Detailed Design Document

for

CSU Campus Assistant

prepared by

Group 2

Kevin Dowling

Taylor Ehrhart

Corey Greer

Drew Polhamus

04/30/15

v3.0

# 

# Abstract

This is the Detailed Design Document for the Columbus State University Campus Assistant. The information contained in this document defines the design using software requirements and the model defined in the System Requirements Document and the Architectural Design Document (ADD). This design document complies with the Detailed Design Document (DDD) from the IEEE Standard.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Table of Contents

1. Introduction

1.1 Purpose

1.2 Scope

1.3 List of terms used

1.3.1 Definitions

1.3.2 Abbreviations

1.4 Documents

1.4.1 Documents Applicable

1.5 Overview

2. Standards

2.1 Design Standards

3. Components

3.1 Menu

3.2 Map

3.3 Facilities

3.4 Facility

3.5 Get Me There

3.6 Web Service

3.7 Database

3.8 Info

Appendices

A Source Code listings

B Testing

C Development Environment

# Section 1

# Introduction

The CSU Campus Assistant will be composed of six subsystems, each of which will play a role in demonstrating the services offered by the application. In this design document, the features themselves will be defined, as well as the relationships between the different subsystems. This is important, as the different subsystems will share information amongst one another, in order to provide full functionality to the user.

1.1 Purpose

The purpose of the Detailed Design Document is to define the detailed design for all components (features) of the CSU Campus Assistant, which are specified in the System Requirements Document. The need for this definition, is so that all members of the development team can have a consistent reference to use when any questions or concerns arise. This prevents team members from straying off of the schedule, as well as the requirements for the application.

1.2 Scope

This document will cover terms used within the document itself, how each feature will be designed, and how each feature will interact with the other features. Each feature covered in this document will include a picture of what the feature should look like upon completion.

1.3 List of terms used

**1.3.1 Definitions**

|  |  |
| --- | --- |
| Terms | Definitions |
| Agent | Tool used to retrieve and execute jobs |
| Application | A non-interactive data processing application consisting of executables,  scripts and/or auxiliary data files that reads one or more input data files  and writes one or more output files |
| Client | Application that is used by all the users except the developer who  uses the agent application |
| Customer | CSU students and faculty |
| Job | Specification of application, configuration data, input and/or output data  files and scheduler specific data (priority, preferred resource, etc). |
| Project | a collection of jobs with specified access rights to which users (project  members) can be assigned |
| Resource Providers | users that offer time on their devices to the Campus Assistant. They can write reviews, suggest changes and give general feedback |
| Role | the actions and activities assigned to a person. |
| Campus Assistant | a tool that helps user navigate the main campus and informs them of facilities, resources, and services |
| Campus Assistant Software | software developed by Group Two to be distributed for informative purposes |

**1.3.2 Abbreviations**

|  |  |
| --- | --- |
| **Abbreviations** | **Terms** |
| ADD | Architectural Design Document |
| IEEE | Institute of Electrical and Electronics Engineers |
| DDD | Detailed Design Document |
| ART | Android Runtime |
| SPP | Software Project Plan |
| SRD | Software Requirements Document |
| UML | Unified Modeling Language |

1.4 Documents

**1.4.1 Documents Applicable**

|  |  |
| --- | --- |
| **[SRD]** | **Software Requirements Document, Group Two, version 2.0, 23 February 2015** |
| **[SPP]** | **Software Project Plan, Group Two, version 1.0, 3 March 2015** |
| **[SUM]** | **Software User Manual, Group Two, TBD** |

1.5 Overview

This design document is organized as follows:

# Section 1 (which you are currently in) contains introductory information that is needed in order to follow the document such as key terms, and a table of contents.

