**Detailed Design Document**

**Badger Social Media Application**

**Prepared by: Spencer Bialt, Kevin DeMoura, Zackary Flake, Danielle Lopez, Zachary Mason**

**Class: Software Engineering - Fall 2016**

**Instructor: Dr. Baliga**

**1. Introduction**

This is the Detailed Design Document for the Badger Social Media Application. Badger will allow users to add friends, award badges to them, and create groups. This system will be developed for the Android platform in Java with an Object-Oriented design. The application will be hosted on Amazon Web Services, using a MYSQL database. Ruby will be used to create the endpoints. Badger will include these 2 high level modules:

* User Interface- includes modules for LoginPage, HomePage, FriendList, FriendPage, BadgeDetail, BadgeList, GroupPage, GroupCreate, and GroupList
* Database- includes modules for User, Badge, and Group

HIGH LEVEL ARCHITECTURE



**2. Module Design**

**2.1 User Interface Module**

**Purpose**

The purpose of this module is to provide the components needed for user interaction with the application. This module contains 9 submodules- LoginPage, HomePage, FriendList, FriendPage,, BadgeDetail, BadgeList, GroupPage, GroupCreate, and GroupList.

**Rationale**

This module was created to hold all of the components needed for the user to navigate and use the application.

**High Level Module Design**

This module consists of the LoginPage, HomePage, FriendList, FriendPage, BadgeDetail, BadgeList, GroupPage, GroupCreate, and GroupList.

UI MODULE



**Provided Interface**

*From the Provided interface of LoginPage:*

**void createUser(String name, String userID, String email)**

Definition: creates a new User object in the database.

Parameters: the name, userID, and email address of the new user

*From the Provided interface of HomePage:*

**void setTrophyCase(List<Badge> trophyCase)**

Definition: allows the user to select a list of badges for their trophy case

Parameters: the badges selected in their trophy case

*From the Provided interface of FriendList:*

**void addFriend(String email)**

Definition: add another user to the friends list via email

Parameters: the email of the friend to be added

*From the Provided interface of GroupCreate:*

**void createGroup(String name)**

Definition: Create a group object in the database

Parameters: the name of the new group, the user ids of the group members

**Required Interface**

*From Provided Interface of Database Module:*

**User getUser(String userID)**

Definition: Getter for the User.

Parameters: the userID for the requested user

Returns: the user that is requested.

**Badge getBadge(String badgeID)**

Definition: Getter for the Badge.

Parameters: the badgeID for the requested badge

Returns: the badge that is requested.

**Group getGroup(String groupID)**

Definition: returns the group ID

Parameters: the requested group’s ID

Returns: the requested group

**List<Badge> getTrophyCase(String userID)**

Definition: returns the badges selected in the user’s trophy case

Parameters: the current user’s ID

Returns: a list of the current user’s trophy case badges

**List<Badge> getSentBadges(String userID)**

Definition: returns the badges sent by the designated user

Parameters: a user ID, either the logged-in user or a friend

Returns: a list of the user’s sent badges

**List<Badge> getReceivedBadges(String userID)**

Definition: returns the badges received by the designated user

Parameters: a user ID, either the logged-in user or a friend

Returns: a list of the user’s received badges

**2.1.1 LoginPage Module**

**Purpose**

The purpose of this module is to allow the user to login to the application or create a new account using their email address.

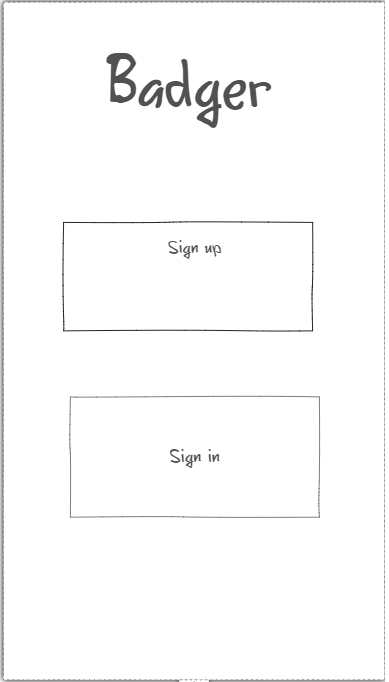
**Rationale**

This module is needed so the user can gain secure access to the application. Logins will be done using emails only for now so that other applications’ APIs can be implemented easily later.

