# TedsRate Documentation 3.0 (April 26, 2016)

## Table of Contents

# Introduction

## Abstract

The primary goals of version three were to extend the functionality of the rating system to allow for addition data: demographics, comments and screenshots to be attached to assessments. In order for this to work efficiently the data schema had to be refactored. Along with the addition of tables for storing demographic questions and their associations, the userRating table was removed and replaced with an assessment table and a set of configuration tables. This allowed for multiple users' assessments to be tied to the same layout simplifying the admin system and reducing data redundancy.

Currently, a second level of grouping is being applied to configurations. these new tables will group configurations to be used with the welcome.php page and at the end screen of assessments so users can quickly see their progress across assessments, move between them and, when giving away gift cards, incentivize the user to continue.

After the development of the new rating system, the focus of the project shifted to include real-time reporting features for the admin side so that we could track the effectiveness of a project immediately and provide data to clients quickly.

## Project Information

Developer – William Menten-Weil

Github Repository:

Development Environment: LAMP stack

- Linux – Ubuntu 16.04 LTS
- Apache2
- MySQL – 5.9, InnoDB tables
- PHP – 5.5 & 7.04

Production Environment: LAMP Stack

- Apache – 2.2.31
- MySQL – 5.5.18
- PHP - 5.6.18

# Server-side – PHP

## Overview

The PHP components of this project can be broken down into two subsections, the vanilla functional scripts and the object-oriented CI (Code Igniter) models and controllers. Tedsrate 3.0 has also sought to move towards a client-server architecture where new AngularJS services communicate with the REST and internal API's. This has helped to maximize our limited system resources by placing more of the computing on the client and limiting redundant data retrieval.

## Vanilla

There are three legacy primary files, only adminproc is still in use:

- Rater.php – displayed the assessment form to the client.
- Process.php – processes a partial or completed assessment form from rater.php.
- Adminproc.php - handles login and logout for the admin site. Formerly, it handled numerous other admin site services such as adding of project, scenarios, artifacts, and personas.

There are also four new primary files:

- Start.php - handles the routing logic for generating new users and assessments for a given configuration.
- Assessment.php - a misnomer, it contains the Angular template for issuing the assessments and no longer contains any PHP code.
- Upload.php - runs the screenshot uploads to the server and provides feedback of success to the client.
- Welcome.php – this file is currently a work in progress. It will be the landing page for configuration groups so that raters can easily navigate between configurations and so we can better tailor the landing page for our advertisements.

### Other Services

Along with the 5 primary vanilla files are a handful of smaller services which are either not in use or being phased out.

Those being phased out:

- Vanilla functional models – these are a directory of entity models and their parent routers. Currently they are used in some of the operations for the Angular assessment app.
- PHP includes – now that the project is centralized into two Angular applications there is no need for include files as the two application require different headers and footers.

Those not in use:

- Database modification scripts – the files addMD5.php and convertAssessments are files which were used to adapt old data into the current database schema.
- Legacy Rating system – While still functional, the rater.php, process.php and UAProxy.php system has been succeeded by the assessment.php system.
- Ajax_service – This file used to be the target of all Ajax connections from the admin site. Now it remains because, while it is not in use currently it hold the template for issuing mail from the Tedsrate mailer.

## Code Igniter

The use of the Code Igniter framework was a late addition in the development of the new version of Tedsrate. However, it proved to be a more efficient method to interact with the data layer as it not only provided the same functionality as in the vanilla PHP models, but also allowed for the creation of a REST API.

### Rest API

The REST API comprises the majority of the CI components. It is the primary means of interacting with the database and allows for CRUD operations on all entities in our data schema.

### Internal API

The internal API handles specific process such as getting project overviews and generating report tables.

# Client-side – JS

## Overview

The client-side components of Tedsrate have gain increased significance in version three. While there still exists some vanilla legacy JS components the majority of the client side logic is contained in two Angular applications one for administration and one for assessments.

