Form Builder MVP

Contents

[Landing Page 2](#_Toc482977861)

[MY FORMS 2](#_Toc482977862)

[“MY FORMS” menu option is provided on the landing page. 2](#_Toc482977863)

[“MY FORMS” Page 2](#_Toc482977864)

[Form Actions 3](#_Toc482977865)

[Create Form 3](#_Toc482977866)

[Form Builder 4](#_Toc482977867)

[Form Elements 5](#_Toc482977868)

[Adding Controls 7](#_Toc482977869)

[Control Properties 7](#_Toc482977870)

[Saving Form Design 9](#_Toc482977871)

[Publish 9](#_Toc482977872)

[Form Submission 9](#_Toc482977873)

[Viewing Form Submissions 9](#_Toc482977874)

[Submissions UI 10](#_Toc482977875)

[Planning 11](#_Toc482977876)

[Marketable Features 11](#_Toc482977877)

[Database Design 12](#_Toc482977878)

[Collections 12](#_Toc482977879)

[Database Initalization 12](#_Toc482977880)

[Security 12](#_Toc482977881)

[Setting up Development Environment 13](#_Toc482977882)

[Install Software 13](#_Toc482977883)

[Workflow – User 13](#_Toc482977884)

[Workflow - Design 13](#_Toc482977885)

# Landing Page

Design a landing page

* + Navigation Menu
    - My Forms
    - Login
    - Signup
  + Content
  + Footer
    - About
    - Github

