16:332:567 Software Engineering

Biometric Health Monitoring

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

Group 3

Project Members:

Ajinkya Bilolikar

Amanbir Singh

Jagbir Singh

Siddhesh Surve

Swapnil Sarode

Yanze Zhang

Table of Contents

A.Individual Contributions Breakdown.....3
B. Summary of Changes.....4
1. Customer Statement of Requirement.....5
a) Problem Statement.....5
b) Glossary of Terms.....9

3. System Requirements.....11
a) Enumerated Functional Requirements.....11
b) Enumerated Non-Functional Requirements.....13
c) On-Screen Appearance Requirements.....13
d) Acceptance Tests.....15

4) Functional Requirements Specifications.....18
a) Stakeholders.....18
b) Actors and Goals.....18
c) Use Cases.....19
d) System Sequence Diagrams.....28
e) Use Case System Sequence Diagrams.....30

5. Effort Estimation using use case Points.....41
5.1 Unadjusted Use case Points.....41
5.2 Technical Complexity Factor.....45
5.3 Environmental Complexity Factor.....47
5.4 Calculating the Use Case Points..... 48
5.5 Deriving Project Duration from Use-Case Points.....48

6. Domain Analysis.....50

a)Domain Model.....50
b)System Operation Contracts.....59

7. Interaction diagrams.....62

8. Class Diagram and Interface Specification.....71

8.1 Class Diagram.....71

8.2 Data Types and Operation Signature.....72

8.3 Traceability Matrix.....76

8.4 Design Patterns.....77

8.5 OCL Contract Specifications.....77

9. System Architecture and System Design……80

a. Architectural Styles……80

b. Identifying Subsystems……80

c. Mapping Subsystems to Hardware ……81

d. Persistent Data Storage……81

e. Network Protocol……83

f. Global Control Flow……83

g. Hardware Requirements……83

10. Algorithms and Data Structures……84

a. Algorithms……84

b. Data Structures……85

11. User Interface Design and Implementation……87

12. Design of Tests……93

13. History of Work,Current Status and Future Work……99

14. References ……105

A.Individual Contributions Breakdown

Contribution Breakdown:-

	Amanbir Singh	Ajinkya Bilolikar	Jagbir Singh	Siddhesh Surve	Swapnil Sarode	Yanze Zhang
Summary of Changes(5 points)	40%					60%
Customer Statement of Requirements(6points)	50%		50%			
Glossary of Terms (4 points)		50%				50%
System Requirements (6 points)		50%	50%			
Functional Requirements Specification (30 points)	25%		30%	25%	20%	
Effort Estimation (4)		75%	25%			
Domain Analysis (25 points)	30	30	40%			
Interaction Diagrams (30+10 points)	20%	20%	10%	20%	20%	10%
Class Diagram and Interface Specification (10+10 points)	20%	20%	20%	20%	20%	
System Architecture and Design (15 points)			34%	33%	33%	
Algorithms and Data Structures (4 points)			100%			
User Interface (11 points)	50%		50%			
Design of Tests (12 points)			34%	33%	33%	
History of Work, Current Status, and Future Work (5 points)				50%		50%
Project Management (13 points)	33%		67%			
Software Coding	18%	9%	55%	9%	9%	
Total = 200	38.3	31.30	56.10	31.40	31.20	11.70

B. Summary of Changes

Revisions Full REPORT:

- In Problem statement page 7, we have included a section of graphical data which gives description of all the graphs.
- In requirements on page 11, we replaced the req 9 because it was not used in our project.
- We have also added a line of highest priority requirement of 3a on page 11.
- In acceptance test on page 14 and 15, we have added additional details in req 1a, req2a and req 5b.
- Traceability Matrix: Updated the traceability matrix, now also including the weights.
- We have added Object Constrained Language (OCL) Contract Specifications.
- System Sequence Diagram: Provided with an alternative scenario for the use cases along with the original success scenario and added names to every diagrams.
- We have included a new section of effort estimation using use case points on page 39.
- In interaction diagrams we added the names which were missing for the use cases.
- In class diagram on page 80, we updated and changed the diagram and added user interface class and changed the traceability matrix.
- In user interface we have added the details of how every tab on the screen and added various details about types of graphs.
- In design of tests we have provided examples of the tests performed for very case.
- We have updated the plan of work and added the diagram from Microsoft project.
- We have updated the references in the last section.

1. Customer Statement of Requirements

a) Problem Statement

Biometrics devices can take unique information about you from your eye, or your hand prints, or your thumb prints and use it to identify you. This information can be used to obtain the valuable healthcare information. The personal health monitoring of each individual is considered very important because of rise in health problems in today's world. The increasing stressful lifestyle is taking maximum toll on the public health. With the ever increasing queues at hospitals and ever increasing number of patients, the doctor fees have sky-rocketed which is affecting especially those patients who cannot afford the fee or who are not suffering from major ailments but get to know so only after paying a hefty fee to the doctor. The researchers and surveys often demonstrate that most of the major health ailments are the result of careless attitude towards the minor health ailments. Majority of these issues can be solved by just following a good diet, proper sleep pattern and regular exercising. But how does a patient know what diet is good or what exercise he/she should follow and more importantly whether the plan that he is following is working effectively for him. The absence of such a mechanism makes the task of patient a difficulty, thus landing him with an option either to go to the doctor that means heavy fee or ignore the ailment that is more dangerous.

Moreover the increase in patients has also led to the decrease in the relative number of doctors per patient which results in vicious cycle where ignored or delayed diagnostics of an ailment makes the patient more dependent on doctor’s check-up. But is it necessary that every time the user faces some issue or requires advice, that are not that serious, like somebody wants to get rid of some extra fat or somebody wants to improve his/her stamina, he/she is required to go to doctor?

Well, though it is advisable to visit the doctor whenever possible, but as discussed above if due to unavailability of specialist due to some reason, the Health Monitoring Devices offer and effective alternative. These days it is advisable to each individual to monitor and maintain good health by using biometric health monitoring devices and keep modifying their diet so as to improve their health stats. Thus there is a need for software that utilizes the data available from the device, uploads it to the website, gets feedback from the doctors via internet and show health reports. Doctor should be able to get data anytime he wants for analysis.

The biometric health devices can record the various data like metabolism rate, sleep hour, sedentary activity while being in contact with the user and this data from user can be used by physicians to recommend any changes to user’s routine. Our project attempts to use the information obtained using such devices to give the detailed analysis of health of a patient/individual that can help in getting a prompt and timely advice from a doctor. Currently healthcare monitoring is extensively doctor depending .Our System is basically designed for those customers who can self-monitor their health indicators to check the progress that they are making

through a diet plan change or exercise routine modification. An option will also be provided to customer to send the report to the doctor for expert advice in case the customer feels he/she is not completely satisfied by the diet routine, sleep hours or any other general habit he is acquiring. Different types of health indicators would be used for the analysis.

Assumption:-

1. The User already possesses a biometric health device that would be used to record the health parameter. These recorded parameters are internal to the device and using our application software user would be provided with some real time user friendly parameters that are defined under parameters sections and these parameters would be displayed on user screen upon request.

2. This device need to be worn by the user/patient on his arm for a particular duration of time, for which he wants his health parameters to be monitored.

3. User would be able to connect this device to his/her PC and would be able to generate the excel sheet in agreed format using the device software, this excel would be kept in user machine and application developed by our team will process data in this excel once after uploading it.

Device Operation:-

The device used during this project is The Sense Wear Armband[6][7]. The user will be wearing this device for some period of time. This device contains sensors that collect the body movements of the user and record data based on those body movements and body temperature and sweating to provide information about what type of activity user is doing. Details of device monitoring are given below:-

- Motion: The Armband contains an accelerometer, a device that measures motion.
- Steps: The Armband counts your steps, using the accelerometer to measure the distinct patterns created by walking and/or running.
- Galvanic Skin Response: This measures the electrical conductivity of the skin, which changes in response to sweat and emotional stimuli.[8][6]
- Skin Temperature: A sensitive electronic thermometer measures the temperature of the skin.
- Heat Flux: Measures the amount of heat dissipating from the body.[17]

Once the user has worn the device and he/she wishes to get his activity log over a period of time using some more real-life parameters, user can connect this device to his Personal computer and

the device software(the software internal to the device and not this project software) will provide user with an excel file that would contain the information recorded by the device. This information contained in the excel would be used by our application to display the parameter explained later in this text.

Once the user has the generated this excel, the user would log into his account on our application and reach his/her homepage. User will have an option to upload this excel and once the excel is uploaded to proceed with the execution to process the information. Once the processing is done the user will be provided with the information mentioned in next section.

Parameters provided :-

1. **Total energy Expenditure**: - This field will display the total no. of calories burned by the user over a period of time.The vigorous activities would lead to faster rate of burning and this rate would be depicted better through the graphical repersentation.

2. **No. of Steps** : - Total no. of steps taken by the user over the window period of wearing the device.

3. **Lying Down Time**: -Total no. hours the user has spent lying down.This time will also include the sleeping time

4. **Sleeping time:** - The total time that user has spent while sleeping.

5. **Sedentary:** - The total time spent while sitting by the user.

6. **Physical Activity Duration**: - The total time for which the user performed the physical activity over a period of time. Any activity with 3 MET(see glossary) would be considered as Physical Activity.

Graphical Data:- For making the system more user friendly to all the users, the above data would be shown with the help of different meters and graphs. We believe it would prove very useful to the doctor especially while providing the feedback about the health condition of a Patient. It is planned to employ the three major graphs Bar Graph, Line graph and Meter Graph.

The System will basically consists of three different users that would be interacting with the aim of providing better healthcare service through mutual utilization of self-monitoring and the consultation from a specialist. The mutual interaction exists because user will have the option of sending the data for analysis to the doctor, getting the feedback and then acting on his advice. The three users are:-

1. Patient

2. Doctor

The Patient will register himself with system initially by providing the various personal details that includes his name, age, sex ,etc., along with the doctor's name with which he wants to consult ,if at all. The registration part is mandatory before the user is able to use the system for his task completion. Then the user proceeds with login using his/her unique id and password. Upon logging into the application the user will have a window screen with few options. One of the option user will have is to upload the file that the system will be using to display the data to the user interface. This data file is generated as a result of the data collected by the health monitoring device that the user wears on his arm-band for a particular duration of time. This file can be uploaded into the user desktop by connecting device through Bluetooth or USB cable. It is assumed that the data would be available in the Excel File. The data in this file is not present in user friendly manner and therefore is not of much use to the customer. The application that is being developed will make this data suitable for user display and analysis and also make provisions to send the data to the doctor. Once this file is uploaded into the system, system extracts the data. This would result in the user getting a clear picture of all the parameters which were recorded by the device over that particular period of time.

The total duration that the user slept during that time would also be provided, including other activities like number of steps taken, that is especially useful for patients aiming at weight reduction monitoring. The working professionals whose workplace activities involve lot of sedentary activities would be interested in knowing the time that they spent sitting. The Sedentary parameter would be available for the user to monitor that. Such diversified parameters would make the application cater to the requirements of various categories of people, thus achieving higher customer base. The customer should have an additional feature to view the whole data captured in a graphical form for a better analysis as we believe graphs are much easier to analyze and understand as compared to other forms, hence making the customer's and doctor's job much easier.

The role of doctor is vital to any healthcare process and we would provide the provision in our project to get doctors feedback .The patient should be able to send the report to the doctor, who was specified by the customer during his/her registration. But this functionality should be entirely upon the discretion of the user that whether he/she chooses doctor's analysis or not. If the customer feels, he/she is satisfied with his improvement and performance after monitoring the data provided by our application, then patient can logout of the system.

The Doctor should be able to register for the application using the same process and then log in the system using his unique credential. Once the doctor gets logged in, he/she should be able to see the data of a patient through a user interface that allows him to select the patients. Once the patient gets selected the data should be available for the doctors analysis as it was presented to the

customer, with the exception that doctor should be able to add his/her comments or feedback after seeing the details of the patient thus issuing his/her advice. It must be noted that the data available for monitoring should not be editable either by the patient or the doctor. Once the doctor's feedback is posted, user/patient would be able to see that once after logging into his account.

Doctor Analysis and Feedback:- The doctor would analyze the parameter mentioned in above specific to particular patient. Doctor would take into consideration the information of patient medical history maintained in patients profile during the registration of patient.

For Ex.:- A patient is suffering from obesity and is undergoing process to reduce his/her weight, the doctor would check the weight from user profile and compare the amount of physical activity that user is undergoing. Based on whether the patient is doing enough or less, doctor can suggest his/her advice. For this doctor would be provided a field in his profile and after checking the user data he/she would be able to enter the prescription/advice in that field and send to the user. This would provide for an efficient and productive communication between the patient and doctor.

The designed application will be extremely user friendly since the users can be of any age. Simplicity in presenting the application to the user will be the key to success of this system.

2. GLOSSARY OF TERMS:

User- A person who wants to upload his personal health information on the web site.

Doctor- A person who monitors the health information of the user and provides feedback to the user.

Database – Entity that stores all the system's information.

Website – An interface that the user and doctor can use to register, and upload or download the health information.

Graph- A graph is an abstract representation of a set of objects where some pairs of the objects are connected by links.

Report- A report is a textual data of the user's health made with the specific intention of relaying information or recounting certain events in a widely presentable form.

Acknowledgement- A feedback given by the doctor to the user to make changes in his/hers diet.

Software application- An application is computer software designed to help the user to perform specific tasks which manage and integrate a computer's capabilities.

Web Server- a server is a physical computer dedicated to running one or more services to serve the needs of the users of other computers on the network.

Unique ID- Identification is a process whereby the subject assimilates an aspect, property, or attribute of the other and is transformed, wholly or partially, after the model other provides.

Diet- the sum of the food consumed by an organism or group.

Heart Beat- Heart rate is the number of heartbeats per unit of time, typically expressed as beats per minute (bpm).

Energy Expenditure- It refers to the amount of energy (calories), that a person uses to breathe, circulate blood, digest food, and be physically active.

Number of Steps- It is the total number of steps travelled by the user to cover a particular distance.

Threshold- A limiting value of a parameter below which the health of the user gets degraded.

MET : - The **Metabolic Equivalent of Task (MET)**, or simply **metabolic equivalent**, is a physiological measure expressing the energy cost of physical activities and is defined as the ratio of metabolic rate (and therefore the rate of energy consumption) during a specific physical activity to a reference metabolic rate[8].

3. System Requirements

a) Enumerated Functional Requirement

The functional requirements are tabulated below [1]

REQ1a	5	The system shall allow new users to register an account on the website.
REQ1b	5	The doctor and the patient have to register themselves with the administrator.
REQ2a	5	The system should provide a specific Login ID to every Registered user and doctor.
REQ2b	5	The system should be able to differentiate between the user's and doctors while login.
REQ2c	4	The system shall identify the user based on his login ID and Password.
REQ3a	5	The system should be able to upload the data in excel form using a user interface upon user request and process it.
REQ3b	4	The system should be able to convert the rawdata from excel and store it in the database.
REQ4a	2	The system should be able to display the data to the user in a tabular form. This data would include energy expenditure, metabolic rate, sleeprate, etc.
REQ4b	2	On user request the System should be able to display the graphs corresponding to data displayed in the REQ7.
REQ4c	1	The system should provide the user with the option to save the data on his PC.
REQ5a	4	The system should send the data to the doctor.
REQ5b	3	The user should provide the doctor name he wishes to send the data to during registration.
REQ6a	5	The system should display the list of all the user's mapped to a particular doctor.
REQ6b	5	The system shall allow the patient data to be received by only one doctor to avoid conflict of feedbacks from doctors.
REQ6c	5	The system shall allow the doctor to view only one patient data at one point of

		time.
REQ7a	4	The system should display the data of the selected patient.
REQ7b	3	The data can be either in tabular or graphical format.
REQ8a	3	The system should check if the feedback from the doctor is available for the particular patient.
REQ8b	5	The system should be able to display the feedback from doctor to the patient.
REQ8c	5	The system should allow the patient to view the feedback written by the doctor.
REQ9	3	The doctor can write the date in the feedback so that patient knows which date he sent the data.
REQ10	5	The system should not allow the patient or doctor to modify the data.

The REQ1a has a high priority as only registered patients and doctors will be able to communicate with the system. The REQ2a also has a high priority as unique identity has to be given to the users. REQ2b is paramount as during the login, the system has to be able to differentiate between a patient and a doctor. Depending on the type of user separate windows and functionalities are provided. REQ3a has a high priority because our application can only take files which are in (.xls or excel) format. REQ4c has the least priority since the patient being able to save the report to this pc is not our main agenda. REQ9 has been given a lower priority as we may have to include its functionality depending on the time and resources available. REQ10 implements that the users should not be able to change the data as it would result in the incorrect analysis of the patient information.

THE NOT LIST-What This Project is not about-In this project we only work on the information captured by the health monitoring device. We are not concerned with how the device works or how the user actually manages to upload his data onto the system. We also do not pass the patient record to all the registered doctors. Only the doctor whose name has been entered by the patient gets the patient record. The doctor does not receive an alert on his mobile phone whenever a patient uploading his data, references that particular doctor. He only gets to know about it when he logs into his account.

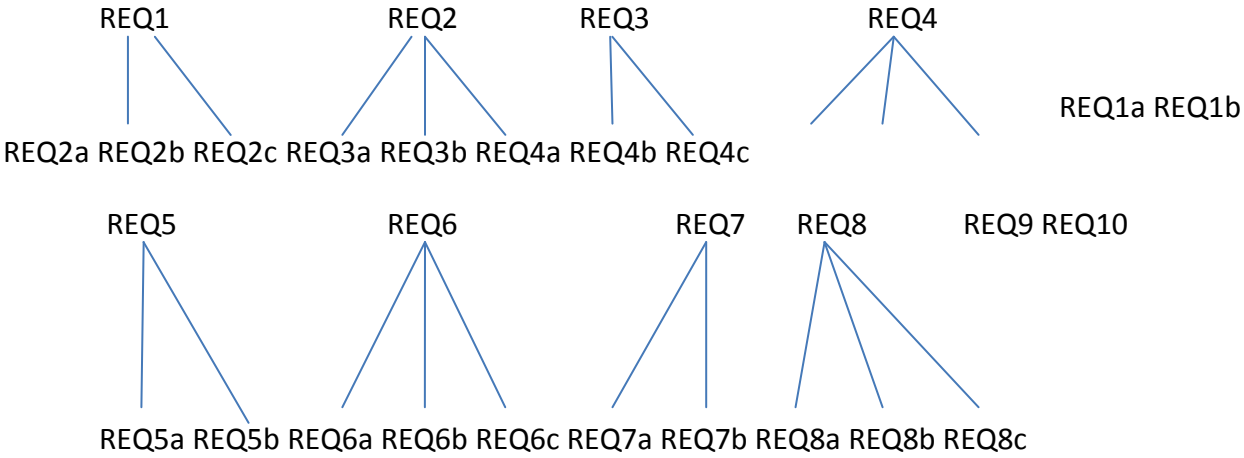
Business Policy:

TTT-BP01: No more than one patient data can be viewed at a time by a doctor.

While the doctor is viewing a patient's record, he must not be allowed to open another patient's record. Only after the doctor has given his feedback for the current patient can he access the next patient's record.

TTT-BP02: The patient can view the feedback only after the doctor uploads it.

If the doctor still has not prescribed his medication then the patient should see "no comments available" on his screen.



b) Enumerated Non Functional Requirements:

The non functional requirements are tabulated below [1]

ID	PW	Requirement
REQ11	5	Authorized access to user is provided as a patient, doctor or System Administrator
REQ12	4	User is allowed to change the password and notified when he attempts to change his password.
REQ13	3	User is logged, off if he is idle for most of his login duration.
REQ14	2	User satisfaction is ensured by providing quick computing, and importing and exporting data.
REQ15	4	The number of transactions per hour does the system need to be able to handle
REQ16	4	The amount of data does the system need to be able to store

c) On-Screen Appearance Requirements:

On-screen requirements are stated below [1]

ID	PW	Requirements
REQ17	5	The system is designed such that it can work on Windows, iOS and Linux operating systems.
REQ18	3	The user is shown his expected results onscreen as well as provided with an option for printing.
REQ19	2	The system must have a consistent look across different browsers and screen resolutions.

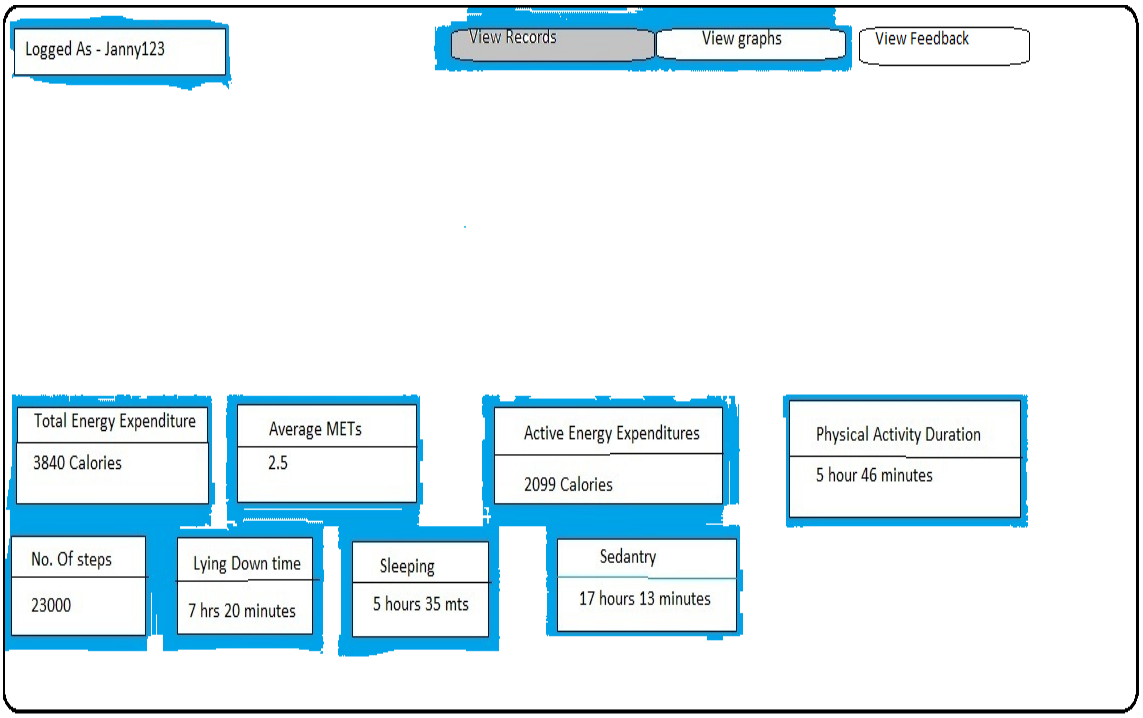


Figure above shows a customer-provided initial sketch of the user interface appearance.
This appearance pertains to patient view of homepage. The screen Homepage contains different tabs displayed as explained below.

Tabs:-

1. **View Records** : - This tab will enable the user to see the records in tabular form. This is also the default view and the records would be displayed in this form until the user choose to go to graphical form.

2. **View Graphs** : -This tab will show the user parameters in graphical form to provide a better idea of user activity over a larger span of time.

3. **View feedback** :- This button is used by the patient for viewing the feedback of the doctor on his reports.

d) Acceptance Tests

Acceptance test case for REQ1a:

ATC 1a.01: Ensure connectivity between user and network which hosts the webpage.

ATC 1a.02: Ensure connectivity remains intact until registration of user completes.

This is imperative as connectivity has to be maintained so that the user can upload his data. We assign users with a specific session which makes sure that if the user is unresponsive for a long time then he is logged out.

Acceptance test case for REQ1b:

ATC 1b.01: The system must make sure that all the required fields are filled up the registering user.

Acceptance test for REQ2a:

ATC 2a.01: Every user (patient or doctor) should be assigned with a unique Login ID to prevent overlapping of login information.

ATC 2a.02: Ensure that the user's password is associated with his respective Login ID.

It should be made sure that same user id is not given to two users. The database has to be checked to make sure that the id being entered by the user is not already taken.

Acceptance test for REQ2b:

ATC 2b.01: Differentiate between a Doctor, a User or an Administrator during login.

Acceptance test for REQ2c:

ATC2c.01: After login ensure that correct user profile is selected based on login information.

Acceptance test for REQ3a:

ATC 3a.01: Ensure every user can upload his data in excel and process the needed parameters from the user interface.

ATC 3a.02: Ensure that the data uploaded by a user is associated with his profile.

Our application only accepts files of .xls format and the file format is checked before the file can be uploaded. This prevents non .xls files to be present in the system.

Note that the process in ATC 5.01 can be a process like check how many hours of sleep the user had or keep a tab on sedentary work hours. The profile of user in ATC 5.02 is his registration information.

Acceptance test for REQ3b:

ATC 3b.01: Ensure that the data from excel is entered into the database where it is stored for analysis.

Acceptance test for REQ4a:

ATC 4a.01: Ensure validity of data stored display it to user and get confirmation from the user.

ATC 4a.01 can be a simple test where the user confirms that the data entered by him which was later transferred to the database is the same and correct data.

Acceptance test for REQ4b:

ATC 4b.01: Ensure correct graphical representation by displaying graphs of parameters requested by the user.

Acceptance test for REQ4c:

ATC 4c.01: Ensure backup of the data by allowing user to have one, for future references.

Acceptance test for REQ5a:

ATC 5a.01: Ensure that the doctor associated with users profile receives the data when user sends it.

Acceptance test for REQ5b:

ATC 5b.01: Ensure that the patient selects the doctor name from the drop down menu.