## Vanilla

Version three has left only one file of vanilla javascript, notice.js this is a small script for parsing the query string for notices regarding login and authorization access for the admin application.

## AngularJS

The majority of the client logic is stored in two Angular applications, Assessment.js and Admin.js

### Assessment.js

This application contains all the logic for a user to run an assessment. Because it is consumer facing it is the highest priority for

maintenance. It has notable features such as auto-saving, progress tracking, rerouting of the user in the case of assessment-user mismatch and screenshot uploading with progress feedback. It has templating to account for the various different UI options which can be specified in the configuration.

### Admin.js

The admin application is the second of the two angular apps. Currently it supports the creation of project and their associated entities: artifacts, scenarios, personas, roles, configurations and assessments. It also allows administrators to view the results of assessments in the form of pivot tables for average and std. dev. Information and bar graphs for demographic question distributions. These stats are accessible in three groupings: grouped by scenario for an artifact, grouped by artifact across a scenario and grouped by assessment across a configuration. While the first two groupings provide aggregate data for each cell the third grouping allows for a granular view of each individual. This can aid in the identification of bad or suspect data.

### Report.js

Report is a third minor application that is exposed to outside parties. This app allows them to view reports without having access to other admin functionalities.

### tedsModels.js

This file contains the services used to connect with the vanilla models and CI APIs on the server for both Angular apps.

### Directives

The project has a few custom directives: dropdown, filterList and pivotTable, Dropdown is used in the current display of projects for the admin application. FilterList and pivotTable are used to format and present project data.

## Third-party

Bootstrap.js – required for interactive ui components, eg. modals, popovers, tooltips, collapse.

Jquery.1.11.0.js – used for interaction in legacy files

D3.min.js – framework for DOM manipulation regarding the display of data driven components

Nv.d3.js – framework for charting on top of d3

### Angular Modules

UI-Bootstrap – Converts bootstrap.js components into angular directives

UI-Validate – improved form validation directives

UI-Router – Allows for routing of URI in the admin application

UI-Grid – Used to provide highly configurable data tables.

UI-Select – Provides search-multi-select boxes to be used in data table filtration.

