# Create MeteorJS project get initial dashboard template functional

using <https://gogo-react.coloredstrategies.com/app/dashboards/default>

Reference: <https://gogo-react-docs.coloredstrategies.com/docs/gettingstarted/introduction>

## Create MeteorJS project

Reference: <https://www.meteor.com/tutorials/react/components>

* using gogo react theme: (attached file: themeforest-473QZraZ-gogo-react-bootstrap-4-admin-dashboard.zip)
* Source Repository: <https://github.com/nastia723/arcadia_web/invitations>

<https://github.com/nastia723/arcadia_web.git>

Create Branch, branch name is: ui

## Make login page

<https://gogo-react.coloredstrategies.com/user/login>

# Make Dashboard

## Main Menu

The main menu will use the Gogo React template format by using both the first Main menu and then displaying a secondary menu (including icons with text below on the main menu) with nested tree hierarchy as outlined below:

Main Menu-[**Dashboard**](#_Dashboard) (use icon simple-icon-home)>Users Custom Dashboard View(SPA Dashboard Home for user, no secondary menu)

FYI: <https://gogo-react.coloredstrategies.com/app/ui/components/icons>

Main Menu-[**Site Viewer**](#_Site_Viewer) (use icon iconsminds-space-needle)>Secondary Menus-(Top Level) NORTHEAST,NORTHWEST,SOUTHEAST,SOUTHWEST,SOUTH,NORTH,ISLAND and then list each site that is associated with the region listed below.

For example, to find in site view REGION NE SITEID 12345 in the list, a user would click Site View and a secondary menu appears. There are 7 grey collapsed options, each of them being a region. When each region is clicked, it will show a listing of all the sites from that region. A user can then click on that site ID from the list and be taken to the Site Viewer Tool, where information from our database is populated into a pre formatted template based on Pages>Product>Details in the Gogo react template.

Main Menu- **File Manager** (use icon simple-icon-cloud-upload)>Secondary Menu

Main Menu-**Notifications** (use icon simple-icon-feed)>Take user to notifications page and display notifications according to the below notifications page definition below.

Main Menu-**Reporting**(use icon simple-icon-printer)>Go to Report Generator Page for now \*\*TODO:Make this more in depth

## Dashboard

Similar to

<https://gogo-react.coloredstrategies.com/app/pages/product/data-list>

# Make Profile Page

Gogo Theme: Home>Pages>Profile>Social

Link: <https://gogo-react.coloredstrategies.com/app/pages/profile/social>

Each user should be able to add status updates, upload photos and videos to their profile (utilizing their profile Azure Blob Storage). There should also be the default above Links in the profile (Profile, Images, Friends) and should work very much like any other social media network.

Under the Profile edit, the user should be able to edit the following information on their public profile:

* Full Name
* Photo (add that into the Azure Blob for their profile-name it profile.jpg or profile.png)
* Banner Photo (add that into the Azure Blob for their profile, fit into the box without making it look weird or add some options like twitter does)
* About Me(place where it says the filler content about the person being a web developer)
* Location
* Responsibilities (with the roles found above in the same format found in the template example)
* Contact Social Media Links (Facebook, Twitter, LinkedIn)
* Contact Email
* Contact Phone Number (must be direct dial, preferably a cell phone)
* Can call or cannot call (check box or something, can prevent Twilio calling for user)

Under this, show a box that displays all of the sites the user has enabled for notifications.

Super Admin should be able to see additional information under the users profile, such as their parent account and in the future permissions/sites they are able to see.

We want to show all the information above, including under contact adding an icon-telephone that will open a small box asking for the users telephone number. They will enter the number and be connected via Twilio to the users telephone number without knowing it).

# New Site

We would like to have a Menu Item that takes us to a wizard to add a new site.

<https://gogo-react.coloredstrategies.com/app/ui/forms/wizard>

Step One: Site Name, Site Antenna Structure Registration (if any),

Step Two: Site Address, Site City, Site State, Site Zip, Site Region.

