

National College of Ireland
BSc in Computing
2015/2016

Jordan Daly
13105272
jordan.daly@student.ncirl.ie



Final Year Project
Technical Report

Table of Contents

Executive Summary	5
1 Introduction	6
1.1 Background.....	6
1.2 Aims.....	6
1.3 Technologies	7
2 System.....	8
2.1 Requirements	8
2.1.1 Functional requirements.....	8
2.1.2 Data requirements.....	10
2.1.3 User requirements.....	10
2.1.4 Usability requirements.....	10
2.2 Design and Architecture.....	11
2.3 Implementation	13
2.4 Testing.....	25
2.4.1 Unit Testing	26
2.4.2 User Testing.....	26
2.5 Graphical User Interface (GUI) Layout.....	28
3 Conclusion.....	56
4 Further development or research.....	57
5 References	58
6 Appendix.....	60
6.1 Project Proposal	60
1. Objectives.....	60
2. Special resources required	64
3. Project Plan	65
4. Technical Details	65
5. Evaluation.....	65

6.2	Project Plan	66
6.3	Monthly Journals.....	66
6.3.1	September.....	66
6.3.2	October	68
6.3.3	November.....	69
6.3.4	December.....	70
6.3.5	January	71
6.3.6	February.....	73
6.3.7	March	73
6.3.8	April	74
6.4	Requirement Specification	75
1.	Introduction.....	78
6.5	Purpose	78
6.6	Project Scope	78
6.7	Definitions, Acronyms, and Abbreviations.....	79
2.	User Requirements Definition	79
1.	Search for particular Colleges, Courses, Modules & Clubs/Societies	79
2.	Read reviews of Colleges, Courses, Modules & Clubs/Societies	79
3.	Post reviews of Colleges, Courses, Modules & Clubs/Societies	79
4.	Login with a social network credentials	80
5.	Register new email account	80
6.	Login with email account.....	80
7.	Save favorites	80
8.	Search history	80
9.	Subscribe to updates.....	80
10.	Settings	80
11.	Intuitive User Interface Navigation.....	80
7	3. Requirements Specification	81
7.1	Functional requirements	81
12.	Use Case Diagram	81
13.	Requirement 1 Search for Colleges, Courses, Modules & Clubs/Societies : Use Case A1	81

14.	Requirement 2 Read & Create Reviews : Use Case B1&B3	83
15.	Requirement 3 Email User Account management; Register and Login : Use Case F1&F3	85
16.	Requirement 4 Login with Social network account : Use Case G1 ..	86
17.	Requirement 5 Static data maintenance; CRUD Colleges, Courses, Modules and Clubs/Societies : Use Case D1	88
18.	Requirement 6 Flag Spam: Use Case C2	89
19.	Requirement 7 Moderate Reviews : Use Case C1	90
20.	Requirement 8 Mark as Helpful: Use Case B2	90
21.	Requirement 9 Post Advert & Pay for Advert: Use Case E1 & E2 ..	90
22.	Requirement 10 Save Favourites & view Search History: Use Case A2 & A3	91
23.	Requirement 11 Subscribe to Updates: Use Case A4	91
24.	Requirement 12 Search Colleges on Google Maps interface: Use Case A1	91
7.2	Non-Functional Requirements	92
25.	Performance/Response time requirement	92
26.	Availability requirement	92
27.	Recover requirement	92
28.	Robustness requirement	92
29.	Security requirement	92
30.	Reliability requirement	92
31.	Maintainability requirement	92
32.	Portability requirement	92
33.	Extendibility requirement	93
34.	Reusability requirement	93
35.	Resource utilization requirement	93
3.	Interface requirements	93
7.3	GUI	93
7.4	Application Programming Interfaces (API)	100
4.	System Architecture	100
5.	System Evolution	102

Executive Summary

StuRevu's mission is to allow current students and graduates to provide advice to prospective students and feedback to Educational Institutions. The aim is to provide this functionality through the means of an Android App.

The main problems StuRevu is attempting to address are the **student dropout rates** in third level education and the lack of awareness of the differences in **graduate unemployment rates** between different graduate employment areas. These can be caused by a lack of information available to prospective students regarding their chosen college or course.

StuRevu will strive to help solve these problems by creating a central, independent database of advice and information. It will basically facilitate peer recommendations in the education market and may evolve into a communication/CRM tool for the education community to more closely connect staff, students and alumni.

1 Introduction

1.1 Background

There a lack of information and advice available when researching education options in Ireland. Prospective students need a way to access word-of-mouth advice from their peers who have been through the higher education system already. Ideally they should even be able to contact these graduates or current students to ask specific questions or queries. In a way the idea is to crowd-source career/education guidance.

Recent HEA survey data has shown that there is a significant difference in graduate salaries between graduates of different courses.

“computer science/information and communications technology (ICT) honours bachelor degree graduates were the highest earners, with 62 per cent earning €29,000 or over. Arts and humanities graduates earned the least, with 25 per cent earning less than €13,000.

The earnings gap between arts and other disciplines is striking. While 25 per cent of arts graduates earned below €13,000, the equivalent figure for computer science was just 5 per cent and for engineers it was 6 per cent.” (The Irish Times, 2016)

There is also a problem with progression rates that has been highlighted in the media.

“6,500 students, or one in six of all first years in higher education in Ireland, did not progress to second year between 2012 and 2014.

One-third of computer science students across all institutes of technology are dropping out after first year in college. “(The Irish Times, 2016)

1.2 Aims

The aim is to develop a Data Driven Android App that collects and displays reviews of third level Institutions and their Courses.

Prospective Students will gain access to the collective knowledge and experience of thousands of current students and graduates. They can use the App as a comparison tool when deciding on where and what to study.

This will hopefully result in higher progression and graduate employment rates from informed decision making in choosing courses and improvements in education quality from transparent feedback.

1.3 Technologies

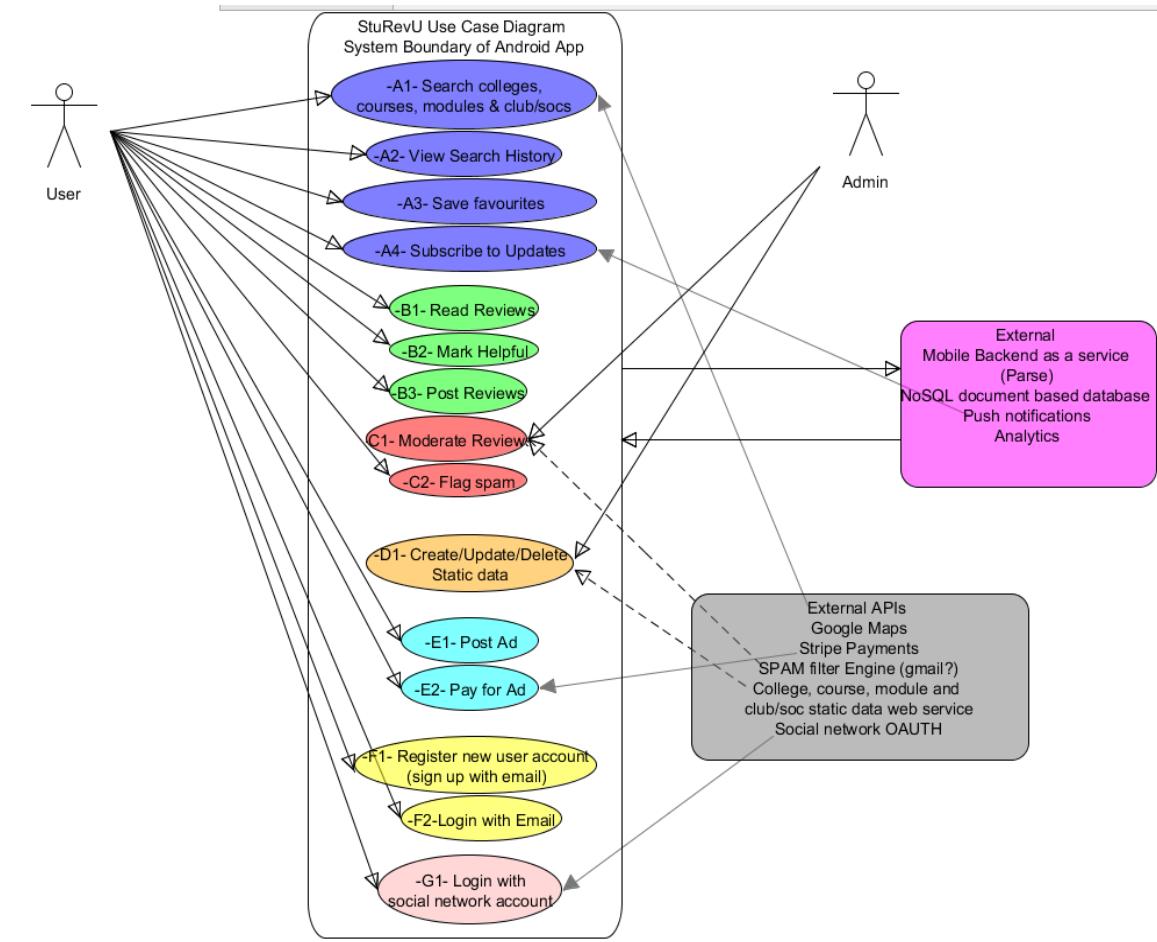
The Android App is implemented in Java and XML and works by making API calls to a backend service called Parse. Parse provides a “Core Data” service to allow CRUD functionality through its API to connect an Android App to a NoSQL database (Mongo DB) on its backend service cloud platform. Parse also provides Cloud Code, the ability to process data on the server side with JavaScript functions. The App integrates Google Maps API, Facebook Graph API and Google Cloud Messaging.

2 System

2.1 Requirements

2.1.1 Functional requirements

Figure 2-1 Use Case Diagram



<u>Done</u>	<u>Still in Backlog</u>	<u>Ideas (user feedback)</u>
Login & Create Account (Email/Username/Password)	Social Login (Google & LinkedIn)	anti-spambot check when user tries to create a new review to avoid abuse
List Colleges & Courses	Post & Pay Ad	report : list top rated colleges/courses

Read reviews (College & Course)	Module & Club/Soc (list, search, read and create reviews)	report : list of similar courses to compare fees, qualifications and other details etc
Create reviews (College & Course)	Limit to one “flag spam”/”vote helpful” per user per review	To highlight obvious revenge reviews instead of an average rating by review scores, display a graph of all review ratings so that outliers are clearly visible
Flag review as Spam	View Search History	New section for MOOCs to compare or to suggest an add-on to compliment the college course
Vote review helpful	User profile section	display/integrate LinkedIn profile details so users can see the professional/qualification credentials of a reviewer
Calculations of Average rating and counts (averageCollegeRating, averageCourseRating, countCollegeReviews, countCourseReviews, countCollegeCourses, countReviewComments)	Advanced Course Search	ability for user to ask a question to a college or course and for them to respond (post comments linked to college/course)
Google Map of Colleges		“Give to Get” model
Social Login (Facebook)		Share on Facebook
Save to Favourites (add & remove favourite College/Course/College review/Course review)		
Push Notifications (sent to user when a new review is added to a favourite or		

when a new comment/reply is added to a review/comment that they created)		
Search Courses		
Post comments linked to review and post reply linked to comment (ability for a user to ask a question to a reviewer and for them to respond)		
Navigation drawer menu (for quick access to favourites, my reviews, course search and college list)		
Toolbar (for actions such as refresh list, favourite, map, filter college list, search etc)		

2.1.2 Data requirements

College and Course data is required for testing and performing demos to show users and get feedback.

2.1.3 User requirements

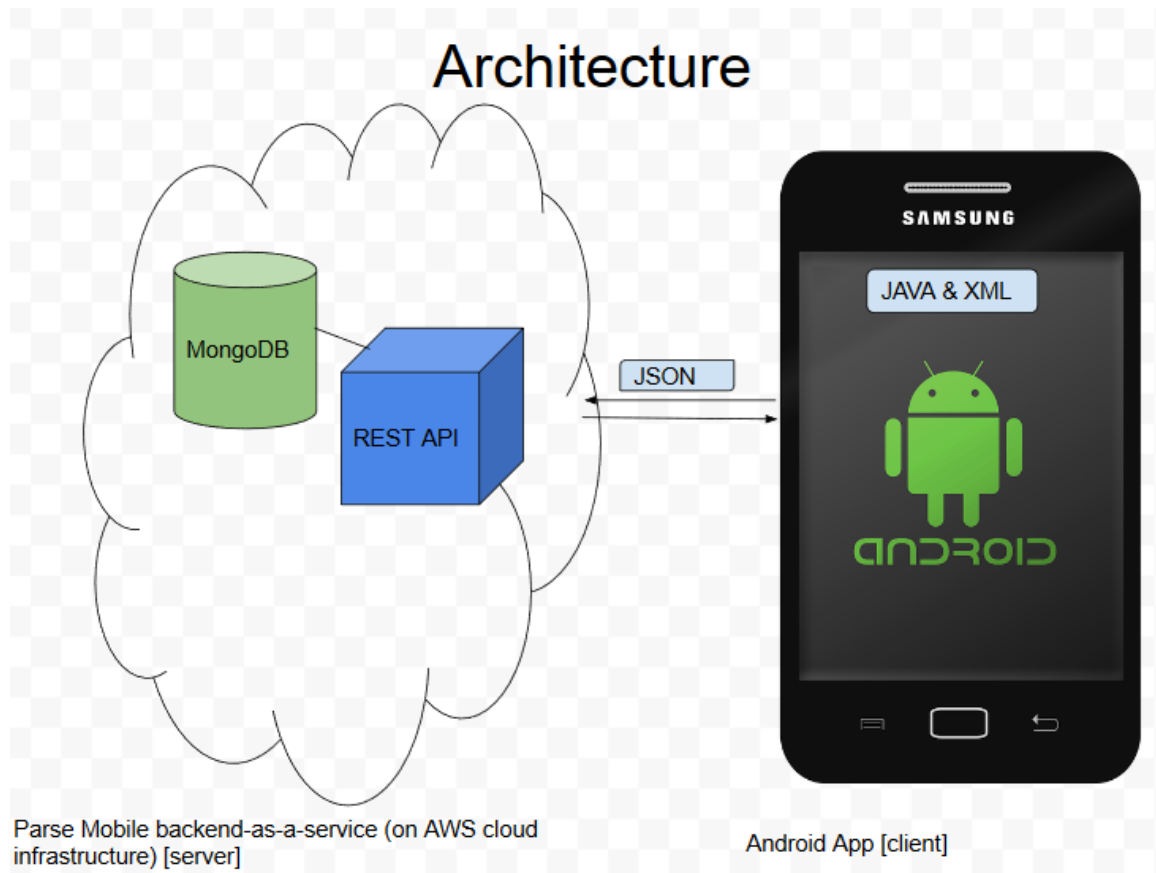
User requires an Android device, an Internet connection and Google Play Store.

2.1.4 Usability requirements

- Understandable: The User Interface should be simple and easy to understand
- Operable: The functionality should be consistent and easy to operate. Any errors should have explanatory error messages to help user to address the issue.
- Ease of use: the App should be easy to learn how to use with intuitive actions and navigation
- Attractiveness: the design of the layout and colour theme should be eye - catching and aesthetically pleasing

2.2 Design and Architecture

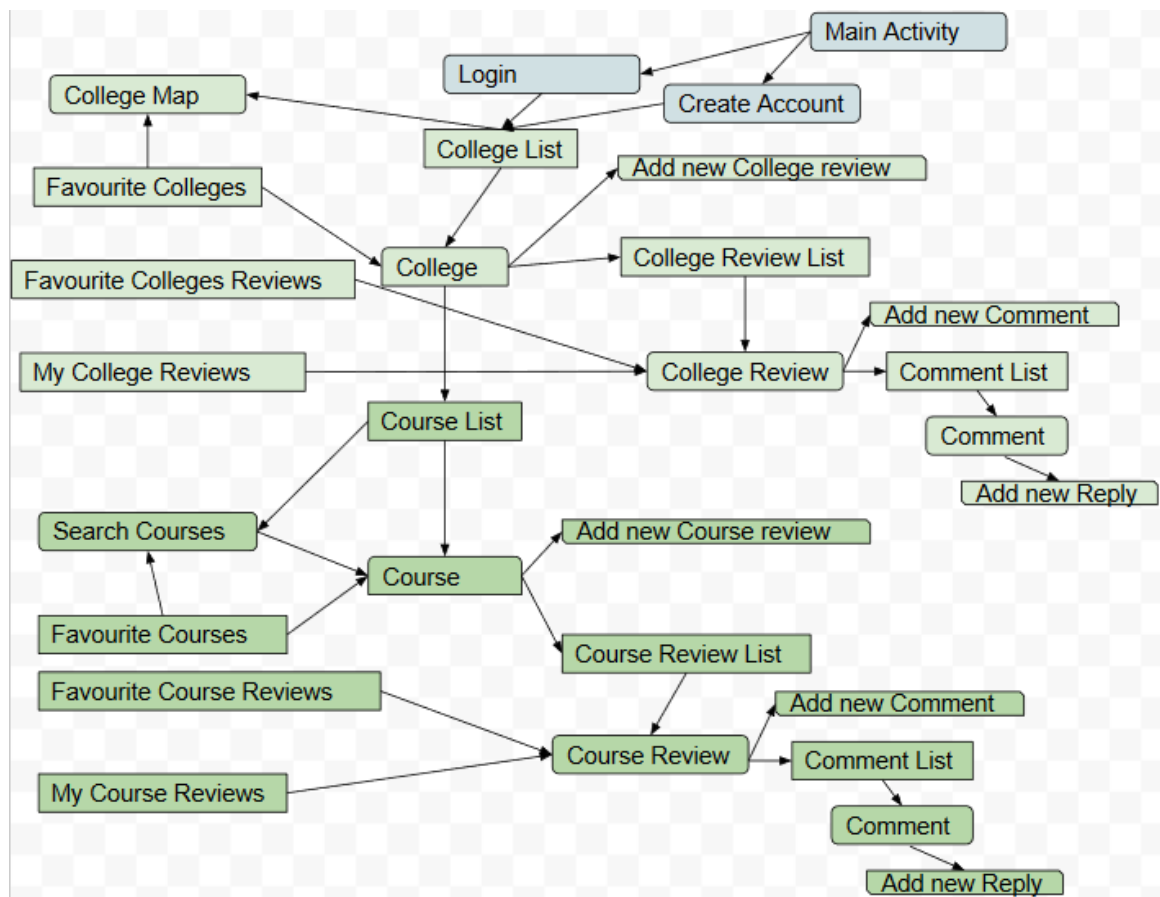
Figure 2-2 Architecture Overview



The Design of the App's structure is illustrated below to show the screen flow of the Android Activities.

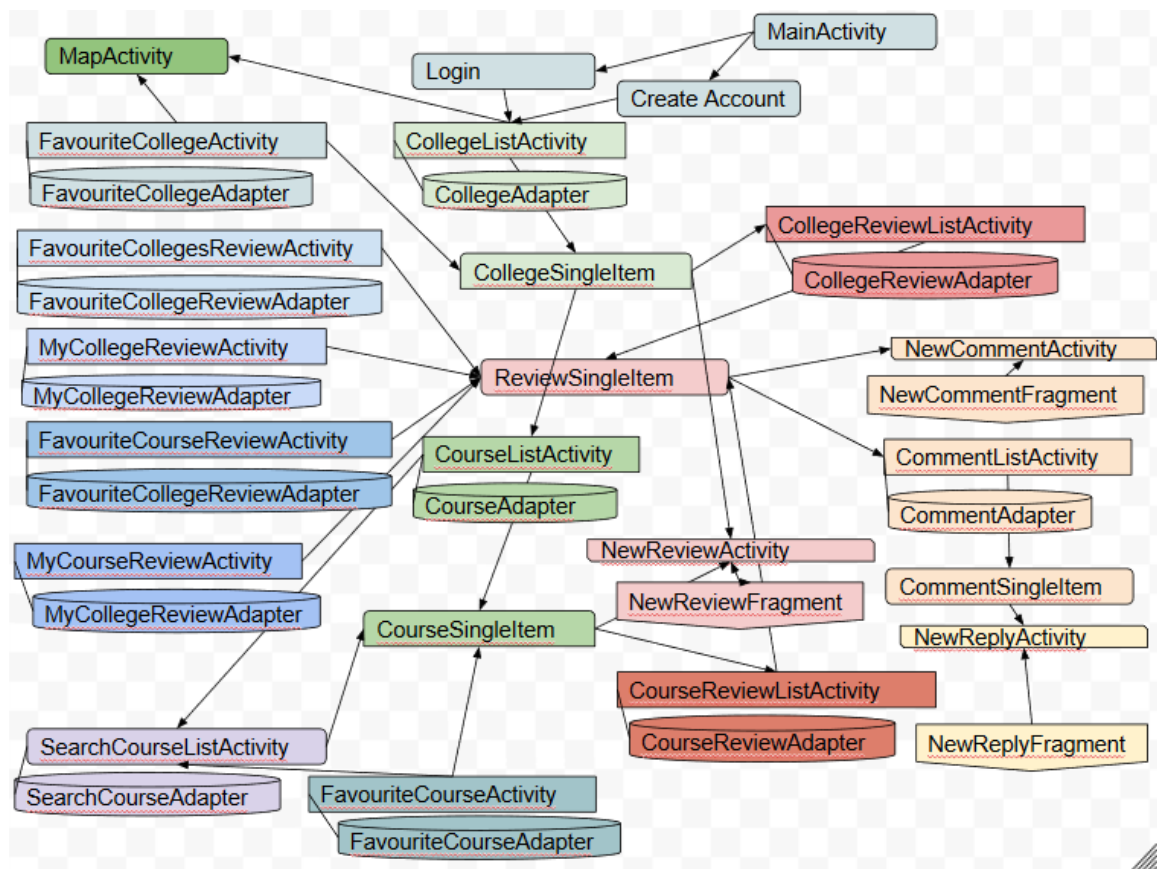
The parts in light green are related to College while the parts in dark green are related to Course.

