

# Package ‘polished’

September 29, 2020

**Type** Package

**Title** Authentication and Administration for 'shiny' Apps

**Version** 0.2.0

**Maintainer** Andy Merlino <andy.merlino@tychobra.com>

**Description** Easily add modern authentication and user administration to your 'shiny' apps. Customize user sign in and registration pages to match your brand. Control who can access one or more of your 'shiny' apps.

**License** MIT + file LICENSE

**URL** <https://github.com/tychobra/polished>, <https://polished.tech>

**BugReports** <https://github.com/tychobra/polished/issues>

**Encoding** UTF-8

**LazyData** true

**Imports** digest, dplyr, DT, htmltools, httr, jose, jsonlite, lubridate, purrr, R6, rlang, shiny, shinycssloaders, shinydashboard, shinydashboardPlus, shinyFeedback, shinyjs, shinyWidgets, stringr, tibble, tidyr, uuid

**RoxygenNote** 7.1.1

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Andy Merlino [aut, cre],  
Patrick Howard [aut]

**Repository** CRAN

**Date/Publication** 2020-09-29 06:50:02 UTC

## R topics documented:

default_admin_ui_options . . . . .	2
email_input . . . . .	3

firebase_dependencies . . . . .	3
firebase_init . . . . .	4
global_sessions_config . . . . .	5
password_input . . . . .	6
profile_module . . . . .	7
profile_module_ui . . . . .	7
providers_ui . . . . .	8
remove_query_string . . . . .	8
secure_server . . . . .	9
secure_static . . . . .	10
secure_ui . . . . .	11
send_password_reset_email_module . . . . .	12
send_password_reset_email_module_ui . . . . .	12
Sessions . . . . .	13
set_config_env . . . . .	15
sign_in_check_jwt . . . . .	15
sign_in_js . . . . .	16
sign_in_module . . . . .	16
sign_in_module_2 . . . . .	17
sign_in_module_2_ui . . . . .	17
sign_in_module_ui . . . . .	18
sign_in_ui_default . . . . .	18
sign_out_from_shiny . . . . .	19

<b>Index</b>	<b>20</b>
--------------	-----------

---

default\_admin\_ui\_options

*Default Options for the Admin UI*

---

## Description

This function specifies the default logos that are displayed in the "Admin Panel".

## Usage

```
default_admin_ui_options()
```

## Value

a list of html for branding the Admin UI

---

email_input	<i>A 'shiny' email input</i>
-------------	------------------------------

---

**Description**

This is a replica of `shiny::textInput()` with the html input "type" attribute set to "email" rather than "text".

**Usage**

```
email_input(
  inputId,
  label = tagList(icon("envelope"), "Email"),
  value = "",
  width = NULL,
  placeholder = NULL
)
```

**Arguments**

inputId	The input slot that will be used to access the value.
label	Display label for the control, or NULL for no label.
value	Initial value.
width	The width of the input, e.g. '400px'.
placeholder	A character string giving the user a hint as to what can be entered into the control. Internet Explorer 8 and 9 do not support this option.

---

firebase_dependencies	<i>Load the Firebase JavaScript dependencies into the UI</i>
-----------------------	--

---

**Description**

Under the hood, 'polished' uses Firebase JavaScript dependencies to handle user authentication. This function loads the required Firebase JavaScript dependencies in the the UI of your 'shiny' app.

**Usage**

```
firebase_dependencies(services = c("auth"), firebase_version = "7.15.5")
```

**Arguments**

services	character vector of Firebase services to load into the UI. Valid strings are "auth", "firestore", "functions", "messaging", and "storage"
firebase_version	character string of the Firebase version. Defaults to 7.15.5.

**Value**

the html `<script>` tags for the Firebase JavaScript dependencies

**Examples**

```
firebase_dependencies()
```

---

firebase_init	<i>Initialize Firebase</i>
---------------	----------------------------

---

**Description**

Executes a couple lines of JavaScript to initialize Firebase. This function should be called in your 'shiny' UI immediately after `firebase_dependencies`.

