Systems Development Life Cycle Course Project

IS316 – Computer Systems Analysis and Design II

Table of Contents

Abstract …………………………………………………………………………………………...5

System Description ……………………………………………………………………………….6

Key Aspects of Technical, Economic, and Organizational Feasibility……………………..…….6

System Requirements…………..…………………………………………………………………8

System Processes, Data Model, and Data Flow.…………………………………………..……..10

Figure 1—Use Case ID: 1………………………………………………………………..11

Figure 2—Use Case ID: 2………………………………………………………………..12

Figure 3—Use Case ID: 3………………………………………………………………..12

Figure 4—Use Case ID: 4………………………………………………………………..13

Figure 5—Use Case ID: 5………………………………………………………………..14

Figure 6—Logical ERD………………………………………………………………….15

System Design…………………………………………………………………………………...16

Acquisition Strategy……………………………………….……………………………………..16

Alternative Matrix………………………………………………………………………………..17

Figure 7—Alternative Matrix……………………………………………………………18

System Specifications…………………………………………………...……………………….18

Architecture Design…………………………………………………………………………..….19

Figure 8—Architecture Design Requirements…………………………………………..20

Figure 9—System Architecture Design………………………………………………….21

Hardware and Software Specifications………………………………………………………..…22

Figure 10—Hardware and Software Specifications……………………………………..23

Interface Design………………………………………………………………………………….23

Table of Content Continued…

Figure 11—User (Claimant) Interface Design…………………………………………...23

Figure 12—Clerk Interface Design………………………………………………………25

Physical Process Model………………………………………………………………………….25

Figure 13—Physical DFD: Module 1……………………………………………………27

Figure 14—Physical DFD: Module 2……………………………………………………28

Figure 15—Physical DFD: Module 3……………………………………………………28

Figure 16—Physical DFD: Module 4……………………………………………………28

Figure 17—Physical DFD: Module 5……………………………………………………29

Program Design………………………………………………………………………………….29

Figure 18—Module 1.0: Create a User Account………………………………………...29

Figure 19—Module 2.0: Create Clerk Account………………………………………….30

Figure 20—Module 3.0: User Account Access………………………………………….30

Figure 21—Module 4.0: Upload Claim File……………………………………………..30

Figure 22—Module 5.0: Query Claim…………………………………………………...31

Figure 23—Program Specifications……………………………………………………...32

Database and File Specifications………………………………………………………………...37

Figure 24—Database Specifications……………………………………………………..38

Figure 25—Modified Data Dictionary…………………………………………………...39

Physical Data Model……………………………………………………………………………..39

Figure 26—Physical ERD………………………………………………………………..40

Test Plan………………………………………………………………………………………….40

Figure 27—Test Plan…………………………………………………………………….41

Table of Content Continued…

References………………………………………………………………………………..………44

Abstract

System development requires an extensive and specific process to meet the client’s goals and vision. The Systems Development Life Cycle (SDLC) evolved from this concept and was later broken down into four phases: planning, analysis, design, and implementation. With this process, IS professionals can more effectively gather information and requirements to create a plan of action suitable to the client’s needs, analyze such information to make sure it maintains focus on the client’s goal, design it as it should be expected, and lastly, build it with the intent to release and eventually deploy the newly developed system. In IS315-Computer Systems Analysis and Design I, the planning and analysis phases were analyzed and discussed in conjunction with the South Dakota Department of Labor, Workers’ Compensation Division’s need for a new system to be developed to satisfy their needs while providing business value. The planning phase results yield five different documents in the form of deliverables including a system request, a feasibility analysis (study), a project plan, a staffing plan, a standards list, and a risk assessment. The analysis phase results yield deliverables in the form of a system proposal, requirements definitions, system processes, and system data flow. This course project assignment intends to continue developing the to-be system for the South Dakota Department of Labor, Workers’ Compensation Division, but this time, focusing on the last two phases of the SDLC: the design and implementation phases.

*Keywords: Phase, Planning, Analysis, Design, Implementation, SDLC, Scenario 2, South Dakota Department of Labor.*

Systems Development Life Cycle Course Project

**System Description**

The South Dakota Department of Labor, Workers’ Compensation Division currently employs a system that is based on filing paper claims. If a claimant wishes to inquiry about the status of his/her job-related claim, a department clerk must take a message, look in a spiral notebook where the department maintains a running log, retrieve the paper file, review the status, and ultimately call the claimant back to provide them with the requested information. As one can see, the as-is system has flaws. The overall process is time-consuming, inefficient, and inconvenient. The need for a digitalized solution is of the utmost importance for their office since it would alleviate many negative factors that currently impact their day-to-day operations. The proposed or to-be system should be a digital solution (web-based) where claim reports could be submitted online, and the department clerks could query a data store while simultaneously talking to the claimants in person or over the phone to provide such information faster. The requirements for the new system should improve the current process. It should be digital, faster, efficient, and alleviate their problems at the office.

**Key Aspects of Technical, Economical, and Organizational Feasibility**

As provided as a deliverable in the planning phase to assess these three areas (Feasibility Analysis), technical feasibility includes familiarity with the application, familiarity with the technology, the project size, and the system compatibility (Dennis et al., 2018, p. 19). For the South Dakota Department of Labor, Workers’ Compensation Division it was demonstrated that undertaking this project is feasible with overall moderate to high risk, as there is no familiarity with the application since there is no previous digital system in place, but the development is fresh, from the ground up. Additionally, familiarity with technology showed a high risk due to similar reasons as familiarity with the application. Since the current technology is a paper filing process, there is no existing system or technology. The project size for the to-be system was said to be of a medium size project. The project scope was limited to online document submission, creating user (claimant) and clerk accounts, querying data stores for information, and limiting it to a website and one data store. Lastly, the system compatibility was said to be of low risk. The as-is system is paper filing, whereas the to-be system is a digitalized solution.

