CS 646 iPad/iPhone Mobile Application Development
Fall Semester, 2018
Assignment 5

San Diego State University -- This page last updated 10/25/18

Assignment 5 - SDSU Class Registration

Due Nov 18 23:59
Version 1.1

Goals

Network

App Description

The app will allow users to register for classes. A student needs to enter their personal information - first name, last name, SDSU red id, email address and a password. The user can register for up to three courses and add themselves to waitlists for courses that are full. The student can also drop classes that they are registered for or drop themselves from waitlists.

The app will allow students to select from a list of courses filtered via major (or subject like Computer Science, Physics, etc), level of the course (lower division - 100 & 200 level courses, upper division courses 300, 400, 500 level courses, and graduate courses (500 level and higher), and time of day (classes starting after a given time and/or ending before a given time). The server has a list of over 4,000 courses so you do not want to display all courses in one list.

The app should display the courses the student is enrolled in and the courses that they are waitlisted for. The app should store the students personal data on the device so that they do not have to enter the data each time they use the app.

Server Interaction Overview

A brief overview of the commands that you can send to the server and what they do for you.

subjectlist Returns a list of majors or subjects. Includes the name of the subject, the college it is in allowing the classes to be grouped by college.

classidslist Returns a list of courses based on subject(s), level, and time. Returns just the ids of the courses.

classdetails Given a course id returns information about the course: title, instructor, meeting time and place, etc.

addstudent Given the personal information about the student added the student to the server so they can add classes.

registerclass Given a course id, student's red id and password registers the student in the class.

waitlistclass If a class is full given a course id, student's red id and password adds the student to the wait list of a class. If a student drops the course the server does not enroll a student from the waitlist in the course.

unregisterclass Drops a student from the course.

unwaitlistclass Removes the student from a course waitlist.

resetstudent Drops the student from all courses and removes them from all waitlists.

Server Protocol

When a request is successful it will return a response with the HTTP status 200. If there is a problem with the request the response will have a HTTP status of 404, 400, or 500. When you request a URL that does not exists you will get a 404 response and the body of the response will be HTML. This can happen when you mistype the URL. Other errors will return a response with a 400 or 404 status and the body of the response will be JSON. This can happen if you do not include a required parameter, misspell a parameter or the wrong type. All 200 responses have JSON data as bodies.

If your URL is correct and all the required parameters are included the response will have HTTP status 200. If the values of your parameters are invalid, say an incorrect password, the response will contain an JSON object with the single key "error". Examples are given below.

All the URLs are HTTPS.

Note that the URLs given below are case sensitive. The path and all query parameter names are lower case.

URLs

Subject (Major) list: Get a list of subject (majors) offered

URL: https://bismarck.sdsu.edu/registration/subjectlist

Method: GET only

Parameter: None

Returns: JSON array of JSON objects. Each JSON object contains 4 key-value pairs. The "title" is the name of the subject (or major or department). The "id" is used to get the list of classes in that subject (or major). "classes" is the number of classes in that subject. "college" is the college at SDSU to which the subject belongs.

Sample return value (truncated to save space):

```
  [{
     "title": "Astronomy",
     "id": 2,
     "college": "Science",
     "classes": 33},
{
     "title": "Computer Science",
     "id": 8,
     "college": "Science",
     "classes": 65}]
```

Course Ids list: Get a list of IDs of courses that are in a given subject or major.

URL (GET): https://bismarck.sdsu.edu/registration/classidslist?subjectid=AnInteger

Method: GET or POST

Parameters GET: subjectid, level, starttime, endtime

Parameters POST: subjectids, level, starttime, endtime

Returns: JSON array of integers. The integers are the ids of the classes that are in the given subject and meet the given criteria.

Parameters Explained

subjectid - required for GET the id for the give subject obtained from the URL above. The request will return all the classes in the given subject that are in the subject that meet the given criteria.

subjectids - required for POST, is a JSON array of subjects from the URL above. The request will return all the classes in the given subjects that are in the subject that meet the given criteria.

level - optional. Three legal values: lower, upper, and graduate. **lower** will return classes in the given subject with course numbers between 100 and 299. **upper** will return classes in the given subject with course numbers between 300 and 599. graduate will return classes in the given subject with course numbers between 500 and 899

starttime - optional. Value is a string representing time in 24 hour format like "0930" and "2100". Note that in a GET URL there are no quotes around the number. If given only courses that start at or after the given time will be returned. In the second example below all CS courses that are graduate level (500 and above) that start at or after "1730" are returned.

endtime - optional. Value is a string representing time in 24 hour format like "0930" and "2100". Note that in a GET URL there are no quotes around the number. If given only courses that end by or before the given time will be returned.

