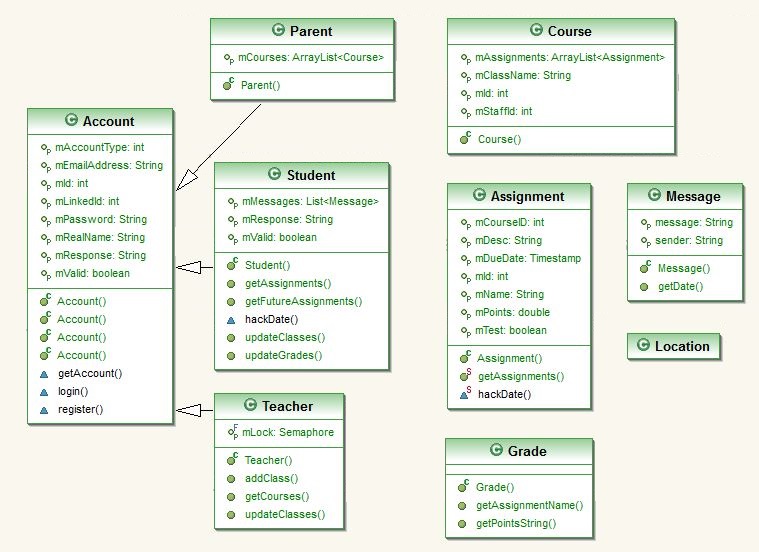
|  |
| --- |
| University of MISSOURI-kansas city |
| SternerLearn |
| Increment 3 Report |
|  |
| **Connor Ledgerwood and Devin Turner** |
| **4/12/2013** |