Cookies – Provides access to cookies within the angular applications

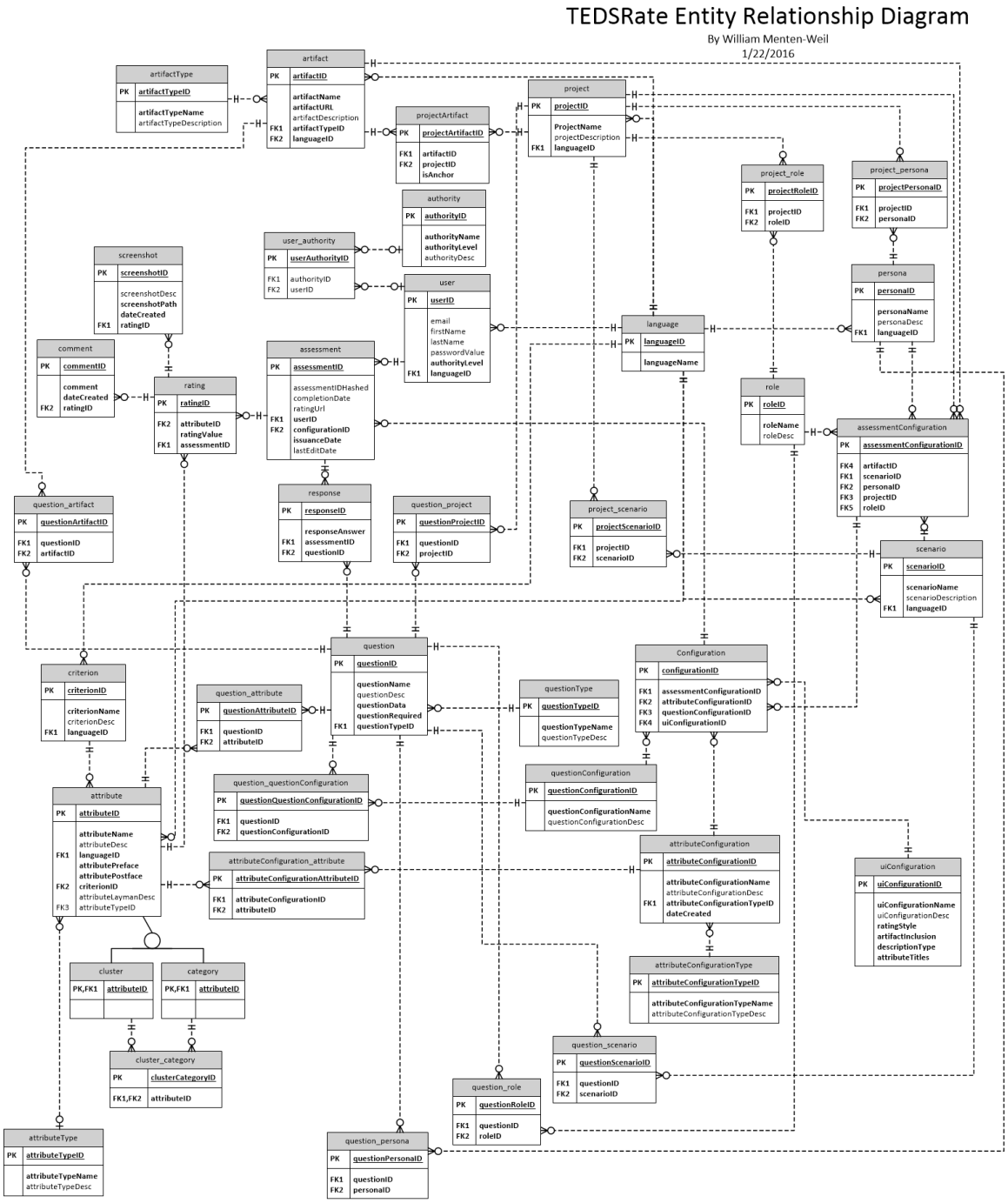
Animate – provides animations for angular events

Bootstrap-lightbox – Lightbox directive

Elastic – dynamically sized textarea directive

Nvd3 – nvd3 charting components converted into angular directives

# MySQL
## Current ERD



# Table Schema

## Project

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| projectID | int(11) | No | *None* | AUTO_INCREMENT |
| projectName | varchar(255) | No | *None* | |
| projectDescription | varchar(255) | Yes | *NULL* | Should be ‘Desc’ |
| LanguageID | int(11) | No | 1 | |

## Artifact

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| artifactID | int(11) | No | *None* | AUTO_INCREMENT |
| artifactName | varchar(255) | No | *None* | |
| artifactDescription | varchar(255) | Yes | *NULL* | Should be ‘Desc’ |
| languageID | int(11) | No | 1 | |

## Scenario

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| scenarioID | int(11) | No | *None* | AUTO_INCREMENT |
| scenarioName | varchar(255) | No | *None* | |
| scenarioDescription | varchar(255) | No | *None* | Should be ‘Desc’ |
| languageID | int(11) | No | 1 | |

## Persona

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| personaID | int(11) | No | *None* | AUTO_INCREMENT |
| personaName | varchar(255) | No | *None* | |
| personaDesc | varchar(255) | No | *None* | |
| languageID | int(11) | No | 1 | |

## Role

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| roleID | int(11) | No | *None* | AUTO_INCREMENT |
| roleName | varchar(255) | No | *None* | |
| roleDesc | varchar(255) | No | *None* | |
| languageID | int(11) | No | 1 | |

## Configuration

The configuration entity serves as a container for four configuration modules: attribute, assessment, question and UI. It also provides an obscured ID in the form of a hash, so new users and assessment can be generated via

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| configurationID | int(11) | No | *None* | AUTO_INCREMENT |
| configurationIDHashed | varchar(64) | Yes | *NULL* | |
| attributeConfigurationID | int(11) | No | *None* | |
| assessmentConfigurationID | int(11) | No | *None* | |
| questionConfigurationID | int(11) | No | *None* | |
| uiConfigurationID | int(11) | No | *None* | |

the start.php script.