The patient has to enter the doctor that he wants to be mapped to while registering. This makes sure that only that particular doctor can view the patient data.

Acceptance test for REQ6a:

ATC 6a.01: Ensure that the list of users handled by a doctor is updated every time there's a new registration with the same doctor.

ATC 6a.02: The Doctor should be able to view any of his user's (patient's) data.

Acceptance test for REQ6b:

ATC 6b.01: Ensure that the patient data is received by only one doctor.

ATC 6b.02: The doctor's comments are unique to that patient so only one doctor shall comment for a particular patient.

Acceptance test for REQ6c:

ATC 6c.01: Ensure that only one patient data can be accessed at a time by the doctor.

ATC 6c.02: The doctor should write his comments to the current patient before being allowed to access the next patient.

Acceptance test for REQ7a, REQ7b:

ATC 7.01: Ensure that the doctor is viewing the data of the correct user.

ATC 7.02: Doctor should be able to choose whether to view data in tabular format or graphical format.

Acceptance test for REQ8a:

ATC 8.01: Ensure that the doctor gives feedback to the correct user.

The test confirms that the comments made by doctor on profile he is currently working on are linked to the correct user.

Acceptance test for REQ8b:

ATC 8a.01: Ensure that feedback provided by doctor is associated with user under observation.

Acceptance test for REQ8c:

ATC 8c.01: Ensure that user gets the feedback provided by the doctor.

Acceptance test for REQ10:

ATC 9.01: Ensure that data is not altered once stored on the database either by the patient (user) or by the doctor.

4. Functional Requirement Specifications

a) Stakeholders

- Patient
- Doctor
- Administrator
- Emergency Services
- Web page designers
- Database Managers

b) Actors and Goals:

- Patient
  - Performs his daily routines with a device attached to his body.
  - Registers himself with the acknowledgment from administrator.
  - Uploads the data collected by the device into the database.
  - Views his health readings.
  - Views the suggestions prescribed by doctor.
- Doctor
  - Registers himself with the acknowledgment from administrator.
  - Views the patient record uploaded.
  - Monitors if the patient condition is improving or not.
  - Prescribes solution to the patient by writing a report.
- Administrator
  - Stores patient and doctor details while registering.
  - Creates the database tables
  - Upgrades the application
  - Monitors the application

Participating:

- Device
  - Captures the patient activity and stores it in the form of data.
- Web Page
  - Allows the doctor and patient to register with the system.
  - Allows the doctor and patient to interact.
- Database
  - Maintains a record of the patient registration.
  - Maintains a record of the doctor registration.
  - Maintains a record of the patient data and doctor feedback.

c) Use Cases

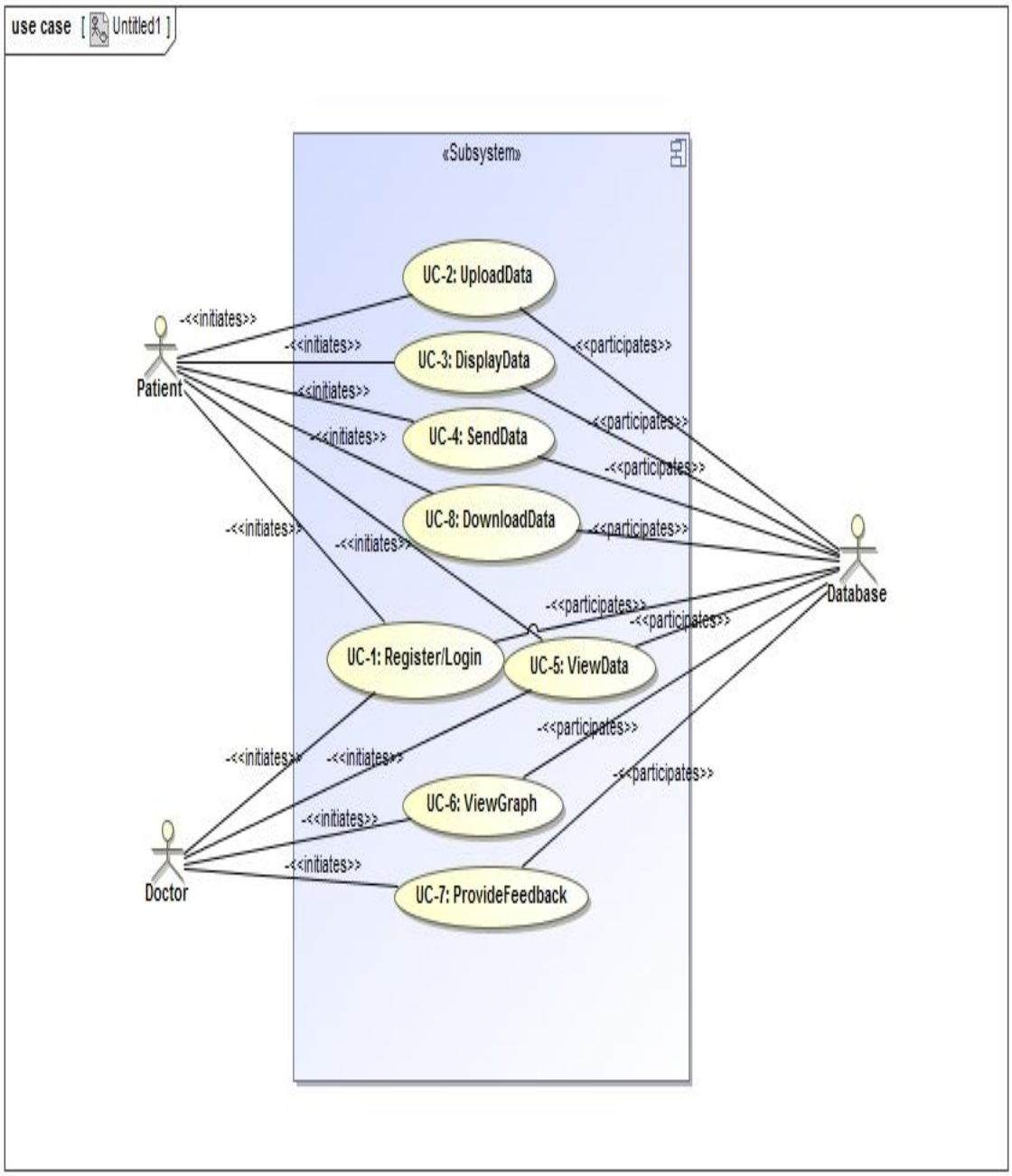
Use cases [1] planned for our project are described below

i. Casual Description:

Use Case	Name	Description	Requirement
UC-1	Register/login	To Register a user/doctor to the website	REQ1a, REQ2a, REQ2b, REQ2c,REQ11,REQ12,REQ13,REQ15
UC-2	UploadData	To upload the raw data file for system to proceed with extraction	REQ3a, REQ3b,REQ16
UC-3	DisplayData	To display the data like energy expenditure, sleep rate, etc. in user friendly manner	REQ4a,REQ4c, REQ10,REQ17,REQ18,REQ19
UC-4	SendData	To send the health information to the doctor	REQ5a,REQ5b,REQ14
UC-5	ViewData	To view the health information of a user by the doctor	REQ7a,REQ7b,REQ6a,REQ6b
UC-6	ViewGraph	To plot the graph of the data	REQ4b, REQ7a,REQ7b

UC-7	ProvideFeedback	To provide the desired feedback to the user	REQ8a,REQ8b
UC-8	Downloaddata	To get the feedback from the doctor and save it.	REQ8c

ii. Use Case Detailed Description:-



Use-Case Diagram

Use-case Details :-As in UC1,User will try to login the application with his/unique username and password if he is already registered with the system. The application/browser will communicate with the database to authenticate the user. If user is authentic he/she will be allowed to continue his/her session. If the credentials are mismatching, the user will be returned with an error saying the username and password do not match. The new users will have to register with the application and information will be stored in the database. Like UseCase 2 shows correctly logging in with username and password, the user will be able to upload the data in excel sheet. This excel should be in proper format as specified with device specifications in customer requirements. This data uploaded will be saved in the system and the acknowledgment would be sent to the browser that the data is correctly uploaded and saved in the system. In case of failure, proper error would be reported. The user would be able to process the file and all the specified parameters would be shown on user screen. In Use case4, the user after viewing his health parameters would be able to send his report to the doctor for getting feedback on his progress. User will choose the button and send report and all his details would be sent to the doctor. A message acknowledging the data sending would be sent to the patient screen. In case of failure the reported error with failure message would be displayed. Doctor would be able to see the data in tabular and graphical forms as depicted in use case diagram in Use Case 5 and 6 .Once after the data is analysed by the doctor , the doctor will write his feedback comments in the field provided for feedbacks. After entering the comments, when saved. The database would be updated with the doctors comments and are read for user view. User would be able to see this feedback in his profile after logging in.

- **Use case 1: Register/login**

  Initiating Actor- User

  Actor Goal- To register a new user, user should be able to register himself with the application

  Requirement Addressed - REQ1a, REQ2a, REQ2b, REQ2c,REQ11,REQ12,REQ13,REQ15.

  PRE-CONDITION- No precondition

  POST CONDITION-The user is successfully registered with the system

  Flow of events :

  →The user, if new, requests to registers himself with the system
  ←The system asks for the personal details of the user.
  → User enters the personal details including chosen credentials, i.e Username/password and submit the registration form
  ←System differentiates between different users, and accordingly saves them in the system database and registration is successful.

- **Use case 2: UploadData**

  Initiating Actor- User

  Actor Goal- To upload the raw data file in the system so that data can be extracted to a user friendly form.

  Requirement Addressed- REQ3a, REQ3b,REQ16

  PRE-CONDITION- User is already registered

  POST CONDITION-The raw data file is uploaded and ready to use for the system.

  Flow of events:

  →user logs into the account
  →user request the system to upload the raw data file from the local upload address
  ←the file is successfully uploaded into the system

- **Use Case 3: DisplayData**

  Initiating Actor- User/Patient
  Actor Goal- To extract the data from the user uploaded file and display it in user-friendly manner

  Requirement Addressed - REQ4a,REQ4c, REQ10,REQ17,REQ18,REQ19

  PRE-CONDITION- the user has already uploaded the raw data file to the system

  POST CONDITION-the data is displayed to the user on screen

Flow of events

→ the patient triggers the data extraction from uploaded raw file by clicking the execute button

← system displays the data for user analysis in tabular form

→ user requests data in graphical form

← data is displayed in graphical form

- **Use Case 4: SendData**

  Initiating Actor- User/Patient
  Actor Goal- To send the data displayed in UC-3 to the doctor for analysis

  Requirement Addressed- REQ5a,REQ5b,REQ14

  PRE-CONIDTION- the user has already extracted the data and data has been displayed on the system

  POST CONDITION - data displayed will be saved in the database for doctors analysis

Flow of events:

→ the patient opts for sending the data to the doctor
← system saves the data for the customer

- **Use Case 5: ViewData**

  Initiating Actor: Doctor
  Actor Goal: To analyze the data reported by the user through application and user interaction

  Requirement Addressed: REQ7a,REQ7b,REQ6a,REQ6b

PRE-CONDITION: Patient has requested the data to be sent to the doctor and asked for feedback.

POST CONDITION – The Doctor Selects the desired patient for providing feedback.

Flow of events

→ The doctor registers his account on the webpage.
→ The doctor makes login to his account by using his login ID and password.
→ The doctor selects the patient from an available list of patients.

- **Use Case 6: ViewGraph**

Initiating Actor: Doctor

Actor Goal: to analyze the data reported by the user through application using graphical representation.

Requirement Addressed: REQ4b, REQ7a,REQ7b

PRE-CONDITION: The data for specified patient is already present in tabular form.

POST CONDITION: The data is displayed using the graphical representation

Flow of events

→ Doctor requests the data of that patient to see.

←Data is displayed in tabular form.

→ Doctor requests the data of that patient in graphical form.

←Data is displayed in graphical form after analysis.

- **Use Case 7: ProvideFeedback**

Initiating Actor: Doctor

Actor Goal: To provide the feedback on user health after analyzing the data available for that patient in tabular and graphical form

Requirement Addressed: REQ8a,REQ8b

PRE-CONDITION: The data for specified patient is already present.

POST CONDITION: Feedback comments would be saved in the system for patient use.

Flow of events

→ The doctor analyze the data using different forms.

→ Doctor posts his/her feedback comments after the data analysis

←Feedback is saved in the database.

- **Use Case 8: DownloadData**

Initiating Actor: Patient

Actor Goal: to see the feedback on user health after analyzing the data available for that patient in tabular and graphical form

Requirement Addressed: REQ8c

PRE-CONDITION: The Feedback comments for specified patient is already present in system

POST CONDITION: Patient would be able to see the feedback comments through user interface.

Flow of events

→ The user requests to see the feedback comments.

←Feedback comments are displayed on the user interface.

i. Traceability Matrix:

	PW	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8
REQ1a	5	X							
REQ1b	5								
REQ2a	5	X							
REQ2b	5	X							
REQ2c	4	X							
REQ3a	3		X						
REQ3b	4		X						
REQ4a	2			X					
REQ4b	2						X		
REQ4c	1			X					
REQ5a	4				X				
REQ5b	3				X				
REQ6a	5					X			
REQ6b	5					X			
REQ6c	5								
REQ7a	4					X	X		
REQ7b	3					X	X		
REQ8a	3							X	
REQ8b	5								
REQ8c	5								
REQ10	5			X					
REQ11	5	X							
REQ12	4	X							
REQ13	3	X							
REQ14	2				X				X
REQ15	4	X							
REQ16	4		X						
REQ17	3			X					
REQ18	3			X					
REQ19	2			X					
MAX POW		5	4	5	4	4	3	3	2
TOTAL POW		35	11	16	9	17	9	3	2

Figure 4.c.(iii) Traceability Matrix

According to the traceability matrix[1] derived above we see that the UC-1 has the highest priority followed by the UC-3. Hence we will be focusing to accomplish the use cases according to the priorities derived as above.

d) SystemSequenceDiagrams

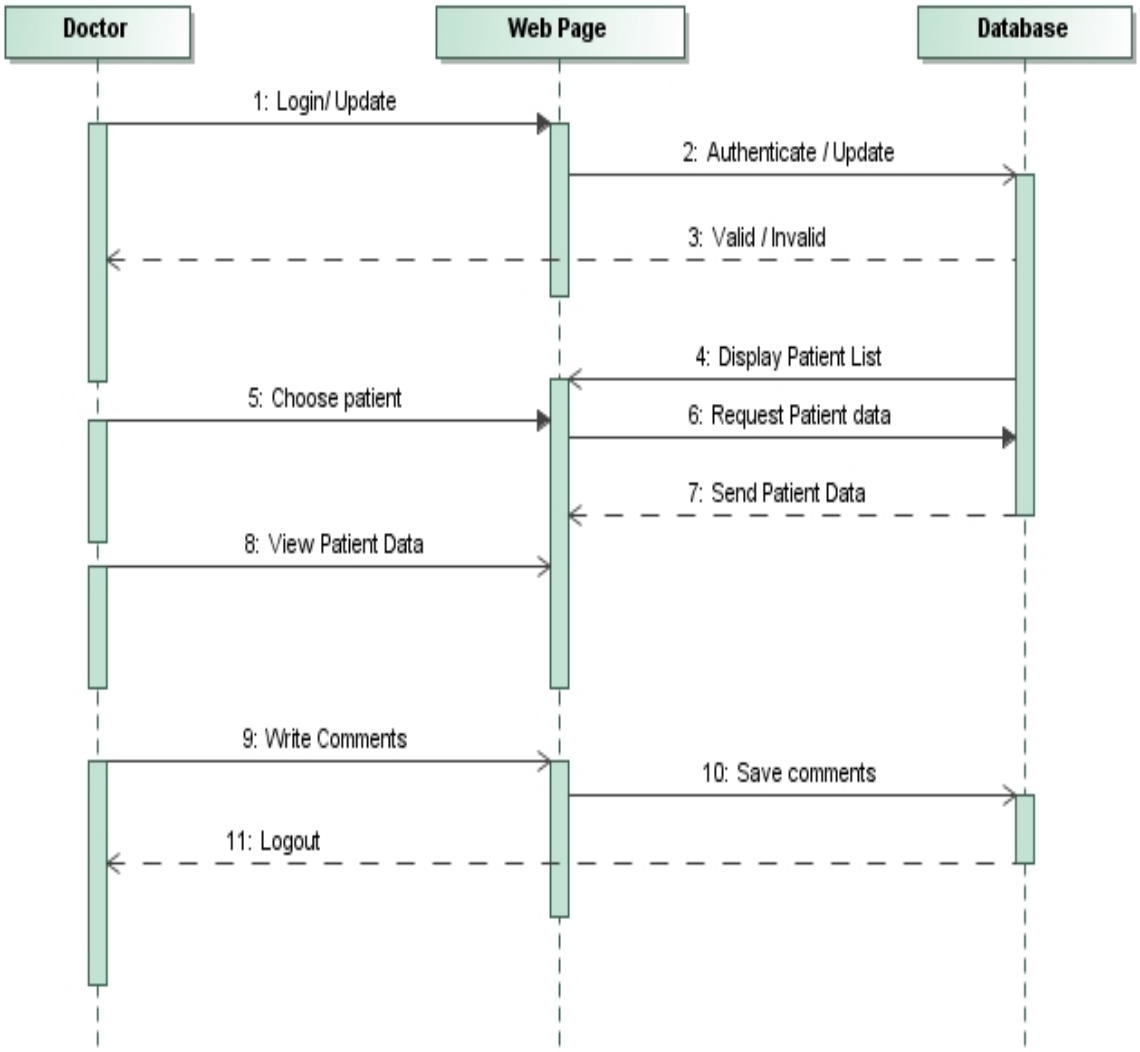


Figure 4.d.1 Sequence followed for User: Doctor

All the sequence diagrams are as per UML specifications[4][5]

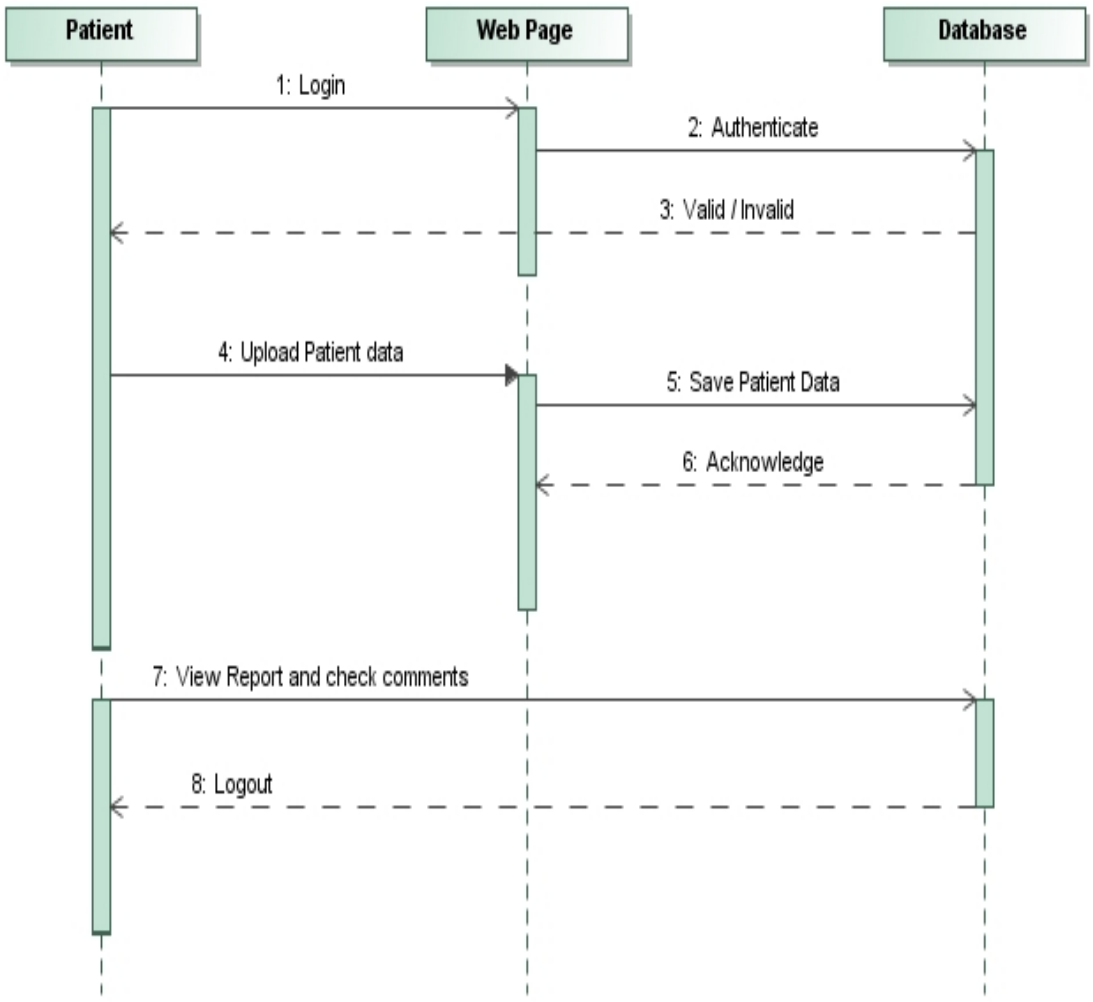


Figure 4.d.2 Sequence followed for User: Patient

e) Use Case System Sequence diagrams

Use case 1: Register/login

In this case User will try to login the application with his/unique username and password if he is already registered with the system. The application/browser will communicate with the database to authenticate the user. If user is authentic he/she will be allowed to continue his/her session. If the credentials are mismatching, the user will be returned with an error saying the username and password do not match.The new users will have to register with the application and information will be stored in the database.

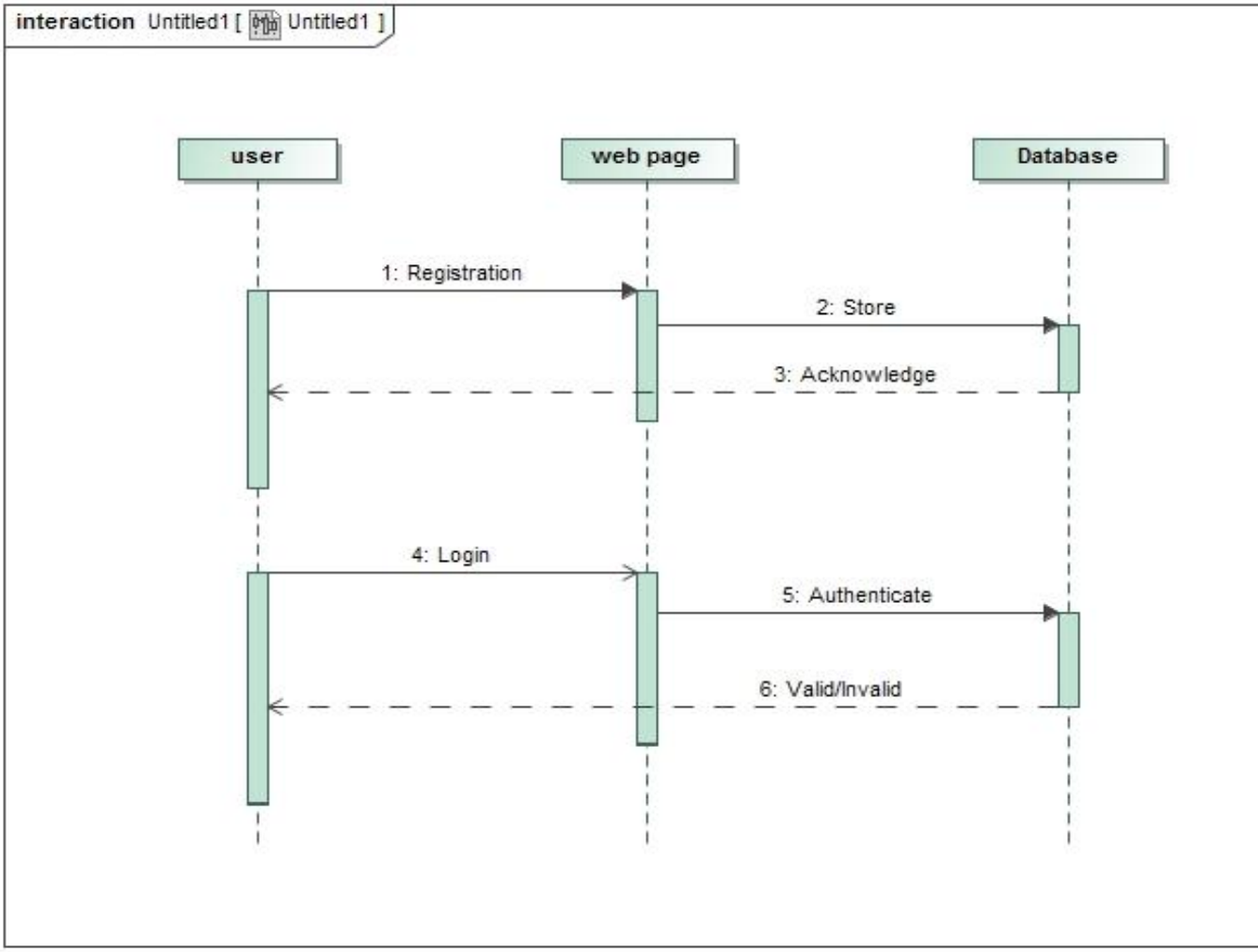


Figure 4.e.1 Register/Login

Alternate Scenario:

In the alternate scenario, a new user(doctor, patient) can register with a unique id. If the database finds that this ID already exists then it sends a message back to the user asking him to register using a different ID as its already taken.

