Software Design Specification

CSUEB Computer Science Department

CS4310 - PROJECT

ontime

Submitted By: Group Gamma

Version 1.0

Description of Project

DOCUMENT NO:	2
VERSION:	1.1
LINK:	<here>
CONTACT:	Group Gamma
EMAIL:	CSUEBGamma@gmail.com
DATE:	11/30/15

Disclaimers

The information contained in this document is the proprietary and exclusive property of CSUEB except as otherwise indicated. No part of this document, in whole or in part, may be reproduced, stored, transmitted, or used for design purposes without the prior written permission of CSUEB.

The information contained in this document is subject to change without notice.

The information in this document is provided for informational purposes only. CSUEB Computer Science Department specifically disclaims all warranties, express or limited, including, but not limited, to the implied warranties of merchantability and fitness for a particular purpose, except as provided for in a separate software license agreement.

Privacy Information

This document may contain information of a sensitive nature. This information should not be given to persons other than those who are involved in the ***ontime*** project or who will become involved during the project life-cycle

Version History

REVISION CHART

Version	Author(s)	Description of Changes	Date Completed
1.0	Andreas Slovacek	Completed Document	11/16/2015
1.0	Jialiang Zhong	Completed Document	11/16/2015
1.0	David Odza	Completed Document	11/16/2015
1.0	Chris Guerra	Completed Document	11/16/2015
1.0	Daniel Jacob	Completed Document	11/16/2015

Document Owner

The primary contact for questions regarding this document is:

Author: Gamma Group

Project Name: ontime

Email: CSUEBGamma@gmail.com

Document Approval

Document Name: Software Design Specification for ontime

Publication Date: 11/16/2015

Contract Number: N/A

Project Number: 1.0.0

Prepared by: Group Gamma

Approval: _____

Name and Organization ***Professor Hank Stalica***

Table of Contents

Detailed Document Description

This section describes the contents of each section of the Software Design Specification.

1. Introduction

Purpose:

This Software Design Specification (SDS) provides an overview of the proposed ontime Android application. It will encompass in detail the basic outline of our project and represent a basis for the development process. This will also allow critical analysis of the logical and functional aspects of the design before any commitment is made to actual code. The ontime Android application is a tool designed as a prototype to demonstrate an application interface that can utilize the public BART API functionality for the Android mobile OS platform. We will also consider publishing as a free application.

Document Overview

Below is the outline of the each section described in this document.

- Chapter 1 – Document Description
- Chapter 2 - System Overview
- Chapter 3 – Design Considerations
- Chapter 4 – Architectural Strategies
- Chapter 5 – System Architecture (Use Cases from SRS)
- Chapter 6 – Policies and Tactics
- Chapter 7 - Design Documents
    - Black Box Design
    - White Box Design
    - Database Design

Scope:

The scope of the design document is to illustrate the functionality of the public BART API. This prototype application is an online tool. It will use the public BART API and the Android mobile platform OS to demonstrate available BART requests.

The design document will also show interactions between the web services, between different requests both riders and potential riders who can utilize Android applications who are the main actors in the design.

SDS will be used by the Professor Hank Stalica and the Gamma development team.

Revision History

Date	Revision	Description	Author
11/16/2015	1.0	template, cover page, formatting/editing Sections: 1, 2, 3, 5, 7.1, 7.2.3,7.2.4, Glossary	Daniel Jacob
11/12/15	1.0 2.0	Use Cases 2,3,4,5,6,7 & UML diagrams. added map, fare, fare2, display, time, and main activity class diagrams. Also added data flow diagram. added to section 6/ added uml class model descriptions.	Chris Guerra
11/30/2015	1.0, 2.0	Section 4, 7.1, 7.2.1 Use Cases 1,8 (UML diagrams and data flow diagrams), formatting, 2.0: Revised Section 4 explaining every method. Revised 7.2.1	David Odza
11/16/2015	1.0	Section 2 cleanup, 5 clearnup, 7 cleanup, 7.2.3 (UML for BartApi)	Andreas Slovacek
11/27/2015	1.2		Andreas Slovacek
		backlog completed.	Back log

References:

Online Reference and Printed Materials

BART API
http://api.bart.gov/docs/overview/index.aspx

Additional References:

Professor Hank Stalica, CSUEB (Client,) California, CA.

Methodology, Tools and Techniques

Microsoft Word document, OmniGraffle diagrams, GenMyModel diagrams, Photoshop images and Android Studio are the tools used to create this design document. Use cases and UML diagrams are created to describe the scenarios.

Key Stakeholders

Project stakeholders are below:

- Prof. Dr. Hank Stalica, CSUEB (client)
- Group Gamma, students, creators,
  - Chris
  - Andreas
  - David
  - Jay
  - Daniel Jacob

Points of Contact

- Gamma Group, CSUEB. csuebgamma@gmail.com

Definitions, important terms, acronyms, or abbreviations:

GUI	Graphical User Interface - a visually based application that serves to provide an interactive medium between the user and the application.
SRS	Software Requirements Specification - the explicit requirements definition used to maintain product consistency during the development process
Web site	A hierarchy of linked HTML-encoded text files that display on a web browser as a series of related text pages with embedded graphics and controls
DFD	Data Flow Diagram

1.1 Overview of Document

Section 1.0 introduces the project. Section 2.0 provides an abstract view of the system architecture, including the components, structure and relationships, and user interfaces. Section 3.0 describes each of these components in more detail, including design and architectural decisions. Section 4.0 explores the relationships to other products. Section 5.0 discusses design decisions, trade-offs, and the reasoning behind these decisions. Section 6.0 is reserved for policies and tactics. It also discusses design patterns that can be applied. Section 7.0 has detailed diagrams. It has both black box model and white box model.

2. System Overview

Module	Description
User Splash Screen	The starting page which asks for initiates the user. Also user is expected to click the start button to insure human interaction is in place. In other words, there is not automatic software or script trying to access to the page.
Estimate Cost	Provides accurate cost accounting for user trip request.
Delay Advisory	Contains the details of the latest Delay Advisory provided from BART.
System Map	This module displays the BART map.
Estimate Arrival	This module display the estimated arrival based on the user requested configuration provided by this form.
Support Page	The pages that provide the information to the User and enables email functionality to product support account.

3. Design Considerations

This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution.

Assumptions and Dependencies

The diagrams, Images and Screenshots in this document were created through Omnigraffle Diagram tool, Adobe Photoshop, GenMyModel, Android Studio or Nexus 4 device.

Related software or hardware

This program will be coded in Google Android Studio 1.4, 2015. Physical device testing will be Google Nexus 4 running stock Android OS Version Marshmallow. Software version system will be git. Github will be used for code sharing and collaboration. Document and issue tracking will be through Google Drive.

End-user characteristics

1- ontime User - The users of the ontime system.

2- Network – the LAN that exists between the two machines that will be involved in this system.

3- Administrator – One of the creators of the ontime system. This user can configure system administrative controls.

4- Internet – the Internet connection of the ontime User's Android device to be able to use the ontime real-time tool.

General Constraints

1) Hardware or software environment

   Installation and capable expertise in Android Studio 1.4.1, Git, Github.

2) End-user environment

   ontime User should have network/Internet connectivity. They configure their device accordingly

3) Availability or volatility of resources

   This depends on the network and Internet connection. Real time values will be done via online. Stability and availability can be measured with the number of failures on the Internet connection.

4) Legal compliance

The BART API is a publicly provided web service available for free. This application is dependant upon this service

5) Standards compliance

None

6) Interoperability requirements

None

7) Interface/protocol requirements

Network connectivity and TPC/IP support are required.

8) Data repository and distribution requirements

None

9) Security requirements (or other such regulations)

None

10) Memory and other capacity limitations

5MB/10MB Storage capacity is required.

11) Performance requirements

No internal failures are acceptable. The only known and accepted failures are from the network or Internet connection that is providing the communication between the two machines.

12) Network communications

Network should be up all the time as part of the functionality is to be able to use the ontime BART tool on the network or Internet.

13) Verification and validation requirements (testing)

WIFI, and mobile connectivity environments will be tested. Functionality via Nexus 4 demonstrating features.

14) Other means of addressing quality goals

None

15) Other requirements described in the requirements specification

a) Online User Documentation and Help System Requirements
The instructions for ontime can be found online at:
https://drive.google.com/drive/folders/ontimeUserdocs-HelpReqs

b) Design Constraints
Adherence to Android Material design standards.
https://developer.Android.com/design/material/index.html

c) Purchased Components

Android Studio, Git, Github, and google services all utilizing free services. Photoshop and Omnigraffle were previously acquired. CSUEB tuition expense.

d) Interfaces

i) User Interfaces

Online application and user interface will be designed on paper, and executed with Android Studio using XML and JAVA languages.

ii) Hardware Interfaces

Web Server needs to be installed and configured.

iii) Software Interfaces

No database software will be used.

iv) Communications Interfaces

BART API and Android phone application.

e) Licensing Requirements

None, public freeware.

f) Legal, Copyright, and Other Notices

g) Applicable Standards

Android API 15 minimum Ice Cream Sandwich supporting 94% of Android mobile OS users.

