ABET General System Documentation

By JMRJM and Computer Science Goons

Compiled by Logan Falkensten

Table of Contents:

Foreword:

This system was designed with efficiency and convenience in mind. It has gone through many changes and has resulted in a design that while solid, was too much for us to feasibly implement in the time provided and with the problems that occured within the semester.

It is encouraged that whoever takes up the task of completing this project review this design document thoroughly and completely, as much of our design philosophy as well as how the general systems work given the code provided to you is contained in here, as well as in our main requirements document and the original Design document.

As a last note, technically nothing is set in stone, should you opt to rebuild parts of or the entire system it is welcome. However, that may cause issues due to some software choices used as well as how many necessities the system needs despite our progress, as well as just how large the system is in general compared to most capstone projects. This document is here for your convenience and to assist with getting up to speed on where we left off, what we were thinking, and how we were implementing things.

General System Expectations/Requirements:

This system has a multitude of expectations for it, and all of the functionalities on record will be listed here, big or small, and implemented or non-implemented. These should be clarified and expanded upon when the project is picked up, as undoubtedly thoughts will change and things might be added or removed.

-The system is less confusing to use, namely in the UI elements and UI design of the system. The system should be able to be understood relatively easily by someone who has never seen it before. This comes in the form of overhauling every element of the UI to be up to modern standards, and bringing the system more in line with the mainline UNT systems. Heuristics should be employed, as well as proper user testing of finalized pages and designs.

-The database is modular and sustainable. The original database had a multitude of bad practices associated with it, and as such none of it was salvageable. The new build must be modular in creation, must be sustainable for a potentially never ending increase in semester data.

-The system must have a roles system that can allocate roles to users. The defined roles were: Admin, Coordinator, Professor, TA, and Student. The system should be properly integrated with the UNT system in this respect, utilizing EUID to define user roles and subsequentially what pages they have permission to access, and how they are able to interact with the system.

-The user must be able to login with their EUID, and the login should be querying the UNT server for access granting. In turn with this, proper security measures should be in place on login and throughout the system to ensure that a user does not get access to pages and data that they shouldn't, as well as locking out users who attempt to access the site in an improper way.

-Professors should be able to comment on resulting data, and upload files to the database as proof of outcomes taught. In turn these files need to be able to be downloaded for viewing from the database, and the datasets themselves need to be accessible by permissible users.

-Microsoft Access should be used to handle the database storage. In our planning, this was the one piece of software that was heavily desired and requested be kept from the original system.

-The system must have a variety of filters for admins to use in order to find specific data types quickly, and view specific sections in a timely manner.

General System Design:

This section covers the wide view of the system. No specifics given.

The system's general design revolves around reducing the visual clutter, bringing the website visually in line with other UNT websites and systems, and making the website more sustainable and modular. The current/previous website does not meet a number

of these standards, on top of having some potential security flaws and vulnerabilities due to the ambiguity in how it gains access to EUID usernames and passwords.

The login and logout functionality in the system is updated to be able to handle EUID credential handling through the UNT LDAP server, which assists massively with security handling.

The roles within the website are intended to determine what landing page a user has access to based upon their role, as well as what pages and permissions they can do.

The database is rebuilt from the ground up to be modular and more sustainable than the current implementation. It currently uses Microsoft Access as the storage handler. Specifics will be elaborated upon in the database section.

We built the system utilizing the Angular framework, Angular typescript 8 to be exact. It is a full-stack development suite that utilizes many packages similar to python in order to accomplish more with less coding. It is strongly recommended to continue using the software, though it does have a learning curve to it. You can find how to set up the software, as well as some extensive documentation and learning tools for the system here: https://angular.io/docs.

Login/Logout:

The login and logout system is fairly straightforward: Allow users to sign in to the ABET system using their EUID. This is a critical requirement, and as such requires a higher level of security. The real strain is getting the proper authentication setup to handle users.

