NovaConnect Design Specification

CS189A
March 4, 2013
Chandra Krintz
Geoffrey Douglas

Members
Andrew Lee (leader)
Tim Chanpuelksa (scribe)
Ernesto Cojulun
Justin Phang
Brandon Newman

Table of Contents

Introduction

1.1 Project Overview

One of the most important functions of a business/technology conference is networking and the ability to generate and keep track of “leads”. Leads are contact information that may result in a business deal or other future opportunities. Currently, there exists no easy, efficient, or cost-effective way for conference attendees to keep track of conference leads. NovaConnect from Novacoast x Garbage Collectors is a mobile web app aiming to improve the current state of lead-tracking. With an easy to use interface, users will be able to track conference generated lead in an organized manner without much effort. With this new ability from NovaConnect, many attendees to these conferences will find that conference attendance will not only be much easier, but also a lot more productive.

2. Components

NovaConnect consisted of three major components: the web interface, the mobile interface, and the backend web service database.

Component Name	Description
2.1. Web Interface	The front end of the application that lets people set up conferences and admins and users.
2.2 Mobile Interface	The mobile version of the front end of the application. Also has the options for a QR code scanner, manual entry, or via a picture of their business card to retrieve leads.
2.3 Backend Web Services Database	The backend exposes an API to handle HTTP requests from both the frontend web application, and mobile application. It is RESTful and is connected to a MySQL database.

2.1. Web Interface

Requirement	Description
Functional for conference users and administrators	Administrators are capable of creating and maintaining event information, users are able to

	access limited portions of the service
Connect to back end	This requires to be able to connect to the database in order to handle information and retrieve stored data
Connect to mobile interface	This allows necessary information to be exchanged and displayed to the user regardless of whether they're using a browser or an iOS device
Interactive UI	Smooth, intuitive, and easy to use for the target audience

For our front end web interface, we are building app using html5, embedded ruby blocks, and bootstrap. With html5, we can build basic webpages with various functionalities. By embedding ruby code in blocks with proper tags. ERB (Embedded Ruby Blocks) allows us to evaluate instance variables declared in the associated controllers, display flash messages, and dynamically control the structure of the page with a few conditional statements and iterations/loops. Bootstrap is a front-end framework and web design kit for building modern websites. Bootstrap is a default mobile-first approach that helps us in developing the mobile web app. It has compatibility with various browsers, ease of access, CSS grid system, and many other utilities.

2.2 Mobile Interface

Requirement	Description
Integration with phone hardware	This allows us to take advantage of the hardware for exclusive features such as the camera for taking pictures,QR.
Connect to web interface	This allows necessary information to be exchanged and displayed to the user regardless of whether they're using a browser or an iOS device.
Interactive UI	Smooth, intuitive, and easy to use for the target audience.