During Login, the database authenticates the validity of the username and password. The database sends back a reply saying that the username and password do not match.

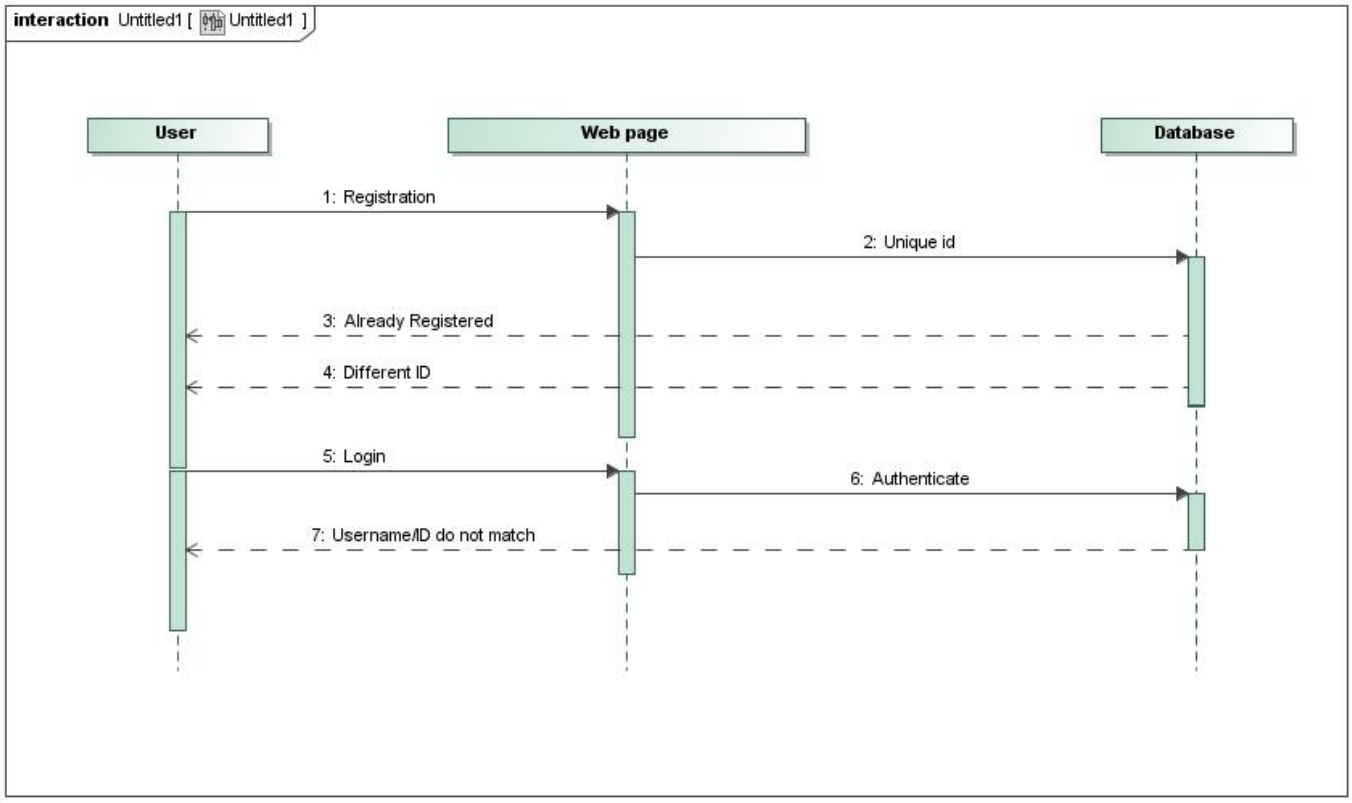


Figure 4.e.2 Register/Login Alternate

Use case 2: UploadData

In this case, after correctly logging in with username and password, the user will be able to upload the data in excel sheet. This data uploaded will be saved in the system and the acknowledgment would be sent to the browser that the data is correctly uploaded and saved in the system. In case of failure, proper error would be reported.

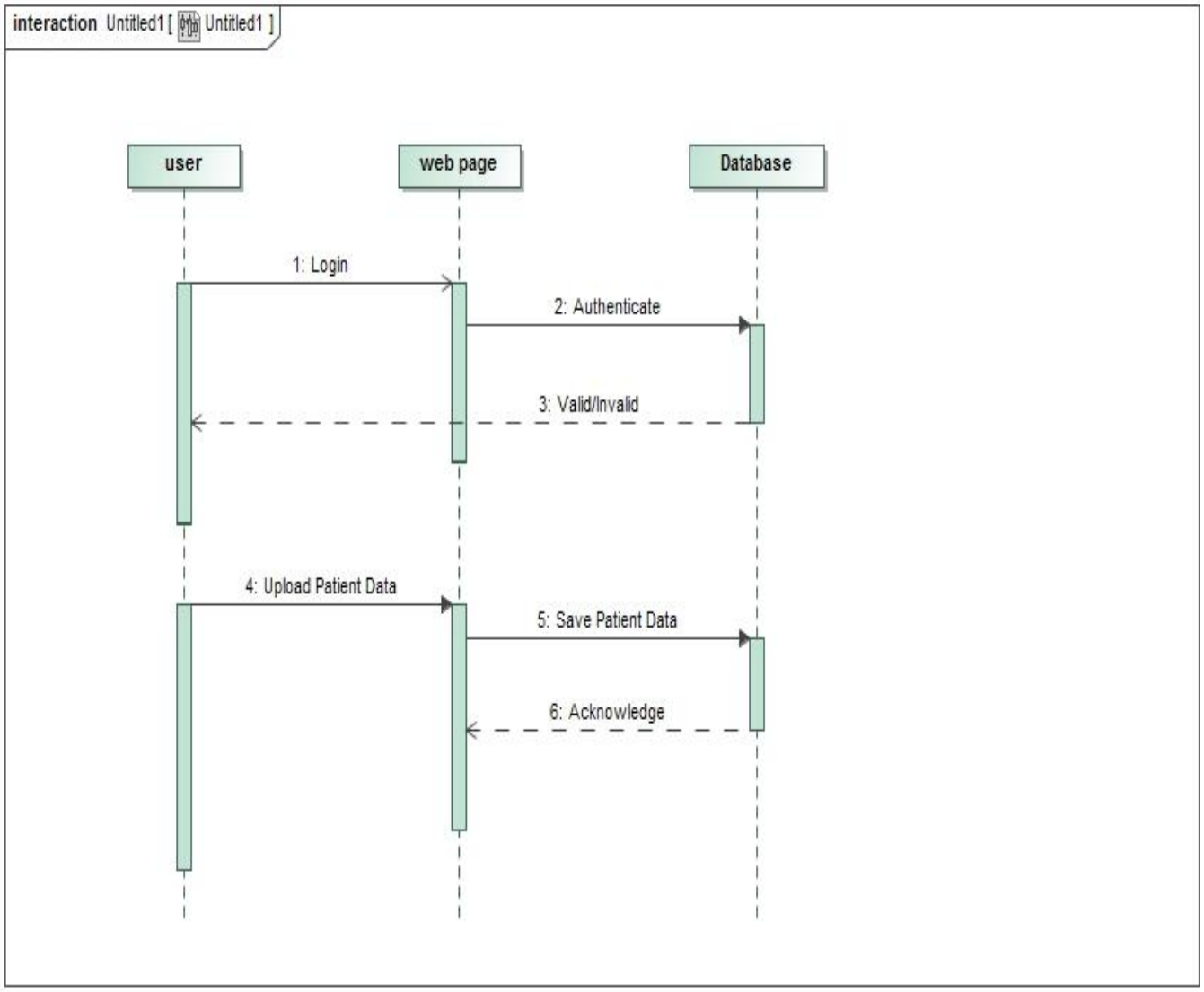


Figure 4.e.3 Upload Data

Use Case 3: Display Data

After uploading the data if the user wish to see the parameters defined customer requirements in graphical form, user will be able to chooses that and the parameters would be displayed in graphical form, displaying how the user activity has varied over a period of time.

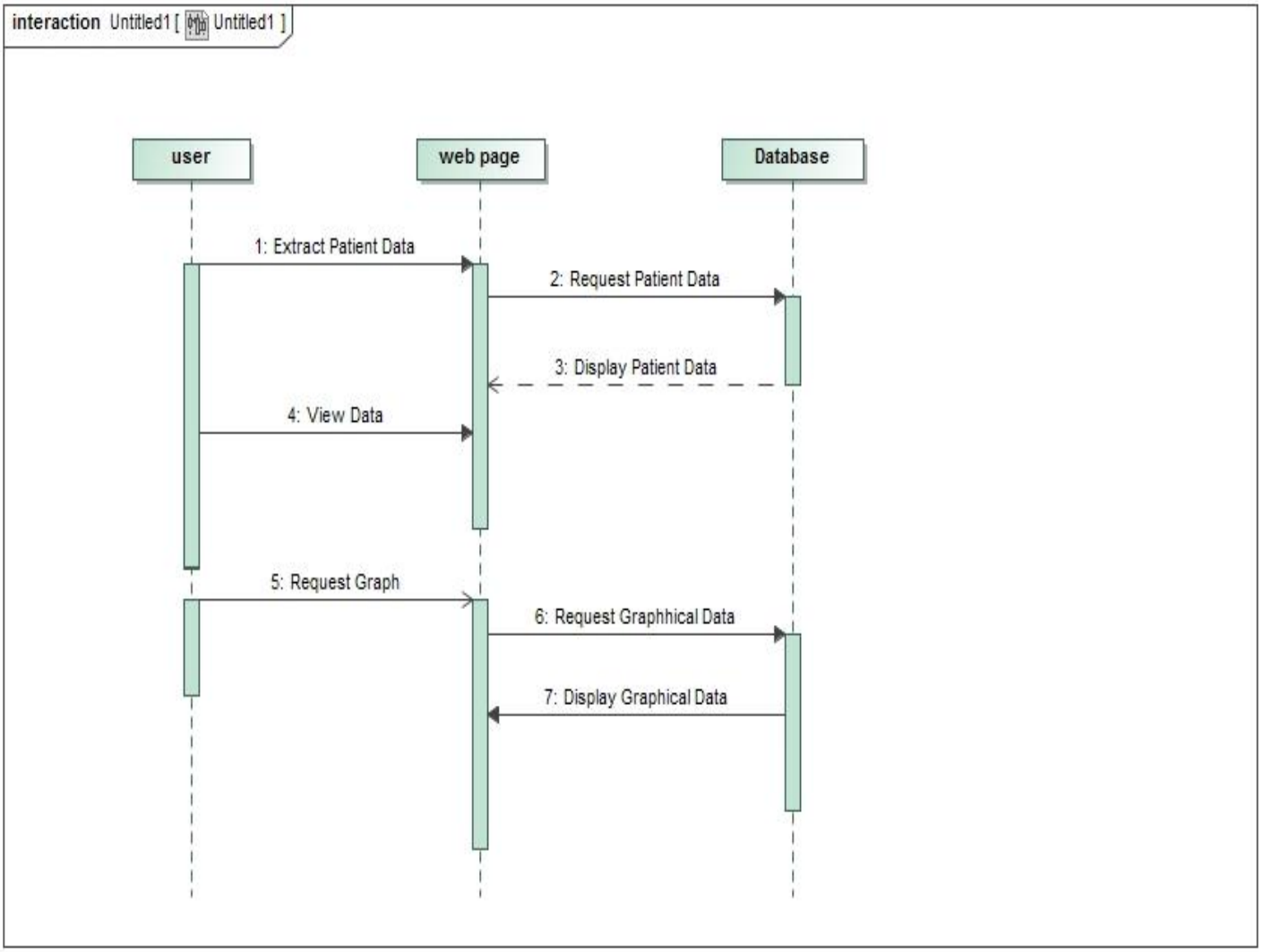


Figure 4.e.4 Display Data

Alternate Scenario:

The user tries to upload a file through the webpage after he has logged into his account. The File Format is checked before it can be uploaded. The Webpage sends back a invalid file format back to the user when he uploads a non .xls file.

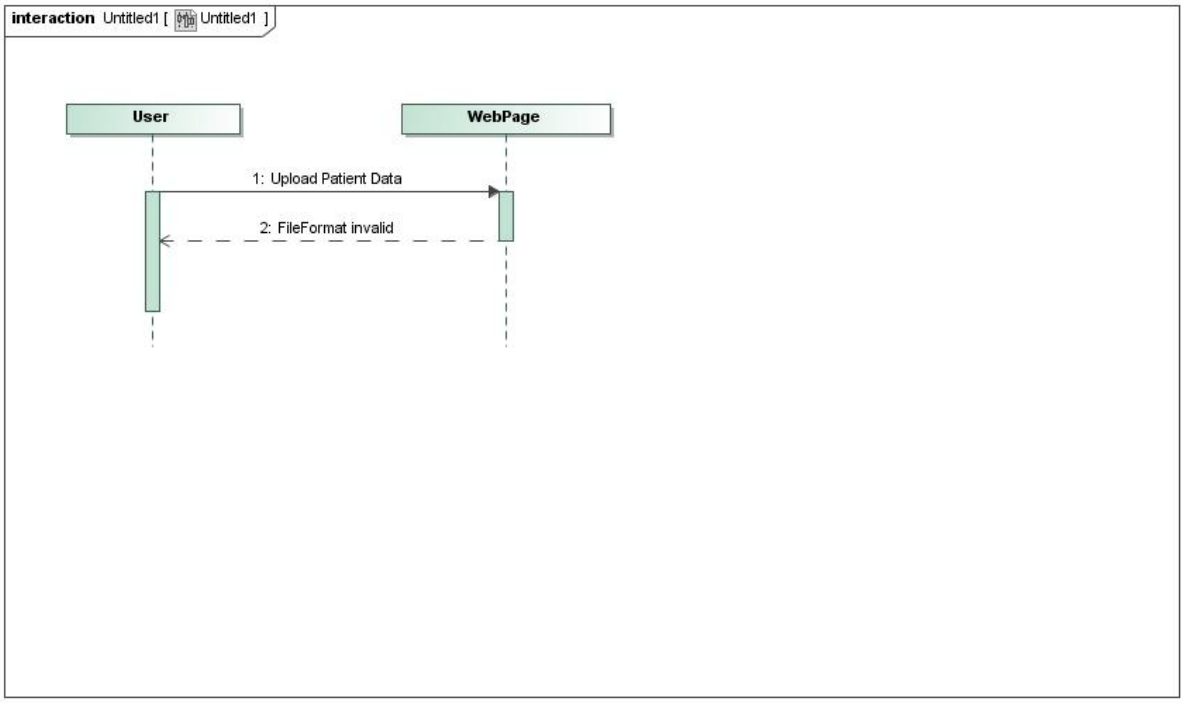


Figure 4.e.5 Display Data Alternate

Use Case 4: SendData

The user after viewing his health parameters would be able to send his report to the doctor for getting feedback on his progress. User will choose the button and send report and all his details would be sent to the doctor. A message acknowledging the data sending would be sent to the patient screen. In case of failure the reported error with failure message would be displayed.

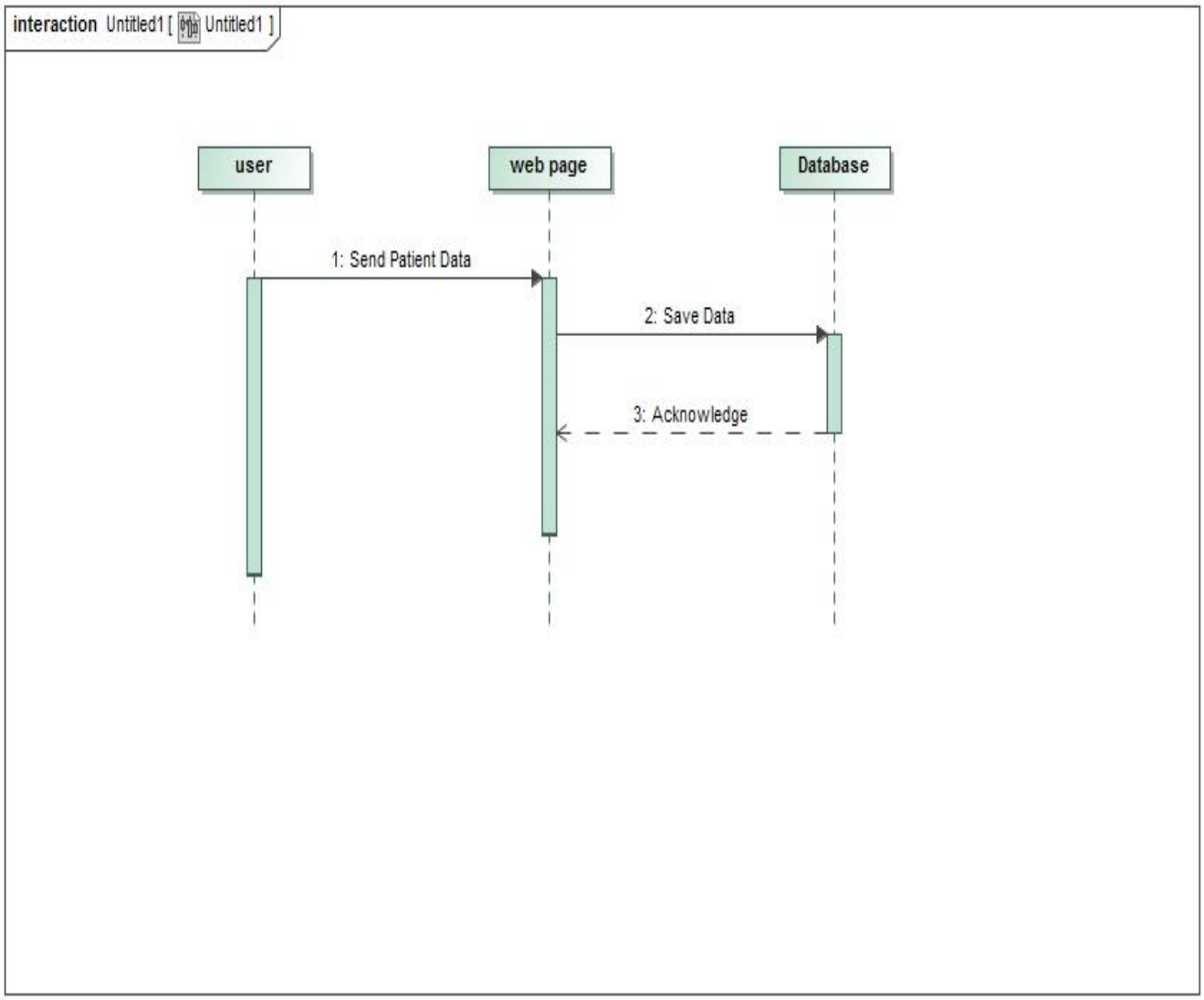


Figure 4.e.6 SendData

Use Case 5: ViewData

The Doctor would be able to view the data pertaining to a specific patient after logging in with his/her details. Doctor would get choose from within the a list of patients that would be displayed at his interface and then choose the patient. The data would be displayed as soon as the patient is selected. A message saying “No data present ” would be displayed in case no data is present for the chosen patient.

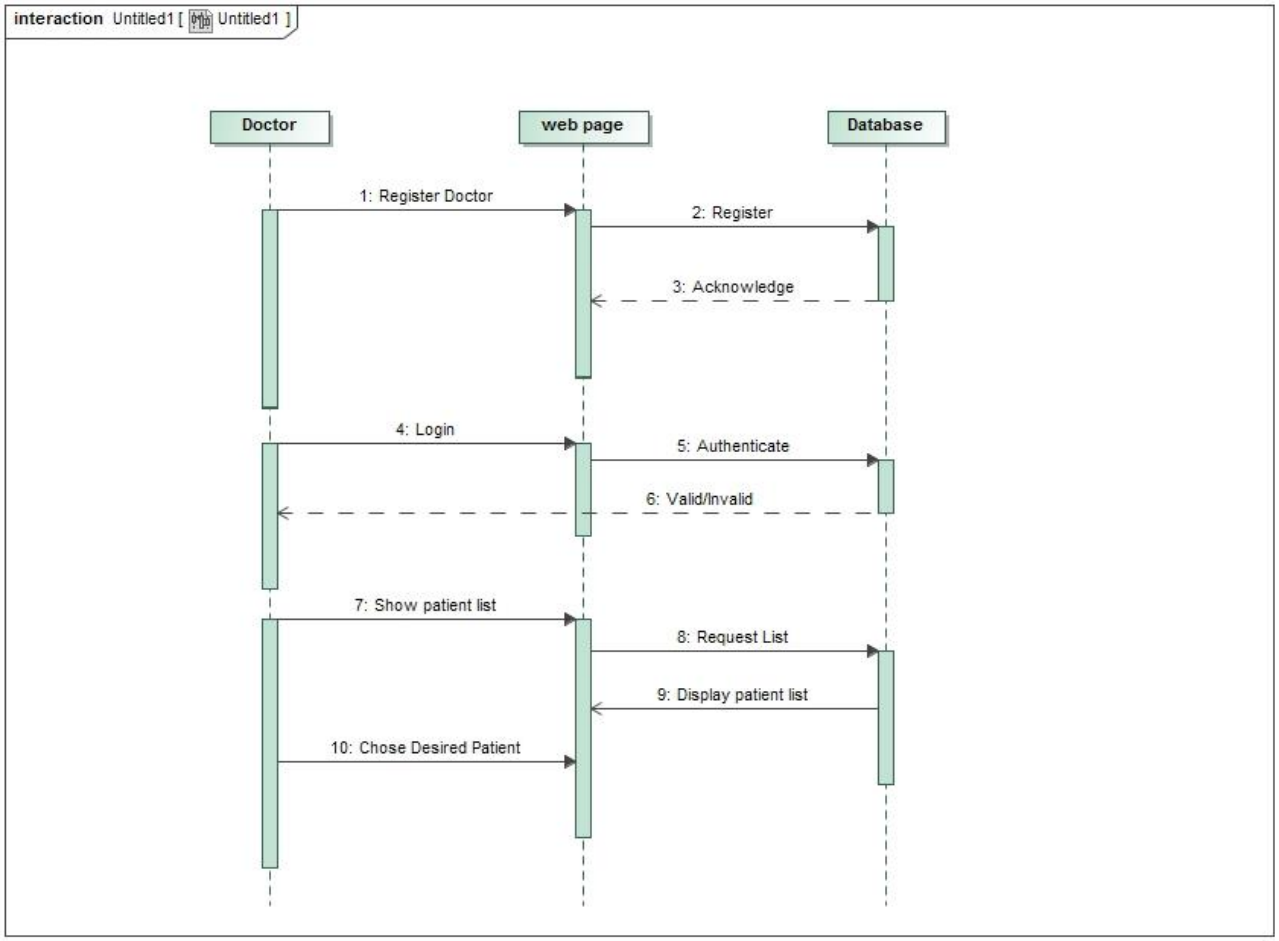


Figure 4.e.7 ViewData

Alternate Scenario:

After the patient uploads and sends the data to the doctor then the doctor should be able to view the file. An error may occur if the mapping between the patient and doctor is wrong which prevents the doctor from seeing this uploaded data.

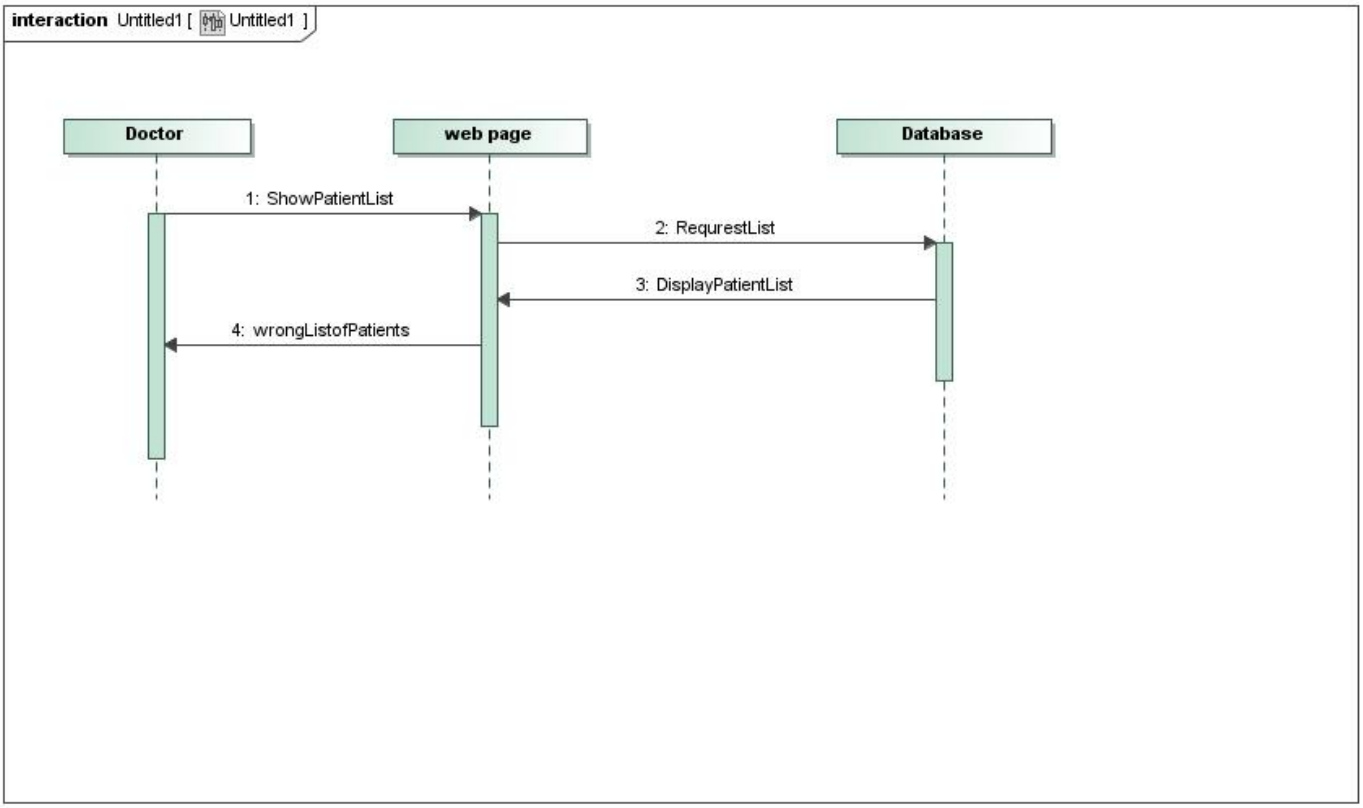


Figure 4.e.8 ViewDataAlternate

Use Case 6: ViewGraph

When the doctor is not satisfied with his data analyses using the tabular data ,Doctor would be able to same data in the graphical forms. Doctor would request the data from browser and required graphs would be displayed.

