



---

# CAMPUS STUDENT HELPER

## FINAL REPORT

---

School of Science and Engineering



CAPSTONE DESIGN  
FALL 2015  
AYMANE ELKHROUFI  
SUPERVISED BY: OMAR IRAQUI HOUSSAINI

# Table of Contents

<b>Feasibility Study</b> .....	5
<b>User and System Requirement</b> .....	9
<b>Functional Requirements</b> .....	9
<b>Use Case Diagram</b> .....	12
<b>Non-functional Requirements</b> .....	13
<b>STEEPLE Analysis</b> .....	15
1. <b>Social</b> .....	15
2. <b>Technological</b> .....	15
3. <b>Environmental</b> .....	15
4. <b>Economical</b> .....	15
5. <b>Political</b> .....	15
6. <b>Legal</b> .....	16
7. <b>Ethical</b> .....	16
<b>Technology Enablers</b> .....	17
1. <b>Server Side Analysis</b> .....	17
2. <b>Data Storage and Management</b> .....	19
3. <b>Client Side Analysis:</b> .....	20
4. <b>Conclusion</b> .....	21

<b>System Architecture and Design</b> .....	22
1. <b>System Architecture</b> .....	22
2. <b>System Design</b> .....	24
<b>Implementation and Testing</b> .....	33
1. <b>Database Management System</b> .....	33
2. <b>Server Side</b> .....	34
3. <b>Client Side</b> .....	37
4. <b>Testing</b> .....	37
5. <b>Results</b> .....	37
<b>Conclusion</b> .....	40
<b>References</b> .....	41

## Acknowledgement

I'm taking this opportunity to express my gratitude to, my advisor at Al Akhawayn University in Ifrane (AUI) , Professor Omar Iraqi Houssaini for not only sharing his knowledge about this field but also for all the moral support he had provided throughout my studies in AUI. I would like to thank him for the greatness of his advices and his guidance throughout this capstone design project. He was able to successfully provide a familiar and comfortable environment during our meetings. I consider myself a very lucky individual as I had such a professor by side. His knowledge about the Information technology field helped me clarify so many aspects in this field. I was provided with an opportunity to know my weaknesses, and more importantly, to have a clear idea on how could I fix some of those weaknesses.

A handwritten signature in blue ink, consisting of a stylized 'A' inside an oval, with the date '04/M/15' written below it.

## **Abstract**

The objective of this capstone project is to facilitate the students' social life within a campus through a social mobile application that provides a variety of services; from rating teachers and courses to carpooling, lost and founds, and sell and buys.

This is a software that would make student life more sociable than ever, where every student may advertise about anything he or she needs, then get feedback from students all around the campus. It is a virtual yet very social and ethical application.

The very first procedure is to analyse the need of student, how they are dealing with problem such as losing a phone or traveling back home, how they get initial background about courses and professors and so on. A clear understanding of what target users need would make the project more feasible.

The phase that would follow is an analysis of the market; what similar software looks like, are they doing fine, and the added values that this may bring. Only after that we could conclude if the project is feasible, and proceed.

The next phase is to analyse the requirements through understanding what students really needs, keeping in mind the non-functional requirement. A very important step would follow which is the design phase. The architecture of the application would be designed in such a way to produce a secure, high performing, reliable, and extendible software. Once this is clear, implementing the project would be the easy part, yet the longest. The product would be tested while being implemented before deploying it, and maintained afterwards.

The application will meet with the common standards of modern mobile application and fully documented.

# **Feasibility Study**

## **Project Context**

While surfing on Facebook, I noticed a great amount of Facebook pages related to our campus social life, pages such as lost & found. Why not a mobile application that puts all of that in one place; a nice social application that would make information available for students on many matters that would involve asking around people. This application would need to be accessed anywhere, anytime; so it is reasonable to think of mobile application at this context to get advantage of its mobility.

## **Project Initial Scope**

Services that this application provides:

Academically related services:

- Students feedback on teachers
- Students feedback on courses
- Add & drop friendly deals

Socially related services:

- Lost & founds
- Sells & buys
- Carpooling

## **Client Analysis**

The application targets university student only; so the question to be asked is: do they really need such services? For the academic services, student use the opinion of others to build initial knowledge about their courses and professors. However, student asks only their friend but always feels in need for more information. An application that allow students to communicate more easily, and share any information related to their courses and professors. Also for the social services, students need a way to advertise their lost/found items, sell/buy from one another, and carpool each other.

We could say that this a feasible project on the marketing side. It would be a full free application downloadable through Google store, and implemented to fulfil high quality modern mobile application standards.

## **Market Analysis**

Facebook is the place where I thought of this application; an obvious question that would come to mind, is why people may need a mobile application for this purpose while Facebook pages provide a familiar environment for student to find such information. Well, those Facebook pages are not monitored nor meant for such services. People may not even know a page of carpooling exists due to organisation issues.

While searching, I could not find an application or a website that holds all those services, yet there are solutions for each individual service. For instance, ratemyprofessors.com and ratemyteachers.com are well implemented popular websites for rating international teachers, I used it myself for feedback on some of my professors. However, it is not very popular in our campus as one could not find information on every teacher.

Also there exist plenty of applications for “lost & founds” or “sells & buys”; however, it would be more practical to have it as an independent, campus related, application.

Carpooling is common around the globe, and it provides a great tool for people to travel. In Morocco, websites such as taxikbir.com and covoituragemaroc.com are doing just fine. They become very popular due to the comfortless in Moroccan’s transportation. Moreover, because student travel in approximately the same periods, holidays for instance, a campus related application for carpooling is very feasible.

### **Project Final Scope:**

Although the project is very feasible and interesting as whole, there are some constraints that one want to consider. Because of the limited time given to implement the application, it would almost impossible and unhealthier to work on such project as one individual. This is why my advisor and I decided to limit the scope of the application to the following:

Academically related services:

- Rating professors with relation to the courses they Taught

Socially related services:

- Lost & founds
- Carpooling

### **Conclusion**

While asking around in campus, the idea seems to be interesting as it was expected after the client analysis. The idea is feasible not only because students need it



but also because most of its services are doing well in the market yet there is no famous application that holds all of that. Also, limiting its scope to only campus students would make it more reasonable since students share more or less the same experiences through their academic studies.

# User and System Requirement

## Functional Requirements

### 1. Profile

#### 1.1. Actors:

##### 2.1.1. Student (Authenticated)

#### 1.2. Functionalities:

**1.2.1. Edit profile:** Authenticated users shall be able to edit every detail of their profile.

**1.2.2. View another user profile:** users are allowed to view other users' profiles.

### 2. Academic

#### 2.1. Actors

##### 2.2.1. Student (Authenticated)

#### 2.2. Functionalities

**2.2.1. Request a new professor rating:** users shall be able to request a new professor rating where they provide the professor name, the course that he/she teach, and a description. After submitting, the system checks on the existing ratings then if a similar rating exist, the system shall show it to the user, before he/she decides to submit his new request. When a request is submitted it should be displayed under academic in a wall.

- 2.2.2. Rate professor:** users shall be able to rate professors, where they find requests to rate professors in the wall. Users rate professors by stars from 1 to 5 and a description about the professor. Also, they may choose to change their ratings at any time.
- 2.2.3. Search for a rated professor:** users shall search for a professor rating using the professor name, or course name.

### **3. Social**

#### **3.1. Lost and founds**

##### **3.1.1. Actors**

###### **3.1.1.1.Student** (Authenticated)

##### **3.1.2. Functionalities**

**3.1.2.1.Post lost item:** users shall be able to post their lost item here, where they provide a description about the lost item and a picture if any. New added posted are visible in the wall under social and under lost and founds.

**3.1.2.2.Post found item:** users also shall be able to add a new post of a found item with a description and a picture, then it would be displayed under social and under lost and founds.

**3.1.2.3.Search for lost/found item:** users shall be able to search using a key word to find a post about a lost or found item, or search in the wall for their desired post. Then he/she could contact the poster by viewing his/her profile.

