|  |
| --- |
| University of MISSOURI-kansas city |
| SternerLearn |
| Increment 2 Report |
|  |
| **Connor Ledgerwood and Devin Turner** |
| **3/12/2013** |

|  |
| --- |
|  |

# Design

## Android Application Class Diagram

## C:\Users\Connor\Desktop\Android App.JPG

## Android Application Sequence Diagrams

# C:\Users\Connor\Desktop\GPS Tracking.JPG

# C:\Users\Connor\Desktop\SMS Receiver.JPG

# Implementation

For the first increment, we implemented the bulk of the first two web services, along with the underlying database structure. We purposefully created this backend first, so that we could begin work on the UI without interruption, and to allow more thorough testing of the individual services and UI functionality.

## Database

These are the changes that have been made to the database tables since iteration 1.

### Texts

This is a straightforward table containing text messages sent by a student. It can be retrieved by parents since they will have their student’s ID.

**id(int)**

Account ID for this student.

**sender(int)**

Name or number of the sender.

**message(int)**

Message body.

## StudentDataService

This is the largest web service planned for our application. This is where all of the school data is retrieved and added: classes, assignments, grades, etc. To allow easier manipulation, several serializable classes were created, which effectively correspond to their SQL tables.

### Assignment

// Correspond to C# equivalents of the Assignments columns

public int mAssignment;

public int mCourseID;

public string mName;

public string mDescription;

public float mPoints;

public DateTime mDueDate;

public bool mIsTest;

### Infraction

// Correspond to C# equivalents of the Infractions columns

public int mInfractionID;

public int mStudentID;

public StudentDataService.INFRACTION\_TYPE mInfractionType;

public string mDescription;

public DateTime mDate;

### Grade

// Correspond to C# equivalents of the Grades columns

public int mGradeID;

public int mAssignmentID;

public int mStudentID;

public double mPointsEarned;

public DateTime mDateSubmitted;

### Functions

public bool addClass

(

int aTeacherID, // Teacher ID to verify and to have teaching the class

string aPassword, // Teacher's password

string aClassName // Name of the class to add

)

public bool addStudentToClass

(

int aTeacherID, // Teacher ID used to verify

string aPassword, // Teacher's password

int aClassID, // Class ID to add student to

int aStudentID // Student to add

)

public List<int> getClasses

(

int aStudentID // Student ID to get classes for

)

This retrieves the classes for a student. Currently it simply returns the ID numbers of the classes, but we plan to modify this to return Class objects, like getAssignments below, to make it more robust.

public List<Assignment> getAssignments

(

int aClassID // Class ID to get assignments for

)

This returns a list of Assignment objects. Because Assignments are serializable and we have the GenerateScriptType on our WebService, these should be readable from other languages’ web service calls.

public bool addGrade

(

int aTeacherID, // Teacher ID to verify

string aPassword, // Password for teacher to verify

int aStudentID, // Student ID to add grade for

int aAssignmentID, // Assignment ID to add grade to

double aPointsReceived, // Points the student received

DateTime aDateSubmitted // Date the student submitted it

)

public List<Grade> getGrades

(

int aStudentID, // Student ID to get grades for

int aClassID // class ID to get grades for

)

public bool addInfraction

(

int aTeacherID, // Teacher ID to verify

string aPassword, // Password to verify

int aStudentID, // Student ID to add infraction to

INFRACTION\_TYPE aInfractionType, // Type of infraction

string aDescription, // Description of infraction

DateTime aDate // Date the infraction occurred

)

public List<Infraction> getInfractions

(

int aStudentID // Student ID to get infractions for

)

public int getParentID

(

int aStudentID // Student ID to get the parent for

)

public int getStudentID

(

int aParentID // Parent ID to get student for

)

protected bool verifyTeacher

(

int aTeacherID, // Teacher ID to verify

string aPassword, // Teacher's password

SqlConnection aConnection // Connection to use. Caller must open and close it

)