Because this system handles confidential information such as student feedback, profesor feedback, analytics, and grade aggregates, as well as creating semesters and classes and putting faculty in the system, it is critical that security access be up to UNT requirements. This is why we interface with the UNT LDAP server for security authentication, so that the UNT servers have a record of all access requests for EUID’s, and can authenticate each request itself, saving us having to develop an entirely new set of security protocols.

See the following or the separate file in the Documentation folder for Login security setup.

Authors: Elton Wee + James Durflinger
Information Courtesy of Jim Byford. Would strongly recommend contacting him when starting this project regardless.

NetIQ eDirectory

There are two eDirectory databases -- IDTREE and AUTHTREE. Use AUTHTREE for simple authentication needs (euid, password, employee/student status). More complex needs may require access to the IDTREE database.

Search Base: ou=people,o=unt

IDTREE: ldaps://ldap-id.untsystem.edu (load-balanced across both datacenters)

IMPORTANT: Clients within a datacenter **MUST** use their local LDAP service.

- GAB datacenter clients **must** use ldaps://ldap-id.gabdcn.unt.edu
- DP datacenter clients **must** use ldaps://ldap-id.dpdcn.unt.edu

AUTHTREE: ldaps://ldap-auth.untsystem.edu (load-balanced across both datacenters)

IMPORTANT: Clients within a datacenter **MUST** use their local LDAP service.

- GAB datacenter clients **must** use ldaps://ldap-auth.gabdcn.unt.edu
- DP datacenter clients **must** use ldaps://ldap-auth.dpdcn.unt.edu

Network Encryption Required

You are required to use a secure (encrypted) network connection. The IDTREE and AUTHTREE server farms use TLS certificates signed by the InCommon certificate authority. Windows, Mac, some Linux/UNIX systems (add the ca-certificates package), and most distributions of the Java runtime should already have the required self-signed "root" certificate.

If needed, you can obtain the InCommon (a.k.a. "Sectigo", "Comodo") self-signed "root" certificate from the InCommon web site. (Hint: You want the "USERTrust Secure" certificate).

Popular IDTREE/AUTHTREE attributes

- euid
- emplid
- sn

- givenname
- mail (not available from AUTHTREE)
- eduPersonScopedAffiliation

NOTE: Other attributes are available, but often require consultation to understand the limits of the available data and appropriate uses.

Microsoft Active Directory

UNT System maintains several Active Directory domains, all within the same Active Directory forest.

DNS domain name	LDAP Search Base	LDAP Connection URL	Use
ad.unt.edu	DC=ad,DC=unt,DC=edu	ldaps://ad.unt.edu:636	System-wide IT Services
its.ad.unt.edu	DC=its,DC=ad,DC=unt,DC=edu	ldaps://its.ad.unt.edu:636	System-wide IT Services
unt.ad.unt.edu	DC=unt,DC=ad,DC=unt,DC=edu	ldaps://unt.ad.unt.edu:636	Denton / Dallas / System employees
hsc.ad.unt.edu	DC=hsc,DC=ad,DC=unt,DC=edu	ldaps://hsc.ad.unt.edu:636	HSC employees
dallas.ad.unt.edu	DC=dallas,DC=ad,DC=unt,DC=edu	ldaps://dallas.ad.unt.edu:636	Abandoned, formerly Dallas employees
students.ad.unt.edu	DC=students,DC=ad,DC=unt,DC=edu	ldaps://students.ad.unt.edu:636	Denton / Dallas / HSC students

Please notice how the LDAP search base matches the fully-qualified DNS name of the domain. When you see the LDAP distinguished name for a user object, You can tell which domain it came from by looking at the end of the distinguished name.

Network Encryption Required

You are required to use a secure (encrypted) network connection. All our Active Directory domain controllers offer LDAP service over TLS. The certificates used for TLS are signed by the Active Directory forest's own certificate authority (not InCommon). Windows computers joined to any of our AD domains should already hold a copy of the required certificates. If your LDAP client is connecting from a computer that is not joined to one of our Active Directory domains, then you may need to import the following X.509 certificate into your application or operating system certificate database (sometimes called a "keystore").

