Courseworks App Design Documentation
Pre-Alpha Build

Alexander Roth
UNI: air2112

May 21, 2014

1 Overview

The goal of mobile application is to compliment the features of Columbia's Courseworks web site and increase accessibility for students.

As of May 21, 2014, this document outlines the development process and future iterations of the Columbia Courseworks app. **Note:** This application is still in development in a *pre-alpha build* and is currently being built for Android only.

2 Features

Again, this application is in the pre-alpha development cycle. Thus, this section is subject to heavy change depending on justification in further design and development cycles.

2.1 Current Features

Features currently implemented within the application include:

1. "Remember Me!" functionality
2. OAuth 2.0 Resource Owner Password Credentials Protocol
3. OAuth 2.0 Implicit Grant Protocol

2.2 Future & Proposed Features

1. OAuth 2.0 Authorization Code Grant Protocol
2. Current Semester Homepage
3. Course Names instead of Course numbers

4. Seamless calendar integration with native mobile calendar applications.
5. A viewable roster
6. Easy downloads under “Files & Resources”
7. Notification hub for emails and updates from professors to students.

3 Implementation & Efficiency

3.1 Workflow

The current design of the app works in a three-step process: the preamble, the authenticat ion, and the main process.

1. **Preamble Process**
   The preamble process involves the use of the Splash screen. To the user, a splash screen with Columbia's logo will be displayed. On the back-end, the application will be checking if the “auth.xml” exists. If this file is present, then a login attempt will begin. If this file is not present, then the login activity will be invoked.
2. **Authentication Process**
   After the preamble process, the application moves into authentication. If the auth file exists, the system goes to auto-login using the credentials of the last user. If this is not the case, the user will be redirected to the login screen, where he or she will provide credentials for the system to use.
   Once credentials are provided, the application runs OAuth's *Resource Owner Password Credential* protocol. The user's credentials are sent to the authentication web server, where they are authenticated by the server. Meanwhile, the application waits for an access token after verification occurs. When an access token is granted, the application moves to the main process.
3. **Main Process**

3.2 Resource Allocation

Since this application is native to the mobile device, there will be very little overhead, except for the HTTP requests and responses when communicating with the servers. As of May 21, 2014, there are only two requests being sent to the Courseworks servers: The POST and GET requests from the Authentication process.

4 Class Design

4.1 AuthPreferences

A helper class for the Login activity and the OAuthClient class. The class contains a number of the methods that either write to or read from the SharedPreference file "auth .xml".

AuthPreferences The constructor for the class. Links to the "auth.xml" file depending on the context in which it is created. Controls the ciphers that will read and write to the auth file. Also holds onto the encryption protocol used throughout the program.

initCiphers Initializes the ciphers for encryption and decryption.

getIV Creates an initialization vector for the ciphers to use.

getSecretKey Constructs a secret key from a given byte array.

createKeyBytes Hashes the given encryption key by using SHA-256 as the hashing algorithm.

put Adds key-value pair to the SharedPreference file. If the value already exists, it is overwritten.

containsKey Checks if the given key is within the SharedPreference file.

removeValue Removes the specified value from the SharedPreference file.

getString Checks if the SharedPreference file contains the given key value; if present, retrieve the encoded value and decode it.

clear Clears all information from the auth file in case the user does not want to save his or her credential information.

toKey Encrypts a given key value.

putValue Encodes a value and stores it in the SharedPreference file.

encrypt Encrypts a plain text string into an encoded string.

decrypt Decrypts an encoded string into a plain text stirng.

convert Encrypts or decrypts data in a single-part operation, or finished a multiple-part operation. The data is encrypted or decrypted, depending on how the cipher was initialized.

4.2 ImplicitGrant

Authentication based off of Oauth2.0's ImplicitGrant Authorization flow.

1. From the Login activity, launches a WebView activity that redirects to the Columbia CAS/Oauth2.0 servers with the already supplied Client ID and Redirec URI.
2. Confirms user and authorizes the app to access the user's information.
3. Parse the token from the response URI.

4.3 Main

Not sure yet, haven't really built anything here since Authentication is still being worked on.

4.4 Login

This activity controls the Login screen.

onCreate Creates the login form and the functionality for the activity.

1. Sets up the login form.
2. Checks if there is a SharedPreference file containing the previous user information. If there is previous user information, it automatically logs in.
3. If there is no previous user information, waits for the user input and user command to log in.

onCreateOptionsMenu Creates the options menu. Standard for any Android application.

attemptLogin Attempts to sign into the account specified by the login form. If there are form errors (i.e. invalid uni, missing fields, etc.), the errors are presented to the user and no actual login request is made.

showProgress Shows the progress UI and hides the login form.

doInBackground Launches the login task from OAuthClient in an Asynchronous task.

4.5 ResourceOwnerCredential

The real meat of the program so far. It will contain all the authentication methods for the application. Currently, it only contains the login method.

login Uses OAuth's *Resource Owner Password Credentials* protocol to log the user into Courseworks.
The process falls into a number of steps:

1. Takes in the user's credentials and a URL to the Courseworks servers.
2. Creates an HTTPS connection between application and server.
3. Sets the security of the connection to TLS.
4. Prepares the connection to be a POST request.
5. Adds necessary data values for the Resource Owner Password Credentials . This includes 4 variables:
   - The grant type
   - The username
   - The password
   - The client ID number
6. These values are then passed to getQuery which encodes the variables onto the POST request.
7. Sends POST request to the server.

getQuery Takes in a value-paired list of parameters which are then encoded into proper formatting for the POST request.

4.6 RestGrant

An alternative to both OAuth implementations mentioned beforehand. Implements CAS RESTful API to make a series of calls to the CAS Authentication servers.

login Logs the user into Columbia's CAS servers using calls to CAS' RESTful API.

logout Logs the user out of the CAS server by calling DELETE to the RESTful API.

getGrantingTicket Makes a POST call to CAS authentication servers in order to return a Ticket Granting Ticket Resource.

getServiceTicket Gets the Service Ticket from the CAS Authentication server .

postUserToServer Sends a POST call to Columbia's CAS Authentication server with the user's information in order to request a Ticket Granting Resource.

postTicketToServer Sends a POST call to Columbia's CAS Authentication server with the Ticket Granting Resource in order to request a Service ticket

4.7 Splash

The Splash activity acts as a buffer between the Login activity (on first access of the application) or the Main activity on the application.

run Runs the splash screen for a set period of time.

5 Risks

5.1 Security

Since the application is native, it is more conducive to store the user's credentials on the mobile device; thus allowing the application to hold the user's credentials in a file system. However, this ease opens up a large security risk. The file holding the user's sensitive information (e.g. uni and password) is not secure at the present moment. There are three workarounds to this issue:

1. We encrypt the SharedPreference file holding the user's information.
2. We remove the "Remember Me!" feature.
3. We move to web authentication through OAuth's *Authorization Code* protocol.
4. We move to web authentication through OAuth's *Implicit Grant* protocol.

As of May 21, 2014, I have implemented an encryption method using Advanced Encryption Standard (AES). However, the SharedPreference file is not guaranteed completely secure; I just increased the difficulty for someone to access the information.

6 ToDo

1. Test Authentication Process.
2. ~~Secure private information of user.~~
3. Update feature list based on student response.
4. Move to alpha build once the Courseworks servers are prepared.