Figure 2-3 Activity flow



The diagram below represents the actual Java Classes which contain the application logic.

Figure 2-4 Class Diagram



2.3 Implementation

List implementation

The Lists are implemented using List Views that are populated by Parse Query Adapters. An Inner Query is used to populate the course list with only courses that are linked to the college in question.

Figure 2-5 Course Adapter (Parse Query) code snippet

```
public class CourseAdapter extends ParseQueryAdapter<Course> {

    public CourseAdapter(Context context) {

        super(context, new ParseQueryAdapter.QueryFactory<Course>() {

            public ParseQuery<Course> create() {
                // Here we can configure a ParseQuery to display
                // only associated courses.
                ParseQuery innerQuery = new ParseQuery("College");
                String objectId = CourseListActivity.collegeId;

                Log.d("DEBUG", "collegeId3 is " + objectId);

                innerQuery.whereEqualTo("objectId", objectId);

                ParseQuery query = new ParseQuery("Course");
                query.whereMatchesQuery("College_Id", innerQuery);

                query.orderByAscending("Name");
                return query;
            }
        });
    }
}
```

Parse Proxy Object Open Source Solution

To pass the object identifiers from list activities to single item activities an open source solution “Parse Proxy Object” is used.

Figure 2-6 Parse Proxy Object Class (Open Source Solution) code snippet

```
// By Jamie Chapman, @chappers57
// License: open, do as you wish, just don't blame me if stuff breaks ;-)

import com.parse.ParseObject;
import com.parse.ParseUser;

import java.io.Serializable;
import java.util.Date;
import java.util.HashMap;

public class ParseProxyObject implements Serializable {

    private static final long serialVersionUID = 1L;
    private HashMap<String, Object> values = new HashMap<>();
    private String ObjectId;
    private Date CreatedAt;
    private Date UpdatedAt;

    public ParseProxyObject(ParseObject object) {
```

Parse Cloud Code JavaScript functions

To avoid having to fetch all rows of data onto the device to perform calculation and count operations Parse Cloud Code JavaScript functions are used to send a parameter to the cloud function and receive just the required result.

Figure 2-7 Parse Cloud Code Function code snippet

```
Parse.Cloud.define("averageCollegeRating", function(request, response) {
    var query = new Parse.Query("Review");
    query.equalTo("College_Id",
        {__type: "Pointer",
         className: "College",
         objectId: request.params.College_Id
        });
    query.find({
        success: function(results) {
            var sum = 0;
            for (var i = 0; i < results.length; ++i) {
                sum += results[i].get("Rating");
            }
            response.success(sum / results.length);
        },
        error: function() {
            response.error("college rating lookup failed");
        }
    });
});
```

Google Map of Colleges

Below is a code snippet of doMapQuery() method from MapActivity class (Google map of Colleges) showing markers being displayed based on the latitude and longitude of Colleges.

Figure 2-8 Google Map code snippet 1

```
// Colleges to show on the map
Set<String> toKeep = new HashSet<>();
// Loop through the results of the search
for (College college : objects) {
    // Add this college to the list of map pins to keep
    toKeep.add(college.getObjectId());
    // Check for an existing marker for this college
    Marker oldMarker = mapMarkers.get(college.getObjectId());
    // Set up the map marker's location
    MarkerOptions markerOpts =
        new MarkerOptions().position(new LatLng(college.getLocation().getLatitude(), college
            .getLocation().getLongitude()));
    // Set up the marker properties based on if it is within the search radius
    if (college.getLocation().distanceInKilometersTo(myPoint) > radius * METERS_PER_FEET
        / METERS_PER_KILOMETER) {
        // Check for an existing out of range marker
        if (oldMarker != null) {
            if (oldMarker.getSnippet() == null) {
                // Out of range marker already exists, skip adding it
                continue;
            } else {
                // Marker now out of range, needs to be refreshed
                oldMarker.remove();
            }
        }
        // Display a green marker with the college information
        markerOpts =
            markerOpts.title(college.getName()).snippet(college.getInitials())
                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN));
    }
}
```

Figure 2-9 Google Map code snippet 2

```
    } else {
        // Check for an existing in range marker
        if (oldMarker != null) {
            if (oldMarker.getSnippet() != null) {
                // In range marker already exists, skip adding it
                continue;
            } else {
                // Marker now in range, needs to be refreshed
                oldMarker.remove();
            }
        }
        // Display a green marker with the college information
        markerOpts =
            markerOpts.title(college.getName()).snippet(college.getInitials())
                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN));
    }
    // Add a new marker
    Marker marker = mapFragment.getMap().addMarker(markerOpts);
    mapMarkers.put(college.getObjectId(), marker);
    if (college.getObjectId().equals(selectedCollegeObjectId)) {
        marker.showInfoWindow();
        selectedCollegeObjectId = null;
    }
}
// Clean up old markers.
cleanUpMarkers(toKeep);
});
```

[Facebook Login](#)

The Login activity is integrated with Facebook Login to allow the user to sign in with their Facebook credentials. In the below code snippets you can see how ParseFacebookUtils (Parse API) allows implementation of Login with Facebook and how a GraphRequest (HTTP GET method) allows a user's Facebook details to be retrieved as JSON data.

Figure 2-10 Facebook login code snippet 1

```
mBtnFb.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        ParseFacebookUtils.logInWithReadPermissionsInBackground(Login.this, Permissions, new LogInCallback() {
            @Override
            public void done(ParseUser user, ParseException err) {

                if (user == null) {
                    Log.d("StuRevu", "Uh oh. The user cancelled the Facebook login.");
                } else if (user.isNew()) {
                    Log.d("StuRevu", "User signed up and logged in through Facebook!");
                    getUserDetailsFromFB();
                    startActivity(new Intent(Login.this, CollegeListActivity.class));
                } else {
                    Log.d("StuRevu", "User logged in through Facebook!");
                    getUserDetailsFromParse();
                    startActivity(new Intent(Login.this, CollegeListActivity.class));
                }
            }
        });
    }
});
```

Figure 2-11 Facebook login code snippet 2

```
private void getUserDetailsFromFB() {
    Bundle parameters = new Bundle();
    parameters.putString("fields", "email,name,picture");
    new GraphRequest(
        AccessToken.getCurrentAccessToken(),
        "/me",
        parameters,
        HttpMethod.GET,
        new GraphRequest.Callback() {
            public void onCompleted(GraphResponse response) {
                /* handle the result */
                try {
                    email = response.getJSONObject().getString("email");
                    mEmailID.setText(email);
                    name = response.getJSONObject().getString("name");
                    mUsername.setText(name);
                    JSONObject picture = response.getJSONObject().getJSONObject("picture");
                    JSONObject data = picture.getJSONObject("data");
                    // Returns a 50x50 profile picture
                    String pictureUrl = data.getString("url");
                    new ProfilePhotoAsync(pictureUrl).execute();
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        })
        .executeAsync();
}
```

Sliding Navigation Drawer

Sliding Navigation Drawer functionality is implemented in the code snippet below. This is added to every activity to ensure consistent navigation options throughout the application.

Figure 2-12 Navigation Drawer code snippet

```
private void addDrawerItems() {
    String[] osArray = {"College List", "Search Courses", "Favourite Colleges", "Favourite Courses", "Favourite College Reviews", "Favourite Course Reviews"};
    mAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, osArray);
    mDrawerList.setAdapter(mAdapter);

    mDrawerList.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            Toast.makeText(CollegeListActivity.this, "Navigation drawer!", Toast.LENGTH_SHORT).show();
            switch (position) {
                case 0: {
                    Intent intent = new Intent(CollegeListActivity.this, CollegeListActivity.class);
                    startActivity(intent);
                    break;
                }
                case 1: {
                    Intent intent = new Intent(CollegeListActivity.this, SearchCourseListActivity.class);
                    startActivity(intent);
                    break;
                }
                case 2: {
                    Intent intent = new Intent(CollegeListActivity.this, FavouriteCollegeActivity.class);
                    startActivity(intent);
                    break;
                }
                case 3: {
                    Intent intent = new Intent(CollegeListActivity.this, FavouriteCourseActivity.class);
                    startActivity(intent);
                    break;
                }
            }
        }
    });
}
```

Save Favourites and Push Notifications

An example of Add and Remove Favourite (College/Course/College Review/Course Review) functionality is seen in the below code snippet. This method queries the database for a record that matches the user and course and if it already exists then it removes the favourite (delete object and unsubscribe from parse push notification channel based on objectId of College/Course/Review). If it doesn't already exist then it adds the favourite (put object and subscribe to parse push notification channel based on objectId of College/Course/Review).

Figure 2-13 saveFavourite code snippet

```
private void saveFavourite() {
    ParseQuery query = new ParseQuery("Favourite");
    ParseObject user_id = ParseObject.createWithoutData("_User", ParseUser.getCurrentUser().getObjectId());
    query.whereEqualTo("User_Id", user_id);
    ParseObject course_id = ParseObject.createWithoutData("Course", courseID);
    query.whereEqualTo("Course_Id", course_id);
    query.getFirstInBackground(new GetCallback<ParseObject>() {
        public void done(ParseObject object, ParseException e) {
            if (e == null) {
                //object exists
                object.deleteInBackground();
                ParsePush.unsubscribeInBackground(courseID);
                Toast.makeText(getApplicationContext(), "Course Removed from Favourites!"
                    , Toast.LENGTH_LONG).show();
            } else {
                if (e.getCode() == ParseException.OBJECT_NOT_FOUND) {
                    //object doesn't exist
                    ParseObject favourite = ParseObject.create("Favourite");
                    ParseUser currentUser = ParseUser.getCurrentUser();
                    String userId = currentUser.getObjectId();
                    Log.d("DEBUG", "userId is " + userId);
                    favourite.put("User_Id", ParseObject.createWithoutData("_User", userId));
                    favourite.put("Course_Id", ParseObject.createWithoutData("Course", courseID));
                    favourite.saveInBackground();
                    ParsePush.subscribeInBackground(courseID);
                    Toast.makeText(getApplicationContext(), "Course Added to Favourites!"
                        , Toast.LENGTH_LONG).show();
                } else {
                    //unknown error, debug
                }
            }
        }
    });
}
```

In the NewReviewFragment class a parse push notification (using Google Cloud Messaging) is sent to the College/Course objectId channel as seen in the code snippet below. This ensures that users who have added a College/Course as a favourite are sent a push notification when a new review is created.

Figure 2-14 NewReviewFragment code snippet 1

```
//send push notification and include college name
ParseQuery<ParseObject> query = new ParseQuery("College");
queryInBackground(collegeId, new GetCallback<ParseObject>() {
    public void done(ParseObject object, ParseException e) {
        if (e == null) {
            // object will be your college
            String collegeName = object.getString("Name");
            ParsePush push = new ParsePush();
            push.setChannel(collegeId);
            push.setMessage("A new review has been created for one your favourite Colleges: " + collegeName);
            push.sendInBackground();
        } else {
            // something went wrong
        }
    }
});
}
```

Figure 2-15 NewReviewFragment code snippet 2

```
//send push notification and include course description and college name
ParseQuery<ParseObject> query = new ParseQuery("Course");
query.include("College_Id");
query.getInBackground(courseId, new GetCallback<ParseObject>() {
    public void done(ParseObject object, ParseException e) {
        if (e == null) {
            // object will be your college
            String courseDesc = object.getString("Description");
            String collegeName = object.getParseObject("College_Id").getString("Name");
            ParsePush push = new ParsePush();
            push.setChannel(courseId);
            push.setMessage("A new review has been created for one your favourite Courses: " + courseDesc + " at " + collegeName);
            push.sendInBackground();
        } else {
            // something went wrong
        }
    }
});
}
```

In the NewCommentFragment class there are two types of Push Notifications. The first is sent to users who have added the College/Course Review as a favourite and is implemented using a channel based on the Review objectId. The second is implemented using an installation query and is sent to the user who created the College/Course Review.

Figure 2-16 NewCommentFragment code snippet 1

```
//send fav college/course review push notification and include course description and college name
ParseQuery<ParseObject> query_fav = new ParseQuery("Review");
query_fav.include("College_Id");
query_fav.include("Course_Id");
query_fav.include("Course_Id.College_Id");
query_fav.getInBackground(reviewId, new GetCallback<ParseObject>() {
    public void done(ParseObject object, ParseException e) {
        if (e == null) {
            // object will be your review
            String reviewTitle = object.getString("Title");

            if (object.getParseObject("Course_Id") != null) {
                String courseDesc = object.getParseObject("Course_Id").getString("Description");
                String collegeInitials = object.getParseObject("Course_Id").getParseObject("College_Id").getString("Initials");

                ParsePush push_fav_course = new ParsePush();
                push_fav_course.setChannel(reviewId);
                push_fav_course.setMessage("A new comment has been created for one your favourite Course Reviews, Title: " + reviewTitle + " for " + courseDesc);
                push_fav_course.sendInBackground();
            } else if (object.getParseObject("College_Id") != null) {
                String collegeInitials = object.getParseObject("College_Id").getString("Initials");

                ParsePush push_fav_college = new ParsePush();
                push_fav_college.setChannel(reviewId);
                push_fav_college.setMessage("A new comment has been created for one your favourite College Reviews, Title: " + reviewTitle + " at " + collegeInitials);
                push_fav_college.sendInBackground();
            } else {
                // something went wrong
            }
        }
    }
});
}
```

Figure 2-17 NewCommentFragment code snippet 2

```
//send push notification to author of college/course review
ParseQuery<ParseObject> query_author = new ParseQuery("Review");
query_author.include("User_Id");
query_author.include("College_Id");
query_author.include("Course_Id");
query_author.include("Course_Id.College_Id");
query_author.getInBackground(reviewId, new GetCallback<ParseObject>() {
    @Override
    public void done(ParseObject object, ParseException e) {
        if (e == null) {
            String author = object.getParseObject("User_Id").getObjectId();
            Log.d("DEBUG", "review UserId is " + author);
            // object will be your review
            String reviewTitle = object.getString("Title");
            Log.d("DEBUG", "review title is " + reviewTitle);

            if (object.getParseObject("Course_Id") != null) {
                String courseDesc = object.getParseObject("Course_Id").getString("Description");
                String collegeInitials = object.getParseObject("Course_Id").getParseObject("College_Id").getString("Initials");

                // Create our Installation query
                ParseQuery pushQuery = ParseInstallation.getQuery();
                ParseObject user_id = ParseObject.createWithoutData("_User", author);
                pushQuery.whereEqualTo("user", user_id);

                // Send push notification to query
                ParsePush push_author_course = new ParsePush();
                push_author_course.setQuery(pushQuery); // Set our Installation query
                push_author_course.setMessage("A new comment has been added to a Course Review that you created, Title: " + reviewTitle + " for " + courseDesc);
                push_author_course.sendInBackground();
            } else if (object.getParseObject("College_Id") != null) {
```

In NewReplyFragment class the push notification is implemented using an installation query and is sent to the user who created the Comment.

Figure 2-18 NewReplyFragment code snippet

```
//set up query for author of reply
ParseQuery<ParseObject> query = new ParseQuery("Comment");
query.include("User_Id");
query.getInBackground(commentId, (object, e) -> {
    if (e == null) {
        String author = object.getParseObject("User_Id").getObjectId();
        Log.d("DEBUG", "reply UserId is " + author);

        // Create our Installation query
        ParseQuery pushQuery = ParseInstallation.getQuery();
        ParseObject user_id = ParseObject.createWithoutData("_User", author);
        pushQuery.whereEqualTo("user", user_id);

        // Send push notification to query
        ParsePush push_author = new ParsePush();
        push_author.setQuery(pushQuery); // Set our Installation query
        push_author.setMessage("A new reply has been added to a comment that you created");
        push_author.sendInBackground();
    } else {
        // something went wrong
        Log.d("DEBUG", "something went wrong");
    }
});
```

Search Courses

The Search Course functionality is a basic search on course name using the `whereStartsWith()` method from the ParseQuery API, similar to the LIKE operator in SQL.

Figure 2-19 SearchCourseAdapter code snippet

```
public class SearchCourseAdapter extends ParseQueryAdapter<Course> {  
  
    public SearchCourseAdapter(Context context) {  
  
        super(context, new ParseQueryAdapter.QueryFactory<Course>() {  
  
            public ParseQuery<Course> create() {  
                // Here we can configure a ParseQuery to search  
                // based on course name starts with.  
                ParseQuery query = new ParseQuery("Course");  
                String course_name = SearchCourseListActivity.course_name;  
                Log.d("DEBUG", "course_name is " + course_name);  
                query.whereStartsWith("Name", course_name);  
                query.orderByAscending("Name");  
                query.include("College_Id");  
                query.setLimit(200);  
                return query;  
            }  
        });  
    }  
}
```

Complex relations with ParseObject & ParseQuery

An example of a more complex ParseQuery is in the FavouriteCourseReviewAdapter class. Dot notation is used to pull data with an include through multiple object pointers. Here we are getting objects from the Favourite table where;

- the User_Id column matches the current logged in user
- the Review_Id column is not null and if so we are including the related Review object in our query result
- the Course_Id column of the related Review object is not null and if so we are matching Review_Id column on the Review objects using an innerquery and including the related Course and _User objects in our query result.

Figure 2-20 FavouriteCourseReviewAdapter code snippet

```
public class FavouriteCourseReviewAdapter extends ParseQueryAdapter<Favourite> {  
  
    public FavouriteCourseReviewAdapter(Context context) {  
        super(context, new ParseQueryAdapter.QueryFactory<Favourite>() {  
            public ParseQuery<Favourite> create() {  
                // Here we can configure a ParseQuery to display  
                // Favourite Course reviews  
                ParseQuery query = new ParseQuery("Favourite");  
                ParseObject user_id = ParseObject.createWithoutData("_User", ParseUser.getCurrentUser().getObjectId());  
                query.whereEqualTo("User_Id", user_id);  
                query.whereExists("Review_Id");  
                query.include("Review_Id");  
                ParseQuery innerQuery = new ParseQuery("Review");  
                innerQuery.whereExists("Course_Id");  
                query.whereMatchesQuery("Review_Id", innerQuery);  
                query.include("Review_Id.Course_Id");  
                query.include("Review_Id.User_Id");  
                query.orderByAscending("createdAt");  
                return query;  
            }  
        });  
    }  
}
```

The Intent becomes more complex also when working with multiple nested relations (data type in parse is pointer).

Figure 2-21 FavouriteCourseReview Intent code snippet

```
private void setUpReviewItems() {  
    reviewListView.setOnItemClickListener(new OnItemClickListener() {  
        public static final String TAG = "buttonClick reviewlist";  
  
        @Override  
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
            Log.d(TAG, "onItemClick");  
  
            ParseObject favourite = reviewAdapter.getItem(position);  
            ParseProxyObject favourite_ppo = new ParseProxyObject(favourite);  
  
            Intent intent = new Intent(FavouriteCourseReviewActivity.this, ReviewSingleItem.class);  
            intent.putExtra("review", favourite_ppo);  
            intent.putExtra("reviewID", favourite.getParseObject("Review_Id").getObjectId());  
            intent.putExtra("reviewTitle", favourite.getParseObject("Review_Id").getString("Title"));  
            intent.putExtra("rating", favourite.getParseObject("Review_Id").getDouble("Rating"));  
            intent.putExtra("username", favourite.getParseObject("Review_Id").getParseObject("User_Id").getString("username"));  
            intent.putExtra("studentType", favourite.getParseObject("Review_Id").getString("Student_Type"));  
            intent.putExtra("contentPros", favourite.getParseObject("Review_Id").getString("Content_Pros"));  
            intent.putExtra("contentCons", favourite.getParseObject("Review_Id").getString("Content_Cons"));  
            intent.putExtra("contentAdvice", favourite.getParseObject("Review_Id").getString("Content_Advice"));  
            intent.putExtra("helpfulVoteCount", favourite.getParseObject("Review_Id").getInt("Helpful_Vote_Count"));  
            intent.putExtra("flaggedSpamCount", favourite.getParseObject("Review_Id").getInt("Flagged_Spam_Count"));  
            startActivity(intent);  
        }  
    });  
}
```

2.4 Testing

GenyMotion is a third party Android Emulator which was used to simulate an App running on different devices. I also connected a real Android device (OnePlus2) to my laptop to test by USB connection.

2.4.1 Unit Testing

During implementation as each new piece of functionality was added (as Java/XML code) Unit Tests were carried out to test for syntax errors at compile time and logical/semantic errors at run time. These Unit Tests were closely derived from the functional requirements.

The output of the android logcat displayed the full details of errors, for example java null pointer exceptions. The Android logging tool was used to display the value of variables for debugging purposes, for example the objectIds of database objects.

The Genymotion emulator was used to test basic functionality like UI, CRUD data operations and anything else not requiring integration with other systems. A real android device was used to test integration with Google Maps, Facebook Login and Push Notifications.

2.4.2 User Testing

To get feedback from a broad range of potential users, a few different profiles (prospective students, current students and graduates) were interviewed after testing StuRevu.

- Conor, 16, Secondary School Student with plan to go on to Full Time Third level Education. He likes StuRevu especially the Push Notifications but finds most of the functionality of the App could be implemented as a website to allow access on any type of device. He suggests including CAO points info for each Course.
- Kelly, 20, Retail Employee with plan to attend a college course part time. She is impressed by the Save Favourite functionality and the sliding drawer menu. She suggests a link to Apply for a Course online and an option to share StuRevu content on Facebook.
- Amy, 22, College Graduate with plan to do a Postgraduate course. She likes the concept of information sharing among the student community. She

suggests a list of top rated colleges and courses and integration with LinkedIn to see the profile of a reviewer.

- Andrew, 29, College Graduate working full time and planning to do part time courses to upskill. Andrew found StuRevu easy to use with an attractive layout but was disappointed that it was only available on Android as he is a loyal iPhone user. He suggested some sort of Anti-Spam bot protection to stop an attack from filling up the database with spam content or overloading the server with a DDOS (Distributed Denial of Service) attack.
- Jennifer, 18, Current College Student. Jennifer really liked the idea behind StuRevu but found that a user should be able to browse content before having to log in and should only have to log in when creating content. She suggested adding the ability to ask a question to a College/Course Admin user. She also thought a section for MOOCs (Massively Open Online Course) should be included as students are using these online courses to complement their traditional education.
- Micheal, 40, College Graduate working in Higher Education as Lecturer. He thought StuRevu was a useful idea and found the interface intuitive. He suggests a “Give to Get” model to encourage users to contribute content before they can access content.