TIP: Choose "binary" for Windows. Choose "text" for everything else (copy the text from your web browser into a plain text file, including the begin/end lines).

[binary] [text] "UNT AD Root CA" (self-signed "root" certificate, signed using SHA-256 (a.k.a. "SHA-2") hash algorithm)

Popular Active Directory attributes:

- employeeID - holds the EUID
- employeeNumber - holds the EMPLID
- sn
- givenname
- mail
- telephoneNumber
- eduPersonScopedAffiliation

NOTE: Other attributes are available, but often require consultation to understand the limits of the available data and appropriate uses.

Active Directory Global Catalog

Active Directory consists of separate directory databases called "domains". While there is some "trust" involved amongst the domains inside a forest, it is not possible to connect to one domain via LDAP to query another domain. This is just how Microsoft designed Active Directory. With the LDAP protocol, you must connect to the domain you wish to query.

If you have a need to query all domains in the forest, Microsoft provides a "global catalog" service. Use the global catalog **only when needed** to query all AD domains in our AD forest. The global catalog is a separate LDAP service that handles LDAP over TLS on TCP port 3269. Some attributes from the Active Directory domains are **not** stored in the global catalog.

IMPORTANT:

- The TCP port number for the AD global catalog is **3269** (not 636).
- The LDAP search base is always an **empty string**.
- A DNS query for **gc._msdcs.ad.unt.edu** returns a list of servers running the global catalog service (then do a reverse DNS lookup on the IP address and **use the DNS name of that server** for TLS to work properly)

WARNING: Active Directory tends to be case-sensitive with some things like attribute names where a normal LDAP server would not care.

LDAP Encryption

This section should help you to properly configure your LDAP clients for TLS encryption. There is no valid excuse for not using encryption or not configuring it properly. If your system is not documented here, we are happy to work with you.

TLS encryption

Transport Layer Security (TLS) is a newer and more secure version of SSL. With TLS, it is possible (but not required) to initiate a non-encrypted connection and then issue a

"STARTTLS" command to begin encryption. TLS can also operate with encryption from the beginning just like SSL did. TLS does **not** require you to start without encryption.

TLS vulnerabilities

By now, you should be using TLS 1.3. All systems should be configured to **ALLOW ONLY TLS 1.2 or higher** due to security weaknesses in earlier versions.

OpenLDAP

OpenLDAP may use one of two TLS code libraries -- OpenSSL or GNUTLS. In either case, you must configure encryption options in the OpenLDAP **ldap.conf** (LDAP client configuration) file. The **TLS_CACERT** directive tells OpenLDAP the location of your trusted root certificates (all trusted roots in a single file).

Distribution	Location of ldap.conf	Location of Trusted Certificates File	TLS library	Certs package
Debian	/etc/ldap/ldap.conf	TLS_CACERT /etc/ssl/certs/ca-certificates.crt	GNUTLS	ca-certificates
Ubuntu	/etc/ldap/ldap.conf	TLS_CACERT /etc/ssl/certs/ca-certificates.crt	GNUTLS	ca-certificates
Fedora	/etc/openldap/ldap.conf	TLS_CACERT /etc/ssl/certs/ca-bundle.crt	GNUTLS	ca-certificates
CentOS	/etc/openldap/ldap.conf	TLS_CACERT /etc/pki/tls/certs/ca-bundle.crt	GNUTLS	ca-certificates
RedHat	/etc/openldap/ldap.conf	TLS_CACERT /etc/pki/tls/certs/ca-bundle.crt	GNUTLS	ca-certificates
SuSE	/etc/openldap/ldap.conf	TLS_CACERTDIR /etc/ssl/certs	OpenSSL	openssl-certs

With Windows, C:\ldap.conf (PHP >= 5.3.1) or C:\OpenLDAP\sysconf\ldap.conf (earlier PHP versions) are the typical locations for this file.