If the site is not found in the estated API, ask user for Latitude and Longitude information and see if the estated api returns any information for the location.

If it fails, tell the user cannot find property/location and end the process. Otherwise, we would like to pull the Geocoded address and display a number of data points from our partner API’s about the property.

**Each new site record created should have a new ISLA\_SITE\_ID associated with it at the time of creation.**

First, we would like to query the Estated API and save the following information to our new site record we are creating:

**The following Metadata will be pulled from the Estated API and inserted into the SITEID record being created as a new site:**

property\_id(int),

UPI(string),

published\_at(datetime),

updated\_at(date\_time),

**The following address data will be pulled from the Estated API and inserted into the SITEID record as well:**

street\_number(string),

street\_pre\_direction(string),

street\_suffix(string),

street\_post\_direction(string),

formatted\_street\_address(string),

city(string),

state(string),

zip\_code(string),

zip\_code\_plus\_four(string)

**The following geocoding data will be pulled from the Estated API and inserted into the SITEID record:**

street(object),

parcel\_centroid(object),

rooftop(object).

**The following Geocoding(address) information will be pulled from the Estated API and inserted into the SITEID record:**

latitude(decimal),

longitude(decimal).

**The following parcel information will be pulled from the Estated API and stored in the SITEID record:**

depth(integer),

width(integer),

size(integer),

acres(decimal),

range(string),

township(string),

section(string),

block(string),

lot (string),

location\_description(string),

building\_count(integer),

boundaries(object)

**The following parcel.boundaries information will be pulled from the Estated API and stored in the SITEID record:**

srid(integer),

wkt(string),

geojson(object).

**The following parcel.boundaries.geojson information will be pulled from the Estated API and stored in the SITEID record:**

type(string),

coordinates(decimal)

**The following land\_use information will be pulled from the Estated API and stored in the SITEID record:**

classification\_code(string),

category(string),

subcategory(string),

source\_code(string),

zoning(string).

**The following structures array will be pulled from the Estated API and saved (may be multiple structures returned, save them all and stored in the SITEID record:**

is\_primary(Boolean), year\_built(integer), effective\_year\_built(integer), identifier(string), rooms\_count(integer), beds\_count(integer), half\_baths\_count(integer), full\_baths\_count(integer), quarter\_baths\_count(integer), three\_quarter\_baths\_count(integer), stories\_count(decimal), units\_count(integer), total\_size(integer), finished\_size(integer), unfinished\_size(integer), basement\_size(integer), garage\_size(integer), height(integer), architecture\_type(string), exterior\_wall\_type(string), air\_conditioning\_type(string), basement\_type(string), building\_type(string), category(string), condition(string), construction\_type(string), fireplace\_count(integer), foundation\_type(string), flooring\_type(string), garage\_type(string), heating\_type(string), mobile\_home(Boolean), parking\_spaces(integer), parking\_type(string), pool\_type(string), quality(string), roof\_type(string), roof\_style\_type(string), electricity\_type(string), fuel\_type(string), sewer\_type(string), water\_type(string).

**The following owners array will be pulled from the Estated API and saved. Will be multiple records of all current and previous owners on file, Save all to SITEID record**: corporate\_flag(Boolean), name(string), second\_name(string), address(string), city(string), state(string), zip\_code(string), first\_name(string), middle\_name(string), last\_name(string), suffix(string), ended\_at(date), started\_at(date).

**The following assessments array will show all tax assessments on file for the site, may be multiple save all to SITEID record:** land(integer), improvement(integer), total(int), exemptions(array), tax\_amount(int), tax\_year(int), details(array), outstanding\_tax(decimal), appraised\_land(int), appraised\_improvement(int), appraised\_total(int).

