Microsoft Office Teams Apps Consulting Practice

Cognitive Convergence is Subject Matter Expert in Office 365, Dynamics 365, SharePoint, Project Server, Power Platform: Power Apps-Power BI-Power Automate-Power Virtual Agents. Our Microsoft Office Teams Consulting, add-in Development, Customization, Integration services and solutions, can help companies maximize business performance, overcoming market challenges, achieving profitability and providing best customer service.

Contents

OBJECTIVE

This case study is a brief introduction about the core concept of Microsoft Teams' App, its fundamentals, building architecture & its development tracks.

INTRODUCTION

App for Microsoft Teams are a great way to interact with the data, and there are a variety of integration points to choose from.

- Messaging extensions with search commands - Search external systems and share the results as an interactive card.
- Messaging extensions with action commands - Collect information to insert into a data store or perform advanced searches.
- Tabs - Create embedded web experiences to view, work with, and share data.
- Connectors and webhooks - A simple way to push data into, and send data out of the Teams client.
- Task modules - Interactive modal forms from wherever you need them to collect or display information.

INITIATE WORKFLOWS AND PROCESSES

Sometimes user just need a quick way to start a process or workflow in an external system.

- Messaging extensions action commands - Trigger from messages, allowing your users to quickly send the contents of a message to your web services.
- Task modules - Open them from a tab, a bot, or a messaging extension to collect information before initiating a workflow.
- Conversational bots - Interact with your users through text and rich cards.
- Outgoing webhooks - A good choice for a simple back-and-forth interaction when you don't need to build an entire conversational bot.

SEND NOTIFICATIONS AND ALERTS

Send asynchronous notifications and alerts to your users in Teams. Use interactive cards to provide quick access to commonly used actions and links to additional information.

- Conversational bots - Send proactive messages to groups, channels, or individual users.
- Connectors & incoming webhooks - Allow a channel to subscribe to receive messages. With a connector let users tailor the subscription with a configuration page.

ASK QUESTIONS AND GET ANSWERS

People have questions. Users have probably got a lot of the answers stored away somewhere. Unfortunately, its often quite difficult to connect the two together.

1. Conversational bots - Natural language processing, AI, machine learning, all the buzzwords. Use a bot powered by the intelligent cloud to connect your users to the answers they need.
2. Tabs - Embed your existing web portal in Teams, or create a Teams-specific version for added functionality

ANYTHING USERS CAN DO IN A SINGLE PAGE APP (SPA)

Tabs are embedded web pages. Pretty much anything that users can do in a SPA, users can do in a tab in Teams. Just be sure to pay attention to scope - group and channel tabs are for shared experiences, personal tabs are for ... personal experiences. The team's list of stuff goes on the channel tab, the list of the stuff goes in the personal tab.

TEAMS, CHANNELS AND GROUP CHATS

Teams, channels and group chats allow multiple people to collaborate. Apps in this context make themselves available to all members of the group or conversation, typically focusing on enabling additional collaborative workflows or unlocking new social interactions. Your app will have access to APIs allowing it to get information about the members in the conversation, the channels in a team, and metadata about the team or conversation.

They can be extended with:

- **Conversational bots** interacting with members of the conversation through chat, and responding to events (like a new member being added, or a channel being renamed). All conversations with a bot in this context are visible to all members of the channel or group, so you'll need to ensure the conversation is relevant to everyone.
- **Configurable Tabs** providing a full-screen embedded web experience configured for the channel or group chat it is installed in. All members will interact on the same shared web-app, so a stateless single page app experience is typical.
- **Webhooks and Connectors** enabling external services to post messages to the conversation, and your users to send messages to your service. You can take advantage of cards and card actions to to create rich, actionable messages.

BUILD APPS WITH THE MICROSOFT TEAMS TOOLKIT AND VISUAL STUDIO CODE

The Microsoft Teams Toolkit enables the user to create custom Teams apps directly within the Visual Studio Code environment. The toolkit guides you through the process and provides everything you need to build, debug, and launch your Teams app.

Understand the use cases

The Microsoft Teams platform offers a large variety of extensibility points and UI elements that the app can take advantage of. Each method of interacting with the users has it's own strengths and weaknesses. Building an awesome Teams app is all about finding the right combination to meet the user's needs. If the user wants to meet those needs, he should first need to understand them.

What problem are you trying to solve?

Every good app has a core problem (or need) it is trying to solve - before the user start building, the user needs to articulate what that problem is. At its heart, Teams is a collaboration platform, so apps looking to solve collaboration problems are a great fit. It's also a social platform, is natively cross-platform, sits at the heart of Office 365, and offers a personal canvas for users to create apps on. There is an incredibly wide variety of needs that can be solved with a Teams app, just be sure you understand which one you're trying to solve.

