Bus Stop Buddy

by: Swole Team 6
Lamis Alqafshat, Brian Freeman, Luke Jeries, Dennis Kellogg,
Taylor Shephard, Dan Wiseman

SLACK CHANNEL: CSC-383.slack.com

Final Document

For: Dr. Mark Allison

On: 4/18/17

Available at: https://github.com/TacticalLuke43/BusStopBuddy

Table of Contents

ABSTRACT

Swole Team 6 proposes creating a bus tracking system called Bus Stop Buddy that includes many features. Some of those features will be showing the route of a specific bus (selected by the user), the predicted arrival of the bus and if it is late, and enabling the students to log into and display this information in real time. The system will also maintain information about the parents and students for contact information in case the need arises.

The application will also have the ability to track multiple busses at once. Any user can get the notifications of the buses whereabouts and if they are on schedule. The application will also allow the user to input whether a pickup is necessary or not and if the student is authorized to ride a specific bus.

1. Introduction

In this chapter, we will introduce the motivation for building the system

1.1. Purpose of System

The purpose of the system is to give students and parents a way to track their associated school bus and know what time it should arrive at each stop.

1.2. Scope of System

The system will allow parents to track the bus and get real time notification of when their children are being picked up from school and dropped off at home. The system will also allow the parent to view the bus driver's information and vice versa so if problems arise, they can contact one another.

1.3. Development Methodology

Since agile is all about working with the business people to get exactly what they want throughout all phases of the project we have a few ways to do so. Currently one of our members has a family member that works as a bus driver and he is acting as our business person. One of our developers is working with him to figure out exactly what kinds of features he would want in a bus tracking software system.

Since then our developer has gotten a list of requirements from him along with some extras that our other developers came up with. At our next meeting, we are going to talk about all these requirements that are necessary to make this project tick along with some extras that are going to make it above and beyond expectation.

1.4. Definitions, Acronyms, and Abbreviations

Actors: External entities that interact with the system.

Agile: A method of development.

School: Educational institution for children.

ETA: Estimated Time of Arrival.

GPS: Global Positioning System.

1.5. Overview of Document

The grand scheme of this is to deliver an application that parents can install on their phones so they can pinpoint exactly when their children will be home.

2. Current System

Not applicable

3. Project Plan

3.1. Project Organization

Phase 1 (1/16/2017-2/3/2017)

Brian Freeman	Editor
Dennis Kellogg	System Architect
Luke Jeries	Leader
Taylor Shephard, Lamis Alqafshat	Minute Keeper
Dan Wiseman	Secretary/Diary Keeper

Phase 2 (2/3/2017-3/2/2017)

Taylor Shephard	Leader
Brian Freeman, Dan Wiseman	Validater/Architect
Lamis Alqafshat	Minute Keeper
Luke Jeries	Diary Keeper/Tester
Dennis Kellogg	Editor

3.2. Software and Hardware Requirements

Hardware: Smartphone and the ability to download an application

Software: iOS, Android OS

3.3. Work Breakdown

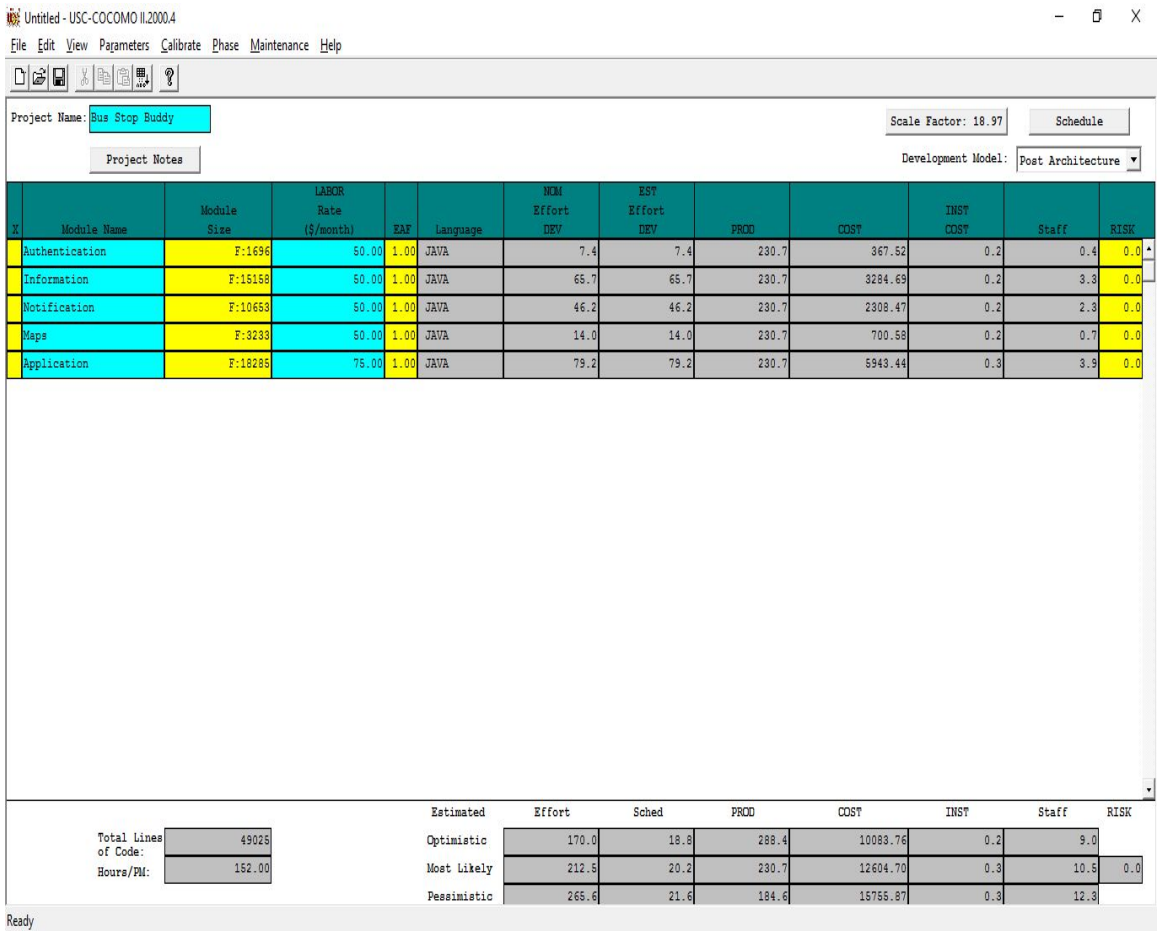
Task #	Task	Description	Duration	Dependencies
1	Team Introduction	Get to know team members, consider strengths and weaknesses, brainstorm project topic	7 days	
2	Project Idea Development	Cultivate existing project idea, consider all potential uses, consider potential challenges	7 days	1 (M1)
3	Use Case Creation	Refine proposed features into clear use cases	3 days	
4	Use Case Completion	Submit Use Case diagram for review, consider future Contextual Use Case details	4 days	3 (D1)
5	Persona Research	Consider relevant users, begin consideration of Personas	3 days	
6	Documentation - SFD	SFD Stages 1-4.3	3 days	2, 4 (M2) (D2)
7	Software Architecture	Divide Project into subsystems, identify objects, finalize member roles for development	3 days	6
8	Object Design	Develop models into workable code/design	14 days	7
9	Implementation Phase 1	Begin coding and auxiliary software implementation	20 days	8
10	Review of Implementation 1	Members all reconvene for group reflection	4 days	9 (M3)
11	Implementation Phase 2 - Final	Finalize basic development	14 days	
12	"Hands on" Testing	Rigorous testing while working with key developer(known problems)	7 days	9, 11 (M4) (D3)
13	"Hands off" Testing	Testing with no insight from key developer(finding unknown problems)	7 days	
14	System Familiarity Development	Bring all members up to speed with status/details of sub systems	2 days	9, 11
15	Creation of FD	Complete FD, begin Power Point	20 days	(M5) (D4)
16	Final Presentation	Present and submit FD	1 day	15

M – Milestone D – Deliverable

3.4 Risk Management Table

Risk	Category	Probability	Impact (1 Lowest to 5 Highest)	RMMM	Risk Exposure
Problem Implementing Authentication Module	Schedule Risk	4.00%	2	Monitoring - Help catch problems early	0.04
Problem designing Mapping Module	Performance Risk	3.00%	5	Monitoring - Help catch problems early	0.15
Designed a product no one wants	Performance Risk	5.00%	5	Management - Talk with the product owners, consider redesigning product	0.25
Hardware Failure	Cost Risk/Schedule Risk	10.00%	3	Mitigation - Backup computers	0.3
Member of the team falls ill/drops out and is unable to work	Schedule Risk	10.00%	3	Management - Be able to recruit new members/have backups/delegate tasks effectively	0.3
Problem Implementing Notification Module	Schedule Risk	12.00%	3	Monitoring - Help catch problems early	0.36

3.5 COCOMO COST ESTIMATE



Untitled - USC-COCOMO II.2000.4

File Edit View Parameters Calibrate Phase Maintenance Help

Project Name: Bus Stop Buddy

Scale Factor: 18.97 | Schedule

Project Notes

Development Model: Post Architecture

X	Module Name	Module Size	LABOR Rate ($/month)	EAF	Language	NOM Effort DEV	EST Effort DEV	PROD	COST	INST COST	Staff	RISK
	Authentication	F:1696	50.00	1.00	JAVA	7.4	7.4	230.7	367.52	0.2	0.4	0.0
	Information	F:15158	50.00	1.00	JAVA	65.7	65.7	230.7	3284.69	0.2	3.3	0.0
	Notification	F:10653	50.00	1.00	JAVA	46.2	46.2	230.7	2308.47	0.2	2.3	0.0
	Maps	F:3233	50.00	1.00	JAVA	14.0	14.0	230.7	700.58	0.2	0.7	0.0
	Application	F:18285	75.00	1.00	JAVA	79.2	79.2	230.7	5943.44	0.3	3.9	0.0

Total Lines of Code: 49025

Hours/PM: 152.00

Estimated	Effort	Sched	PROD	COST	INST	Staff	RISK
Optimistic	170.0	18.8	288.4	10083.76	0.2	9.0	
Most Likely	212.5	20.2	230.7	12604.70	0.3	10.5	0.0
Pessimistic	265.6	21.6	184.6	15755.87	0.3	12.3	

Ready

4. Requirements of System

The system we are proposing

4.1. Functional and Nonfunctional Requirements

Functional: App, GPS tracking, ETA Estimates, Contact Information, Map, All these features for all busses.
Nonfunctional: Show route on map, program to display bus info at school, Features to get ETA, Alarm Notification for student pickup and drop off, student to bus correlation, Ability to select other busses, RFID

4.2. IDENTIFY Personas

Parents: Busy people with tight schedules that need to know when their child or children will be leaving school and arriving home.

