Design Document

Version 0
Group 16
COMPSCI 4ZP6
Dr. Mehdi Moradi
Jan 24, 2025

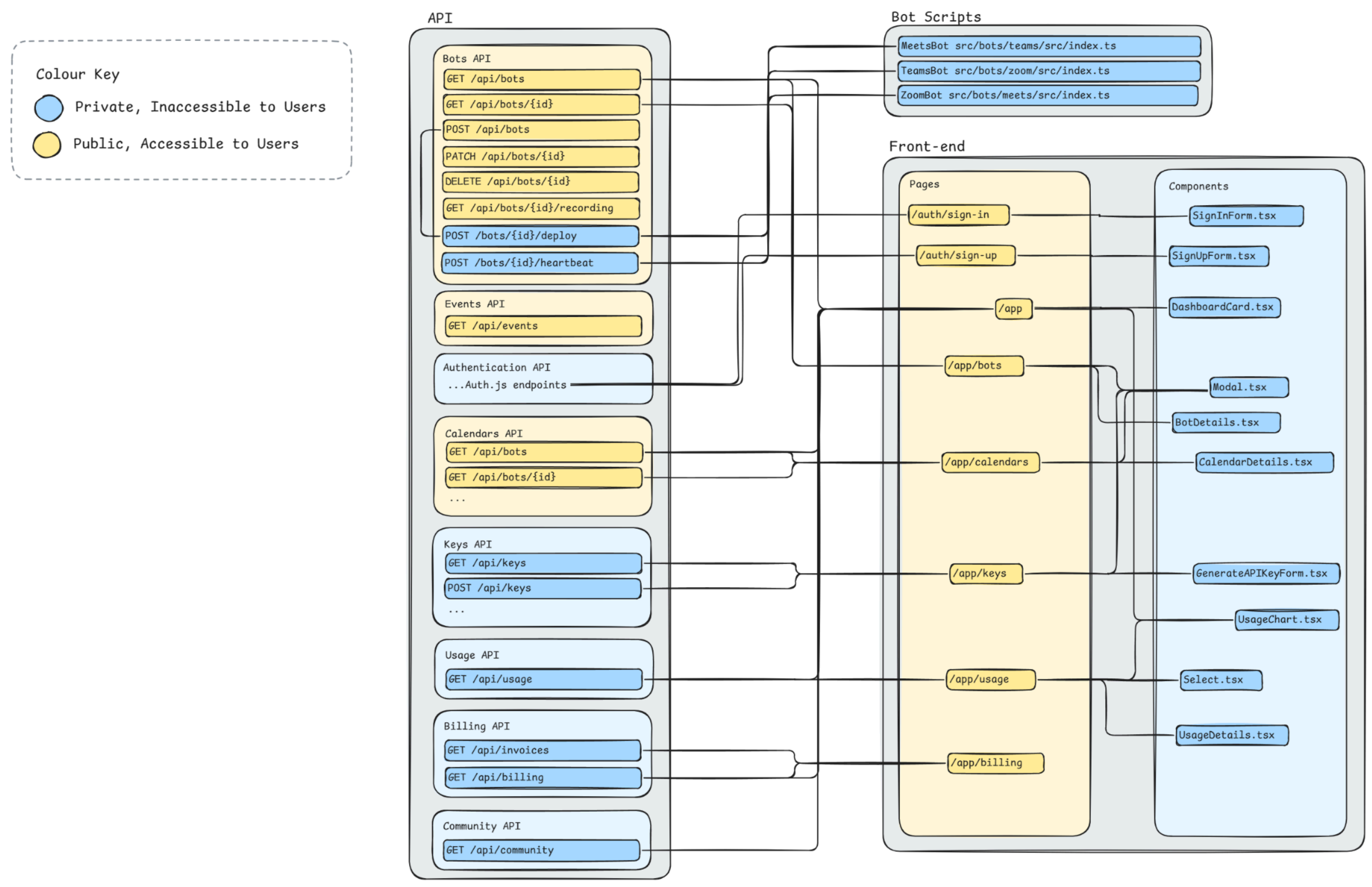
Alex Eckardt	eckardta@mcmaster.ca	400390784
Owen Gretzinger	gretzino@mcmaster.ca	400407289
Jason Huang	huanj168@mcmaster.ca	400374849
Sahib Khokhar	khokhs5@mcmaster.ca	400396918
Sarah Simionescu	simiones@mcmaster.ca	400363648

Brief

The *MeetingBot* project aims to provide a service to companies and individuals who wish to build a bot for video meetings, but don't have the resources to have a dedicated team build and maintain integrations for all meeting platforms. This document aims to document the core systems of this service and the interfaces between them. Our service is ***configured*** through **a front-end dashboard** and ***utilized*** by integrating with our **API**.

Component Diagram

View on Excalidraw in Full Detail!



To satisfy the page length constraints, some details for the private/helper components (in blue) have been redacted. We will focus mainly on the public components (in yellow) and their relationships.

Component Behaviour

The following is a list of undesired behaviours that apply to each of the following API endpoints. Components with additional undesired behaviours will be outlined in their own section.

Undesired Behaviours	Incorrect Input to the API (where applicable) Internal component breakdown of communication. Input is wrong (ID does not exist in the database, type is incorrect)

Bots API

(getBots) /api/bots GET	
Normal Behaviour	Returns a list of every active bot along with their statuses.