2.5 Graphical User Interface (GUI) Layout

Figure 2-22 Main Activity

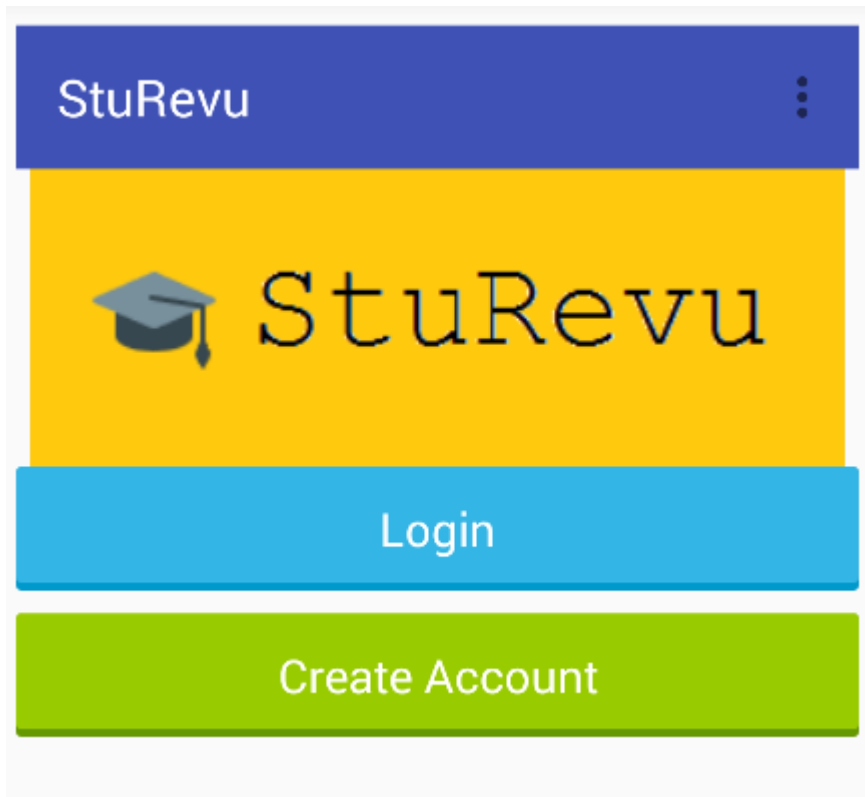
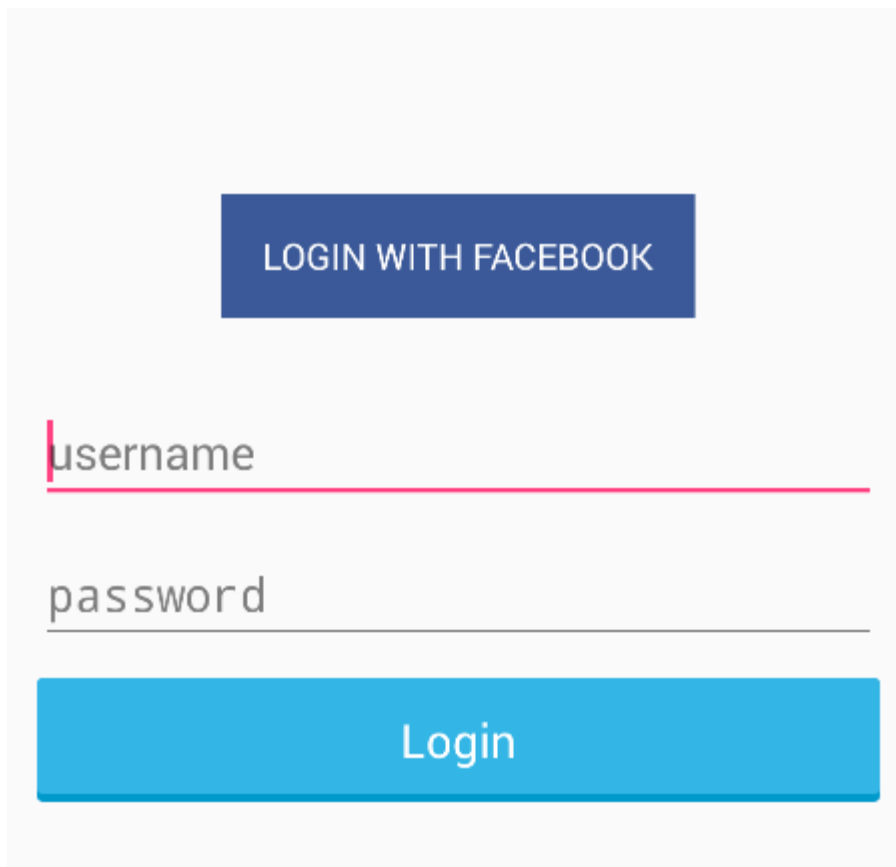
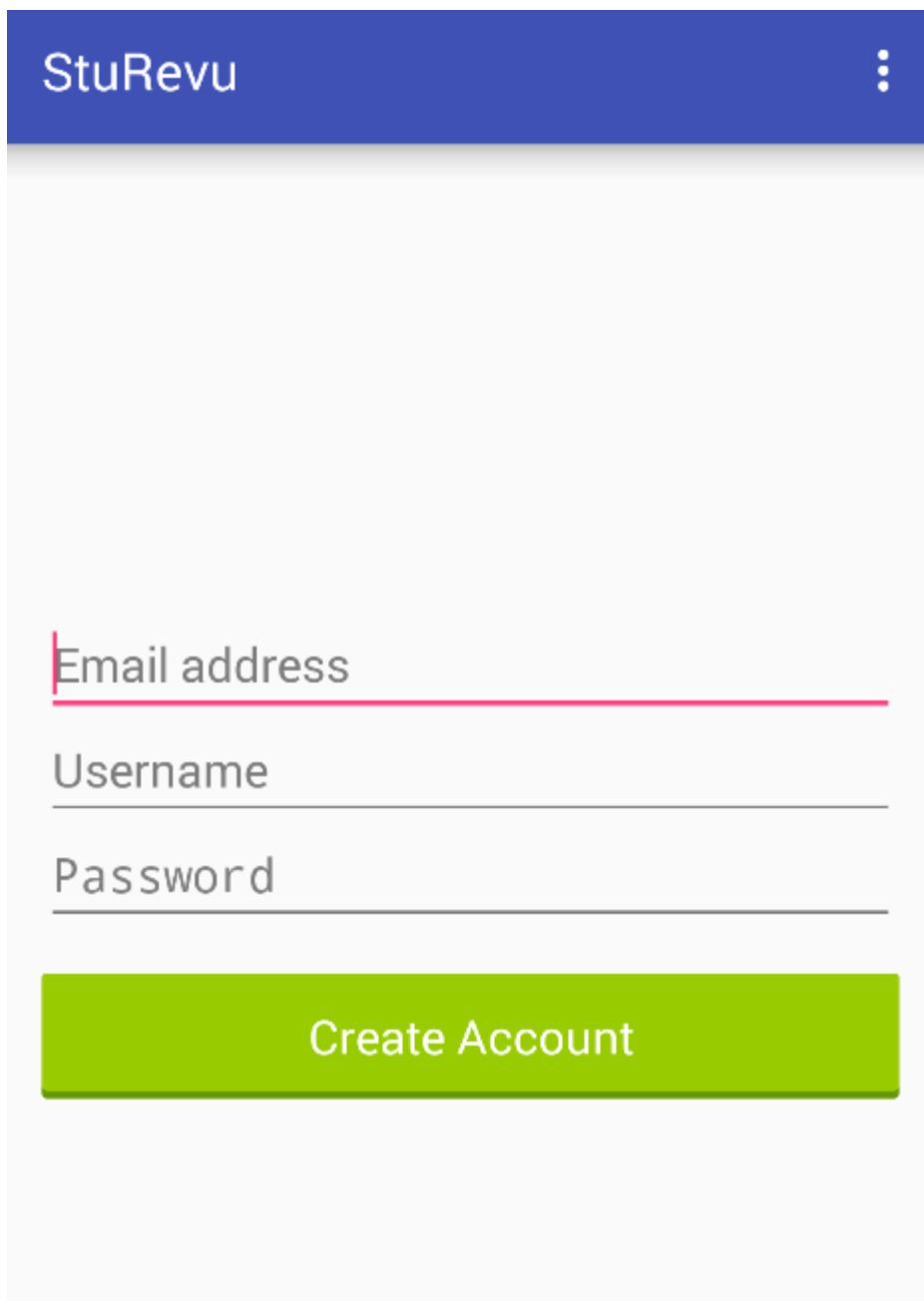


Figure 2-23 Login



A login form interface. At the top is a dark blue button with the text "LOGIN WITH FACEBOOK" in white. Below this are two input fields: the first is labeled "username" with a pink underline, and the second is labeled "password" with a grey underline. At the bottom is a large blue button with the text "Login" in white.

Figure 2-24 Create Account



The image shows a mobile application interface for creating an account. At the top is a blue header bar with the text "StuRevu" on the left and a vertical ellipsis menu icon on the right. Below the header is a light gray background area. The form consists of three input fields: "Email address" with a pink underline, "Username" with a gray underline, and "Password" with a gray underline. Below these fields is a large green button with the text "Create Account" in white.

StuRevu

Email address

Username

Password

Create Account

Figure 2-25 College List



College List	
	Dublin City University <i>University</i>
	Dublin Institute of Technology <i>Institute of Technology</i>
	Dundalk Institute of Technology <i>Institute of Technology</i>
	Institute of Technology Blanchardstown <i>Institute of Technology</i>
	Institute of Technology Tallaght <i>Institute of Technology</i>
	National College of Art and Design <i>University</i>

Figure 2-26 College

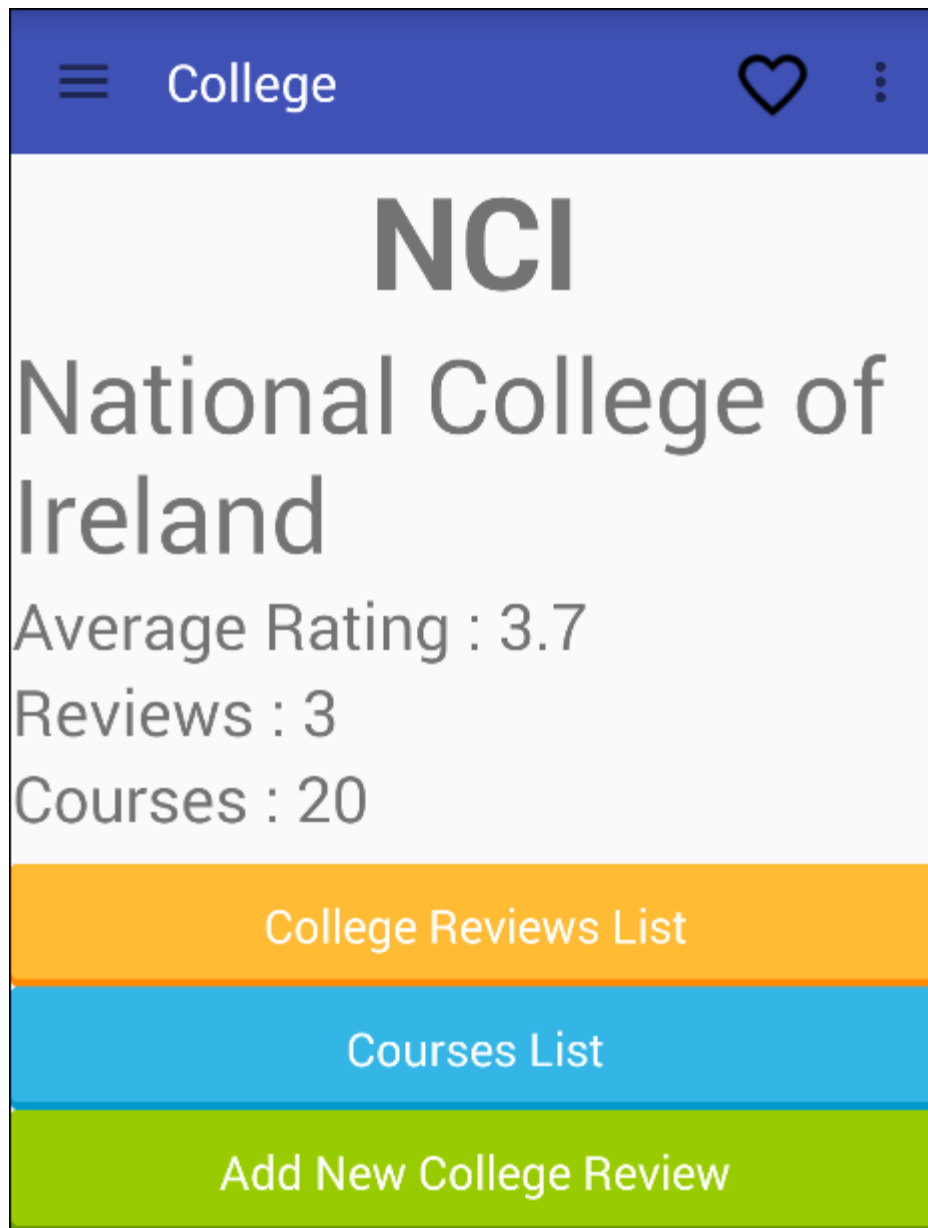





Figure 2-27 College Review List






← College Review List	
Example review title	
Rating: 3	
Date: Fri Dec 04 11:07:39 EST 2015	
test	
Rating: 3	
Date: Fri Dec 11 11:45:59 EST 2015	
central location	
Rating: 5	
Date: Sat Feb 06 08:13:39 EST 2016	

Figure 2-28 College Review

 Review  

central location

Author: *Test*
Reviewed: **NCI**
Student Type: **Part Time Postgraduate**
EU

Rating:     

Pros: **convenient distance to walk from work**
Cons: **luas lines not linked up yet**
Advice: **dont lock bikes outside, unsafe area**

Comments: 0
Helpful Votes : 0
Spam Flags : 0

Comment List

Vote Helpful

free for personal use **Flag Spam**

Figure 2-29 College Review (contd)



Figure 2-30 Course List

Course List	
BA (Hons) in Accounting and Finance	<i>Full Time</i>
Higher Certificate in Business	<i>Full Time</i>
BA (Hons) in Business	<i>Part Time</i>
BA (Hons) in Business	<i>Full Time</i>
Masters in Business Administration (MBA)	<i>Part Time</i>
Higher Certificate in Science in Business Computing	<i>Full Time</i>
BSc (Hons) in Business Information Systems	<i>Full Time</i>
BSc (Hons) in Computing	<i>Full Time</i>
Higher Certificate in Computing Applications and Support	<i>Full Time</i>
MSc in Data Analytics	<i>Full Time</i>
MSc Entrepreneurship	<i>Full Time</i>
MSc in Finance	<i>Full Time</i>
Higher Diploma in Business in Finance	<i>Part Time</i>
BA (Hons) in HRM Strategy and Practice	

Figure 2-31 Course

The image shows a mobile application interface for a course. At the top is a blue header bar with a hamburger menu icon on the left, the word "Course" in the center, a heart icon on the right, and a vertical ellipsis icon on the far right. Below the header, the course title "BA (Hons) in Business" is displayed in a large, bold, dark grey font, followed by "Part Time" in a smaller, italicized, dark grey font. Below the title, several course details are listed in a dark grey font: "CAO Code: n/a", "Qualification: Honours Bachelor of Arts Degree", "NFQ Level: 8", "Duration: 4 years", "Course Level: Undergraduate", "Fees: 3800", and "Department/Faculty: School of Business". Below these details, the "Average Rating : 3.5" and "Reviews : 1" are shown. At the bottom of the screen, there are two large, rounded rectangular buttons: an orange one labeled "Course Reviews List" and a green one labeled "Add New Course Review".

☰ Course ♥ ⋮

BA (Hons) in Business

Part Time

CAO Code: n/a

Qualification: **Honours Bachelor of Arts Degree**

NFQ Level: **8**

Duration: **4 years**

Course Level: **Undergraduate**

Fees: **3800**

Department/Faculty: **School of Business**

Average Rating : 3.5

Reviews : 1

Course Reviews List

Add New Course Review

Figure 2-32 Course Review List

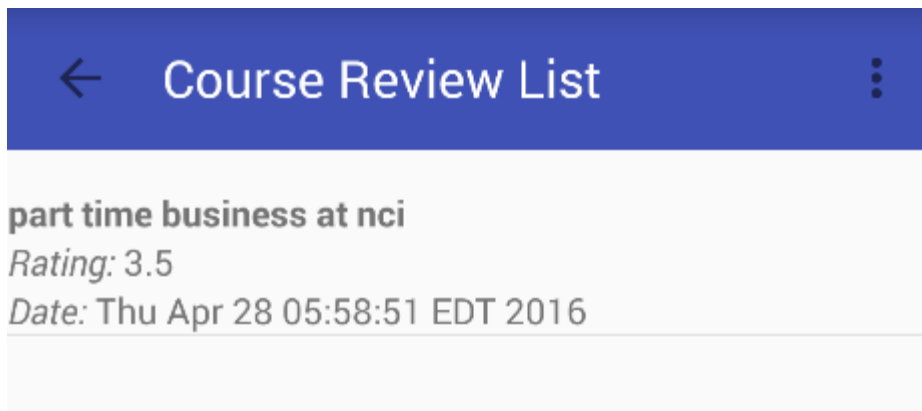





Figure 2-33 Course Review






 Review  

part time business at nci

Author: *userno1*

Reviewed: **BA (Hons) in Business at NCI**

Student Type: **Part Time Undergraduate EU**

Rating:     

Pros: **flexible schedule , good projects and material**

Cons: **exams heavily weighted for grading**

Advice: **learn how to memorise for exams**

Comments: 11

Helpful Votes : 1

Spam Flags : 0

Comment List

Vote Helpful

free for personal use

Figure 2-34 Course Review (contd)

Flag Spam


Add New Comment

free for personal use

Figure 2-35 Add New Review (College/Course)

New Review Title

Rating:



Student Type

Pros:

Cons:

Advice:

Save

Cancel

Figure 2-36 Comment List

← Comment List ↺ ⋮	
better than DBS?	
<i>Date:</i>	Thu Apr 28 06:00:13 EDT 2016
continuous assessments	
<i>Date:</i>	Thu Apr 28 06:10:49 EDT 2016
books required	
<i>Date:</i>	Thu Apr 28 06:14:17 EDT 2016
another comment	
<i>Date:</i>	Thu Apr 28 06:37:42 EDT 2016
push notification test	
<i>Date:</i>	Thu Apr 28 06:45:26 EDT 2016
another test	
<i>Date:</i>	Thu Apr 28 06:55:24 EDT 2016
comment push	
<i>Date:</i>	Thu Apr 28 07:11:17 EDT 2016
testing	
<i>Date:</i>	Thu Apr 28 08:34:06 EDT 2016
new comment test	
<i>Date:</i>	Thu Apr 28 08:38:22 EDT 2016

Figure 2-37 Comment

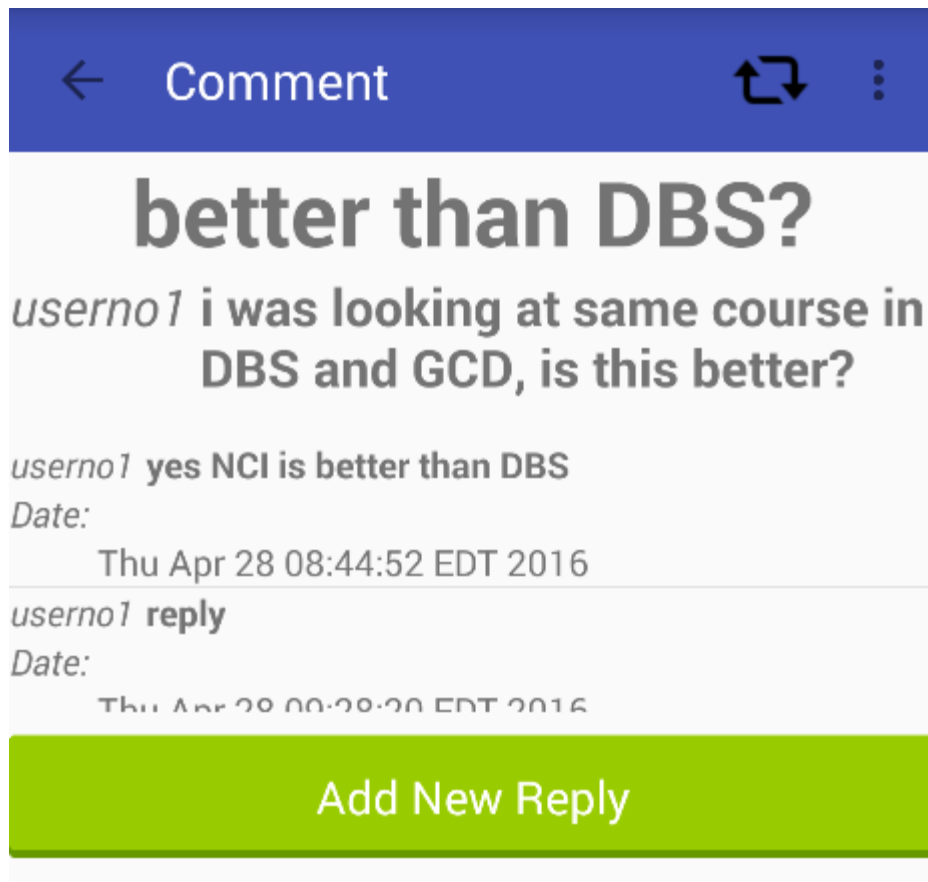
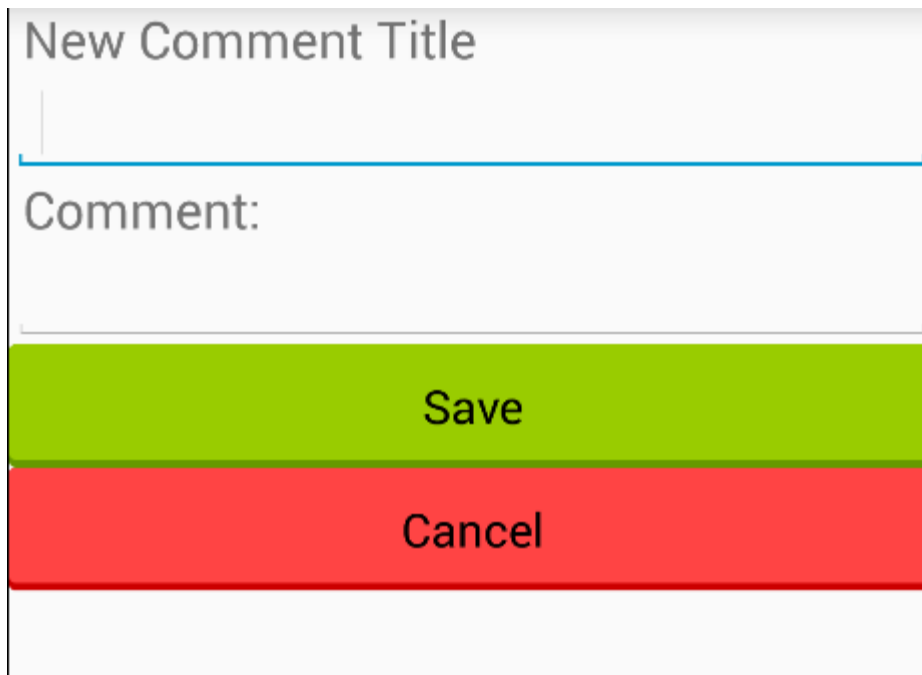


Figure 2-38 Add new Comment



The image shows a form for adding a new comment. It consists of a light gray container with a white background. At the top, there is a text input field with the placeholder text "New Comment Title". Below this is another text input field with the placeholder text "Comment:". At the bottom of the form, there are two buttons: a green "Save" button and a red "Cancel" button. The buttons are stacked vertically and have a slight shadow effect.

