# SmartSeat

**SmartSeat Bus System**

# Software Design Document

Ben Sewards

John Longo

John Piccolomini

Lab Section: COMP 566-02

Workstation: WENTW 210/DOBBS 308

Date: (02/19/2013)

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| [**1.**](#_bookmark0) | [1.1](#_bookmark1) | [**INTRODUCTION**](#_bookmark0)  [Purpose](#_bookmark1) | [**2**](#_bookmark0)  [2](#_bookmark1) |
|  | [1.2](#_bookmark2) | [Scope](#_bookmark2) | [2](#_bookmark2) |
|  | [1.3](#_bookmark3) | [Overview](#_bookmark3) | [2](#_bookmark3) |
|  | [1.4](#_bookmark4) | [Reference Material](#_bookmark4) | [2](#_bookmark4) |
|  | [1.5](#_bookmark5) | [Definitions and Acronyms](#_bookmark5) | [2](#_bookmark5) |
| [**2.**](#_bookmark6) |  | [**SYSTEM OVERVIEW**](#_bookmark6) | [**2**](#_bookmark6) |
| [**3.**](#_bookmark7) |  | [**SYSTEM ARCHITECTURE**](#_bookmark7) | [**2**](#_bookmark7) |
|  | [3.1](#_bookmark8) | [Architectural Design](#_bookmark8) | [2](#_bookmark8) |
|  | [3.2](#_bookmark9) | [Decomposition Description](#_bookmark9) | [3](#_bookmark9) |
|  | [3.3](#_bookmark10) | [Design Rationale](#_bookmark10) | [3](#_bookmark10) |
| [**4.**](#_bookmark11) | [4.1](#_bookmark12) | [**DATA DESIGN**](#_bookmark11)  [Data Description](#_bookmark12) | [**3**](#_bookmark11)  [3](#_bookmark12) |
|  | [4.2](#_bookmark13) | [Data Dictionary](#_bookmark13) | [3](#_bookmark13) |
| [**5.**](#_bookmark14) |  | [**COMPONENT DESIGN**](#_bookmark14) | [**3**](#_bookmark14) |
| [**6.**](#_bookmark15) |  | [**HUMAN INTERFACE DESIGN**](#_bookmark15) | [**4**](#_bookmark15) |
|  | [6.1](#_bookmark16) | [Overview of User Interface](#_bookmark16) | [4](#_bookmark16) |
|  | [6.2](#_bookmark17) | [Screen Images](#_bookmark17) | [4](#_bookmark17) |
|  | [6.3](#_bookmark18) | [Screen Objects and Actions](#_bookmark18) | [4](#_bookmark18) |
| [**7.**](#_bookmark19) |  | [**REQUIREMENTS MATRIX**](#_bookmark19) | [**4**](#_bookmark19) |
| [**8.**](#_bookmark20) |  | [**APPENDICES**](#_bookmark20) | [**4**](#_bookmark20) |

### INTRODUCTION

* 1. **Purpose**

This software design document describes the architecture and system design of the SmartSeat Bus System, intended to specify and give insight on the analyzed system to the System Developers, Application Developers, and Engineers on the team.

* 1. **Scope**

The given School will rely on a central database dedicated to each bus route mapped by the school. Real-time weight and the number of children for each bus stop on a corresponding bus route will be analyzed by the given hardware and transferred to the Bus System Application. From this onboard Application, the data will be compared with the central database to provide an accurate analysis for notification conditions on each bus route stop.

Along with these notification conditions will include real-time geo-location and updated accelerometer readings between bus stops that will compare against the central database for other parental notifications. These parental notifications will vary based on the design on the database, but should be considered a two part system that branches from pick-up and drop-off of students on the bus route. Keep in mind that both system branches will utilize the Bus System application, central database, and real-time systems.

SmartSeat will become beneficial in cases where Parents see the lack of supervision to and from school. This monitoring system will provide real-time up to date notifications to these given parents in terms of the corresponding bus route, when and if their child boards the bus, and approximation times for pick-up and drop-off to alleviate the unknowingness that a child is left behind.