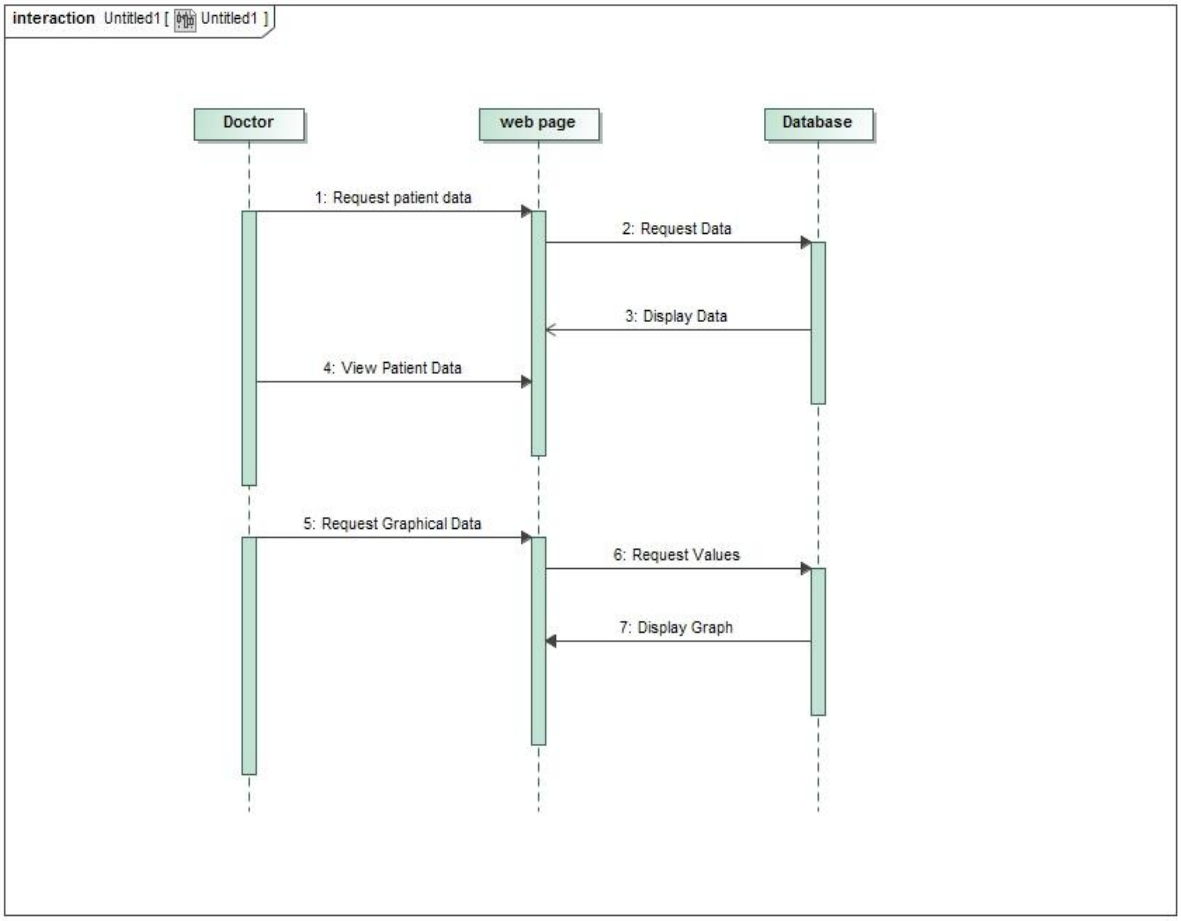


Figure 4.e.9 ViewGraph

Alternate Scenario:

When the patient tries to view the data which he has not uploaded yet into the system then the database sends back an error message saying unable to display the graphs for the tabular column.

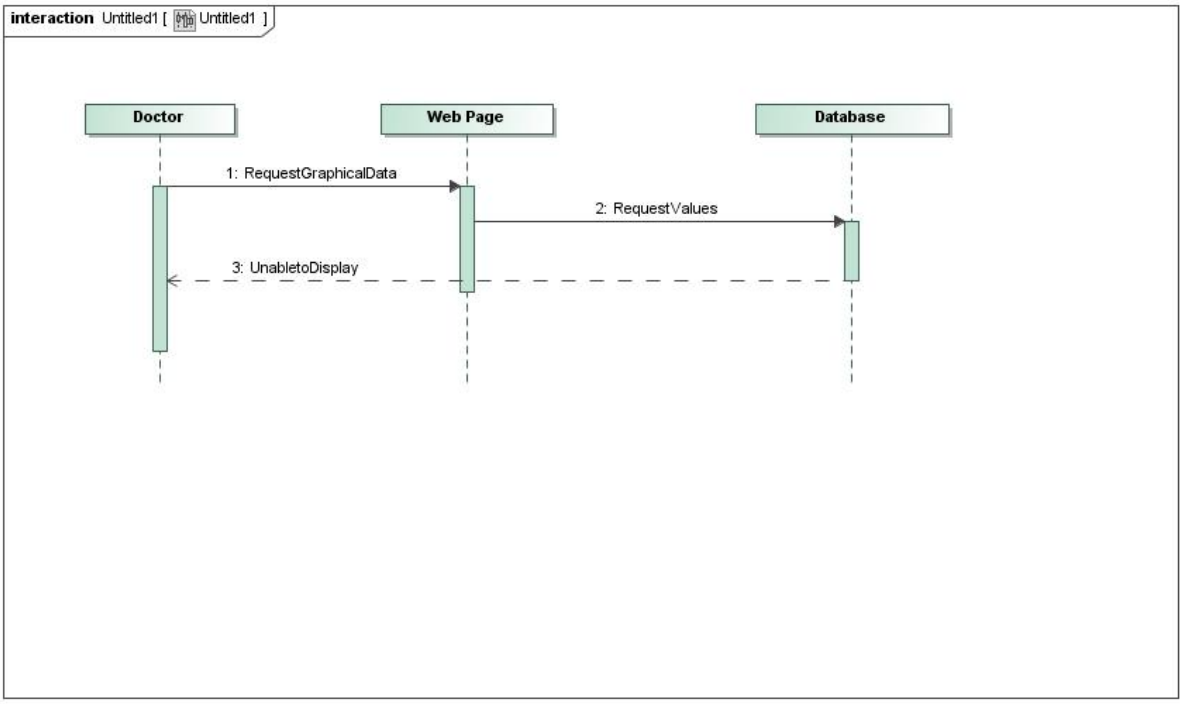


Figure 4.e.10 ViewGraph Alternate

Use Case 7: ProvideFeedback

Once after the data is analyzed by the doctor, the doctor will write his feedback comments in the field provided for feedbacks. After entering the comments ,database would be updated with the doctors comments and are ready for user view.

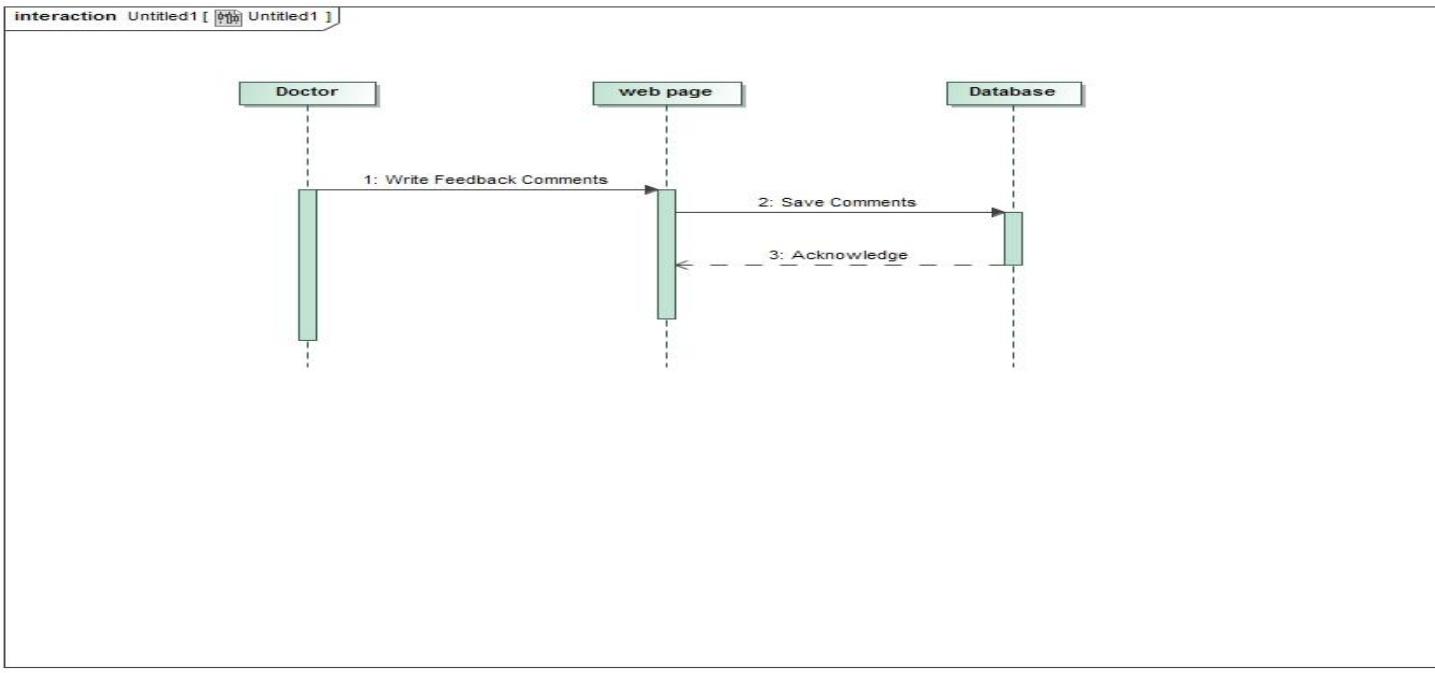


Figure 4.e.11 Provide Feedback

Use Case 8: Download Feedback

After logging in ,the patient would be able to choose his report for which he wants to see his/her feedback. If the comments are available patient would be able to see the feedback comments. However there is no separate communication provided that whether the doctors' comments are available or not. If the comments field is empty it is understood that the doctors' comments are not available.

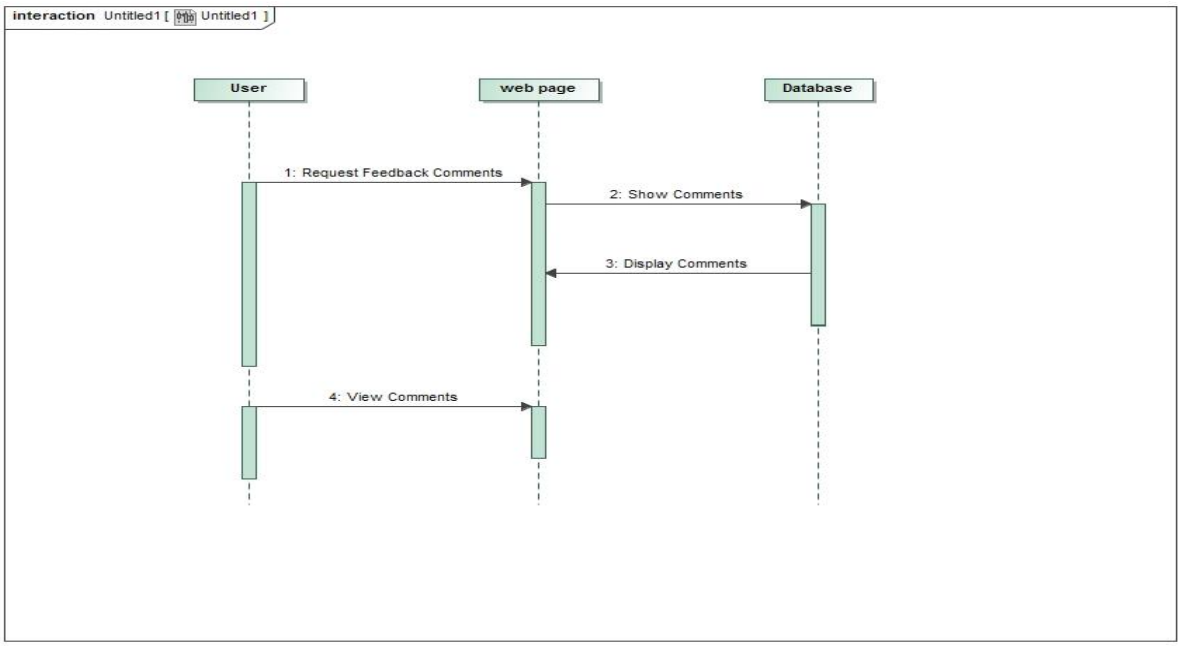


Figure 4.e.12 DownloadFeedback

5. Effort Estimation using Use Case Points

Use Case Points (UCP) method provides the ability to estimate the person-hours a software project requires based on its use cases. UCP method analyzes the use case actors, scenarios, nonfunctional requirements, and environmental factors and joins them in a simple equation: UCP = UUCP * TCF * ECF.

- Unadjusted Use Case Points (UUCP) – Measures complexity of functional requirements
- Technical Complexity Factor (TCF) – Measures complexity of nonfunctional requirements.
- Environmental Complexity Factor (ECF) – Assesses development teams experience and their development environment

5.1 Unadjusted Use Case Points

UUCP are computer as a sum of the following two components:

- Unadjusted Actor Weight (UAW) – Combined complexity of all the actors in all the use cases
- Unadjusted Use Case Weight (UUCW) – Total number of activities contained in all the use case scenarios

UUCP = UAW + UUCW

5.1.1 Unadjusted Actor Weight

The weights for Actor classification are computed via the following table: Actor classification and associated weights

Actor	Description of Actor Type	Weight
Simple	The actor is another system which interacts with our system through a defined application programming interface (API)	1
Average	The actor is a person interacting through a text-based user interface, or another system interacting through a protocol, such as a network communication protocol	2
Complex	The actor is a person interacting via a graphical user interface	3

Actor Classification for Health Monitoring System

Actor	Description of Characteristics	Complexity	Weight
User	User is interacting with the system through a graphical user interface.	Complex	3
Doctor	Doctor is interacting with the system through a graphical user interface.	Complex	3
Database	Database is another system interacting through a protocol.	Average	2
Web Server	Web server is interacting through http.	Average	2

Table 5.1.1 Unadjusted Actor Weight

UAW (Health monitoring system) = 2 * Average + 3 * Complex
= 2*2 + 3*2
= 10

5.1.2 Unadjusted Use Case Weight

The weights for use cases are computer via the following table:

Use Case Category	Description of Category	Weight
Simple	Simple user interface. Up to one participating actor (plus initiating actor). Number of steps for the success scenario: no more than 3. If presently available, its domain model includes no more than 3 concepts.	5
Average	Moderate interface design. Two or more participating actors. Number of steps for the success scenario: 4 to 7. If presently available, its domain model includes between 5 and 10 con- cepts.	10
Complex	Complex user interface or processing. Three or more partic- ipating actors. Number of steps for the success scenario: at least 7. If available, its domain model includes at least 10 concepts.	15

Table 5.1.2 Unadjusted Use Case Weight

Use case classification for Health Monitoring System

Use Case	Description	Category	Weight
Register/Login (UC-1)	Simple user interface. 8 steps for main success scenario. Three participating actors (User, Database, web server).	Complex	15
Upload Data (UC-2)	Simple user interface. 3 steps for main success scenario. Three participating actors (User, Database, web server).	Simple	5
Display Data (UC-3)	Simple user interface. 4 steps for main success scenario. Three participating actors (User, Database, web server).	Average	10
Send Data (UC-4)	Simple user interface. 2 steps for main success scenario. Two participating actors (User, Database, web server).	Simple	10
View Data (UC-5)	Simple user interface. 3 steps for main success scenario. Three participating actors (Doctor, Database, web server).	Simple	5
View Graph (UC-6)	Simple user interface. 4 steps for main success scenario. Three participating actors (Doctor, Database, web server).	Average	10
Provide Feedback (UC-7)	Simple user interface. 3 steps for main success scenario. Three participating actors (Doctor, Database, web server).	Simple	5
Download Feedback (UC-8)	Simple user interface. 2 steps for main success scenario. Three participating actors (User, Database, web server).	Simple	5

5.1.3 **Computing Unadjusted Use Case Points**

UUCW(Health monitoring system) = 5 * Simple + 10*Average + 15 * Complex
= 5*4 + 10*2 + 15*1
= 55

UUCP(Health Monitoring system) = UAW + UUCW = 10 + 55 = 65

5.2 Technical Complexity Factor

Technical Complexity Factor (TCF) is computed using thirteen standard techni- cal factors to estimate the impact of productivity of the nonfunctional requirements for the application. We then need to assess the perceived complexity of each tech- nical factor in the context of the project. A perceived complexity value is between
0 and 5, with 0 meaning trivial effort, 3 meaning average effort and 5 meaning major effort. Each factors weight is then multiplied by perceived complexity factor to produce calculated factor. Two constants are used with TCF. The constants utilized are C1 = 0.6 and C2 = 0.01.

$$TCF = \text{Constant1} + \text{Constant2} \times \text{Technical Factor Total} = C_1 + C_2 \cdot \sum_{i=1}^{13} W_i \cdot F_i$$

Technical complexity factors and their weights:

Technical Factor	Description	Weight
T1	Distributed system	2
T2	Performance objectives	1
T3	End-user efficiency	1
T4	Complex internal processing	1
T5	Reusable design or code	1
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portable	2
T9	Easy to change	1
T10	Concurrent use	1
T11	Special security features	1
T12	Provides direct access for third parties	1
T13	Special user training facilities are required	1

Technical complexity factors for Health Monitoring System:

PC = Perceived Complexity, CF = Calculated Factor

Technical Factor	Description	Weight	PC	CF
T1	Distributed, web-based system	2	3	6
T2	User expects good performance, but will tolerate network latency	1	3	3
T3	End-user expects efficiency, which is achieved through caching	1	4	4
T4	Internal processing required multiple interactions with other subsystems	1	4	4
T5	No requirement for reusability	1	0	0
T6	No user installation required	0.5	2	1
T7	Ease of use was very important	0.5	5	2.5
T8	Portable since it runs in a browser	2	2	4
T9	Relatively simple to add new features	1	2	2
T10	Concurrent use is required	1	4	4
T11	Security handled by Facebook	1	0	0
T12	No direct access for third parties	1	0	0
T13	No training required	1	0	0

Table 5.2 Technical Complexity Factors

TCF = 0.6 + (0.01 × 32.5) = 0.925.

This results in a decrease of the UCP by 7.5 %.

5.3 Environment Complexity Factor

The Environment Complexity Factor (ECF) is computed utilizing eight standard environmental factors to measure the experience level of the people on the project and the stability of the project. We then need to assess the perceived impact based on perception that factor has on projects success. 1 means strong negative impact,
3 is average and 5 means strong positive impact.

TCF is computed utilizing thirteen standard technical factors to estimate the impact of productivity of the nonfunctional requirements for the application. We then need to assess the perceived complexity of each technical factor in the context of the project. A perceived complexity value is between 0 and 5 with 0 meaning that it is irrelevant, 3 means average effort and 5 means major effort. Each factors weight is then multiplied by perceived complexity factor to produce calculated factor. Two constants are used with ECF. The constants utilized are C1 = 1.4 and C2 = -0.03.

$$ECF = Constant1 + Constant2 \times EnvironmentalFactorTotal = C_1 + C_2 \cdot \sum_{i=1}^{8} W_i \cdot F_i$$

Environmental complexity factors and their weights:

Environmental Factor	Description	Weight
E1	Familiar with the development process	1.5
E2	Application problem experience	0.5
E3	Paradigm experience	1
E4	Lead analyst capability	0.5
E5	Motivation	1
E6	Stable requirements	1
E7	Part-time staff	-1
E8	Difficult programming language	-1

Environmental Complexity Factors for Health Monitoring System:

PI = Perceived Impact, CF = Calculated Factor

Environment Factor	Description	Weight	PI	CF
E1	Beginner familiarity with UML-based development	1.5	1	1.5
E2	Half of team has familiarity	0.5	3	1.7
E3	Some knowledge of object-oriented approach	1	3	3
E4	Average lead analyst	0.5	2	1
E5	Highly motivated overall	1	4	4
E6	Requirements were stable	2	5	10
E7	Student staff (part-time)	-1	4	-4
E8	Used new programming languages but resources were readily available	-1	5	-5

Table 5.3 Environmental Complexity

ECF = 1.4 - (0.03 × 12) = 1.04

This results in an increase of UDP by 4%.

5.4 Calculating the Use Case Points

As mentioned earlier, UCP = UUCP × TCF × ECF.

From above calculations, UCP variables have the following values: UUCP = 65

TCF = 0.925

ECF = 1.04

UCP = 165 × 0.925 × 1.04 = 62.53 or 63 use case points.

5.5 Deriving Project Duration from Use-Case Points

UCP is a measure of software size. We need to know the teams rate of progress through the use cases. We need to utilize the UCP and Productivity Factor (PF) to determine duration. The equation for computing Duration is:

$$\text{Duration} = \text{UCP} \times \text{PF}$$

Productivity Factor is the ratio of development person-hours needed per use case point. Assuming a PF = 28, the duration of our system is computed as follows:

$$\text{Duration} = \text{UCP} \times \text{PF} = 63 * 28 = 1764$$

1764 person-hours for the development of the system.

This does not imply that the project will be completed in 1764/24 = 74 days. A reasonable assumption is that each developer on average spent 10 hours per week on project tasks. With a team of six developers, this means the team makes 6 *15 = 90 hours per week. Dividing 1764 person-hours by 90 hours per week, we obtain the total of approximately 20 weeks to complete this project. We have spent 16 weeks approximately on the project so far which gives us 4 weeks left to complete this project according to our estimation. The reason for the large estimate is due to the highly over-estimated productivity factor.

6. Domain Analysis

a) Domain Model:-

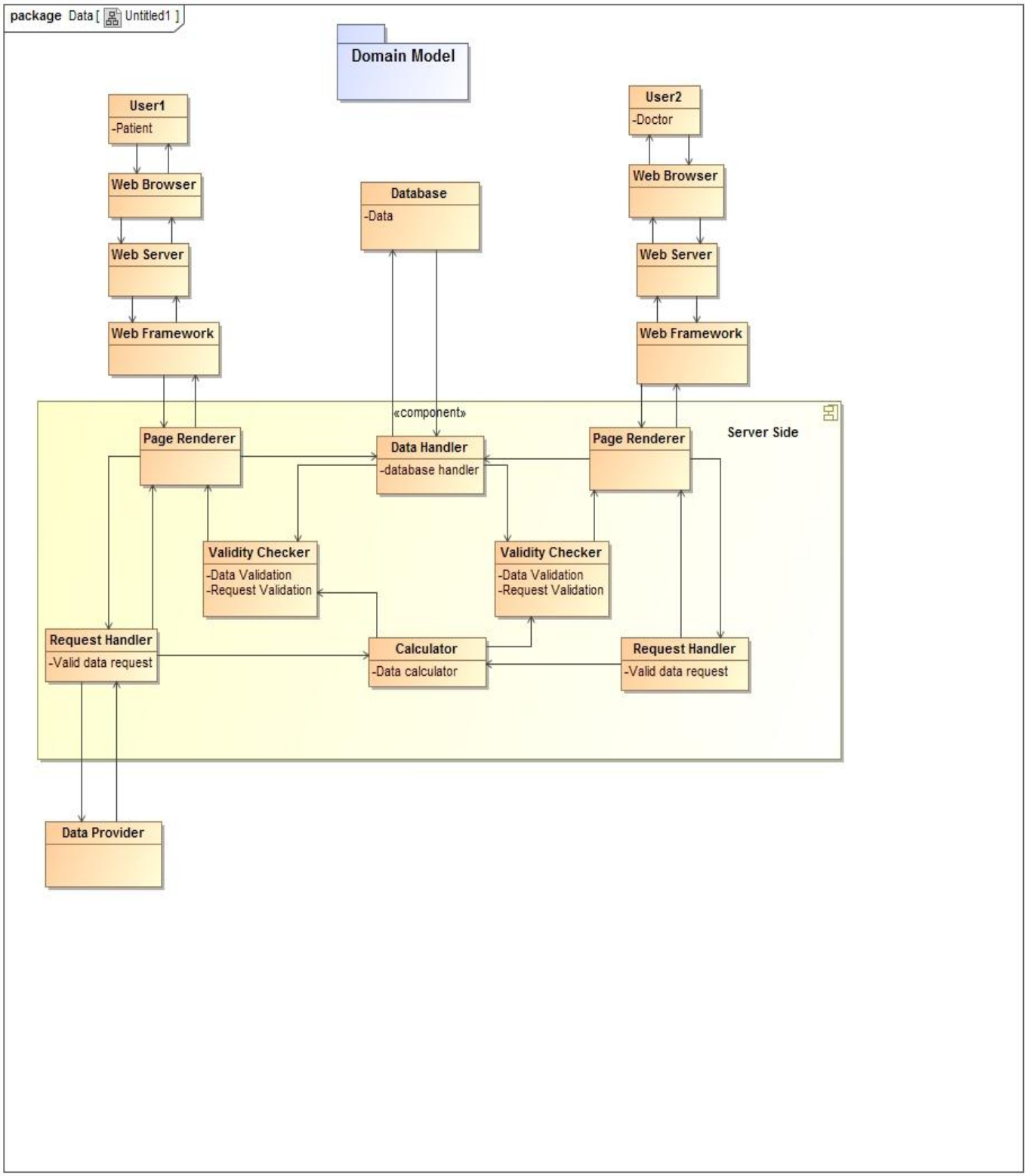


Figure 6.a.1 Domain Model

Figure general Domain Model as a whole. The subsequent diagrams will give insight into how the concepts work to satisfy the key user cases of the website. This diagram shows the general flow of information and request ordered and how it is processed. The database would get updated periodically by requesting new information every time and whenever data is stored or calculated.

