Proposal

**Executive Summary**

Senior Rights Victoria (SRV) faces significant operational challenges in managing the high volume of calls from the elderly community. This proposal outlines a strategic plan to implement an online ticketing system aimed at improving response times, prioritizing urgent cases, and enhancing overall service delivery for elder abuse support.

**Current System and Challenges**

* **Long Wait Times:** Extended waiting periods for voicemail leave vulnerable seniors without timely support.
* **Inefficient Call Handling:** The lack of efficient call management systems results in backlogs and delayed responses.
* **Turnaround Time (TOT):** A slow response rate of up to one week, inappropriate for urgent needs.
* **Impact on Elder Abuse Response:** Delays adversely affect SRV’s mission to provide prompt and effective support for elder abuse issues.

**Proposed Solution: Online Ticketing System**

**Objective:** To establish an accessible, efficient, and responsive ticketing system for SRV that:

1. **Reduces Wait Times:** Implements a digital queue system allowing for immediate ticket submissions.
2. **Prioritizes Urgent Cases:** Features an urgency categorization system for prioritizing critical cases.
3. **Enhances Call Handling:** Utilizes AI and automation to streamline the ticket creation and management process.
4. **Improves Turnaround Time:** Targets a reduction in response time with real-time tracking and analytics.
5. **Supports Elder Abuse Response:** Ensures rapid action and follow-up on elder abuse reports.

**System Components (based on user stories)**

* **IVR Integration:** To filter urgent cases and guide users in ticket submission.
* **Multilingual Support:** Ensures inclusivity for non-English speakers.
* **Accessibility Features:** For users with visual and hearing impairments.
* **User Profiles:** Streamlined creation and management through CRM integration.
* **Chatbots and FAQ:** For immediate assistance with common issues.
* **Appointment Scheduling:** Online system for efficient service access.
* **Feedback and Analytics:** Real-time dashboards for service quality monitoring.

**Technical Considerations**

* **Framework:** Secure and scalable to support a high volume of users.
* **Database:** Compliant with privacy regulations, utilizing encryption for sensitive data.
* **Software Integration:** Compatibility with existing systems for a seamless transition.
* **User Authentication:** To ensure data security and privacy.
* **APIs:** For efficient frontend and backend communication.

**Ethical and Security Measures**

* **Data Privacy:** Adherence to consent management and information transparency.
* **Access Control:** Role-based access to sensitive data.
* **Secure Development:** Best practices to prevent data breaches and vulnerabilities.

**Implementation Timeline**

* **Research and Selection:** Review and choose appropriate software (1 month).
* **Development and Testing:** Backend, frontend, and database development with subsequent testing phases (4-6 months).
* **Training and Transition:** Training staff and gradually transitioning to the new system while running parallel with the old system (2 months).
* **Go-Live:** Official switch to the new system with support and monitoring in place (1 month).

**Budget Considerations**

* Development, licensing, and maintenance costs of the proposed systems.
* Training expenses for staff.
* Long-term savings from improved efficiency and reduced need for phone line resources.

**Conclusion**

The implementation of an online ticketing system is a transformative step for SRV, addressing the immediate challenges and laying the foundation for a scalable and robust service delivery model. This will ultimately empower SRV to better serve its community, responding to elder abuse cases with the urgency and sensitivity they require.

Appendix   
  
**User Stories and Solutions**

User Story: Immediate Assistance Request

**Story:** As a senior facing an urgent issue, I want to quickly submit a ticket via the SRV web app. **Solution:** Implement a prioritized ticketing system with an urgency dashboard. This allows employees to see real-time updates and historical information, enabling them to prioritize and address urgent cases swiftly.

User Story: Follow-up on Existing Case

**Story:** As a senior who has previously contacted SRV, I want to check the status of my case or add new information. **Solution:** Create a user portal within the ticketing system where clients can view the status of their cases and submit additional information. Provide SRV employees with a transparent ticket management system to track progress.

User Story: Prompt Assistance Regarding Someone Close

**Story:** As an individual close to someone experiencing elder abuse, I want to quickly communicate my scenario. **Solution:** Integrate an IVR system with prompts that guide the caller through issuing a ticket or directly connecting to a representative. Implement a categorization system within the ticketing platform to route these cases appropriately.

User Story: Prompt Assistance Regarding My Client

**Story:** As a professional working with the elderly, I need to talk to someone who understands the elder abuse my client may be facing. **Solution:** Establish dedicated lines for professionals through the IVR system, offering immediate assistance or the option to create a detailed ticket that will be escalated to the appropriate SRV staff.

User Story: Help with Common Issues

**Story:** As a senior with a simple problem, I want to resolve my issue without delay. **Solution:** Integrate a chatbot within the IVR and web app to answer frequently asked questions, providing immediate solutions to common issues.

User Story: Scheduling Appointments

**Story:** As a user of SRV, I want to schedule appointments online without waiting on the phone. **Solution:** Develop an online scheduling system integrated with the ticketing platform that asks for user consent and allows clients to book and manage appointments conveniently.

User Story: Multilingual Support

**Story:** As a non-English speaking senior, I want to communicate effectively for support. **Solution:** Offer multilingual support within the IVR system and on the web platform, ensuring language is not a barrier to accessing services.

User Story: Accessibility Support

**Story:** As a senior with visual or hearing impairments, I need an accessible phone screening system. **Solution:** Ensure the web app and IVR system are designed with accessibility in mind, featuring high-contrast interfaces, adjustable font sizes, and compatibility with assistive technologies like screen readers.

# Project Ideation research

Target audience – SRV employees

Implementation: Web Application (to **schedule appointments and submit inquiries online)**

