|  |
| --- |
| 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

Most of our services have already been created in Iteration 1, but we did some refactoring, added some tables, and web services.

## 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.

## TableDisplay

Because we are now deploying the database to our cloud instance, it will become more difficult to view the database contents conveniently. In this light, we created the TableDisplay page, which displays the contents of all of the tables in the database. Then, instead of needing to remotely log on to the server and look at the table, we will simply be able to go to this website and conveniently view them.

This also supports hiding and showing specific tables in the database. Particularly once the database becomes larger, we will not want to display multiple tables at the same time, and so this improves the convenience of the tool.

## ParentalManagementService

This service was added as part of Iteration 2. Currently it only contains the location service management, where locations can be added and retrieved.

### Location

// Correspond to C# equivalents of the Locations columns

public int mLatitude;

public int mLongitude;

public DateTime mTime;

### Functions

public bool addLocation

(

int aStudentID,

string aPassword,

int aLatitude,

int aLongitude,

DateTime aTime

)

addLocation will simply add a new location for the student at the specified time and location.

public List<Location> getLocations

(

int aStudentID,

DateTime aStartTime,

DateTime aEndTime

)

getLocations is provided a start and end time for a given student. It then retrieves all of the Location objects in the given time frame.

## StudentDataService

The only changes to this service for this iteration were code refactoring, to improve the quality and reusability of the code. The largest changes are described here.

### Course

// Correspond to C# equivalents of the Course columns

public int mCourseID;

public int mStaffID;

public string mClassName;

public List<Course> getClasses

(

int aStudentID // Student ID to get classes for

)

Previously getClasses was only returning a list of course IDs. However, to provide more information to the caller, we modified it so that it also returns a Course object with the actual table data.

### Serializable Object Constructors

Previously our objects had several constructors, taking the data required from a given row of the table. This made callers have to use the SqlDataReader objects themselves, which introduced a clear possibility of code duplication. So instead, we passed the SqlDataReader to the objects and let the objects themselves use the reader to obtain the data. An example for the Course object is shown below.

public Course

(

SqlDataReader aReader // Reader to get data from

)

{

int? aCourseID = aReader["courseID"] as int?;

int? aStaffID = aReader["staffID"] as int?;

string aClassName = aReader["className"] as string;

mCourseID = aCourseID.HasValue ? aCourseID.Value : -1;

mStaffID = aStaffID.HasValue ? aStaffID.Value : -1;

mClassName = aClassName;

}

## StudentDataService

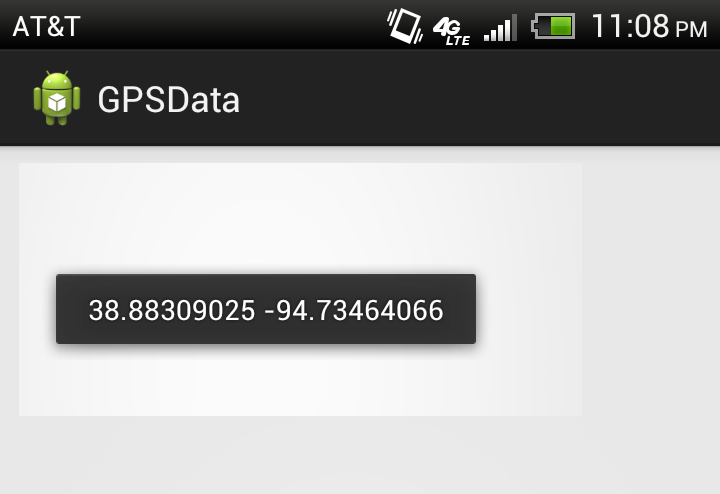
Add your changes here

# 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.

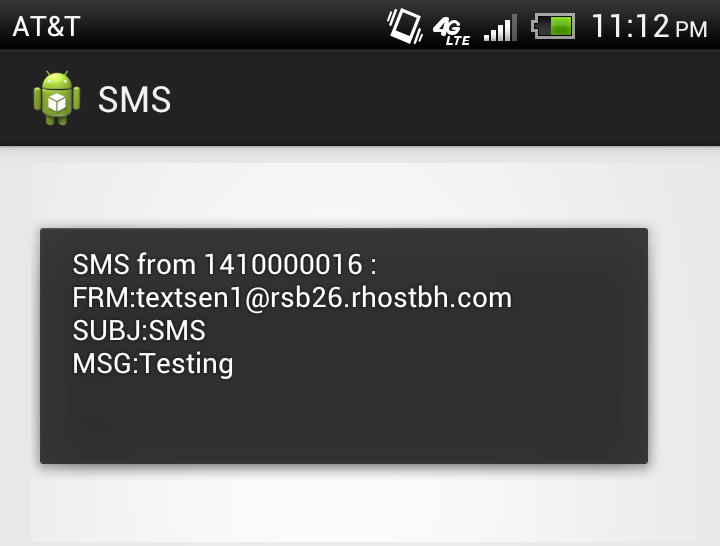
### GPS Tracking

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.