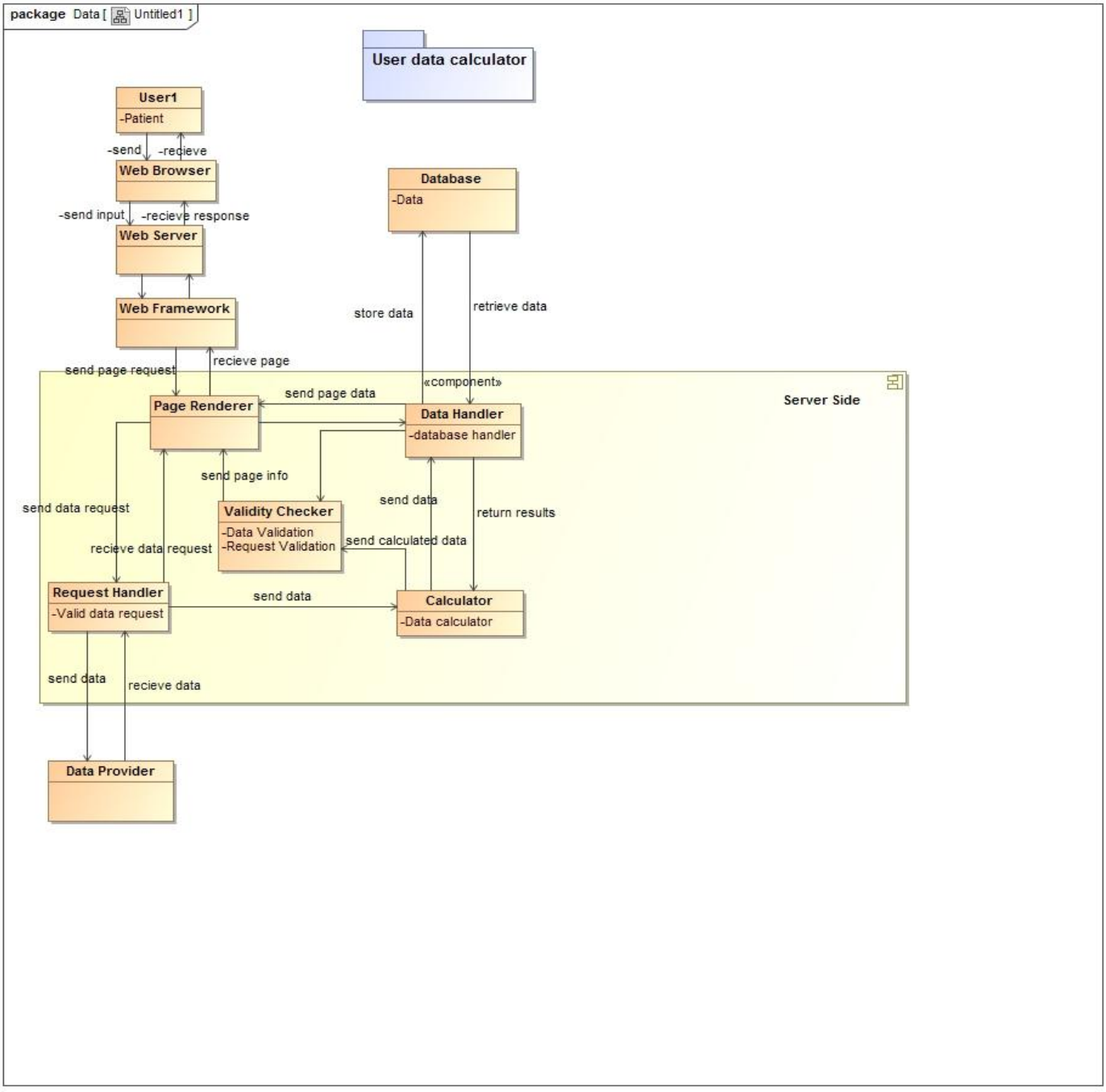


Figure 6.a.2

Figure 6.a.2 will represent UC1, UC2, UC3. It represents login, register upload data information, calculate and display information. The user will request for data to be uploaded and the request goes to request handler from web Framework via page renderer. The request handler will communicate with the system from which the data will be fetched and then it is uploaded to the database after processing by the calculator. Also the user requests for data to be calculated and displayed. The request will be send from web framework to the page renderer and then to the request handler which checks which type of request to be process. For calculating result request then goes to data handler which fetches the stored data from the database and then this data goes to the calculator. The calculator process the data and determine various parameters like total energy expenditure, sleep time, steps etc and then validity checker checks if the results calculated are right.

. Then the calculated results are send to page rendered which generates a user friendly page to display data. The web framework then process the page and displays it to the user.

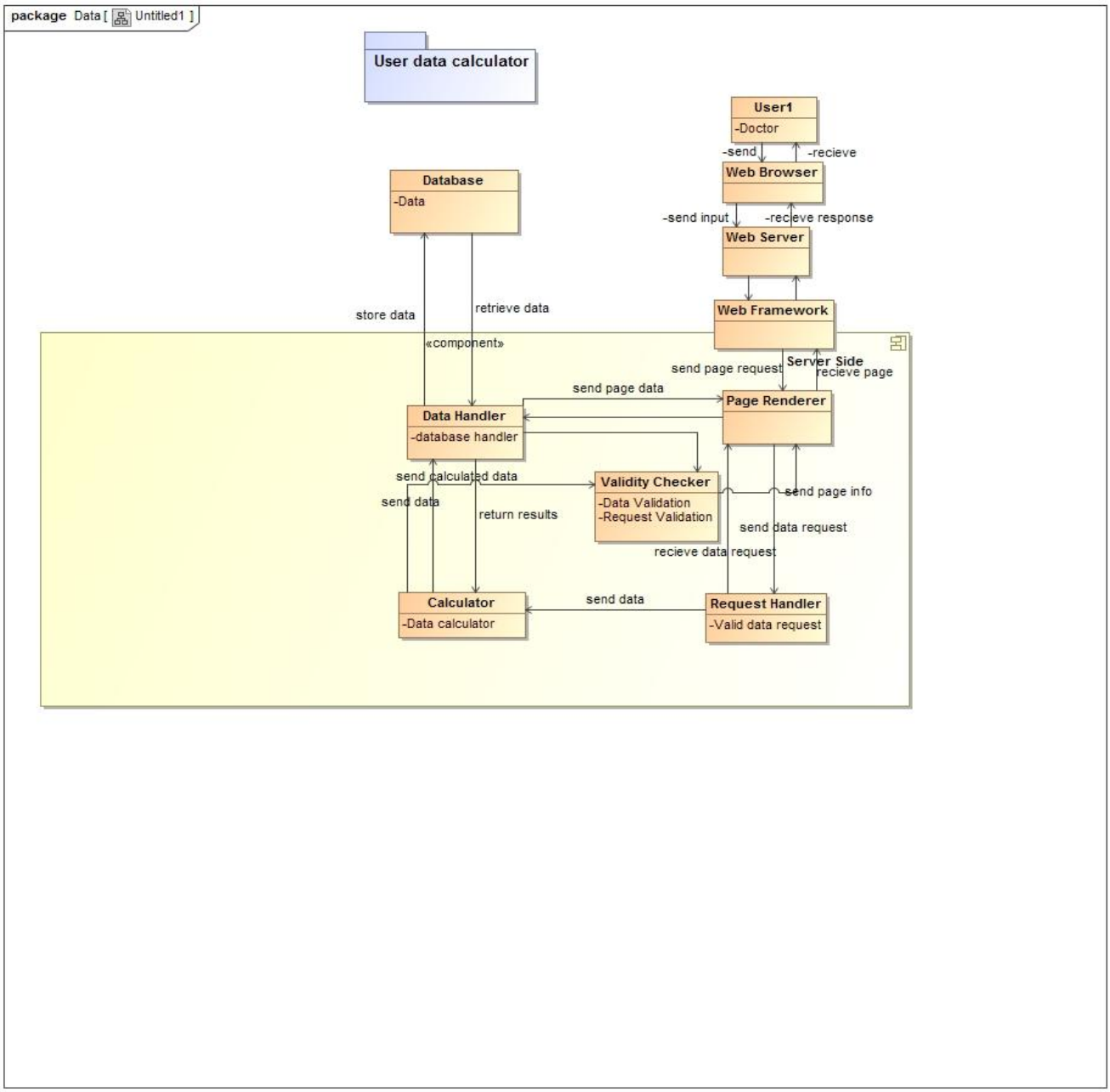


Figure 6.a.3

Figure 6.a.3 will represents UC1, UC5, UC6, UC7. It represents login, register of doctor, view data by doctor and feedback. The user(doctor) will send the request and the request goes to request handler from web Framework via page renderer. The request handler checks which type of request to be process. For calculating result request then goes to data handler which fetches the stored data from the database and then this data goes to the calculator. The calculator process the data and determine various parameters like total energy expenditure, sleep time, steps etc and then validity checker checks if the results calculated are right. Then the calculated results are send to page rendered which generates a user friendly page to display data. The web framework then process the page and displayed to the user.

To give feedback the user request for feedback window and then writes the feedback which is then goes to data handler. It then saves the feedback for that particular patient in the database which can be viewed by him.

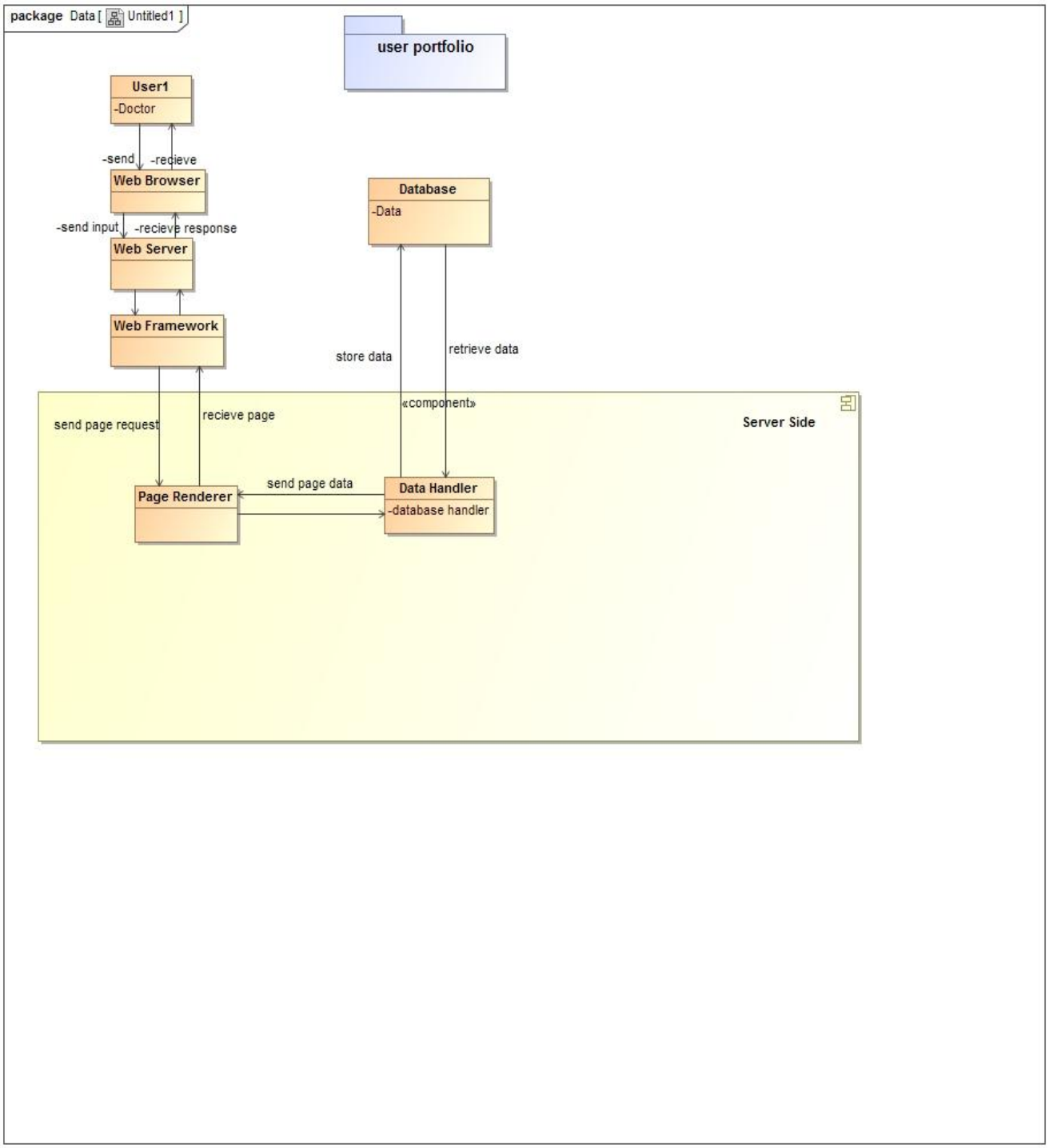


Figure represents UC1. The user queries about the profile view and then the request is sent from the web framework to the page renderer which then goes to data handler. The data handler fetches the reused data from the database and it is sent back to the page renderer. The page renderer generates the page for the web framework to send to the user for viewing.

i. Concept Definition

The Definitions of our concepts are as follows:

User
Definitions: Monitoring user health through feedback from doctor.
Responsibilities:
_ Manage portfolio
_ Make requests for feedback
_ send data
_ Navigate through website

Web Browser
Definitions: The user's browser which runs from the user's device.
Responsibilities:
_ Take requests from the user
_ Send requests to the Web Server
_ Get responses from the Web Server
_ Display the response from the Web Server

Web Server
Definitions: HTTP web server, running on a web host
Responsibilities:
_ Receive requests from Web Browser
_ Send requests to Web Framework
_ Get responses from Web Framework
_ Send responses to the Web Browser

Web Framework
Definitions: APIs to help display user-friendly output
Responsibilities:
_ Receive requests from Web Server
_ Sends request to appropriate handler: application or database
_ Receive rendered pages in the form of structured data
_ Send responses to the Web Server

Page Renderer
Definitions: Takes user requests and creates a page which is user-friendly
Responsibilities:
_ Determine the information required to be rendered and request it

_ Receive the required information
_ Convert the information into user-friendly format
_ Send rendered pages to the Web Framework

Request Handler
Definitions: Application which takes the input from the patient and processes it to display the graph.
Responsibilities:
_ Receive requests from Web Framework
_ Determine what the request is and readies for manipulation
_ Request updated calculated data info
_ Transmit necessary information to other concepts

Calculator
Definitions: Calculate the total or average of the real time Data
Responsibilities:
_ Receive requests for calculation
_ Request information from request handler
_ Retrieve information from data handler
_ Send real-time calculated data to be stored for application's use

Validity Checker
Definitions: Checks if a trade is valid
Responsibilities:
_ Receive updated order information
_ Request and receive portfolio data
_ Determine if correct data is provided by the user
_ Determine if calculated data is correct for the desired user
_ Send updated portfolio information if necessary

Data Handler
Definitions: Communicates with Database to service data requests
Responsibilities:
_ Receive and send every kind of data used in system
_ Request data from Database
_ Send data to be stored in Database

ii. Association Definitions

The following association definitions are provided for the domain models that model not only for the important use cases, but also any alternative models for said use cases

Table 6.ii.1 Associative Definitions

Concept Pair	Association Description	Association Name
Web Browser Web Server	User interacts with browser	send input, send response
Web Framework $ Request Handler	Passes data, User ID and league ID	send data request
Web Framework $ Page Renderer	Request to visit page, sends rendered page in form of data	send page request, send page
Page Renderer $ Data Handler	Requests data to correctly render page, passes necessary data	send data request, send page data
Page Renderer $ calculator	Asks for calculated data, send calculated data	request data data, send data
Page Renderer $ Validity Checker	Passes necessary data for the page to be rendered	send page info
request Handler $ Stock Query	Passes necessary data	send data
Validity Checker $ Data Handler	Asks for user's portfolio information for validity purposes, passes user's portfolio information, passes updated portfolio information following	request portfolio data, return portfolio data, send new portfolio data
Validity Checker $ calculator	Sends updated data to be checked	send updated data
calculator $ Data Handler	Sends calculated information, returns new value	send info, return request
Data Handler $ Database	Stores incoming data, request certain data, retrieve needed data	store data, request data, retrieve data

iii. Attribute Definitions

Most of our concepts do not need to hold their own data, as our website is dynamic and web-based. We also have not yet made the decision to cache data.
Thus, nearly all data is stored in a single database. The sparse attributes that must be accounted for are as follows:
Concept

Concept	Attribute	Meaning
Data Handler	databaseHandle	Interacts with the database.
Database	data	Stores data for future use. Includes all data used in the system, including League ID, User ID, stock volume and price data, league settings, fund settings, and portfolio data such as transaction history.
calculator	Calculate data	Calculates the average value or the total value
Page Renderer	su_cientRenderData	Determines if the required data to render the page is there.
Request Hander	validOrderRequest	Checks to see if there is all the required data.
Validity Checker	Data valid,user valid	Compares users with database checks settings to make sure a user and data are valid. Determines if identity is a success.

Table 6.iii.1 Attribute Definition

iv. Traceability Matrix

UC	PW	User Patient	User Doctor	Web Browser	Web Server	Web Framework	Page Renderer	Request Handler	Calculator	Data Handler
UC01-03	35	X		X	X	X	X	X	X	X
UC05-07	17		X	X		X	X	X	X	X
UC08	35	X		X	X	X	X			X

Table 6.iv. 1 Traceability Matrix

b) System Operation Contracts

i. Register User:

Preconditions: None

Postconditions:

- User has a profile linked to database and every user has a doctor who will review his profile.
- Database keeps track of user's information.

ii. Upload Data:

Preconditions:

- Make sure enough space is available on in database.
- User is logged in.

Postconditions:

- Data is updated to database.

iii. Display data:

Preconditions:

- Database has valid information.
- Parameters have been calculated.

Postconditions:

- Data is displayed in user friendly version.
- Graphical representation is used for easy understanding.

iv. Send to Doctor:

Preconditions:

- Doctor is logged in.
- Parameters have been calculated.
- User has requested a review from his Doctor.

Postconditions:

- Data is sent to the doctor specified by the user during registration.

v. Feedback:

Preconditions:

- Doctor has received data pertaining to user who has requested areview.
- Doctor has viewed all graphical charts of user.

Postconditions:

- Doctor writes comments for user based on his report.
- Feedback is uploaded to the database.

vi. Graphical Representation:

Precondition:

- User/Doctor is logged in.
- Valid information is present in the database.

Postcondition:

- Graphical version of the user information is generated.

vii. Feedback to User:

Precondition:

- User is logged in.
- Doctor has updated feedback to the database for user.

Postcondition:

- User gets the feedback meant for him/her from the database.

viii. Downloadable Feedback:

Precondition:

- Database contains the information requested by the user.

Postcondition:

- Information is downloaded and saved on user's system.

ix. View and Edit Information:

Precondition:

- User/Doctor is logged in.

Postcondition:

- Profile Information is modified and updated in database.

7. Interaction Diagrams:-

7.1 Use Case -1(Register/Login)

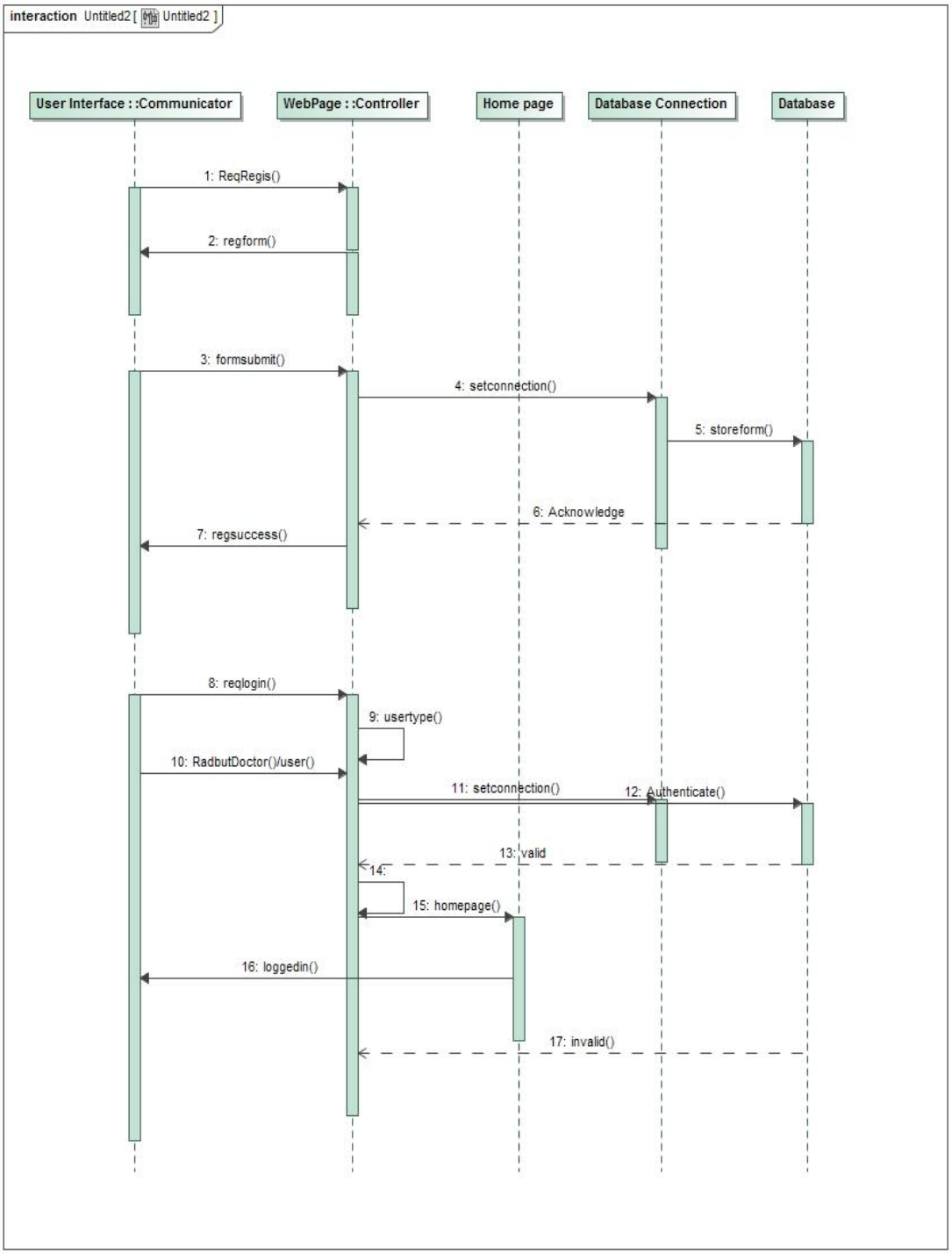


Figure 2.1 Use Case 1.

The above interaction diagram(Figure2.1) 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 radio button. 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 displays 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. All Interaction Diagrams are as per UML specifications[4][5].