## **3.2. Carpooling**

### **3.2.1. Actors**

**3.2.1.1. Driver-**extends student (Authenticated)

**3.2.1.2. Passenger-** extends student (Authenticated)

### **3.2.2. Functionalities**

**3.2.2.1.Post a new trip (for drivers):** users shall be able to create new carpooling trip as driver, with the following information: destination, car type, number of seats available, time, price, and any additional description.

**3.2.2.2.Search for a trip (for passengers):** also, the user shall be able to act as a passenger and look for a trip in the wall of carpooling, or search for the trip by its destination. (The wall displays only carpooling post from the same university).

**3.2.2.3.Check-in trip (for passengers):** the user shall be able to choose to check-in for a trip. Then the application notifies the driver.

**3.2.2.4.Share in social media (for passengers & drivers):** users are allowed to share any post (academic or social) in their social media accounts.

## Use Case Diagram

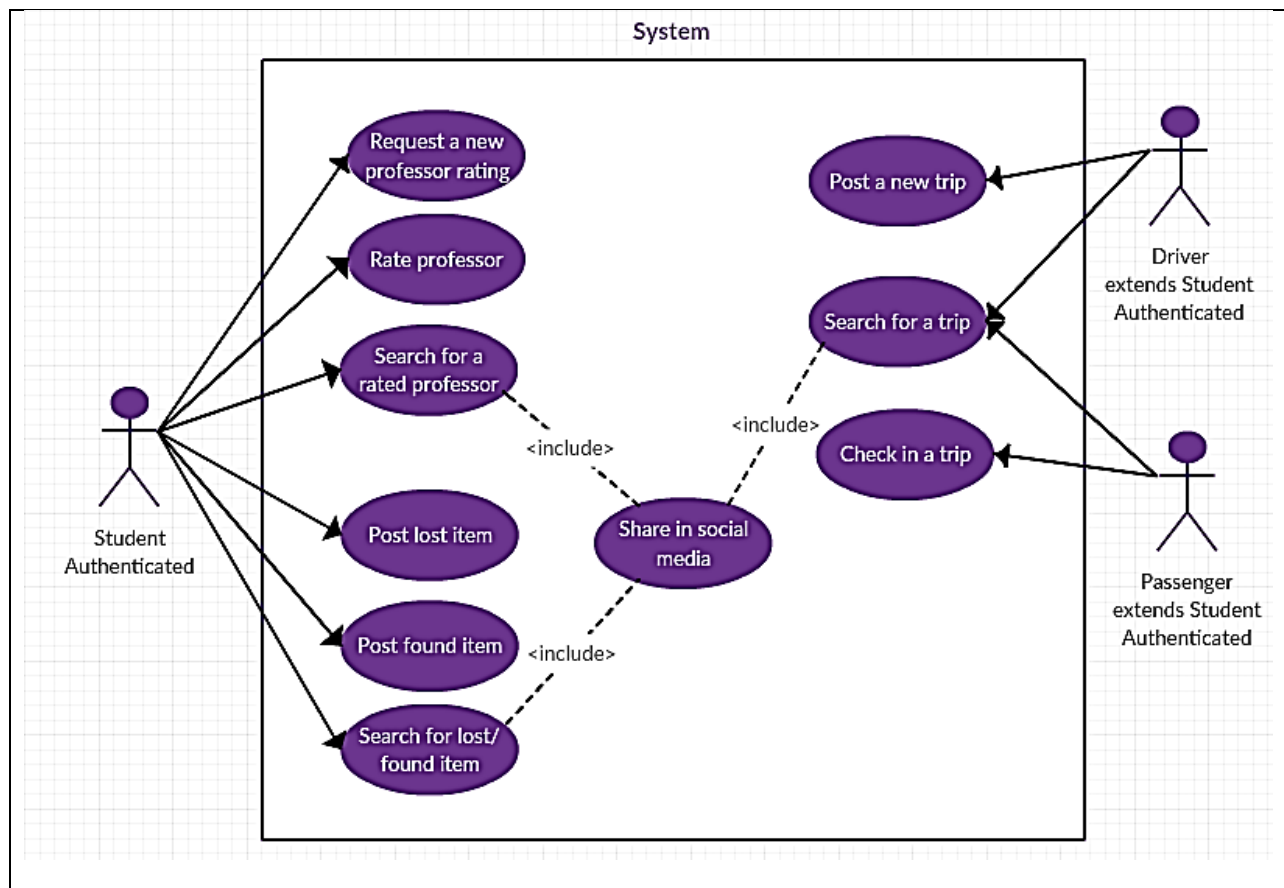


Figure 1. Use Case Diagram

## **Non-functional Requirements**

### **1. Performance**

The application response time should be minimized, and also the number of users connected in parallel should be maximized.

### **2. Scalability**

The application should be highly scalable; since it is meant to be used by university students anywhere, then the demand may increase whenever a new university starts using the system.

### **3. Extensibility**

The application should be extensible to allow for adding other services in the future, services such as sell and buys.

### **4. Integration**

The system should be interruptible to allow other application to use its services. Also, the application shall use Facebook services as a third party services as well as being able to integrate other APIs such as Google maps.

### **5. Security**

The system should be highly secure since only authenticated users gain access to the server. It should respect the following:

**5.1. Confidentiality:** Users shall view data related to his campus and his campus only. Also, the application shall insure the privacy of his profile information, the user shall be able to choose what information would be visible for others.

**5.2 Integrity:** Only authorized actors shall be able to modify data using authorized operations.

**5.3.Availability:** This is a main concern, the application shall be always available as students may need to access it any time. The server up-time should be 99.99% which allows for 53 min/year.

## **6. Maintainability**

The system should be easily maintainable to allow for additional upgrades that may need to be implemented in the future.

## **STEEPLE Analysis**

### **1. Social**

This in fact is a social application, it helps student to deal with some social issues in an automated manner. Student will after all socialize with each other more easily while they communicate to each other a lost\found item, a carpooling trip, or ratting their teachers.

### **2. Technological**

People are becoming highly dependent on their smartphones because of its mobility. This application uses this technology to deliver the service anywhere, any time in a secure manner.

### **3. Environmental**

The application is environmentally friendly since it does not cause any harm to the environment, but helps it. Carpooling service may decrease the number of cars traveling at the same time which may decrease a portion of CO2.

### **4. Economical**

The Application is entirely free for student to use; moreover, it may even be beneficial if we are taking about carpooling that may decrease the traveling fees.

### **5. Political**

The application is unrelated to any governmental activity, yet it respects all rules of any academic university. Information is initially generated by student, so in order for them to register in the system they need to approve a set of rules among them not to get in any kind of political discussions.



## **6. Legal**

The application is fully legal since it is a social campus related application. However, in case of carpooling, the driver should have an insurance that the passengers should check before hitting the road.




## **7. Ethical**

Client confidentiality should be kept: all information related to trips' history should only be communicated to their respective user

# Technology Enablers

## 1. Server Side Analysis

The server side shall be implemented to fulfil all non-functional requirement. A wise choice of technologies to be used is crucial to achieve a high performing, scalable, extendible, secure, and maintainable web services. The following table, shows the application servers that I considered:

<b>JAVA</b>		Java is the most famous, widely used, language for implementing web services. Java oriented web services run its EAR and WAR files with JVM that enables it to be deployed in different OS systems. It also uses needs an application server such as glassfish or jBoss to run the application.
<b>JavaScript</b>		JavaScript were used mainly for client side web development as a dynamic programing language until 2009 where Node.js was developed to allow programmers to implement standalone web application using this technology.[2]
<b>Microsoft .Net</b>		Another application server is .net technologies that was developed by Microsoft to run in windows.

**Table 1: Considered Application Servers**

## 1.1. Technologies comparison

### 1.1.1. Performance

There are actually some studies that suggest that Node.js is faster than JEE and .NET and vice versa, yet those studies are never conclusive.

Performance is related to other criteria such as the architecture of the application, the hardware in use; and therefore, we cannot conclude that a technology is better performing than another.

### 1.1.2. Scalability

Java EE is highly scalable since it distribute its components across multiple servers. So, scaling in and out is easier since we would need only to add more machines (processing power) to scale out and in.