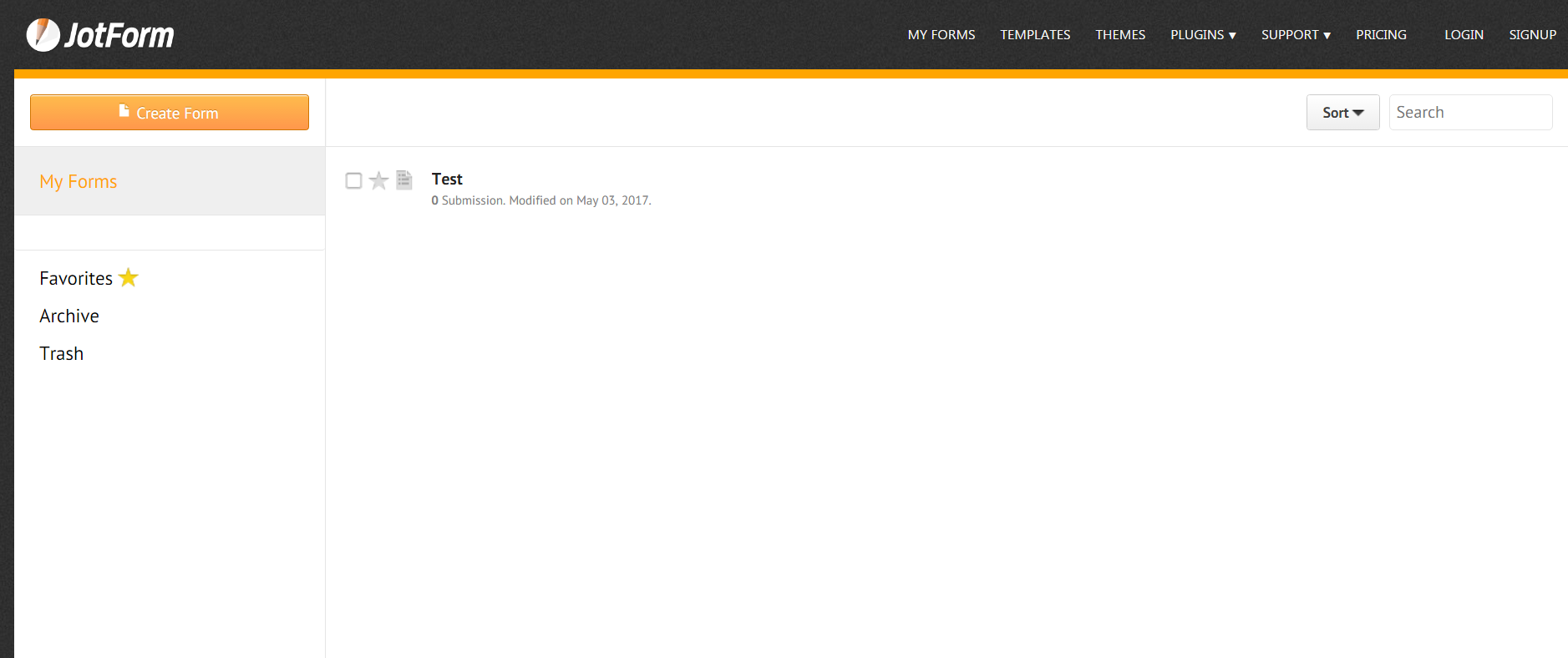
# MY FORMS

## “MY FORMS” menu option is provided on the landing page.

## “MY FORMS” Page

User doesn’t need to sign up or login to access “MY FORMS” page.

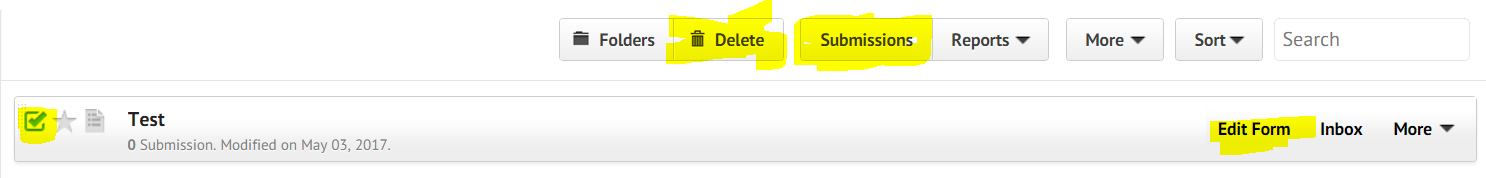
NOTE: We need a way of identifying the guest user so that the forms can be presented to the user when he/she visits “My Forms” page. One way of doing this is to use session cookie.



1. Show list of created forms.
2. Create Form button at the top left.

### Form Actions

When user selects a form, action menu is shown :



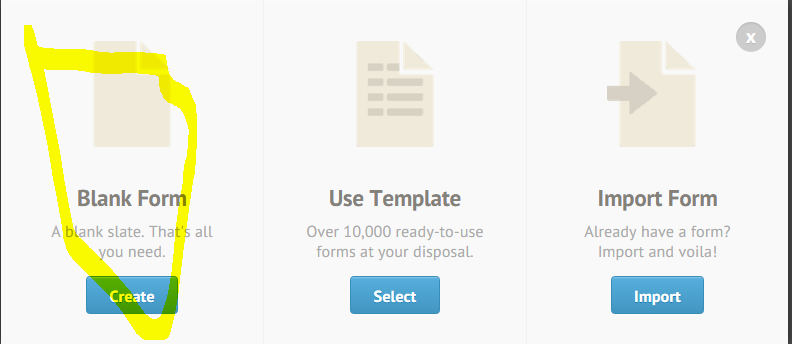
1. Allow user to select a form
2. Delete form
3. Check submissions
4. Edit Form option on the right

# Create Form

User needs to click on “Create Form” button.

