**9-2 Final Project Submission: Software Design of a Healthcare Insurance Web-based Application System**

Allyssa Evergreen

Department of Information Technology, Southern New Hampshire University

IT-632-Q2724 Software Design & Modeling 20TW2

Dr. Harry Omoregie, Ph.D.

February 7, 2021

**Healthcare Insurance Web-based Application – System Overview**

Modern technology has expanded to improve the lives of individuals and organizations across the globe. Technologies specializing in information, communication, and individual participation have especially increased and improved everyday life in the healthcare industry. For instance, a 2019 study analyzing participation in rehabilitation methods proved that the usage of information and communication technologies improved the patients’ rates of participation and perceptive quality of the support (Zonneveld, Asaba, & Guidetti, 2019). While there are many approaches to leveraging technology in this industry, this report will detail the software design of a web-based application that will allow users to review, select, and enroll in healthcare insurance plans.

**The Purpose**

A web-based application that allows the user to select and enroll in healthcare plans empower the individual to make a benefit plan selection that is right for them. For instance, Human Resources (HR) Specialists explain benefit plans typically in-person during open enrollment season to employees yearly. By providing an alternate method to learn about the healthcare plans that are offered and making their selection on a 24/7 accessible, web-based application improves the productivity of the HR role by allowing HR Specialists to focus on other end-of-the-year projects for their organization. This also benefits organizations financially by closing the distance between organizations using this technology with their stakeholders (i.e., customers, employees) and helps add business value to their company (Riggins, 2009). Completing the software design of a web-based application that leverages the ability for the end-user to review, select, and process that selection also allows for building a software template and design for other purposes, especially if the application in the healthcare industry continues to produce positive metrics and results. For instance, the web-based application could be extended to projects such as having an online menu ordering system or choosing a photo package that is right for the end-user, allowing the foundation of the software design to become a template for other business ventures.

**End-users**

There are several potential end-users for a healthcare insurance exchange web-based application. As stated earlier, employees who are participating in open enrollment at their companies may utilize the application to review the benefit options provided by employers for the incoming year. The web-based application could detail plans’ summaries, carrier information, premiums, expected payments (i.e., co-pays, monthly premiums), and delineate any contingencies before enrolling into the plans that will best benefit the employee, their budget, and their situation. Once they select an eligible plan that fits in their budget, they then may enroll in the plan with little to no assistance from HR Specialists. Another end-user that benefits in this scenario would also be the HR staff. While they may also enroll in the same way as their employees, they could also use the information submitted on the web application to accurately assess items such as which plans were more favorable with their employees this incoming year and gather data for their YTD reporting and statistics. The end-users are also not limited to just employees or HR staff. This web-based application is also useful in reaching a larger audience. An effective example of this is the implementation of the Affordable Healthcare Program, which allows individuals across the United States to review health plans in their database called the Marketplace, and select/enroll into a plan once individuals decide what is best for them and their budget in the comfort of their own home (USA.gov, 2020).

**Organizations**

The organizations that would benefit from this web-based application are insurance carriers, employers, and government programs. In 2019, there were almost 6000 insurance companies in the United States alone (Insurance Information Institute, 2020). Many insurance companies are finding creative ways to utilize technology to expand their consumer reach and presence in the insurance market. Providing a way for users to review, select, and enroll in medical insurance plans in the comfort of their own home will provide a more relatable alternative and shopping method in the market. As discussed, employers and organizations may also use the web-based application to allow employees to enroll in benefit plans when they are first hired or during open enrollment. This reduces the amount of paperwork and time spent on obtaining all enrollment information for employees, and more time on other HR-related tasks. Another type of organization that benefits from the usage of applying and enrolling in plans online are government programs. Government programs could leverage this technology to expand their footprint and reach to demographics and populations that may not have known about the government programs, which allows these programs to provide more affordable healthcare options than privatized companies. This list of organizations may also expand if the software design of the web-based application is used as a template for other uses (i.e., food ordering system, package system, surveys, online newsletters).

**Operating Environment**

The web-based software application must contain a work area that displays content within a web browser that can operate from a desktop computer setup or mobile device. The web design must be accessible across different browsers, such as Google Chrome, Microsoft Edge, Mozilla Firefox, and MacIntosh Safari. The site must also be easy-to-read in the desktop version or mobile device view within these browsers. Once users are provided the website link or QR code (which will direct them to the site) and create an account, the user must be able to navigate through the web-based application with little to no assistance; navigation buttons, instructions, and usability must be easy to understand and is ergonomic and natural to the user. The software requirement for this software design must also provide a secure database when collecting benefit information and transmit it securely into a local area network (LAN) for decryption before storing in a cloud environment’s secure and private storage, which is then only accessible to an authorized database administrator and/or pre-authorized individuals such as HR Specialists who need access to the information (Anderson, 2015).   
**Conclusion**

The usage of a web-based application that provides the ability to select and enroll in healthcare insurance plans will empower users to have visibility in the benefits they choose, while also benefiting the organizations they reside in. For instance, HR Specialists were known to present benefit and carrier plans before open enrollment periods to deliver important carrier changes, benefit plan information, and how it will affect their payroll. The usage of information systems as a tool alleviates these in-person demands on HR specialists by placing these responsibilities and actions on their employees, thus freeing their schedule and employee user satisfaction (Shibly, 2011). The application will be designed to be versatile across different browsers, while also providing security features such as LAN decryption and pre-authorization to access the data collected. The software design will also be created as a template for future business ventures outside of just enrolling in healthcare insurance plans. The software design will demonstrate the usability, functionality, and reliability of the web-based application. In the next section, we will analyze the software design and the use cases that support it.