**Usage**

```
firebase_init(firebase_config)
```

**Arguments**

```
firebase_config  
list of firebase configuration
```

**Value**

a character string of JavaScript code to initialize Firebase

**Examples**

```
## Not run:  
my_config <- list(  
  apiKey = "your Firebase API key",  
  authDomain = "your Firebase auth domain",  
  projectId = "your Firebase Project ID"  
)  
  
firebase_init(my_config)  
  
## End(Not run)
```

---

`global_sessions_config`*Configuration for global sessions*

---

## Description

This is the primary function for configuring 'polished'. It configures your app's instance of the Sessions class that manages your user's 'polished' sessions. Call this function in your "global.R" file. See [https://github.com/Tychobra/polished/blob/master/inst/examples/polished\\_example\\_01/global.R](https://github.com/Tychobra/polished/blob/master/inst/examples/polished_example_01/global.R) for a complete example.

## Usage

```
global_sessions_config(  
  app_name,  
  api_key,  
  firebase_config = NULL,  
  admin_mode = FALSE,  
  is_invite_required = TRUE,  
  api_url = "https://api.polished.tech",  
  sign_in_providers = "email",  
  is_email_verification_required = TRUE  
)
```

## Arguments

<code>app_name</code>	the name of the app.
<code>api_key</code>	the API key. Either from polished.tech or your on premise polished API deployment.
<code>firebase_config</code>	a list containing your Firebase project configuration. This list should have the following named elements: <ul style="list-style-type: none"><li>• <code>apiKey</code></li><li>• <code>authDomain</code></li><li>• <code>projectId</code></li></ul>
<code>admin_mode</code>	FALSE by default. Set to TRUE to enter the polished Admin Panel without needing to register and sign in. This is useful during development for inviting the first users to your app. Make sure to set 'admin_mode' to FALSE before deploying your app.
<code>is_invite_required</code>	TRUE by default. Whether or not to require the user to have an invite before registering/signing in
<code>api_url</code>	the API url. Defaults to "https://api.polished.tech".
<code>sign_in_providers</code>	the sign in providers to enable. Valid values are "google" "email", "microsoft", and/or "facebook". Defaults to "email".

```
is_email_verification_required
    TRUE by default. Whether or not to require the user to verify their email before
    accessing your Shiny app.
```

### Examples

```
## Not run:
# global.R

global_sessions_config(
  app_name = "<your app name>",
  api_key = "<your API key>"
)

## End(Not run)
```

---

password\_input      *A modification of 'shiny::passwordInput'*

---

### Description

This modified version of Shiny's passwordInput() does not actual send the password to our Shiny server. It is just a regular password input that always keeps your user's password on the client. The password is used to sign the user in and then converted to a JWT by Firebase, all on the client, before it is sent to your Shiny server.

### Usage

```
password_input(
  input_id,
  label = tagList(icon("unlock-alt"), "Password"),
  value = "",
  style = "",
  placeholder = NULL
)
```

### Arguments

input_id	The input slot that will be used to access the value.
label	Display label for the control, or NULL for no label.
value	Initial value.
style	Character string of in-line css to style the input.
placeholder	A character string giving the user a hint as to what can be entered into the control. Internet Explorer 8 and 9 do not support this option.

---

profile_module	<i>Profile Module Server</i>
----------------	------------------------------

---

**Description**

The server logic to accompany the [profile\\_module\\_ui](#).

**Usage**

```
profile_module(input, output, session)
```

**Arguments**

input	the Shiny server input
output	the Shiny server output
session	the Shiny server session

---

profile_module_ui	<i>Profile Module UI</i>
-------------------	--------------------------

---

**Description**

Generates the UI for a user profile dropdown button to be used with the 'shinydashboard' package.

**Usage**

```
profile_module_ui(id, other_lis = NULL)
```

**Arguments**

id	the Shiny module id.
other_lis	additional <code>&lt;li&gt;</code> html tags to place between the email address and the Sign out button in the user profile dropdown. This is often used to add a user "My Account" page/app where the user can set their account settings.