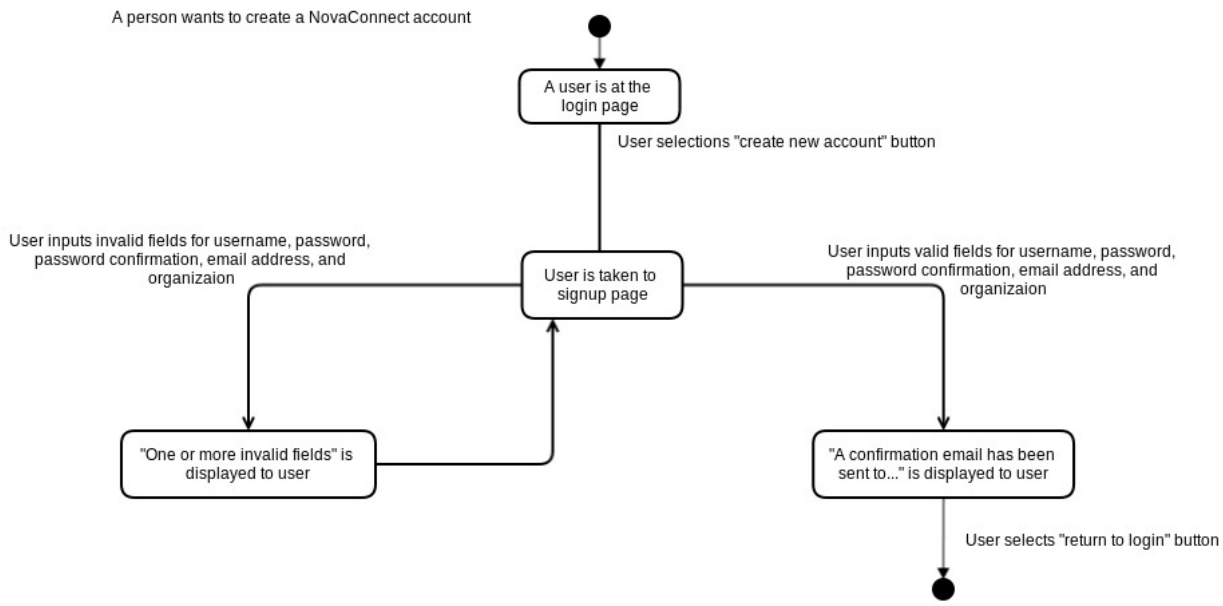
2.3 Database

Requirement	Description
Auto Backup	The database will periodically backup its contents to another backup partition.
Scalability	Ability to scale as well as store huge amounts of data.
Host on cloud	Make AWS instance.

Our database contains several different tables in MySQL for different situations in MySQL. For example, we have an implemented tables for the creation of conferences, users, and admins. From MySQL tables, we can input and pull information from the front-end web interface, using simple embedded ruby block codes. MySQL is a high performing database that provides scalability and flexibility. One could easily learn to create/access a MySQL database because of it's wide use on the internet.

3. User Stories

3.1 An Individual Wants to Create a NovaConnect Account



This user story is the first step towards many of the goals desired to be achieved by NovaConnect. By being able to successfully create an account, a user can take part in the forthcoming functionality, such as creating a conference and adding new leads.

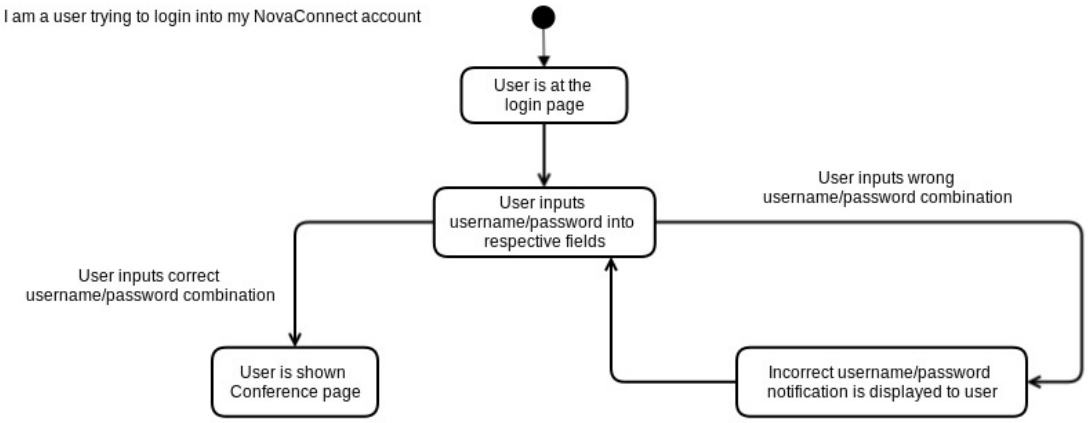
3.1.1 Successful Account Creation

An account creation is considered successful when a user has valid inputs for username, password, password confirmation, email address, and organization on the signup page. Validation will be confirmed through email.