|  |
| --- |
|  |

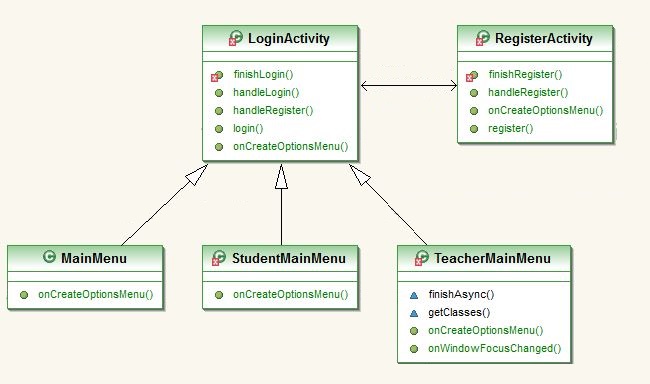
# Design

## Android Application Class Diagram

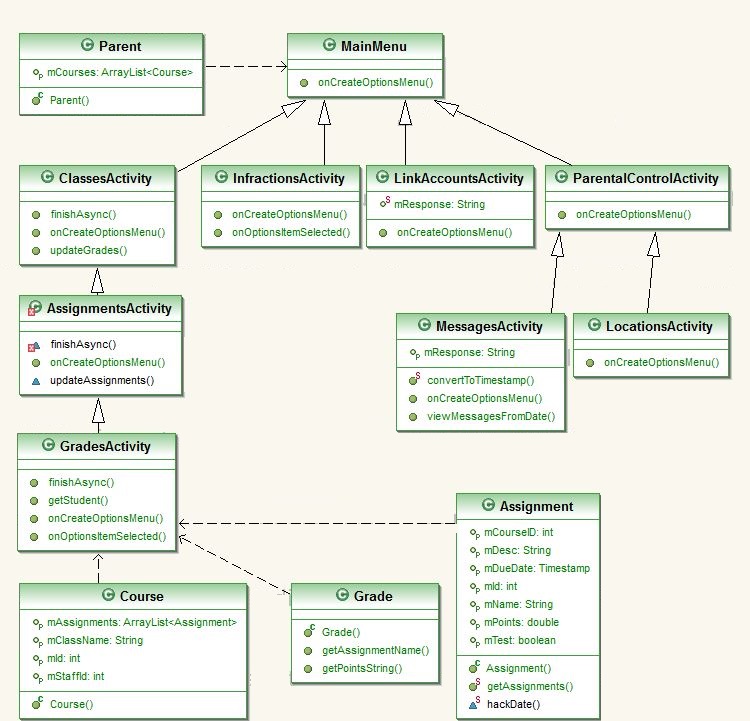
### Class Files



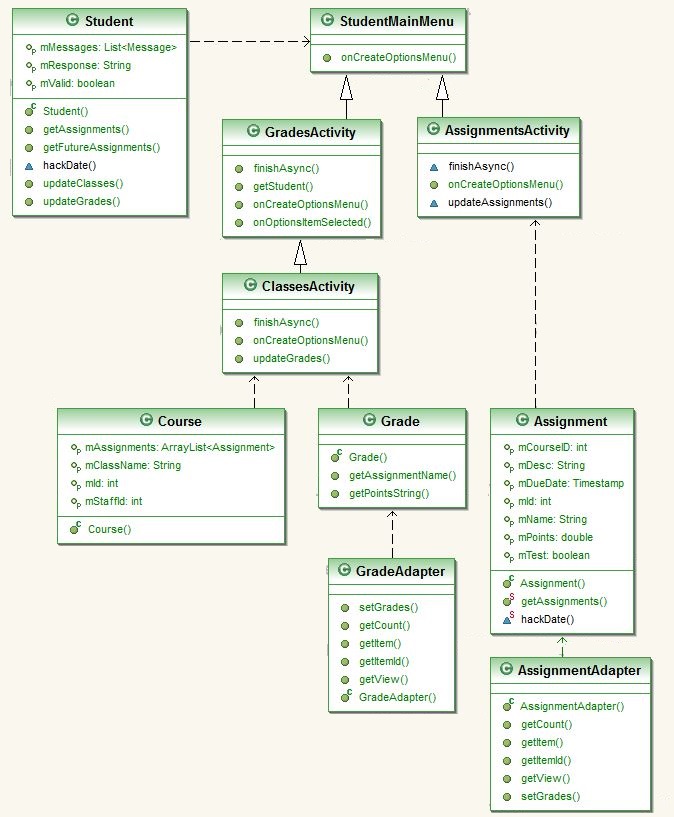
### Login Structure



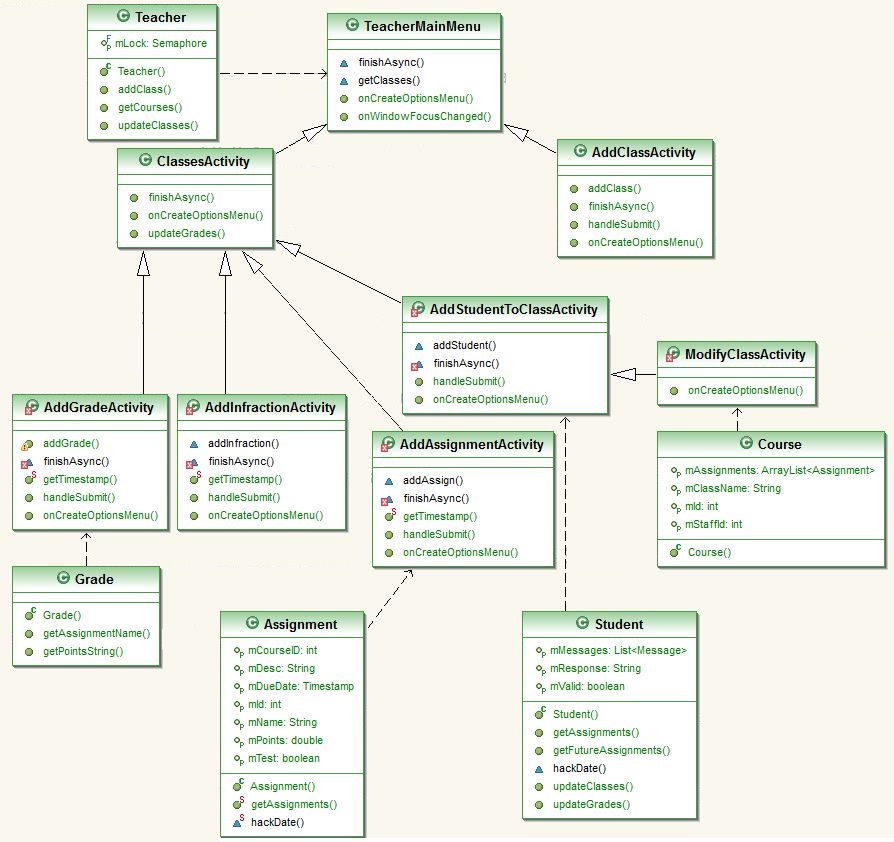
### Parent-based Application



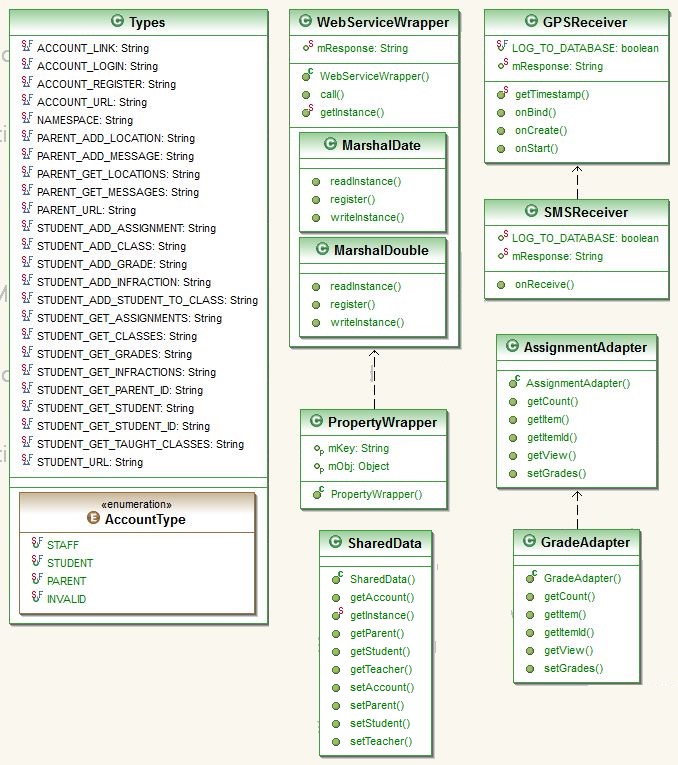
