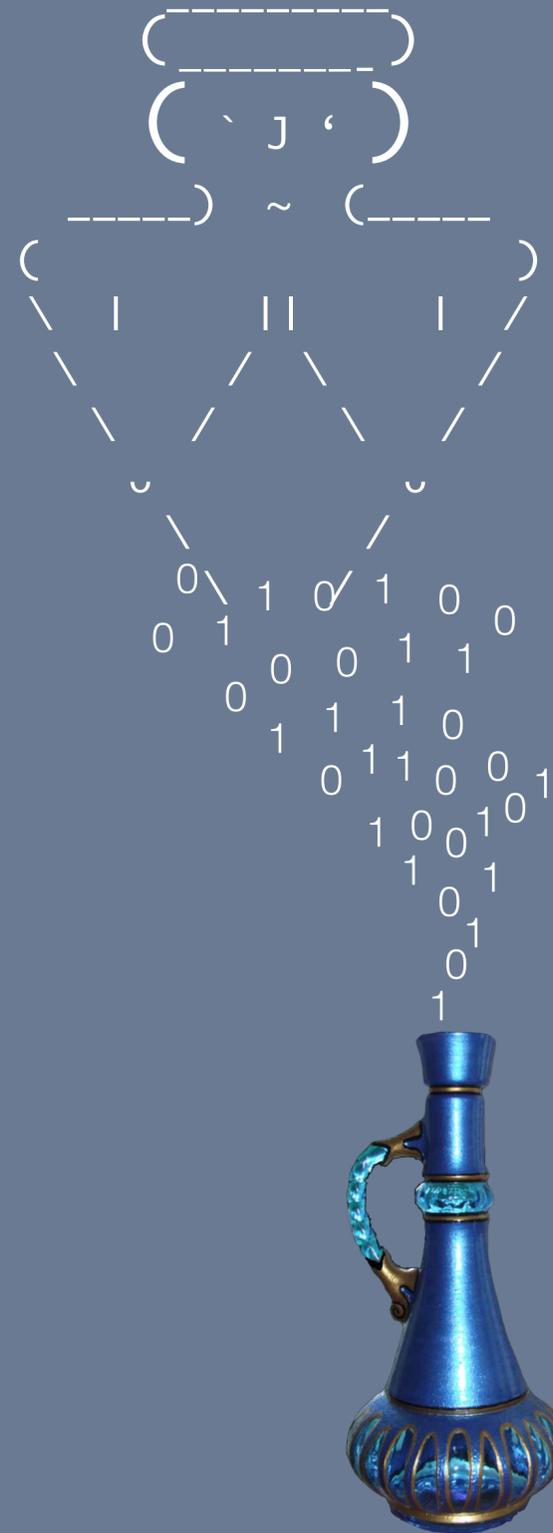




Bringing bash
to
Xcode

i.e., letting the genie out
of the bottle





Leslie N. Helou

Senior Professional Services Engineer
Jamf

What we will accomplish:

- Create an application based on a bash script.
- Customized GUI asking for the computer name, providing feed back to the user.
- Build the application so that it is properly signed.

Goal 1: Create the app.

Start with a bash script* to rename a computer:

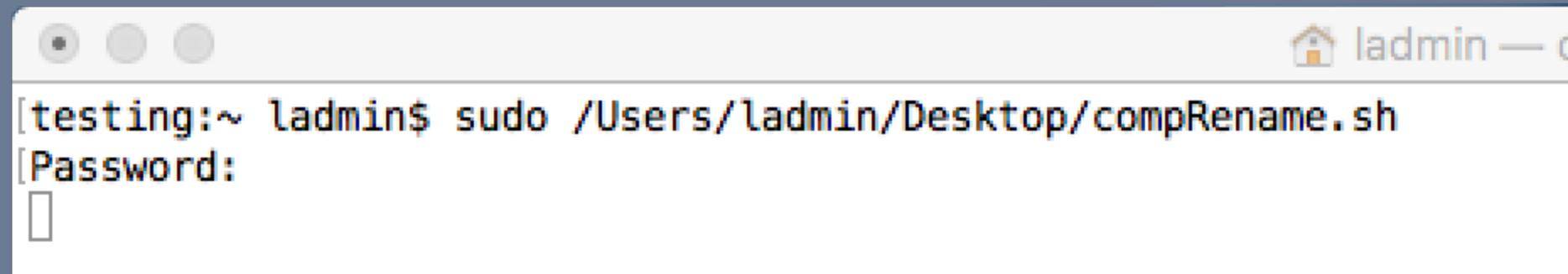
```
#!/bin/bash
# check that script is run as root user
if [ $EUID -ne 0 ]
then
    /bin/echo $'\nThis script must be run as the root user!\n'
    exit
fi
# capture user input name
while true;do
    name=$(osascript -e 'Tell application "System Events" to display dialog "Please enter the name for your computer or select Cancel." default answer "" -e 'text
returned of result' 2>/dev/null)
    if [ $? -ne 0 ];then # user hit cancel
        exit
    elif [ -z "$name" ];then # loop until input or cancel
        osascript -e 'Tell application "System Events" to display alert "Please enter a name or select Cancel... Thanks!" as warning'
    elif [ -n "$name" ];then
        # user input
        break
    fi
done
scutil --set ComputerName "$name"
scutil --set LocalHostName "$name"
scutil --set HostName "$name"

jamf recon
```

*Author - Daniel Mintz

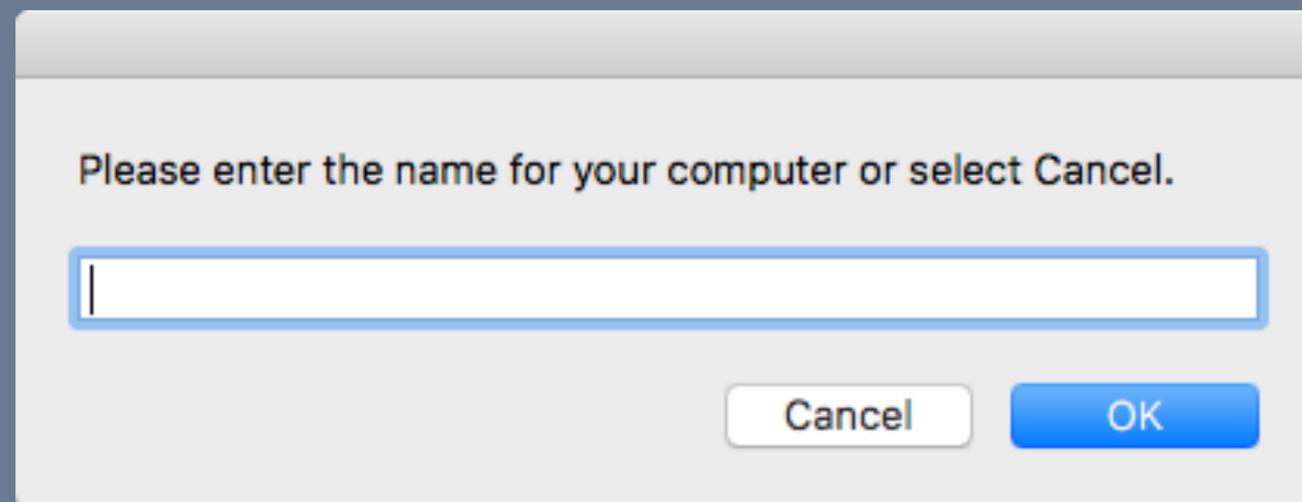
<https://github.com/BIG-RAT/MacAdmins2017/blob/master/compRename.sh>

Quick look at the script

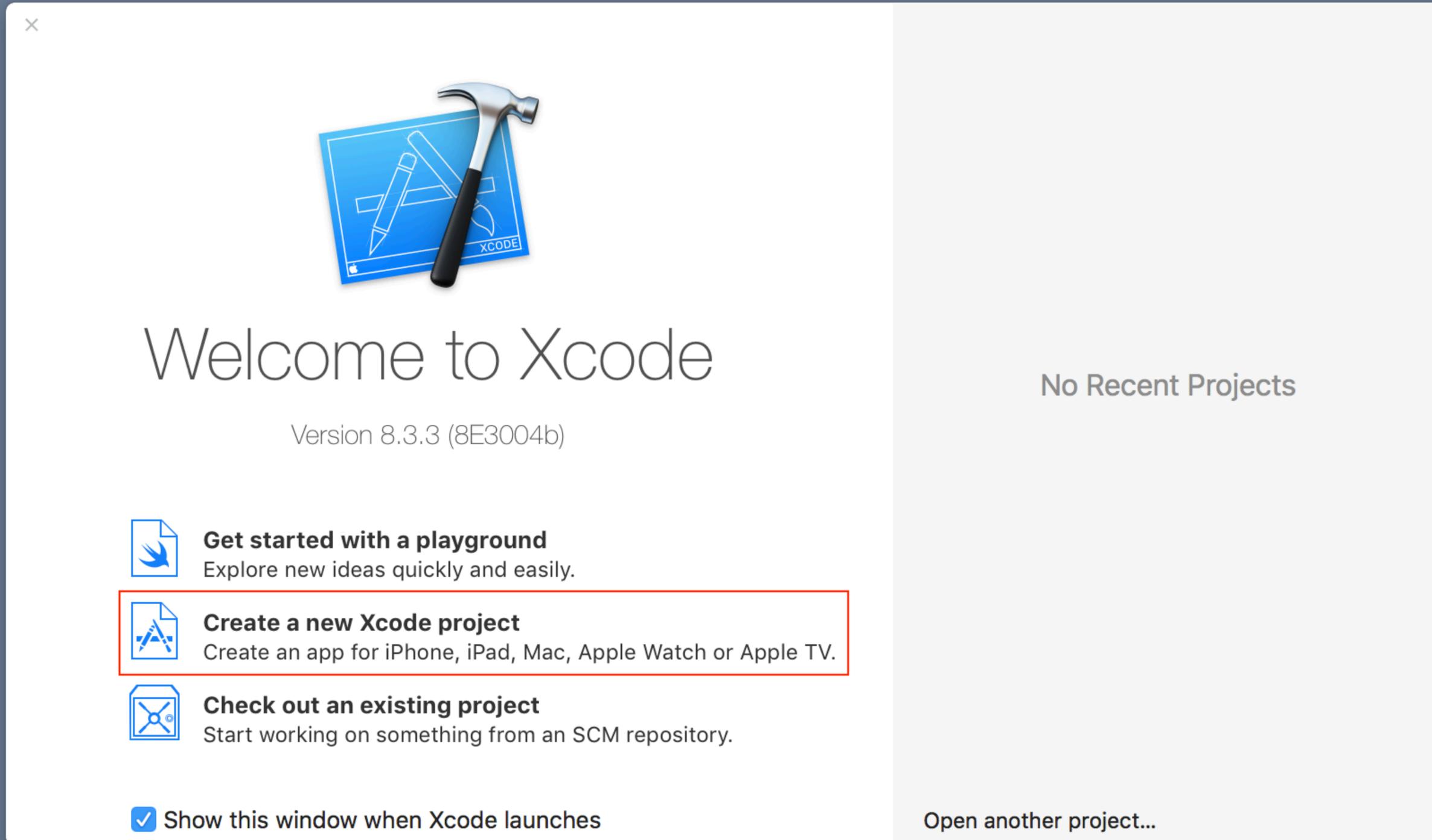


```
admin — o
[testing:~ ladmin$ sudo /Users/ladmin/Desktop/compRename.sh
[Password:
]
```

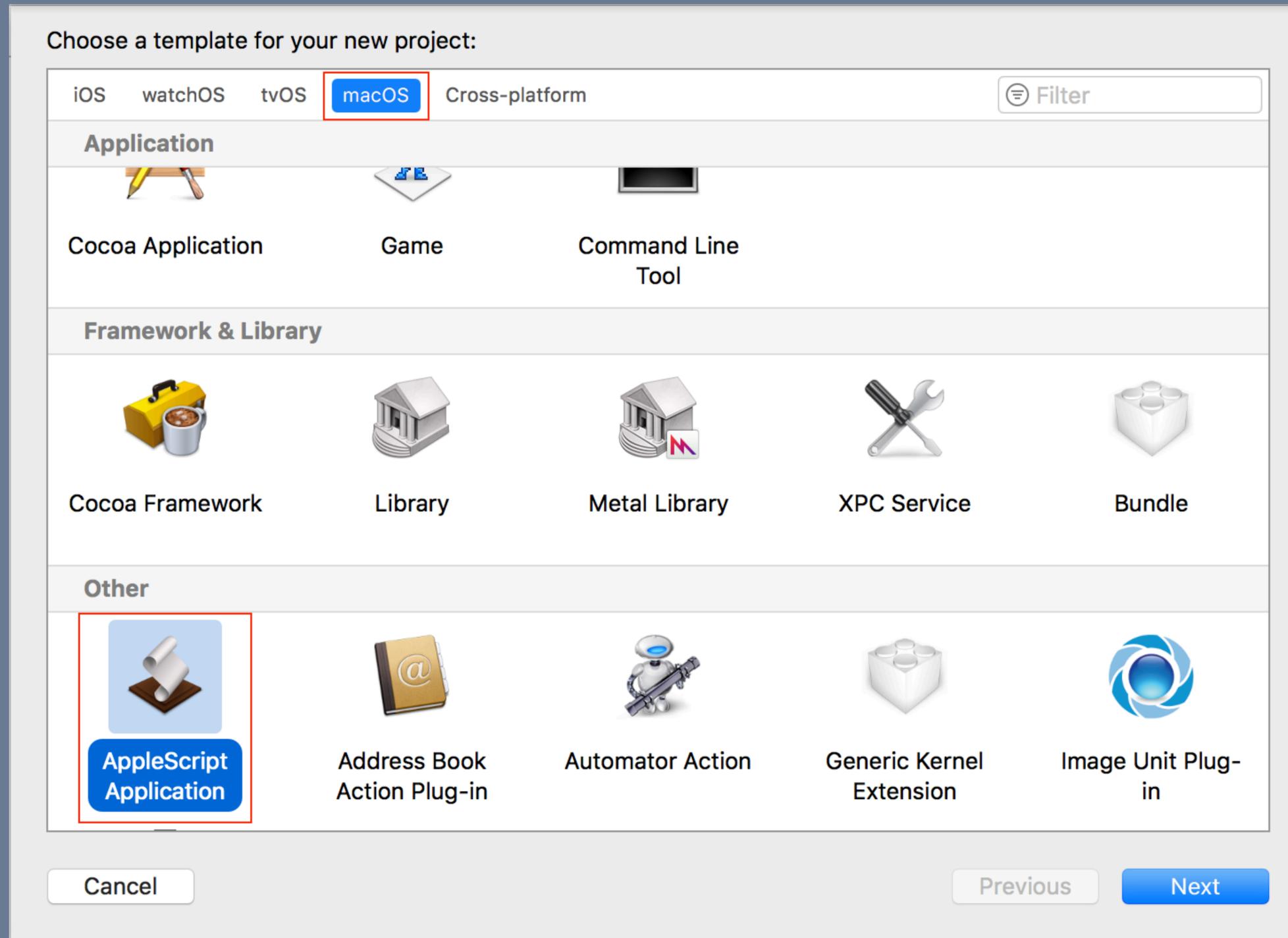
We're greeted with a simple window.



