

Rule-Based Decision Tree Chatbot with AI

Sharan Baskaran
ChainSys
Madurai

Overview

This documentation provides a comprehensive overview of the functionality and implementation of a chatbot application that manages leave applications, service request, desktop support. It also includes email notification handling for leave application and error management.

Decision Tree - Logic for Leave Application Process:

1. Node1

- **Type:** Input
- **Text:** "Hi! I am Chainbot, an online assistant for the employees!"
- **Purpose:** Greet the user and introduce the chatbot.

2. Node2

- **Type:** Question
- **Text:** "Welcome to our leave application service"
- **Answers:**
 - **"Leave form!"** – Moves to Node 3 for leave form processing.
 - **"Check Leave balance"** – Moves to Node 3 (likely intended to check balance but leads to the same processing as leave form).
 - **"Cancel"** – Moves to Node12 to exit the leave application process.
- **Purpose:** Present options to the user regarding leave application or checking balance.

3. Node3

- **Type:** Input
- **Text:** "Please provide your employee ID"
- **Purpose:** Collect the employee ID to proceed with the leave application process.

4. NodeCheckEligibility

- **Type:** Statement
- **Text:** "Checking your eligibility..."
- **Purpose:** Inform the user that their eligibility for leave is being checked.

5. NodeEligibilityResult

- **Type:** Question
- **Text:** "Hello [employeeName], you are eligible to apply for leave. Your available leave balance is [leaveBalance], Casual Leave Balance: [casualLeaveBalance], Medical Leave Balance: [medicalLeaveBalance]."
- **Answers:**
 - **"Apply!"** – Moves to Node4 to start the leave application.
 - **"Cancel"** – Moves to Node13 to end the process.
- **Purpose:** Inform the user of their leave eligibility and balances, and provide options to apply for leave or cancel.

6. Node4

- **Type:** Input
- **Text:** "Type of leave"
- **Purpose:** Collect the type of leave the employee is applying for.

7. Node5

- **Type:** Input
- **Text:** "Reason for the leave"
- **Purpose:** Collect the reason for the leave application.

8. Node6

- **Type:** Input
- **Text:** "No. of days"
- **Purpose:** Collect the number of days the employee wants to take leave.

9. Node7

- **Type:** Input
- **Text:** "Please provide the start date for your leave. (Format: YYYY-MM-DD)"
- **Purpose:** Collect the start date of the leave.

10. Node8

- **Type:** Input
- **Text:** "Please provide the end date for your leave. (Format: YYYY-MM-DD)"
- **Purpose:** Collect the end date of the leave.

11. Node9

- **Type:** Question
- **Text:** "Are you sure you want to submit the form?"
- **Answers:**
 - **"Yes, submit"** – Moves to Node10 to confirm submission.
 - **"No, make changes"** – Moves back to Node4 to modify the leave details.
- **Purpose:** Confirm the user's intent to submit the leave application.

12. Node10

- **Type:** Question
- **Text:** "Your leave application has been successfully submitted! You will receive a confirmation email shortly. Is there anything else I can help you with?"
- **Answers:**
 - **"Yes"** – Moves to Node12 for further assistance.
 - **"No"** – Moves to Node13 to end the interaction.
- **Purpose:** Confirm successful submission of the leave application and offer additional help.

13. Node11

- **Type:** Question
- **Text:** "Your leave balance is 0. Please note that your leave will be marked as loss of pay."
- **Answers:**
 - **"Ok"** – Moves to Node16 to start the leave application.
 - **"No"** – Moves to Node13 to end the process.
- **Purpose:** Inform the user if their leave balance is zero and the leave will be marked as loss of pay, and provide options to proceed or cancel.

14. Node12

- **Type:** Question
- **Text:** "Ok! Do you need any other assistance?"
- **Answers:**
 - **"Apply leave"** – Moves to Node2 to restart the leave application process.
 - **"Service Request"** – Moves to Node14 to switch to Service Request options.
 - **"Desktop Support"** – Moves to NodeDesktop to switch to Desktop Support options.
 - **"No"** – Moves to Node13 to end the interaction.
- **Purpose:** Offer additional assistance or options to the user.

15. Node13

- **Type:** Input
- **Text:** "Thank You"
- **Purpose:** End the interaction with a thank you message.

Decision Tree - Logic for Bug Request Process:

1. NodeBug

Type: Input

Text: "Please enter your employee ID to start the Bug report process."

Purpose: Collect the employee ID to begin the bug report process.

2. NodeBugTitle

Type: Input

Text: "Please provide a title for the Bug report."

Purpose: Collect the title of the bug report.

3. NodeBugClient

Type: Dropdown

Text: "Select the client associated with the bug."

Options: [List of clients]

Purpose: Identify the client related to the bug.

4. NodeBugInstanceName

Type: Input

Text: "Please enter the Instance Name where the bug was found."

Purpose: Collect the instance name related to the bug.

5. NodeBugProductTeam

Type: Input

Text: "Specify the Product or Team associated with the bug."

Purpose: Identify the product or team affected by the bug.

6. NodeBugComponent

Type: Input

Text: "Describe the Component & Sub Component where the bug occurred."

Purpose: Collect details about the component and sub-component.

7. NodeBugSeverity

Type: Dropdown

Text: "Select the severity level of the bug."

Options: [Trivial, Minor, Major, Critical]

Purpose: Assess the severity of the bug.

8. **NodeBugPriority**

Type: Dropdown

Text: "Select the priority level of the bug."

Options: [Low, Medium, High, Urgent]

Purpose: Determine the priority of the bug.

9. **NodeBugPhase**

Type: Dropdown

Text: "Select the phase where the bug was found."

Options: [Production, System Testing, Unit Testing]

Purpose: Identify the phase during which the bug was discovered.

10. **NodeBugFoundIn**

Type: Dropdown

Text: "Specify where the bug was found."

Options: [Product, Code Review, Design, Requirement, Test Scenario, User Manual, Test Review]

Purpose: Determine the context in which the bug was found.

11. **NodeBugIdentifiedBy**

Type: Dropdown

Text: "Was the bug identified by appBots?"

Options: [Yes, No]

Purpose: Specify if the bug was identified by automated systems.

12. **NodeBugTestEnvironment**

Type: Input

Text: "Provide details about the Test Environment where the bug was observed."

Purpose: Collect information about the test environment.

13. **NodeBugAssignedTo**

Type: Input

Text: "Enter the name of the person or team assigned to address the bug."

Purpose: Specify who is responsible for fixing the bug.

14. **NodeBugCreatedFor**

Type: Input

Text: "Describe the vulnerability that the bug was created for."

Purpose: Provide details about the vulnerability addressed by the bug.

15. **NodeBugPlannedReleaseVersion**

Type: Input

Text: "Enter the Planned Release Version for fixing the bug."