API	**Output**	Returns a list of all bots.
Implementation		Interfaces with Postgres Database on AWS to select all bots from the table.

(getBot) /api/bots/{id} GET		
Normal Behaviour		Gets a specific bot by its ID.
API	**Input**	`"id": 2147483647`
	Output	See **Example Bot Output** In the appendix.
Implementation		Interfaces with Postgres Database on AWS to select a bot with a matching id from the table.
Additional U.B		Id does not match one belonging to a bot

(createBot) /api/bots POST		
Normal Behaviour		Creates a new bot with the specified configuration.
API	**Input**	See **Example Bot Input** in the appendix.
	Output	See **Example Bot Output** in the appendix.
Implementation		Interface with Postgres DB and input values as a new row.

(updateBot) /api/bots{id} PATCH		
Normal Behaviour		Updates an existing bot's configuration.

API	**Input**	`"id": 2147483647`
		`{`
		`  "data": [Object]`
		`}`
		
		Where [Object] is an instance like **Example Bot Input**

	Output	See **Example Bot Output** in the appendix.
Implementation		Interface with Postgres DB and input values as a new row.
Additional U. B.		User could attempt to malform the patch data, which could conflict with DB columns

(deleteBot) /api/bots/{id} DELETE		
Normal Behaviour		Deletes a bot by its ID. This is not fully intended to be used, but rather to complete the CRUD operations.
API	**Input**	`"id": 2147483647`

	Output	`{`
		`  "message": "string"`
		`}`

		A message from the DB.
Implementation		Interface with Postgres DB and removes the row.
Additional U. B.		Users could attempt to delete something that does not exist.

(getRecording) /api/bots/{id}/recording GET		
Normal Behaviour		Retrieves the recording associated with a specific bot.
API	**Input**	`"id": 2147483647`
	Output	`{ "recording": "string" }`
Implementation		Interfacing with Postgres DB and returns the recording link (S3 bucket).
Additional U. B.		Could have issues if trying to get the recording before uploading is done. If the recording gets moved, then the return link would be faulty.

[PRIVATE] (deployBot) /bots/{id}/deploy POST		
Normal Behaviour		Deploys and begins the bot's life cycle
API	**Input**	`"id": 2147483647` An additional **config** file, see appendix.
	Output	See **Example Bot Output** in the appendix.
Implementation		Interfaces with PostgresDB to get access to the configuration data, then instructs EC2 to spin up a docker container with the image of the requested bot.
Additional U. B.		Id provided is of a bot that is/was already active Provision cloud resources fails (lack of permissions, connectivity etc.) The bot failing on startup Bot does not start sending heartbeats so state is not updated

Events API

(getEventsForBot) /api/events/bot/{botId} GET		
Normal Behaviour		Retrieves a list of all events associated with a specific bot
API	**Output**	A list of objects, see the **Example Event Output** in the appendix.
Implementation		Query with PostgresDB and return it's result
Additional Undesired Behaviours		This would be the place to think what can go wrong. Think simultaneously about testing and verification here. Although the details on this topic will go int V&V

Bot Component

(Teams/Zoom/Meets) Bot - src/bots/{type}/src/index.ts		
Normal Behaviour		A single bot instance will join either a Teams/Google/Meet online meeting, record it, and upload the recording to the backend.
API	**Input**	An **.env** file requiring the following items to be set up by the user ahead of time. These are user secrets (provided by AWS) which are necessary for the bot to function as intended: `AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, AWS_BUCKET_NAME, AWS_REGION` A **config.json** file, which is customized for each meeting. The format of this file can be found in the appendix.
	Output	The bot's id will be passed back to the backend as a query parameter, so it knows which bot just finished. The bot will also create side effects, such as uploading meeting recordings to S3, which the backend is aware of.
Implementation		`MeetingBot`, the interface for a meeting bot. The following are the required methods and their purpose: `sendHeartbeat`, sends a ping back to the backend. `joinMeeting`, the logic for joining a meeting `leaveMeeting`, logic for leaving a meeting. `startRecording`, starts up the recording process `stopRecording`, ends the recording process `meetingActions`, what should happen when the bot joins the meeting (Start Recording and wait until the meeting is over). Then, leave the meeting. `MeetsBot, TeamsBot, ZoomBot`, the implementations of the above for each of the platforms. Each instance of a bot will run on its own `main` thread, which instantiates a bot class and defines the other logic: `getS3Client`, makes a connection to the desired S3 bucket. `uploadRecording`, uploads the recording to S3. `heartbeatLoop`, asynchronously loops, calling the bot's ping call. `sendHeartbeat()` calls the related private endpoint ***/api/bots/{id}/heartbeat*** to update the backend.
Potential Undesired Behaviours		Could not connect to S3 Client (bad **.env** parameters). Could not connect to S3 Client (bad connection). Recording Upload to S3 Fails. The Heartbeat Loop does not end, lasting forever. Bot is unable to join the meeting for various reasons. Bot is unable to leave the meeting for various reasons. Meeting ends early, but the Bot is not aware and records dead time.

	A Popup appears, eating all scripted inputs. Bot is labeled as "suspicious" by Google and cannot join the Meet A meeting platform changes their UI, which interrupts the predefined traversal sequence logic.

Front-end Pages
For brevity, I will specify here **potential undesired behaviours** that apply to all of the front-end pages.