Launch Xcode and create a new project



Select macOS and AppleScript Application



Name and save project

Product Name:

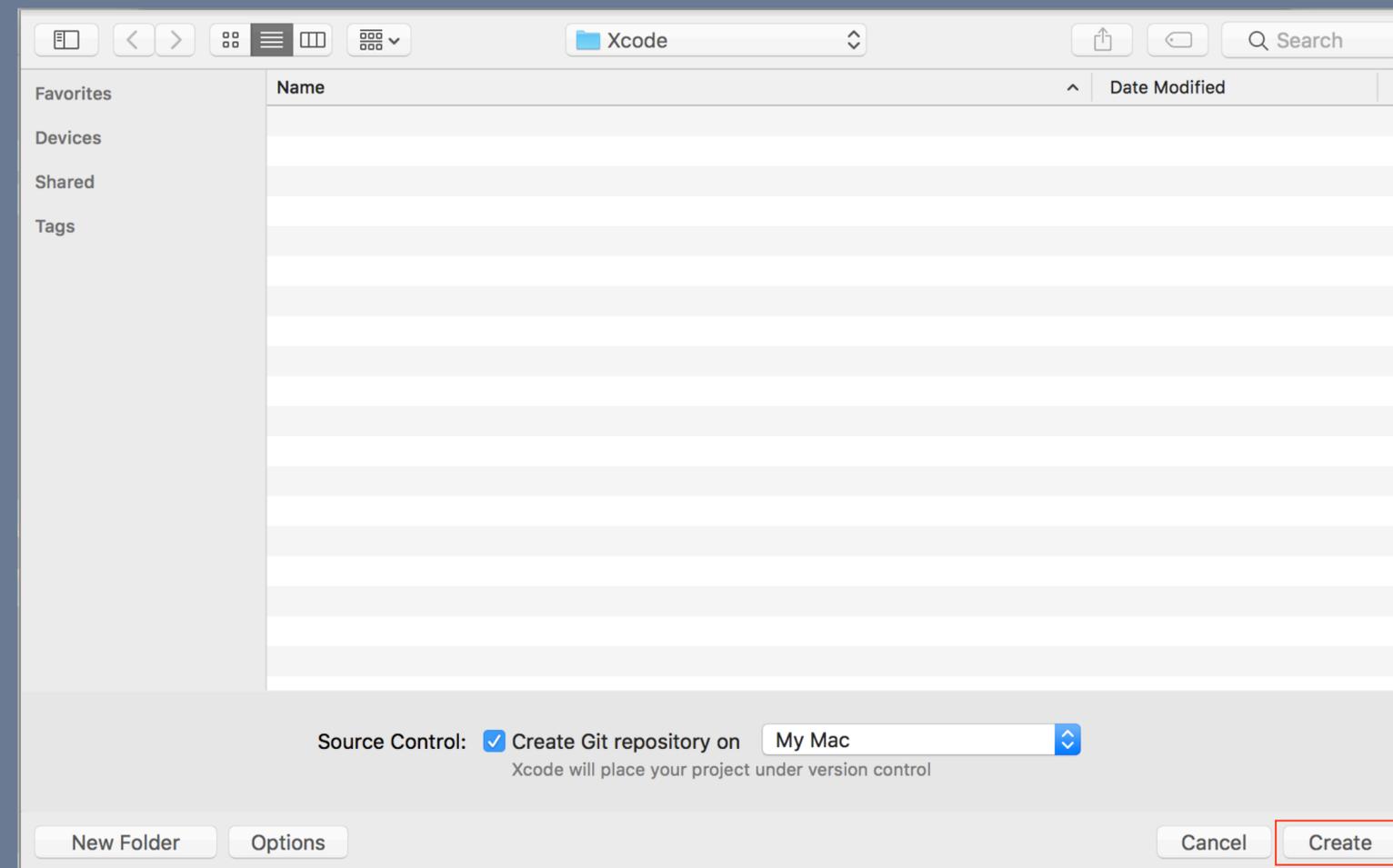
Team:

Organization Name:

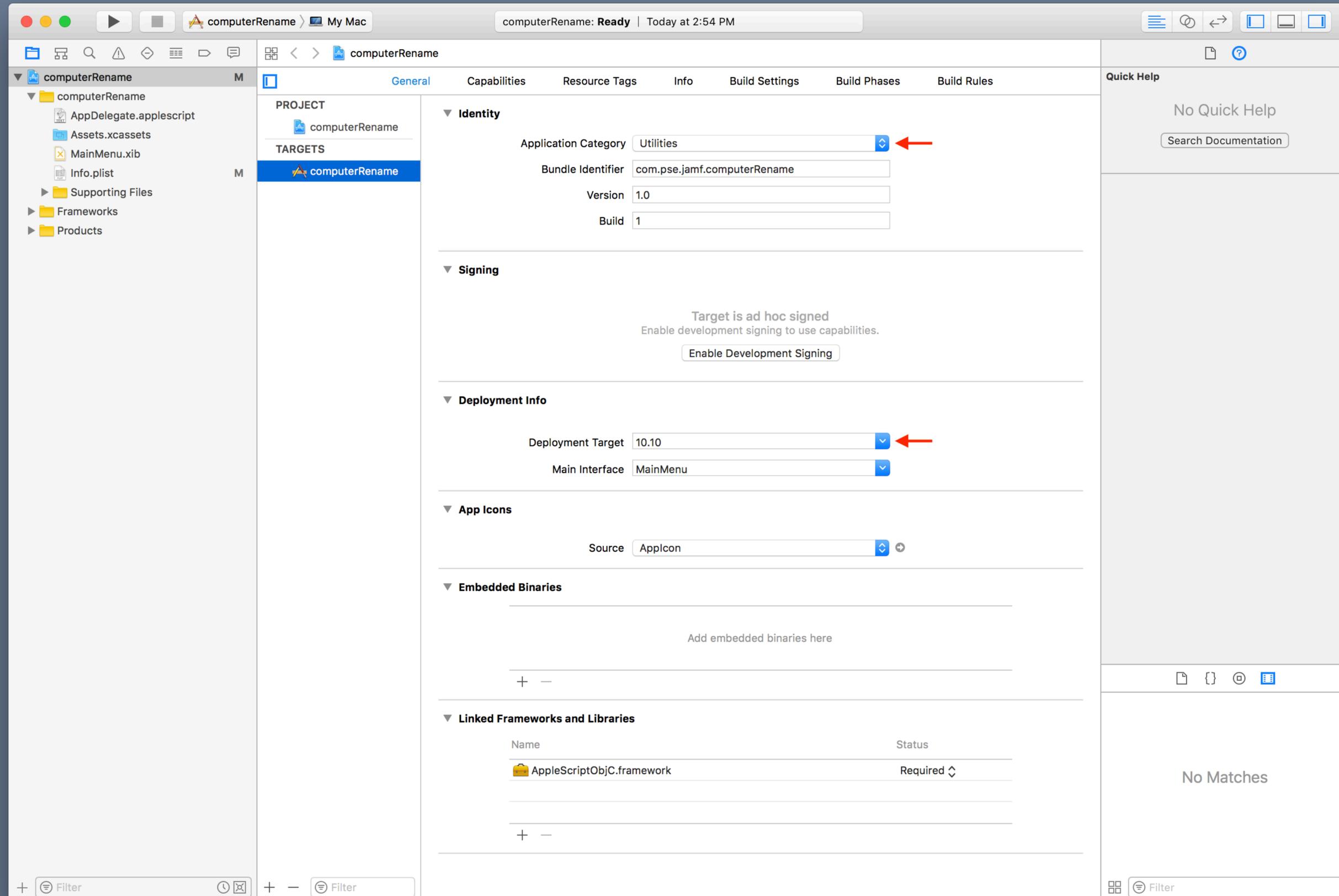
Organization Identifier:

Bundle Identifier:

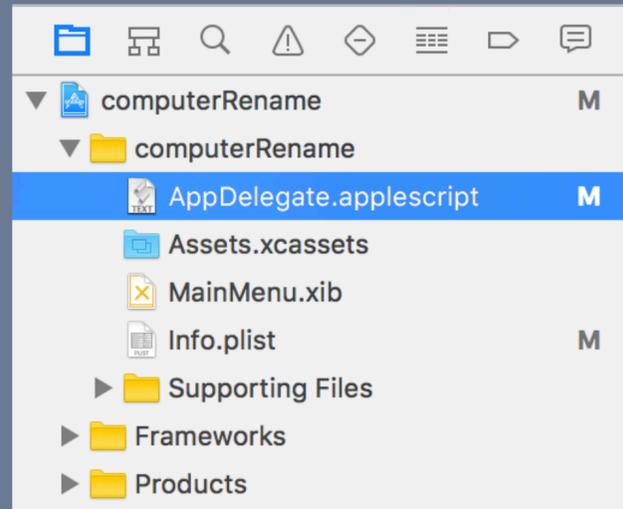
Create Document-Based Application



The main window for our project:



Bring the bash script into our project my first clicking AppDelegate.applescript, then paste the script into the applicationWillFinishLaunching_ section.



```
on applicationWillFinishLaunching_(aNotification)
    -- Insert code here to initialize your application before any files are opened
    #!/bin/bash
    # check that script is run as root user
    if [ $EUID -ne 0 ]
    then
    /bin/echo $'\nThis script must be run as the root user!\n'
    exit
    fi
    # capture user input name
    while true;do
    name=$(osascript -e 'Tell application "System Events" to display dialog "Please enter the name for your computer
        or select Cancel." default answer "" -e 'text returned of result' 2>/dev/null)
    if [ $? -ne 0 ];then # user hit cancel
    exit
    elif [ -z "$name" ];then # loop until input or cancel
    osascript -e 'Tell application "System Events" to display alert "Please enter a name or select Cancel... Thanks!"
        as warning'
    elif [ -n "$name" ];then
    # user input
    break
    fi
    done
    scutil --set ComputerName "$name"
    scutil --set LocalHostName "$name"
    scutil --set HostName "$name"

    jamf recon

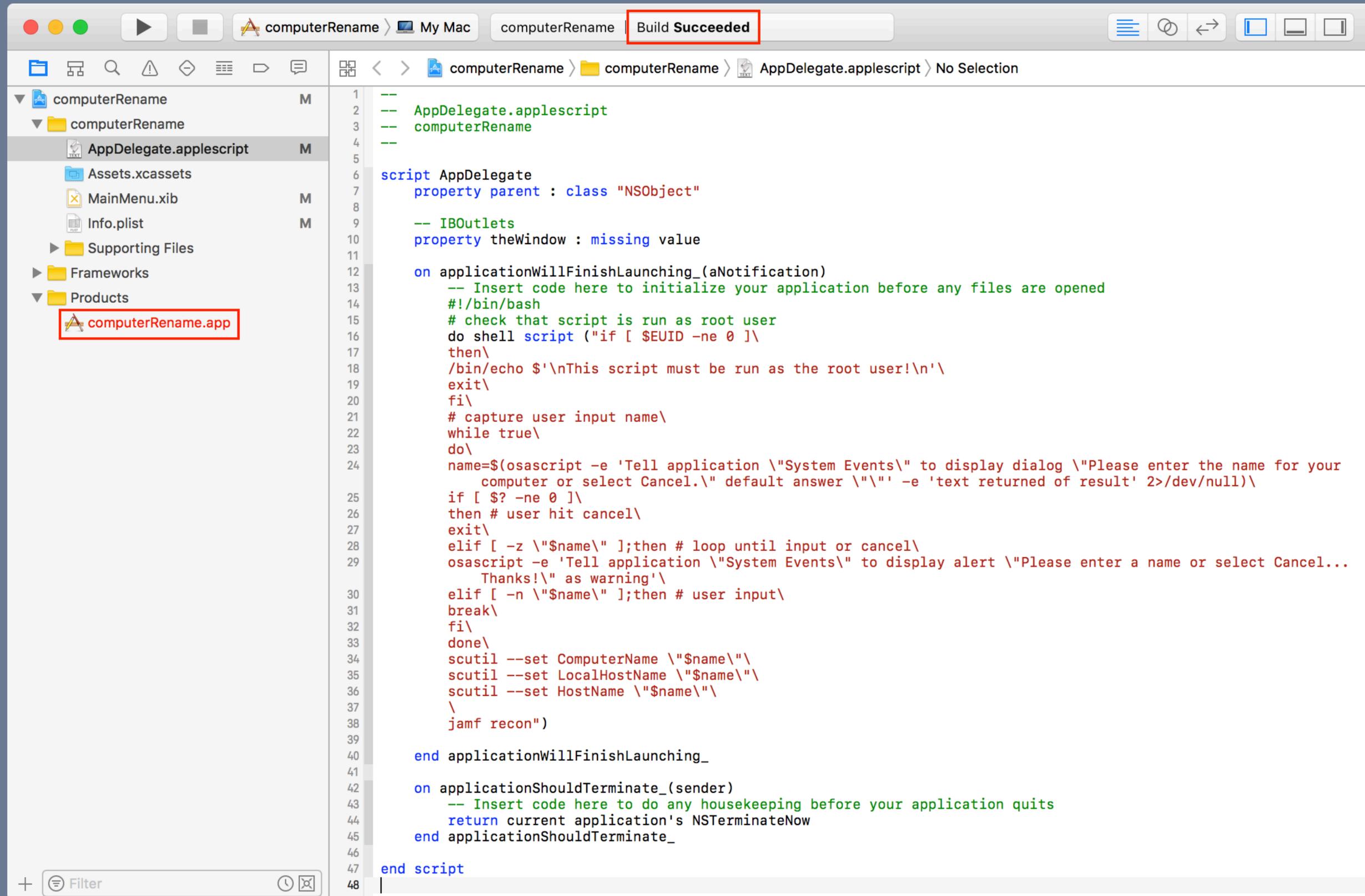
end applicationWillFinishLaunching_
```

As we're in an AppleScript environment we need to enclose the bash script in a 'do shell script("...")'. Escape existing quotes (\") and newlines (add \ to the end of each line).

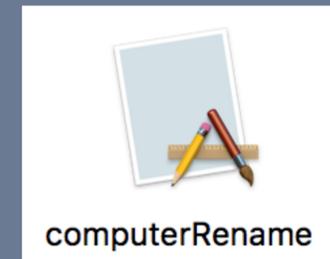
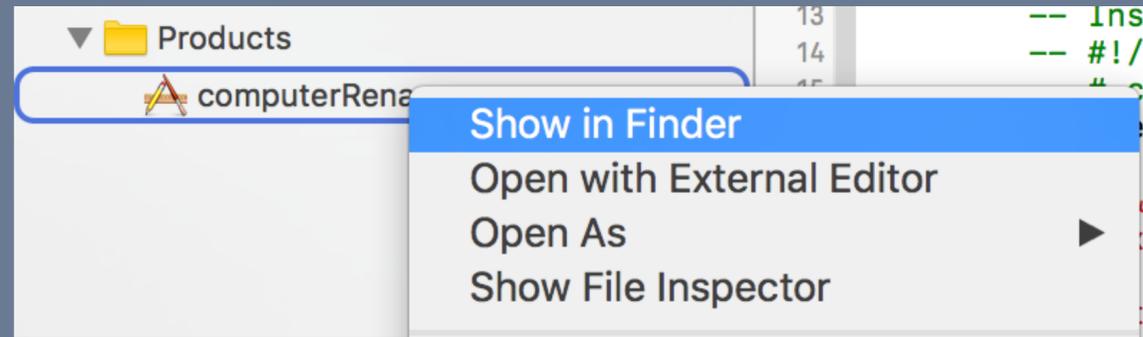
```
--Insert code here to initialize your application before any files are opened
#!/bin/bash
# check that script is run as root user
do shell script ("if [ $EUID -ne 0 ]\
then\
/bin/echo $\nThis script must be run as the root user!\n'\
exit\
fi\
# capture user input name\
while true\
do\
name=$(osascript -e 'Tell application \"System Events\" to display dialog \"Please enter the name for your
computer or select Cancel.\" default answer \"\" -e 'text returned of result' 2>/dev/null)\
if [ $? -ne 0 ]\
then # user hit cancel\
exit\
elif [ -z \"$name\" ];then # loop until input or cancel\
osascript -e 'Tell application \"System Events\" to display alert \"Please enter a name or select Cancel...
Thanks!\n\" as warning'\
elif [ -n \"$name\" ];then # user input\
break\
fi\
done\
scutil --set ComputerName \"$name\"\
scutil --set LocalHostName \"$name\"\
scutil --set HostName \"$name\"\
\
jamf recon")
```

https://github.com/BIG-RAT/MacAdmins2017/blob/master/applicationWillFinishLaunching_1.txt

Let's see if we can build the application - command+b

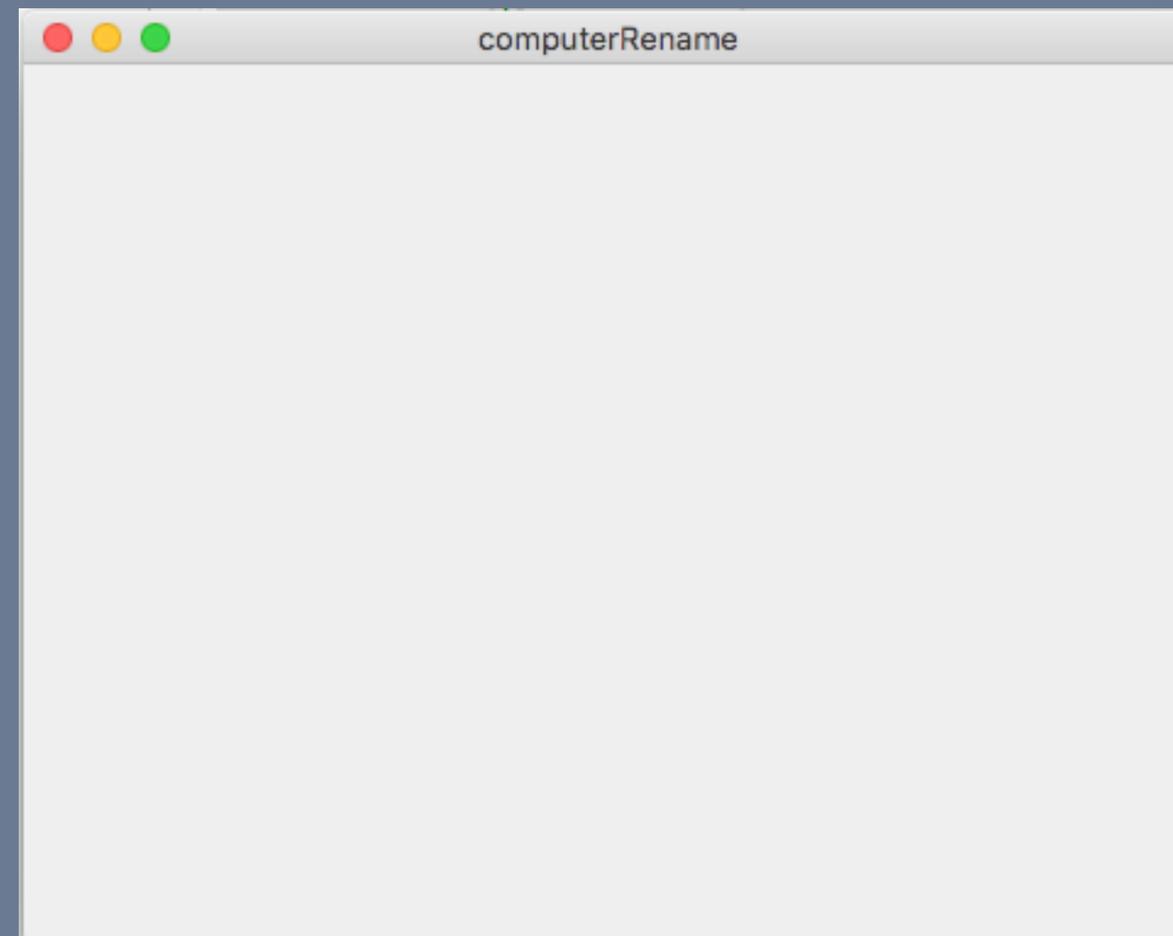


Now let's give it run - on a test machine of course.