New Comment Title

Comment:

Save

Cancel

Figure 2-39 Drawer Navigation Menu

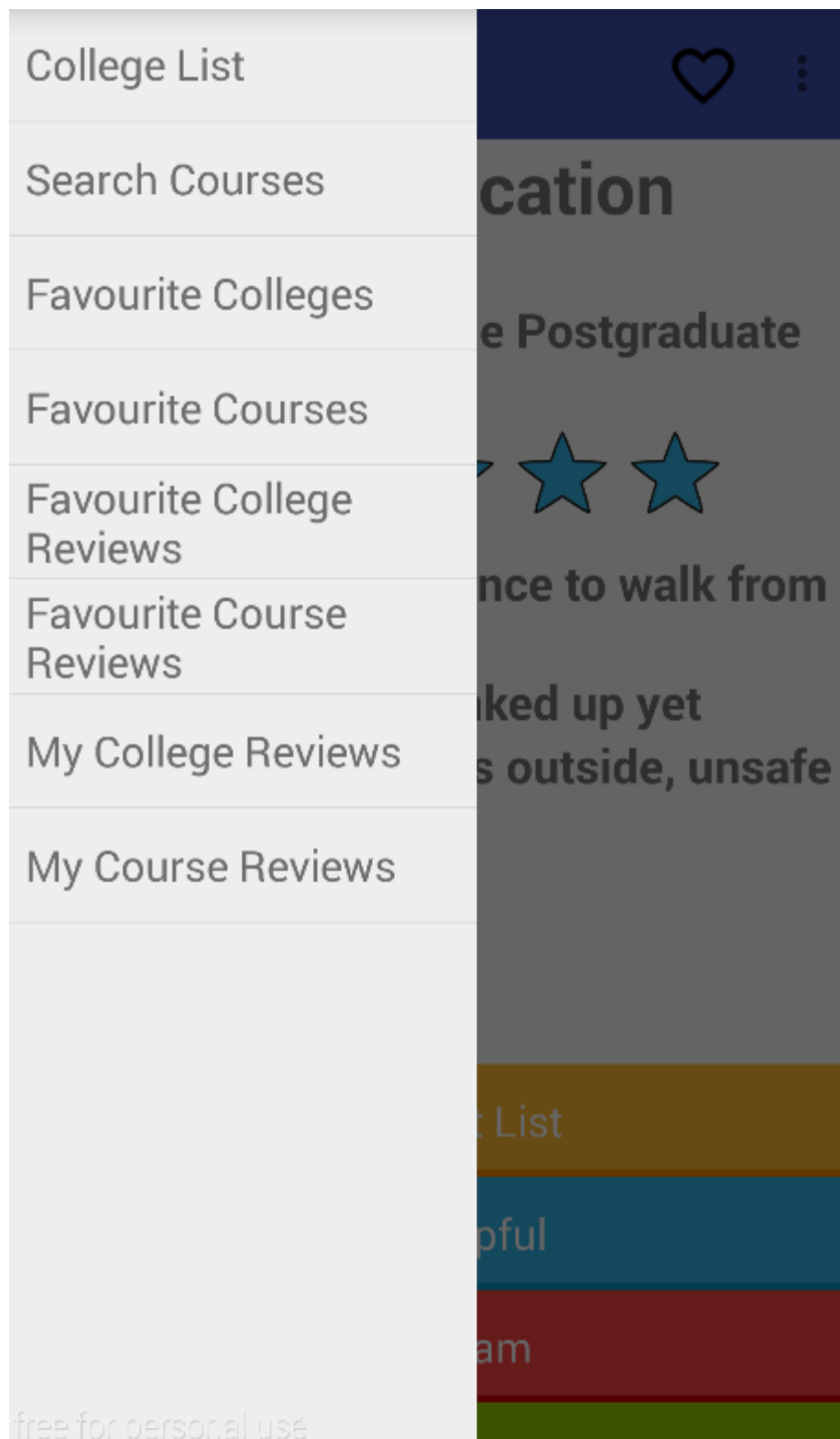


Figure 2-40 Search Courses

←

Search Courses

⋮

Bus

Search

Higher Certificate in Business

NCI

Full Time

BA (Hons) in Business

NCI

Part Time

BA (Hons) in Business

ITB

Full Time

BA (Hons) in Business

NCI

Full Time

Masters in Business Administration (MBA)

NCI

Part Time

Higher Certificate in Science in Business Computing

NCI

Full Time

BSc (Hons) in Business Information Systems

NCI

Full Time

Figure 2-41 Favourite Colleges

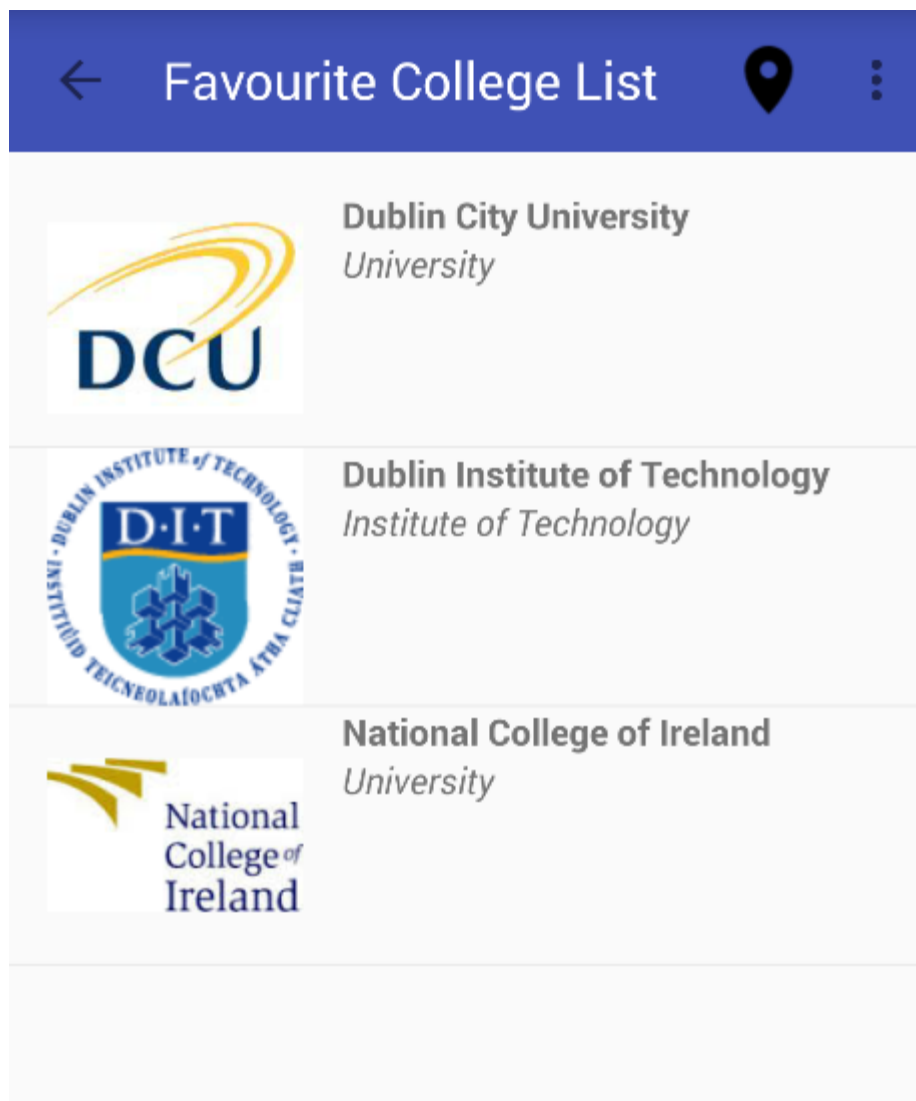


Figure 2-42 Favourite Courses

← Favourite Course List 🔍 ⋮	
Higher Certificate in Computing Applications and Support	
NCI	
Full Time	
Higher Certificate in Business	
NCI	
Full Time	
BSc (Hons) in Computer Applications	
DCU	
Full Time	

Figure 2-43 Favourite College Reviews

← Favourite College Review L... ⋮	
test dcu college review	
DCU	
Rating: 3	
Date: Fri Dec 11 10:26:41 EST 2015	
MSC Environmental Science	
TCD	
Rating: 2.5	
Date: Wed Feb 03 19:33:21 EST 2016	

Figure 2-44 Favourite Course Reviews

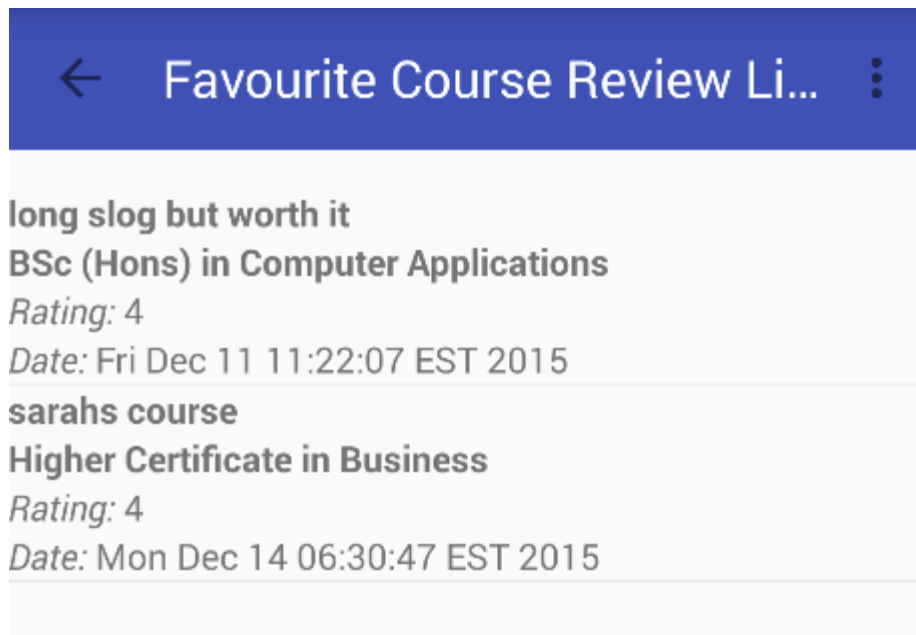


Figure 2-45 My College Reviews

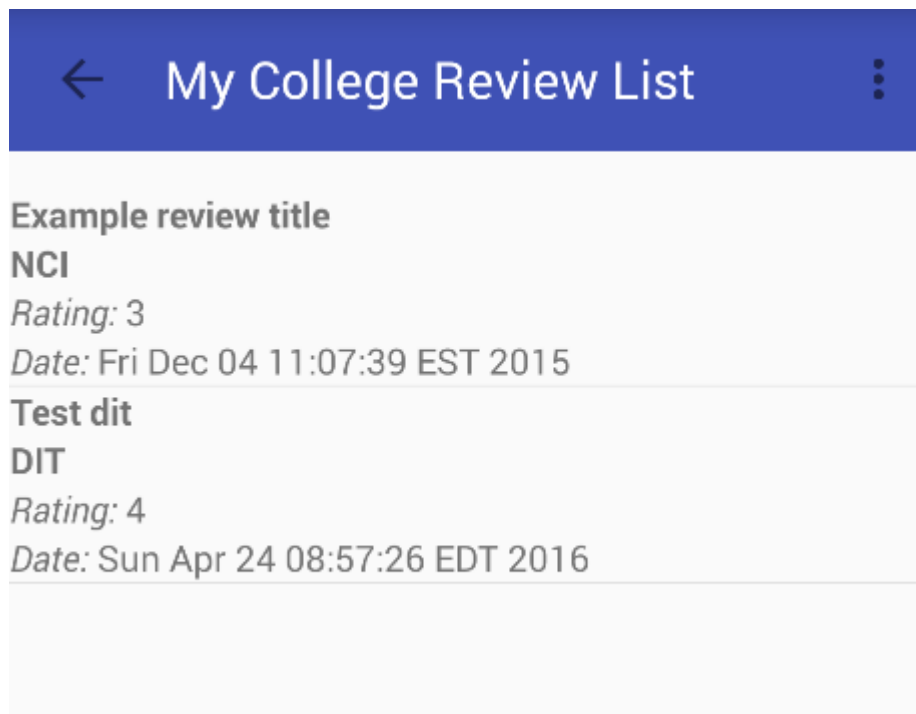


Figure 2-46 My Course Reviews

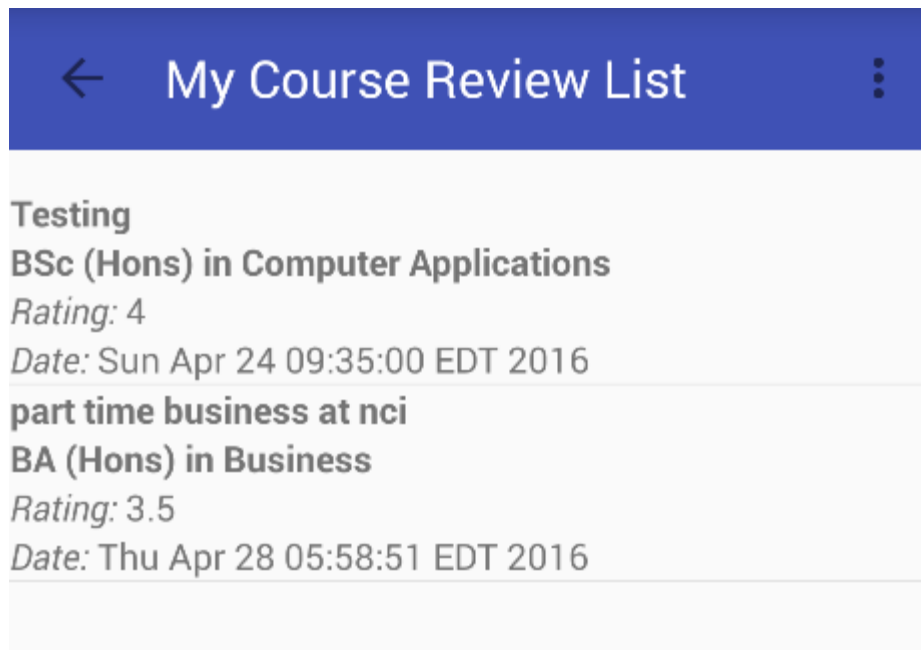


Figure 2-47 Favourite College New Review Push Notification

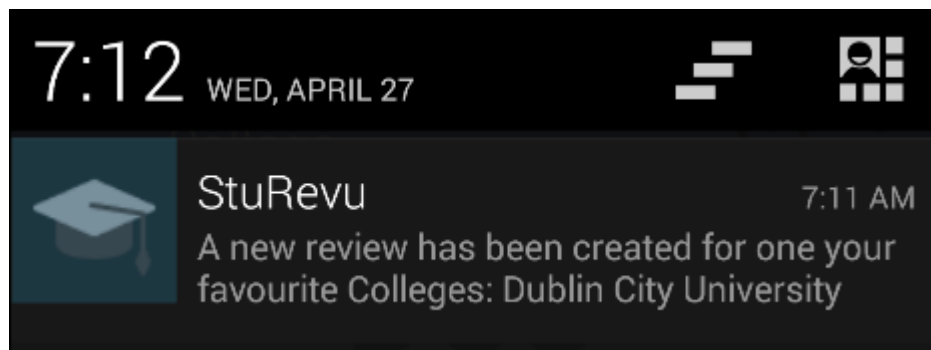


Figure 2-48 Favourite Course New Review Push Notification

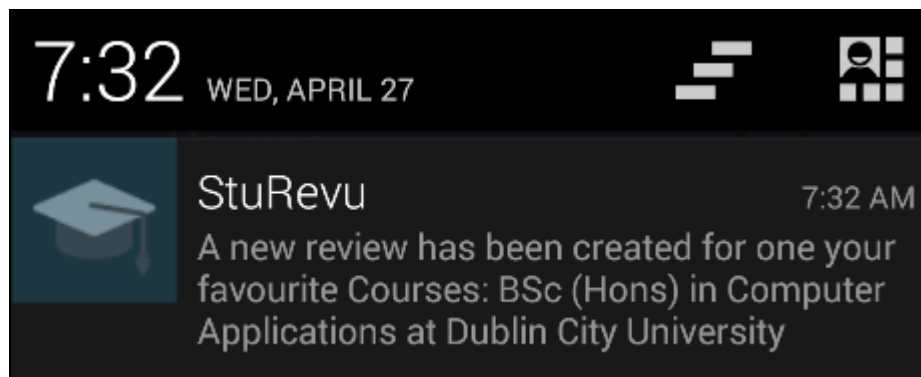


Figure 2-49 Favourite College Review New Comment Push Notification

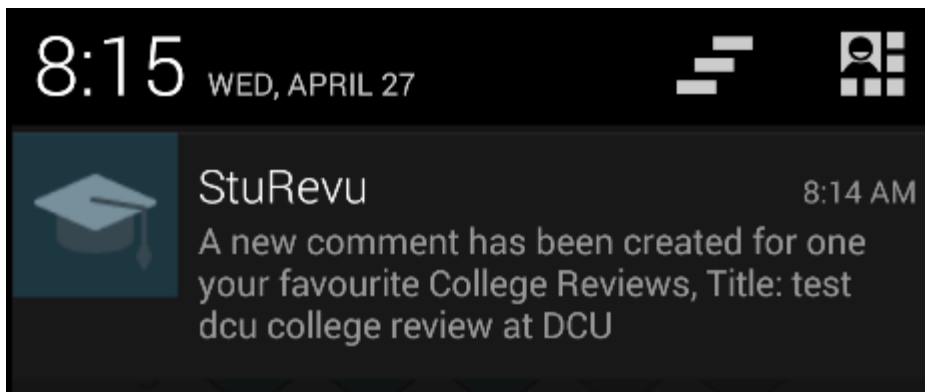


Figure 2-50 Favourite Course Review New Comment Push Notification

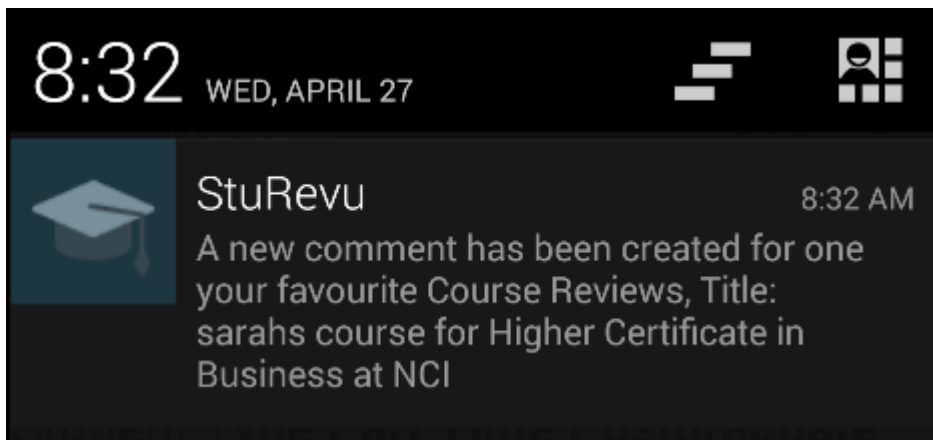


Figure 2-51 My College Review New Comment Push Notification

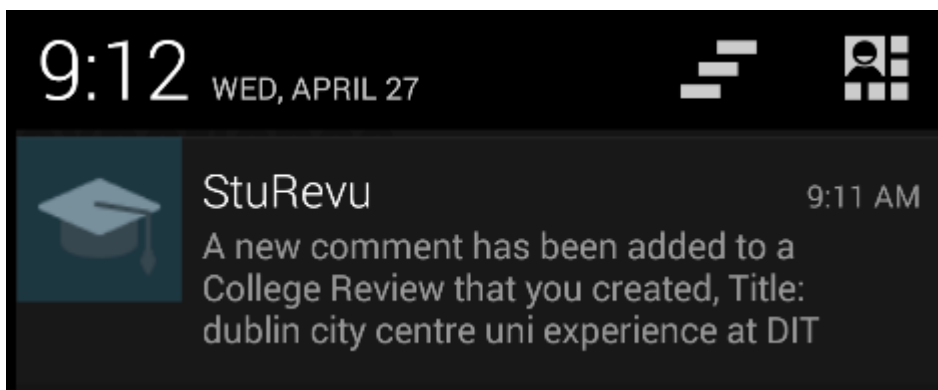


Figure 2-52 My Course Review New Comment Push Notification

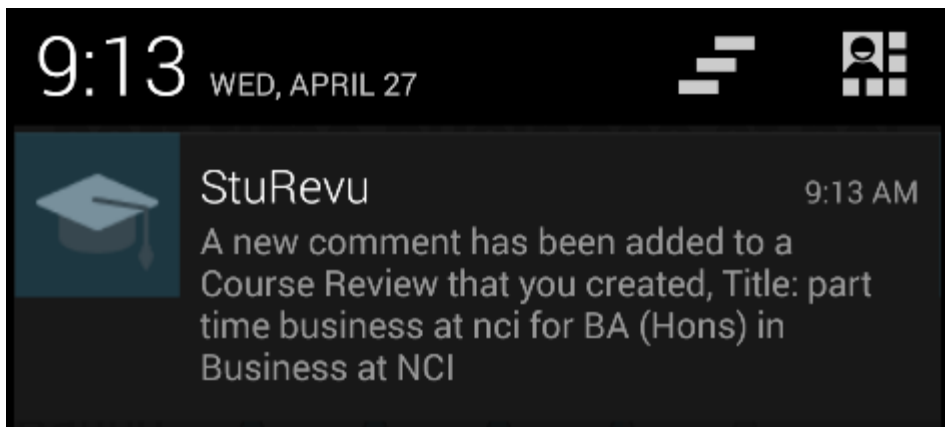


Figure 2-53 My Comment New Reply Push Notification

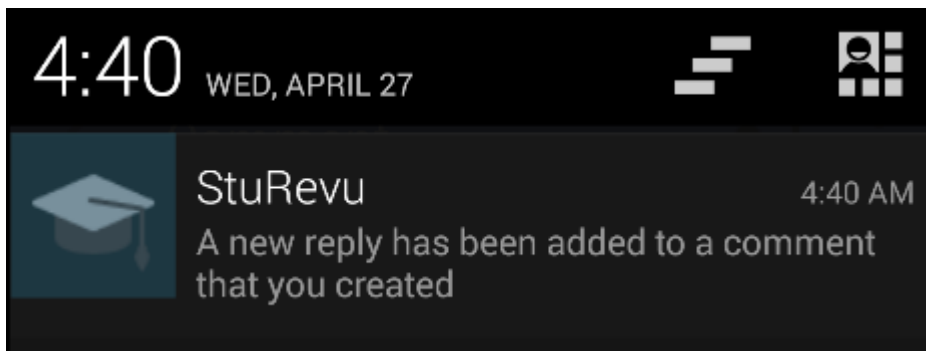


Figure 2-54 Facebook Login (not logged in yet)

