Team B: 007 Insurance Management System Final Report

Scrum Master: Izzy Austin

Jonathan Courts, Adam Crozier, Connor Gallun,

Maddie Neely, Luka Popovic, Blake Robbins

SE 361: Introduction to Software Engineering

Dr. Panos Linos

April 21, 2022

Table of Contents

Abstract

The 007 Insurance application is an insurance claim database management system designed for SE 361: Intro to Software Engineering. 007 Insurance was created using the C# language running on the Microsoft .NET framework in Visual Studios. The database backend was constructed in SQL and offloaded in the cloud through AWS. The application divides users into four groups: client, financial manager, claim manager, and administrator. Each role has its own kind of needs and the application addresses these needs accordingly. A more detailed description of user story functionality is listed on the next page and throughout the report.

Chapter 1: Introduction

Team 007 was tasked with creating an application that can cover a wide range of insurance company and management processes, including providing relevant information to the company to support seamless and effective decision making for users. As mentioned in our abstract, the users of this application are divided into four groups – clients, administrators, financial managers, and claim managers – who all have varying needs in our application. Through a user-friendly interface, clients should be able to file claims, upload documents, communicate with their financial and claim managers, and download claim reports. Furthermore, administrators should be able to view user details, organize client information, register users, and grant user privileges. Claims managers should be able to view their clients' profiles, access client documents, validate and transfer files, and communicate with their clients, other managers, and administrators. And, lastly, financial managers should be able to calculate client estimates, validate and transfer client amount details, view their clients' profiles, and communicate with their clients, other managers, and administrators.

Before getting into the background of our project's creation, let's introduce the team members and their roles in this project. Izzy Austin was the Scrum Master and worked mostly on the graphic user interface (GUI), or frontend, of the project. Maddie Neely worked on the frontend as well as the database side, or backend, of the project. She was our full-stack member. Connor Gallun worked on the backend. Jonathan Courts worked on the frontend. Blake Robbins worked on the backend. And Luka Popovic worked on the frontend as well as some of the

business logic. When we say someone worked on the frontend or backend, however, we mean that the majority of their input was on that side of the project, but it is fair to say everyone worked on both sides of the project at one point or another.

At the beginning of each two-week Sprint, the team would create goals based on the natural progression of the project. After everyone agrees on the goals for the Sprint, we would go through and choose user stories to accompany our goals. Next, we would break up our goals into tasks and start working. In the next few chapters of this report, we will be detailing what our project Sprints looked like, what we concluded and learned through this project, and how successful we were as a team.

Chapter 2: Project Sprints

As previously mentioned, we had six two-week Sprints throughout this project. Each Sprint had the same structure consisting of a planning meeting where we discussed our product backlog and goals for the next two weeks, daily standup meetings where each team member discussed what they were working on and the obstacles they encountered, and retrospectives where we discussed what we did well and what we wanted to do better. In the following paragraphs, we will detail what each of those processes looked like.

Our team's Sprint planning meetings looked the same for every Sprint.We would start by creating a Google Doc, shared amongst all team members, and write down our tentative goals for the current sprint – which we established at the end of the previous Sprint, – which user stories/functional requirements would make the most sense to complete given the application's progression, a concrete goal list, and an optional goal list dependent on the completion of our concrete goals. From there, we would separate every concrete goal into step-by-step tasks and assign members to each task. This was usually done by the Scrum Master who assigned members to tasks that either are a progression of something they were already working on or a task that fell into their range of skills. Team members, of course, were able to assign themselves to tasks as well. After the tasks were created and assigned to members, we would add them to our Trello board and begin working.

As for our Product Backlog, which was a list of all of the project's functional requirements, we created a shared Google Doc between the team members in which we

organized them into sections based on the user stories' topic. For example, some of our topics included Login, Messaging, and Profile user stories. Once they were organized, all user stories were broken down into frontend tasks and backend tasks. We did this in order to help us decide which user stories naturally came next given the progression of our application. From this list, the team would choose a few user stories to complete every Sprint and carry over the uncompleted stories from the last Sprint. From there, we would check off user stories/functional requirements as we finished them.

Similar to our Sprint planning meetings, our team's daily standups would look the same every Sprint. At the beginning of each of our meetings, we would discuss each member's progress. This included what they were currently working on, what obstacles they ran into, and what they planned to work on next. During each member's analysis, others would chime in if they felt they were able to solve one of their problems or gave them feedback on a way to do something better/more efficiently.