**The following sales(deeds) array will show all sales records on file for the property, may be multiple so save all to SITEID record:** date(date), recording\_date(date), price(int), type(string), seller(string), buyer(string), transaction\_id(string), document\_year(int), sale\_code(string), multi\_apn(Boolean), hpi\_adjusted\_price\_2010(int), deed\_book(string), deed\_page(string), document\_id(int).

**The following mortgages(deeds) array will return all mortgage records on file. There may be multiple so save all to SITEID record:** amount(int), loan\_type(string), deed\_type(string), date(date), term(int), due\_date(date), lender\_name(string), details(array).

**The following legal(parcel) details of the property will be returned for the property and need to be saved to SITEID record:** apn\_original(string), apn\_unformatted(string), tax\_account(string), legal\_description(string), planned\_urban\_development(string).

**The following valuation will be returned for the property and will be saved for the SITEID record:** value(int), low(int), high(int), confidence(decimal), market\_value\_change\_year(decimal), market\_value\_change\_quarter(decimal), forecast\_1\_year(decimal), suggested\_rental(int).

**The following postal information(address) will be returned for the property. Save all for the SITEID record:** carrier\_code(string), delivery\_point\_code(int), delivery\_point\_check\_digit(int), validated(Boolean), deliverable(string).

**The following geographies information will be returned and need to be saved into the SITEID record:** congressional(string), neighborhood(string).

**The following geographies.county information will be returned and need to be saved into the SITEID record:** name(string), state\_fips\_code(string), fips\_code(string).

**The following geographies.census information will be returned and need to be saved into the SITEID record:** census\_tract(int), census\_block(string).

**The following geographies.cbsa will be returned for the Core based statistical area information on the property and needs to be saved into the SITEID record:** name(string), code(string), type(string).

**The following geographies.school\_elementary will be returned for the properties district and may be multiple so save all to SITEID record:** geoid(string), name(string), low\_grade(string), high\_grade(string), type(string).

**The following geographies.school\_secondary will be returned for the properties district and may be multiple so save all to SITEID record:** geoid(string), name(string), low\_grade(string), high\_grade(string), type(string).

**The following geogaphies.state\_legislative\_upper will be returned for the property about the State Legislative Upper House and saved to SITEID record:** name(string).

**The following geographies.state\_legislative\_lower will be returned for the property about the State Legislative Lower House and saved to SITEID record:** name(string).

**The following geographies.police will return information on Police about the property, save all to SITEID record:** name(string), type(string), city(string), state\_fips(string), county\_fips(string).

**The following geographies.fire will return information on Fire Department for area, save all to SITEID record:** fdid(string), name(string), address(string), address\_2(string), city(string), zip\_code(string), phone(string), mailing\_address(string), department\_type(string), organization\_Type(string), website(string), stations\_count(int), career\_firefighters\_count(int), volunteer\_firefighters\_count(int), paid\_per\_call\_firefighters\_count(int), non\_firefighting\_staff\_count(int), interpolated\_latitude(decimal), interpolated\_longitude(decimal, county(string), state(string).

**The following geographies.flood information will be returned and needs to be saved in the SITEID record:** fima\_floode\_zone(string), annual\_flood\_risk(string), firm\_panel\_id(string).

# Site Viewer

(NE,NW,SE,SW,S,N,ISLAND)

should have nested below a **list of states** for the region as assigned in the site. After the State has been clicked, further selectable is the **city** of the site as assigned. After the city has been selected, now we see **SITEID-Name** and are able to select that site to take us to the record we are needing to see.