**Provided Interface**

**void createUser(String name, String userID, String email)**

Definition: creates a new User object in the database.

Parameters: the name, userID, and email address of the new user.

**Required Interface**

**User getUser()**

Definition: get User information from the Database

Parameters: None

Returns: the user that has signed in.

**2.1.2 HomePage Module**

**Purpose**

The purpose of this module is to allow the user to view their created and received badges, and choose a set of particular badges to display in a “trophy case”.

**Rationale**

This module is likely where the user will spend their most time. From here, they can manage their collection, and navigate to elsewhere in the application.

**Provided Interface**

**void setTrophyCase(List<Badge> trophyCase)**

Definition: allows the user to select a list of badges for their trophy case

Parameters: the badges selected in their trophy case

**Required Interface**

*From the Provided Interface of the Database Module:*

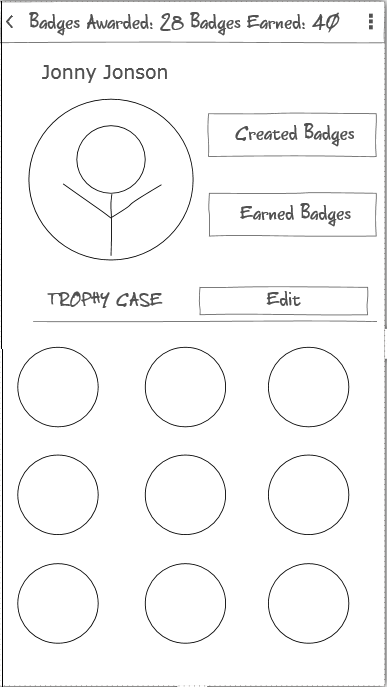
**List<Badge> getTrophyCase(String userID)**

Definition: returns the badges selected in the user’s

trophy case

Parameters: the current user’s ID

Returns: a list of the current user’s trophy case badges

**List<Badge> getSentBadges(String userID)**

Definition: returns the badges sent by the designated user

Parameters: a user ID, either the logged-in user or a friend

Returns: a list of the user’s sent badges

**List<Badge> getReceivedBadges(String userID)**

Definition: returns the badges received by the designated user

Parameters: a user ID, either the logged-in user or a friend

Returns: a list of the user’s received badges

**Badge getBadge(String badgeID)**

Definition: returns the requested badge

Parameters: the badge ID

Returns: the requested badge

**2.1.3 FriendList Module**

**Purpose**

The purpose of this module is to allow the user to interact with their list of friends.

Stretch goals for the friend list allow the user to designate a list of “close friends”, and also to display the friend’s “top badge” in their trophy case.

**Rationale**

This module is needed so the user can navigate to the profiles of their friends.