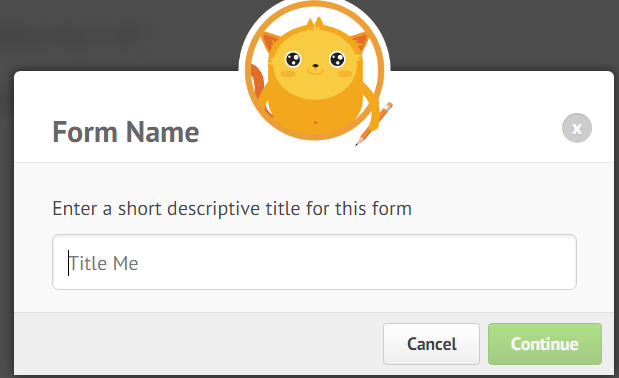


1. Form Template Selection dialog is shown. For MVP, we will do only “Blank Form”.



1. User clicks on “Create” button.

“Form Name” dialog is shown asking user to enter form name

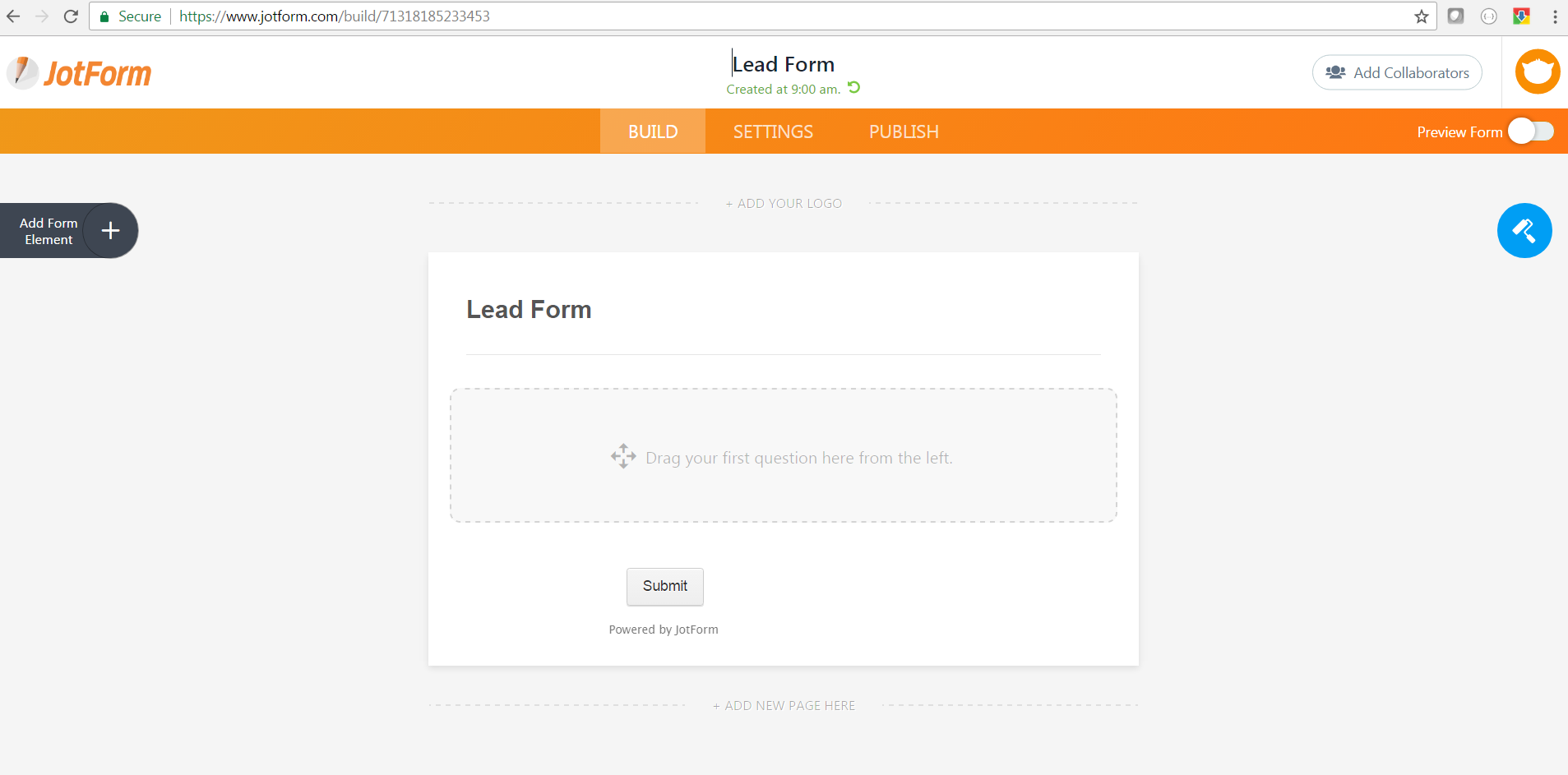


Clicking on “Cancel” closes the popup.

Clicking on “Continue” takes the user to the Form Builder UI.

“Continue” button is disabled if there is no text entered in “Title” textbox. When user enter any text, it gets enabled.