Undesired Behaviours	The components could look incorrect once the page window is resized. If there are any internal or external links on the page, these links may not work once clicked. Additionally, if the page makes API requests, it is possible that the state of the rendered components may not accurately reflect the current state of the API request, or the API request might be improperly formatted.

Authentication ***/auth/sign-in /auth/sign-up***	
Normal Behaviour	These pages will allow a user to create or sign-in to an account using a form.
Implementation	**Subcomponents:** `SignInForm.tsx`, a form for users to sign-in and `SignUpForm.tsx`, a form for users to sign-up **Related Endpoints:** We will use the Auth.js auth endpoints to authenticate or create the user

Main Dashboard ***/app***	
Normal Behaviour	This will be the first page shown upon authentication. If the user *has not yet* created an API key, there will be cards to guide the user to getting started (link to the API page, link to the docs). Otherwise, for returning users, cards will showcase important concise details (e.g. current balance, current number of active bots) and relevant links for the users to navigate through the platform.
Implementation	**Subcomponents:** `DashboardCard.tsx` will render a dashboard card in a consistent style given a header, icon, body and link (external or internal) passed through props. `UsageChart.tsx` will render the respective time-graph for the current week within one of the cards. **Related Endpoints:** GET `/api/bots/` to get the count of active bots for the current user, GET `/api/calendars/` to get the count of calendars for the current user, GET `/api/usage` will provide usage details (aggregated by the given time period provided in the body) for the current user, GET `/api/community` to list the discord and GitHub updates from the community, GET `/api/billing` to provide an object of important billing details

API Keys ***/app/keys***	
Normal Behaviour	This page will list the API keys and generate new API keys.
Implementation	**Subcomponents:** `GenerateAPIKeyForm.tsx` to create a new API key in a `Modal.tsx` component. **Related Endpoints:** POST `/api/keys` to generate a new key for the current user, GET `/api/keys` to list all the api keys for the current user

Bots */app/bots*	
Normal Behaviour	This page will list all the bots and allow the user to view properties about any particular bot.
Implementation	**Subcomponents:** `BotDetails.tsx` will render all details for a given bot ID (passed through props) within a `Modal.tsx` component **Related Endpoints:** GET `/api/bots/` to list the bots for the current user, GET `/api/bots/{id}` to get details about a specific bot

Calendars */app/calendars*	
Normal Behaviour	This page will list all the calendars and allow the user to view properties about any particular calendar.
Implementation	**Subcomponents:** `CalendarDetails.tsx` will render all details for a given calendar ID (passed through props) within a `Modal.tsx` component **Related Endpoints:** GET `/api/calendars` to list the calendars for the current user, GET `/api/calendars/{id}` to get details about a specific calendar

Usage */usage*	
Normal Behaviour	This page will display the user's usage in the last year, month or week (by their selection) on a line graph.
Implementation	**Subcomponents:** `Select.tsx` will render the select component for "This Year," "This Month," and "This Week", and `UsageChart.tsx` will render the respective time-graph for the current selection passed through props. **Related Endpoints:** GET `/api/usage` will provide usage details (aggregated by the given time period provided in the body) for the current user

Billing */billing*	
Normal Behaviour	This page displays the user's current balance, payment method and reload status. It links them to external links where they may add more funds, manage their payment method and manage their reload options. The user may also view their invoice history and recent usage.
Implementation	**Subcomponents:** `DashboardCard.tsx` will render each card in a consistent style given a header, icon, body and link (external or internal) passed through props. `UsageChart.tsx` will render the respective time-graph for the current week within one of the cards. **Related Endpoints:** GET `/api/invoices` to list of invoices for the current user, and GET `/api/billing` to provide an object of important billing details

Relationship Between Components and Requirements (P0)

A bot should be able to join a meeting.	***P0***
Related Components	(createBot) /api/bots POST, (deployBot) /bots/{id}/deploy POST, (Teams/Google/Meets) Bot, API Keys /app/keys

Implementation	createBot: Called by the client to create and either immediately launch the bot, or schedule it to launch later. deployBot: Deploys the previously created bot to handle the meeting (Teams/Google/Meets) Bot: The bot that runs to attend the meeting API Keys: used by the user to call createBot

The system should be able to detect when the call ends, and stop the bot.	***P0***
Related Components	(Teams/Google/Meets) Bot
Implementation	(Teams/Google/Meets) Bot: The bot joins the meeting and detects when all participants have left the call, and stops itself using `leaveMeeting` and `stopRecording`.

The system should be able to capture the audio stream from the meeting.	***P0***
Related Components	(Teams/Google/Meets) Bot, (getRecording) /api/bots/{id}/recording GET
Implementation	(Teams/Google/Meets) Bot: When the bot joins the meeting, we start recording with `startRecording`. At the end of the meeting, we upload it with `uploadRecording`. getRecording: We can retrieve the captured audio stream from the meeting using this endpoint. getRecording: This is used by the client to retrieve the audio recording when the meeting is complete.

User Interface

The designs are available in the Appendix.

Style

We honoured the style of our chosen component library, ShadCN, through leveraging the community figma component library. We set a clear colour scheme: **primary colour (slate/900), secondary colour (slate/500) and neutral colours (white/black)** to create a high-contrast display and establish a clear hierarchy among our components. Additionally, we leveraged a single font (Inter) with a few select styles: **h2 (semi-bold, size 30, black)**, p (regular, size 16, black) and **subtle (semibold, size 14, secondary)** (plus small variations) to establish a clear hierarchy of text within and amongst components. Lastly, we used the icon library ShadCN ships with, Lucide, for consistency.