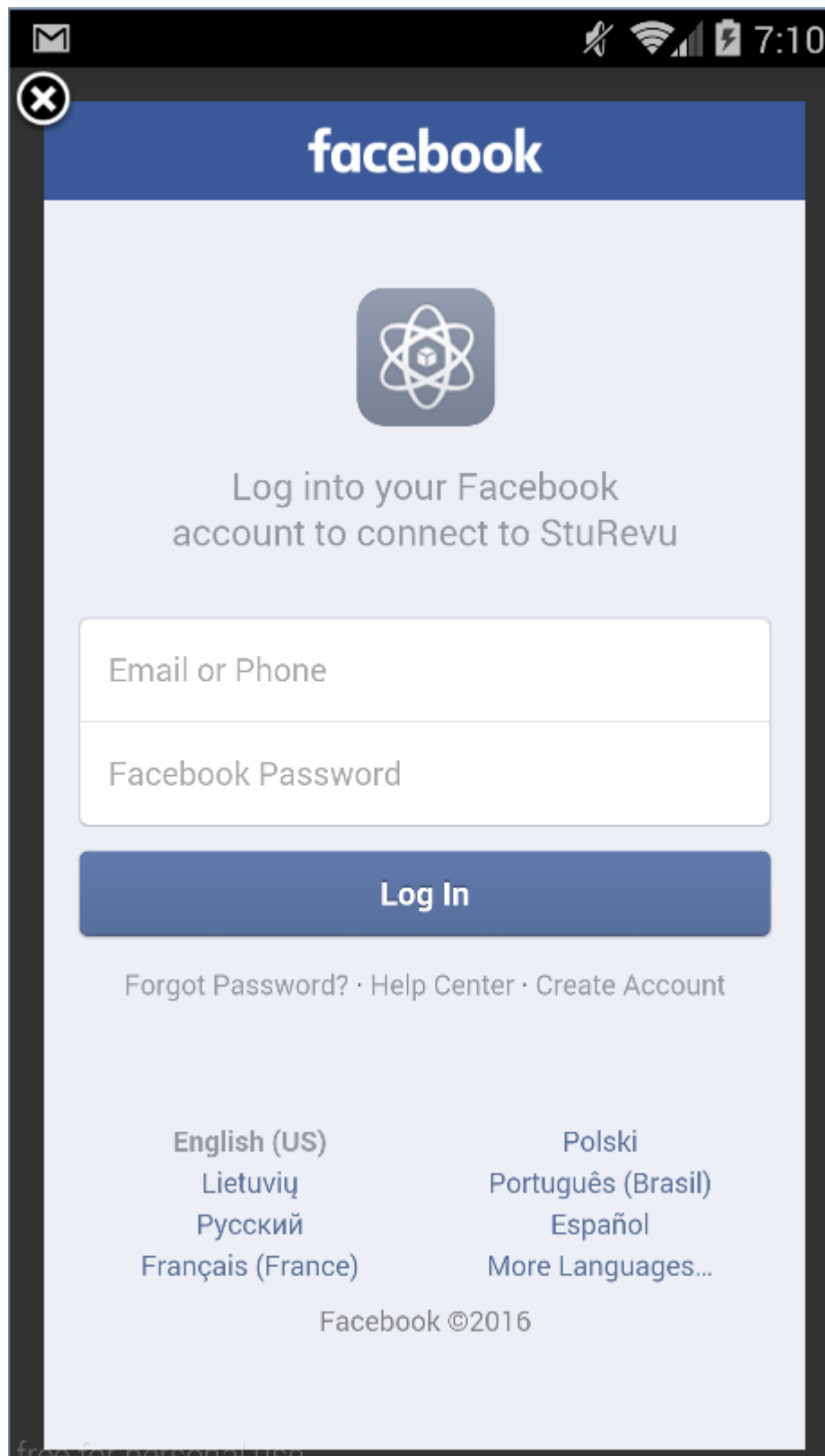
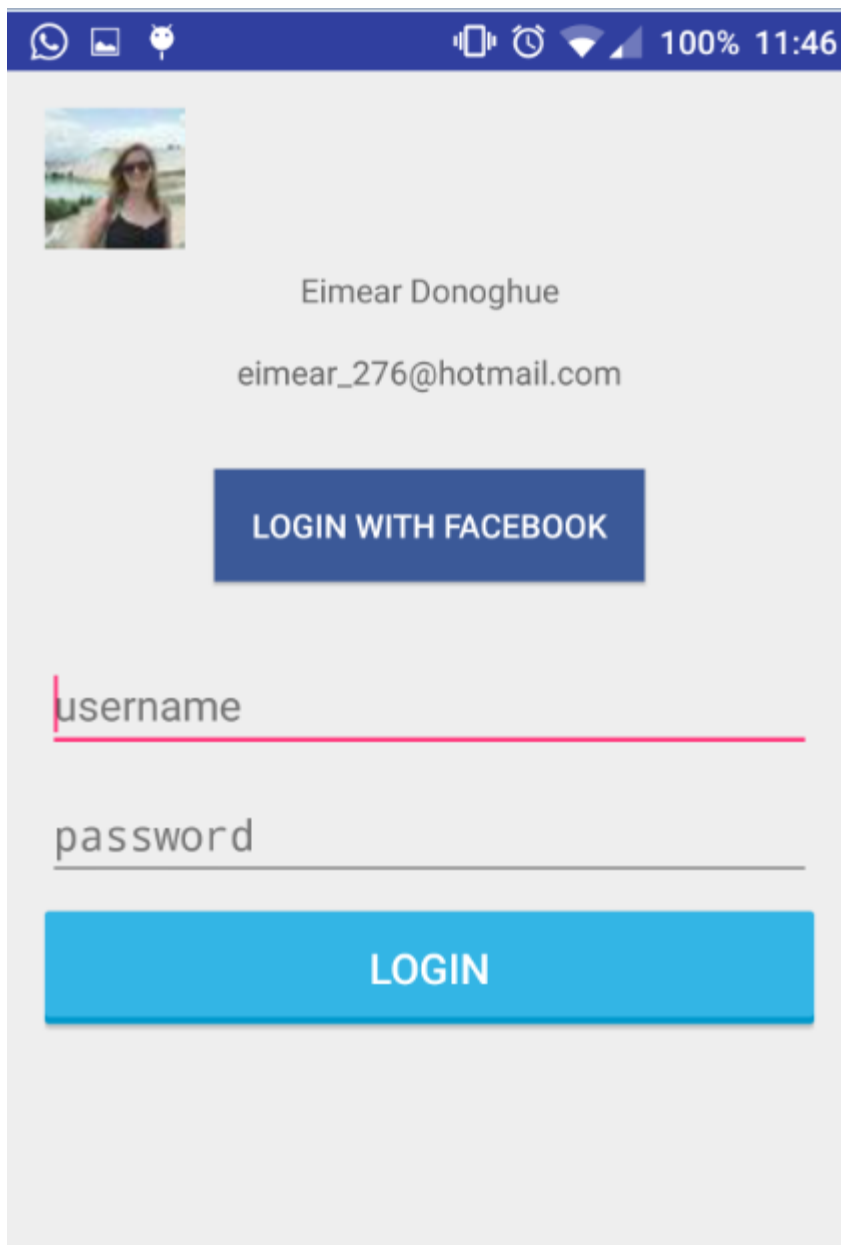


Figure 2-55 Facebook Login (logged in details displaying)



The image shows a mobile application interface for Facebook login. At the top is a dark blue status bar with icons for WhatsApp, a gallery, a robot, a mobile phone, an alarm, Wi-Fi, and battery status at 100% with the time 11:46. Below this is a light gray background. On the left is a square profile picture of a woman with sunglasses. To the right of the picture, the name 'Eimear Donoghue' is displayed in a medium gray font, followed by the email address 'eimear_276@hotmail.com' in a smaller gray font. Below the email is a dark blue rectangular button with the text 'LOGIN WITH FACEBOOK' in white. Further down are two text input fields. The first is labeled 'username' in a light gray font, with a pink underline and a pink cursor at the start. The second is labeled 'password' in a light gray font, with a gray underline. At the bottom is a large blue rectangular button with the text 'LOGIN' in white.

Eimear Donoghue
eimear_276@hotmail.com

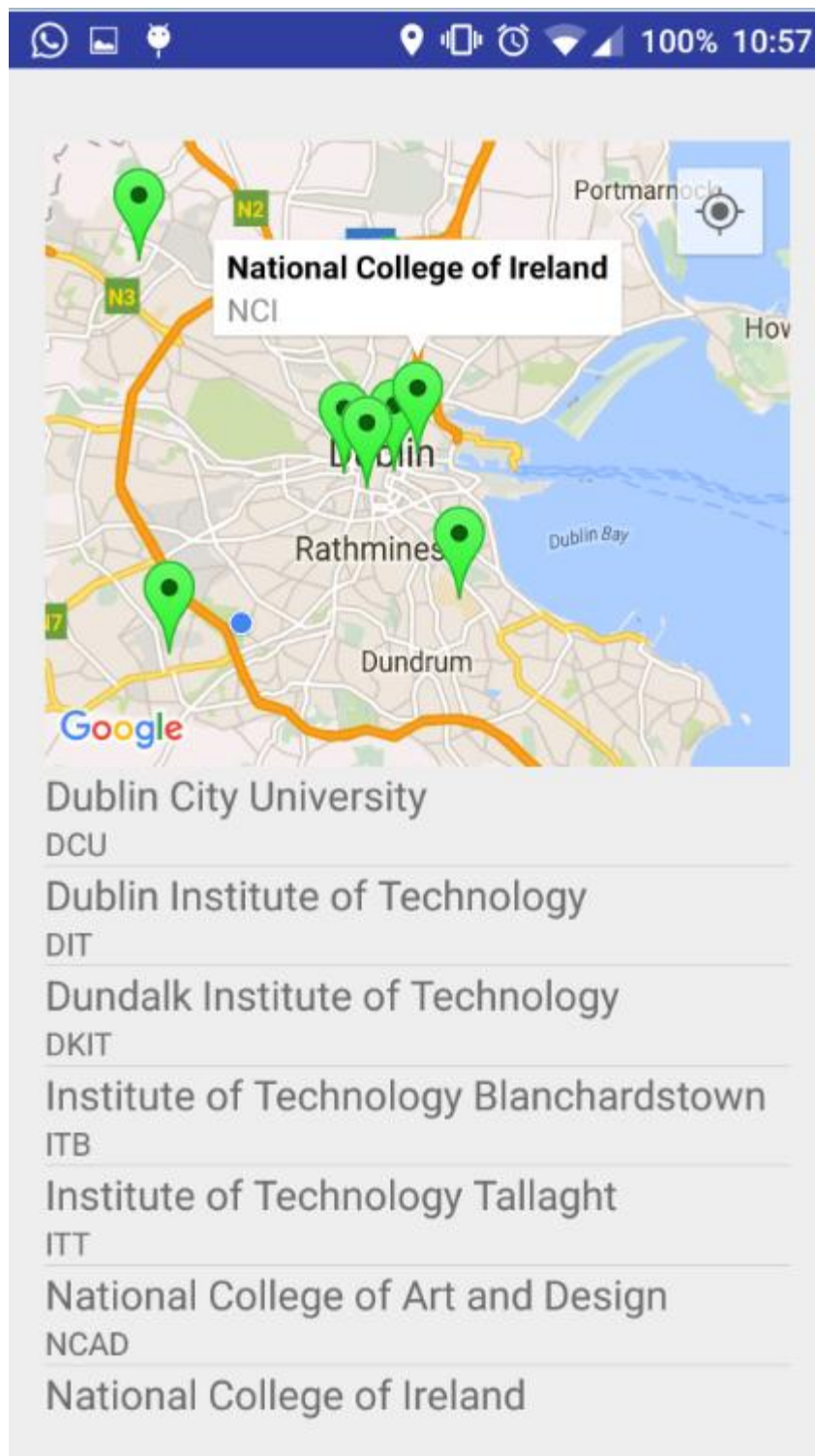
LOGIN WITH FACEBOOK

username

password

LOGIN

Figure 2-56 Google Map of Colleges



3 Conclusion

The inspiration for StuRevu came from Glassdoor, a review app that allows employees to share information about companies, salaries and interviews. The application that I have developed has the potential to bring insider reviews to the education market.

The Course and College data is time consuming to insert manually into the database. To achieve scalability a web scraper or web service integration would be required to get data from an external source in an automated process. Alternatively a College Admin web interface login could be implemented to allow self-service in bulk creation and maintenance of College/Course static data.

Following on from this and after analysis of the requirements specification it was decided not to include Clubs & Societies and Modules in the scope of this project. Login with LinkedIn was also not implemented. These features could however be added later if required or desired by users.

The use of a noSQL database on the backend should ensure scalability for fast read and write operations. The use of Parse backend service would become cost inefficient at a certain point of growth in users and traffic but for development and testing purposes it remains free and provides excellent features including push notifications and analytics.

Unfortunately Parse recently announced end of life of its service will be January 2017. For this App to go Live and remain available after that date a migration is required from Parse hosted platform to MongoDB and Parse Server (both Open Source and free) on AWS or similar cloud infrastructure providers.

4 Further development or research

To turn this project into a commercial venture;

Step 1: Incentivise graduates and current students to create reviews to generate the value.

This could involve a partnership with the likes of GradIreland to gain access to the email marketing databases of current students and graduates to collect reviews in exchange for entry to a raffle to win a tablet or smartphone for example.

Step 2: Market/Advertise the App to prospective students to grow the user base.

This could involve visits to secondary schools, social media campaigns, TV & Radio exposure and blogs to promote the App to prospective higher education students.

Step 3: Implement advertising revenue model.

The App could utilise google AdMob or allow third level institutions to post paid adverts for open course applications.

5 References

AndroidBegin. (2013). *Android Parse.com Simple ListView Tutorial - AndroidBegin*. [online] Available at: <http://www.androidbegin.com/tutorial/android-parse-com-simple-listview-tutorial/> [Accessed 10 May 2016].

Developer.android.com. (2016). *Getting Started | Android Developers*. [online] Available at: <https://developer.android.com/training/index.html> [Accessed 10 May 2016].

Facebook Developers. (2016). *Getting Started - Android SDK - Documentation - Facebook for Developers*. [online] Available at: <https://developers.facebook.com/docs/android/getting-started> [Accessed 10 May 2016].

Gist. (2016). *A Parse.com Serializable ParseObject Proxy*. [online] Available at: <https://gist.github.com/jamiechapman/5375115> [Accessed 10 May 2016].

GitHub. (2016). *codepath/android_guides*. [online] Available at: https://github.com/codepath/android_guides/wiki/Using-the-App-ToolBar [Accessed 10 May 2016].

Google Developers. (2016). *Getting Started*. [online] Available at: <https://developers.google.com/maps/documentation/android-api/start> [Accessed 10 May 2016].

Grafix Artist. (2015). *Integrate Facebook Login with Parse - Part 2*. [online] Available at: <http://blog.grafixartist.com/facebook-login-with-parse-part-2/> [Accessed 10 May 2016].

Inthecheesefactory.com. (2016). *How to install Google Play Services on Genymotion Step by Step*. [online] Available at: <https://inthecheesefactory.com/blog/how-to-install-google-services-on-genymotion/en> [Accessed 10 May 2016].

Parse.com. (2016). *Parse Android Anywall Tutorial*. [online] Available at: <https://www.parse.com/tutorials/anywall-android> [Accessed 10 May 2016].

Parse.com. (2016). *Parse Android Guide*. [online] Available at: <https://parse.com/docs/android/guide> [Accessed 10 May 2016].

Parse.com. (2016). *Parse Android Mealspotting Tutorial*. [online] Available at: <https://www.parse.com/tutorials/mealspotting> [Accessed 10 May 2016].

Parse.com. (2016). *Parse Query Adapter Tutorial*. [online] Available at: <https://parse.com/tutorials/parse-query-adapter> [Accessed 10 May 2016].

Parse.com. (2016). *Parse Questions*. [online] Available at: <https://parse.com/questions/> [Accessed 10 May 2016].

Stackoverflow.com. (2016). *Newest parse.com Questions*. [online] Available at: <http://stackoverflow.com/questions/tagged/parse.com> [Accessed 10 May 2016].

The Irish Times. (2016). *Computer science graduates in demand, but arts earn least*. [online] Available at: <http://www.irishtimes.com/news/education/computer-science-graduates-in-demand-but-arts-earn-least-1.2632396> [Accessed 10 May 2016].

The Irish Times. (2016). *Concern over drop-out rates in computer science courses*. [online] Available at: <http://www.irishtimes.com/news/education/concern-over-drop-out-rates-in-computer-science-courses-1.2491751> [Accessed 10 May 2016].

Treehouse Blog. (2015). *How to Add a Navigation Drawer in Android - Treehouse Blog*. [online] Available at: <http://blog.teamtreehouse.com/add-navigation-drawer-android> [Accessed 10 May 2016].

www.tutorialspoint.com. (2016). *Android Tutorial*. [online] Available at: <http://www.tutorialspoint.com/android/> [Accessed 10 May 2016].

6 Appendix

6.1 *Project Proposal*

Project Proposal

ClassDoor

Jordan Daly, 13105272, jordan.daly@student.ncirl.ie

BSc (Hons) in Computing

Specialisation - Networking and Mobile Technologies

2nd Oct 2015

1. Objectives

The objective of this project is to create an education review Android App for students.

My idea is to create a peer recommendation app similar to [Glassdoor](#) / [Tripadvisor](#) / [Yelp](#) for prospective students. I believe that the an information sharing application for students would help them to make more informed decisions about which college, course, modules and clubs/societies to choose.

