Voice Biometric Integration

Product Requirements Document, Version 2
November 28, 2016

Team Tres Commas

Members:
Jonathan Easterman, Sasha Shams,
Arda Ungun, Carson Holoien, Vince Nicoara

Mentors:
Mike Weaver, Colin Kelley, James Brown
Chandra Krintz, Ankita Singh

Table of Contents

Introduction

PROBLEM

We want to bring authentication to the 21st century. Existing methods of automated user authentication over telephony have many drawbacks; entering codes via touch-tone is cumbersome and error-prone, existing robo-callers are unreliable and repetitive.

VISION

We will utilize modern voice biometrics to identify and authorize users in a seamless process. No more having to remember years-old pins and passwords. No more having to share private, sensitive information with strangers over the phone as you answer security questions. Using our technology, an individual's identity will be ascertained in the utterance of a sentence.

Implementing voice authentication will improve convenience and user experience for customers, while automation will be cheaper and more efficient for businesses. Authenticating a user can be 100% autonomous, which will speed up the process for both users and call center employees.

COMPETITIVE ANALYSIS

There are many APIs that offer a voice biometrics service via the Internet, but there is only one company that offers a complete voice authentication service. Nuance offers voice authentication as a costly and monolithic product[1]. We plan to offer an open-source, configurable product that will target smaller companies.

INNOVATION

Our solution is secure, lightweight, easy to deploy, and modular. We will offer improved security by integrating automated voice authentication, and target small to mid level customers through our lightweight architecture and ease of deployment. The plug-and-play nature of our application allows other developers to leverage our product to the services that suit their needs. We implement Microsoft's voice authentication API, but future developers can choose to replace that portion of our project with an improved or proprietary solution instead.

[1] We tried contacting Nuance about their product and potential support for developers and have not received a reply.

Project Description

OBJECTIVES

Our project aims to add another layer of security to Invoca's call systems by adding voice authentication and shortening the amount of time spent by employees authenticating users.

BACKGROUND

In existing solutions, when users call to a make changes to one of their accounts, they must verify their identity by answering a series of security questions. These security questions can include things like their social security number and other personal questions. Going about verification in this manner has a few issues that can be improved upon. First of all, it can take a long time – maybe the user does not remember their specific account information ("What is your favorite animal?"). Secondly, it is not 100% secure – if a nefarious character (or even the individual on the other end of the call) got ahold of their personal information, they can access the user's account and wreak havoc. Our voice authentication vision seeks to provide another layer of security as well as make the entire process quicker for the user and ultimately improve efficiency for call center administrators.

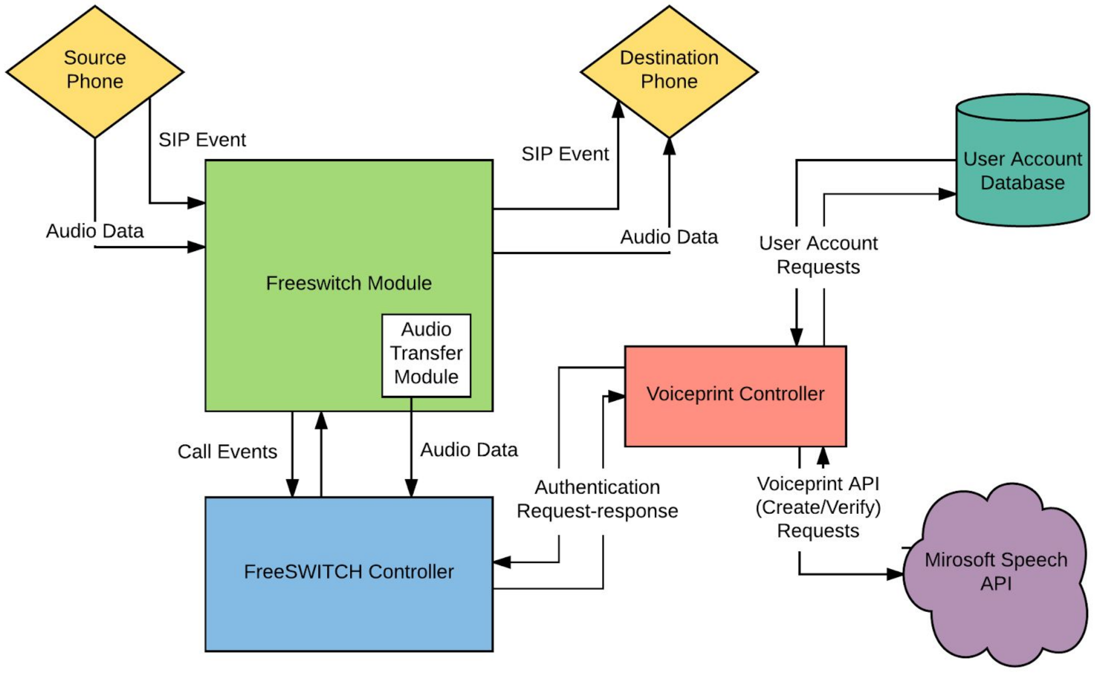
ASSUMPTIONS

- Users desire faster authentication when accessing their account over the phone
- Users desire to have more security when accessing their account (2FA)
- Account administrators desire to have users with more secure accounts, so that they do not have to deal with the issues of compromised accounts
- Account administrators desire to have call centers with improved efficiency as time is wasted authenticating users manually
- Users' voices may fluctuate short term and change over time, so alternative methods of authentication should be in place to enable users to update their voice profiles
- Vulnerabilities in voice authentication technology exist, therefore there must be preventative measures in place such as randomization of voiceprint phrases

System Architecture Overview

HIGH-LEVEL DIAGRAM

Illustrated below is our preliminary system architecture. Each piece handles specific responsibilities, which are outlined. The Freeswitch Module, Ruby Main Controller, Voiceprint Controller, and User Account Database are all within their own docker containers and communicate over sockets.



FreeSWITCH Module (Open Source Telephone Platform):

This module handles all of the protocols to initiate a phone call and conversate. The freeswitch module monitors call events such as calling, answering, hanging up, and entering digits. These events are transferred to the FreeSWITCH controller in the form of call events.