User Experience

We utilized our colours and fonts to establish hierarchies to guide users to complete their goals. For example, the main dashboard for a user who has not yet created an API key highlights the "Create API Key" button in the primary colour and bold font at the top left to draw the user's attention. Whereas for returning users, the dashboard displays key metrics which guide them to the respective pages where they can learn more and achieve their goals.

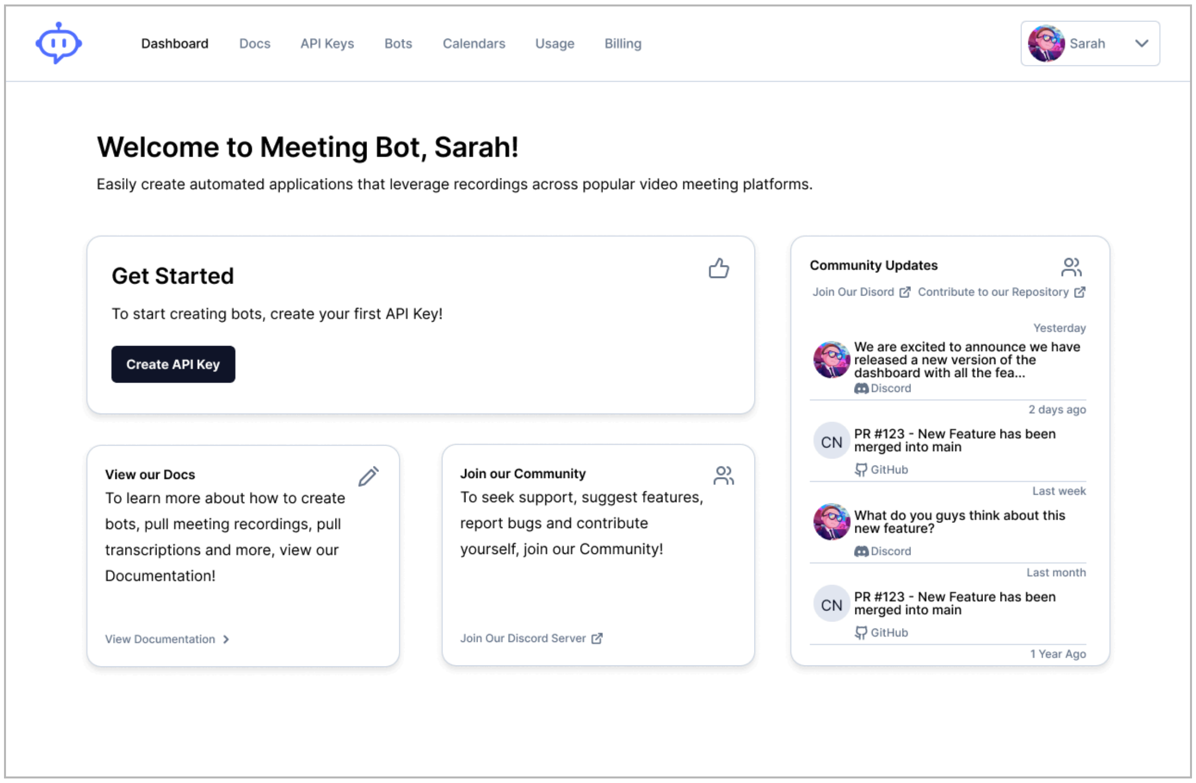
Additionally, we established memorable and significant user experience patterns. For example, all navigation links use the **subtle** look and are often placed at the bottom of the card. If the link is internal, the symbol is a right arrow; otherwise, it's the external link symbol. Another example is all "View" buttons (identical styling) on the API Keys, Bots, and Calendar pages open a modal with more information about the row item.

Appendix

UI Design

Main Dashboard

Before the user has generated their first API key



Welcome to Meeting Bot, Sarah!

Easily create automated applications that leverage recordings across popular video meeting platforms.

Get Started

To start creating bots, create your first API Key!

Create API Key

View our Docs

To learn more about how to create bots, pull meeting recordings, pull transcriptions and more, view our Documentation!

View Documentation

Join our Community

To seek support, suggest features, report bugs and contribute yourself, join our Community!

Join Our Discord Server

Community Updates

Join Our Disord Contribute to our Repository

Yesterday

We are excited to announce we have released a new version of the dashboard with all the fea...

Discord

2 days ago

CN PR #123 - New Feature has been merged into main

GitHub

Last week

What do you guys think about this new feature?