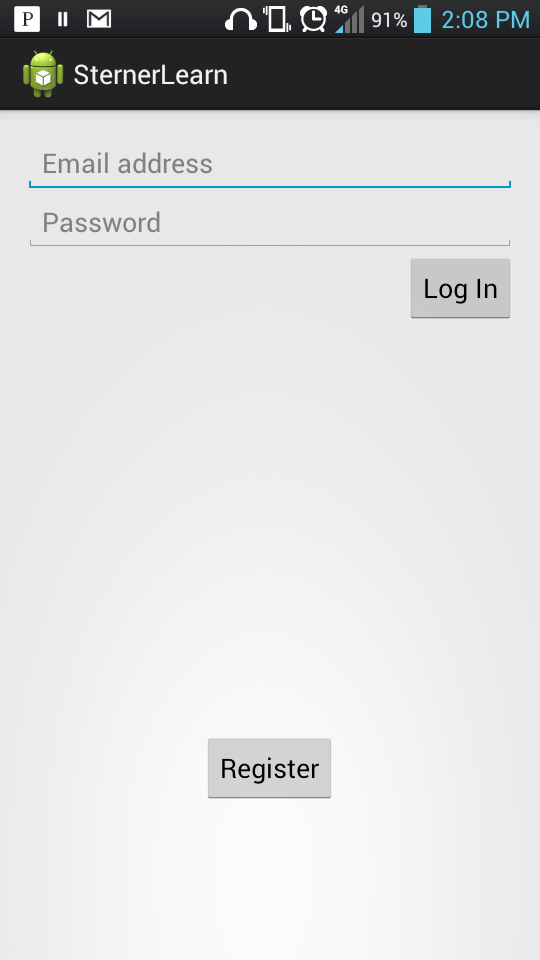
### SMS Receiving

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.