**Existing ticketing systems**

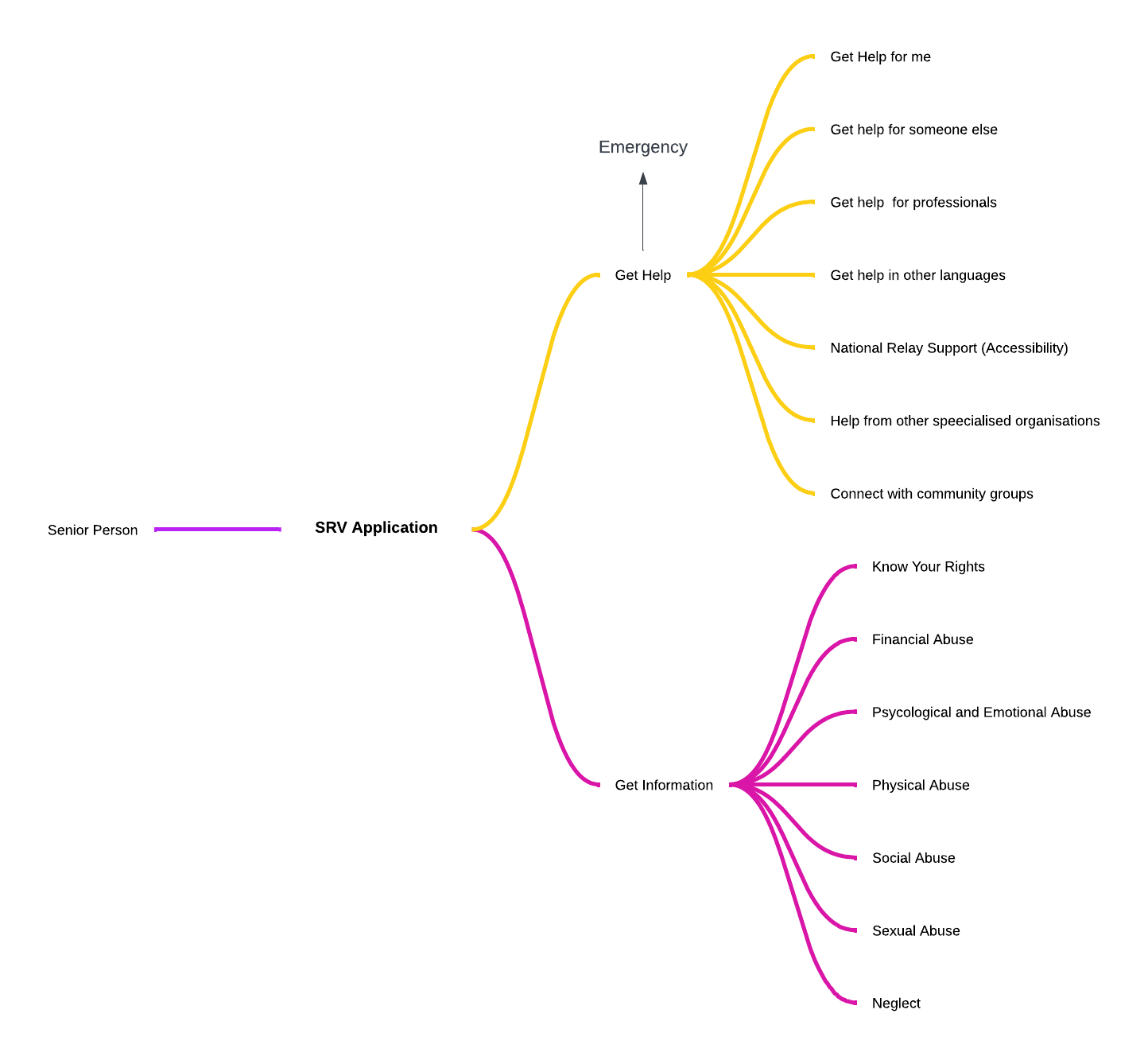
(Rohani)

A **ticketing system** is a software tool used to manage and track customer inquiries and support requests. It allows users to submit tickets for assistance, which are then assigned to appropriate agents. It provides efficient communication between customers and support staff, ensuring issues are addressed promptly.

Below are several ticketing systems suitable for customer service, each offering different pricing plans. When selecting a ticketing system for SRV, it's important to choose one that balances affordability with the essential features needed. While free ticketing systems are available, they may pose data privacy concerns due to unclear data handling practices. Opting for a paid ticketing system ensures better control over data security and adherence to privacy regulations.

|  |  |  |  |
| --- | --- | --- | --- |
| Ticketing System | Annual cost per agent | Features | Cons |
| 1. **Zendesk** | US $ 1,068 | **Multichannel support-** agents can manage and respond to customer inquiries from email, chat, phone and social media.  **Self-service Options -** AI chatbots to help customers with inquiries and provide instant responses.  **Reporting call analytics –** dashboards to track key performance metrics.  **UI** - user-friendly interface for agents to manage tickets, communicate with customers. | Complex interface- Learning the features may be time consuming for the agents.  More expensive |
| 2. **Zohodesk** | US $ 422.4 | **Multichannel support -** agents can manage and respond to customer inquiries from email, chat, phone, social media and self-service portals.  **AI-powered self-service** via a virtual assistant.  **Reporting call analytics** – reporting dashboards and analytics to track key performance metrics.  **UI -** user-friendly interface for agents to manage tickets, communicate with customers. | Agents need to switch between different interfaces for chat, phone and ticketing, rather than managing ongoing conversations in a single view |
| 3. **Jira service management** | US $650 | **Multichannel support -** agents can manage and respond to customer inquiries from email, chat, phone, social media and self-service portals.  **AI-powered self-service** via a virtual assistant.  **Reporting call analytics** – reporting dashboards and analytics to track key performance metrics.  **UI -** user-friendly interface for agents to manage tickets, communicate with customers. | Complex interface. |
| **4. Intercom** | US $468 | **Multichannel support -** agents can manage and respond to customer inquiries from email, live chat, social media and self-service portals.  **Unified Inbox -**Provides a centralized inbox where support tickets, messages and inquiries from various are consolidated into one interface.  **AI-powered self-service** for inquiries.  **Reporting call analytics** – reporting dashboards and analytics to track key performance metrics | Advanced automation and chatbots features are more expensive |

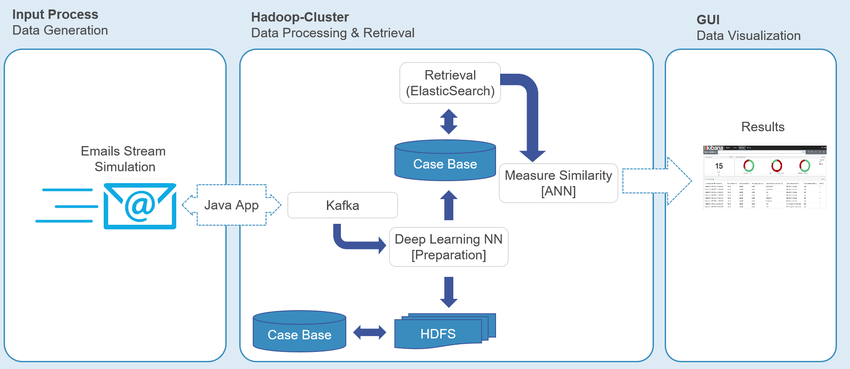
(Muhammad) Existing SRV Site Structure:



**Shanti**

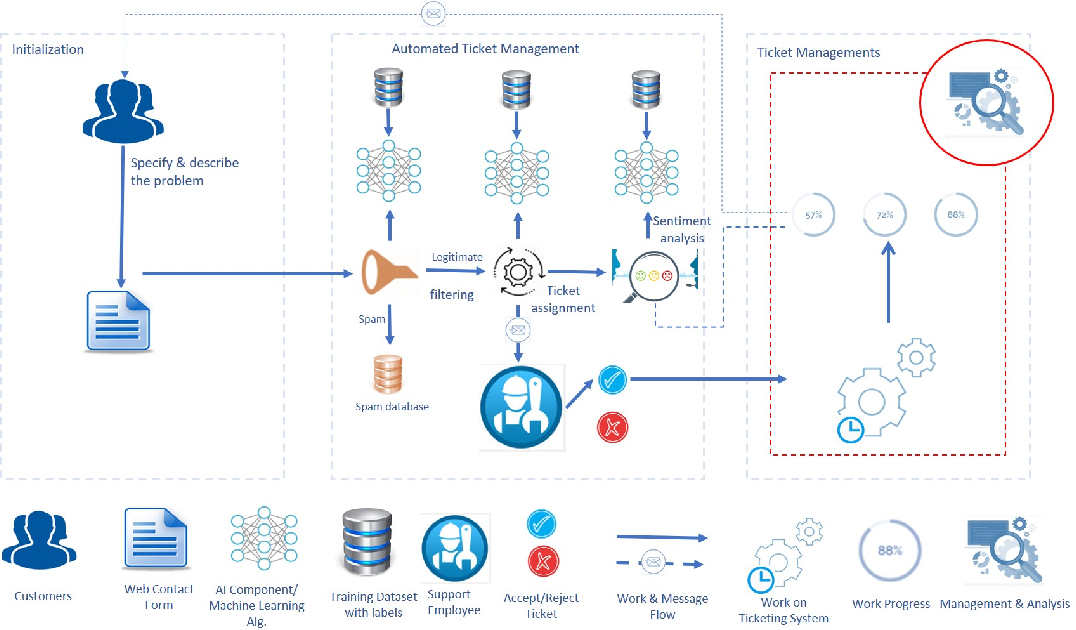
**Suggested System Framework**

**Automation and Workflow:**

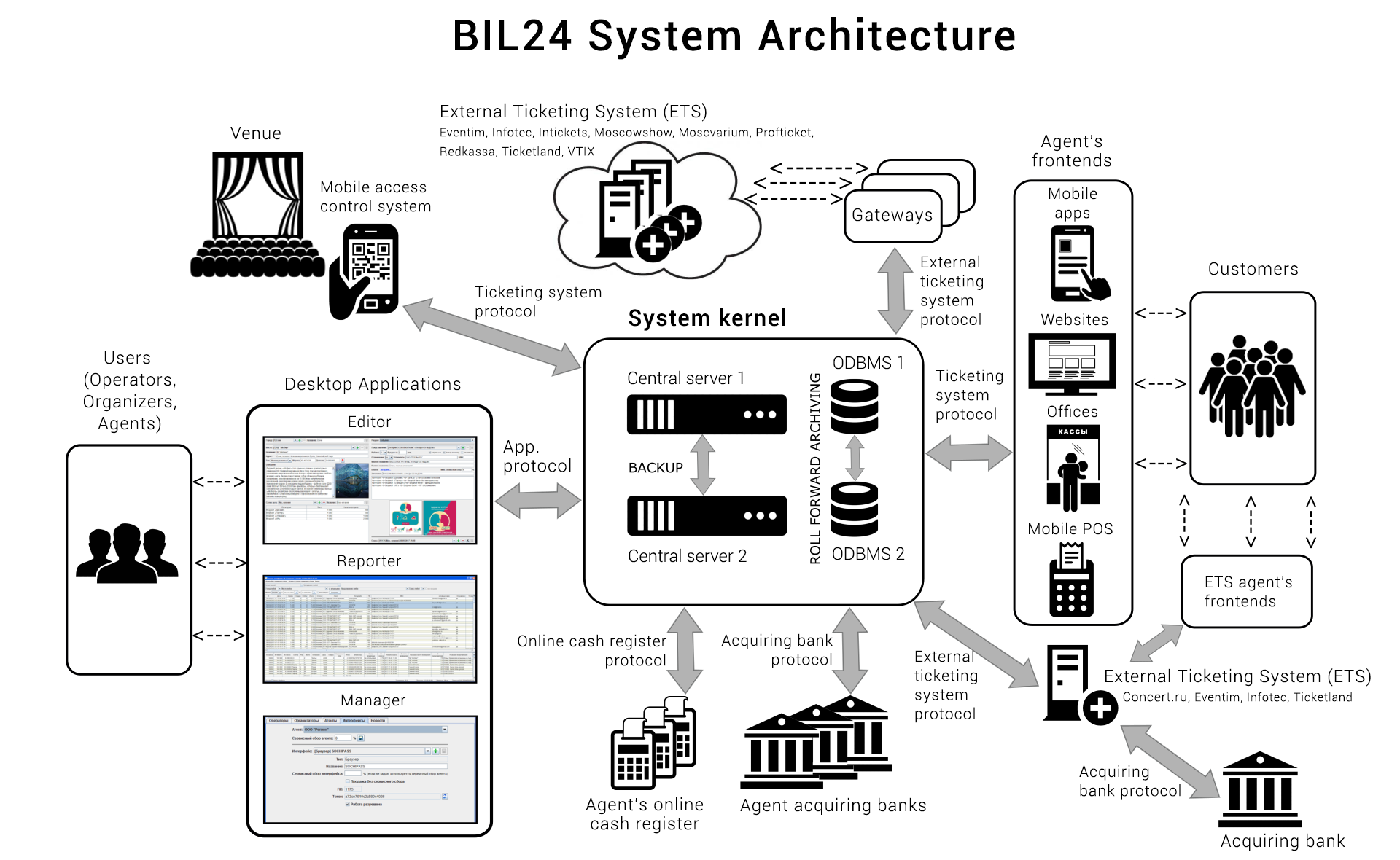


An automated ticket management process, starting from a customer specifying and describing a problem, through various automated steps, like spam filtering, sentiment analysis, and ticket assignment, culminating in ticket management and analysis (Amin et al., 2020).

**Data Processing and Retrieval:**



Data processing and retrieval flow within a Hadoop cluster environment, involving streams of emails, a Java application interfacing with Kafka, and the use of ElasticSearch and deep learning neural networks for case retrieval and similarity measurement (Qamili et al., 2018)

**Overall System Architecture:**  


A detailed system architecture for a ticketing system called "BIL24," indicating how various users (agents, customers, organizers) interact with the system through multiple interfaces, such as mobile apps, desktop applications, and websites. This architecture also illustrates the back-end processes, including database management, backup servers, and protocols for integration with external systems, like external ticketing systems and banking protocols (bil24, nd)

Components:

**Customer Interaction**: Users report issues or request services through various front-end applications.

**Ticket Generation**: Reported issues are converted into tickets, which may involve spam detection and sentiment analysis to prioritize them.

**Processing and Management**: Tickets are assigned to support staff or automated systems. There might be a case base for managing past and current tickets, which aids in retrieving similar cases and speeding up resolution.

**Back-End Infrastructure**: Involves data storage, processing clusters (like Hadoop), and data retrieval systems (like ElasticSearch). The architecture is designed for high availability and reliability, often including multiple servers and backup systems.

**Integration**: There's often a need to interface with other systems, which requires various protocols and gateways to ensure seamless data exchange.

**Data Visualization**: GUI elements to display the results, statuses, and other relevant analytics to the support staff and management for quick decision-making.

References  
Amin, Kareem & Kapetanakis, Stelios & Polatidis, Nikolaos & Althoff, Klaus-Dieter & Dengel, Andreas. (2020). DeepKAF: A Heterogeneous CBR & Deep Learning Approach for NLP Prototyping. 1-7. 10.1109/INISTA49547.2020.9194679.