For the economic feasibility (costs associated with the project) of the to-be system for the South Dakota Department of Labor, Workers’ Compensation Division, it was stated that the economic feasibility was adjusted based on South Dakota’s Occupational Employment and Wage Statistics found in the U.S. Bureau of Labor Statistics, combined with cost-benefit analysis. Since the to-be system was considered to be a medium size project under technical feasibility, it was stated that it requires at least the participation of five IS professionals, including a project manager, BSA, team lead, and two programmers for front and back-end. It was stated that a total amount of $596,610 was going to be the overall cost of the new system developed for the first year, breaking this cost as follows: $451,610 for labor, $15,000 for office space and equipment, $50,000 for software, and $80,000 for hardware. One must keep in mind that these numbers were generated assuming some assets are already available to the IS system development project team. Lastly, Since the budget for the South Dakota Department of Labor was said to be assigned based on yearly funds by the state, quantifying ROIs and NPVs is hard. The department does not receive any monetary gains due to being a public-funded office, no service is sold. This IS system will improve productivity in the Workers’ Compensation Division.

As for organization feasibility goes, it was stated for the South Dakota Department of Labor, Workers’ Compensation Division that from an organizational perspective, the to-be IS system had a moderate risk, due to the project being strategically aligned with the business needs. Top management support is provided by the Director of the Worker’s Compensation Division, who strongly supports this project. Moreover, organization management is on board in undertaking this project. However, few oppose the change as they are used to the as-is system and are not familiar with other technologies. Additionally, claimants will be delighted to know there is the potential to have an easier way to submit paperwork on a website. To achieve this, the user interface will have to be as straightforward as possible. Lastly, clerks at the South Dakota Department of Labor, Workers’ Compensation Division support this project. This system will reduce time, increase productivity in the workplace, and resolve confusion among claimants and clerks.

**System Requirements**

System requirements refer to “statements of what the system must do or what characteristics it needs to have” (Dennis et al., 2018, p. 88). Five categories are included when talking about system requirements: *functional, nonfunctional, business, system*, and *user*. Functional requirements describe “end user specifically demands as basic facilities that the system should offer” (GeeksforGeeks, 2020). As part of the functional requirements for the South Dakota Department of Labor, Workers’ Compensation Division, the claimant should be able to upload medical files. Additionally, the to-be system should store them, and the department clerks should be able to query a data store using the claimant’s personal information or claim number. Lastly, claimants and clerks should be able to create an account to log in, upload documents, and access them.

Nonfunctional requirements describe “quality constraints that the system must satisfy according to the project contract” (GeeksforGeeks, 2020). For nonfunctional requirements, it was stated that claimants and clerks should log in through a two-step verification process. The to-be system should use data stores to store files using software like MySQL or similar. File naming convention should be by year and case number, and written in a well-known programming language. Lastly, the querying process should be immediate, not exceeding ten seconds.

As it concerns business requirements or “what the business needs,” it was specified that for the South Dakota Department of Labor, Workers’ Compensation Division, paper filing is inefficient and requires a digital solution to improve their daily operations (Dennis et al., 2018, p. 89). Their office is the only one that requires it, and the newly developed system should allow the upload and query of medical files. The goal remains the same, as their office wishes to provide value by improving and speeding up the current process.

As previously stated in the analysis phase, the system requirements of the South Dakota Department of Labor, Workers’ Compensation Division needs the operating system to be Windows 10 or newer and supported only on PCs. The RAM should be 1 GB for 32-bit, and 2 GB for 64-bit computers, while the hard drive space should be 128 GB. It must require a minimum processor speed of 1 GHz with a graphics card of DirectX9 or later, and display at 800 x 600 pixels. For the user requirements, it was stated that the system should present a website to the claimant with a login tab, presenting an option to select whether the user is a claimant or clerk. Both login screens present an option for an account to be created. Lastly, once both individuals have logged in, the interface will change slightly where the claimant will have the option to upload documents, while a department clerk will see a search box to query claims.

**System Processes, Data Models, and Data Flow**

The following DFDs were composed, and are based on use cases composed during the analysis phase for the South Dakota Department of Labor, Workers’ Compensation Division. DFDs are a logical representation, meaning they should show logical processes only. Therefore, the DFDs shown below might not explicitly show all the inputs and outputs in the use cases shown above. However, each DFD has an explanation for how it relates to its respective inputs and outputs. Please refer to the following DFDs below:

Starting with use case ID: 1, the user/caller creates an account, and the data flow atop represents the initial user (external entity) interaction. This interaction is needed for this process to start, as it marks the user’s initiation to create a web-based account by inquiry. Since this initiation is not solely logical, it does not fit in as part of the logical processes, but it is mentioned to better understand the flow of the system. Focusing on the user and the first set of data flow, one can see input and output reaching the process “Process User Data.” Here, the user provides a bundle of data (personal information for account creation) to process 1.1 as input, as requested by the process. Once the process acquires this information, it moves to verification and creates the user’s account, as seen in process 1.2, “Verify and Create Account.” From here, the account information gets stored in a data store (D1). Each data store has a response. For this instance, the account-creating response will return a value that will tell whether the account creation is successful. Process 1.2 will send the user a notification message, letting the user know whether the account creation was a success or failure.

**Figure 1—Use Case ID: 1**

Diagram

Description automatically generated

Use case ID: 2, clerk creates an account, is similar to use case ID: 1, where the data flow atop represents the initial clerk (external entity) interaction. Again, since this initiation is not solely logical, it does not fit in as part of the logical processes, but it is mentioned to better understand the flow of the system. The same will be for all DFDs. Two arrows from process 2.1 (Process Clerk Data) are shown to represent the interaction between the clerk (external entity) and process 2.1. The clerk provides a bundle of data in the form of personal information for account creation, as requested by the process. Once the process acquires this information, it moves to verification and creates the clerk’s account, as seen in process 2.2, “Verify and Create Account.” From here, the account information gets stored in a data store (D1). Each data store has a response. For this instance, the account-creating response will return a value that will tell whether the account creation is successful. Process 2.2 will send the clerk a notification message, letting the user know whether the account creation was a success or failure.

**Figure 2—Use Case ID: 2**

Diagram

Description automatically generated

In use case ID: 3, the user logs in, and data flow shows the interaction between process 3.1, “Process User Data,” and the external entity, the user. In this case, process 3.1 will request the user’s name and password, while the user inputs such data. The system will show a welcome screen with a login tab, where the user will click to log in. The user’s information (name and password) will then move to process 3.2, “User Authorization,” where the system will determine whether the user has permission to access the actual system. The system will verify that the information entered is correct by accessing the data store (D1), and the data store will return a validation response to process 3.2, with a grant access response back to the user. The system will take the user to an account home page.