# Form Builder



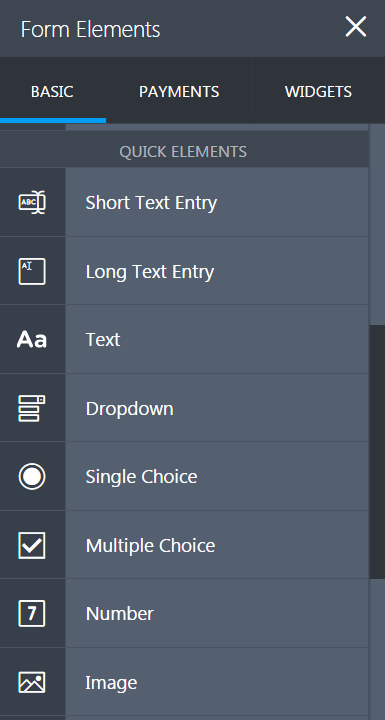
Above is the initial blank canvas for Form Builder.

MVP Features:

1. We will maintain the same layout i.e Header, Menu Bar, Main Canvas Page
2. Header:
   1. Logo on the left
   2. Form Title
3. Menu Bar
   1. Build
   2. Publish
   3. Preview Form
4. Main Canvas Page
   1. Add Form Element Panel. We will not show the “+” button instead show the controls panel directly on the left.
   2. No “Add Your Logo”, “Add New Page Here”
   3. No “Paint” icon as shown above on the right side.

## Form Elements

Form Elements (Controls Panel) will be shown on the left side.

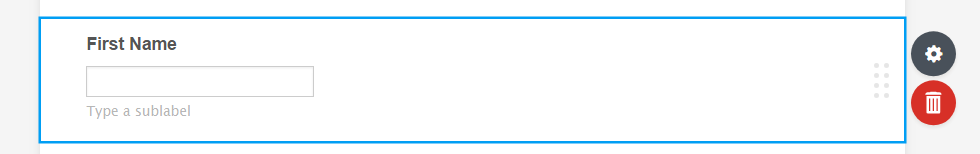


MVP Features:

1. Only Basic tab
2. No ‘X’ button
3. Only Quick Elements
   1. Short Text
   2. Long Text
   3. Dropdown
   4. Checkbox
   5. Radio
   6. Text
   7. Date
   8. Number

# Adding Controls

User has to drag and drop a control onto the center content. We will continue to have “Add Row” option. We will not show WYSWIG UI for the dropped control.

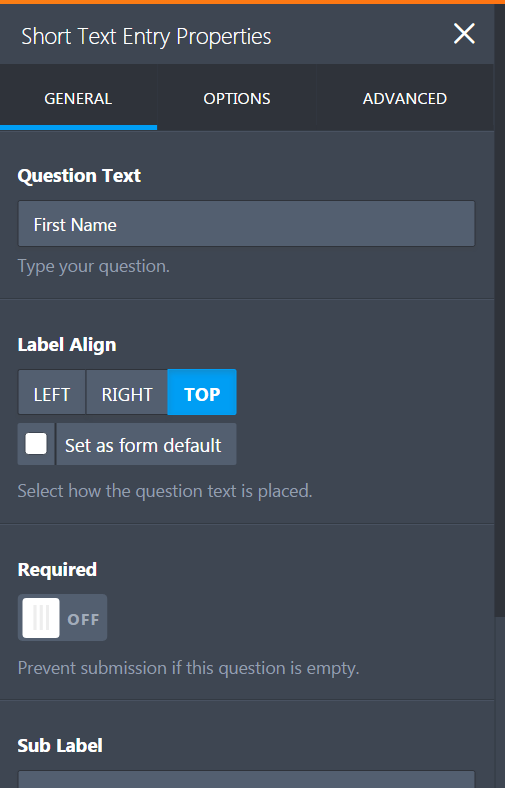


Show 2 options as shown above:

1. Properties
2. Delete

# Control Properties

Control properties UI will be shown on the right side.



MVP Features:

1. General and Advanced tabs
2. General
   1. Label
   2. Required
   3. Placeholder text
   4. Read Only
3. Advanced
   1. Control specific properties

# Saving Form Design

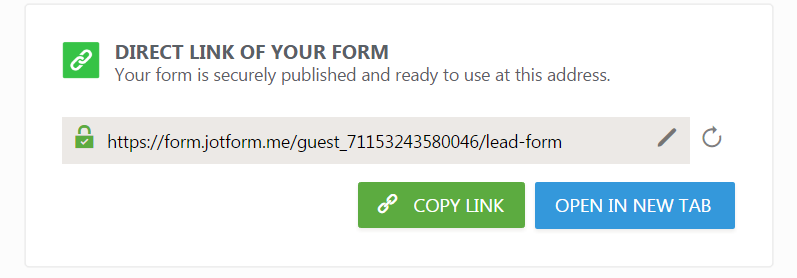
We will provide a “Save” button which will save the form. No auto saving option for MVP.

