# Abstract

In this project we build an application that is a combination of a personalized assistant and an agenda namely, a course manager and a scheduler as well as a localized positioning system to assist in navigating Brock University campus.

# Table of Contents

Contents

[1 Abstract 1](#_Toc353467522)

[2 Table of Contents 1](#_Toc353467523)

[3 Glossary 1](#_Toc353467524)

[4 Introduction 2](#_Toc353467525)

[5 Technical Details 2](#_Toc353467526)

[5.1 Overview 2](#_Toc353467527)

[5.2 Implementation 3](#_Toc353467528)

[5.2.1 Scheduler 3](#_Toc353467529)

[5.2.2 CourseManager 5](#_Toc353467530)

[5.2.3 Tour 18](#_Toc353467531)

[5.2.4 Maps 21](#_Toc353467532)

[5.2.5 Help 27](#_Toc353467533)

[5.3 Miscellaneous 27](#_Toc353467534)

[5.4 Management 28](#_Toc353467535)

[5.5 Acknowledgements 28](#_Toc353467536)

[5.6 Bibliography 29](#_Toc353467537)

# Glossary

OOP –Object Oriented Programming

Manifest File- AndroidManifest.xml

SDK – Standard Development Toolkit

ADT- Android Development ToolKit

# Introduction

BrockButler is being developed to ease the life of Brock Students. For a freshman student or perhaps a returning student, organizing their courses and managing their time can be a daunting task. Locating class rooms, organizing courses and assignments associated with those, remembering weekly schedules can be quite stressful especially under a heavy workload. BrockButler will provide a software solution to this problem by utilizing a combination of school navigation, agenda, and memos to be an all in one program for a student to plan out their school activities; with an added functionality to navigate a student to a desired location within the school or do a virtual tour of the school like Google Street View. This application was designed for the Android operating system and will keep a student’s class, and assignment schedule and place reminders and convenient times to ensure the student will be able to appropriate time responsibly.

# Technical Details

## Overview

BrockButler was designed for the Android platform to work with a minSdkVersion 14 and minSdkVersion 16 this is specified in the manifest file.

The System UI has 6 activity files which are MainActivity, TourActivity, HelpActivity, MapsActivity, CourseManagerActivity, AddContactsActivity all the activities associated with one of the five components are included with it.

The system is divided into five components by functionality as well as the user interface associated with each component:

1. *Scheduler*: Tasks can be viewed, added, modified or removed from within the scheduler

2. *Course Manager:*  Courses can be viewed, added, modified or removed from within the course manager

3. *Maps:* Displays map of Brock University with all rooms, location in detail, enables you to set the destination and finds the optimum route to the destination

4. *Tour:* Gives a picture view of the interior of the buildings enabling navigation by providing navigation arrows.

5. *Help:* Displays a help *with* overview, feature list and help topics in addition each of the other modules can launch the help functionality

## Implementation

### Scheduler

#### Description

The scheduler consists of 3 classes to accomplish its functionality; they are SchedulerActivity, AddTaskActivity and Task, they along with the database accomplish the task of maintaining schedules.These 3 classes are included in the package edu.seaaddicts.brockbutler.scheduler

#### Class SchedulerActivity

public class SchedulerActivity extends Activity

SchedulerActivity class populates the list view with the appropriate schedules and interacting with the user and responds to the different user inputs.

|  |  |
| --- | --- |
| **Field Summary** | |
| Private static final int | *VISIBLE is used to set the visibility on a textView this is initially set to 0* |
| Private static final int | GONE *is used to set the visibility on a textView to hide this is initially set to 8* |
| Private CourseHandler | mCourseHandle *points to the current course handler* |
| Private ListView | mRegisteredCoursesListView *is the listView for displaying the schedule* |
| Private ArrayList<Course> | mRegisteredCoursesList *is an ArrayList of courses that has been registered for* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | populateCoursesLayout() *populates the listView with all the courses* |
| Void | onCreate(Bundle savedInstance*) is a method that calls the populateCoursesLayout method and also initializes a coursehandle* |
| Void | showHideToolBar(View view, int position) *is a method that starts the animation on expanding the animation for the item and adds the offerings registered* |
| Void | showHelp() *is a method that shows help related to scheduler* |
| Boolean | onCreateOptionsMenu(Menu menu) *method is invoked when the menu is pressed* |
| Void | sendEmail() *sends an email to the appropriate client as specified* |

#### 6.2.1.4 Class AddTaskActivity

public class AddTaskActivity extends Activity

This class is responsible for creating a UI for the user to add a task and responds to the various user interaction .

|  |  |
| --- | --- |
| **Field Summary** | |
| Private static final int | DATE\_DIALOG\_*ID is set to 0 which is used as a helper field for the DatePickerDialog* |
| Private static final String | TAG *is set to AddTaskActivity initially and is used in debugging* |
| Private Button | mDueDateButton *button used to invoke the due date functionality* |
| Private Button | mSaveButton *is the button used to save your data* |
| Private Button | mCancelButton *is used to cancel your current task* |
| Private ArrayList<Courses> | mRegCourses *keeps all the registered courses* |
| Private TextView | mDueDateTextView *is used to display the due date* |
| Private EditText | mTaskTitle *is used to display the title* |
| Private EditText | mTaskWeight *is used to display the weight* |
| Private EditText | mTaskBase *is used to display the base mark* |
| Private EditText | mTaskMark *is used to display the task mark* |
| Private Spinner | mCourseSpinner *is used select one course value from a set* |
| Private Spinner | mPrioritySpinner *is used to select the priority value from a set* |
| Private CourseHandler | mCourseHandle *is used to access the functionality of the course manager* |
| private int | mYear *holds the year* |
| Private int | mMonth *holds the month* |
| Private int | mDay *holds the day* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | Init() *is the method that initializes all the views and sets the Button’s onClickListener* |
| Void | onCreate(Bundle savedInstance*) sets the contentView and calls the above method* |
| Void | onCreateDialog(int) *displays the date time picker on click* |