Drivers: Pick the kids up from school and drop them off at home. Need to be able to let parents know when the kids are being picked up and being dropped off. Along with alerting parents of emergencies

Students: The children being picked up and dropped off. They need to be able to check into the bus and Check out of the bus.

School Faculty: Work at the school and look out for the general well being of the children. Need to know when the students get on the bus and when the students are dropped off the bus. Also need to be able to track the bus to see where it's at and if has ran into any issues.

4.3. Use Case Diagram

The next figure depicts the interaction between the actors and the previously described use cases.

A description for each actor follows.

Parent: The main user of the application, guardian of the student.

Student: Rides the bus, could be in elementary/middle/high school

Driver: Operates the bus

School: Owner of the bus, manages the bus

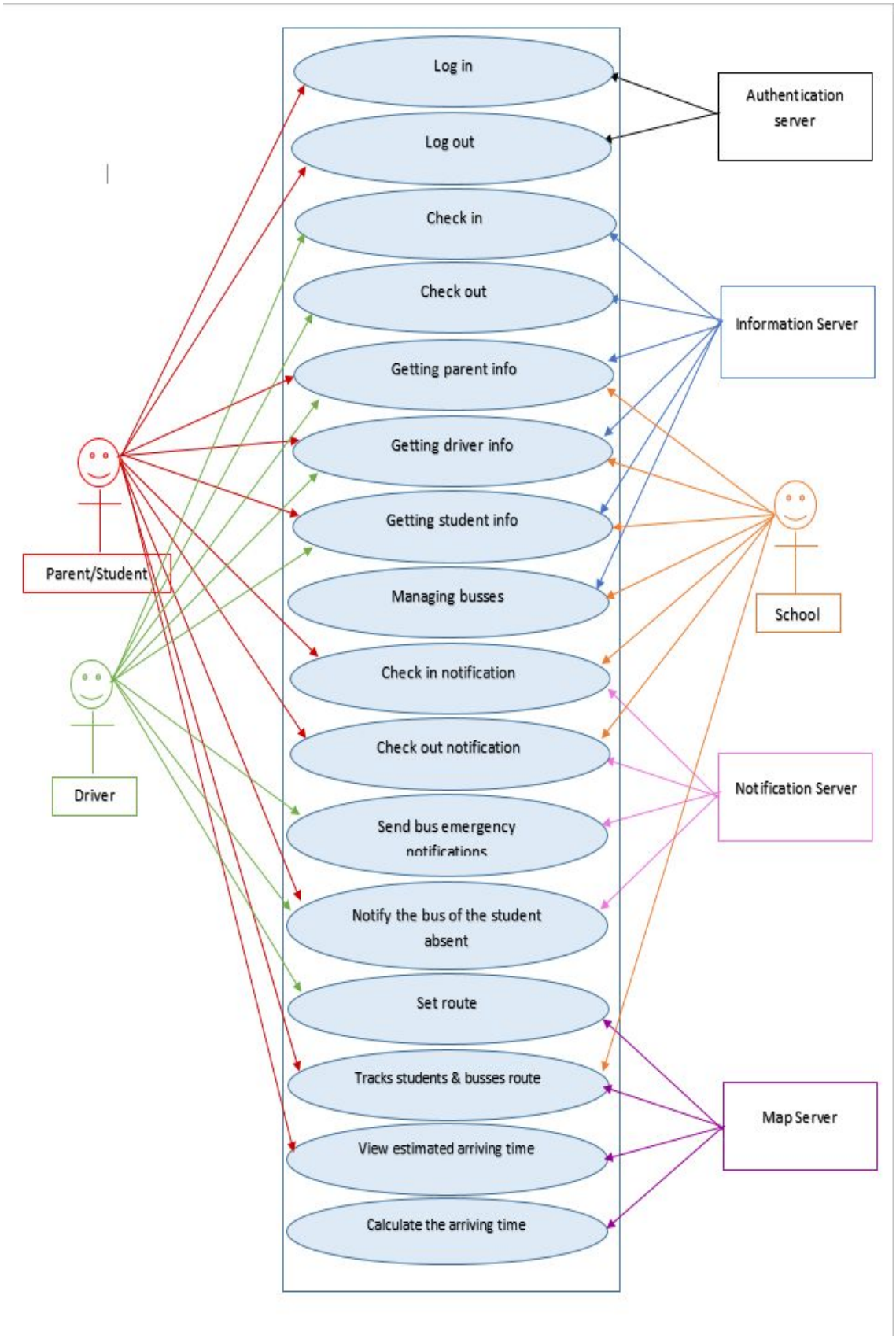


Figure 1: Use Case Diagram

4.4. Requirements Analysis

After considering our own goals for the project as well as the concerns of some of our stakeholders(both bus drivers and parents), we believe that our use case and the requirements it reflects wholly encompass the needs of anyone hoping to use the app. Keeping the user experience at the forefront of our design process, we can move forward towards fleshing out our system details and implementing all of the planned helpful functions of Bus Stop Buddy.

5. Software Architecture

.

5.1. Overview

Upon considering the inherent nature of an app like ours, following a Client-Server architectural pattern made the most sense for this project. Every user's device will connect with our FireBase cloud server in order to ensure everyone views the same data.

