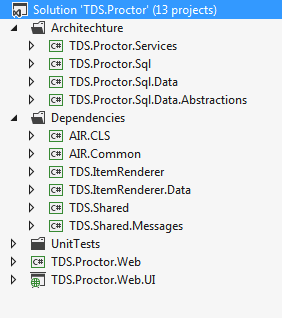
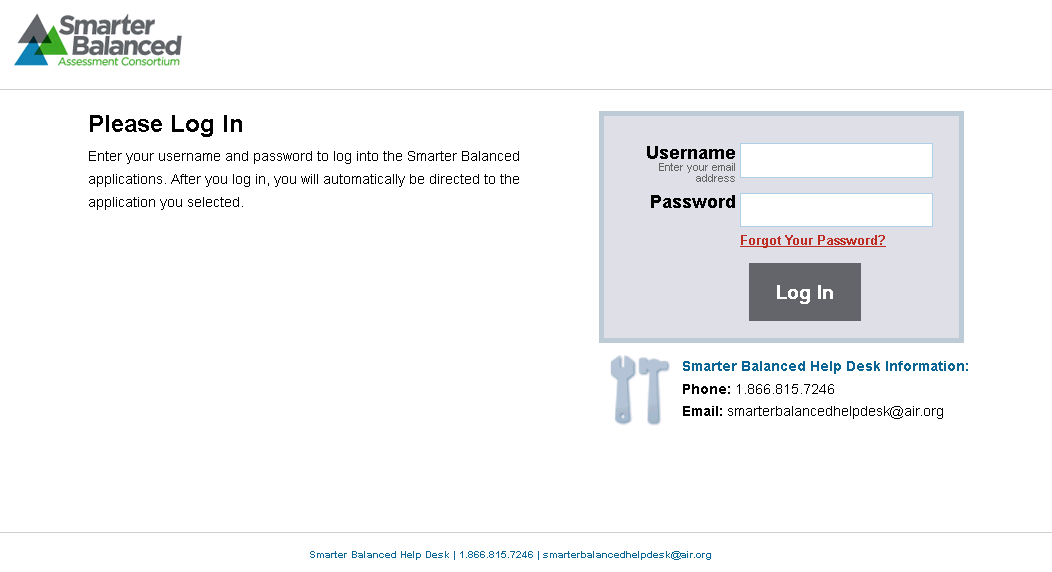
# Proctor App Summary:

1. Test Proctor URL: <https://tds2.airws.org/test_testadmin_2012/login.aspx?c=SBAC&clslogin=false>
   1. UserNames: [ClientUser1@air.org](mailto:ClientUser1@air.org); [ClientUser2@air.org](mailto:ClientUser2@air.org)
   2. Password: Air1234$
2. Test Student URL: <https://tds2.airws.org/test_student_2012/default.aspx?c=SBAC>
   1. Student: just use GUEST student with real session that created by the proctor.
3. Databases:
   1. 38.118.83.146.TDSCore\_Test\_Session\_2012
4. CVS:
   1. Module: TDSCore
   2. TDSCore\AppsCurrent\Proctor\
5. Solution file: TDSCore\AppsCurrent\Proctor\TDS.Proctor.sln