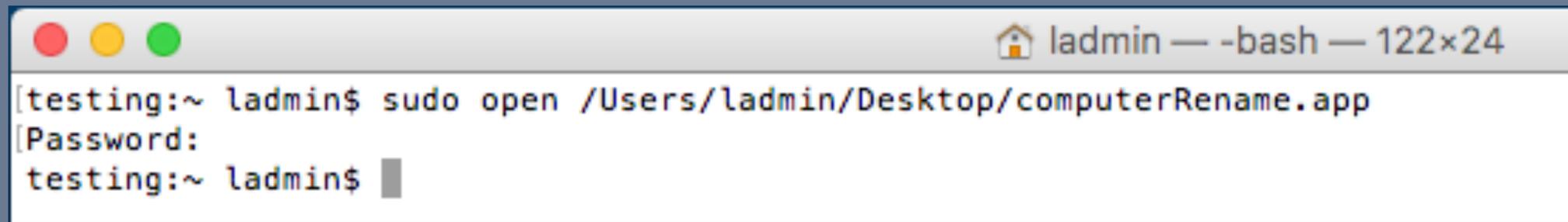


Double-click to launch.

We should see the following, just a blank window.

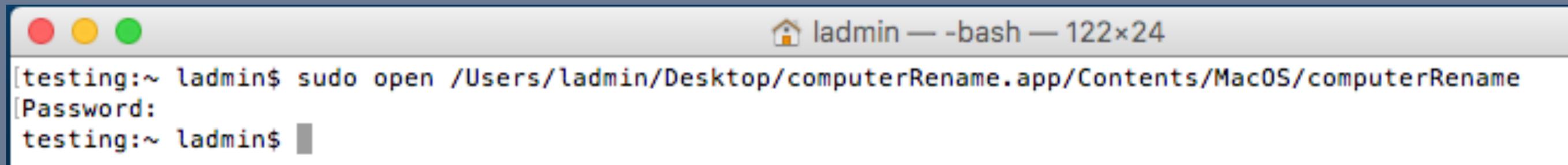


So let's launch the application with elevated



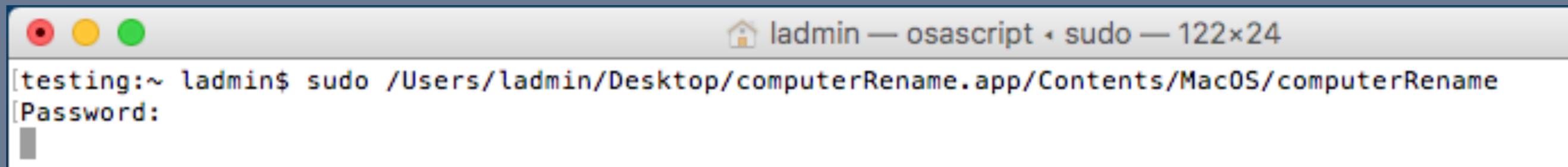
```
admin — -bash — 122x24
[testing:~ ladmin$ sudo open /Users/ladmin/Desktop/computerRename.app
[Password:
testing:~ ladmin$
```

Nope, try a different approach.



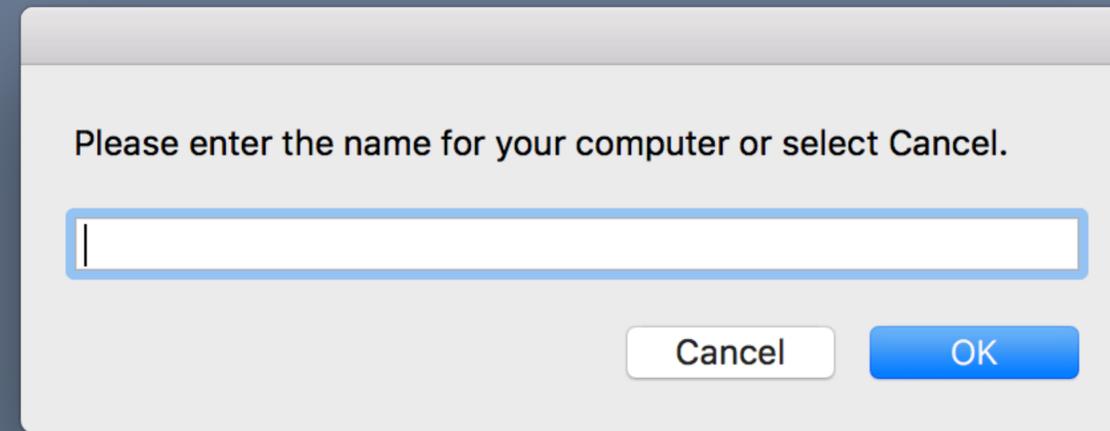
```
admin — -bash — 122x24
[testing:~ ladmin$ sudo open /Users/ladmin/Desktop/computerRename.app/Contents/MacOS/computerRename
[Password:
testing:~ ladmin$
```

Third time's a charm?



```
admin — osascript + sudo — 122x24
[testing:~ ladmin$ sudo /Users/ladmin/Desktop/computerRename.app/Contents/MacOS/computerRename
[Password:
█
```

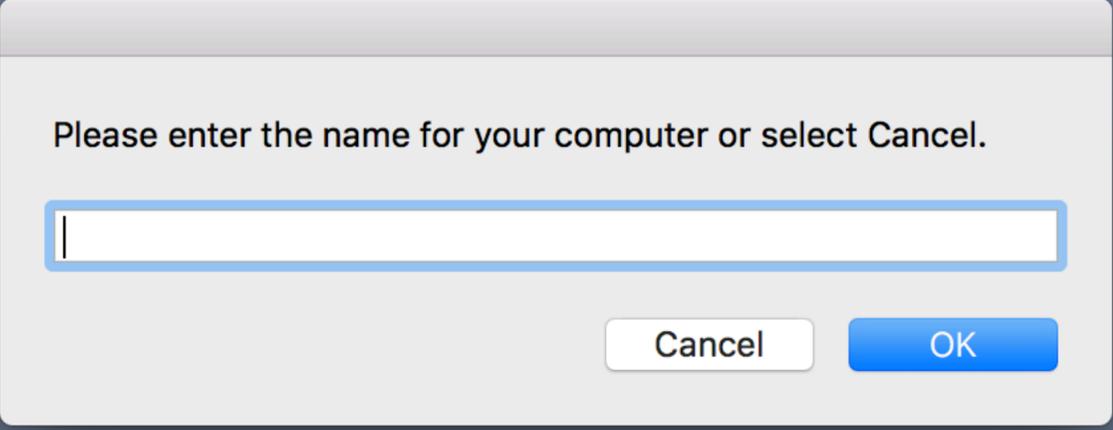
SCORE! We now get prompted for the computer name.



Might as well continue and see if it works...

Now let's dress it up a bit.

Go from here...



Please enter the name for your computer or select Cancel.

A macOS-style dialog box with a light gray background and a white title bar. The text "Please enter the name for your computer or select Cancel." is centered at the top. Below the text is a white text input field with a blue border and a vertical cursor on the left. At the bottom right, there are two buttons: a white "Cancel" button and a blue "OK" button.

Now let's dress it up a bit.



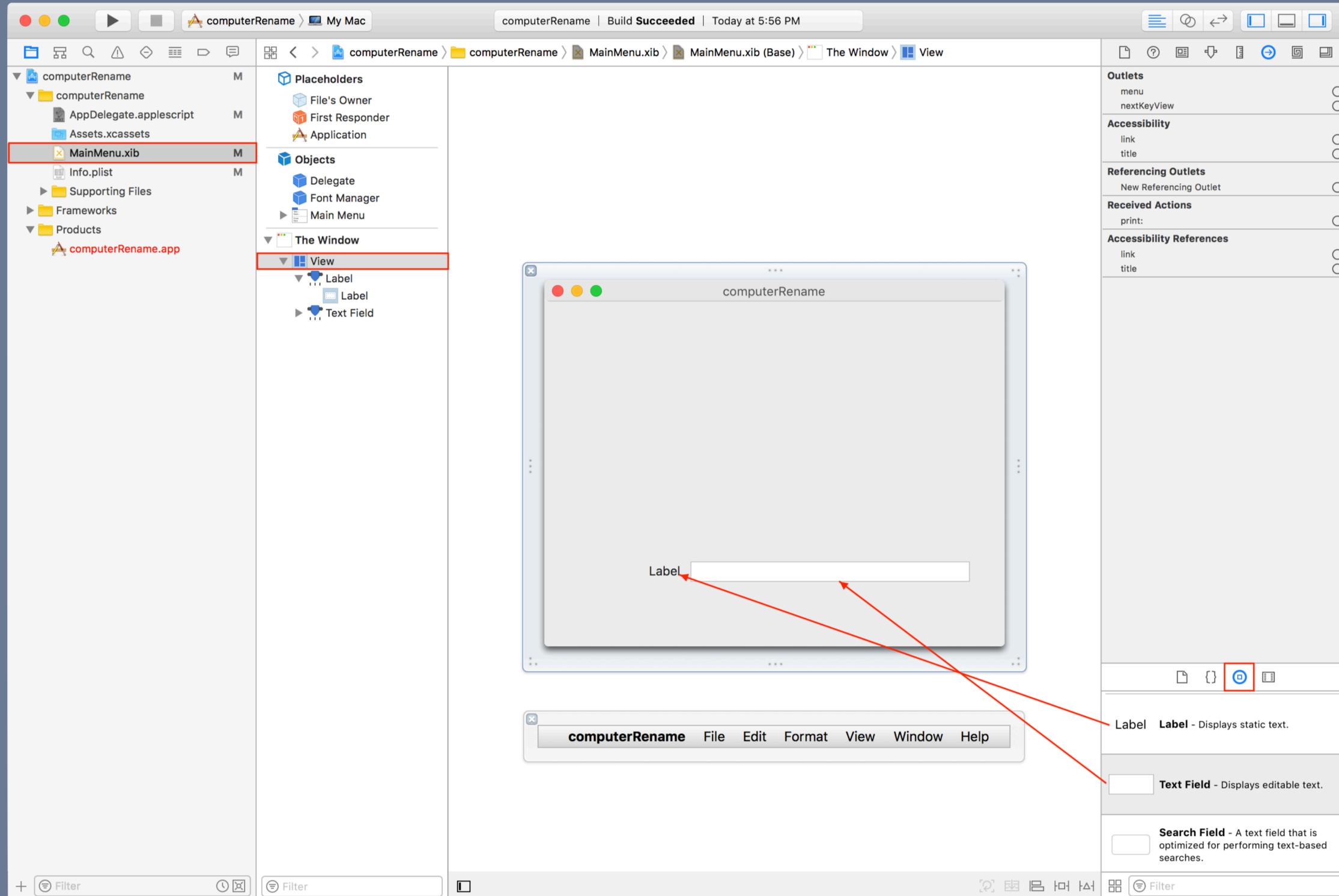
...to here.

Comment out the existing script and add a new section.

```
    -- Insert code here to initialize your application before any files are opened
#!/bin/bash
# check that script is run as root user
#
# do shell script ("if [ $EUID -ne 0 ]\
# then\
# /bin/echo $'\nThis script must be run as the root user!\n'\
# exit\
# fi\
# # capture user input name\
# while true\
# do\
# name=$(osascript -e 'Tell application \"System Events\" to display dialog \"Please
enter the name for your computer or select Cancel.\" default answer \"\" -e 'text returned
of result' 2>/dev/null)\
# if [ $? -ne 0 ]\
# then # user hit cancel\
# exit\
# elif [ -z \"$name\" ];then # loop until input or cancel\
# osascript -e 'Tell application \"System Events\" to display alert \"Please enter a
name or select Cancel... Thanks!\" as warning'\
# elif [ -n \"$name\" ];then # user input\
# break\
# fi\
# done\
# scutil --set ComputerName \"$name\"\
# scutil --set LocalHostName \"$name\"\
# scutil --set HostName \"$name\"\
# \
# jamf recon")
```

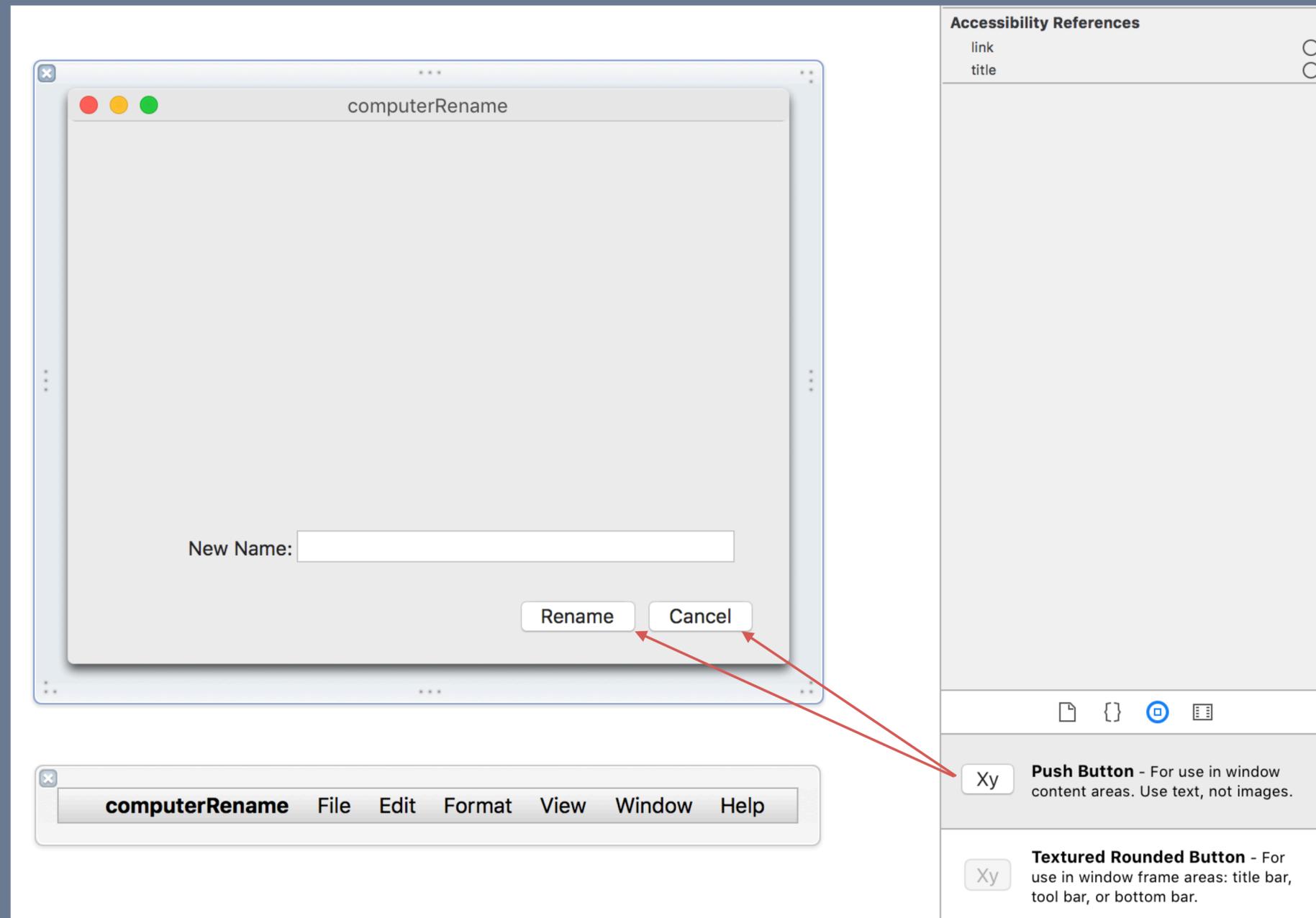
Hint: Select the block of code to comment out, then press command+/'

Goal 2: The GUI



Add a label along with a text field to type the computer name into.

Add a couple buttons.



Double click the label and buttons to add appropriate names.

Back to the code, let's add some lines.