---

providers_ui	<i>UI for the Firebase authentication providers buttons</i>
--------------	---

---

### Description

Creates the html UI of the "Sign in with \*" buttons. These buttons are only necessary if you enable social sign in via the `sign_in_providers` argument passed to [global\\_sessions\\_config](#).

### Usage

```
providers_ui(
  ns,
  sign_in_providers = c("google", "email"),
  title = "Sign In",
  fancy = TRUE
)
```

### Arguments

<code>ns</code>	the 'shiny' namespace function created with <code>shiny::NS()</code> .
<code>sign_in_providers</code>	the sign in providers to enable. Valid values are "google" "email", "microsoft", and/or "facebook". Defaults to "email".
<code>title</code>	The title to be used above the provider buttons. Set to <code>NULL</code> to not include
<code>fancy</code>	Should the buttons be large and colorful?

### Value

the html UI of the "Sign in with \*" buttons.

---

remove_query_string	<i>Remove the URL query</i>
---------------------	-----------------------------

---

### Description

Remove the entire query string from the url. This function should only be called inside the server function of your 'shiny' app.

### Usage

```
remove_query_string(
  session = shiny::getDefaultReactiveDomain(),
  mode = "replace"
)
```



**Arguments**

session	the Shiny session
mode	the mode to pass to <code>shiny::updateQueryString()</code> . Valid values are "replace" or "push".

---

secure_server	<i>Secure your 'shiny' app's server</i>
---------------	---

---

**Description**

This function is used to secure your 'shiny' app's server function. Make sure to pass your 'shiny' app's server function as the first argument to `secure_server()` at the bottom of your 'shiny' app's "server.R" file.

**Usage**

```
secure_server(
  server,
  custom_sign_in_server = NULL,
  custom_admin_server = NULL,
  allow_reconnect = FALSE,
  account_module = NULL,
  splash_module = NULL
)
```

**Arguments**

server	A Shiny server function (e.g <code>function(input, output, session) {}</code> )
custom_sign_in_server	Either NULL, the default, or a Shiny module server containing your custom sign in server logic.
custom_admin_server	Either NULL, the default, or a Shiny module server function containing your custom admin server functionality.
allow_reconnect	argument to pass to the 'shiny' <code>session\$allowReconnect()</code> function. Defaults to FALSE. Set to TRUE to allow reconnect with shiny-server and Rstudio Connect. Set to "force" for local testing. See <a href="https://shiny.rstudio.com/articles/reconnecting.html">https://shiny.rstudio.com/articles/reconnecting.html</a> for more information.
account_module	the server code for the user account module.
splash_module	the server code for the splash page.

---

secure_static	<i>Secure a static html page</i>
---------------	----------------------------------

---

### Description

secure\_static() can be used to secure any html page using 'polished'. It is often used to add 'polished' to ".Rmd" htmloutput and flexdashboards.

### Usage

```
secure_static(  
  html_file_path,  
  global_sessions_config_args,  
  sign_out_button = shiny::actionLink("sign_out", "Sign Out", icon =  
    shiny::icon("sign-out-alt"), class = "polished_sign_out_link")  
)
```

### Arguments

html\_file\_path the path the to html file. See the details for more info.

global\_sessions\_config\_args  
arguments to be passed to [global\\_sessions\\_config](#).

sign\_out\_button  
action button or link with inputId "sign\_out". Set to NULL to not include a sign out button.

### Details

To secure a static html page, place the html page in a folder named "www" and call secure\_static() from a file named "app.R". The file structure should look like:

- app.R
- www/
  - index.html

See an example here: [https://github.com/Tychobra/polished\\_example\\_apps/tree/master/05\\_flex\\_dashboard](https://github.com/Tychobra/polished_example_apps/tree/master/05_flex_dashboard)

### Value

a Shiny app object

---

secure_ui	<i>Secure your 'shiny' UI</i>
-----------	-------------------------------

---

### Description

This function is used to secure your 'shiny' app's UI. Make sure to pass your 'shiny' app's UI as the first argument to `secure_ui()` at the bottom of your 'shiny' app's "ui.R" file.