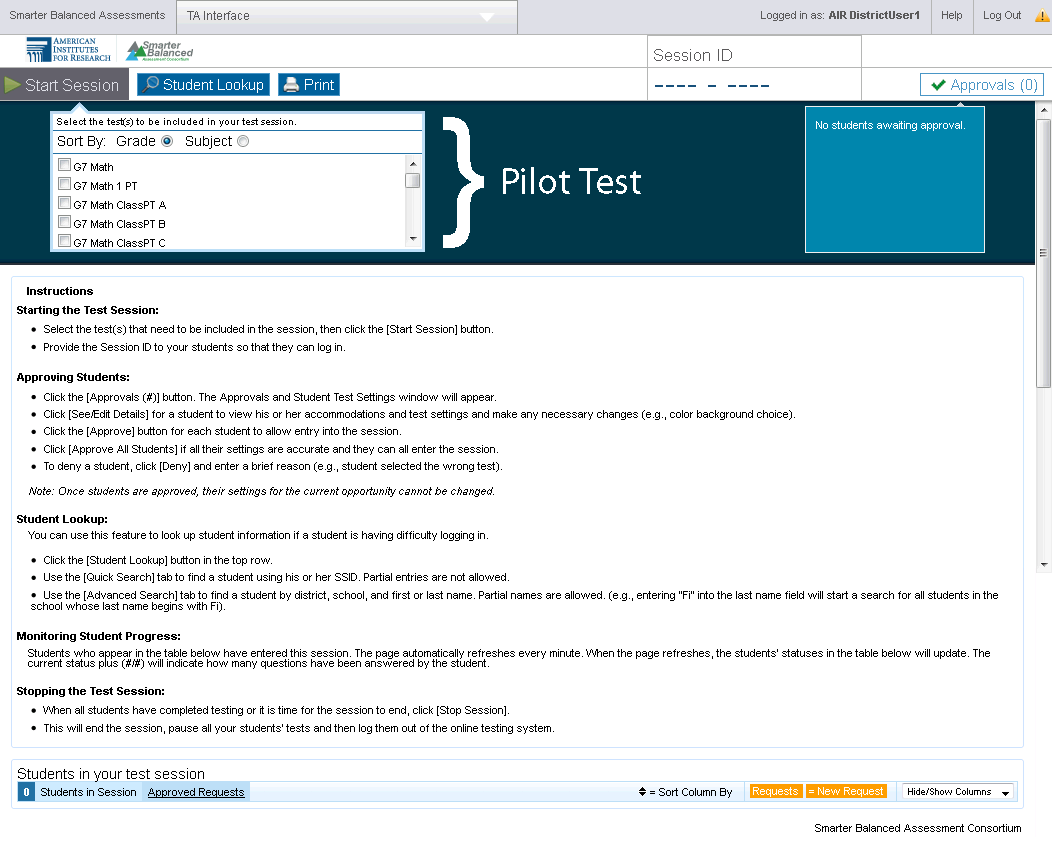
* 1. Architecture:
     1. TDS.Proctor.Service: All available service APIs in the proctor app, BLL
     2. TDS.Proctor.Sql: All database access layer APIs to database store procedures, DAL
     3. TDS.Proctor.Sql.Data: All data classes
     4. TDS.Proctor.Sql.Data.Attractions: All interface to service classes
  2. Dependencies:
     1. AIR.CLS 🡪 TDSCore\CLS\AIR.CLS\
     2. AIR.Common 🡪 TDSCore\Common\
     3. TDS.ItemRenderer 🡪 TDSCore\ItemRenderer\
     4. TDS.ItemRenderer.Data 🡪 TDSCore\ItemRenderer\
     5. TDS.Shared 🡪TDSCore\Shared\
     6. TDS.Shared.Messages 🡪TDSCore\Shared\
  3. TDS.Proctor.Web: All presenters, web controls, and XHR handlers
  4. TDS.Proctor.Web.UI:
     1. All ASPX files
     2. All Javascripts files
     3. Html/CSS

# Login Page (TDS.Proctor.Web.UI\Login.aspx): Proctor login



1. **Presenter**: TDS.Proctor.Web\LoginPresenter.cs
2. **XHR Handler:** None
3. **BLL:** TDS.Proctor.Service\ProctorUserService.cs;
4. **DAL**: TDS.Proctor.Sql \ProctorRepository.cs
5. **Database Store Procedures**: P\_ValidateProctor
6. **On Login Click**:
   1. Call presenter.DoLogin
   2. On Success:
      1. Create a Form authentication cookie
      2. Redirect to the default home page (TDS.Proctor.Web.UI\Default.aspx)
   3. On Failure:
      1. Display the error message