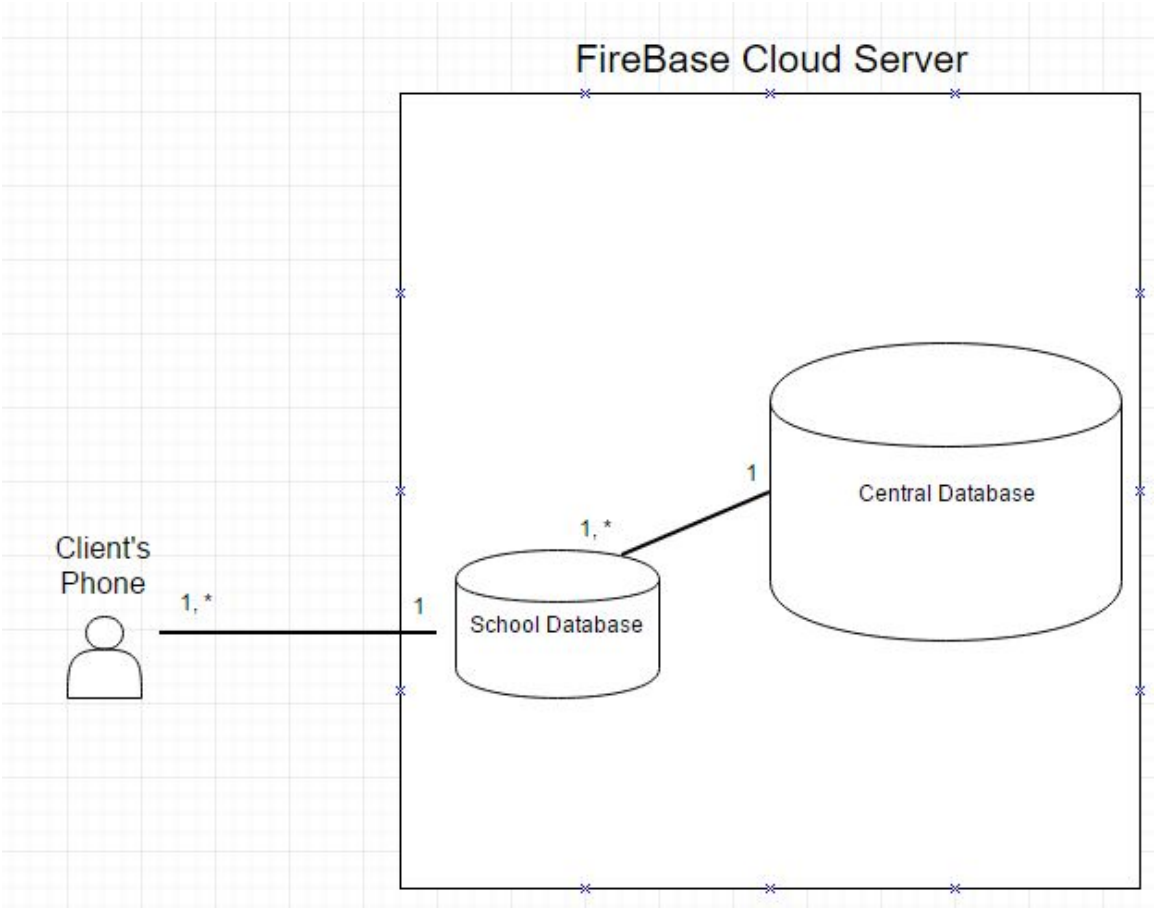
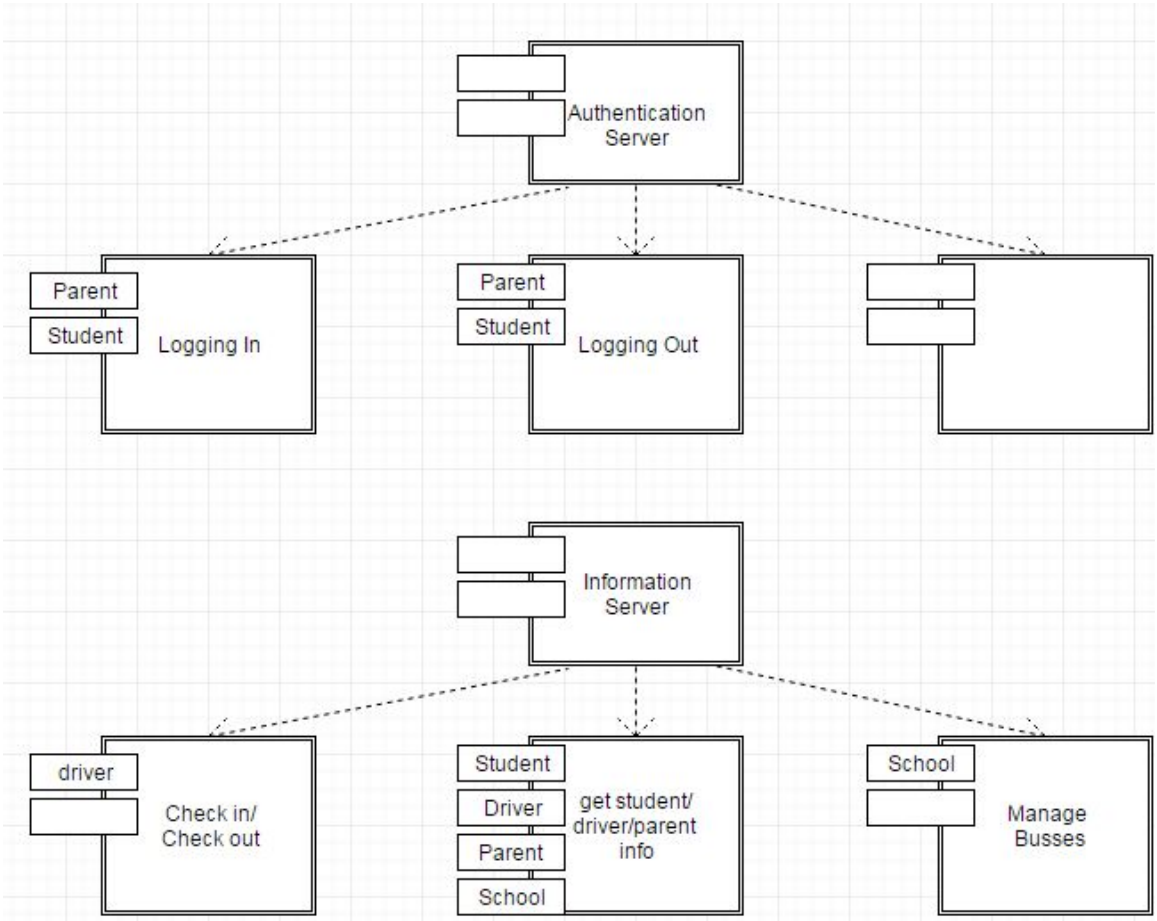
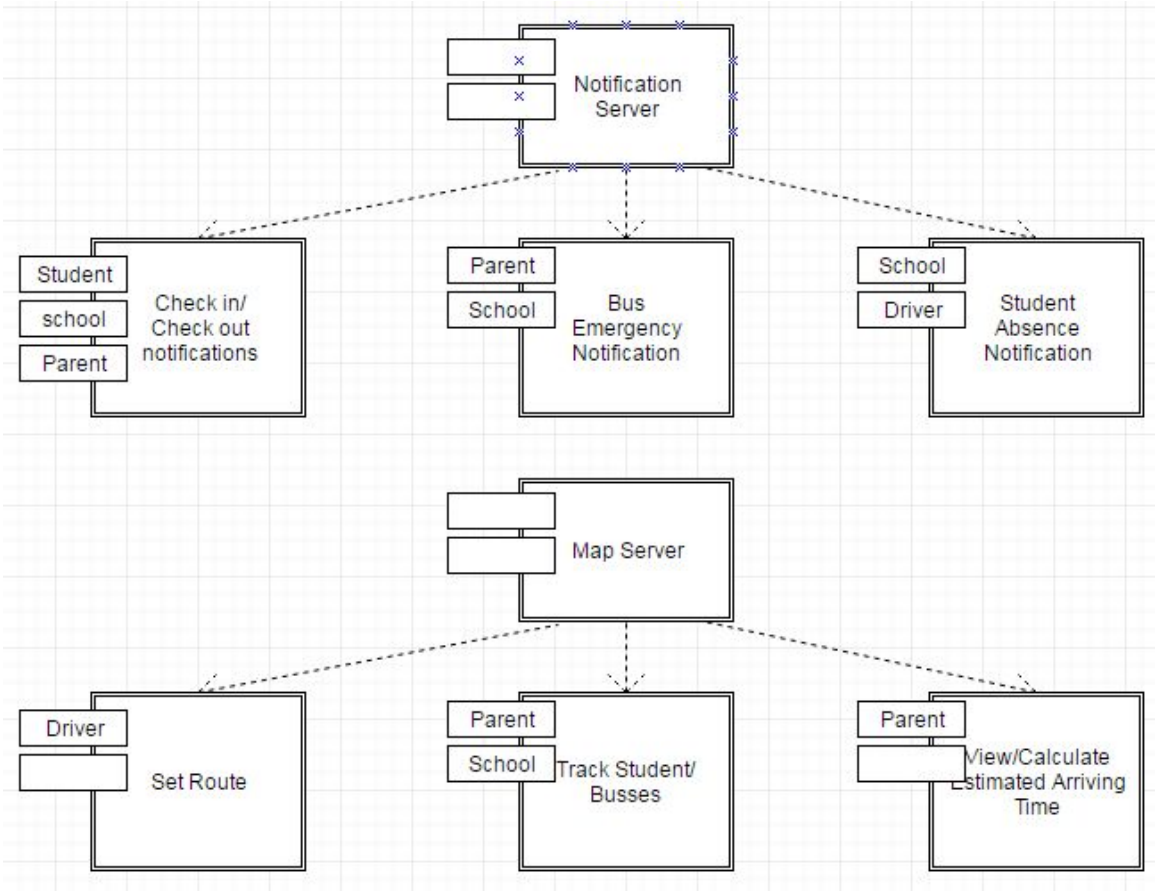


Figure 1: Basic Client-Server Architecture

5.2. Subsystem Decomposition.





5.3. Persistent Data Management

Firebase(Cloud) Storage

- Driver Information
- Bus Routes
- Student Information
- Parent Information
- School Information

Device Storage

- Personal Settings
- Remember Username
- Remember Password

6. Object Design

The architecture we chose was client-server architecture. The main reasons we chose this architecture is because the way the users interact with each other.

6.1. Overview

This application was designed with the evolutionary model. The main reason we chose this model is because we knew that there were going to be changes made in the future depending on what works and what doesn't.

6.2. Object Interaction

The way the object interact in our application is that the user's interact with the server. The driver object interacts with the server by telling the server his/her information along with the notifications that a specified child has gotten on/off the bus or if a incident has occurred. The parent object interacts with the system by retrieving data from the driver about the whereabouts of their child and what the estimated time of arrival of their child is. Also the parent interacts with the server by uploading their contact information. The student object interacts with the driver object to let the driver know that they are on the bus.

6.3. Detailed Class Design

Driver

- Collection: Persons
- Count: int (drivers = # of busses)
- File = Number of driver info files

+ Driver()
+ getNumberOfDrivers(): int
+ addPerson(String firstName, String lastName, String address, String city, String state, String zip, String phone)
+ getFullNameofPerson(int index): String
+ getOtherPersonInformation(int index): String[]
+ updatePerson(int index, String address, String city, String state, String zip, String phone)
+ removePerson(int Index)
+ sortByName()
+ sortByZip()
+ printAll()
+ getFile(): File
+ getTitle(): String
+ setFile(File file)
+ getChangedSinceLastSave(): boolean
+ setChangedSinceLastSave(boolean changedSinceLastSave)

Parent

- Collection: Persons
- Count: int (parent = #of students)
- File = Number of parent info files

+ Parent()
+ getNumberOfParents(): int
+ addPerson(String firstName, String lastName, String address, String city, String state, String zip, String phone)
+ getFullNameofPerson(int index): String
+ getOtherPersonInformation(int index): String[]
+ updatePerson(int index, String address, String city, String state, String zip, String phone)
+ removePerson(int Index)
+ sortByName()
+ sortByZip()
+ printAll()
+ getFile(): File
+ getTitle(): String
+ setFile(File file)
+ getChangedSinceLastSave(): boolean
+ setChangedSinceLastSave(boolean changedSinceLastSave)

Student

- Collection: Persons
- Count: int (Student = #ofStudents)
- File = Number of Student info files

+ Student()
+ getNumberOfStudents(): int
+ addPerson(String firstName, String lastName, String address, String city, String state, String zip, String phone)
+ getFullNameofPerson(int index): String
+ getOtherPersonInformation(int index): String[]
+ updatePerson(int index, String address, String city, String state, String zip, String phone)
+ removePerson(int Index)
+ sortByName()
+ sortByZip()
+ printAll()
+ getFile(): File

+ getTitle(): String
+ setFile(File file)
+ getChangedSinceLastSave(): boolean
+ setChangedSinceLastSave(boolean changedSinceLastSave)

---

7. Testing Process

7.1. User Experience Tests

Testing the system largely fell to us, the developers. In order to ensure that we had a good and functional product, we each used the apps on our own phones and simulated specific tasks in order to ensure that all features were functional. These primarily consisted of viewing the various information contained on the menus(View Routes, View Student Check Ins, etc.), and creating new entries for the various users(Edit Student/Parent/Driver info). From these tasks, we made slight adjustments in the design of our user interface in order to make it consistent across all screens within the app, as well as make it more intuitive and easy to use.

7.2. Systems Tests

The system tests proved to be invaluable, as we did very little in the way of integration testing. Many bugs and inconsistencies had arisen during the development of our different modules that were not caught due to our lack of proper use of version control. Thankfully, the systems test we performed caught a lot of the abnormal behavior, and we were able to correct it.

7.3 Subsystems Tests

Due to the more simplistic nature of our in class demo compared to the scope and functionality of our entire planned project, our subsystems tests were mostly done to verify basic functionality. Making sure that all buttons behave as expected proved to be the most time consuming part of this process. Copy and pasted code proved to be a major problem, as this lack of care led to many difficult to track down bugs in the simple navigation of our app. In the end though, with each member contributing to the testing of their modules during their respective development phases, each subsystem behaved as expected. Presented are several test cases which unveiled major issues while creating the system.