### Usage

```
secure_ui(
  ui,
  sign_in_page_ui = NULL,
  custom_admin_ui = NULL,
  custom_admin_button_ui = admin_button_ui("polished"),
  admin_ui_options = default_admin_ui_options(),
  account_module_ui = NULL,
  splash_module_ui = NULL
)
```

### Arguments

<code>ui</code>	UI of the application.
<code>sign_in_page_ui</code>	Either NULL, the default, or the HTML, CSS, and JavaScript to use for the UI of the Sign In page.
<code>custom_admin_ui</code>	Either NULL, the default, or a list of 2 Shiny module UI functions to add additional 'shinydashboard' tabs to the Polished admin panel. The list must be in the form <code>list("menu_items" = &lt;your_custom_admin_menu_ui('custom_admin')&gt;, "tab_items" = &lt;your_custom_admin_tabs_ui('custom_admin')&gt;)</code> .
<code>custom_admin_button_ui</code>	Either <code>admin_button_ui("polished")</code> , the default, or your custom UI to take admins from the custom Shiny app to the Admin panel.
<code>admin_ui_options</code>	list of html elements to customize branding of the "Admin Panel". Valid list element names are "title", "sidebar_branding", and "browser_tab_icon". See <a href="#">default_admin_ui_options</a> for an example.
<code>account_module_ui</code>	the ui for the user's account module.
<code>splash_module_ui</code>	the ui for the splash page.

### Value

Secured Shiny app UI

---

`send_password_reset_email_module`*server logic for Shiny module to send a password reset email*

---

**Description**

This function sends a request to the polished.tech API to reset a user's password.

**Usage**

```
send_password_reset_email_module(input, output, session, email)
```

**Arguments**

<code>input</code>	the Shiny server input
<code>output</code>	the Shiny server output
<code>session</code>	the Shiny server session
<code>email</code>	A reactive value returning the email address to send the password reset email to.

---

`send_password_reset_email_module_ui`*the UI for a Shiny module to send a password reset email*

---

**Description**

the UI for a Shiny module to send a password reset email

**Usage**

```
send_password_reset_email_module_ui(id)
```

**Arguments**

<code>id</code>	the Shiny module id
-----------------	---------------------

---

Sessions

*R6 class to track polished sessions*

---

## Description

An instance of this class handles the 'polished' user sessions for each 'shiny' app using 'polished'. The 'shiny' developer should not need to interact with this class directly.

## Methods

### Public methods:

- `Sessions$config()`
- `Sessions$sign_in_social()`
- `Sessions$get_invite_by_email()`
- `Sessions$find()`
- `Sessions$sign_in_email()`
- `Sessions$register_email()`
- `Sessions$refresh_email_verification()`
- `Sessions$set_signed_in_as()`
- `Sessions$get_signed_in_as_user()`
- `Sessions$set_inactive()`
- `Sessions$sign_out()`
- `Sessions$get_admin_mode()`
- `Sessions$clone()`

**Method** `config()`: polished Sessions configuration function

*Usage:*

```
Sessions$config(  
  firebase_config = NULL,  
  admin_mode = FALSE,  
  is_invite_required = TRUE,  
  sign_in_providers = "email",  
  is_email_verification_required = FALSE  
)
```

*Details:* This function is called via `global_sessions_config()` in `global.R` of all Shiny apps using polished.

**Method** `sign_in_social()`: verify the users Firebase JWT and store the session

*Usage:*

```
Sessions$sign_in_social(firebase_token, hashed_cookie)
```

*Arguments:*

`firebase_token` the Firebase JWT. This JWT is created client side (in JavaScript) via `firebase.auth()`.