**Web-Based Health Insurance Exchange - Software Analysis and Use Cases**

The analysis and design of a software system must have an outline or plan of intention on how the system will interact with intended users of the system. Identifying the actors and illustrating their interactions and relationships through methods such as UML use case diagrams and use case scenarios are an excellent way to visualize and explain the business processes and workflows that the system should perform. Functional requirements should also be outlined to demonstrate what the system should be able to accomplish, as well as its constraints to ensure efficiency and accuracy. These items are delineated for the health insurance exchange on a web-based application scenario and help form the software design and architecture of the system.

**Identifying the Actors**

The ability to utilize a web-based application to select, manage, and enroll in health care insurance exchange insurance plans outlines involves different actors that will interact with the system in at least one form. Actors may be separated and categorized as a primary actor, a supporting actor, and an offstage actor. The primary actor represents the main user that will interact with the system and whose goals are primarily satisfied, the supporting actor represents an entity that performs a service within the system, whereas the offstage actor has vested interest in the system and the behavior of the use cases, but does not have direct interaction (Larman, 2005, p. 66).

Here are the actors within the health care insurance exchange system:

* **Primary Actors**
  + ***Employee/User.*** These reflect the users that do not have high-level permissions to access and manipulate the benefits. These users are interacting with the software system to only select, enroll, and update their benefit plans.
  + ***HR Representatives.*** HR Representatives are the individuals with higher-level permissions that will be able to review company employees or a group of users’ benefits. They will also be responsible for reviewing and approving benefits.
* **Supporting Actors**
  + ***Health care insurance server***. This actor represents the software system’s server. This actor supports the actions of the primary actors and will respond based on these requests/actions. For instance, when a primary actor updates a benefit plan, the supporting actor of “health care insurance server” will update the system to reflect that instance.
* **Offstage Actors**
  + ***Health care insurance carriers***. These represent the health insurance carriers that will not have access to the system, nor make, select, or enroll in the benefit plans. However, they have a vested interest in the approved benefits and receive reports (with the permission of the HR Representatives) to upload into their system.

**UML Use Case Diagrams and Use Case Scenarios**

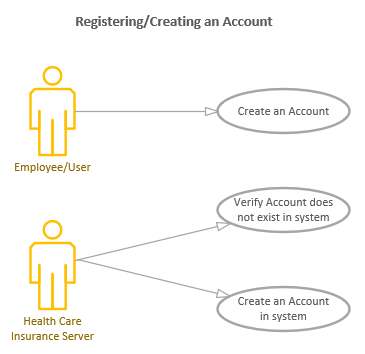
UML use case diagrams are excellent ways to simplify the relationships between actors and interactions within the system. The UML use case diagram also helps illustrate workflows and business processes to visually present these interactions (Larman, 2005, p. 92). While there are many different potential interactions within the health care insurance exchange web-based application system, here are six different use case diagrams illustrating and explaining the intended software design of this system. To further strengthen the impact and comprehension of the software system design, use case scenarios are also included to help further explain these interactions.

**Use Case #1**

* **Use case name:** Registering/Creating an Account.
* **Actors:**
  + ***Primary*:** Employee/User.
  + ***Supporting*:** Health care insurance server.
* **Description of the functionality:** The user will follow the appropriate steps to register an account within the health care insurance website. The server will take these steps, translate the data, and then create an account on the website for the user.
* **Precondition:** The user should not have an existing account on the website.
* **Assumptions:** The user has access to the internet and the health care insurance website. The user does not have an existing account on the website and follows the appropriate steps to register an account.
* **Flow of events:** The user opens the website in an internet browser. The user clicks on the “register” link. The user follows the pre-distinguished registration steps. The user registers and creates an account for current and future use.
* **Postconditions/goals:** The success scenario would be that the user has an active, referenceable, and accessible user account for current and future use with 24/7 website access (if the website is functioning and live).
* **Alternative Paths:** The failure scenario would be that the account could not be registered due to not accessing the server properly or the user not following the registration steps appropriately. Another alternative path – if the assumptions are not followed – is that the user would already have an account created on the website and cannot complete the registration steps for a new account due to a pre-existing account.