Test Case ID	1			**Created Date**	3/17/2017
Priority	High			**Created By**	Luke Jeries
Module Name	Application				
Title	View Parents Information				
PreConditions	User is logged in as admin/driver				
Step#	**Test Step**		**Test Data**	**Expected result**	
1	Select the "View Parent Info" button form menu		View Driver Info screen is launched, any interaction causes crash	View Parent Info activity is launched	
2	Select student from drop down list		Crash	valid students are presented in drop down l	
3	Select Update Button		Crash	Form is updated with currently selected student's parent info	
4	Select Log Out Button		Log In screen is presented	Log in screen is presented	
Conclusions:	Class association or intents have a coding error				
Fixes:	Review XML for View Parent Info and View Driver Info Activity				
	Review Java file for View Parent Info and View Driver Info Activity				
	Review Intents for all activity transfers				

Figure 1: Application Module Test - Class association error

Test Case ID	3			**Created Date**	3/19/2017
Priority	Low			**Created By**	Luke Jeries
Module Name	Maps				
Title	Instanced View Buses Test				
PreConditions	User is logged in as student				
Step#	**Test Step**		**Test Data**	**Expected result**	
1	Select the "View Buses" button from menu		All routes are loaded	Load only those routes student is registered for	
2	Select Log Out Button		Log In screen is presented	Log in screen is presented	
Conclusions:	Array of routes not filtered correctly				
Fixes:	Review student route registration				
	Review array filter in View Buses activity				

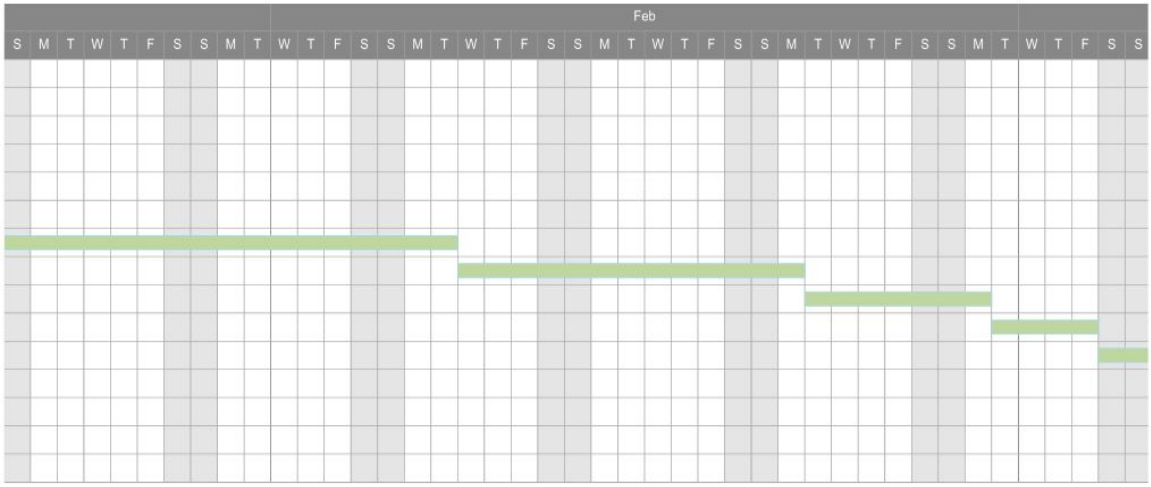
Figure 2: Maps Module Test - Instanced user maps error

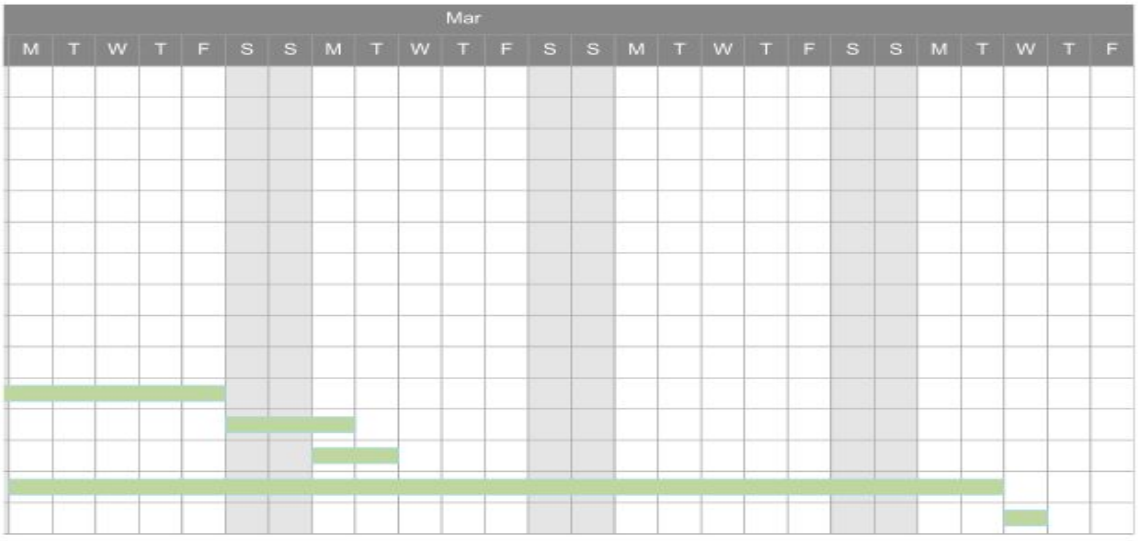
8. GLOSSARY

9. APPENDIX

9.1. APPENDIX A – GANTT CHART

	Task Name	Duration	Start	Finish
1	Team Introduction	1d	01/04/17	01/04/17
2	Project Idea Development	5d	01/05/17	01/11/17
3	Persona Research	4d	01/10/17	01/13/17
4	Use Case Creation	2d	01/14/17	01/16/17
5	Documentation -SFD	3d	01/15/17	01/17/17
6	Software Architecture	3d	01/17/17	01/19/17
7	Object Design	14d	01/19/17	02/07/17
8	Implementation - Phase 1	9d	02/08/17	02/20/17
9	Implementation Phase 2 - Final	5d	02/21/17	02/27/17
10	Review of Implementation 1	4d	02/28/17	03/03/17
11	"Hands on" Testing	6d	03/04/17	03/10/17
12	"Hands off" Testing	2d	03/11/17	03/13/17
13	System Familiarity Development	2d	03/13/17	03/14/17
14	Creation of FD	17d	03/06/17	03/28/17
15	Final Presentation	1d	03/29/17	03/29/17





9.2. Appendix B – Use Cases

Use Cases

Use Case ID: BSB01 - Manage Routes

Scenario:

Actor: Administrator

Pre-conditions:

1. Administrator is logged in to the system.

Description:

1. Use case begins when the Administrator selects the “Administrator Features” option from the app menu.
2. The system presents the user with the Administrator Features menu.
3. Administrator selects the Manage Routes option from the menu.
4. System retrieves route list for Administrator’s registered school.
5. System presents list of routes to user.
6. Administrator selects the route they wish to modify.
7. System pulls up detail form for selected route.
8. Administrator changes desired details of route and hits “submit changes”
9. Use Case ends when system updates form and returns Administrator to Administrator Features menu.

Post Conditions:

1. School’s route list has been permanently updated.

Alternate Course of Action:

1. In Step D6, Administrator may select to add a new route rather than modify existing.

Related Use Cases:

Log in

Decision Support:

Frequency: After initial set up, route adjustments should be limited. About 3 per month in regular use.

Criticality: High. Creation of routes is the core feature of setting up the system.

Risk: Medium. Use case employs modification of server info.

Constraints:

Route list should be available to modify >90% of each 24 hour day.

Use Case ID: BSB02 – Send Bus Emergency Notification

Scenario:

Actor: Driver

Pre-conditions:

1. User is logged in to the system.

Description:

1. Use case begins when a registered Driver selects “Send Emergency Update” from app menu.
2. System presents the Update form to user.
3. Users enters text detailing desired broadcast message.
4. System sends notification to all users registered to that Driver’s route.
5. Use case ends when system returns user to app menu.

Exceptions:

1. User is not a registered Driver.
2. Driver has no associated routes.

Related Use Cases:

Log in

Decision Support:

Frequency: Low. Updates should be few and far between.

Criticality: Low. Emergency Updates are a non-essential feature, merely quality of life.

Risk: Medium. Use case employs modification of server info and push notifications.

Constraints:

Emergency Updates are pushed to users within 1 minute.

Use Case ID: BSB01 - Manage Routes

Scenario:

Actor: Administrator

Pre-conditions:

1. Administrator is logged in to the system.

Description:

1. Use case begins when the Administrator selects the “Administrator Features” option from the app menu.
2. The system presents the user with the Administrator Features menu.
3. Administrator selects the Manage Routes option from the menu.
4. System retrieves route list for Administrator’s registered school.
5. System presents list of routes to user.
6. Administrator selects the route they wish to modify.
7. System pulls up detail form for selected route.
8. Administrator changes desired details of route and hits “submit changes”
9. Use Case ends when system updates form and returns Administrator to Administrator Features menu.

Post Conditions:

1. School’s route list has been permanently updated.

Alternate Course of Action:

1. In Step D6, Administrator may select to add a new route rather than modify existing.

Related Use Cases:

Log in

Decision Support:

Frequency: After initial set up, route adjustments should be limited. About 3 per month in regular use.

Criticality: High. Creation of routes is the core feature of setting up the system.

Risk: Medium. Use case employs modification of server info.

Constraints:

Route list should be available to modify >90% of each 24 hour day.

Use Case ID: BSB02 – Send Bus Emergency Notification

Scenario:

Actor: Driver

Pre-conditions:

1. User is logged in to the system.

Description:

1. Use case begins when a registered Driver selects “Send Emergency Update” from app menu.
2. System presents the Update form to user.
3. Users enters text detailing desired broadcast message.
4. System sends notification to all users registered to that Driver’s route.
5. Use case ends when system returns user to app menu.

Exceptions:

1. User is not a registered Driver.
2. Driver has no associated routes.

Related Use Cases:

Log in

Decision Support:

Frequency: Low. Updates should be few and far between.

Criticality: Low. Emergency Updates are a non-essential feature, merely quality of life.

Risk: Medium. Use case employs modification of server info and push notifications.

Constraints:

Emergency Updates are pushed to users within 1 minute.

Use Case ID: **Calculate Arriving Time**

Scenario:

Actor: Parent/Student user.