# Home Page(TDS.Proctor.Web.UI\Default.aspx): No existing session



1. **Presenter**: TDS.Proctor.Web\HomePresenter.cs
2. **Handlers**: TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs
3. **BLL**:
   1. TDS.Proctor.Service\ProctorUserService.cs; TestSessionService.cs
   2. TDS.Proctor.Sql.Data\TestSession.cs; SessionDTO.cs;
   3. TDS.Proctor.Sql.Data.Abstractions\ITestSessionService.cs;
4. **DAL**: TDS.Proctor.Sql \TestSessionRepository.cs; ProctorRepository.cs
5. **Database Store Procedures**:
   1. P\_GetConfigs: get application configs and settings
   2. AppMessagesByContext: get all messages and translations
   3. P\_GetCurrentSessions: get current session
   4. P\_GetAllTests: get all selectable tests
   5. IB\_GetSegments: get a list of test segments
   6. P\_GetSessionTests: get a list of session tests
   7. P\_GetUnAcknowledgedAlertMessages: get a list of unacknowledged alert messages
   8. P\_CreateSession: create a session
   9. P\_InsertSessionTest: insert a session test
   10. P\_GetTestsForApproval: get a list of students waiting for approval
6. **On Initialization:** render the home page without an existing session
   1. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: GetInitData
      1. Data Elements: SessionDTO.cs
         1. Current session if exists
         2. List of selectable tests
         3. List segments
         4. List of unacknowledged alert messages
         5. If session exists, get a list of session tests
         6. If session exists, get a list of students waiting for approval if this session has students are waiting for approval
         7. If session exists, get a list of students belong to this session
      2. On Success: TDS.Proctor.Sql.Data\SessionDTO JSON object
      3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>
7. **On “Start Session” Click:** Start a new session
   1. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: InsertSessionTests

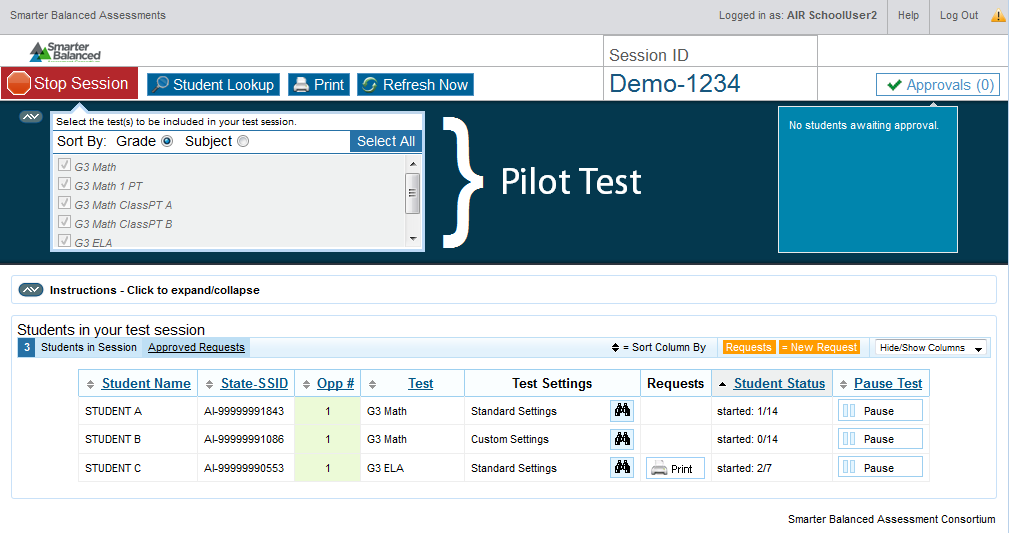
If there isn’t a session key, create one by call CreateSession TDS.Proctor.Web\ProctorAppTasks.cs 🡪 TestSessionTasks.CreateSession

For each selected test, call TDS.Proctor.Web\ProctorAppTasks.cs 🡪 TestSessionTasks.InsertSessionTest

* + 1. Data Elements: SessionDTO.cs
    2. On Success: TDS.Proctor.Sql.Data\SessionDTO JSON object
    3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>

1. **On “Student Lookup” Click:** Render quick or advance search dialog
   1. Popup an iframe TDS.Proctor.Web.UI\Lookup.aspx page
2. **On “Alerts” Click:** View the current alerts
   1. Popup an iframe TDS.Proctor.Web.UI\Alerts.aspx page
3. **On “Logout” Click:** Do logout
   1. Redirect to TDS.Proctor.Web.UI\Shared\Logout.aspx page to logout