3.1.2 Unsucessful Account Creation

An account is considered unsuccessful when a user does not input valid information for any of field of username, password, password confirmation, email address, and organization. A unsuccessful login will redirect the user back to the signup page.

3.2 A User wants to Login to their NovaConnect Account



Once a user has successfully been able to log into his or her NovaConnect account, they will be shown to the main page, which allows them to take part in lead entering functionality. Thus, to log into his or her account is an essential aspect of NovaConnect.

3.2.1 A User Inputs His/Her Username/Password Combination

Once the user is on the login page, they will have fields corresponding to username and password. This is where they enter their respective information.

3.2.2. A User Unsuccessfully Logs Into His/Her Account

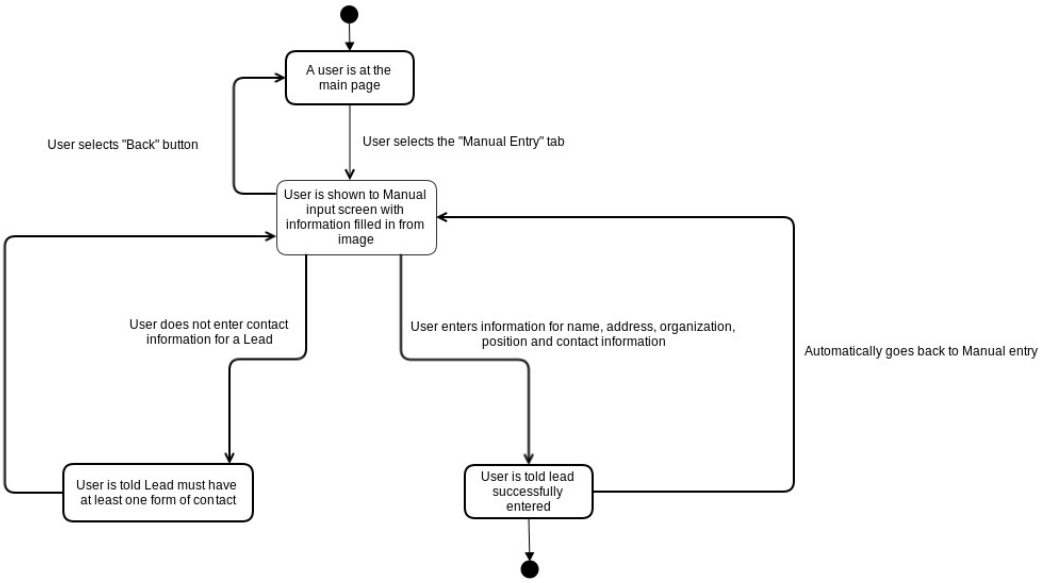
If a user submits an invalid combination of username/password the user will be alerted that he or she has done so. He or she will be brought back to the login page.

3.2.3. A User Successfully Logs Into His/Her NovaConnect Account

If a user submits a valid combination of username/password, the user is brought to the Main page.

3.3 A User Wants to Manually Add a Lead

The default method for submitting a new lead into the system is to enter the lead's information manually. This is done through various text fields.



3.3.1 A User Selects the Manual Entry Icon

When a User selects the Manual Entry icon the User will be presented with various text fields corresponding to the Lead's name, address, organization, position, and contact information.