Student-based Application



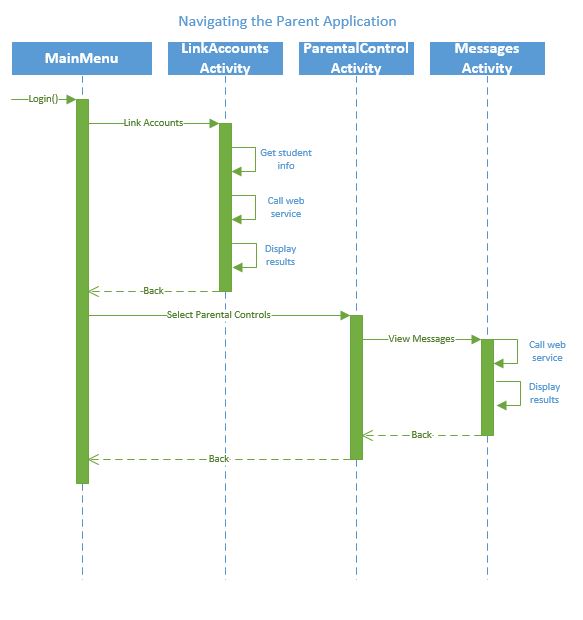
### Teacher-based Application

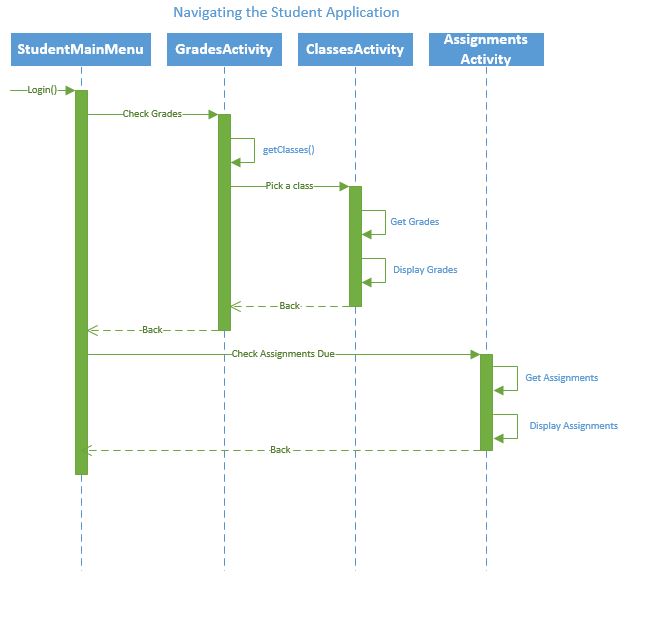


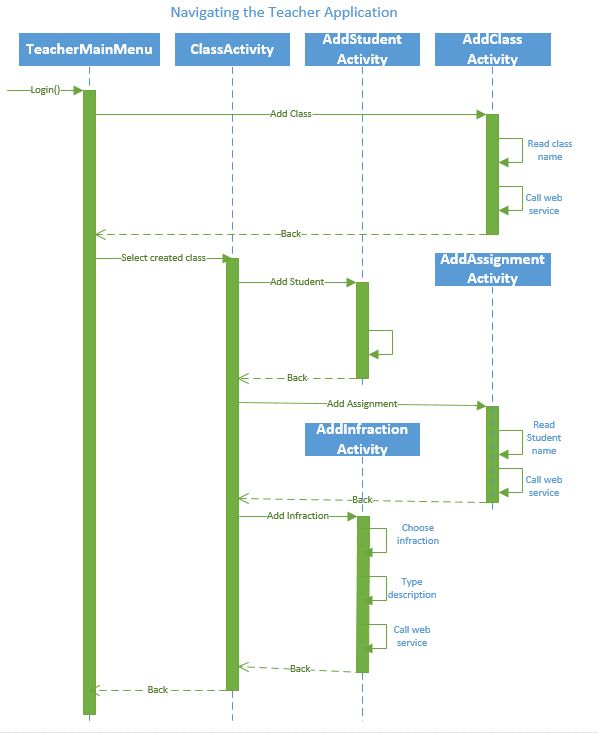
### Back End

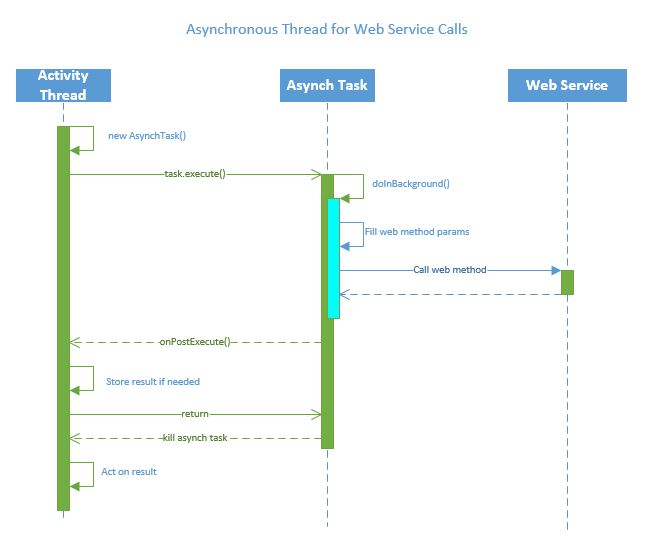


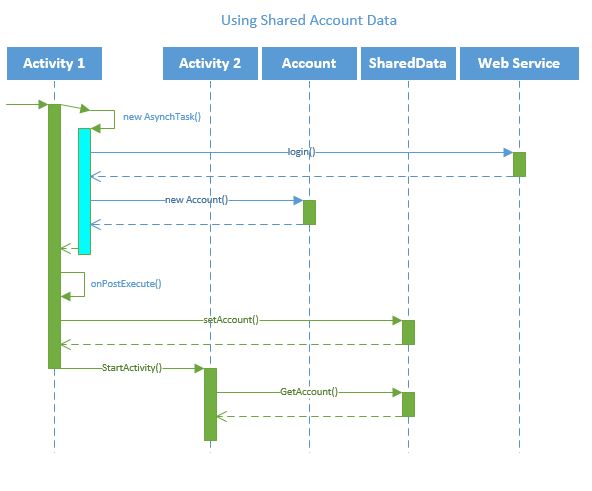
## Android Application Sequence Diagrams











# Implementation

The bulk of the work for this iteration was in the creation of the mobile application. The web service was only modified to support new pages that we had not anticipated in the original scope of the project, or to fix small bugs that we discovered. Below we list the software changes that occurred since Iteration 2.