**Figure 1.1**

UML Use Case Scenario – Registering/Creating an Account

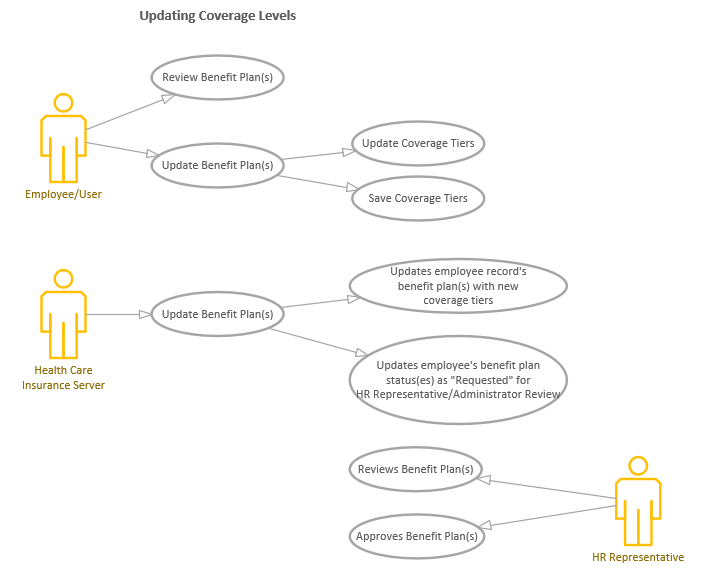


**Use Case #2**

* **Use case name:** Updating Coverage Levels.
* **Actors:**
  + ***Primary*:** Employee/User, HR Representative.
  + ***Supporting*:** Health care insurance server.
* **Description of the functionality:** The user will log into the web-based application, review the benefit plan, click “update” to change the coverage level of their health plan, and then click “save” to solidify the request. The system will then reflect the updated coverage level for that user’s specific health plan. The system will also update the benefit plan in the “requested” status for HR Representative or Administrator review for approval.
* **Precondition:** There is a benefit plan with changeable coverage tiers in the system.
* **Assumptions:** The user has access to the internet and the health care insurance website. The user already has an existing account within the web-based application that is active and accessible. There are also no limitations to changing the coverage tiers during the time the user is accessing the system (for instance, the user must be within the open enrollment period to make modifications. Otherwise, special circumstances must be taken into consideration – like qualifying events – to make necessary changes to the coverage tiers).
* **Flow of events:** The user logins into the web-based application. The user reviews their benefit plans and selects the benefit plan with changeable coverage tiers that they would like to modify. They select the desired coverage tier and save the coverage tier. The system then updates the coverage tier to reflect the user’s new coverage tier and updates the benefit plan status as “Requested” for HR Representative or Administrator review.
* **Postconditions/goals:** After the user selects the new coverage tier, the benefit plan reflects the updated coverage tier for current and future use.
* **Alternative Paths:** The coverage tier did not update and generates an error with a reason to the user (for instance, if they are accessing the benefit plan outside of open enrollment – which prevents any changes being made unless they have qualifying event changes).

**Figure 1.2**

UML Use Case Scenario – Updating Coverage Levels

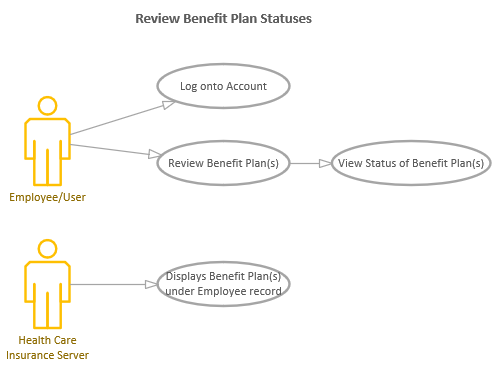


**Use Case #3**

* **Use case name:** Reviewing Benefit Plan Statuses
* **Actors:**
  + ***Primary*:** Employee/User.
* **Description of the functionality:** The user logs onto the web-based application and reviews their benefits. The benefits will list all current benefits along with their status (i.e., approved, requested, in progress).
* **Precondition:** The user has benefit(s) in the system and is not terminated from their company (if the user is an employee).
* **Assumptions:** The user has access to the internet and the health care insurance website. The user already has an existing account within the web-based application that is active and accessible.
* **Flow of events:** The user will log into the web-based application. Under the “Current Benefits” tab, the user will select/click on the “Current Benefits” link and the system will return a table of the current benefits and its relevant information (such as coverage tiers, monthly premiums, and deductions as applicable).
* **Postconditions/goals:** The user can click on the “Current Benefits” link and a table with all the benefits they have requested are displayed with pertinent information such as status, monthly premiums, and deductions as applicable.
* **Alternative Paths:** If an employer would not like to show the employees’ requested benefits for any reason – and just want to show approved benefits— the employer could pre-select a condition to only display approved benefits, which would then trickle down to the employee user’s accessing the web-based application by only showing approved benefits and omitting requested or in progress benefits.

**Figure 1.3**

UML Use Case Scenario – Reviewing Benefit Plan Statuses

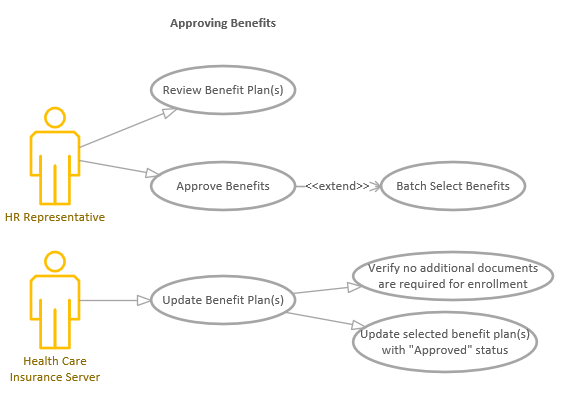


**Use Case #4**

* **Use case name:** Approving Benefits.
* **Actors:**
  + ***Primary*:** HR Representative.
  + ***Supporting*:** Health care insurance server.
* **Description of the functionality:** The HR Representatives may access the web-based application to review, batch select (select multiple employees/benefits at a time), and approve any requested or in progress benefits in the system submitted by employees.
* **Precondition:** The HR Representative must have permissions and access to view other employees’ benefits.
* **Assumptions:** The user has access to the internet and the health care insurance website. The user already has an existing account within the web-based application that is active and accessible.
* **Flow of events:** The HR Representative will log into the website and be able to see all current requested or in progress benefits by their company employees. The HR Representative can then batch select which benefits to approve. The system will approve the selected benefits.
* **Postconditions/goals:** The HR Representative can select specific benefits and approve them. The system must change the status of these benefits from requested or in progress to approve. User and HR Representative must see them as “approved”.
* **Alternative Paths:** Some benefits require an estimate of insurability (EOI) form to be submitted before approving the benefit. The system should not allow approval of a benefit that requires this form.