#### 6.2.1.5 Class Task

This class encapsulates a task

|  |  |
| --- | --- |
| **Field Summary** | |
| Publis String | mSubj *to hold the subject* |
| String | mCode *is to hold the course code* |
| Boolean | mIsPastDue *a Boolean to indicate the status of the task* |
| Float | mMark *to hold a mark for the task* |
| Float | mBase *to hold a base mark* |
| Float | mWeight *to hold the weight of the task* |
| Int | mAssign *to hold the assignment number* |
| String | mName *to hold the name of the task* |
| String | mCreationDate *to hold the creation date* |
| String | mDueDate *to hold the due date* |
| Int | mPriority *to hold the priority of the task* |

|  |  |
| --- | --- |
| **Constructor** | |
| Void | Task() *creates a new ArrayList of contacts* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Boolean | isPastDueDate() *is used to check whether the task is past due date returns a boolean* |

### CourseManager

#### Description

The CourseManager consists of 11 classes to accomplish its functionality; they are AddCourseActivity, Brocku, Course, CourseHandler, CourseListHandler, CourseManagerActivity, CurrentCoursesHandler, MasterCourse, ModifyCourseActivity, Offerings, OfferingTime this along with the database accomplish the task of maintaining the CourseManager. These 11 classes are included in the package edu.seaaddicts.brockbutler.coursemanager

#### Class AddCourseActivity

public class AddCourseActivity extends Activity

This class encapsulates the responsibility of populating the respective views with all the courses, their offerings and all the information related to them and it also allows for interactivity with the user by responding to different events that could be trigger by the user interaction with the application interface.

|  |  |
| --- | --- |
| **Field Summary** | |
| Private static final int | DATE\_DIALOG\_*ID is set to 0 which is used as a helper field for the DatePickerDialog* |
| Private static final String | TAG *is set to AddTaskActivity initially and is used in debugging* |
| Private static final int | VISIBLE *is set to 0 initially and it used to turn the visibility on* |
| Private static final int | INVISIBLE *is initially set to 4* |
| Private static final int | GONE *is used to set the visibility on a textView to hide this is initially set to 8* |
| ArrayList<String> | mLecs *is used to hold all the lectures associated with the course* |
| ArrayList<String> | mLabs *is used to hold all the labs associated with the course* |
| ArrayList<String> | mTuts *is used to hold all the tutorials associated with the course* |
| ArrayList<String> | mSems *is used to hold all the semesters associated with the course* |
| ArrayList<Offering> | mLecsOfferings *is used to hold all the lecture offerings* |
| ArrayList<Offering> | mTutsOfferings *is used to hold all the tutorial offerings* |
| ArrayList<Offering> | mSemsOfferings *is used to hold all the semester offerings* |
| Private String | mSubject *to store the subject* |
| Private String | mCode *to store the course code* |
| Private Course | mCourse *to store the course name* |
| Private Button | mSaveButton *is the button used to save your data* |
| Private Button | mCancelButton *is used to cancel your current task* |
| Private ArrayList<Offering> | mOfferings *keeps all the registered courses* |
| Private CourseHandler | mCourseHandle *is used to access the functionality of the course manager* |
| Private Spinner | mSubjectSpinner *is used select one subject value from a set* |
| Private Spinner | mCodesSpinner *is used to select the code value from a set* |
| Private ListView | mLecsListView *to display the lectures* |
| Private ListView | mSemsListView *to display the semesters* |
| Private ListView | mTutsListView *to display the tutorials* |
| Private ListView | mLabsListView *to display labs* |
| private int | mYear *holds the year* |
| Private int | mMonth *holds the month* |
| Private int | mDay *holds the day* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | Init() *is the method that initializes all the views and sets the Button’s onClickListener* |
| Void | onCreate(Bundle savedInstance) *sets the contentView and calls the above method* |

#### Class Brocku

public class Brocku extends AsyncTask<Void, Void, ArrayList<MasterCourse>>

This class uses AsyncTask to do operations off the main thread it goes to the brock registrar’s url and obtains the latest timetable to process that information to be shown in the application

|  |  |
| --- | --- |
| **Method Summary** | |
| Protected ArrayList<MasterCourse> | doInBackground(void…void*) this returns an ArrayList of courses after all the information has been gathered and processed* |

#### Class Course

public class Course

This class encapsulates a course

|  |  |
| --- | --- |
| **Field Summary** | |
| Public int | mId *A course has an Id* |
| Public String | mSubject *a course has a subject* |
| Public String | mCode *a course has a code* |
| Public String | mDesc *a course has a description* |
| Public String | mInstructor *stores the instructor information* |
| public ArrayList<Offering> | mOfferings *stores the offering information* |
| public String | mInstructor\_email *stores the instructor email* |
| public ArrayList<Task> | mTasks *stores the tasks* |
| public ArrayList<Contact> | mContacts *stores the contacts* |

|  |  |
| --- | --- |
| **Constructor** | |
| Public | Course() *creates a new offerings, tasks and contacts ArrayList to begin processing* |