## Database

The only change here was in a small modification to the “Text” table, which was renamed to “Messages” and given more columns.

### Messages

**studentID(int)**

Account ID for this student.

**sender(string)**

Name of the sender.

**message(string)**

Message body.

**time(datetime)**

The time at which the message was sent.

## AccountService

### Account

This class was created to serialize the contents of a login or register call to provide the caller with the full information of the account they are using.

### Register

This web method now returns a serialized account that was generated from the function call, instead of hardcoded strings.

### Login

This method also returns an Account object.

## ParentalManagementService

This web service was simply changed to update it for the new Messages table.

### Message

This serializable class stores the contents of a row from the Messages table to return to callers.

Two new web methods were also created to set and get messages.

public bool addMessage

(

int aStudentID, // Student ID to associate the message with

string aPassword, // Password for the student

string aSender, // Sender’s name

string aMessage, // Message contents

DateTime aTime // Time the message was sent

)

public List<Message> getMessages

(

int aStudentID, // Student ID to get messages for

DateTime aStartTime, // Start time to retrieve messages

DateTime aEndTime // End time to retrieve messages

)

## StudentDataService

This web service received a couple new web methods and one bug fix. The bug was in addStudentToClass, and it fixed multiple rows (in this case students) being affected by a single call.

public List<Course> getTaughtClasses

(

int aTeacherID // Teacher ID to get classes for

)

getTaughtClasses was added to retrieve the list of courses a teacher was teaching. This was required for the teachers to be able to affect their classes using the UI in any way. Using this, the application is able to display a teacher’s courses and let them modify them individually.

public int getStudent

(

int aTeacherID, // Teacher ID used to verify

string aPassword, // Teacher's password

string aStudentName // Student's name

)

getStudent was required because there was no way to go from a name to an identification number, so there was no way for a teacher to get any information about a student, since they didn’t have access to the identification numbers.

## Application

Nearly all of our work for this iteration was contained in the application. We created many new pages of the UI, added functionality, and added the GPS and SMS services.

### Objects

#### Account

These objects represent individual accounts. They store the exact same data as the Accounts from the web service, but in the Java language.

#### Assignment

These represent individual assignments, and also correspond to the web service assignments. Some web service calls are located here to retrieve a list of assignments for a course.

#### Course

This represents an individual course, and simply stores the data.

#### Grade

This stores the grade data as well as the Assignment object associated with this grade.

#### Location

This stores the same data as Locations from the web service.

#### Message

This stores the same data as Messages from the web service.

#### Parent

A parent currently only stores an account object, but will be extended to store more parental control data for their student.

#### Student

A student contains all of the data associated with the student. This means the account, courses, grades, and assignments. The grades and assignments are stored in a hash map with the courses being the keys, and a list of assignments or grades, respectively, being the values. The web service calls to retrieve data for a student are within the Student class itself.

#### Teacher

A teacher contains the data associated with a student. The main web service functionality stored within the teacher itself is the ability to update their list of classes. Because of a threading issue with updating the page while retrieving new classes, this is now semaphore protected.

### List Adapters

List adapters are used in Android to lay out list items. Because we wanted to display more than a single string per list item to the user, we chose to override the basic list adapters to lay out more data per list item.

#### GradeAdapter

This is for the list of grades This will display the assignment name and the points received out of the total.

#### AssignmentAdapter

This is for a list of upcoming assignments. This displays the assignment name, description, total points, and due date, all in a single list item.

### Pages

We chose to use a single Android Activity for each of our pages. This simplified much of the logic and design, and does not seem to detrimentally affect the performance. Each of these activities has an associated XML layout file, which is not described here, but can be accessed in our git repository. Nearly all of these pages required web service calls, and so required the use of a background task. We chose to use the AsyncTask for this, which can retrieve the data on a background thread, and then return it to be accessed by the UI thread. This was the design scheme of all of the pages.