Purpose: Specify the version in which the bug is planned to be fixed.

16. NodeBugDetailedDescription

Type: Input

Text: "Provide a detailed description of the bug."

Purpose: Collect a comprehensive description of the bug.

17. NodeBugStepsToReproduce

Type: Input

Text: "List the steps to reproduce the bug."

Purpose: Detail the steps required to reproduce the bug.

18. NodeBugExpectedResult

Type: Input

Text: "What is the expected result?"

Purpose: Define what the expected outcome should be.

19. NodeBugActualResult

Type: Input

Text: "What is the actual result?"

Purpose: Describe what actually happened.

20. NodeBugScreenshot

Type: Upload

Text: "Please upload a screenshot if available."

Purpose: Collect any relevant screenshots.

21. NodeBugConfirm

Type: Question

Text: "Are you sure you want to submit the Bug report?"

Answers:

- "Submit" – Moves to Node12 to conclude the process.
- "Cancel" – Moves to Node12.

Purpose: Confirm the submission of the bug report.

Decision Tree - Logic for Change Request Process:

1. NodeChangeRequest

Type: Input

Text: "Please enter your employee ID to start the Change Request process."

Purpose: Collect the employee ID to begin the change request process.

2. NodeChangeRequestTitle

Type: Input

Text: "Please provide a title for the Change Request."

Purpose: Collect the title of the change request.

3. NodeChangeRequestClient

Type: Dropdown

Text: "Select the client associated with the change request."

Options: [List of clients]

Purpose: Identify the client related to the change request.

4. NodeChangeRequestInstanceName

Type: Input

Text: "Please enter the Instance Name related to the change request."

Purpose: Collect the instance name for the change request.

5. NodeChangeRequestProductTeam

Type: Input

Text: "Specify the Product or Team associated with the change request."

Purpose: Identify the product or team affected by the change request.

6. NodeChangeRequestComponent

Type: Input

Text: "Describe the Component & Sub Component related to the change request."

Purpose: Collect details about the component and sub-component.

7. NodeChangeRequestSeverity

Type: Dropdown

Text: "Select the severity level of the change request."

Options: [Trivial, Minor, Major, Critical]

Purpose: Assess the severity of the change request.

8. **NodeChangeRequestPriority**

Type: Dropdown

Text: "Select the priority level of the change request."

Options: [Low, Medium, High, Urgent]

Purpose: Determine the priority of the change request.

9. **NodeChangeRequestIdentifiedBy**

Type: Dropdown

Text: "Was the change request identified by appBots?"

Options: [Yes, No]

Purpose: Specify if the change request was identified by automated systems.

10. **NodeChangeRequestTestEnvironment**

Type: Input

Text: "Provide details about the Test Environment related to the change request."

Purpose: Collect information about the test environment.

11. **NodeChangeRequestAssignedTo**

Type: Input

Text: "Enter the name of the person or team assigned to handle the change request."

Purpose: Specify who is responsible for addressing the change request.

12. **NodeChangeRequestCreatedFor**

Type: Input

Text: "Describe the vulnerability that the change request is related to."

Purpose: Provide details about the vulnerability addressed by the change request.

13. **NodeChangeRequestPlannedReleaseVersion**

Type: Input

Text: "Enter the Planned Release Version for implementing the change request."

Purpose: Specify the version in which the change is planned to be implemented.

14. **NodeChangeRequestDetailedDescription**

Type: Input

Text: "Provide a detailed description of the change request."

Purpose: Collect a comprehensive description of the change request.

15. **NodeChangeRequestScreenshot**

Type: Upload

Text: "Please upload a screenshot if available."

Purpose: Collect any relevant screenshots.

16. NodeChangeRequestConfirm

Type: Question

Text: "Are you sure you want to submit the Change Request?"

Answers:

- "Submit" – Moves to Node12 to conclude the process.
- "Cancel" – Moves to Node12.

Purpose: Confirm the submission of the change request.

Decision Tree - Logic for Support Request Process

1. **NodeSupportRequest**

Type: Input

Text: "Please enter your employee ID to start the Support Request process."

Purpose: Collect the employee ID to begin the support request process.

2. **NodeSupportRequestTitle**

Type: Input

Text: "Please provide a title for the Support Request."

Purpose: Collect the title of the support request.

3. **NodeSupportRequestClient**

Type: Dropdown

Text: "Select the client associated with the support request."

Options: [List of clients]

Purpose: Identify the client related to the support request.

4. **NodeSupportRequestInstanceName**

Type: Input

Text: "Please enter the Instance Name related to the support request."

Purpose: Collect the instance name for the support request.

5. **NodeSupportRequestProductTeam**

Type: Input

Text: "Specify the Product or Team associated with the support request."

Purpose: Identify the product or team affected by the support request.

6. **NodeSupportRequestComponent**

Type: Input

Text: "Describe the Component & Sub Component related to the support request."

Purpose: Collect details about the component and sub-component.

7. NodeSupportRequestSeverity

Type: Dropdown

Text: "Select the severity level of the support request."

Options: [Trivial, Minor, Major, Critical]

Purpose: Assess the severity of the support request.

8. NodeSupportRequestPriority

Type: Dropdown

Text: "Select the priority level of the support request."

Options: [Low, Medium, High, Urgent]

Purpose: Determine the priority of the support request.

9. NodeSupportRequestIdentifiedBy

Type: Dropdown

Text: "Was the support request identified by appBots?"

Options: [Yes, No]

Purpose: Specify if the support request was identified by automated systems.

10. NodeSupportRequestTestEnvironment

Type: Input

Text: "Provide details about the Test Environment related to the support request."

Purpose: Collect information about the test environment.

11. NodeSupportRequestAssignedTo

Type: Input

Text: "Enter the name of the person or team assigned to handle the support request."

Purpose: Specify who is responsible for addressing the support request.

12. NodeSupportRequestCreatedFor

Type: Input

Text: "Describe the vulnerability that the support request is related to."

Purpose: Provide details about the vulnerability addressed by the support request.

13. NodeSupportRequestPlannedReleaseVersion

Type: Input

Text: "Enter the Planned Release Version for addressing the support request."

Purpose: Specify the version in which the support is planned to be addressed.

14. NodeSupportRequestDetailedDescription

Type: Input

Text: "Provide a detailed description of the support request."

Purpose: Collect a comprehensive description of the support request.

15. NodeSupportRequestScreenshot

Type: Upload

Text: "Please upload a screenshot if available."

Purpose: Collect any relevant screenshots.

16. NodeSupportRequestConfirm

Type: Question

Text: "Are you sure you want to submit the Support Request?"

Answers:

- "Submit" – Moves to Node12 to conclude the process.
- "Cancel" – Moves to Node12.

Purpose: Confirm the submission of the support request.