In addition to our daily standups, the tools we used during our Sprints really improved our efficiency as a team. We used the following tools during the course of our entire project. Trello was used for project tracking and assigning members to tasks. We created a new board for each Sprint. Slack was used for team communication such as sharing links. We found that using a SMS group chat also improved efficient communication. Visio was a great tool that helped us map out and visualize our application. Lastly, we used Github to track the changes made throughout the course of our project.

During the creation of this project, our team used the Scrum format which consisted of

six two-week “Sprints”. In our first Sprint, Sprint 0, we worked out the basics of our team. This was done through establishing a Scrum Master and team name, selecting our project – Insurance Claim Management System, – and discussing our strengths as team members. Additionally, we set up elements to track our process including a Trello board and Slack group. Lastly, we set up and reviewed the tools needed to be successful in this class (i.e. reviewing the Scrum documents and installing Visio and Visual Studio).

Now that we had all the tools necessary for success, we began our project. During Sprint 1, we created our login process and homepage prototype as well as set up our database and created our loginCredentials table to store login information. We also delineated our team’s definition of “done.” In Sprint 2, we implemented log out functionality and created the registration process for new users. The next Sprint consisted of creating the claim filing process for clients as well as different homepages for different users. We chose to accomplish this task in Sprint 3 so we could start implementing and testing the admins ability to grant user privileges. In Sprint 4 we continued implementing the claim filing process. We also started implementing the ability for financial managers, or FMs, and claim managers, or CMs, to view client profiles. Additionally, we started working on the user interface for messaging as well as its functionality. During our last Sprint, Sprint 5, we refined our current processes as well as implemented new processes like messaging, downloading claim reports, transferring documents, and the Equitable Claim Database Management, or ECDM. The EDCM is a way for our application to assign clients FMs and CMs equitably.

Furthermore, at the end of each Sprint we would discuss the things that worked for us, the things that didn't work for us, and how we would proceed. The following is a list of things that worked for us in order of most helpful to least helpful. We think one of the biggest attributes of our success was our shared Google Drive. This helped communication, in terms of what we were working on, flow seamlessly, especially since we were all on different schedules. We also found the Trello board extremely helpful because, not only could you see all the tasks we were currently working on, but you could also see who was working on them. It made it much easier to consult with team members on any given task. The SMS group chat also increased our efficiency in communication. The SMS chat started as more of a notification system for when messages were sent in Slack, but, over time, it ended up replacing Slack, more or less, and Slack was only used as a place for sharing links between team members. Github was obviously very useful in terms of knowing who worked on what and when, however, it caused some problems and required a big learning curve.

As previously mentioned, the biggest thing that didn't work for us was the group chat in Slack. We found that the notification system for Slack, in regards to the team member's phones, wasn't the best. Some members would not get notifications about messages and we were not sure if messages had been seen by all members. Obviously, this became an issue that needed to be solved immediately; therefore, we created the SMS group chat to ensure that group members would see important messages in a timely manner.

Chapter 3: Conclusions and Lessons Learned

Applying an iterative and agile development approach to this project shaped and changed all of our team member's views about software engineering in general. During the college curriculum, there is not much opportunity to work with a team on a project for a long duration of time. This class allowed us to experience this and is the closest experience we will get to a real world development process. It also changed what a lot of us expected from software engineering. Before coming into this class, a lot of us expected to not be able to do much without the previous "step" of the project being done, but through using Scrum, we realized that this was not the case and it felt like we were able to get more done. Once we got started, using Scrum also made it less of a daunting task to get the project done and meet all of the requirements that were asked by the product owner.

After working on this project for the entire semester, our experience with following an object-oriented approach was generally very positive. When first starting, we did not follow the object-oriented design approach very closely. We would put code wherever we needed it without much thought or care, as long as the code functioned like the owner wanted it to. As the class progressed, we all learned more about object-oriented design and applied it to all existing and new code. By doing this, it was not only easier to read the program and work with it, but it was also easier to figure out where things were. An example of this is using database handlers to manage the data uploading and retrieval. At first, we would put the code wherever it was needed (i.e. on a button event), but after adding the class, we could reuse the code. Additionally, if an

issue occurred, then we knew it was in one section of the code rather than each individual section where the search or upload might have occurred.

The object-oriented design also allowed for easier debugging and problem solving. With the object oriented approach, it was very easy to roughly see what was going wrong (i.e. being able to tell if we were reading data from a list box wrong versus not uploading it properly). If we did not apply this design approach, it would have been much more difficult to test and figure out what part of the code was giving us issues.

