GRADEBADGE

DEVELOPMENT OF A CLOUD-BASED REWARD APPLICATION

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Erwin Toni Soekianto

April 2013

GRADEBADGE

DEVELOPMENT OF A CLOUD-BASED REWARD APPLICATION

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Erwin Toni Soekianto

April 2013

Approved by:

David Turner, Chair, Computer Science and Engineering	Date
Richard J. Botting	
Arturo I. Concepcion	

ABSTRACT

The purpose of this project is to investigate the use of cloud-based services to deliver cutting-edge applications. For this purpose, a prototype of a reward application using badges was developed to illustrate and explore this emerging paradigm. The cloud services utilized by this application include Heroku (for the application server), MongoLab (for the database), and Facebook (for authentication and social network integration). The application server is written in Javascript and runs inside a Nodejs execution environment. The application is accessed through a web browser running on either desktop or mobile computers.

On the client side, this application makes use of numerous web technologies, including HTML5, CSS, Bootstrap, and jQuery. The project also made use of the git version control system to manage source code and deployment of the application server to Heroku. The source code repository was stored remotely through the cloud-based service called GitHub.

The purpose of the GradeBadge application is to help organizations interact with and motivate their members in a fun way. It keeps the members engaged by giving badges as rewards for their efforts or achievements. In order to facilitate adoption amoung users, GradeBadge is integrated with the social networking site Facebook.

ACKNOWLEDGEMENTS

I would like to thank all the people with whom I have worked while pursuing my master's degree at California State University, San Bernardino (CSUSB). Studying in the School of Computer Science and Engineering at CSUSB has been a great learning experience. I would like to thank the faculty of the School of Computer Science and Engineering who supported this project by serving on my committee: Dr. David Turner, Dr. Arturo Concepcion and Dr. Richard Botting.

TABLE OF CONTENTS

Abstract iii

Acknowledgements iv

List of Tables ix

List of Figures x

1. Introduction 1

1.1 Background 1

1.2 Technology Overview 3

1.2.1 Facebook 3

1.2.2 Heroku 3

1.2.3 MongoDB 3

1.2.4 MongoLab 4

1.2.5 Git 4

1.2.6 Bootstrap 4

1.2.7 Jquery 4

1.2.8 Node.js 4

1.3 Project Purpose 5

1.4 Project Scope 5

1.5 Related Work 5

1.6 Definitions, Acronyms, and Abbreviations 6

2. Software Requirements Specification 8
2.1 External Interfaces Requirement 8
2.1.1 Hardware Interfaces 8
2.1.2 Software Interfaces 8
2.1.3 Communication Interfaces 8
2.2 Functional Requirement 9
2.2.1 Create group 9
2.2.2 Create Badge 10
2.2.3 Add Member 10
2.2.4 Issue Badges to Members 10
2.2.5 View Badge Earned 10
2.2.6 Share Badge to Social Networking 10
2.3 Performance Requirement 10
2.4 Design Constraint 10
2.5 Software System Attributes 11

3. System Architecture 12
3.1 Overview 12
3.2 Deployment Workflow 13
3.3 Developer/Release 14
3.4 Heroku 15
3.5 MongoLab 15
3.6 Facebook 15
3.7 Client Browser 16

4. System Design 17
4.1 Design Overview 17
4.2 Model View Controller Architecture 17

4.3 Server-side Architecture Design 18
4.4 Mapping of Model Classes to MongoDB 19
4.5 Request Handler Operation 20
4.6 Client-side Architecutre Design 20

5. Database Design 22
5.1 Overview 22
5.1.1 MongoDB 22
5.1.2 MongoLab 22
5.1.3 Documents 22
5.1.4 Collections 23
5.2 Data Model 23
5.2.1 User Collection 24
5.2.2 Group Collection 24
5.2.3 Badge Collection 24
5.2.4 Group Admin Links Collection 25
5.2.5 Group Member Links Collection 25
5.2.6 Badge User Links Collection 26
5.2.7 Group Badge Links Collection 26

6. Project Implementation 28
6.1 Loading Screen 29
6.2 Login Screen 30
6.3 Group Page 32
6.4 Add New Group Page 33
6.5 Group Page 34

7. Conclusion and Future Direction 35

7.1 Conclusion 35
7.2 Future Direction 35

APPENDIX A: SOURCE CODE 37

References 39

LIST OF TABLES

5.1 User Collection 24
5.2 Group Collection 24
5.3 Badge Collection 25
5.4 Group Admin Links Collection 25
5.5 Group Member Links Collection 26
5.6 Badge User Links Collection 26
5.7 Group Badge Links Collection 27

LIST OF FIGURES

2.1 Use Case Diagram 9

3.1 Deployment Diagram 12

3.2 System Integration 13

6.1 GradeBadge Loading Screen 29

6.2 GradeBadge Login Screen 31

6.3 GradeBadge Group Page 32

6.4 GradeBadge Add New Group Page 33

6.5 GradeBadge Group Page Added 34

1. INTRODUCTION

1.1 Background

A long time ago, businesses used to produce their own electric power. Due to an engineering breakthrough in electric generator and transmission methods, it became easier to produce and transmit electricity to businesses that once produced their own electricity. As more businesses started buying electric power, the production of electricity become less expensive, encouraging more compnies to purchase rather than make their own electricity.

And today, just like the utilities, instead of buying servers to run websites and applications, businesses rent servers and various services from cloud computing providers. These cloud resources are provided to companies in a way that resembles people renting apartments in a single building; even though you are in the same building with other people, you still have your own space. As more people rent and buy computing service from large-scale providers, the cost of these services are decreasing.

Today there are many cloud computing providers that provide different types of services. Some provide application hosting, databases, code repositories, authentication services, social network integration, etc. Some famous providers are Amazon, Google, Microsoft, Facebook, GitHub and Heroku.

Google App Engine provides infrastructure to build web applications on the same scalable systems that power Google applications. Google App Engine supports Python, Java and the Go programming languages. Google App Engine also provides several options for storing data, including App Engine Datastore, Google Cloud SQL, and

Google Cloud Storage. Windows Azure provides similar services as Google App Engine but supports a different set of programming languages, including ASP.NET, VB.NET, Java, Nodejs and Python.

Amazon also provides scalable cloud computing services, which includes the popular EC2 virtual machine instances, where users choose the type of operating system and configuration they need. The virtual machine service provided by Amazon is more flexible than language-specific executation environments such as Google App Engine and Heroku, but require more time and expertise to set up and manage.

In this project, Heroku is used as a cloud application platform for running Javascript in the Nodejs executation environment. Heroku also provide alternative execution environments that support Scala, Ruby, PHP and others. One of the benefits of using Heroku (and other similar application hosting services) is that the bandwidth and CPU capacity can be scaled up or down almost instantly to accomodate rapidly changing demand. Also, by relying on Heroku fto manage the hardware and system and network administration, developers gain the reliability, performance and security that is provided by a larger company with staff dedicated to these purposes.