FreeSWITCH Controller (Ruby):

This module contains the main “brains” of the whole system. It facilitates the IVR (Interactive Voice Response) tree. In our system, the IVR tree maintains the current state of the caller and transfers them accordingly based on their user input (Ex: “Prompt: Press 1 for Directions, 2 for operator”). Based on the current state of the caller, the FreeSWITCH Controller interfaces with the Voiceprint Controller to retrieve user info and issue requests for authentication, creation, and deletion.

Voiceprint Controller (Sinatra Web Framework):

The voiceprint controller handles the logistics behind creating and retrieving users and their voiceprints in the system. The Voiceprint controller is built on a Sinatra framework and maintains a web server for accepting requests and serving the web interface..The FreeSWITCH controller issues requests in the form of API calls to our internally built Sinatra API.

To create a voiceprint, a request is sent from the FreeSWITCH controller to the Voiceprint controller with the user's phone number and 3 recorded voice samples. The Voiceprint Controller then takes this data and forms an API request to Microsoft Speech API to enroll the user and returns a profile UUID. The Voiceprint controller then uses this profile UUID to create a customer account in our User Accounts Database with attributes such as First Name, Last Name, Phone Number, and Verification Profile UUID.

When the controller receives a verify request, it will retrieve the voiceprint UUID from the database as a parameter for the API request to verify. The response is passed to the FreeSWITCH controller and ultimately presented to the customer.

User Account Database (SQLite3) :

Due to privacy reasons, Microsoft Speaker Recognition API does not store personal info with a user's voiceprint and simply creates a verification profile ID. Therefore, we must maintain our own database of that a user's name, phone number, and verification profile ID for later retrieval.

For ease of setup, we implemented our User Account Database using SQLite3. The Voiceprint controller then interfaces with database methods to insert, update, and delete entries in the database.

Voice Authentication API (Microsoft Speaker Recognition API):

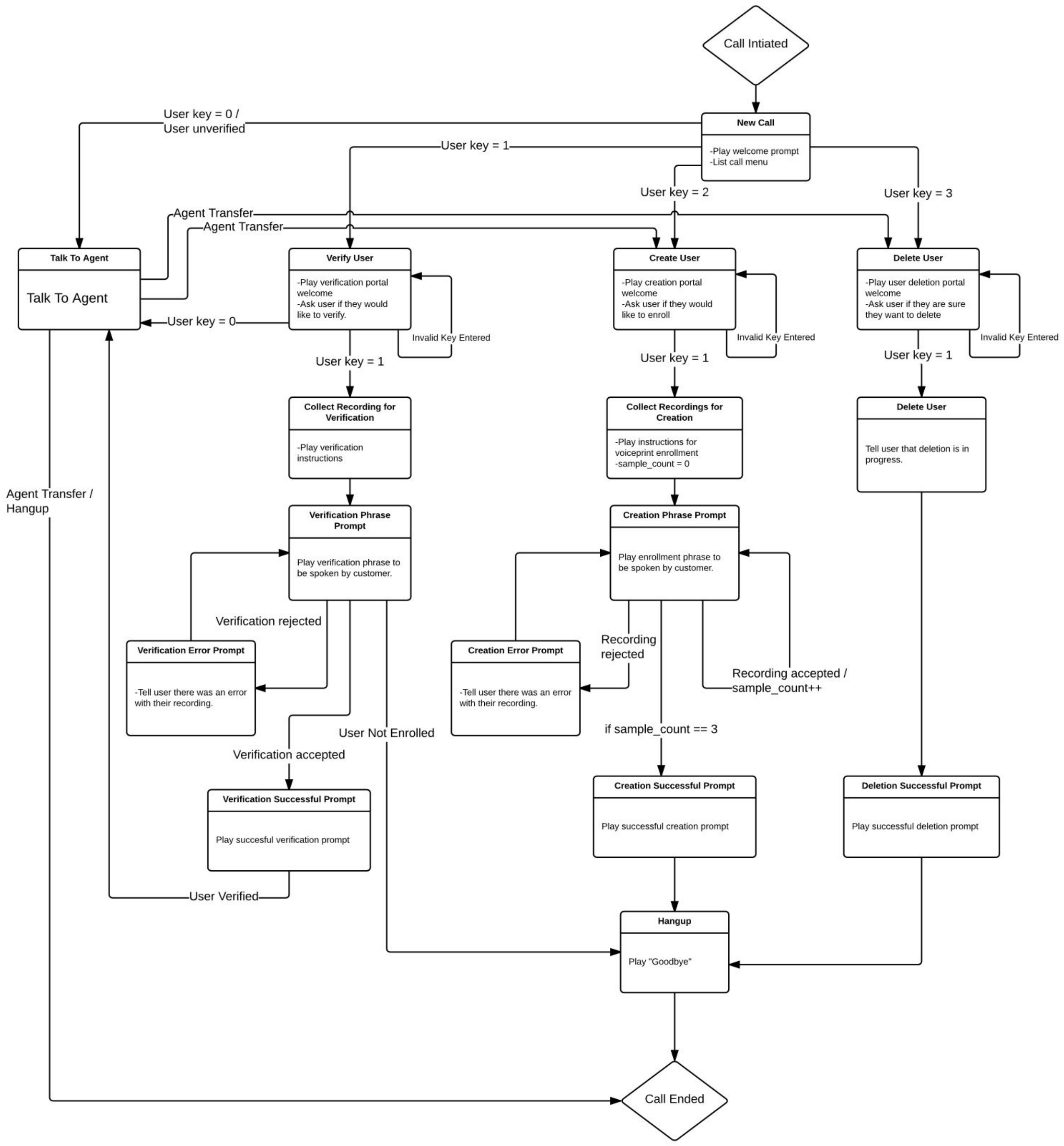
After much analysis of the voice authentication APIs (Voice API Analysis) we decided on Microsoft Cognitive Services - Speaker Recognition API. All of the voice APIs shared similar limitations, only specific phrases provided by the API could be spoken by the user to be verified. These phrases bring out certain characteristic patterns that are used to create a unique user's voiceprint. The API will be used to handle all speech processing.

To enroll with the Speaker Recognition API, the customer needs to speak a given phrase 3 times - after they are enrolled they are ready to authenticate with their voice.

To verify with the Speaker Recognition API, the customer needs to speak the *same* phrase as enrollment 1 time. The API then processes the voice sample and returns a confidence score on authentication - our application only accepts high confidence responses to ensure security.

