**Software Requirements Specification**

**(SRS)**

*February 6, 2015*

INSURANCE MANAGEMENT SYSTEM

**DanRamFar Insurance**

*Ramiz Cohadarevic*

*Farooq Abrahem*

*Daniel Walker*

**(Group 1)**

CS 319 Software Engineering

# Table of Contents

[Table of Contents i](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487619)

[1.0. Introduction 1](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487621)

[1.1. Purpose .1](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487622)

[1.2. Scope 1](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487623)

[1.3. Glossary 2](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487624)

[1.4. References 3](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487625)

[1.5. Overview 3](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487626)

[2.0. Project Description 4](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487627)

[2.1 Product perspective 4](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487628)

[2.2 Product functions 4](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487629)

[2.3 User characteristics 5](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487630)

[2.4 Constraints 5](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487632)

[2.5 Assumptions and dependencies 5](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487634)

[3.0. Functional Requirements 6](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487650)

[3.1 User Case Diagrams 6](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487653)

[3.2 Use Case Description 10](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487654)

[3.3 Behavior requirements 14](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487655)

[3.4 System models 22](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487656)

4.0. Non-Functional Requirements 47

[4.1 Performance Requirements 47](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487659)

[4.2 Dependability Requirements 47](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487660)

[4.3 Other Quality Attributes 47](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487661)

5.0  [Appendices 48](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487662)

[5.1 Documentation 48](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487663)

[5.2 Hardware Description 48](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487664)

[5.3 Database system description 48](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487665)

[5.4 Business rules and policies 48](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487666)

[Index 48](file:///C:\Users\Ramiz\AppData\Local\Temp\SRSExample-webapp.doc#_Toc77487668)

# 1.0. Introduction

## 

## 1.1. Purpose

The purpose of this software requirements specification document is to in greater detail describe the architecture and design of our insurance management system. It will explain the different levels and their requirements. The SRS will describe what our software is intended to do how and how should perform. Features of our system include the ability for a customer to research and buy different polices offered, mange existing polices, make a payment, view payments made, contact our organization as well as get support at any time. The system is supposed to allow the user to browse our site and explore different kinds of polices offered, select desired polices, make payments and receive assistance when it’s needed. This document is intended for the developers of our team who can more easily understand the guidelines for future development. It is also intended for any customer who wishes to know more about what our system is capable of.

**1.2. Scope**

DanRamFar Insurance is an insurance management company offering several policy types for our customers, customers can browse through different types offered, make payments, view previous payment history, and get customer assistance at any point in time. This document cover all the functionality that our insurance management application will provide.

From the functional requirements describing in detail every use case for each module and what the system should do. The non-functional requirements describing how the system will work as a whole.

The goals of our insurance management company is to produce a product that will meet the customer’s needs in buying different kinds of insurance policies. We plan to be fully done and our product to be fully functional by April 29 2015. We will deliver on all listed features, at minimal to no cost, and produce an efficient and useful product by the deadline.

## 1.3. Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Customer | The user of the product. |
| Admin | The administrator who has control of system. |
| Policy | The specific type of insurance. |
| SRS | Software requirements specification. |
| Module | Area containing info to a certain set of tasks. |
| Use-Case | Diagram showing representation of how user will use programs interface. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## 

## 1.4. References

(n.d.). Retrieved February 10, 2015, from https://neiu.desire2learn.com/d2l/le/content/2880867/viewContent/357054/View

(n.d.). Retrieved February 10, 2015, from https://neiu.desire2learn.com/d2l/le/content/2880867/viewContent/357055/View

(n.d.). Retrieved February 10, 2015, from http://www.cse.chalmers.se/~feldt/courses/reqeng/examples/srs\_example\_2010\_group2.pdf

Customized Options for Restaurant Owners. (n.d.). Retrieved February 10, 2015, from http://www.farmers.com/business/industry/restaurant/

## 1.5. Overview of Document

The rest of the SRS will cover the overall description of the product. It will provide the project perspective, project functions, user characteristics, constraints, and assumptions and dependencies. Next it will cover the function requirements. This section will describe in detail what the system should do. It will include the use case diagrams, the use case descriptions, the behavior requirements, system models and the entities used. All of these things will be covered for each module. We then have the non-functional requirements which include the performance Requirements, dependability requirements, and other quality attributesional requirements which will provide an overview of how the system should work. The last two things are the appendices which will include additional l information if needed, and lastly the index on where key terms can be located on the SRS document.

**2.0 Overall Description**

***2.1 Project Perspective***

This project intends to make a fully functional insurance program that has an easy to use system for customers. Using simple navigation customers will easily be able to locate links due to descriptive information on the exact function of each page. The business processes should be quick and function to serve several clients.

***2.2 Project Functions***

Within this project the users will have several functions to create and manage their account. They will be able to select preferences to help them choose a plan that best suits them. Then they will be able to choose a plan, as well as a payment options for that plan. They can then manage the policy and make changes as need, as long they fit the stated deadlines.

The customer will also be able to submit payments. They will be able to set a payment plan. Then can view information on their policy, invoices and their balance. Admin will be able to update the data on invoices. They will also be able to view the charges of each payment, to verify the amount asked in the payment history. They will also be able to view when a payment is due.

The customer will be able to contact the customer service by email, a phone number, or by posting a question on the support forum. There will also be a list directory of several employees and departments so that the customer can easily contact their plan representative.

***2.3 User Characteristics***

The customer will need to be able to speak English as the site will be written in that language. The customers will need to able to keep track of their invoice numbers, so that when they need support their account will easily be able to find. They also will need to be responsible and make payments on time or their service will be charged extra. Continued failure to make payments on time will result in termination of the services and legal action will be carried out.

***2.4 Constraints***

The project’s success will be factored by the time management of the team, as far as making deadline. The level of knowledge of PHP could cause struggle in completing certain tasks. Team Communication is also very important and if not done correctly can halt the projects movement. Failure to complete a part on time could halt other functions of the project.

***2.5 Assumptions and Dependencies***

This project depends on the ability for customer to be able to make payments through the system. It depends on maintenance to address customer concerns. This project assumes that all the pieces will properly connect and pass the data correctly between the user and the server.

**3.0. Functional Requirements**

***3.1 Use Case Diagrams***

**Research/Buy Policies and Manage my Policies Modules**

*Research/Buy Policies Module*

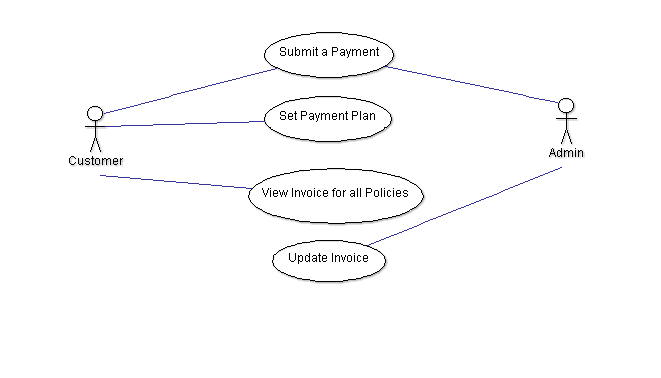


*Manage my policies Module*

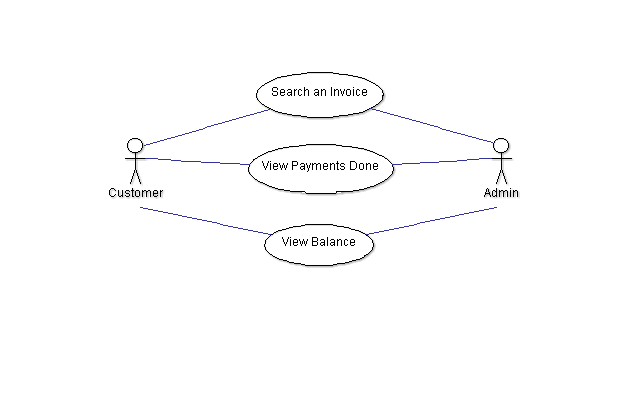


**Billing and Billing History Modules**

*Billing Module*

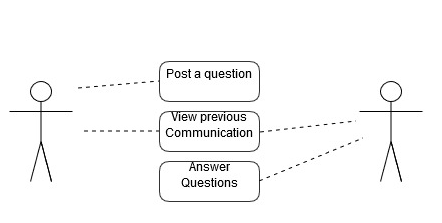
****

*Billing History Module*

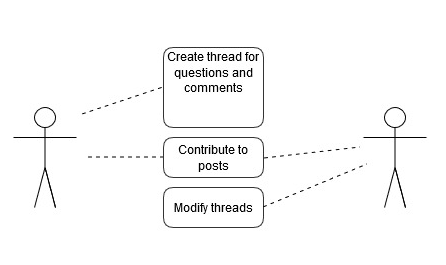
****

**Modules Contact and Support**

*Contact Module*

****

*Support Module*

****

***3.2 Use Case Description***

**Modules Research/Buy Policies**

Customer and Admin will be able to search and find the right policy for their products.