#### Class CourseHandler

public class CourseHandler

This class provides all the functionality associated with a course

|  |  |
| --- | --- |
| **Field Summary** | |
| CurrentCoursesHandler | CH *provides the course context* |
| CourseListHandler | courseList *provides the courseList context* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | getAllCourses() *gets the data from the registrars timetable and inserts data into the masterlist table* |
| Course | getCourse(final String subj, final String code*) gets all information for a given course subject and code* |
| Void | updateCourse(Course course) *updates all the information for a given course* |
| ArrayList<String> | getSubjects()throws Exception  *gets a list of subjects available from the master list* |
| ArrayList<String> | getCodes(String subj) *gets a list of codes for a given subject from the master list* |
| Course | getCourseOfferings(String subj, String code) *returns all offerings offered for a given course* |
| Int | addCourse(Course course) throws Exception *adds information for a course into the database* |
| Int | removeCourse(Course course) *deletes all information from the database for a course* |
| ArrayList<Course> | getRegisteredCourses()*returns all information for all courses in the*  *current courses database* |
| ArrayList<Offering> | getOfferings(String subj, String code) *get all offerings for a certain course* |
| ArrayList<Task> | getTasks() *gets all tasks from the database* |
| Int | addTask(Task task) *adds a given task to the task table in the database* |
| Int | addTask(Course course) *adds the tasks for a given course to the task table in the database* |
| Int | removeTask(Task task) *deletes task information from the database for a given task* |
| Float | getMark(Course course) *returns the calculated progress mark for a course* |
| int | getSize() *gets the number of courses in the courseList* |
| void | removeContact(Contact contact) *removes a specified contact* |
| Cursor | Query(String query) *returns a cursor with results for a custom query* |

|  |  |
| --- | --- |
| **Constructor** | |
| Public | CourseHandler**(**Context context**)** *creates a new context and also creates a database and opens it* |

#### 6.2.2.6 Class CourseListHandler

public class CourseListHandler extends SQLiteOpenHelper

This class provides all the information associated with the different courses it also performs database housekeeping by creating a database table for a full list of offerings on the registrar's timetable and allows the table to have inserts or be read

|  |  |
| --- | --- |
| **Field Summary** | |
| private static final int | DATABASE\_VERSION *A course has an Id has value 1* |
| private static final String | DATABASE\_NAME *The database name* |
| private static String | DB\_PATH *holds the database path initially it is* /data/data/edu.seaddicts.brockbutler.cousemanager/databases |
| private SQLiteDatabase | myDataBase *holds the databse context* |
| private static final String | TABLE\_MCOURSES *refers to the master list* |
| private static final String | KEY\_DAYS *refers to the days* |
| private static final String | KEY\_TIME *refers to the time* |
| private static final String | TABLE\_COURSES *refers to the courses* |
| private static final String | TABLE\_TASKS *refers to the tasks* |
| private static final String | TABLE\_OFFERINGS refers to the offerings |
| private static final String | TABLE\_OFFERING\_TIMES *refers to the offering times* |
| private static final String | TABLE\_CONTACTS *refers to the contacts* |
| private static final String | KEY\_SUBJ *refers to the subj* |
| private static final String | KEY\_CODE *refers to the code* |
| private static final String | KEY\_DESC *refers to the description* |
| private static final String | KEY\_INSTRUCTOR *refers to the instructor* |
| private static final String | KEY\_ID *refers to the id* |
| private static final String | KEY\_TYPE *refers to the type* |
| private static final String | KEY\_SEC *refers to the section* |
| private static final String | KEY\_DAY *refers to the day* |
| private static final String | KEY\_TIMES *refers to the start time* |
| private static final String | KEY\_TIMEE *refers to the end time* |
| private static final String | KEY\_LOCATION *refers to the location* |
| private static final String | KEY\_DUR *refers to the duration* |
| private static final String | KEY\_ASSIGN *refers to the assignment* |
| private static final String | KEY\_NAME *refers to the name* |
| private static final String | KEY\_MARK *refers to the mark* |
| private static final String | KEY\_BASE *refers to the base mark* |
| private static final String | KEY\_WEIGHT *refers to the weight* |
| private static final String | KEY\_DUE *refers to the due date* |
| private static final String | KEY\_CREATE\_DATE *refers to the creation date* |
| private static final String | KEY\_CID *refers to the cid* |
| private static final String | KEY\_FNAME *refers to the first name* |
| private static final String | KEY\_LNAME *refers to the last name* |
| private static final String | KEY\_EMAIL *refers to the email* |
| private static final String | KEY\_PRIORITY *refers to the priority* |
| private static final String | KEY\_INSTREMAIL *refers to the instructors email* |
| Context | context |

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | onCreate(SQLiteDatabase db) *creates the courses table for the master list of courses* |
| Void | createDataBase() throws IOException *create a database into the default system path of the application* |
| Boolean | checkDataBase() *check if the database has been opened for reading and returns a Boolean* |
| void | copyDataBase() throws IOException *copies the contents of one database into another* |
| Void | openDataBase() throws SQLException establishes a connection with the *the database* |
| Void | onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) *upgrading the database will drop the table and recreate* |
| Void | addCourse() *adds information for a course into the database* |
| ArrayList<MasterCourse> | getCourses(String subj, String code) *returns a list of offerings for a particular subject and code* |
| ArrayList<String> | getSubjects()*returns a list of subjects from the database* |
| ArrayList<String> | getCodes(String subj) *returns a list of codes for a subject from the database* |
| int | size() *gets the number of courses* |

|  |  |
| --- | --- |
| **Constructor** | |
| Public | CourseListHandler(Context context) *creates a new context and also gets the database absolute path* |