This is not a web service, but it is widely used. For all of the “addX” services, we call this to verify that the teacher’s identification is valid. This is used to prevent other users from accessing functionality which should only be available for teachers.

protected bool generateGenericInsertCommand

(

SqlConnection aConnection, // Connection to execute command with

params string[] aList // Variable length list of parameters

)

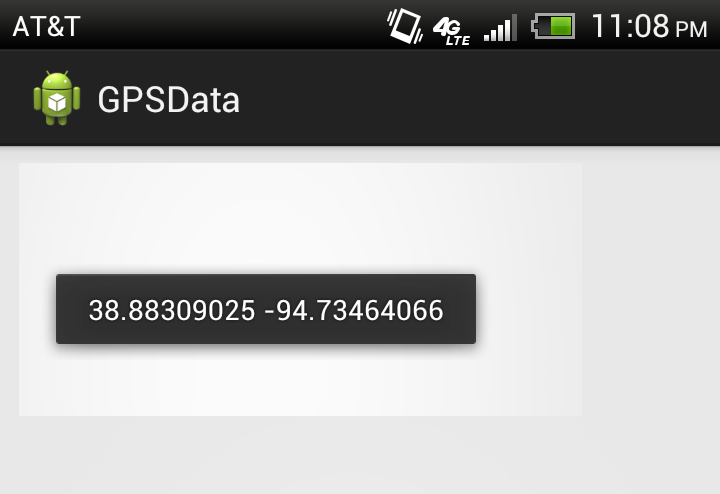
This is not a web service, but it is used for most of the “addX” services. It was used to make generating the command string in a simpler fashion.

# Testing

Android API features were tested by installing feature-specific apps on an android device and seeing that the core functionality of the features work correctly.

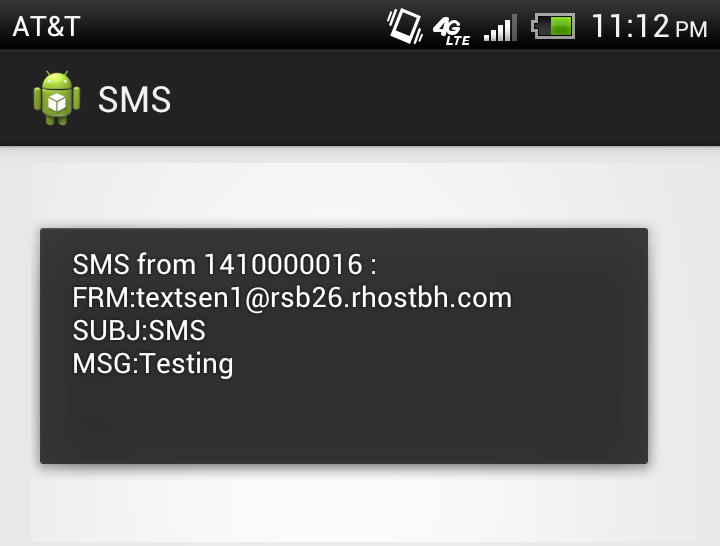
### GPS Tracking

The current implementation consists of a defined call-back function that is asynchronously called in the main application activity when the user has changed location by a magnitude of at least 10 yards. The sensitivity will be less accurate in future iterations and the power consumption will be low. Using GPS requires permission to view location in the project manifest.



### SMS Receiving

This application watches for Android system broadcasts and requires permission to read and receive SMS messages in the project manifest. The application receives the message before the Android operating system and proceeds to decrypt it, showing the sender and message content.



# Project Management

We have no issues or concerns with our project’s progress so far. We were able to complete virtually all of the tasks that were originally planned for Iteration 1. The only tasks we did not complete were either determined unnecessary, or we decided it would be more appropriate to work on at a later date. In addition, we completed several tasks from Iteration 2, which opens up additional time in our schedule to potentially work on some additional features.