The “policy search” use case has the following:

1. User will be able to write a text and search for it in the policy search field.
2. User click on policy, the policy will show deductibles, and policy description.
3. User will be able to see the total of deductibles and items.
4. User will be able to click the buy policy.

The “Buy policy” use case has the following:

1. Policy agreement terms.
2. System requires the user to accept the agreement terms.
3. After agreement terms have been accepted, the buy policy should be active.

**Modules Manage My Policies**

This module allows customer and admin to edit policies items, like deductibles.

The “Manage my Policy” use case has the following:

1. Edit deductibles.
2. Edit covered item description.

The “Submit changes” use case has the following:

1. User can submit changes made on their current policy.
2. User agrees to the changes total by submitting changes.

**Module Billing**

Any customer can submit a payment with the system. Administrators can also submit payments on behalf of their customer. The system will validate the payment information to ensure the user is putting in the right information. Once the payment is confirmed the user will get a message and the amount paid will be reduced from their balance.

The “Submit a Payment” use case contains the following basic steps:

1. User fills payment information and submits data.
2. System validates data.
3. System confirms payment and sends confirmation message.

Customers can set a payment plan on how they wish to be charged for payments of invoices. The system will give them the option to pay for the invoice manually or automatically pay monthly, every 3 months, or annually.

The “Set Payment Plan” use case contains the following basic steps:

1. User enter the number of the invoice they wish to set payment.
2. The system confirms data and displays all available payment option.
3. User then must select one payment option and submit data.
4. The system validates data and send confirmation on user selection.

Customers can view the current pending invoices of all their policies per plan that is setup on their account. This will allow the user see when payments are due for invoices.

The “View Invoice” use case contains the following basic steps:

1. User selects a policy they wish to view current invoices and submits.
2. User selects invoice ID # from the list to view more specific information on invoice.
3. The system retrieves data and displays all current invoices for selected policies

Administrators can update the information for invoices for their clients to view. This information can pertain several things such as whom charged the invoice, the amount charged, the deducted amount, etc.

The “Update Invoice” use case contains the following basic steps:

1. User enters invoice id and submits.
2. The system displays the data on invoice
3. User can edit data on invoice and submit.
4. Systems sends back confirmation.

**Module Billing History**

Both the customer and administrators can search for invoices. The user can submit the invoice id or selects a date range to view all invoices open during that time.

The “Search an Invoice” use case contains the following basic steps:

1. The user enters the invoice id or selects date range and then submits.
2. The system will display either the invoice matching id or invoices with date range.
3. User can select one to view information on it.
4. The systems displays the selection.

Both the customers and administrators can see the payments done for customer. The user enters the account ID and selects a date range to view all payments made during that time.

The “View Payments Done” use case contains the following basic steps

1. The user enters the account id or selects date range and then submits.
2. The system will display all payments made during the date range.
3. User can select one to view information on it.
4. The systems displays the selection.

Both the customers and administrators can view the balance for customer. The user enters the account ID and submits. The system will display the current balance for that account ID. It will also display pending changes if any payment transfer are in effect.

The “View Balance” use case contains the following basic steps

1. The user enter their account id and submits
2. The system displays the current balance with pending charges if applicable for user.

**Modules Contact**

Post a comment-

Any current customer register should be able to post a question and await a response from a representative. The user will be allowed to ask as many questions he or she may like. The admin will not have the ability to ask questions.

“Post a question” use has these basic steps:

1. User fills and submits form.
2. Systems requests user to type in a code displayed as security measure to avoid spam.
3. System validates code.
4. System creates a thread with customer’s question.

View previous communication-

Both customer and Administrator will have the ability to view all previous communication with each other. Customer will have the ability to see all the communication made with our organization, for every thread made. Admin will have the ability to see all the answer made on questions as well as the answers made by fellow coworkers.

“View previous communication” use has these basic steps:

1. User has an option to click “All communication”.
2. System retrieves and displays every thread made by user.

The admins will have the ability to post answer to any of the questions submitted by the customers. Any admin has this ability and may at any time view questions made by customers and answer if he or she chooses.

“Answer questions” use has these basic steps:

1. System retrieves all questions posted by customers.
2. Admin fills reply form and submits answer.
3. System creates a replay message within the customer’s original question.

**Module Support**

Create thread for questions and comments-

Customers will have the ability to post questions in a support forum. Whenever a customer wishes to ask a question in the forum a thread will be made. Each question will have its own thread.

“Create thread for questions and comments” use has these basic steps:

1. User fills in and submits a new thread.
2. System requires security code as spam precaution.
3. System validates code.
4. System creates a new thread if correct, refresh with error if not.

Contribute to posts-

Both customer and Admin will the ability to view thread already made. Customers and admins will have the ability to contribute to posts. This will allow not only admins to assits with question, but customers who are eager to help one another as well.

“Contribute to posts” use has these basic steps:

1. User and admin reply to threads already made.
2. System posts the comment.

Modify threads-

Admins will have the ability to monitor and control threads. With features like the ability to edit and delete thread and posts. They may delete inappropriate thread as well as modify comments and thread their discretion.

“Modify threads” use has these basic steps:

1. Admin attempts to edit thread or posts.
2. System updates thread or post.
3. System displays modified thread or post.

***3.3 Behavior Requirements***

**Module Research/Buy Policies**

|  |  |
| --- | --- |
| Use Case ID:1 | View Insurance |
| Description | Both customers and Admins will be able to view current insurance information through the search/buy policy page. Customers will be able to view their own personal insurance information whenever accessing the website. Admins will be able to view all customer insurance information when accessing the website via a customer list.  After a user’s level of access has been identified through the System login process they will be presented with all information relevant to their level of authorization. |
| Input | * View Insurance link available through website navigational bar (Customer) * Customer name selected from database for individual customer insurance (Admin) |
| Output | * + Customer will be presented with results pertaining to their level of access. If a patient selects “view insurance” they will be presented with their current insurance provider and any information pertaining to that provider that is deemed necessary.   + Admins will be able to search for individual customers insurance providers and will be presented with the same information that is presented to the customer. |
| Preconditions | The User has already set up an account with their appropriate level of access. If a first time customer user they will need to input first time insurance information. |
| Post Conditions | After selecting “view insurance” the customer will be provided with the requested page that contains their current insurance information  If admins, after searching for a customer they will be provided with the customer information they requested. |
| Action | System will validate the following rules before displaying insurance information:   * Current insurance information has already been entered by customer/admins for individual customer viewing. * Level of access has been established via the login process to display the user with the appropriate options (customer database only available to admins) |