Among all the programming language execution environments supported by Heroku, this project uses Nodejs, which supports server-side programming in Javascript. On the client side, the application uses HTML5, Javascript, Jquery library and the Bootstap framework. The applciation also relies on the MongoDB database as provided by the MongoLab service provider.

Node.js and MongoDB are often used together to build scalable web applications. The following describe the cloud computing service providers used in this project.

1.2 Technology Overview

1.2.1 Facebook

Facebook is a popular social networking website more than one billion users. It has proven to be a good platform to use for web applications that take advantage of its social networking features, such as friend lists, wall posting, etc [8]. Facebook also allows other applications to access user data with their authorization using its API.

1.2.2 Heroku

Application hosting is a form of cloud computing that enables developers to publish applications that require Internet connectivity. Heroku is one of the first companies to offer a remote Nodejs executation environment. Nodejs applications are deployed to Heroku using git [12]. To depoy, or redeploy, an applicaiton, the developer pushes a branch of a git repository to a remote git repository provided by Heroku. At deployment, Heroku extracts the application files from the git repository and runs the main executable, which is a server process. Heroku work is a partner of Facebook and so is designed to work easily with it.

1.2.3 MongoDB

There are many different types of cloud-based datastore services to choose from. For this project we used MongoDB because it works well with Node.js and Heroku. MongoDB is a scalable, high-performance, open source, NoSQL document-based database. MongoDB features include document-oriented storage, indexes, replication, high availability, auto-sharding, and querying [21].

1.2.4 MongoLab

MongoLab is the cloud computing provider for MongoDB database that was used in this project [22]. MongoLab has a free tier service that facilitates experimentation by developers, and thus was convenient for this project.

1.2.5 Git

Git is a distributed version control system [9]. This project uses GitHub to store and manage a remote git repository of application source code. Git was convenient for the project because Heroku uses git as a means to deploy web applications to its servers. Heroku's git-based deployment system allows easy creation of testing, staging, and production versions of the application.

1.2.6 Bootstrap

Bootstrap is a freely avaialble CSS and Javascript library created by Twitter. It provides a responsive design framework that works well for applications that run inside browsers in desktop computers, tablets and smart phones [4]. The application's user interface was contructed using Bootstrap.

1.2.7 Jquery

The GradeBadge application uses Jquery for AJAX transactions and DOM manipulation [19]. Bootstrap also depends on Jquery.

1.2.8 Node.js

Cloud-based services support apps written in several different programming languages, such as Java, Python, PHP, Javascript, Ruby and more. For this project we used Javascript running in a Node.js context. Node.js is a platform built on

Chrome's JavaScript runtime. It's purpose is to allow construction of fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient when used for I/O intensive applications such as Web applications [23].

1.3 Project Purpose

The purpose of this project is to explore the current technologies that enable rapid development and deployment of desktop and mobile Web applications that can scale to accomodate any number of users.

1.4 Project Scope

Project does not include database sharding features of MongoDB, which would allow a greater degree of scalability.

The GradeBadge application provides the following functionalities:

- Create Group
- Create Badge
- Add Member
- Issue Badges to Members
- View Badges Earned
- Share Badges with Social Networking Contacts

1.5 Related Work

There are other similar mobile Web application and cloud computing demonstration projects that were built as master's degree projects at CSUSB. One of these projects

was completed by Manoj Kulkarni [32]. This project used Google App Engine for an application hosting provider and Google App Engine datastore and Java programming language. It also used Jquery Mobile as UI framework. The GradeBadge project uses different set of technologies, and is the first project at CSUSB that utilizes Nodejs and MongoDB. Most other similar projects at CSUSB have relied on either Java, .NET or PHP.

1.6 Definitions, Acronyms, and Abbreviations

The definitions, acronyms, and abbreviations used in the document are described in this section.

- GradeBadge: The name of this project.
- API: Application Programming Interface, which is a set of routines that an application uses to request and carry out low-level services performed by a computer's operating system; also, a set of calling conventions in programming that defines how a service is invoked through the application [10].
- Cloud computing: the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet) [**?**].
- JQuery: A javascript library for building web based applications [19].
- DOM: Document Object Model, which is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents [5]. item I/O: Input/Output.
- UI: User Interface.
- CSUSB: California State University, San Bernardino.

- HTML: HyperText Markup Language, which is the authoring language used to create documents on the Web [29].
- HTTPS: Hyper-text Transfer Protocol Secure, which is a secure network protocol used to encrypt data transferred between server and client [14].
- MVC: Model-View-Controller is an architectural pattern used in software engineering to isolate business logic from user interface considerations [33].
- UML: The Unified Modeling Language, which is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems [28].
- Microsoft Azure: Cloud computing platform provided by Microsoft [31].
- Google App Engine: Cloud computing platform provided by Google.
- Amazon Web Services: Cloud computing platform provided by Amazon [2].
- Heroku: Cloud application platform provided by Heroku [12].
- Android: Mobile operating system provided by Google [3].
- IOS: Mobile operating system provided by Apple [16].
- NoSQL: Types of databases that use key-value pairs for storing data unlike traditional relational databases. [24].
- JSON: Javascript Object Notation, which is a data representation format using key-value pairs [20].
- Ajax: Asynchronous JavaScript and XML, which is a method for Web applications to communicate between client browsers and servers [1].
- OOP: Object Oriented Programming, which is a computer programming paradigm where application logic is organized into objects that provide an interface to encapsulated state. [30].

2. SOFTWARE REQUIREMENTS SPECIFICATION

2.1 External Interfaces Requirement

2.1.1 Hardware Interfaces

The application is hosted in the Heroku application cloud service. The Web server communicates over HTTPS to ensure that data transferred between client and server is untampered and private. The system is a Web based application; users are required to use a high-speed Internet connection and use an up-to-date Web browser.

2.1.2 Software Interfaces

JavaScript will be implemented throughout the website in order to display the correct feature the user requested. And HTML5 may be implemented throughout the website in order to display the correct feature the user requested.

2.1.3 Communication Interfaces

This application is designed to be viewed on any Internet Web browser provided that JavaScript and image features are enabled and the browser is HTML5 compatible. Performance may vary slightly between browsers; however, the functionality of the site should not be impaired.

2.2 Functional Requirement

The functions specified in this section directly correspond to work that will be conducted in this project as shown in the use case diagram in Figure 2.1.