RedHat / CentOS / Fedora

Copy the certificate authority's self-signed ("root") certificate to the /etc/pki/ca-trust/source/anchors/ directory and issue the update-ca-trust export command.

Java

For all implementations of LDAP based upon Java, you add trusted root certificate to the Java "keystore" (a file named **cacerts**). The location of the cacerts file varies and there is often more than one copy on the computer. You must install the correct root certificate in the correct cacerts file for the LDAP functions to verify the server certificates correctly.

```
keytool -import -trustcacerts -keystore $JAVA_HOME/jre/lib/security/cacerts -storepass changeit -alias short-name-for-this-certificate -import -file file.cer
```

Special configuration notes for PHP LDAP on Windows

NOTE: These instructions for PHP on Windows are VERY old and may no longer work.

PHP uses the OpenLDAP library, which uses the OpenSSL library. You need to tell OpenLDAP where to find the trusted root certificates. This is the correct way to configure PHP LDAP for encryption on Windows.

1. Install PHP (recommended path is C:\php and version >= 5.3.1)
2. Make sure that libeay32.dll and ssleay32.dll are in the Windows path (typically C:\php)
3. Modify php.ini to tell PHP the location of the extensions directory (extension_dir = "c:\php\ext")
4. Modify php.ini to enable the LDAP extension (extension=php_ldap.dll)
5. Create C:\ldap.conf file (plain text, use Notepad or similar)
6. Add only the line TLS_CACERT c:\cacerts.pem to the C:\ldap.conf file
7. Create C:\cacerts.pem file (plain text, use Notepad or similar)
8. Obtain a copy of the USERTrust Secure certificate from InCommon
9. Add that certificate (in PEM/Base64 encoded format) to the C:\cacerts.pem file

NOTE: You do not need to enable the PHP OpenSSL extension. Yes, the LDAP module uses the OpenSSL code library, but that is not the same thing as the PHP extension for OpenSSL.

NOTE: The **cacerts.pem** file can contain multiple root certificates. You may wish to add the InCommon (a.k.a. Sectigo, Comodo) root ("USERTrust Secure") and the root certificate we use for Active Directory.

NOTE: Some of the comments on the PHP web site imply that PHP might look for the **ldap.conf** file on a drive other than the C: drive, especially if your web server is installed on another drive. It is not possible to relocate the **ldap.conf** file. In rare circumstances, it may be necessary to use the FileMon utility to find out where PHP is looking. Try these paths where X is the drive letter of any hard disk (starting with C:).

1. X:\ldap.conf
2. X:\OpenLDAP\sysconf\ldap.conf

OpenSSL command reference

View certificate details (if needed, add "-inform DER" or "-inform PEM"):
```
openssl x509 -in input-file.pem -text
```

Convert certificate file format from binary (".cer") to Base64 (".pem"):
```
openssl x509 -inform DER -in input-file.cer -outform PEM -out output-file.pem
```

Convert certificate file format from Base64 (".pem") to binary (".cer"):
```
openssl x509 -inform PEM -in input-file.pem -outform DER -out output-file.cer
```

Roles System:

This section is smaller, as much of its integrations are located in other parts of the system functionality wise.

Author: Jorge Hernandez

- Roles permissions need to be validated before allowing a user to see any screen other than the login screen. This is done via a database API query. Check database documentation for more details on Access database queries.
- The database has a “Roles” table which has a Boolean value for each permission. Access is modified using that table.
    - Location: “Backend\AbetData.accdb”
- Each user is given a role upon creation, which determines their access in the webapp.
- When the user logs in, each page verifies if the Boolean for that specific page is enabled or not before displaying.
    - Also, each element of the “home” page verifies the user’s access. Only buttons to which the user has access are displayed.

File: backendHelperFunctions.ts
Location: angular-example\src\app\roles

Class Name: RolesBackendHelperFunctions

Author: Matthew Harris

This file is intended to hold several helper functions to assist with the backend after a user has already logged in. The functions currently contained test a user’s role so that a page will display content specific to that role.