Node.js is event driven that can delegate processes such I/O operations to other components which gives it time to process other requests. [2] It is also single threaded so it doesn't take advantage of high performing CPU(s). [2]

.NET as java, support physical scalability.

### 1.1.3. Developer Productivity

Features	Java EE	Node.js
IDE Support	Yes, multiple choices including Eclipse sublime and idea	Yes, multiple choices including visual studio, eclipse sublime
Dependency Management	Maven	NPM

Database support	Yes	Yes
Runtime Environment	JVM	Google's V8 engine, an open-source runtime environment used by node.js
ORM frameworks	Hibernate	Sequelize, a promise-based ORM for Node.js
MVC support	JSF	Sails.js, a realtime MVC Framework
Security – access control	Spring Security + others	Cross-origin resource sharing of Express.js
Testing frameworks	Yes	Yes

**Table 2: JavaEE vs NodeJS**

Node.js is a recent technology, yet it managed to achieve a high performance and scalability. In fact, JavaScript are adopted everywhere, they are the future of programming in my opinion. It's very encouraging to discover this technology that equals years of oracle development. This project would be a great opportunity to discover this recent technology. For that reason, the following table shows the frameworks this technology supports in comparison with JEE.

## **2. Data Storage and Management**

JavaScript and NoSQL both uses JavaScript Object Notation (JSON) so it is recommended to use a NoSQL DBMS for storing and managing data as Node.js had no ORM framework.

However, recently, Sequelize was developed as a framework for ORM support. SQL works greatly with application servers that has ORM support. NoSQL uses document to store data, opposed to SQL that uses a relational model to store data as tables. NoSQL is used for application that may need to grow rapidly with a high throughput of data which not the case in our application.

For this project we are going to work with MySQL the most widely used SQL DBMS.

### 3. Client Side Analysis:

The ideal case: client can access the services in multiple ways: using their browser, phone, and desktop.

Web access	Users are able to access the web services with HTTP protocol using their browsers.
Mobile access	Android is an open source operating system developed by Google that is used by 84% of worldwide smartphone.  iOS is another operating system developed by apple to be used in their smartphones.
Desktop access	A program that runs directly on top of operating system.

**Table 3: Client Access**





As we have a time frame to respect, it would be highly recommended to limit its scope to only android development since the project is highly depending on mobility criteria of smartphone. However, the project can expend in the future to include web and desktop access.

In order for the server and client to communicate a protocol supported by both sides is needed. Simple Object Access Protocol (SOAP) and REpresentational State Transfer (REST) are two answers to the same question: how to access Web services. The choice initially may seem

easy, but at times it can be surprisingly difficult. In this project, I would rather use REST as it's supported by node.js and android, and also, to discover this protocol.

#### 4. Conclusion

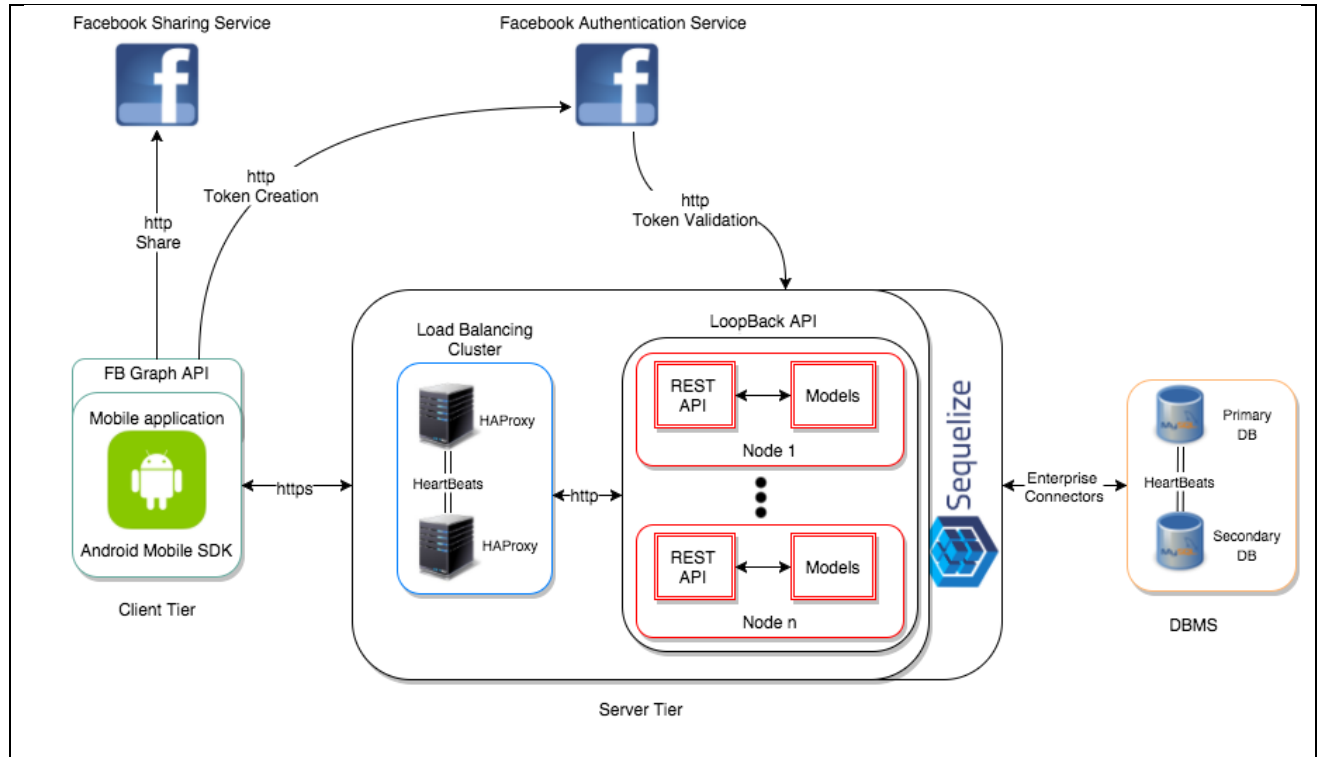
The choice of a technology is always relative to the application as there is no best technology, but only suitable ones. For this project, here are the technologies that we will adopt in designing the architecture of our application as well as implementing it:

DBMS	 MySQL	
Server-Side	 Node.js	 & Express (web framework)
Client side	 Android developer	

**Table 4: Technologies to be used**

# System Architecture and Design

## 1. System Architecture



**Figure 2: System architecture**

The system design follows the standard approach of the three tiers architecture; a client tier, a server tier, and an EIS tier. The design was produced with all the non-functional requirements in mind.

For reliable enterprise information system architecture, the tier has two databases one replicates the other. The two databases keep synchronizing data between each other to insure consistency. The first database receives queries from the system tier to persist and process while the other stands by. The two databases exchange heartbeats so that each one of them knows that the other didn't fail. If the primary database fails, the secondary one replaces it and continues the

data persistency and processing. This insures that the system information system is always up and running.

The system tier includes a load balancer and a loopback API server. The load balancer which is an HA proxy distributes the load of the coming connections over the clusters of the loopback server. The HA proxy insures then the system reliability and provides a higher performance to the system by distributing the work over the clusters. The clusters, also, provide a high scalability as we need to add one more node if the number of users increases. Moreover, the server side exposes a RESTfull API that can be accessed by any REST client.

