Programming Pangolins

CSC 470

Mr. Jenkins

10/23/22

# Project Requirements

* FURPS+
  + F = Functionality
  + U = Usability
  + R = Reliability
  + P = Performance
  + S = Supportability
  + D = Design Constraints
  + Others not necessary

|  |
| --- |
| **Functional Requirements** |
| 1. Must Requirements    1. Compile payroll data into a pay stubs PDF and a NACHA files (F)       1. User selects the employees for payroll from a list (UF)       2. User inputs the number of hours for each hourly employee (UF)       3. User selects the save location for each file (UF)       4. The system builds a PDF for the pay stubs (F)       5. The system builds the NACHA file (F)       6. The system displays a message stating the process is completed (UF)       7. Net pay cannot be a negative amount (F)    2. Add a new employee to the database (SUF)       1. User inputs nearly all employee information (required) (UF)          1. Employee ID is automatically calculated by the database (F)          2. Deductions will be entered at a later point (F)       2. An employee should either be “Hourly” or “Salary” with an input box for the dollars per hour or salary per pay period respectively (F)    3. Delete employee from the database (SUF)       1. User chooses the employees to delete (UF)       2. The user cannot delete every user with access to payroll (F)          1. User cannot delete itself (F)    4. Change business information in the database (SF)    5. Login to access payroll functions (UF)       1. The user needs an employee ID and password (F)    6. No information is saved when closing any screen early (UF)    7. The system validates any information before doing the required operation (adding, updating, deleting) to the database (F)       1. To validate, the input must not be empty and must fit the required format (F) 2. Should Requirements    1. Display invalid input message to the user (UF)       1. The system displays a message stating the entered information is incorrect (UF)       2. The system should not erase text from the screen when displaying this message (UF)    2. Edit existing employee in the database (SUF)       1. Cannot edit employee ID (F)    3. Add/Delete payroll deductions for an employee (SUF)       1. Deductions can be percentages or flat but not both (F)       2. Deductions cannot be a negative amount (F)    4. View employee information (F)       1. Any employee can view his/her information with the correct login (UF)    5. When successfully finishing editing, deleting, or adding an employee, display a message saying that the operation was successful (UF)    6. When finishing editing or adding an employee or any other information (company, deductions) and the system successfully saves it, clear any text from the screen and stay on that screen (UF) 3. Could Requirements    1. The system should automatically close the employee information screen after 30 seconds of inactivity    2. Allow user to change his/her password    3. User has the option to input the number of overtime hours during payroll for each hourly employee    4. Implement a time clock    5. Archive old employees and their information    6. Have another option for payroll besides hourly/salary: “Other”       1. This would be used for commission employees or someone similar    7. Print a summary page at the end of the pay stubs 4. Won’t Requirements    1. Expand how the system calculates federal tax    2. Computer year-to-date payroll information       1. Requires long-term storage of payroll data in the database |

|  |
| --- |
| **Nonfunctional Requirements** |
| 1. Must/Should Requirements    1. Operational Requirements       1. The system will be compatible with Windows 10/11 Desktop Computers (D)       2. The database will be available 20 hours a day at minimum (R)          * The database will be AWS (DS)       3. All needed extra software will be installed by the installation files (SR)    2. Performance Requirements       1. Displaying any screen will take less than 5 seconds (P)       2. 5 users should be able to use the software at the same time (P)       3. Connecting to and querying databases will take less than 3 seconds (P)       4. Compiling the payroll information to a PDF file and a NACHA file should take less than 30 seconds (P)       5. Displaying an invalid input message takes less than 2 seconds (P)       6. Starting the application should take less than 10 seconds (P)    3. Security Requirements       1. None of the employee’s information will be stored long-term outside of the database (excluding the pay stubs and NACHA files) (F)       2. Ensure that sensitive information (SSN, Password) is stored securely and encrypted (F)          * The information also needs to be able to be decrypted (F)       3. No requirements for password but they are always of length 4 (F)       4. Only users with the proper permissions should be able to access the payroll operations (F)    4. Cultural and Political Requirements       1. Our team will try our best to comply with a simplified version of payroll law (RD) 2. Could/Won’t Requirements    1. Operational Requirements       1. The system will work on Linux and Mac computers (D)    2. Performance Requirements    3. Security Requirements       1. The program will have salted sensitive information (F)    4. Cultural and Political Requirements       1. The program will be fully compliant with payroll law (RD) |