<https://gogo-react.coloredstrategies.com/app/pages/product/details>

* On the left side of the container (where you see food), a **mapbox plugin** should be displayed with the following checkboxes that can enable different layers: Map View (showing a map of the site), Boundary View, and 3D view (enabling the display of our threejs gltf for the site, which will be located in REGION>SITEID>3DASSETS>main.gltf. This GLTF file will be a summation of all assets produced in relation to the site. The quality of the GLTF is to be determined). This mapbox container should also display local traffic as well as any weather data available. Boundary View will use the boundary information as recorded in parcel.boundaries in the geojson object.
* **Additionally the following information should be showed on the Top Right Box replacing the lorem ipsum and remove the likes/comments icons**: Current Weather from AccuWeather API, Forecast Weather from Accuweather API, Current Traffic from Waze API or Google Traffic API. Leave tags and add a small text entry that will autopopulate popular tags used in other sites or add a new tag in the form of a #hashtag. This should be with the site, and should be logged in the site history with date and user who created the tag.

**AT THE END OF THIS WE SHOULD HAVE THE FOLLOWING TABS LOCATED IN A SINGLE BOX:**

1. **General = Estated V4 Address**
2. **Property = Parcel**
3. **Structure = Structure**
4. **Owner = ?**
5. **Tax = ?**
6. **Sales = ?**
7. **Mortgages = ?**
8. **Legal = ?**
9. **Other = ?**

* **The following site information should be pulled from the database and displayed in the General Tab(Replace the Details Tab in the example with General):** Site ID, Owner\_Name, Formatted\_Street\_Address, City, State, Zip\_Code\_plus\_four at the top of the site record in bold large letters to signify a Title.
* **Add Property tab next to General and the following should be shown:** Depth (ft), Width (ft), Size, Acres, Range, Township, Section, Block, Lot, Location Description, Building Count, and Boundaries (need to figure out how to show this object but will be inside of threejs/mapbox).

**Under the same Property tab the following information should be shown further:** Classification Code, Category, Sub Category, Source Code, Zoning Code.

* **Add Structure tab to the right of the Property tab, the following information should be shown (found in the estated structures api array):** Year Built, Year Updated (linked to effective\_year\_built), Identifier, Rooms Count, Beds Count, Baths Count, Half Baths Count, Full Baths Count, Quarter Baths Count, Three Quarter Baths Count, Stories Count, Units Count, Total Size, Finished Size, Unfinished Size, Basement Size, Garage Size, Height, Architecture Type, Exterior Wall Type, Air Condition Type, Basement Type, Building Type, Category, Condition, Construction Type, Fireplace Count, Fireplace Type, Foundation Type, Flooring Type, Garage Type, Heating Type, Mobile Home (can be X or check mark based on False or True return), Parking Spaces, Pool Type, Quality, Roof Type, Roof Style Type, Electricity Type, Fuel Type, Sewer Type, Water Type.
* **Under the Owner tab display all array information with each record having its own small bubble of data shown. Stack the containers from newest first, to oldest last. The information should be in the form of:** Corporate Flag(Boolean true or false, display X if false green check if TRUE), Name, Second Name, Address, City, State, Zip Code, First Name, Middle Name, Last Name, Suffix, Ended At, Started At.

**ALL MENTIONS OF A “BUBBLE” should be in the preformat of “Data List” in GogoReact Template and are found under HOME>PAGES>PRODUCT>DATA LIST.**