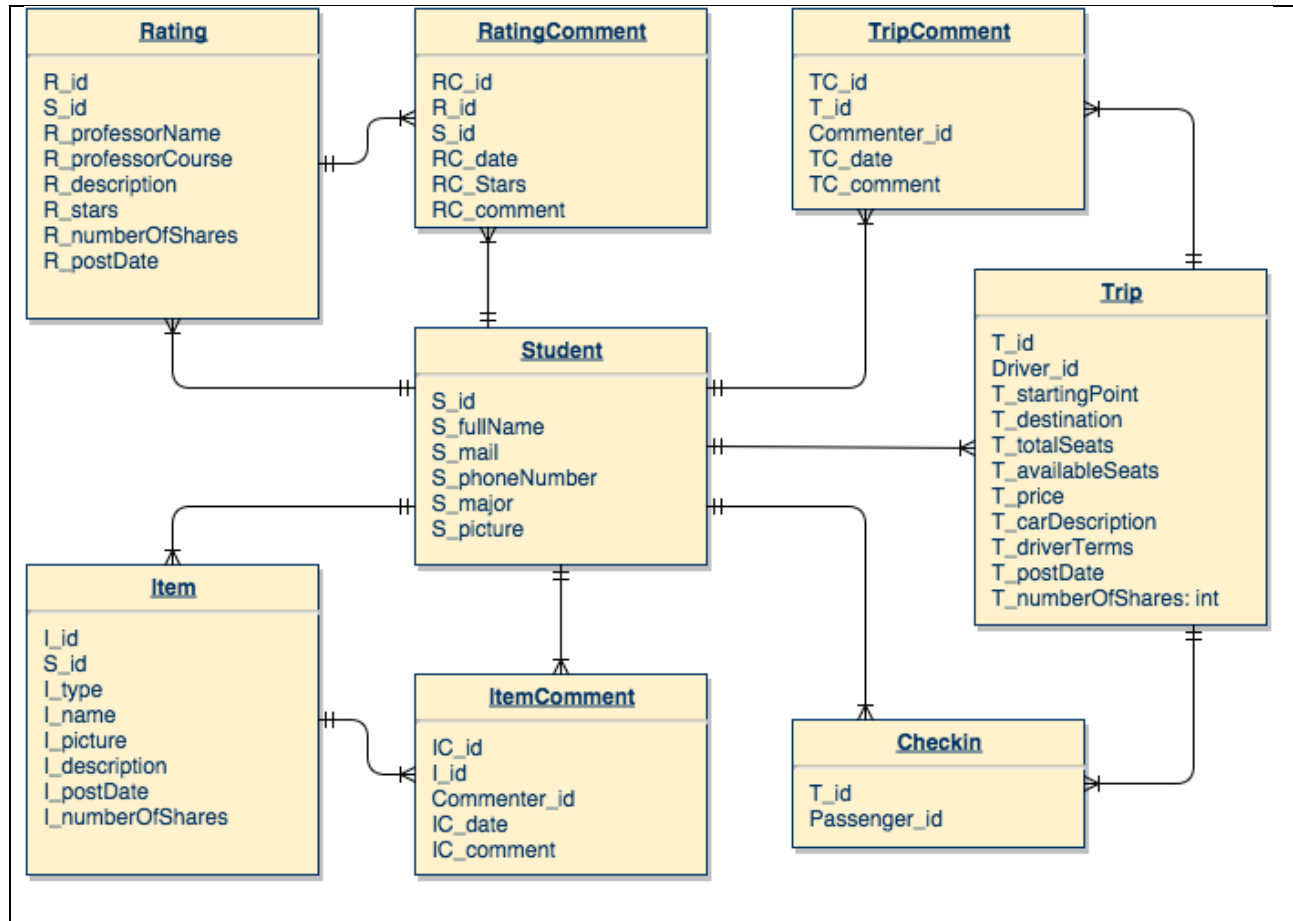
The client tier is an android application as we limited the scope of the application to cover only android mobiles. Since the server and the client sides both support REST then it's wise to use REST to communicate data between the two tiers.

In addition to all that, the system shall use some of the Facebook services as a third party services. For the time being, we are to use Facebook authentication service for users to authenticate using their Facebook accounts. Another third party service is the Facebook sharing service that is linked to the android client tier to allow our users to share any post on their social media accounts.



## 2. System Design

### 2.1. Enterprise Information System – Entity Relation Diagram

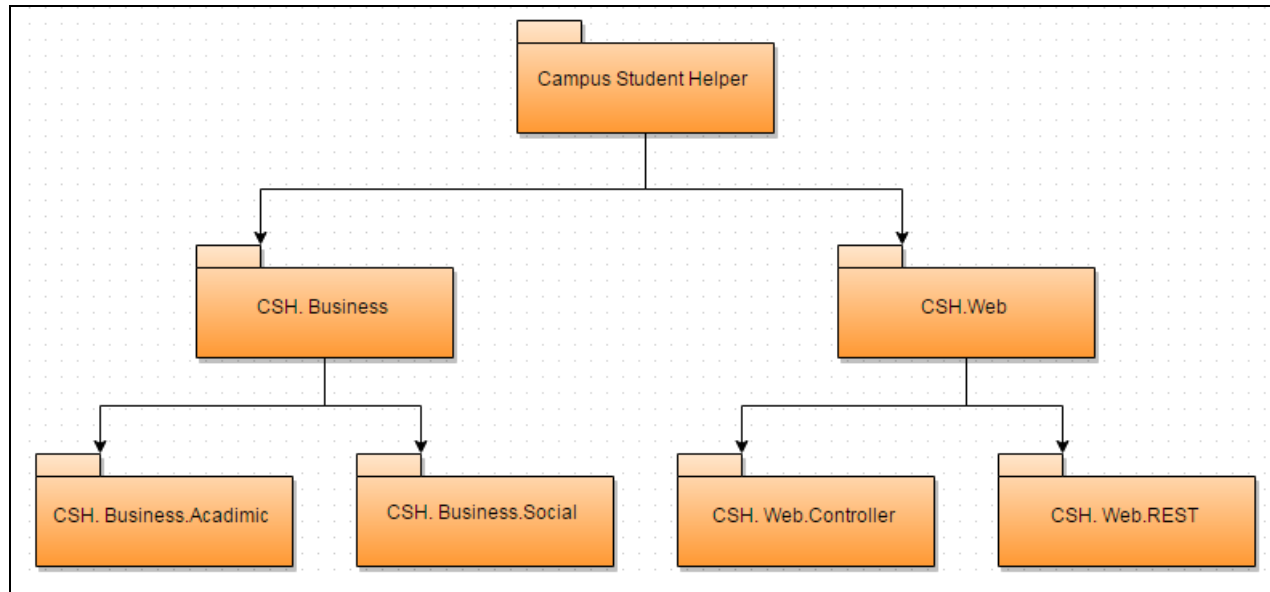


**Figure 3: Entity Relation Diagram**

The Enterprise Information System is composed of 8 tables. Student table, is the table that holds all information related to each student. Then, there are 3 tables linked to the Student; Item, Trip, and rating with a 1 to many relationship; for instance, a student request many ratings, and every rating is requested by one student. Also there are 3 other tables for comment management: ItemComment, RatingComment, TripComment, and check in. Those tables are bridge tables between item and student, Rating and student, and trip and student (many-to-many relationships). Figure 3, shows all tables with their attribute and relationships in an Entity Relation Diagram.

## 2.2.Server Side Design

### 2.2.1. Package Hierarchy



**Figure 4: package Hierarchy**

The package hierarchy shows all the packages that the system has. Indeed, the system follows the MVC framework with model, controllers and views. The model is illustrated under the CSH.Business which in its turn include a package holding academic services and another holding the social services. In addition, the controller that controls the system users' access to the data is implemented under the CSH.Web. In addition, the system is exposed through a RESTfull API considred as view.