Discord

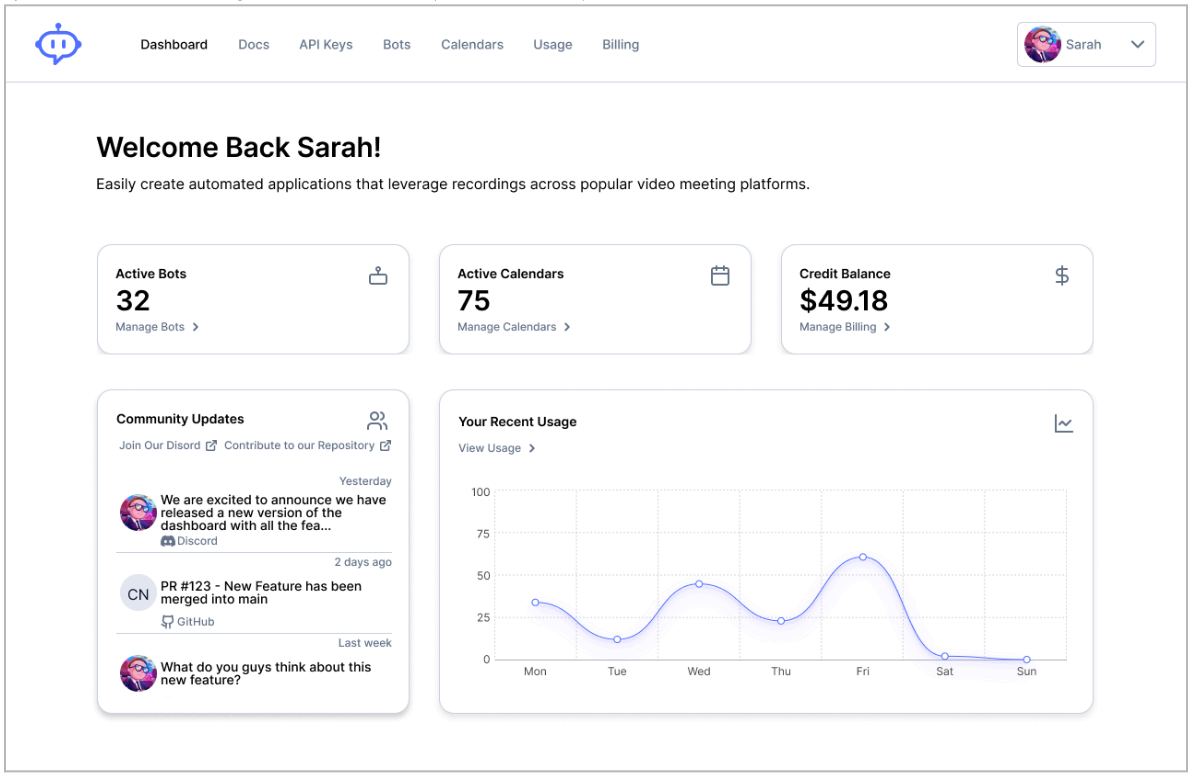
Last month

CN PR #123 - New Feature has been merged into main

GitHub

1 Year Ago

After the user has generated their first API key



Welcome Back Sarah!

Easily create automated applications that leverage recordings across popular video meeting platforms.

Active Bots

32

Manage Bots

Active Calendars

75

Manage Calendars

Credit Balance

$49.18

Manage Billing

Community Updates

Join Our Disord Contribute to our Repository

Yesterday

We are excited to announce we have released a new version of the dashboard with all the fea...

Discord

2 days ago

CN PR #123 - New Feature has been merged into main

GitHub

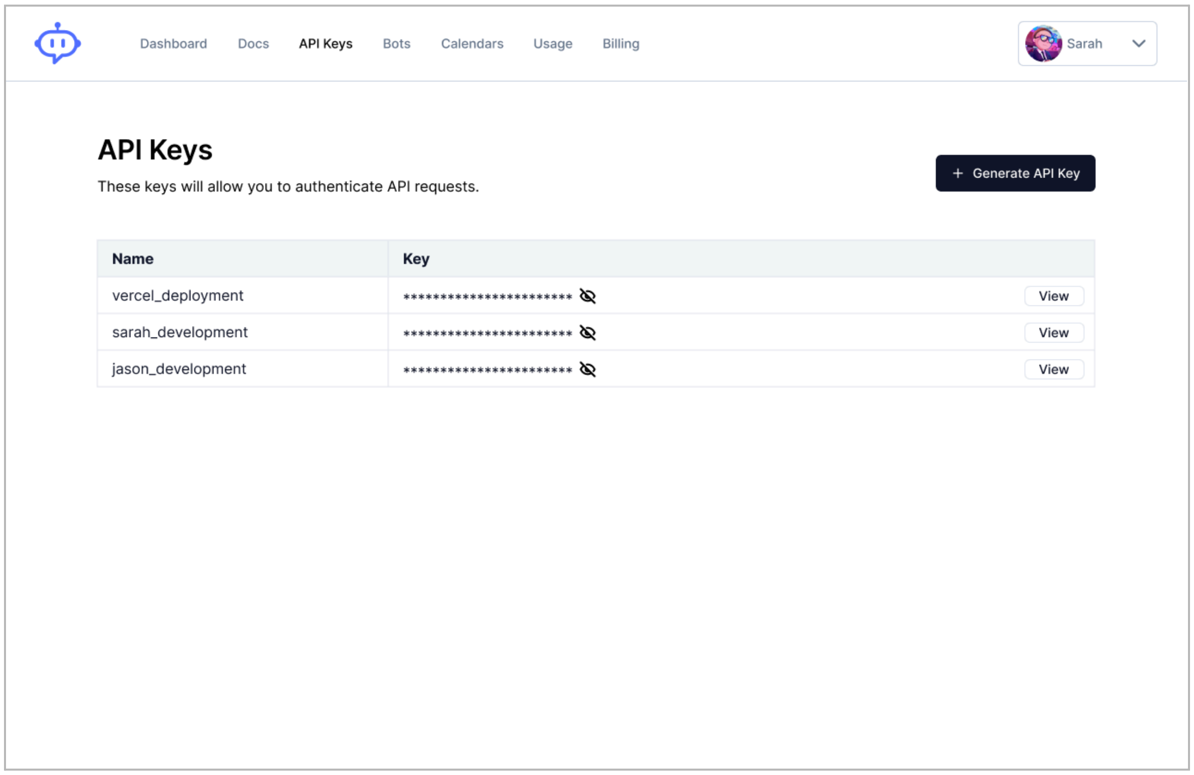
Last week

What do you guys think about this new feature?