<https://www.semanticscholar.org/paper/An-Intelligent-Framework-for-Issue-Ticketing-System-Qamili-Shabani/211d6c781bf23d72b55feb6674612586d75728c1>

<https://bil24.pro/architecture_en.html>

# Technical Research

(Rohani)

Creating a web application would streamline the process for clients of the SRV helpline, **allowing them to schedule appointments and submit inquiries online**. This shift from phone calls to online interactions could significantly reduce wait times on the helpline and make the support process more convenient and efficient for both staff and clients.

* **How will we know if our solution and its implementation are better than the current system?**
* We can measure the reduction in wait times on the helpline by comparing average response times before and after the implementation of the web application.
* Integrating chatbots into the web application can further enhance efficiency by providing instant responses to common inquiries, freeing up staff time for more urgent issues. The impact of this feature can be measured by monitoring user satisfaction ratings and the precision and efficacy of chatbot responses.
* The integration of a ticketing system with the web application allows for better organization and tracking of client inquiries.
* Implementing Interactive Voice Response (IVR) within the web application can streamline the initial call routing process for urgent inquiries. By categorizing calls based on urgency levels and directing urgent calls to priority queues, we can ensure that critical issues receive prompt attention. Tracking the average response time for urgent inquiries and client feedback on the IVR system can help evaluate its effectiveness.

# Technical implementation

Research and choose appropriate software for the solution

* What applications will be used
* Will we create a website for the callers, or create a solution for SVR?

Zunga

# **2. Technical Implementation**

In order to proficiently execute an adequate solution for the SRV Phone Screening Project, an outline of the necessary technological applications will need to be provided. Various applications may need to be drawn upon, as the next stages of the project are divided into the implementation of different additions.

## **2.1. Required Application – IVR System**

**Why an IVR System?**

SRV are looking to significantly improve their current IVR system, in order to provide a better client experience. An Interactive Voice Response (IVR) is an automated voice answering system, which provides unique responses or cooperation to incoming calls based on the specific input the caller selected. These systems are beneficial for phone service lines, as they can assist in directing the caller to the correct employee, or effectively help the caller without transferring them to an employee, but by relaying informative automated responses, in regard to their selected IVR options (Global Call Forwarding, n.d.). IVR systems can provide immense benefit to any organisation if implemented correctly.

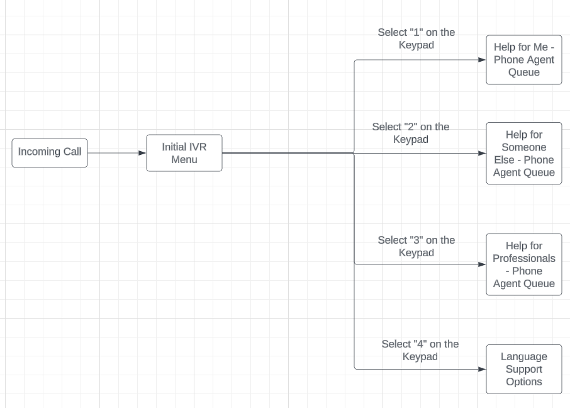
**Step 1 of Developing an Optimal IVR System: Selection of Required Departments**

In order to implement an IVR system correctly, various aspects must be taken into consideration.

The first step should be to select the departments within the organisation that will be included in the IVR’s options (Global Call Forwarding, n.d.). In the context of SRV, this would include the customer service officers who answer the centres default phone line. However, this can be further organised by dividing the specific IVR options into the four main types of support described on the SRV’s website: Help for me (callers who may have been personally subject to elder abuse), help for someone else (callers calling on behalf of an individual who may be experiencing elder abuse), help for professionals (industry professionals whose clients may be experiencing elder abuse), and language support (any individual concerned about or expiring elder abuse that is undergoing a language barrier preventing them from speaking with a phone agent that speaks English (Seniors Rights Victoria, n.d.).

Now that the core four avenues of SRV’s specific categories for services have been identified, they can now be possibly imbedded within the IVR system, as four separate IVR options which have the potential to make answering client calls much easier can be used. This ultimately ensures that any individual looking to speak to a specific SRV officer will be instantly transferred to the extension per the IVR that is best suited for their needs. For example, an individual who is experiencing elder abuse can simply select the designated IVR option for this service, and be sent through to a designated officer who is well versed with the nature of these calls. The departmentalisation that an enhanced IVR provides will also make expecting incoming calls far easier for the phone agents. Agents who have been rostered on the “help for someone else” IVR option will always be expecting and prepared for calls in which the caller isn’t the one experiencing the abuse, but talking on behalf of someone who is. This also makes the call experience much more fluid and professional, as agents are readily prepared for how the discussion may go, and callers are assured they are speaking to someone who can competently gauge their problem.

Based on the four key components that define how SRV provides assistance to enquiries, the IVR can initially be developed to cater to these sections. Figure 1 below showcases how the IVR can be curated in accordance with these 4 departments and assist in the organisation of client calls.





**Step 2 of Developing an Optimal IVR System: Decide on Single Level or Multilevel IVR**

This step involves the deciding of how much “levels” will be imbedded within the IVR. Levels are simply tiers within an IVR system that can be actioned by callers if they undertake a specific sequence of IVR option selections. Single level IVR is quite limited in the amount of available IVR sequences and typically only allows for the selection of one option, or automatically joins a phone queue without allowing users to select any IVR options. Multilevel IVR is an IVR system that contains various IVR options that incoming callers can utilise to join their desired phone queue (Global Call Forwarding, n.d.).

Based on the multiple services and operators that SRV has readily available and the succinct IVR map provided in Figure 1, a multilevel IVR would be best suited for this project. It enables callers to have a greater range of freedom in regard to the specific operator they are hoping to speak to. This will not only improve the level of customer service callers will experience, but also provides a heightened level of call visibility for the SRV staff, as they can easily monitor and distinguish the calls which are being received, and what the caller is looking to discuss. This also allows executive SRV staff to gauge an understanding of their most frequent enquiries, as the IVR can identify where their client base is mostly coming from or looking to know, providing further opportunities for expansion of departments with the most traffic.

**Step 3 of Developing an Optimal IVR System: Choose a “Voice” for the System**

The voice of an IVR system is the tone and audible aesthetic of the audio response that responds to incoming calls and informs the caller of the available IVR options. This voice should be clear, concise, and easy for callers of various backgrounds to understand. The voice for the SRV Phone System IVR should be a clear, calm, and feminine voice with an Australian accent. The qualities of this voice can assist in easing the client as they navigate their way through the IVR and it will be easily understandable as it will be exhibiting an Australian cadence, which majority of the callers will be comfortable with (Global Call Forwarding, n.d.).

**Step 4 of Developing an Optimal IVR System: Choose and pay for an IVR Provider**

This step involves the selection of a specific IVR provider and paying them accordingly, in regard to the chosen service option. There are various IVR providers available who can provide quality services based on an organisation’s needs (Global Call Forwarding, n.d.).