**Figure 3—Use Case ID: 3**

Diagram

Description automatically generated

Use case ID: 4, user uploads medical files, data flow shows the interaction between process 4.1, “Process User File,” and the external entity, the user. In this DFD, the system will show an upload document dialog box. As input, the user will click on the upload dialog box, process 4.1 will show the user's file explorer (request user file), and the user will select the desired files to be uploaded. Once uploaded, process 4.2, “Store File,” will store the file and send it to D1, “Workers Compensation” data store. The data store will then return a notification response to the user through process 4.2, with a notification message of a successful upload.

**Figure 4—Use Case ID: 4**

Diagram

Description automatically generated

Use case ID: 5, clerk queries a database for claims, data flow shows the interaction between user, clerk, and clerk database query. In this DFD, a user calls the South Dakota Department of Labor, Workers’ Compensation Division to inquire about the claim status. Through process 5.1, “Process User Claim Info,” the user provides a claim ID (case number), a clerk receives the claim ID, and processes it through process 5.2, “Retrieve Claim Info,” where the system will show a query box to input such information. Through the same process, the system will search and retrieve claim records based on the search criteria from data store D1, “Workers Compensation.” From here, the data store will return the queried claim record with the requested claim ID back to the clerk through process 5.2.

**Figure 5—Use Case ID: 5**

Diagram

Description automatically generated

The following diagram is the entity relationship diagram (ERD) for the South Dakota Department of Labor, Workers’ Compensation Division to-be system shown during the analysis phase. An ERD is a type of “flowchart that illustrates how entities such as people, objects, or concepts relate to each other within a system” (What Is an Entity Relationship Diagram (ERD)?, n.d.). Each box represents a data entity. These entities are the User, Clerk, Account, Medical File, and Claim. Each of these entities contains attributes that describe the properties of entities. For instance, a user entity has a first and last name. The asterisk next to an attribute indicates that that attribute is used to identify the entity. Furthermore, the entities are related to one another, and this relationship is shown by the lines among the entities. Each relationship also has cardinality and modality, and it is portrayed by the symbols at the ends of each relationship line. For example, the user has a relationship to Medical File. A user can have 0 to many Medical Files. This is known as cardinality. Since it is 0, this means that the modality is “optional”, User can exist without a Medical File. Looking at it the other way around, a medical file can have one and only one User (cardinality). The modality for this one is required. For a file to exist, it must be related to one and only one user. Lastly, the text on the lines labels the relationship.

By analyzing the ERD further, it becomes evident that a User submits a Medical File. It is submitted by only one User as seen by the two-line symbol in the relationship line. Users can submit none or many Medical Files as seen by the circle and crow’s foot symbol. Medical Files are included in Claims. Each Medical File is included in one and only one Claim. A claim can include none or many Medical Files. Claims are included in Accounts and are searched for by Clerks. Many Clerks can search many accounts and many accounts can be searched by many Clerks as shown by the crow’s foot symbol. Clerks and Users both create accounts. The remaining symbols are the same as those explained. Having this in mind, the following image represents an ERD for the South Dakota Department of Labor, Workers’ Compensation Division.

**Figure 6—Logical ERD**

Diagram, schematic

Description automatically generated

**System Design**

During the design phase of the SDLC, system development refinement happens. It is an essential phase in software development since this is the phase where decisions are made, in terms of “how the system will operate, hardware and software specifications, and the network infrastructure that will be in place” (Dennis et al., 2018, p. 10). The design phase is composed of four steps. The first step includes a design strategy. Here, IS professionals must clarify the Acquisition strategy for the to-be system. A decision needs to be made for the development of the to-be system to be a custom development (in-house), a packaged solution (purchased or obtained from ASPs), or outsourced and created by a third party. Once a decision has been made, the system development moves into the second step: the development of the architectural design. During this step, IS professionals further refine all requirements gathered during the analysis phase. Hardware, software, and network specifications are described, and an interface design is created. Step three of the design phase deals with data stores and file specifications. These two mechanisms define data and its storage. Lastly, IS professionals can move into the fourth step: program design. The program design provides information for IS teams to construct the to-be system with specifications and what each component should do. As a deliverable of the design phase, taking into consideration the four steps described, a system specification is produced.

**Acquisition Strategy**

For the South Dakota Department of Labor, Workers’ Compensation Division's new digital system solution, the acquisition strategy should be based on the information found in the planning and analysis phase’s deliverables. As an example, the outcome of the planning phase was a system request document. During the planning phase, the business needs, business requirements, business value, and constraints were mentioned, as well as a feasibility analysis that included technical, economic, and organizational feasibility. It was stated for the South Dakota Department of Labor, Workers’ Compensation Division that the new system must be a digitalized solution to increase workflow productivity while decreasing clerk buffering time, the specifications or what the Labor’s office wishes to achieve, etc. Most importantly, in the feasibility analysis, it was stated that developing this digitalized solution system is doable as a medium-sized project, compatible with the as-is system since there is no current system in place, and economically makes sense, among other factors.

Although some feasibilities were said to be of moderate to high risk, there might be packages available that could check all the boxes and offer value to the Labor’s office, or even the notion to outsource the system development completely. In a real-world scenario, it is best to develop in-house to develop better IS professionals and teams, and pass that knowledge along. This way, the business “gets exactly what they want, allows team flexibility and creativity, unique solutions are created for strategic advantage, and the new system is consistent with existing technology and standards” (Dennis et al., 2018, p. 225). Therefore, the best acquisition strategy for developing the to-be digitalized system for the South Dakota Department of Labor, Workers’ Compensation Division is custom development.

**Alternative Matrix**

An alternative matrix describes “what can be used to organize the pros and cons of the design alternatives so that the best solution will be chosen in the end” (Dennis et al., 2018, p. 234). An alternative matrix contains the technical, economical, and organizational feasibility analysis from the analysis phase, and other related information, placed into a matrix or table, for easier comparison, visually offering alternative routes for the to-be system. Please refer to the alternative matrix below for the South Dakota Department of Labor, Workers’ Compensation Division’s to-be system. For the first alternative listed, custom development, the cost estimations in the alternative matrix were based on South Dakota’s Occupational Employment and Wage Statistics found in the U.S. Bureau of Labor Statistics, combined with a cost-benefit analysis. For the second and third alternatives listed, packaged solution and outsourced development, the cost estimation averages were obtained from an article titled, *How much does software development cost?* (refer to *References Section*).