**Module Manage My Policies:**

*The following table outlines the “Edit/Modify Insurance” use case.*

|  |  |
| --- | --- |
| Use Case ID: 1 | Manage my policies |
| Description | A customer will have the ability to edit/modify their current policy information at any time. The customer will select “edit policy” when accessing the insurance module of the system. The customer will then fill out the provided online form with the required information. If new policy exists within the system the customer will select the company from the provided drop down list. Customer will then select “submit” and will be presented with a notification page that displays the changes the customer made.  An administrator will be able to edit/modify any customer’s policy information at any time. Administrator will access the customer database. Then after selecting the desired customer will select “edit policy” fill out the provided form with required changes. Administrator will the select submit and be presented with a notification that the changes were made and the system has been updated. |
| Input | User credentials upon login  Appropriate changes to policy information  Submission of changes. |
| Output | Notification from the System that the insurance has been updated and changes have been made. |
| Pre-conditions | The user is already registered with the system, and his or her account is active.  Previous policy information already exists in the system |
| Post Conditions | The submitted policy changes are stored in the system  User is notified that the changes have been accepted. |
| Frequency of Use | Only when an update to current insurance information is required. |

**Modules Billing and Billing History**

|  |  |
| --- | --- |
| Use Case ID: B1 | Submit a Payment |
| Description | There are two types of users that can access this module in billing. Both have access to submit a payment. Customer can use their account number to pay. Admins can also submit a payment on behalf of customer. |
| Input | * Account Number * First Name * Last Name * Credit Card Number /or Selected Payment Plan. |
| Output | If transaction is successful the systems will display a confirmation message. |
| Preconditions | The Account Number does not exist in the system. |
| Post Condition | If succeeded, the balance for the account will reduced by the amount of the payment. |
| Action | The system will then update it’s database. |

|  |  |
| --- | --- |
| Use Case ID: B2 | Set Payment Plan |
| Description | The Customer will have access to this module. The system will give them the option to pay for the invoice manually or automatically pay monthly, every 3 months, and annually. |
| Input | * Account Number * Invoice Number * Payment Plan Selection |
| Output | If transaction is successful the systems will display a confirmation message. |
| Preconditions | The Account Number does not exist in the system. The Plan does not apply to policy. |
| Post Condition | If succeeded, the plan will be set for the invoice. |
| Action | The system will remember the user’s selection for billing module. |

|  |  |
| --- | --- |
| Use Case ID: B3 | View Invoice for all policies |
| Description | The Customer will have access to this module. They can find all open invoices for their policies on account. |
| Input | * Policy Selection * Invoice Selection |
| Output | If transaction is successful the systems will display all the policies and all the current invoices that apply. Users then can select an invoice to view data on it. |
| Preconditions | The Account does not have existing policies in the system. |
| Post Condition | If succeeded, the nothing will change. |
| Action | The system will be the same. |

|  |  |
| --- | --- |
| Use Case ID: B4 | Update Invoice |
| Description | Only Admins can update invoice. They can change the invoice for their clients. They can add or remove data from the invoice. |
| Input | * Account Number * Invoice ID * Data |
| Output | If transaction is successful the systems will display a confirmation message. |
| Preconditions | The Account Number does not exist in the system. Invoice ID does not exist. |
| Post Condition | If succeeded, the invoice’s data will be changed to the new update. |
| Action | The system will update the invoice data. |

|  |  |
| --- | --- |
| Use Case ID: B5 | Search an Invoice |
| Description | Both user and admin types have access to viewing invoices. This includes current and completed invoices. |
| Input | * Invoice Number * Data Range |
| Output | Data for the searched Invoice. |
| Preconditions | The Invoice Number does not exist in the system. No invoices are within data range. |
| Post Condition | If succeeded, user will see the information. |
| Action | The system will remain same. |

|  |  |
| --- | --- |
| Use Case ID: B6 | View payments done |
| Description | Both user and admin types have access to viewing payments done. |
| Input | * Account Number * Date range |
| Output | If is successful It will list the more recently paid first payments to older payments within the data range. |
| Preconditions | The Account Number does not exist in the system. No invoices are within data range. |
| Post Condition | If succeeded, the payments will be shown. |
| Action | The system will remain same. |

|  |  |
| --- | --- |
| Use Case ID: B7 | View Balance |
| Description | Both user and admin types have access to viewing payments down. |
| Input | * Account Number |
| Output | If transaction is successful the systems will display a confirmation message. |
| Preconditions | The Account Number does not exist in the system. |
| Post Condition | If succeeded, the balance for the account will be displayed. |
| Action | The system will be same. |

**Modules Contact and Support**

|  |  |
| --- | --- |
| Use Case ID:1 | Post a question |
| Description | A user has the ability to post a question at any time by filling and submitting a form and a thread will be created. The only precaution required is a simple number/letter code to be type by user to reduce spam. If entered incorrectly system will refresh with error message on incorrect input. |
| Input | * First Name * Last Name * Question * Security code |
| Output | If successful new thread will be created with users question, if failed it will refresh with error message for missing or incorrect input. |
| Preconditions | User is registered and logged in. |
| Post Conditions | If successful thread with question will be created. |
| Action | System validates:   * All fields have values |

|  |  |
| --- | --- |
| Use Case ID:2 | View previous communication |
| Description | Both user and admin have the ability to view previous communication with each other. Only requirement here is that user is logged in, with one click system will display all previous questions and answers between the user and admin. |
| Input | * No input necessary |
| Output | User should see all previous communication with organization. Admin should be able to see history as well as coworkers answers. |
| Preconditions | User has logged in. |
| Post Conditions | User and admin will see all previous communication. |
| Action | System retrieves:   * Previous communication made by user and admin. |

|  |  |
| --- | --- |
| Use Case ID:3 | Answers to Questions |
| Description | The administrator has the ability to view Questions submitted by customers. Admin can then answer the question he or she submitted. |
| Input | * Admin Name * Admin Answer |
| Output | Answer to customer question. |
| Preconditions | Logged in as administrator. |
| Post Conditions | No post Conditions. |
| Action | System Displays:   * Creates a response to a question submitted by user. |

**Support Module**

|  |  |
| --- | --- |
| Use Case ID:1 | Create thread for Question or comment |
| Description | User will have the ability to create a thread. User must fill required information in order to create a thread as well enter correct code used to reduce spam. If information is entered correctly thread is created, if not error message on input not valid. |
| Input | * Subject of thread * Question or comment in box |
| Output | If information is valid, and code is correct thread is created, other message on input invalid. |
| Preconditions | Logged in as user |
| Post Conditions | User redirected to his or her thread. |
| Action | System validates:   * Input is valid * System creates and displays thread |

|  |  |
| --- | --- |
| Use Case ID:2 | Contribute to posts |
| Description | Both customer and Admin may contribute to existing thread by posting a comment. Simply by hitting user or admin can comment and reply on whole thread or simply on a specific post.  To register with the system, a user must fill in a list of required fields before submitting the form. If the transaction is successful the user account will be created within the system. |
| Input | * Comment in comment box |
| Output | System displays comment on bottom if replying to thread, or right under post if replying to specific comment. |
| Preconditions | User/Admin logged in |
| Post Conditions | No post conditions |
| Action | System displays:   * User or admin fills comment box hit post and comment will display |