`hashed_cookie` the hashed polished cookie. Used for tracking the user session. This cookie is inserted into the "polished.sessions" table if the JWT is valid.

*Returns:* NULL if sign in fails. If sign in is successful, a list containing the following:

- email
- email\_verified
- is\_admin
- user\_uid
- hashed\_cookie
- session\_uid

**Method** `get_invite_by_email()`:

*Usage:*

`Sessions$get_invite_by_email(email)`

**Method** `find()`:

*Usage:*

`Sessions$find(hashed_cookie, page)`

**Method** `sign_in_email()`:

*Usage:*

`Sessions$sign_in_email(email, password, hashed_cookie)`

**Method** `register_email()`:

*Usage:*

`Sessions$register_email(email, password, hashed_cookie)`

**Method** `refresh_email_verification()`:

*Usage:*

`Sessions$refresh_email_verification(session_uid, firebase_token)`

**Method** `set_signed_in_as()`:

*Usage:*

`Sessions$set_signed_in_as(session_uid, signed_in_as, user_uid = NULL)`

**Method** `get_signed_in_as_user()`:

*Usage:*

`Sessions$get_signed_in_as_user(user_uid)`

**Method** `set_inactive()`:

*Usage:*

`Sessions$set_inactive(session_uid, user_uid)`

**Method** `sign_out()`:

*Usage:*

`Sessions$sign_out(hashed_cookie, session_uid)`

**Method** get\_admin\_mode():

*Usage:*

```
Sessions$get_admin_mode()
```

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```
Sessions$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

---

set\_config\_env

*Automatically set the config environment*

---

### Description

Determines if the app is deployed to a server or running locally, and adjusts the config environment to "production" or "default" respectively. This function is almost always called in the "global.R" file of a shiny app immediately before the configuration in the "config.yml" is read in.

### Usage

```
set_config_env(override = NULL)
```

### Arguments

override	Set the environment to "default" or "production" manually. <b>CAUTION:</b> Be sure you know the difference between "default" & "production" configuration environments. Using the "production" environment will affect the database of the deployed application.
----------	--

---

sign\_in\_check\_jwt

*Check the JWT from the user sign in*

---

### Description

This function retrieves the JWT created by the JavaScript from [sign\\_in\\_js](#) and signs the user in as long as the token can be verified. This function should be called in the server function of a shiny module. Make sure to call [sign\\_in\\_js](#) in the UI function of this module.

### Usage

```
sign_in_check_jwt.jwt, session = shiny::getDefaultReactiveDomain())
```

### Arguments

jwt	a reactive returning a Firebase JSON web token for the signed in user.
session	the shiny session.

---

`sign_in_js`*Sign in and register pages JavaScript dependencies*

---

**Description**

This function should be called at the bottom of your custom sign in and registration pages UI. It loads in all the JavaScript dependencies to handle polished sign in and registration. See the vignette for details.

**Usage**

```
sign_in_js(ns)
```

**Arguments**

`ns` the ns function from the Shiny module that this function is called within.

---

`sign_in_module`*Server logic for the sign in and register pages*

---

**Description**

This server logic accompanies the [sign\\_in\\_module\\_ui](#).

**Usage**

```
sign_in_module(input, output, session)
```

**Arguments**

`input` the Shiny input  
`output` the Shiny output  
`session` the Shiny session



---

sign_in_module_2	<i>Server logic for the sign in and register pages</i>
------------------	--

---

**Description**

This server logic accompanies [sign\\_in\\_module\\_2\\_ui](#).

**Usage**

```
sign_in_module_2(input, output, session)
```

**Arguments**

input	the Shiny input
output	the Shiny output
session	the Shiny session

---

sign_in_module_2_ui	<i>UI for the sign in and register pages</i>
---------------------	--

---

**Description**

Alternate sign in UI that works regardless of whether or not invites are required. The UI displays email sign in inputs on the left, and social sign in options on the right. [sign\\_in\\_module\\_2](#) must be provided as the argument "custom\_sign\_in\_server" in [secure\\_server](#) for proper functionality.