Your Recent Usage

View Usage

API Keys

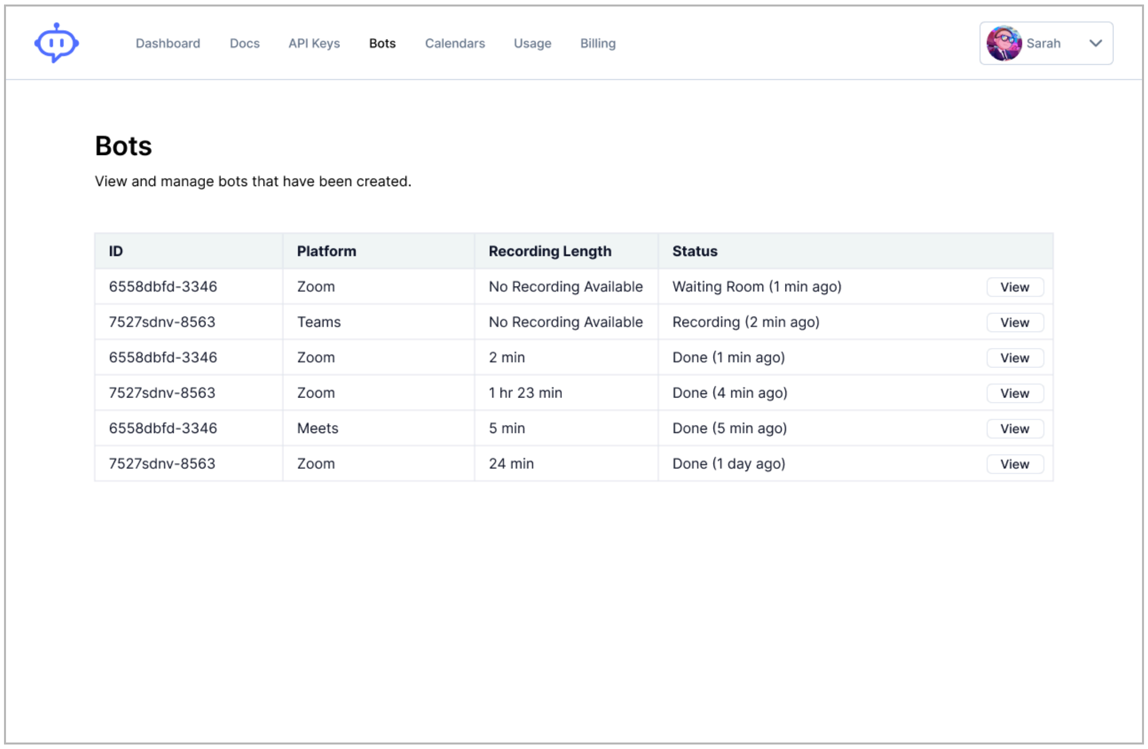


API Keys

These keys will allow you to authenticate API requests.

Name	Key	
vercel_deployment	***********************	View
sarah_development	***********************	View
jason_development	***********************	View