## Overview

The Software Design Document (SDD) is a 7-piece documentation guide and reference for the Developers and Engineers. Starting from the Introduction, System Overview, System Architecture, Data Design, Component Design, Human Interface Design, and finally the Requirements Matrix, these 7 components will be thoroughly defined in each given section.

## Reference Material

*This section is optional.*

List any documents, if any, which were used as sources of information for the test plan.

## Definitions and Acronyms

*SOAP* | [Simple Object Access Protocol] is a simple XML-based protocol to let applications exchange information over HTTP.

*SQLite* | An express Database for storing local app data.

### SYSTEM OVERVIEW

The overall SmartSeat Bus System will require 4 sub-system packages:

1. Student Identification Retrieval System (SIRS)

- Retrieves Student data via communication between the Hardware ID System.

2. On-Board Bus System Application (OBBS)

- Interprets the above system, comparing to the central database to detect all possible conditions.

- On Drop-off BusRoutes, a sub-service called DropOffMappingService will be utilized to detect delays and accelerometer readings between BusStops.

3. Notification Service System (NSS)

- All conditional data will trigger different notification schemes based on the services listed above. The notification schemes are sent to the Bus Application and to the Mobile SmartSeat for Parents System.

4. SmartSeat for Parents [Mobile System]

- A Mobile notification system that is a read-only service for parental advisory.

- The NSS will provide data communication to this system

- This system will have its own local SQLite Database to store local data sent to the system.

### SYSTEM ARCHITECTURE

* 1. **Architectural Design**

In this section, the system modules stated above in the System overview will be explained, including their responsibilities, collaborations with the other modules, and desired functionality.

The main overview of the systems will be displayed graphically to represent module flow.

SIRS [Student ID Retrieval System]

*Responsibilities*

- Since the hardware aspect of SmartSeat isn’t in the scope specified by the SRS, raw controlled data will act as the hardware system to identify fake students and then to be interpreted by the SIRS as data that matches the central DB criteria.

*Functionality*

- Data interpreted by this system will need to be processed with data type validation and stored temporarily and eventually retrieved by the OBBS.

*Collaborations*

- Interaction between the central DB will be necessary in order to compare the raw data at every bus stop between drop-off and pick-up intervals.

- The OBBS below is dependent on the verification of students via the SIRS.

- The NSS below is also dependent on the verification per student.

OBBS [On-Board Bus System]

*Responsibilities*

- Drop-off bus routes will require real-time geo location in order to track the bus between bus stops.

*Functionality*

- All data presented by the geo-location API and accelerometer API will need to be processed, validated, and a reliable system will need to be in place in-case the use of the above API’s communication drop and data is unjustified to other systems.

- The accelerometer API will be used to predict travel intervals, so that when the bus starts up in these intervals, the geo-location API can be called to get the current location. This locational data is then sent to the NSS below, where it is verified and processed.

*Collaborations*

- The SIRS will initially send student data to this system.

- Data from this system will be sent to the NSS.

NSS [Notification Service System]

*Responsibilities*

- Before notifications can be sent to parents, we need to determine what kinds of notifications are being built through what system sending over data. This is where the NSS comes into play.

*Functionality*

- Acting as a data validation and notification service for various types of data, this data, in many forms, will need to be sent over to the SmartSeat for Parents with certain tags on the objects carrying the data.

- The service itself will need to include a wireless communication bridge to the SmartSeat for Parents that will be able to carry data reliable and in a timely manner.

*Collaborations*

- The two services above, SIRS and OBBS, will make service calls to this system.

- From this system, all notifications will be sent to the SmartSeat for Parents (and possibly back to the OBBS to be displayed differently on the OBBS interface).

SmartSeat for Parents [Mobile App System]

*Responsibilities*

- This mobile application has two sub-services. The ConnectMyStudent and RetrieveNotifications services.

- Both services functionality is described below.

*Functionality*

- ConnectMyStudent service initially triggers on start-up and is based on whether the Parent is in fact the Parent of the student. Once verified through some sort of authentication code emailed to the parent, this service will provide an authentication gateway to the next underlying service below and the mobile interfaces.