Throughout the entirety of this project, our team encountered a lot of challenges, mistakes, successes, failures, and outcomes that were sometimes favorable, and sometimes not. A major challenge we had through the entire development process were personal ones that would occasionally make it difficult to work on the project. If a member had a difficult week due to personal reasons, the rest of the team were understanding and would work on things that they were intending on working on.

The biggest challenge that we faced as a team was the accidental reversion of our project to one that was a week old. In order to fix this, we first tried to revert back to the most recent version of the project through Git control. This proved to be difficult as we were all still very unfamiliar with Github and how to do tasks outside of fetching, pulling, and pushing. In order to fix this issue, the team created a second Github repository and copied the most recent code (that we had on a member's laptop) into that repository. This allowed us to keep the code that was lost and only had to give up an hour's worth of work.

Chapter 4: Team organization and roles

Overall, it's fair to say that we all did a little bit of everything. While specific team members usually stayed in the sector that they were most skilled in (GUI/database), we all helped, where we could, to solve problems and implement error checking. With that being said, the following procedure was consistent throughout the duration of our application's creation. First, Izzy and/or Jonathan would create the GUI of a form. Next, Maddie and the form's creator would work on the business logic. Adam and Connor would, then, work on getting the form connected to the database, if needed. Luka would work on the respective user stories' specificities, like search by date or listbox functionality, and Blake would add the form to our diagram(s).

For example, this was what happened during the creation of our login process. Izzy created a homepage GUI prototype while Jonathan created the login GUI. Maddie, then, connected the login page to the homepage. The backend group, Adam, Connor, and Blake, worked on getting it connected to the database and Luka worked on testing and debugging. The following paragraphs detail every team member and their respective contribution to the project throughout its creation.

Izzy Austin:

Izzy was the Scrum Master for this group. As the Scrum Master, she spearheaded group communication and organization, such as creating the Google Drive, which was used for sharing project information, Sprint feedback, and other necessary documentation, throughout the project.

Aside from her role as Scrum Master, Izzy mostly worked on the frontend of the project. This included creating GUI forms, and implementing their respective business logic. The aforementioned GUIs includes, but is not limited to, the individual user homepages, profile page, settings page, claim report form, and the pages used to view user/client information (i.e. viewUsers and viewProfiles). Izzy also worked on the implementation of email address error checking and functionality that allows the user to download claim reports along with Connor. Lastly, she also did the frontend and backend of the forgot password process and security questions, as instructed by the Product Owner, but as the project progressed the team decided it wasn't worth continuing in the final stretch since it was not a functional requirement.

Maddie Neely:

Maddie primarily worked on connecting the GUI frontend to the database backend through the creation of classes and database management classes to handle all of the necessary data. This began with her and Connor working to implement the user login and registration logic. She, then, worked closely with Izzy to make the user profile and settings forms functional. Next, she worked with Connor to implement the necessary business logic to allow for users to see their claims as well as to allow for claim managers to view the claims that were assigned to them. Furthermore, Maddie also worked closely with Jonathan to make the messaging system fully functional. She created the database, database handler, and message class and implemented the logic to ensure the appropriate messages appeared depending on who was logged in. Overall, Maddie acted as a full stack developer, helping with GUI when necessary as was the case with the Profile and Profile Edit form, and working with the database when tables needed to be

created or edited, but she primarily focused on connecting the frontend to the backend to make all of the features truly functional.

Jonathan Courts:

Primarily, Jonathan worked on producing the GUI for the lots of the forms and constructing the applications layout. Starting off with the login/registration, these were the first forms that were created for the application. We based the entire frontend of the application based on the original blue and cyan theme that was used in the login form. Within the application there is a sidebar that allows the user to navigate the application which was made by Jonathan as well. This sidebar is the core feature within the application that is a part of each form. The business logic functionality to navigate between forms was also done while building the GUI part. As for other forms, Jonathan worked on the messages form, with Maddie, to fully build out that feature with Jonathan primarily working on the frontend and Maddie building the backend. The scheduling form was completely done by Jonathan, front and back end. This was more of a side project to better understand C# that also was applicable to the project, and, hence, implemented.