Pre-conditions:

1. User has successfully logged onto the system.
2. Web page has been activated.

Description:

1. Use case begins when parent/student user clicks on the Calculate Arrival Time Button in the selection menu
2. The system shall validate the information
3. User shall select which bus they desire to view (request)
4. The user shall then send the request by selecting the **send** button.
5. The system shall then notify the parent/student user if the request was submitted correctly.
6. When the request is received, the system shall generate the ETA
7. System shall display the map and time of arrival
8. Use case ends when the user closes

Post-conditions:

1. The estimated time of arrival is estimated
2. Estimated time of arrival is updated and stored in the system

Alternative Courses of Action:

1. In step D.4 (step 4 of Description section) the user has the option to cancel the request.
2. In step D.6 if any of the required fields are incorrect the system shall request the user to make a correction in the appropriate field.

Exceptions:

1. There are no busses running at this time.

Related Uses Case:

None.

Decision Support:

Frequency: On average 10 requests are made daily by parent/student user.

Criticality: High. Main objective of the program

Risk: High. Implementing this use case employs drivers login, gps tracking systems, refresh rates

Constraints:

Non-functional requirements

Modification History:

Owner: Swole Team 6

Initiation date: 02/15/2017

Date last modified: 02/15/2017

Use Case ID: **Calculate Arriving Time**

Scenario:

Actor: Parent/Student user.

Pre-conditions:

1. User has successfully logged onto the system.
2. Web page has been activated.

Description:

1. Use case begins when parent/student user clicks on the Calculate Arrival Time Button in the selection menu
2. The system shall validate the information
3. User shall select which bus they desire to view (request)
4. The user shall then send the request by selecting the **send** button.
5. The system shall then notify the parent/student user if the request was submitted correctly.
6. When the request is received, the system shall generate the ETA
7. System shall display the map and time of arrival
8. Use case ends when the user closes

Post-conditions:

1. The estimated time of arrival is estimated
2. Estimated time of arrival is updated and stored in the system

Alternative Courses of Action:

1. In step D.4 (step 4 of Description section) the user has the option to cancel the request.
2. In step D.6 if any of the required fields are incorrect the system shall request the user to make a correction in the appropriate field.

Exceptions:

1. There are no busses running at this time.

Related Uses Case:

None.

Decision Support:

Frequency: On average 10 requests are made daily by parent/student user.

Criticality: High. Main objective of the program

Risk: High. Implementing this use case employs drivers login, gps tracking systems, refresh rates

Constraints:

Non-functional requirements

Modification History:

Owner: Swole Team 6

Initiation date: 02/15/2017

Date last modified: 02/15/2017

Use Case ID: Get driver information

Scenario:

Actor: driver, school, info server

Pre-conditions:

1. driver register to the system
2. driver Enter his/her information
3. driver Submit request
4. School sends information to the parents and students

Description:

1. Use case begins driver download the application
2. When the driver open the application the start screen should show log in and register options
3. driver will click in register and the system will take him to the register page
4. The system will ask the user to enter the ID number.

5. Driver enter ID number
6. System will display the information page.
7. Driver will enter his/her information: name, contact
8. Driver click in Submit bottom to send the request.
9. School receive the driver request and save the new driver information in the info server
10. Use case ends send information to the students or parents in the area the driver will go to

Post-conditions:

1.

Alternative Courses of Action:

1. In step D.8 (step 4 of Description section) the user has the option to cancel the request.
2. In step D.8 if any of the required fields are blank the system shall request the user to make an entry in the appropriate field.
3. In step D.4 the system will choose the category (student, parent, or driver) from the ID number and display the correct application.

Exceptions:

Related Uses Case:

None

9.3. Appendix C – User Interface Designs

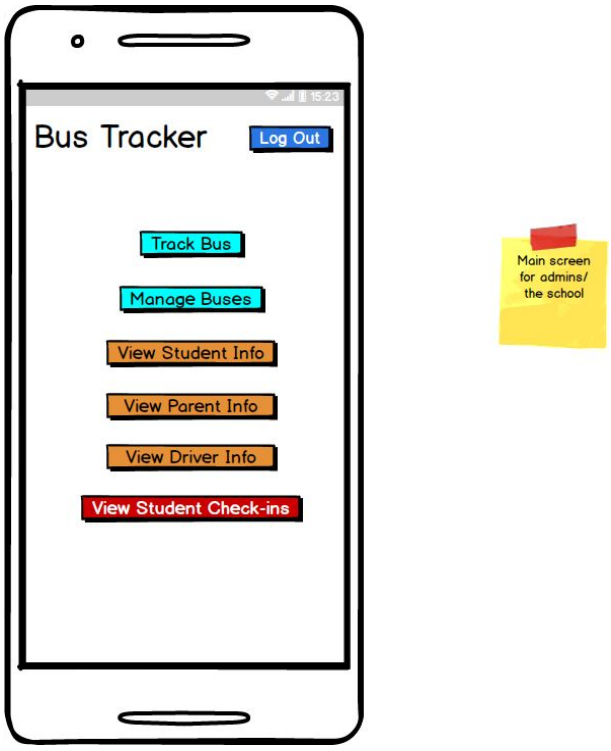


Figure 1: Admin Main Menu

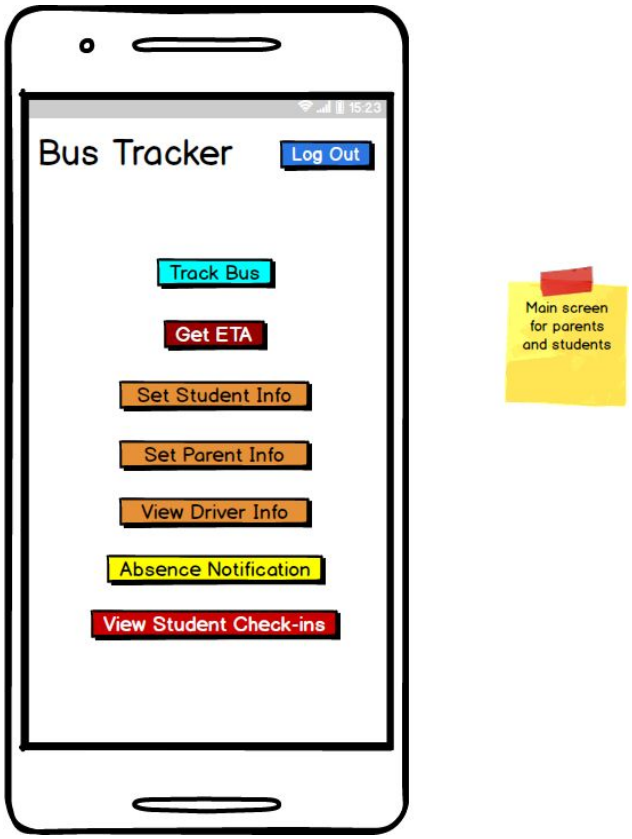


Figure 2: Parents Main Menu

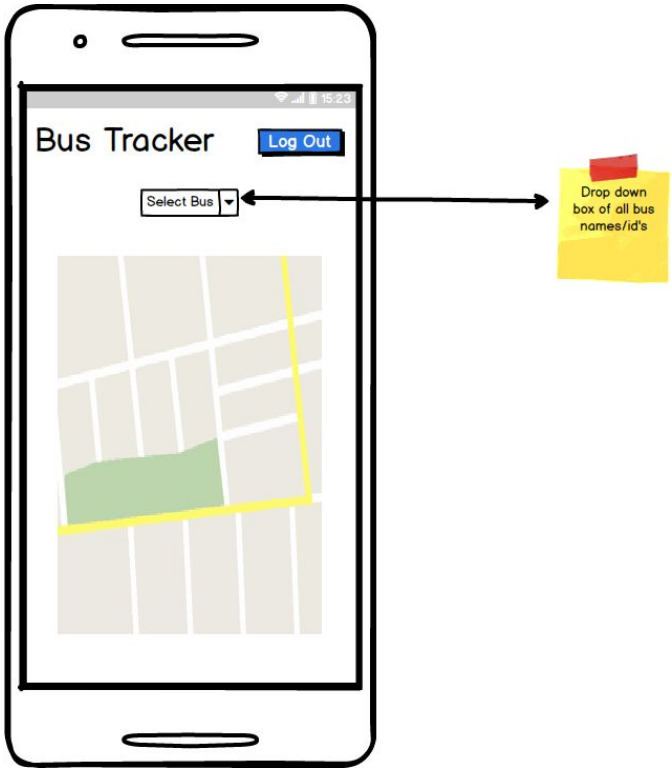


Figure 3: Bus Tracker Main Screen

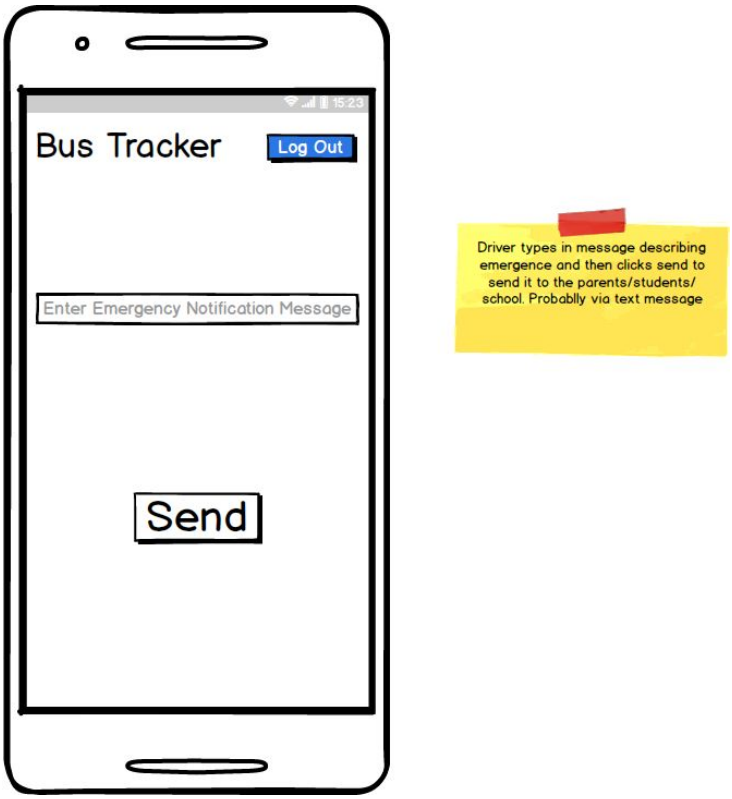


Figure 4: Emergency Notifications

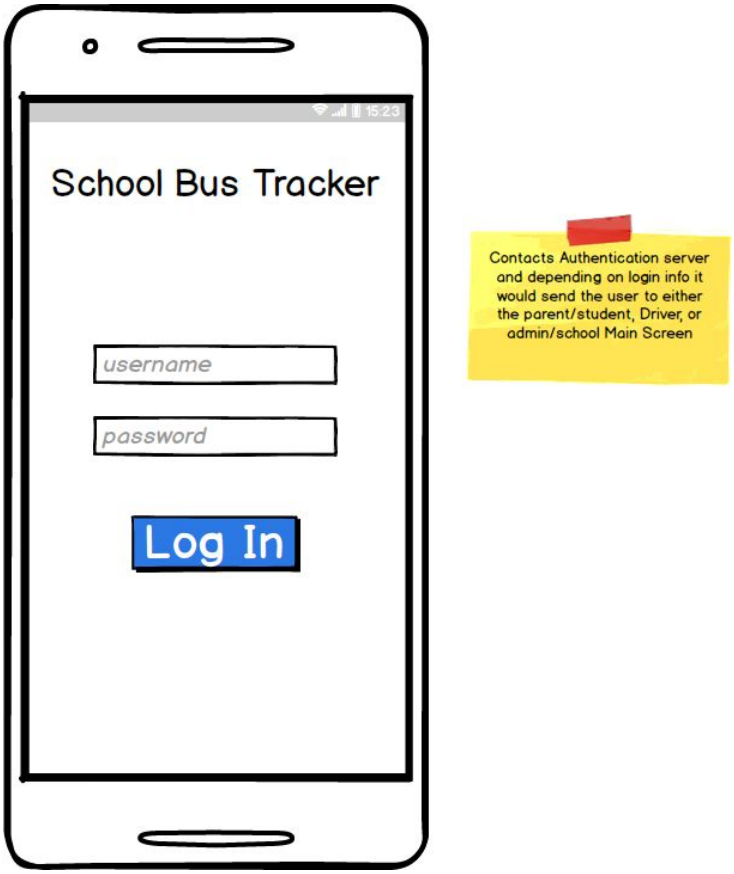


Figure 5: Log In Screen

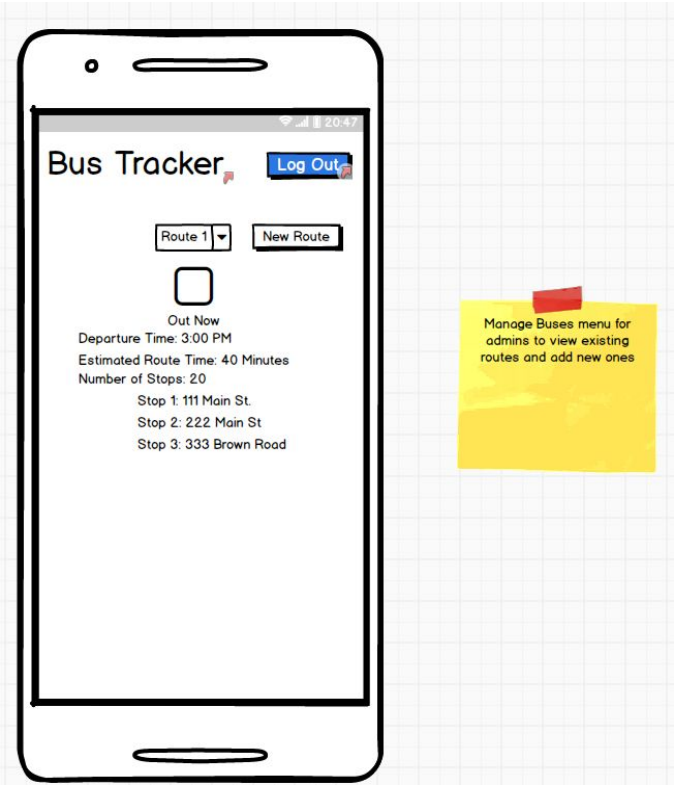


Figure 6: Manage Buses Screen

Figure 7: Student Check In Screen

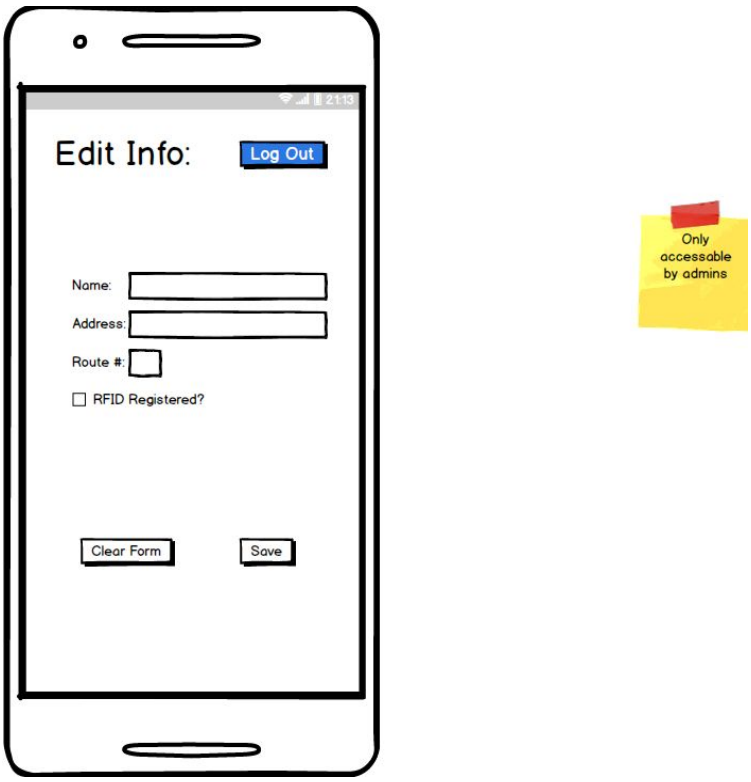


Figure 8: Edit Info Screen

Figure 9: View Info Screen

9.5. Appendix D – Class Interfaces for Implemented Subsystems

Bus Stop Buddy

SWOLE TEAM SIX

PROJECT BY:

BRIAN FREEMAN
DAN WISEMAN
DENNIS KELLOGG
LAMIS ALQAFSHAT
LUKE JERIES
TAYLOR SHEPARD

What is Bus Stop Buddy?

- Bus tracking application via GPS

Features of Bus Stop Buddy

- Alert System
- Notifies user of arrival times for pickup and drop-off
- Creation and editing accounts
- Allow the driver to notify parents/school of problems
- Bus selection for tracking
- Viewing information

Why Bus Stop Buddy?

- This application appeals to busy parents
- Allows for precision timing
- Safety
- Updates on problems

Requirements

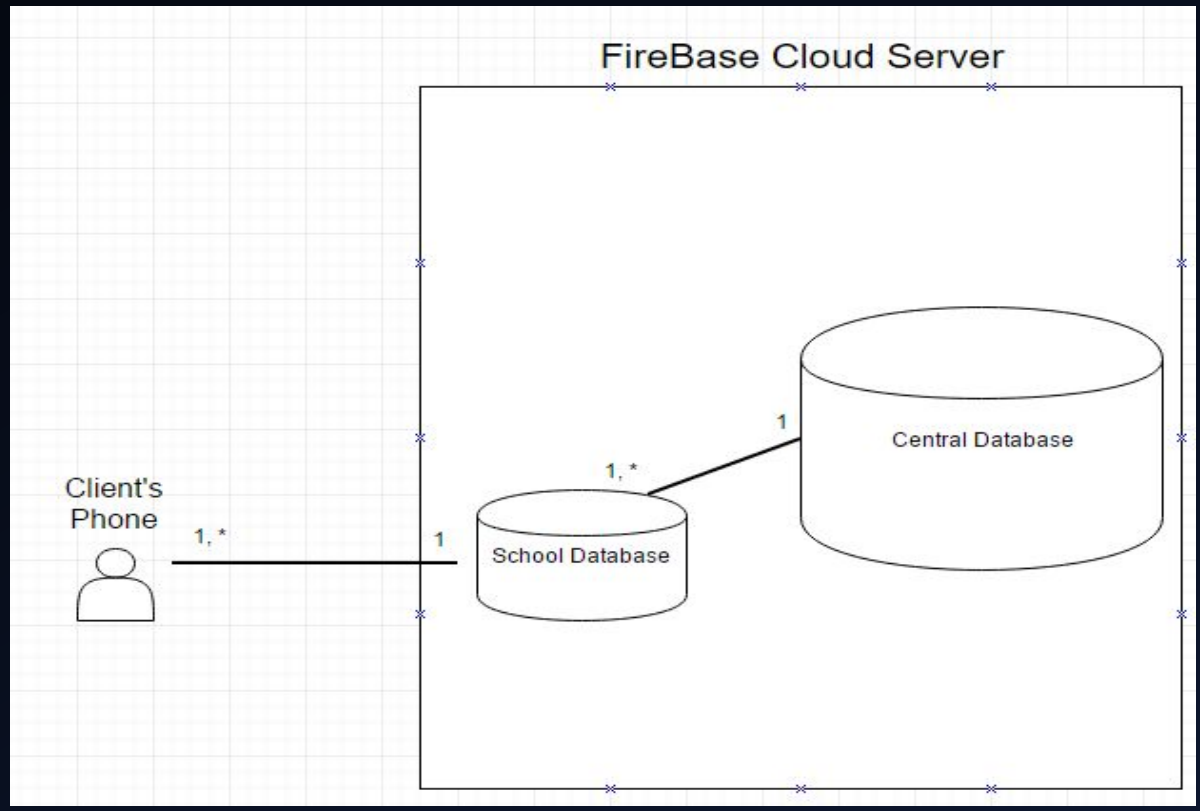
- Application
- GPS Tracking
- Estimated time of arrival/pickup
- Contact Information
- Maps