7.2Use case 2 (Upload Data)

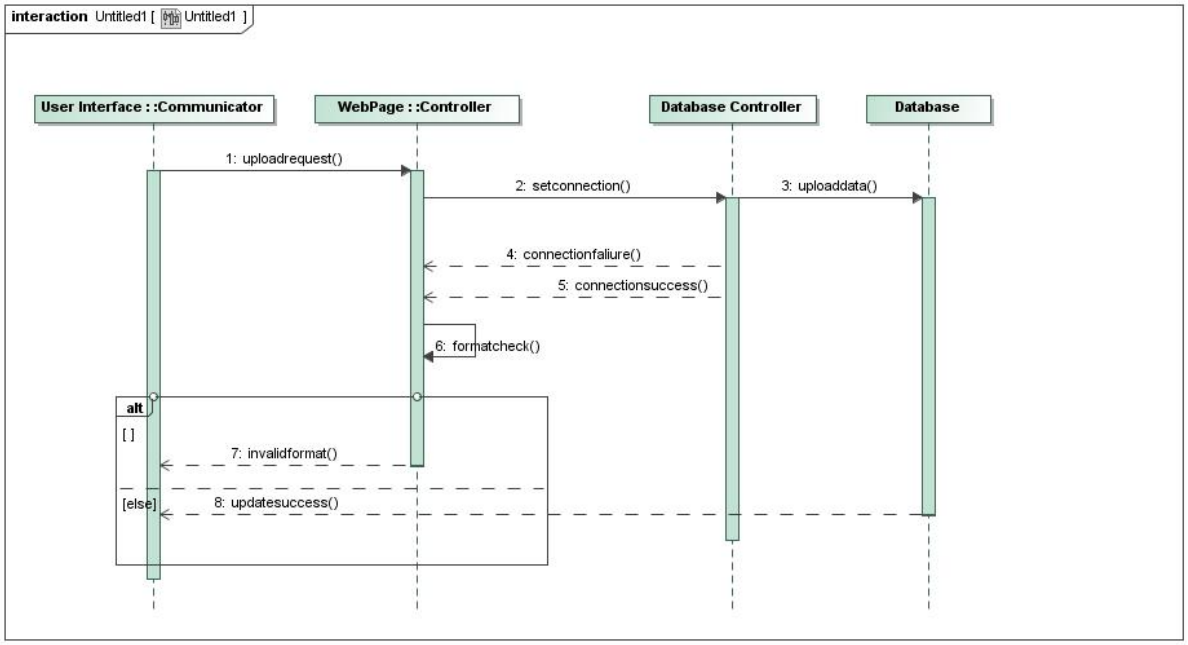


Figure 2.2 Use Case2.

The above interaction diagram(Figure 2.2) 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.

2.2 Use case 3 & 4(Display Data/Send Data)

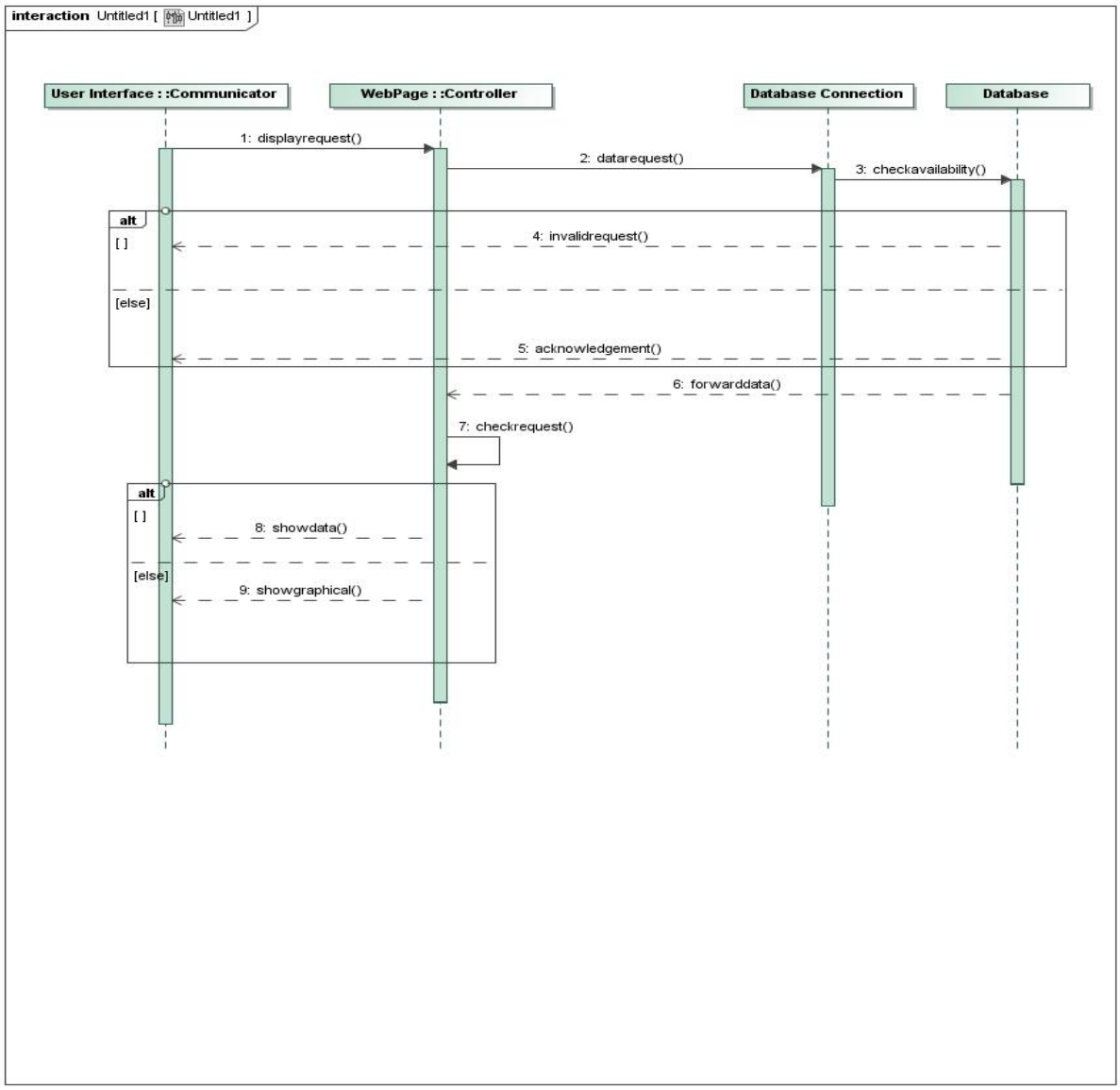


Figure 2.3 Use Case 3 and 4.

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.

7.3 Use case 5 (View Data)

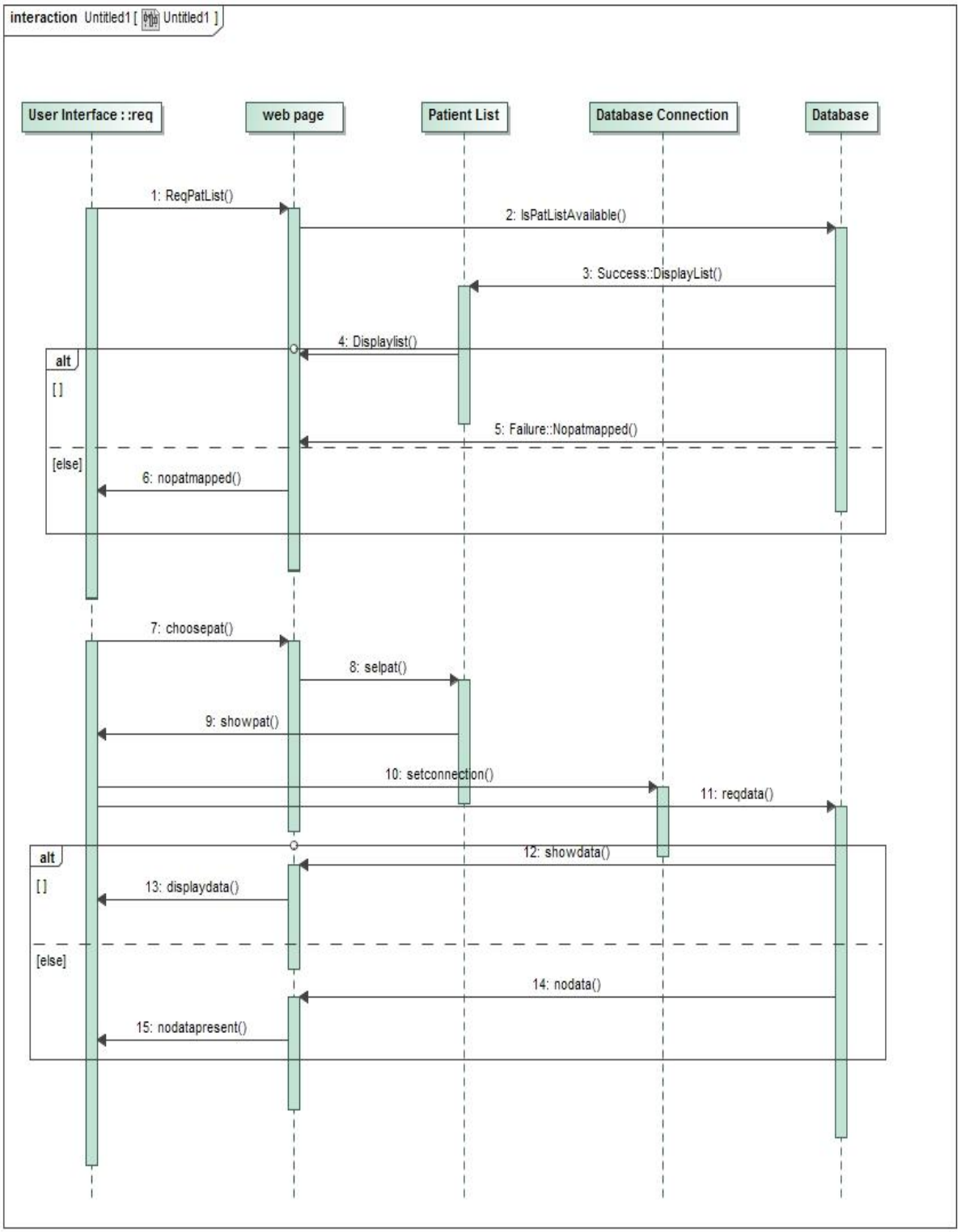


Figure 2.4 Use Case 5.

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.

7.4 Use Case 6&7(View Graph / Provide FeedBack)

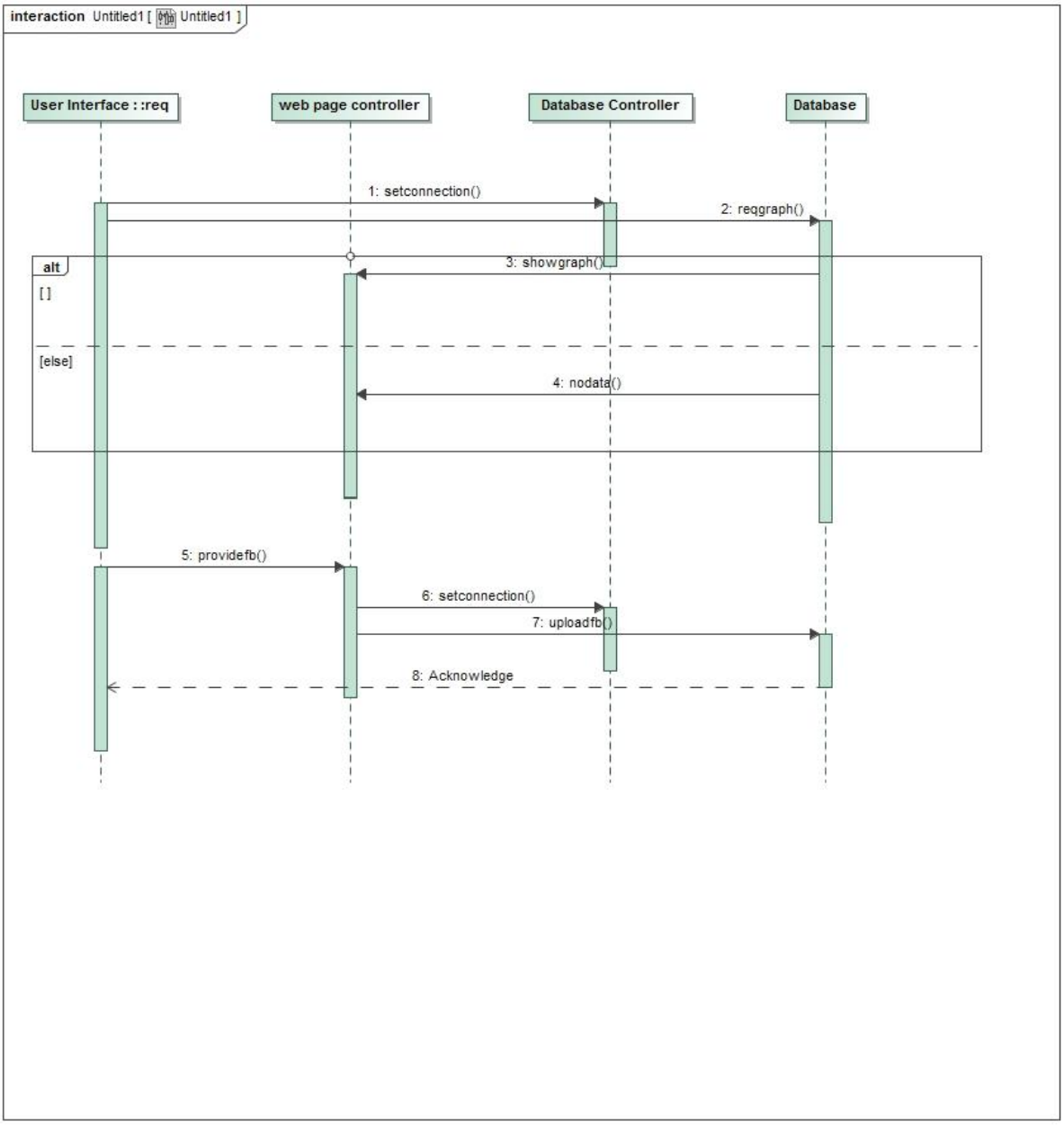


Figure 2.5 Use Case 6 and 7

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.

7.5 Use Case 8(Download Data)

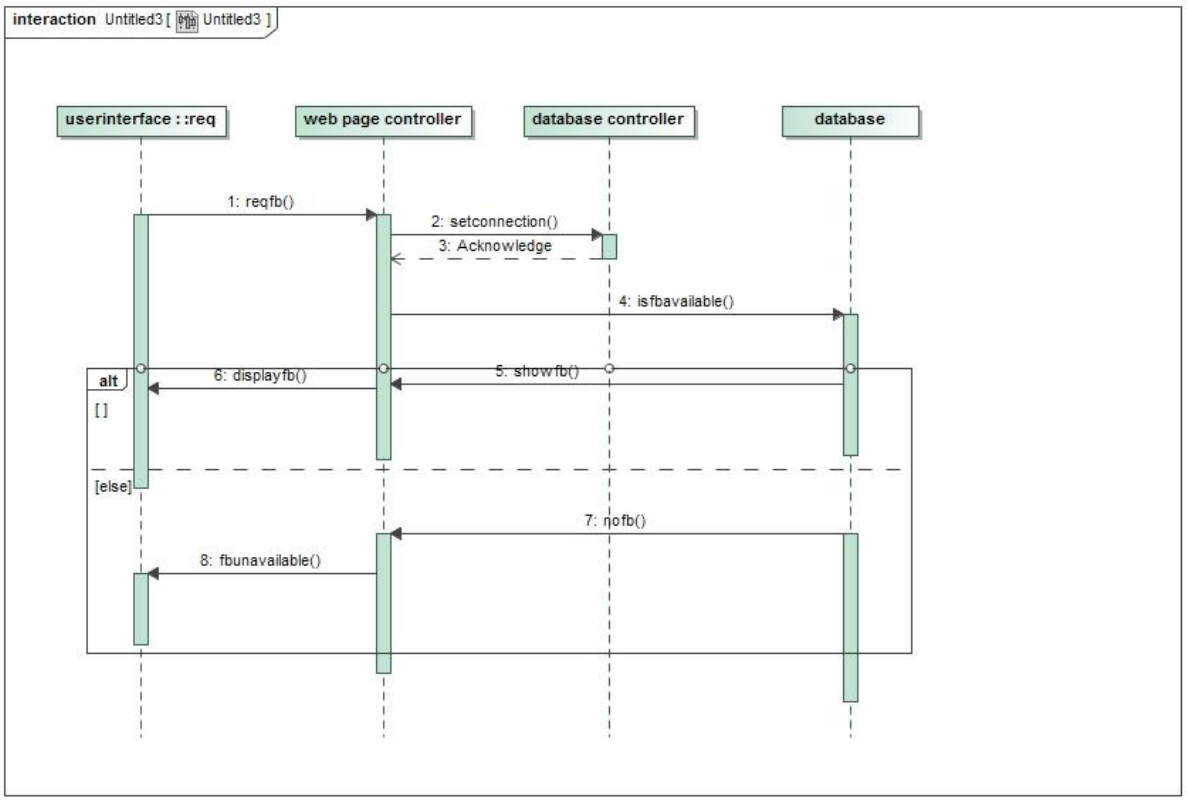


Figure 2.6 Use Case 8.

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.

Design Pattern

In the diagrams above we employed the MVC (Model View Controller) pattern[8], which allows for a modular and scalable approach to software development. The view is the interface the user interacts with (for example, by entering the user action like viewing data, graphs). The controller handles the input event from the user interface, often via a registered call back and converts the event into an appropriate user action, understandable for the model. The controller then notifies the model of the user action, possibly resulting in a change in the model's state. A view queries the model in order to generate an appropriate user interface. The user interface waits for further user interactions, which restarts the control flow cycle. This pattern makes agile development easier and allows for a more secure deployment.

8. Class Diagram and Interface Specification

a. Class Diagram:-

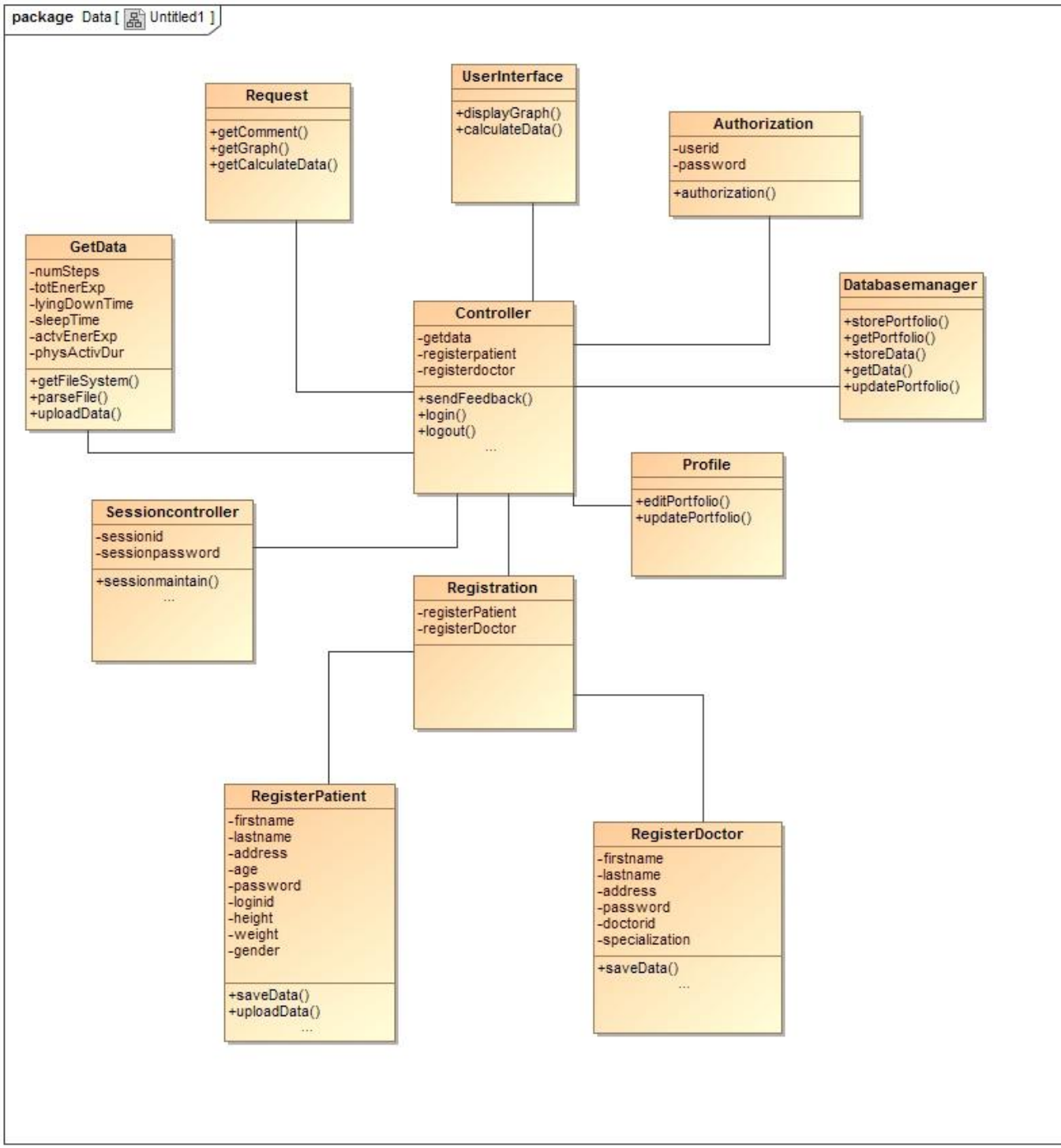


Figure 8(a)Class Diagram [5]

b. Data Types and Operation Signature

1. Controller:-

- **Operations:-**

  - +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
  - stringlyingDownTime
  - stringsleepTime
  - stringactvEnerExp
  - stringphysActivDur

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

8. **UserInterface:-**

   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.

9. **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.

10. 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

The classes explained above were derived from the previous traceability matrix[1]. 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[3][9] 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.

Domain Concepts	GetData	Controller	Request	RegisterDoctor	RegisterPatient	Databasemanager	SessionController	Profile	Authorization
User	X			X	X	X			
Web Browser	X			X	X	X	X	X	
Web Server	X			X	X	X	X	X	
Page Renderer		X				X			
Request Handler		X	X	X	X	X		X	
Calculator		X				X			
Validity Checker		X	X			X			X
Data Handler	X					X	X	X	X

Figure 8(c)Traceability Matrix.

During our first understanding of system to be, we developed the concept of WebFramework, however while working on the system we realized that most of the task involved could be managed by making some changes to concepts like WebServer, Controller and request handler hence that concept is not shown in the traceability matrix. From the traceability matrix we prioritized the classes and organized the functions in each class accordingly. From the traceability matrix it was found out that databasemanager class was the most important or the highest priority class and the rest were surrounded around it to support its functioning.

Design Patterns

We employed several design patters in different sections of our code. For the most part, we used the Command Pattern and Delegation in the way that any click of button in the GUI would delegate the next responsibility to some other class via a method call, and that would sometimes delegate to yet another class. The command pattern is also used in Update in the way that it is 'commanded' to update certain variables/parameters. Furthermore, we use direct Request-Based communication in two ways: method calls to the class Update return nothing but they update variables in various objects throughout the program whereas method calls to Compute do not change any parameters, it simply reads certain data to produce graphs so each method in Compute returns a graph. Some of these patterns arose out of our initial intention of how the program should be structured. Others were seen as more advantageous as time went on and were then implemented by changing the way we had originally designed it.

Object Constrained Language (OCL) Contract Specifications

Controller:-

Invariant :- co-ordinates all the actions like registration ,getting data from file and displaying the graphical and user data as well as saving and displaying the feedback.

Pre-Condition:- Application should be opened in User machine.

Post Conditions:- The user is able to register, viewdata or view feedback if use is Patient and should be able to submit feedback as well if user is Doctor. Session Maintenance is also present.

Request:-

Invariant:-User request for viewing doctor feedback , view graph or calculating the data. User should be a valid patient or Doctor

contextController inv:

user == valid(doctor) || valid(patient)

Pre-Condition:-User must be a logged in the system.

Post Condition :- The User request is passed to the concerned event handler depending upon the request.

GetData:-

Invariant:-Must be a valid file type as mentioned in User requirement Section

contextController inv:

filetype == valid(excel)

Pre-Condition:-The file to be uploaded should placed in the system and must contain data in valid format as specified in User requirements.

Post Condition :-Data from Excel file is parsed into the system and saved in the database.

SessionController:-

Invariant :-should receive a request for logoff, must track the time user is logged into the system and request for log- off generated after specified period of inactivity.

Pre-Condition:- User is logged into the system.

Post-Condition :-User Session should be logged out after inactivity of 15 minutes and maintained otherwise until requested log-off

Registration:-

Invariant :-User must be an unregistered user with the system.

Pre-Condition:- All the mandatory personal details asked should be entered by the user, additionally in case of Patient, the doctor(consultant mapped) name must be entered.

Post-Condition :-User gets registered with the system and is ready to proceed with his/her health monitoring.

Authorization:-

Invariant:- User must be an registered user with the system.

contextController inv:

User == valid;

Pre-Condition:- User enters a valid username and password

Post-Condition:- User is logged into the system.

DatabaseManager:-

Invariant:-should create a connection with database only if correct DB credentials are passed.

contextController inv:

DBusenamr == valid;

DBpassword == valid;

Pre-Condition:- Request sent to fectch or save userinformation,userdata or feedback into the database.

Post-Condition:- Requested data is sent to the Controller.

Profile:-

Invariant:-User must be a registered user with the system and must be logged in system.

contextController inv:

user == valid;

Pre-Condition:- Request for viewing or editing profile data.

Post-Condition:- Profile is edited and changes are saved in the database.

UserInterface:-

Invariant:-User must be logged in to receive a request to display profile, calculate data or view graph.

Pre-Condition:-A request is made in user profile for ex. View graphs.

contextController pre:

Bool req_viewgraphs;

If(Bool)

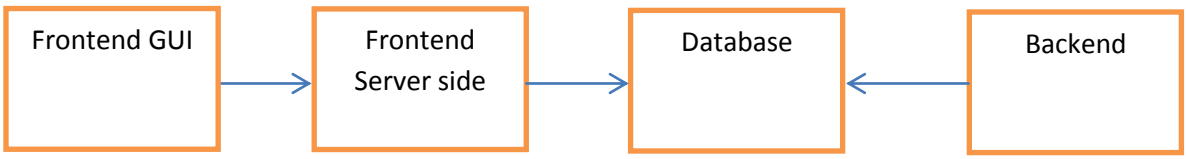
Post-Condition:- Graphs are displayed on the user Interface.