Who are you solving it for?

Sometimes this can be obvious - "My team's monitoring system needs to send alerts somewhere, we need to be able to discuss them quickly, and none of us want to check our email." Sometimes the target audience can grow over time - "Our sister team is jealous of our alerting system, and now they want in on the action." Understanding who the users are will help you identify the right distribution model, but more importantly, will help to identify how they use Teams.

- Are they primarily front-line workers on mobile clients?
- Do you expect a lot of guest users to need access to your app?
- Do they use teams and channels, or primarily group chats?
- How technically sophisticated are they?
- Will you need a thorough onboarding experience, or will a few pointers do?

Do you need authentication?

user should identify early on if they going to need to protect the services you're exposing, and at what level. Remember the web services you'll be exposing in your Teams app are publicly available over the internet, so if the requirement is to secure them start thinking about how now.

Should the entire app be in Teams?

Whether the user starts building something entirely new or bringing an existing solution into Teams, it is important to decide if the entire app is going to be inside the Teams client, or if it makes sense to only bring in a portion of the experience. With a combination of tabs, messaging extensions, task modules, interactive cards, and conversational bots that can be build complex apps completely inside of Teams. However, that doesn't always make sense. Remember who your users are, and the problem you're trying to solve. Do they already have a system for solving most of the problems, and the user just needs to extend a sub-set of the functionality into Teams? Typically if the user is only going to bring in a portion of the solution you should focus on sharing, collaboration, and initiating and monitoring workflows.

What will the onboarding experience be like?

Users’ onboarding experience can be the difference between success or failure for the app. For each capability of the developed app, and each context, that capability can be installed in, the user should have a plan for how the users are going to introduce yourself. How the user will introduce the conversational bot when it is installed in a channel with a thousand people will probably be different than when it is installed in a one-to-one chat. What happens when a user first configures your tab in a channel? If you're sharing cards with a messaging extension, does it make sense to add a small link to a "learn more" page to help introduce users to what else your app can do?

Knowing who the users are will help you craft the right experience. Do you expect most people to already have some context of what the app is for, or to have already used the previously developed services in another context? Or are they coming to the app with no prior knowledge? Craft the user’s onboarding experience with the key users in mind. Remember too that users can discover the newly developed app in a variety of ways - they might be the ones installing it, or they might be introduced to the app when another team member uses it to share content.

Above all else, remember that nobody likes spam. Blasting away with personal and channel messages is a good way to get un-installed quickly!

Create, share and collaborate on items in an external system

App for Microsoft Teams is a great way to interact with the organization’s data, and there are a variety of integration points to choose from.

- Messaging extensions with search commands - Search external systems and share the results as an interactive card.
- Messaging extensions with action commands - Collect information to insert into a data store or perform advanced searches.
- Tabs - Create embedded web experiences to view, work with, and share data.
- Connectors and webhooks - A simple way to push data into, and send data out of the Teams client.
- Task modules - Interactive modal forms from wherever you need them to collect or display information.

Initiate workflows and processes

Sometimes the requirement is to find a quick way to start a process or workflow in an external system.

1. Messaging extensions action commands - Trigger from messages, allowing your users to quickly send the contents of a message to your web services.
2. Task modules - Open them from a tab, a bot, or a messaging extension to collect information before initiating a workflow.
3. Conversational bots - Interact with your users through text and rich cards.
4. Outgoing webhooks - A good choice for a simple back-and-forth interaction when you don't need to build an entire conversational bot.

Send notifications and alerts

Send asynchronous notifications and alerts to the users in Teams. Use interactive cards to provide quick access to commonly used actions and links to additional information.

- Conversational bots - Send proactive messages to groups, channels, or individual users.
- Connectors & incoming webhooks - Allow a channel to subscribe to receive messages. With a connector let users tailor the subscription with a configuration page.

Ask questions and get answers

People have questions. Users may get a lot of the answers stored away somewhere. Unfortunately, its often quite difficult to connect the two, together.

1. Conversational bots - Natural language processing, AI, machine learning, all the buzzwords. Use a bot powered by the intelligent cloud to connect your users to the answers they need.
2. Tabs - Embed your existing web portal in Teams, or create a Teams-specific version for added functionality.

Start small

Not sure where to start? Feeling a bit overwhelmed with the awesome variety of options available to you? Don't fret, choose a core feature of your app, and start there. Once the user gets a feel for the flow of information through the various contexts in Teams, it will be a lot simpler to picture a more complex interaction.