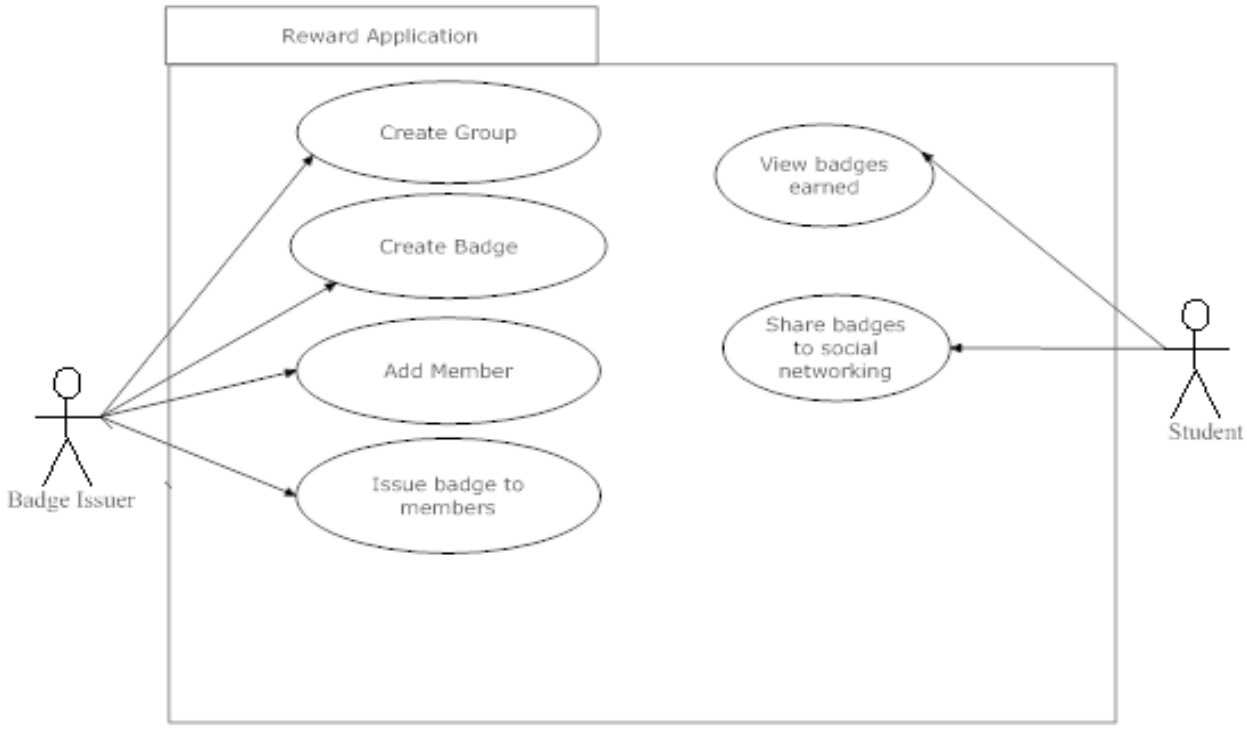

Fig. 2.1: Use Case Diagram

2.2.1 Create group

This functionality allows organizers or badge issuers to create groups, which will have a set badge collection.

2.2.2 Create Badge

This functionality allows badge issuers to create badge to be added to badge collection. A badge would have at least badge name and description.

2.2.3 Add Member

This functionality would allow organizers or badge issuers to add members into the groups as badge recipients. Member would have at least email address and name.

2.2.4 Issue Badges to Members

This functionality would allow badge issuers to issue badges to members.

2.2.5 View Badge Earned

This functionality would allow badge recipients to view all the badges that they have earned.

2.2.6 Share Badge to Social Networking

This functionality would allow badge recipients to share the badge to their social networking site such as Facebook

2.3 Performance Requirement

This application is going to be hosted in the Heroku cloud server, so the performance of this application would be high.

2.4 Design Constraint

This application requires internet-enabled devices and internet connection to perform. And every user must have Facebook account to be able to use this.

2.5 Software System Attributes

The author will keep coding standard with proper commenting and documentation.

3. SYSTEM ARCHITECTURE

3.1 Overview

This application uses HTTPS exclusively for security reasons, except in the local developer environment, where we use unencrypted HTTP, as shown in Figure 3.1.

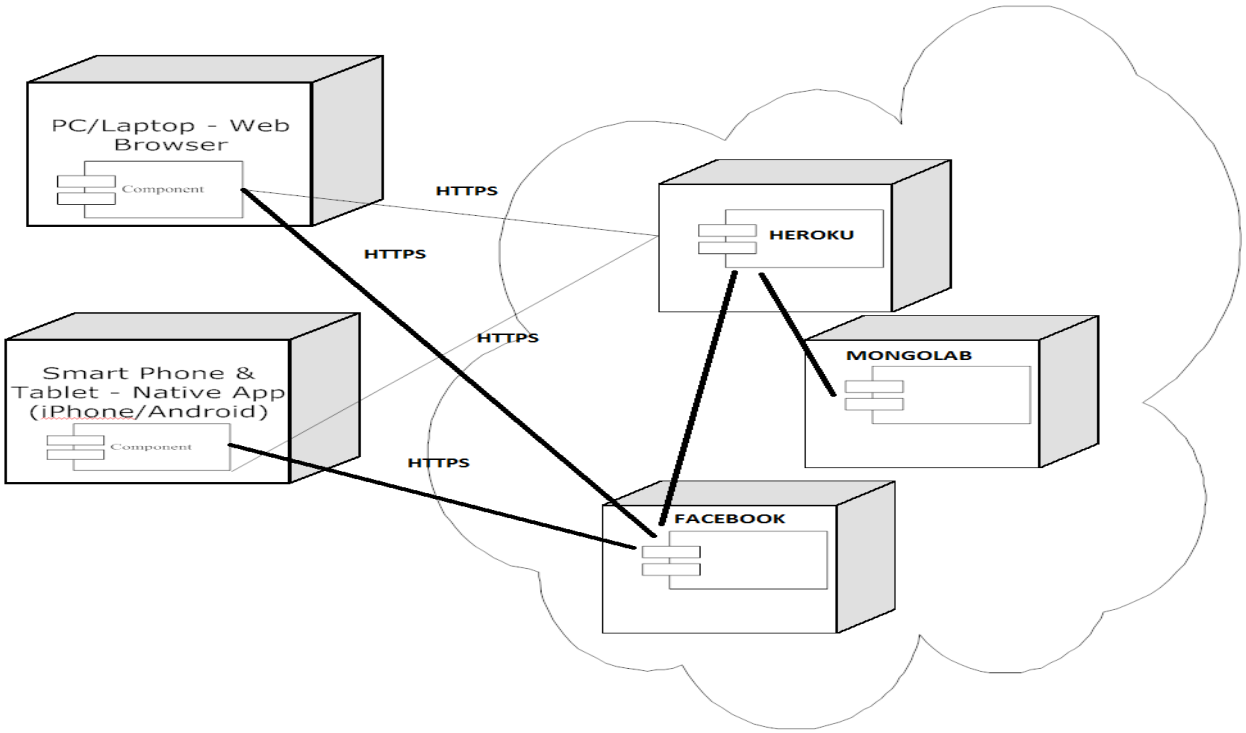

Fig. 3.1: Deployment Diagram

3.2 Deployment Workflow

There are three type of environments used in the deployment workflow: development, staging and production. Figure 3.2 illustrates the deployment workflow.