Samples:

https://bismarck.sdsu.edu/registration/classidslist?subjectid=8

https://bismarck.sdsu.edu/registration/classidslist?subjectid=8&level=graduate&starttime=1730

Return value

```
[7033,7036,7038,7041,7046,7049,7051,7052]
```

Post Sample:

URL: https://bismarck.sdsu.edu/registration/classidslist

Body: {"subjectids":[8], "level":"graduate", "starttime":"1730"}

Content Type: Must be application/json

Class Details: Get the details of a class.

URL (GET): https://bismarck.sdsu.edu/registration/classdetails?classid=AnInteger

Method: GET or POST

Parameters GET: classid

Parameters POST: classids

Returns: GET version -JSON object with 22 key-value pairs.

Returns: POST version -A JSON array of JSON objects with 22 key-value pairs.

Parameters Explained

classid - GET only. Id (integer) of the class obtained from the course id list url.

classids - POST only. JSON array of integers - the Ids of the class obtained from the course id list url.

GET Sample:

https://bismarck.sdsu.edu/registration/classdetails?classid=7036

Post Sample:

URL: https://bismarck.sdsu.edu/registration/classdetails

Body: {"classids":[7036]}

Content Type: Must be application/json

Sample output GET version. The meaning of most of the keys should be clear. "seats" is the number of students that are allowed to enroll in the course. "enrolled" is the number of students enrolled in the course. "Waitlist" is the number students in the waitlist for the class. Note that the values in quotes are strings. The values not in quotes are integers.


```
{
   "description": "Principles, ….",
   "department": "Computer Science",
   "suffix": "",
   "building": "GMCS",
   "startTime": "1900",
   "meetingType": "Lecture",
   "section": "01",
   "endTime": "2015",
   "enrolled": 0,
   "days": "TTH",
   "prerequisite": "Computer Science 310.",
   "title": "SOFTWARE INTERNATNLIZATN",
   "id": 7036,
   "instructor": "W. ROOT",
   "schedule#": "21073",
   "units": "3.0",
   "room": "425",
   "waitlist": 0,
   "seats": 48,
   "fullTitle": "Softwre Internationalization",
   "subject": "CS",
   "course#": "540"}
```

Register a Student: Register a student for a class.

A student can only add a class once. A student can register for at most 3 classes. A student can not register for classes that meet at the same time.

URL (POST): https://bismarck.sdsu.edu/registration/addstudent

{"firstname":"Roger","lastname":"Whitney","redid":"123456003","password":"password","email":"whitney@sdsu.edu"}

Method: POST.

Parameters POST: firstname, lastname, redid, password, email.

Returns: JSON object with one key, either "ok" or "error". See examples below.

Parameters Explained

firstname - String, first name of student

lastname - String, last name of student

redid - String, nine characters long. Only red ids already known to the server will be accepted. To avoid conflicts between students each students has a set of assigned red ids. Your set of red ids is any 9 digit string that ends with the same four digits as your SDSU red id. So if your red id is 822326003 then any red id from 000006003 through 999996003 will work. If you use an invalid redid the value returned will be

{"error": "Invalid Red Id"}

password - String and least 8 characters long and containing 7 different characters. If the password is not valid the returned value will be an error. For example:

{"error": "Password too short"}

email - String, a valid email address. If the email is not valid the return value will be a JSON object with the key "error". For example:

{"error": "Invalid email no @" }

Post Sample:

URL: https://bismarck.sdsu.edu/registration/addstudent

Body:
{"firstname":"Roger","lastname":"Whitney","redid":"123456003","password":"password","email":"whitney@sdsu.edu"}

Content Type: Must be application/json

Return: {"ok": "Student Added"}

If the same request is repeated the return value is:

{"error": "Red Id already in use"}

Other error responses include:

{"error": "Invalid Red Id"}

{"error": "Password too short"}

{"error": "Invalid email no @"}

Register for a Class: Register a student in a class.

A student can only add a class once. A student can register for at most 3 classes. A student can not register for classes that meet at the same time.

URL (GET): https://bismarck.sdsu.edu/registration/registerclass?redid=ARedId&password=APassword&courseid=ACourseID

URL (POST): https://bismarck.sdsu.edu/registration/registerclass

{"redid": ARedId, "password": "APassword", "courseid":ACourseID}

Method: GET or POST

Parameters GET & POST: redid, password, courseid

Returns: GET & POST version - JSON object with one key, either "ok" or "error"

Parameters Explained

redid - String, valid red id of a student already registered with server

password - String, password of the registered student.

courseid - Integer, id of the course to add the student to.