Goals and Guidelines

1) Apply The KISS principle ("Keep it simple and straightforward!").

The eight requirements that identify for a good design which are well structured, simple, efficient, adequate, flexible, practical, implementable and standardized are the guidelines to create this design.

2) Emphasis on speed versus memory use. Native Android libraries will be utilized before thirds parties.

3) Working, looking, or "feeling" like an existing Android application. Utilizing Android Material design guidelines.

4) Explore the Android platform and how best display the transit information to users.

5) Explore the BART API to discover potential features that could be extended from existing features.

The goal of this project is to deliver the product application on time. Use all the recommended models in the design document during coding. At the end we will demonstrate a prototype tool that uses the BART API Web services.

4. Architectural Strategies:

Design Patterns Description:

The Ontime BART app has a MVC (Model View Controller type of architecture).

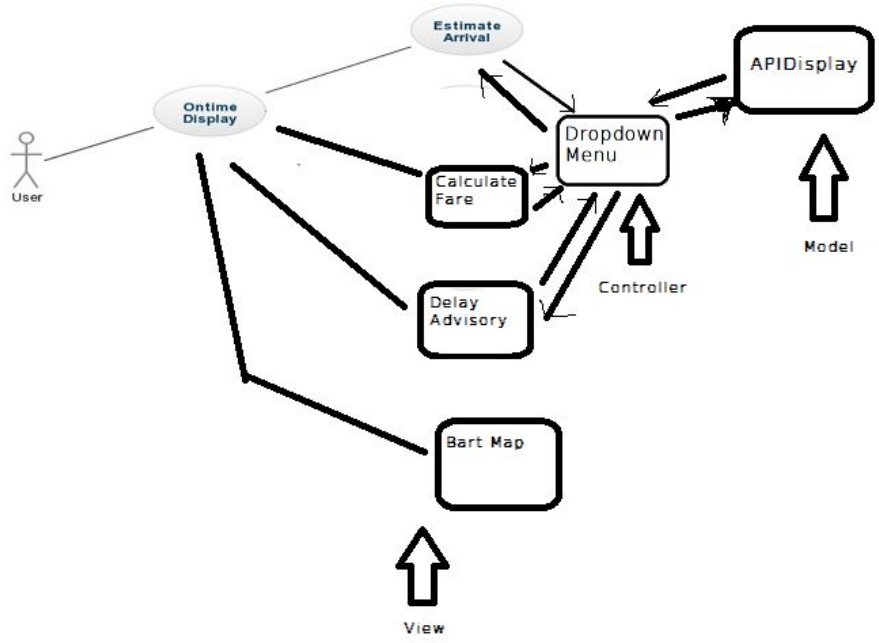
The Model is built into the APIDisplay class. The APIDisplay class is responsible for retrieving data from the BART API. The APIDisplay class handles and manipulates all the data that the app needs. The APIDisplay behaves as an engine to provide data to the controller, so that the controller can display it in the view when applicable. The APIDisplay is constructed in a way that allows easy compatibility with the controller, by having three strings an input. These strings specify to the model what data it needs, and for what BART stations this data is being needed for. The data is also retrieved and parsed into a string for easy use by the controller and view. The model of all the data is in the form of strings, which makes the data easy to display and pass around.

The controller is handled by the DropDown class. This class is used for taking inputs from the user which are made in the view, and sending that data to the model in the correct format. The DropDown class converts the data input into the view into something more friendly for the APIDisplay to work with. Then the model provides DropDown with the data that is requested. DropDown puts the data from model into a nicer format and sends it to the view to be displayed.

The view is broken down into the rest of our apps classes. These classes are: display, fare, fare2, MainActivity, map and time. These classes are for the user to navigate to appropriate pages, and make requests. These are our GUI pages, and they have nice displays, buttons and pictures. These classes are made to be visually pleasing, as well as convenient and simple to use. The view class don't know how the model is getting data or presenting data, because the controller will provide the view with the data in a nicely formatted way. The view is responsible for telling the controller what data to retrieve.

The Ontime BART apps architectural strategies make it a very versatile piece of code. This app is designed to be compatible with changes made to any components of the app. The compatibly of the code on all ends makes it easy to combine code together, add new features, and change existing code without doing a full remodel of the apps front and back end.

Use Case 1 Diagram 1 Software Design Specification



Documentation:

Set of procedures:

Model:

public class BartApi - this class is responsible for the backend, mostly calling the BART API and getting data.

public String printDestinationTime(String origin, String destination)- this function sends the arrival and departure information to the controller in the form of strings.

public String printFare(String origin, String destination)- this function sends the fare information to the controller in the form of strings.

public String printAdvisory()- this function sends the advisory information to the controller in the form of strings.

public class ApiDisplay **extends** AsyncTask<String, Void, String> - this is a built in java method that allows multi-threaded processing. It's used to make the API calls on a seperate thread.

protected String doInBackground(String...params)- This is the code which the Async task will execute, in our case calling the API when the user has pressed submit.

protected void onPostExecute(String result)- This is what to do after executing the async task. In our case, this will print the results to the screen.

View:

public class display **extends** Activity - This class is the main page of our app. It is used to navigate to any functions of the app.

public void displaymap(View v)- This is executed when the map button is pressed. It sends the user to the map page.

public void advisory(View v)- This is executed when the advisories button is pressed. It sends the user to the advisories page.

public void fare(View v)- This is executed when the fare button is pressed. It sends the user to the fare page.

public void time(View v)- This is executed when the arrival/departure times button is pressed. It sends the user to the arrival/departure times page.

public class fare **extends** Activity - This class is where the functions for the fare page are defined.

protected void onCreate(Bundle savedInstanceState)- Once an instance of the fare class has been called, there are some preliminary steps to prepare the page for the user. They are kept here.

public void addListenerOnButton()- This function is called in onCreate(). It prepares the button to pick up stations selected on the dropdown menus once the user presses the button. It also checks to make sure the user selected two different stations.

public class advisory **extends** Activity

protected void onCreate(Bundle savedInstanceState)- Once an instance of the fare class has been called, there are some preliminary steps to prepare the page for the user. They are kept here.

public void addListenerOnButton()- This function is called in onCreate(). It prepares the button to pick up advisory selected on the dropdown menu once the user presses the button.

public class MainActivity **extends** AppCompatActivity - This is the splash page. It displays our app's logo and offers a button to enter the app.

public void onbuttonclick(View v)- This defines the splash pages button function. When the user presses the button on the splash page, they are directed to the app menu.

public class map **extends** Activity

protected void onCreate(Bundle savedInstanceState)- Once an instance of the map class has been called, there are some preliminary steps to prepare the page for the user. They are kept here.

public class time **extends** Activity

protected void onCreate(Bundle savedInstanceState)- Once an instance of the time class has been called, there are some preliminary steps to prepare the page for the user. They are kept here.

public void addListenerOnButton()- This function is called in onCreate(). It prepares the button to pick up stations selected on the dropdown menus once the user presses the button. It also checks to make sure the user selected two different stations.

Controller:

String stationToSymbol(String station)- This function converts the full name of the BART station selected from the view by the user, into the 4 character symbol used by the model to make the API call. Here is a short example:

```
if (station.equals("24th St. Mission")) {
   stationSymbol = "24TH";
}
if (station.equals("Ashby")) {
   stationSymbol = "ASHB";
}
```

This is repeated for every BART station.

Other procedures may be added before going live with the app. All procedures have comments which describe their usage and purpose in the code.

Domain knowledge

The most tricky part of this app is the API calls. They are made by putting in a URL, for instance

```
public String returnString(String dep, String arr)

String URL = "http://api.bart.gov/api/sched.aspx?cmd=depart&orig="+ dep + "&dest=" + arr + "&date=now&key=MW9S-E7SL-26DU-VV8V&b=2&a=2&l=1";
```

This procedure will take two 4 character strings as arguments. It will create a call to the BART API and then make the call. The specifications of this particular URL make the call return with departure and arrival times for the two stations which were passed in as 4 character symbols. Therefore access to the Internet is absolutely required to run the app.

Environmental constraints:

The environment that this app is run on will be an Android phone or emulator. The version of Android that this app can be run on is Android 2.2 “Froyo”. An Internet connection is essential to this app because it makes calls to the BART API via the Internet to retrieve all of its information. Time constrains prohibited us from including an animation to represent the data, so the user needs to be able to read and calculate times from a textual display. The phone being used must have smart phone Android capabilities and have a screen which can show data. This app uses XML and Java languages. This app shows .png pictures. Errors caught via code in the controller such as these lines from ApiDisplay:

```
class ApiDisplay extends AsyncTask<String, Void, String>:

try ..

catch (IOException e)

{

   return "Not Working";

}

catch (ParserConfigurationException e)

{

   e.printStackTrace();

}

catch (SAXException e)

{

   e.printStackTrace();

}

return "Not Working";
```

The previous code can be modified to show exactly what error is occurring, if errors arise during the controller portion of the code.