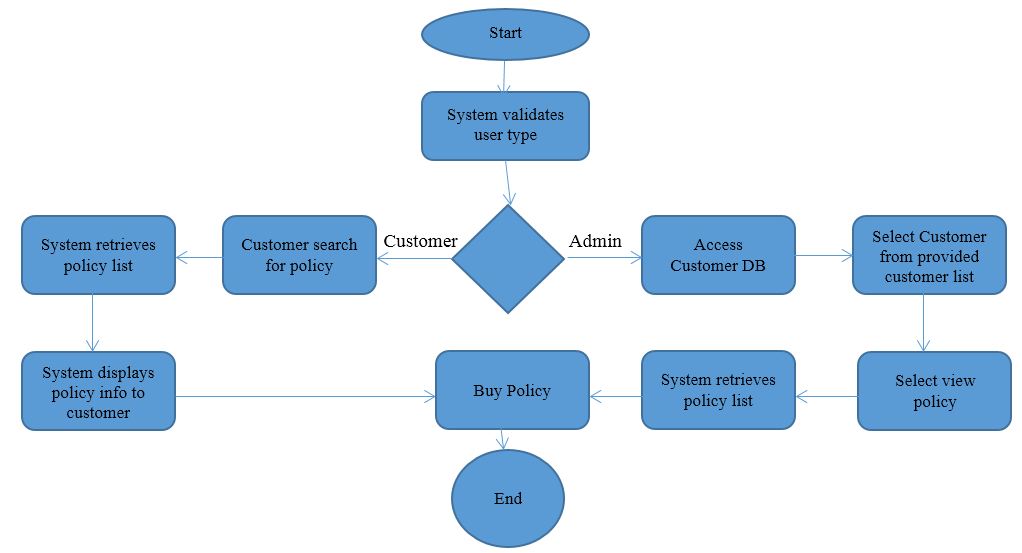
|  |  |
| --- | --- |
| Use Case ID:3 | Modify threads |
| Description | Admins have the power to control, monitor and modify thread at will. Editing if they see necessary, and deleting threads deemed inappropriate. |
| Input | * No input necessary. |
| Output | Message saying ”Successfully modified” |
| Preconditions | Logged in as Administrator |
| Post Conditions | System will display abilities that allow admin to modify threads. |
| Action | System performs operation:   * Based on chosen command, system will perform ability specified. |

**3.4 System models**

**Module Research/Buy Policies**

Use Case: 1

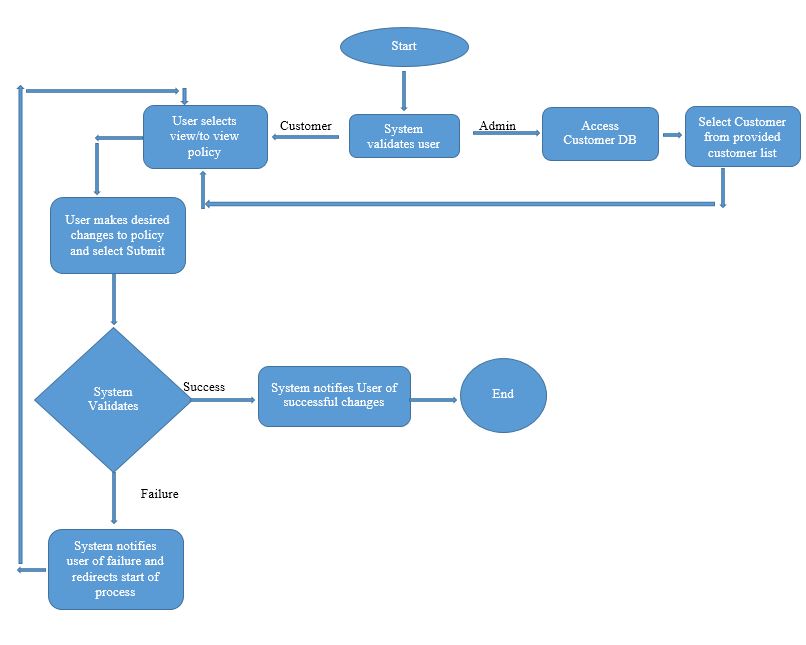
The diagram below shows the process of viewing customer policy pertaining to level of access. Customers will be able to search for a policy and buy it. Administrators have the capability to view policies that available for a customer. The sequence diagram below illustrates the process of viewing patient insurance. It shows the processes of validating the type of user that is attempted to retrieve the insurance information. Once the validation process is completed and the system determines the desired information exists it will display the information to the user. If no information exists then no information will be displayed.



**Module Manage My Policy.**

Use Case: 2

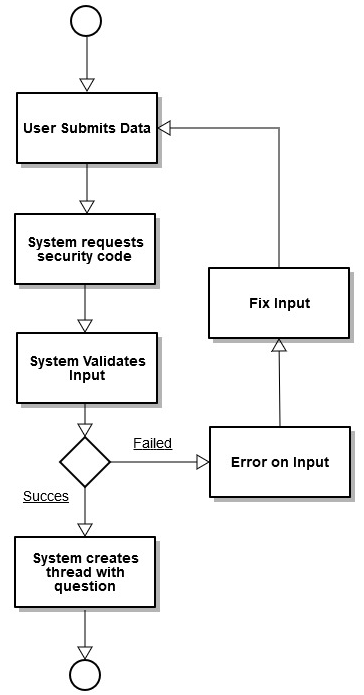
The System model below shows the process of managing customer policy. Once the system validates the type of user and the level of access assigned to them they will be able to continue the process of editing their personal policy or edit any current customer’s policy (Admins). While a user is currently viewing their policy they will have the option to edit/modify their policy deductibles. The process is then completed in very much the same way as the initial addition of policy. Once the customer/Admin has completed the process they will submit the changes and will either receive a success message, or an error message. If an error message is received they will have the option to try again. The process for Admin is identical in completion. However, the Admin process does differ in that staff will have access to the customer database and will have the ability to change any customer’s current policy as well as update any billing information, and deductions regarding insurance claims.



**Activity Diagrams**

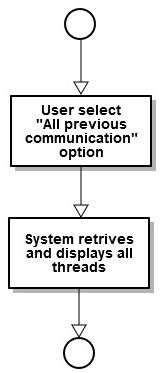
*Contact Module*

Use case # 1 “Post a question”



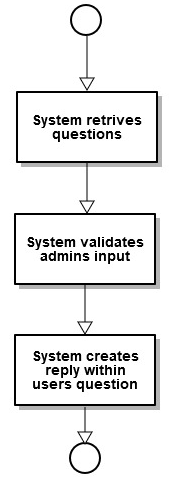
The use case # 1 of the contact module describes the process of posting a question. User will submit all the required information, followed by a security used to limit spam, system will then attempt to validate input, if data is incorrect, and error message will return on the part of the input were error occurred, the user can then fix errors and the process will restart. If the data was in the correct format the system will create a thread with question.

Use case # 2 “View previous communication”



The use case # 2 of the contact module describes the process of viewing previous communication. This is a simple use case requiring no input. User or Admin must simply click “All previous communication” and the system will retrieve and display all threads.

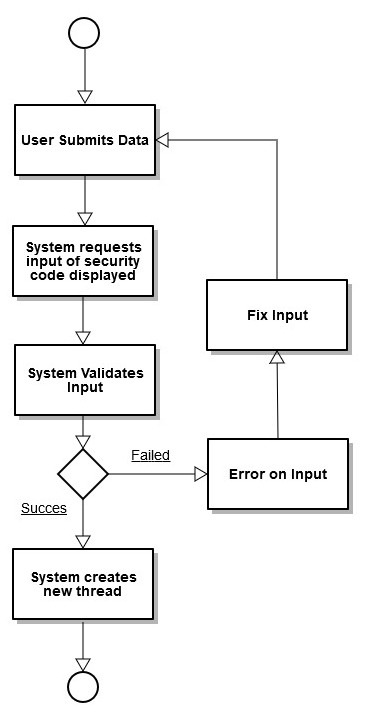
Use case # 3 “Answer Questions”



The use case # 3 of the contact module describes the process of answering questions. The admin has the ability to view previous questions asked by customers. The system will then attempt to validate the admins input, the system will then create a reply within a user’s question.

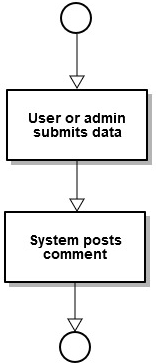
**Support Module**

Use case # 1 “Create thread for question or comment”



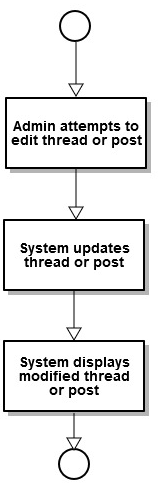
The use case # 1 of the support module describes the process of creating a thread for a question or comment. Unlike the contact module, the user is able to post a thread with their question or comments in a forum. By doing so this create a community where not only can admins answer questions but other user can contribute to posts. In this diagrams user submits the data, followed by a security code, system attempts to validate input, if fails it will display error on incorrect input, user can attempt to fix input and go through the process again, if the validation was correct the first time, the system creates a new thread.