# Publish

1. Publish option on the menu bar.



1. Clicking on “Publish” shows the following UI:



# Form Submission

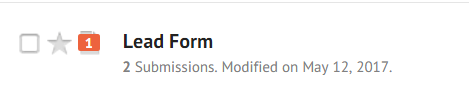
User needs to open the published form url in a browser. The form will be shown to the user. User can enter the details and click “Submit”.

On “Submit” any validations will be shown

On successful validations, form data will be saved and a message will be shown to the user.

# Viewing Form Submissions

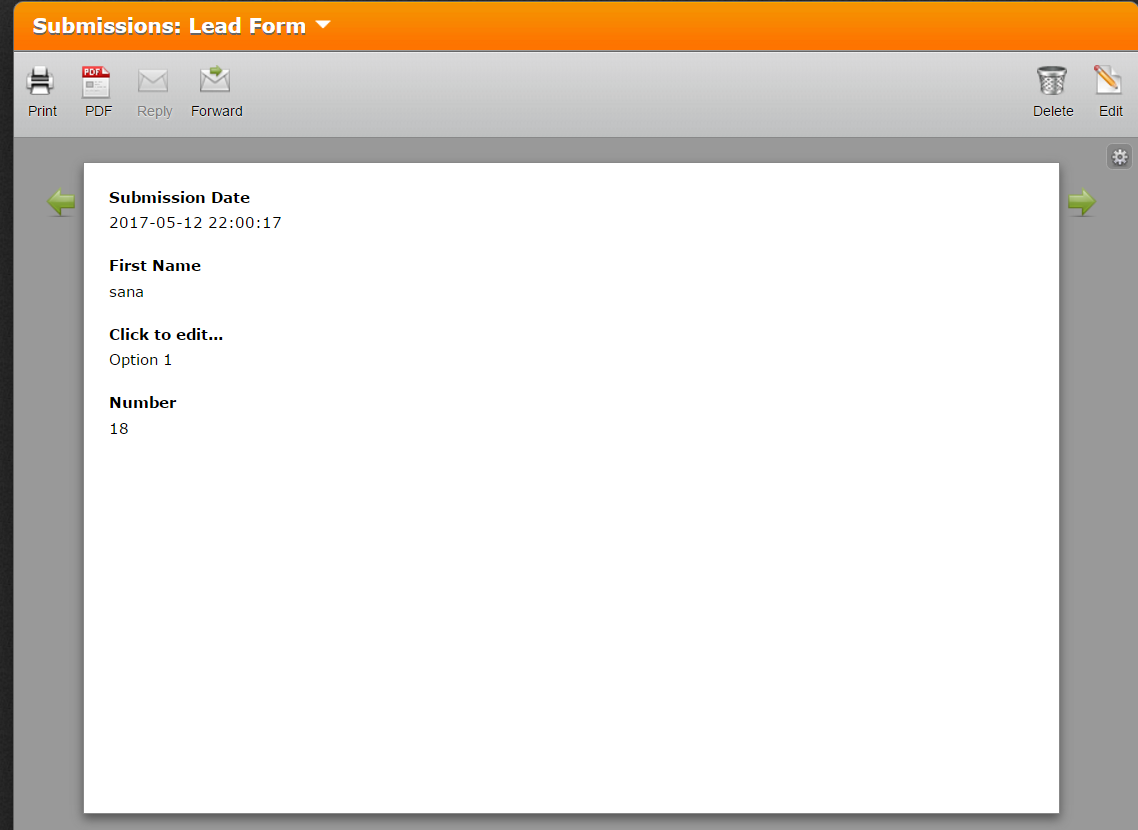
The submitted form data by the users will be shown on the “MY FORMS” page.