Pseudocode:

Functions

isAdmin(user: User)

Takes a User, returns true if that user’s role is Admin.

isCoordinator(user: User)

Takes a User, returns true if that user’s role is Coordinator.

isProfessor(user: User)

Takes a User, returns true if that user's role is Professor.

isTA (user: User)

Takes a User, returns true if that user's role is TA.

isStudent (user: User)

Takes a User, returns true if that user's role is Student.

Front End Design:

The general idea of the front end is to bring it within the same design principles visually as other UNT sites. On top of this, we wanted to make the website more user friendly in general, as many many pages were stuffed inside of dropdowns with little to no clarity on what each dropdown had and what certain page names meant.

All pages for the front end are components within angular so that they are linked together via the on screen buttons. Many of them are just the main template again but with a name referencing what page they should be, they just need a javascript form and API call.

Professor/TA and student UI has not had development on it yet, however the idea we had was to make it look more like Canvas with a 'card system'. The students would log in to the ABET system, and their landing page would just be a page with all of their current classes on it. They would click a classes' 'card' and it would direct them to the survey for that class. Once done, they would return to the main menu again and they could choose another class, with the class they just did being marked as 'done'.

Likewise for the professors and TA's, they would log in and see the classes they are associated with in the database. They would click on a class, enter their information and comments, as well as upload assignments as proof of a course outcome being met. TA's can do the same thing for these classes, and they can do it in place of their professors. However, professors have final submissions rights, so the data is not locked into the database as final until the professor hits the submit on the entered data.

Usability and usability testing suggestions:
We were making use of Nielsen's 10 Heuristics, which I strongly encourage making use of and following as you iterate further. You can find them here:
https://www.nngroup.com/articles/ten-usability-heuristics/
I would encourage doing a fair bit of both moderated and unmoderated testing throughout the semester and iterate on UI early, as that was a major issue we ended up having.

UI Files in repository package:
Location:Old_UI folder
File(s):ABETLogin.css, .xaml, and .svg

These are the original files for the login page that we created using a tool called lunacy. I would not recommend utilizing Lunacy and its iOS counterpart Sketch to assist in UI creation, only a previewer if you're in need of one.

The SVG file is very much the most useful of all of the files, as it provides a visualization of what the code is supposed to look like in practice when placed in Github, and i would imagine other locations as well. It should help with establishing a theme, as well as seeing some of our design philosophy in action. The CSS file and the XAML file are the raw code of the page, though the XAML file's use is limited due to it being Microsoft only.

Location:Old_UI folder
File(s):ABETTemplate.css, .xaml, and .svg

Like the previous fileset, this is another set that was created in Lunacy. The SVG file is the most important once again. This file was the template page for the admin side UI, and as such it would be used across the entire admin side of the system. Specifics will be elaborated upon in the new UI page.

Location: Master Branch
Files: index.html

This is the updated Admin template. It contains a top set of responsive dropdown tabs that have had their names revamped from the old dropdowns, and had some locations moved around to more fitting tabs.

The buttons along the side are intended to be quick access buttons from any page, and if a form is to go far down a page then they should be stickied. What pages you opt to place on these buttons is up to preference of the sponsor. These buttons are not intended to be replacements for the respective page's placement in the dropdowns. If a button goes onto the quick access bar, then it should still have a location in the dropdowns.

The central space is empty, as this is a template document. That space is intended for javascript forms to show up on a given page. Thus you can copy and paste the template to a multitude of pages to save time, and only need to encode the javascript form and the API call to submit the data to the database on data entry completion.

Location: Master Branch
File(s): ABETLogin.html, index.php

This is the login page UI file. It has some formatting errors now, however it works with a mock backend, and when security is integrated should work just fine. Index.php is the current php backend authenticator, but might need to be changed as it was running into LDAP server issues, and might be better handled on the Angular side with packages.

Database and API:

Authors: Rory Spralls and Basil Holloway

1. **The tables:**

    1. **Course Templates:** General data on courses like semester established and course description.