9. System Architecture and System Design

a. Architectural Styles

Before speaking of the kinds of architectural styles[1], 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[8], which is considered to be a Separated-Presentation architectural style[8]. 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



A subsystem is, effectively, a special kind of package that only has interfaces as public elements[1]. The interfaces provide a layer of encapsulation, allowing the internal design of the subsystem to remain hidden from other model elements. The concept subsystem is used to distinguish it from "ordinary" packages, which are semantic-free containers of model elements; the subsystem represents a particular usage of packages with class-like (behavioural) properties.

The frontend GUI[8] 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[9] including field names and types.

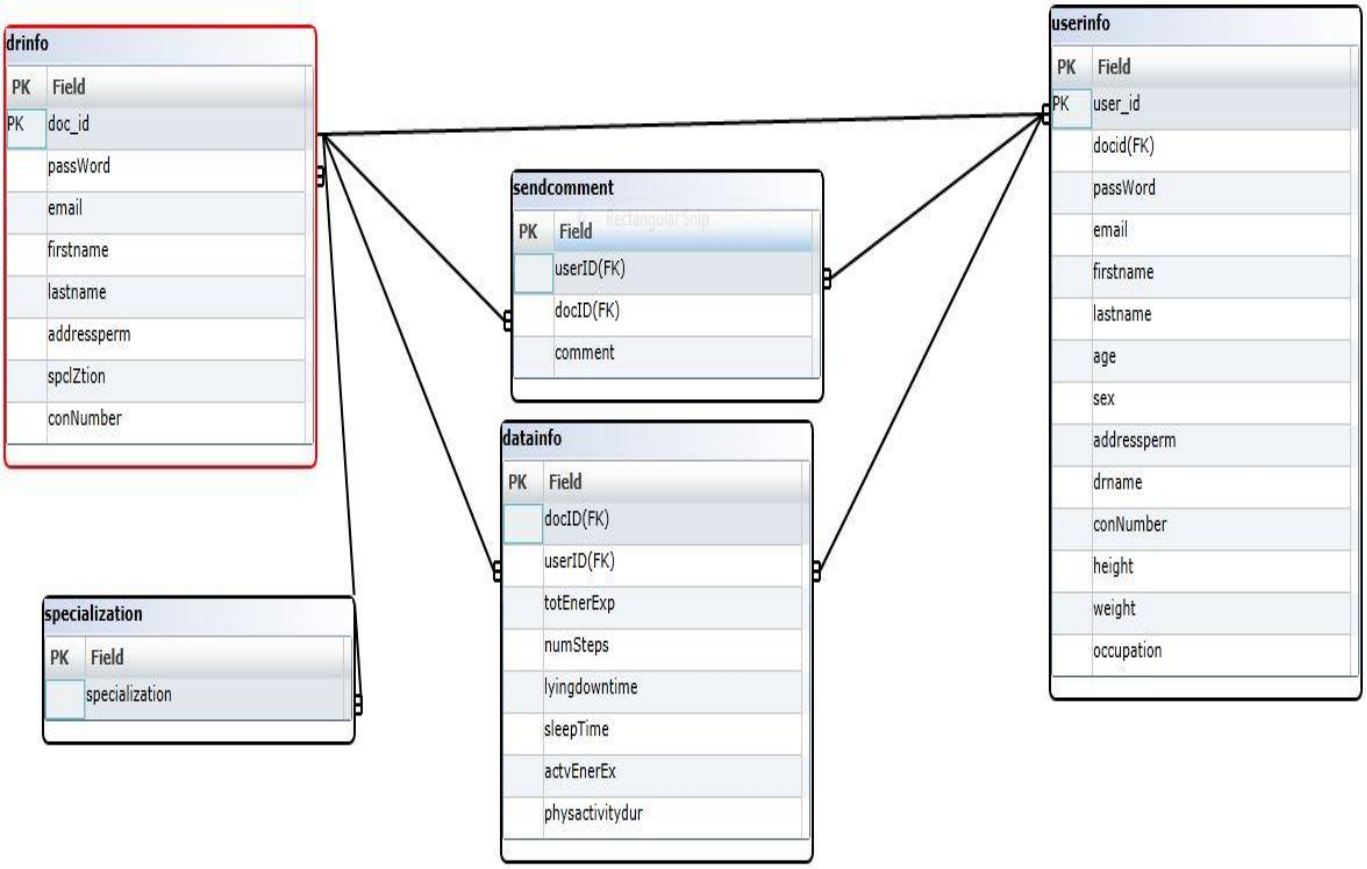


Figure 9.d.1 Database tables

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[18]. The client is comprised of html, JSP[12][13] 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[15]. The backend uses the JDBC driver to connect to the MySQL database[18].

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).

10. Algorithms and Data Structures

Algorithms

BMI:- The body mass index (BMI)[8], or Quetelet index[8], is a heuristic proxy for estimating human body fat based on an individual's weight and height.

BMI = (mass(in lbs)/(height(in inch))^2) * 703

Metabolic Eqquivalent Task(MET) : The Metabolic Equivalent of Task (MET)[8], or simply metabolic equivalent, is a physiological measure expressing the energy cost of physical activities and is defined as the ratio of metabolic rate (and therefore the rate of energy consumption) during a specific physical activity to a reference metabolic rate, set by convention to 3.5 ml $O_2 \cdot kg^{-1} \cdot min^{-1}$ or equivalently:

$$1\ \mathrm{MET} \equiv 1\frac{\mathrm{kcal}}{\mathrm{kg} * h} \equiv 4.184\frac{\mathrm{kJ}}{\mathrm{kg} * h}$$

Physical activity	MET
Light intensity activities	**< 3**
Sleeping	0.9
watching television	1.0
writing, desk work, typing	1.8
walking, 1.7 mph (2.7 km/h), level ground, strolling, very slow	2.3
walking, 2.5 mph (4 km/h)	2.9
Moderate intensity activities	**3 to 6**
bicycling, stationary, 50 watts, very light effort	3.0
walking 3.0 mph (4.8 km/h)	3.3
calisthenics, home exercise, light or moderate effort, general	3.5
walking 3.4 mph (5.5 km/h)	3.6
bicycling, <10 mph (16 km/h), leisure, to work or for pleasure	4.0

bicycling, stationary, 100 watts, light effort	5.5
Vigorous intensity activities	**> 6**
jogging, general	7.0
calisthenics (e.g. pushups, situps, pullups,jumping jacks), heavy, vigorous effort	8.0
running jogging, in place	8.0
rope jumping	10.0

On the basis of different ranges of MET from this table we are able to distinguish different types of physical activities. In the raw data we receive MET per minute and on the basis of this table we can distinguish between moderate, vigorous and very vigorous activities.

Also to calculate the active energy we have used the formula mentioned below

Active Energy = Total Energy – Sedentary Energy expenditure

Based on the above table we have divided the energy expenditure in four different parts for our system as explained below

Sedentary Energy expenditure = Energy when MET is < 1.8

Moderate Energy expenditure = Energy when 2.3 < MET <3.6

Vigorous Energy expenditure = Energy when 3.6 < MET <7

Very Vigorous Energy expenditure = Energy when 7 < MET <10

Data Structures

The main data structure in our system is the Database[9]. 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[9] 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.

11. User Interface Design and Implementation

Login Page:

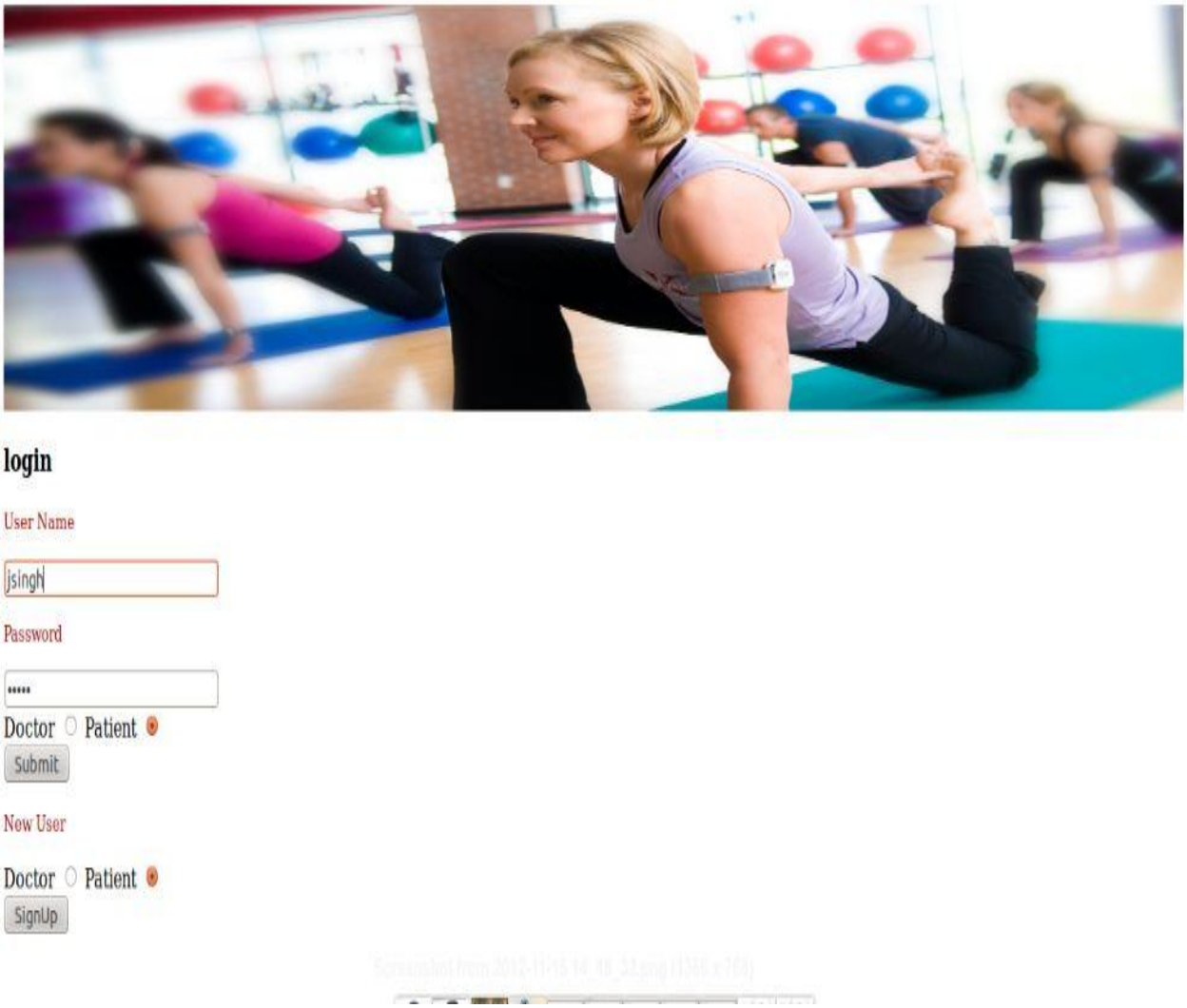


Figure 11.1 Login Page

In the User Interface design[16], In login Screen User can login the application using unique username and password provided during registration. If the user has not registered yet he can go with sign up option where the User has to enter all the information related to him and should specify the doctors' name while registering. This Doctor is then associated with his profile and any review request from this user will be directed to him. Unlike many user interfaces this promises to be more effective because of its simplicity since usually patients are made to enter too many private details like SSN, Nationality, Race, which don't have anything to do with users' diagnosis with which the user may not be comfortable answering. After the registration the user can feel free to access his account with his ID anytime.

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:

Fig 11.2 Home Page

The Homepage as shown in Fig11. 2 provides new menus tabs on the screen. Managing user account, uploading data, calculate, contact information and logout can be found in tab on the home screen. We felt this is an easy and user friendly way to integrate most of the functionality on a single page. The file upload option from the system has also been provided on the same page to reduce the user effort for these functions.

Data Interpretation Using Graphs :-

Number of Steps:-

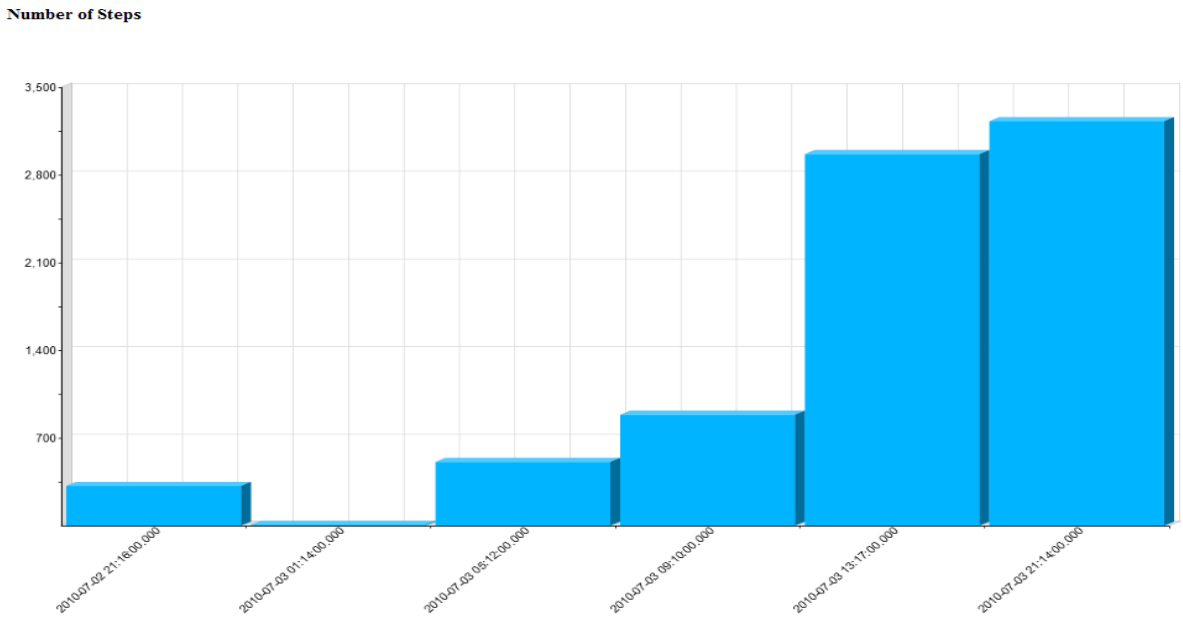


Fig 11.3 Number of Steps

The graph in Fig 11.3 depicts the number of steps taken by the patient over the period of time for which data is requested by the user. The x-axis of graph displays the time duration and y-axis the number of steps. The x-axis is divided into six equal parts of the total time duration chosen. For ex. If data is requested for 24 hours, each bar will depict the steps taken in each four hour duration window. In above graph, we see around 350 steps taken in first 4 hours.

Energy Expenditure Graph:-

Fig 11.4 Energy in calories per minute

The graph in fig 11.4 depicts the Energy expenditure per minute of the patient over the given period of time for which data request is made. The x-axis shows the duration and y-axis shows the energy spent in calories per minute. The red color depicts the increase in energy expenditure from past minute reading while black color shows the fall in energy expenditure as compared to previous minute reading.

Total and Active Energy Expenditure:-

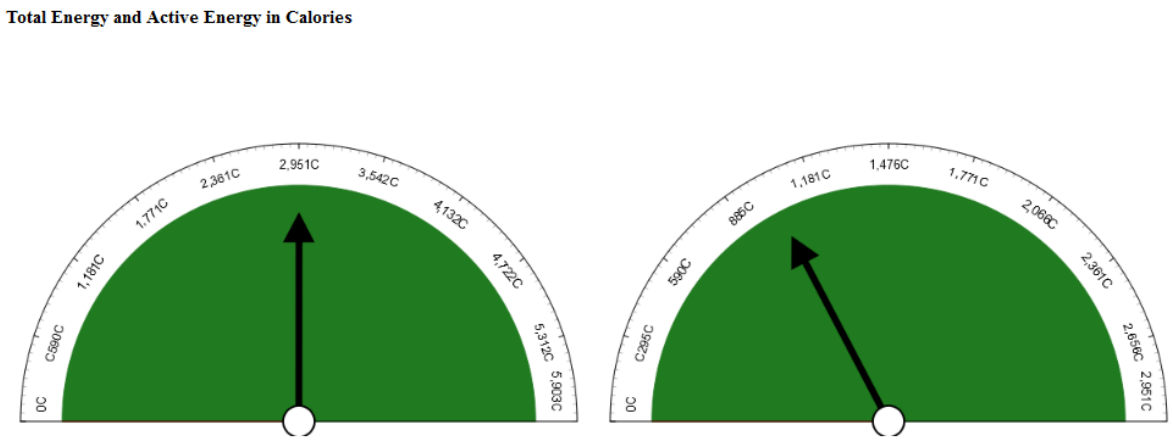


Fig 11.5 Total and Active Energy

In Fig 11.5, the first meter graph shows the total energy expenditure in calories for the whole duration chosen by the user, while the meter on Right hand side shows the active energy expenditure of the patient for the complete duration. Here Active energy means the energy spent while doing some actual physical activity and does not include the sedentary energy expenditure.

Total Physical activity Duration:-

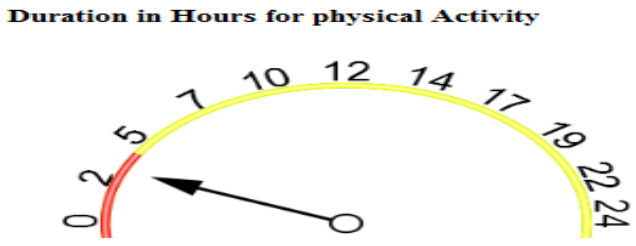


Fig 11.6 Total Physical activity Duration

The meter graph in Fig11. 6 displays the total time spent in hours while doing a physical activity[8] by the user. This physical activity can be of four different types sedentary,

vigorous, very vigorous and moderate .For ex. In above graph time spent is almost 3 hours.

Percentage duration per activity:-

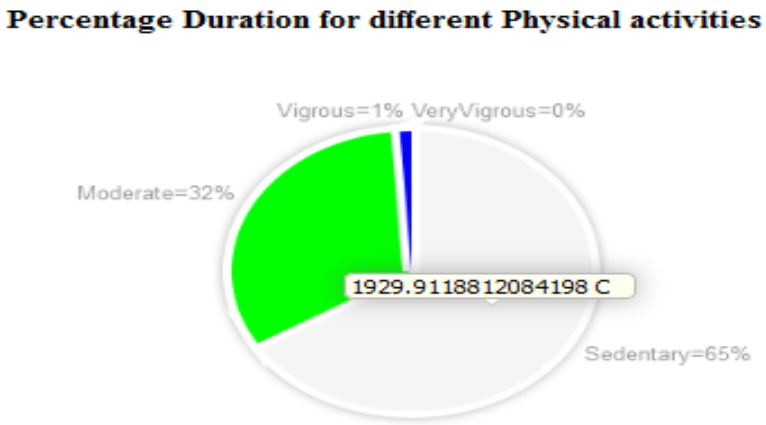


Fig 11.7 Percentage duration

The fig 11.7 graph shows the percentage breakdown of duration for different types of physical activities as explained above. Another feature that we added is that upon clicking on any activity section, we can see the total energy spent in calories while doing that activity.

Sleep Duration:-

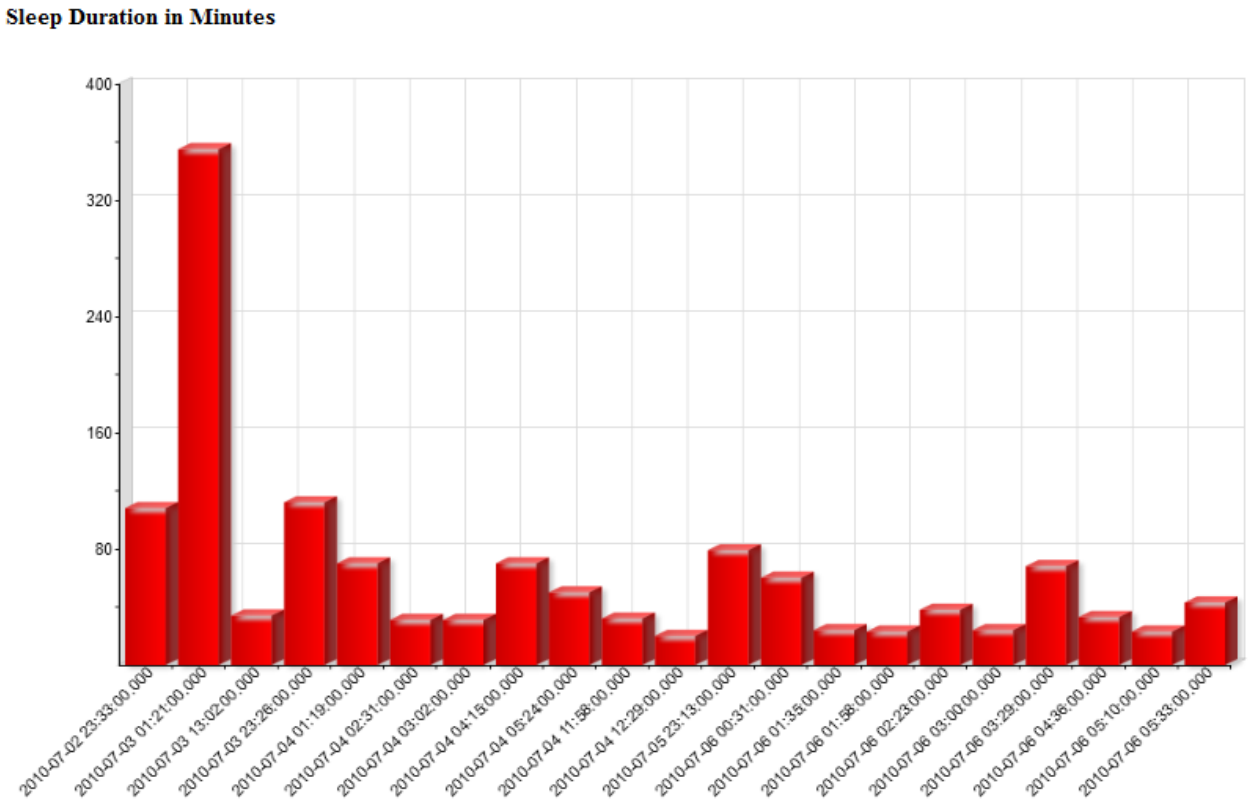


Fig 11.8 Sleep Duration

The graph in figure 11.8 shows the sleep pattern of a patient for past one week. The x-axis shows the time at which a user starts his sleep and y-axis shows the number of minutes the user sleeps for. An important feature that can distinguish this application from other standard software's is that in sleeping time, lying down time (i.e. person lying down but not sleeping) is not included in the sleeping time. Every bar will break as the patient gets awakened and starts again as the patient sleeps again. Regular discontinuous small bars depicts that the patient is suffering from some kind of insomnia.

Lying Down Time:-

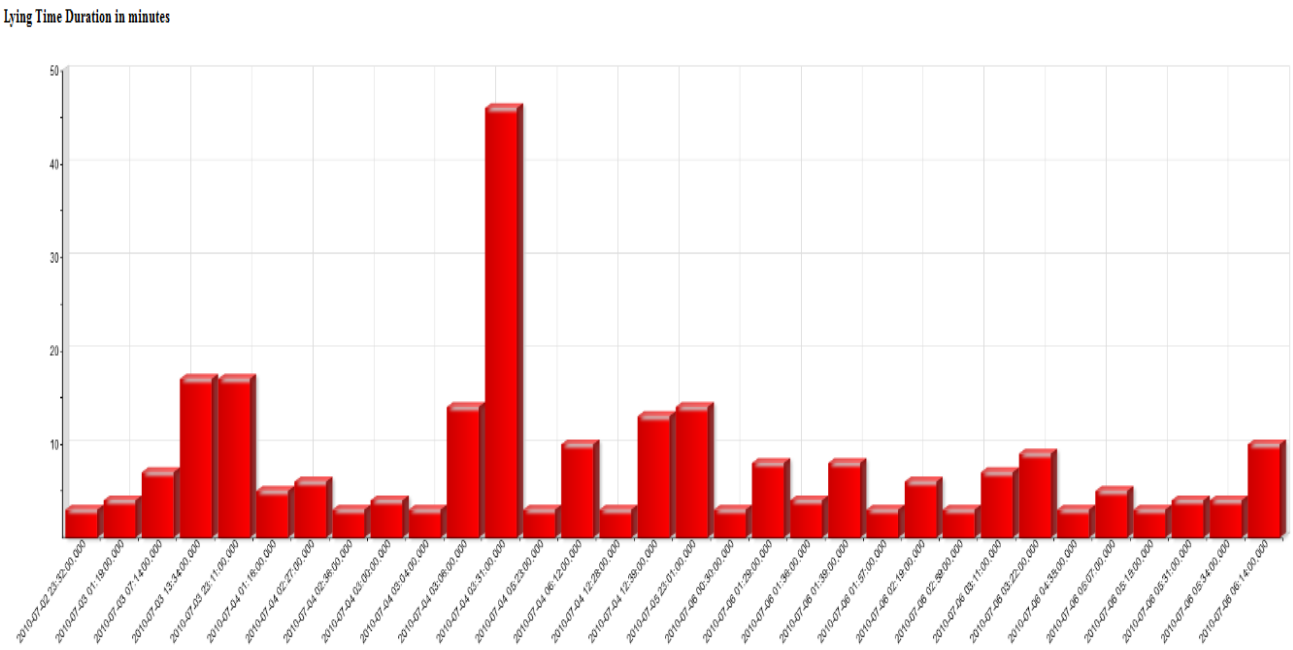


Fig11. 9 Lying down time

Figure 11.9 displays the lying down pattern similar to the sleeping time depicted and explained above in figure 11.8.In this lying down time, sleep time is not included. For instance, long bars show that the person has been doing rest for longer durations.