## 2.2.2. Class diagram

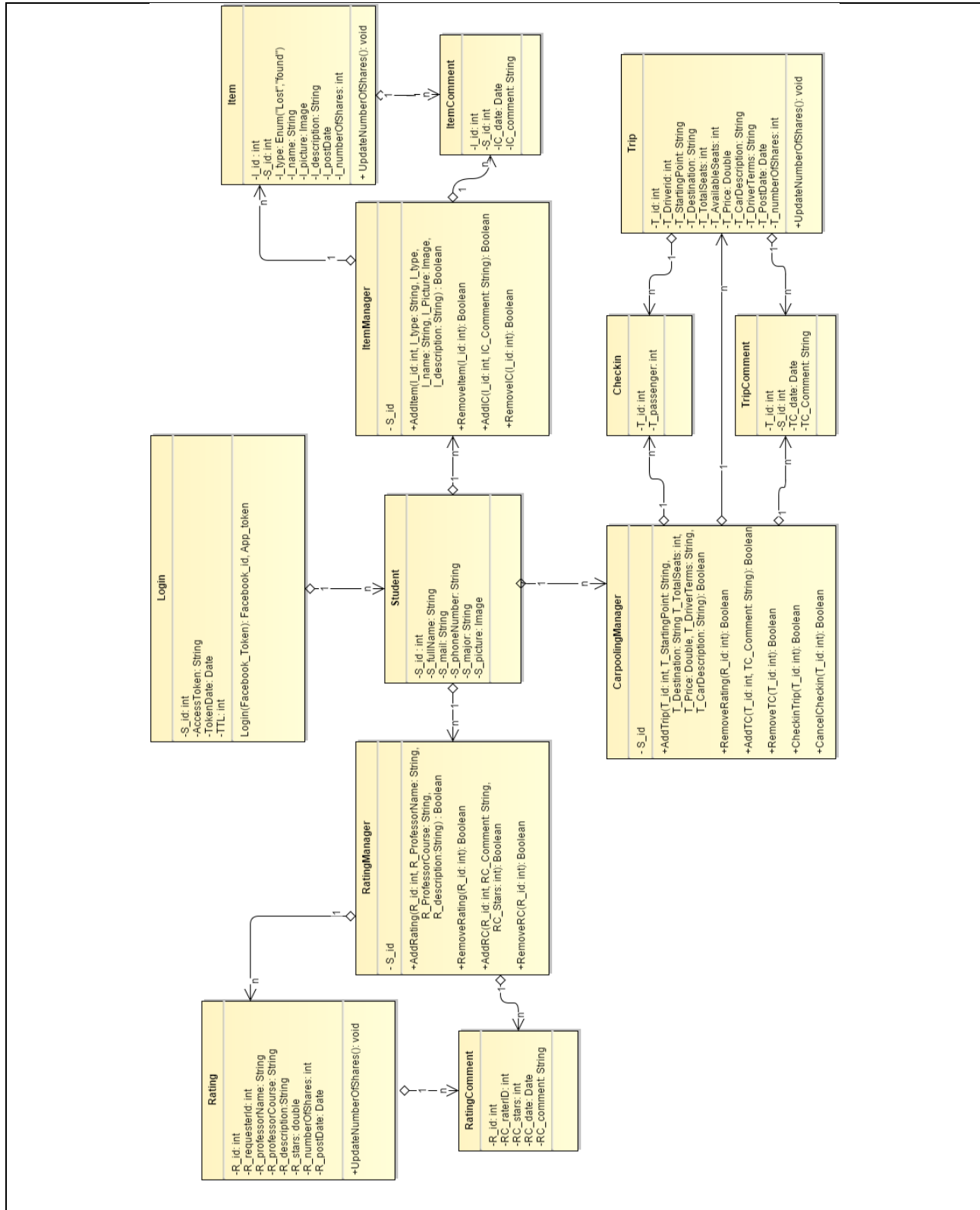


Figure 4: Class Diagram

The class diagram, figure 3, shows the different classes that our system is composed of. There are 8 classes representing the tables from the database and persisted using the SequelizeJS technologies as an ORM provider. Those classes are named as their names in the database; Student, Item, Rating, Trip ItemComment, RatingComment, Check-in, and TripComment. In addition to those there are 3 classes for the management of the services the system provides; CarpoolingManager, ItemManager, and RatingManager. Those 3 classes allow clients to add, delete, and edit any of their post in addition to commenting on other posts or check-in a trip. Table 5 describe the system classes with their attributes and methods.

<i>Class</i>	<i>Method</i>	<i>Description</i>	<i>Input</i>	<i>Output</i>
<i><b>Login:</b> the class that manages clients authentication to the system</i>	<b>Login</b>	The login method allows users to gain access to the application through the Facebook Authentication on server.	<b>Facebook_Token:</b> the actual Facebook token received from Facebook Authentication server after successfully Authenticating.	<b>Facebook_id:</b> the id of the user on Facebook. <b>App_token:</b> an token created to every specific user authentication.
<i><b>RatingManager:</b> A class that manager the students requests to rate professors so as to manage</i>	<b>AddRating</b>	The method allows to add new request to rate a professor	<b>R_id:</b> reference to request id <b>R_ProfessorName:</b> professor name <b>R_ProfessorCourse:</b> professor course <b>R_description:</b> additional description <b>R_Stars:</b> the average stars of all ratings	<b>Boolean:</b> indicate if the request succeeded or failed

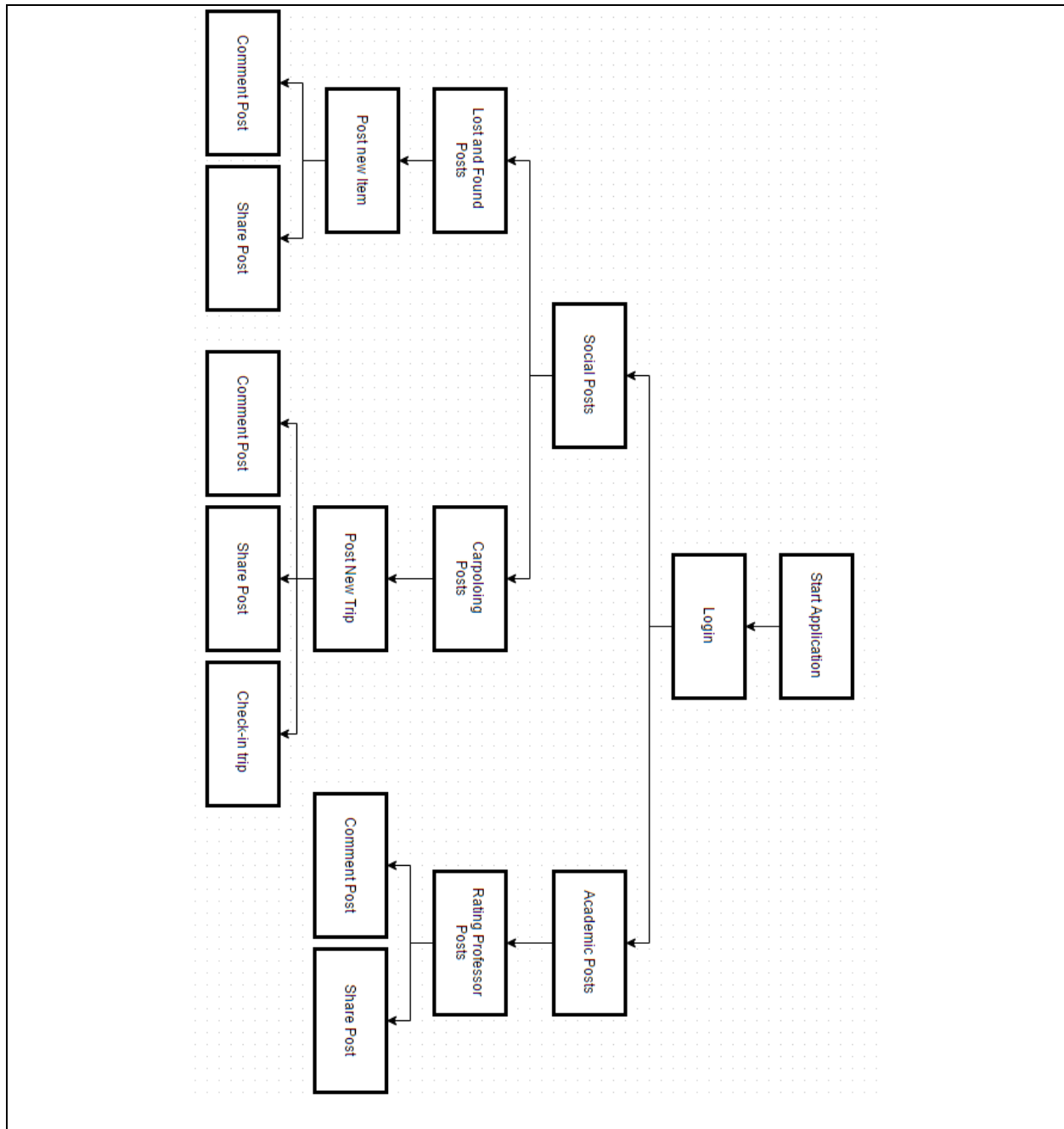
<i>the student's ratings.</i>	<b>RemoveRating</b>	The method allows to remove a request for rating a professor	<b>R_id:</b> reference to the request ID	<b>Boolean:</b> indicate if the request succeeded or failed
	<b>AddRC</b>	The method allows to add new rating comment that actually holds the rating	<b>R_id:</b> reference to the request ID <b>RC_Comment:</b> the actual comment	<b>Boolean:</b> indicate if the request succeeded or failed
	<b>RemoveRC</b>	The method allows to remove a rating comment	<b>R_id:</b> reference to the request ID	<b>Boolean:</b> indicate if the request succeeded or failed
<b>ItemManager:</b> <i>A class that manager the students requests to rate professors so as to manage the student's ratings.</i>	<b>AddItem</b>	The method allows to add new lost or found item	<b>I_id:</b> reference to item ID <b>I_type:</b> type of item (Lost / found) <b>I_name:</b> Item name <b>I_Picture:</b> Item picture <b>I_description:</b> description of the item	<b>Boolean:</b> indicate if the request succeeded or failed
	<b>RemoveItem</b>	The method allows to remove a lost or found item	<b>I_id:</b> reference to item ID	<b>Boolean:</b> indicate if the request succeeded or failed
	<b>AddIC</b>	The method allows adding comments to Item posts.	<b>I_id:</b> reference to item ID. <b>IC_Comment:</b> The actual comment.	<b>Boolean:</b> indicate if the request succeeded or failed.