**Figure 7—Alternative Matrix**

Table

Description automatically generated

**System Specifications**

As previously stated, the system specification document is the deliverable resulting from finalizing the design phase. It is composed of the architectural design, interface design, database and file specifications, and program design. It “conveys exactly what system the project team will implement during the implementation phase of the SDLC” (Dennis et al., 2018, p. 223). For the South Dakota Department of Labor, Workers’ Compensation Division, this means that an acquisition strategy for the to-be system has been selected, and an alternative matrix has been composed and provided to show alternative courses of action (as shown above). Physical data and physical process models will be constructed to convert logical data/processes into physical data/processes, while program design specifications will be addressed to further develop, maintain, and support the to-be system. Additionally, an interface design will be developed that guides the system through a series of inputs and outputs. Also, a data storage design will be included. Use case entries will get updated, and the CRUD matrix will be revised during the design phase, as the project moves along. Lastly, an architectural design that complements and provides value to the to-be system will be suggested. Hardware and software specifications will be defined in detail, making the development process more efficient and effective. Please see below the suggested architecture design, along with hardware and software specifications.

**Architecture Design**

For the architectural design of the to-be system, nonfunctional requirements have to be further analyzed into four subcategories, including operational, performance, security, and cultural and political requirements. These requirements concerning the South Dakota Department of Labor, Workers’ Compensation Division are addressed below:

**Figure 8—Architecture Design Requirements**

**Table

Description automatically generated**

By analyzing nonfunctional requirements, the architecture design options become less perplexing. The to-be system will employ a combination of a web-based architecture with a thin client-server architecture, and be referred to as the User Architecture, for explanation purposes. An installed application architecture with a thin client-server architecture will be referred to as the Clerk Architecture for explanation purposes. Additionally, the to-be system will also employ cloud computing. The User Architecture will mainly be intended for the user (claimant) to upload medical files and claims documentation. The Clerk Architecture will be intended for the clerks at the labor’s office to query users’ claims. The cloud computer will act as the server for both, the web and the client. Moreover, some cloud computing components will be AWS DynamoDB and AWS S3 Storage. Please refer to the diagram below, created in app.diagrams.net (refer to *References Section*):

**Figure 9—System Architecture Design**

Graphical user interface, application

Description automatically generated

As shown above, on the Clerk side (left side), the installed software will handle the presentation logic, along with some application logic. On the User side (right side), the web-based application will handle the presentation logic with some application logic. Cloud computing will handle the data storage, access logic, and most of the application logic (back-end). S3 or “Amazon Simple Storage Service (Amazon S3) is an object storage service” that will store the claimant’s documents, and once submitted, a clerk from the labor’s office will be able to query them (What Is Amazon S3? - Amazon Simple Storage Service, n.d.). Additionally, DynamoDB, “a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability” will be used (What Is Amazon DynamoDB? - Amazon DynamoDB, n.d.). It will store the Claim database data, user accounts, etc. Lastly, Amply, another AWS service, will function as the back-end host. This is where the main application logic and data access logic lives. Through this service, S3 and DynamoDB will be reached and sent back to the client devices.

The use of cloud computing has been selected because it provides several advantages. First and foremost, the main advantage of using cloud computing is that the South Dakota Department of Labor, Workers’ Compensation Division does not have to worry about maintaining servers since the implementation of the to-be system will use AWS services. This way money, assets, hardware, personnel (IS professionals), etc., can be saved or used in other areas of interest. Furthermore, AWS is scalable, and the cost is dependent on service consumption. Additionally, cloud computing allows for connectivity anywhere there is an internet connection, allowing clerks to work remotely. Not to mention, the various cross-platform frameworks associated with cloud computing, and the many different system applications that can be added or developed in the future, catering to the South Dakota Department of Labor, Workers’ Compensation Division’s everchanging business strategy.

**Hardware and Software Specifications**

During the design phase of the SDLC, hardware, and software specifications need to be well-defined. The following diagram is composed of hardware and software specifications needed for the South Dakota Department of Labor, Workers’ Compensation Division’s future system. One must keep in mind that the specifications for the Standard Application Server, Standard Storage Server, and Standard Database Server depicted in the diagram below are for specificity. However, the South Dakota Department of Labor, Workers’ Compensation Division does not need to worry much about these specifications, since AWS manages those systems. This is an enormous advantage of cloud computing.

**Figure 10—Hardware & Software Specifications**

Table

Description automatically generated

**Interface Design**

The appropriate interface design for the South Dakota Department of Labor, Workers’ Compensation Division should be constructed in a way that is simple to use, fulfills all business needs, requirements, and presents all design principles and arguments presented so far in this document. For the South Dakota Department of Labor, Workers’ Compensation Division, their new system’s user interface prototype should resemble the following:

**Figure 11—User (Claimant) Interface Design**

Graphical user interface, application

Description automatically generated

The above graphical user interface prototype explains the user interface's claimant (user) side. The prototype design starts by showing the South Dakota Department of Labor, Workers’ Compensation Division’s website. Since the architectural design of the to-be system employs a combination of web-based architecture with a thin client-server architecture for the user, adding additional features to their existing website is the best course of action to take. The user can take two paths. Once the user opts to mouse-click on “Login,” the display screen will change to reflect the next step: account login or create a new account. If the user already has an account, the first path is fulfilled, and the user can proceed to his/her account homepage. The second path takes the user into creating an account. Once the user has inputted all necessary information in the provided fields, an account will be created, saved, and the account homepage will be displayed. Once the user is on his/her account homepage, the option for uploading claims will be present. Under the menu “Claims,” a user can mouse-click the sub-menu “Upload Claim,” to upload his/her medical claim. Once the user has done so, the display screen will reflect the next action: upload medical files by dragging them into the provided box or clicking the link that will take the user into browsing his/her computer hard drive. Once this has been done, the user can mouse-click “Done,” and the process is finalized.