The attribute configuration module allows administrators to specify the selection of attributes used in the configuration. The options for this include the 40 original categories and the 12 compressed clusters.

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| attributeConfigurationID | int(11) | No | *None* | AUTO_INCREMENT |
| attributeConfigurationName | varchar(255) | No | *None* | |
| attributeConfigurationDesc | varchar(500) | Yes | *NULL* | |
| attributeConfigurationTypeID | int(11) | No | *None* | |
| dateCreated | timestamp | No | *CURRENT_TIMESTAMP* | |

The assessment configuration module specifies the keys which define the study: the project, artifact (usually a website or mobile app), scenario, persona and role.

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| assessmentConfigurationID | int(11) | No | *None* | AUTO_INCREMENT |
| projectID | int(11) | No | *None* | |
| artifactID | int(11) | No | *None* | |
| scenarioID | int(11) | No | *None* | |
| personaID | int(11) | No | *None* | |
| roleID | int(11) | No | *None* | |

The question configuration module serves as a target to associate with a group of questions. These can then be used throughout the project on multiple configurations or even across projects.

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| questionConfigurationID | int(11) | No | *None* | AUTO_INCREMENT |
| questionConfigurationName | varchar(255) | No | *None* | |
| questionConfigurationDesc | varchar(500) | Yes | *NULL* | |

The UI configuration module specifies the specific UI tweaks for the given configuration: The style of the rating input mechanism (Likert Scale or

Textbox), whether or not to include the artifact, the type of description to use with each attribute and if attribute names should be displayed.

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| uiConfigurationID | int(11) | No | *None* | AUTO_INCREMENT |
| uiConfigurationName | varchar(255) | No | *None* | |
| uiConfigurationDesc | varchar(500) | Yes | *NULL* | |
| ratingStyle | varchar(100) | No | *None* | |
| artifactInclustion | varchar(100) | No | *None* | |
| descriptionType | varchar(100) | No | *None* | |
| attributeTitles | int(11) | No | *None* | |

## Assessment

Assessment serves as the reference point to bundle ratings responses, comments and screenshots. It contains a hashed form of its ID to be exposed in URIs for assessment.php. It also hold relevant date information.

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| assessmentID | int(11) | No | *None* | AUTO_INCREMENT |
| assessmentIDHashed | varchar(64) | Yes | *NULL* | |
| configurationID | int(11) | No | *None* | |
| userID | int(11) | No | *None* | |
| completionDate | datetime | Yes | *NULL* | |
| issuanceDate | datetime | No | *CURRENT_TIMESTAMP* | |
| lastEditDate | datetime | Yes | *NULL* | |
| ratingUrl | varchar(2083) | Yes | *NULL* | Backward Compat |

## Attribute

Attribute specifies the name, type (depreciated), parent criterion and descriptions for the Teds model attribute/category/cluster. There are two descriptions, one academic (attributeDesc) and one simple (attributeLaymanDesc). It also hold the specific preface and postface to be used with the attribute if the UI dictates they are to be included.