Connor Gallun:

Connor was responsible for backend work on the insurance system. At the start, his primary focus was setting up the database as well as figuring out a handful of the database handler classes that would access and manage the data from forms or objects. Specifically, these were the database handlers for the user profiles and login credentials, some work on the claim information database handler, and an abstract database handler class that was used with all of the database handlers. Connor implemented a method to store files a user would upload and, along

with Izzy, download files and images that were related to the claim, as well as the ability to send extra files to others, such as the claim manager. Connor also worked on equitable claims distribution, as well as assisting in debugging and adding error checking for incorrect data or missing data when it was attempted to be accessed, such as a case where the user did not upload a document and the claim manager would try to download it.

Blake Robbins:

Primarily working on the backend, Blake worked on details such as the organization of tables/data as well as some of the implementation involved. With the organization involved, this quickly developed into the creation of the UML Diagrams. The Use Case diagrams were created at the beginning in attempts to better visualize the user stories. Each of these highlighting the necessary capabilities for each user: client, claim manager, finance manager and administrator. Class Diagrams then became a large focus for the rest of the semester. There was an issue with the automatic generation of the Class Diagrams so most of the connections and sort had to be created by hand. The Class Diagram then continued to grow throughout the semester, with the addition of some new classes and such resulting in a frequent update necessary. Sequence Diagrams were finally created to guide the users through the GUI as well as each of the functional requirements for each user typer.

Adam Crozier:

Adam served the team on the backend and assisted the team by finding a way to streamline the creation of database objects in the Microsoft .NET framework. While other handlers ended up being implemented in the final project, this technique helped make the profile

carousel, which inspired the design format for the View Profiles and View Users pages. Adam also helped to establish the early ins and outs for the backend team, specifically table creation in SQL and connections for the Amazon Web Services (AWS) database. Adam coined the phrase for the Equitable Claim Distribution Management System, which assigns claims to Claim Managers based on need in a queue. Connor fleshed out a ton of this out on the backend. Adam also assisted Blake with the creation of the Class and UML diagrams. Additionally, Adam planned on implementing a robust financial planning tool for FMs, which had to be placed on the back burner for more important features like messaging and filing a claim.

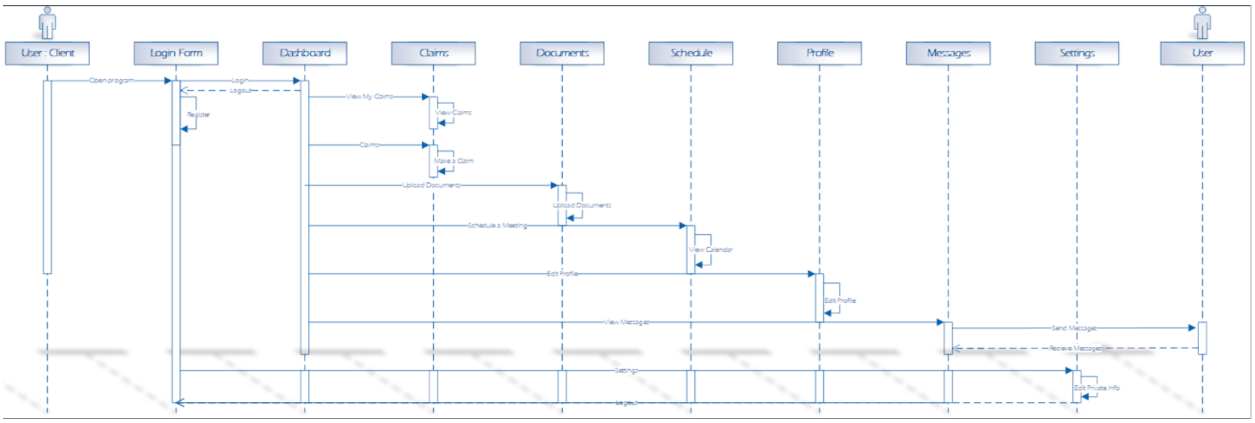
Luka Popovic:

To better organize the group, Luka sorted the user stories into categories that related with other user stories. This was done to target a certain functionality among all roles for each Sprint. Luka, then, worked with the database side of the project and implemented a visual for certain roles to see the contents of certain databases. He began with the functionality to alter user roles. This required the user to view the database containing all of the other users and select a user before retyping the new desired role. This continued by restricting this privilege to only the administrators. Luka, then, continued to make this more intuitive by creating a visual that allowed the administrator to view a list of users to choose from. He took this idea and implemented it in the same fashion for other user roles. For clients, they were able to browse their filed claims similar to the administrator's ability to select from a list of users. For financial managers and claims managers, their lists contained only the clients that they were working with. This was partly backend and partly frontend as the visual had to be read from the databases.