# Section 2 describes the standards this project follows.

# Section 3 describes each of the features of the application in a consistent way.

# Section 4 contains a reference to where the build procedure of the Campus Assistant can be found.

# Section 5 lists the files in which the components reside.

# Section 6 contains test plans for each feature, as well as the traceability matrix used in testing.

# 

# 

# 

# 

# 

# 

# 

# 

# 

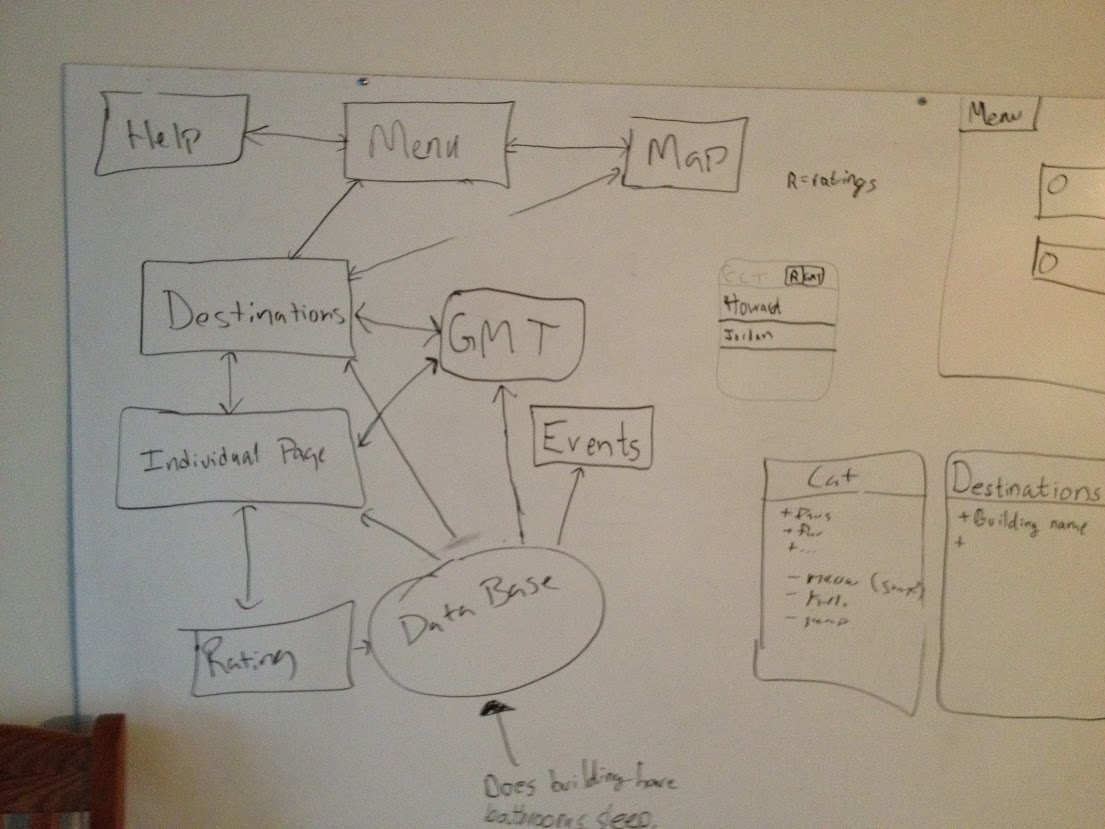
