keep everyone informed.

In addition Siddhesh and Yanze has the responsibility of maintaining our project website.

As a group we regularly have been commenting and looking over each others work on Google Docs to give constructive criticism. This has been a good implicit tool for everyone as it has caught a lot of bugs in our reports ranging from simple types to the misinterpretation of an idea described by another group member.

Plan of Work

As of the due date of Report 2, we will have completed three deliverables including the Proposal, Report 1, and Report 2. A brief of overview of the three deliverables are as follows:

The Proposal was a brief overview of the group members working on this project as well as a short description of the Health Monitoring System with a detailed explanation of our proposed changes

Report 1 was compiled as our Systems Specification document. It included more specific details of our project such as the Customer Statement of Requirements, enumerated Functional Requirements, Use Case descriptions, and Use Case Diagrams. This included an initial mock-up of the potential User Interface and the Domain Analysis of our problem.

Report 2 encompasses our System Design of the Health Monitoring System. This report holds key design aspects such descriptions to how the classes and modules will interact with each other in the interaction diagrams and Class Diagram and Interface Specification. In addition it also holds our Design of Tests to perform unit as well as integration testing once the coding of each module has been completed.

Developing, Coding, & Unit Testing

The following describes the modules developed or currently under development, and the member primarily responsible for the development of said module:

Login/Registration	:	Amanbir/Siddhesh
Edit/Update profile	:	Jagbir/Swapnil
Parsing Raw Data	:	Jagbir
Total/Active Energy Expenditure	:	Jagbir/Amanbir
Sleep time/lying	:	Ajinkya/Swapnil
Phyisical Activity/Lyingdown Time	:	Siddhesh/Jagbir
Graphs/Charts	:	Amanbir/Jagbir

Coordination of Integration

This coordination effort will be overseen by Jagbir. Jagbir has contributed greatly in producing an efficient scheme of the organization of our simulation program.

Integration Testing

Integration testing will be performed by Ajinkya, Swapnil, Siddhesh to make sure that the code modules combined so far behave and interact properly. The testing will utilize the test cases generated by Ajinkya, Swapnil. Ajinkya , Swapnil will portray as standard users and follow a simple outlined procedure as it was envisioned by us of how to use our software. The simple procedure will be available to everyone in the help section of the GUI, which a user can navigate to from the initial GUI screen.

By performing the integration testing and going through sample cases that would usually be performed by our users, Ajinkya, Swapnil, Siddhesh will be able to easily gauge our applications ease of use and check whether user effort is comparable to what we projected it to be. Our goal is for users to be able to use our application with little to no training. To encourage this, users will be provided with access to documentation such as a User manuals explaining the complete functioning,usage and advantages of the system.

9. References

1. Marsic, Ivan. Software Engineering. 2012.

2. ruegge, Bernd, and Allen H. Dutoit. Object-oriented Software Engineering: Using UML, Patterns, and Java. Boston: Prentice Hall, 2010

3. Bardi, James A. Hotel Front Office Management. Hoboken, NJ: John Wiley & Sons Inc., 2007.

4. http://sensewear.bodymedia.com/

5. http://medicaldevice-network.com/

6. http://google.com,http://Wikipedia.org

7. Ramakrishnan, Raghu, and Johannes Gehrke. Database Management Systems. Boston: McGraw-Hill, 2003.

8. Website Template – sites.google.com.

9. "Java Server Pages". http://baike.baidu.com/view/3387.htm.

10. Ding Dong. Introduction of JSP. 2008

11. Shi Zhiguo, Xue Weimin, Dong Jie. Application Course Of JSP. Oct 2004. TSINGHUA UNIVERSITY PRESS.

12. Russ Miles and Kim Hamilton. Learning UML 2.0. O'Reilly Media. May 2, 2006

13. Sasha Nakhimovksy, Alexander Nakhimovsky. Professional Java XML Programming with Servlets and JSP. Wrox Press Ltd. Birmingham, 1999.

14. "UserInterface". http://baike.baidu.com/view/362528.htm?fromenter=user%20interface.

15. Danielsson ULF; (1990): Convective heat transfer measured directly with a heat flux sensor. Journal of Applied Physiology , 68 (3): 1275-1281