**Provided Interface**

**void addFriend(String email)**

Definition: add another user to the friends list via email

Parameters: the email of the friend to be added

**Required Interface**

*From the Provided Interface of the Database Module:*

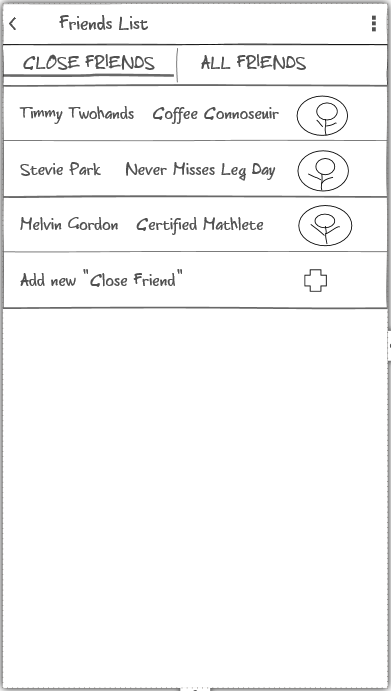
**List<User> getFriends(userID)**

Definition: returns a list of friends of the current user

Parameters: the current user’s ID

Returns: the current user’s friends

**User getFriend(userID)**

Definition: returns the user object of a selected friend

Parameters: the ID of the friend selected on the friends list

Returns: the requested user object

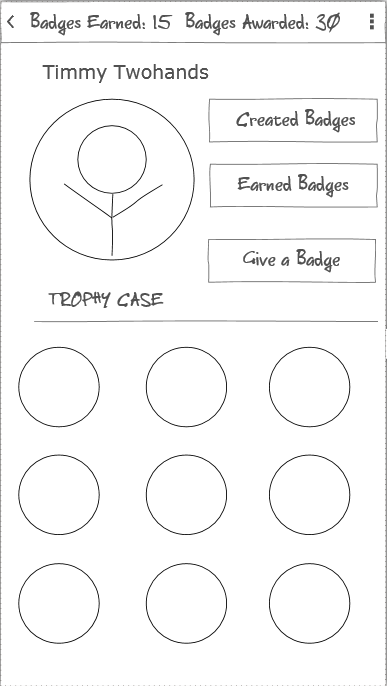
**2.1.5 FriendPage Module**

**Purpose**

The purpose of this module is to allow the user to interact with the page of a friend, viewing their trophy case, created badges and earned badges.

**Rationale**

This module is needed so the user can navigate through the the page of a friend.



**Provided Interface**

This module does not have a provided interface.

**Required Interface**

*From the Provided Interface of the Database*

*Module:*

**List<Badge> getTrophyCase(String userID)**

Definition: returns the badges selected in the user’s

trophy case

Parameters: the current user’s ID

Returns: a list of the current user’s trophy case badges

**List<Badge> getSentBadges(String userID)**

Definition: returns the badges sent by the designated

user

Parameters: a user ID, either the logged-in user or a friend

Returns: a list of the user’s sent badges

**List<Badge> getReceivedBadges(String userID)**

Definition: returns the badges received by the designated user

Parameters: a user ID, either the logged-in user or a friend

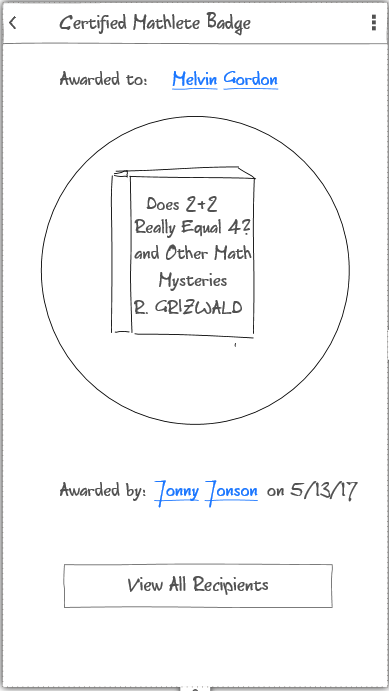
Returns: a list of the user’s received badges

**Badge getBadge(String badgeID)**

Definition: returns the requested badge

Parameters: the badge ID

Returns: the requested badge

**2.1.6 BadgeDetail Module**

**Purpose**

The purpose of this module is to allow the user to view a large and close up look at a badge. A stretch goal includes the ability to view all recipients of the badge.

**Rationale**

This module is needed to show a larger view of a chosen badge, as well as other details about the badge that would not fit on other pages.