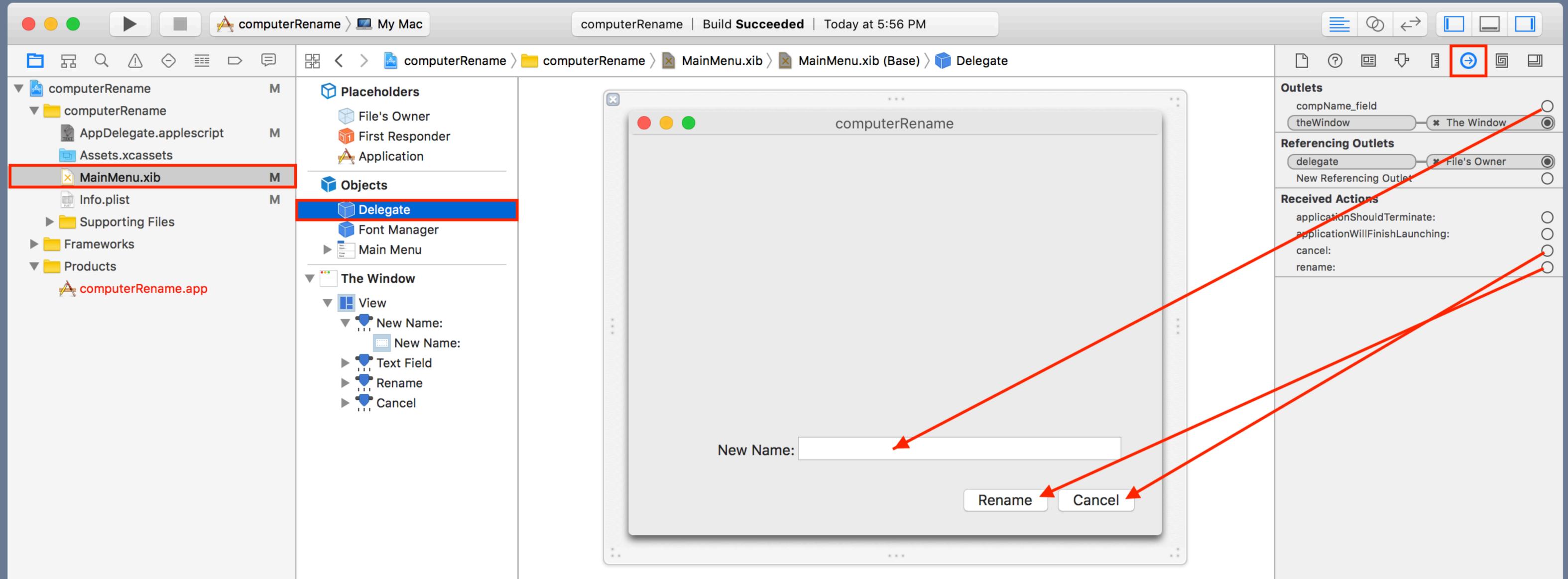
```
1  --
2  -- AppDelegate.applescript
3  -- computerRename
4  --
5
6  script AppDelegate
7      property parent : class "NSObject"
8
9      -- IBOutlets
10     property theWindow : missing value
11     property compName_field : missing value
12
13     on rename_(sender)
14         set newName to compName_field's stringValue()
15
16         do shell script ("
17         /usr/sbin/scutil --set ComputerName \"\" & newName & "\"\"
18         /usr/sbin/scutil --set LocalHostName \"\" & newName & "\"\"
19         /usr/sbin/scutil --set HostName \"\" & newName & "\"\"
20         \
21         jamfBin=$(which jamf)\
22         $jamfBin recon")
23
24         quit
25     end rename
26
27     on cancel_(sender)
28         quit
29     end cancel_
30
31     on applicationWillFinishLaunching_(aNotification)
32         -- Insert code here to initialize your application before any files are opened
```

For the computer name field

For the Rename button

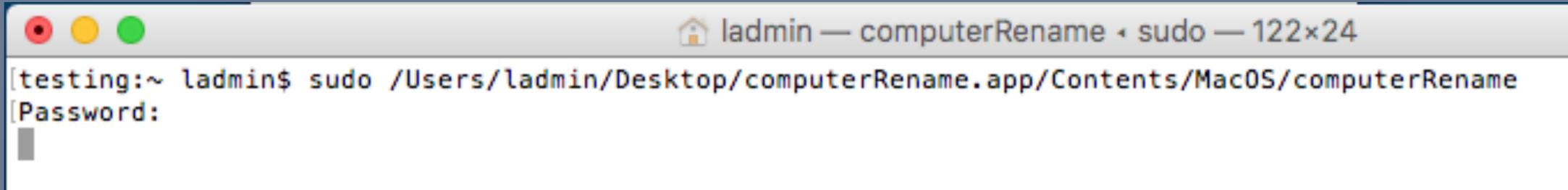
For the Cancel button

Connect our new lines of code to the GUI objects.



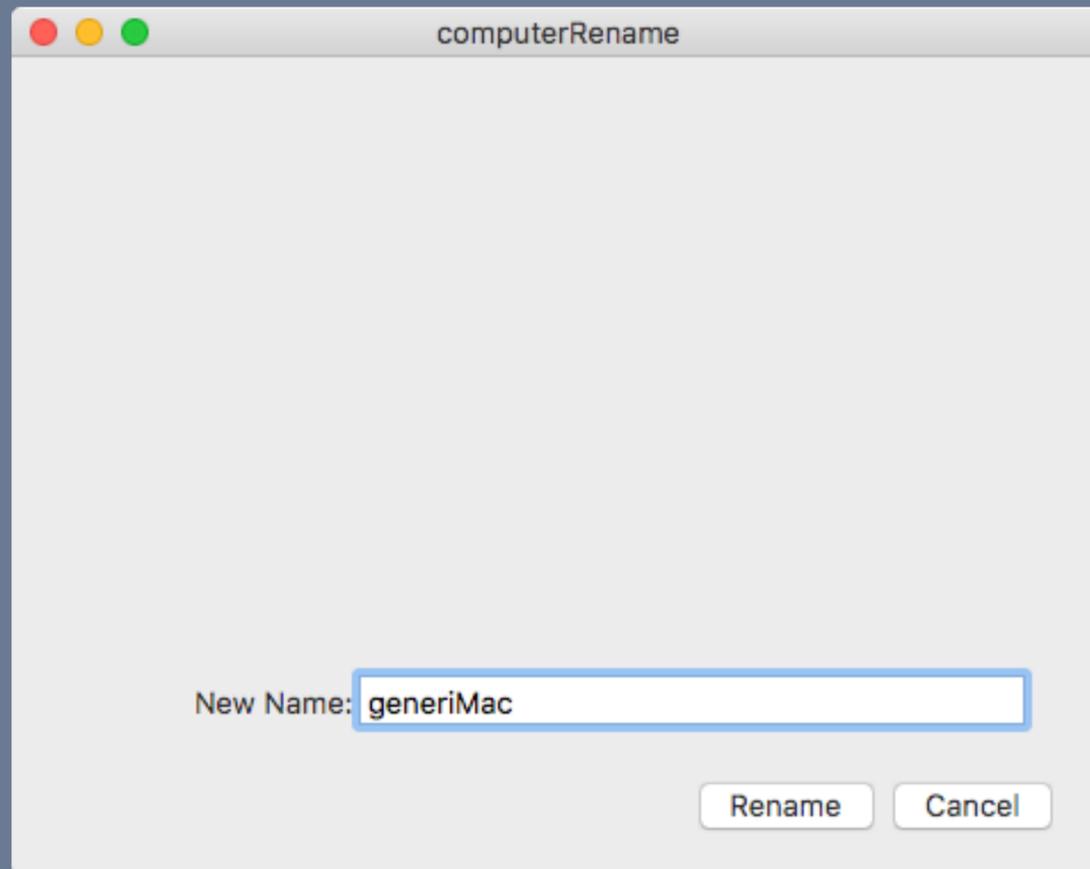
Click and drag the open circle to the associated object.

Build the app, command+b, and test it. Remember to launch it through Terminal.



```
ladmin — computerRename • sudo — 122x24
[testing:~ ladmin$ sudo /Users/ladmin/Desktop/computerRename.app/Contents/MacOS/computerRename
[Password:
█
```

Enter a new name and click 'Rename'.



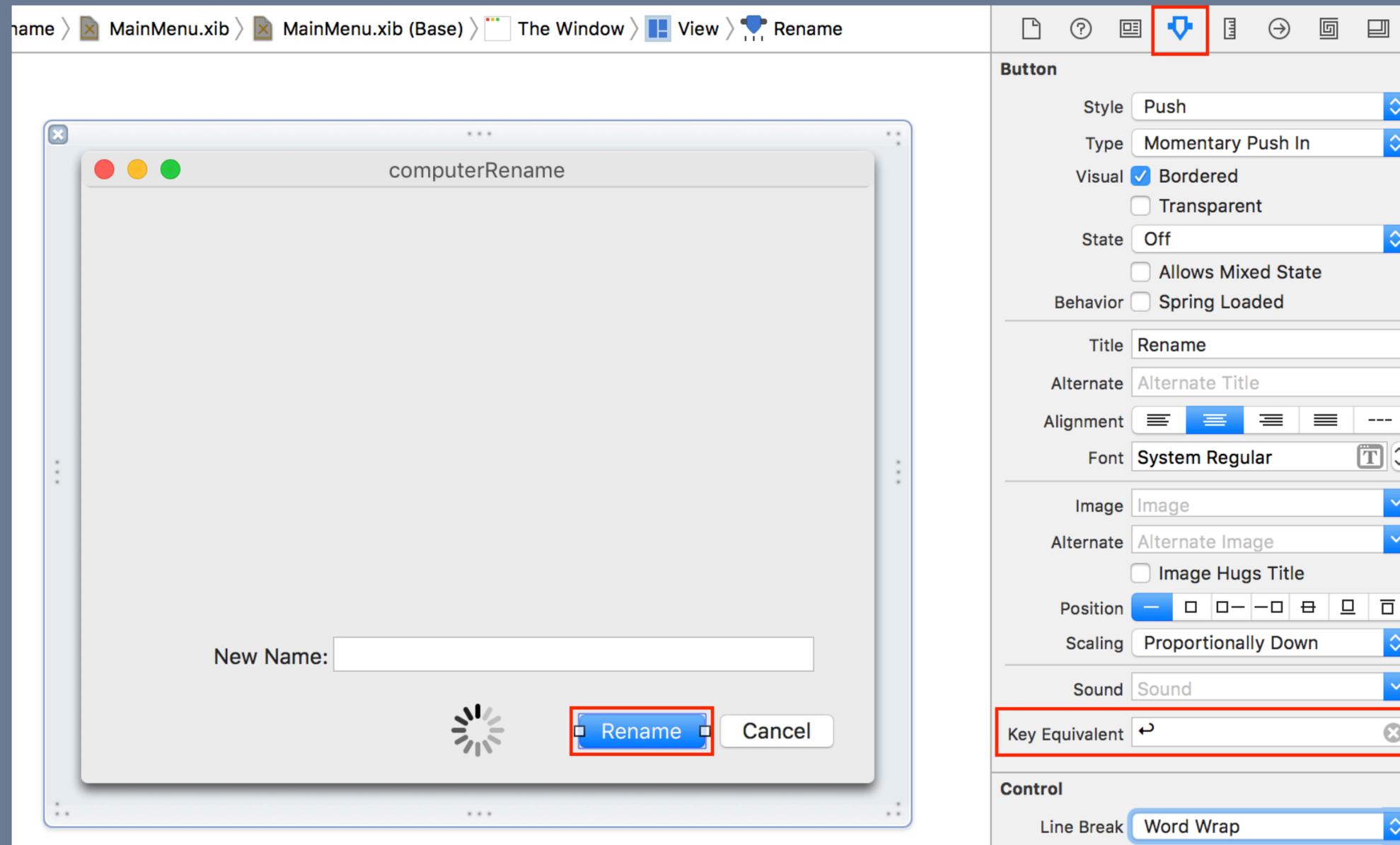
Dress up the GUI a little more, add a progress indicator (spinner).

The image shows a GUI design tool interface. On the left is a sidebar with a tree view under 'The Window' > 'View' > 'Circular Progress...'. The main canvas displays a 'computerRename' dialog box with a 'New Name:' text field, a spinner, and 'Rename' and 'Cancel' buttons. A red arrow points from the spinner in the dialog to the 'Progress Indicator' settings panel on the right. The settings panel includes:

- Progress Indicator**
 - Style: Spinning
 - Behavior: Display When Stopped
 - Indeterminate
 - Value: 0 (Minimum) to 100 (Maximum)
 - Current: 0
- View**
 - Tag: -1
 - Focus Ring: Default
 - Drawing: Hidden, Can Draw Concurrently
 - Autosizing: Autosizes Subviews

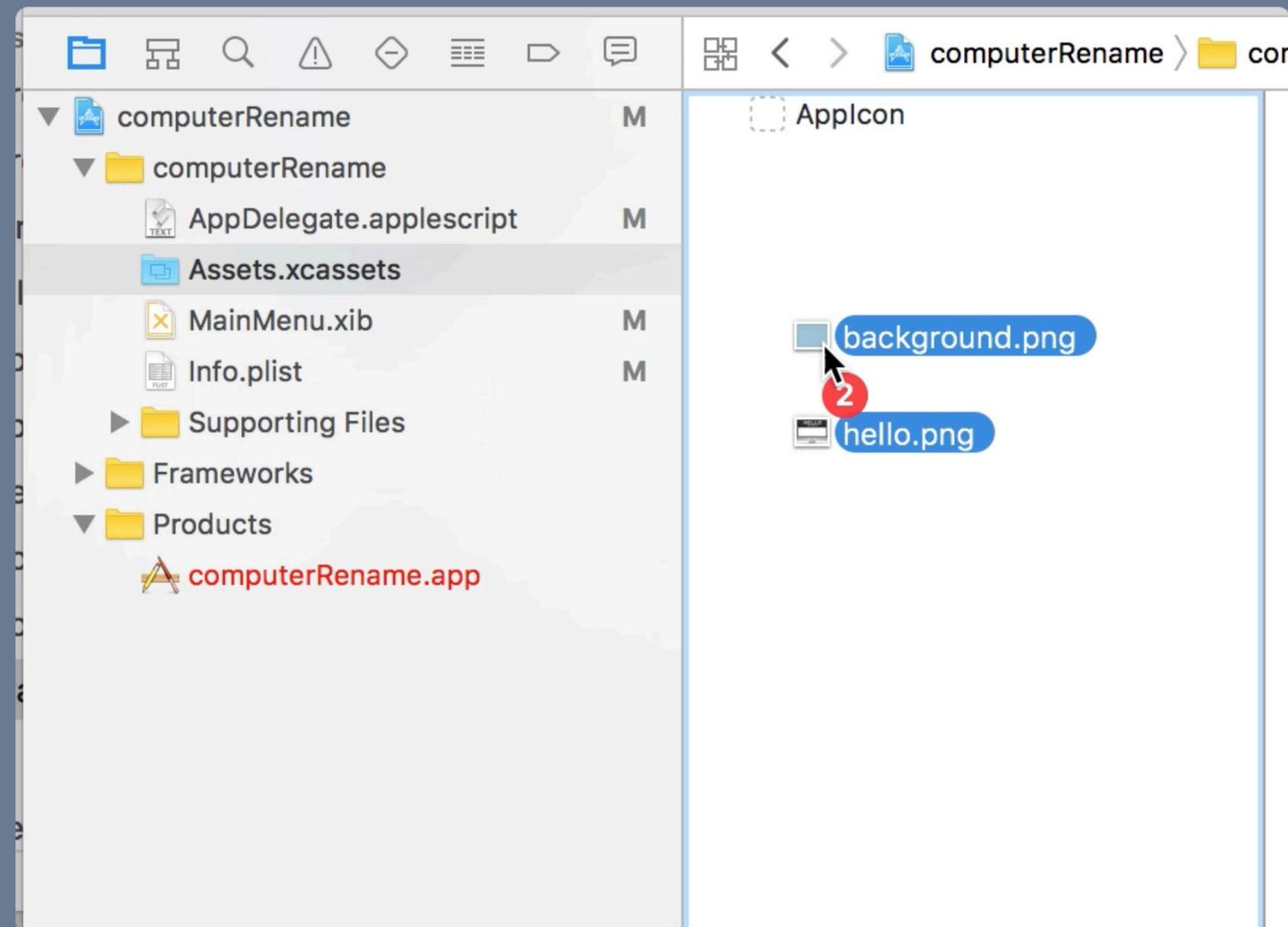
At the bottom right, a legend entry shows a spinner icon and the text: **Indeterminate Circular Progress Indicator** - Shows task progress, with continuous animation.

Let's also set the Rename button as the default.

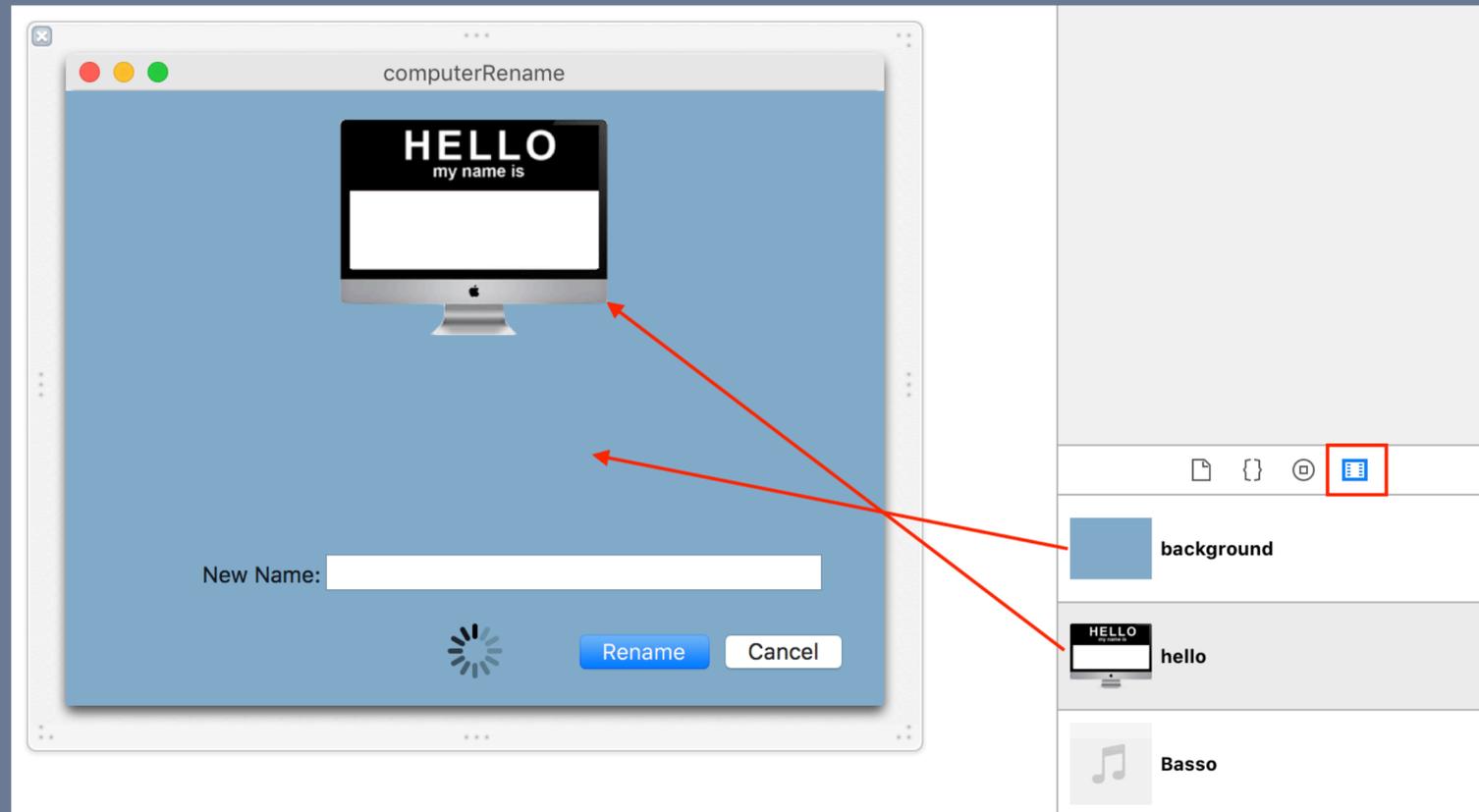


Click in the 'Key Equivalent' field and hit return.