Personas

- Parents
- Drivers
- Students

Architecture

- Client server



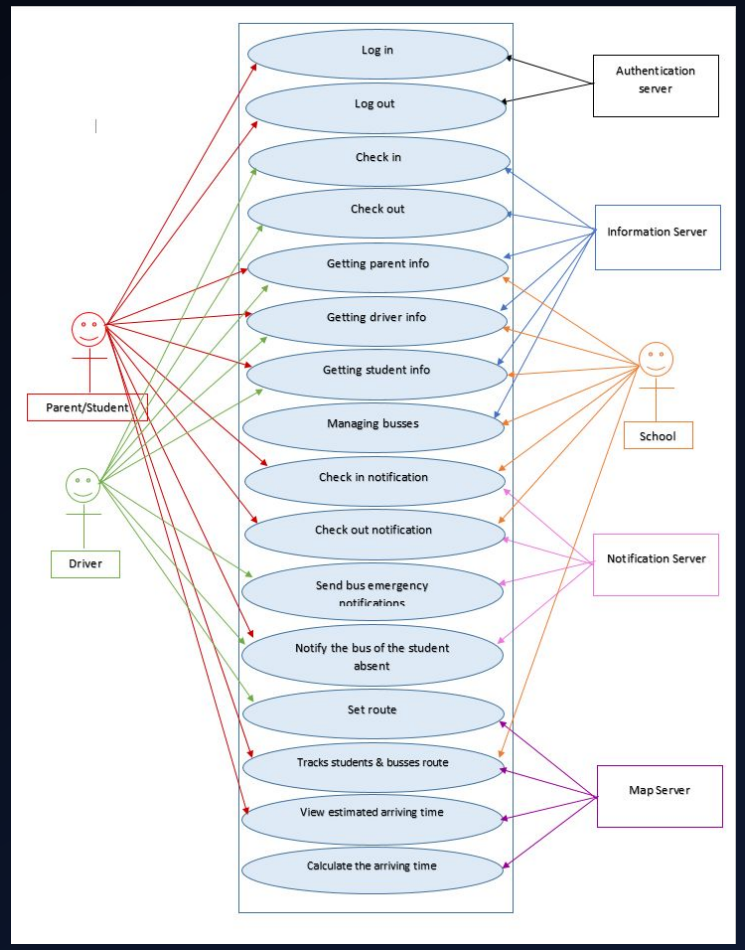
Use Case Diagram

Actors

- Parents
- Students
- Drivers

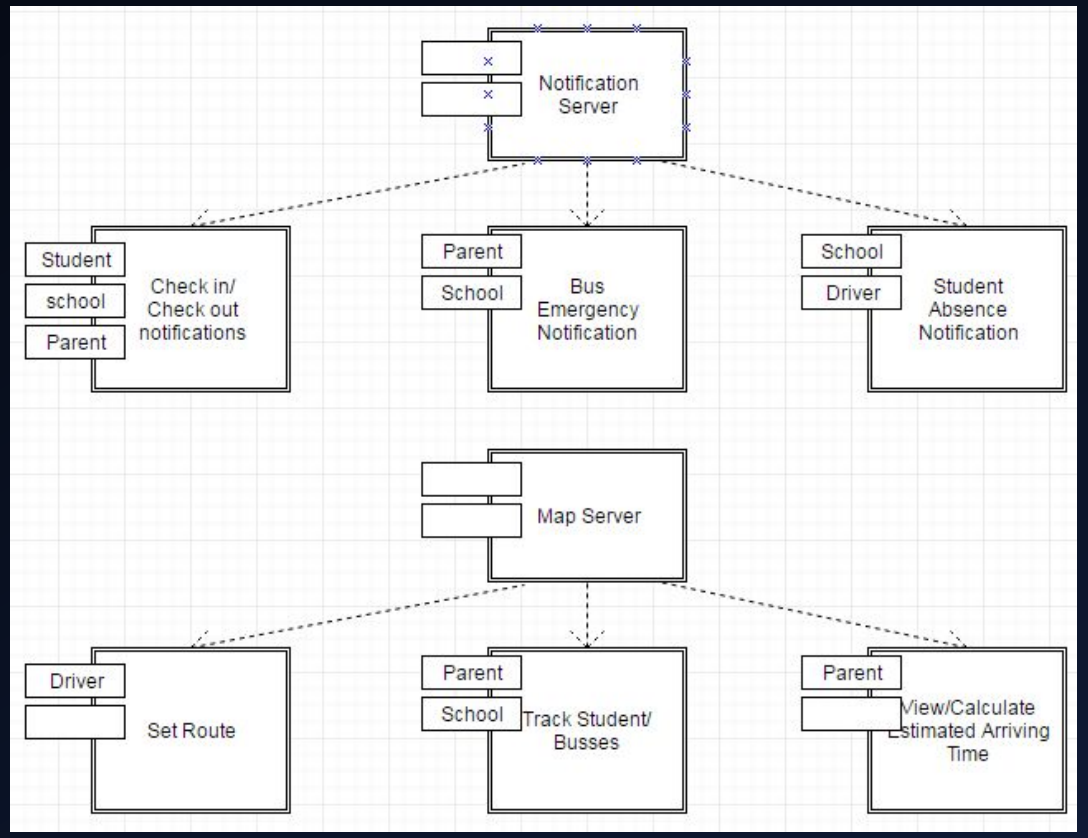
Secondary Actors

- School
- Databases



Subsystem Decomposition

- Examples



Persistent Data Management

FireCloud Database

- User Information
- Routes

Personal Device

- Username/Password
- Personal Settings

Text Use Cases

EXAMPLE 1

Use Case ID: BSB01 - Manage Routes

Scenario:

Actor: Administrator

Pre-conditions:

1. Administrator is logged in to the system.

Description:

1. Use case begins when the Administrator selects the “Administrator Features” option from the app menu.
2. The system presents the user with the Administrator Features menu.
3. Administrator selects the Manage Routes option from the menu.
4. System retrieves route list for Administrator’s registered school.
5. System presents list of routes to user.
6. Administrator selects the route they wish to modify.
7. System pulls up detail form for selected route.
8. Administrator changes desired details of route and hits “submit changes”
9. Use Case ends when system updates form and returns Administrator to Administrator Features menu.

EXAMPLE 2

Use Case ID: BSB02 – Send Bus Emergency Notification

Scenario:

Actor: Driver

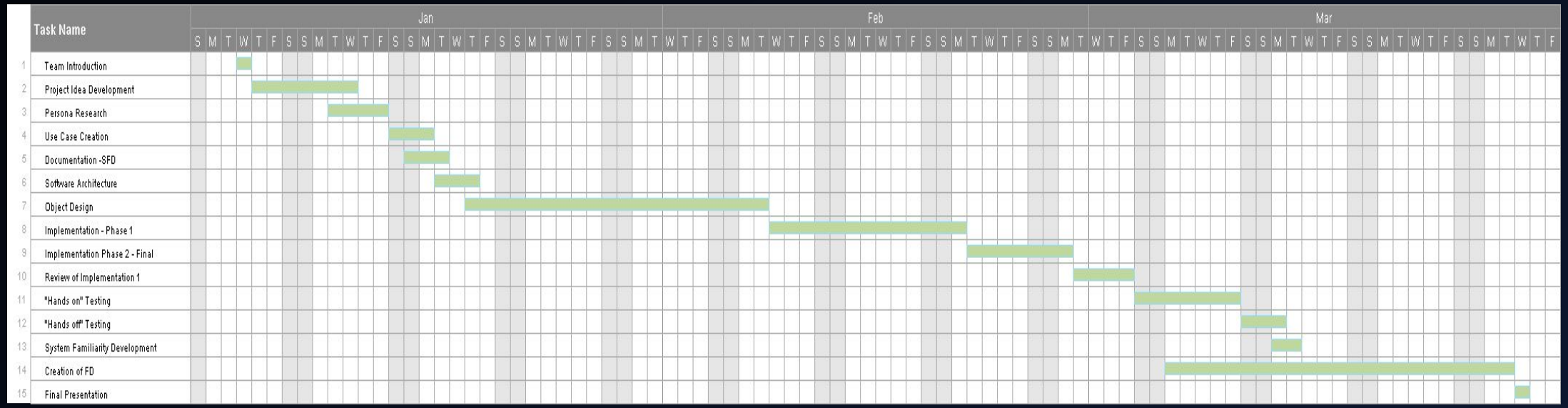
Pre-conditions:

1. User is logged in to the system.

Description:

1. Use case begins when a registered Driver selects “Send Emergency Update” from app menu.
2. System presents the Update form to user.
3. Users enters text detailing desired broadcast message.
4. System sends notification to all users registered to that Driver’s route.
5. Use case ends when system returns user to app menu.