Use case # 2 “Contribute to posts”



The use case # 2 of the support module describes the process of contributing to existing threads by making posts. A user or admin may at any point make a post, the users data will be submitted with required fields, the system then will post the comment.

Use case # 3 “Modify threads”



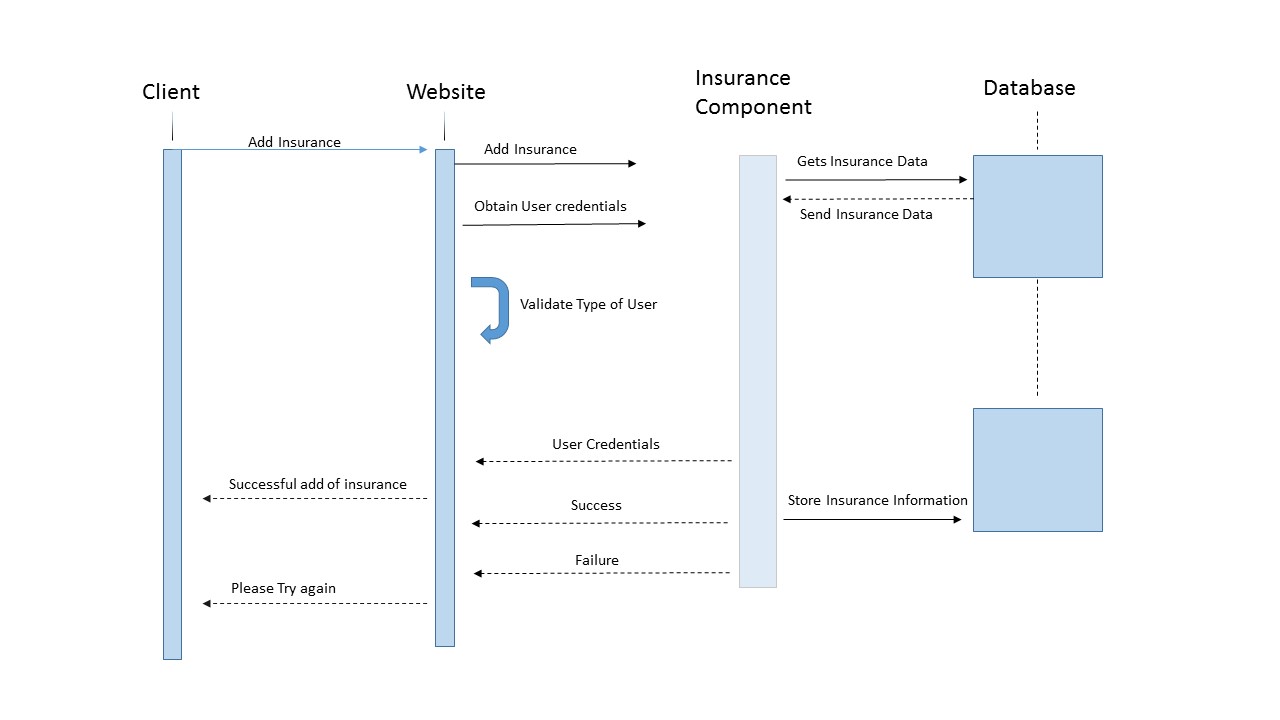
The use case # 3 of the support module describes the process of Modifying threads. The admins have the power to control the flow of the threads in the forum. They are granted several abilities such as the power to edit or delete existing threads or posts.

**Sequence Diagrams**

***Module Research/Buy Policies***

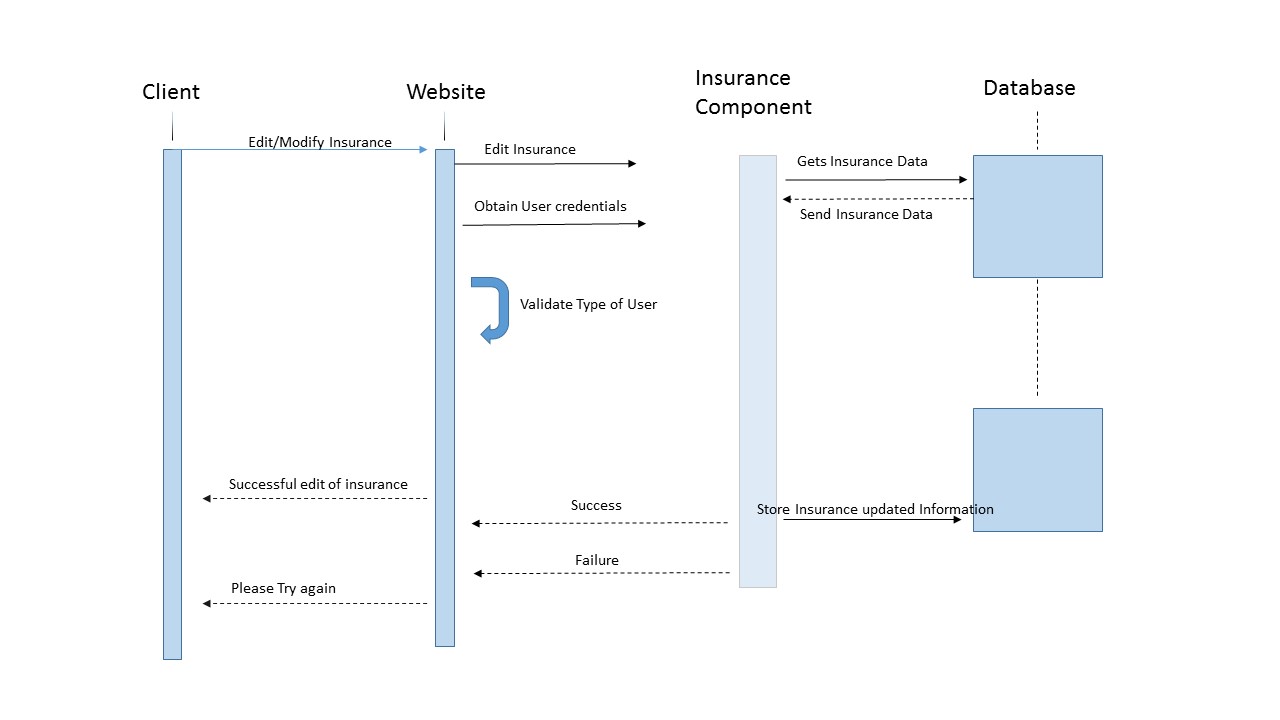
Use Case: 1

The Sequence Diagram below illustrates the process of validating which type of user is attempting to buy the policy information (customer or administrator) and the process of successfully storing the new policy information. If the system successfully adds the new insurance the user will receive a message notifying the success of the process. If process is not successful the user will receive notification that the process failed and to try again.