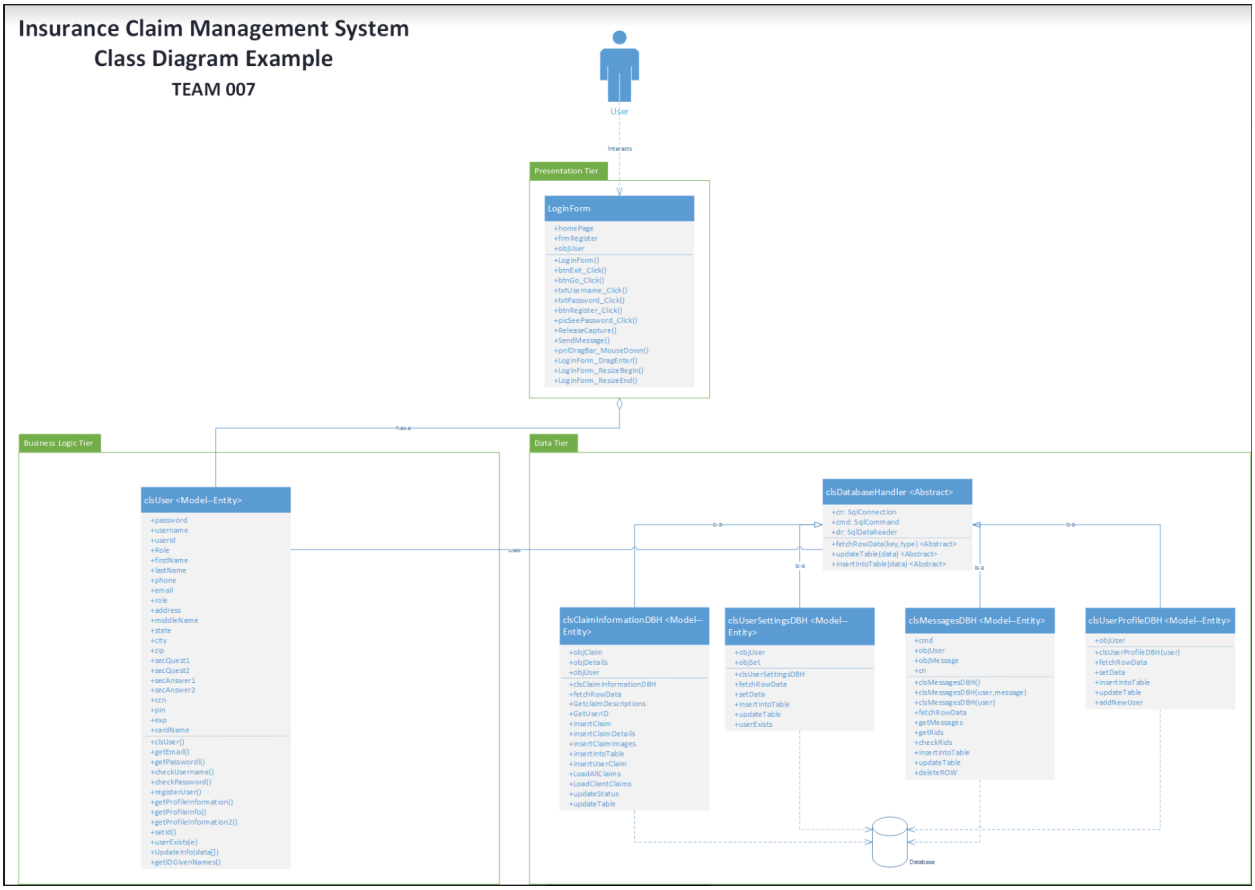
Then, Luka set out to give the administrator an additional privilege that let them sort through claims by date. This consisted of adding a date selection variable on the GUI and altering the code behind that to simply let administrators search through claims on the selected date. Towards the end of the project, Luka aimed to create a way for administrators to create folders and sort users into said folders. Unfortunately, time and code complications prevented this functionality from meeting the team's definition of done within the Sprint 5 timeline.