1. Number of submisisons will be shown to the user as shown above
2. User selects the form and clicks on “Submission” button to view the Submissions



## Submissions UI



1. Maintain the same layout
2. Only Delete option for MVP
3. Form Submissions will be shown in LIFO order i.e latest submissions first and so on
4. User can use the navigation arrows to switch between different submissions

# Planning

## Marketable Features

1. ~~Design and Architecture~~ (6) **DONE**
   1. Identify Tech stack
      1. FrontEnd – Angular , TypeScript, JavaScript, Angular Material, SCSS, Angular CLI, Jasmine, NPM, Chrome Dev Tools, VS Code,
      2. Server - Visual Studio, Node JS (> 7.6.0), Express, ES6
      3. Database – MongoDB
      4. Source Control – git , GitHub, code structure
      5. Project Management – Visual Studio Online
      6. Cloud – Azure
      7. CI/CD – Chef, Puppet, Travis
   2. High level design
      1. Front End
         1. Component and module architecture
         2. SCSS framework - TEJAS
         3. ~~UI wrapper~~ - **DONE**
         4. ~~Routing -~~ **DONE**
            1. Primary Routes (<router-outlet> in app)

home

Myforms

Design/:id/edit (design/0/edit for new)

Submissions/:id

Preview/:id

* + - 1. ~~Setup base project -~~ **~~DONE~~**
         1. FrontEnd

FormBuilder

App

Myforms

Formdesigner

Submissions

Preview

Shared

FormRenderer

UI

Router

FormSubmitter

* + 1. Server
       1. ~~Design RESTful API and document~~ - **DONE**
       2. ~~Authentication using cookie/header~~ - **DONE**
       3. ~~Setup base project~~ - **DONE**
    2. ~~Database -~~ **~~DONE~~**
       1. ~~Data modelling~~
       2. ~~Indexing~~
       3. Security
  1. ~~Setup projects~~ **DONE**
  2. ~~Source code control~~ **DONE**
  3. Identify tech spikes
     1. ~~Using Angular CLI. Setting up. –~~ **~~DONE~~**
     2. ~~How to not set up git when creating using cli? Not possible. Instead delete .git folder. - DONE~~
     3. ~~Production build of Angular using CLI. How to deploy angular prod build to server. - DONE~~
     4. using Node for publishing angular prod build
     5. ~~Deploying Node to Azure – Do it as part of Deployment MF~~
     6. ~~CI/CD using Azure and GitHub - Do it as part of Deployment MF~~
     7. ~~Designing UI wrapper module so that underlying UX framework can be easily swapped. For eg: Angular Materia, Kendo UI~~

1. Landing Page (10)
2. My Forms (20)
3. Form Builder (New, Edit) (40)
4. Save Form (10)
5. Publish Form (20)
6. View Form Submissions (40)
7. Do Form submissions (40)
8. Deployment (40)
   1. <https://www.visualstudio.com/en-us/docs/build/get-started/aspnet-4-ci-cd-azure-automatic>
   2. MongoDb security
   3. Understand domain registration and pricing
   4. Resetting base path for Angular apps. Its set by default to <base href=”/”> in index.html

Using Angular CLI: ng build –base-href /formbuilder/

## Database Design

We will be using MongoDb as the database.

### Collections

|  |  |  |
| --- | --- | --- |
| Name | Schema | Index |
| formsMeta | {  Id:’’,  userId:’’,  formName:’’,  meta:{}  } | Id, userId |
| formsData | {  formId,  data:{  }} | formId |
| Users | Id  isGuest  loginId  salt  hash | Id |

### Database Initalization

We need to write code when creating MongoDB to set up the following:

1. Create above indexes when creating database and collections. Use db.collection.createIndex API

### Security

TBD in deployment

# Setting up Development Environment

## Install Software

|  |  |  |
| --- | --- | --- |
| Node | >7.6.0 | Nodejs.org |
| NPM |  |  |
| MongoDB |  |  |
| VS Code |  |  |
| Visual Studio 2015 > |  |  |
| Node JS Tools for Visual Studio |  |  |
| Git |  |  |
|  |  |  |

**## Scaffolding for FormBuilder and FormSubmitter**

1. Requirements - Node, NPM, Angular CLI
2. To generate app:

**ng new FormBuilder --prefix fb --routing -sg true --style scss**

|  |  |
| --- | --- |
| **--style scss** | **Styles should use SASS** |
| **--prefix fb** | **Use fb prefix** |
| **-sg true** | **Skip git** |
| **--routing** | **Add a default routing module** |
|  |  |

1. To generate artifacts
2. To serve

**ng serve -o –w**

1. To lint

ng lint –-format stylish

1. To build

Ng build

**## VS Code Settings**

File > Preferences menu or the Command Palette (Ctrl+Shift+P) search for File Icon Theme

Select Seti

# UI Design

## Approach

Create UI module which provides wrappers for underlying UI elements. This allows swapping the UI framework without impacting the entire application.