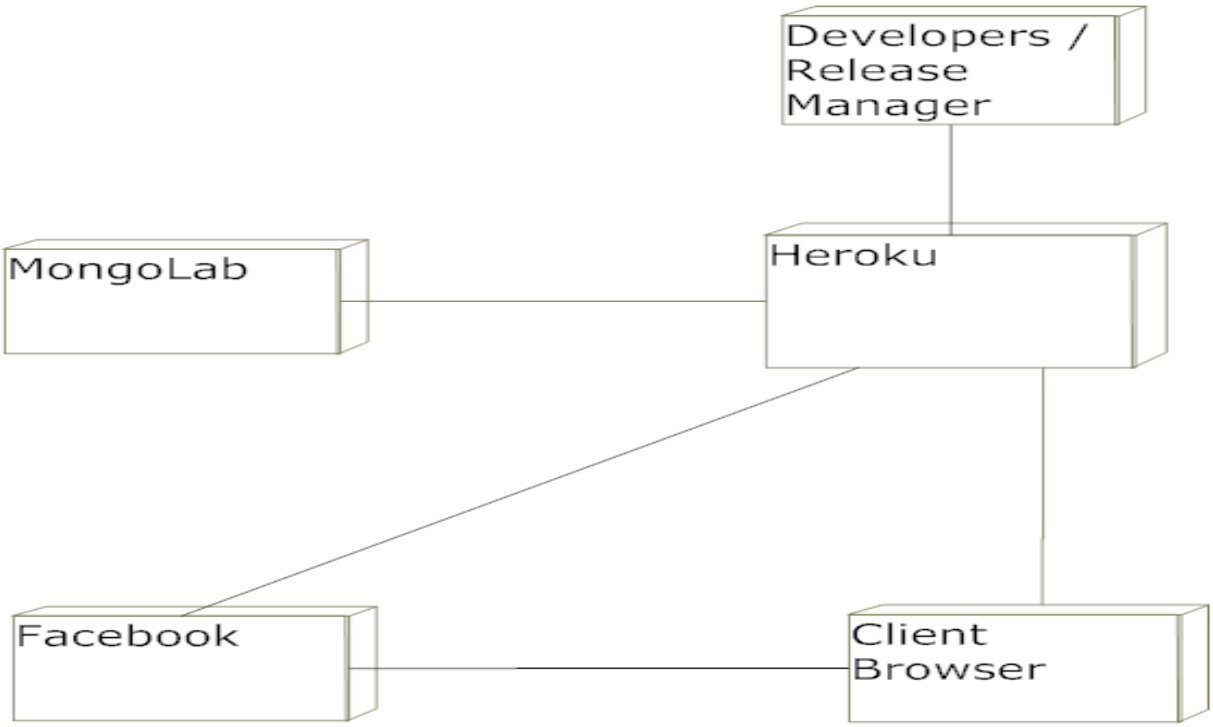

Fig. 3.2: System Integration

- Developers work on new features or bug fixes in development branches. Only minor updates are committed directly to the stable development branch.
- Once features are implemented and/or set of bugs are fixed, they are merged in to staging branch and deployed to staging environment for testing and quality assurance
- After testing is completed, the snapshop of staging branch is kept for production

deployment, otherwise the process will repeat until the testing is completed.

- On the release date, the working staging branch is deployed to production environment.

On this project, git is used as code repositories, to manage developments, staging and production branch. And Heroku toolbelt is also used to set the enviroment config variable for each deployment. Heroku allows users to use git to deploy automatically from local repositories.

3.3 Developer/Release

In this project, developers first do unit testing in local machine, then after it reached certain point, developer himself or assign release manager use git to push the changes to staging or production repositories that connected to respective Heroku servers.

Below are the command the developer/release manager uses to start the application, then commit and push the changes to remote repository.

```
$ foreman start
18:43:16 web.1   : started with pid 5540
18:43:16 web.1   : listening on 5000

$ git status

$ git add .

git commit –m "message here"

$ git push master remote
$ git push staging remote
$ git push production remote
```

3.4 Heroku

In this project there are two sets of heroku instances used: staging and production. The application running in Heroku connects to a database server running inside MongoLab using the MongoDB protocol to read and/or write data to database. Heroku also talks to Facebook servers via Facebook's Open Graph API.

Below are the sample of Heroku command to create staging environment and edit configuration variables

```
$ heroku create --remote staging

$ git config heroku.remote staging
```

3.5 MongoLab

In this project there are three sets of mongo databases used: development, staging and production. It is important to keep the versions of databases since new versions of changes may include changes in database structure, so rolling back or forward the application version would not cause any error.

Below are the command how to connect to remote Mongo Database that hosted in MongoLab

```
$ mongo <servername>.mongolab.com:<portno>/<dbname>
-u <dbuser> -p <dbpassword>
```

3.6 Facebook

Facebook is playing an important role in this project. Facebook provides user authentication and social media integration. Facebook allows connection using the Facebook API and Open Graph API.

3.7 Client Browser

Client browser uses HTTPS GET for static content, and HTTPS POST for AJAX request to Heroku. And client browser also connects to Facebook server directly using Facebook API and Open Graph API in HTTPS.

4. SYSTEM DESIGN

4.1 Design Overview

GradeBadge application uses a client-server architecture model, and it uses MVC (Model View Controller) architecute in the client and server side. And all incoming AJAX requests are submitted using HTTP POST and contain data encoded using JSON.

4.2 Model View Controller Architecture

This application is based on Model View Controller (MVC) Architecture. Model View Controller (MVC) architecture is a software design pattern for separating different components of a software application [33]. There are three main categories in the MVC architecture:

- Model : it represents data in the application. All the business rules are handled in the model.
- View: it represents UI components in the application. The UI components are responsible for presenting the model and for collecting user inputs.
- Controller: it is responsible for updating the data in the model and notifying the view about changes in the model.

4.3 Server-side Architecture Design

All Nodejs modules that start with req_*.js are request handler that get requested from router.js. All ajax requests will go through req_op.js, that verifies the user logged-in to Facebook and app_version is current. Every req_op.js request must contains Facebook access_token and app_version. If the user is not logged-in to Facebook, req_op.js returns the following JSON document, login:true, if the version is not current, then req_op.js return the following JSON document, ver:true.