Decision Tree - Logic for Desktop Support Process:

1. NodeDesktopSupport

Type: Input

Text: "Please enter your employee ID to start the Desktop Support process."

Purpose: Collect the employee ID to begin the desktop support request process.

2. Node Desktop Support Module

Type: Dropdown

Text: "Select the module for which you need desktop support."

Options: [Software, Hardware]

Purpose: Identify the module (software or hardware) requiring desktop support.

3. Node Desktop Support Title

Type: Input

Text: "Please provide a title for the Desktop Support issue."

Purpose: Collect the title of the desktop support issue.

4. NodeDesktopSupportContact

Type: Input

Text: "Provide a contact number where you can be reached."

Purpose: Collect the contact number to reach the employee.

5. NodeDesktopSupportLocation

Type: Dropdown

Text: "Select your location."

Options: [Chennai, Madurai]

Purpose: Identify the employee's location for desktop support.

6. NodeDesktopSupportExpectedDate

Type: Input

Text: "Enter the expected date for the support request."

Purpose: Collect the expected date when desktop support is needed.

7. NodeDesktopSupportDescription

Type: Input

Text: "Provide a detailed description of the problem."

Purpose: Collect a comprehensive description of the desktop support issue.



8. Node Desktop Support Screenshot

Type: Upload

Text: "Please upload a screenshot if available."

Purpose: Collect relevant screenshots to better understand the issue.

9. NodeDesktopSupportConfirm

Type: Question

Text: "Are you sure you want to submit the Desktop Support form?"

Answers:

"Submit" – Moves to Node12 to conclude the process.

"Cancel" – Moves to Node12.

Purpose: Confirm the submission of the desktop support form.

ChatbotComponent (Purpose, Logic):

ngOnInit()

- **Purpose:** Initialize the component.
- **Logic:**
 - Fetch the decision tree configuration from `couchdbService`.
 - Set the initial node based on the decision tree's start node.
 - Add the initial bot message to the conversation.
 - Add an event listener to scroll to the bottom of the chat content.

ngAfterViewChecked()

- **Purpose:** Ensure the chat view is always scrolled to the bottom.
- **Logic:** Call `scrollToBottom()` to maintain the scroll position.

addMessage(speaker: string, text: string)

- **Purpose:** Add a message to the conversation.
- **Logic:**
 - Push the message to the `conversation` array.
 - Call `scrollToBottom()` to ensure the latest message is visible.

selectAnswer(answer: Answer)

- **Purpose:**

This function handles the user's answer selection, processes it, and triggers specific actions based on the user's choice.
- **Logic:**
 1. **Record User's Answer:**
 - The function starts by recording the user's response in the conversation using `this.addMessage('user', answer.text)`.
 2. **Handle Leave Type Selection:**

- If the current node's text is "Type of leave", the selected leave type is stored in `this.Details['Type of leave']`.
- The function then validates the leave type using `this.validateLeaveType(answer.text)`. If the validation fails, the function exits early without further processing.

3. Handle Form Submissions:

- If the user's answer is "Submit", the function sets the `submitted` flag to `true` to indicate the form submission process has started.
- Based on the current node's text, the function triggers the appropriate form submission method:
 - **Leave Application Form:** If the current node's text is "Are you sure you want to submit the form?", it calls `this.sendLeaveApplicationEmail()`.
 - **Bug Report Form:** If the text is "Are you sure you want to submit the Bug report form?", it calls `this.submitBugTicket()`.
 - **Change Request Form:** If the text is "Are you sure you want to submit the Change Request form?", it triggers `this.submitChangeRequest()`.
 - **Service Request Form:** If the text is "Are you sure you want to submit the Service Request form?", it calls `this.submitServiceRequest()`.
- After handling form submission, the function exits early to prevent any further processing.

4. Handle Other Answers:

- If the answer is not "Submit", the `submitted` flag is reset to `false`.
- The function then updates the `currentNode` to the next node based on `answer.next` and calls `this.processNode()` to continue the conversation flow.

submitInput()

- **Purpose:** Handle input submission from the user.
- **Logic:**
 - Validate user input based on the current question.
 - If input is valid, update `leaveDetails` and process it.
 - Handle specific cases like start date and end date validations for leave.

checkOverlappingLeaveDates(employeeId: string, startDate: string, leaveDays: number)

- **Purpose:** Check for overlapping leave dates.
- **Logic:**
 - Fetch existing leave records for the employee.
 - Determine if the new leave overlaps with any existing leave.
 - If there is an overlap, show an error message; otherwise, proceed.

validateLeaveDays(days: string)

- **Purpose:**

This function validates the number of leave days requested by the user.
- **Logic:**
 1. **Numeric Validation:**
 - The input is parsed into an integer.
 - If the input is not a valid number, an error message is displayed: "Please enter a valid number for leave days."
 - If the number of leave days is less than or equal to 0, an error message is displayed: "Number of days must be greater than 0."
 2. **Leave Balance Check:**
 - Based on the type of leave selected (`Casual Leave` or `Medical Leave`), the function checks if the requested leave days exceed the available balance.
 - If the requested days exceed the available balance, an error message is displayed informing the user of the remaining balance, and the user is redirected to a specific node (`node6`) to reselect or reconsider their input.

- If the balance is 0 and the user selects a leave type, the function considers it a "Loss of Pay" leave, updates the leave details, and returns **true**.

3. Fallback Handling:

- If the leave type is not recognized or an unexpected error occurs, an error message is displayed: "Invalid leave type or unexpected error."
- The function returns **false** if any of these conditions are met.

4. Final Update:

- If all validations pass, the leave details are updated with the number of days, and the function returns **true**.

validateLeaveType(type: string)

- **Purpose:**

This function validates the type of leave selected by the user.

- **Logic:**

1. Balance Check:

- The function checks if the selected leave type has a sufficient balance.
- If the balance is 0 for **Casual Leave**, an error message is displayed: "You have 0 casual leave days available. Please select another type of leave."
- Similarly, if the balance is 0 for **Medical Leave**, an error message is displayed: "You have 0 medical leave days available. Please select another type of leave."