**Figure 1.4**

UML Use Case Scenario – Approving Benefits

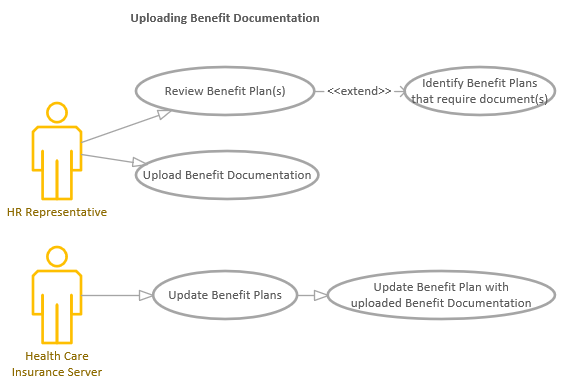


**Use Case #5**

* **Use case name:** Uploading Benefit Documentation.
* **Actors:**
  + ***Primary*:** Employee/User, HR Representatives.
  + ***Supporting*:** Health care insurance server.
  + ***Offstage*:** Health care insurance carriers.
* **Description of the functionality:** Users of the system may upload benefit documents for specific benefits that request this information.
* **Precondition:** The system can upload documents on each benefit plan.
* **Assumptions:** The user has access to the internet and the health care insurance website. The user already has an existing account within the web-based application that is active and accessible.
* **Flow of events:** The user – either the employee or HR Representative – log into the website. The user will then select “Current Benefits” and have an option to upload benefit documentation. A file upload window pops up for a file to be selected from their local computer and uploaded into the system.
* **Postconditions/goals:** The system reflects a readable, uncorrupted benefit document and attaches it to the specific and applicable benefit plan. A notification is sent to the HR Representative that the requested or in progress benefit now has a document for review.
* **Alternative Paths:** Not all benefit plans require benefit documentation to enroll in them. If the benefit plan was not designated as a benefit plan requiring documentation, the upload function should not appear or apply for these benefit plans. This will mitigate user errors where some benefit plans may be accidentally uploaded to an incorrect corresponding benefit plan.

**Figure 1.5**

UML Use Case Scenario – Uploading Benefit Documentation

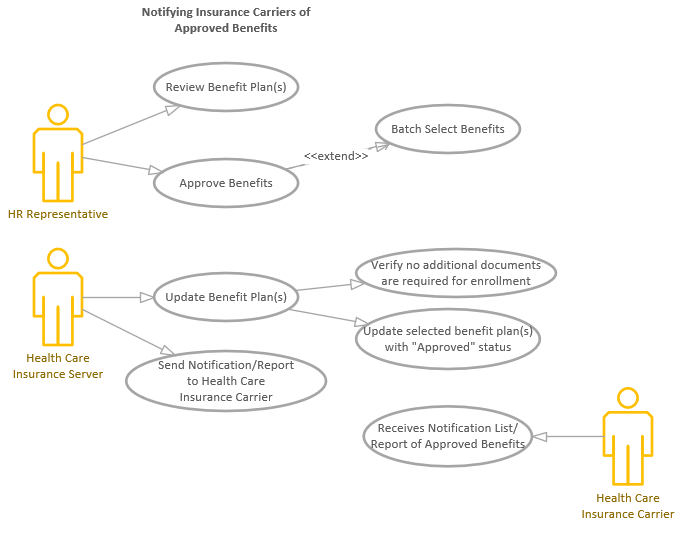


**Use Case #6**

* **Use case name:** Notifying Insurance Carriers of Approved Benefits.
* **Actors:**
  + ***Primary*:** HR Representative.
  + ***Supporting*:** Health care insurance server.
  + ***Offstage*:** Health care insurance carriers.
* **Description of the functionality:** To remove the additional step of HR Representatives reaching out to carriers themselves, the HR Representative will have the ability to create a report that lists all employees that are in approved benefits and send this information to the carrier with a click of a button after review.
* **Precondition:** There should be at least one employee benefit that is approved to send the health care insurance carrier(s) information.
* **Assumptions:** The user has access to the internet and the health care insurance website. The user already has an existing account within the web-based application that is active and accessible.
* **Flow of events:** The HR Representative will access the website and select/click on “Review all Current Benefits.” The HR Representative should be able to filter down this table to only approved benefits. Once everything is reviewed and contains all required information, the HR Representative will then select the “Notify the Carrier of Approved Benefits” button to send the report to their pre-designated carrier contact(s).
* **Postconditions/goals:** The HR Representative can view the table with approved benefits, review the information, and the carriers are successfully notified of all approved benefits.
* **Alternative Paths:** Due to safety and security concerns, HR Representatives may be able to toggle on and off the ability to automatically send notifications to the carriers once a benefit is in the approved benefit status. If this toggle is turned “off” or the HR Representative does not have permission to send the report, the option of “Notify the Carrier of Approved Benefits” should not be accessible and a manual report or electronic file will have to be submitted outside the system.