**Step 5 of Developing an Optimal IVR System: Uploading and Configuration of Required Prompts**

Once the ideal IVR service method has been decided on, project developers can then upload the specific prompts that will be relayed within the IVR and configure the relevant steps and pathways that the IVR will showcase (Global Call Forwarding, n.d.).

**What IVR Applications Can be Used?**

If looking to create the IVR solution from “scratch” and actively program it, the Python programming language can be used. The concisely written steps of how to use Python to do this are outlined below (Lau, 2022):

1. In order to do this the Python add on, Django will need to be downloaded, as well as ngrok and a working phone line.
2. A Python virtual environment will need to be set up.
3. After this, the Django project will need to be created within this Python directory.
4. Then a digital network tunnel will need to be established with ngrok.
5. The phone line to be used must be confirmed to be working.
6. The IVR system can then begin to be built with Python.
7. Proficient coding prompts will need to be relayed within the program’s directory.

The following link provides further detail into the specifics of what these written steps must include: <https://www.codemotion.com/magazine/languages/2-ways-to-build-a-phone-ivr-system-with-python/>

Twilio Studio can be used to create an IVR system without any coding. It provides an array of useful steps and prompts that users can follow to create their ideal IVR program Twilio (n.d.). Twilio is clearly a suitable software for the solution.

However, there is various other credible potential software options that can aid in the development of an IVR system without coding (Capterra, c.2024):

* Aircall
* Zendesk Suite
* Voice Guide IVR
* Justcall
* RingCentral MVP
* Bitrix 24

Voice API is a technological application that grants developers with more control, in regard to how they can curate the intricacies of an IVR system. It also allows developers to implement functional voice solutions within the IVR’s code. Voice API is essentially an array of API’s that allow for the customisation of IVR menu options. Voice API can allow for useful add ons within the IVR such as speech recognition, answering machine detection, and text to speech. Voice API would be ideally implemented if it’s placed within an existing IVR scenario, that has been created with a desired programming language. This is a viable option if the addition of a competent addition to an existing IVR system is a priority (Karlović, 2023).

**Ticketing System**

A ticketing system is a technological tool that assists in the automation, management, and processing of service requests and strives to provide an adequate resolution. They provide a digital trail of tasks that need to be tended to by staff members. The chosen system will then monitor each tickets status, whilst staff members tend to resolve them. Tickets also record all communication and progress that occurs in regard to this ticket, ensuring the outcome is digitally recorded, such as backlogs of staff members calling the client being added within the ticket. Such an addition will be pivotal in creating better call visibility for the staff of the SRV Phone System, as there is a greater sense of organisation, prioritization, and consolidation in order to support client requests. The ticketing system can possibly work by transforming client calls who haven’t received an answer after a certain amount of time into actionable tickets that can ensure future correspondence can be met for their enquiry. These tickets not only provide enhanced visibility in regard to client matters, but also provide leadership members with analytical insights, which can allow them to better improve the SRV service. A list of potential ticketing system applications is below (Zendesk, c.2024):

1. Zendesk: Provides key features such as tagging and routing, unified omnichannel experience, reporting and analytics, AI powered self-service, and an extensive library of 1,000 + integrations. Its starting price is $55 per agent/month.
2. Zoho Desk: Provides key features such as tagging and routing, unified omnichannel experience, reporting and analytics, AI powered self-service, and an extensive library of 1,000 + integrations. Its starting price is $20 per agent/month.
3. Freshdesk: Provides key features such as tagging and routing, unified omnichannel experience, reporting and analytics, AI powered self-service, and an extensive library of 1,000 + integrations. Its starting price is $0 per month (up to 10 agents).
4. HappyFox: Provides key features such as tagging and routing, unified omnichannel experience, reporting and analytics, AI powered self-service, and an extensive library of 1,000 + integrations. Its starting price is $39 per agent/month.
5. Help Scout: Provides key features such as tagging and routing, unified omnichannel experience, reporting and analytics, AI powered self-service, and an extensive library of 1,000 + integrations. Its starting price is $25 per user/month.

## **2.2 What Will the Technical Solution Look Like?**

The ideal technical solution for all of SRV’s current phone dilemmas should be the implementation of a refined IVR system and a functional ticketing system, instead of a website. Considering a significant number of individuals calling into the SRV phone are likely elderly, curating a new version of the website that attempts to showcase features present in the IVR or ticketing system, may be too difficult for this demographic to grasp, as they statistically have lower levels of digital literacy. However, an enhanced IVR and a new ticketing system will serve as mere additions to the phone system, allowing for a smoother transition for the elderly users as they are already accustomed to speaking via the phone. The complex and sensitive nature of the themes that SRV provides aid for may also deter users from indicating this information on a website form. The IVR and Ticketing System will also provide greater visibility than a website, as they can monitor a similar interface to the one that receives incoming calls and not have to assimilate to an additional website portal for the website specific enquiries. Considering that users who currently use the SRV service are used to relaying their enquiries via the phone, conducting a new website may be too drastic of a change to be undertaken, however, the IVR and ticketing system still rely on phone use. The refining of the IVR system shouldn’t provide too much difficulty for the elderly users as other phone services that they often use may have also adopted IVR prompts, such as local councils and other government services, providing a smoother transition than a website. The elderly users are also much more used to speaking via the phone if conducting digital communication, instead of inputting information into an online site. It is evident that the implementation of an IVR system and ticketing system is far more of a suitable solution than the creation of a CRM style website for the clients.

# **3. References**

Capterra. (c.2024). IVR software - Comparison & Reviews. <https://www.capterra.com.au/directory/30231/ivr/software>

Global Call Forwarding. (n.d.). How to Set Up Interactive Voice Response. <https://www.globalcallforwarding.com/how-to-set-up-interactive-voice-response/>

Karlović, M. (2023). A step-by-step guide on how to create an IVR system. Infobip. <https://www.infobip.com/blog/how-to-create-an-ivr-system>

Lau, G. (2022). 2 ways to build a phone IVR system with python. Codemotion Magazine. <https://www.codemotion.com/magazine/languages/2-ways-to-build-a-phone-ivr-system-with-python/>

Seniors Rights Victoria. (n.d.). Seniors Rights Victoria - Home. <https://seniorsrights.org.au/>

Twilio. (n.d.). How to build an IVR with Twilio Studio. <https://www.twilio.com/docs/studio/tutorials/how-to-build-an-ivr>

Zendesk. (c.2024). 17 top ticketing systems + tools for 2024. Zendesk. <https://www.zendesk.com/au/service/help-desk-software/ticketing-system/>

# Software requirements

* How will we integrate the calls and voicemail messages with a software solution that can filter/sort the messages?
* What software/programs/applications can do this?