2. Return Value:

- The function returns **false** if the balance for the selected leave type is 0, prompting the user to select a different leave type.
- If the selected leave type has a sufficient balance, the function returns **true**.

validateDate(date: string)

- **Purpose:** Validate date format and range.
- **Logic:**
 - Ensure the date is in **YYYY-MM-DD** format.
 - Check if the date is within the valid range (e.g., The year must be the current year, month should be between 01 to 12, date should be between 01 to 31).

checkEmployeeEligibility(employeeId: string)

- **Purpose:** Check if the employee is eligible for leave.
- **Logic:**
 - Fetch employee and leave data using the provided employee ID.
 - Determine eligibility based on leave balance and eligibility criteria.
 - Update the decision tree node based on the eligibility.

async processUserInput(input: string)

- **Purpose:** To process user input, correct spelling errors, and map the corrected input to the appropriate decision tree node.
- **Logic:**
 1. **Correct Spelling Errors:**
 - a. Start with the original input text.
 - b. Checks if a corrected version of the input is already in the cache using **this.geminiService.getCorrectionCache()**.
 - c. If a cached correction is found, it uses that corrected text.
 - d. If no cached correction is found and the decision tree has not yet been entered, it utilizes the **suggestCorrection** method of **GeminiService** to correct spelling errors.
 - e. If an error occurs during correction, it falls back to the original input.
 2. **Map to Next Node:**
 - a. Uses predefined keyword mappings to determine the next node in the decision tree.
 - b. Keywords and their corresponding nodes are defined in **keywordMappings**.

- c. Normalizes the corrected input by converting it to lowercase and trimming whitespace for accurate keyword matching.
 - d. Checks if the normalized corrected input contains any predefined keywords and maps it to the corresponding node.
3. Handle No Match:
 - a. If no keywords are matched, it defaults to node 'node15'.
4. Decision Tree Management:
 - a. Marks that the decision tree has been entered if it hasn't already.
 - b. Resets the flag if the next node is 'node13' or 'node12'.
5. Process Node:
 - a. Updates the **currentNode** based on the mapped next node.
 - b. Calls **processNode()** to handle further processing based on the updated node.

Features:

- Spelling Correction: Uses **GeminiService** to correct spelling errors and cache corrections.
- Keyword Mapping: Maps user input to decision tree nodes based on predefined keywords.
- Fallback Handling: Provides a fallback mechanism to handle errors and unmatched keywords.
- Decision Tree Management: Handles decision tree state and node processing.

processNode()

- **Purpose:** Process the current node in the decision tree.
- **Logic:**
 - Skip prompts if the employee ID is already stored.
 - Update answers based on the current node type and text.
 - Add the bot's message to the conversation.

replaceVariables(text: string)

- **Purpose:** Replace placeholders in the text with actual values from **Details**.
- **Logic:**
 - Replace placeholders in the format **[key]** with the corresponding value from **Details**.

sendLeaveApplicationEmail()

Purpose: This method handles the process of sending a leave application email to the respective manager using EmailJS. It also fetches employee data from CouchDB, prepares email parameters, and handles success or failure responses.

Key Steps:

1. Extract Employee ID and Leave Details:

- Retrieves the employee ID from `this.Details` using the key 'Please provide your employee ID'.

2. Fetch Employee Data:

- Constructs the URL for the employee data using the extracted employee ID.
- Sets up HTTP headers with Basic Authentication for CouchDB access.
- Sends a **GET** request to CouchDB to fetch the employee data.
- Handles the HTTP response to extract employee details such as name, email, and manager's email.

3. Prepare Email Parameters:

- Creates an object `templateParams` that contains the following details:
 - `employee_name`: The employee's name.
 - `employee_email`: The employee's email address.
 - `manager_email`: The manager's email address.
 - `employee_id`: The employee ID.
 - `leave_type`: Type of leave, defaulting to 'Loss of Pay' if not specified.
 - `leave_reason`: The reason for the leave.
 - `leave_days`: The number of leave days.
 - `start_date`: Start date of the leave in YYYY-MM-DD format.
 - `end_date`: End date of the leave in YYYY-MM-DD format.

4. Send Email Using EmailJS:

- Defines a `sendEmail` function to send emails using EmailJS with provided service ID, template ID, and public key.
- Attempts to send the email using the primary service ID, template ID, and public key.

- **On Success:** Logs success message and status. Calls `updateLeaveBalance()` to update the leave balance with the number of leave days.
- **On Failure:** Logs error and retries sending the email using a secondary service ID, template ID, and public key.
 - **On Success (Retry):** Logs success message and status. Calls `updateLeaveBalance()` to update the leave balance.
 - **On Failure (Retry):** Logs error and does not perform any additional actions.

5. Error Handling:

- **If Employee Data is Not Found:** Logs an error message indicating no employee data was found.
- **If Fetching Employee Data Fails:** Logs an error message indicating the error during the data fetch operation.

`updateLeaveBalance(employeeId: string, leaveDays: string)`

Purpose: Updates the leave balance for a specific employee and handles different leave types.

Key Steps:

1. Retrieve Employee Data:

- Sends a `GET` request to fetch the employee's current leave balance.

2. Check Leave Balance:

- Verifies if there is enough leave balance based on the type of leave (Casual or Medical) and whether there is a loss of pay.

3. Update Leave Balance:


- Deducts the leave days from the balance if applicable.

4. Create and Save Leave Document:

- Generates a unique leave document and saves it using a `PUT` request.

5. Handle Success or Failure:

- On success, updates the leave balance in the database and navigates to the appropriate node.
- On failure, displays an error message and navigates to an error node.



saveTicketDocument(ticketDocument: any, url: string, headers: HttpHeaders, ticketId: string): void

Purpose: Saves a ticket document to CouchDB, including handling the addition of any file attachments.

Key Steps:

1. Save Ticket Document:

- Sends a PUT request to save the ticket document at the specified URL with the provided headers.

2. Handle Success:

- On successful save, logs a success message, notifies the user with the ticket ID, and navigates to the appropriate node.

3. Handle Failure:

- On failure, logs an error message and notifies the user about the failure, then navigates to an error node.

handleFileInput(event: Event): void

Purpose: Handles the file input event to allow file selection.

Key Steps:

1. Select File:

- Retrieves the selected file from the input event and stores it for further processing.

uploadScreenshot(): void

Purpose: Uploads a selected screenshot as a base64 encoded image and moves to the next process node.

Key Steps:

1. Check File Selection:

- Verifies if a file has been selected for upload.

2. Encode File to Base64:

- Reads the selected file as a base64 encoded string using FileReader.

3. Move to Next Node:

- Proceeds to the next node in the decision tree after encoding the file.

submitBugTicket(): void

Purpose: Submits a bug ticket with the relevant details and optional screenshot attachment.

Key Steps:

1. Prepare Ticket Data:

- Collects and organizes the bug ticket details from user input.

2. Generate Document ID:

- Creates a unique ID for the ticket document.

3. Prepare Ticket Document:

- Constructs the ticket document including all details and optionally adds a screenshot attachment if selected.

4. Save Ticket Document:

- Saves the prepared ticket document to CouchDB and handles both scenarios where a screenshot is included or not.

submitChangeRequest(): void

Purpose: Submits a change request ticket with relevant details and optional screenshot attachment.

Key Steps:

1. Prepare Ticket Data:

- Collects and organizes the change request details from user input.

2. Generate Document ID:

- Creates a unique ID for the change request document.