USER EXPERIENCE DIAGRAM

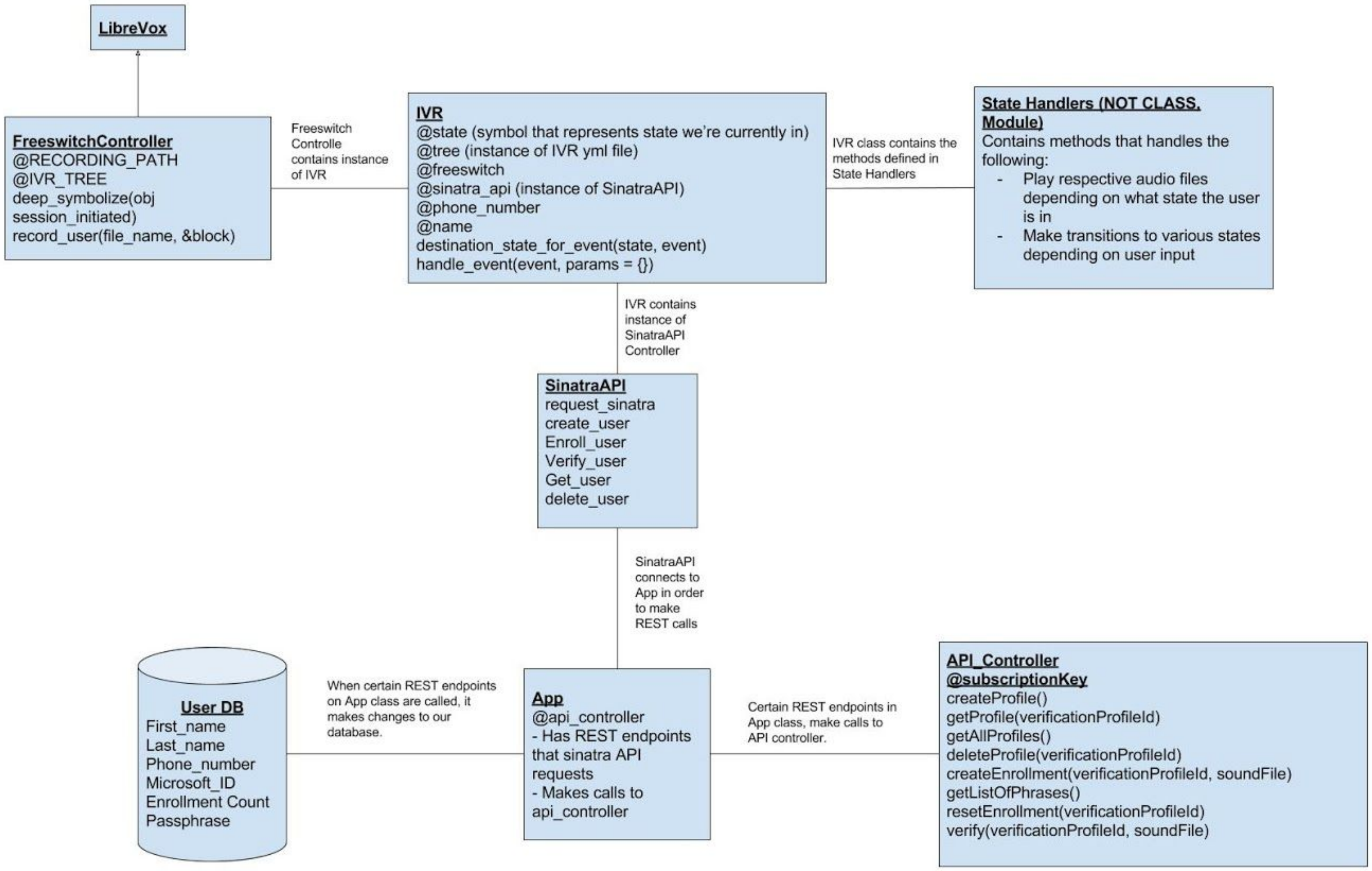
In addition to the high level system diagram, we have charted the user's experience through calling into our system. In the diagram below, each container is a state with its corresponding output in the description - in most cases this is an audio prompt played to the user. User inputs/Agent transfers are indicated by transitions between states. This diagram helps illustrate our user stories.



Call Intiated
New Call
-Play welcome prompt
-List call menu
User key = 0 / User unverified
User key = 1
User key = 2
User key = 3
Agent Transfer
Agent Transfer
Talk To Agent
Talk To Agent
Verify User
-Play verification portal welcome
-Ask user if they would like to verify.
User key = 0
Invalid Key Entered
Create User
-Play creation portal welcome
-Ask user if they would like to enroll
Invalid Key Entered
Delete User
-Play user deletion portal welcome
-Ask user if they are sure they want to delete
Invalid Key Entered
User key = 1
User key = 1
User key = 1
Collect Recording for Verification
-Play verification instructions
Collect Recordings for Creation
-Play instructions for voiceprint enrollment
-sample_count = 0
Delete User
Tell user that deletion is in progress.
Agent Transfer / Hangup
Verification Phrase Prompt
Play verification phrase to be spoken by customer.
Creation Phrase Prompt
Play enrollment phrase to be spoken by customer.
Verification rejected
Verification Error Prompt
-Tell user there was an error with their recording.
Creation Error Prompt
-Tell user there was an error with their recording.
Recording rejected
Recording accepted / sample_count++
if sample_count == 3
User Not Enrolled
Verification accepted
Verification Successful Prompt
Play succesful verification prompt
Creation Successful Prompt
Play successful creation prompt
Deletion Successful Prompt
Play successful deletion prompt
User Verified
Hangup
Play "Goodbye"
Call Ended

UML/CLASS DIAGRAM

Below we have outlined the classes as well as the database in our project. It illustrates the functionality as well as the relationship between all of the high level elements in our design. To clarify, LibreVox is an open source wrapper around FreeSWITCH's mod_event_socket API; it gives the ability to establish a TCP connection and send commands to FreeSWITCH.