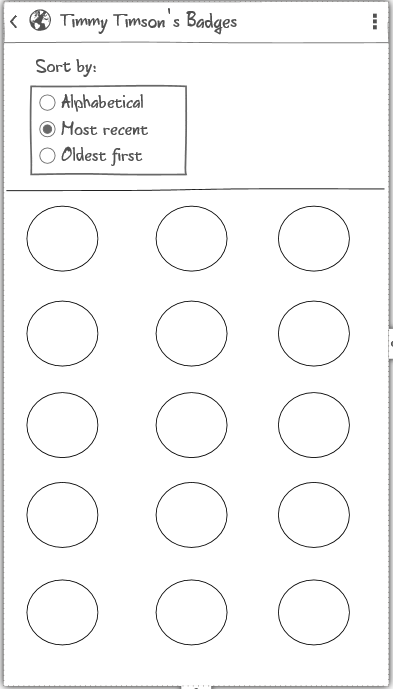
**Provided Interface**

This module does not have a Provided Interface.

**Required Interface**

This module does not have a Required Interface.

**2.1.7 BadgeList Module**

**Purpose**

The purpose of this module is to allow the user to view a list of all of the badges they have been awarded.

Clicking on a badge will bring up the BadgeDetail view.

A stretch goal would be for the badges to be sortable.

**Rationale**

This module is needed to show a view of a user’s badges, either the active user or one of their friends.

**Provided Interface**

This module does not have a provided interface.

**Required Interface**

*From the Provided Interface of the Database Module:*

**Badge getBadge(String badgeID)**

Definition: returns the requested badge

Parameters: the badge ID

Returns: the requested badge

**2.1.8 GroupList Module**

**Purpose**

The purpose of this module is to allow the user to view a list of all of the groups they are members of.

**Rationale**

This module is needed to show a view of the user’s groups.

**Provided Interface**

This module does not have a provided interface.

**Required Interface**

*From the Provided Interface of the Database Module:*

**List<Group> groupList(userID)**

Definition: returns the list of groups

Parameters: the current user’s ID

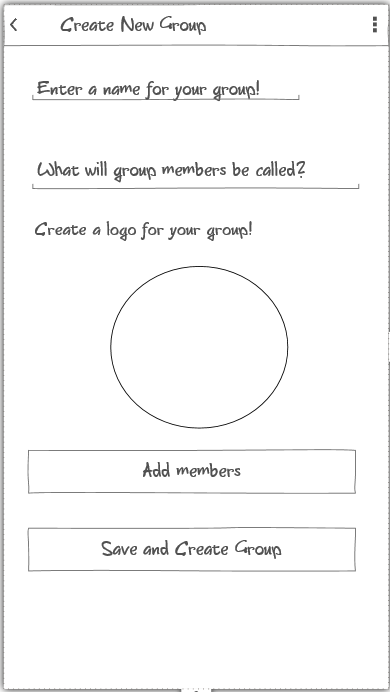
Returns: the current user’s list of groups



**2.1.9 GroupCreate Module**

**Purpose**

The purpose of this module is to allow the user to create a new group.



**Rationale**

This module is needed so the user can create a group.

**Provided Interface**

**Group createGroup(String userID, List<String> members, String groupName, String membersName)**

Definition: Create a new Group in the database.

Parameters: the current user’s ID, a List of selected users to join the group, the name of the group, and what members of the group will be called

Returns: the newly created group

**Required Interface**

This module has no required interface.

**2.1.10 GroupPage Module**

**Purpose**

The purpose of this module is to allow the user view the homepage of a specified group.

**Rationale**

This module is needed so the user can view a group.