3. Prepare Ticket Document:

- Constructs the change request document including all details and optionally adds a screenshot attachment if selected.

4. Save Ticket Document:

- Saves the prepared ticket document to CouchDB and handles both scenarios where a screenshot is included or not.

submitSupportRequest(): void

Purpose: Submits a service request ticket with relevant details and optional screenshot attachment.

Key Steps:

1. Prepare Ticket Data:

- Collects and organizes the service request details from user input.

2. Generate Document ID:

- Creates a unique ID for the service request document.

3. Prepare Ticket Document:

- Constructs the service request document including all details and optionally adds a screenshot attachment if selected.

4. Save Ticket Document:

- Saves the prepared ticket document to CouchDB and handles both scenarios where a screenshot is included or not.

toggleMinimize(event?: Event)

Purpose: Toggles the minimization of the chat window.

Key Steps:

1. Toggle State:

- Changes the **isMinimized** flag to show or hide the chat window.



closeChat(event?: Event)

Purpose: Closes the chat window with a confirmation prompt.

Key Steps:

1. **Confirm Close:**

- Displays a confirmation dialog before closing the chat.

scrollToBottom()

Purpose: Automatically scrolls the chat content to the bottom if enabled.

Key Steps:

1. **Scroll Check:**

- Scrolls to the bottom if **autoScrollEnabled** is true.

onUserScroll()

Purpose: Handles user-initiated scrolling and updates the auto-scroll state.

Key Steps:

1. **Update Scroll State:**

- Updates **autoScrollEnabled** based on user scroll position.

displayErrorMessage(message: string)

Purpose: Displays an error message to the user.

Key Steps:

1. **Add Message:**

- Uses **addMessage()** to display the error message.

EmailService

Purpose: Manages email notifications using EmailJS.

Key Methods:

- **sendEmail(templateParams: any):** Sends an email with the provided parameters using EmailJS.

GeminiService

Purpose:

The **GeminiService** integrates with Google Generative AI to provide text correction suggestions, manage rate limiting, and maintain a cache for storing results. It also tracks message history using a reactive data store.

Key Methods:

1. Service Initialization:

- **GoogleGenerativeAI Instance:** The service initializes the **GoogleGenerativeAI** instance using an API key. This instance is used to interact with the Gemini model for text generation and correction.
- **Cache System:** A private static object (**correctionCache**) is used to store the corrected text results. The cache is also persisted across browser sessions using **localStorage**.
- **Rate Limiting Parameters:** The service tracks API requests to prevent exceeding the predefined request limit within a set time frame. It uses **lastRequestTime**, **requestCount**, **REQUEST_LIMIT**, and **TIME_FRAME** variables to manage rate limiting.
- **Message History:** A **BehaviorSubject** is used to store and observe the message history. The context window size, which defines how many messages to consider, is set to 100.

2. Suggest Correction:

- **Cache Lookup:** Before making a request to the API, the service checks if the input has already been corrected and stored in the cache (**correctionCache**).
- **Rate Limiting:** If the input is not cached, the service checks whether the rate limit has been exceeded. If necessary, it waits before sending a request to the API.
- **API Request:** The input is sent to the Gemini AI model with a prompt asking for text correction. The response is cleaned up (e.g., removing asterisks) and then stored in both the global cache and **localStorage**.

- **Cache Management:** The corrected result is cached to avoid redundant API calls for the same input.
- **Rate Limit Tracking:** After processing, the service updates the rate limit counters.

3. Rate Limiting:

- **handleRateLimit():** This method manages the rate limit by pausing further requests if the number of requests within the set time frame exceeds the allowed limit. It waits for the necessary time before resuming.
- **updateRateLimit():** This method resets or updates the rate limit counters based on the elapsed time since the last request.

4. Cache Management:

- **loadCache():** Loads the cached corrections from **localStorage** during service initialization to ensure persistence across browser sessions.
- **saveCache():** Saves the current cache to **localStorage** whenever a new correction is added.

5. Message History:

- **getMessageHistory():** Provides an observable to retrieve the message history. The observable allows components to reactively listen to changes in the message history. The service currently maintains a context window of the last 100 messages.

To get API KEY:

https://aistudio.google.com/app/apikey?_gl=1*qqdls*_ga*NTczMzU3MDAzLjE3MjMxMDY2OTI.*_ga_P1DBVKWT6V*MTcyMzgxmDczMC41LjAuMTcyMzgxmDczMC42MC4wLjE0MjI5NTcxMTA

CouchdbService

Purpose: Provides methods to interact with CouchDB for various data operations.

Key Methods:

- **getDecisionTree():** Fetches the decision tree from CouchDB.
- **getUsers():** Retrieves user details for authentication.
- **authenticate(username: string, password: string):** Validates user credentials.
- **getAllEmployeeLeaves():** Fetches all leave records for employees.
- **getEmployeeDetails():** Retrieves details of all employees.

CouchDB Urls

1. Decision Tree - <https://192.168.57.185:5984/decison-tree-db/decisiontree> 12
2. Employee Database - https://192.168.57.185:5984/employee-db/employee_2_{employeeId}
3. Leave Record Database - https://192.168.57.185:5984/employee-db/leave_2_{employeeId}
4. User credentials - <https://192.168.57.185:5984/decison-tree-db/users>

Home Component:

1. **Home:**

The central point of the dashboard, the **Home** section allows you to return to the main dashboard view. From here, you can see all available options and navigate to different features of the home dashboard and how to use ChainBot .

2. **My Details:**

Allows users to view their personal employee information.

3. **Leave Menu:** This button toggles a dropdown for leave-related options:

- a. **Leave Details:** Users can track their leave balances, including Casual and Medical Leave.
- b. **Leave Report:** Users can review detailed records of their leave usage, organized by year and including types and reasons.

4. **Service Requests:**

This option provides access to service requests, where users can view their SR IDs and data.

5. **Desktop Support:**

This option provides access to support tickets related to desktop issues.

6. **Logout:** An option to securely exit the application.

7. **ChainBot:** Chainbot, Online Assistant. I'm here to help you with Leave Application, Desktop Support and Service Requests..

Admin Component:

1. Home:

- a. The central point of the dashboard, the **Home** section allows you to return to the main dashboard view. From here, you can see all available options and navigate to different features of the admin dashboard.

2. Leave Summary:

- a. This section provides a high-level overview of employee leave data, segmented by time periods:
- b. **No. of Employees took leave Today:** It is a title card that shows the count of employees who have taken leave today and click to view the date wise Leave Summary.
- c. **No. of Employees took leave this Week:** It is a title card that shows the count of employees who have taken leave this week and click to view the week wise Leave Summary.
- d. **No. of Employees took leave this Month:** It is a title card that shows the count of employees who have taken leave this Month and click to view the month wise Leave Summary.
- e. **No. of Employees took leave this Year:** It is a title card that shows the count of employees who have taken leave this Year.