Next we'll add some graphics to our project so that we're able to set a background along with a custom image. Click on Assets.xcassets and drag in the images, drop them just below 'AppIcon'.

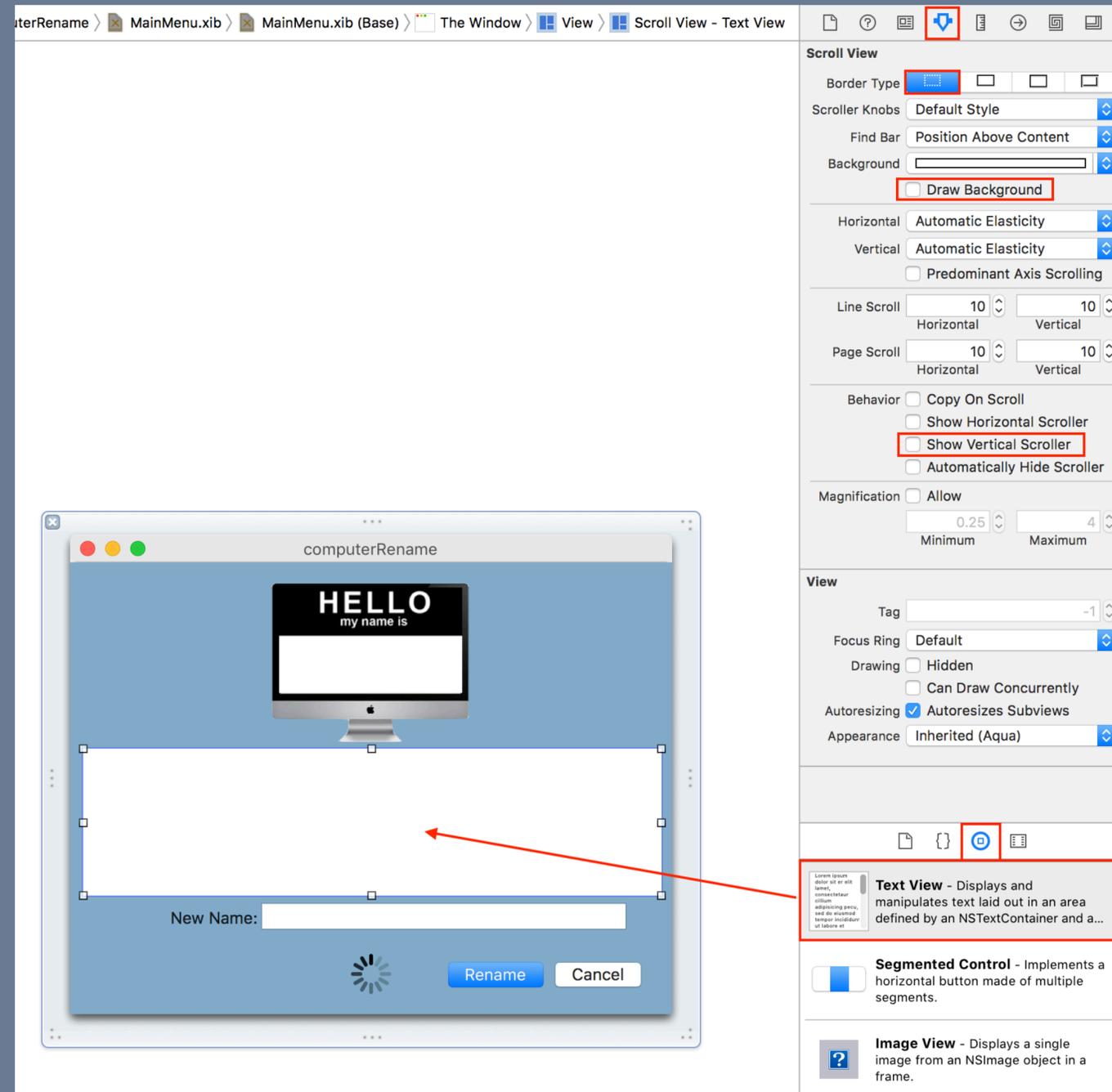


Back to the MainMenu.xib, add the images.



Click and drag the objects into the GUI. Be sure to move the background to the back: Editor —> Arrange —> Send to Back

In addition to the spinner providing evidence the application is running, let's add a text field to display recon output.



Set the text field to blend into the background.

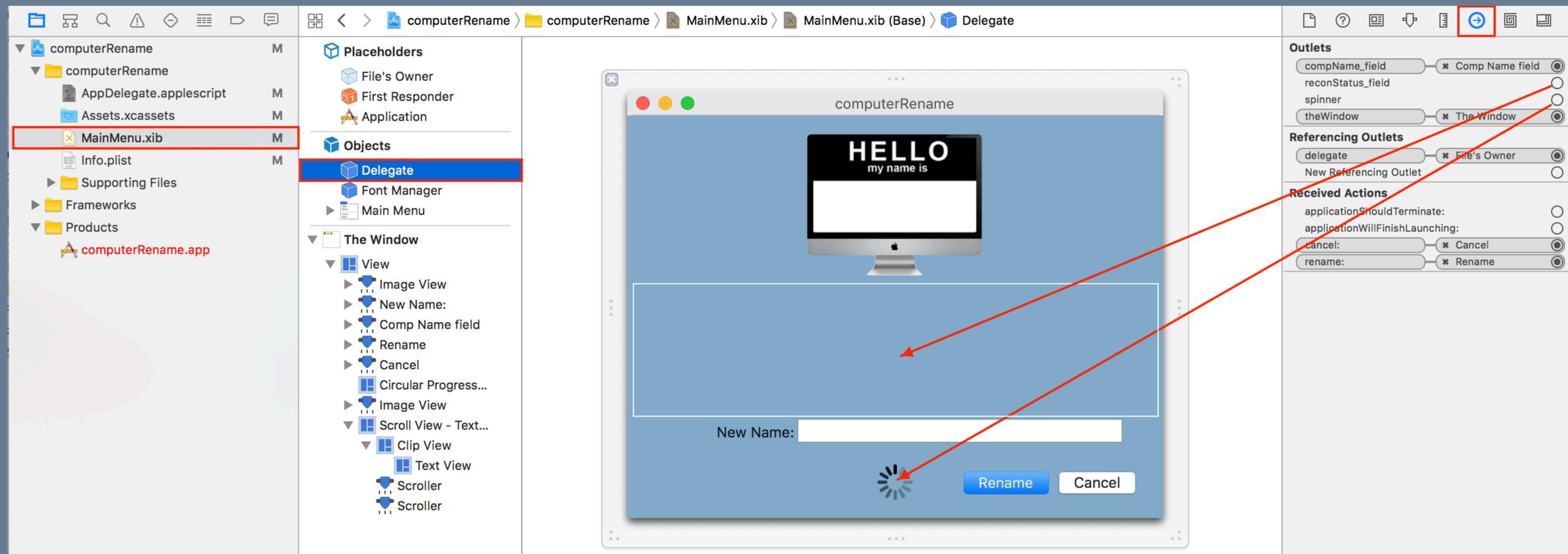
The screenshot displays the Xcode development environment. On the left, the 'The Window' sidebar shows a tree view where 'Text View' is selected and highlighted with a red box. The main canvas shows a 'Rename' dialog box with a 'HELLO my name is' computer monitor icon and a 'New Name:' text field. On the right, the 'Text View' inspector is open, with several settings highlighted by red boxes: the 'Text' field, the 'Editable' checkbox (which is unchecked), the 'Draws Background' checkbox (which is unchecked), and the 'Automatic Spelling Correcti...' checkbox (which is unchecked). The dialog box in the center has a light blue background and a white text field, demonstrating the 'blend into the background' effect.

Now, back to coding. We added a spinner and text field to display some output, let's utilize them.

```
script AppDelegate
    property parent : class "NSObject"

    -- IBOutlet
    property theWindow : missing value
    property compName_field : missing value
    property spinner : missing value
    property reconStatus_field : missing value
```

Two new properties. We need to link them to their objects in the GUI.



Update the rename_ function:

Make sure we don't have a blank name.

start the spinner

output redirection

Display output as recon is running.

stop the spinner

```
on rename_(sender)
  set newName to compName_field's stringValue()
  if newName as string is equal to "" then
    display dialog "Oops, forgot to enter a computer name."
    return
  end if

  spinner's startAnimation_(me)

  do shell script ("\
/usr/sbin/scutil --set ComputerName \"\" & newName & \"\" \
/usr/sbin/scutil --set LocalHostName \"\" & newName & \"\" \
/usr/sbin/scutil --set HostName \"\" & newName & \"\" \
\
jamfBin=$(which jamf)\
$jamfBin recon > /tmp/recon.txt 2>&1 &")

  try
    set recon_check to do shell script ("ps -ax | grep recon | grep -v grep")
    on error
      set recon_check to "complete"
    end try

    repeat while recon_check is not equal to "complete"
      set log_text to do shell script ("tail -n3 /tmp/recon.txt")
      reconStatus_field's setString_("Running Recon: " & text of log_text)
      delay 0.2
      try
        set recon_check to do shell script ("ps -ax | grep recon | grep -v \"grep\\|tail\\")
        on error
          set recon_check to "complete"
        end try
      end try
    end repeat

    reconStatus_field's setString_(text of log_text & return & "Updated.")

    spinner's stopAnimation_(me)

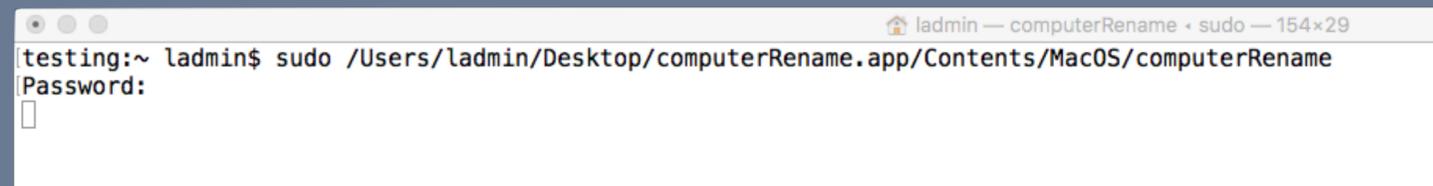
    display dialog "Rename complete!" buttons {"OK"}
    quit
  end rename
```

Lastly, cleanup the code we're not using in `applicationWillFinishLaunching_` and add a function to bring the application to the foreground when launched.

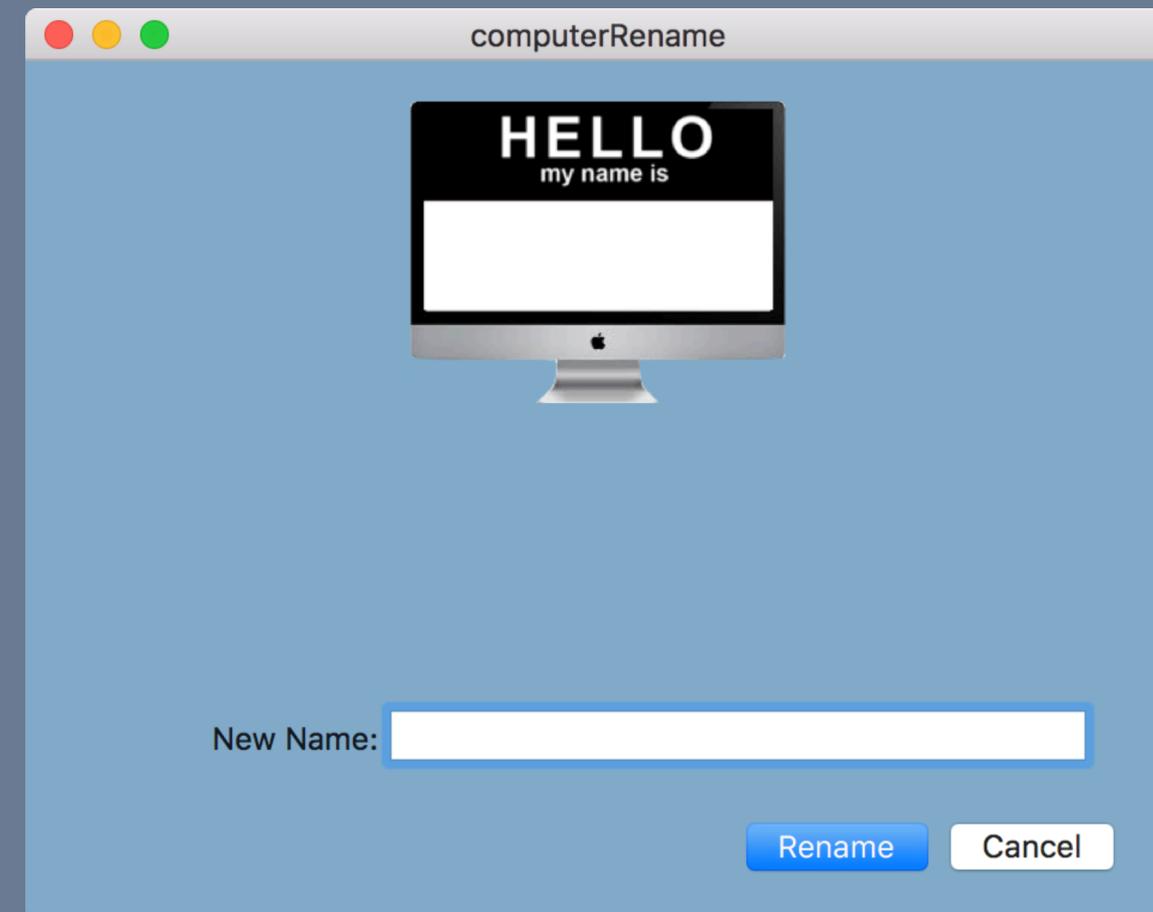
```
on applicationWillFinishLaunching_(aNotification)
    -- Insert code here to initialize your application before any files are opened
end applicationWillFinishLaunching_

on applicationDidFinishLaunching_(sender)
    activate me
    theWindow's makeFirstResponder_(compName_field)
end applicationDidFinishLaunching_
```

Build (command+b) then launch the application through Terminal.



```
ladmin — computerRename • sudo — 154x29
testing:~ ladmin$ sudo /Users/ladmin/Desktop/computerRename.app/Contents/MacOS/computerRename
Password:
█
```



Goal 3: Properly sign the application.