Bots

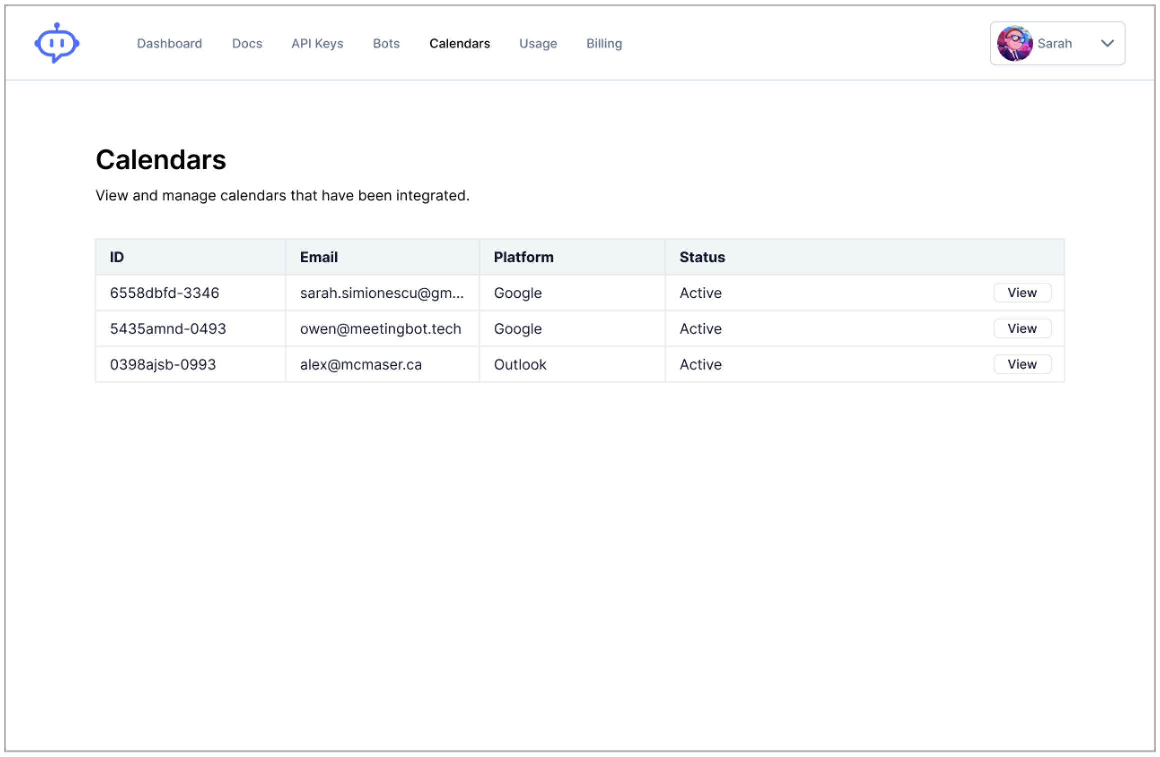


Bots

View and manage bots that have been created.

ID	Platform	Recording Length	Status	
6558dbfd-3346	Zoom	No Recording Available	Waiting Room (1 min ago)	View
7527sdnv-8563	Teams	No Recording Available	Recording (2 min ago)	View
6558dbfd-3346	Zoom	2 min	Done (1 min ago)	View
7527sdnv-8563	Zoom	1 hr 23 min	Done (4 min ago)	View
6558dbfd-3346	Meets	5 min	Done (5 min ago)	View
7527sdnv-8563	Zoom	24 min	Done (1 day ago)	View

Calendars

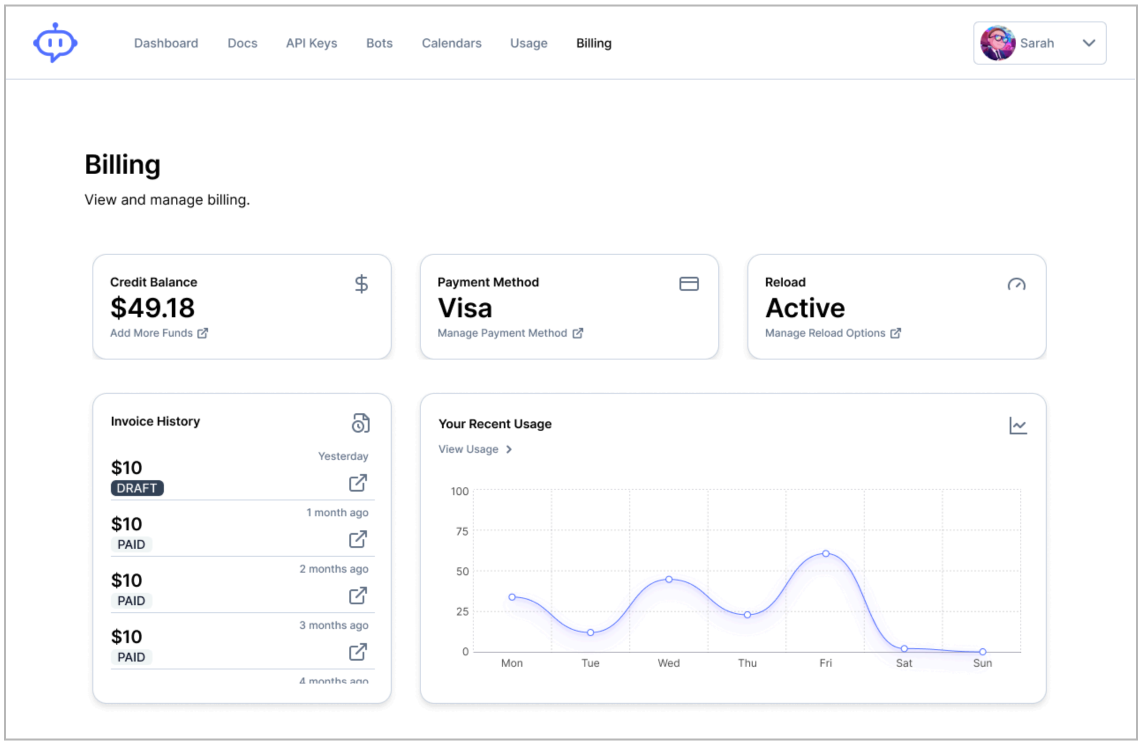


Calendars

View and manage calendars that have been integrated.

ID	Email	Platform	Status	
6558dbfd-3346	sarah.simionescu@gm...	Google	Active	View
5435amnd-0493	owen@meetingbot.tech	Google	Active	View
0398ajsb-0993	alex@mcmaser.ca	Outlook	Active	View

Billing



Billing

View and manage billing.

