CSAL Database Project Introduction

Craig Kelly

October 27, 2014

Contents

1 Introduction **2**

2 Use Cases **2**

2.1 Direct Logging 2

2.2 Logging via ReST Endpoint 2

2.3 Teacher Status Check 2

2.4 Data Administration 2

3 Database **2**

3.1 Class 3

3.2 Student 3

3.3 Lesson 3

3.4 Student Actions 3

4 CSALMongo DLL **3**

5 CSALMongo Web API **4**

5.1 ReST API 4

5.2 User Interface 4

5.3 Authentication 4

5.4 Logging 4

5.5 Deployment 5

6 Sequence Diagrams **5**

6.1 Example of User Interaction 5

1 Introduction

This repository contains code for storing and displaying data stored as part of the CSAL project. The use cases and the code produced are described below. The short version is that the data is stored in a MongoDB instance, there is a C# library for accessing the database, and there is a Web API wrapping the DLL. In addition, the Web API server provides a very simple ASP.NET MVC application for viewing the data. Because of the dependence of the Web API on the "core" library, the only deployment information is below in 5.5 Deployment

2 Use Cases

2.1 Direct Logging

Services running on the same server as the database (i.e. ACE) want the ability to write JSON data log entries without POST'ing to an HTTP endpoint. Note that while the architecture of the project allows this use case (and it is currently used in production), a safe alternative would be to force all applications to write via the ReST API (see 2.2 Logging via ReST Endpoint).

2.2 Logging via ReST Endpoint

Applications may post a JSON record representing an ACE turn to a public endpoint for persisting in the database. Note that this is the recommended way to save data to the database (contrast with 2.1 Direct Logging)

2.3 Teacher Status Check

Teachers need to be able to see students' progress in the lesson. There should be a way to see how the entire class is doing, how the class is doing on a lesson, how a student is doing across the lessons, and how a specified student is doing on a specific lesson. Note that if this application grows, this Use Case should be broken out into small chunks

2.4 Data Administration

There must be a way for administrators (not teachers) to initialize and edit the database. Specifically, class, lesson, and student data should be configured for expected logging (via either use case 2.1 Direct Logging or 2.2 Logging via ReST Endpoint).

3 Database

The main documentation for the JSON logging record is available in a Google doc. Please see the documents "CSAL Data" at `https://docs.google.com/document/d/19nJZMReWpTat_tjNeOvA7oa8rDD0KmQ5oKaHnn2HmwE` and "AutoTutor Conversation Engine (ACE) Web API (CORS Version)" at `https://docs.google.com/document/d/1ZRlj7e5u4PQSlggCD--yZ5EsB2KCZzt7P8MEI6HB9So`

The various database entities are described below. The JSON logging data described above is stored in the entity described in 3.4 Student Actions . You may also see how the C# classes for this data (both the JSON

logging format and the database entities below) are structured by looking in the CSALMongo project or the CSALMongo.chm compiled documentation in this directory.

The server is autotutor.x-in-y.com and the MongoDB database should be named csaldata. There are four collections: classes, lessons, students, and studentActions.

3.1 Class

There is one document in this collection per class. It contains a list of the students in the class and the lessons used. Please see 3.4 Student Actions for details on auto-creation and updating.

3.2 Student

There is one document in this collection per student. Please see 3.4 Student Actions for details on auto-creation and updating.

3.3 Lesson

There is one document in this collection per lesson. Please see 3.4 Student Actions for details on auto-creation and updating.

3.4 Student Actions

There is one document in this collection per student per lesson. Any time a JSON logging record is saved for a student in a lesson, it is appended to the Turns list in the corresponding document in this collection. Although it is preferred to have the class, lesson, and student documents matching this information pre-populated, a minimal version of each entity will be created if it is not present when the data is logged. Auto-created entities will have a property named AutoCreated set to true.

To help with queries, we also update the class, student, and lesson documents when we receive turn data like so:

- 3.1 Class - update the fields Students and Lessons
- 3.3 Lesson - update the fields LastTurnTime, Students, AttemptTimes, StudentsAttempted, StudentsCompleted, and URLs
- 3.2 Student - update the fields LastTurnTime and TurnCount

4 CSALMongo DLL

The "base" or "core" DLL contains the model classes for JSON logging format, the model classes for the database, the actual database interface class, and some supporting code. The project is documented via XML documentation which has been compiled into the CSALMongo.chm in this directory.

There is also a unit test project named CSALMongoUnitTest. It uses the Unit Testing facilities availble with Visual Studio 2013. The tests are broken into five categories of testing:

1. Model Testing for methods added the model classes for parsing, information, etc.

2. Database Operations Testing for actual database operations exposed by the main class
3. Database Utility Testing for helper or utility methods exposed by the main database class
4. Utility Testing for helper or utility functions **outside** the main database class.

5 CSALMongo Web API

5.1 ReST API

The ReST API is exposed via a .NET Web Api project (that also houses a GUI - see 5.2 User Interface).

Since this is a ReST API, there is a URL namespace complete with expected verbs and payloads. They are documented below. It should be assumed that for local workstation testing the url would begin with `http://localhost:62702`. For the production URL, the proper prefix would be `http://autotutor.x-in-y.com/csaldb`.

1. TODO

5.2 User Interface

The User Interface is an ASP.NET MVC web application, served by the Home controller class, rendered via Razor. It uses jQuery and Bootstrap for UI automation and styling. Various jQuery UI plugins are also used (notably DataTables and Sparklines).

The application maintains a URL namespace similar to the ReST API, but under the Home directory. As above, It should be assumed that for local workstation testing the url would begin with `http://localhost:62702`. For the production URL, the proper prefix would be `http://autotutor.x-in-y.com/csaldb`.

1. TODO

5.3 Authentication

Authentication is handled via Google OAuth2. Administrators are identified by email address in the web.config file. Teachers are given access to class information if the address from their OAuth2 profile matches the teacher name for the class.

5.4 Logging

This document generally refers to logging to identify ACE turn records sent in JSON format and stored in the MongoDB collection studentActions. However, the Web API and GUI must log records as well. Currently this is fairly simple; in addition to default IIS logging, the Elmah library is used to log unhandled exceptions and any errors displayed via the custom user error page. The Elmah log is only stored in-memory (and so is transient) and can be accessed at the main URL at `http://autotutor-x-in-y.com/csaldb/elmah`.

5.5 Deployment

The entire application should be deployed via Visual Studio packaging and IIS application import. Please note that the application is already deployed to its own App Pool on the production server, so a new deployment should be an overwrite (not a new application).

Any deployment should also be accompanied by an annotated tag in the Git repository.

6 Sequence Diagrams

6.1 Example of User Interaction