#### 6.2.2.7 Class CourseManagerActivity

public class CourseManagerActivity extends Activity

CourseManagerActivity is responsible for providing a UI for the course manager it also accepts different user inputs and responds to them

|  |  |
| --- | --- |
| **Field Summary** | |
| public static final int | CODE\_COURSE\_MODIFIED  *is set to 0 to indicate that the course code has been modified* |
| public static final int | CODE\_COURSE\_UNMODIFIED *is set to 1 to indicate that the course code has been unmodified* |
| public static final int | CODE\_ADD\_COURSE *is set to 2 to indicate that the course code has been added* |
| public static final String | CODE\_COURSE\_SUBJECT *refers to the course subject* |
| public static final String | CODE\_COURSE\_CODE *refers to the course code* |
| public static final String | CODE\_COURSE\_DESC *refers to the course description* |
| public static final String | CODE\_COURSE\_INSTRUCTOR *refers to the course instructor* |
| public static final String | CODE\_COURSE\_INSTRUCTOR\_EMAIL *refers to the course instructor email* |
| public static final String | CODE\_COURSE\_OFFERINGS *refers to the course offerings* |
| private static final String | TAG *has the CourseManagerActivity as its value and is used in debugging* |
| private static final int | *VISIBLE is used to set the visibility on a textView this is initially set to 0* |
| private static final int | GONE *is used to set the visibility on a textView to hide this is initially set to 8* |
| private ArrayList<Course> | mRegisteredCoursesList *is an arrayList of registered courses* |
| private CourseHandler | mCourseHandle *points to the current course handler* |
| private ListView | mRegisteredCoursesListView *is the listView for displaying the schedule* |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | onCreate(Bundle savedInstanceState) *creates the courseHandler and sets the contentView* |
| Void | onResume() *calls the method below after resuming* |
| Void | populateCoursesLayout()*Populates the ListView with registered classes and brief details, i.e.instructor name and class times.* |
| Void | showHelp(MenuItem item) *launches the help* |
| void | onCreateContextMenu(ContextMenu menu, View v,ContextMenuInfo menuInfo) *populates the menu with the appropriate menu items* |
| Boolean | onContextItemSelected(MenuItem item) *Determines which MenuItem was selected and acts appropriately depending on choice* |
| Void | onActivityResult(int requestCode, int resultCode, Intent data) *adds a course or modifies the course depending on user interaction with the interface* |
| Void | showHideToolbar(View view, int position) *Shows/hides verbose description of course.* |
| Boolean | onCreateOptionsMenu(Menu menu) *Inflate the menu; this adds items to the action bar if it is present.* |

#### 6.2.2.8 Class ****CurrentCoursesHandler****

public class CurrentCoursesHandler extends SQLiteOpenHelper

This class Handles database table creation and queries for course information

|  |  |
| --- | --- |
| **Field Summary** | |
| private static final int | DATABASE\_VERSION *is set to 1 to refer to the database version* |
| private static final String | DATABASE\_NAME *refers to the database name* |
| private static final String | TABLE\_COURSES *refers to the courses* |
| private static final String | TABLE\_TASKS *refers to the tasks* |
| private static final String | TABLE\_OFFERINGS *refers to the offerings* |
| private static final String | TABLE\_OFFERING\_TIMES *refers to the offering times* |
| private static final String | TABLE\_CONTACTS *refers to the contacts* |
| private static final String | KEY\_SUBJ *refers to the subjects* |
| private static final String | KEY\_CODE *refers to the code* |
| private static final String | KEY\_DESC *refers to the description* |
| private static final String | KEY\_INSTRUCTOR *refers to the instructor* |
| private static final String | KEY\_ID *refers to the id* |
| private static final String | KEY\_TYPE *refers to the type* |
| private static final String | KEY\_SEC *refers to the section* |
| private static final String | KEY\_DAY *refers to the day* |
| private static final String | KEY\_TIMES *refers to the start time* |
| private static final String | KEY\_TIMEE *refers to the end time* |
| private static final String | KEY\_LOCATION *refers to the location* |
| private static final String | KEY\_DUR *refers to the duration* |
| private static final String | KEY\_ASSIGN *refers to the assignments* |
| private static final String | KEY\_NAME *refers to the name* |
| private static final String | KEY\_MARK *refers to the mark* |
| private static final String | KEY\_BASE *refers to the base mark* |
| private static final String | KEY\_WEIGHT *refers to the weight* |
| private static final String | KEY\_DUE *refers to the due date* |
| private static final String | KEY\_CREATE\_DATE *refers to the creation date* |
| private static final String | KEY\_IS\_DONE *refers to whether the key is done* |
| private static final String | KEY\_CID *refers to the cid* |
| private static final String | KEY\_FNAME *refers to the first name* |
| private static final String | KEY\_LNAME *refers to the last name* |
| private static final String | KEY\_EMAIL *refers to the email* |
| private static final String | KEY\_PRIORITY *refers to the priority* |
| private static final String | KEY\_INSTREMAIL *refers to the instructor email* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) *on an upgrade drop tables and recreate* |
| Void | addCourse(Course course) *adds all information for a course to the database adding course, offerings, tasks and contacts information, if information exists* *for the course then an update is done, otherwise and insert is done* |
| Void | deleteCourse(Course course) *removes all information for the given course from the database* |
| Course | getCourse(String subj, String code) *retreives all information for the given course* |
| Void | addOfferings(Course course) *adds all offerings offered by a particular course as well as their offering times* |
| Void | deleteOffering(Offering offering) *removes all information from the databse for the given offering* |
| Void | addTasks(Course course) *adds all tasks associated with a given course* |
| Void | addContacts(ArrayList<Contact> contacts) *add contacts to the contacts table in the database for the given list of contacts* |
| void | addTasks(Task task) *adds a task for a certain course* |
| ArrayList<Offering> | getOfferings(String subj, String code) *gets all offerings for a given subject and code* |
| ArrayList<Task> | getTasks()*gets all tasks a person may have from the databas* |
| ArrayList<Task> | getTasks(Course course) *gets all tasks for a particular course* |
| ArrayList<Contact> | getContacts(Course course) *get all contacts for a specified course* |
| void | removeTask(Task task) *deletes a given task from the tasks table of the database* |
| void | removeContact(Contact contact) *deletes a given contact from the contacts table* |
| ArrayList<Course> | getRegCourses()*gets all courses added to the courses table of the*  *database and all of it's components* |
| Cursor | Query(String s) *a general method to allow a query to be done that has not been specified. it returns a cursor to allow the data to be read.* |