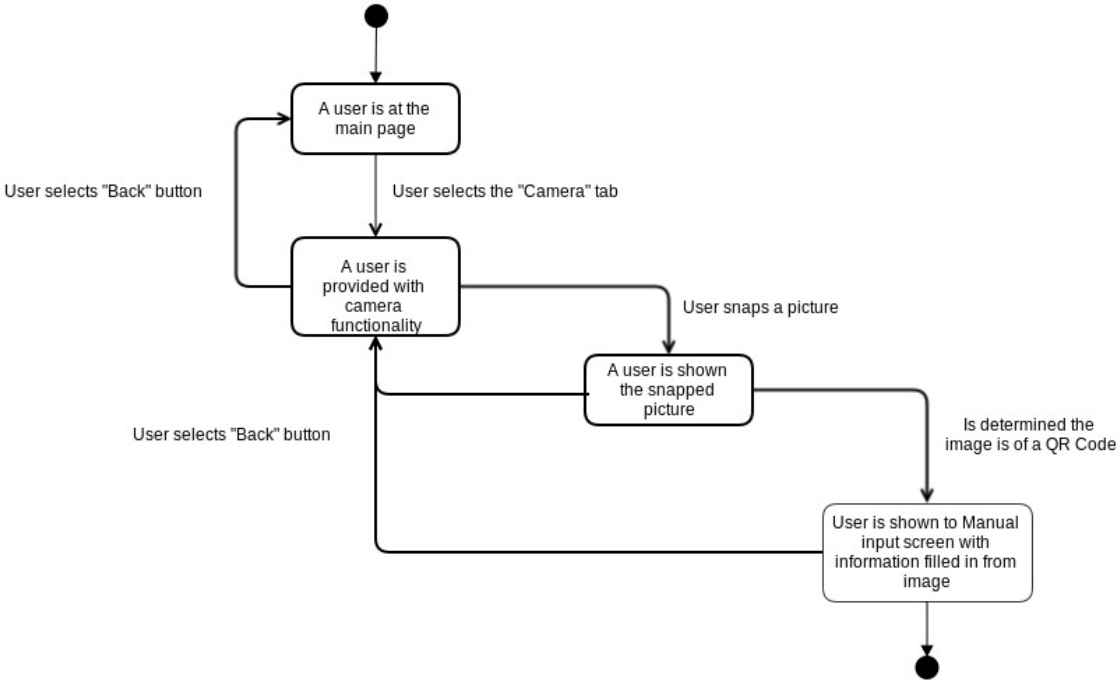
3.3.2 A User Submits a Lead with Invalid Contact Information

If a User submits a Lead without valid contact information, the lead is considered invalid. The User is prompted that the Lead must have at least one form of contact information to be added to the Leads.

3.3.3 A User Submits a Lead with Valid Contact Information

If a User submits a Lead with valid contact information, that Lead will be entered into the database. The User will be told that the Lead was successfully added, and they will be brought back to the Manual Entry screen.

3.4 A User Wants to Add a Lead via QR Code



The ability to add a Lead to the database using a QR code will have the desired effect of speeding up the process of adding a Lead. This would be useful in cases where many leads need to be added at once or very quickly.

3.4.1. A User Selects the Camera Icon

When the User selects the Camera icon they will be taken to the camera interface. Here they will be able to snap a picture of a QR code or a business car.

3.4.2. A User Snaps a Picture

When the User snaps a picture, her or she will be shown this image. The user then has the ability to confirm whether or not they will use this image. If they choose not to use this image, they will be brought back to the camera functionality.