# Use Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:** Get Payroll Information | | | **ID Number:** 1 | | |
| **Short Description:** The system receives the payroll information from the user. | | | | | |
| **Trigger:** The user hits the “Compile Payroll” button  **Type:** External | | | | | |
| **Major Inputs:**  Description   1. Employees 2. Hours | Source   1. Database 2. User | **Major Outputs:**  Description   1. Payroll Employee List | | Destination   1. Use Case 2 | |
| **Major Steps Performed:**   1. The system retrieves a list of all employees and their information. 2. The system displays a prompt of all the employees (names and id). 3. The user selects the employees from the list for the current payroll. 4. The user confirms the selected choices. 5. The system displays a prompt for hours worked for each hourly employee. 6. The user enters the hours worked for each employee. | | | | |
| **Information for Steps/Alternate Flows:**   1. The user must select at least 1 employee or cancel the payroll. 2. The number of hours must be greater than 0 and can have 2 decimal places. | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:** Compile Pay Stubs and NACHA Files | | | **ID Number:** 2 | | |
| **Short Description:** The system compiles the pay stubs and NACHA files. | | | | | |
| **Trigger:** Get Payroll Information use case finishes  **Type:** External | | | | | |
| **Major Inputs:**  Description   1. Payroll Employee List 2. Tax Rates 3. Company Information | Source   1. Use Case 1 2. Database 3. Database | **Major Outputs:**  Description   1. Pay Stub PDF 2. NACHA File 3. Success/Fail Message | | Destination   1. User Chosen Destination 2. User Chosen Destination 3. Screen | |
| **Major Steps Performed:**   1. The system gets the save location for the PDF (for the pay stubs) from the user. 2. The system queries the database for the FICA/Med tax rates and state tax rates (federal tax rate for each employee is included in the Payroll Employee List). 3. The system calculates each tax, other deductions, and gross and net pay. 4. The system builds a pay stub page for each employee in the list in the PDF. 5. The system gets the save location for the NACHA file from the user. 6. The system builds the NACHA file. 7. The system displays a message to the user telling him/her that the files were successfully created and saved. | | | | |
| **Information for Steps/Alternate Flow:**   1. The NACHA file will follow the format here: <https://www.treasurysoftware.com/ach/how-do-i-create-an-ach-nacha-file-detailed-file-format.aspx> 2. If unable to build the files or the user does not choose a valid location, quit the process, and return to the previous screen. | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:** Add Employee to the Database | | | **ID Number:** 3 | | |
| **Short Description:** A privileged user adds a new employee to the database. | | | | | |
| **Trigger:** The user clicks the “Add New Employee” button  **Type:** External | | | | | |
| **Major Inputs:**  Description   1. Employee Information | Source   1. User | **Major Outputs:**  Description   1. Insert Query 2. Success/Fail Message | | Destination   1. Database 2. Screen | |
| **Major Steps Performed:**   1. The system displays the add-employee screen. 2. The user enters the necessary information into the boxes. 3. The system prompts the user with a confirmation message.    1. If confirmed, the insert query is sent to the database, and the user is returned to the add-employee screen.    2. If declined, any information the user entered is not saved, and the user is returned to the admin screen. 4. The system displays a message of success or failure to the user. | | | | |
| **Information for Steps/Alternate Flow:**   1. A user can only add one user at a time. 2. The system will validate user input for correct formats and duplicate employee IDs before sending it to the database. If problem is found, tell the user the problem and let the user fix any mistakes. 3. Most of the employee information will be required. | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:** Login Process | | | **ID Number:** 4 | | |
| **Short Description:** A user logs into the application with a Username and Passcode. | | | | | |
| **Trigger:** The user launches the application, and a login window appears  **Type:** External | | | | | |