5. System Architecture

In this section high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components are provided. Detail about the individual components themselves will be discussed in the detailed design part of this document.

The ontime Android application will be an online tool using BART API Web services. It will have the ability to demonstrate the web services functionality. Ontime Users will be able to request Estimated Cost, Estimate Arrival, Delay Advisory, Product System Map, Feedback functionalities.

Users will be able to make as many requests as desired.

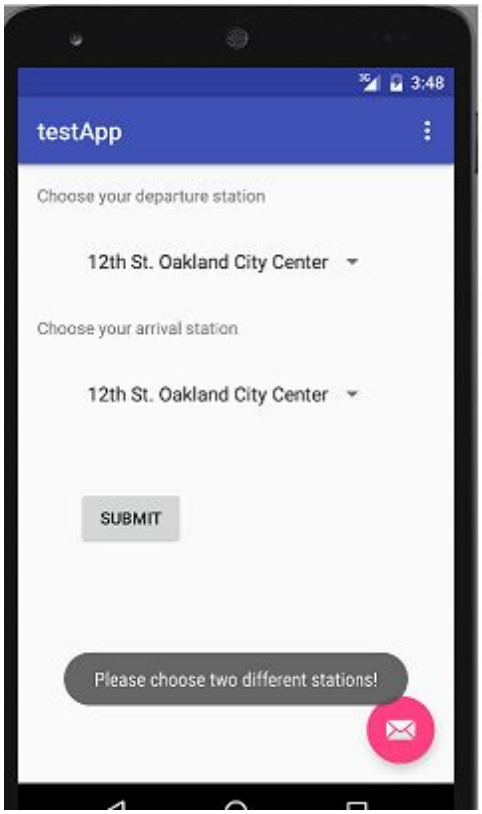
Use cases from the SRS Document

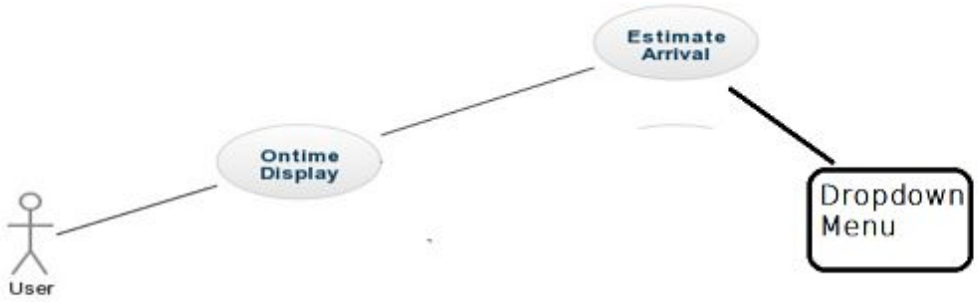
5.2.1 Use Case 1 Specification:

Use Case ID:	**1**		
Use Case Name:	**Departure and Arrival with the same station**		
Created By:	David Odza	Last Updated By:	David Odza
Date Created:	11/12/15	Date Last Updated:	11/12/15

Primary Actors:	User
Secondary Actors:	N/A
Description:	This use case is designed to test the functionality of calling the API to get arrival and departure times to and from the same station. At first this action would crash the app, therefore a test was put in place to prohibit the submission of the same station twice.
Trigger:	Pressing the submit button while the drop down menus have the same stations selected.
Preconditions:	The drop down menus have the same stations selected.
Post-conditions:	Proper usage of the app after executing the call to retrieve departure and arrival times.
Normal Flow:	The user turns on the app. The user navigates to the departure/arrival times section of the app. The user selects two stations with the same name. The user presses submit. The app responds to please choose two different stations and re-submit.
Alternative Flows:	None.
Exceptions:	None.
Includes:	None
Priority:	Low
Frequency of Use:	Low
Business Rules:	None
Special Requirements:	None
Open Issues	None
Assumptions:	The Android phone using this application has a network and Internet connection.
Notes and Issues:	Any Internet connection or network connection issue could cause access problems.

Use Case 1 Diagram 1 Software Design Specification



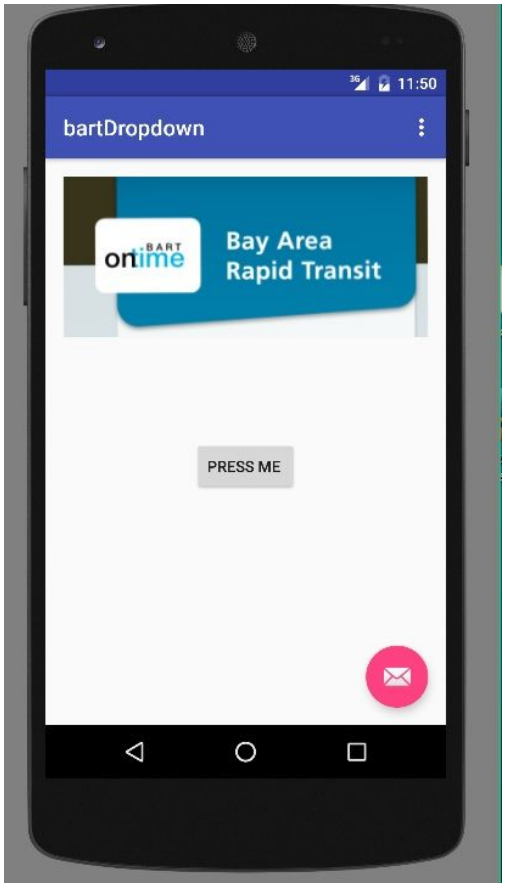


5.2.2 Use Case 2 Specification:

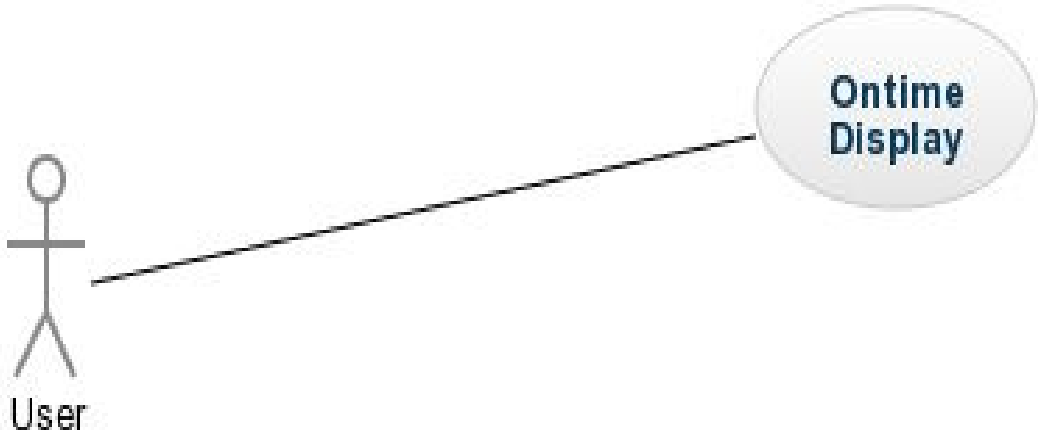
Use Case ID:	**2**		
Use Case Name:	Display Page		
Created By:	Christopher Guerra	Last Updated By:	Christopher Guerra
Date Created:	11/12/2015	Date Last Updated:	11/12/2015

Primary Actors:	User
Secondary Actors:	Platform running app/XML file
Description:	This use case displays how the user will interact with the phone
Trigger:	Once the button "press me " is clicked a new activity will load.
Preconditions:	Actor, The platform must support Android and have the current operating system for android.
Normal Flow:	Actor loads the app and is prompted with a display featuring our logo along with one button.
Alternative Flows:	None
Exceptions:	If the button is not pressed the user will not advance any further.
Includes:	None
Priority:	High
Frequency of Use:	High
Business Rules:	None
Special Requirements:	None
Open Issues	None
Assumptions:	Actor can read English and has fully functioning fingers or an object to press the screen.
Notes and Issues:	None

Use Case #2 Diagram 1:



Use Case #2 Diagram 2:

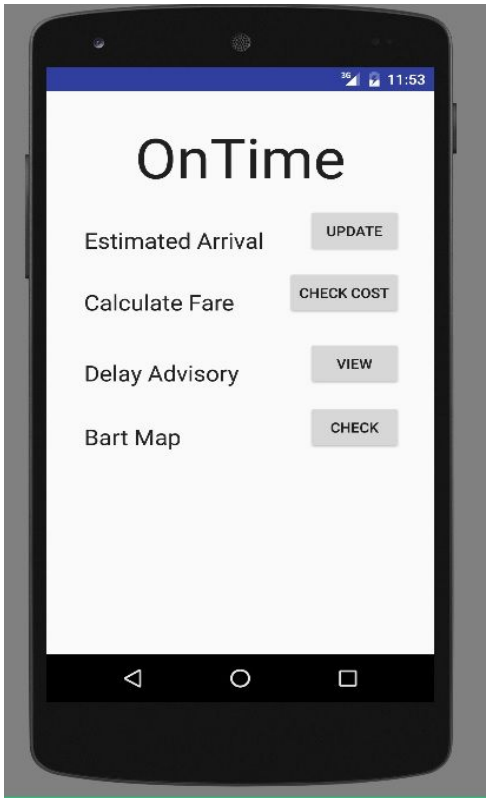


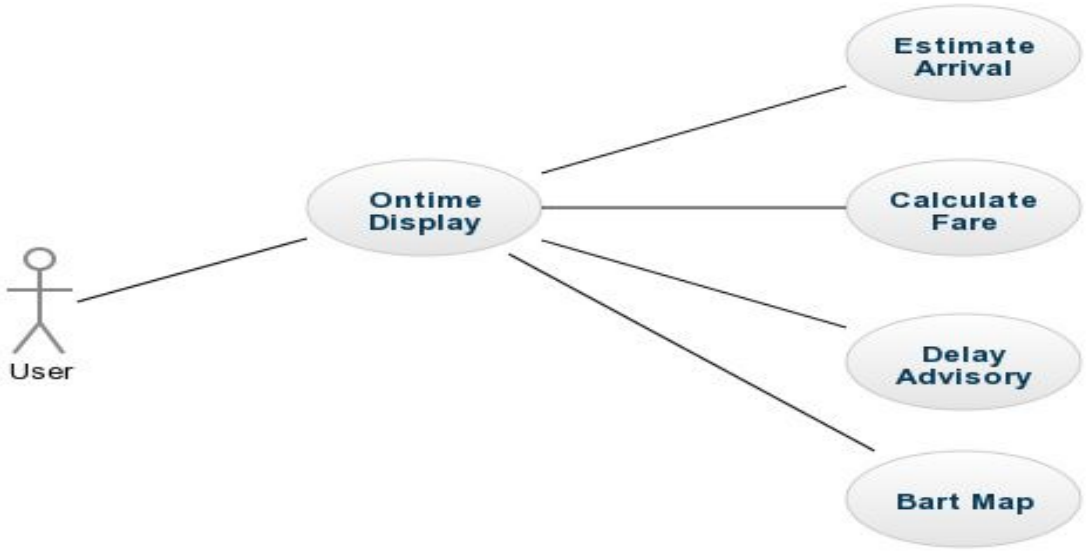
5.2.3 Use Case 3 Specification:

Use Case ID:	**3**		
Use Case Name:	**Ontime Menu**		
Created By:	Christopher Guerra	Last Updated By:	Christopher Guerra
Date Created:	11/14/2015	Date Last Updated:	11/14/2015

Primary Actors:	User
Secondary Actors:	XML files, Phone running the app.
Description:	This use case describes how the user will interact with the apps main menu.
Trigger:	After the button on the display page is pressed the main menu XML will launch displaying a new menu.
Preconditions:	Actor has pressed the “Press Me” button.
Post-conditions:	Actor will be prompted with four options and needs to choose one or will not leave the main menu page.
Normal Flow:	Main menu loads, the actor is presented with four text options along with four button options. Each is linked to a separate XML file and will create a new activity dependent on the button pressed.
Alternative Flows:	None.
Exceptions:	App may crash. If this happens just reload.
Includes:	None
Priority:	High
Frequency of Use:	High
Business Rules:	None
Special Requirements:	None
Assumptions:	Actor can read English and has fully functioning fingers or an object to press the screen.
Notes and Issues:	None

Use Case #3 Diagram 1:





Use Case #3 Diagram 2:

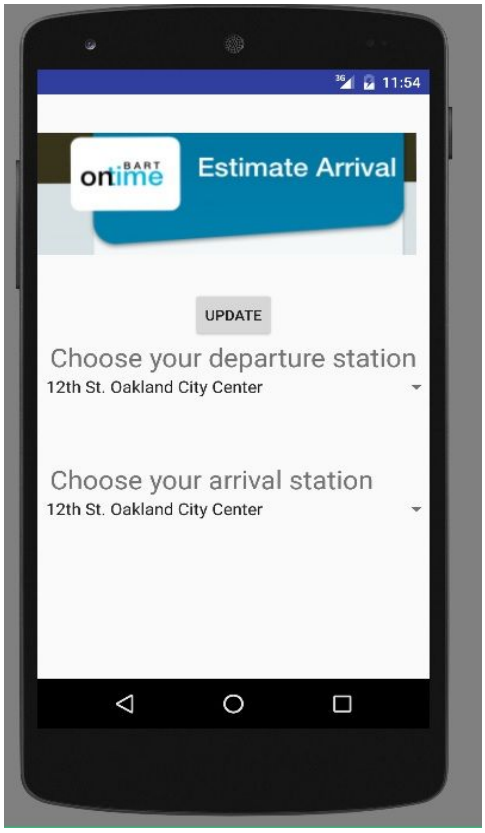
5.2.4 Use Case 4 Specification:

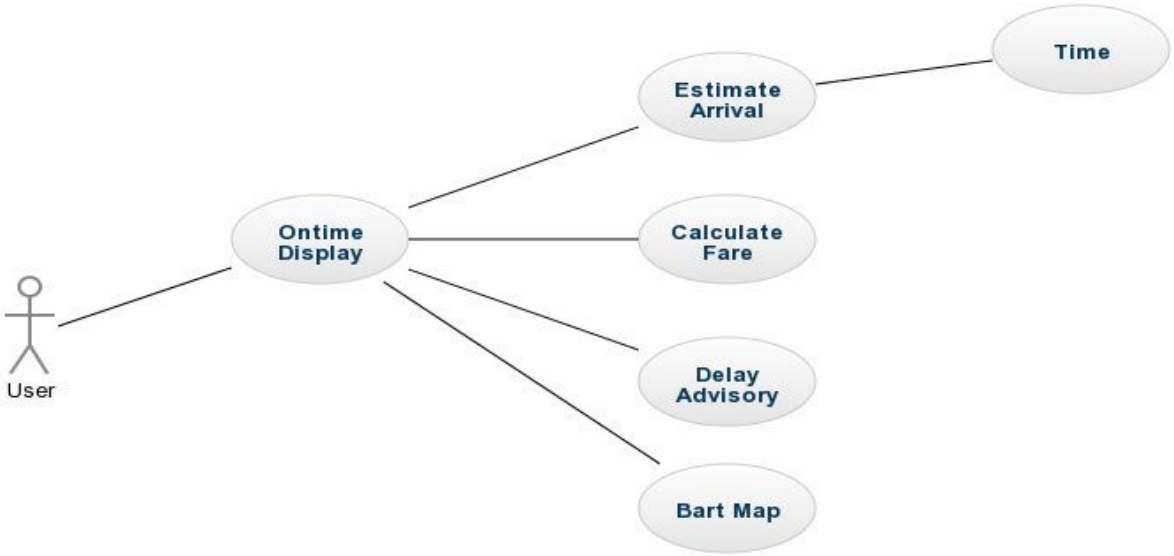
Use Case ID:	**4**		
Use Case Name:	**Display Estimate Arrival**		
Created By:	Christopher Guerra	Last Updated By:	Christopher Guerra
Date Created:	11/14/2015	Date Last Updated:	11/14/2015

Primary Actors:	User
Secondary Actors:	Time XML file, Data connection, Phone running app
Description:	This use case shows what the actor will see when they choose the menu item estimate arrival
Trigger:	User presses the “Update” button.
Preconditions:	Actor has pressed the “Press Me” button.
Post-conditions:	None.
Normal Flow:	Actor presses the “Update” button which loads a new activity called fare. The fare activity shows the user two spinner menus which display all of the Bart stations upon choosing two stations and pressing the update button, the actor will then see the times of arrival based on the two choices.
Alternative Flows:	None.
Exceptions:	If there is no Internet connection the activity will not have the ability to display current train arrivals.
Includes:	None
Priority:	High
Frequency of Use:	High
Business Rules:	None
Special Requirements:	None
Open Issues	None
Assumptions:	Actor can read English and has fully functioning fingers or an object to press the screen.
Notes and Issues:	Any Internet connection or network connection issue will cause access problems.