12. Design of Tests:-

These are test cases for determining the correctness of implemented structures in the program [1][8]

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 (Eg. “Invalid User ID/Pssword”)	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.

Table 12.1 Login_TC

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.(Eg.”User ID already in use.Pleaseselect a different ID”).	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.

Table 12.2 Register_TC

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.(Eg. “Wrong file selected”).	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. (Eg.”Invalid Format ”).	Pass if result meets the expected result.	Only .xls files are allowed to be uploaded.

Table 12.3 Upload_TC

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.

Table 12.4 Mapping_TC

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.

Table 12.5 UserReview_TC

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.

Table 12.6 EditProfile_TC

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. (Eg.”Check your date and time”).	Pass if result meets the expected result.	This test shows that user cannot request information that is not present in the database.

Table 12.7 ViewInfo_TC

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.

Table 12.8 Viewformat-TC

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.

Table 12.9 DoctorFeedback_TC

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.

Table 12.10 ViewFeedback_TC

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.

13. History of Work, Current Status, and Future Work

We started this project with an novel idea that we all as team members believed can be really innovative and useful in the field of Biometric healthcare Monitoring. Considering the challenges and opportunity that it presented ,we were very confident from the beginning that we have a great deal to learn through this project .There were a lot of key accomplishments and milestones that occurred throughout the duration of the creation of the project. Even some of the smallest feats were actually huge accomplishments toward the success of the overall project. Getting two functions to work together was a milestone each and every time it worked without failure. Seeing a code compile without any runtime or exceptions for the first time was also a big deal each time it happened. Each little thing that worked correctly contributed to the overall success of the project and with one miniscule error, the whole project could have potentially not worked or just flat out crashed altogether.

We started our project work by searching for a health monitoring device by contacting the various firms to get details about their product. Initially we faced this problem that most of the companies were not willing to share the details about their products at the research level. Then we came across this sensewear band and after we communicated with their product department they agreed to provide us the patient data to work-on, However they insisted upon buying the licence in case we required the information and decoding algorithms to obtain that data from device. But, once we talked about this with Professor Marsic, he assured it would be fine and novel idea if we can develop an application using the raw data provided by the company. It made sense as primarily our project was a software project and our project proposal contained lot of Software engineering implementation challenges.

We submitted our proposal on September 22 and formed a schedule of meeting twice in a week for discussing the progress and problems faced in our assigned responsibilities. Although most of the team members were good at programming Since not all were familiar with JSP MVC framework (we felt it was best web framework to work on in our project), we assigned the responsibilities accordingly. All the team members were very committed and dedicated from the very first day and most of the times we were able to meet the submission deadlines by days in ahead. However as is always in case of challenging project, we were bound to face challenges.

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 were having was 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 relied 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 relied 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 was that everyone can keep track of each others' progress.
Also since most of our other courses were different, so managing our schedules was really tough, however we solved this problem by following a proper schedule decided earlier and strictly following it. We prepared the Customer requirements after careful analyses of the problem at our hand from the user's perspective and we arrived at the conclusion that we would require about 10 Use cases to wholly cover the problem domain. However, later on we came to decision that we can complete the project in a modular way by using 8 Use cases. Domain Analyses and System Sequence diagrams were completed on October 9 We submitted our First project report on October 14,2012.

While we started working on our project demo and report 2, we started receiving the report 1 reviews from professor and other groups. Most of the reviews, except few, were really encouraging, helpful and thorough in appreciating the workdone by our group and also in pointing out the scope for improvements in our report. So we divided each section of report 1 between each member and we made sure that all parts were updated and corrected wherever required according to the reviews and our own understanding. We had decided to implement the MVC software architecture (using JSP). The class structure that evolved from domain concepts gave us a clear idea that we were heading towards the right direction. It also gave us the confidence that we would be able to meet all the use cases and even exceed in providing some functionality in some cases. For example, in case of displaying the sleep graph, initially we had planned to display the bar graph showing the only the total sleep hours per day. But after some brainstorming we came with a novel idea (i.e. presently implemented, refer Userinterface) that would allow a lot more useful information to be extracted from the graphical data.

Since most of our work was always complete well ahead in time, we were relatively less affected by the disruptions caused by the Sandy Disaster that also caused the delay in class first project demo. We had already submitted the first part of report two until before our demo. We presented our first demo on October 9 and were able to implement almost half of the use-cases by then.
The last one month was really hectic as we hurried up towards completing the integration testing of all the test cases. Since our project included lot of graphical data and not all members were familiar with this development platform, we encountered few issues as

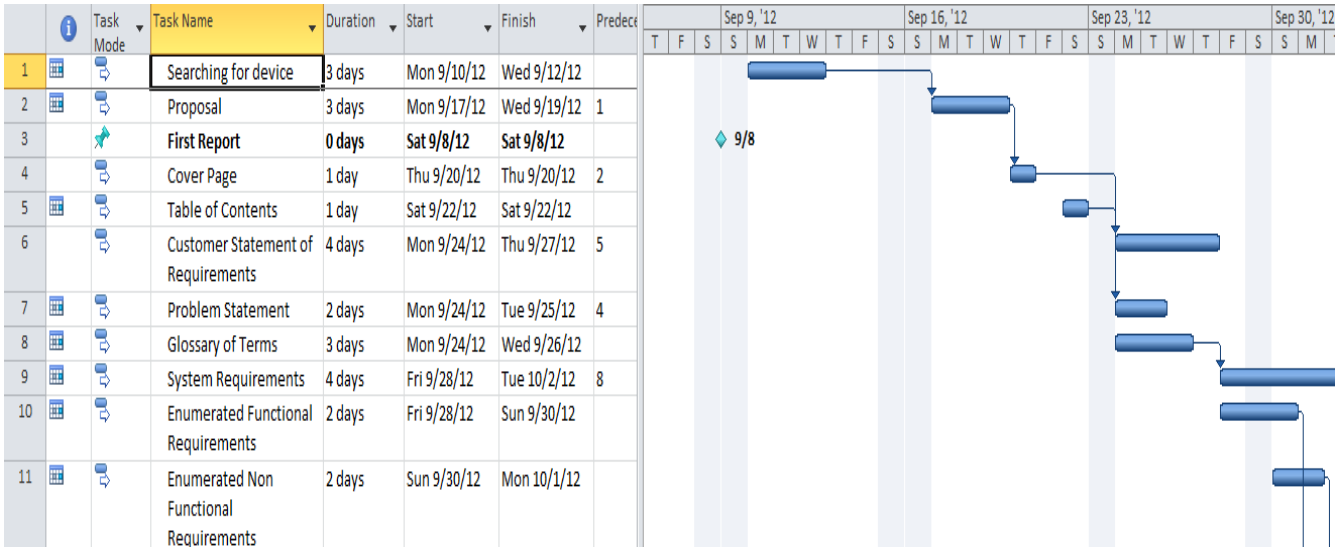
sometimes the graphs would not come up even while entering the correct date values. However careful insight into the issues and constant never ending debugging allowed us to move ahead each such issue.
Our coding and Unit testing was completed in first week of the December, however after integrating the code we found out few of the modules were not working proper. It took us one week to test the integrated code and solve all issues associated.
We were able to complete our project by 10th of December with complete coding effort and integration testing of all the test cases. In Overall we completely implemented all the test cases that we presented in the first report and our application met most of the requirements that we identified after customer requirement analyses. We believe we have been able lay a very good base for a research project at a large scale with lot of future scope and possibilities of implementations with innumerable enhanced features.

Future work:-

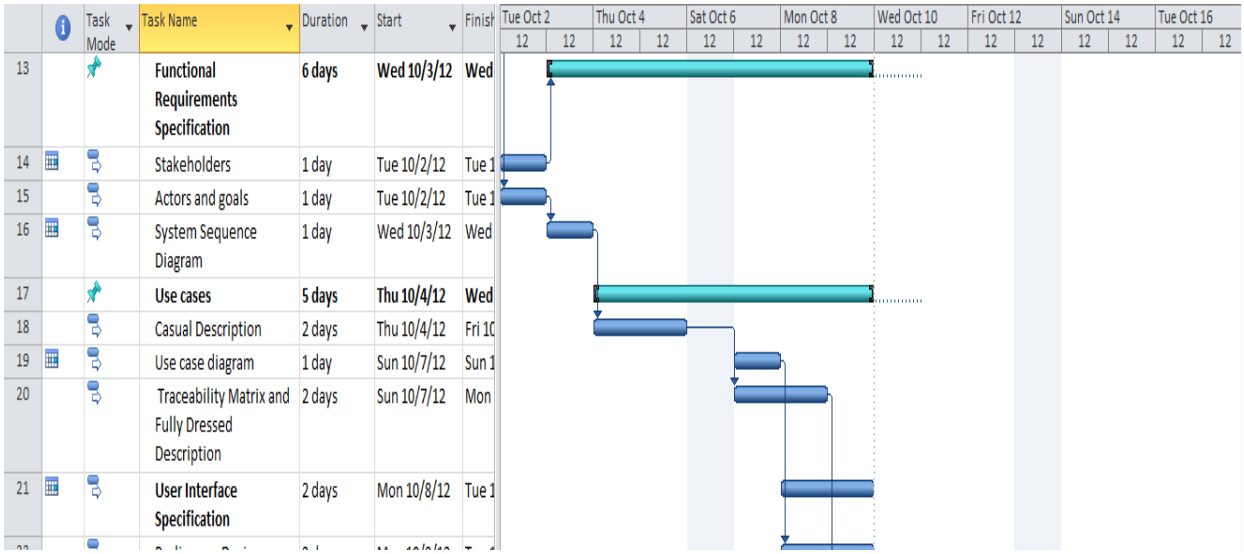
We started working on this project with an eye on developing an application for long term benefits. As of now, lot of useful features have already been incorporated into this application and we feel it has met its primary objective of saving time and unnecessary botheration by communicating timely diagnoses and feedback from doctor to the patient. However we feel that still there are lot many features that can be included in this application in the long run. Some features that we plan to implement in near future are allowing the direct communication from patient to doctor(like feedback is present from doctor to patient) and the as advised by Professor Marsic changing the sleep and sedentary time bar graphs where we can display one discontinuous bar per day where the gaps in bar would show time the user is not sleeping.

This application can be extended into an android application that can provide enhanced mobility to the user as well as doctor. As they say *"there is nothing called free lunch"* there are few doctors who are not willing to provide their services free of cost. Hence we plan to include the PayPal system in this application so that user can be provided with wide range of specialized doctors.

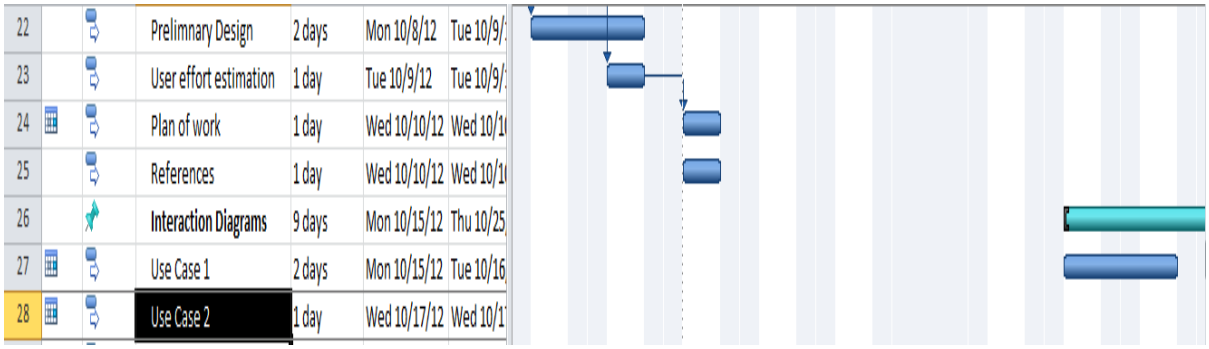
As of now the application provides and analyses the data in static mode. We can extend its application for the scenarios when user is wearing the device armband and he/she will be provided with the current progress and statistical indicators of his activity/health.



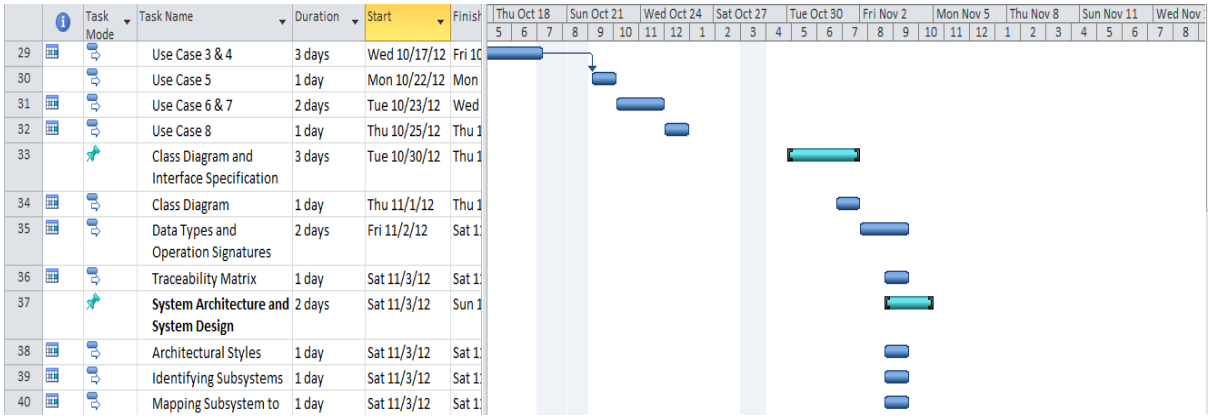
	Task Mode	Task Name	Duration	Start	Finish	Predece
1		Searching for device	3 days	Mon 9/10/12	Wed 9/12/12	
2		Proposal	3 days	Mon 9/17/12	Wed 9/19/12	1
3		**First Report**	**0 days**	**Sat 9/8/12**	**Sat 9/8/12**	
4		Cover Page	1 day	Thu 9/20/12	Thu 9/20/12	2
5		Table of Contents	1 day	Sat 9/22/12	Sat 9/22/12	
6		Customer Statement of Requirements	4 days	Mon 9/24/12	Thu 9/27/12	5
7		Problem Statement	2 days	Mon 9/24/12	Tue 9/25/12	4
8		Glossary of Terms	3 days	Mon 9/24/12	Wed 9/26/12	
9		System Requirements	4 days	Fri 9/28/12	Tue 10/2/12	8
10		Enumerated Functional Requirements	2 days	Fri 9/28/12	Sun 9/30/12	
11		Enumerated Non Functional Requirements	2 days	Sun 9/30/12	Mon 10/1/12	



	Task Mode	Task Name	Duration	Start	Finish
13		**Functional Requirements Specification**	**6 days**	**Wed 10/3/12**	**Wed**
14		Stakeholders	1 day	Tue 10/2/12	Tue 1
15		Actors and goals	1 day	Tue 10/2/12	Tue 1
16		System Sequence Diagram	1 day	Wed 10/3/12	Wed
17		**Use cases**	**5 days**	**Thu 10/4/12**	**Wed**
18		Casual Description	2 days	Thu 10/4/12	Fri 10
19		Use case diagram	1 day	Sun 10/7/12	Sun 1
20		Traceability Matrix and Fully Dressed Description	2 days	Sun 10/7/12	Mon
21		**User Interface Specification**	2 days	Mon 10/8/12	Tue 1

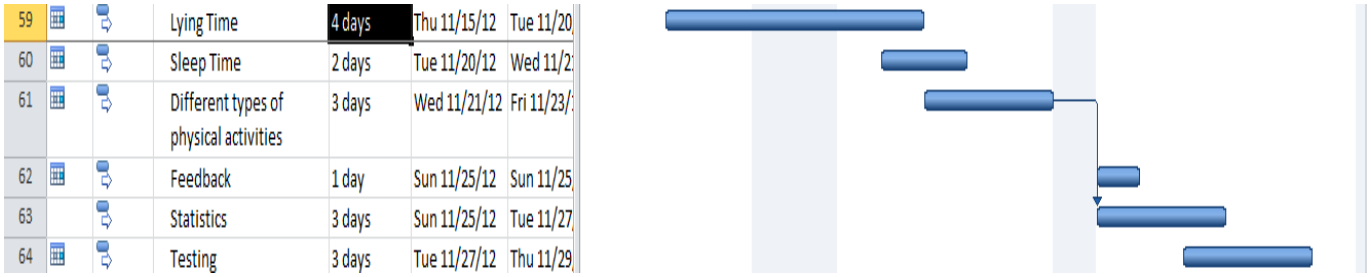


22		Prelimnary Design	2 days	Mon 10/8/12	Tue 10/9/
23		User effort estimation	1 day	Tue 10/9/12	Tue 10/9/
24		Plan of work	1 day	Wed 10/10/12	Wed 10/1
25		References	1 day	Wed 10/10/12	Wed 10/1
26		**Interaction Diagrams**	9 days	Mon 10/15/12	Thu 10/25
27		Use Case 1	2 days	Mon 10/15/12	Tue 10/16
28		Use Case 2	1 day	Wed 10/17/12	Wed 10/1

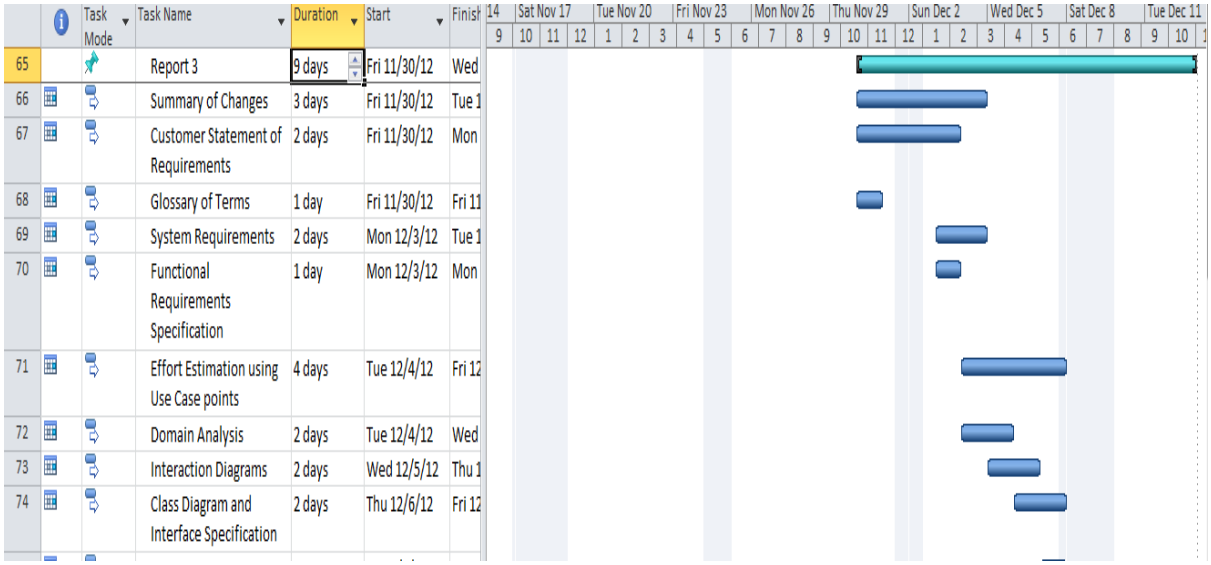


	Task Mode	Task Name	Duration	Start	Finish
29		Use Case 3 & 4	3 days	Wed 10/17/12	Fri 10
30		Use Case 5	1 day	Mon 10/22/12	Mon
31		Use Case 6 & 7	2 days	Tue 10/23/12	Wed
32		Use Case 8	1 day	Thu 10/25/12	Thu 1
33		Class Diagram and Interface Specification	3 days	Tue 10/30/12	Thu 1
34		Class Diagram	1 day	Thu 11/1/12	Thu 1
35		Data Types and Operation Signatures	2 days	Fri 11/2/12	Sat 1
36		Traceability Matrix	1 day	Sat 11/3/12	Sat 1
37		**System Architecture and System Design**	2 days	Sat 11/3/12	Sun 1
38		Architectural Styles	1 day	Sat 11/3/12	Sat 1
39		Identifying Subsystems	1 day	Sat 11/3/12	Sat 1
40		Mapping Subsystem to	1 day	Sat 11/3/12	Sat 1

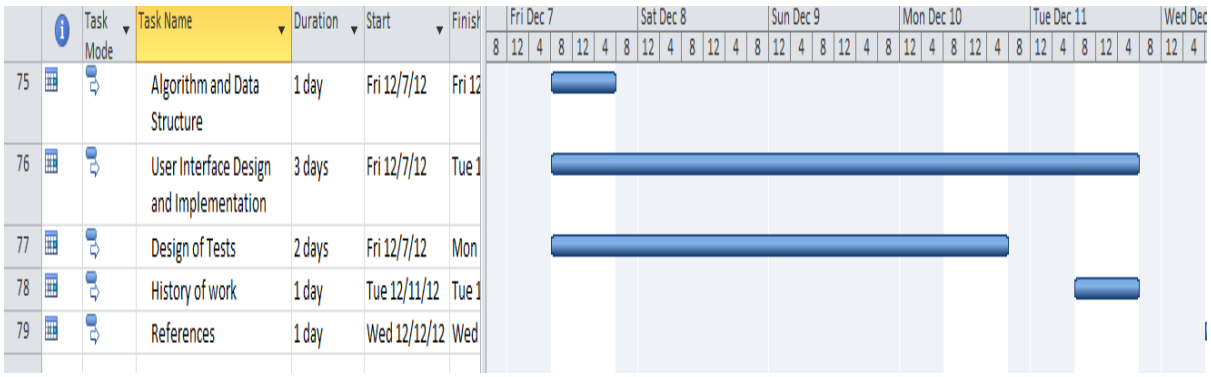
41		Persistent Data Storage	1 day	Sun 11/4/12	Sun 1
42		Network Protocol	1 day	Sun 11/4/12	Sun 1
43		Global Control Flow	1 day	Sun 11/4/12	Sun 1
44		Hardware Requirement	1 day	Sun 11/4/12	Sun 1
45		Algorithm	1 day	Mon 11/5/12	Mon
46					
47		Data Structure	1 day	Mon 11/5/12	Mon
48		User Interface Design and Specification	2 days	Mon 11/5/12	Tue 1
49		Design of Tests	1 day	Tue 11/6/12	Tue 1
50		Project Management and Plan of Work	1 day	Tue 11/6/12	Tue 1
51		Demo1	3 days	Tue 11/6/12	Thu 1
52		Making ppt	2 days	Tue 11/6/12	Wed
53		Making fliers	1 day	Wed 11/7/12	Wed
54		Preparing for presentation	1 day	Thu 11/8/12	Thu 1
55		Implementing the 6 parameters in graphical format	14 days	Thu 11/8/12	Mon
56		Number of steps	3 days	Thu 11/8/12	Sat 1
57		Total Energy	3 days	Sat 11/10/12	Tue 1



59	Lying Time	4 days	Thu 11/15/12	Tue 11/20
60	Sleep Time	2 days	Tue 11/20/12	Wed 11/2
61	Different types of physical activities	3 days	Wed 11/21/12	Fri 11/23/
62	Feedback	1 day	Sun 11/25/12	Sun 11/25
63	Statistics	3 days	Sun 11/25/12	Tue 11/27
64	Testing	3 days	Tue 11/27/12	Thu 11/29



	Task Mode	Task Name	Duration	Start	Finish
65		Report 3	9 days	Fri 11/30/12	Wed
66		Summary of Changes	3 days	Fri 11/30/12	Tue 1
67		Customer Statement of Requirements	2 days	Fri 11/30/12	Mon
68		Glossary of Terms	1 day	Fri 11/30/12	Fri 11
69		System Requirements	2 days	Mon 12/3/12	Tue 1
70		Functional Requirements Specification	1 day	Mon 12/3/12	Mon
71		Effort Estimation using Use Case points	4 days	Tue 12/4/12	Fri 12
72		Domain Analysis	2 days	Tue 12/4/12	Wed
73		Interaction Diagrams	2 days	Wed 12/5/12	Thu 1
74		Class Diagram and Interface Specification	2 days	Thu 12/6/12	Fri 12



	Task Mode	Task Name	Duration	Start	Finish
75		Algorithm and Data Structure	1 day	Fri 12/7/12	Fri 12
76		User Interface Design and Implementation	3 days	Fri 12/7/12	Tue 1
77		Design of Tests	2 days	Fri 12/7/12	Mon
78		History of work	1 day	Tue 12/11/12	Tue 1
79		References	1 day	Wed 12/12/12	Wed

14. 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://www.uml.org/ (UML software source).

5. http://www.sparxsystems.com/resources/uml2_tutorial/uml2_classdiagram.html (UML tutorials)

6. http://sensewear.bodymedia.com/

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

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

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

10. Website Template – sites.google.com.

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

12. Ding Dong. Introduction of JSP. 2008

13. Shi Zhiguo, XueWeimin, Dong Jie. Application Course Of JSP. Oct 2004. TSINGHUA UNIVERSITY PRESS.

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

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

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

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

18. http://www.mysqltutorial.org/ (For mySQL basics).