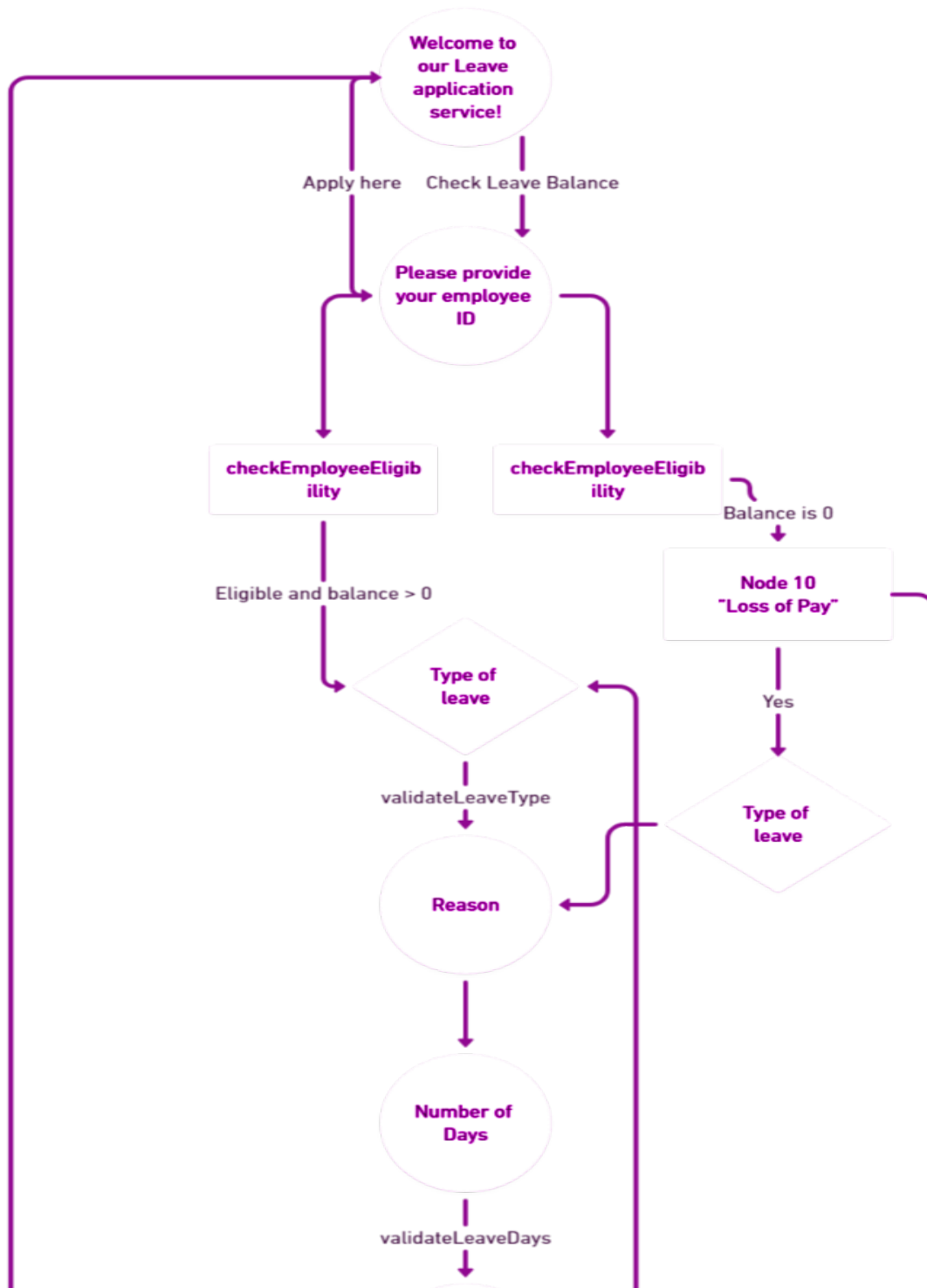
3. Leave Summary of all Employees:

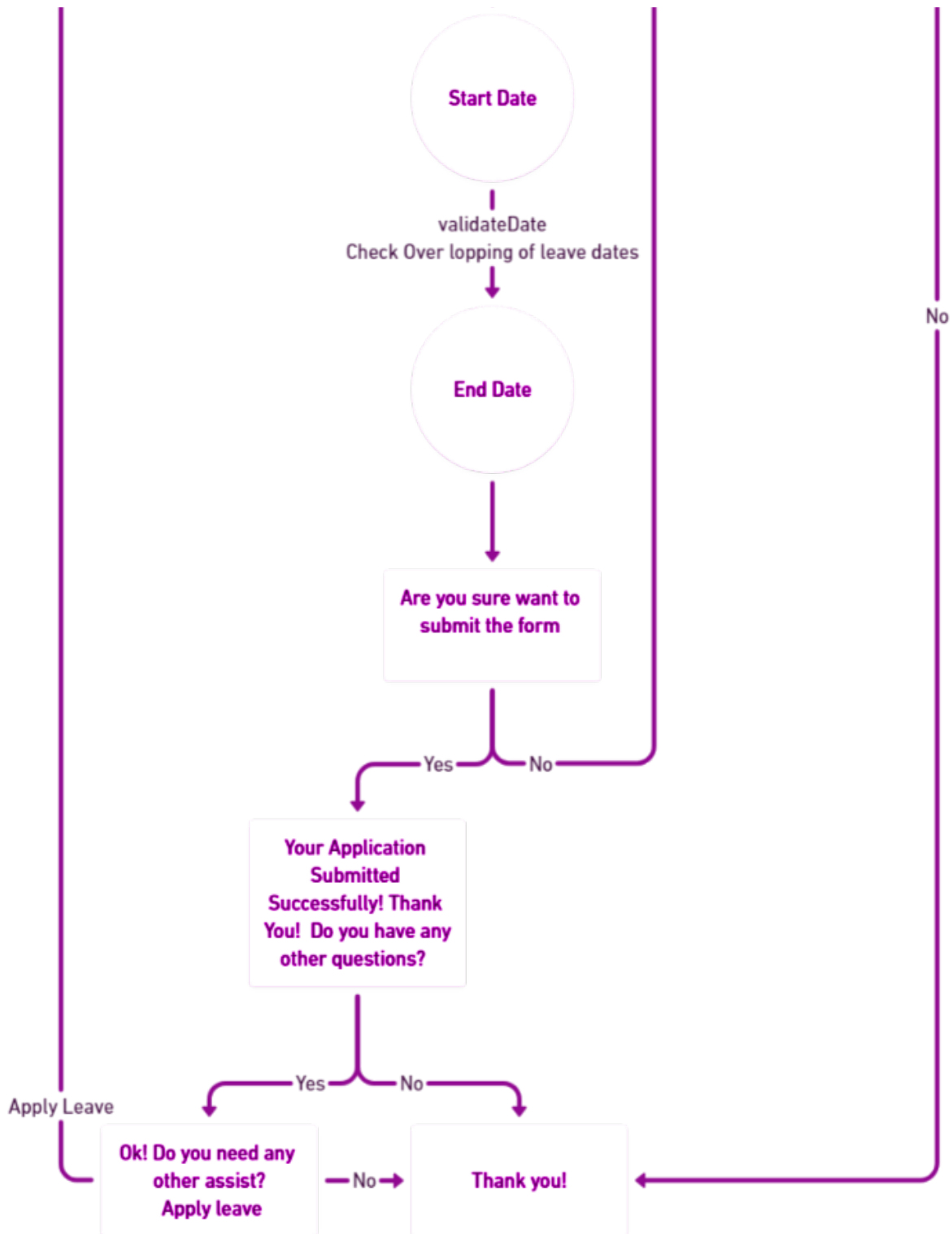
- a. Here, you can check the detailed leave records for all employees. This feature allows for tracking, enabling you to manage an employee's leave balance.