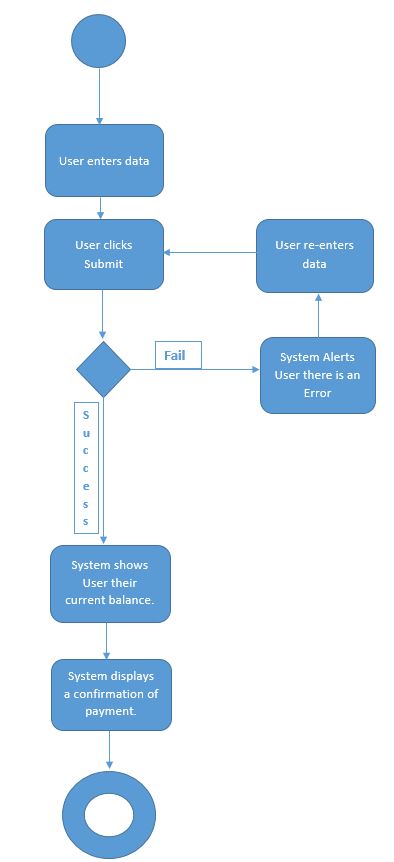
***Manage my Policy:***

Use Case: 2

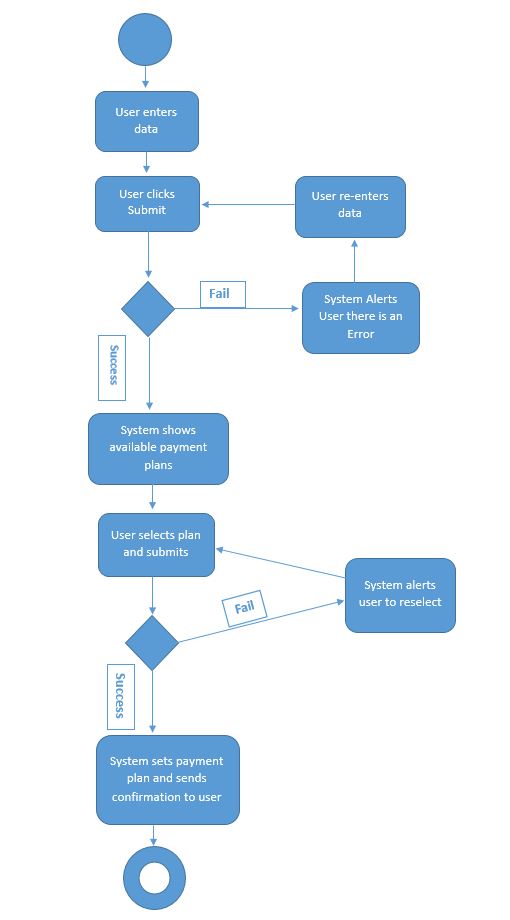
The sequence diagram below demonstrates the process as it accesses the various levels of the website all the way down to the database layer. The steps in this process are almost identical to the “Buy policy” process with the main difference being the edit, and the submission of the edit. Once the edit is completed the user will be notified of the success of the process. If the process fails the user will repeat the edit process.