Use Case 1 Diagram 1 Software Design Specification

Use Case #4 Diagram 1:





USE CASE #4 DIAGRAM 2:

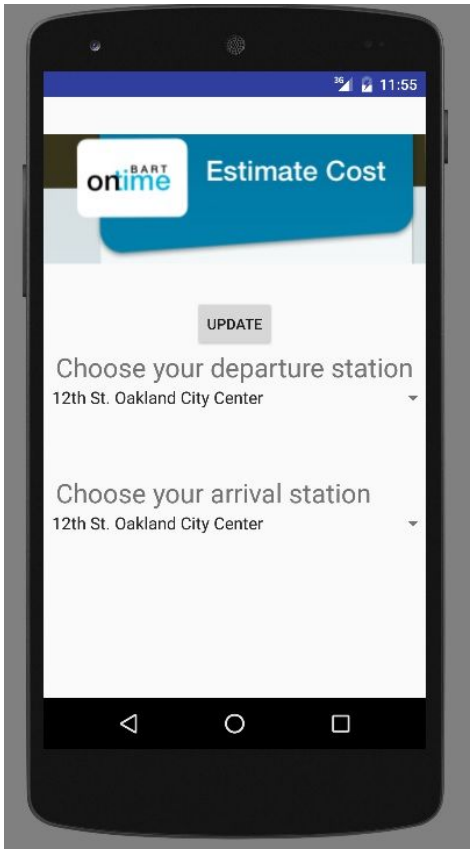
5.2.5 Use Case 5 Specification:

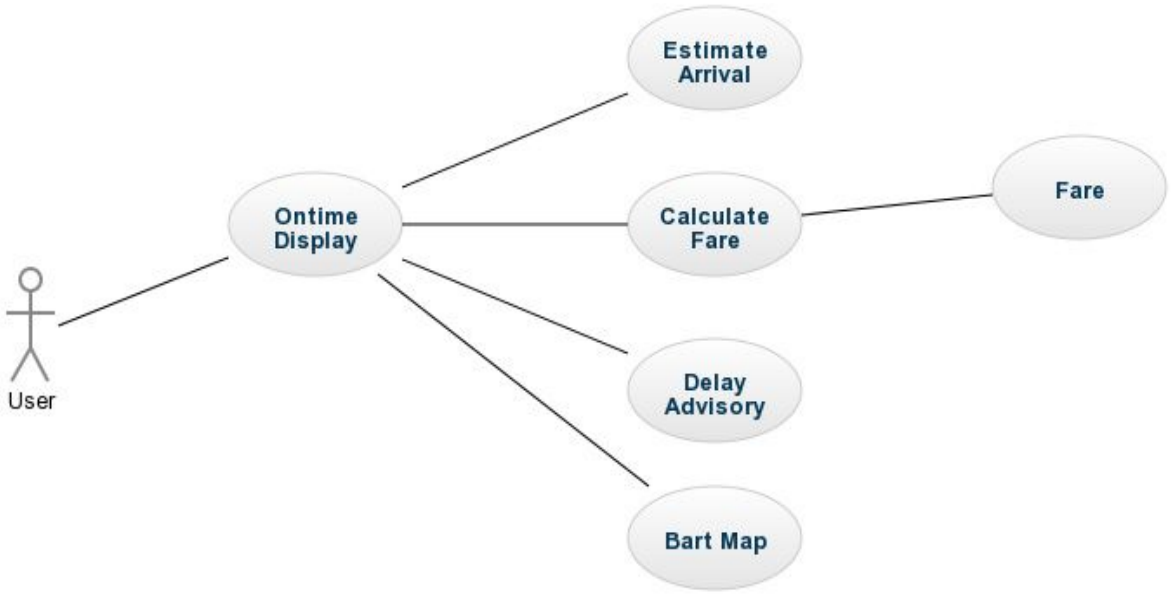
Use Case ID:	**5**		
Use Case Name:	**Calculate Fare**		
Created By:	Christopher Guerra	Last Updated By:	Christopher Guerra
Date Created:	11/14/2015	Date Last Updated:	11/14/2015

Primary Actors:	User
Secondary Actors:	Fare XML file, Data connection, Phone running the app.
Description:	This use case shows what the actor will see when they choose the menu item calculate fare.
Trigger:	Actor presses the “Check Cost” button which will load a new fare activity.
Preconditions	Actor has pressed the “Press Me” button.
Normal Flow:	Actor press the “Check Cost” button on the main menu which loads a new XML activity called fare. In this new activity the actor will see two spinner menus containing station information and an “update” button. Upon entering two stations and pressing update the actor will see the cost to travel between the two stations.
Alternative Flows:	None.
Exceptions:	None
Includes:	None
Priority:	High
Frequency of Use:	High
Business Rules:	None
Special Requirements:	None
Open Issues	None
Assumptions:	Actor can read English and has fully functioning fingers or an object to press the screen.
Notes and Issues:	Any Internet connection or network connection issue will cause access problems.

Use Case 1 Diagram 1 Software Design Specification

Use Case #5 Diagram 1:



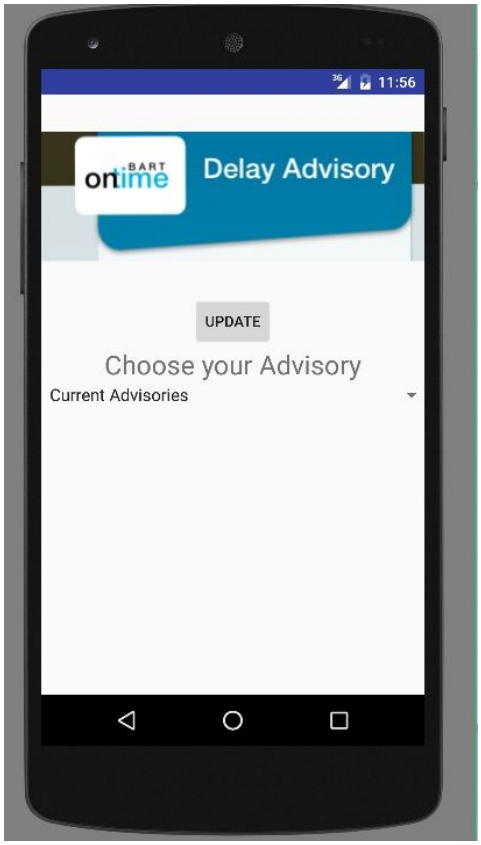


5.2.6 Use Case 6 Specification:

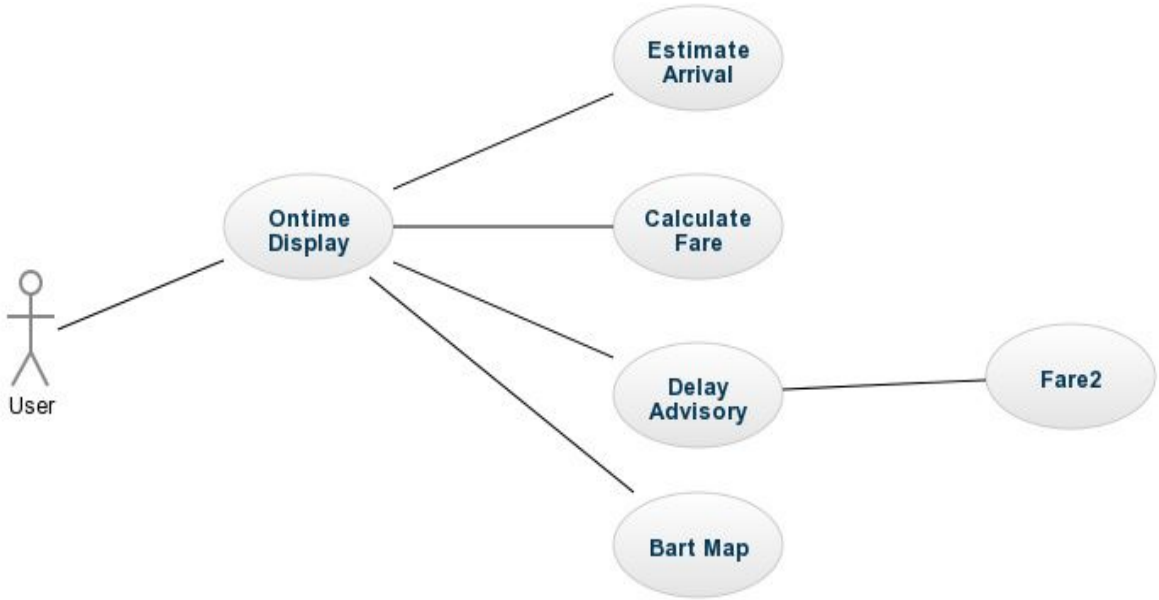
Use Case ID:	**6**		
Use Case Name:	**Delay Advisory**		
Created By:	Christopher Guerra	Last Updated By:	Christopher Guerra
Date Created:	11/14/2015	Date Last Updated:	11/14/2015

Primary Actors:	User
Secondary Actors:	Fare2 XML file, Internet connection, Phone running the app.
Description:	This use case shows what the actor will see when the delay advisory button is pressed.
Trigger:	Actor presses the “View” button on the main menu.
Preconditions:	Actor has pressed the “Press Me” button.
Post-conditions:	None
Normal Flow:	Actor will press the “View” button which will load the Fare2 XML file into a new activity. The actor will see one spinner containing three advisories check.Once the update button is press each advisory will either yield current issues in the Bart station or no issues at all.
Alternative Flows:	None.
Exceptions:	None
Includes:	None
Priority:	High
Frequency of Use:	High
Business Rules:	None
Special Requirements:	None
Open Issues	None
Assumptions:	Actor can read English and has fully functioning fingers or an object to press the screen.
Notes and Issues:	Any Internet connection or network connection issue will cause access problems.