# 

# Section 2

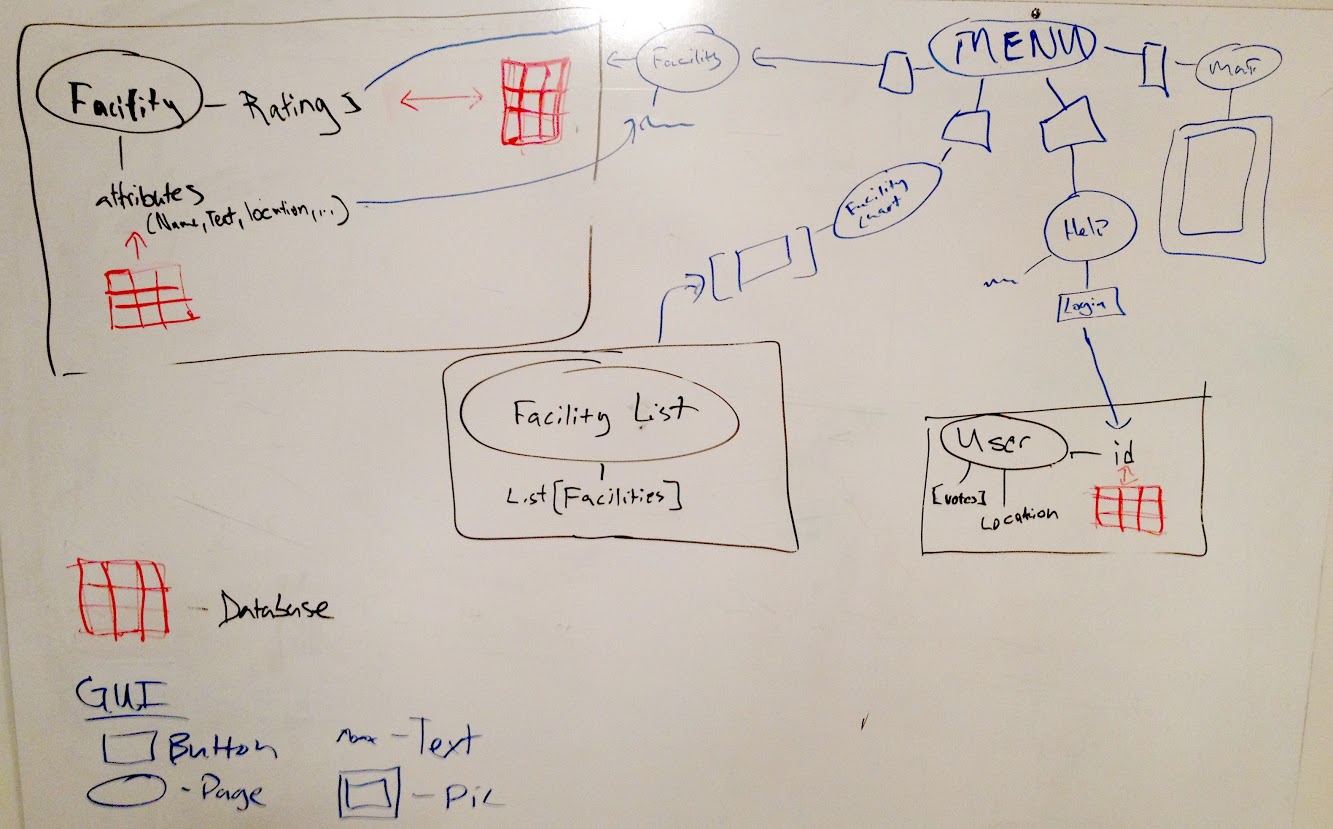
# Standards

2.1 Design standards

The Campus Assistant will be powered by 4 classes; User, Facility, Facility List, and Amenity. Each class has it’s own attributes, and methods that the classes will use to perform it’s activities. Unified Modeling Language is used to show the design of each feature.



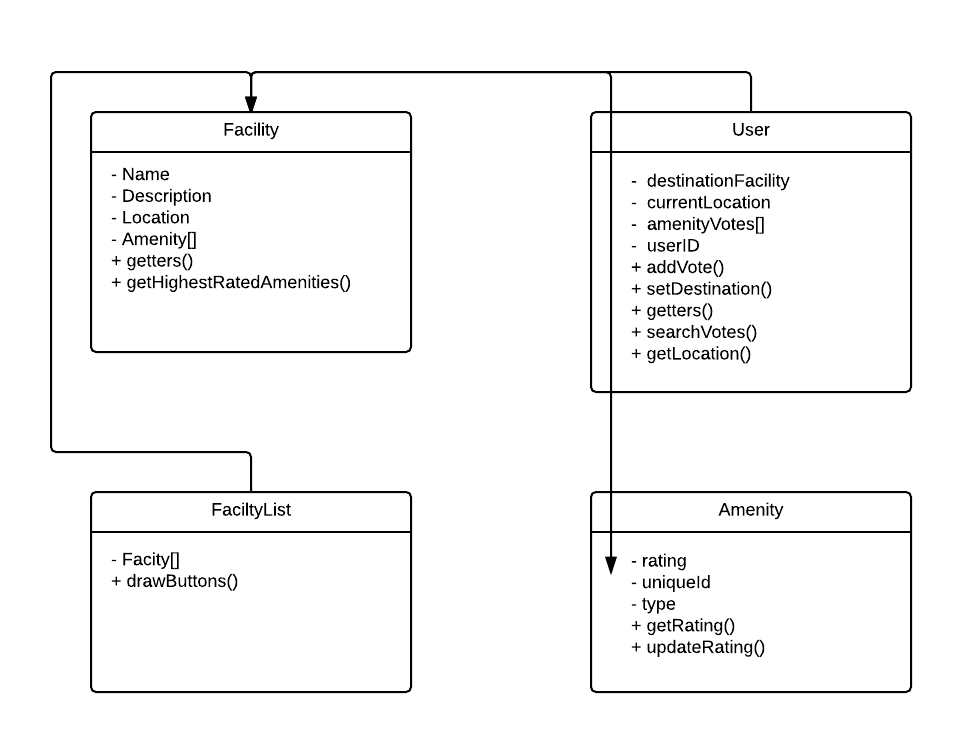
*First draft of UML that illustrates how data will be shared between the different features.*.