LibreVox
FreeswitchController
@RECORDING_PATH
@IVR_TREE
deep_symbolize(obj
session_initiated)
record_user(file_name, &block)
Freeswitch Controlle contains instance of IVR
IVR
@state (symbol that represents state we're currently in)
@tree (instance of IVR yml file)
@freeswitch
@sinatra_api (instance of SinatraAPI)
@phone_number
@name
destination_state_for_event(state, event)
handle_event(event, params = {})
IVR class contains the methods defined in State Handlers
State Handlers (NOT CLASS, Module)
Contains methods that handles the following:
- Play respective audio files depending on what state the user is in
- Make transitions to various states depending on user input
IVR contains instance of SinatraAPI Controller
SinatraAPI
request_sinatra
create_user
Enroll_user
Verify_user
Get_user
delete_user
SinatraAPI connects to App in order to make REST calls
User DB
First_name
Last_name
Phone_number
Microsoft_ID
Enrollment Count
Passphrase
When certain REST endpoints on App class are called, it makes changes to our database.
App
@api_controller
- Has REST endpoints that sinatra API requests
- Makes calls to api_controller
Certain REST endpoints in App class, make calls to API controller.
API_Controller
@subscriptionKey
createProfile()
getProfile(verificationProfileId)
getAllProfiles()
deleteProfile(verificationProfileId)
createEnrollment(verificationProfileId, soundFile)
getListOfPhrases()
resetEnrollment(verificationProfileId)
verify(verificationProfileId, soundFile)

SYSTEM SEQUENCE DIAGRAMS

DIAGRAM 1 - CREATE ACCOUNT

Below we have diagramed the system sequence for creating an account in our system. The elements at the top of the diagram are color coded to correspond with blocks of the high level system diagram illustrated earlier.

Preconditions:

- User has called and established connection with FreeSWITCH
- FreeSWITCH has played initial menu prompt