Putting it all together

That being said, the best apps usually combine multiple features, creating an app that engages users in the right context with the right functionality at the right time. Don't try to force functionality into a place it doesn't belong - just because the user has got a good one-to-one conversational bot doesn't mean he should just add it to a team. Different extensibility points are good for different things; play to their strengths and the app will shine.

Installing the Teams Toolkit

The Microsoft Teams Toolkit for Visual Studio Code is available for download from the Visual Studio Marketplace or directly as an extension within Visual Studio Code.

Set up a new Teams project

1. Create a workspace/folder for your project in your local environment.
2. In Visual Studio Code, select the Teams icon from the activity bar on the left side of the window.
3. Select Open the Microsoft Teams Toolkit from the command menu.
4. Select Create a new Teams app from the command menu.
5. When prompted, enter the name of the workspace. This will be used as both the name of the folder where the user's project will reside and the default name of the name app.
6. Press Enter and you will arrive at the Add capabilities screen configure the properties for the new app.
7. Select the Finish button to complete the configuration process.

Import an existing Teams app project

1. In Visual Studio Code, select the Teams icon from the activity bar on the left side of the window.
2. Select the Import app package from the command menu.
3. Choose your existing Teams app package zip file.
4. Choose the Select publishing package button. The configuration tab of the toolkit should now be populated with your app's details.
5. In Visual Studio Code, select File -> Add Folder to Workspace to add your source code directory to the Visual Studio Code workspace.

Configure your app

At its core, the Teams app embraces three components:

1. The Microsoft Teams client (web, desktop, or mobile) where users interact with your app.
2. A server that responds to requests for content that will be displayed in Teams, e.g., HTML tab content or a bot adaptive card.
3. A Teams app package consisting of three files:
    - The manifest.json
    - A color icon for your app to display in the public or organization app catalog
    - An outline icon for display on the Teams activity bar.

When an app is installed, the Teams client parses the manifest file to determine needed information like the name of your app and the URL where the services are located.

1. To configure your app, navigate to the **Microsoft Teams Toolkit** tab in Visual Studio Code.
2. Select the **Edit app package** to view the **App details** page.
3. Editing the fields in the App details page updates the contents of the manifest.json file that will ultimately ship as part of the app package.

Package your app

Modifying the app details page or updating the manifest, or .env files in the app's .publish folder will automatically generate your Development.zip file. Users will need to include two icons in that same folder.

Install and run the app locally

Refer to the *Build and Run* content on the project homepage for detailed instructions for packaging and testing the app. In general, the users need to install the app's server, get it running, then set up a tunneling solution so that Teams can access content running from the localhost.

Run your app in Teams

- Prerequisites:
    - Enable Teams developer preview mode

1. Navigate to the activity bar on the left side of the Visual Studio Code window.
2. Select the **Run** icon to display the **Run and Debug** view.
3. You can also use the keyboard shortcut Ctrl+Shift+D.

Validate your app

The Validate page allows user to check your app package before submitting the app to AppSource. Simply upload the manifest package and the validation tool will check the app against all manifest related test cases. For each failed test, the description provides a documentation link to help to fix the error. For the tests that are hard to automate, the Preliminary checklist details 7 of the most common failed test cases as well as links to a complete submission checklist.

Publish the app to Teams

On the project home page, users can upload your app to a team, submit the app to the company custom app store for users in the organization, or submit the app to App Source for all Teams users. The company’s IT admin will review these submissions. Users can return to the Publish page to check on the submission status and learn if the app was approved or rejected by the IT admin.

WRITING MESSAGES

Your app can help users craft more effective messages by enabling them to search, or take action, in an external system, and insert the results in a rich, structured format complete with actionable buttons.

There are three ways your app can help users create better messages:

1. **Messaging Extension - search commands** allowing them to quickly search an external system, preview the results of that search, then insert the result into the chat as a rich card.
2. **Messaging Extension - link unfurling** allows your app to monitor web domains you're interested in. When a URL containing that domain is pasted into the compose message box, your app's API will be invoked, allowing you to add a rich card to the message with additional information about the item being linked to.
3. **Messaging Extension - action commands** present your user with a modal form (a task module), submit the results of the form to your app, then either insert a message into the conversation directly, or create part of a message the user can edit before sending to the conversation.