<b>Carpooling Manager:</b> the class that manages users' trips, check in, and comments.	<b>RemoveIC</b>	The method allows removing item comments.	<b>I_id:</b> reference to item ID.	<b>Boolean:</b> indicate if the request succeeded or failed.
	<b>AddTrip</b>	The method allows to add new trip.	<b>T_id:</b> reference to trip ID <b>T_StartingPoint:</b> the meeting point or city <b>T_Destination:</b> additional description <b>T_TotalSeats:</b> Total number of seats. <b>T_Price:</b> Trip price <b>T_DriverTerms:</b> additional drivers' terms. <b>T_CarDescription</b> : car description.	<b>Boolean:</b> indicate if the request succeeded or failed.
	<b>RemoveRating</b>	The method remove an existing trip.	<b>T_id:</b> reference to trip ID.	<b>Boolean:</b> indicate if the request succeeded or failed.
	<b>AddTC</b>	The method allows to add new trip comment to ask about more information.	<b>T_id:</b> reference to Trip ID. <b>TC_Comment:</b> The actual comment.	<b>Boolean:</b> indicate if the request succeeded or failed.
	<b>RemoveTC</b>	The method allows to remove a trip comment.	<b>T_id:</b> reference to Trip ID.	<b>Boolean:</b> indicate if the request succeeded or failed.

<i>Trip, item, and rating: 3 entity classes persisted with the database</i>	<b>CheckinTrip</b>	The method allows users to check in a selected trip.	<b>T_id:</b> reference to Trip ID.	<b>Boolean:</b> indicate if the request succeeded or failed.
	<b>CancelCheckin</b>	The method allows users to cancel their check in trip.	<b>T_id:</b> reference to Trip ID.	<b>Boolean:</b> indicate if the request succeeded or failed.
	<b>UpdateNumberOfShares</b>	The method updates the number of shares related to an academic or social post	<b>NO INPUT PARAMETER</b>	<b>Void</b>

**Table 5: System Methods**

### 2.3. Client Side Design



**Figure 5: System Activity**

This activity diagram, figure 5, shows the different activities that a user may do using there android application. The user first get into a logging activity where he/she log-in to the system using his/her Facebook account. Then he can navigate through social or academic posts. From



there, there variety of choses to make; new post, comment on an existing post, share in social media, and so on. Table 6 shows the different activities that the system shall have.

Level	Activity name	Activity description
1	<b>Login Activity</b>	The user's authentication process that goes through Facebook Authentication Server. The user provides his Facebook identification username and password to be send to the Facebook Server. Then, if authenticated the Facebook Server responds to the system server with an access token to guarantee a secure authentication to the user.
2	<b>Social Posts</b>	Once authenticated the user can view all social posts choose to do one of the underneath activities
2	<b>Academic Posts</b>	Once authenticated the user can view all social posts choose to do one of the underneath activities
3	<b>Lost and found posts</b>	The activity where student can search for a lost or found post, comment a post, or add a new lost or found post.  In this activity, student can also share a post on social media.
3	<b>Carpooling posts</b>	This activity allows student to add a new trip setting its attributes. Then other student may check in this trip or ask for additional information through adding comments.  Also, student may choose to share a post on social media.
3	<b>Rating Teachers posts</b>	This activity allows student to exchange detailed information about any teacher and the course he teaches, in addition to sharing the post in their social media

**Table 6: activities description**

# Implementation and Testing

In this project the implementation process went hand in hand with testing; each requirement was implemented then tested afterward. This way fixing bugs and errors was easier, and made me focus on each requirement individually.

## 1. Database Management System

The implementation of the database was through MySQL Workbench 6.3 E. figure 6 shows the EER diagram generated by MySQL workbench after implementing all tables showing primary keys (key), foreign keys (red diamond), and all other attributes:

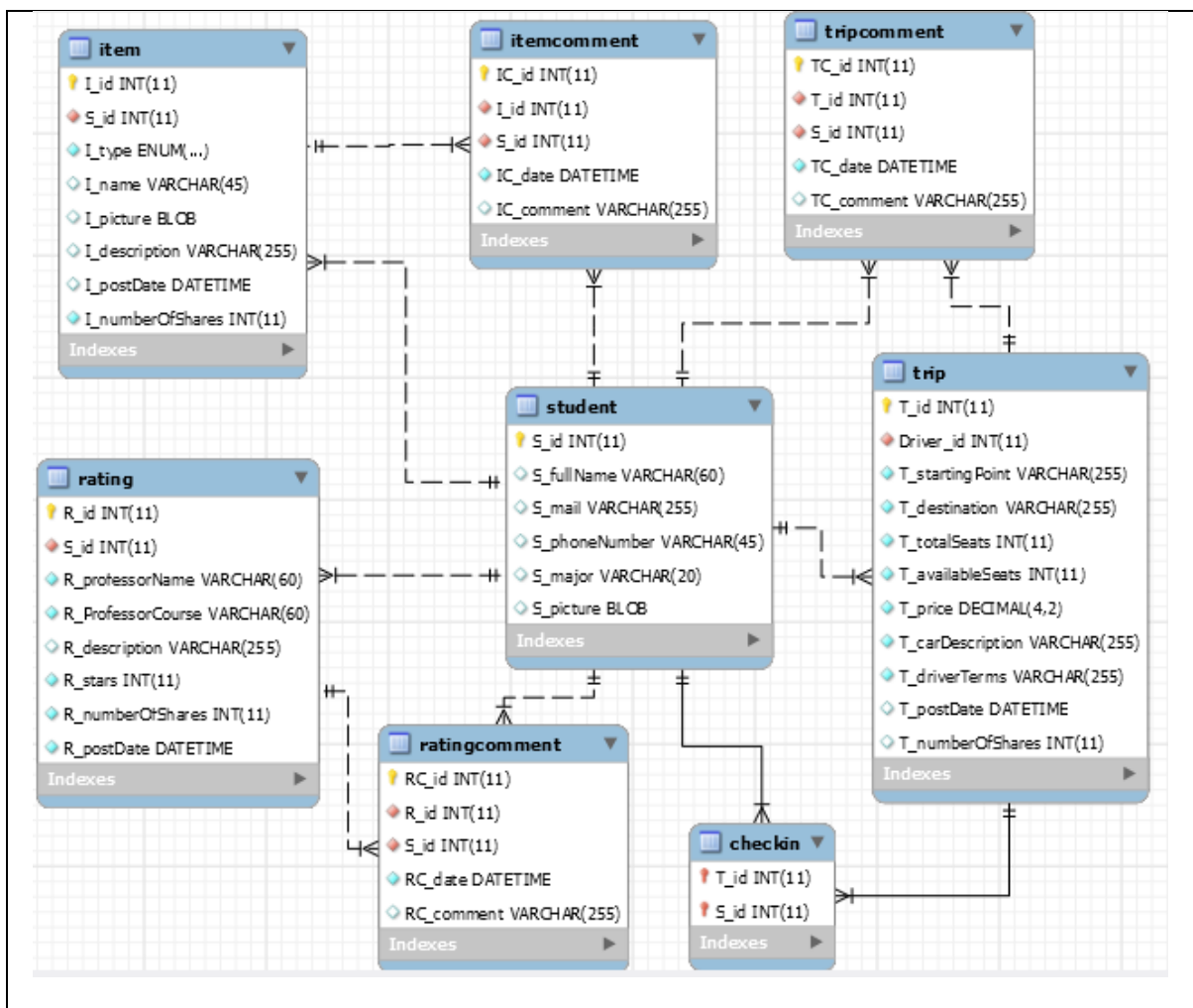
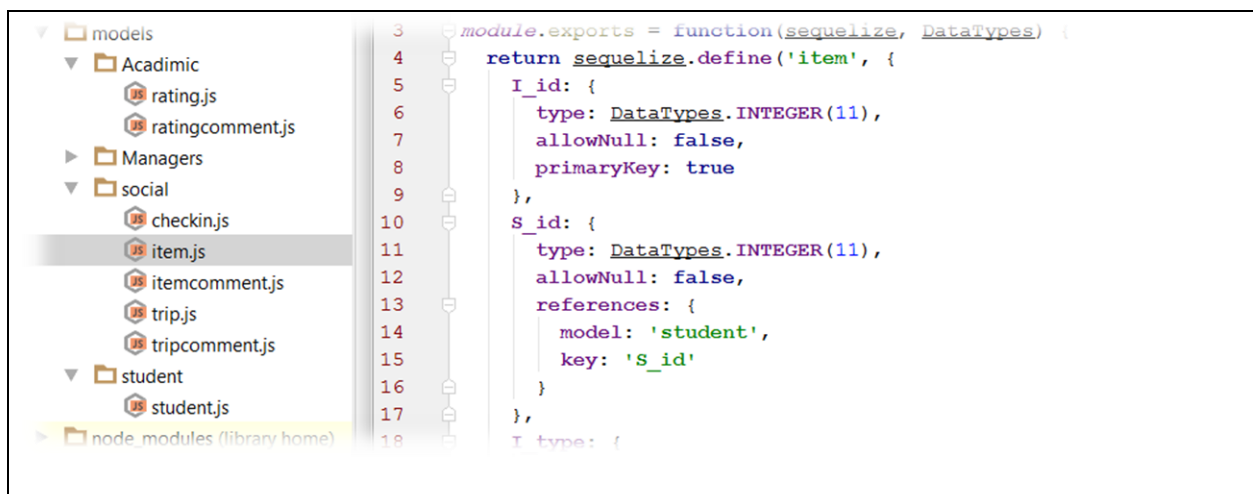