#### AddAssignmentActivity

This page is for teachers to add a new assignment to a particular class. It allows them to select the name, description, points, and due date.

#### AddClassActivity

This page is for teachers to add a new class. It allows them to select the name of the class.

#### AddGradeActivity

This page is for teachers to add a new grade to a particular assignment and student. It allows them to select the name of the student, assignment, and what they received.

#### AddInfractionActivity

This page is for teachers to add a new infraction to a particular student. It allows them to select the type of infraction, the student receiving the infraction, and an additional description of the infraction.

#### AddStudentToClassActivity

This page allows teachers to add a student to a given class.

#### AssignmentsActivity

This page provides a list of upcoming assignments for a student, which are laid out using the AssignmentAdapter.

#### ClassesActivity

Contrary to its name, this page lists the grades the student has received for the particular class. The list items are laid out with the GradeAdapter.

#### GradesActivity

Also contrary to its name, this page lists the classes the student is participating in. By clicking on a class, they are taken to the ClassesActivity for that class.

#### InfractionsActivity

This page displays the infractions a student has received. It is still a work in progress.

#### LinkAccountsActivity

This page allows a parent to enter passwords for themselves and their student to link their accounts. This allows the parent to access their student’s data.

#### LocationsActivity

This page is still under construction. For iteration 4 we plan to display the locations the student was at on a map view for the parent.

#### LoginActivity

This page simply lets the user log in with their username and password. They can also select to register a new account.

#### MainMenu

This is actually the parent’s main menu page, which allows them to select from a pre-generated list of options to view data on their student.

#### MessagesActivity

This page will contain a list of messages from the student’s phone, viewable by the parent. This page is still a work in progress.

#### ModifyClassActivity

This page lists the options for a teacher to edit a class. They can either add an infraction to one of the students, add a student to the class, add an assignment to the class, or assign grades. Clicking on each brings them to the appropriate new page to enter the information.

#### MyListActivity

This was a basic ListActivity created to allow classes with pre-generated list items to more easily add them to the page. It is used by ModifyListActivity and the three main menu pages.

#### ParentalControlActivity

This page displays the options for a parent to view data regarding their student. This is currently just a skeleton.

#### RegisterActivity

This allows the user to enter all of the information required to register a new account. If the registration is successful, they are taken to the main menu for that new account.

#### StudentMainMenu

This is the student’s main menu page. It allows them to view their grades or upcoming assignments.

#### TeacherMainMenu

This is the teacher’s main menu. This simply has a list of their current classes. Clicking on one will take them to the ModifyClassActivity.

### Services

These services will only be activated for student type accounts, and will continually run in the background after the student is logged in.

#### GPSReceiver

This logs their locations continually and as they move. The locations are sent to the server with a web service, and will later be retrievable by the parent to display on their Android device.

#### SMSReceiver

This catches new text messages as they are received, and uploads them to the server.

### Utility

These are utility classes, which are used throughout the application to assist with various tasks.

#### PropertyWrapper

This is simply a wrapper for the SOAP object properties. To allow more convenient manipulation and sharing of these properties, a list of PropertyWrappers can be created.

#### SharedData

This is where all of the data that should be shared between activities is stored. It is a singleton that stores the current account information, such as student, teacher, or parent specific data.

#### Types

Shared strings and enumerations are stored here to be more easily accessed by all of the activities and objects.

#### WebServiceWrapper

This is where the core of the web service usage occurs. Using the call() function with a web service URL, method name, and list of PropertyWrapper, it will return the envelope which contains the data the caller requires.

# Testing

Because of the nature of our application and how web service connectivity is essential to our application’s functionality, we were unable to use NUnit for automated testing. Instead we manually tested all of our web services and application UI. Not only did we continually test pages and services as they were being created and used in our application, we also did a round of end-to-end testing at the completion of development for Iteration 3. We believe this to have been sufficient for our purposes.

### Application UI

### Web Service Calls

When creating the application UI, we were able to more thoroughly test the web services we previously created. To our surprise, we only discovered one bug in the addStudentToClass service. This was due to a misunderstanding of the SqlDataReader, and was promptly fixed.