Use Case #6 Diagram 1:



Use Case #6 Diagram 2:

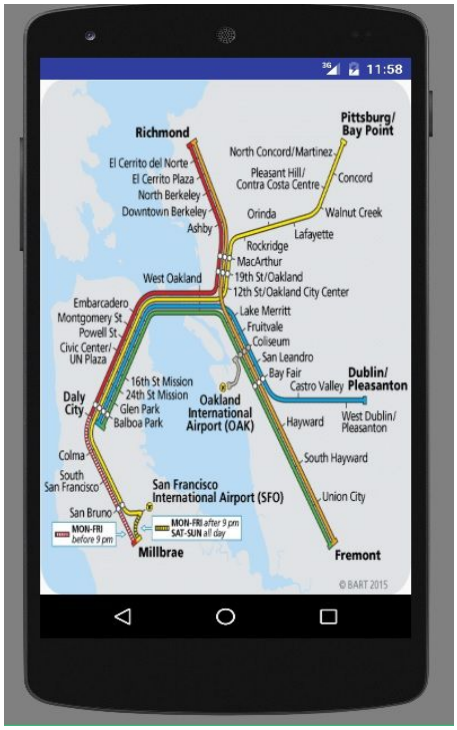


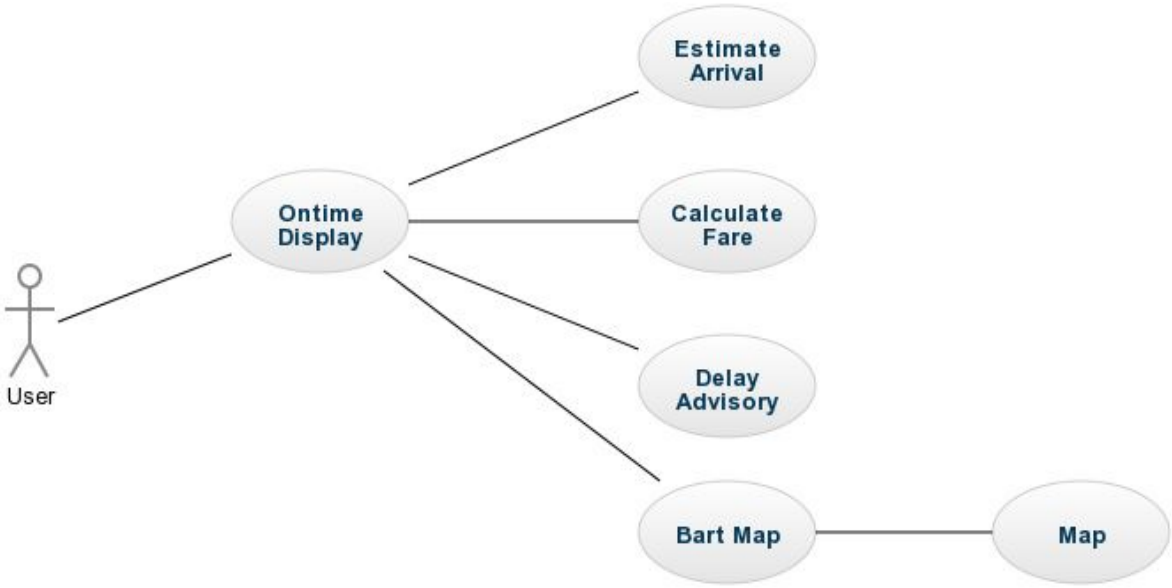
5.2.7 Use Case 7 Specification:

Use Case ID:	**7**		
Use Case Name:	**System Map**		
Created By:	Christopher Guerra	Last Updated By:	Christopher Guerra
Date Created:	11/14/2015	Date Last Updated:	11/14/2015

Primary Actors:	User
Secondary Actors:	Map XML file, Phone running the app.
Description:	This use case shows what the user will see when the “Check” button is pressed.
Trigger:	Actor pressed the “Check” button.
Preconditions:	Actor has pressed the “Press Me” button.
Post-conditions:	None.
Normal Flow:	Actor presses the “Check” button on the main menu which loads the map XML file into a new activity. Once the activity has loaded the user is presented will a detailed map of the Bart system.
Alternative Flows:	None.
Exceptions:	None
Includes:	None
Priority:	High
Frequency of Use:	Moderate
Business Rules:	None
Special Requirements:	None
Open Issues	None
Assumptions:	Actor can read English and has fully functioning fingers or an object to press the screen.
Notes and Issues:	Map is displayed in portrait filling the entire phone screen.

Use Case #7 Diagram 1:



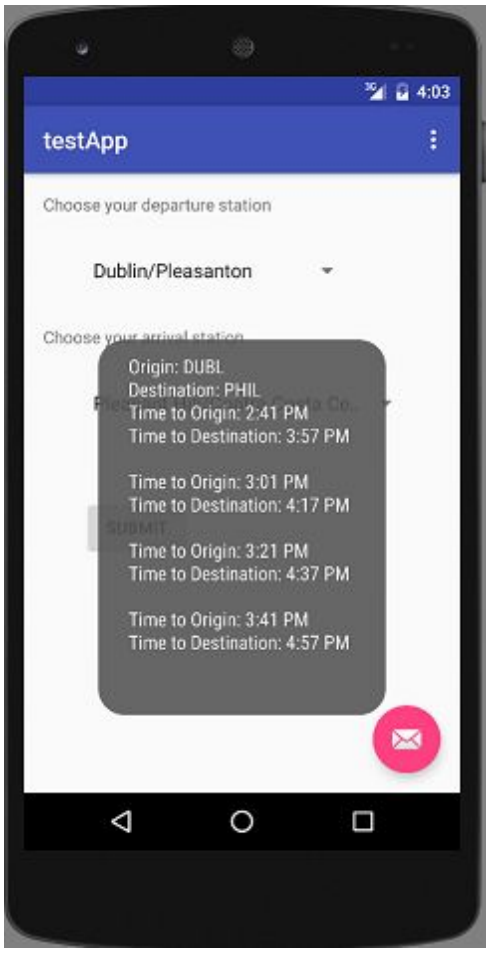


5.2.8 Use Case 8 Specification:

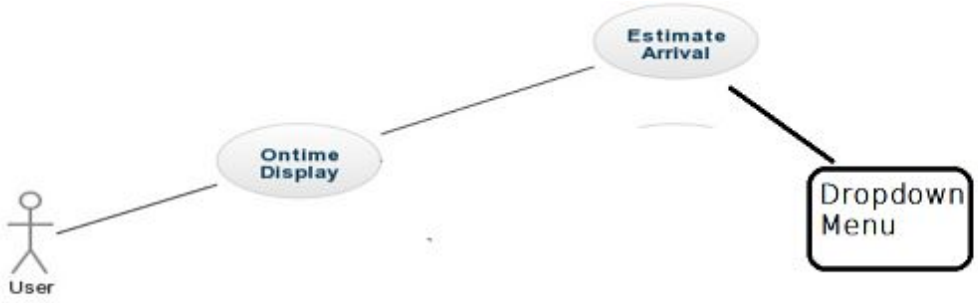
Use Case ID:	**8**		
Use Case Name:	**Departure and Arrival at different stations**		
Created By:	David Odza	Last Updated By:	David Odza
Date Created:	11/15/15	Date Last Updated:	11/15/15

Primary Actors:	User
Secondary Actors:	N/A
Description:	This use case is designed to test the functionality of calling the API to get arrival and departure times to and from the different stations. For this test I chose the departure station Dublin/Pleasanton, and arrival station Pleasant Hill/ Contra Costa Centre.
Trigger:	Pressing the submit button while the drop down menus have different stations selected.
Preconditions:	The drop down menus have different stations selected.
Post-conditions:	The app should display the time of arrival to the departure station. Then the arrival time to the arrival station based on the trip planned. The times should be the 4 times closest to when the call to the API is made.
Normal Flow:	The user turns on the app. The user navigates to the departure/arrival times section of the app. The user selects two stations with different names. The user presses submit. The app calls the API to get arrival and departure times based on the users selected stations. The app displays the arrival and departure times for the specified trip.
Alternative Flows:	None.
Exceptions:	None.
Includes:	None
Priority:	High
Frequency of Use:	High
Business Rules:	None
Special Requirements:	None
Open Issues	None
Assumptions:	The Android phone using this application has a network and Internet connection.
Notes and Issues:	Any Internet connection or network connection issue could cause access problems.