MICROSOFT TEAMS MODULES

Classes

ChildAppWindow
GlobalVars
ParentAppWindow

Interfaces

Context	
DeepLinkParameters	
File	File object that can be used to represent image or video or audio
FrameContext	
IAppWindow	
LocaleInfo	Represents OS locale info used for formatting date and time data
MessageRequest	
MessageResponse	
SdkError	
ShowNotificationParameters	

TabInformation	Represents information about tabs for an app
TabInstance	Represents information about a tab instance
TabInstanceParameters	Indicates information about the tab instance for filtering purposes.
TaskInfo	
TeamInformation	Represents Team Information

Enums

ChannelType	**The type of the channel with which the content is associated.**
ErrorCode	
FileFormat	Enum for file formats supported
FrameContexts	
HostClientType	
NotificationTypes	
TaskModuleDimension	
TeamType	Indicates the team type, currently used to distinguish between differe for Education (team types 1, 2, 3, and 4).
UserTeamRole	Indicates the various types of roles of a user in a team.

Functions

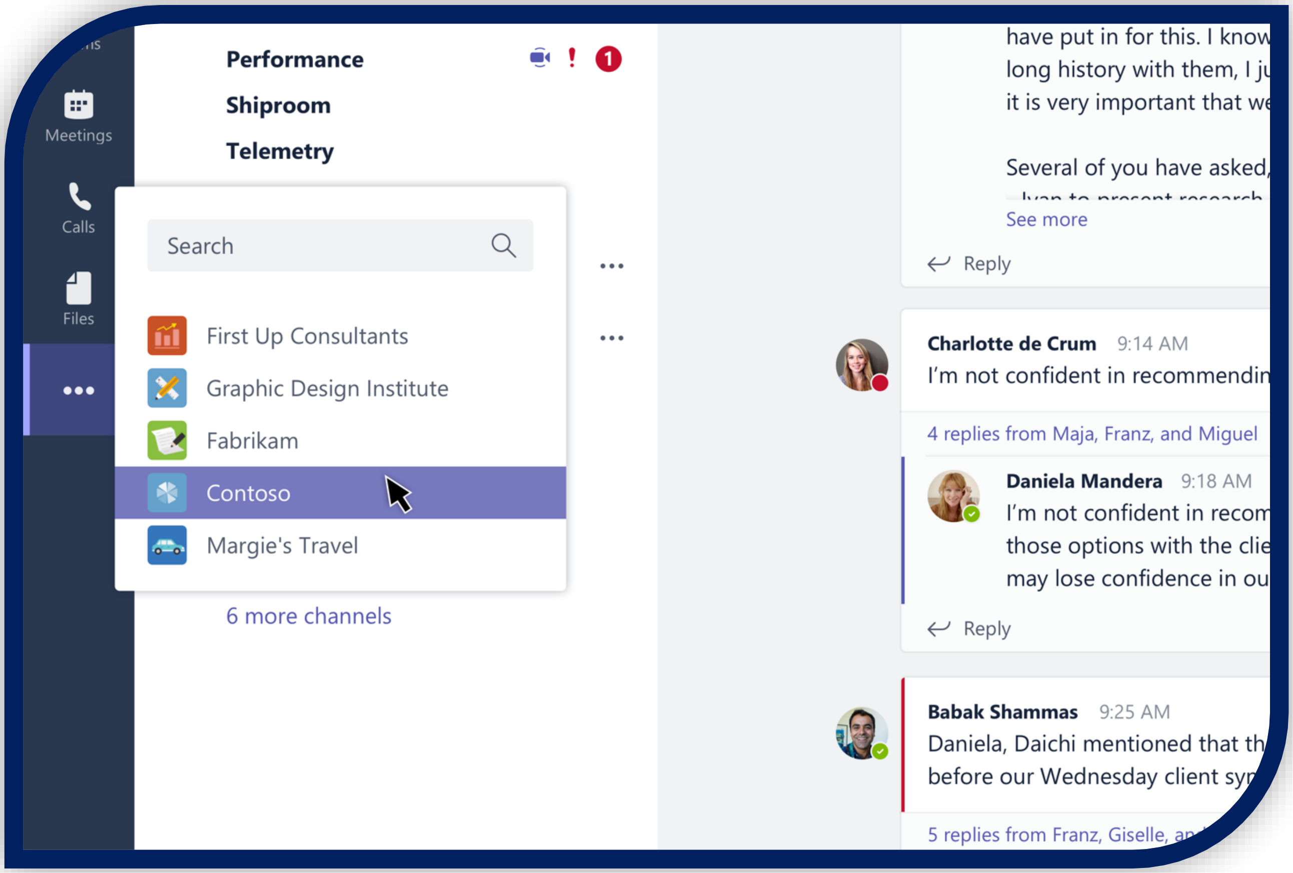
captureImage((error: SdkError, files: File[]) => void)	**Launch camera, capture image or choose image from gallery and return the images as a File[] object to the callback. Callback will be called with an error, if there are any. App should first check the error. If it is present the user can be updated with appropriate error message. If error is null or undefined, then files will have the required result. Note: Currently we support getting one File through this API, i.e. the file arrays size will be one. Note: For desktop, this API is not supported. Callback will be resolved with ErrorCode.NotSupported.**
compareSDKVersions(string, string)	Compares SDK versions.
enablePrintCapability()	Enable print capability to support printing page using Ctrl+P and cmd+P