* **Under the Tax tab display all array information in the same above format with each record in the array having its own bubble generated showing information for:** Land (currency USD), Improvement (currency USD), Total (currency USD), Exemptions, Tax Year (bold and on upper left of the container to stand out), Tax amount (currency USD), Details, Outstanding Tax (currency USD), Appraised Land Value (currency usd), Appraised Improvement Value (currency USD), Appraised Total (currency USD).
* **Under the Sales Tab display all array information in the same above format with each record in the array having its own bubble generated for showing information in the sales array from the Estated API. Each bubble should be the same size showing Date, Price, Type as the basic information and when clicked show the rest of the information in a auto sized bubble**: Date, Recording Date, Price, Type, Seller, Buyer, Transaction ID, Document Year, Sale Code, Multi APN (true or false, can be red X or green check), HPI Adjusted Price 2010, Deed Book, Deed Page, Document ID.
* **Under the Mortgages Tab display all array information in the same above format with each record in the array having its own bubble generated showing information:** Amount, Loan Type, Deed Type, Date, Term, Due Date, Lender Name, Details.
* **Under the Legal Tab display all legal information from the estated API:** APN Original, APN Unformatted, Tax Account Number, Legal Description, Planned Urban Development.
* **Under the Other Tab display all other information grouped together in boxes or well organized/sectioned:** (Geographies.county)Name, State FIPS Code, FIPS Code / (Geographies.census)Census Tract, Census Block / (Geographies.cbsa)Name, Code, Type/ (geographies.police)Name, Type, City, State FIPS, County FIPS/ (geographies.fire)Fire Department ID (linked to fdid), Name, Address, Address Second, City, Zip Code, Mailing Address, Department Type, Organization Type, Website (with clickable link), Stations Count, Career Firefighters Count, Volunteer Firefighters Count, Non Firefighting Staff, County, State/ (geographies.flood)Fima Flood Zone, Annual Flood Risk, Firm Panel ID.

# File Viewer Tool

On the **left Menu**, the file viewer tool should take the user to a single screen containing a **mapbox map** on **the left** and a **sites display box** on the **right**. The user should be able to add sites into the map, and click on the sites to view a **list** of files below the map and to the left of the sites field on the right. User should also be able to see when the file was added, who added it, and any comments they left.

# Upload Tool

**File and Media Storage/Upload Tool**

[**https://medium.com/@stuarttottle/upload-to-azure-blob-storage-with-react-34f37805fdfc**](https://medium.com/@stuarttottle/upload-to-azure-blob-storage-with-react-34f37805fdfc)

**Very Similar to the above link, but the ability to tag files (can be hashtag or select media type, but needs to be some way to store all of them, and make them available for viewing and retrieval via download or email)**

Our platform needs the ability to easily upload various file content by multiple users throughout the evolution of our project. We would like our data storage platform to be Azure Blobs. When a user would like to upload content, they will go to the Upload Tool on the Menu and first be given a search box with a region. The user will select the region (NE,NW,SE,SW,S,N,ISLAND) which will be then followed by a site ID. This ID may be found from various fields in our data, such as: CUSTOMER\_SITE\_ID,TOWER\_SITE\_ID,ASR\_SITE\_ID,ISLA\_SITE\_ID. After the site has been selected, populate information to the user to ensure the site matches the site they would like to upload for and have permission to do so. Then we would like the user to be able to select files from their computer, or also drag them into the browser window and attach them for upload into the Azure blob storage. We would like the blob storage to be organized by REGION>SITE\_ID>MEDIA\_TYPE where MEDIA\_TYPE can be: Laser Scan (.las, .e57, .pts, .xyz), Excel (.xls, .xlsx, .xlxb), Word (.txt, .doc, .docx, .odf), PDF (.pdf), Tower (.txn), AutoCAD (.dwg, .dxf), Photos (.jpg, .png), Video (.mp4, .avi, .mov),MeshGun (.mkv, .mjpeg), 3D Assets (.obj, .fbx, .ply, .gltf, .3ds, .maya, .usd).

**Laser Scan  
Excel  
Word  
PDF  
Tower  
AutoCAD  
Photos  
Video  
MeshGun  
3D Assets**

Once the files have been attached, we would like to show the file size as well as automatically categorize them by the associated file types and then uploaded to their respective folders inside of the REGION>SITE\_ID>MEDIA\_TYPE. For example, if a user uploaded a .las file the system for a Region NE site 12345 would automatically recognize it as a laser scan file, and deposit that file into NE>12345>LaserScan and save it as DATEOFUPLOAD-UPLOADERUSERID-originalfilename.las. After the files have been successfully saved, we would like an email to go out and notifications to be sent via our platform informing users of a new upload and who uploaded/when etc.