The graphical user interface prototype below explains the clerk’s side of the user interface. The prototype design starts by showing the South Dakota Department of Labor, Workers’ Compensation Division’s installed software program called “*South Dakota Dept. of Labor & Regulation Claims Processing System*,” for explaining purposes. In architecture design, it was stated that an installed application architecture with a thin client-server architecture was going to be used for the department. Having this in mind, the graphical user interface prototype for the clerks shows a login screen that contains two paths: login and new user. Similar to the user’s interface design, if a clerk does not have an active account, the system will display a “Create Account” window, once the clerk mouse-clicks “Create Account”. Similarly, if the clerk has an active and existing account, the system will display a “Claim ID” window, where the clerk will be able to query medical claim files uploaded by claimants (users), fulfilling the system purpose, and all the business needs for the South Dakota Department of Labor, Workers’ Compensation Division.

**Figure 12—Clerk Interface Design**

Graphical user interface, application

Description automatically generated

**Physical Process Model**

It is important to mention the distinction between a logical and physical process model. During the analysis phase for the South Dakota Department of Labor, Workers’ Compensation Division, logical data flow diagrams (DFDs) were created to aid in the development of the to-be system. These diagrams focused on “the business and business activities…describing the business events that take place and the data required for each event” (Logical Vs. Physical Data Flow Diagram, n.d.). In simplest terms, it answered the question of “what” the business needs. On the other hand, during the design phase, these logical DFDs get converted into physical DFDs (process models), to map out the process of information in the to-be system, but this time, these models “depict how the data system will work, such as the hardware, software, paper files, and people involved,” answering the question “how” the business needs will get accomplished (Logical Vs. Physical Data Flow Diagram, n.d.). For the South Dakota Department of Labor, Workers’ Compensation Division, the following physical process models below will represent their to-be system physical DFDs.

One must keep in mind to create physical data flow diagrams from logical ones, a few steps need to be followed. These steps include taking the existing logical DFDs from the analysis phase and adding implementation references, setting human-machine boundaries, and adding system-related data stores, data flows, and processes. It is important to mention that updating data elements in the data flow and updating metadata in the CASE repository are steps in turning a logical process model into a physical one “will not be apparent on the physical DFD” (Dennis et al., 2018, p. 324). Referencing the use cases, steps previously mentioned, and logical DFDs from the analysis phase, the explanation for the physical DFDs remains similar to the logical ones discussed previously, with some exceptions. The difference lies in the first three steps of creating a physical process model. For the South Dakota Department of Labor, Workers’ Compensation Division, this entails the separation of human-machine boundary, represented by dotted lines. Moreover, during the architecture design, it was stated that the database of the to-be system was going to employ a combination of a web-based architecture with a thin client-server architecture. Thus, the data store of choice for this system is DynamoDB for the system’s account, user, file, and claims information. For the claimant’s documents, S3 is used (shown in processes 4.1 and 4.2 and the data store). Additionally, the data flow in the physical DFDs represents the system-related data stores and processes associated with how the system works. As an example of the changes required for a physical DFD, below, processes 1.1 and 1.2 are shown to be written in Java. The User for these two processes is shown with dotted lines to represent the human-machine boundary. The data store where the claimant’s information will be saved is in the Account Table on DynamoDB. The data flow arrows show the type of data that is being transmitted at each process. For instance, process 1.2 sends an Account Created object transmitted through a DynamoDB Record Save instance. All other data flows contain this extra information. This applies to all the physical DFDs below. There is one difference to be noted in the physical DFD of processes 5.1 and 5.2. This one shows an ASCII table that is used to keep track of claim ids that are logged. This is for auditing purposes and also shows step 3 of going from a logical to a physical DFD.

**Figure 13—Physical DFD: Module 1**

Graphical user interface, website

Description automatically generated

**Figure 14—Physical DFD: Module 2**

Graphical user interface, website

Description automatically generated

**Figure 15—Physical DFD: Module 3**

Graphical user interface, application, Teams

Description automatically generated

**Figure 16—Physical DFD: Module 4**

Graphical user interface, application, Teams

Description automatically generated

**Figure 17—Physical DFD: Module 5**

Graphical user interface, application, Teams

Description automatically generated

**Program Design**

Within the program design for the to-be system for the South Dakota Department of Labor, Workers’ Compensation Division, two additional models need to be depicted. This includes structure charts and program specifications. By using structure charts, IS professionals can follow sequences, selections, and iterations among the programming aspect of the new system, since these types of charts are structured and arranged in a hierarchical form, being read from left to right. Please refer to below for South Dakota Department of Labor, Workers’ Compensation Division structure charts:

**Figure 18— Module 1.0: Create a User Account**

A picture containing application

Description automatically generated

**Figure 19— Module 2.0: Create Clerk Account**

A picture containing application

Description automatically generated

**Figure 20— Module 3.0: User Account Access**

Graphical user interface, application, Teams

Description automatically generated

**Figure 21— Module 4.0: Upload Claim File**

Graphical user interface, application, Teams

Description automatically generated

**Figure 22— Module 5.0: Query Claim**

Graphical user interface, application, Teams

Description automatically generated

The diagrams above show processes 1.0: *Create User Account*, 2.0: *Create Clerk Account*, 3.0: *User Account Access*, 4.0: *Upload Claim File*, and 5.0: *Query Claim*. Each process is broken down into sub-modules in the second and third levels, respectively. Each structure chart shows specific arguments that ensure “all the components of code that must be included in a program at a high level” are present (Dennis et al., 2018, p. 327). For example, the diagram above shows process 1.0: *Create User Account*, with two processes: 1.1 *Process User Data* and 1.2 *Verify and Create Account*. To process user data, there is a need for an additional module, 1.1.1: *Enter User Personal Info.,* where the user will input data. To verify and create an account, another module is needed, 1.2.1: *Save User Personal Info*., which saves the user data. The diagram also shows how the information is moved between DFD levels, and shows the account creation response from 1.2 as a notification (depicted by a control couple), going back to the calling module, 1.0. The rest of the processes follow a similar pattern, as they are self-explanatory.