***Billing Module***

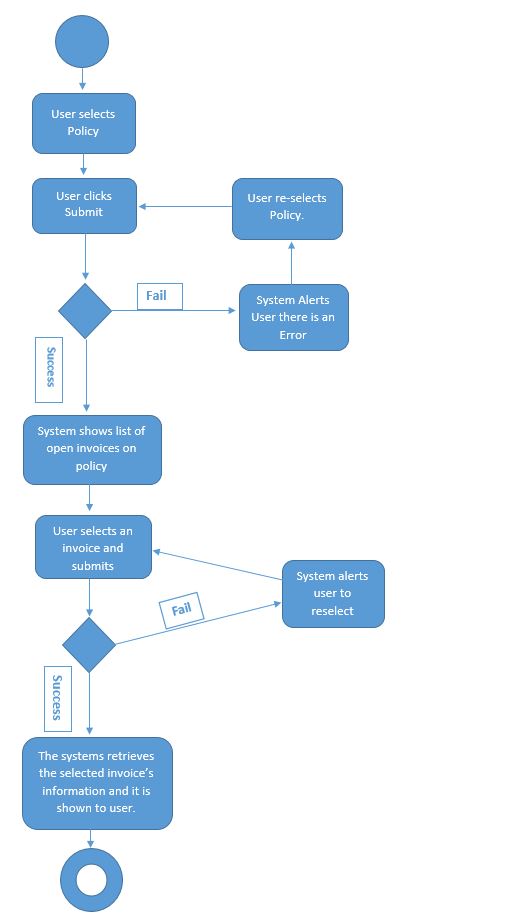
Use case: Submit a Payment



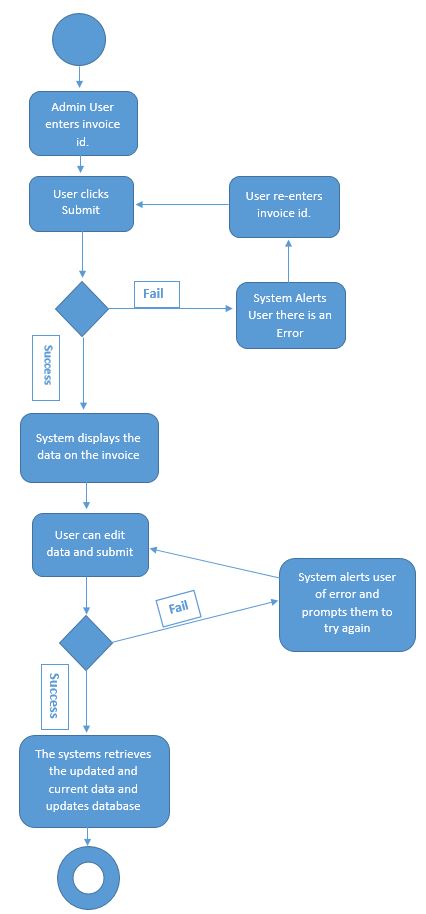
*Use case: Set Payment Plan*



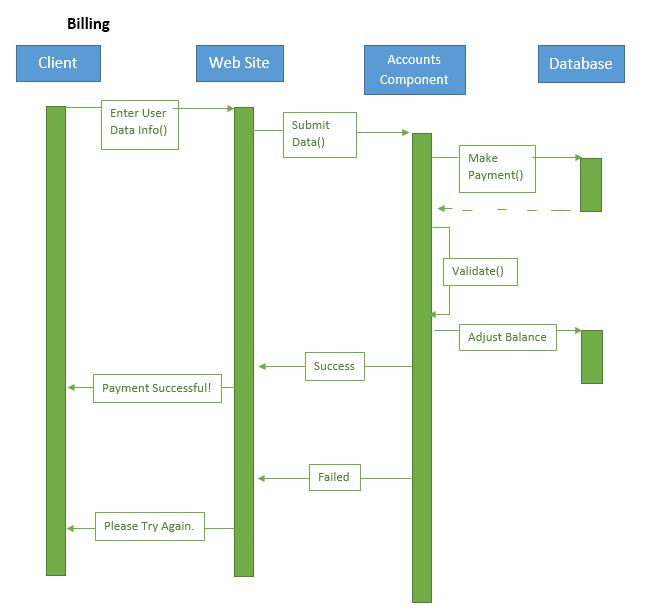
*Use case: View Invoice*



*Use case: Update Invoice*

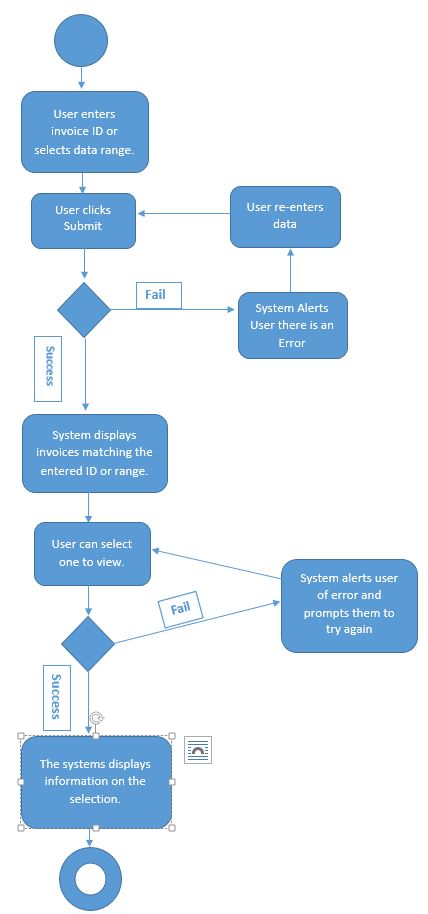


*Sequence Diagram: Billing*

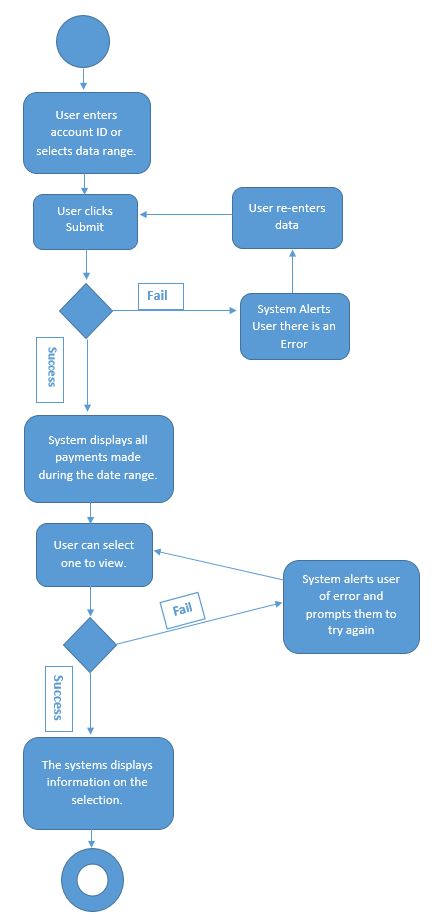


***Billing History Module***

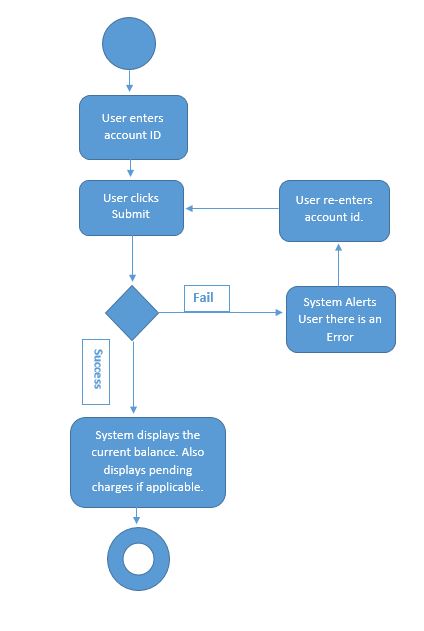
Use case: Search an Invoice



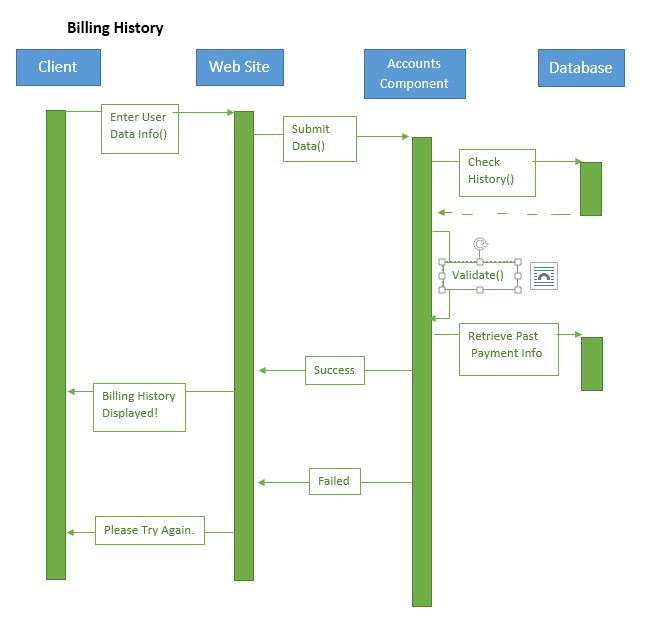
Use case: View Payments Done



Use case: View Balance

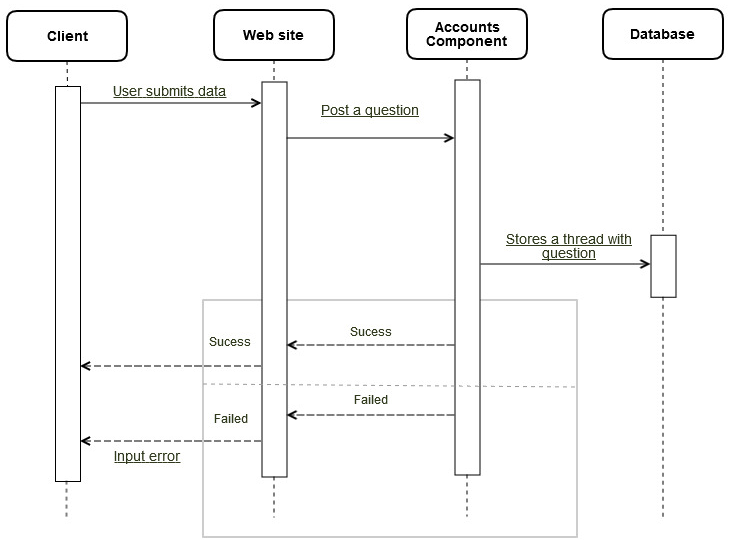


Sequence Diagram: Billing History



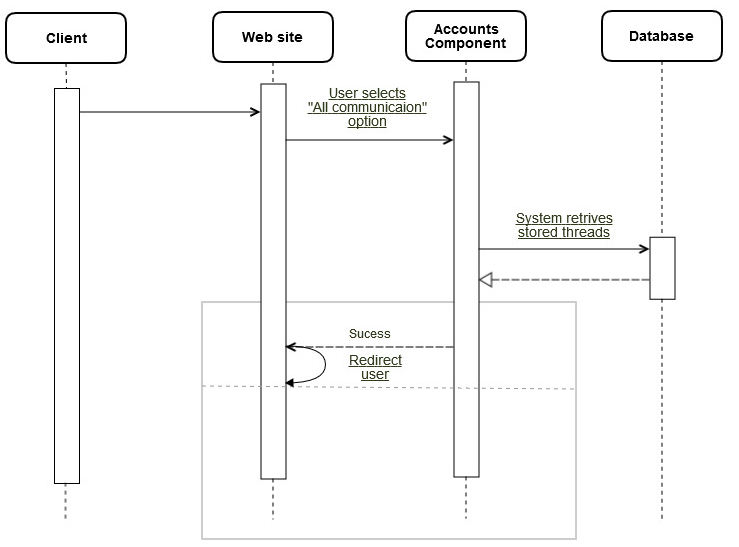
**Contact Module**

Use case # 1 “Post a question”



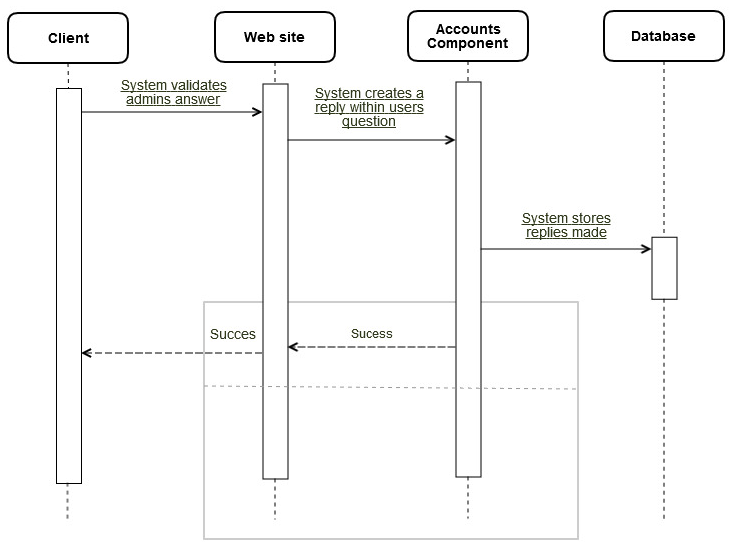
This sequence diagram shows pretty much what the activity diagrams shows. The user submits data to website it could be successful, or it can fail with an input error message returned. If successful the user will attempt to post question which can be a success or a failure at the accounts component. If successful the thread will the question will be store in the database.