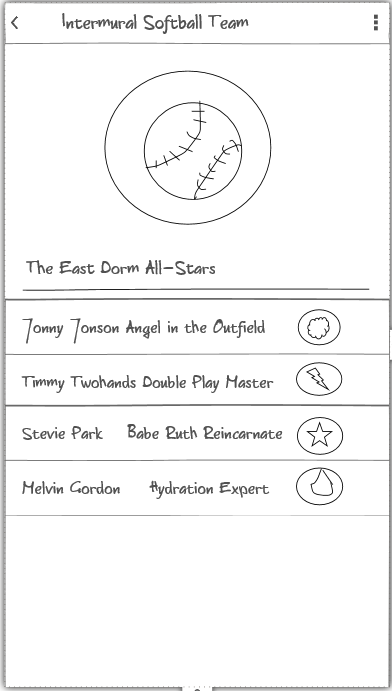
**Provided Interface**

This module has no provided interface.

**Required Interface**

*From the Provided Interface of the Database Module:*

**Group getGroup(groupID)**

Definition: returns the group ID

Parameters: the requested group’s ID

Returns: the requested group

**3.1 Database Module**

**Purpose**

The purpose of this module is to provide the components needed for user the database. This module contains 3 submodules- User, Badge, and Group.

**Rationale**

This module was created to hold all of the components needed to store all data needed for the application.

**High Level Module Design**

This module consists of the User, Badge, and Group modules.

DATABASE MODULE



**Provided Interface**

**User getUser(String userID)**

Definition: Getter for the User.

Parameters: the userID for the requested user

Returns: the user that is requested.

**Badge getBadge(String badgeID)**

Definition: Getter for the Badge.

Parameters: the badgeID for the requested badge

Returns: the badge that is requested.

**Group getGroup()**

Definition: get Group information from the Database

**Provided Interface**

*From the Provided interface of LoginPage in the UI Module:*

**Void createUser(String name, String userID)**

Definition: Creates a User in the database.

Parameters: the name and userID

*From the Provided interface of GroupCreation in the UI module:*

**void createGroup(String name, String Description)**

Definition: Create a group in the database

Parameters: the name of the new group

**3.1.1 User Module**

**Purpose**

The purpose of this module is to store all data needed for a user.

**Rationale**

This module is needed so the application can store user data.

**Provided Interface**

**User getUser(String userID)**

Definition: Getter for the User.

Parameters: the userID for the requested user

Returns: the user that is requested.

**Required Interface**

**void createUser(String name, String userID, String email)**

Definition: Creates a User in the database.

Parameters: the name, userID, and email of the new user

**3.1.2 Badge Module**

**Purpose**

The purpose of this module is to store all data needed for a badge.

**Rationale**

This module is needed so the application can store badge data.

**Provided Interface**

**Badge getBadge(String badgeID)**

Definition: Getter for the Badge.

Parameters: the badgeID for the requested badge

Returns: the badge that is requested.

**Required Interface**

This module does not have a required interface

**3.1.3 Group Module**

**Purpose**

The purpose of this module is to store all data needed for a group.

**Rationale**

This module is needed so the application can store group data.

**Provided Interface**

**Group getGroup(groupID)**

Definition: returns the group ID

Parameters: the requested group’s ID

Returns: the requested group

**Required Interface**

**void addGroup(String name, String description)**

Definition: adds a Group to the database.