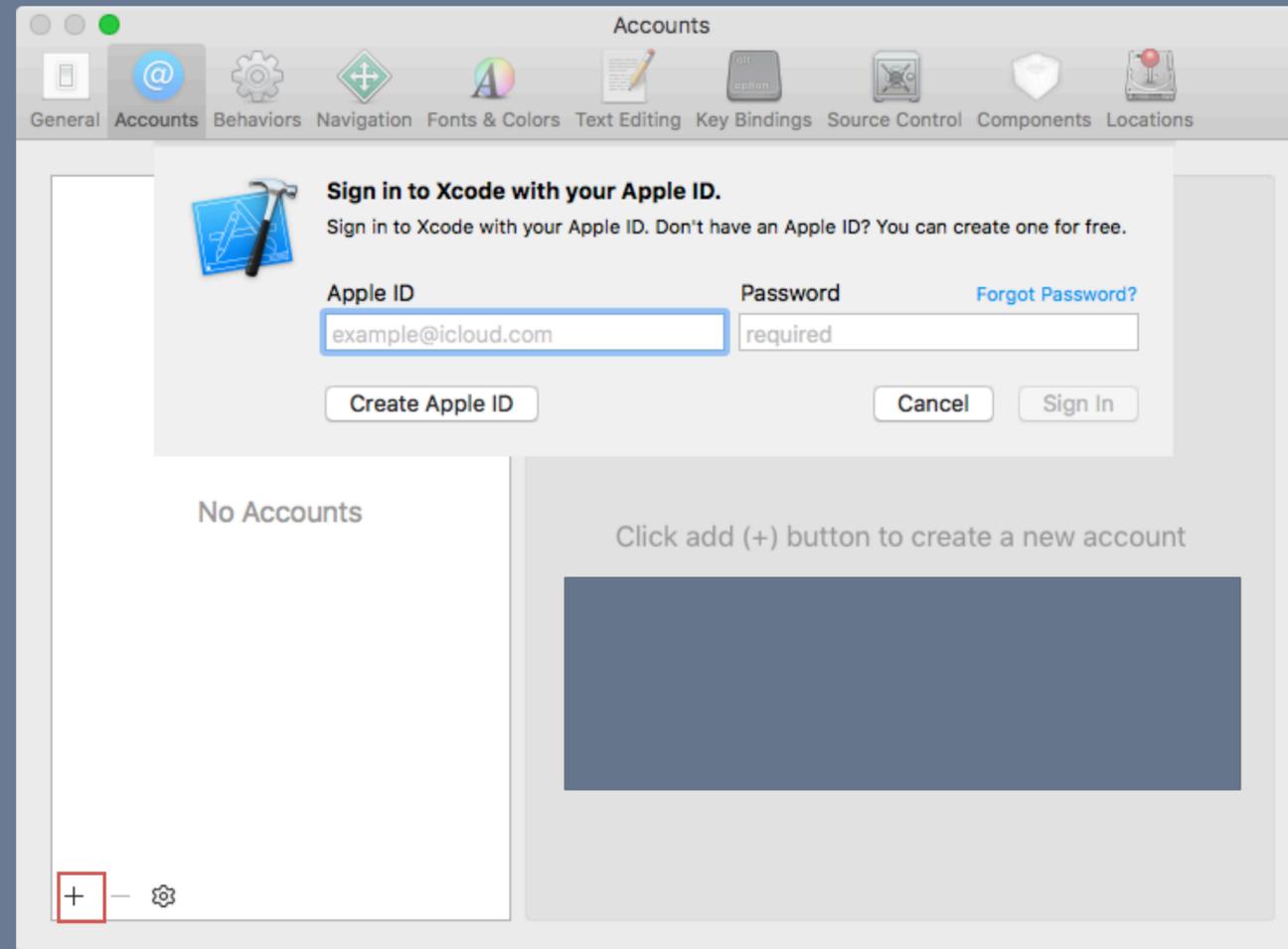
We can check the application signature current state using `codesign -dv` in Terminal.



```
ladmin — -bash — 154x29
[testing:~ ladmin$ codesign -dv /Users/ladmin/Desktop/computerRename.app
Executable=/Users/ladmin/Desktop/computerRename.app/Contents/MacOS/computerRename
Identifier=com.pse.jamf.computerRename
Format=app bundle with Mach-O thin (x86_64)
CodeDirectory v=20100 size=524 flags=0x2(adhoc) hashes=11+3 location=embedded
Signature=adhoc
Info.plist entries=22
TeamIdentifier=not set
Sealed Resources version=2 rules=13 files=3
Internal requirements count=0 size=12
testing:~ ladmin$
```

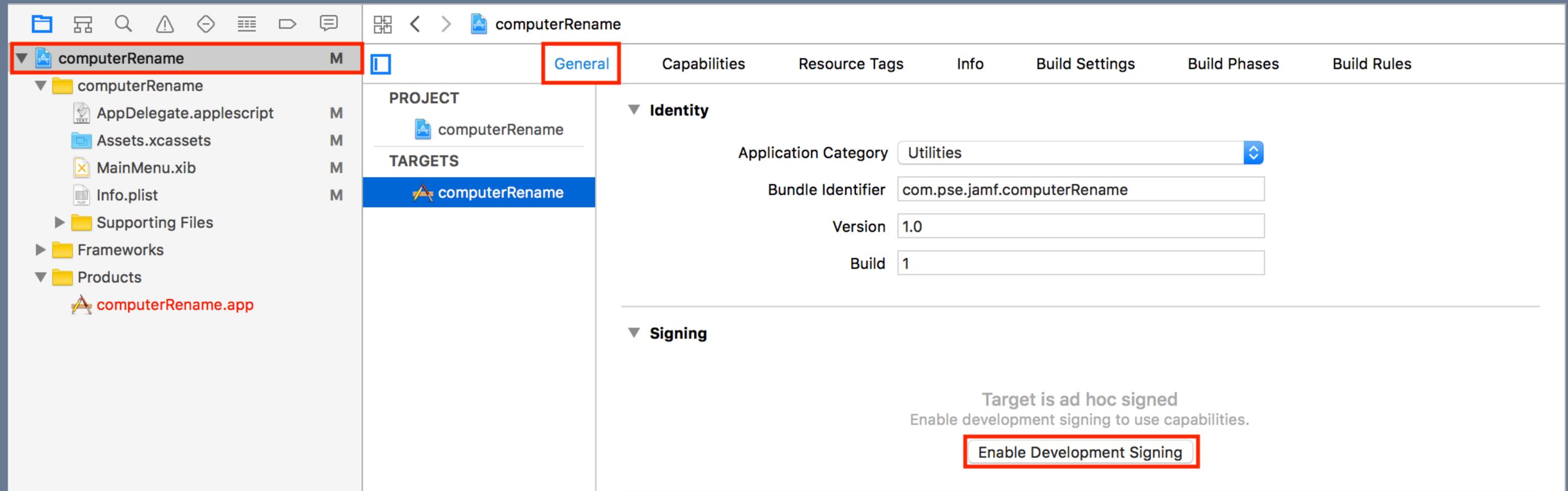
Signature=adhoc tells us the application is not signed by a known developer.

If you haven't already, add your (paid) developer account to Xcode. From the menu bar, Xcode -> Preferences -> Accounts, then click the + sign.



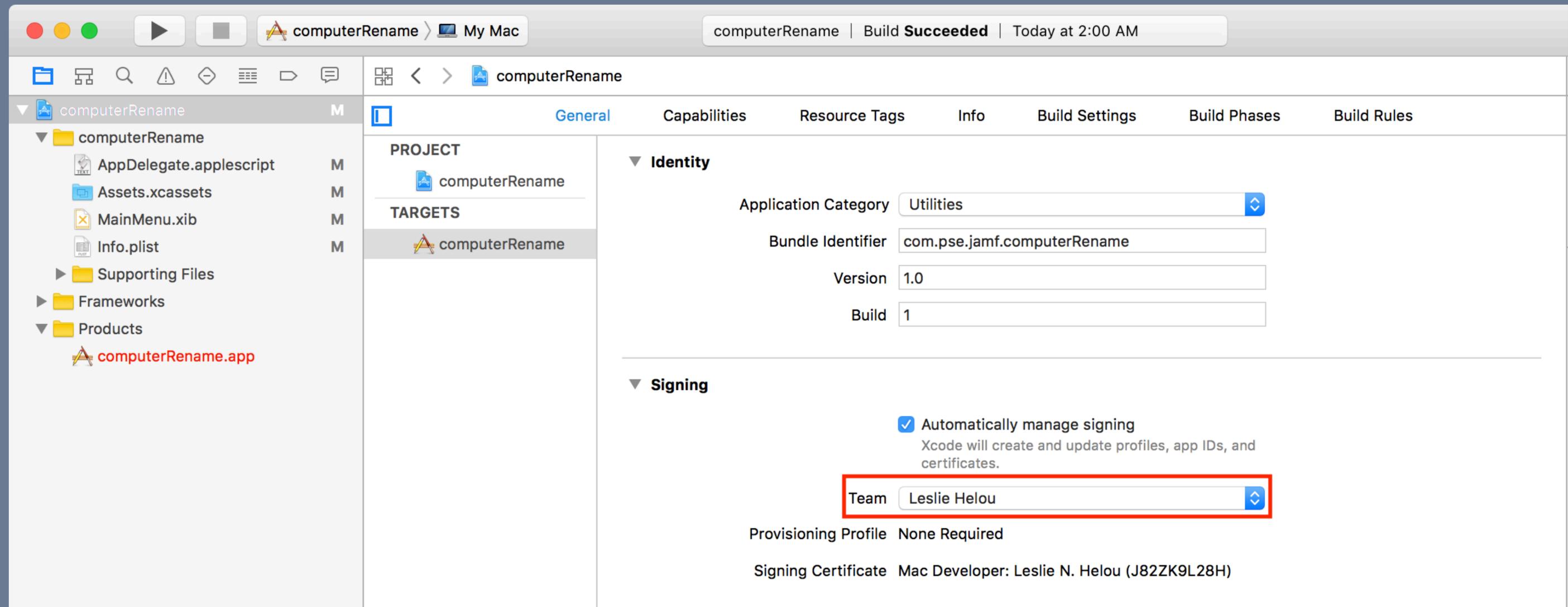
Enter the appropriate credentials and click 'Sign In'. Close the preferences window.

Back to the main window, click the top level of the project and make sure we're on the General tab.



Click 'Enable Development Signing'.

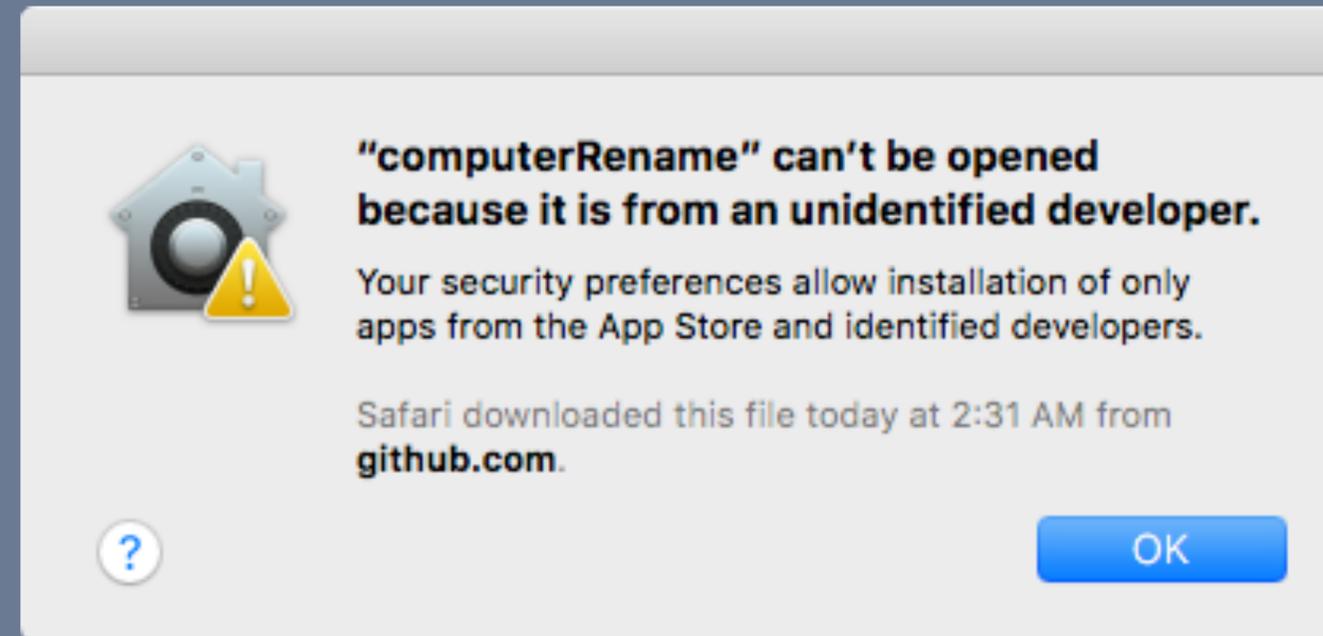
If necessary select a team from the drop-down.



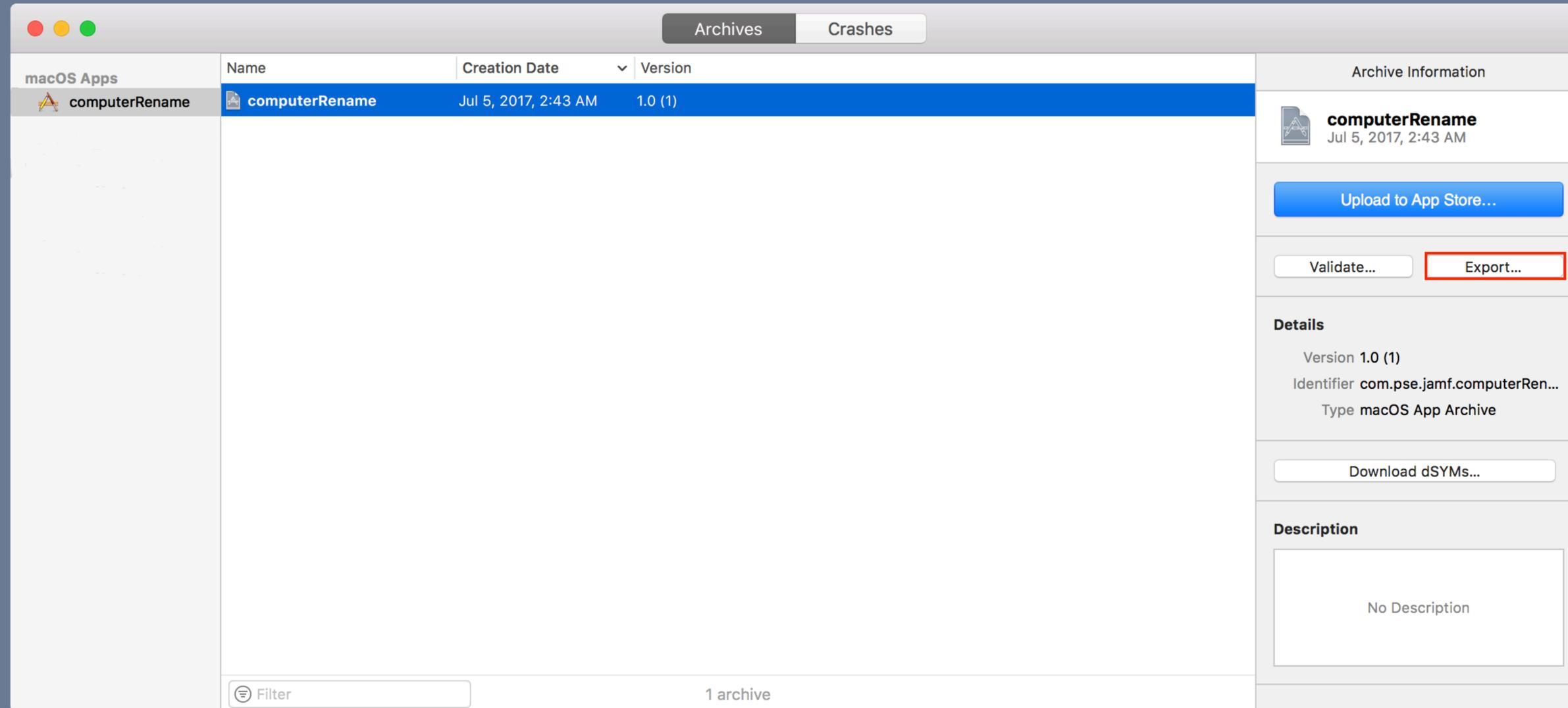
Build and check the application to see if it is signed. Use `codesign -dvv` to get a little extra info.

```
ladmin — -bash — 154x29
[testing:~ ladmin$ codesign -dvv /Users/ladmin/Desktop/computerRename.app
Executable=/Users/ladmin/Desktop/computerRename.app/Contents/MacOS/computerRename
Identifier=com.pse.jamf.computerRename
Format=app bundle with Mach-O thin (x86_64)
CodeDirectory v=20200 size=539 flags=0x0(none) hashes=11+3 location=embedded
Signature size=4686
Authority=Mac Developer: Leslie N. Helou (J82ZK9L28H)
Authority=Apple Worldwide Developer Relations Certification Authority
Authority=Apple Root CA
Signed Time=Jul 5, 2017, 2:00:12 AM
Info.plist entries=22
TeamIdentifier=PS2F6S478M
Sealed Resources version=2 rules=13 files=3
Internal requirements count=1 size=188
testing:~ ladmin$ █
```

Now even though the application is signed, it is not ready to be shared, say through GitHub. If the application is downloaded in its current state, we'll likely see the following when it is opened.



As the final task, we need to 'Archive' the project. From the menu bar, Product -> Archive.



Click Export...

Select Export a Developer ID-signed Application

Select a method for export:

- Save for Mac App Store Deployment
Sign and package application for distribution in the Mac App Store.
- Export a Developer ID-signed Application
Save a copy of the application signed with your Developer ID.
- Export a Development-signed Application
Save a copy of the application signed with your Development identity.
- Export as a macOS App
Export a copy of the application without re-signing.