Use case # 2 “View previous communication”



This sequence diagram shows the view previous communication of the contact module. Here no input is required, the user must only select to view “All previous communication” link from the website. The system will then retrieve and displays all previous threads from the database. User will then be redirected to the appropriate page.

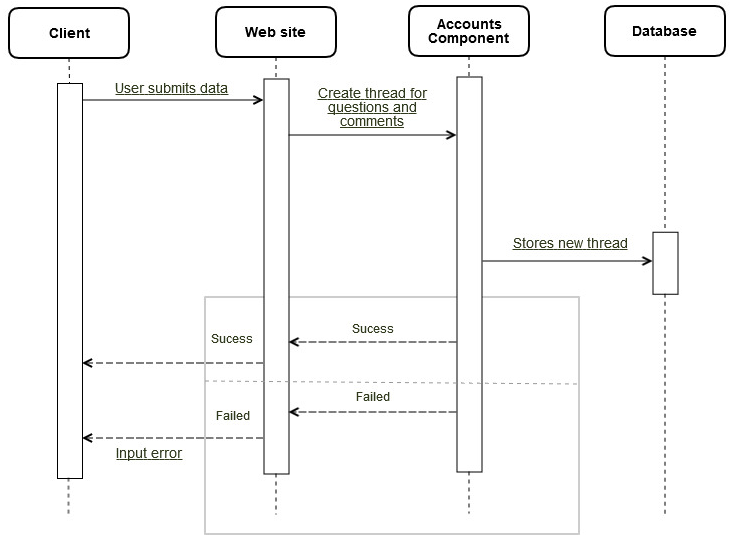
Use case # 3 “Answer Questions”



This sequence diagram shows the answer questions use case of the contact module. The system will validate admins input at clien level. There is no check for failure. The system creates a replay within the user’s questions. The system will then store the replies made.

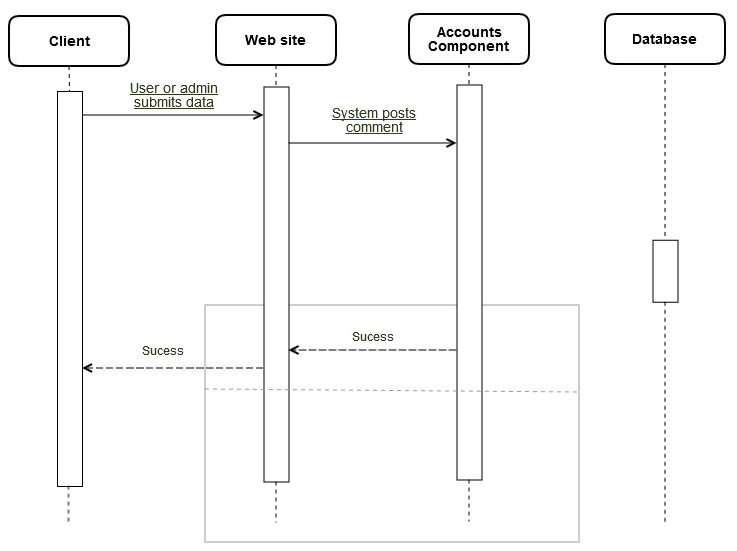
**Support Module**

Use case # 1 “Create thread for question”



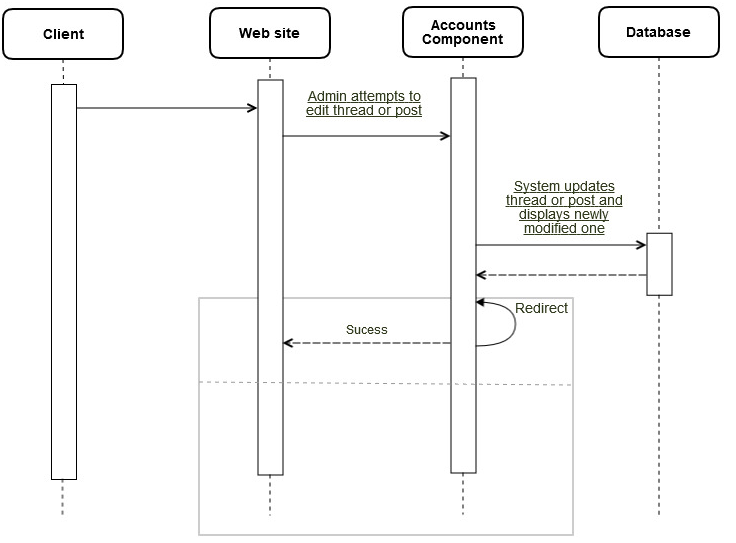
This sequence diagram shows create a thread for questions or comments in the support section. User submits data, if failed message on input error will be returned. If successful user may process to create a thread for questions or comments on website level. New thread is then stores in the database,

Use case # 2 “Contribute to posts”



This sequence diagram shows Contribute to posts use case of the support module. The user or admit submits data. No failure checks are done in this use case. The system posts the the comments at the website level. Nothing is stored in the database.

Use case # 3 “Modify threads”



This sequence diagram shows the use case which grants the admins the ability to modify threads, this use case belongs to the support module. No input is necessary here, admin will attempt to edit threads or posts at website level. It will always be successful. System will then update the thread or post and display the newly modified one, user will be redirected to the appropriate page.

## Domain Analysis

Main entities for contact module



**User**/**Customer**:

This entity represents the customer/user, since the only other entity is the admin. It has a list of the abilities the customer/user for the contact module such as the ability to ask questions as well as view previous posts.

**Administrator**:

Main entities for support module



**User**/**Customer**:

This entity represents the customer/user, since the only other entity is the admin. It has a list of the abilities the customer/user for the contact module such as the ability to create thread for questions or comments in a forum as well as making posts.

**Administrator**:

This entity is in charge kettering to the customer/user needs, making their lives easier. The admins can make post, they also have the power to modify threads at will, with the ability to edit as well as delete inappropriate threads or posts.

# Non-Functional Requirements

**4.1. Performance Requirements**

***Maintainability***

1. Every program module must be assessed for maintainability according to procedure xx. 70% must obtain “highly maintainable” and none “poor”. Production of a simple change into the web site, should take no more than 50 seconds.
2. The cyclomatic complexity of code must not exceed 7. No method in any object may exceed 200 lines of code.
3. Installation of a new version shall leave all database contents and all personal settings unchanged.
4. The product shall provide facilities for tracing any database field to places where it is used.

**4.2. Dependability Requirements**

***Reliability***

1. Resistance to failure.
2. Ability to perform a required function under stated condition for a specified period of a time.
3. The web site defect rate shall be less than 1 failure per 1000 hours of operation.
4. No more than 1 per 1000000 transactions shall result in a failure requiring a system restart.

***Availability***

1. The system shall meet or exceed 99.99% uptime.
2. The system shall not be unavailable more than 1 hour per 1000 hours of operation. Less than 20 seconds shall be needed to restart the system after a failure 95% of the time

**4.3. Other Quality Attributes**

***Security***

1. The application shall identify all of its client applications before allowing them to use its capabilities.
2. The application shall ensure that the name of the employee in the official human resource and payroll databases exactly match the name printed on the employee’s social security card.
3. At least 99% of instruction shall be detected within 10 seconds.

***Portability***

1. No more than 5% of the system implementation shall be specific to the operating system.
2. The meantime needed to replace the current Relational Database System with another Relational Database System shall not exceed 2 hours. No data loss should ensue.

# 5.0. Appendices

# Farmers Insurance:Quote & Buy Online. (n.d.). Retrieved February 18, 2015, from <http://www.farmers.com/>

Allstate Insurance (n.d.). Retrieved February 18, 2015, from <http://www.allstate.com/‎>

# 6.0. Index

<<What pages are important key terms located>>