### Threading

During implementation of the TeacherMainMenu, a flaw in the AsyncTask usage was discovered. This necessitated thorough testing of all of the activities which used AsyncTask to make sure there was no modification of data from underneath the UI thread. To fix the issue in the TeacherMainMenu, semaphore protection was introduced.

### Other

# Project Management

## Work Completed

We completed far more work than originally scheduled for Iteration 3. Originally we were only planning on completing perhaps 1/3 to 1/2 of the user interface, but after recommendations from the TA and professor we decided to complete as much as possible in the Iteration 3, rather than Iteration 4. As a result, we completed about ¾ of the UI. Both the Teacher and Student portions of the UI are fully functional. They allow convenient viewing of classes, grades, assignments by the students. The teachers are able to add new classes, add students to their classes, add assignments, assign grades, and give out infractions. The parents UI framework is there, and they are able to link accounts with a child, and view their child’s text messages. In total we spent 96 hours of work on Iteration 3, about twice as much as Iteration 2.

|  |  |  |  |
| --- | --- | --- | --- |
| Project | Task | User | Spent effort (hours) |
| SternerLearn | Create Add Class page | Devin Turner | 3.00 |
| SternerLearn | Create Add Grade page | Devin Turner | 3.00 |
| SternerLearn | Create Add Infraction page | Devin Turner | 4.00 |
| SternerLearn | Create Add Student to Class page | Devin Turner | 4.00 |
| SternerLearn | Create Assignments list page | Devin Turner | 6.00 |
| SternerLearn | Create Classes page | Devin Turner | 4.00 |
| SternerLearn | Create GPS Receiver service | Connor Ledgerwood | 8.00 |
| SternerLearn | Create Grades page | Devin Turner | 6.00 |
| SternerLearn | Create Link Accounts page | Connor Ledgerwood | 6.00 |
| SternerLearn | Create SMS Receiver service | Connor Ledgerwood | 6.00 |
| SternerLearn | Create Text Messages list page | Connor Ledgerwood | 8.00 |
| SternerLearn | Create UML documentation | Connor Ledgerwood | 4.00 |
| SternerLearn | Create additional web services that were required | Devin Turner | 2.00 |
| SternerLearn | Create additional web services that were required | Connor Ledgerwood | 2.00 |
| SternerLearn | Create main account page | Devin Turner | 2.00 |
| SternerLearn | Create report | Devin Turner | 4.00 |
| SternerLearn | Create report | Connor Ledgerwood | 4.00 |
| SternerLearn | Create teachers' Class list page | Devin Turner | 4.00 |
| SternerLearn | End-to-end testing | Devin Turner | 8.00 |
| SternerLearn | End-to-end testing | Connor Ledgerwood | 8.00 |

## Work to be Completed

For Iteration 4, we plan to complete the UI of our mobile application. The bulk of the work will be in completing the Parent section, and in particular the Location tracker page. We plan to use the Google Maps API available on the Android platform to create a map view of the student’s locations. We will also refine the pages which were created in Iteration 3, and ensure they are easy to use and bug free. In particular, allowing teachers to more easily add material to their classes. Currently they must type in the names of students or assignments to affect, but clearly it would be more convenient to allow them to select from a list. Of course, we will also begin on creating our video presentation of our application. More tasks will be created as we begin to do more comprehensive planning of the final iteration.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | State | Assignees | Effort Left |
| Create Tutorial (optional) | NOT\_STARTED | Devin Turner |  |
| Create Disable Apps page | BLOCKED | Connor Ledgerwood | 12 |
| Create Tracking Location page | NOT\_STARTED | Devin Turner | 16 |
| Refactoring code | NOT\_STARTED | Devin Turner | 8 |
| Completing full functionality of all pages from Iteration 3 | NOT\_STARTED | Devin Turner | 12 |
| Create video presentation | NOT\_STARTED | Connor Ledgerwood, Devin Turner | 16 |

# Deployment

## Agilefant

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

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

## Web Services

<http://170.224.169.101/Iteration3/AccountService.asmx>

<http://170.224.169.101/Iteration3/StudentDataService.asmx>

<http://170.224.169.101/Iteration3/ParentalManagementService.asmx>

## GitHub

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