ensureInitialized(string[])	
executeDeepLink(string, (status: boolean, reason?: string) => void)	execute deep link API.
generateRegExpFromUrls(string[])	
getContext((context: Context) => void)	Retrieves the current context the frame is running in.
getGenericOnCompleteHandler(string)	
getMruTabInstances((tabInfo: TabInformation) => void, TabInstanceParameters)	Allows an app to retrieve the most recently used tabs for this user.
getTabInstances((tabInfo: TabInformation) => void, TabInstanceParameters)	Allows an app to retrieve for this user tabs that are owned by this app. If no TabInstanceParameters are passed, the app defaults to favorite teams and favorite channels.
handleParentMessage(DOMMessageEvent)	
initialize(() => void, string[])	Initializes the library. This must be called before any other SDK calls but after the frame is loaded successfully.
initializeWithFrameContext(FrameContext, () => void, string[])	
isAPISupportedByPlatform(string)	Checks whether the platform has knowledge of this API by doing a comparison on API required version and platform supported version of the SDK
navigateBack((status: boolean, reason?: string) => void)	Navigates back in the Teams client. See registerBackButtonHandler for more information on when it's appropriate to use this method.
navigateCrossDomain(string, (status: boolean, reason?: string) => void)	Navigates the frame to a new cross-domain URL. The domain of this URL must match at least one of the valid domains specified in the validDomains block of the manifest; otherwise, an exception will be thrown. This function needs to be used only when navigating the frame to a URL in a different domain than the current one in a way that keeps the app informed of the change and allows the SDK to continue working.
navigateToTab(TabInstance, (status: boolean, reason?: string) => void)	Navigates the Microsoft Teams app to the specified tab instance.
print()	default print handler

processAdditionalValidOrigins(string[])	Processes the valid origins specifuied by the user, de-duplicates and converts them into a regexp which is used later for message source/origin validation
processMessage(DOMMessageEvent)	
registerBackButtonHandler(() => boolean)	Registers a handler for user presses of the Team client's back button. Experiences that maintain an internal navigation stack should use this handler to navigate the user back within their frame. If an app finds that after running its back button handler it cannot handle the event it should call the navigateBack method to ask the Teams client to handle it instead.
registerChangeSettingsHandler(() => void)	Registers a handler for when the user reconfigurated tab
registerFullScreenHandler((isFullScreen: boolean) => void)	Registers a handler for changes from or to full-screen view for a tab. Only one handler can be registered at a time. A subsequent registration replaces an existing registration.
registerOnThemeChangeHandler((theme: string) => void)	Registers a handler for theme changes. Only one handler can be registered at a time. A subsequent registration replaces an existing registration.
sendMessageEventToChild(string, any[])	Send a custom message object that can be sent to child window, instead of a response message to a child
sendMessageRequestToParent(string, any[])	Send a message to parent. Uses nativeInterface on mobile to communicate with parent context
setFrameContext(FrameContext)	
shareDeepLink(DeepLinkParameters)	Shares a deep link that a user can use to navigate back to a specific state in this page.

PERSONAL APPS

A personal app is a Teams application with a personal scope. As an app developer, you have the option to provide a version of your app that focuses on interactions with a single user. It can be a conversational bot to engage in one-to-one conversations with a user or a personal tab providing an embedded web experience. Personal apps enable users to view their select content in one place. In the following screenshot, Contoso is a personal app in the personal app flyout.



have put in for this. I know
long history with them, I ju
it is very important that we

Several of you have asked,
Ivan to present research

Charlotte de Crum 9:14 AM
I'm not confident in recommendin

Daniela Mandera 9:18 AM
I'm not confident in recom
those options with the clie
may lose confidence in ou