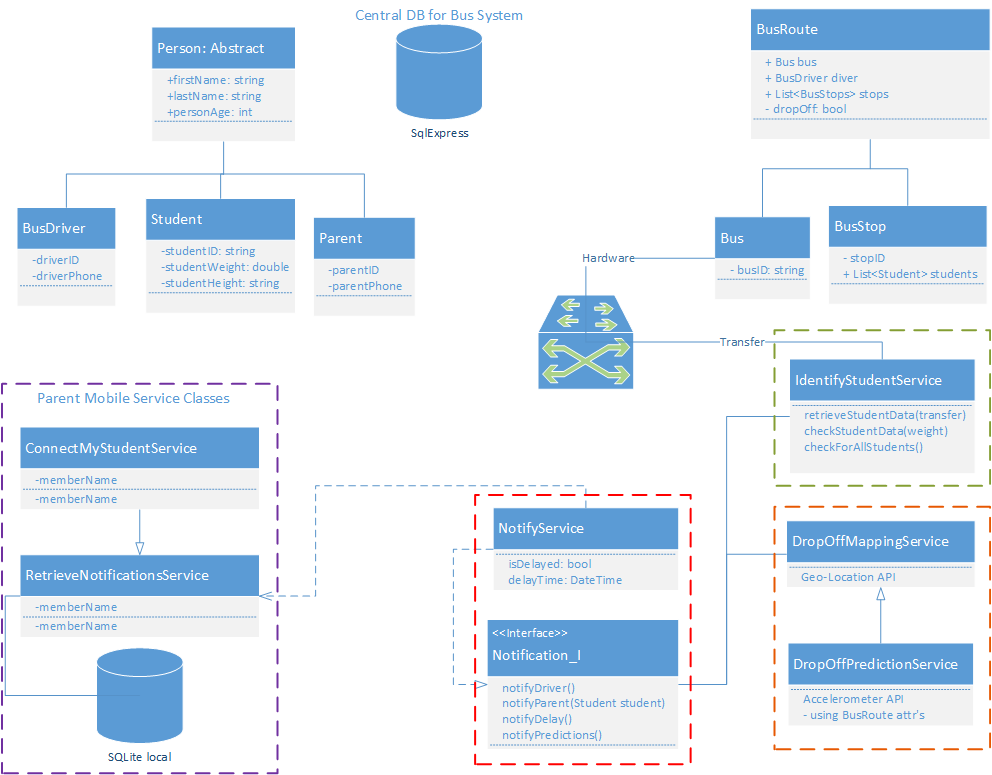
- RetrieveNotifications service is a responding service dedicated to receiving notification data from the NSS using a standard protocol of data communication that has yet to be verified.

*Collaborations*

- ConnectMyStudent service allows/denies the initialization of the RetrieveNotificationService via parent to student authentication codes.

- RetrieveNotifications service, dependent on the above, will then wait for the NSS to send notification data.

*Overall system graph*

**

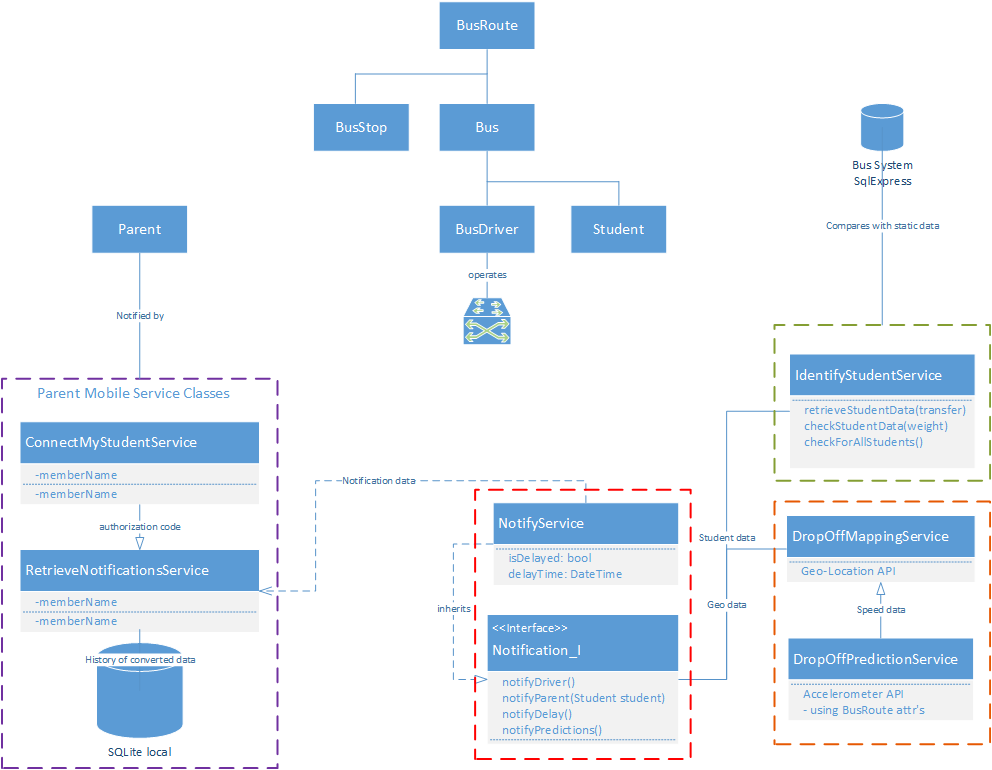
- The colored under-lined sections of this system design mark the application services that are part of each system marked.