**Figure 1.6**

UML Use Case Scenario – Notifying Insurance Carriers of Approved Benefits



**Functional Requirements**

Requirements are defined as the conditions and capabilities that the system must conform to to ensure the system functions as intended (Larman, 2005, p. 54). For the health care insurance exchange web-based application, we will be utilizing a FURPS+ model to identify the different functional requirements to ensure the systems operates as intended. The FURPS+ model consists of functional, usability, reliability, performance, supportability, and the additional requirement of legality (Larman, 2005, pp. 56-57).

The FURPS+ model when applied to the Wild Woods Apartments database:

* **Functionality.** The system must remain functional and accessible from any computer or mobile device with access to an internet browser 24/7, 365 days a year.
* **Usability.** The system must be easy to understand and obtain an intuitive ergonomic design. For instance, each step in adding a benefit plan should make sense to the user and follow an appropriate logical design. Every single action should take no more than 1 minute of consistent and average internet connection speed for the user to update a single component of a benefit plan, refresh the website page, and see the benefit plan reflected as enrolled or updated appropriately.
* **Reliability.** HR Representatives must be able to review and utilize the web-based application at any time, especially during open enrollment periods.
* **Performance.** Multiple users – employees, regular users, and HR Representatives – upwards of 10,000 users at a time should be able to log into the system without any decrease in performance or usability.
* **Support.** The web-based application should be supported by the database administrator or company IT support monthly to ensure the system complies with International Standards of Organization and security compliances and certificates.
* **Legal.** Expanding on ensuring the appropriate compliances and certificates are maintained in the support section, the web-based application will continue to ensure that the appropriate licensures, authorization, and standards are upheld when utilizing the website.

**Developing the Architecture of the System**

The use cases and their associated diagrams may be used to design the overall system’s architecture, especially involving class diagrams and sequence diagrams. Class diagrams represent ideas, things, or objects within the system’s domain (Larman, 2005, p. 136). By utilizing commonalities between the different use case scenarios, as well as common nouns or phrases in the use cases, the conceptual classes are easy to determine. For instance, a common occurrence between the use cases is “benefits.” A conceptual class of “Benefits” can be created that include examples such as benefit plan names, benefit plan description, or benefit plan insurance type. The use case diagrams may also be used to develop the system sequence diagrams (SSDs). For instance, the use case diagram of “Registering/Creating an Account” could be used to exemplify the main topics of interactions in that process but an SSD will delineate exactly what the actor’s actions are and how the server or other actors will interact with that action. In an SSD for this use case scenario, the SSD would delineate the employee/user actors’ selections step-by-step in registering an account. After inputting an account log in, the other actor of “health care insurance server” will verify if that same log is present in the current server. If it is not, then it will send a message to the user to continue forward and produce additional fields for user input. Otherwise, it will terminate the registration request.

**Conclusion**

While use case scenarios and diagrams are simplified forms of the actor(s)’ interactions within the system, they propose helpful information to stakeholders or users of the system. Software developers can review these scenarios and diagrams to ensure their system accommodates the use case scenario appropriately and as efficiently as possible. Highlighting these interactions in diagrams or text first allows developers to ensure they are understanding system expectations before undergoing a very lengthy, meticulous time of writing code for these interactions, support efficient and agile methods. In the next section, we will evaluate the UML sequence diagrams for the system and understanding the system’s expectations and interactions.

**UML Sequence Diagrams for the Healthcare Insurance Web-based Exchange**

UML sequence diagrams are diagrams that represent the interactions between objects and actors and details their interactions (Visual Paradigm, 2020). Sequence diagrams help visualize the different parts that perform these functions within a system. They are also modeled after use case diagrams and use case scenarios.

**UML Sequence Diagrams**

For the healthcare insurance web-based exchange, we have two UML sequence diagrams to illustrate the flow of logic for how users may review and edit benefits, as well as how benefits are approved within the system. These represent important components of the system as it is required for users to be able to review, update, and approve their benefits accordingly. These steps also support some of the overall systematic functional requirements by ensuring users have access to review the benefits within the system at any time (functionality) and uphold the ability to empower users (such as employees) to elect in their benefits – and then allowing appropriate, authorized personnel to review (as applicable) and make the final approval (which represents a legal requirement). For both the UML Sequence Diagrams (as seen in Figure 1.1 and Figure 1.2), the flow logic begins with the actor of either User or HR Representative; the UML sequence diagrams represents one or the other to display the key differences on how the system should operate with a regular user without additional permissions and a user with authorized permissions (such as an HR Representative who must review all benefits). Regardless of the actor, the actor operates on the computer server that represents the computer interface that bridges communication between the actor and the database with the benefits data. The first steps of logging in and authenticating the process with the computer server are similar on both UML sequence diagrams.