Parameters: the name and description of the new group

**3.2 Database Schema**

users

String userID

String username

String email

String password

groups

String groupID

String groupName

String membersName

Array[String] members

userGroups

userID

groupID

userFriends

friendID

trophyCase

userID

badgeID

userSent

userID

badgeID

userReceived

userID

badgeID

badge

badgeID

senderID

imageURL

description

**3.3 RESTful API Specification**

User Functions

**/createUser**

POST http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//createUser

Returns newly created user with id

Sample Request Body: { "username": "myUser", “password” : “fjkD3@@$D!” }

Sample Output: { "id": 16, "username": "myUser" }

### **/readUser**

GET http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//readUser?id=2

Returns user with id 2 in JSON

Sample Output: { "id": 2, "username": "exampleUser1" }

### **/updateUser**

POST http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//updateUser

Updates user of given id with provided parameters

Sample Request Body: {“id”:”5”, "username": "myNewUsername" }

Sample Output: { "id": 16, "username": "myNewUsername" }

### **/deleteUser**

DELETE http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//deleteUser

Deletes user with id

Sample Request Body: { "id": "446" }

Sample Output: { "response": "User deleted." }

Badge Functions

### **/createBadge**

POST http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//createBadge

Returns newly created badge with id

Sample Request Body: { "badge\_name": "My Badge" }

Sample Output: { "badge\_id": 16, “description”: “My Badge” }

### **/readBadge**

GET

Sample input:

http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//readBadge?id=2

Returns badge object in JSON

Sample Output: { "badge\_id": 2, "description": "exampleBadgeName", “badge\_URL” : “my.badge.com/badgeFileName.png” }

### **/updateBadge**

POST http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//updateBadge

Updates badge with id with given parameters

Sample Request Body: {“badge\_id”:”16”, "description": "myNewBadgename" }

Sample Output: { "badge\_id": 16, "description": "myNewBadgename" }

### 

### 

### **/deleteBadge**

DELETE http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//deleteBadge

Deletes the badge with given id

Sample Request Body: { "badge\_id": "446" }

Sample Output: { "response": "Badge deleted." }

Group Functions

**/createGroup**

POST http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//createGroup

Returns newly created group with id

Sample Request Body: { "group\_name": "My Group" }

Sample Output: { "group\_id": 16, “group\_name”: “My Group” }

### **/readGroup**

GET

Sample input:

http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//readGroup?id=2

Returns group object in JSON

Sample Output: { "group\_id": 2, "group\_name": "exampleGroupName" }

### **/updateGroup**

POST http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//updateGroup

Updates group with id with given parameters

Sample Request Body: {“group\_id”:”16”, "group\_name": "myNewGroupname" }

Sample Output: { "group\_id": 16, "group\_name": "myNewGroupname" }

### **/deleteGroup**

DELETE http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//deleteGroup

Deletes the group with given id

Sample Request Body: { "group\_id": "446" }

Sample Output: { "response": "Group deleted." }

Group Functions

**/createGroup**

POST http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//createGroup

Returns newly created group with id

Sample Request Body: { "group\_name": "My Group" }

Sample Output: { "group\_id": 16, “group\_name”: “My Group” }

### **/readGroup**

GET

Sample input:

http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//readGroup?id=2

Returns group object in JSON

Sample Output: { "group\_id": 2, "group\_name": "exampleGroupName" }

### **/updateGroup**

POST http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//updateGroup

Updates group with id with given parameters

Sample Request Body: {“group\_id”:”16”, "group\_name": "myNewGroupname" }

Sample Output: { "group\_id": 16, "group\_name": "myNewGroupname" }

### **/deleteGroup**

DELETE http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//deleteGroup

Deletes the group with given id

Sample Request Body: { "group\_id": "446" }

Sample Output: { "response": "Group deleted." }

userSent Functions

**/addSentBadge**

POST http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//addSentBadge

Will link a particular badge\_ID to a user\_ID to record the badge sender

Sample Request Body: { "badge\_ID": "My Group" }

Sample Output: { "group\_id": 16, “group\_name”: “My Group” }

### **/readSentBadges**

GET

Sample input:

http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//readSentBadges?id=2

Returns the badge\_ids of badges sent by the user

Sample Output: { "user\_id": 2, badge\_id": "20" }

userReceived Functions

**/addReceievedBadge**

POST http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//addReceivedBadge

Will link a particular badge\_ID to a user\_ID to record the badge recipient

Sample Request Body: { "badge\_ID": "20" }

Sample Output: { "user\_id": 2, “badige\_ID”: “20” }

### **/readReceievedBadges**

GET

Sample input:

http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//readReceivedBadges?id=2