#### 6.2.2.9 Class ****ModifyCourseActivity****

public class ModifyCourseActivity extends Activity

This class provides the UI and functionality for course modification

|  |  |
| --- | --- |
| **Field Summary** | |
| private static final String; | TAG *refers to the ModifyCourseActivity and is used in debugging* |
| private static final int | DATE\_DIALOG\_ID *set to 0* |
| private static final int | *VISIBLE is used to set the visibility on a textView this is initially set to 0* |
| private static final int | *VISIBLE is used to set the invisibility on a textView this is initially set to 4* |
| private static final int | GONE *is used to set the visibility on a textView to hide this is initially set to 8* |
| ArrayList<String> | mLecs *refers to the lectures* |
| ArrayList<String> | mLabs *refers to the labs* |
| ArrayList<String> | mTuts *refers to the tutorials* |
| ArrayList<String> | mSems *refers to the semesters* |
| ArrayList<Offering> | mLecsOfferings *refers to the lecture offerings* |
| ArrayList<Offering> | mLabsOfferings *refers to the lab offerings* |
| ArrayList<Offering> | mTutsOfferings *refers to the tutorial offerings* |
| ArrayList<Offering> | mSemsOfferings *refers to the semester offerings* |
| public String | mSubject *refers to the subject* |
| public String | mCode *refers to the code* |
| public String | mDesc *refers to the description* |
| public String | mInstructor *refers to the instructor* |
| public String | mInstructor\_email *refers to the instructor email* |
| public ArrayList<Offering> | mOfferings *refers to the offerings* |
| public ArrayList<Task> | mTasks *refers to the tasks* |
| public ArrayList<Contact> | mContacts *refers to the contacts* |
| private Button | mSaveButton *refers to the save button* |
| private Button | mCancelButton *refers to the cancel button* |
| private CourseHandler | mCourseHandle *refers to the course handler used to access the course functionality* |
| private TextView | mSubjectTextView *refers to the textView to display subject* |
| private TextView | mCodesTextView mSubjectTextView *refers to the textView to display codes* |
| private Bundle | mCourseBundle *refers to the bundle passed with course information* |
| private Intent | mCallingIntent *refers to the calling intent* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | onCreate(Bundle savedInstanceState) *creates a new course handler and sets the content view* |
| Void | init() i*nitialize all views and sets Button OnClickListeners.* |

#### 6.2.2.10 Class ****MasterCourse****

Public class Mastercourse

This class is a wrapper class for course timetable information

|  |  |
| --- | --- |
| **Field Summary** | |
| public String | id *refers to the course id* |
| public String | Code *refers to the course code* |
| public String | Subj *refers to the course subject* |
| public String | Desc *refers to the course description* |
| public String | Type *refers to the course type* |
| public String | Sec *refers to the course sec* |
| public String | Dur *refers to the course duration* |
| public String | Days *refers to the course days* |
| public String | Time *refers to the course time* |
| public String | Location *refers to the course location* |
| public String | Instructor *refers to the course instructor* |

#### 6.2.2.11 Class ****Offering****

public class Offering

This class is a wrapper class for Offering information

|  |  |
| --- | --- |
| **Field Summary** | |
| public int | id *refers to the offering id* |
| public String | mSubj *refers to the offering subject* |
| public String | mCode *refers to the offering code* |
| Public int | mSection *refers to the offering section* |
| Public String | mType *refers to the offering type* |
| public ArrayList<OfferingTime> | mOfferingTimes *refers to the offering times* |
| public int | mOid *refers to the offering time id* |

#### 6.2.2.12 Class ****OfferingTime****

public class OfferingTime

This class is a wrapper class for OfferingTime information

|  |  |
| --- | --- |
| **Field Summary** | |
| public int | mOid *refers to the offering time id* |
| public String | mStartTime *refers to the offering time start time* |
| public String | mEndTime *refers to the offering time end time* |
| Public String | mDay *refers to the offering time day* |
| Public String | mLocation *refers to the offering time location* |

|  |  |
| --- | --- |
| **Constructor** | |
| Public | Offering()*creates a new arrayList of offeringtimes* |

#### Contacts

##### **Description**

The Contacts consists of 4 classes that encapsulate both the user interface as well as all the contacts functionality and these 4 classes are included in the package edu.seaaddicts.brockbutler.contacts

##### **Class AddContactActivity**

public class AddContactActivity extends Activity

This class handles adding of contacts

|  |  |
| --- | --- |
| **Field Summary** | |
| private Button | mSaveButton *refers to the save button* |
| private Button | mCancelButton *refers to the cancel button* |
| private Contact | mContact *refers to the contact* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | onCreate(Bundle savedInstanceState) *creates a new course handler and sets the content view* |
| Void | init() i*nitialize all views and sets Button OnClickListeners.* |

##### **Class Contact**

public class Contact

This class is a wrapper class for Contact information

|  |  |
| --- | --- |
| **Field Summary** | |
| Public String | mSubj *refers to the contact subject* |
| Public String | mCode *refers to the contact code* |
| Public String | mFirstName *refers to the contact first name* |
| Public String | mLastName *refers to the contact last name* |
| Public String | mEmail *refers to the contact email* |
| public int | mId *refers to the contact id* |

##### **Class ContactsActivity**

public class ContactsActivity extends Activity

This class provides the UI for contacts

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | onCreate(Bundle savedInstanceState) *creates a new course handler and sets the content view* |
| Void | onCreateOptionsMenu(Menu menu) *inflates the menu and adds items to the action bar if present* |