# Home Page(TDS.Proctor.Web.UI\Default.aspx): Existing session



1. **Same as the default page above but with active session …**
2. **On “Stop Session” Click:** Stop a current session
   1. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: PauseSession

TDS.Proctor.Web\ProctorAppTasks.cs 🡪 TestSessionTasks.PauseSession

* + 1. Data Elements: SessionDTO.cs
    2. On Success: TDS.Proctor.Sql.Data\SessionDTO JSON object
    3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>

1. **On “Auto Refresh”:** Auto refresh certain data elements every X configurable seconds
   1. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: AutoRefreshData
      1. Data Elements: SessionDTO.cs
         1. Get a list of students waiting for approval if any (TDS.Proctor.Sql.Data\TestOpps.cs)
         2. Get a list of students belong to this session (TDS.Proctor.Sql.Data\TestOpps.cs)
         3. List of unacknowledged alert messages
      2. On Success: TDS.Proctor.Sql.Data\SessionDTO JSON object
      3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>
2. **On “Refresh Now”:** Same as auto refresh
3. **On “Approvals”:** Approve students that are waiting to join the session
   1. Render “Approvals and student test settings” dialog
4. **On “Pause” Test Opportunity:** Pause a test opportunity in a session
5. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: PauseOpportunity
   * 1. Data Elements: None
     2. On Success: ReturnStatus JSON (Status: success; Reason: null)
     3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>
6. **On “Print” Testee Requests:** Print this test opportunity printing requests
7. Open an iframe dialog with Requests.aspx
8. **On “Approved Requests”:** Display all approved requests for this session
9. Open an iframe dialog with ApprovedRequests.aspx

# Home Page(TDS.Proctor.Web.UI\Default.aspx): Approvals and student test settings

1. **On Initialization:** render the approvals and test setting dialog. Also get all configured test accommodations
   1. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: GetAccs
      1. Data Elements: AccsDTO.cs
         1. Get All Accommodations for a test (TDS.Proctor.Services\TestService.cs: GetTestAccs)
      2. On Success: TDS.Proctor.Sql.Data\AccsDTO JSON object
      3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>
2. **On “Approve All Students”:** Approve all students that are waiting
3. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: ApproveOpportunity

Loop through a list of opportunities that are waiting for approval and call ApproveOpportunity

* + 1. Data Elements: SessionDTO.cs
    2. On Success: TDS.Proctor.Sql.Data\SessionDTO JSON object
    3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>

1. **On “Refresh”:** Refresh all students that are waiting for approval
2. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: GetApprovalOpps

Get all students that are waiting for approval

* + 1. Data Elements: SessionDTO.cs
       1. Get a list of TestOpportunity (TDS.Proctor.Sql.Data\TestOpps.cs)
    2. On Success: TDS.Proctor.Sql.Data\SessionDTO JSON object
    3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>

1. **On “Approve” Test Opportunity:** Approve a student that is waiting
2. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: ApproveOpportunity

Approve all student’s accommodations first and then Approve the opportunity itself (TDS.Proctor.Services\TestOpportunityService.cs: ApproveAccommodations, ApproveOpportunity)

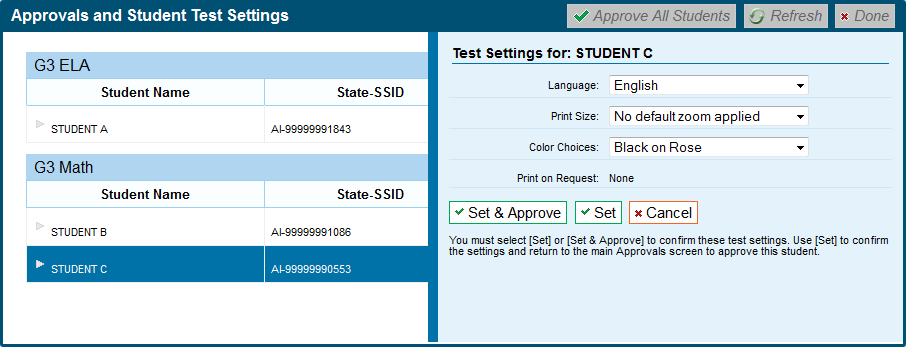
* + 1. Data Elements: None
    2. On Success: ReturnStatus JSON (Status: success; Reason: null)
    3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>