- This is not be confused with a UML for classes or an ERD for the Databases, but an overview of the top-level systems and their relations.

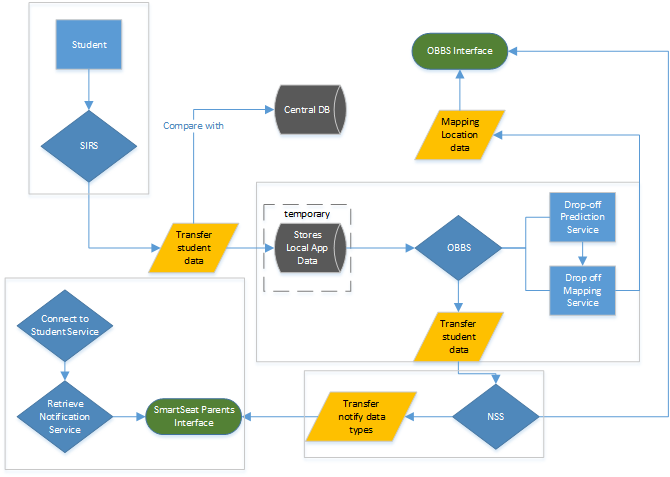
- To view a larger scale PDF of the above graph, refer to Section 8 for more details.

## Decomposition Description

The graph in section 3.1 thoroughly describes the system at large, so the next model will break down the modules into identified relations:



To describe the data work flow, we will show a *top-level data flow diagram* (DFD):

.

## Design Rationale

In this section, I will refer highly to some of the architectural dangers and risk factors that we will encounter when dealing with the above system described in 3.1.

- *Temporary data storage*: Data needs to be temporarily stored by the SIRS before it can be sent to the OBBS. This may not be a problem considering the abstract data types and amount of memory needed for identification.

- *Data Communication between long distance systems*: The NSS has the requirements of sending data over a long range to mobile devices for delays and student notifications. One of the biggest unknowns we have right now is this communication layer.