Use Case #8 Diagram 1:



Use Case #8 Diagram 2



6. Policies and Tactics:

1. Choice of which specific product to use (compiler, interpreter, database, library, etc. ...)

   It will be coded in Android studio with the main language being Java. All testing and compiling will be handled in Android studio.

2. Engineering trade-offs

   None.

3. Coding guidelines and conventions

   All code for classes should be documented with comments congruent to JavaDoc. Each API call will take string inputs of origin and destination stations and return a string parsed from the call. Any data parameters will use the current time and date.

4. The protocol of one or more subsystems, modules, or subroutines

   Three out of the four main menu options will require an Internet connection in order for the classes to send and receive data.

5. The choice of a particular algorithm or programming idiom (design pattern) to implement portions of the system's functionality

We will use an Object Pool creation pattern- creating on BartApi object and calling its methods as needed.

6. Plans for ensuring requirements traceability

   Our SRS document was the guidelines for creating our app, all of the design elements in the document are reflected in our app. Also, the use cases were created to make sure that all the functionality requirements of the document were met.

7. Plans for testing the software

   We will test our app with black box. Features and requirements will be tested using the use case scenarios created.

8. Plans for maintaining the software

   Depending on user feedback we will implement a series of patches to accommodate new changes or fix existing issues.

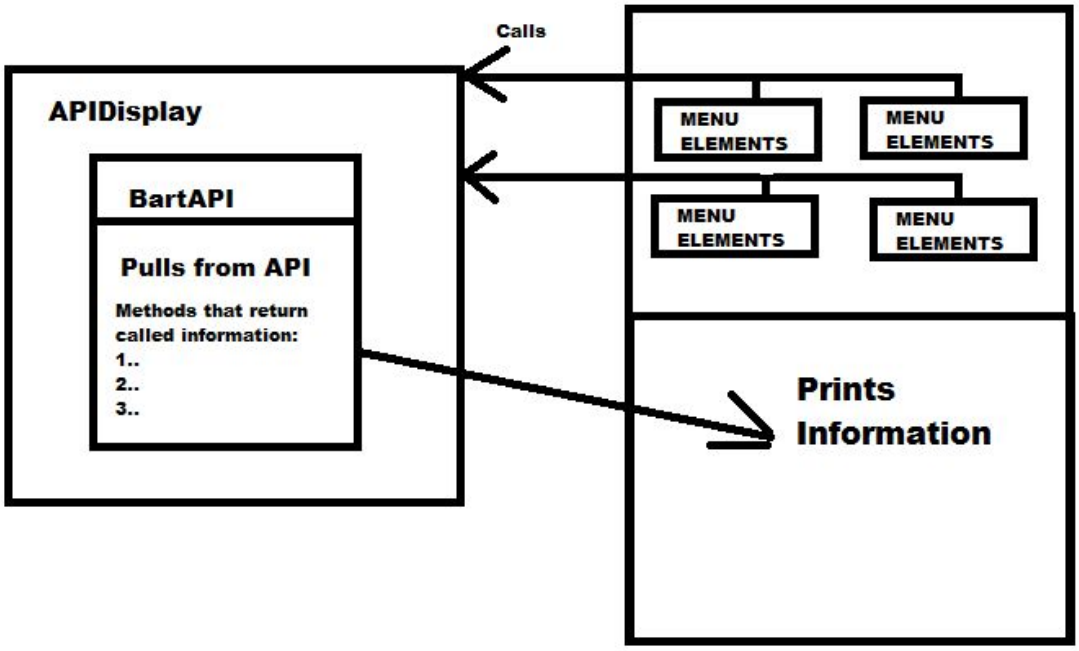
9. Interfaces for end-users, software, hardware, and communications

   Ontime will require end users to have a phone with the most at least API 15 Android ICS version of the mobile OS. A stable data or wifi connection will be needed as well.

6.1 Design Pattern:

The APIDisplay class is a wrapper class that calls the BartAPI class and uses its methods. The purpose of this is so that the basic structure of the Async class can be used without unnecessary duplications of the class's methods ie, background methods and executing methods. With BartAPI being an another class, we can add methods that are similar but serve different parts of the app.

Because the BartAPI class serves as a helper class for the APIDisplay class, most of its methods return string objects that displayed onto the screen. Depending on what menu elements exists, the BartAPI can serve different information given the correct inputs and given the correct context.



The state of the APIDisplay class is not saved as there is no interactions after the a method has been used. However, if there is a need for more layers of interaction between each menu option, the class can "know" by having static variable flags each time a method is used.

7. Design Documents

7.1 Black Box Design for ontime Android application

Black box design of the ontime Android application is done with Data Flow diagram below. This DFD is created from the SRS document provided.

Classification

Class diagrams are drawn for the classes used in this project. Operations and attributes are defined for each class.

Definition

The specific purpose and semantic meaning of the component are below. This black box model is drawn by referring to the the requirements specification document. All the requirements are drawn in this diagram to make it clear for the developer. For additional functionalities main display level is divided into sub levels.

Responsibilities

The primary responsibilities and/or behavior of the interface are:

Main Activity: This is the Home splash page. The user is presented with Press Me button upon application initiation.

Press Me: This is the action the user will take to initiate Display Activity.

Display Activity: This activity will enable the ontime user to access the Update, Check Cost, View, and additional Checks.

Update: User driven Update action will lead to display the Time Activity.

Check Cost: User driven Check Cost action will lead to display the Fare Activity.

View: User driven View action will lead to display the Fare2 Activity.

Additional Check: User driven Additional Check will lead to the Map display.

Time Activity: Displays call data based on configurations.

Fare Activity: Displays call data based on configurations.

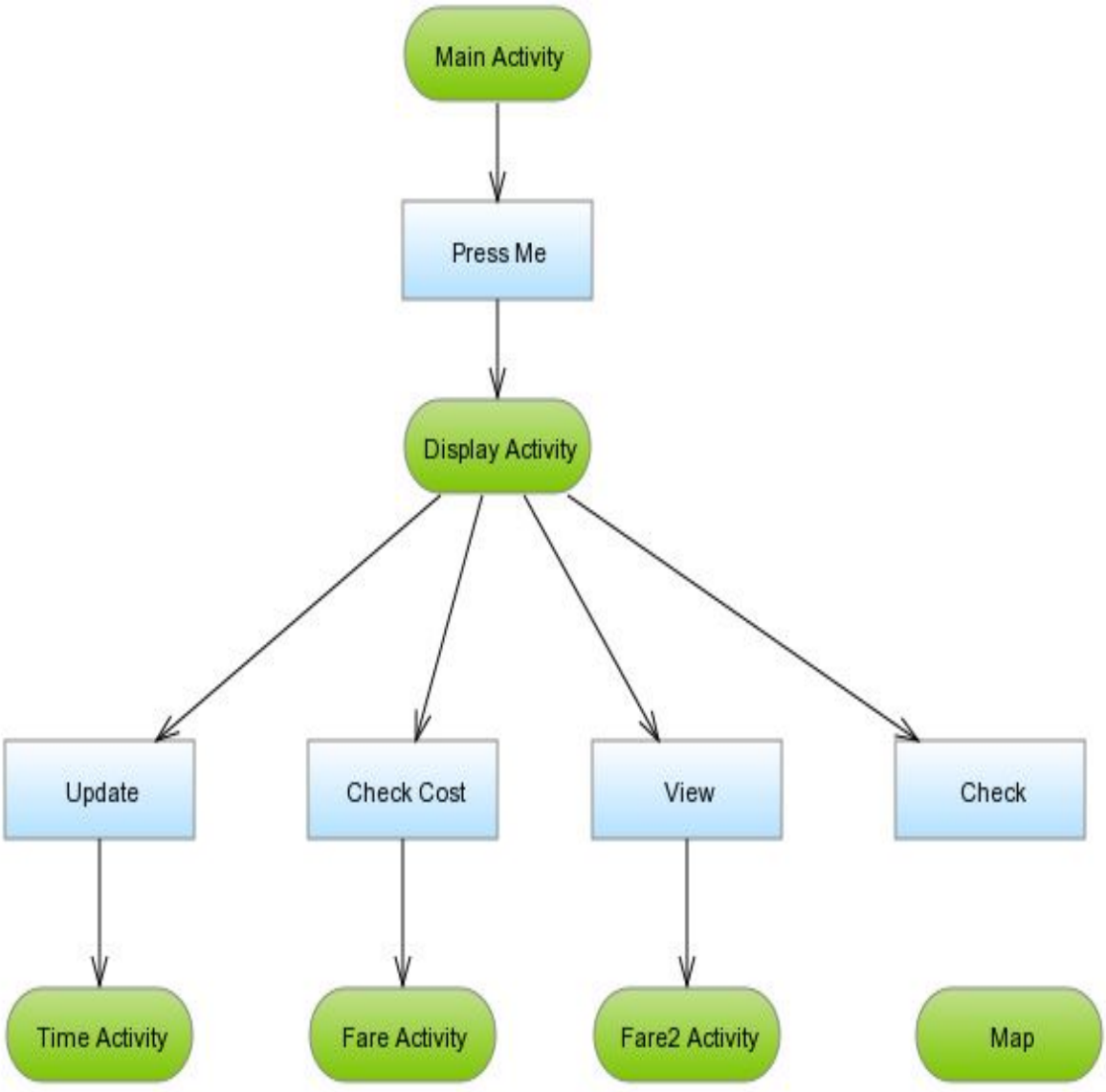
Fare2 Activity:Displays call data based on configurations.