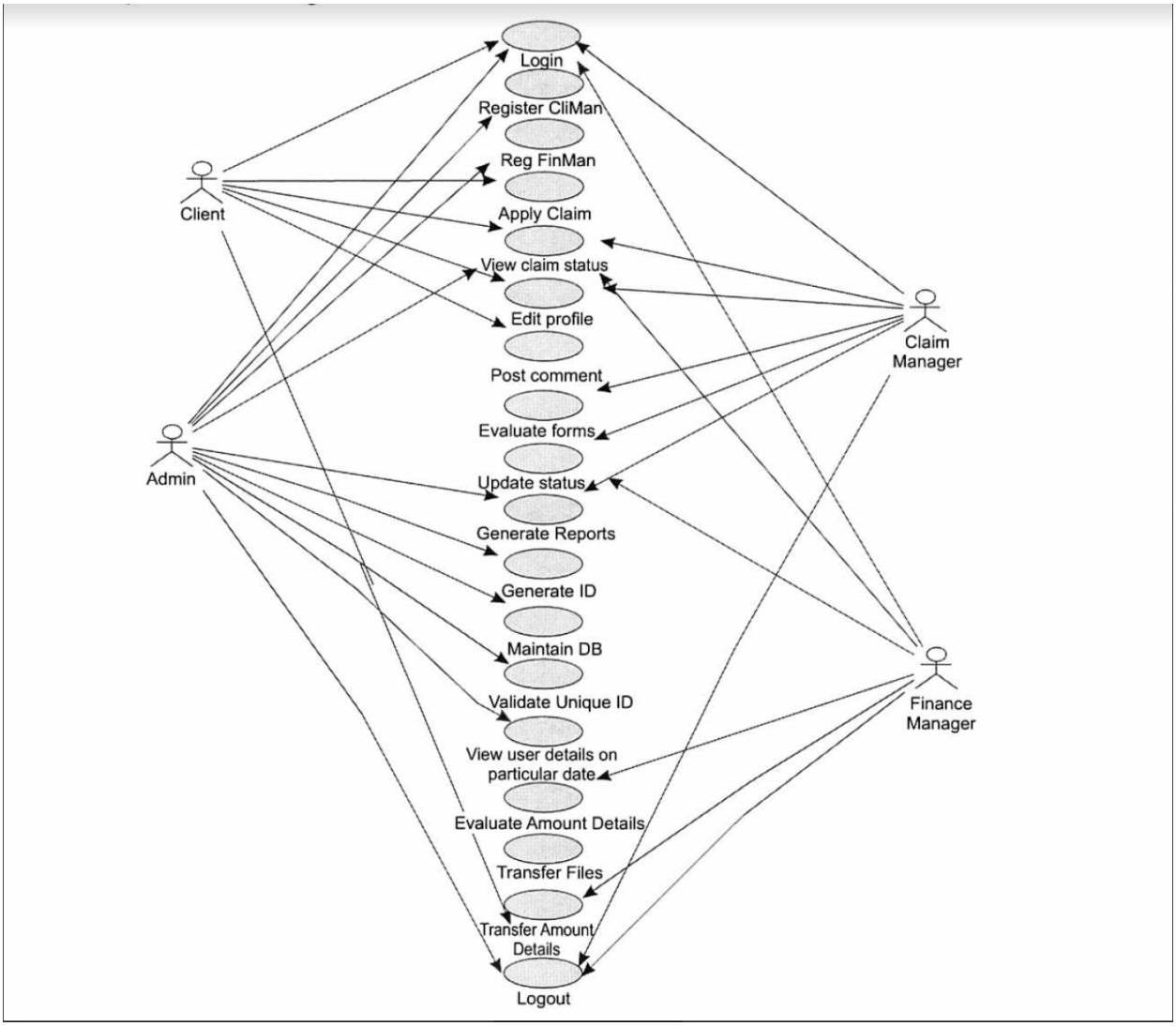
Diagrams

Sequence Diagram - Client



3-Tier Breakdown




Login
Register CliMan
Reg FinMan
Apply Claim
View claim status
Edit profile
Post comment
Evaluate forms
Update status
Generate Reports
Generate ID
Maintain DB
Validate Unique ID
View user details on particular date
Evaluate Amount Details
Transfer Files
Transfer Amount Details
Logout
Client
Admin
Claim Manager
Finance Manager

Appendices

Sprint 0 Presentation

Sprint 1 Presentation

Sprint 2 Presentation

Sprint 3 Presentation

Sprint 4 Presentation

Sprint 5 Presentation

Sprint 0 Peer Evaluation

Sprint 1 Peer Evaluation

Sprint 2 Peer Evaluation

Sprint 3 Peer Evaluation

Sprint 4 Peer Evaluation