    2. **Course Instances:** More specific data like the current coordinator of the course and their comments on the course

    3. **Sections:** Very specific data like the professor, their comments on the course, and the number of students in the course.

4. **Section Comments:** specific comments from students and TAs about a section of a course.

5. **Objectives:** List of objectives, their descriptions, and which major they most closely follow

6. **Objective Thresholds:** Logs how specific majors did in *specific portions* of the course

7. **Section Thresholds:** Logs how specific majors did in the ***entire*** course

8. **Semesters:** Mainly used to speed up queries because comparisons made between semesters is not very common

9. **Staff:** Holds staff ids, and their roles. Includes TAs

10. **Roles:** Handles roles and their permissions like what pages of the system they may view

11. **Assignment Tables:** These are sort of junctions that allow many-to-many connections in the database

More information on the tables can be gained by reading through the MS Access database design document, located in Backend\AbetData.accdb.

2. **Their Relationships:**

   This figure shows a rudimentary picture of the relationships. Remember that PK, or *primary key,* means that in any given entry on any given table, there will *never* be a duplicate set of primary keys. IE: section comments can not have more than one comment that refers to the section 1 of CSCE 4110 and has the comment number 3.

3. **The API: At the end of this document will be resources that contributed to the construction of this portion of the back end.**

1. **General Information**: The API acts as a sort of bridge between the front-end UI and the back-end database using Microsoft's ASP NET server framework. It was written and designed in a way that would enable it to be able to change the database that it referred to as easily as possible, hence the strange combination of a traditional ASP NET API and an API built for communicating with MS Access, which requires a different approach.

2. **AbetController.cs:** This file handles the requests from the front-end and determines the actions required by the API to fulfill requests from the front-end. Has not been built to support MS Access yet, and will require integration with OdbcController.cs. *This portion of the code does not relate very closely to MS Access and can therefore be mostly ignored until edits are needed for it to work with MS Access*

    1. getRaw(string table): returns the raw DbSet. This is used internally, mainly to save on a few typestrokes.

    2. getTable(string table): returns a DbSet as a list to be read asynchronously. This is sent to the user at the front-end.

    3. createEntry(string table, object entry): creates and entry in the table and returns success or failure

    4. editEntry(string table, string id, object entry): edits an entry in the table. Returns no content on success.

    5. deleteEntry(string table, string id): deletes an entry in the table

3. **OdbcController.cs:** This file handles communication between the API and MS Access. There is little code within because the file is incomplete, as we ran out of time to fully build it, as we only learned of MS Access's shortcomings after constructing most of the API.

4. **AbetContext.cs:** This simple file just puts everything that is within ABETmodels.cs into a set of DbSet list objects. These are used to manipulate the data in a way that is closer to SQL. Has little to do with MS Access, unfortunately.

5. **ABETmodels.cs:** This file closely resembles the structure of the MS Access database in terms of what data is stored within each entry of the database, but it is very loosely related as it is the base of a traditional approach to database APIs.

4. **Learning resources used in the construction of the API**

    1. **Tradition API construction:** Can be used to understand the theory behind the structure of the API. Will help in altering the APIs code in the event of changes being necessary.

    2. **MS Access API construction:** Can be used to understand the theory behind the structure of the MS Access portions of the API.

    3. **UNT Free Microsoft Office Subscription:** Needed to use MS Access in a capacity that will allow you to make any serious changes to database structure.

    4. basilfholloway@gmail.com: the email of the main API and database programmer. Please contact him if you have any questions. Will be most helpful on weekends.

Links and Contact Info:

In case you have any questions (as is likely to be the case regardless of how thorough the documentation is).
Make sure to mark emails as important/high priority to reduce chances of going to spam.

Logan Falkensten- Leader of Joint Teams: loganfalkensten@gmail.com
Basil Holloway- Database Designer: basilfholloway@gmail.com
Jorge Hernandez - Roles management: jlhern.91@gmail.com