Credit Balance
$49.18
Add More Funds

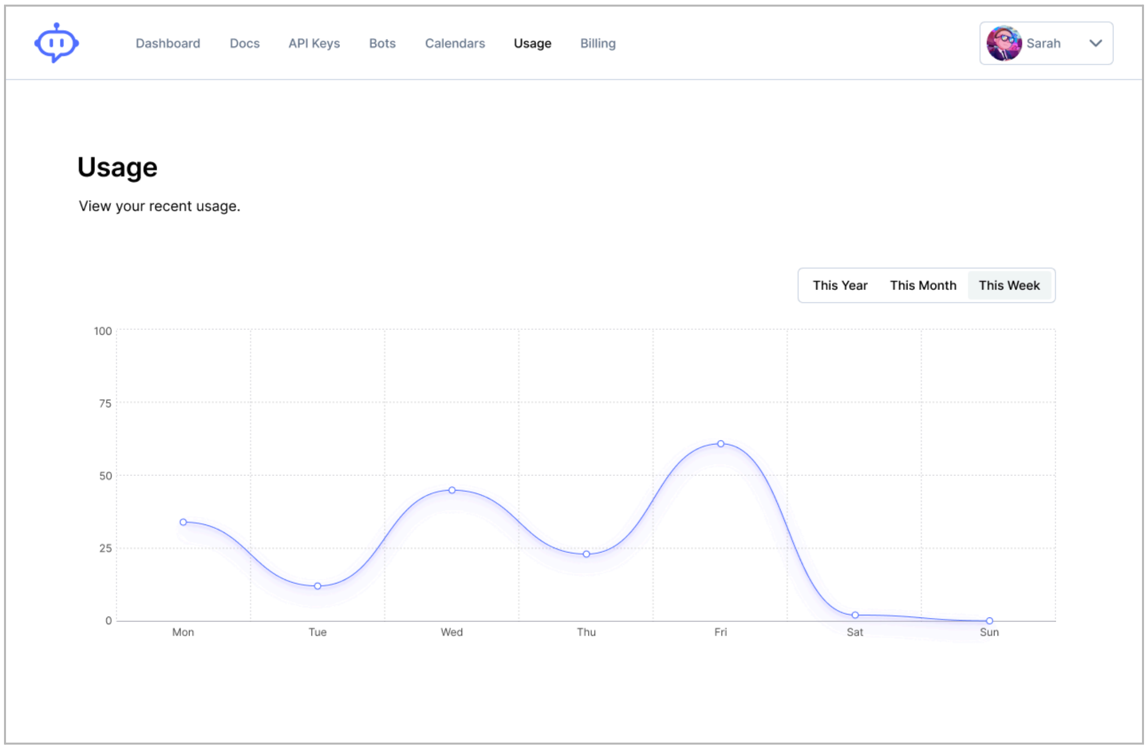
Payment Method
Visa
Manage Payment Method

Reload
Active
Manage Reload Options

Invoice History

$10 DRAFT	Yesterday
$10 PAID	1 month ago
$10 PAID	2 months ago
$10 PAID	3 months ago
	4 months ago

Usage



Usage

View your recent usage.

Bot Config.json Schema

```
{
  meeting_info: {
    platform: [“Meet”|”Zoom”|”Teams”] which platform the bot should join
    meeting_id: [String] The Meeting code, used in different ways depending on the platform.
    meeting_password*: [String] The password used to enter the meeting
    organizer_id*: [String] The meeting organization's identifier (Zoom)
    tenant_id*: [String] — : The meeting tenant’s identifier (Zoom)
    message_id*: [String] The message associated with the meeting (Zoom)
    thread_id*: [String] The discussion associated with the meeting (Zoom)
  }
  user_id: [Any] The user id of the owner of the bot.
  meeting_name: [String] The Name of the Meeting
  start_time: [ISO-DateStr] Date when the bot will schedule to join the meeting
  end_time: [ISO-DateStr] Date when the bot will “expect” the meeting to end
  bot_display_name: [String] Display name of the bot
  bot_image: [String] URL to image bot will use as its avatar
  heartbeat_frequency: [Number] the frequency (in ms) a bot will send a status update to the
   backend.
  callback_url: [String] URL to callback to the backend
  automatic_leave: [Object] Config object which defines how the bot should leave the meeting
}

* - If applicable
```

Example Bot Creation Input

```
{
  "userId": 2147483647,
  "meetingTitle": "string",
  "meetingInfo": {
    "meetingId": "string",
    "meetingPassword": "string",
    "meetingUrl": "string",
    "organizerId": "string",
    "tenantId": "string",
    "messageId": "string",
    "threadId": "string",
    "platform": "zoom"
  },
  "startTime": "2025-01-23T17:09:36.383Z",
  "endTime": "2025-01-23T17:09:36.383Z",
  "lastHeartbeat": "2025-01-23T17:09:36.383Z",
  "createdAt": "2025-01-23T17:09:36.383Z",
  "deploymentStatus": "PENDING",
  "deploymentError": "string"
}
```

Example Bot Output

```
{
  "id": 2147483647,
  "userId": 2147483647,
  "meetingTitle": "string",
  "meetingInfo": {
    "meetingId": "string",
    "meetingPassword": "string",
    "meetingUrl": "string",
    "organizerId": "string",
    "tenantId": "string",
    "messageId": "string",
    "threadId": "string",
    "platform": "zoom"
  },
  "startTime": "2025-01-23T17:09:36.383Z",
  "endTime": "2025-01-23T17:09:36.383Z",
  "lastHeartbeat": "2025-01-23T17:09:36.383Z",
  "createdAt": "2025-01-23T17:09:36.383Z",
  "deploymentStatus": "PENDING",
  "deploymentError": "string"
}
```

Example Event Output

```
 {
     "id": 2147483647,
     "botId": 2147483647,
     "eventType": "PARTICIPANT_JOIN",
     "eventTime": "2025-01-23T17:50:53.103Z",
     "data": {
       "participantId": "string"
     },
     "createdAt": "2025-01-23T17:50:53.103Z"
   }
```