*A diagram, of the classes, that the Campus Assistant will need in order to present the components, as well as show how they will interact with the GUI.*

**Classes**

1. **Facility**
2. Attributes
   1. Name - String that represents the name of the facility
   2. Description - String that is a description about the building including things that make the building unique
   3. Location - This variable will store the facility’s GPS coordinates
   4. Amenity - An array that will hold all of the amenities for the particular facility. Each amenity element will have a rating associated with it that it will read from the database and will be updated with user input.
3. Methods
   1. getters for each attribute
   2. getHighestRatedAmenites - this method will retrieve the highest rated amenities for the particular facility so that they can be displayed on the Facility List page.
4. **User**
   1. Attributes
      1. destination - this String variable will hold the identity of the user’s destination.
      2. currentLocation - this variable will represent the user’s current GPS coordinates
      3. amenityVotes - this array will hold all of the votes cast by the user.
      4. userID - the user ID is a String that the app will use in it’s GUI to identify the person using the app
      5. loggedOn - boolean value that is set to true if the user has submitted their Novell login (tentative, need to research access).
   2. Methods
      1. addVote - This method will have to access the database and will send either +1 or -1 to the votes on a specific facility amenity.
      2. setDestination - This will set the string value of the User destination
      3. getters -
         1. getDestination - returns the User’s destination
         2. getCurrentLocation - returns the User’s destination
         3. getUserID - returns the User’s id
      4. searchVotes - takes in an Amenity id as the parameter and searches through the User’s amentyVotes array list for the id given. Returns a boolean true if the User has voted on the amenity given.
5. **Facility List**
   1. Attributes
      1. facility[]: An array of Facilities organized alphabetically by name.
   2. Methods
      1. drawButtons: sends the information needed for the GUI index by index.
6. **Amenity**
   1. Attributes
      1. uniqueID - int that is unique for each Amenity in the program. Used solely for the User votes.
      2. rating - this is an integer that is used to determine the color of the rating GUI color (red, yellow, green)
      3. type - an enumerator type that determines the icon in the rating GUI
   2. Methods
      1. Getters:
         1. getUniqueID: returns the uniqueID
         2. getRating: gets and returns the rating from the database
         3. getType: returns the type of the Amenity.



# 

# 

# 

# 

# 

# 

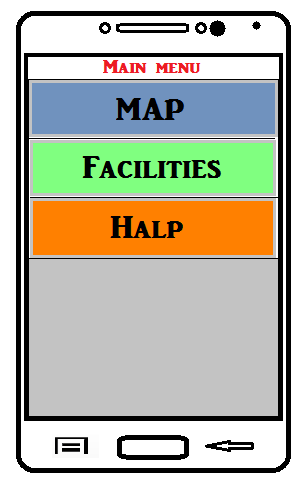
# 

# Section 3

# Components

## 3.1. Menu

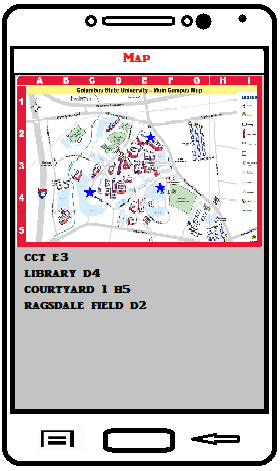
The App’s Main Menu will be comprised of 3 *buttons* that will navigate the *customer* to the individual features. The buttons on the menu will be; **Map**, **Facilities**,and **Help**. At the very top of every page on the app will be a bar that states what page you are on. This includes all feature pages, as well as the main menu.



*The Main Menu*

**3.2. Map**

The Map feature on the Campus Assistant will be very similar to the one found on Columbus State’s [website](http://www.columbusstate.edu/smokefree/images/designatedareas.jpg). Notice the grid layout, and key (located below the map) that will provide customers with the ability to locate their facility of choice and then locate it quickly on the map. The grid system allows areas of the map to be defined by a combination letter and number identifier.



*The Map*

**3.3. Facilities**

The Facilities page will simply be a list of clickable rows, with each row displaying the name of an individual facility. Upon clicking the row, the customer will be taken to that individual facility’s page, where they can learn more about the facility.

**3.4. Individual Facilities**

Each facility will have it’s own individual page that gives the customer all of the information about that facility.It is here that user will be able to view the ratings of the facility’s amenities, as well as rate it themselves if they login with a valid CSU email address. This is also where the customer can access the Get Me There feature via a green button in the top right corner.



**3.5 Get Me There**

The app will be able to give the users directions around campus as the user is walking around in the real world. It will give a compass direction and distance to the place that the user has chosen to get directions to. It will overlay this information on a camera feed.

**3.6 Info**

The “Info” page, is where users will find instructions for using the Campus Assistant. The contents of this page will be as follows:

Main Menu - To navigate, tap your choice of the features.

Facility List - To select a facility, tap your choice of available facilities.

Facility - This page displays the facility name, and has a space for a description provided by the database. In the bottom right corner, you will see the “GMT” button. This is how you access Get Me There.