Figure 6: EER diagram

## 2. Server Side

Server side was implemented with NodeJS and Express framework on top of windows 8. Webstorm 11.0, the student edition, was used as an IDE for implementing the server modules and wrapped them by RESTfull API. The server communicate and persist data with MySQL server through SequelizeJS.



**Figure 7: Sequelize modules; Item module example**

Figure 7 shows all sequelize modules and the package under which they were created in addition to a part of the function defining the item model. Those models are the same as the ones in the database with the same names: student.js, item.js, itemcomment.js, tripcomment.js, checkin.js, rating.js, and ratingcomment.js.



**Figure 8: Managers Modules; login module example**

Figure 8 illustrate the package path of the system managers. The system have 4 managers. ItemManager.js is in charge of managing the users lost and founds items; it provides functions to add or delete, if authorised, any item post or comment. TripManager.js manages the student carpooling trips allowing users to add, delete, checking, or comment a trip post. RatingManager.js allows students to exchange ratings on teachers and courses through requesting a new rating, or rate an already requested rating. Login.js allows users to authenticate to the system and use its functions. All of the above was wrapped using REST API.

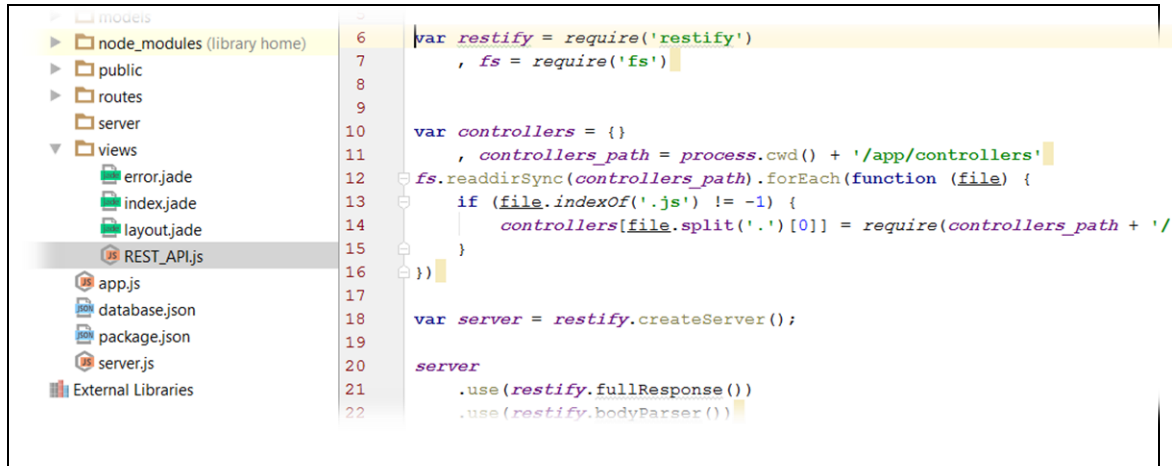
```

"dependencies": {
  "body-parser": "~1.13.2",
  "cookie-parser": "~1.3.5",
  "debug": "~2.2.0",
  "express": "^4.13.3",
  "jade": "~1.11.0",
  "morgan": "~1.6.1",
  "mysql": "^2.9.0",
  "sequelize": "^3.14.2",
  "sequelize-cli": "^2.2.1",
  "serve-favicon": "~2.3.0",
  "facebook-client": "~1.6.1"
}

```

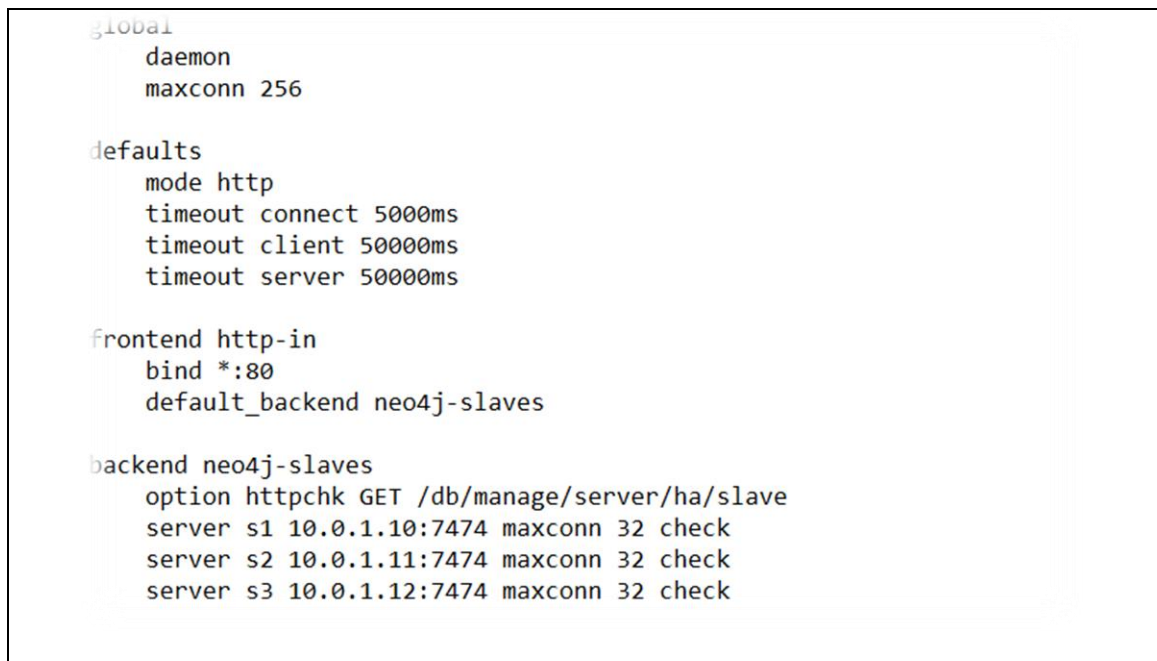
**Figure 9: System Dependencies**

Figure 9 shows the system dependencies; some dependencies were installed with node and express while the others were installed using NPM such as sequelize v3.14, mysql v2.9, and facebook-client 1.6.