## Work Completed

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Product | Iteration | | | Task | User | Spent effort (hours) |
| SternerLearn | | | Iteration 1 | Create Accounts Database table | Devin Turner | 0.50 |
| SternerLearn | | Iteration 1 | | Create Assignments Database table | Devin Turner | 0.50 |
| SternerLearn | | Iteration 1 | | Create Classes Database table | Devin Turner | 0.50 |
| SternerLearn | | Iteration 1 | | Create Grades Database table | Devin Turner | 0.50 |
| SternerLearn | | Iteration 1 | | Create Infractions Database table | Devin Turner | 0.50 |
| SternerLearn | | Iteration 1 | | Create Iteration 1 Report | Devin Turner | 4.00 |
| SternerLearn | | Iteration 1 | | Create Meta Database table | Devin Turner | 0.50 |
| SternerLearn | | Iteration 1 | | Create Notifications Database table | Devin Turner | 0.50 |
| SternerLearn | | Iteration 1 | | Create Students Database table | Devin Turner | 0.50 |
| SternerLearn | | Iteration 1 | | Create UML documentation | Connor Ledgerwood | 6.00 |
| SternerLearn | | Iteration 1 | | End-to-end testing | Connor Ledgerwood | 6.00 |
| SternerLearn | | Iteration 1.5 | | Create AccountManagement::Login Web Service | Connor Ledgerwood | 4.00 |
| SternerLearn | | Iteration 1.5 | | Create AccountManagement::Register Web Service | Connor Ledgerwood | 4.00 |
| SternerLearn | | Iteration 1.5 | | Create StudentData::addAssignment Web Service | Devin Turner | 2.00 |
| SternerLearn | | Iteration 1.5 | | Create StudentData::addClass Web Service | Devin Turner | 2.00 |
| SternerLearn | | Iteration 1.5 | | Create StudentData::addStudentToClass Web Service | Devin Turner | 2.00 |
| SternerLearn | | Iteration 1.5 | | Create StudentData::addTeacher Web Service | Devin Turner | 2.00 |
| SternerLearn | | Iteration 1.5 | | Create StudentData::getAssignments Web Service | Devin Turner | 2.00 |
| SternerLearn | | Iteration 1.5 | | Create StudentData::getClasses Web Service | Devin Turner | 2.00 |
| SternerLearn | | Iteration 2 | | Create StudentData::addGrade Web Service | Devin Turner | 2.00 |
| SternerLearn | | Iteration 2 | | Create StudentData::addGuardian Web Service | Connor Ledgerwood | 2.00 |
| SternerLearn | | Iteration 2 | | Create StudentData::addInfraction Web Service | Devin Turner | 1.00 |
| SternerLearn | | Iteration 2 | | Create StudentData::getGrades Web Service | Devin Turner | 2.00 |
| SternerLearn | | Iteration 2 | | Create StudentData::getInfractions Web Service | Devin Turner | 1.00 |
| SternerLearn | | Iteration 2 | | Create StudentData::getParentID Web Service | Devin Turner | 1.00 |
| SternerLearn | | Iteration 2 | | Create StudentData::getStudentID Web Service | Devin Turner | 1.00 |

## Work to be Completed

There are only two tasks in Iteration 1 which were not completed. Both of these tasks are part of the Text Message feature, which will allow guardians to view their student’s text messages. Because we have not begun development of our mobile application for Iteration 1, we decided to wait to implement the Text Message database table and corresponding web services until then. Once we determine exactly what information Android will be able to provide about the text message data, we will be in a better position to create the table and services.

# Deployment

## Project Website

<http://170.224.169.101/lab5/main.html>

## Agilefant

<https://cloud.agilefant.org/evaluator102/login.jsp>

To log in, use the username “professor” and password “password”.

## Web Services

<http://170.224.169.101/Iteration1/AccountService.asmx>

<http://170.224.169.101/Iteration1/StudentDataService.asmx>

## GitHub

<http://github.com/clkv5/cs551_project/>