CREATE ACCOUNT DIAGRAM

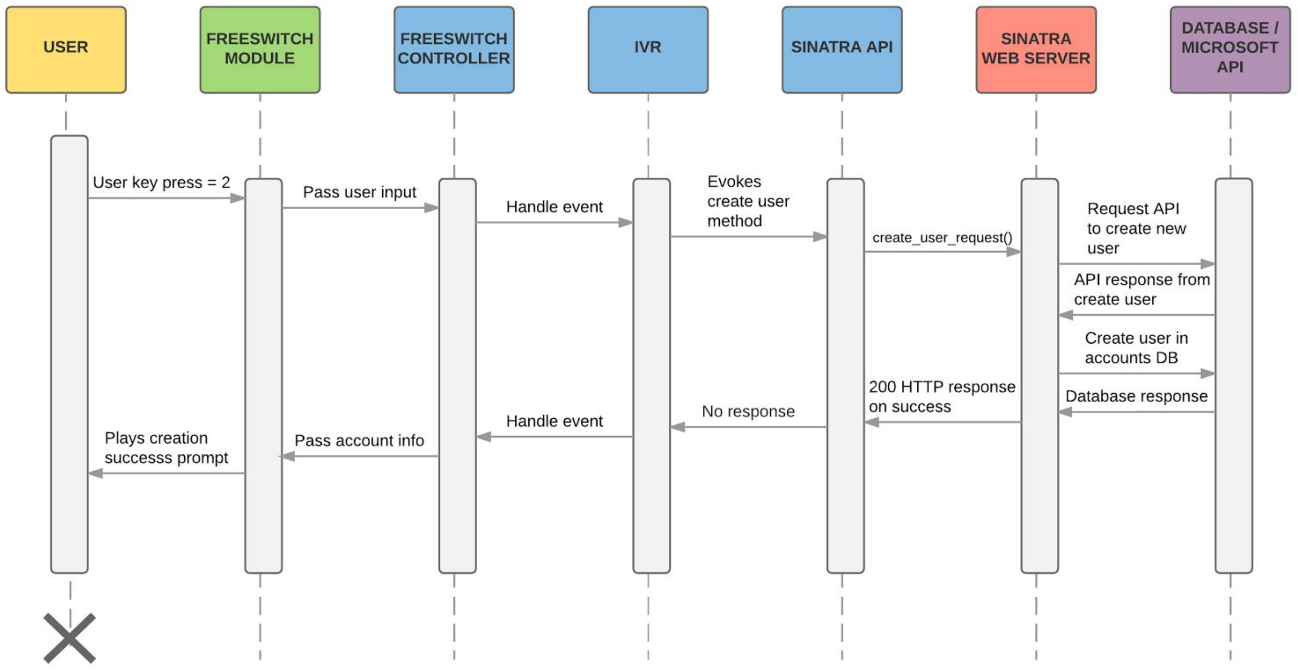


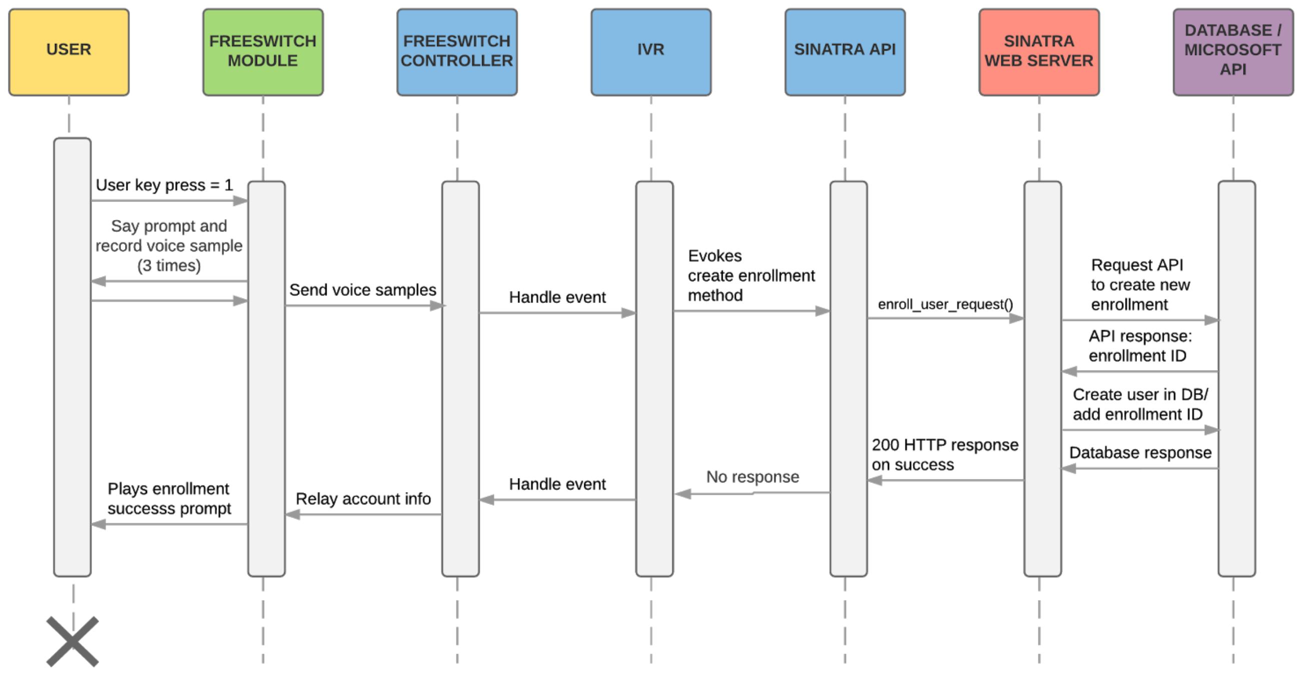
DIAGRAM 2 - CREATE ENROLLMENT

Below we have diagramed the system sequence for enrolling your voice for authentication within our system. The elements at the top of the diagram are color coded to correspond with blocks of the high level system diagram illustrated earlier.

Preconditions:

- User has called and established connection with FreeSWITCH
- FreeSWITCH has played initial menu prompt

CREATE ENROLLMENT DIAGRAM



USER
FREESWITCH MODULE
FREESWITCH CONTROLLER
IVR
SINATRA API
SINATRA WEB SERVER
DATABASE / MICROSOFT API
User key press = 1
Say prompt and record voice sample (3 times)
Send voice samples
Handle event
Evokes create enrollment method
enroll_user_request()
Request API to create new enrollment
API response: enrollment ID
Create user in DB/ add enrollment ID
Database response
200 HTTP response on success
No response
Handle event
Relay account info
Plays enrollment successs prompt

DIAGRAM 3 - VERIFY USER

Below we have diagramed the system sequence for verifying your account with your voice in our system. The elements at the top of the diagram are color coded to correspond with blocks of the high level system diagram illustrated earlier

Preconditions:

- User has an account enrolled for authentication and has already called and established connection with FreeSWITCH
- FreeSWITCH has played initial menu prompt

ACCOUNT VERIFICATION SUCCESS DIAGRAM

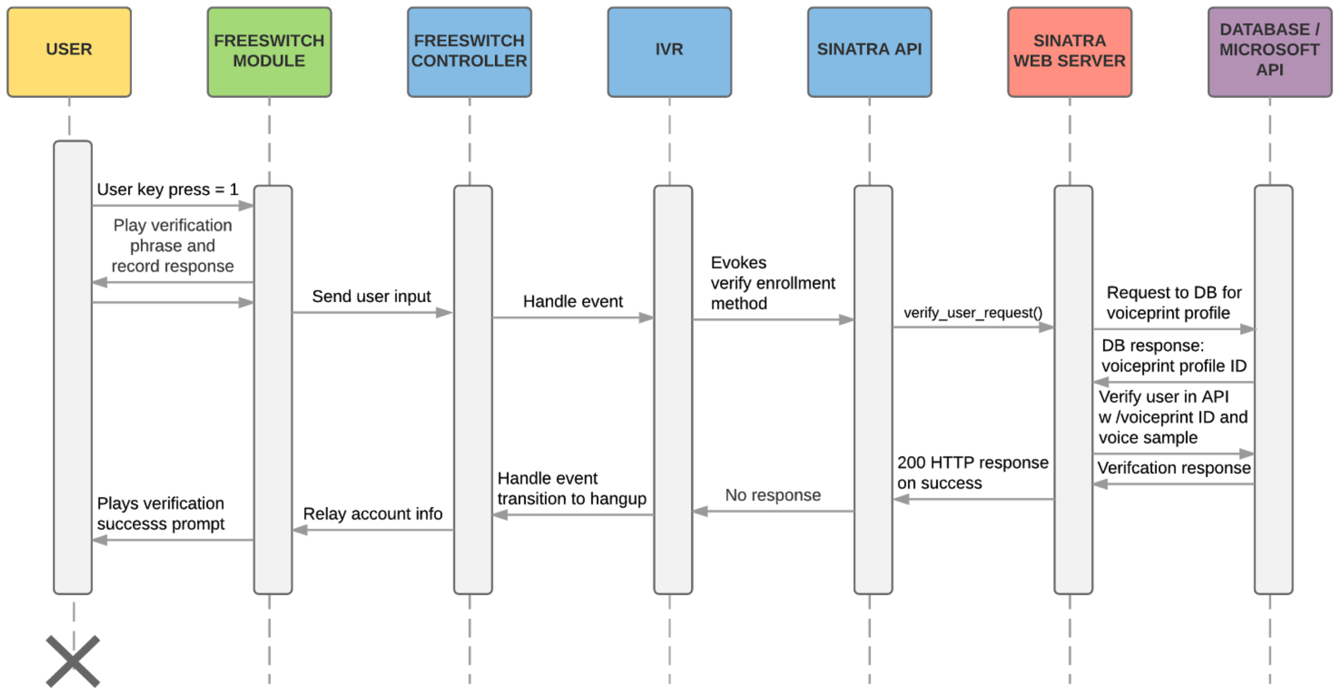

USER
FREESWITCH MODULE
FREESWITCH CONTROLLER
IVR
SINATRA API
SINATRA WEB SERVER
DATABASE / MICROSOFT API
User key press = 1
Play verification phrase and record response
Send user input
Handle event
Evokes verify enrollment method
verify_user_request()
Request to DB for voiceprint profile
DB response: voiceprint profile ID
Verify user in API w /voiceprint ID and voice sample
Verifcation response
200 HTTP response on success
No response
Handle event transition to hangup
Relay account info
Plays verification successs prompt