##### **Class ContactsAdapter**

public class ContactsAdapter extends BaseAdapter implements OnClickListener

This class works as an adapter in that it provides contact information

|  |  |
| --- | --- |
| **Field Summary** | |
| private Context | context *refers to the context* |
| private List<Contact> | mContactsList *refers to the contacts list* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Int | getCount() *gets the number of contacts* |
| Object | getItem(int position) *gets a particular item* |
| long | getItemId(int position) *gets a particular itemid* |
| View | getView(int position, View convertView, ViewGroup viewGroup) *obtains the view for the given parameters and sets the listeners appropriately* |
| Void | onClick(View view) *removes the contact on click and notifies the change* |

### Tour

#### Description

The Tour consists of 5 classes to accomplish its functionality; they are TourActivity, TourHall,TourInfo, TourNode, TourRoom. These 4 classes are included in the package edu.seaaddicts.brockbutler.tour

#### Class TourActivity

public class TourActivity extends Activity

This class is the main activity for the tour functionality

|  |  |
| --- | --- |
| **Field Summary** | |
| private TourNode[] | Nodes *refers to the nodes* |
| private final int | numImages *to store the number of images* |
| private TourInfo | Info *to store tour information* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | onCreate(Bundle savedInstanceState) *initializes all the buttons and all the nodes in preparation to execute tour* |
| Boolean | onCreateOptionsMenu(Menu menu) *inflates the menu with menu items* |
| Void | goBack(MenuItem item) *Pops the previous TourNode from the stack and goes to that node.* |
| Void | turnAround(MenuItem item) *Goes to the node in the tour which is logically turning around.* |
| Void | teleport(MenuItem item) *Teleports the user to the specified block.* |
| Void | endTour(MenuItem item) *Ends the tour and destroys the Activity* |
| Void | onBackPressed()*pop previous TourNode from the stack and goes to that node.* |
| Int | idx(int r) *returns index in `nodes` of TourNode which shows the image `r* |
| Void | room(int r, String roomNumber) *Adds a new TourRoom to nodes.* |
| Void | hall(int r, int ll, int ul, int c, int ur, int lr, int ta) *Adds a new TourHall to nodes.* |
| Void | void hall(int r, int c) *same as above but for a special case* |
| Void | hall(int r, int c, int ta) *same as above but for a special case* |
| Void | hall (int r, int ll, int ul, int c, int ur, int lr) *same as above but for a special case* |
| Void | initNodes() *method to link all of the images together.* |

#### Class TourNode

public class TourHall extends TourNode

This class encapsulates a hallway node in the tour. Hallways have multiple branching points,

and can turn around. Each hallway knows about all of it's branching points.

|  |  |
| --- | --- |
| **Field Summary** | |
| private TourNode[] | Nodes *refers to each of the nodes this node branches off to* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | setOuterLeftNode(TourNode node) *sets the outer left node* |
| Void | setInnerLeftNode(TourNode node) *sets the inner left node* |
| Void | setCenterNode(TourNode node) *sets the center node* |
| Void | setInnerRightNode(TourNode node) *sets the inner right node* |
| Void | setOuterRightNode(TourNode node) *sets the outer right node* |
| Public String | makeTitle(int img) *makes title for the image* |
| Void | paint(final TourInfo info) *hanges the image displayed on the screen and redefines where the buttons lead us to.*  *Also pushes this node onto the TourInfo's history.* |

|  |  |
| --- | --- |
| **Constructor** | |
| Public | TourHall(int img, TourNode ll, TourNode ul, TourNode c, TourNode ur, TourNode lr) *Forward pass constructor. Defines each button, and inits the turnaround location to null.* |
| Public | TourHall(int img, TourNode ll, TourNode ul, TourNode c, TourNode ur, TourNode lr) *Second pass constructor. Defines each button, and also defines the turnaround node. Links `turnAroundNode` back to this node via turning around.* |

#### Class TourInfo

public class TourInfo

Wrapper class for all passing required information around throughout the tour.

|  |  |
| --- | --- |
| **Field Summary** | |
| public RelativeLayout | Rl  *layout of which the background is changed from node to node* |
| public ImageButton[] | Buttons *refers to array of ImageButtons* |
| public int[] | Arrows *refers to arrows for direction* |
| public Context | Context *getApplicationContext(), for toasts* |
| public TourNode | Current *refers to current tour node* |
| public Stack<TourNode> | History *refers to the previous ones* |

|  |  |
| --- | --- |
| **Constructor** | |
| Public | TourInfo(RelativeLayout r, ImageButton[] b, Context c) *initializes all the fields* |

#### Abstract Class TourNode

public abstract class TourNode

Simple abstract wrapper for TourRoom and TourHall*.*

|  |  |
| --- | --- |
| **Field Summary** | |
| protected int | Image *resource for the image on this node* |
| protected TourNode | turnAroundNode *refers to the turn around node* |
| protected String | Title *text for Toast to display, if any* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Boolean | canTurnAround() |
| TourNode | getTurnAroundNode() |
| Void | setTurnAroundNode(TourNode node) |
| Void | setOuterLeftNode(TourNode node) |
| Void | setInnerLeftNode(TourNode node) |
| Void | setCenterNode(TourNode node) |
| Void | setInnerRightNode(TourNode node) |
| Void | setOuterRightNode(TourNode node) |
| Public abstract void | Paint() |

#### Class TourRoom

public class TourRoom extends TourNode

A room node in the tour. Rooms are simple, they don't go anywhere.