**Figure 1.1**

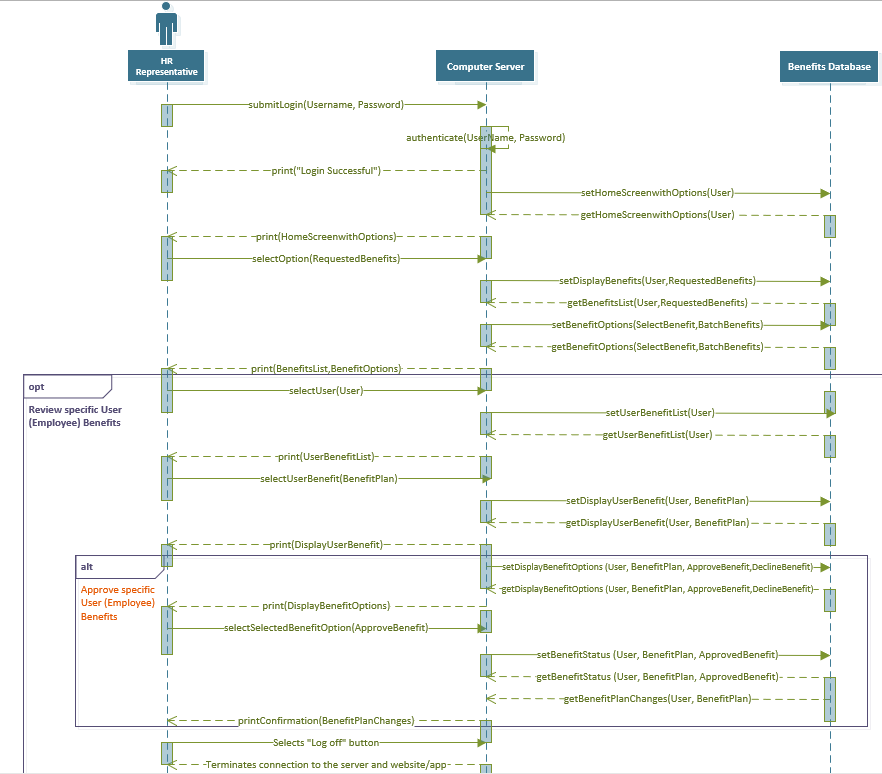
*UML Sequence Diagram of Reviewing and Updating Benefits*



*Note:* The UML Sequence Diagram represents how a user interacts with the computer server to log into their account to review and update their benefit’s coverage tier. While there are many different paths the system can operate on, this path focuses on reviewing the benefits and updating applicable coverage tiers on their benefit(s). The computer server also represents any computer interface, such as a website or mobile application; the computer server is used to authenticate login information, as well as send and receive requests from the database containing benefits data for the specific user. It is important to note that there are implied data limitations for a user without additional permissions. For instance, when the user selects the options to review their current benefits, the system will only display benefits that they have elected and underneath their user profile.

**Figure 1.2**

*UML Sequence Diagram of Approving Benefits*



*Note*: The UML sequence diagram represents the actor of “HR Representative.” The HR Representative represents the user with authority to view multiple users’ benefit(s) data. This is applicable if the healthcare insurance web-exchange is used as a third-party provider for companies to store, review, and retrieve pertinent benefits data for business operations. The HR Representative is responsible for reviewing and approving benefits. This UML sequence diagram examines the capability of the HR Representative to review multiple benefits user data underneath the “requested benefits” selection on the healthcare insurance exchange and then examines the specific interactions with approving specific benefit(s) that are listed as “requested” for each user. Additionally, the user’s permissions plays an important role (which is only implied in the sequence diagram and will later be described in the class diagram) where their user account will allow them access to view all other users’ benefits data in the system as long as it is underneath their permissions and company. The displayed home screen will also look different for those with additional permissions and viewing capability.

**UML Sequence Diagrams in the Software Development Cycle**

The process of creating UML sequence diagrams is an important part of the software development life cycle. While there is a variety of diagrams that help move forward the software development life cycle, UML sequence diagrams help develop the basic, essential building blocks of a model, the relationships, and interactions between objects, as well as provide a graphical representation to help show different perspectives and uses in the system (Waykar, 2013). The software development team may use UML sequence diagrams in combination with other diagrams to help visually document how the system should operate when interacting with objects. The usage of these sequence diagrams also ensures that the appropriate modeling languages are consistent and provide extensibility in the system’s operations. Most importantly, UML sequence diagrams help visualize the time sequence of when and how the objects participate (or are active) in the system, representing both the dimensions in a vertical sense (time) as well as a horizontal sense (through objects) (Waykar, 2013). However, it is a useful software development tool as it helps software developers’ understanding of how the different objects must be implemented and may be used to support definitions and protocols of tasks in the system (Baqais, 2018).

**Conclusion**

The UML sequence diagrams are helpful visual diagrams that further develops the understanding of the behaviors, dynamics, and interactions of actors and objects within the system. These time-sequenced diagrams may also be replicated for each intended, specific behavior, making it an effective tool for software developers to develop the software development life cycle. For the healthcare case insurance web exchange, UML sequence diagrams are a great way to model how users interact with the system to log in, review benefits, and making appropriate changes, such as updating coverage tiers or approving benefits. The strengths of the UML sequence diagram help outline and support the functional requirements of software development projects, such as the healthcare insurance web-based exchange. In the next section, we will be evaluating UML Class diagrams for the system, the characteristics, and methods for these objects.