UI Library

(Kendo, Ng Material)

UI Module

App/Module

## Setting up Angular Material

1. Install 2 packages : angular material, angular animation

Npm install –save @angular/material

Npm install –save @angular/animations

1. Make changes to module to import angular material modules

// Angular Material modules

import {MaterialModule} from '@angular/material';

import {BrowserAnimationsModule} from '@angular/platform-browser/animations';

@NgModule({

imports: [

CommonModule,

MaterialModule,

BrowserAnimationsModule

]

1. Add the following in styles.scss

@import url('https://fonts.googleapis.com/icon?family=Material+Icons');

@import '~@angular/material/prebuilt-themes/deeppurple-amber.css';

<https://github.com/angular/angular-cli/wiki/stories-include-angular-material>

# REST APIs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operation | URL and Method | Headers | Request | Response |
| Get forms for a user/guest | GET  /api/forms/{includeMeta}  includeMeta – 1/0  If 1, send entire meta  If 0 , send formid, form name | **Request**:  Content-type : application/json  Accept : application/json  X-Auth-Header : username:<username>  **Response**:  X-Auth-Header : username:<username>  Cache-Control: private; max-age=3600 |  | **Success:**  200 OK  No username:  Return []  With username:  IForm[]  **Failure:**  400/500  Error code  Error message  In case of no user name, generate a random unique username and send in response header:  X-Auth-Header |
| Get form for a user/guest | GET  /api/form/:id | **Request**:  Content-type : application/json  Accept : application/json  X-Auth-Header : username:<username>  **Response**:  Cache-Control:  private; max-age=3600 |  | **Success:**  200 OK  No username:  Return {}  With username:  IForm  **Failure:**  400/500  Error code  Error message |
| Save form meta | POST  /api/form/:id | **Request**:  Content-type : application/json  Accept : application/json  X-Auth-Header : username:<username>  **Response**: | Form meta JSON | **Success**:  201 Created  200 OK  Return Form meta  **Failure:**  400/500  Error code  Error message |
| Save form data |  |  |  |  |
| Get form data |  |  |  |  |
| List submissions for a form |  |  |  |  |

## Guest Authentication

Server

**Client**

Auth Middleware

API Routes

Users

Local Storage/Cookie

### New

1. User clicks "My Forms"
2. Client reads local storage/cookie for guest id
3. Calls server API:/ api/forms. Passes the guest id (cookie/header)
4. Request reaches server.
5. Auth middleware receives the request (Auth middleware is configured before all API routes)
   1. Reads the userid (request.cookie or request.headers)
   2. If userid is blank, implies new user. So generate a random unique userid
   3. Save the new user in users collection in MongoDb
   4. Sets request.user = {user:<uerid>,isGuest:true/false,isSet:true/false}. Calls next()
   5. Sets the response.header/response.cookie
6. API route receives the request.
   1. uses the request.user object to execute its logic
7. Client receives the response. Reads the response.header/response.cookie and sets the user in local storage/cookie

### Existing

1. User clicks "My Forms"
2. Client reads local storage/cookie for guest id
3. Calls server API:/ api/forms. Passes the guest id (cookie/header)
4. Request reaches server.
5. Auth middleware receives the request (Auth middleware is configured before all API routes)
   1. Reads the userid (request.cookie or request.headers)
   2. Sets request.user = {user:<uerid>,isGuest:true/false,isSet:true/false}. Calls next()
   3. Sets the response.header/response.cookie
6. API route receives the request.
   1. uses the request.user object to execute its logic
7. Client receives the response. Reads the response.header/response.cookie and sets the user in local storage/cookie

# Angular Mysteries

1. Sencha package equivalent in Angular. How to create a reusable library in Angular?
2. Understand and POC for Angular Universal
3. Writing code comments

# 

# Workflow – User

Store User Submissions

Published Forms



# Workflow - Design

Save

Create Form

Form Builder

View Submissions

Edit Form

List Forms

Publish

My Forms

Landing Page