- .env : This is the setup file that contains environment variables. This file only exist in developer local environment, and these values in this file would be set in each Heroku environment config for staging and production.
- .gitignore: This is the setup file that contains list of files or folders that will be ignored when committing or pushing to git repositories.
- .slugignore: This is the setup file that contains list of files or folders that will be ignored when calculating the slug limit in Heroku
- package.json: This is the setup file that contains of list of dependencies and engine version use in the application. This file also contains application name, version and description.
- Procfile: This is the setup file that tells Heroku how to launch the application
- main.js: This is the main module in node js, which contains the code that verify all neccessary environment variables are set correctly. It also invoke initialization in neccesarry modules to start the application, after the initialization is completed, it starts the HTTP request handling loop.
- router.js: This module routes incoming requests to the right module.
- app_ajax.js: This module contains the application wide AJAX handling routines

- app_http.js: This module contains all of HTTP protocol routines for the application, caching headers, compression header and other HTTP based optimization are implemented in this module.
- fb.js: This module contains all code that interact with Facebook.
- logger.js: This module contains application wide logging functionalities.
- model.js: This module initializes the database connnection pool during server start
- req_app.js: This module handles request for application HTML template for badge earner
- req_counter.js: This module handles request for logging counter
- req_file.js: This module handles request for static content.
- req_issuer.js: This module handles request for application HTML template for badge issuers
- req_mem.js: This module handles request for memory usage
- req_root.js: This module handles request for static content under the root URL
- req_op.js : This module handles all AJAX request from client and routes to appropriate modules.

4.4 Mapping of Model Classes to MongoDB

There will be one node js module to represent the mongoDB collection, named model_(collection_name).js. And many-to-many relationships are represented by linking documents, named (a)_(b)_links

- model_group.js: this node.js module represents Groups collection

- model_badge.js: this node.js module represents Badges colletion
- model_user.js: this node.js module represents Users collection
- model_group_admin.js: this node.js module represents group_admin_links collection
- model_group_member.js: this node.js module represents group_member_links collection
- model_user_badge.js: this node.js module represents user_badge_links collection
- model_group_badge.js: this node.js module represents group_badge_links collection

4.5 Request Handler Operation

There will be one node js module to handle ajax request from client, named op_(request).js. Every request may read, write or update to and from more than one collection

- op_read_badges_by_group.js: this operation is for request for all badges in the given group
- op_read_groups_by_admin.js: this operation is for request for all groups in the given admin
- op_save_badge.js: this operation is for request for saving given badge details
- op_save_group.js: this operation is for request for saving given group details

4.6 Client-side Architecutre Design

- app.html : This is the html template for badge earner page
- issuer.html : This is the html template for badge issuer page

- public_root/channel.html : This is the static content required by Facebook
- public_root/favicon.ico : This is the statuc content for icon use in the browser
- public_ver/app.js : This is the client java-script
- public_ver/style.css: This is the css file use

5. DATABASE DESIGN

5.1 Overview

5.1.1 MongoDB

MongoDB is a scalable, high-performance, open source, NoSQL document-based database. MongoDB features include document-oriented storage, indexes, replication, high availability, auto-sharding, and querying. [21]

5.1.2 MongoLab

MongoLab is the cloud computing provider for MongoDB database that was used in this project [22]. MongoLab has a free tier service that facilitates experimentation by developers, and thus was convenient for this project.

5.1.3 Documents

Data in MongoDB is stored in documents, and every document must have a primary key named _id. Like requlars SQL-based database, documents are like row in a table, where in MongoDB, documents have flexible schema, but every document must have _id field, even if you don't speficy the _id field, mongoDB will add that automatically. [21]

Although document structure is not enforced in MongoDB, different data model and structure may have significant impacts on MongoDB and application performace. So it is good to keep some kind of structure or pattern in data model.

5.1.4 Collections

Documents in MongoDB are organized in collection, and basic database operations are performed based on collection. Indexes can be assigned in any field or subfield contained in documents within a MongoDB collection, and they are defined on per-collection level. [21]

5.2 Data Model

In this project, there are three main collections, User, Group and Badge Colletion. And there are four linkings colletions to represent the many-to-many relationship between the main collections.

- user : this collection contains the user information
- group : this collection contains the group information
- badge : this collection contains the badge information
- group_admin_links : this linking collection contains group_id and user_id, which represent the admin of a group
- group_member_links : this linking collection contains group_id and user_id, which represent the member of a group
- badge_user_links : this linking colletion contains badge_id and user_id, which reperesnt badges that user earned
- group_badge_links : think linking collection contains group_id and badge_id, which represent badges that belong to a group

5.2.1 User Collection

This collection contains of user documents, contain the following information, Table 5.1

Tab. 5.1: User Collection

Document Attribute	Sample Data	Description
_id	34728373	Facebook user ID
last_login	12/1/13 12:34:56 PM	User last login timestamp

5.2.2 Group Collection

This collection contains of group documents, Table 5.2

Tab. 5.2: Group Collection

Document Attribute	Sample Data	Description
_id	"51466591b95b2b5023000001"	Auto-generated Group ID
name	Yoga CSUSB	Group name
desc	This is Yoga group in CSUSB	Group description

5.2.3 Badge Collection

This collection contains of badge documents, Table 5.3

Tab. 5.3: Badge Collection

Document Attribute	Sample Data	Description
_id	"514e34507075ae0c0f000001"	Auto-generated Badge ID
name	Attended 10 classes	Badge name
desc	This is badge description	Badge description
pict	pict1.png	Badge Picture Reference

5.2.4 Group Admin Links Collection

This collection represent many-to-many relationship between group and user. User in this collection means the admin of the group. Table 5.4

Tab. 5.4: Group Admin Links Collection

Document Attribute	Sample Data	Description
_id	"514e34507075ae0c0f000001"	Auto-generated Document ID
gid	"514e34507075ae0c0f000123"	Group ID
uid	"514e34507075ae0c0f000456"	User ID as Admin

5.2.5 Group Member Links Collection

This collection represent many-to-many relationship between group and user. User in this collection means the member of the group. Table 5.5

Tab. 5.5: Group Member Links Collection

Document Attribute	Sample Data	Description
_id	"514e34507075ae0c0f000002"	Auto-generated Document ID
gid	"514e34507075ae0c0f000123"	Group ID
uid	"514e34507075ae0c0f000675"	User ID as Member

5.2.6 Badge User Links Collection

This collection represent many-to-many relationship between badge and user. User in this collection means the user who earned the badge. Table 5.6

Tab. 5.6: Badge User Links Collection

Document Attribute	Sample Data	Description
_id	"514e34507075ae0c0f000005"	Auto-generated Document ID
bid	"514e34507075ae0c0f000975"	Badge ID
uid	"514e34507075ae0c0f000456"	User ID as Earner

5.2.7 Group Badge Links Collection

This collection represent many-to-many relationship between group and badge. Badge in this collection means the badge in the group. Table 5.7

Tab. 5.7: Group Badge Links Collection

Document Attribute	Sample Data	Description
_id	"514e34507075ae0c0f000004"	Auto-generated Document ID
gid	"514e34507075ae0c0f000158"	Group ID
bid	"514e34507075ae0c0f000953"	Badge ID

6. PROJECT IMPLEMENTATION

The GradeBadge application is designed to work on mobile, tablet devices and desktop computers. The UI of the application is developed using Bootstrap. When a page requires the data to be loaded from server or modified or deleted, a request is sent to the Web server over HTTPS. The requests are sent to the Web server using Ajax. For handling Ajax requests and responses, this application uses JQuery Ajax API. All UI components are dynamically created or initialized in response to the data received from the Web server.