**UML Class Diagrams for the Healthcare Insurance Web-based Exchange**

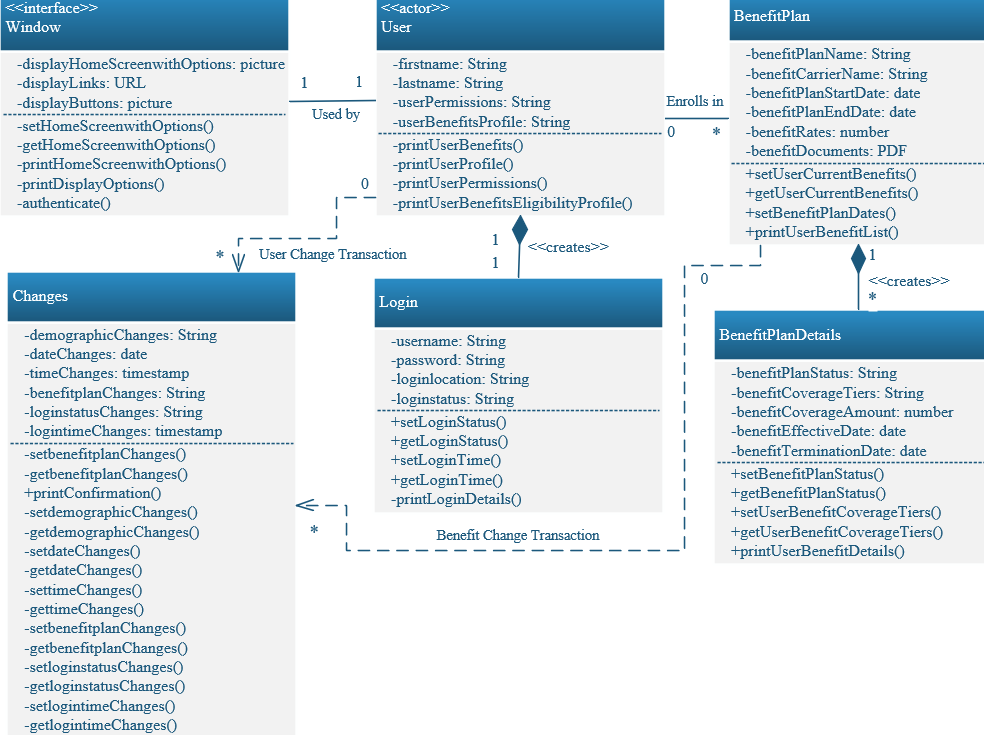
UML Class Diagrams are important tools for the software development team. The UML Class Diagrams are static models that help model the application’s design, delineate the responsibilities of the system, and provide an in-depth analysis of interfaces, objects, and actors within the system (Tutorialspoint, 2020). Class diagrams investigate and display relationships between the classes, applications, specifications regarding coding variables and their types, and methods that support the class in the overall system.

**UML Class Diagrams in the System**

For the healthcare insurance web-based exchange, we have six classes that operate within the system. These classes are broken out by the User (actor), Window (interface), Login, BenefitPlan, BenefitPlanDetails, and Changes classes (as seen in Figure 1.1) The User class represents all external individuals who are utilizing the system to enroll and change their benefits. The window represents the healthcare insurance exchange interface – either through a website on a web browser or mobile application – that users must use to access and utilize the functionalities of the system. It is also important to emphasize the composition of aggregation relationships within the system with the Login and BenefitPlanDetails class respectively. Composition aggregation represents a class that has exclusive ownership over another class, and when that one class is deleted, the rest of the classes connected in the composition aggregation are deleted (Larman, 2005, p. 264). In the UML class diagrams for this system, BenefitPlanDetails is exclusively owned by the BenefitPlan class, and the Login class is exclusively owned by the User class. Finally, we have the Changes class that contains and documents all the changes made by the User, Login, BenefitPlan, and BenefitPlanDetails classes; coding in the Changes class is only updated when there are changes in the classes that it is dependent on in the UML class diagram.

**Figure 1.1**

*UML Class Diagrams for Healthcare Insurance Web-based Exchange system*



*Note*: In the screenshot, we have three different types of relationship connector types: association, dependency, and composition aggregation. The association is represented with a straight solid line. The dependency relationship is represented with a dotted line and arrow towards the class with the dependency. The composition line is a solid line with a dog-eared diamond pointing to the class with exclusive ownership of the connected class.

**UML Class Diagrams in the Software Development Cycle**

UML Class diagrams are an important part of the software development life cycle as they serve as blueprints for actors, interfaces, and objects within the system. Class diagrams allow software developers to understand the classes in a system, its attributes, and the operations/methods that support its functionality (Visual Paradigm, 2020). Class diagrams are versatile in that they may be used in a variety of software development phases, and helps model class diagrams in three different perspectives: conceptual (contains a systematic description that relates to the real world and is independent of software languages), specification (helps detail components or abstractions with interfaces and specifications), and implementation (helps represent the objects for implementation that are specific to a coding language or application) (Guru99, 2020). Software development teams may also use class diagrams to determine how coding should connect within the system and be established before studying and diving into implementing actual code. The utilization of class diagrams supports agile methods as they are very easy to update relationships, coding, methods, and functions to reflect the new changes in the system.

