# Exercise 1: Understanding the components of a Microsoft Teams app

## Task 1: Setup Environment for Teams development

### Install Yeoman and Gulp

To be able to scaffold projects using the Teams generator you need to install the Yeoman tool as well as the Gulp CLI task manager. 1. Open a **Command Prompt** window and execute the following:

```powershell  
npm install yo gulp-cli --global  
```

1. To install Yeoman generator for Microsoft Teams apps execute the following command:

* npm install generator-teams --global

## Task 2: Use Yeoman to create and test a basic Teams app

1. This task shows you how to create a basic Microsoft Teams app. You’ll use the Yeoman generator to scaffold and test the project.

* yo teams
* **Note**: Directory paths can become quite long after you import node modules. As a best practice, don’t use spaces in your dirno ectory names, and create your project in the root folder of your drive. That makes working with the solution easier in the future and protects you from potential issues associated with long file paths. In this example, you use \*\*C:\*\* as the working directory.

1. Yeoman launches and asks a series of questions. Answer them with these values:
   * **What is your solution name?**: MyTeamsApp
   * **Where do you want to place the files?**: Create a subfolder with solution name
   * **Title of your Microsoft Teams App project?**: Learn MSTeams App
   * **Your (company) name? (max 32 characters)**: Contoso
   * **Which manifest version would you like to use?**: 1.5
   * **Enter your Microsoft Partner Id, if you have one?**: (Leave blank to skip)
   * **What features do you want to add to your project?**: A Tab
   * **The URL where you will host this solution?**: https://myteamsapp.azurewebsites.net
   * **Would you like to include Test framework and initial tests?**: No
   * **Would you like to use Azure Applications Insights for telemetry?**: No
   * **Default Tab name? (max 16 characters)**: Learn Teams App
   * **Do you want to create a configurable or static tab?**: Static
2. Once Yeoman is finished generating the solution, navigate to the newly created project directory by executing the command: cd myteamsapp
3. Launch the project in Visual Studio Code by executing the command: code .

### Review the generated solution

The newly created project should now be launched in Visual Studio Code.

Microsoft Visual Studio Code showing expanded left navigation pane.

Microsoft Visual Studio Code showing expanded left navigation pane.

The source code for the application is in the **src\* folder. The** src\* folder contains the assets required to create the Teams app package.

Take a minute or two and familiarize yourself with how the code is organized - you can read more about that in the Project Structure documentation: <https://github.com/OfficeDev/generator-teams/wiki/Project-Structure>

### Build and test the app

Your Tab will be located in the **./src/app/scripts/ MyTeamsApp/LearnTeamsAppTab.tsx** file. This is the TypeScript React based class for your Tab. Locate the **render()** method and observe the code inside the <Surface> tag.

The <Panel> will contain <PanelHeader>, <PanelBody>, and <PanelFooter>.

<Surface>  
 <Panel>  
 <PanelHeader>  
 <div style={styles.header}>This is your tab</div>  
 </PanelHeader>  
 <PanelBody>  
 <div style={styles.section}>  
 {this.state.entityId}  
 </div>  
 <div style={styles.section}>  
 <PrimaryButton onClick={() => alert("It worked!")}>A sample button</PrimaryButton>  
 </div>  
 </PanelBody>  
 <PanelFooter>  
 <div style={styles.footer}>  
 (C) Copyright Contoso  
 </div>  
 </PanelFooter>  
 </Panel>  
 </Surface>

1. Before you can build your Teams App, you must first create a Teams App manifest that you will upload/sideload into Microsoft Teams.
2. Open Terminal in Visual Studio Code. From the **Visual Studio Code** ribbon select **Terminal > New Terminal**.
3. Create the manifest by executing the command: gulp manifest
   1. This will validate the manifest and create a zip file in the **./package** directory.

* Package directory expanded in VS Code.
* Package directory expanded in VS Code.

1. To build your solution you use the gulp build command.
   1. This will transpile your solution into the **./dist** folder.

dist directory expanded in VS Code.

dist directory expanded in VS Code.

### Run your app

To run your app you use the gulp serve command. This will build and start a local web server for you to test your app. The command will also rebuild the application whenever you save a file in your project. You should now be able to browse to **http://localhost:3007/learnTeamsAppTab/** to ensure that your tab is rendering. App running in browser as localhost test.

Leave this local browser session running and then proceed with the next task.

## Task 3: Update runni ng Visual Studio Code solution

Now let’s modify the content inside the header, body and footer of your tab solution using Visual Studio Code.