6.1 Loading Screen

When the GradeBoard application is loaded, a loading screen is presented to the user as shown in the Figure 6.1. The loading screen shows application logo and loading progess bar, and the screen is automatically redirected after the loading completed.

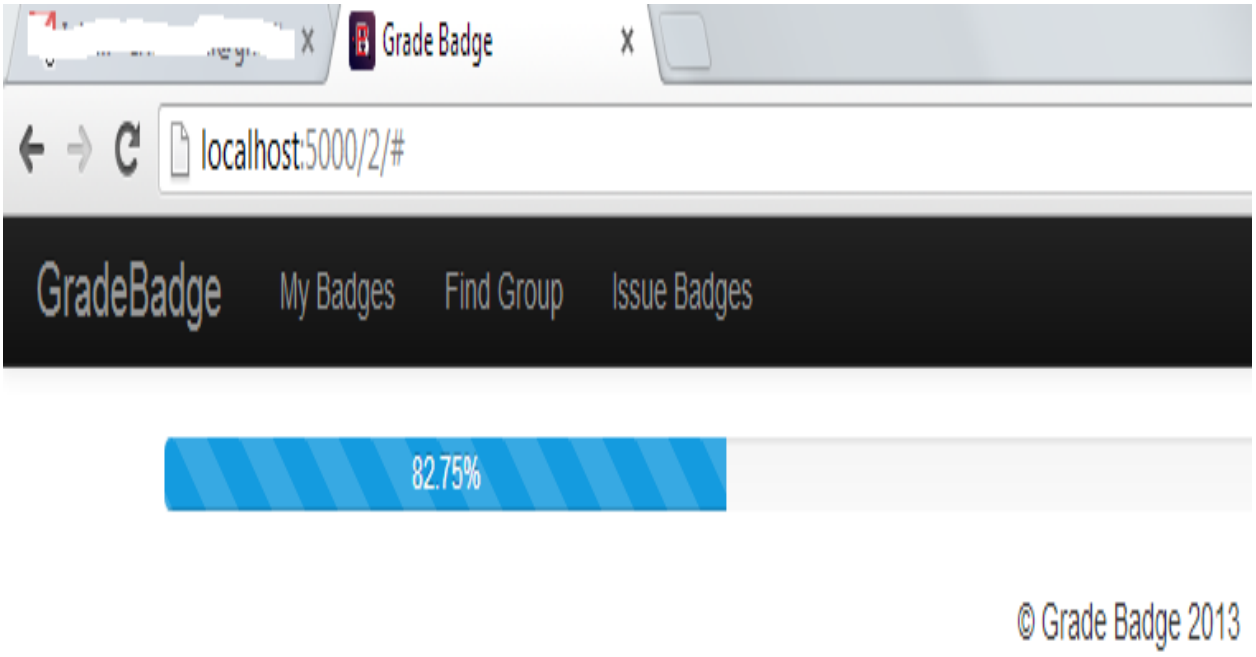

Fig. 6.1: GradeBadge Loading Screen

6.2 Login Screen

GradeBadge uses Facebook account for users to login. When the user is not logged-in to Facebook, the screen is automatically redirected to the Facebook login screen as shown in Figure 6.2. Every user in the system can be badge issuer and badge earner.