The second additional model that needs to be addressed in the program design is the program specifications. IS professionals can utilize program specifications to ensure what is being developed meets all criteria and the business needs, including design problems and anything else related to the development of the new system that could cause trouble. Using the structured charts created for the South Dakota Department of Labor, Workers’ Compensation Division, their respective program specifications are as follows:

**Figure 23— Program Specifications**

|  |  |  |  |
| --- | --- | --- | --- |
| **Program Specification 1.0 for Create User Account** | | | |
| **Module Name:** 1.1.1 Enter Claimant (user) Personal Information | | | |
| **Purpose:** Creates Claimant (user) account | | | |
| **Programmer:** Luis Hurtado | | | |
| **Date Due:** 11/20/22 | | | |
| **C** | **Python** | **Visual Basic** | **Java** |
| **Events:**  Occurs upon the user initiating account creation. | | | |
| **Input Name**  User Personal Info | **Type**  Object | **Provided by**  User | **Notes** |
| **Output Name**  User Personal Info | **Type**  Object | **Used by**  Module 1.2.1 | **Notes** |
| **Pseudocode**  enter\_user\_info(user\_personal\_info):  send(user\_personal\_info); | | | |
| **Other** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Program Specification 1.0 for Create User Account** | | | |
| **Module Name:** 1.2.1 Save Claimant (user) Personal Information | | | |
| **Purpose:** Saves Claimant (user) data in a data store | | | |
| **Programmer:** Luis Hurtado | | | |
| **Date Due:** 11/20/22 | | | |
| **C** | **Python** | **Visual Basic** | **Java** |
| **Events:**  Occurs upon entry of User Personal Info by user | | | |
| **Input Name**  User Personal Info | **Type**  Object | **Provided by**  Module 1.1.1 | **Notes** |
| **Output Name**  Notification | **Type**  String | **Used by**  Module 1.0 | **Notes** |
| **Pseudocode**  Save\_user\_personal\_information(user\_personal\_info):  return Save(user\_personal\_info); | | | |
| **Other** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Program Specification 2.0 for Create Clerk Account** | | | |
| **Module Name:** 2.1.1 Enter Clerk Personal Information | | | |
| **Purpose:** Creates Clerk account | | | |
| **Programmer:** Luis Hurtado | | | |
| **Date Due:** 11/20/22 | | | |
| **C** | **Python** | **Visual Basic** | **Java** |
| **Events:**  Occurs upon the clerk initiating account creation. | | | |
| **Input Name**  Clerk Personal Info | **Type**  Object | **Provided by**  clerk | **Notes** |
| **Output Name**  Clerk Personal Info | **Type**  Object | **Used by**  2.1.2 | **Notes** |
| **Pseudocode**  enter\_clerk\_info(clerk\_personal\_info):  send(clerk\_personal\_info); | | | |
| **Other** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Program Specification 2.0 for Create Clerk Account** | | | |
| **Module Name:** 2.2.1 Enter Clerk Personal Information | | | |
| **Purpose:** Saves Clerk data in a data store | | | |
| **Programmer:** Luis Hurtado | | | |
| **Date Due:** 11/20/22 | | | |
| **C** | **Python** | **Visual Basic** | **Java** |
| **Events:**  Occurs upon entry of Clerk Personal Info by clerk | | | |
| **Input Name**  Clerk Personal Info | **Type**  Object | **Provided by**  Module 2.1.1 | **Notes** |
| **Output Name**  Notification | **Type**  String | **Used by**  Module 2.0 | **Notes** |
| **Pseudocode**  Save\_clerk\_personal\_information(clerk\_personal\_info):  return save(clerk\_personal\_info); | | | |
| **Other** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Program Specification 3.0 for User Account Access** | | | |
| **Module Name:** 3.1.1 Enter User name and Password | | | |
| **Purpose:** Process Claimant (user) account data | | | |
| **Programmer:** Luis Hurtado | | | |
| **Date Due:** 11/20/22 | | | |
| **C** | **Python** | **Visual Basic** | **Java** |
| **Events:**  Occurs upon user initiating account access | | | |
| **Input Name**  Username and Password | **Type**  Object | **Provided by**  User | **Notes** |
| **Output Name**  Username and Password | **Type**  Object | **Used by**  3.1.2 | **Notes** |
| **Pseudocode**  Enter\_username\_password(usernameAndPassword):  Send(usernameAndPassword) | | | |
| **Other** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Program Specification 3.0 for User Account Access** | | | |
| **Module Name:** 3.2.1 Retrieve Claimant (user) account | | | |
| **Purpose:** Retrieves Claimant (user) account data | | | |
| **Programmer:** Luis Hurtado | | | |
| **Date Due:** 11/20/22 | | | |
| **C** | **Python** | **Visual Basic** | **Java** |
| **Events:**  Occurs upon entering username and password by user | | | |
| **Input Name**  Username and Password | **Type**  Object | **Provided by**  Module 3.1.1 | **Notes** |
| **Output Name**  Grant Access Response | **Type**  String | **Used by**  Module 3.0 | **Notes** |
| **Pseudocode**  Retrieve\_account(usernameAndPassword):  return retrieve\_user(usernameAndPassword) | | | |
| **Other** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Program Specification 4.0 for Upload Claim File** | | | |
| **Module Name:** 4.1.1 Enter File | | | |
| **Purpose:** Claimant (user) upload claim file | | | |
| **Programmer:** Luis Hurtado | | | |
| **Date Due:** 11/20/22 | | | |
| **C** | **Python** | **Visual Basic** | **Java** |
| **Events:**  Occurs when the user initializes medical file submission | | | |
| **Input Name**  File | **Type**  Object | **Provided by**  User | **Notes** |
| **Output Name**  File | **Type**  Object | **Used by**  4.1.2 | **Notes** |
| **Pseudocode**  Enter\_file(file):  Send(file) | | | |
| **Other** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Program Specification 4.0 for Upload Claim File** | | | |
| **Module Name:** 4.2.1 Save File | | | |
| **Purpose:** Claimant (user) file data gets saved | | | |
| **Programmer:** Luis Hurtado | | | |
| **Date Due:** 11/20/22 | | | |
| **C** | **Python** | **Visual Basic** | **Java** |
| **Events:**  Occurs upon user submitting medical File | | | |
| **Input Name**  File | **Type**  Object | **Provided by**  Module 4.1.1 | **Notes** |
| **Output Name**  Notification | **Type**  String | **Used by**  Module 4.0 | **Notes** |
| **Pseudocode**  save\_file(file):  return create\_file(file) | | | |
| **Other** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Program Specification 5.0 for Query Claim** | | | |
| **Module Name:** 5.1.1 Provide Claim ID | | | |
| **Purpose:** Claimant (user) provides Claim ID to clerk via call | | | |
| **Programmer:** Luis Hurtado | | | |
| **Date Due:** 11/20/22 | | | |
| **C** | **Python** | **Visual Basic** | **Java** |
| **Events:**  Occurs upon user calling clerk and provides claim ID. | | | |
| **Input Name**  Claim ID | **Type**  Integer | **Provided by**  User | **Notes** |
| **Output Name**  Claim ID | **Type**  Integer | **Used by**  Module 5.1.2 | **Notes** |
| **Pseudocode**  Provide\_claim\_id(claimId):  Send(claimId) | | | |
| **Other** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Program Specification 5.0 for Query Claim** | | | |
| **Module Name:** 5.1.2 Enter Claim ID | | | |
| **Purpose:** Clerk inputs Claim ID from Claimant (user) | | | |
| **Programmer:** Luis Hurtado | | | |
| **Date Due:** 11/20/22 | | | |
| **C** | **Python** | **Visual Basic** | **Java** |
| **Events:**  Occurs upon user providing claim id to clerk | | | |
| **Input Name**  Claim ID | **Type**  Integer | **Provided by**  Module 5.1.1 | **Notes** |
| **Output Name**  Claim ID | **Type**  Integer | **Used by**  Module 5.2.1 | **Notes** |
| **Pseudocode**  Enter\_claim\_id(claimId)  Send(claimId) | | | |
| **Other** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Program Specification 5.0 for Query Claim** | | | |
| **Module Name:** 5.2.1 Retrieve Records | | | |
| **Purpose:** Clerk retrieves records using Claim ID | | | |
| **Programmer:** Luis Hurtado | | | |
| **Date Due:** 11/20/22 | | | |
| **C** | **Python** | **Visual Basic** | **Java** |
| **Events:**  Occurs upon submission of claim id by clerk | | | |
| **Input Name**  Claim ID | **Type**  Integer | **Provided by**  Module 5.1.2 | **Notes** |
| **Output Name**  Claim Records | **Type**  Array | **Used by**  Module. 5.0 | **Notes** |
| **Pseudocode**  Retrieve\_records(claimId):  Return get\_records(claimId) | | | |
| **Other** | | | |