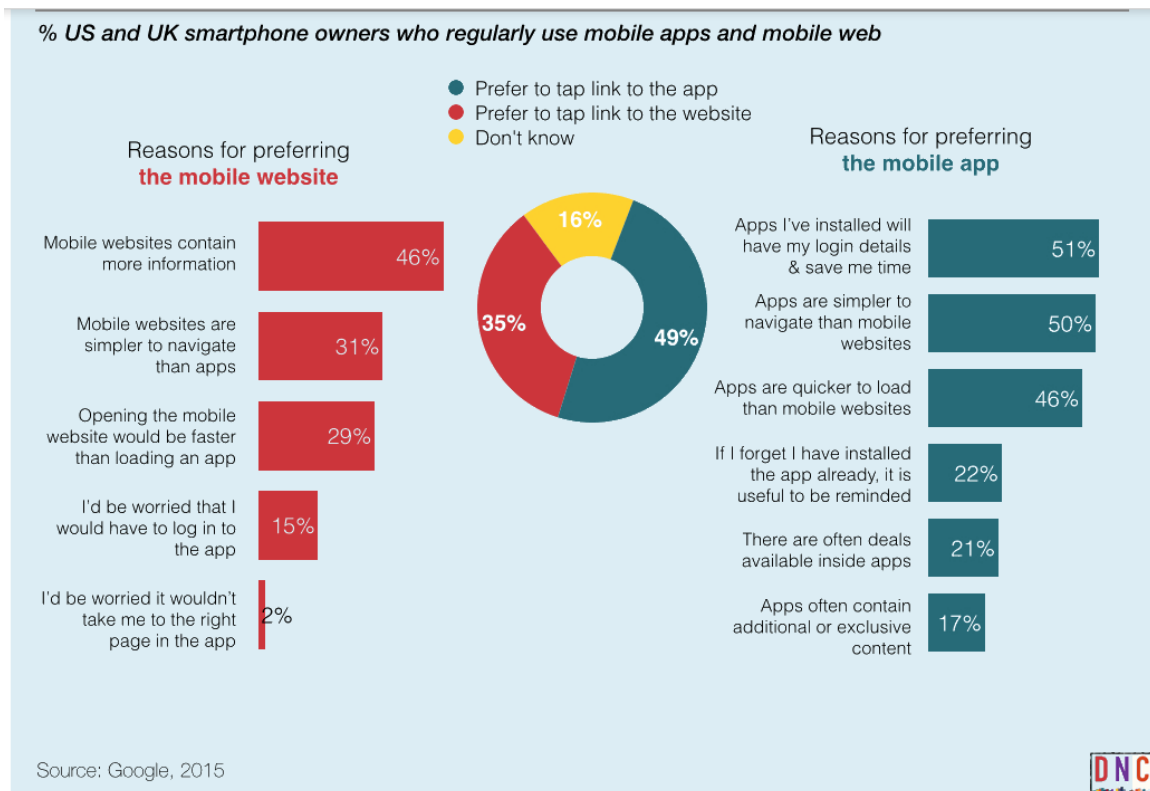
Technical Approach

Brief description of the approach to be followed (Max. 1 Page), Research, literature review, requirements capture, implementation etc...

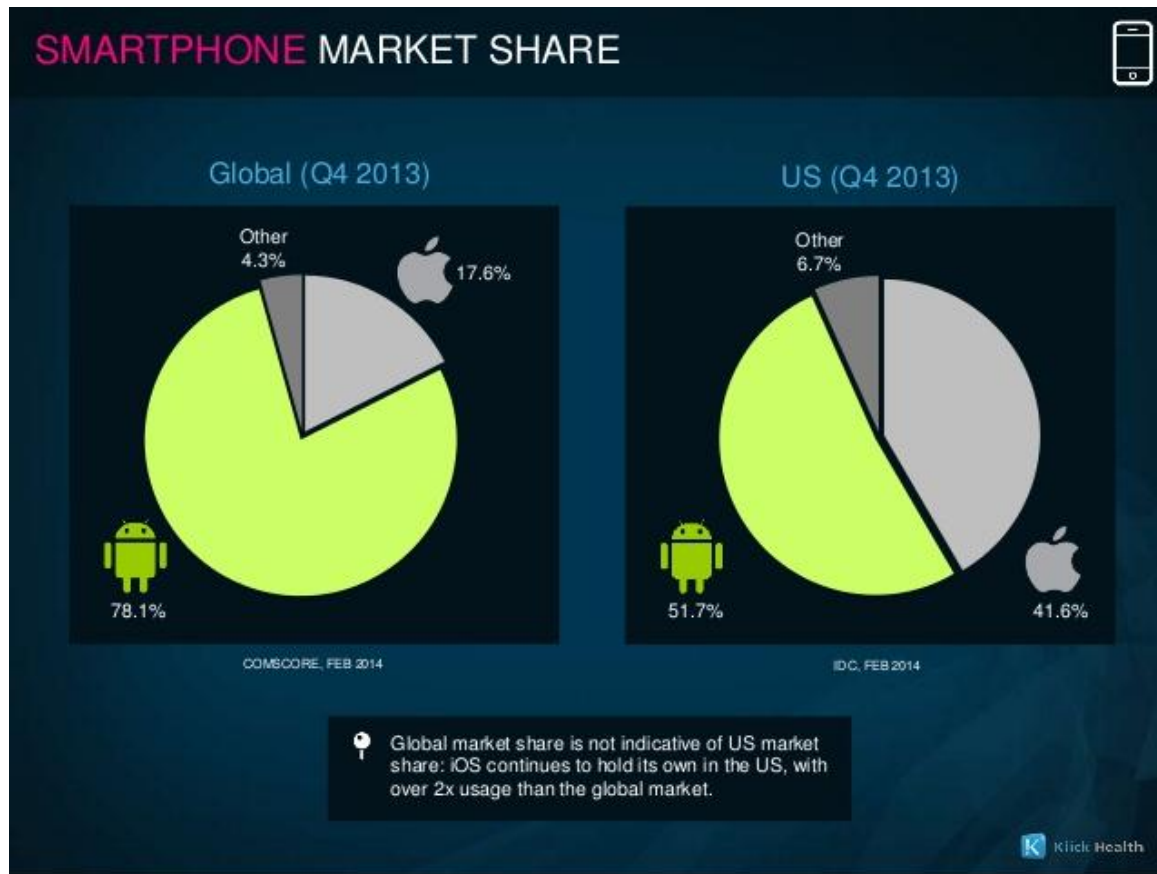
Research

I found that something similar was already existed online in Ireland (a web application for college and course reviews), called [coursehub](#) so I realised that the next step is a native Mobile Application. Currently there are no standout leaders in this area on the Google Play Store when searching Android Apps.

CourseHub was developed in 2011 and at the time received awards for it's design. While a mobile site would suffice for such an application, consumer research has indicated a preference for mobile apps as opposed to mobile websites.



I plan to develop an Android App because ClassDoor's target market will be the consumer market so I want to allow access to as many users as possible. Android is the mobile operating system on the majority of smartphones globally at almost three quarters share and around 50% in the U.S.



Literature Review

<http://www.usnews.com/education/blogs/college-rankings-blog/2014/03/13/freshmen-cite-reputation-cost-factors-in-college-choice-survey>

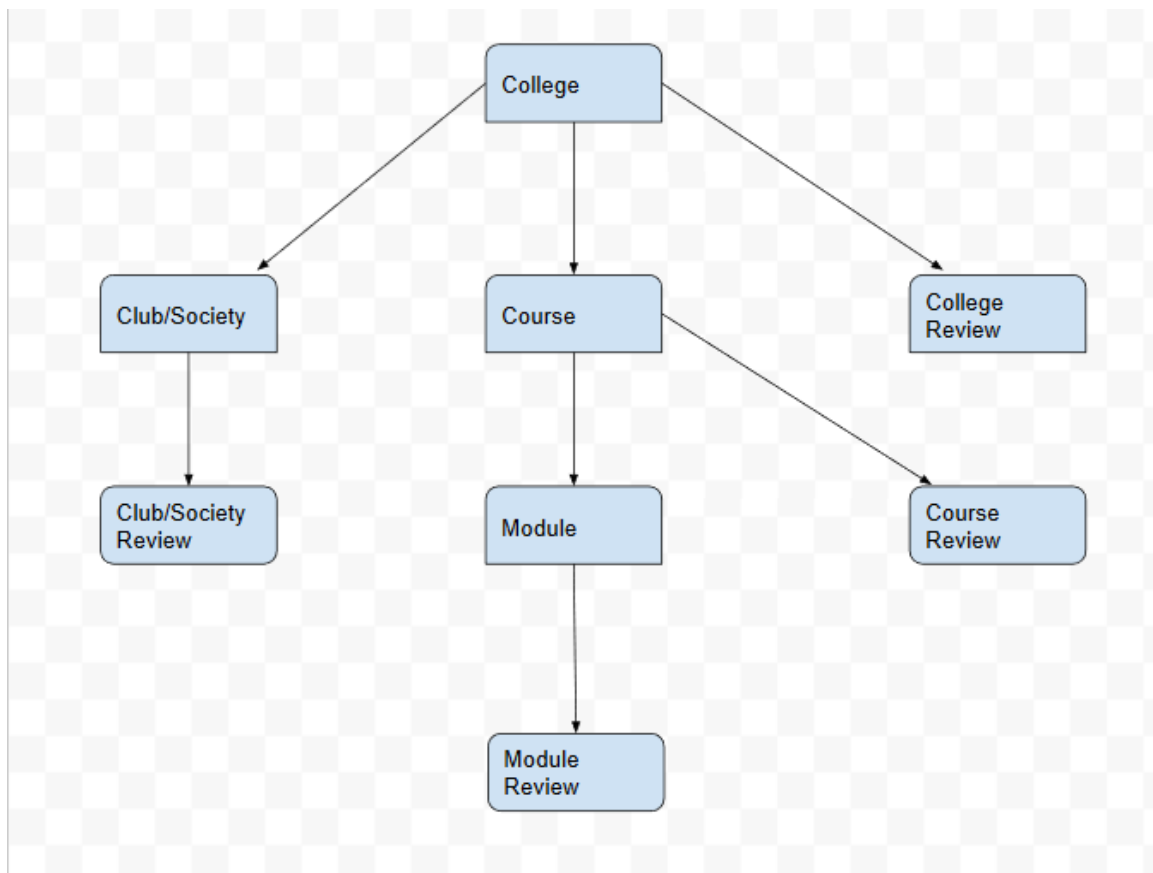
A number of factors have been seen to affect college choice decision making with academic reputation and graduate employability number one and two respectively.

1. College has very good academic reputation (64.0 percent)
2. This college's graduates get good jobs (53.1 percent)
3. I was offered financial assistance (48.7 percent)
4. The cost of attending this college (45.9 percent)
5. College has a good reputation for its social activities (44.1 percent)
6. A visit to this campus (42.9 percent)
7. Wanted to go to a college about this size (37.6 percent)

8. College's grads get into top grad/professional schools (33.0 percent)
9. The percentage of students that graduate from this college (29.7 percent)
10. I wanted to live near home (19.6 percent)
11. Information from a website (18.3 percent)
12. Rankings in national magazines (17.6 percent)
12. Parents wanted me to go to this school (17.6 percent)
14. Could not afford first choice (14.9 percent)
15. Admitted early decision and/or early action (14.3 percent)
16. Not offered aid by first choice (10.9 percent)
17. High school counselor advised me (10.3 percent)
18. Athletic department recruited me (9.4 percent)
19. My relatives wanted me to come here (8.4 percent)
20. Attracted by the religious affiliation/orientation of college (8.3 percent)
20. My teacher advised me (7.3 percent)
22. Private college counselor advised me (4.5 percent)
23. Ability to take online courses (3.8 percent)

Requirements Capture

The high level requirements are listed below and the application architecture is depicted in the diagram below with each entity representing a class/database table. Colleges and courses are the minimum degree of granularity I am planning to implement and I may incorporate clubs and society information into the college review form/table and do the same for courses and modules.



Login with facebook/linkedin/twitter or some other social authentication api

colleges

(search/filter)

create/read/filter reviews/ratings

add/view *photos*

- clubs/societies

(search/filter)

create/read/filter reviews/ratings

add/view *photos*

-- courses

(search/filter)

create/read/filter reviews/ratings

---- modules

(search/filter)

create/read/filter reviews

2. Special resources required

Books

-Learn Android Studio

Build Android Apps Quickly and Effectively

Online resources

<https://parse.com/docs/android/guide>

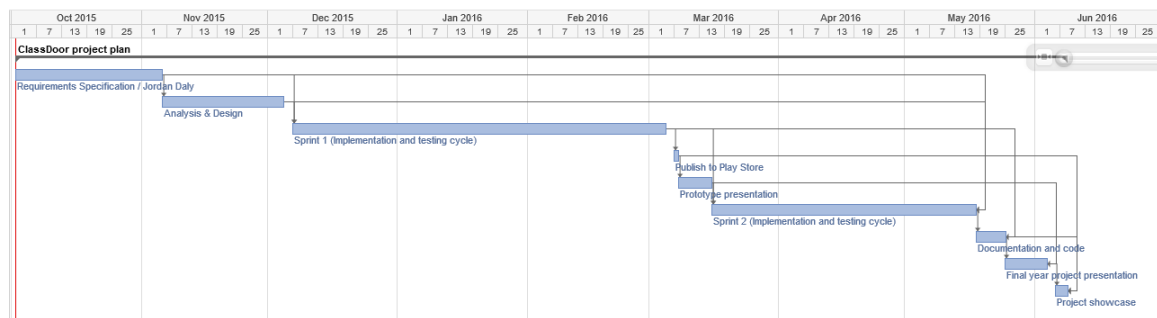
<https://developer.android.com/design/material/index.html>

<https://developer.android.com/training/index.html>

<http://www.tutorialspoint.com/android/>

<https://developers.facebook.com/docs/android/getting-started>

3. Project Plan



4. Technical Details

Implementation language and principal libraries

- IDE - Android Studio, Parse for Android
- back end and database MongoDB (used by Parse MBaaS)
- front end - Android Studio, google material design, Bootstrap responsive
- version control - Github
- testing - android in built framework JUnit

5. Evaluation

Describe how you will evaluate the system with real technical data using system tests, integration tests etc. In addition, where possible describe how you will evaluate the system with an end user.

I will evaluate the system by leveraging the Android Testing Support Library

This library provides a set of APIs that allow you to build and run test code for your apps. The Android Testing Support Library includes the following test automation tools:

AndroidJUnitRunner: JUnit 4-compatible test runner for Android

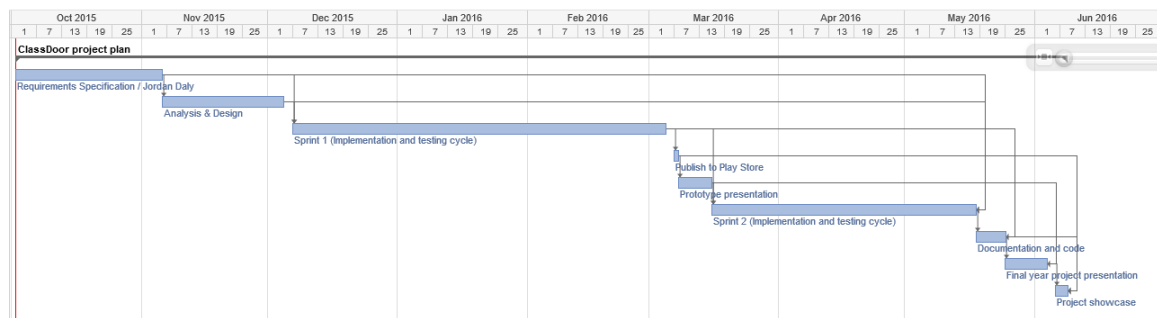
Espresso: UI testing framework; suitable for functional UI testing within an app

I will also ask friends, family and colleagues to manually test the app by downloading it onto their devices and providing feedback on performance, functionality and user experience.

Jordan Daly

Oct 2nd 2015

6.2 Project Plan



6.3 Monthly Journals

6.3.1 September

My name is Jordan, I'm 27 and working full time as a software support analyst for a global company which makes student information systems. The SIS that I support (Quercus) is built on Oracle technology.

Originally I wanted to do a project that was related to my company's product to help me get a move into the development team in work but the only ideas I had were not suitable.

The first was an assessment results integration piece between moodle and Quercus; turned out to be fairly complicated and required mostly php development which I have no knowledge or experience with.

The second idea arose from a real problem I see in my day to day job; the client's oracle databases are filling up too quickly with application trace data (logging info) which is degrading performance and causing them to switch off or greatly reduce the amount of trace they allow to be produced. Trace data is used for debugging application level issues and without it certain issues can't really be investigated.

So a solution is required to store the trace data outside the oracle database, perhaps in a noSQL database on AWS. After exploring this idea I realise it's beyond me because of the lack of examples/resources out there and (most importantly) it's not really a software project it's more of an infrastructure/DevOps project.

So I have decided to go with something I will enjoy instead. My idea is to create a [Glassdoor](#) for students, and of course after a quick google I found that something similar was already existed, called [coursehub](#) so I realised I need to make it better or different in some way.

I plan to make it better by delivering on mobile devices and by designing the interface to be attractive and intuitive.

I created a simple ruby on rails app in 3rd year which I enjoyed so I feel confident enough to learn a new technology in a short space of time once again.

I did some research and started making a list of useful links in my google drive project folder.

Advised by colleagues to look into parse and udemy. Still not sure about the name yet, StuRevu is too generic so going for "ClassDoor" as it has a certain ring to it. Hope I don't regret the decision.

Started to fantasize about leaving my day job to work for myself. Decided that the revenue model will be from paid ads that are posted by a college to advertise courses that are open for application.

6.3.2 October

Proposal done, and afterwards realised that it's not a great idea from a business perspective (SWOT) as there is a huge risk of glassdoor themselves suddenly deciding to target students as well as employees and then ClassDoor is rendered redundant.... possibly need to rethink the name to something more intuitive, StuRevU for example.

Perhaps if there is functionality to allow prospective students to contact graduates for more info it would provide a USP/differentiation. Also considered offering some incentive to verified graduates for their reviews to create quality content.

Once there is a decent user base one strategy would be to offer colleges a way to advertise their chosen courses that are open for application, however there may be a huge conflict of interest between ensuring user content trust (no unfair censorship of reviews due to requests from paying advertisers) and pleasing revenue sources. A less problematic idea may be just to allow ads relevant to the demographic of users in general and not specifically targeting educational institutions.

To ensure users create content and don't just read only, there needs to be a give to get model and/or a give to turn off ads model.

How do you filter user generated content to avoid abuse, is that a huge issue to be solved if it went commercial?

Installed Android studio, Genymotion (fast emulator) and created a parse.com account (mobile backend as a service). Parse describes itself as ruby on rails for mobile apps! if it lives up to that claim i'll be delighted. the irish startup feedhenry (now owned by red hat) looks like the best for enterprise, their junior developer job looks incredibly high spec.

Few people including myself following MOOC tutorials on udemy to get the basics fast. It goes too slow. I don't understand why we have mobile app development module after xmas when it will be too late.

Already having regrets that I don't own Apple hardware so I can run xcode and therefore create an iOS app, (easier to develop according to the internet). iOS developer are also in greater demand it seems. maybe I'll try it after.

Wish I was a much more experienced .NET developer because Xamarin cross platform framework makes it possible to write your app code almost entirely in c# on .NET platform and then it automatically maps the majority of the App to the mobile platforms for you, leaving just a small amount of platform specific UI work to finish it off. Looks like

a game changer for Enterprise mobile app development, applied for a student account, no response after a week, terrible customer service.

Seems that stackoverflow is the single coherent source of help on the internet for issues.

Realised that the company I work for (Ellucian dublin formerly CampusIT) is a graduate of NCI Business incubation centre.

Balancing home life demands with the urge to stay glued to the computer is exhausting.

Considering categorising college by type, and course by dept/faculty and/or mode of study and to easily allow users to decipher type/category consistent colour coding will be used.

Scaling problems may arise around moderation of reviews; need to consider using a spam/abuse engine, need to research existing solutions.

GlassDoor opened an office in Dublin, mostly sales jobs as usual.

Found UMLet a great opensource tool for UML diagrams.
Balsamiq is awesome for wireframe mockups

Watched the 2015 WebSummit on youtube, I will be an entrepreneur one day hopefully!

6.3.3 November

Discussed my mockups with people,

- modules and club/socs are not worth including in the first version of the App(or at least the prototype)
- search by faculty/subject area is important

ideas

1. contact reviewer button to ask more specific queries
2. add fees details to courses
3. report : list top rated colleges/courses
4. report : list of similar courses to compare fees qualification details etc

5. New section of APP for MOOCs to compare them (possible web service api for info? coursera, udacity, edX, codeschool, cloudacademy, lynda.com, pluralsight, udemy)

Need to make the App better for user than the competitor college/course website

whatcollege.ie

How?

- More extensive list of colleges and courses (time consuming)
- More reviews by graduates/students (time and possibly money consuming)
- Nicer UI/UX (javascript/css effort required later) ,
- map of colleges,
- intuitive navigation, easier sign up/in (social integration)
- Give users who contribute reviews the option to remove ads
- Faster performance
- QUick search and read/writes
- COnnect both a web app and a mobile app to same database via backend service API

I bought the domain name sturevu.com

6.3.4 December

Started actual development, gave myself 6 (full day holidays from work) days to start functional requirements. Struggled with the basics due to the ambiguity of java errors and confusing nature of android studio. Spent dozens of hours googling errors and trying lots of suggestions on stack overflow. Eventually hacked my way to a basic prototype.

key problem number 1: how to pass objectId around application with "Intent"
sub problem : how to use objectId from adjacent adapter class (passing variable between classes)

key problem number 2: how to write a nested query to get parse objects based on said objectId

Eventually solved these problems and achieved basic working functionality.
Solved problem 1 with an open source solution; ParseProxyObject
Solved subproblem adapter issue with simple passing java variable around.
Solved problem 2 by trial and error with parse documentation and parse forum post answers on nested query.

Created test reviews , took videos to show demos to people for feedback.

explored the options for creating web app with parse, Express framework is documented on their website, looks similar to rails in terms of minimalism and automation of boilerplate.

6.3.5 January

Continued to develop prototype

Added cloud code javascript functions for calculating average rating and counting related entities. This required making certain pages reload data (example after adding a review) to show up to date average rating and count of reviews.

Added Stars for Rating on the Add new review page instead of drop down list.

Tested in genymotion emulator.