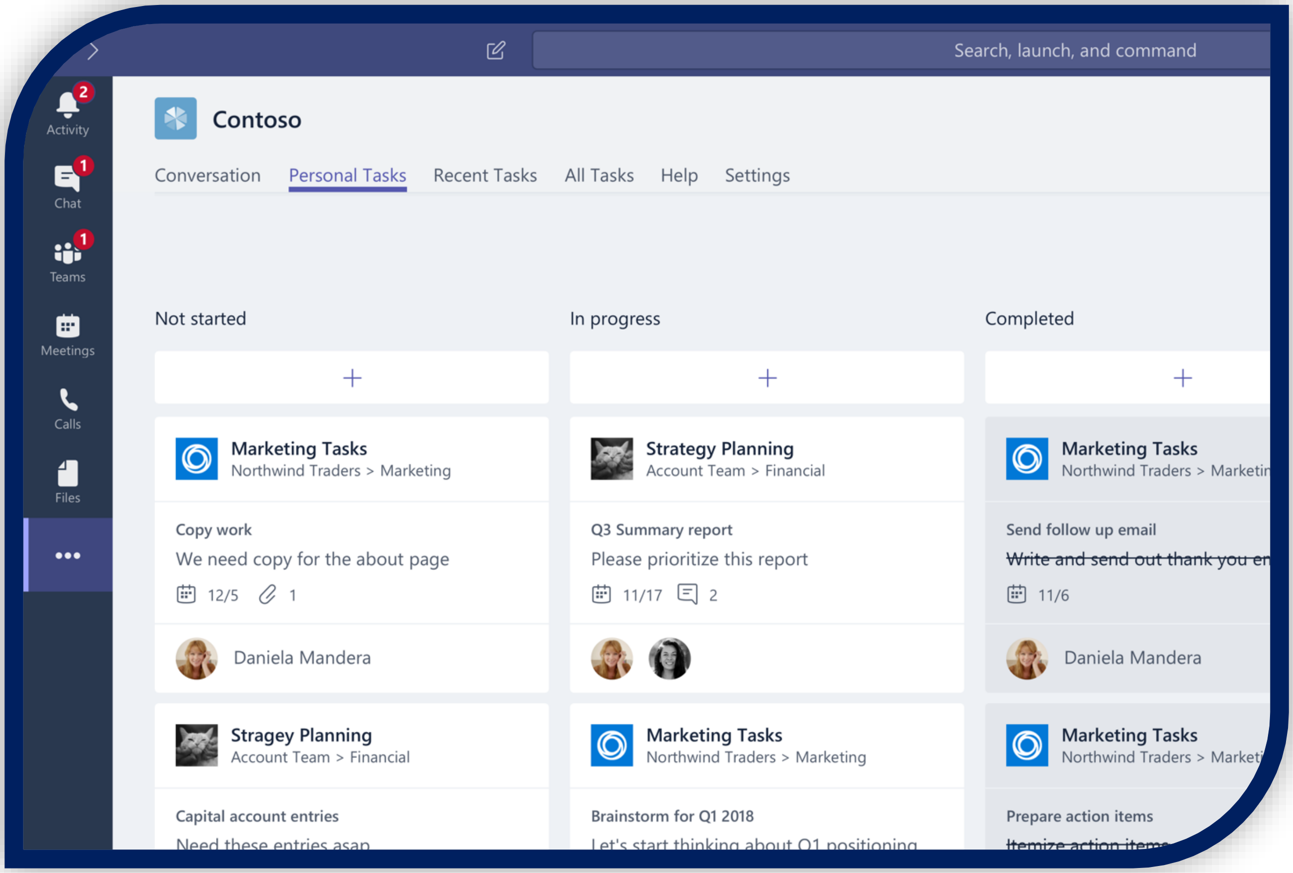
Babak Shammas 9:25 AM
Daniela, Daichi mentioned that th
before our Wednesday client sy

GUIDELINES

A personal app typically contains the following tabs:

The tab

This is where your users will see all their stuff. It's their personal space. The tab can be arranged as a list, a grid, columns, or a single canvas...whatever works best for your application. Since this tab can show items from multiple channels, each item should display its team, channel, and tab so the user can easily see where it originated.



Marketing Tasks
Northwind Traders > Marketing
Copy work
We need copy for the about page
12/5 1
Daniela Mandera

Stragey Planning
Account Team > Financial
Capital account entries
Need these entries asap

Strategy Planning
Account Team > Financial
Q3 Summary report
Please prioritize this report
11/17 2