4. Summary of a particular Employee:

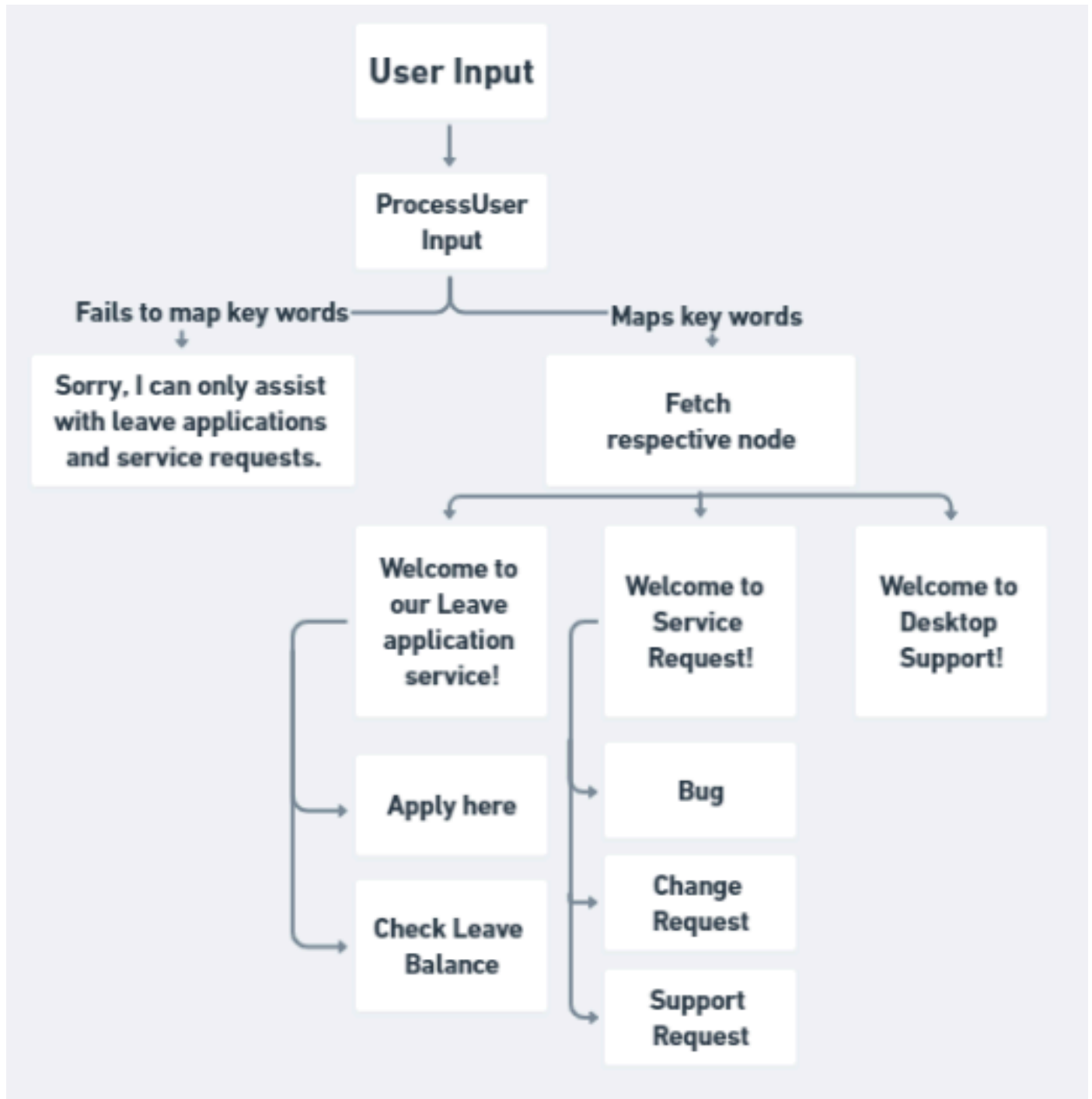
- a. Here, you can check the detailed leave records for any specific employee. This feature allows for individualized tracking, enabling you to manage an employee's leave balance, review their leave history, and ensure they are adhering to company leave policies.

Decision tree Diagram for Leave Application:





Working of the chatbot Diagram:



Sample Output:

Login Page:





Home Page:

- Home
- My Details
- Leave
- Service Requests
- Desktop Support
- Logout

Welcome! Sharan

Home

The central point of the dashboard, the Home section allows you to return to the main dashboard view. From here, you can see all available options and navigate to different features of the home dashboard and how to use ChainBot.

My Details

Allows users to view their personal employee information.

Leave

- Leave Details: Users can track their leave balances, including Casual and Medical Leave.
- Leave Reports: Users can review detailed records of their leave usage, organized by year and including types and reasons.

Service Requests


This option provides access to service requests, where users can view their SR IDs and data.

Desktop Support:

This option provides access to support tickets related to desktop issues.

ChainBot

Hi! I'm Chainbot, Online Assistant. I'm here to help you with Leave Application, Desktop Support and Service Requests.