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | *Paint*(TourInfo info) *sets the background image and adds listeners to the various buttons* |

|  |  |
| --- | --- |
| **Constructor** | |
| Public | TourRoom(int img, String s) *Initializes this room, rooms only need the image resource value* |

### Maps

#### Description

The Maps consists of 3 classes to accomplish its functionality; they are MapsActivity, MapsHandler and MapsTouchImageView. These 3 classes are included in the package edu.seaaddicts.brockbutler.maps

#### Class MapsActivity

public class MapsActivity extends Activity

This class provides the UI for user interaction with the Maps

|  |  |
| --- | --- |
| **Field Summary** | |
| private static final String | TAG *is used for debugging* |
| private TextView | mTemp *refers to textView* |
| private Button | Start *refers to the start button* |
| private Button | Resume *refers to resume* |
| private Position | pTest *refers to the position* |
| private EditText | mSearchEditText *refers to textExit* |
| private Handler | mHandler *refers to handler* |
| private MapsTouchImageView | mMapImage *refers to the MapsTouchImageView* |
| private MapsHandler | mMapsHandler *refers to the maps hadler* |
| private Position | currentLocation *refers to current position* |
| private Position | mStartPosition *refers to start position* |
| private Position | mGoalPosition *refers to goal position* |
| private Astar | School *refers to the Astar object which is used in path finding* |
| private Context | mContext *refers to the context* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Protected void | onCreate(Bundle savedInstanceState) *handles the messages on create* |
| boolean | onCreateOptionsMenu(Menu menu) *inflates the menu* |
| Void | init() *initializes the map* |
| Void | onBackPressed() *pauses the current thread to go backwards* |
| Void | exitMaps(MenuItem item) *exits the map activity* |
| Void | displaySearchDialog(MenuItem item) *Prompts user to search for existence of a location.* |
| Void | displayGetDirectionsDialog(MenuItem item) *Displays AlertDialog for user to enter destination. First the location is determined to exist, then if true path is drawn on map.* |
| Void | showHelp(MenuItem item) *calls the help activity for maps* |
| Void | updatePosition(MenuItem item) |

#### Class MapsHandler

public class MapsHandler extends Handler

This class provides functionality for different user events raised by user interactions with the map

|  |  |
| --- | --- |
| **Field Summary** | |
| public static final int | MAPS\_REQUEST\_UPDATE = 0x001; |
| public static final int | MAPS\_REQUEST\_LOCATION\_EXISTS = 0x002; |
| public static final int | MAPS\_REQUEST\_DIRECTION = 0x003; |
| public static final int | MAPS\_SEND\_POSITION = 0x004; |
| public static final int | MAPS\_SEND\_DIRECTIONS = 0x005; |
| public static final int | MAPS\_ERROR\_NO\_LOCATION = 0x006; |
| public static final int | MAPS\_ERROR\_NO\_WIFI = 0x007; |
| public static final int | THREAD\_REQUEST\_START = 0x008; |
| public static final int; | THREAD\_REQUEST\_STOP = 0x009 |
| public static final int | THREAD\_REQUEST\_PAUSE = 0x010; |
| public static final int | THREAD\_REQUEST\_RESUME = 0x011; |
| public static final int | THREAD\_UPDATE\_POSITION = 0x012; |
| private static final String | tag = "MapsHandler" *used for debugging* |
| private Handler | mMainHandler *refers to the main handler* |
| private Thread | mMapsThread *refers to the maps thread* |
| private Object | mPauseLock *is used for synchronization* |
| private boolean | mIsPaused *keeps the state of the thread* |
| private boolean | mIsFinished *indicates finishing* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | Init() *initializes the map and starts the thread* |
| Void | handleMessage(Message msg) *handles different messages passed by the user interaction with the maps* |

#### Class MapsTouchImageView

public class MapsTouchImageView extends ImageView

This class Allows pinching, zooming, translating, and drawing on an ImageView.

|  |  |
| --- | --- |
| **Field Summary** |  |
| private static final String | TAG = "MapsTouchImageView" |
| private static final int | MAP\_WIDTH = 2000 |
| private static final int | MAP\_HEIGHT = 1100 |
| private static final int | CLICK = 3 |
| private Matrix | mMatrixMap *matrix that represents the map* |
| private static final int | NONE = 0 |
| private static final int | DRAG = 1 |
| private static final int | ZOOM = 2 |
| private int | mode = NONE; |
| private PointF | Last *the destination point* |
| private PointF | Start *the starting point* |
| private float | minScale = 1f |
| private float | maxScale = 8f |
| private float[] | M *floating points to aid in artith* |
| private double | mMapRatio*holds the map ratio* |
| private int | viewWidth, viewHeight *width and height view* |
| private int | oldMeasuredWidth, oldMeasuredHeight *to hold the old width and height* |
| private float | scaleFactor = 1f *represents the scaling factor* |
| private float | origWidth, origHeight *originalHeight and originalWidth* |
| private final | Paint mPathPaint *to plot the points for the path* |
| private ScaleGestureDetector | mScaleDetector *used in scaling the map* |
| private Context | mContext *holds the context* |
| int | actionBarHeight *holds the action bar height* |
| public Position[] | mPosition *holds a list of positions* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | onDraw(Canvas canvas) *draws a line between the points given* |
| void | sharedConstructing(Context context) *set the listeners and responds to user interaction* |
| Void | setMaxZoom(float x) *sets the maximum zoom scale* |

#### Class AStar

public class Astar

Path finding algorithm for navigation route

|  |  |
| --- | --- |
| **Field Summary** | |
| private Position[] | Graph *this is the graph that we are searching.* |

|  |  |
| --- | --- |
| **Method Summary** | |
| boolean | nodeExist(Position node) t *his method returns true or false if a given position exists* *within the graph(school). This method is to be used before pathGeneration.* |
| Position | findPosition(String nodeName) *This method returns a position if it exists within the graph(school). Used by the interface searching feature.* |
| Position[] | pathGeneration(Position startNode, Position goalNode)  *The pathGeneration method is the main part of the A\* algorithm. This method achieves an efficient and route between two positions based on a heuristic score.* |
| Position[] | pathReturn(Position end) throws ClassCastException  *This method takes the generated route from the A\* algorithm and processes it into a usable ArrayList of positions to be passed to the mapping activity for drawing a route on the* *map.* |
|  |  |

|  |  |
| --- | --- |
| **Constructor** | |
| Public | Astar () *Constructor method for the ASTAR class. The constructor creates the searchable graph space that represents the school hallways and classrooms.* |