1. Go back to the solution in Visual Studio Code.
   1. Ensure the **./src/app/scripts/ MyTeamsApp/LearnTeamsAppTab.tsx** file is open.
2. Modify the header content of the tab. Find the <PanelHeader> tag and replace the markup with the <div> as displayed below:

* <PanelHeader>  
   <div style={styles.header}>My First Teams App</div>  
   </PanelHeader>

1. Modify the body content of the tab. Find the <PanelBody> tag and replace the entire div that contains {this.state.entityId} with: <div style={styles.section}>Hello World! Yo Teams rocks!</div>
2. Your <PanelBody> should now look like the code below.

* <PanelBody>  
   <div style={styles.section}>Hello World! Yo Teams rocks!</div>  
   <div style={styles.section}>  
   <PrimaryButton onClick={() => alert("It worked!")}>  
   A sample button  
   </PrimaryButton>  
   </div>  
   </PanelBody>

1. Modify the footer content of the tab. Find the **Copyright** text inside the <PanelFooter> and add the current year.

* <PanelFooter>  
   <div style={styles.footer}>(C) 2019 Copyright Contoso</div>  
   </PanelFooter>

1. Save the solution. Go back to the browser and refresh the running session. Your tab content should now look similar to the image below.

* Showing 2019 Copyright Contoso
* Showing 2019 Copyright Contoso

1. Select on **A sample button**. A dialog will appear with the message **It worked!**. Select **OK**.

* It worked message showing in dialog pop up.
* It worked message showing in dialog pop up.

1. Go ahead and spend a few minutes modifying the header, body and footer with your preferred content and controls.
2. Once you’re done terminate the process by typing **CTRL+C** in the Visual Studio Code terminal window and then type **Y** when prompted to terminate.

Terminal window in VS Code Terminate batch job yes no question.

Terminal window in VS Code Terminate batch job yes no question.

## Task 4: Run your app in Microsoft Teams

Now that you’ve tested your tab, it’s time to run your app inside Microsoft Teams.

**Note**: Microsoft Teams does not allow you to have your app hosted on localhost, so you need to either publish it to a public URL or use a proxy such as ngrok. Good news is that the scaffolded project has this built-in. When you run gulp ngrok-serve the ngrok service will be started in the background, with a unique and public DNS entry and it will also package the manifest with that unique URL and then do the exact same thing as gulp serve.

The generated application is ready to run. The generator created a gulp task to facilitate development.

1. Start this task by running the following command in the Terminal window of the open project in Visual Studio Code: gulp ngrok-serve

* Command Prompt running gulp ngrok-serve
* Command Prompt running gulp ngrok-serve
* **Note**: The gulp serve process must be running in order to see the tab in the Microsoft Teams application. When the process is no longer needed, press **CTRL+C** to cancel the server.

1. Next, upload the app to Microsoft Teams. In the Teams application, select the **Create and join team** link. Then select the **Create team** button.

* Create and join team link > Create team button
* Create and join team link > Create team button

1. Enter a team name and description. In this example, the team is named **Training Content**. Select **Next**.
   1. Optionally, invite others from your organization to the team. This step can be skipped in this lab.
2. The new team is shown. In the side panel on the left, select the ellipses next to the team name. Choose **Manage team** from the context menu.

* Choose Manage team from the context menu
* Choose Manage team from the context menu

1. Select **Apps** tab, then select **More** **Apps**.

* More apps button
* More apps button

1. Select the **Upload a custom app** link at the lower right corner of the application then select **Upload for Contoso**.

* Apps > Upload a custom app
* Apps > Upload a custom app

1. Select the **MyTeamsApp.zip** file from the **package** folder. Select **Open**.

* Open teams-app-1.zip
* Open teams-app-1.zip

1. The app appears and displays the description and icon from the manifest.

* Learn Teams App tile
* Learn Teams App tile

1. Select the app. From the dialog, select **Add**.

* **Note**: Notice there is TODO text for the short and full description. These values can be updated in your solution but for now go ahead and proceed with adding the app.

1. The app is now uploaded into the Microsoft Teams and loads as an app with the custom tab. If you select on the **A sample button**, a dialog should appear saying **It worked!.**
2. If you navigate away from the app, you can access it again by selecting it from the **… More added apps** dialog.

More added apps.

More added apps.

## Review

In this exercise, you:

* Ran Yeoman Teams generator and reviewed the result.
* Updated the manifest for the app and uploaded the app to Teams.
* Added a tab to a team view in Teams.