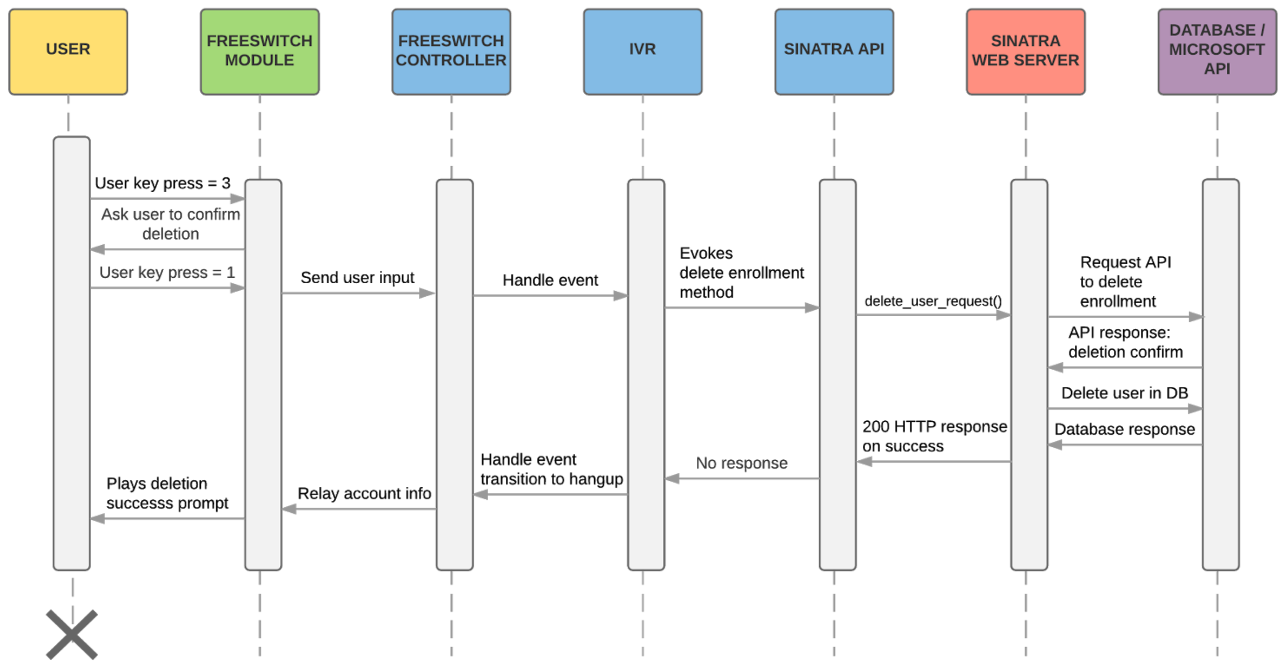
DIAGRAM 4 - DELETE USER

Below we have diagramed the system sequence for creating an account in our system. The elements at the top of the diagram are color coded to correspond with blocks of the high level system diagram illustrated earlier

Preconditions:

- User has an account and has already called and established connection with FreeSWITCH
- FreeSWITCH has played initial menu prompt
- User is authenticated

DELETE ENROLLMENT DIAGRAM



Product Requirements

PRIORITIZED USER STORIES

User Story 1	
Story	As a **customer**, I can call the call center so that I can take care of business.
Acceptance Criteria	A customer can call a phone number and have their call answered properly by the system.
Tests	• Verify that the FreeSWITCH controller can accept a new call and can park it properly. • Verify that the customer's call information is inserted into the call log database.
Prototype	https://github.com/sashashams/InvocaCapstone/tree/prdv2/freeswitch_conf/freeswitch https://github.com/sashashams/InvocaCapstone/tree/prdv2/freeswitch_controller https://github.com/sashashams/InvocaCapstone/blob/prdv2/freeswitch_controller/spec/sinatra_api_spec.rb

User Story 2	
Story	As a **customer**, I can call the call center and create a voice print profile so that I can use voice authentication in the future.
Acceptance Criteria	A customer can record 3 samples of their voice over the phone and create an account in the system with associated information.
Tests	● Verify that the FreeSWITCH controller properly records and saves 3 samples of the customer's voice. ● Verify that the voice controller properly uses the speech API to enroll customers with voice samples. ● Verify that the voice controller properly adds new customers in the account database with customer's phone number, name, and voice profile ID. ● Verify all steps of user story are properly inserted into the call log database.
Prototype	https://github.com/sashashams/InvocaCapstone/tree/prdv2/freeswitch_controller/ivr_tree https://github.com/sashashams/InvocaCapstone/tree/prdv2/freeswitch_controller/freeswitch_files/voice_prompts https://github.com/sashashams/InvocaCapstone/blob/prdv2/freeswitch_controller/spec/sinatra_api_spec.rb

User Story 3	
Story	As a **customer**, I can call the call center and be authenticated using my voice print before talking to an agent so that I can: ● Authenticate faster and painlessly ● Add another level of security ● Keep my personal details private
Acceptance Criteria	A customer can call in, transfer to the verification portal, speak their enrollment phrase, and be verified in the system by their voice.
Tests	● Verify that the FreeSWITCH controller properly records and saves samples of the user's voice. ● Verify that the voice controller properly retrieves an existing user account and associated voice profile ID from the user's phone number. ● Verify that the voice controller properly uses the speech API to verify a user with their voice sample. ● Verify that speech API does not authenticate invalid voice samples. ● Verify all actions in the user story are properly inserted into the call log database.
Prototype	https://github.com/sashashams/InvocaCapstone/blob/prdv2/freeswitch_controller/freeswitch_controller.rb https://github.com/sashashams/InvocaCapstone/tree/prdv2/freeswitch_controller/ivr_tree

User Story 4	
Story	As a **customer support agent,** I can know whether a customer was properly authenticated when their call is transferred to me.
Acceptance Criteria	A customer support agent can see the authentication status of a customer when a customer call is transferred to the them.
Tests	● Verify that the FreeSWITCH controller properly bridges a call with an agent. ● Verify that the customer support agent web interface properly displays an inbound customer's authentication status (authenticated or unauthenticated). ● Verify that speech API does not authenticate invalid voice samples. ● Verify all steps of user story are properly inserted into the call log database.
Prototype	https://github.com/sashashams/InvocaCapstone/tree/prdv2/freeswitch_controller/ivr_tree https://github.com/sashashams/InvocaCapstone/blob/prdv2/sinatra_app/app.rb

User Story 5	
Story	As a **customer,** I can bypass the authentication system so that I can authenticate with a customer support agent directly.
Acceptance Criteria	A customer can call in and immediately opt to speak with an agent.
Tests	● Verify that the FreeSWITCH controller properly bridges a call with an agent. ● Verify that the customer support agent web interface properly displays an unauthenticated status. ● Verify all steps of user story are properly inserted into the call log database.
Prototype	https://github.com/sashashams/InvocaCapstone/tree/prdv2/freeswitch_controller/ivr_tree https://github.com/sashashams/InvocaCapstone/blob/prdv2/sinatra_app/app.rb