#### Class Position

public class Position implements Comparable<Object>

|  |  |
| --- | --- |
| **Field Summary** | |
| public int | xPosition *refers to x position* |
| public int | yPosition *refers to y position* |
| public double | fScore *refers to fscore* |
| public double | gScore *refers to gscore* |
| public double | hScore *refers to hscore* |
| public String | nodeNumber *refers to the node number* |
| public String | nodeName *refers to the node name* |
| public boolean | Visited *refers to a flag to indicate whether the node’s been visited* |
| public Position | From *starting position* |
| public Position | accesible[] *wheelchair accessible nodes* |
| public Position | nonaccesible[] *wheelchair nonaccessible* |

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | setCoordinates (int inputX, int inputY) *Set coordinates* |
| Void | setNumber (String inputNumber) *Set position number* |
| Void | setName (String inputName) *Set position description* |
| Int | getX ( ) *get x coordinate* |
| Int | getY ( ) *get y coordinate* |
| String | getNumber ( ) *get node number* |
| String | getName ( ) *Get node name* |
| boolean | compare (Position node) *Compares this node to another* |
| Int | compareTo (Object node) |
| Void | printCoordinates ( ) *testing method* |
| Void | printNumber ( ) *testing method* |
| Void | printName ( ) *testing method* |

|  |  |
| --- | --- |
| **Constructor** | |
| Public | Position ( ) *Constructor methods for no arguments* |
| Public | Position (int inputX, int inputY) C*onstructor with coordinates set* |
| Public | Position (int inputX, int inputY, String inputName, String inputNumber) *Constructor with all position information set* |

### Help

#### Description

Help is implemented in 4 classes they are HelpActivity, MapsActivity, CourseManagerActivity and schedulerActivity and the HelpActivity is included in the package edu.seaaddicts.brockbutler.help

#### Class HelpActivity

This class provides the help functionality fo r different activities

public class HelpActivity extends Activity

|  |  |
| --- | --- |
| **Method Summary** | |
| Void | onCreate(Bundle savedInstanceState) *creates a helpActivity in a webview and also listens for calls from other activities* |
| Void | *onCreateOptionsMenu(Menu menu) inflates the menu* |

## Miscellaneous

Originally the tables for the database were created when the database was created if it did not exist on first use. Since the data could no longer be retrieved from the Brock University registrar’s website the full list of course offerings had to be included in a database to be shipped with the application. Due to this the database needed to be created in a different way.

On start up of the course manager or scheduler if the database has not been constructed it will be initialized then the tables and information will be copied over from the shipped database. After this initial set up, functionality will continue as it was originally planned.

Due to the new method of database creation addCourse() in CourseListHandler is depreciated until a new set of course offerings are available for the next fall/winter session.

Since addCourse() is depreciated, methods using it will also be depreciated, these methods are updateAll and getAllCourses in the CourseHandler class.

Brocku.java is the class that handled all the online interaction to retrieve all the course offerings offered at Brock is no depreciated as well since currently no offerings exist for it to handle.

Getting data from my.brocku.ca

It was attempted to get student data from a students’ my.brocku.ca account but there was a major security concern with this application having access to that data. Since the data is confidential and accessing it would be security breach on Brock’s computer systems, getting permissions to access the data was denied for an undergraduate project. Due to this security concern this part of the application was not possible to complete.

## Management

Management of the source code related to BrockButler was hosted on GitHub and the project design & development was divided into five phases which had the following phase plan & time estimation

**Definition Phase**: In this phase we defined the group members, defined the scope of the document, a source control was setup namely GitHub, a testing plan was devised, standards document compiled, the group was trained on using Android & module roles were assigned. The rough time estimation for this phase is 3 months

**Design Phase**: In this phase the following were designed System Architecture Diagram , System use case diagram, Modular use case diagram, Modular use case documents, Class Design Documents, Test Case Documents, Database Schema. The rough time estimate for this phase is 1 month.

**Development Phase**: In this phase the following were engineered Virtual Tour Photos taken, Database Constructed, Location Nodes Calculated, Mapping Module Complete, Tour Module Complete, Scheduler Module Complete, Interface Module Complete, & Application Coding Complete. The rough times estimate for this phase being 1 month.

**System Test Phase**: In this phase the following were developed Unit Testing, Integration Testing, Interface Testing, Recovery Testing, Acceptance Testing & User Procedure Testing. The rough time estimates for this phase

**Acceptance Phase:** In this phase the following were accomplished Project Evaluation, Group Evaluation, User Manual, and Technical Manual & Presentation. This phase took roughly 1 week.

## Acknowledgements

We would like to express our thanks to the following individuals:

**Wlodzimierz Wojcik**:We would like to express our gratitude to Vlad for his guidance as well as insights during the course of our projects

**Earl Foxwell:** Earl was instrumental in providing networking advice which was helpful during the course of development

**Erin Sauder:** Erin provided us with the graphic icons which was necessary in making the app look good for which we are grateful

## Bibliography

Foundation, j. (2013, 01 01). *jQuery* . Retrieved from jQuery : http://jquery.com/

GitHub. (2013, 01 01). *GitHub Training*. Retrieved from Github Training: http://training.github.com/

Google. (2013, 01 01). *Developer Training*. Retrieved from Android Developers Site: http://developer.android.com/training/index.html

Ortiz, M. (2013, 01 01). *TouchImageView*. Retrieved from Github: https://github.com/MikeOrtiz/TouchImageView

Pressman, R. (2006). *Software Engineering, 7th Edition.* Conniticut : McGraw-Hill.