Get Me There - This feature, currently just a camera feed, is still under development.

Info - Information about how to use this app.

To navigate to a previous page, tap the ‘back’ button on your Android device.

**3.7 Database**

The CSU Campus Assistant will require a database to store all of the data that is available for the application to display. The relational database that the application will use is powered by Amazon Web Services, and is available for free on a trial basis. The trial will extend beyond the range of Spring 15, which is ample time to present this project. Data integrity will be provided in the form of a security group, which is provided by AWS as well. The security group will only allow access from a specified range of IP addresses, and can be controlled by the database administrator.

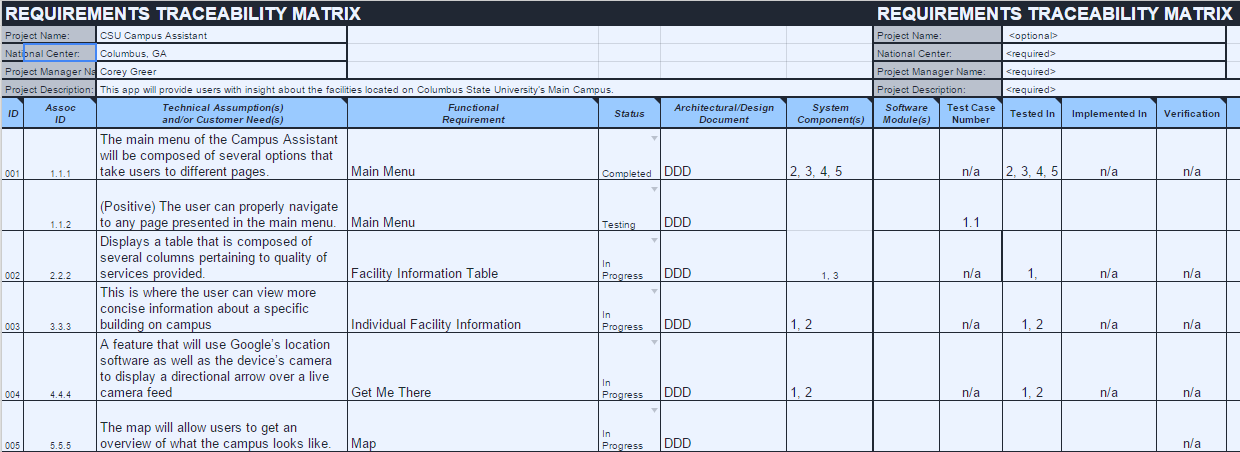
For the database client, the development team will use MS SQL Server, which is free to students through Dreamspark. MS SQL Server is in the list of clients available for an AWS RDS, and will provide all of the features necessary to build the database for this application.

# Appendix A - Source Code

The source code for this application can be found on this same storage device, under the file named “CSUCampusAssistant2”. It is an Android Studio project, and must be opened in that IDE.

Appendix B - Unt and System Testing

|  |  |  |
| --- | --- | --- |
| Functional Requirement ID | Pass Criteria | Fail Criteria |
| 1.1.1 | This requirement allows users to navigate to a variety of pages in the application. If the user can successfully navigate to any page listed on the main menu, then the function will pass. | The requirement will meet fail criteria if the user cannot navigate to pages listed in the main menu. This includes incorrect linking of pages, pages not loading, and/or the process freezing upon navigating. |
| 2.2.2 | This requirement displays a general description of services available to facilities. If the correct content is displayed for each corresponding facility, then the function will pass. | The requirement will meet fail criteria if the correct content is not displayed. This can be a content in error, or absence of content in part or all together. |
| 3.3.3 | This requirement provides more concise facility information. If the correct content is displayed for each corresponding facility, then the function will pass. | The requirement will meet fail criteria if the correct content is not displayed. This can be a content in error, or absence of content in part or all together. |
| 4.4.4 | This requirement allows users to determine a destination on main campus and guide them in the proper direction using their camera feed and a basic overlay. If the function sets destinations correctly and provides accurate orientation, then the function will pass. | This requirement will meet fail criteria if an unexpected destination is set and/or the directional guide malfunctions. Malfunctions include skewed orientation, camera/overlay failure, or a wrong preset destination or no destination specified at all. Additionally failure will occur if the process freezes or crashes the application. |
| 5.5.5 | This requirement offers user a map of the main campus that is lightly interactive. The function will pass if the map displays and zoom functions are capable. | This requirement will meet fail criteria if the map is not displayed at all, zoom functions do not properly execute, or if the process freezes or crashes the application. |

Traceability Matrix

Appendix C - Development Environment

Android Studio is the environment that has been chosen to develop this application with.