GET Sample:

https://bismarck.sdsu.edu/registration/registerclass?redid=123459003&password=password&courseid=7036

Sample Responses:

{"ok": "Course added"}

{ "error": "Student already in course"}

Waitlist for a Class: Add a student to the waitlist for a class.

A student can only be added to the waitlist of a class once and only after the class is full. To aid in testing waitlisting the courses with is 7034, 7035 and 7036 are full. Any attempt to add those courses will fail, but you can add students to the waitlist.

URL (GET): https://bismarck.sdsu.edu/registration/waitlistclass?redid=ARedId&password=APassword&courseid=ACourseID

URL (POST): https://bismarck.sdsu.edu/registration/waitlistclass

{"redid": ARedId, "password": "APassword", "courseid":ACourseID}

Method: GET or POST

Parameters GET & POST: redid, password, courseid

Returns: GET & POST version - JSON object with one key, either "ok" or "error"

Parameters Explained

redid - String, valid red id of a student already registered with server

password - String, password of the registered student.

courseid - Integer, id of the course to add the student to the waitlist.

GET Sample:

https://bismarck.sdsu.edu/registration/waitlistclass?redid=123459003&password=password&courseid=7036

Sample Responses:

{"ok": "Course added"}

UnRegister a Class: Unregister a student in a class.

URL (GET): https://bismarck.sdsu.edu/registration/unregisterclass?redid=ARedId&password=APassword&courseid=ACourseID

URL (POST): https://bismarck.sdsu.edu/registration/unregisterclass

{"redid": ARedId, "password": "APassword", "courseid":ACourseID}

Method: GET or POST

Parameters GET & POST: redid, password, courseid

Returns: GET & POST version - JSON object with one key, either "ok" or "error"

Parameters Explained

redid - String, valid red id of a student already registered with server

password - String, password of the registered student.

courseid - Integer, id of the course to remove the student from.

GET Sample:

https://bismarck.sdsu.edu/registration/unregisterclass?redid=123459003&password=password&courseid=7036

Sample Responses:

{"ok": "Course dropped"}

Unwaitlist a Class: Remove a student from the waitlist of a class.

URL (GET): https://bismarck.sdsu.edu/registration/unwaitlistclass?redid=ARedId&password=APassword&courseid=ACourseID

URL (POST): https://bismarck.sdsu.edu/registration/unwaitlistclass

{"redid": ARedId, "password": "APassword", "courseid":ACourseID}

Rest of the details are similar to unregister.

Student classes: Returns list of class a student is registered in or in a waitlist.

URL (GET): https://bismarck.sdsu.edu/registration/studentclasses?redid=ARedId&password=APassword

URL (POST): https://bismarck.sdsu.edu/registration/studentclasses

{"redid": ARedId, "password": "APassword"}

Method: GET or POST

Parameters GET & POST: redid, password

Returns: GET & POST version - On success a JSON object with two keys: "classes" and "waitlist". The values are a JSON array of class ids.

Sample Result:

{ "classes": [7036], "waitlist": [] }

Reset Student: Drops all students classes and and waitlisted classes.

URL (GET): https://bismarck.sdsu.edu/registration/resetstudent?redid=ARedId&password=APassword

URL (POST): https://bismarck.sdsu.edu/registration/resetstudent

{"redid": ARedId, "password": "APassword"}

Method: GET or POST

Parameters GET & POST: redid, password

Returns: GET & POST version - On success a JSON object with one key: ok or error. error is returned when the password or red id is not correct.

Sample Result:

{:ok "Student reset”}

Grading

Points	Time
5	Add a students personal information to app
8	Add student information to server
10	Can obtain subject list and class ids list from server
13	Register/Unregsiter a student for a class
13	Add/Remove student to waitlist
20	Student can filter classes based on subject(s), time of data, and level
15	Display students schedule and waitlisted classes
8	Quality of GUI, layout of widgets, workflow
8	Code quality, lack of app crashes
100	Total Points

What to Turn in

Create a Xcode project for the assignment. Xcode places the project in its own directory. Place the directory (and all its contents) into a zip file. This assignment we will not use blackboard to turn in assignments. We will use my older course portal (http://bismarck.sdsu.edu/CoursePortal).

Version 1.1

Changed: In “Register a student for a class” the valid red ids now require your last four digits of your SDSU ID.