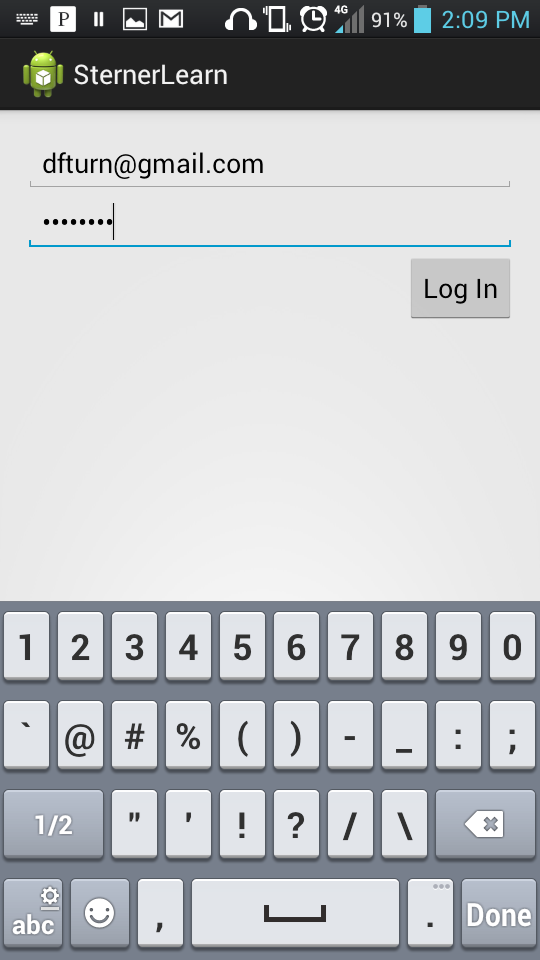


### SternerLearn Application

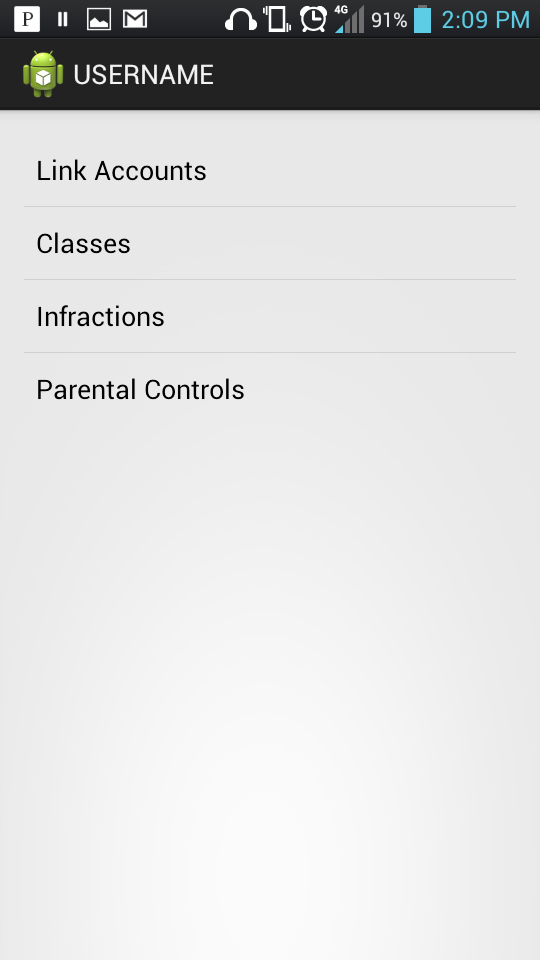
We have begun development on the skeleton of the UI for the SternerLearn application. We decided to develop in native android because of the significant demands on the operating system we will require (GPS, SMS, application data, etc). Below are some screenshots of the current UI along with explanations.



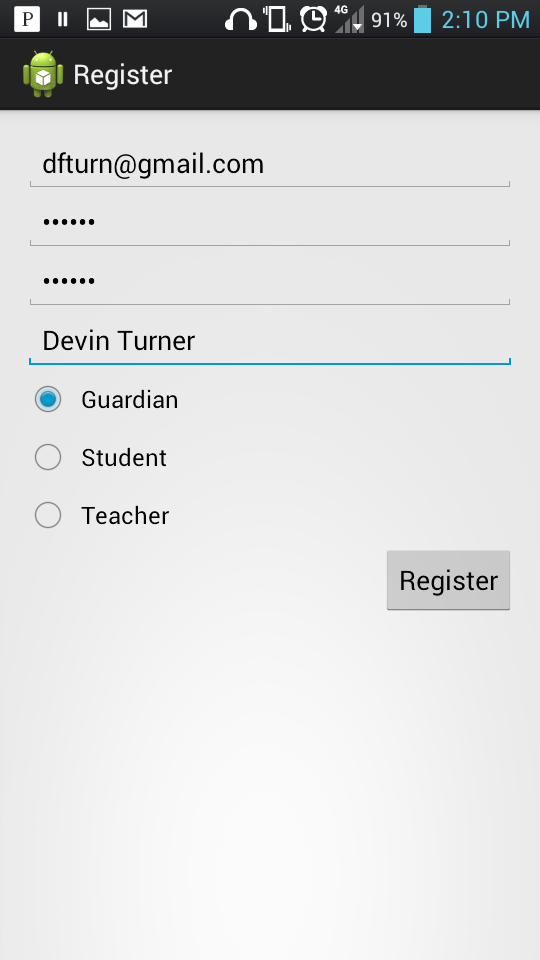
When the user first enters the application, it will display a login screen. Once they log in once, the application will save their username and password to automatically log them in for future uses.