# Software implementation

(Rohani)

* **Test prototype**
* Define the objectives of the web application and gather information from SRV staff and clients. Determine the key features and functionalities that the prototype should include.
* A rough design of the web application components can be created using wireframes such as Figma. This focuses on the layout of the pages and represents the visual appearance.
* Implementation of Interactive prototypes, to allow users to navigate through the web application and interact with its features (buttons etc..) as they would in the final product. Tools such as InVision and Adobe XD can be used for this.
* The prototype provides a visual representation of the product being developed. It is essential in the development process as it helps to test and refine the product before moving forward with full-scale development.
* **Frontend Development**
* After designing the prototype, the next step is frontend development which involves translating the finalized design into functional code and implementing responsive design to ensure accessibility.
* Use the wireframes and interactive prototype as a reference to begin coding the frontend of the web application.
* Break down the design into smaller components such as headers, navigation bars, buttons, forms and write HTML markup to create the structure of the web pages.
* Apply CSS styles to the HTML elements to match the visual appearance shown in the wireframes and prototype.
* Implement interactivity using JavaScript or JavaScript libraries such as React.
* Incorporate animations and transitions to enhance user experience and engagement.
* Test the front-end code thoroughly to find and correct any errors, bugs or inconsistencies.
* Ensure that the web application is accessible to users with disabilities. For example, *maintain sufficient color contrast between text and background colors to ensure readability for users with low vision or color blindness.*
* Use Git to track changes and collaborate with other developers working on the front-end code.
* **Backend Development**
* After completing the front-end development, the next step is to proceed with back-end development. This involves creating the server-side components and logic necessary to support the functionality of the frontend.
* Back-end developers ensure the website performs correctly, focusing on databases, APIs and architecture.
* Frameworks such as Node.js, Django and ruby can be used to streamline backend development.
* Create the database structure based on the application's data requirements.  
   Select an appropriate database management system, such as MySQL.
* Implement database models, tables, and relationships according to the defined schema.
* Write server-side code in the chose framework to handle business logic, data processing and interaction with the database.
* Implement API endpoints to allow communication between the front-end and backend components.
* Conduct unit testing and integration testing to ensure the accuracy and reliability of backend code.
* Use encryption and secure protocols such as HTTPS to protect data transmission between the frontend and backend.
* Use Git to track changes and collaborate with other developers working on the back-end code. Follow branching, committing and merging code changes to ensure a stable and organized development process.

To guarantee safe communication and limit access to specific resources to authorized users only, security measures must be implemented on both the frontend and the backend. This may include implementation of security tokens, user authentication and authorization.

It is essential that the frontend can handle mistakes, such as network failures, unexpected server responses and problems with data validation, that could arise during communication with the backend. The frontend must efficiently display the data provided by the backend, involving the parsing and processing of the received data to present it coherently and understandably to the user.

Effective frontend-backend integration through an API is crucial for seamless application functionality and user experience, ensuring both components work together efficiently.

(Guwani) Frontend Development

Cross-Browser Compatibility and Performance Optimization:

Ensuring the web application is compatible with different web browsers (eg. Chrome, Firefox, Safari) is crucial to provide a consistent user experience across platforms. Optimizing frontend performance involves strategies such as minimizing HTTP requests, optimizing images and assets, implementing lazy loading, and utilizing browser caching. Performance monitoring and testing tools (e.g., Lighthouse, Google Page Speed Insights) play a vital role in identifying and addressing performance bottlenecks.

Backend Development

Scalability and Load Balancing:

Designing the backend infrastructure to handle increasing traffic and user demand is essential as the web application grows.

Concepts such as horizontal and vertical scaling, load balancing, and auto scaling ensure that the application remains responsive and available under heavy loads. Cloud-based solutions (eg, AWS Elastic Load Balancer, Google Cloud Load Balancing) and container orchestration platforms (eg. Kubernetes) help manage scalability and distribute incoming traffic effectively.

# Backend and Database Modelling (if required)

* Will the solution require a database? (Most likely will if we are taking voicemails)
* Research and determine what database tools/languages/programs are appropriate (PostgreSQL, dbeaver, MySQL etc.)
* Create a database model to visualize how data would be stored and managed
* Ethical concerns with storing and managing sensitive data

(Muhammad’s Contribution)

**Database Modelling**

Designing the database for the **Senior Rights Victoria** web-based application involves understanding the entities involved, their relationships, and the requirements for storing and managing data securely. Below is a suggested approach for the database model, technologies, and storage types:

**Database Model:**

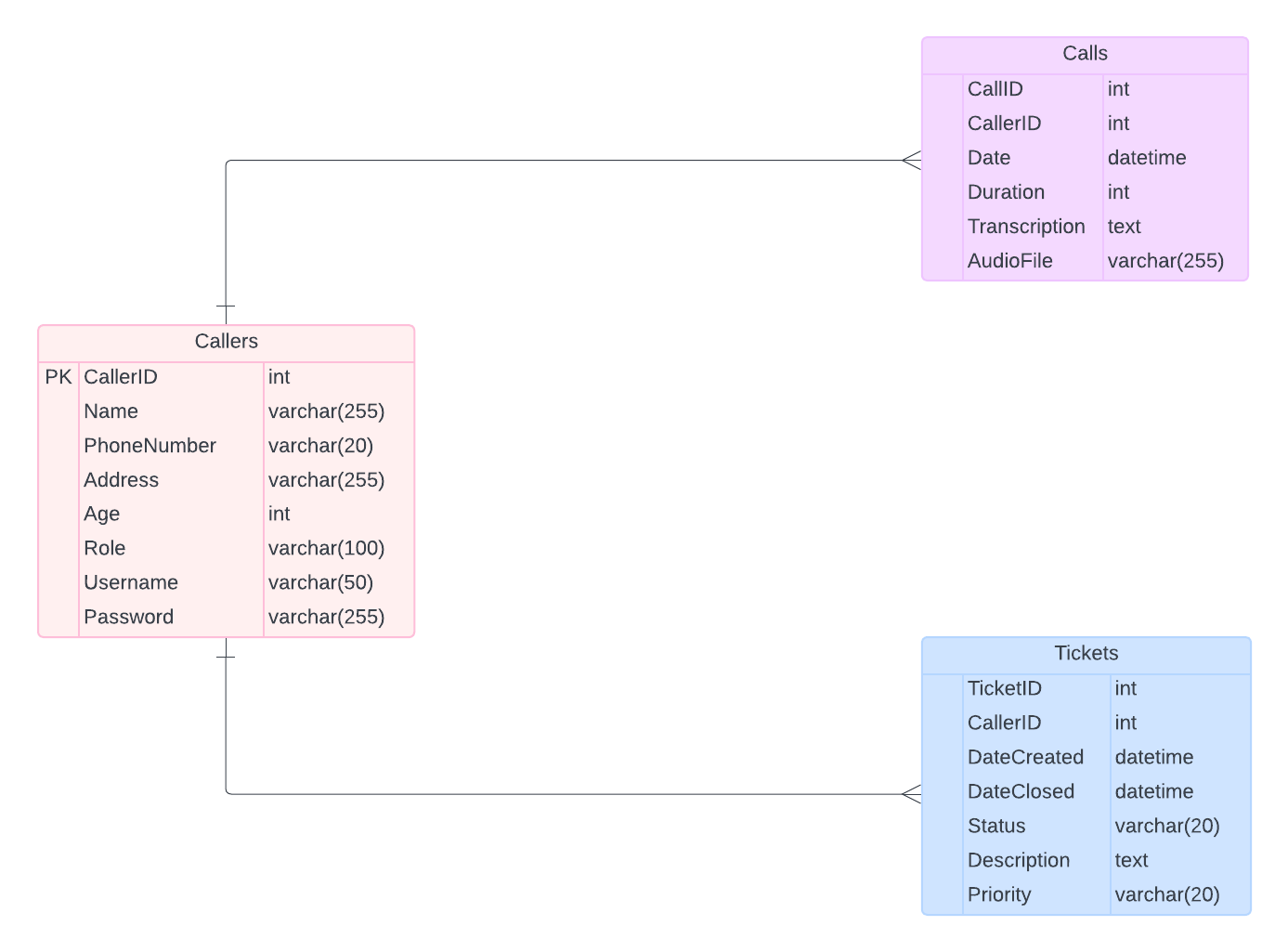
Entities:

1. Callers:
   1. Attributes:
      1. CallerID (Primary Key): Unique identifier for each user.
      2. Name: Full name of the user.
      3. Phone Number: Contact number of the user.
      4. Address: Residential address of the user.
      5. Age: Age of the user.
      6. Username: Unique username for logging into the system.
      7. Password: Securely stored password for user authentication.
2. Tickets:
   1. Attributes:
      1. TicketID (Primary Key): Unique identifier for each ticket.
      2. CallerID (Foreign Key): References the user who raised the ticket.
      3. Date Created: Timestamp indicating when the ticket was created.
      4. Date Closed: Timestamp indicating when the ticket was resolved or closed.
      5. Status: Status of the ticket (e.g., open, in progress, resolved).
      6. Description: Description of the issue or query raised in the ticket.
      7. Priority: Priority level assigned to the ticket (e.g., low, medium, high).
3. Calls:
   1. Attributes:
      1. CallID (Primary Key): Unique identifier for each call.
      2. UserID (Foreign Key): References the user who made the call.
      3. Date: Timestamp indicating when the call was made.
      4. Duration: Duration of the call-in seconds.
      5. Transcription: Textual transcription of the call content.
      6. Audio File: Binary data or reference to the audio recording of the call.

Relationships:

1. One-to-Many Relationship between Callers and Tickets:
   1. Each user can raise multiple tickets.
   2. Each ticket is raised by exactly one user.
   3. This relationship is established through the CallerID attribute in the Tickets table, referencing the CallerID attribute in the Users table.
2. One-to-Many Relationship between Callers and Calls:
   1. Each user can make multiple calls.
   2. Each call is made by exactly one user.
   3. This relationship is established through the CallerID attribute in the Calls table, referencing the CallerID attribute in the Users table.

The Preliminary Data Model will look something like below:



**Database Tools and Languages:**

1. Database Management System (DBMS):
   1. **MySQL, PostgreSQL, or SQLite** are suitable options considering reliability, security, and scalability requirements.
2. Languages:
   1. **SQL** for database querying and management.
3. For the backend of the web application, languages like **Python** (with frameworks like **Django or Flask**), **Node.js, or Java** (with Spring Boot) can be used.

**Storage Types:**

1. Caller Data:
   1. Store in the database with appropriate encryption for sensitive information like passwords.
   2. Use VARCHAR or TEXT for variable-length attributes like name and address.
2. Ticket Information:
   1. Store ticket data in the database with appropriate data types for each attribute.
   2. Use ENUM or similar constructs for fields like status and priority.
3. Call Data:
   1. Store call metadata (date, duration, etc.) in the database.
   2. Store audio files in a file system or cloud storage (e.g., Amazon S3).
   3. Transcriptions can be stored in the database or as text files associated with the call records.

**Data Privacy:**

1. Consent Management:
   1. Implement a robust consent management system where callers explicitly provide consent for their information to be collected, stored, and processed.
   2. Ensure that consent is obtained before recording any calls or collecting personal information.
2. Anonymization and Pseudonymization:
   1. Wherever possible, anonymize or pseudonymize caller information to minimize the risk of identification.
   2. Use identifiers that cannot be easily linked back to individuals.
3. Data Minimization:
   1. the service's purpose.
   2. Avoid storing unnecessary personal details.
4. Transparency:
   1. Provide clear and transparent information to callers about what data is being collected, how it will be used, and who will have access to it.

**Data Security:**

1. Encryption:
   1. Encrypt sensitive data both in transit and at rest using industry-standard encryption algorithms.
   2. Implement end-to-end encryption for communications between the application and the database.
2. Access Control:
   1. Enforce strict access controls to ensure that only authorized personnel can access sensitive data.
   2. Implement role-based access control (RBAC) to limit access based on job roles and responsibilities.
3. Data Integrity:
   1. Implement mechanisms to detect and prevent unauthorized changes to data.
   2. Use checksums or digital signatures to verify the integrity of data.
4. Auditing and Monitoring:
   1. Log all access to sensitive data and regularly review access logs for any unauthorized activities.
   2. Implement intrusion detection systems to detect and respond to security breaches promptly.
5. Secure Development Practices:
   1. Follow secure coding practices to minimize the risk of vulnerabilities such as SQL injection and cross-site scripting (XSS).
   2. Conduct regular security assessments and penetration testing to identify and address potential vulnerabilities.
6. Data Retention and Disposal:
   1. Define clear data retention policies specifying how long different types of data will be retained.
   2. Implement secure data disposal procedures to ensure that data is securely deleted when no longer needed.

**Potential APIs:**

1. User Management:
   1. Create User:
      1. Endpoint: POST /api/users
      2. Description: Create a new user.
      3. Request Body: User details (name, phone number, address, age, role, username, password).
      4. Response: Newly created user details.
   2. Get User Profile:
      1. Endpoint: GET /api/users/{userID}
      2. Description: Get details of a specific user.
      3. Response: User details.
   3. Update User Profile:
      1. Endpoint: PUT /api/users/{userID}
      2. Description: Update user profile information.
      3. Request Body: Updated user details.
      4. Response: Updated user details.
   4. Delete User:
      1. Endpoint: DELETE /api/users/{userID}
      2. Description: Delete a user account.
      3. Response: Success message.
2. Ticket Management:
   1. Create Ticket:
      1. Endpoint: POST /api/tickets
      2. Description: Create a new ticket.
      3. Request Body: Ticket details (callerID, description, priority).
      4. Response: Newly created ticket details.
   2. Get Ticket Details:
      1. Endpoint: GET /api/tickets/{ticketID}
      2. Description: Get details of a specific ticket.
      3. Response: Ticket details.
   3. Update Ticket Status:
      1. Endpoint: PUT /api/tickets/{ticketID}
      2. Description: Update ticket status (e.g., mark as resolved).
      3. Request Body: Updated ticket status.
      4. Response: Updated ticket details.
   4. List User's Tickets:
      1. Endpoint: GET /api/users/{userID}/tickets
      2. Description: Get a list of tickets associated with a specific user.
      3. Response: List of ticket details.
3. Call Handling:
   1. Record Call:
      1. Endpoint: POST /api/calls
      2. Description: Record a call and save call details.
      3. Request Body: Call details (callerID, duration, transcripemtion, audio file).
      4. Response: Success message.
   2. Get Call Details:
      1. Endpoint: GET /api/calls/{callID}
      2. Description: Get details of a specific call.
      3. Response: Call details.
4. Authentication:
   1. User Authentication:
      1. Endpoint: POST /api/auth/login
      2. Description: Authenticate user credentials.
      3. Request Body: User credentials (username, password).
      4. Response: Authentication token.
   2. Token Verification:
      1. Endpoint: POST /api/auth/verify-token
      2. Description: Verify the validity of an authentication token.
      3. Request Header: Authorization token.
      4. Response: Verification status.
5. Data Retrieval:
   1. Search Tickets:
      1. Endpoint: GET /api/tickets/search
      2. Description: Search for tickets based on specified criteria (e.g., status, priority).
      3. Query Parameters: Criteria for search (status, priority, date range, etc.).
      4. Response: List of matching tickets.
   2. Analytics:
      1. Endpoint: GET /api/analytics
      2. Description: Retrieve analytics data (e.g., ticket trends, call volume).
      3. Response: Analytics data.