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| attributeID | int(11) | No | *None* | AUTO_INCREMENT |

| | | | |
|---|---|---|---|
| attributeName | varchar(255) | No | *None* |
| criterionID | int(11) | Yes | *NULL* |
| languageID | int(11) | No | *None* |
| attributeDesc | varchar(5000) | Yes | *NULL* |
| attributePreface | varchar(100) | No | *'Below Expectations'* |
| attributePostface | varchar(100) | No | *'Above Expectations'* |
| attributeTypeID | int(11) | Yes | *NULL* |
| attributeLaymanDesc | varchar(5000) | Yes | *NULL* |

## Rating

Rating stores the rating value of a single attribute for a given assessment. It can have attached comments and screenshots

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| ratingID | int(11) | No | *None* | AUTO_INCREMENT |
| assessmentID | int(11) | No | *None* | |
| ratingValue | decimal(11,1) | Yes | *NULL* | |
| attributeID | int(11) | Yes | *NULL* | |

## Comment

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| commentID | int(11) | No | *None* | AUTO_INCREMENT |
| comment | varchar(64) | Yes | *NULL* | |
| ratingID | int(11) | No | *None* | |
| dateCreated | datetime | No | *CURRENT_TIMESTAMP* | |

## Screenshot

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| screenshotID | int(11) | No | *None* | AUTO_INCREMENT |
| screenshotPath | varchar(1000) | No | *None* | |

| | | | | |
|---|---|---|---|---|
| screenshotDesc | text | Yes | *NULL* | Should be varchar |
| ratingID | int(11) | No | *None* | |
| dateCreated | datetime | No | *CURRENT_TIMESTAMP* | |

## Question

Question holds the information for a demographic questions. This includes the question name, description, requirement status, type (demographic, project, artifact, scenario, persona, role) and the question JSON data. This JSON data specifies the input type of the question as well as the options for the input.

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| questionID | int(11) | No | *None* | AUTO_INCREMENT |
| questionName | varchar(1000) | No | *None* | |
| questionDesc | varchar(2000) | Yes | *NULL* | |
| questionData | varchar(17000) | No | *None* | |
| questionTypeID | int(11) | No | *None* | |
| questionRequired | tinyint(1) | No | *None* | |

## Response

Response stores the answer to a given question for an assessment.

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| responseID | int(11) | No | *None* | AUTO_INCREMENT |
| responseAnswer | varchar(20000) | No | *None* | |
| questionID | int(11) | No | *None* | |
| assessmentID | int(11) | No | *None* | |

## User

User stores the personally identifying information such as email, first name last name and password. It also contains depreciated versions of a user's authority level. The correct way to do this is through the user_authority associative entity.

| Column | Type | Null | Default | Comments |
|---|---|---|---|---|
| userID | int(11) | No | *None* | AUTO_INCREMENT |
| Email | varchar(100) | Yes | *NULL* | |

| | | | | |
|---|---|---|---|---|
| firstName | varchar(45) | Yes | *NULL* | |
| lastName | varchar(45) | Yes | *NULL* | |
| languageID | int(11) | No | *None* | |
| passwordValue | varchar(100) | Yes | *NULL* | Exposed password values |
| authorityLevel | int(11) | No | *1* | Backward Compat |
| authorityID | int(11) | Yes | *NULL* | Backward Compat |

## Associative Entities

The current schema contains many associative entities. They are donated through the use of an underscore '_' in their name. Below you will find a list of associative entity pairs grouped by the primary member of the relationship.

| **Parent** | **Children** |
|---|---|
| Project | Artifact, Scenario, Persona, Role |
| Question | Project, Artifact, Scenario, Persona, Role, Attribute |
| QuestionConfiguration | Question |
| Cluster | Category |
| AttributeConfiguration | Attribute |
| User | Authority |

# Future Development Notice

There are a few main targets for future development.

- Move the project off of the UW shared hosting to a destination where we have greater control over the server and its software. This could be AWS or a personal box at UW.  This would allow us to upgrade the Mysql and unlock the InnoDB engine so we can use transactions and prevent data anomalies.
- Upgrade the PHP version to 7.x and gain the increased speed from version seven’s improvements. (Will happen on UW servers Oct. 2016)
- Assess the feasibility of converting the Angular applications into Angular 2 as it is significantly faster.
- Continue to refactor our vanilla PHP logic and our angular data storage to optimize page loads and reduce bandwidth needs on the server.
- Once the data schema is stable, convert the API models to use stored procedures.