**Figure 10: REST API**

The business functions (managers), shown in figure 8, were exposed through a RESTful API.



**Figure 11: Part of load balancer configuration file**

Figure 11 shows a small part of the load balancer configuration file; the system has 2 load balancers running on top of the UNIX kernel. The figure is a screen shot of the HAProxy.cfg configuration file.

### **3. Client Side**

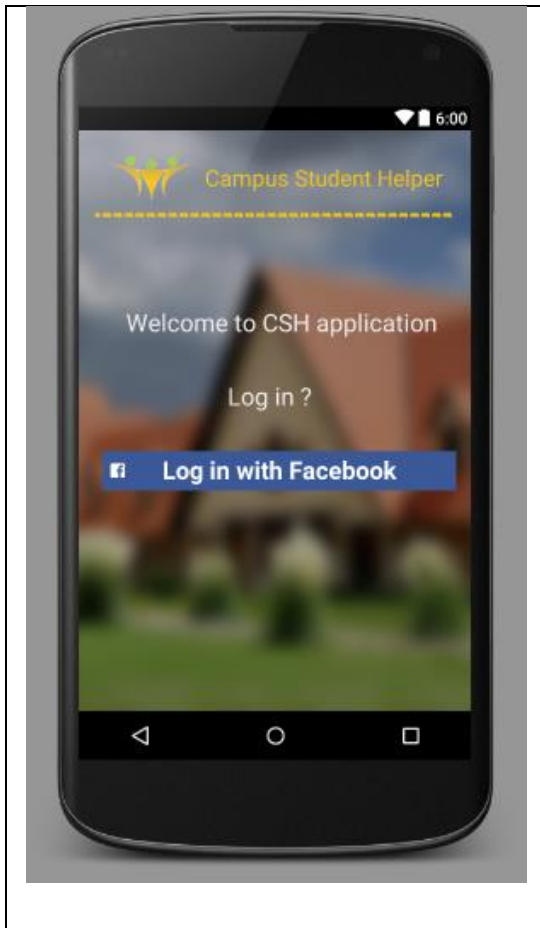
As we limit the scope of the application to cover only clients with android mobile access, the android application was implemented using android studio with Android SDK tools that uses currently Android 6.0 platform. Figures on the results section shows the actual interfaces of the client application.

### **4. Testing**

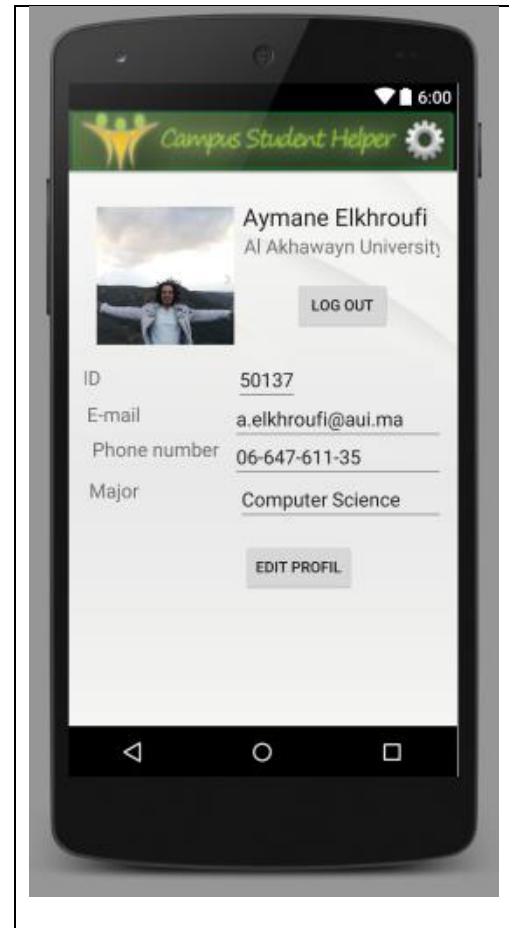
To test the system, unit testing techniques were used where we provide some input to the system and compare the actual output with the expected output. This technique was used to test every single method on the system including client and server side implementations. The system shall not be deployed unless every method passes the unit testing.

### **5. Results**

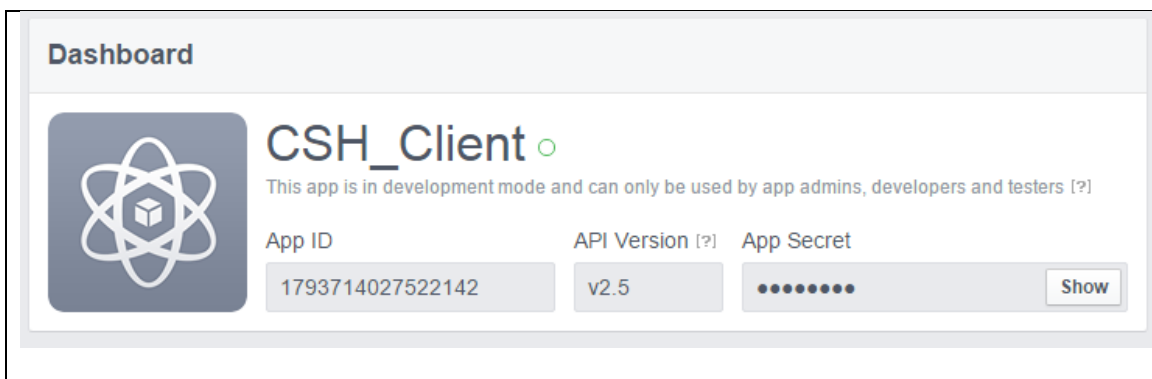
Figures from 13 to 18 shows the actual results from a running android application on a Nexus 4 (4.7") and Nexus 5X (5.0")



**Figure 13: Login Page**



**Figure 14: Profile Page**



**Figure 15: Facebook application ID**



**Figure 16: A Found Items Page**



**Figure 17: Navigation Panel**



**Figure 18: Shared post via CSH\_Client application**



## **Conclusion**

The Campus Student Helper application respects all enterprise class application standards. Therefore, the application is highly scalable, extensible, and performing. In addition, it's support high security and availability standards.

This project result is an android application as a client application connected to high functioning server ready to be deployed on Google Play Store.

I'm grateful for this opportunity that gives me the chance to discover further the enterprise class application development field as it's a very promising field.

## References

- [1] Somerville, Ron. *Software Engineering*. Harlow, England; New York; Addison-Wesley, 2000
- [2] Benjamin San Souci and Maude Lemaire, *An Inside Look at the Architecture of NodeJS*, McGill University. Retrieved from: [mcgill-csus.github.io/student\\_projects/Submission2.pdf](https://mcgill-csus.github.io/student_projects/Submission2.pdf)
- [3] Azat Mardan. *Express.js Guide, The Comprehensive Book on Express.js*. Leanpub. 2014. Retrieved from: [samples.leanpub.com/express-sample.pdf](https://samples.leanpub.com/express-sample.pdf)
- [4] Mark L. Murphy. *The Busy Coder's Guide to Android Development*. CommonsWare, LLC. 2008.
- [5] Leonard Richardson and Sam Ruby. *RESTful web services*. O'Reilly Media 2007.
- [6] *Enterprise Applications Testing –Leverage the Power of Objectivity*. OraclePeopleSoft - ERP White Paper Series. 2010