**Usage**

```
sign_in_module_2_ui(id)
```

**Arguments**

id	the Shiny module id
----	---------------------

---

sign\_in\_module\_ui      *UI for the sign in and register pages*

---

### Description

UI for the sign in and register pages when a user invite is required to register and sign in.

### Usage

```
sign_in_module_ui(id, register_link = "First time user? Register here!")
```

### Arguments

id	the Shiny module id
register_link	The text that will be displayed in the link to go to the user registration page. The default is "First time user? Register here!". Set to NULL if you don't want to use the registration page.

---

sign\_in\_ui\_default      *Default UI styles for the sign-in pages*

---

### Description

Default styling for the sign in and registration pages. Update the sign\_in\_ui\_default() arguments with your brand and colors to quickly style the sign in and registration pages to match your brand.

### Usage

```
sign_in_ui_default(
  sign_in_module = sign_in_module_ui("sign_in"),
  color = "#5ec7dd",
  company_name = "Your Brand Here",
  logo_top = tags$div(style = "width: 300px; max-width: 100%; color: #FFF;", class =
    "text-center", h1("Your", style = "margin-bottom: 0; margin-top: 30px;"), h1("Brand",
    style = "margin-bottom: 0; margin-top: 10px;"), h1("Here", style =
    "margin-bottom: 15px; margin-top: 10px;")),
  logo_bottom = NULL,
  icon_href = "polish/images/polished_icon.png",
  background_image = NULL,
  terms_and_privacy_footer = NULL,
  align = "center"
)
```

**Arguments**

sign_in_module	UI module for the sign in and registration pages.
color	hex color for the background and button.
company_name	your company name.
logo_top	html for logo to go above the sign in panel.
logo_bottom	html for the logo below the sign in panel.
icon_href	the url/path to the browser tab icon.
background_image	the url/path to a full width background image. If set to NULL, the default, the color argument will be used for the background instead of this image.
terms_and_privacy_footer	links to place in the footer, directly above the copyright notice.
align	The horizontal alignment of the sign in box. Defaults to "center". Valid values are "left", "center", or "right"

**Value**

the UI for the sign in page

---

sign\_out\_from\_shiny     *Sign our from your 'shiny' app*

---

**Description**

Call this function to sign a user out of your 'shiny' app. This function should be called inside the server function of your 'shiny' app. See [https://github.com/Tychobra/polished/blob/master/inst/examples/polished\\_example\\_01/server.R](https://github.com/Tychobra/polished/blob/master/inst/examples/polished_example_01/server.R) For an example of this function being called after the user clicks a "Sign Out" button.

**Usage**

```
sign_out_from_shiny(session = shiny::getDefaultReactiveDomain())
```

**Arguments**

session	the Shiny session
---------	-------------------

# Index

default\_admin\_ui\_options, [2](#), [11](#)

email\_input, [3](#)

firebase\_dependencies, [3](#), [4](#)  
firebase\_init, [4](#)

global\_sessions\_config, [5](#), [8](#), [10](#)

password\_input, [6](#)  
profile\_module, [7](#)  
profile\_module\_ui, [7](#), [7](#)  
providers\_ui, [8](#)

remove\_query\_string, [8](#)

secure\_server, [9](#), [17](#)  
secure\_static, [10](#)  
secure\_ui, [11](#)  
send\_password\_reset\_email\_module, [12](#)  
send\_password\_reset\_email\_module\_ui,  
[12](#)

Sessions, [13](#)

set\_config\_env, [15](#)  
sign\_in\_check\_jwt, [15](#)  
sign\_in\_js, [15](#), [16](#)  
sign\_in\_module, [16](#)  
sign\_in\_module\_2, [17](#), [17](#)  
sign\_in\_module\_2\_ui, [17](#), [17](#)  
sign\_in\_module\_ui, [16](#), [18](#)  
sign\_in\_ui\_default, [18](#)  
sign\_out\_from\_shiny, [19](#)