- Home
- My Details**
- Leave
- Service Requests
- Desktop Support
- Logout


Welcome! Sharan

Employee Details

Employee ID:	370
Name:	Sharan
Date_of_Birth :	12-06-2004
Contact Number:	9994244780
Mail ID:	sharan.baskaran@chainsys.com









- Home
- My Details
- Leave**
- Leave Details**
- Leave Report
- Service Requests
- Desktop Support
- Logout

Welcome! Sharan

Leave Details

Casual Leave Taken:	1
Medical Leave Taken:	1
Total Leave Taken:	2





Home

My Details

Leave

Leave Details

Leave Report

Service Requests

Desktop Support

Logout

Welcome! Sharan


All

Leave Records

2024

Start Date	End Date	Leave Type	Reason	Leave Days	Loss of Pay
Aug 8, 2024	Aug 8, 2024	Casual Leave	fuction	1	No
Sep 29, 2024	Sep 29, 2024	Medical Leave	Fever	1	No





Home

My Details

Leave

Service Requests

Desktop Support

Logout

Welcome! Sharan

Filter Service Requests

Show Filters

Service Requests


- Service Request ID: Service_Request_370_bug_1728973102736

Date: 15/10/2024
- Service Request ID: Service_Request_370_change_request_1728971839455

Date: 15/10/2024
- Service Request ID: Service_Request_370_support_request_1728973629610

Date: 15/10/2024





ChainSys

- Home
- My Details
- Leave ▾
- Service Requests
- Desktop Support**
- Logout


Welcome! Sharan

Desktop Support Requests

- Desktop Support ID: Desktop_Support_370_1728974208176 ^


Date: 15/10/2024

Contact Number	9994244780
Location	Madurai
Expected Date	2024-10-15
Problem Description	Older version is too slow
Date	2024-10-15T06:36:48.176Z



ChainBot:


ChainBot - X



Hi! I'm Chainbot, Online Assistant. I'm here to help you with Leave Application, Desktop Support and Service Requests.

Message ChianBot...

ChainBot - X



Hi! I'm Chainbot, Online Assistant. I'm here to help you with Leave Application, Desktop Support and Service Requests.

hi i like to apply for leave


Welcome to our leave application service

Leave form!

Check Leave balance

Cancel

Admin Page:



- Home
- Leave Summary
- Leave Summary of all Employees
- Summary of a particular Employee
- Logout

Welcome! Admin

Home

The central point of the dashboard, the Home section allows you to return to the main dashboard view. From here, you can see all available options and navigate to different features of the admin dashboard.

Leave Summary


- No. of Employees took leave Today: It is a title card that shows the count of employees who have taken leave today and click to view the date-wise Leave Summary.
- No. of Employees took leave this Week: It is a title card that shows the count of employees who have taken leave this week and click to view the week-wise Leave Summary.
- No. of Employees took leave this Month: It is a title card that shows the count of employees who have taken leave this month and click to view the month-wise Leave Summary.
- No. of Employees took leave this Year: It is a title card that shows the count of employees who have taken leave this year.

Leave Summary of all Employees

Here, you can check the detailed leave records for all employees. This feature allows for tracking, enabling you to manage an employee's leave balance.

Summary of a particular Employee

Here, you can check the detailed leave records for any specific employee. This feature allows for individualized tracking, enabling you to manage an employee's leave balance, review their leave history, and ensure they are adhering to company leave policies.



- Home
- Leave Summary
- Leave Summary of all Employees
- Summary of a particular Employee
- Logout


Leave Summary

No. of Employees took Leave Today	No. of Employees took Leave This Week	No. of Employees took Leave This Month	No. of Employees took Leave This Year
1	1	1	4
Click the card to view Date Wise Summary	Click the card to view Week Wise Summary	Click the card to view Month Wise Summary	

Today Leave Summary

No. of employees took leave today: 1

Employee Id	Employee Name	Start Date	End Date	Leave Type	Reason	Leave Days	Loss of Pay
373	Gowtham	Oct 17, 2024	Oct 17, 2024	Casual Leave	To attend function	1	No



- Home
- Leave Summary
- Leave Summary of all Employees
- Summary of a particular Employee
- Logout

Welcome! Admin

Back

2024-10-17


Show All

Search

Leave Details of All Employees (Date Wise)

Thu Oct 17 2024

Emp ID	Emp Name	Start Date	End Date	Leave Type	Reason	Leave Days	Loss of Pay
373	Gowtham	Oct 17, 2024	Oct 17, 2024	Casual Leave	To attend function	1	No



- Home
- Leave Summary
- Leave Summary of all Employees
- Summary of a particular Employee
- Logout

Welcome! Admin


Back

2024-W42

Leave Details of All Employees (Week Wise)

Week 2024-W42

Emp ID	Emp Name	Start Date	End Date	Leave Type	Reason	Leave Days	Loss of Pay
373	Gowtham	Oct 17, 2024	Oct 17, 2024	Casual Leave	To attend function	1	No



[Home](#)

[Leave Summary](#)

[Leave Summary of all Employees](#)

[Summary of a particular Employee](#)

[Logout](#)

Welcome! Admin


[Back](#)

October 2024

Leave Details of All Employees (Month Wise)

October 2024

Emp ID	Emp Name	Start Date	End Date	Leave Type	Reason	Leave Days	Loss of Pay
373	Gowtham	Oct 17, 2024	Oct 17, 2024	Casual Leave	To attend function	1	No



[Home](#)

[Leave Summary](#)

[Leave Summary of all Employees](#)


[Summary of a particular Employee](#)

[Logout](#)

Welcome! Admin

Summary Of all the employees

Emp ID	Emp Name	Total Leave Days	Casual Leave Balance	Medical Leave Balance	Total Leave Balance
370	Sharan	2	5	5	10
371	Dhaneshwar	3	3	6	9
372	Sujan Bose	1	6	5	11
373	Gowtham	2	5	5	10



Home

Leave Summary

Leave Summary of all Employees

Summary of a particular Employee

Logout

Welcome! Admin

Employee Leave Details

373

Search

Employee Details

Employee ID: 373

Name: Gowtham

Leave Details

Casual Leave: 1

Medical Leave: 1

Total Leave Taken: 2

Leave Records

2024

Start Date	End Date	Leave Type	Reason	Leave Days	Loss of Pay
Aug 8, 2024	Aug 8, 2024	medical leave	function	1	No
Oct 17, 2024	Oct 17, 2024	Casual Leave	To attend function	1	No

STEPS TO SETUP THE PROJECT:

Project Url: <https://github.com/SharanBaskaran/CHAINBOT-V2>

1. Download the code base from the Git repository:

- Clone or download the project from the GitHub repository.
- **Unzip the file**(if necessary) to extract the project files.

2. Open the folder in Visual Studio Code:

- Open VS Code and use **`File > Open Folder...`** to navigate to and open the extracted project folder.

3. Open the terminal in VS Code:

- You can open the terminal by clicking on **`Terminal > New Terminal`** from the top menu.

4. Install Angular packages:

- In the terminal, run the following command to install all the dependencies required by the project: **npm install**

5. Install the required packages for email service and Google Gemini AI:

- In the terminal, run the following commands: **npm install --save-dev @types/emailjs-com, npm install @google/generative-ai**

6. Run the application:

- Now start the Angular application by running: **ng serve**