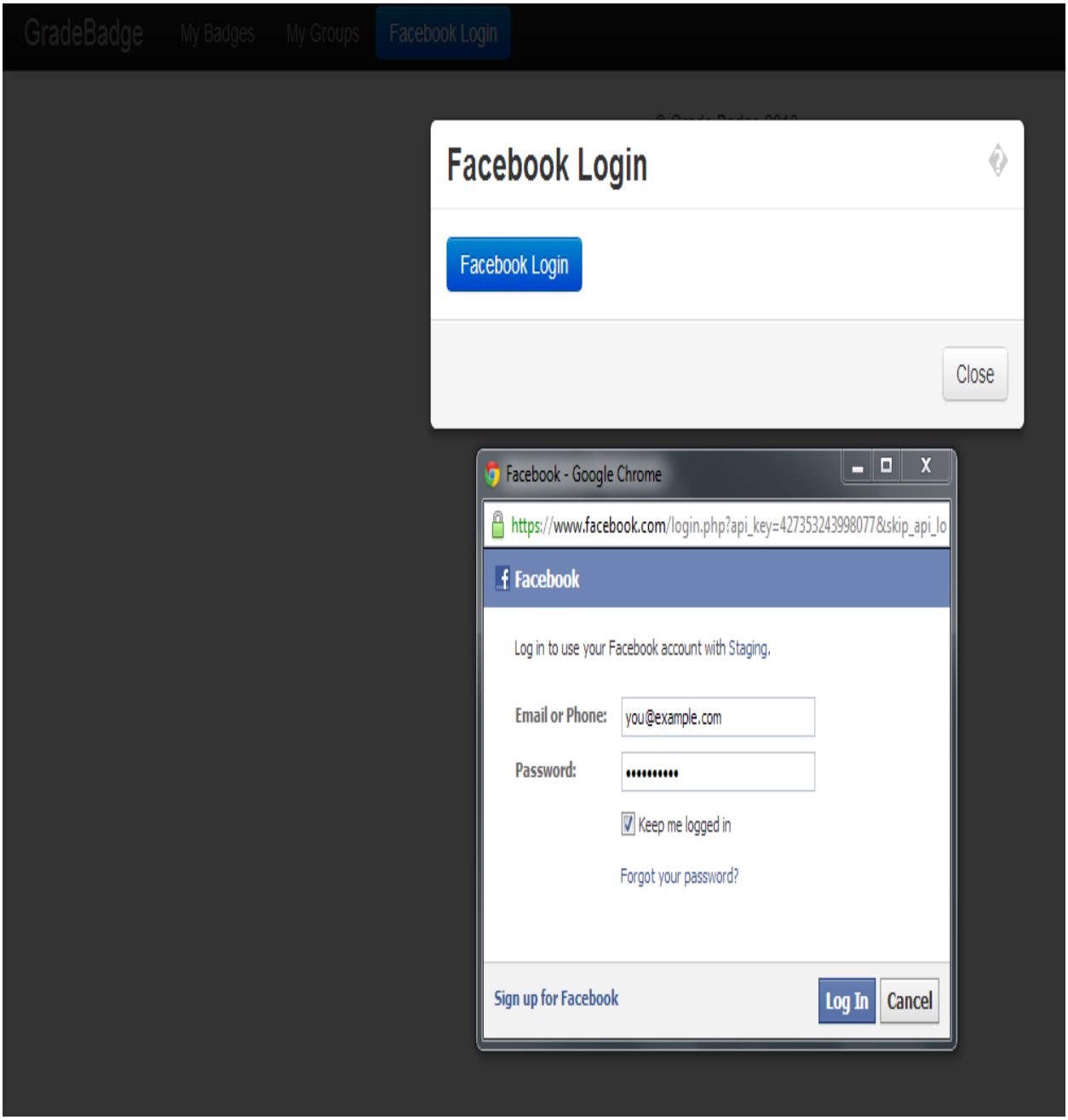

Fig. 6.2: GradeBadge Login Screen

as shown in Figure 6.3.

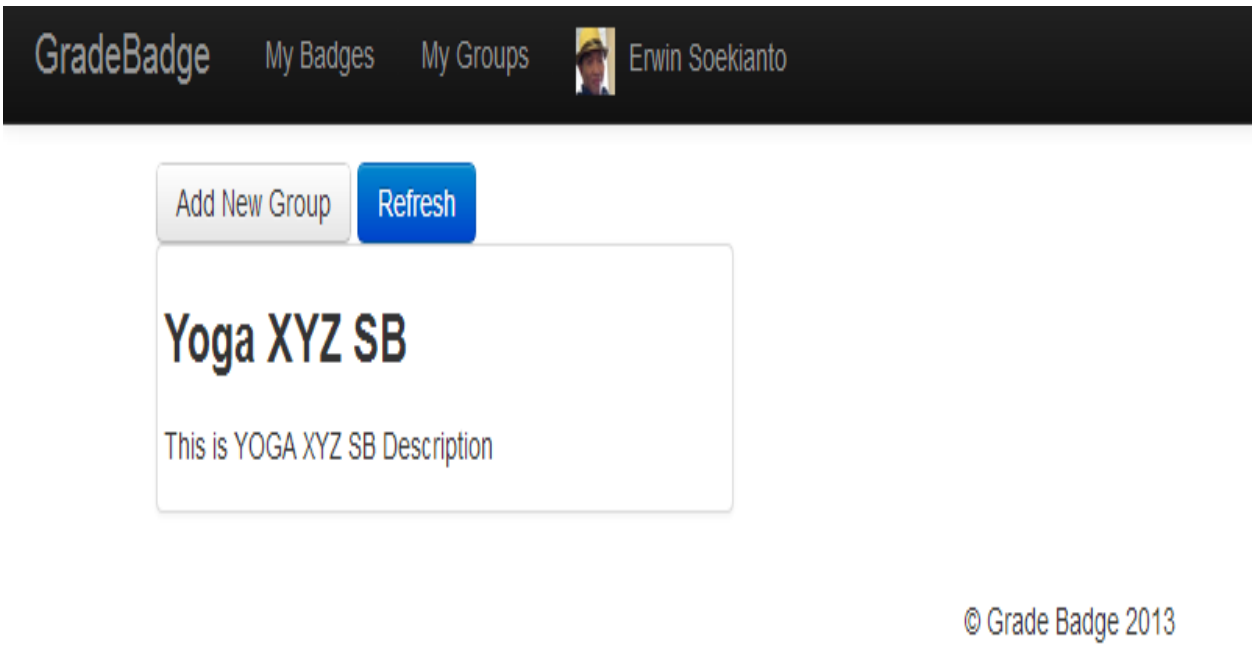

Fig. 6.3: GradeBadge Group Page

as shown in Figure 6.4.

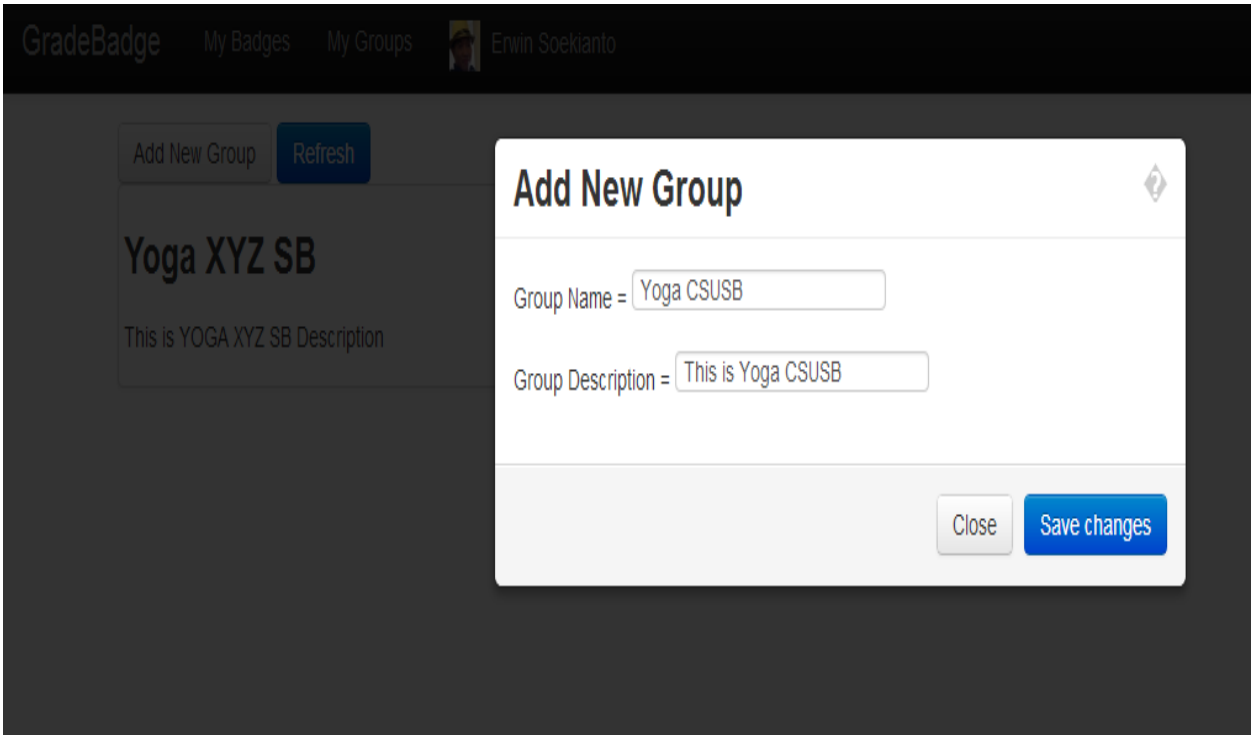

Fig. 6.4: GradeBadge Add New Group Page

as shown in Figure 6.5.