**Database and File Specification**

Database and file specifications are designed to provide IS professionals, especially the programming team, with formal written specifications as to how all data types are to be used, objects declared, and operations defined, concerning the data storage design aspect of system development. For the South Dakota Department of Labor, Workers’ Compensation Division, there are no file specifications, since the whole aspect of their system development addresses only databases, as all data used in conjunction with cloud computing will be saved using Amazon’s S3 (Amazon Simple Storage Service) and DynamoDB, previously stated above under *Architecture Design.* The following table represents the South Dakota Department of Labor, Workers’ Compensation Division suggested data types, uses, and format:

**Figure 24— Database Specifications**

Table

Description automatically generated

As one can see in the database specifications table above, the six types of data that make up the database and data storage service are described. These tables include entities such as User and Clerk information, Medical Documents, Medical Files, Claims, and Account information. The bulk of the data in the system would mostly be text that makes up Users, Medical Files, Claims, Accounts, and Clerks. A relational database on DynamoDB would be able to handle the data effectively. There would also be an image file format for the Medical Documents that are uploaded to S3 (Amazon Simple Storage Service).

As previously mentioned, each of the entities in the database specifications table creates the database and data storage. A data dictionary is a helpful tool since it can further illustrate and break down these entities, as data dictionaries show their attributes and relationships, and assist with data inconsistencies. For the South Dakota Department of Labor, Worker’s Compensation Division, a modified data dictionary (previously constructed during the analysis phase) is shown below. The main advantage of this modified data dictionary is that it shows relational instances among entities which is helpful in the database specifications as seen later in physical data models, and it describes all attributes and relationships for the South Dakota Department of Labor, Workers’ Compensation Division’s system.

**Figure 25— Modified Data Dictionary**

Table

Description automatically generated

**Physical Data Model**

Dennis et al. mention “following the selection of data storage format, the data model created during analysis is modified to reflect this implementation decision” (2018, p. 355). This happens by taking the existing logical entity relationship diagram (ERD) and turning it into a physical data model, or physical ERD. Physical ERDs can show references and implementations in detail while explaining the “how” aspect of the expected system, and how the data will be stored. The following physical ERD was constructed for the South Dakota Department of Labor, Workers’ Compensation Division:

**Figure 26— Physical ERD**

Graphical user interface, application, Teams

Description automatically generated

**Test Plan**

One crucial aspect of the implementation phase, the last phase in the Systems Development Life Cycle (SDLC), is developing a test plan. Dennis et al. mention “each test has a specific objective, describing a set of very specific test cases to examine, and define the expected results and the actual results observed” (Dennis et al., 2018, p. 392). For IS professionals, a test plan ensures the quality of a system since all program specifications are tested, and should an error occur, the possibility to fix it. A test plan is found below for the South Dakota Department of Labor, Workers’ Compensation Division. The test plan created shows different aspects of the to-be system. Account Creation, Medical File Upload, Query Claim, and User Accesses Account are the primary requirements addressed in the test plan that are crucial for the expected system to function properly. For simplification purposes, Account Creation and User Accesses Account for both the user and clerk are not shown, but the procedure should be similar, if not the same for testing purposes.

