# Pluto Project Spec

## Overview

The objective is to create an enterprise architecture tool, which will create a conceptual model of an organization, all parts of an organization - business, data, technology, applications (BDAT).

This conceptual model is consist of **objects**, **attributes** and **relationships**. The model will then be viewed through dashboards, diagrams and views. **Dashboards** will be delivered using Apache Superset, **Diagrams** will be delivered using diagrams.net and **views** will be created as part of the project.

The model is structured using folders, in which the diagrams and objects will reside. Dashboards and views will not be stored in the folder structure but they will be accessed directly from a list.

## Content Examples

### Objects, Attributes & Relationships

**Object**: An entry in the database, aligned to the metamodel. For example an object aligned to the metamodel object type ‘Operating System’ might be called ‘Windows XP’

**Attributes**: An entry in the database, aligned to the metamodel AND an object. For example an object called ‘Windows XP’ might have a metamodel attribute type ‘Cost’. The attribute could then be ‘$5,000’

**Relationships**: A link between two objects, aligned to the metamodel and TWO objects. For example, Two objects ‘Windows XP’ and ‘MS SQL 2008 R2’ might exist in a model which has a defined relationship type ‘composition’. This would enable a relationship to be added ‘Windows XP is composed of MS SQL 2008 R2’, which could be viewed as ‘MS SQL 2008 R2 is part of Windows XP’



### Diagrams, Dashboards & Views

**Diagram**: A picture of objects, attributes and relationships. The diagram can have different notations, a notation is the style of the picture, for example the color of the box.

A green tiled wall and a sign

Description automatically generated

**View**: A configurable table for objects, attributes & Relationships. For example a view might be a simple table showing operating systems and costs. Or it could be a table showing operating systems, costs, start date and end dates, then you select an operating system to see any object with a ‘composition’ relationship from that operating system

**Dashboard**: An Apache Superset BI dashoard, using data from the database which updates live. For example a pie chart showing Operating Systems and their cost. When the data changes, the pie chart updates.

# Scope

## Deployment

The application needs to be delivered to customers using distinct URLs, for example [customer1].architech.live. Within each deployment, the application will have one **system administrator** and many users who could be either an **administrator** or a **user**.

## Users

The system administrator will provide access to all other users. The administrator will define the metamodel, manage the structure, roles and rights for users. Users can view dashboards, create folders, files, views, and diagrams.

**System Administrator**: Selects an administrator, defines technical setup of application (AD Integration settings, timeout settings, etc…)  
**Administrator**: One who will manage all the users, their roles and Permissions  
**User**: One who will view content and create diagrams

A system administrator is a default user in the application, the username cannot be changed and they have a default password until they login. The default password must be changed upon login and should be set to something like ‘password’. No-one can be added to the system until the administrator logs in for the first time. When a new user is added by the system administrator, that user gets an email, they click on the email and enter their first name, last name and password. The process for SSO is automated.

The login screen will include a password reset button, SSO this is not necessary. Each user can be added to a user group if desired, the user group is a group of users. This will simplify permissions to content.

## Log-In / Models:

The Login section will let the admin log into the system with their unique and secure credentials. They must enter email id and password to authenticate themselves (unless they use SSO).

When a user logs-in they are entered into a default model. There is a button under user settings which allows a drop-down menu to switch the user between any number of models to which they have been invited. There will be other features such as:

• A model is equivalent to a database

• When a model is created, it is created within a site

• A site is an instance of the application, this site is going to be deployed 1 per customer

• Information can never be accessible across sites

• Administrators see the complete list of models at all times and can manage users ability to see these models when these users login

• Each model has a default flag, accessible ONLY by administrators – a check box, when this is set, that model is set as the default model to login for UNLESS the user has previously specified a different default flag

• Each user can set their own default model, which would overwrite the default set by the administrators

• If the administrator has not set the default flag, it is set to the first model by alphabetical order

• ALL content – metamodel and model, belongs to one and ONLY ONE model. By switching between models, you see different content and potentially a different metamodel

• It is possible to create a copy of a model (metamodel and model content) if you are an administrator

• It is possible to move just the metamodel from one site to another, ‘migrate configuration’ which will APPEND the metamodel to the other model.

# Content

## Create Objects, Attributes & Relationships

Objects, Attributes & Relationships are created by any user, these users define their own permissions for other users, by default the objects adopt permissions from the folder in which they are created. Administrators always see all objects, attributes & relationships – they are stored in models and organized by folders.

## Create diagrams

Diagrams are created by any user, these users define their own permissions for other users, by default the diagrams adopt permissions from the folder in which they are created. Administrators always see all diagrams. Diagrams are stored in models and organized by folders.