(Guwani) Data Privacy and Security Measures:

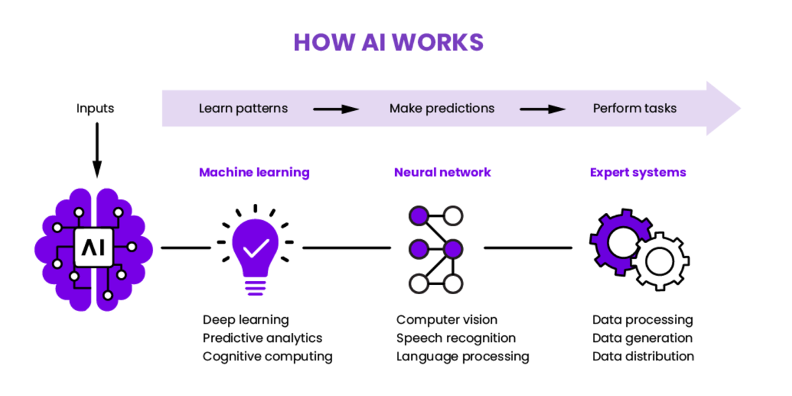
In handling sensitive information such as caller details and inquiries, data privacy and security are paramount.

Implementing measures such as consent management, anonymization, encryption, access control, and data retention policies is essential to protect user data and ensure compliance with privacy regulations.

Transparency and user consent regarding data collection and usage are crucial aspects that need to be emphasized.

|  |  |
| --- | --- |
|  |  |
|  |  |

AI/ML - Enhancing Customer Experience and Operational Efficiency Through AI-Powered Ticket Routing



* Develop AI-powered chatbots to automate responses to common inquiries and provide instant support to customers. Train these chatbots using machine learning algorithms to improve their accuracy and responsiveness over time.

* **Recurrent Neural Networks (RNNs):** Useful for sequential data processing and can effectively model the context of conversations and understand the dependencies between words in sentences.
* **Reinforcement Learning (RL):** Can be used to train chatbots to interact with users in a goal-oriented manner and learn from the feedback received after each conversation turn.
* **Supervised Learning Algorithms (Logistic Regression):** These algorithms can be used to analyze user feedback (ratings, reviews) and classify it into relevant categories to identify areas for model improvement.
* **Hierarchical Reinforcement Learning**: This approach enables chatbots to engage in multi-turn conversations by learning to navigate through dialogue trees and select appropriate responses based on the current dialogue context.
* Use NLP techniques to improve the analysis of customer inquiries. This includes techniques such as sentiment analysis to understand the mood of messages, entity recognition to identify important details, and topic modeling to group inquiries into relevant categories.

* Utilize machine learning models to predict customer behavior and anticipate their needs. For example, use customer data to predict the volume of inquiries during peak hours or identify patterns in customer inquiries to address common issues.

* **Regression Analysis**: Apply linear regression, polynomial regression, logistic regression to identify relationships between various factors such as time of day, day of week, customer demographics and the volume or type of inquiries. These models can help predict future inquiry volumes based on relevant predictors.
* **Classification Models**: D trees, random forests, support vector machines (SVM) can be used to classify inquiries into different categories. By analyzing historical inquiry data and identifying patterns in customer inquiries, these models can predict the likelihood of certain types of inquiries occurring during specific time periods or under certain conditions.
* **Clustering Analysis**: Apply clustering algorithms such as k-means clustering or hierarchical clustering to group similar inquiries together based on their characteristics (e.g., content, keywords, customer attributes). By clustering inquiries into meaningful groups, these models can identify common issues or topics and predict the occurrence of similar inquiries in the future.
* **Neural Networks:** Utilize deep learning models such as recurrent neural networks (RNNs) or long short-term memory networks (LSTMs) to analyze sequences of customer inquiries over time, and make predictions. These models can capture complex patterns and dependencies in the data and provide accurate forecasts of future customer behavior.

* Develop algorithms to automatically route incoming tickets to the most appropriate staff members based on the nature of the inquiry and their expertise. This can help streamline ticket management and reduce response times.
* **Machine Learning-based Routing**: Train machine learning models on historical ticket data to learn patterns and correlations between ticket attributes and the appropriate staff members. These models can then predict the best routing option for new incoming tickets based on their characteristics.
* **Rule-based Routing**: Create predefined conditions to route tickets based on specific criteria such as keywords in the inquiry, customer attributes or ticket attributes. For example, tickets containing the keyword "Emergency" could be routed to the emergency department, while tickets related to technical issues could be routed to the IT department.
* **Content-based Filtering**: Analyze the content of incoming tickets using natural language processing techniques to extract relevant features such as keywords, topics. Then, match these features with the skills of staff members to determine the best routing option.
* **Reinforcement Learning**: Apply reinforcement learning techniques to dynamically adjust routing decisions based on feedback and outcomes. This approach allows the system to learn and adapt its routing strategy over time, optimizing for efficiency and customer satisfaction.

Additional Points (guwani)

Interpretability and Explainability:

Ensure that the AI models used for ticket routing and customer inquiries are interpretable and explainable. This means that the decisions made by the models can be understood and justified by human operators. Techniques such as feature importance analysis, model visualization, and generating explanations for model predictions can enhance trust and transparency in the AI system.

Continuous Model Monitoring and Maintenance:

Implement mechanisms for continuous monitoring and maintenance of the AI models to ensure they remain effective and up to date. This involves monitoring model performance metrics, detecting drift or degradation in model accuracy, and retraining models with fresh data periodically to adapt to changing patterns and trends in customer inquiries.

Ethical Considerations and Bias Mitigation:

Address ethical considerations related to AI usage, such as bias in model predictions and potential negative impacts on certain demographic groups. Implement bias detection and mitigation techniques, such as fairness aware algorithms and bias audits, to minimize discriminatory outcomes and ensure fairness and inclusivity in the ticket routing process.