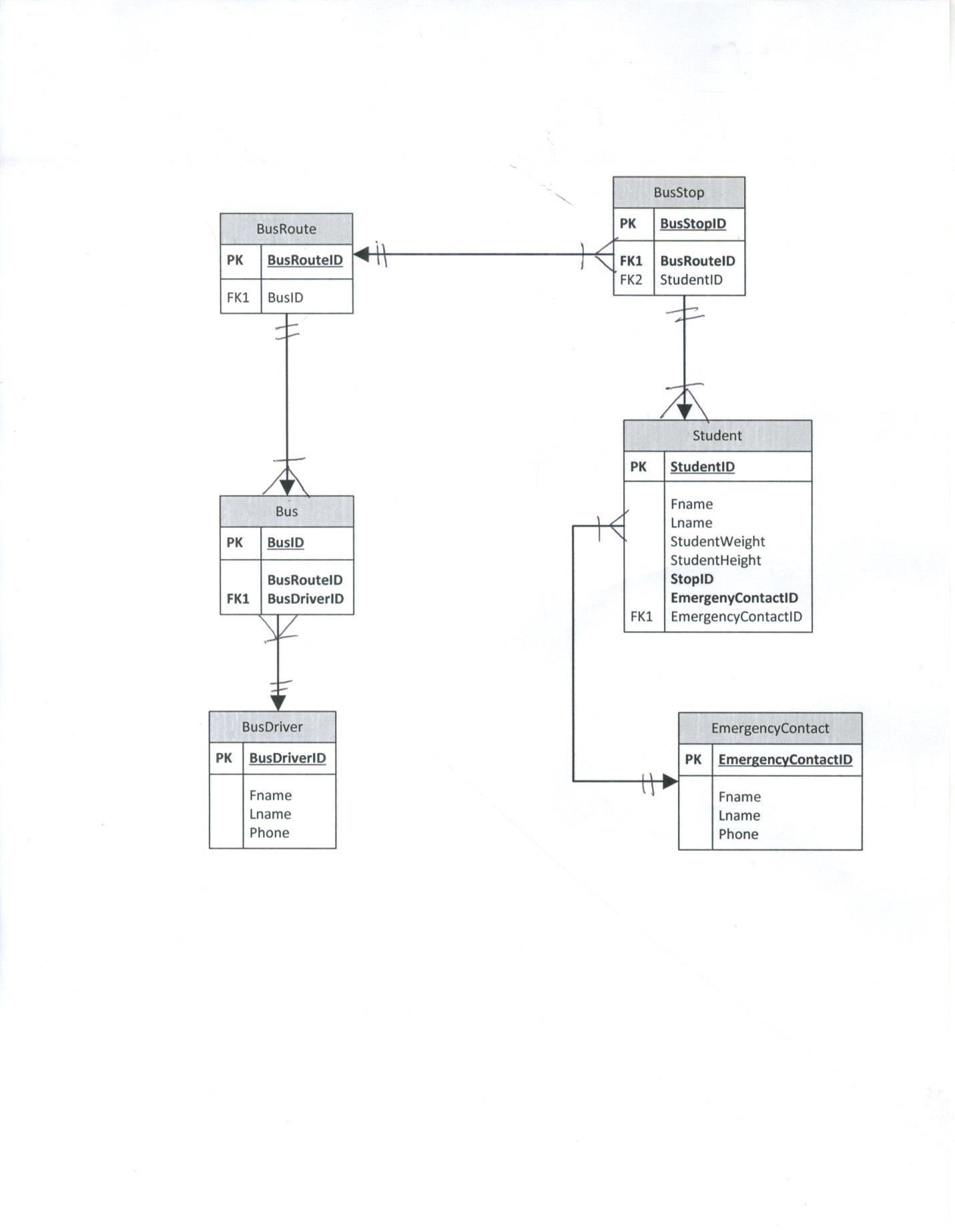
What we may want to try is creating a Web Service from the OBBS. How this will work is as so: The SmartSeat for Parents notification service will request different types of notification data through a SOAP format message to the NSS. The NSS will then respond to that given SOAP request by sending back the corresponding information. The request may carry a method call to request different notification data based on what is clicked on in the interface.

For instance, the typical way of refreshing a service on the mobile app would be to slide your finger down and release, showing an animated loading .gif and creating a request through the back-end.

### DATA DESIGN

* 1. **Data Description**

The network majors have provided the system transformed into data structures by implementing an ERD that includes entities, relations, cardinalities, and associative entities:



## Data Dictionary

**Student (**StudentID, Fname, Lname, StudentWeight, StudentHeight, *StopID, EmergencyContactID*)

**EmergencyContact** (EmergencyContactID, Fname, Lname, Phone)

**BusStop** (BusStopID, *BusRouteID*)

**BusRoute** (BusRouteID)

**Bus** (BusID, *BusRouteID, BusDriverID*)

**BusDriver** (BusDriverID, Fname, Lname, Phone)

### COMPONENT DESIGN

The objects being transmitted from system to system through the services require different data communications and certain data structures per service.