**Figure 27— Test Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **South Dakota Department of Labor, Workers’ Compensation Division To-be System Test Plan** | | | | | |
| **Program ID:**PID001 | **Version Number:**1 | | | | |
| **Tester:**John Doe | **Date Designed:**12/04/2022 | | **Date Conducted:**12/04/2022 | | |
| **Results: PASSED** | | | | | |
|  | | | | | |
| **Test ID:**T01 | **Requirement Addressed:** Account Creation | | | | |
| **Objective:** To verify that an account is created for the user and an account is created for the clerk. | | | | | |
| **Test Cases** | | | | | |
| **Interface ID** | | **Data Field** | **Value Entered** | **Actual Result / Notes** | **Expected Result** |
| 1. REQ-001 User creates account | | User Personal Info | {firstName: "Tim", lastName: "Doe", SSN: "232-23-2323", email:"test@email.com", pass: "pass", confirmPass: "pass", username: "username"} | "Account Creation was Successful" success message | "Account Creation was Successful" success message |
| 2. REQ-002 User creates account with blank input | | User Personal Info | Blank | "Please enter perosnal info" error message | "Please enter perosnal info" error message |
| 3. REQ-003 User creates account with numbers as name | | User Personal Info | {firstName: "123", lastName: "123", SSN: "232-23-2323", email:"test@email.com", pass: "pass", confirmPass: "pass", username: "username"} | "Invalid name" error message | "Invalid name" error message |
| 4. REQ-004 User creates account with existing SSN | | User Personal Info | {firstName: "123", lastName: "123", SSN: "232-23-2323", email:"test@email.com", pass: "pass", confirmPass: "pass", username: "username"} | "Account for that SSN already exists" error message | "Account for that SSN already exists" error message |
| 5. REQ-005 User creates account and passwords don't match | | User Personal Info | {firstName: "123", lastName: "123", SSN: "232-23-2323", email:"test@email.com", pass: "pass", confirmPass: "pazz", username: "username"} | "Passwords don't match" error message | "Passwords don't match" error message |
| 6. REQ-006 User creates account with wrong email format | | User Personal Info | {firstName: "123", lastName: "123", SSN: "232-23-2323", email:"test@@@email.com", pass: "pass", confirmPass: "pass", username: "username"} | "Email address is unvalid" error message | "Email address is invalid" error message |
| **Script** | | | | | |
|  |
|  | | | | | |  |
|  |
| **Test ID:**T02 | **Requirement Addressed:** Medical File Upload | | | | |  |
| **Objective:** To verify that the user can upload medical files | | | | | |  |
| **Test Cases** | | | | | |  |
| **Interface ID** | | **Data Field** | **Value Entered** | **Actual Result / Notes** | **Expected Result** |  |
| 1. REQ-101 User uploads medical file | | File | Valid image (JPEG, PDF, etc.) | "Successfully uploaded file" message | "Successfully uploaded file" message |  |
| 2. REQ-102 User uploads medical file with invalid format | | File | Invalid image (MP4) | "Invalid file format" error message | "Invalid file format" error message |  |
| 3. REQ-103 User uploads medical file with size over the limit | | File | Valid image size 4GB | "File too big" error message | "File too big" error message |  |
| **Script** | | | | | |  |
|  |
|  | | | | | |  |
|  |
| **Test ID:**T03 | **Requirement Addressed:** Query claim | | | | |  |
| **Objective:** To verify that the clerk can query for claims with a provided claim ID. | | | | | |  |
| **Test Cases** | | | | | |  |
| **Interface ID** | | **Data Field** | **Value Entered** | **Actual Result / Notes** | **Expected Result** |  |
| 1. REQ-201 Clerk queries with a valid claim ID | | Claim ID | "10-12345" | "1 record(s) found" message / record is shown | "1 record(s) found" message / record is shown |  |
| 2. REQ-202 Clerk queries with an invalid claim ID | | Claim ID | "ruffus123" | "0 record(s) found" message | "0 record(s) found" message |  |
| 3. REQ-203 Clerk queries with blank as claim ID | | Claim ID | blank | "Please provide claim id" error message | "Please provide claim id" error message |  |
| **Script** | | | | | |  |
|  |
|  | | | | | |  |
|  |
| **Test ID:**T04 | **Requirement Addressed:** User accesses account (logs in) | | | | |  |
| **Objective:** To verify that the user can access account. | | | | | |  |
| **Test Cases** | | | | | |  |
| **Interface ID** | | **Data Field** | **Value Entered** | **Actual Result / Notes** | **Expected Result** |  |
| 1. REQ-301 User accesses account with valid username and password | | Username and Password | {username: "username", password: "pass"} | User is logged in | User is logged in |  |
| 2. REQ-302 User accesses account with wrong password | | Username and Password | {username: "username", password: "wrong"} | "Invalid password" error message | "Invalid password" error message |  |
| 3. REQ-303 User accesses account with invalid username | | Username and Password | {username: "usernamewrong", password: "pass"} | "Invalid username" error message | "Invalid username" error message |  |
| 3. REQ-304 User accesses account with blank | | Username and Password | blank | "Please enter a valid username and password" error message | "Please enter a valid username and password" error message |  |
| **Script** | | | | | |  |
|  |

**References**

Assistant Secretary for Public Affairs. (n.d.). *Use Cases | Usability.gov*. Retrieved September 25, 2022, from https://www.usability.gov/how-to-and-tools/methods/use-cases.html

Dennis, A., Wixom, B., & Roth, R. M. (2018). *Systems Analysis and Design* (7th ed.) [E-book]. Wiley.

*Diagrams.net - free flowchart maker and diagrams online*. (n.d.). Retrieved November 6, 2022, from https://app.diagrams.net/

GeeksforGeeks. (2020, April 29). *Functional vs Non-Functional Requirements*. Retrieved October 3, 2022, from https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/

*How much does software development cost?* (2020, August 26). thumbtak.com. Retrieved October 6, 2022, from https://www.thumbtack.com/p/software-development-costs

*Logical vs. Physical Data Flow Diagram*. (n.d.). Lucidchart. Retrieved November 20, 2022, from https://www.lucidchart.com/pages/data-flow-diagram/logical-vs-physical-data-flow-diagram

*What is Amazon S3? - Amazon Simple Storage Service*. (n.d.). Retrieved November 6, 2022, from https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

*What is an Entity Relationship Diagram (ERD)?* (n.d.). Lucidchart. Retrieved October 3, 2022, from https://www.lucidchart.com/pages/er-diagrams

*What is Amazon DynamoDB? - Amazon DynamoDB*. (n.d.). Retrieved November 6, 2022, from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html