Returns the badge\_ids of badges received by the user

Sample Output: { "user\_id": 2, badge\_id": "20" }

trophyCase Functions

**/addTrophyBadge**

POST http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//addTrophyBadge

Will link a particular badge\_ID to a user\_ID to be placed in the user’s trophy case

Sample Request Body: { "badge\_ID": "20" }

Sample Output: { "user\_id": 2, “badige\_ID”: “20” }

### 

### 

### 

### **/readTrophyBadges**

GET

Sample input:

http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//readTrophyBadges?id=2

Returns all badge\_ids attached to a user\_id’s trophy case

Sample Output: { "user\_id": 2, badge\_id": "20" }

**API Spike**

Prototypes of two API endpoints were implemented as a spike for this project. The idea was to that we could successfully set up a web server that would provide resources from our database. The result is two functions that are live and actually manipulate the database that we are using:

### **/createUser**

POST http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//createUser

Returns newly created user with id

Sample Request Body: { "username": "myUser", “password” : “fjkD3@@$D!” }

Sample Output: { "id": 16, "username": "myUser" }

### **/readUser**

GET http://badgerapi.e3rxnzanmm.us-west-2.elasticbeanstalk.com//readUser?id=2

Returns user with id 2 in JSON

Sample Output: { "id": 2, "username": "exampleUser1" }

**4. COMPRISES Diagram**





**4. USES Diagram**



**5. Testing Plan**

**Test Cases:**

**createUser(String name, String userID, String email)**

**Description:** Tests ability to create a new User object in the database.

**Data Input:** the name, userID, and email address of the new user

**Expected Output:** None

**void setTrophyCase(List<Badge> trophyCase)**

**Description:** Tests ability to allow the user to select a list of badges for their trophy case.

**Data Input:** the badges selected in their trophy case

**Expected Output:** None

**void addFriend(String email)**

**Description:** Ability to add another user to the friends list via email.

**Data Input:** the email of the friend to be added.

**Expected Output:** None.

**void createGroup(String name)**

**Description:** Create a group object in the database

**Data Input:**  the name of the new group, the user ids of the group members

**Expected Output:** None

**User getUser(String userID)**

**Description:** Getter for the User.

**Data Input:**  the userID for the requested user

**Expected Output:** the method should show proper retrieval of the user that is requested.

**Badge getBadge(String badgeID)**

**Description:** Ability to get the Badge.

**Data Input:**  the badgeID for the requested badge

**Expected Output:** Returns the badge that is requested.

**Group getGroup(String groupID)**

**Description:** Ability to return the group ID

**Data Input:**  the requested group’s ID

**Expected Output:** Returns the requested group

**List<Badge> getTrophyCase(String userID)**

**Description:** Tests ability to return the badges selected in the user’s trophy case

**Data Input:**  the current user’s ID

**Expected Output:** Proper retrieval of a list of the current user’s trophy case badges

**List<Badge> getSentBadges(String userID)**

**Description:** Ability to return the badges sent by the designated user

**Data Input:**  a user ID, either the logged-in user or a friend

**Expected Output:**  Returns a list of the user’s sent badges

**List<Badge> getReceivedBadges(String userID)**

**Description:** Tests ability to return the badges received by the designated user

**Data Input:**  a user ID, either the logged-in user or a friend

**Expected Output:** Returnsa list of the user’s received badges

**List<User> getFriends(userID)**

**Description:** Ability to return the user object of a selected friend

**Data Input:** the ID of the friend selected on the friends list

**Expected Output:** the requested user object

**List<Group> groupList(userID)**

**Description:** Ability to return the list of groups

**Data Input:** the current user’s ID

**Expected Output:** the current user’s list of groups.

**Group createGroup(String userID, List<String> members, String groupName, String membersName)**

**Description:** Tests the ability to create a new Group in the database.

**Data Input:** the current user’s ID, a List of selected users to join the group, the name of the group, and what members of the group will be called

**Expected Output:** the newly created group