| **Major Inputs:**  Description  1. Username (ID)  2. Passcode | Source   1. User 2. User | **Major Outputs:**  Description   1. Login Unsuccessful Message 2. Main menu screen | | Destination   1. Screen 2. Screen | |
| **Major Steps Performed:**   1. The user enters his/her username and passcode. 2. The user hits the “Login” button. 3. The system queries the database to confirm if login credentials are valid.    1. The system displays a login successful or login unsuccessful message. 4. The system shows each user information based on their privileges. | | | | |
| **Information for Steps/Alternate Flow:**   1. A user must have a previous account to login into the application. 2. If the user’s information is incorrect, they do not advance beyond the login screen, and the system displays a login unsuccessful message. | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:** Delete Employee from the Database | | | **ID Number:** 5 | | |
| **Short Description:** The system deletes selected employee(s) from the database. | | | | | |
| **Trigger:** The user clicks the “Delete Employee” button  **Type:** External | | | | | |
| **Major Inputs:**  Description   1. Employees 2. Selected Employee(s) | Source   1. Database 2. User | **Major Outputs:**  Description   1. Deletion query 2. Deletion confirmation | | Destination   1. Database 2. Screen | |
| **Major Steps Performed:**   1. The system retrieves a list of all employees and their information from the database. 2. The system displays that list to the user. 3. The user selects employees from the list for deletion. 4. The system displays a confirmation prompt for the user to confirm/decline.    1. If confirmed, the system sends the deletion query to the database and returns to the main screen.    2. If declined, the user is taken back to the employee-deletion select screen. 5. The system displays either a success or error message for the deletion query. | | | | |
| **Information for Steps/Alternate Flow:**   1. The user must select at least 1 employee or cancel the deletion. 2. The user cannot delete the only employee who has permission for payroll. | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:** Edit Employee Information in the Database | | | **ID Number:** 6 | | |
| **Short Description:** A privileged user edits an employee to the database. | | | | | |
| **Trigger:** A privileged user clicks the “Edit Employee” button  **Type:** External | | | | | |
| **Major Inputs:**  Description   1. Employee ID 2. Employee Information | Source   1. User 2. User | **Major Outputs:**  Description   1. Confirmation message 2. Update Query | | Destination   1. Screen 2. Database | |
| **Major Steps Performed:**   1. The system launches the edit-employee screen. 2. The user enters in the employee ID to change. 3. The user edits the information. 4. The system prompts the user with a confirmation message.    1. If confirmed, the update query is sent to the database, and the user is returned to the edit-employee screen.    2. If declined, any information the user entered is not saved, and the user is returned to the admin screen. 5. The system displays a message of success or failure to the user depending on if the update query was successful. | | | | |
| **Information for Steps/Alternate Flow:**   1. Validate user input for correct formats and to ensure they are not empty. 2. Most of the employee information will be required. 3. The user cannot update the employee ID but can edit any other piece of information. 4. If user enters in invalid ID in step 2, display message indicating so. | | | | |

# User Stories

Compile Payroll

*As a payroll manager, I select the employees for payroll from a list that is displayed by the program. For each hourly employee, I enter in the number of hours he/she worked in the pay cycle. When I finish entering in the hours, the system should build and save the pay stubs and NACHA file to the location I specified.*

Login

*As a user, I will start the application. Upon launch, I will be presented with a login screen and will enter my username and password. Based on my credentials provided, I will be able to access the payroll functions on the main screen or I will not be able to. If I accidentally mistype my username or password, I should not be directed to the main screen but instead be urged to reenter them.*