SIRS

*Communication transfer*: Without the hardware prototypes, data will need to be pushed into the system through a manual testing service that acts like the hardware detecting students. This test data will replace the communication transfer and will not be worried about until the electrical engineers provide a communication module with the hardware.

*Object structure:* The test data will include light identification information and will most likely not need security. The passing Student object will be processed and stored in an Array list of type String so that it can be compared easily with the central DB.

OBBS

*Object structure:* Geo-location and Accelerometer API’s will use their own embedded objects for data that will be transferred to the NSS. Any other OBBS data, such as student information compared by the SIRS will still be passed as a List because the length of data may change based on notification conditions.

NSS

*Communication transfer & Data structure*: Notification objects above will be securely embedded in SOAP envelope responses.

SmartSeat for Parents

*Communication transfer & Data structure*: The web service described above in 3.2 will be initialized and a SOAP request envelopes with corresponding notification methods and possible XML parameters will bring back the NSS response with Notification data.

### HUMAN INTERFACE DESIGN

* 1. **Overview of User Interface**

*OBBS interface overview*

- Before entering the simplistic view below, the bus driver will need to choose from a list of bus routes in a descriptive list. These Bus Routes will be sorted between pick-up and drop-off and will be data bound by the central DB based on the Bus ID.

- Once a bus route is chosen, the bus driver will have a 1 screen simplistic view with a 2 column layout. This is necessary so that the driver will have all the displayed information needed without being distracted.

- The larger left layout will include the current bus route through GPS. Once again, this service will be started by selecting a bus route. The GPS service will define the bus route by mapping out each bus stop as flags.

- The side-column right layout will provide time estimations between the current location and the next bus stop, and most importantly when the bus arrives at a bus stop, student information is displayed for the number of students required to pick-up or drop-off, depending on the route.

- At the bus stops, as we know that the SIRS will send student data to the OBBS, below the number of students will display an actual pick-up/drop-off live count. Once all of the students are on/off the bus at the given stop, the right column feed will give a final notification to the driver in the right layout regarding the student count.

*SmartSeat for Parents interface overview*

- This mobile app will include 2 parts, student connection interface and the live notification interface.

- The student connection interface will have 1 input textbox and validation messages below the textbox if the student authorization code is incorrect. A help link will also be displayed on the bottom right of the screen. Once clicked, a pop over will display helpful information regarding the student authorization code and where to sign-up.

- The live notification interface will provide the parent with a history of notification types, the current day’s activity, and the student’s basic information including name, bus route and bus stop. These 3 components will be displayed by touching the corresponding menu bar at the mid-top of the layout. Each component will be a sub-layout that alternates depending on the menu bar selected.

*Parents giving feedback*

- Feedback is crucial to the projects user base. Parents will be able to provide feedback after a set time of using the mobile application. After this set time, a pop-up will display over the normal interface asking them to provide feedback or remind them again later. Once feedback is submitted through an interactive touch based rating form, it is sent through an email handler to our development team.

## Screen Images

Please refer to the EarlyStageWireFrames\_SSBS.PDF document to view the wire-frames listed below:

SmartSeat for Student early-stage wire-frames

*ConnectMyStudent*

*RetrieveStudentNotifications*

On-Board Bus System wire-frames

*BusDriverLogIn*

*BusRouteListSelection*

BusRouteNavigator

## Screen Objects and Actions

The screen objects, or the interface classes, will get the stored requested notification data from the local SQLite DB on the mobile device. This local app data will be featured in different layouts through XML’s version of repeaters, by data binding the data in a List structure to a page control.

### REQUIREMENTS MATRIX

|  |  |  |  |
| --- | --- | --- | --- |
| Functional Requirement | Bus Database (SQL Express) | Parent Mobile Service Classes (SQL lite) | Hardware Components |
| Weight Resistance Detector and Recorder | YES | NO | YES |
| Real-time Notification System | NO | YES | YES |

### APPENDICES

Models

*Overall System Graph* | see attached document “OverallClass\_Design”

Wire-frames  
*Early Stage Wire-frames |* see attached PDF “EarlyStageWireFrames\_SSBS.pdf”