User Story 6	
Story	As a **customer,** I can delete my voiceprint so that I can remove my voiceprint data and customer account from the system.
Acceptance Criteria	If a customer is authenticated, they can delete their voiceprint and customer account data from the system using the voiceprint deletion portal.
Tests	• Verify that the voice controller properly removes customer account information from the accounts database. • Verify that the voice controller uses the speech API to remove all enrollments for the customer. • Verify all steps of user story are properly inserted into the call log database.
Prototype	https://github.com/sashashams/InvocaCapstone/tree/prdv2/freeswitch_controller/ivr_tree https://github.com/sashashams/InvocaCapstone/blob/prdv2/sinatra_app/app.rb

User Story 7	
Story	As a **customer,** I can update my voiceprint so that I can use a more recent voiceprint in the future.
Acceptance Criteria	If a customer is authenticated, a customer can update their voiceprint data from the voiceprint update portal.
Tests	● Verify that the voice controller properly uses the speech API to authenticate the customer. ● Verify that the voice controller properly uses speech API to remove all enrollments for the current customer. ● Verify that the FreeSWITCH controller properly records and saves 3 samples of the customer's voice. ● Verify that the voiceprint controller properly uses the speech API to enroll the customer's *new* voice samples. ● Verify that the customer account information is properly updated in the accounts database. ● Verify all steps of user story are properly inserted into the call log database.
Prototype	https://github.com/sashashams/InvocaCapstone/blob/prdv2/sinatra_app/voice_api_controller/api_controller.rb

User Story 8	
Story	As a **call center manager**, I can see the current state of a call in the system by its call ID.
Acceptance Criteria	A call center manager can retrieve the current state of a customer's call in the system from the customer's UUID.
Tests	● Verify call center manager interface displays current state of call. ● Verify that the interface updates as the customer progresses through the call plan. ● Verify all steps of user story are properly inserted into the call log database.
Prototype	https://github.com/sashashams/InvocaCapstone/blob/prdv2/sinatra_app/views/call_log.erb https://github.com/sashashams/InvocaCapstone/blob/prdv2/sinatra_app/app.rb

User Story 9	
Story	As a **customer support agent,** I can see the user's history of states in a call so that I can offer better support.
Acceptance Criteria	When a customer is transferred to an agent, the agent can view the full history of states the customer visited in a web interface.
Tests	● Verify customer support agent interface displays full history of customer's call states. ● Verify that the interface updates as the customer progresses through the call plan. ● Verify all steps of user story are properly inserted into the call log database.
Prototype	https://github.com/sashashams/InvocaCapstone/blob/prdv2/sinatra_app/views/call_log.erb https://github.com/sashashams/InvocaCapstone/blob/prdv2/sinatra_app/app.rb

User Story 10	
Story	As a **call center admin,** I can setup the product for development and use.
Acceptance Criteria	• A call center admin can set up the product by downloading Docker and deploying dockerfiles. • A call center admin can set up a testing suite using a continuous integration platform. • A call center admin can setup a Microsoft Speech Recognition API account for use in the system.
Tests	• Manually verify that dockerfile configures installation properly. • Manually verify that CI platform properly executes tests. • Manually verify that API is interfacing properly.
Prototype	https://github.com/sashashams/InvocaCapstone/tree/prdv2/docker_files https://github.com/sashashams/InvocaCapstone/tree/prdv2/documentation

User Story 11	
Story	As a **customer support agent**, I can be connected to the next available customer's call.
Acceptance Criteria	A customer support agent can use the web interface to initiate the next call.
Tests	• Verify that FreeSWITCH properly transfers the call from the IVR system to the customer support agent. • Verify that system properly tracks customer support agents that are available and customers in queue. • Verify that customer support agent's status is changed from available to unavailable • Verify that the customer support agent's actions are logged in the database.

User Story 12	
Story	As a **call center admin**, I can set incoming calls to bypass the voiceprint system so that the support agents can continue receiving calls if other system components fail.
Acceptance Criteria	A call center admin can access the platform settings page to enable/disable the platform. When the platform is disabled, normal phone operations are implemented. When the platform is enabled, the IVR call plan is implemented.
Tests	● When disabled, verify that calls are properly bridged with an agent without use of the IVR tree. ● When enabled, verify that calls are properly transitioned into the IVR tree to be managed by the phone system.

User Story 13	
Story	As a **customer**, I can be put on hold until a customer service agent is available to take my call.
Acceptance Criteria	In the event of call backlog in the system due to unavailable customer support agents, the system will automatically hold an inbound call and add it to a queue.
Tests	● Verify that system checks whether any customer support agents are available. ● Verify that the customer is played smooth jazz while on hold. ● Verify that the system can transfer a held call to a newly available customer support agent. ● Verify all actions in the user story are properly inserted into the call log database.

User Story 14	
Story	As a **customer support agent**, I can put the customer call on hold so that I can retrieve additional information for the customer.
Acceptance Criteria	A customer support agent can pause and reconnect to a call through the call web interface.
Tests	● Verify that a customer support agent can hold the call through the web interface. ● Verify that the customer is played smooth jazz while on hold. ● Verify that the system can reconnect the held call. ● Verify that customer support agent is not shown as “free” in the system ● Verify all steps of user story are properly inserted into the call log database.

User Story 15	
Story	As a **customer support agent**: After authentication, I can transfer the customer to a specific state of the call plan so that a customer can create, update, or delete their voiceprint.
Acceptance Criteria	A customer support agent can properly transfer a customer to another state of the call plan using a web interface.
Tests	● Verify that calls are properly transitioned into the IVR tree to be managed by the phone system ● Verify that customer support agent is now marked as free in system. ● Verify all steps of user story are properly inserted into the call log database.