Cancel Previous Next

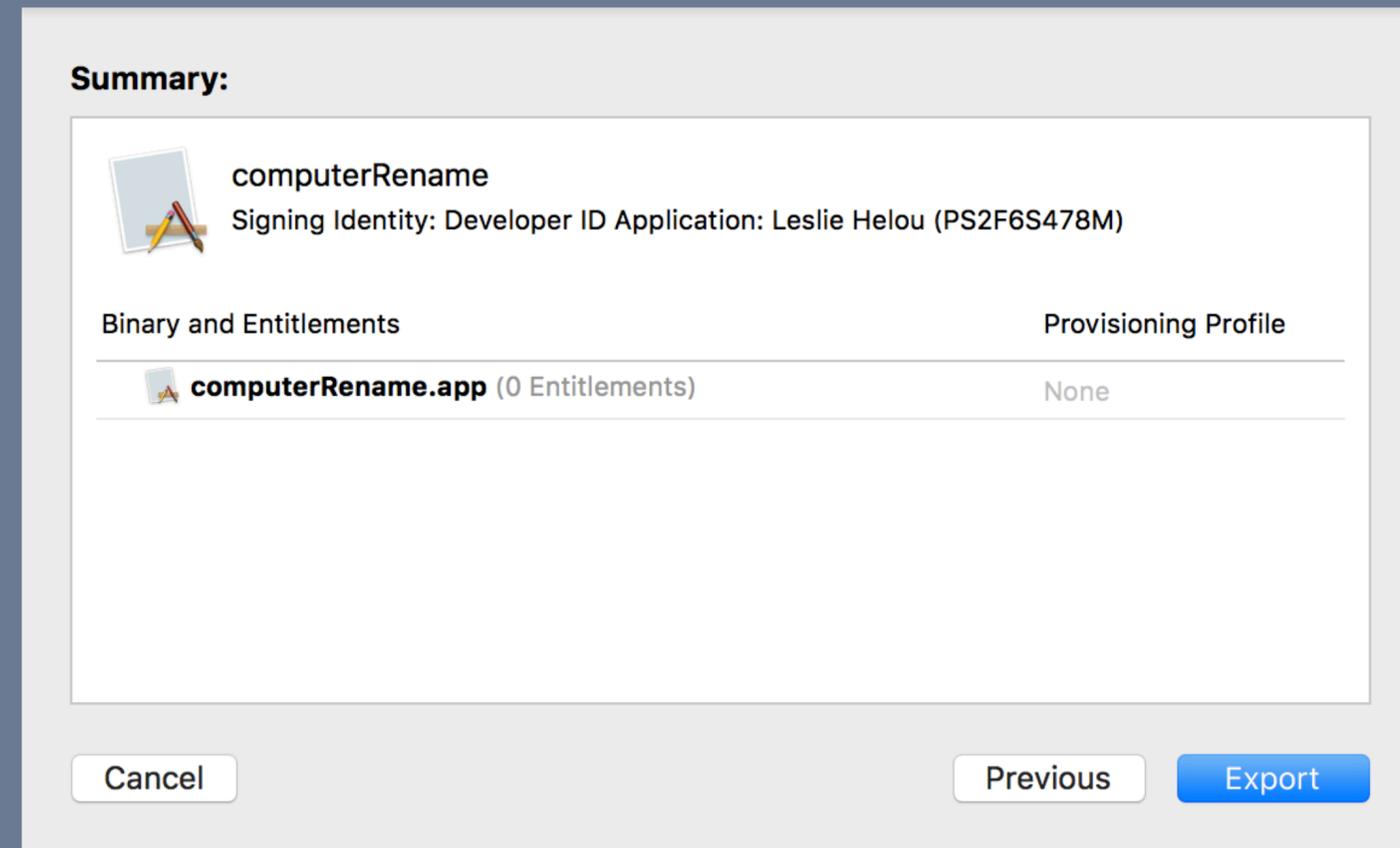
Click Next, ensure we have the right developer account and click Next again.

To export a Developer ID-signed Application, select a Development Team to use for provisioning:

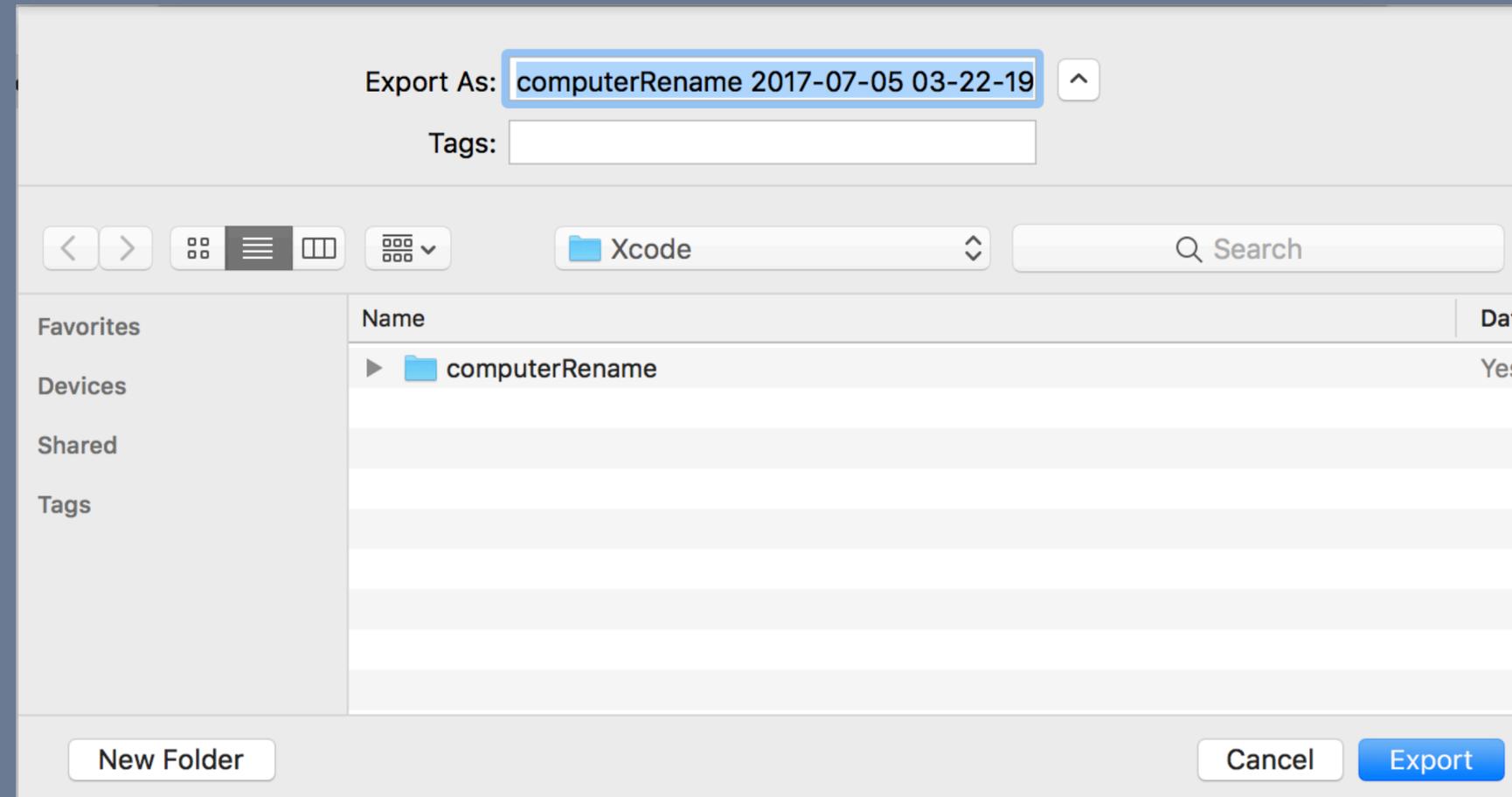
Leslie Helou

View Accounts... Cancel Choose

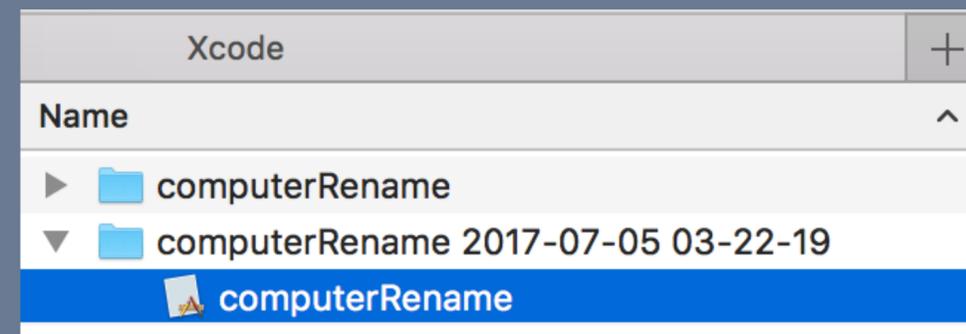
There will be some processing then you'll be presented with the application to export.



Select a name and location for the application and click Export.



The application is now ready to be shared.

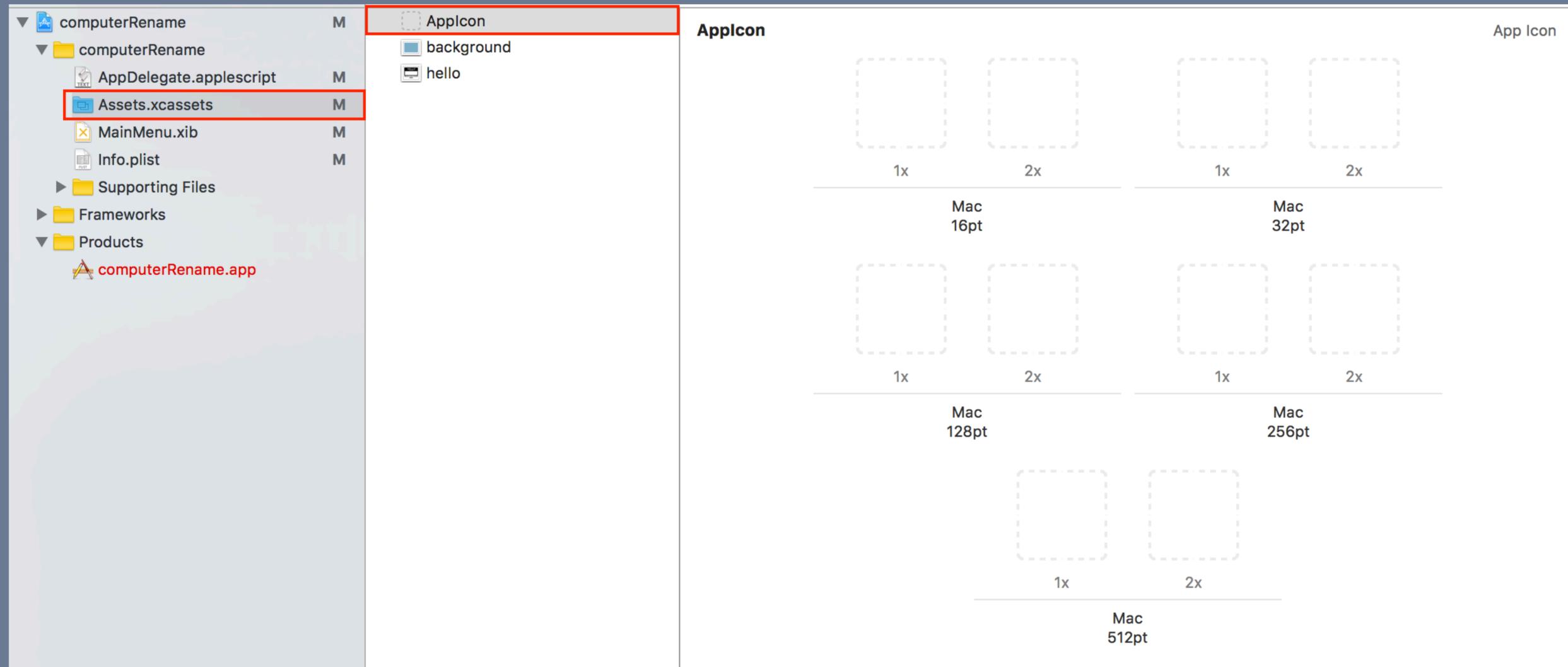


Thank you!

Extra time, extra credit...

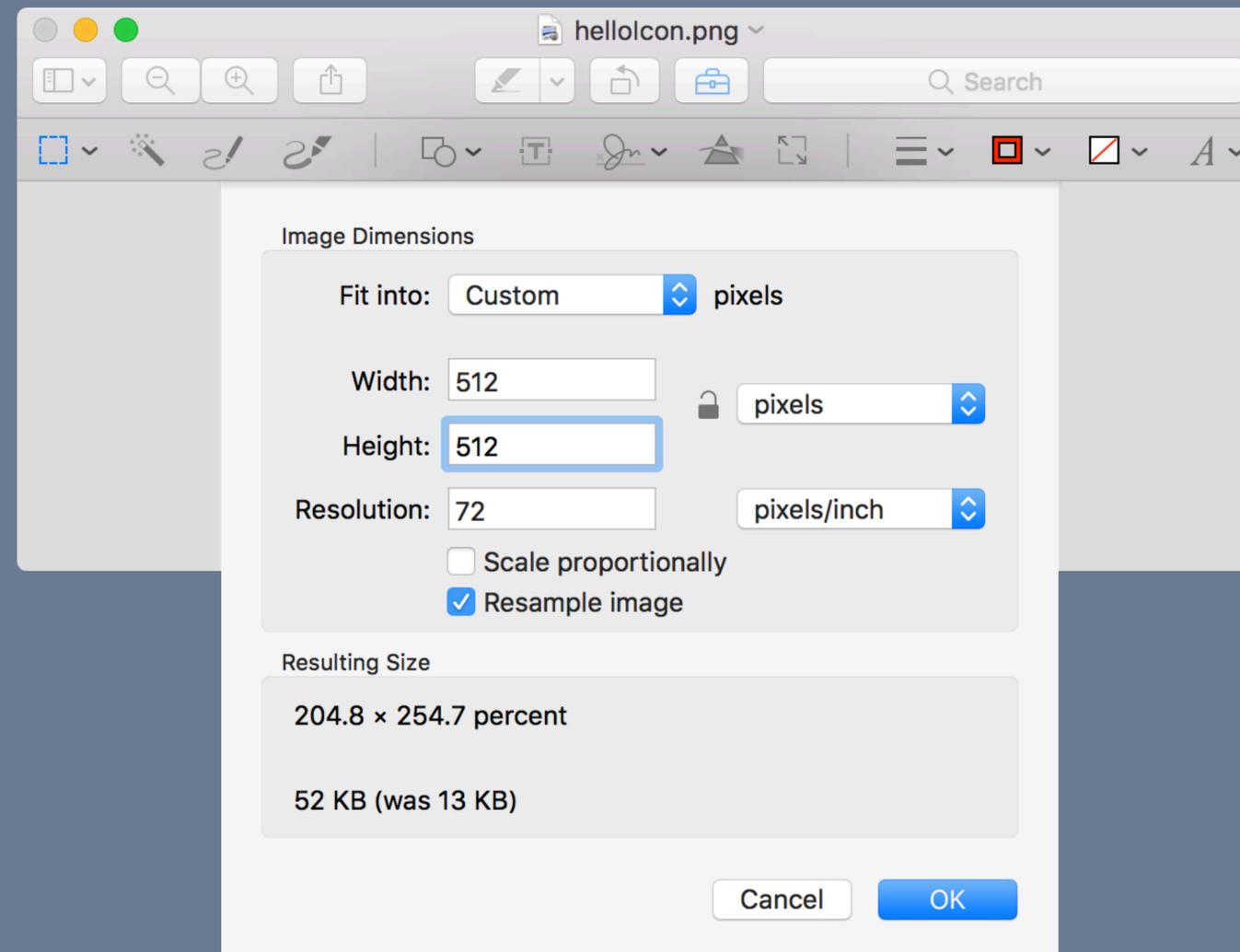
- Add an icon for the application.
- Enable the application to work when launched from the Finder.
- Tweak the application window.

Looking at the options for icons for the application we see several sizes are available.



Start with the hello.png from the application. Open with Preview and resize the image to 512x512 pixels.

- Make a copy of hello.png, call it something like hellolcon.png
- Open hellolcon.png with Preview, click Tools -> Adjust Size...
- Click the lock so that it is unlocked and select pixels from the drop down next to the lock
- Set the width and height to 512



We'll need several other sizes, but rather than creating them manually let's use a script.