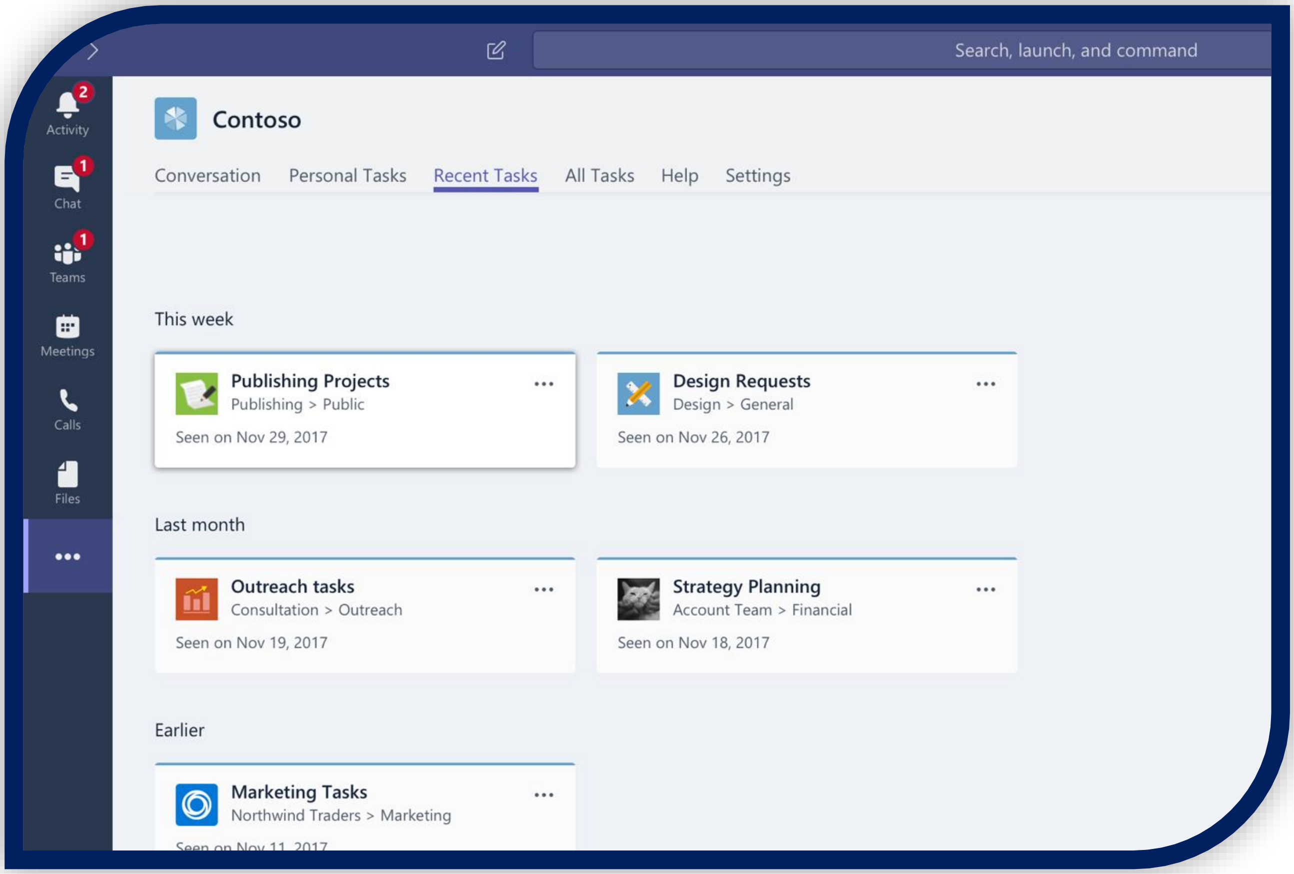
Marketing Tasks
Northwind Traders > Marketing
Brainstorm for Q1 2018
Let's start thinking about Q1 positioning

Marketing Tasks
Northwind Traders > Marketin
Send follow up email
~~Write and send out thank you er~~
11/6
Daniela Mandera

Marketing Tasks
Northwind Traders > Marketi
Prepare action items
~~Itemize action items~~

Recent

Recent tab lets someone browse everything they've recently viewed in the app. It's listed in chronological order (from most to least recent). Clicking on an item in this list will navigate the user to that item's channel and tab.