Functional requirements breakdown;

Done	Still in Backlog	Ideas
Login & Create Account (Email/Username/Password)	Social Login (Facebook & LinkedIn)	
List College & Course	Search College & Course	
	Module & Club/Soc (list, search, read and create reviews)	
Read reviews (College & Course)		
Create reviews (College & Course)		
Flag review as Spam	Limit to one flag per user per review	

Vote review helpful	Limit to one vote per user per review	
	View Search History	
	Save to Favourites	
	Subscribe to Updates	
Calculations of Average rating and counts (college & course)		
		anti-spambot check when user tries to create a new review to avoid abuse
	map of colleges	
		report : list top rated colleges/courses
		report : list of similar courses to compare fees qualification details etc
		To highlight obvious revenge reviews instead of an average rating by review scores, display a graph of all review ratings so that outliers are clearly visible
		New section for MOOCs to compare or to suggest an add-on to compliment the college course
		display/integrate linkedin profile details so users can see the professional/qualification credentials of a reviewer
		ability to ask a question to a college or course and for them to respond
		ability to ask a question to a reviewer and for them to respond (post comments linked to review)

6.3.6 February

Summary of commits to code repository:

Fixed some issues with formatting;

- reformatted to 2 decimal points => average rating on college & course pages
- reformatted to 2 decimal points => average rating on college & course pages only used if data is double (not reformatted for integers to avoid error)

Implemented a “reload” of page to see updated data for average and count calculations after review created;

- onResume restart activity for college and course after new review activity

Updated button colors to ensure consistency (orange for list, green for save or create new, blue for general navigation/ vote helpful and red for cancel/ flag spam;

- color of buttons updated

Didn't do much work on project in feb, back to college modules.

Mid point presentation was very disappointing. The 2nd examiner was impatient and dismissive of the whole project concept. He wasn't even interested in seeing my working demo all he cared about was requirements documentation. Scored a disappointing 62.

6.3.7 March

Summary of commits to code repository:

- Implemented a map interface of colleges, that shows a map above a list of colleges so that when you click on a college name the map brings you to its location with a marker. Took a while to figure out how to get the google maps API working because the location service didn't work on the genymotion emulator, only works on a physical device.
- Implemented Login with facebook, which was fairly straightforward, it's well documented. Once again only works properly on a physical device.
- Replaced the default (old) action bar with a toolbar (new material design) and found a workaround for actions not displaying on the toolbar.
- Added a sliding navigation drawer to left of activity but couldn't find a solution for issue with hamburger menu icon not displaying on list, it displays a back arrow always.
- Implemented a basic Search Course functionality that matches exactly on first few letters of course name.

March was a busy month for continuous assessments for the other modules.

6.3.8 April

Summary of commits to code repository.

- added favourite functionality, for college and course, add and delete and list
- added comment functionality, list & add new (to reviews)
- added reply functionality, list & add new (to comments)
- updated icons
- added push notification functionality to be triggered when a new review is added to a favourite college/course
- added favourite college & course reviews functionality
- added drawer navigation and toolbar to reviews and comments
- added push notification functionality to be triggered when a new comment is added to a favourite review (college/course)
- updated reviewsingleitem page to display rating using stars & updated search button styling on searchcourses page
- reviewsingleitem made into a scrollview to allow for content that pushes buttons out of view
- added more push notifications; to be sent to review author when new comment is added & to be sent to comment author when new reply is added
- added my college review & my course review lists to the navigation drawer menu
- added countReviewComments cloud code function to ReviewSingleItem

April was a busy month with Exams etc

6.4 Requirement Specification

NCI BSHCENM4
<h1>Requirements Specification (RS)</h1>
StuRevU

Jordan Daly 11/6/2015

Requirements Specification (RS)

Document Control

Revision History

14/10/2015	1	Create			
21/10/2015	2	Update			

Distribution List

Eamon Nolan	Lecturer	

Related Documents

Title of Use Case Model	
Title of Use Case Description	

Table of Contents ☐

[Requirements Specification \(RS\)](#)

[Document Control](#)

[Revision History](#)

[Distribution List](#)

[Related Documents](#)

[1 Introduction](#)

[1.1 Purpose](#)

[1.2 Project Scope](#)

[1.3 Definitions, Acronyms, and Abbreviations](#)

[2 User Requirements Definition](#)

[2.1.1 Search for particular Colleges, Courses, Modules & Clubs/Societies](#)

[2.1.2 Read reviews of Colleges, Courses, Modules & Clubs/Societies](#)

[2.1.3 Post reviews of Colleges, Courses, Modules & Clubs/Societies](#)

[2.1.4 Login with a social network credentials](#)

[2.1.5 Register new email account](#)

[2.1.6 Login with email account](#)

[2.1.7 Save favorites](#)

[2.1.8 Search history](#)

[2.1.9 Subscribe to updates](#)

[2.1.10 Settings](#)

[2.1.11 Intuitive User Interface Navigation](#)

[3 Requirements Specification](#)

[3.1 Functional requirements](#)

[3.1.1 Use Case Diagram](#)

3.1.2 Requirement 1 Search for Colleges, Courses, Modules & Clubs/Societies : Use Case A1

3.1.3 Requirement 2 Read & Create Reviews : Use Case B1&B3

3.1.4 Requirement 3 Email User Account management; Register and Login : Use Case F1&F3

3.1.5 Requirement 4 Login with Social network account : Use Case G1

3.1.6 Requirement 5 Static data maintenance; CRUD Colleges, Courses, Modules and Clubs/Societies : Use Case D1

3.1.7 Requirement 6 Flag Spam: Use Case C2

3.1.8 Requirement 7 Moderate Reviews : Use Case C1

3.1.9 Requirement 8 Mark as Helpful: Use Case B2

3.1.10 Requirement 9 Post Advert & Pay for Advert: Use Case E1 & E2

3.1.11 Requirement 10 Save Favourites & view Search History: Use Case A2 &A3

3.1.12 Requirement 11 Subscribe to Updates: Use Case A4

3.1.13 Requirement 12 Search Colleges on Google Maps interface: Use Case A1

3.2 Non-Functional Requirements

3.2.1 Performance/Response time requirement

3.2.2 Availability requirement

3.2.3 Recover requirement

3.2.4 Robustness requirement

3.2.5 Security requirement

3.2.6 Reliability requirement

3.2.7 Maintainability requirement

3.2.8 Portability requirement

3.2.9 Extendibility requirement

3.2.10 Reusability requirement

3.2.11 Resource utilization requirement

4 Interface requirements

4.1 GUI

4.2 Application Programming Interfaces (API)

5 System Architecture

1. Introduction

6.5 *Purpose*

The purpose of this document is to set out the requirements for the development of a mobile application.

The intended users are students.

For example;

- Second level students who want to research third level or further education options may want to view college and course reviews in their country or city.
- Current undergraduate students who may want to research reviews of optional/specialisation modules and clubs and societies as well as postgraduate college and course reviews.
- Graduates who are already working and may want to research part time/short term professional qualifications by college and course and module.
- Exchange students and International students who are looking for peer recommendations.

6.6 *Project Scope*

The scope of the project is to develop a native mobile application that can be downloaded from the Google Play Store as an Android App.

The application will list educational institutions and organisations (colleges) and allow a user to search and filter by location such as country and city, and other options such as college type and initials (short name from acronym).

Once a college is selected a list of reviews of the college will be displayed in chronological order and the overall average rating and 'recommend to a friend' score displayed at the top.

Functionality to search for related courses, modules and clubs/societies will be accessible from the college page tabs.

The user will have the following options;

- A user will have the ability to filter reviews by date, star rating and student type (full time, part time, undergraduate, postgraduate student, exchange, international).

- A user will have the option to post a review of a college, giving star based ratings of the college relating to different attributes and characteristics
- A user will be able to filter courses by subject area, mode of study (full time/part time/online), course level (undergraduate/postgraduate) and qualification type (degree/dip/cert/masters/PhD). A user will be able to post a review of a course and rate different areas and features.

The same pattern of search, filter and review and rating will be applied to modules and clubs/societies with modules linked to courses and clubs/societies linked to colleges.

The requirements have been gathered from a mix of personal student experience and from analysing leading review apps in other user markets and domains such as GlassDoor which is a company review app for employees.

My knowledge regarding the requirements of a student derives from my own background as a student with experience as the following; full time undergraduate student in a three year Arts degree course, full time postgraduate student in a one year Information Technology skills conversion course and part time undergraduate student in a four year Computer Science degree course.

I also have two years' experience working for a company which develops software for the higher education market so I have in-depth domain knowledge.

6.7 Definitions, Acronyms, and Abbreviations

CCM&C/S Colleges, Courses, Modules and Clubs/Societies

.....

2. User Requirements Definition

1. Search for particular Colleges, Courses, Modules & Clubs/Societies

The user must be able to search based on text input from the device's keyboard and be able to filter results.

2. Read reviews of Colleges, Courses, Modules & Clubs/Societies

The user must be able to access a list of reviews related to their selected entity. The list should display the reviews chronologically to ensure up to date content.

3. Post reviews of Colleges, Courses, Modules & Clubs/Societies

The user must be able to post reviews.

4. Login with a social network credentials

The user must be able to login with Facebook/LinkedIn/Twitter (OAUTH) for easy access.

5. Register new email account

The user must have the option to sign up with email if they don't choose to login with a social network account.

6. Login with email account

Login with personal email address.

7. Save favorites

Save favourite Colleges, Courses, Modules and Clubs/Societies for easy navigation and user friendliness.

8. Search history

List past searches for user reference.

9. Subscribe to updates

Allow user to subscribe to email/push notification updates from Colleges, Courses, Modules and Clubs/Societies.

10. Settings

Settings may include the following;

- Rate StuRevU in Play store
- Share App
- Terms & Conditions
- Privacy Policy
- Logout (social/email account)

11. Intuitive User Interface Navigation

The user must always know how to go back to the previous page. For example a cancel or back button.

7 3. Requirements Specification

7.1 Functional requirements

12. Use Case Diagram

Figure 0-1 Use Case Diagram



13. Requirement 1 Search for Colleges, Courses, Modules & Clubs/Societies : Use Case A1

1. Description & Priority

The search requirement is a core functional requirement because the app's other functionality depends on this (basically it won't work without it) and its priority is the highest at P1. This requirement is essential to the overall system.

2. Use Case

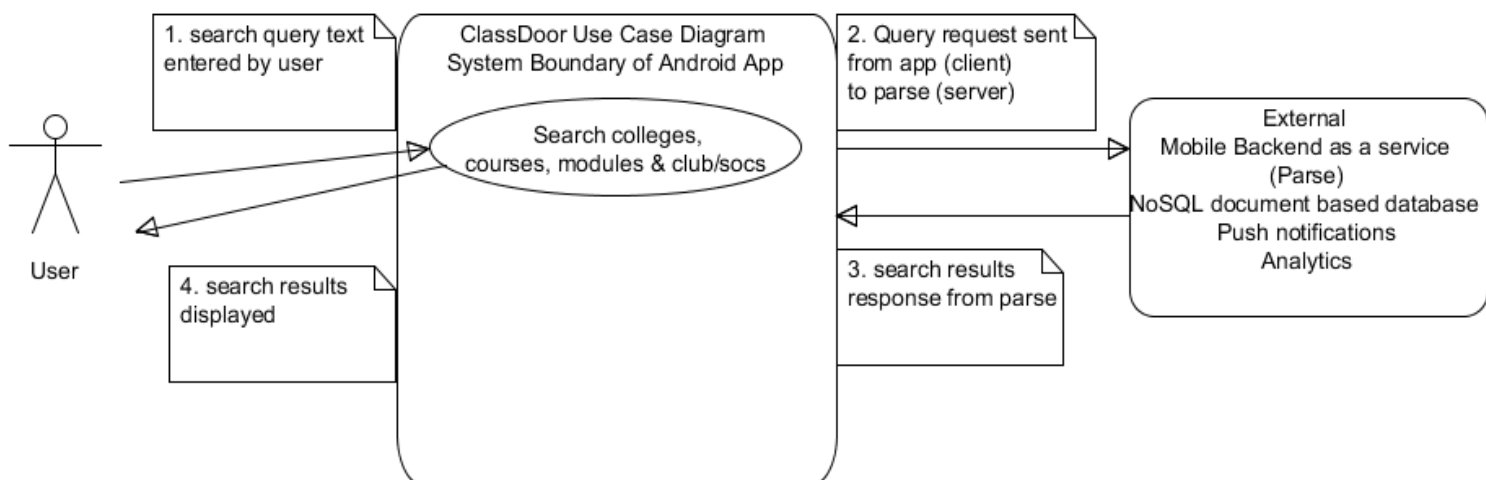
Scope

The scope of this use case is to allow the user to search for CCM&C/S and

Description

This use case describes the user entering search parameters (text or item selected from list of suggestions).

Use Case Diagram



Flow Description

Precondition

The app has been downloaded and the app has been launched.

Activation

This use case starts when a User enters text in a search box and touches the Search button.

Main flow

1. The system detects the search request.
2. The system displays a list of results.(See A1)
3. The system displays the user's expected result.(See E1)
4. The user selects their desired result.

Alternate flow

1. A1 : Incorrect results found
2. The system displays a list of similar but not matching results.
3. The user tries a different search.
4. The use case continues at position 4 of the main flow

Exception flow

1. E1 : No results found
2. The system does not find any results matching the search input parameter(s).
3. The user tries again with a new search.
4. The use case continues at position 3 of the main flow

Termination

The system presents the search results page (college, course, module or club/society).

Post condition

The system goes into a wait state

14. Requirement 2 Read & Create Reviews : Use Case B1&B3

1. Description & Priority

Read and create reviews of CCM&C/S, top priority P1. The functionality around allowing a user to view and post reviews (which are the main value generating content of the App) is essential to the overall system.

2. Use Case

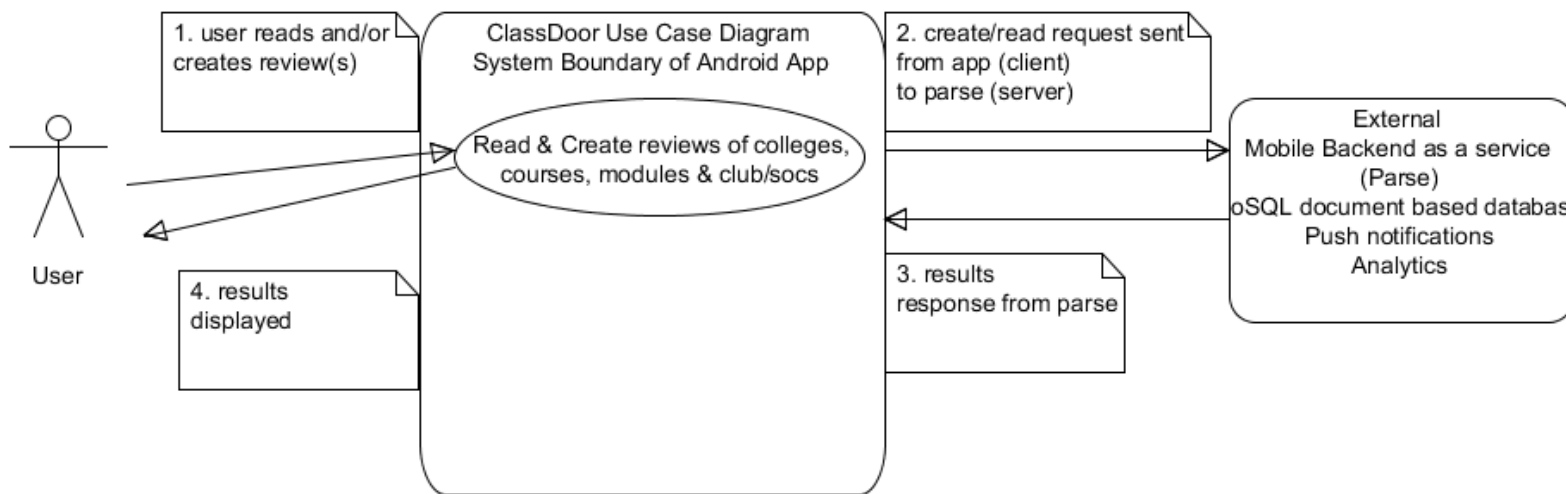
Scope

The scope of this use case is to allow users to view reviews and add new reviews.

Description

This use case describes the process of a user reading and creating reviews.

Use Case Diagram



Flow Description

Precondition

The user is logged in.

Activation

This use case starts when a user touches the reviews tab of a College, Course, Module or Club/Society (read review) and then touches the Add Review button.

Main flow

5. The system identifies the Add Review request.
6. The User fills the create review form and hits submit. (See A1)
7. The system displays review added successfully message.(See E1)
 8. The user can view their saved Review.

Alternate flow

A1 : Validation error

1. The system detects required field data missing.
2. The User adds the missing data.
3. The use case continues at position 3 of the main flow

Exceptional flow

E1 : Network connectivity error

4. The system displays error due to network connectivity issues.
5. The User addresses the network issue on their device.
6. The use case continues at position 4 of the main flow

Termination

The system presents the updated list of Reviews.

Post condition

The system goes into a wait state

15. Requirement 3 Email User Account management; Register and Login : Use Case F1&F3

1. Description & Priority

For users who don't want to login with an existing social network they need the option to set up a user account by email and login.

2. Use Case

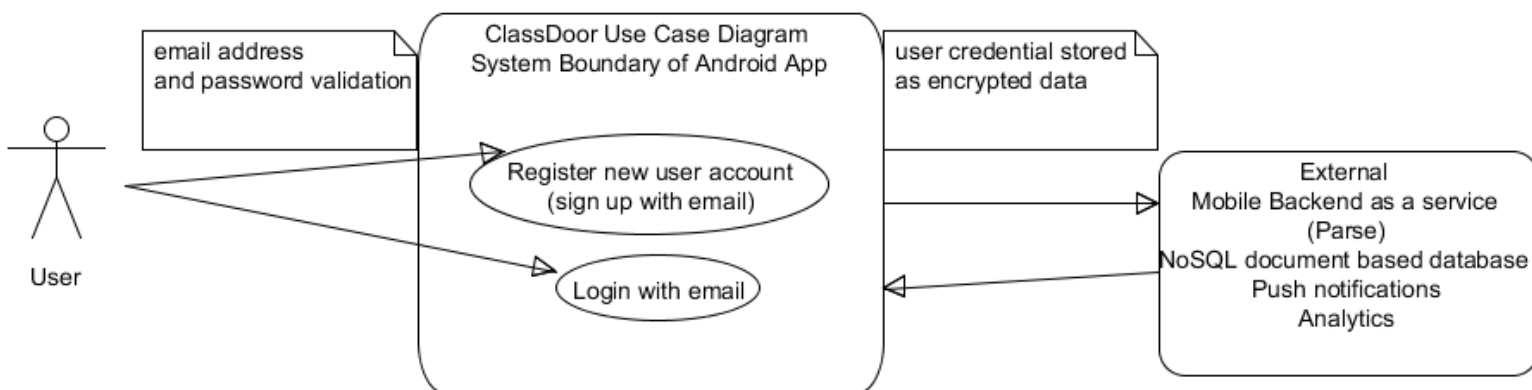
Scope

The scope of this use case is to allow users to register a new account by email address and then log in.

Description

This use case describes the process of email user account registration and login.

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode.

Activation

This use case starts when a User launches the App after downloaded.

Main flow

9. The system presents the Login/Register new account page.
10. The User creates a new user account and confirms their email address.(See A1)
11. The system(See E1)
12. The user logs in with username and password.

Alternate flow

A1 : username/password requirements validation error

7. The system displays error message stating above
8. The User corrects the issue
9. The use case continues at position 3 of the main flow

Exceptional flow

E1 : username/email address already in use, uniqueness requirement error

10. The system displays error message stating above
11. The user enters a different username/email address or uses Forgot my password for the email address.
12. The use case continues at position 4 of the main flow

Termination

The system presents the home page of the App.

Post condition

The system goes into a wait state

16. Requirement 4 Login with Social network account : Use Case G1

1. Description & Priority

Allow users to login with existing social network accounts for handiness. Examples include Login with Google, Facebook, LinkedIn and Twitter.

2. Use Case

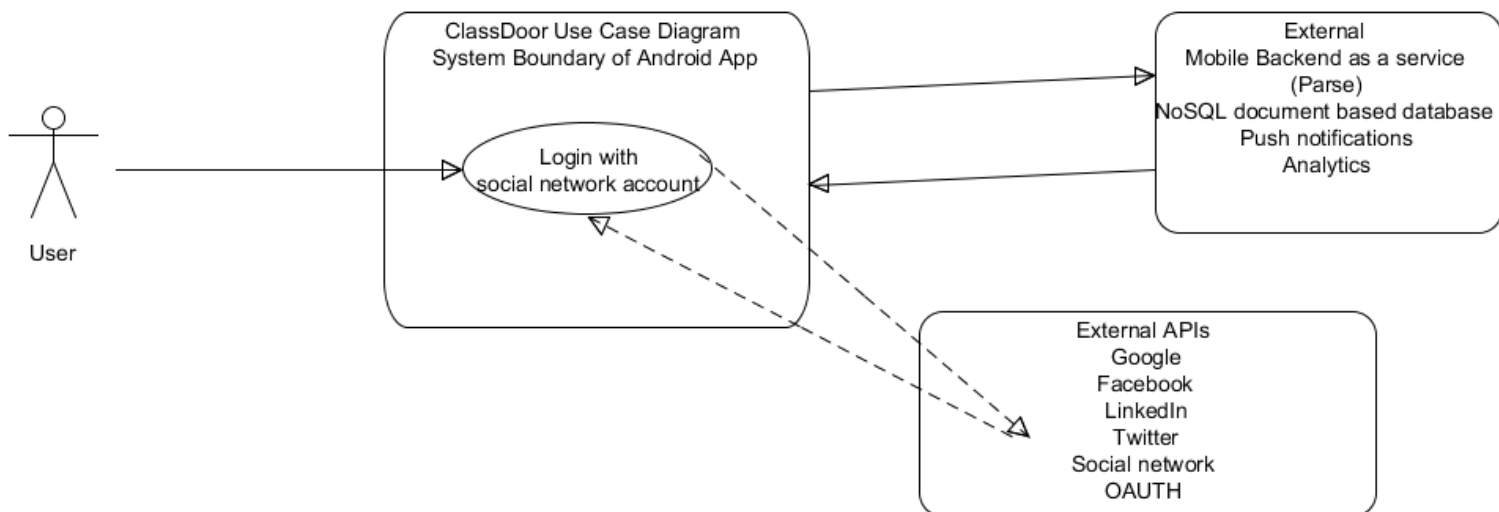
Scope

The scope of this use case is to allow the user to use Login with Google/Facebook/LinkedIn/Twitter functionality for user friendliness.

Description

This use case describes the Login process with a social network account plugin.