<https://github.com/BIG-RAT/MacAdmins2017/blob/master/createlconSet.sh.zip>

```
#!/bin/bash

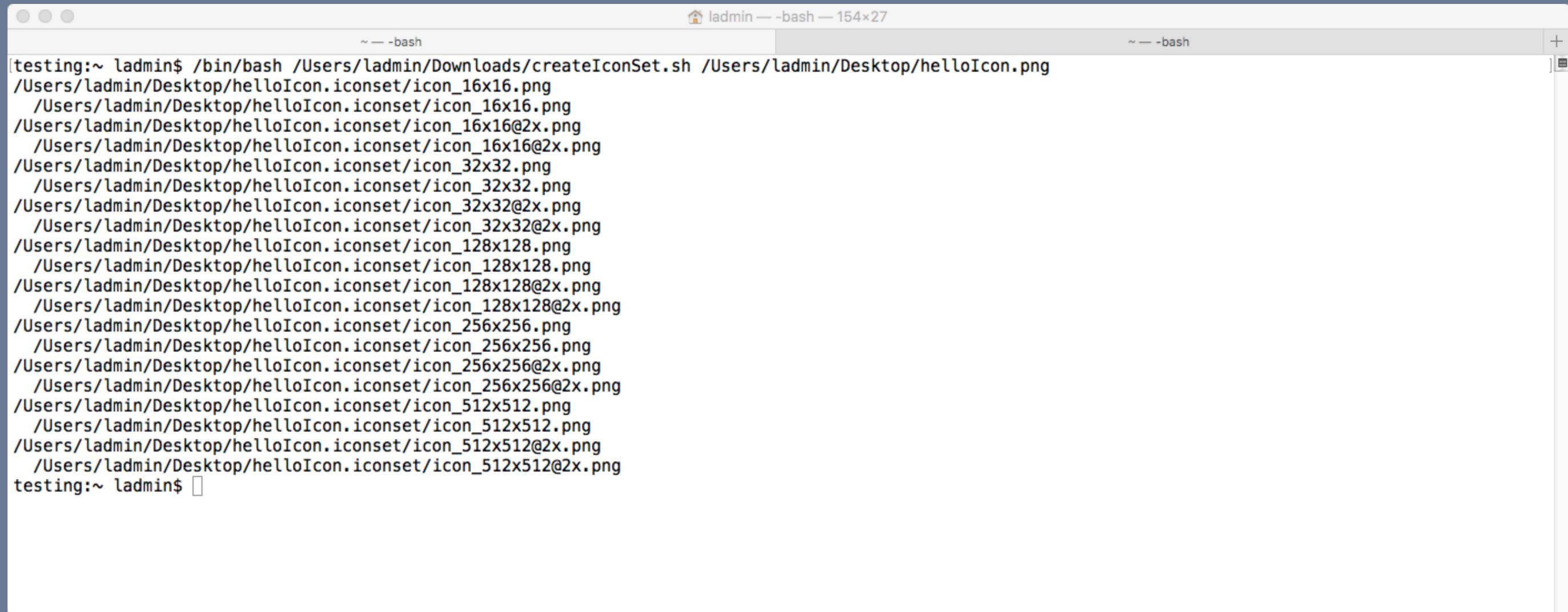
## used to create square png files from single png file
## helpful when creating icons for Xcode apps

theImage="$1"
fileName=$(basename "$1" | sed -e 's/.png//')
username=$(stat -f%Su /dev/console)
mkdir "/Users/$username/Desktop/$fileName.iconset"

for i in 16 32 128 256 512; do
    newName="icon_"$i"x"$i".png"
    retinaName="icon_"$i"x"$i"@2x.png"
    /bin/cp "$1" "/Users/$username/Desktop/$fileName.iconset/$newName"
    /bin/cp "$1" "/Users/$username/Desktop/$fileName.iconset/$retinaName"
    /usr/bin/sips "/Users/$username/Desktop/$fileName.iconset/$newName" -z $i $i
    /usr/bin/sips "/Users/$username/Desktop/$fileName.iconset/$retinaName" -z $(( i*2 )) $(( i*2 ))
done

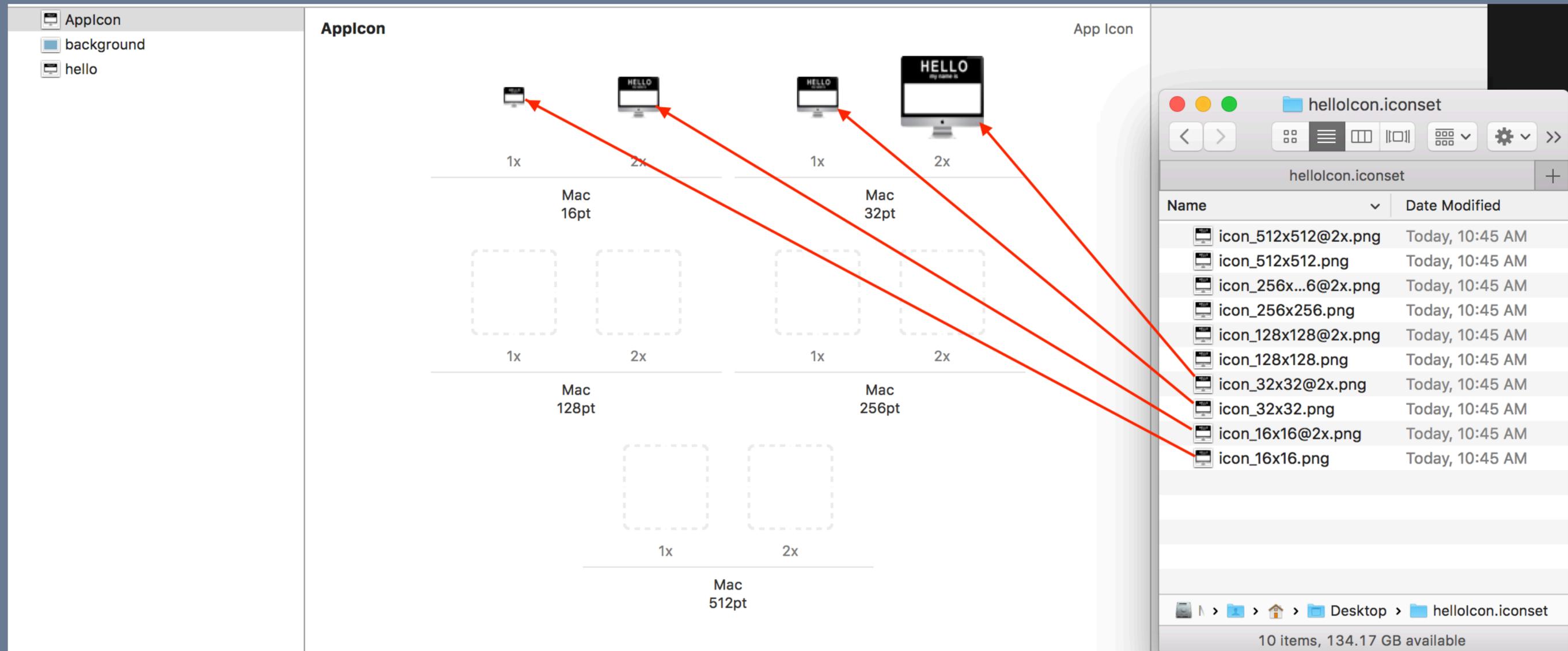
/usr/bin/iconutil -c icns "/Users/$username/Desktop/$fileName.iconset"
```

Run the script in Terminal, passing the helloIcon.png as a parameter.



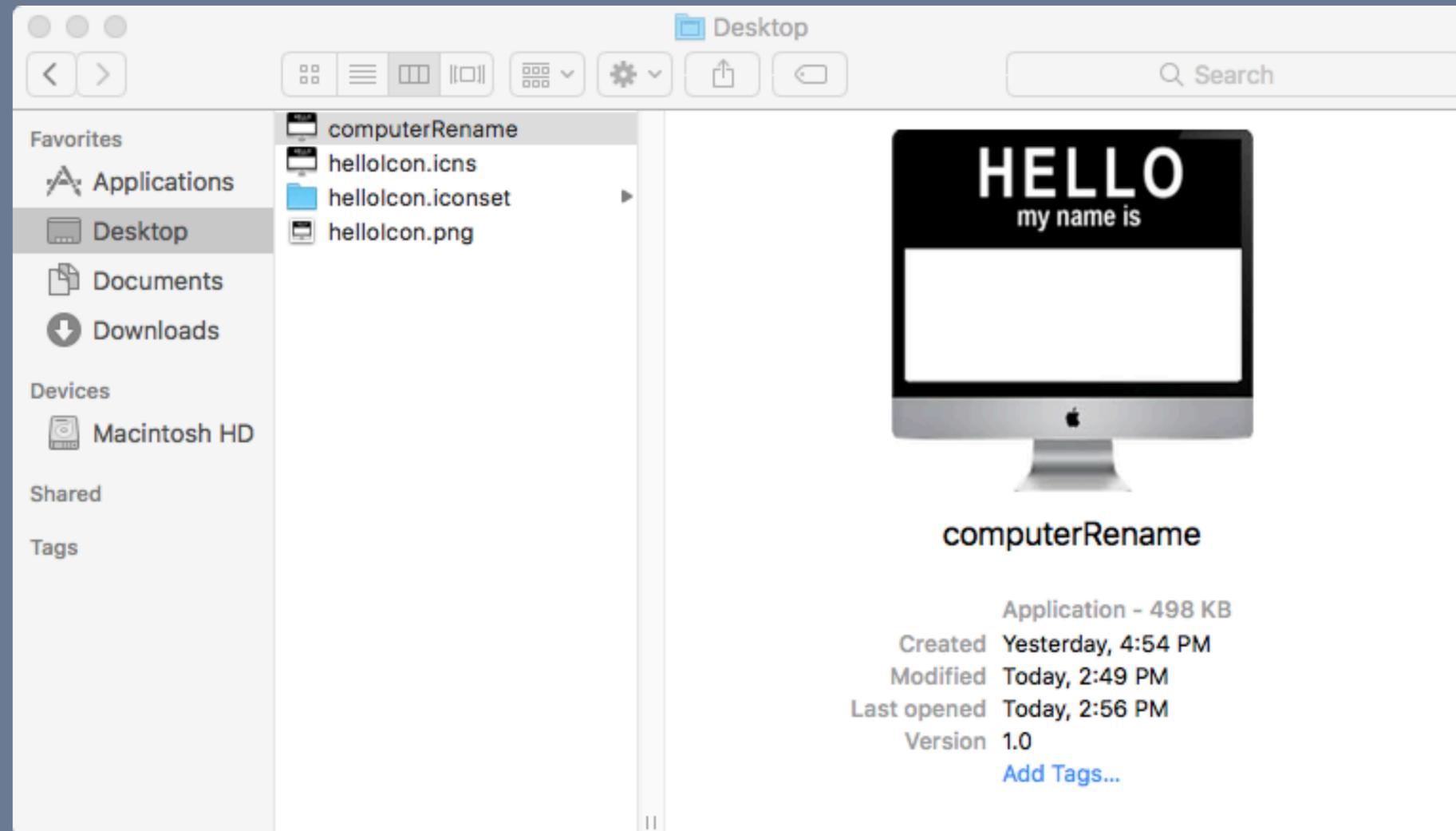
```
ladmin — -bash — 154x27
~ — -bash
[testing:~ ladmin$ /bin/bash /Users/ladmin/Downloads/createIconSet.sh /Users/ladmin/Desktop/helloIcon.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_16x16.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_16x16.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_16x16@2x.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_16x16@2x.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_32x32.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_32x32.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_32x32@2x.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_32x32@2x.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_128x128.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_128x128.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_128x128@2x.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_128x128@2x.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_256x256.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_256x256.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_256x256@2x.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_256x256@2x.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_512x512.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_512x512.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_512x512@2x.png
/Users/ladmin/Desktop/helloIcon.iconset/icon_512x512@2x.png
testing:~ ladmin$ ]
```

From the folder created on the Desktop, drag the images to the appropriate location in the project.



Once all the images have been placed rebuild the application.

Locate the application in the Finder, the new icon should be displayed.



Time now to make the application function properly if double-clicked.

Current Code:

```
spinner's startAnimation_(me)

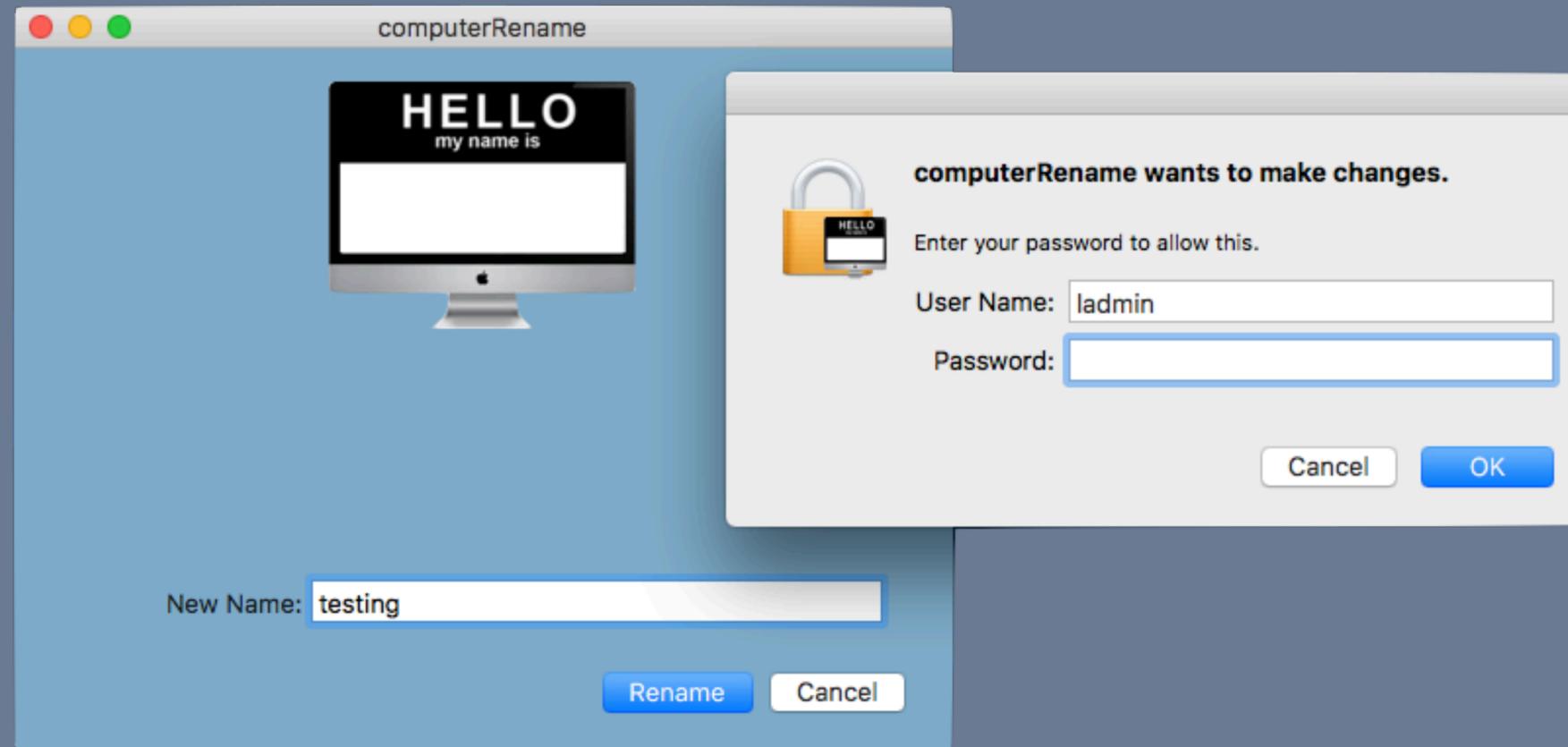
do shell script ("
/usr/sbin/scutil --set ComputerName \"\" & newName & "\"
/usr/sbin/scutil --set LocalHostName \"\" & newName & "\"
/usr/sbin/scutil --set HostName \"\" & newName & "\"
\
jamfBin=$(which jamf)\
$jamfBin recon > /tmp/recon.txt 2>&1 &")
```

Updated Code:

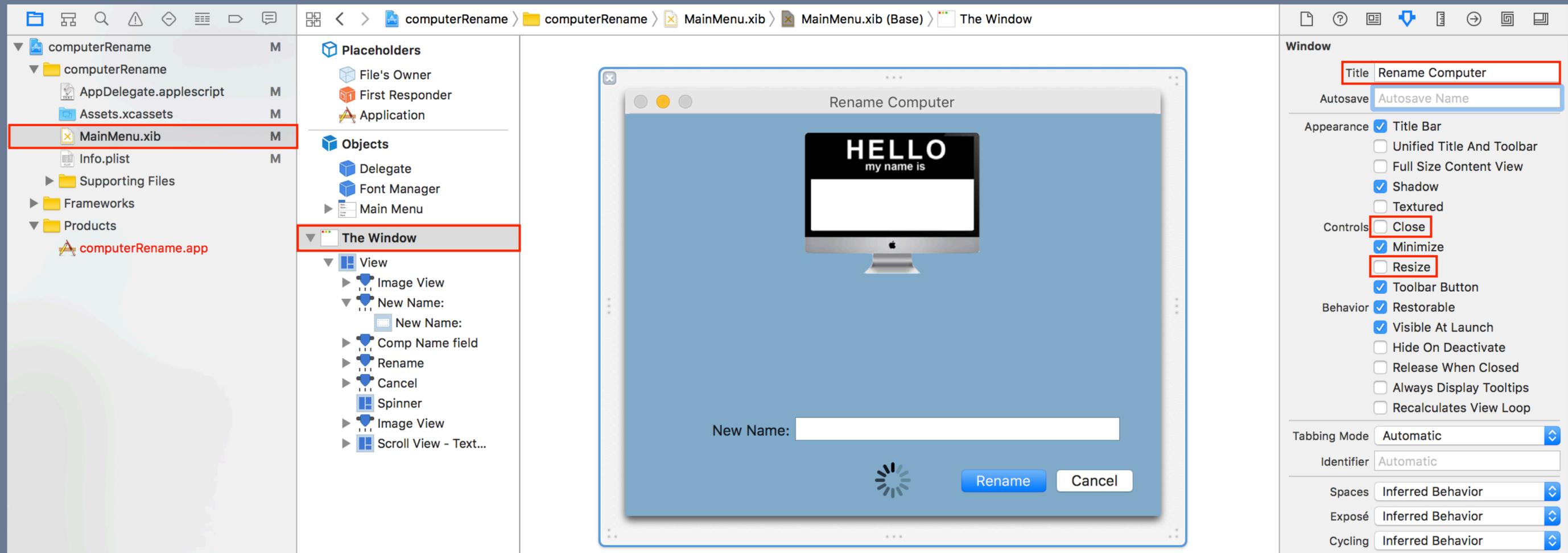
```
do shell script ("
/usr/sbin/scutil --set ComputerName \"\" & newName & "\"
/usr/sbin/scutil --set LocalHostName \"\" & newName & "\"
/usr/sbin/scutil --set HostName \"\" & newName & "\"
\
/usr/local/bin/jamf recon > /tmp/recon.txt 2>&1 &") with administrator privileges

spinner's startAnimation_(me)
```

Let's build the application and launch it from the Finder.



Minor adjustments to the application window.



One more time, build the application. How does it look? Can we resize the window?

Thank you!
(again)

Feedback: <https://bit.ly/psumac2017-141>