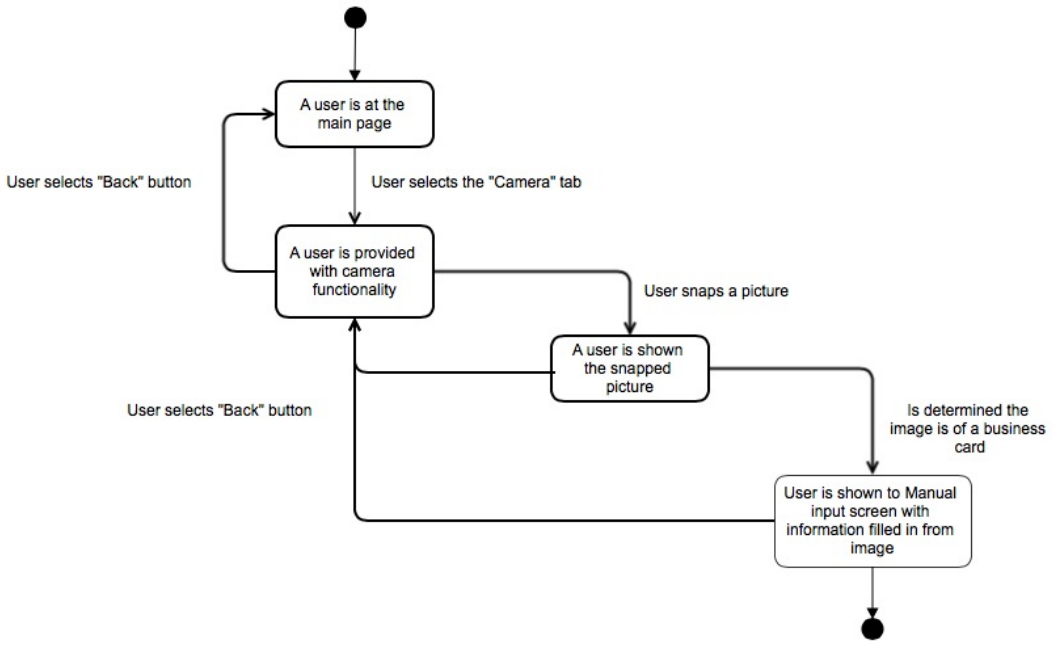
3.4.3. A User Confirms a Picture

If a User confirms the picture he or she, the User will be brought to the Manual Entry screen. Here they will find the QR code information populated to the respective fields. If a User is not happy with the information gathered from the QR code, the ability to go back to the camera functionality is given.

3.4.4. A User Confirms the Information

If a User submits a Lead with valid contact information, that Lead will be entered into the database. The User will be told that the Lead was successfully added.

3.5 A User Wants to Add a Lead Via a Business Card



The ability to add a Lead to the database using an image of a business card will have the desired effect of speeding up the process of adding a Lead. This would be useful in cases where many leads need to be added at once or very quickly.

3.4.1. A User Selects the Camera Icon

When the User selects the Camera icon they will be taken to the camera interface. Here they will be able to snap a picture of a QR code or a business card.

3.4.2. A User Snaps a Picture

When the User snaps a picture, her or she will be shown this image. The user then has the ability to confirm whether or not they will use this image. If they choose not to use this image, they will be brought back to the camera functionality.