Use Case Diagram



Flow Description

Precondition

The App is downloaded and launched.

Activation

This use case starts when the User touches one of the Login with Google/Facebook/LinkedIn/Twitter buttons.

Main flow

13. The system identifies the request to the third party API and send the request to it which results in a response which presents the Allow access login page.
14. The User allows access and Logs in.
15. The system displays login failed. (See E1)
16. The User logs in successfully.

Exceptional flow

E1 : Social login failed

13. The system presents the Allow access login page again.
14. The User tries again.
15. The use case continues at position 4 of the main flow

Termination

The system presents the home page of the App.

Post condition

The system goes into a wait state

17. Requirement 5 Static data maintenance; CRUD Colleges, Courses, Modules and Clubs/Societies : Use Case D1

1. Description & Priority

The App requires the ability to keep up to date tables of data regarding Colleges, Courses, Modules and Clubs/Societies. The Admin user must be able to maintain the data manually through an interface or file upload functionality. P2 Important but not the core part of the App's user facing functionality.

2. Use Case

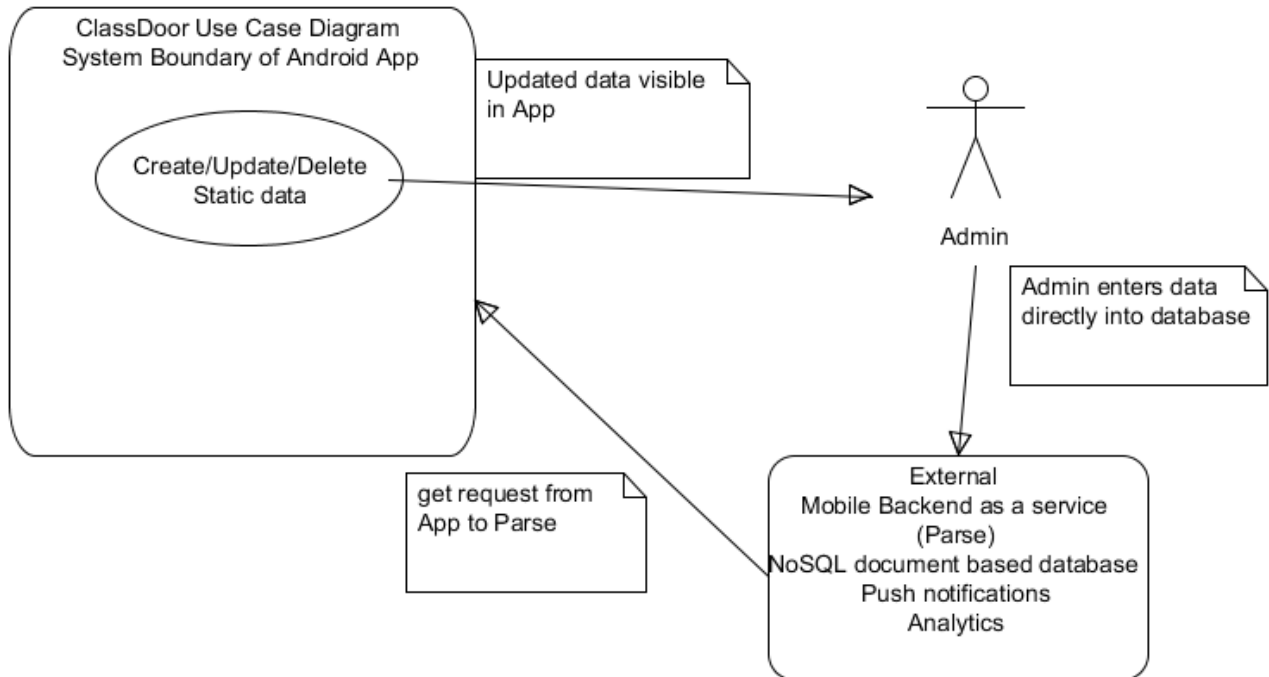
Scope

The scope of this use case is for an Admin to create, update and delete data for Colleges, Courses, Modules and Clubs/Societies.

Description

This use case describes the CRUD functionality relating to Colleges, Courses, Modules and Clubs/Societies by Admin.

Use Case Diagram



Flow Description

Precondition

The Admin is logged into Parse.

Activation

This use case starts when the Admin enters data into tables in parse.

This functionality is outside scope of the App itself in regards to flow of interaction with App.

18. Requirement 6 Flag Spam: Use Case C2

1. Description & Priority

The user must be able to flag reviews as spam.

2. Use Case

Scope

The scope of this use case is to allow users to flag reviews as spam if they have inappropriate, offensive or obviously bogus content.

Description

This use case describes the user flagging a review as spam. P3 not very important but necessary.

19. Requirement 7 Moderate Reviews : Use Case C1

1. Description & Priority

When a user has flagged a review as spam the Admin user must be able to update and delete reviews.

2. Use Case

Scope

The scope of this use case is to allow the Admin user to update and delete reviews as required, especially if marked as spam. P3 not very important but necessary.

20. Requirement 8 Mark as Helpful: Use Case B2

1. Description & Priority

The user must be able to mark a review as helpful. This helps in highlighting valuable content. The higher the helpful count the more valuable the content of the review and

2. Use Case

Scope

The scope of this use case is to allow the user to mark reviews as helpful and display the number of times a reviews has been marked as helpful. P3 not very important but necessary.

21. Requirement 9 Post Advert & Pay for Advert: Use Case E1 & E2

1. Description & Priority

The user must be able to create and pay for an ad. The target user for ads will be products and services aimed at 17-35 year olds.

2. Use Case

Scope

The scope of this use case is to allow users to post and pay for adverts. P3 not very important but necessary to contribute to running costs of the App.

22. Requirement 10 Save Favourites & view Search History: Use Case A2 &A3

1. Description & Priority

The user must be able to save their favourite CCM&C/S and view search history to check which ones they viewed in the past. This is for user friendliness and for convenience.

2. Use Case

Scope

The scope of this use case is to allow the user to view their search history as a list and to save favourite items for quick access upon return to the App at a later point in time. P3 not very important but necessary for user experience reasons.

23. Requirement 11 Subscribe to Updates: Use Case A4

1. Description & Priority

The user must be able to subscribe to updates to be alerted of new reviews of certain CCM&C/S.

2. Use Case

Scope

The scope of this use case is to allow users to Follow certain CCM&C/S so that they get push notifications when new reviews are added.

24. Requirement 12 Search Colleges on Google Maps interface: Use Case A1

1. Description & Priority

The user must be able to search for colleges on a map.

2. Use Case

Scope

The scope of this use case is to allow the user to browse a map interface of college near them. P3 not very important but a nice feature to have.

7.2 Non-Functional Requirements

25. Performance/Response time requirement

The typical crash rate is 1-2%. The general rule of thumb is to optimize to around 1 second response time when taking API latency into account.

Users may have some tolerance for slower response times, but anything over 3-4 seconds total response time is unacceptable for the majority of users.

26. Availability requirement

The App should be available 24/7/365.

27. Recover requirement

It is basically testing how well a system recovers from crashes, hardware failures. The App must launch correctly after a crash or after the user's device powers off. Actual server side hardware issues are not in scope because Parse is being used.

28. Robustness requirement

Robustness is the ability of a computer system to cope with errors during execution. The necessary error handling mechanisms and error messages will be included in the App to deal with possible errors.

29. Security requirement

The App's API keys should not be exposed by decompilation or in Github.

30. Reliability requirement

This is the point at which the system fails. The App should be able to handle thousands of concurrent users reading and writing data.

31. Maintainability requirement

Regular updates of the App must be released to allow users to install bug fixes and new features.

32. Portability requirement

The App must work on all devices running the Android Operating System.

33. Extendibility requirement

The App must be extendible to allow for integration with third party services such as Advertising, App monitoring, Static Data feeds and Spam engine services.

34. Reusability requirement

The design must be reusable for development of an iOS App.

35. Resource utilization requirement

The target is to stay below the free threshold for as long as possible on Parse to avoid expensive running costs.

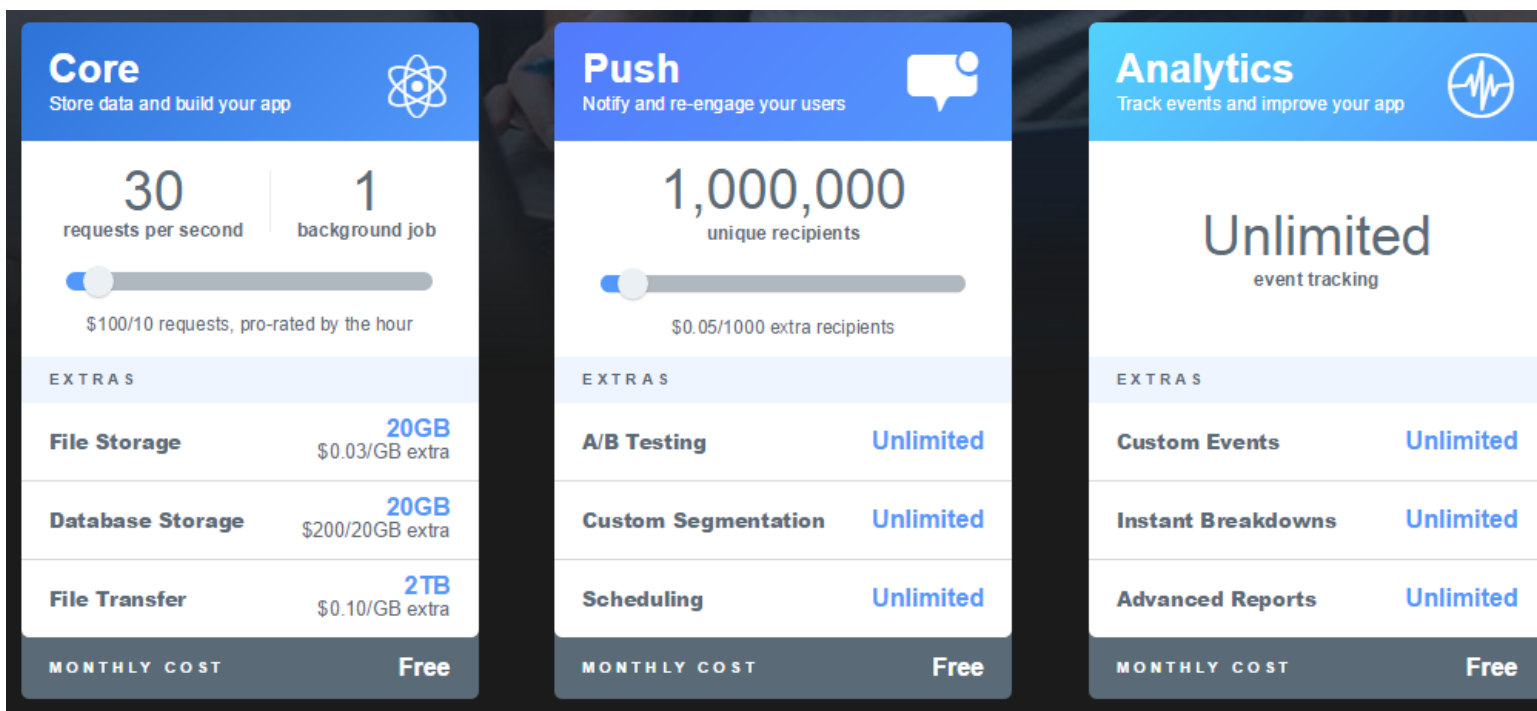


Figure 2: Parse (2015). Available from: <https://parse.com/plans>

3. Interface requirements

7.3 GUI

Figure 3-1 Login page



Figure 3-2 Home page

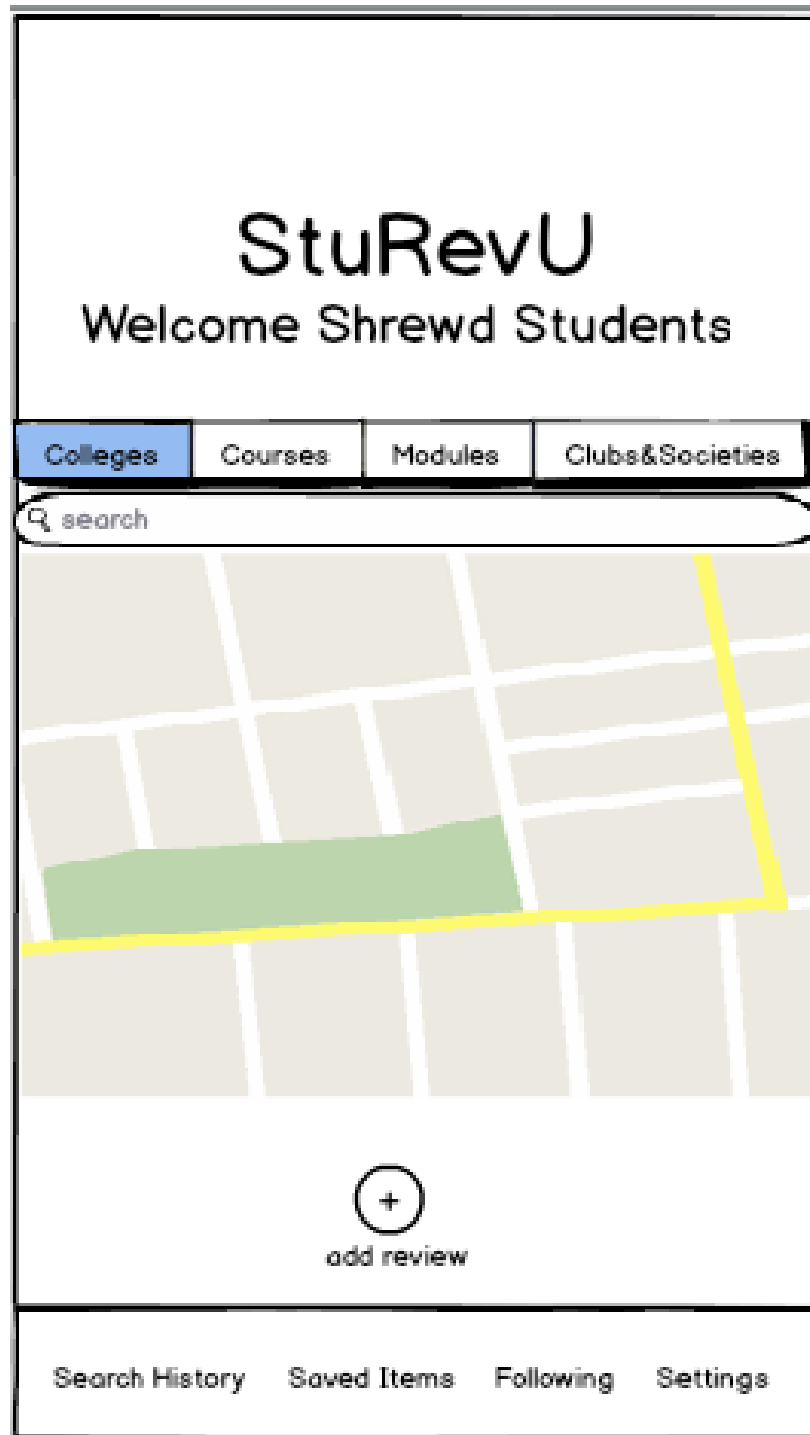


Figure 3-3 Search page

StuRevU

CancelSearch

Q search Colleges

Colleges

Courses

Modules

Clubs&Societies

recent college searches

[clear](#)

NCI

UCD

DCU

QWERTYUIOP

ASDFGHJKL

⬆ZXCVBNM⬇

123🌐🎤spacereturn

Search History

Saved Items

Following

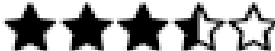
Settings

Figure 3-4 College page

[Home](#) > [Colleges](#) > [NCI](#) >

NCI

National College of Ireland

3.5 

Reviews

Courses

Modules

Clubs&Socs

[11](#) | [22](#) | [144](#) | [16](#)

[Filter](#)

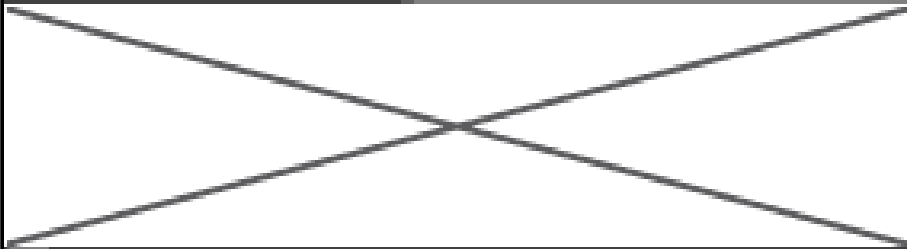
Review Dated Most recent


-

-

-

review dated least recent





add review

Search History

Saved Items

Following

Settings

Figure 3-5 Review page

[Home](#) > [Colleges](#) > [NCI](#) > Reviews

NCI

National College of Ireland

3.5

★

★

★

★

☆

Reviews

Courses

Modules

Clubs&Socs

11

22

144

16

[Filter](#)

Date

2.5

★

★

★

☆

☆

Title

Student type

Content

Pros

...

Cons

...

Advice

...

Helpful (0)

+

add review

Search History

Saved Items

Following

Settings

Figure 3-6 Add Review page

[Home](#) › [Colleges](#) › [NCI](#) › [Reviews](#) › Add Review

Cancel

Add Review

Submit

Choose Rating



Student Type 

Your review is completely anonymous

Review Title

Pros

...

Cons

...

Advice

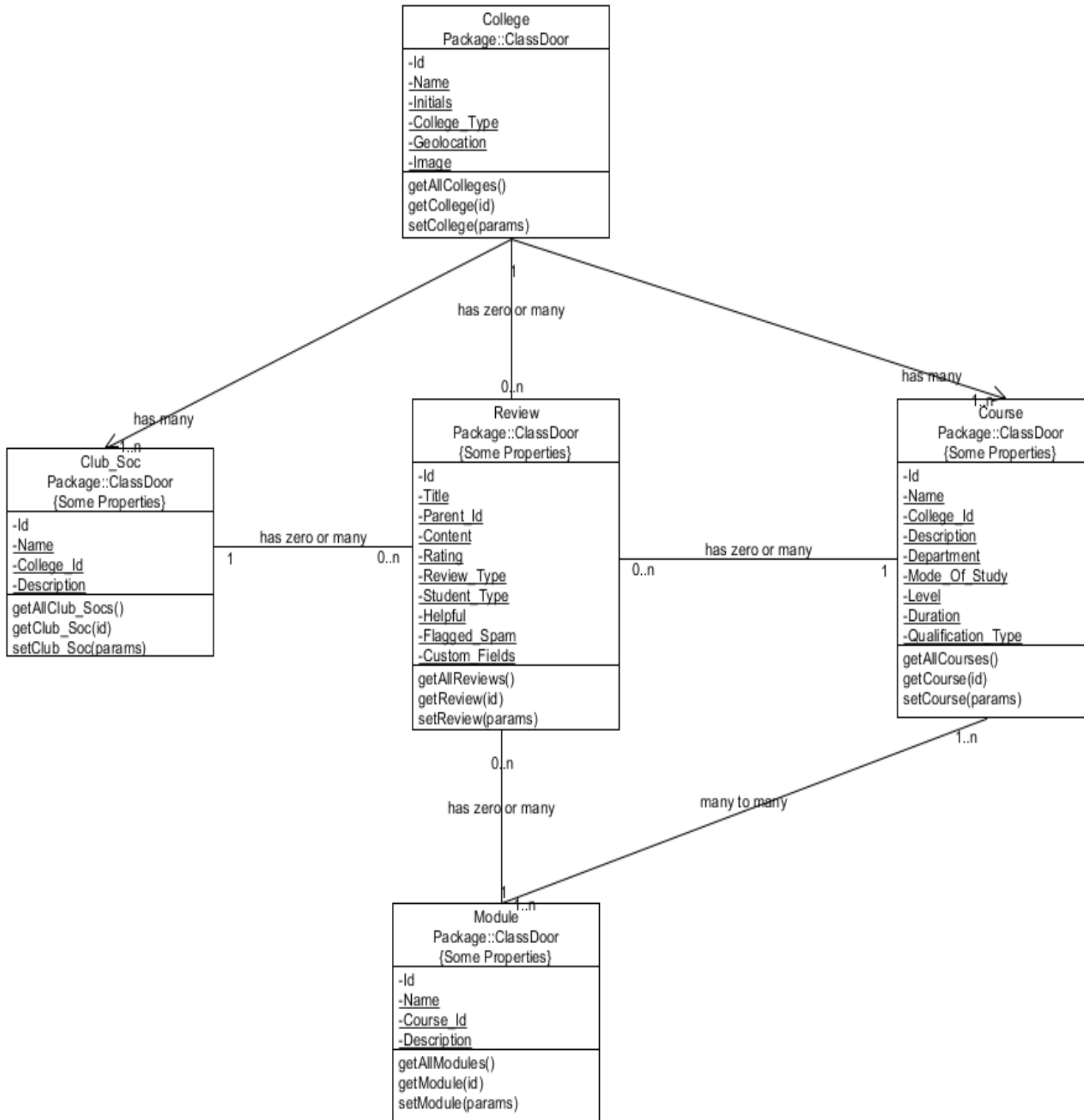
7.4 Application Programming Interfaces (API)

Explain which interfaces your system offers or which are used by your system. Examples include Google maps and Weka.

- Parse for Mobile Backend Service
- Google Maps for map search user interface
- Stripe for payments,
- Facebook, LinkedIn, Google and Twitter for Social Login OAUTH

4. System Architecture

Below is a Class diagram to outline the structure of the system. To avoid repetition I made one review class linked to all other classes by parent_id instead of having a corresponding review class for each entity class.



5. System Evolution

The use of cross platform technology such as Xamarin could allow the development of both Android and iOS Apps from the same code which would reduce the time and financial cost of running and maintaining the App.

The App will initially have static data for Ireland, then the UK and eventually all other English speaking countries such as US, Canada, Australia and New Zealand. Localization is required to gain traction in countries where English is not the first language.