This week

Publishing Projects
Publishing > Public
Seen on Nov 29, 2017

Design Requests
Design > General
Seen on Nov 26, 2017

Last month

Outreach tasks
Consultation > Outreach
Seen on Nov 19, 2017

Strategy Planning
Account Team > Financial
Seen on Nov 18, 2017

Earlier

Marketing Tasks
Northwind Traders > Marketing
Seen on Nov 11, 2017

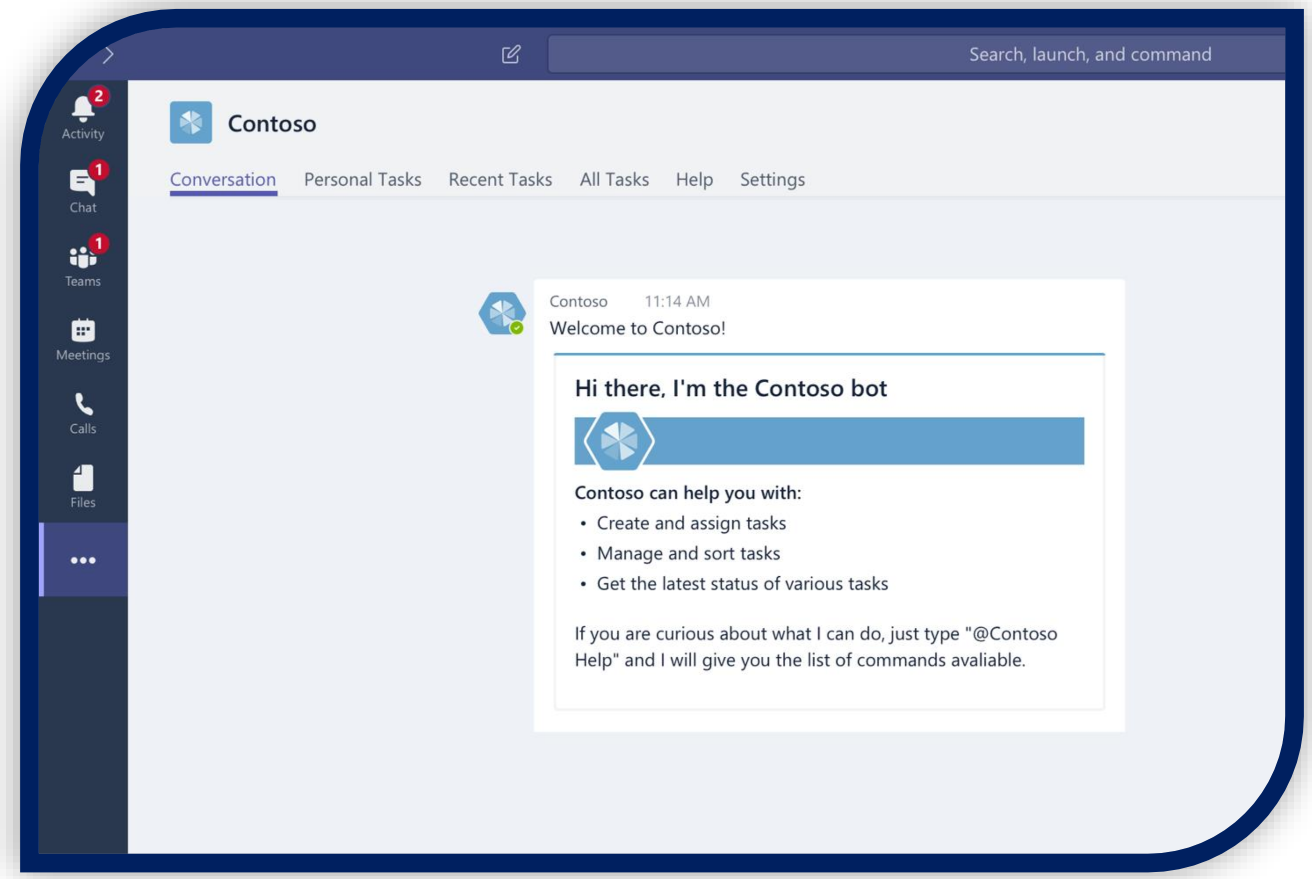
All

This is a list of all the tabs in the person's organization (the ones they have access to, anyway). In other words, it shows them everywhere the app is being used. As with the Recent tab, selecting something in the list will bring the user straight to the relevant channel and tab.

Bot

A bot isn't required, but it's a great way to communicate directly and privately with the users. Notification is one of the most important functions of a personal app and what better way to notify than with direct communication?

Bots deliver messages in the form of cards, which can provide specific information (like an alert that new content is available) or broad updates (like a daily to-do list).



Contoso 11:14 AM
Welcome to Contoso!

Hi there, I'm the Contoso bot

Contoso can help you with:
- Create and assign tasks
- Manage and sort tasks
- Get the latest status of various tasks

If you are curious about what I can do, just type "@Contoso Help" and I will give you the list of commands avaliable.

Help and Settings

Help content enables users to discover the nuances of the developed app. Add a Settings tab to give them the ability to further customize it.

About

Include an About tab to provide information like version number, capabilities, privacy, and permissions links.

CONCLUSION

In this case study, a brief introduction about the Microsoft Teams' Apps, its architecture, development & deployment ways & a demo of the introductory level is discussed in detail.

Our Microsoft Teams Consulting, add-in Development, Customization, Integration services, and solutions, can help companies maximize business performance, overcoming market challenges, achieving profitability, and providing the best customer care service.