User Story 16	
Story	As a **customer support agent**: After authentication, I can add and modify phone numbers associated with a customer account, so that the customer can authenticate from multiple phone numbers.
Acceptance Criteria	A customer support agent can add and modify additional phone numbers to a customer's user account in the accounts database using a web interface.
Tests	● Verify that a customer can properly authenticate from multiple phone numbers. ● Verify that the customer's account is properly updated with new information. ● Verify all steps of user story are properly inserted into the call log database. ● Verify that interface updates to reflect new information

User Story 17	
Story	As a **customer support agent**, I can delete a customer's voiceprint and account data so that the customer can be removed from the system.
Acceptance Criteria	If a customer is authenticated, a customer support agent can remove the customer from the system using a web interface.
Tests	● Verify that customer's account information is removed from the accounts database. ● Verify that the customer's speech data is removed from the Speech API. ● Verify that a success message is displayed to the customer support agent on successful deletion. ● Verify all actions in the user story are properly inserted into the call log database.

User Story 18	
Story	As a **customer support agent**, I can update a customer’s account info on the customer’s behalf.
Acceptance Criteria	If a customer is authenticated, a customer support agent can update any account information in the accounts database using a web interface.
Tests	● Verify that the customer’s account info is properly updated in the accounts database. ● Verify all actions in the user story are properly inserted into the call log database.

User Story 19	
Story	As a **customer support agent**, I can bridge the call to a **different customer support agent** so that the customer can be routed to more appropriate service agents.
Acceptance Criteria	A customer support agent can bridge/transfer an active call to another customer support agent using a web interface.
Tests	● Verify that FreeSWITCH properly bridges active call to new customer support agent. ● Verify that previous agent is no longer a member of the call and marked “available”. ● Verify that new agent is marked “unavailable” ● Verify all actions in the user story are properly inserted into the call log database.

User Story 20	
Story	As a **customer support agent**, I can notify a call center manager to join the call to provide assistance with a customer.
Acceptance Criteria	A customer support agent can notify all available call center managers that assistance is needed in a call through the web interface.
Tests	● Verify that call center managers are properly notified. ● Verify that when a call center manager joins a call, the assistance notification is cleared. ● Verify all actions in the user story are properly inserted into the call log database.

User Story 21	
Story	As a **call center manager**, I can perform all customer support agent operations, so that I can provide support.
Acceptance Criteria	A call center manager can perform all the operations a customer support agent can through a similar web interface.
Tests	● Verify all functions of a customer support agent are available to the call center manager. ● Verify that any actions performed by a call center manager are properly logged.

User Story 22	
Story	As a **call center manager**, I can view the state of all calls currently in the system, so that I can manage customer support agents and calls.
Acceptance Criteria	A call center manager can use the web interface to view all current calls in the system with associated call ID's and current call state.
Tests	● Verify that multiple calls can be viewed in the call center manager web interface. ● Verify that when calls end, they are no longer displayed in the call center manager's web interface. ● Verify that when calls are initiated, they are properly updated in the call center manager's web interface.

User Story 23	
Story	As a **call center manager**, I can connect to an existing call so that I can monitor the quality of support, or provide assistance to a call.
Acceptance Criteria	● A call center manager can connect to any current call through the web interface. ● A call center manager can connect to a call after responding to a notification from a customer support agent.
Tests	● Verify that the FreeSWITCH allows a call center manager to listen in on calls selected from the web interface. ● Verify that the call center manager can properly connect to a call after responding to a customer support agent notification. ● Verify that a call center manager can properly exit a call, and allow that call to resume, after finishing with their assistance.

User Story 24	
Story	As a **call center manager**, I can retrieve previously recorded agent/customer conversations so that I can provide “quality assurance” for “training purposes”
Acceptance Criteria	A call center manager can use the web interface to access a database of recorded customer support agent sessions.
Tests	● Verify that non-sensitive conversations with a customer support agent are properly recorded in a database. ● Verify that interface properly shows customer support agent sessions. ● Verify that the customer support agent can replay recordings of customer support agent conversations.

User Story 25	
Story	As a **customer support agent**, I can see a queue of inbound calls and the authentication status of each call, so that I can select a call to connect to.
Acceptance Criteria	A customer support agent can use the web interface to connect to a specific queued call.
Tests	● Verify that the system can transfers a held call to the customer service agent ● Verify that call no longer appears in web interface ● Verify all steps of user story are properly inserted into the call log database.

User Story 26	
Story	As a **call center admin**, I can see the status of all calls in the system, and the agents they are connected with, so that I can monitor the system.
Acceptance Criteria	A call center admin can view the status of a calls, amount of inbound calls, availability of agents, and other system metrics from a web dashboard.
Tests	● Verify that the system metrics are properly updated and displayed in the web dashboard.

User Story 27	
Story	As a **call center admin**, I can view all the customers in the database, so that I can manage the database.
Acceptance Criteria	A call center admin can view customer accounts in the accounts database and retrieve other database metrics in a web interface.
Tests	● Verify that the database metrics are properly updated and displayed in the web dashboard.

User Story 28	
Story	As a **call center admin**, I can modify all the customers from the database, so that I can manage the database.
Acceptance Criteria	A call center admin can view customer accounts in the accounts database and modify the entries in the web interface.
Tests	● Verify that the database metrics are properly updated and displayed in the web dashboard. ● Verify that a call center admin can properly update information of an existing account entry.

User Story 29	
Story	As a **call center admin**, I can delete customers from the database, so that I can manage the database
Acceptance Criteria	A call center admin can view customer accounts in the accounts database and delete entries in the database in the event of system closure.
Tests	● Verify that the database metrics are properly updated and displayed in the web dashboard. ● Verify that a call center admin can properly delete information of an existing account entry. ● Verify that a call center admin can properly delete all entries in the user accounts database.

Appendix

TECHNOLOGIES EMPLOYED

Product

Docker	Development platform for building, shipping and running applications
Freeswitch	Open source telephony server
Ruby Sinatra	Lightweight web server used offer interface for call center agent
Librevox	Ruby wrapper for mod_event_socket Freeswitch interface
Microsoft Speaker Recognition API	Voice Authentication
SQLite3	User account database

Internal

Slack	Team Communication
Github	Source Version Control and code repository management.
Waffle.io	Agile/Scrum Task Board
Codeship	Continuous Integration code testing

FEATURES NOT IMPLEMENTED:

1. Conversational automated voice authentication

We will not authenticate users in the background based on a natural conversation. Users will have to speak predetermined phrases in order for our solution to work. This is a result of the limitations of the available voice authentication APIs.

2. Conversational manual voice authentication

We will not specify how a call center agent should authenticate a person in the case where automated authentication fails. This is up to the particular business to determine if they would like to use security questions, pins, password, etc.