1. **On “Deny” Test Opportunity:** Deny a student that is waiting
2. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: DenyOpportunity

Deny approval for this opportunity (TDS.Proctor.Services\TestOpportunityService.cs: DenyOpportunity)

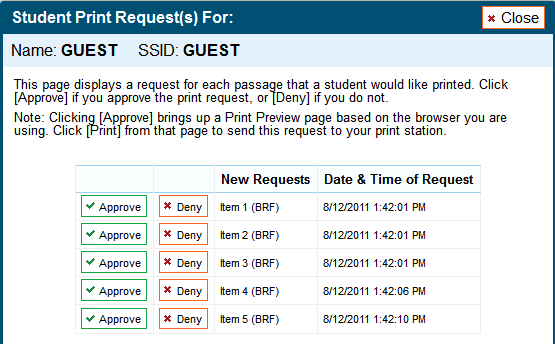
1. Data Elements: None
2. On Success: ReturnStatus JSON (Status: failed; Reason: <reason of failure>
3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>

# Home Page(TDS.Proctor.Web.UI\Default.aspx): student test settings



1. **On “Set & Approve”:** Same action as on “Approve” above
2. **On “Set” and “Cancel”:** No server side calls here…

# Requests Page(TDS.Proctor.Web.UI\Requests.aspx): Student’s requests that are waiting for approval



1. **On Initialization:**
   1. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: GetCurrentRequests

Get all current requests for this testee and render the table

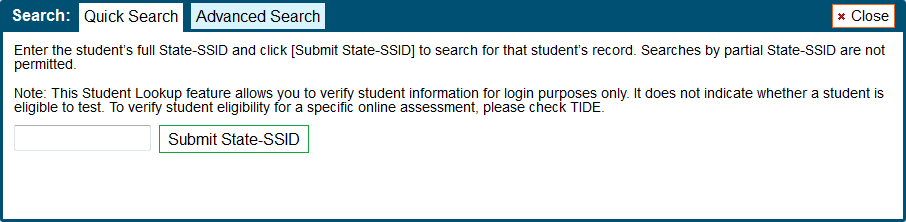
* + 1. Data Elements: TesteeRequestsDTO.cs
       1. Get All current testee’s requests (TDS.Proctor.Services\TesteeRequestService.cs: GetCurrentTesteeRequests)
    2. On Success: TDS.Proctor.Sql.Data\TesteeRequestsDTO JSON object
    3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>

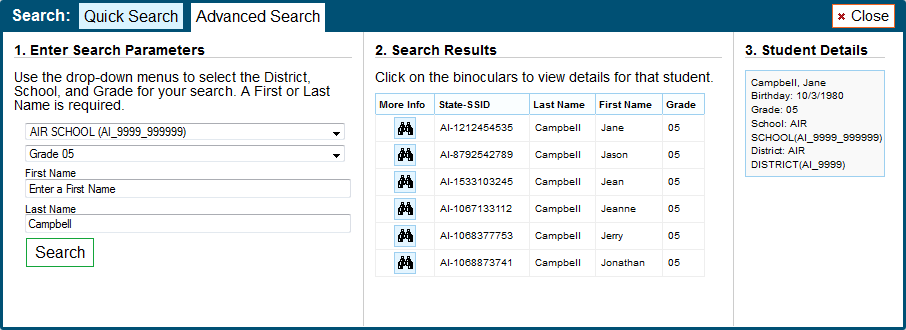
1. **On “Approve”:** Approve a student that is waiting
   1. Jump to PrintRequest.aspx page
2. **On “Deny”:** Deny a testee’s printing request
3. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: DenyTesteeRequest

Deny testee request for this opportunity (TDS.Proctor.Services\TesteeRequestService.cs: DenyTesteeRequest)

1. Data Elements: None
2. On Success: ReturnStatus JSON (Status: True; Reason: null
3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>