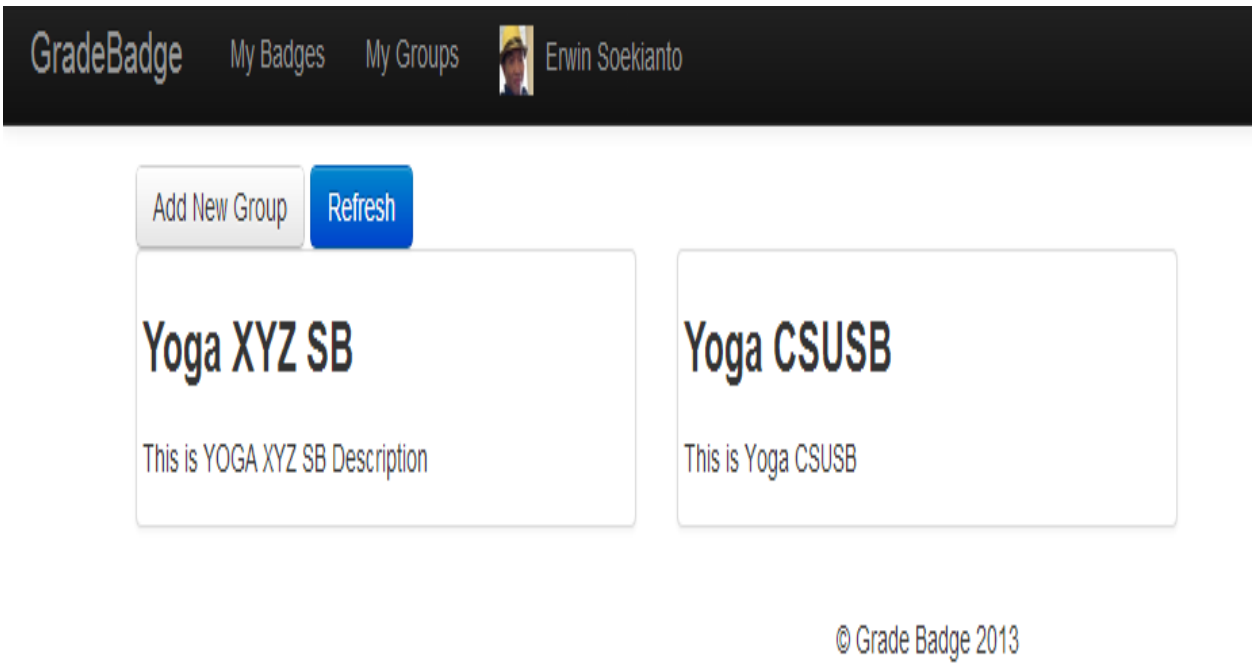

Fig. 6.5: GradeBadge Group Page Added

7. CONCLUSION AND FUTURE DIRECTION

7.1 Conclusion

GradeBadge is cloud-based web application that is hosted in Heroku, uses Nodejs and MongoDB, that provide responsive design that works well to be run inside browsers in desktop computers, tablets and smart phones. GradeBadge also enables users to login with their Facebook account to simplify authentication, and integrated with their Facebook.

Cloud computing provides significant cost-saving to developers to build application that can be scaled up or down almost instantly to accomodate rapidly changing demand.

7.2 Future Direction

GradeBadge application can be used as sample to showcase the development of cloud-based cross-platform application. This application can also be extended and enhanced in the future, as described in the following

- Auto-sharding: Implementation of auto-sharding in MongoDB to support greater scalability.
- Other social networking: Implementation of authentication and integration with other social networking applications such as Twitter, Tumblr, Google+, etc.
- Native App: Development of native IOS , Android and Windows Mobile Appli-

cation.

- API: Development of API to enable other applications to integrate a reward system module to their applications.

APPENDIX A

SOURCE CODE

```
//main.js
```

REFERENCES

[1] Ajax (programming). http://en.wikipedia.org/wiki/Ajax_(programming).

[2] Amazon Web Services. http://aws.amazon.com/.

[3] Android Mobile Computing Platform. http://developer.android.com/about/index.html.

[4] Bootstrap. http://twitter.github.com/bootstrap/.

[5] Dom Document Object Model. http://www.w3.org/DOM.

[6] Eclipse Fouccation. http://www.eclipse.org/.

[7] Entities, Properties, and Keys. https://developers.google.com/appengine/docs/java/datastore/entities.

[8] Facebook Developers. https://developers.facebook.com.

[9] Github. https://github.com/.

[10] Glossary. http://technet.microsoft.com/en-us/library/bb742416.aspx.

[11] Google Developers Academy. https://developers.google.com/appengine/training/intro/whatisgae/.

[12] Heroku How It Works. http://www.heroku.com/how.

[13] How Entities and Indexes are Stored. https://developers.google.com/appengine/articles/storage_breakdown.

[14] Http secure. http://en.wikipedia.org/wiki/HTTPS.

[15] Index Definition and Structure. https://developers.google.com/appengine/docs/python/datastore/indexes.

[16] IOS Mobile Computing Platform. http://en.wikipedia.org/wiki/IOS.

[17] Jetty (web server). http://en.wikipedia.org/wiki/Jetty_(web_server).

[18] Jquery Mobile 1.2 Reference Document. http://www.jquerymobile.com/demos/1.2.0/docs/about/features.html.

[19] Jquery Official Document. http://www.jquery.com/.

[20] Json. http://en.wikipedia.org/wiki/Json.

[21] Mongodb Agile and Scalable. http://www.mongodb.org/.

[22] Mongolab. http://www.mongodb.org/.

[23] Node.js Manual and Documentation. http://nodejs.org/api/.

[24] Nosql. http://en.wikipedia.org/wiki/NoSQL/.

[25] Official Documentation of Google App Engine Java Datastore. https://developers.google.com/appengine/docs/java/datastore/.

[26] Official Java Technology Document. http://www.java.com/en/download/faq/whatis_java.xml.

[27] PolyModel. https://developers.google.com/appengine/docs/python/ndb/polymodelclass.

[28] Unified Modeling Language. http://www-01.ibm.com/software/rational/uml/.

[29] W3C community. http://www.w3.org/TR/REC-html40/.

[30] What is an Object? http://docs.oracle.com/javase/tutorial/java/concepts/object.html.

[31] Windows Azure. http://www.windowsazure.com/en-us/develop/overview/.

[32] Manoj Kulkarni. *GradeBoard: A Cloud-Based Solution for a Student Grading System.* CSUSB - Masters Project, 2013.

[33] Ph.D Steve Burbeck. Mvc How to use Model-View-Controller. http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html.