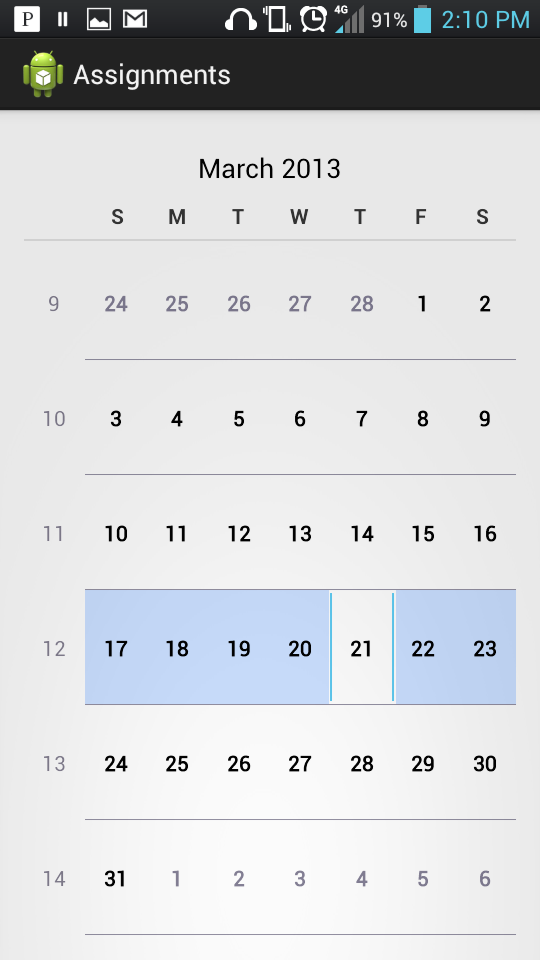
The text entry fields have attributes which specify how they should be used. For instance, the email keyboard displays “@” and “.com”, and the password field displays the dots instead of characters.



This is the skeleton of a parent’s main account page. This will be fleshed out once our database is deployed to the cloud instance and we can retrieve the data needed to be displayed on each of these pages.



On the register page, the user can enter the information required to create a new account. This data will be sent to the server and entered into the database.



The assignments page currently just uses a CalendarView to display a basic calendar. However, the built-in Android CalendarView does not support everything that we require. We will have to either develop a new calendar view widget, or use an open-source one. Optimally we would like a calendar display similar to the Gmail Android application, but with our custom data.

# Project Management

We completed nearly all of the tasks originally scheduled for Iteration 2, and were able to add several new tasks to the iteration, as well as completing several tasks from Iteration 3. Our original plan did not have us starting on the application development until Iteration 3, and we decided that this was too delayed for the rapid development involved in this project. So we began development on the application UI, using native Android, and also did some research on the Android OS capabilities that our application will require.

## Work Completed

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Product | Iteration | Task | User | Spent effort (hours) | |
| SternerLearn | Iteration 2 | Android GPS access research | Connor Ledgerwood | | 4.00 |
| SternerLearn | Iteration 2 | Android root access research | Connor Ledgerwood | | 4.00 |
| SternerLearn | Iteration 2 | Android text messages research | Connor Ledgerwood | | 4.00 |
| SternerLearn | Iteration 2 | Create ParentalManagement::addLocation Web Service | Devin Turner | | 1.00 |
| SternerLearn | Iteration 2 | Create ParentalManagement::getLocations Web Service | Devin Turner | | 1.00 |
| SternerLearn | Iteration 2 | Create web page to conveniently display database tables while on the cloud instance | Devin Turner | | 2.00 |
| SternerLearn | Iteration 2 | Deploy database to cloud instance | Connor Ledgerwood | | 12.00 |
| SternerLearn | Iteration 2 | End-to-end testing | Devin Turner | | 1.00 |
| SternerLearn | Iteration 2 | End-to-end testing | Connor Ledgerwood | | 1.00 |
| SternerLearn | Iteration 3 | Create Base page with Login/Register buttons | Devin Turner | | 3.00 |
| SternerLearn | Iteration 3 | Create Login page | Devin Turner | | 4.00 |
| SternerLearn | Iteration 3 | Create Register page | Devin Turner | | 3.00 |
| SternerLearn | Iteration 3 | Create main account page | Devin Turner | | 4.00 |

## Work to be Completed

There were only two tasks not completed for this iteration. The first, making a teacher website for adding grades, was deferred to a later iteration. We wanted to make sure we had a fully featured application for Android as soon as possible, so we decided this was an optional task, and would only be completed if we had time after the application was essentially complete. The second task was postponed. The research into disabling applications is still continuing, but it is not clear whether this feature will be possible on an un-rooted Android phone. If this feature is not possible, we will replace it with another one.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Product | Iteration | Task | User | Effort remaining |
| SternerLearn | Iteration 2 | Create teacher website to conveniently add grades etc. | Devin Turner | 16 |
| SternerLearn | Iteration 2 | Create applications disabled database and web service | Connor Ledgerwood | 16 |

# 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>

Need to add the ParentalManagementService and the TableDisplay here (and update the services that have already been deployed)

## GitHub

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