# Lookup Page(TDS.Proctor.Web.UI\Lookup.aspx): Quick and advance student lookup





1. **Presenter**: TDS.Proctor.Web\LookupPresenter.cs
2. **Handler**: TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs
3. **BLL**:
   1. TDS.Proctor.Service\TesteeService.cs; InstitutionService.cs
   2. TDS.Proctor.Sql.Data\Testee.cs; Institution.cs; Grade.cs
   3. TDS.Proctor.Sql.Data.Abstractions\ITesteeService.cs; IInstitutionService.cs
4. **DAL**: TDS.Proctor.Sql \TesteeRepository.cs; InstitutionRepository.cs
5. **Database Store Procedures**:
   1. P\_GetRTSTestee: Get a testee information
   2. GetRTSUserRoles: Get a list of RTS User Roles and Institution ID, name, type
   3. R\_GetSchoolGrades: Get a list of grades in this school
   4. R\_GetSchools: Get a list of schools within a district
   5. GetSchoolStudents: Get a list of testees within that match with selected school, grade, firstname, lastname
6. **On Quick Search tab Initialization:** Render the quick search dialog, no extra server side code calls
7. **On “Quick Search” OR “Submit” click:** Make call to the handler and then render the returned JSON object
   1. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: GetTestee

Get testee information.

* + 1. Data Elements: Testee.cs
       1. Get testee’s information (TDS.Proctor.Services\TesteeService.cs: GetTestee)
    2. On Success: TDS.Proctor.Sql.Data\Testee JSON object
    3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>

1. **On Advance Search tab click:** Render a list of user’s institutions and grades.
   1. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: GetInstitutions

Get a list of user institutions that this user belongs to

* + 1. Data Elements: Institution.cs
       1. Get a list of user institutions that this user belongs to (TDS.Proctor.Services\InstitutionService.cs: GetUserInstitutions)
    2. On Success: TDS.Proctor.Sql.Data\InstitutionList JSON object
    3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>

1. **On select an institution:** If selected institution type is district, get a list of schools else get a list of grades. Render schools/grades
2. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: GetSchools

Get a list of schools for selected district

* + 1. Data Elements: Schools.cs
       1. Get a list of schools for selected district
    2. On Success: TDS.Proctor.Sql.Data\Schools JSON object
    3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>

1. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: GetGrades

Get a list of grades for selected school

* + 1. Data Elements: Grades.cs
       1. Get a list of grades for selected school
    2. On Success: TDS.Proctor.Sql.Data\Grades JSON object
    3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>

1. **On “Advance Search” button click:** Perform advance search

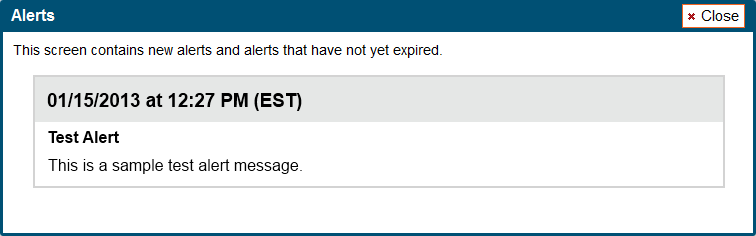
Make XHR call to Search based on selected school, grade, firstname, lastname

1. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: GetSchoolTestees

Get a list of testees based on selected school, grade, first name and/or last name

* + 1. Data Elements: Testees.cs
       1. List of testees
    2. On Success: TDS.Proctor.Sql.Data\Testees JSON object
    3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>)

# Alerts Page(TDS.Proctor.Web.UI\Alerts.aspx): List alerts for the proctor



1. **On Initialization:** Render the list of current alerts for this proctor
   1. XHR Handler Name**:** TDS.Proctor.Web\Handlers\ActiveSessionXHR.cs: GetCurrentAlertMessages

Get a list of current alert messages

* + 1. Data Elements: AlertMessages.cs
       1. AlertMessages (TDS.Proctor.Services\AlertMessageService.cs: GetCurrentAlertMessages)
    2. On Success: TDS.Proctor.Sql.Data\AlertMessages JSON object
    3. On failure: ReturnStatus JSON (Status: failed; Reason: <reason of failure>