Map: Displays Image request based on configurations.

Constraints

There won’t be any constraints on completing this project. It will be completed on time.

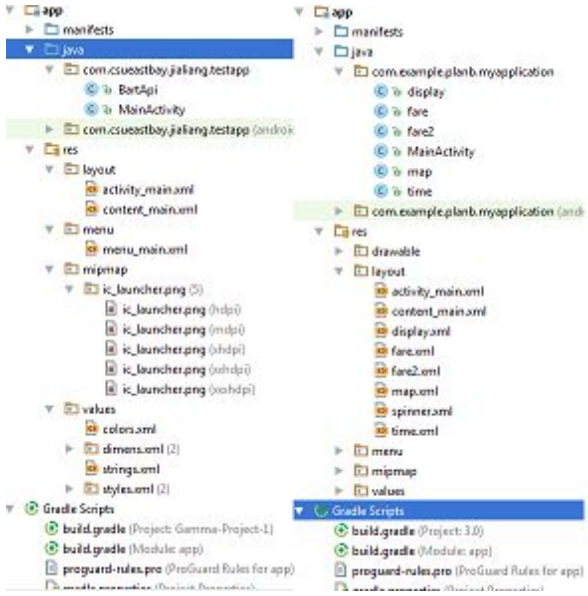
Data Flow Diagram



Uses/Interactions

The interactions between the classes are defined in the class diagrams drawn below.

Resources



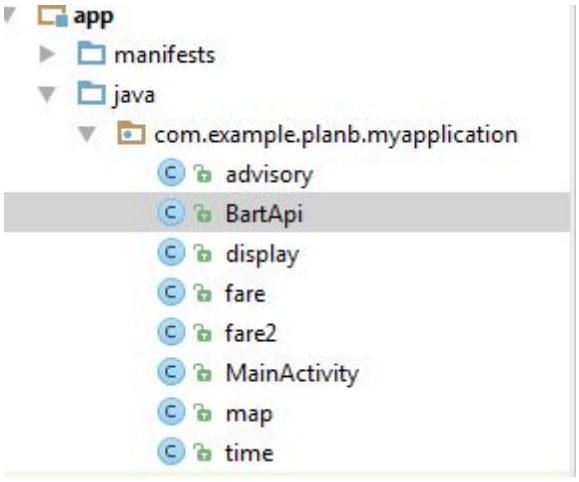
These two apps will be combined to create our almost finished product of the BART Ontime Android application. This is a user, API tool. Most functions will be running based on data provide by the API call, and managed by the controller. Shown in the view. Above is the directory for the two apps which will be combined into one.

Error Processing

Handling of exceptional conditions is done in each module. All the scenarios that can cause errors will be handled.

7.2 System Classes

7.2.1 ontime system classes



This is our java class folder. The java classes contain all the functions which correspond to the XML pages displayed by our app. Descriptions of all classes and functions inside of these classes are provided in section 4.

7.2.2 Main Activity system class

oncreate - is a function that runs when a new java activty is running or created. This function can also join other classes by extending an activity to them.

setContentView - this function passes xml files to the device screen. R.layout.activity_main is java class that also contains an xml file.

onbuttonclick - is a click listener, View v is a number that show if a button was pressed or not. We compare the value and start running the corresponding activity.

Display Class

oncreate - is a function that runs when a new java activty is running or created. This function can also join other classes by extending an activity to them.

setContentView - this function passes xml files to the device screen. R.layout.display is java class that also contains an xml file.

displaymap - is a click listener, View v is a number that show if a button was pressed or not. We compare the value and start running the corresponding activity.

testbutton1 - is a click listener, View v is a number that show if a button was pressed or not. We compare the value and start running the corresponding activity.

fare - is a click listener, View v is a number that show if a button was pressed or not. We compare the value and start running the corresponding activity.

time - is a click listener, View v is a number that show if a button was pressed or not. We compare the value and start running the corresponding activity.

Time Class

oncreate - is a function that runs when a new java activty is running or created. This function can also join other classes by extending an activity to them.

setContentView - this function passes xml files to the device screen. R.layout.time is java class that also contains an xml file.

Fare Class

oncreate - is a function that runs when a new java activty is running or created. This function can also join other classes by extending an activity to them.

setContentView - this function passes xml files to the device screen. R.layout.fare is java class that also contains an xml file.

Fare2 Class:

oncreate - is a function that runs when a new java activity is running or created. This function can also join other classes by extending an activity to them.

setContentView - this function passes xml files to the device screen. R.layout.fare is java class that also contains an xml file.

Map Class

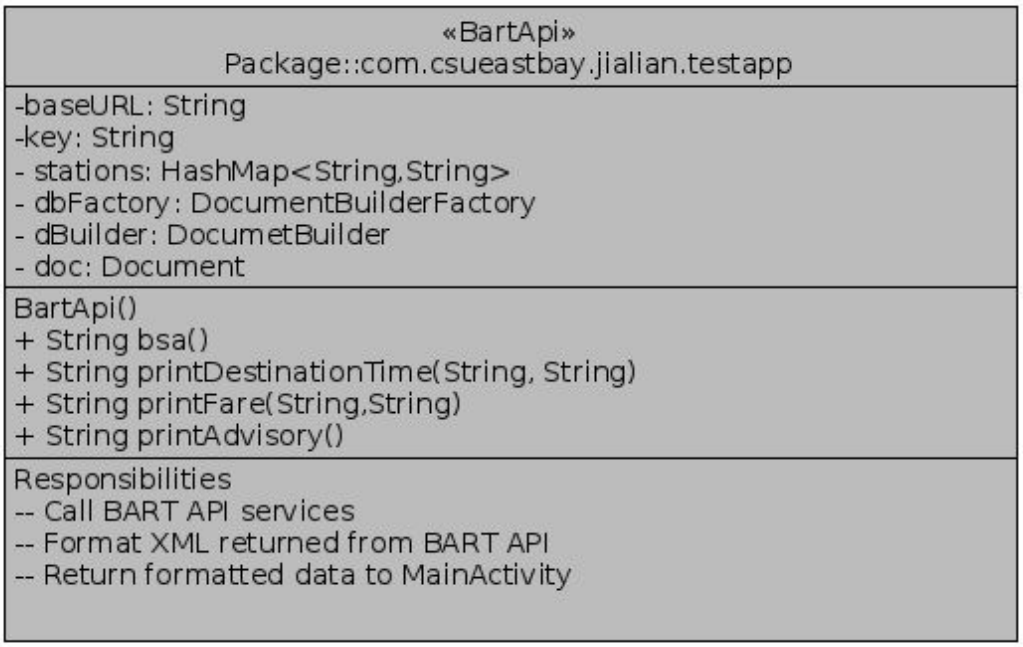
oncreate - is a function that runs when a new java activty is running or created. This function can also join other classes by extending an activity to them.

setContentView - this function passes xml files to the device screen. R.layout.map is java class that also contains an xml file.

7.2.3 BART API Services Library Classes

BART API Services Library classes are extracted from the original source code where the web services were created.

BartApi Class



7.2.4 Software Classes

Software classes are also extracted from the source code created. As there are too many properties involved and image can not fit in an A4 size document, use the link below to access the class diagrams folders for classes and all the other diagrams in this SDS document.

https://drive.google.com/open?id=0By9OP-J8mRR8UmQtVERVcFJPODg

(above link is location of team artifacts, including docs, images, additional documentation and group data. Professor Stalica has been provided access.)

Glossary

SRS: Software Requirements Specification

SDS: Software Design Specification

DFD: Data Flow Diagram

GUI: Graphical User Interface

XML: extensible markup language

BART: Bay Area Rapid Transit