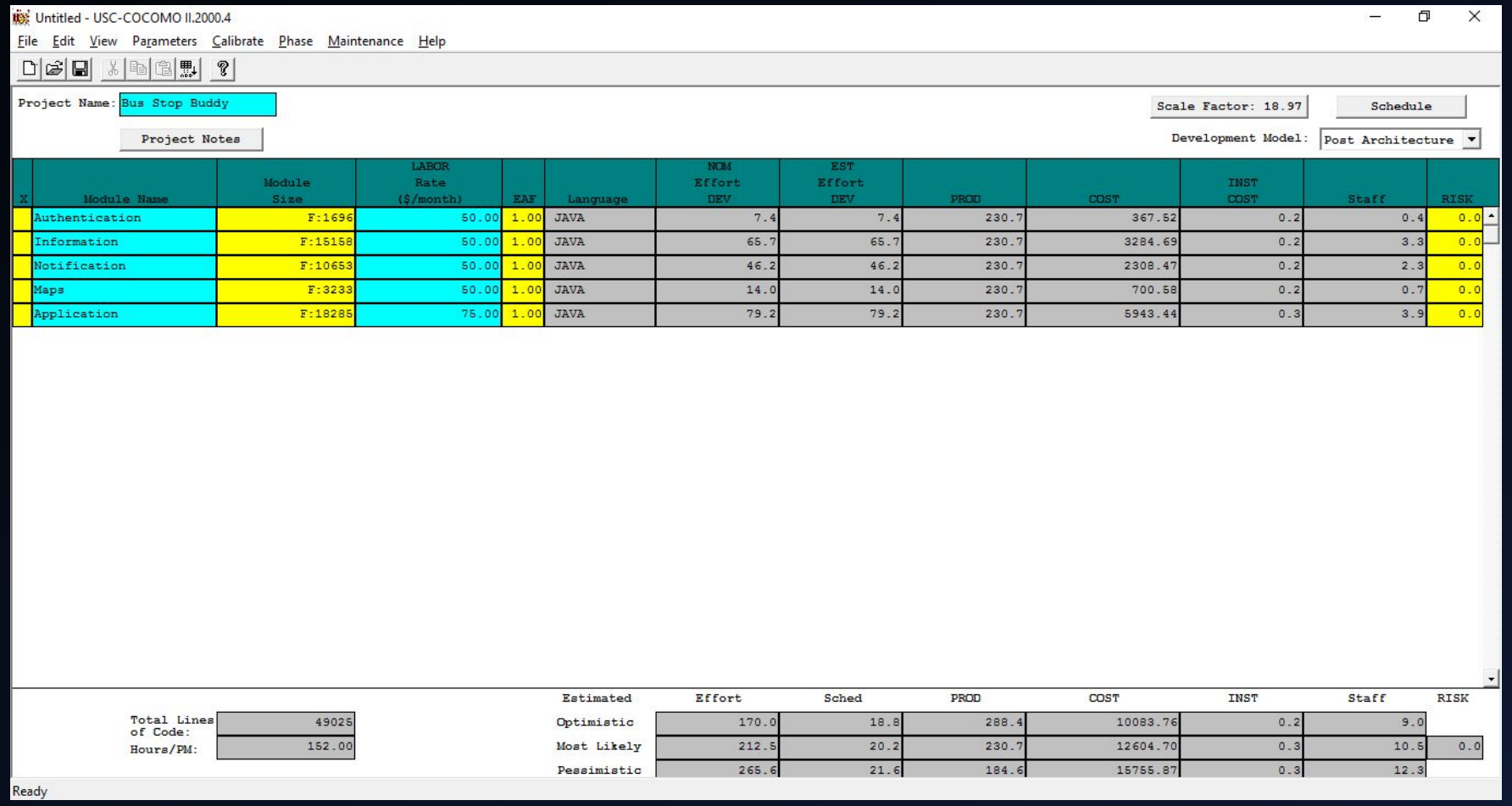
Gantt Chart


Task Name
Jan
Feb
Mar
1 Team Introduction
2 Project Idea Development
3 Persona Research
4 Use Case Creation
5 Documentation -SFD
6 Software Architecture
7 Object Design
8 Implementation - Phase 1
9 Implementation Phase 2 - Final
10 Review of Implementation 1
11 "Hands on" Testing
12 "Hands off" Testing
13 System Familiarity Development
14 Creation of FD
15 Final Presentation

Risk Management

Risk	Category	Probability	Impact (1 Lowest to 5 Highest)	RMMM	Risk Exposure
Problem Implementing Authentication Module	Schedule Risk	4.00%	2	Monitoring - Help catch problems early	0.04
Problem designing Mapping Module	Performance Risk	3.00%	5	Monitoring - Help catch problems early	0.15
Designed a product no one wants	Performance Risk	5.00%	5	Management - Talk with the product owners, consider redesigning product	0.25
Hardware Failure	Cost Risk/Schedule Risk	10.00%	3	Mitigation - Backup computers	0.3
Member of the team falls ill/drops out and is unable to work	Schedule Risk	10.00%	3	Management - Be able to recruit new members/have backups/delegate tasks effectively	0.3
Problem Implementing Notification Module	Schedule Risk	12.00%	3	Monitoring - Help catch problems early	0.36

COCOMO


Untitled - USC-COCOMO II.2000.4
File Edit View Parameters Calibrate Phase Maintenance Help
Project Name: Bus Stop Buddy
Project Notes
Scale Factor: 18.97
Schedule
Development Model: Post Architecture
X | Module Name | Module Size | LABOR Rate ($/month) | EAF | Language | NOM Effort DEV | EST Effort DEV | PROD | COST | INST COST | Staff | RISK
Authentication | F:1696 | 50.00 | 1.00 | JAVA | 7.4 | 7.4 | 230.7 | 367.52 | 0.2 | 0.4 | 0.0
Information | F:15158 | 50.00 | 1.00 | JAVA | 65.7 | 65.7 | 230.7 | 3284.69 | 0.2 | 3.3 | 0.0
Notification | F:10653 | 50.00 | 1.00 | JAVA | 46.2 | 46.2 | 230.7 | 2308.47 | 0.2 | 2.3 | 0.0
Maps | F:3233 | 50.00 | 1.00 | JAVA | 14.0 | 14.0 | 230.7 | 700.58 | 0.2 | 0.7 | 0.0
Application | F:18285 | 75.00 | 1.00 | JAVA | 79.2 | 79.2 | 230.7 | 5943.44 | 0.3 | 3.9 | 0.0
Total Lines of Code: 49025
Hours/PM: 152.00
Estimated | Effort | Sched | PROD | COST | INST | Staff | RISK
Optimistic | 170.0 | 18.8 | 288.4 | 10083.76 | 0.2 | 9.0
Most Likely | 212.5 | 20.2 | 230.7 | 12604.70 | 0.3 | 10.5 | 0.0
Pessimistic | 265.6 | 21.6 | 184.6 | 15755.87 | 0.3 | 12.3
Ready

Demo

Scope too Large?

- Too many plans
- Not enough expertise
- Schedule changes
- Functional Vs. Nonfunctional

Too Many Features, Not Enough Time!

RFID

- Simplifies student check in
- Nonfunctional Requirement

CLOUD DATABASE

- Single storage location for information.

Too Many Features, Not Enough Time!(cont)

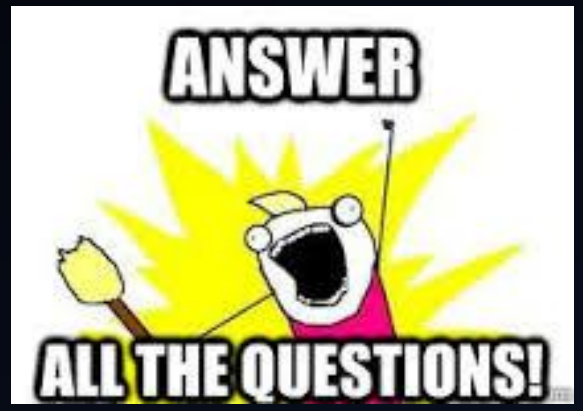
NEW ROUTE

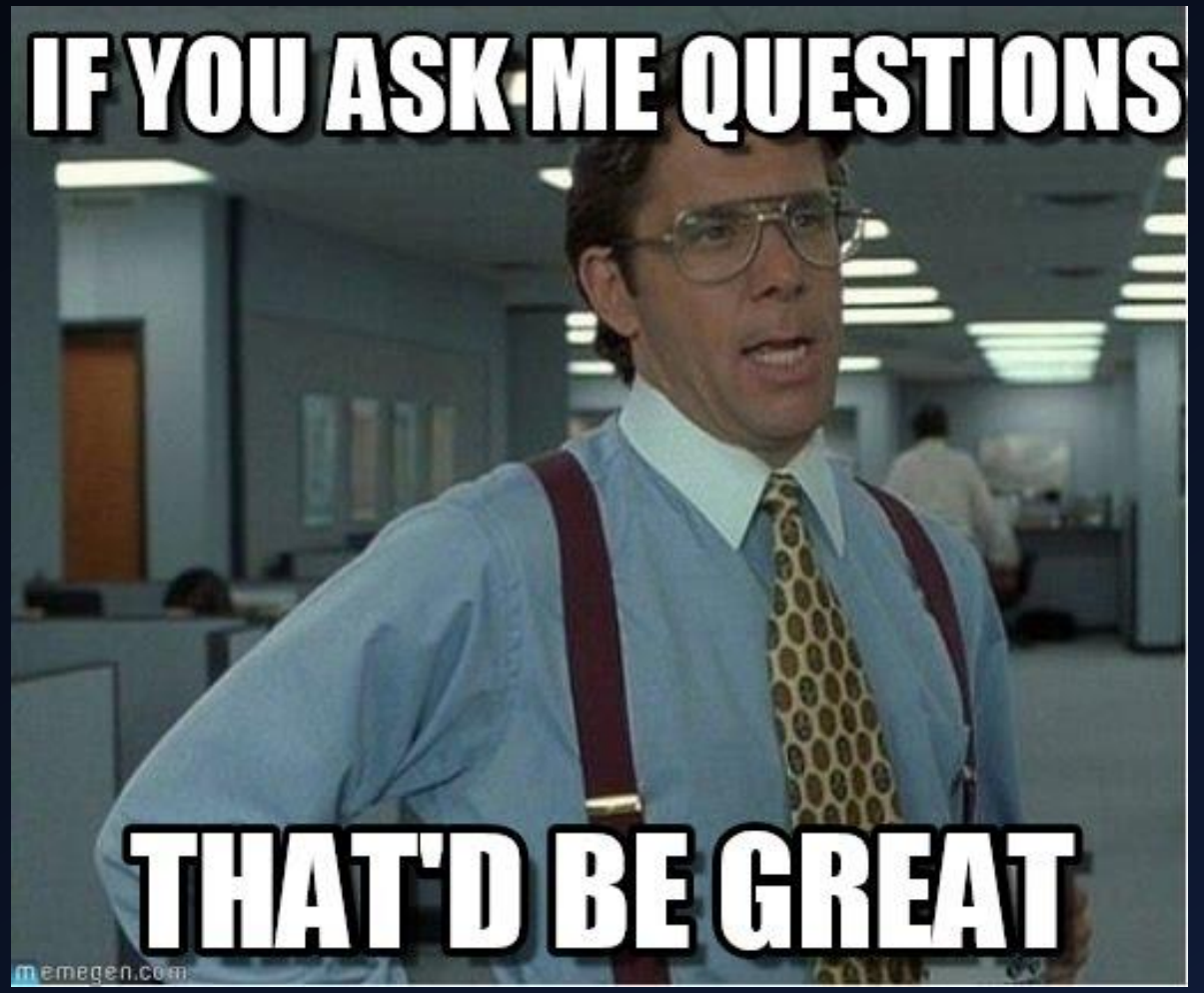
- Allows updating of routes incase routes get longer or shorter
- New route creation

ETA

- Traffic
- Amount of kids in queue
- nonfunctional

Questions?


ANSWER
ALL THE QUESTIONS!


IF YOU ASK ME QUESTIONS
THAT'D BE GREAT
memegen.com

Thanks for Listening!

- Be sure to leave us feedback so we can fine-tune our project!