**Conclusion**

UML class diagrams are helpful and essential tools in the software development lifecycle. Class diagrams further the understanding of the system by delineating all classes’ specifications before coding is fully studied and implemented. In the healthcare insurance web-exchange, six classes are important to the system: the Window, User, Login, BenefitPlan, BenefitPlanDetails (specific details of these benefit plans for each user), and the Changes class. These classes encompass and represent coding, objects, and methods that are crucial to the user’s experience and interaction with the healthcare insurance web-based exchange. By representing these classes in a UML class diagram, software developers can communicate different perspectives – such as conceptual, specification, and implementation—in various stages of the software development cycle, providing the necessary visibility of system operations and functionality in different phases of a software development project. In the final section of this report, we will be investigating the steps required to assist development teams to implement this software design.

**Preparing Software Design and Implementation for Development Teams**

Now that the initial drafts of software design, models, and analysis are complete, designers must ensure that the steps and expectations laid out are clear and easy-to-understand. Software development teams must be able to assess the design and understand what they need to do to begin implementation of the design. The goal of all designs is to ensure that software development teams receive the requests and have no questions on the purpose or design of the request, promoting efficient development models and processes.

**Guidelines**

Before requesting development on a software design, another evaluation of the design must be completed. Designers must review proposals to ensure that all necessary workflows and designs are provided in the request, along with specific databases and software tools that must be used to create the development and specific pages and details of what will be created (Azarian, 2020). It is also important that the software design document defines system capabilities and limitations to demonstrate a clear scope of the project, while also ensuring that additional requirements and development objectives do not become an impossible venture (Guinn, 2019). Communication and collaboration are the keys to ensuring software development teams are on board with the new request. Another set of guidelines that could also be used in the software design proposal is highlighting key components of the ADKAR change model, which help transition any initial resistance to the change and help foster a new perspective on the proposal at hand:

1. **Awareness.** The problem with the current system at hand needs to be clear and recognizable.
2. **Desire**. Benefits and consequences need to be incentivized. It is also important that clear goals of the new development are defined to show why it is desired and how it will optimize the current system.
3. **Knowledge**. Knowledge must be gathered to ensure the creation and maintenance of change.
4. **Ability**. Any skill gaps that may restrict changes need to be closed in terms of ability. If additional training is needed, this needs to be clearly defined so software developers understand the programs that they are working with as well as what to expect.
5. **Reinforcement**. Proposals and implementation procedures should always have clear measurable processes and reward wins. Focusing on continuous improvement will also help support the long-term implementation phase for software developers (Guinn, 2019).

**Conclusion**

An in-depth analysis is required for every software design proposal. In this report, the healthcare web-based exchange system was analyzed for its purpose, functions, stakeholders, organizations, and operating environments. From there, UML sequence and class diagrams were completed to demonstrate the usability and applicability of the system to prepare for software development implementation. By having clear objectives and goals outlined in the software design, software development teams can achieve a smooth transition from the design to the implementation phase efficiently and with ease.

**References**

Anderson, J. (2015, April 22). US20160314252A1 - Method and system for health insurance enrollment. Google Patents. https://patents.google.com/patent/US20160314252A1/en.

Azarian, I. (2020, March 26). *Key Phases of Software Development Projects*. Segue Technologies. https://www.seguetech.com/key-phases-software-development/.

Baqais, A. A. B., & Alshayeb, M. (2018). Sequence diagram refactoring using single and hybridized algorithms. *PLoS ONE*, *13(8)*. e0202629. https://doi.org/10.1371/journal.pone.0202629

Guinn, J. (2019, August 23). *5 Critical Steps for Your Software Implementation Plan*. Software Advice. https://www.softwareadvice.com/resources/software-implementation-plan/.

Guru99. (2020, December 10). *UML Class Diagram Tutorial with Examples*. Guru99. https://www.guru99.com/uml-class-diagram.html.

Insurance Information Institute. (2020). Facts + Statistics: Industry overview. https://www.iii.org/fact-statistic/facts-statistics-industry-overview.

Larman, C. (2005). In Applying UML and patterns: an introduction to object-oriented analysis and design and the unified process (pp. 54, 56-57, 66, 92, 136, 264). Prentice-Hall.

Riggins, F. J. (2009). A Framework for Identifying Web-Based Electronic Commerce Opportunities. Journal of Organizational Computing and Electronic Commerce, 297–310.

Shibly, H. A. (2011). Human Resources Information Systems success Assessment: An integrative model. Australian Journal of Basic and Applied Sciences, 5(5), 157–157.

Tutorialspoint. (2020). *UML - Class Diagram*. Tutorialspoint. https://www.tutorialspoint.com/uml/uml\_class\_diagram.htm.

USA.gov. (2020). Apply for Health Insurance. HealthCare.gov. https://www.healthcare.gov/apply-and-enroll/how-to-apply/.

Visual Paradigm. (2020). What is Sequence Diagram? https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/.

Visual Paradigm. (2020). What is Class Diagram? https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/.

Waykar, Y. (2013). MANAGELIZATION. INDIA; ResearchGate. https://www.researchgate.net/publication/322991992\_Importance\_of\_UML\_Diagrams\_in\_Software\_Development.

Zonneveld, M., Patomella, A.-H., Asaba, E., &amp; Guidetti, S. (2019). The use of information and communication technology in healthcare to improve participation in everyday life: a scoping review. Perspectives in Rehabilitation, 3416–3423.