## Create dashboard

The administrators can create different dashboards. Allowing users to see them according to the permissions. Administrators see all dashboards saved, they can edit the permissions on each dashboard to share it with a user or user group which makes it visible / invisible to that user when they select the dashboard tab.

# Minimum Viable Product (MVP)

## Overview

Dashboard will be delivered using Apache Superset, diagrams using diagrams.net. It will be necessary to change the diagram types available in diagrams.net to have some new diagrams (BPMN, UML, ArchiMate and a custom set of diagrams to be defined). These shapes will be linked to object types in the metamodel, so when a box is added it will CREATE an object reference inside the model. This linkage should be by some kind of ID on the shape type.

## High Level Features

### Metamodel Management (75% done)

Define object types, relationship types and attribute types. Currently this is done and working, but there are more requirements which have not been implemented such as selecting a currency for money values, custom drop down menus and more.

### CRUD Object / Attribute / Relationship (75% done)

Adding objects, attributes and relationships. This is done, but requires more refinement, primarily on the UI side.

### User Administration

The ability to add users aligned to the types: System Administrator, Administrator and User.

### API

It must be possible to use the REST API to import / export objects, object types, attributes, attribute types, relationships, relationship types. This is required for integrations to external systems.

### Security

A variety of security features such as data encryption at rest and in transit, the ability to restrict access to the application according to VPN connectivity, SSO and more.

### Environment Creation per Customer

The ability to create a different deployment of the application for each customer with subdomains to enable each customer to be separated.

### BI Integration

It must be possible to CRUD dashboards in Apache Superset and view these dashboards content inside of the ArchiTech application. There should be no need to open a new tab to see these dashboards. It can be necessary to run superset (in a new tab if necessary) to create dashboards

### Model Management

Change access to models, change default model for users, migrate configurations, other model based actions. A model is basically a database.

### View Structured Content

Create a ‘View’, this could be a list of objects with attributes and / or relationships. The view is similar to a diagram, it is stored and saved so it can be re-run and it shows objects, relationships and attributes in the model.

### Folder / Storage Location Management

CRUD folders. Move content (objects, diagrams) between folders.

### CRUD Diagram - Linked to Model

The ability to CRUD diagrams in diagrams.net, with the ability to drag / drop objects from the model into the diagram and match new shapes by name to the objects in the model. Ability to view / match relationships (lines) in diagrams to relationships in the model.

### ML / AI / Social

Newsfeed showing activity inside the model (new diagrams, new dashboards, etc…)

### Feedback / Commenting

Ability to store a comment against a diagram for all users to see.

## Detailed Level Features

### Diagram Navigation

The ability to link diagrams together, so it will be possible to click a button to go directly from one diagram to another diagram

### Hard / Soft Delete Object

When an object is deleted it can either be deleted from the diagram / folder and left inside the model OR it can be deleted from the database altogether

### Heatmap Diagram

When viewing a diagram, select objects within the diagram and see attributes, these attributes can then be used to color the diagram according to these attribute values.

### State Modeling

Enable relationships to be exposed through attributes, objects should be viewable through an attribute drop down menu. The results of this attribute can then be viewed in a custom built report.

### Reuse Object

If an object name is entered, and it already exists there should be the option to NOT add a new object, but rather re-use the object. Re-use could mean 1) in a diagram that the object is referenced, 2) The object is linked to a new folder / view but not duplicated. Re-used objects are just references to the existing object.

### Basic Search

Ability to search for objects and diagrams based on name, with partial matches, not case sensitive

### Change Control – Collaboration

Prevent two users from simultaneously editing the attributes for an object.

# Datasource

## Data Refresh

**Diagrams**:1-2 seconds. Example when the name of an object changes, the diagram must be updated to display this new object name within 1-2 seconds.

**Dashboards**: 10-20 seconds

**Objects / Views**: Instant, updates can be made directly against this content

# Content

## Amount of Content

The solution must be able to store the below without any performance decrease:

**Objects, relationships & Attributes**: > 500,000 (average 5,000)

**Diagrams**: >10,000 (average 100)

**Dashboards**: >500 (average 10)

**Views**: >200 (Average 50)

## Technology Stack

This application will support web browser as mentioned in the table below:

OS/Browser: Chrome Firefox Safari

Windows: 74+ 66+ Not supported

Mac: 74+ Not supported 12.1+

We will use the below technology for the website and back - end development:

1 Website: Angular, Canvas, HTML5, SCSS, TypeScript.

2 Webservices: Java or C#, Elastic Search, WebSocket.io

3 Database: MySQL Server.

The application will be able integrate with other 3rd party platform

Testing Environment

• The Application will be tested and supported on Chrome 74+ (in windows and Mac OS), Safari 12.1+ (in Mac OS) and Firefox 66+ (in windows OS).

Hosting:

Hosting - Cloud based Package Configuration