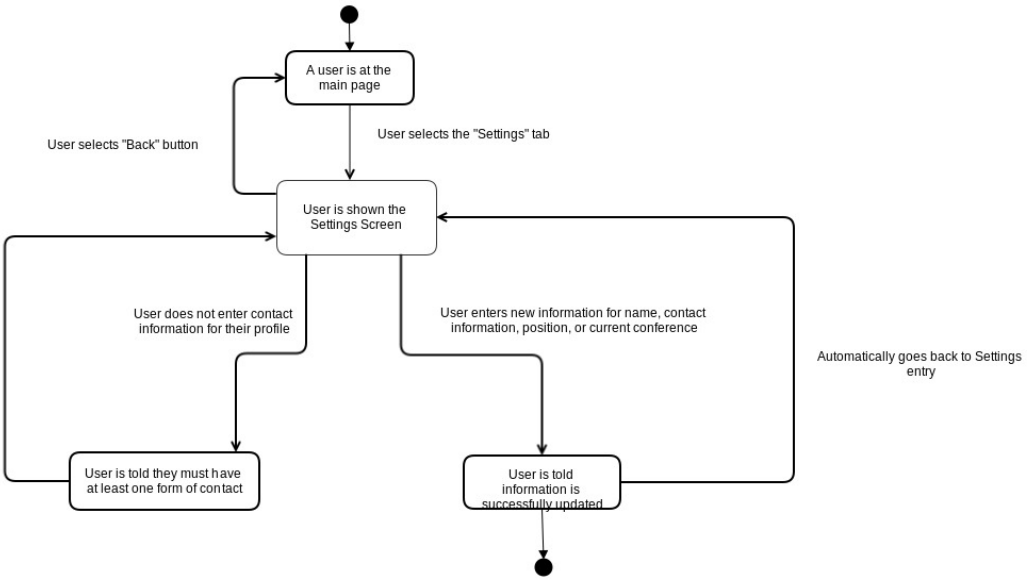
3.4.3. A User Confirms a Picture

If a User confirms the picture he or she, the User will be brought to the Manual Entry screen. Here they will find the business care information populated to the respective fields. If a User is not happy with the information gathered from the business card, the ability to go back to the camera functionality is given.

3.4.4. A User Confirms the Information

If a User submits a Lead with valid contact information, that Lead will be entered into the database. The User will be told that the Lead was successfully added.

3.6 A User Wants to Access and Change Personal Settings



Having an accurate personal profile is important for any conference attendee. In the Settings menu, a User has the ability to keep his or her information up to date.

3.6.1 A User Wants to View Personal Information

When a User selects the Settings icon, they will be brought to a list of his or her own personal information. This includes a picture, name, address, contact information, position, and the conference currently being attended.

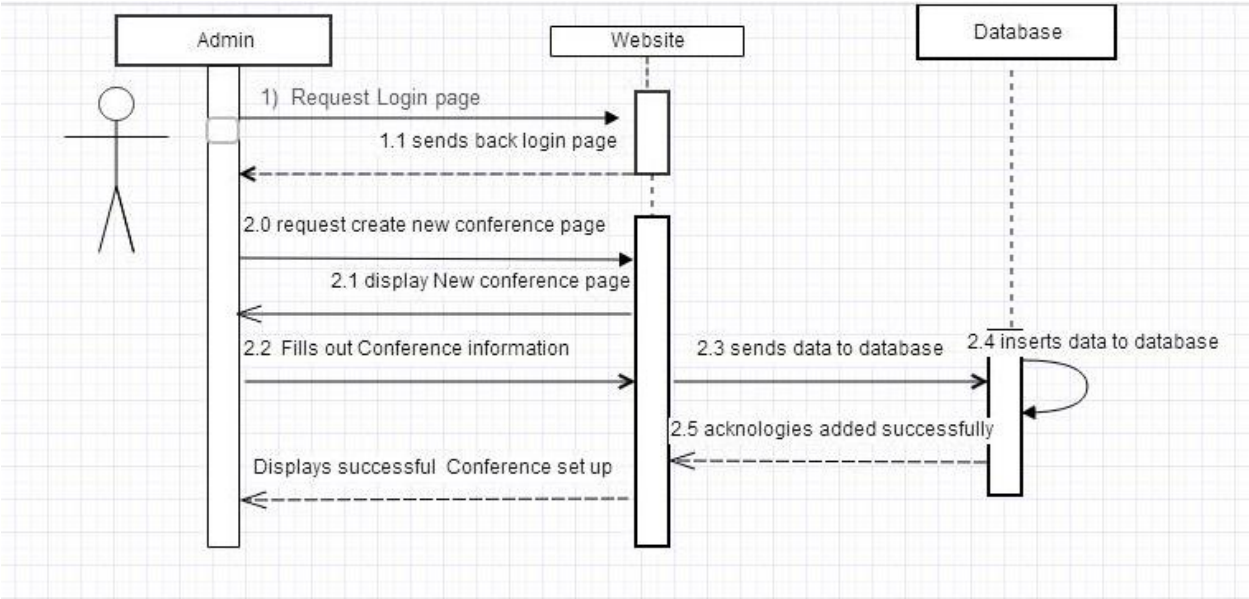
3.6.2. A User Inputs Invalid Contact Information

If a User make a change to his or her own personal information, and the contact information is found to be invalid, the User will not be allowed to make this change. A notification telling the User that invalid contact information will be displayed.

3.6.3 A User Input Valid Contact Information

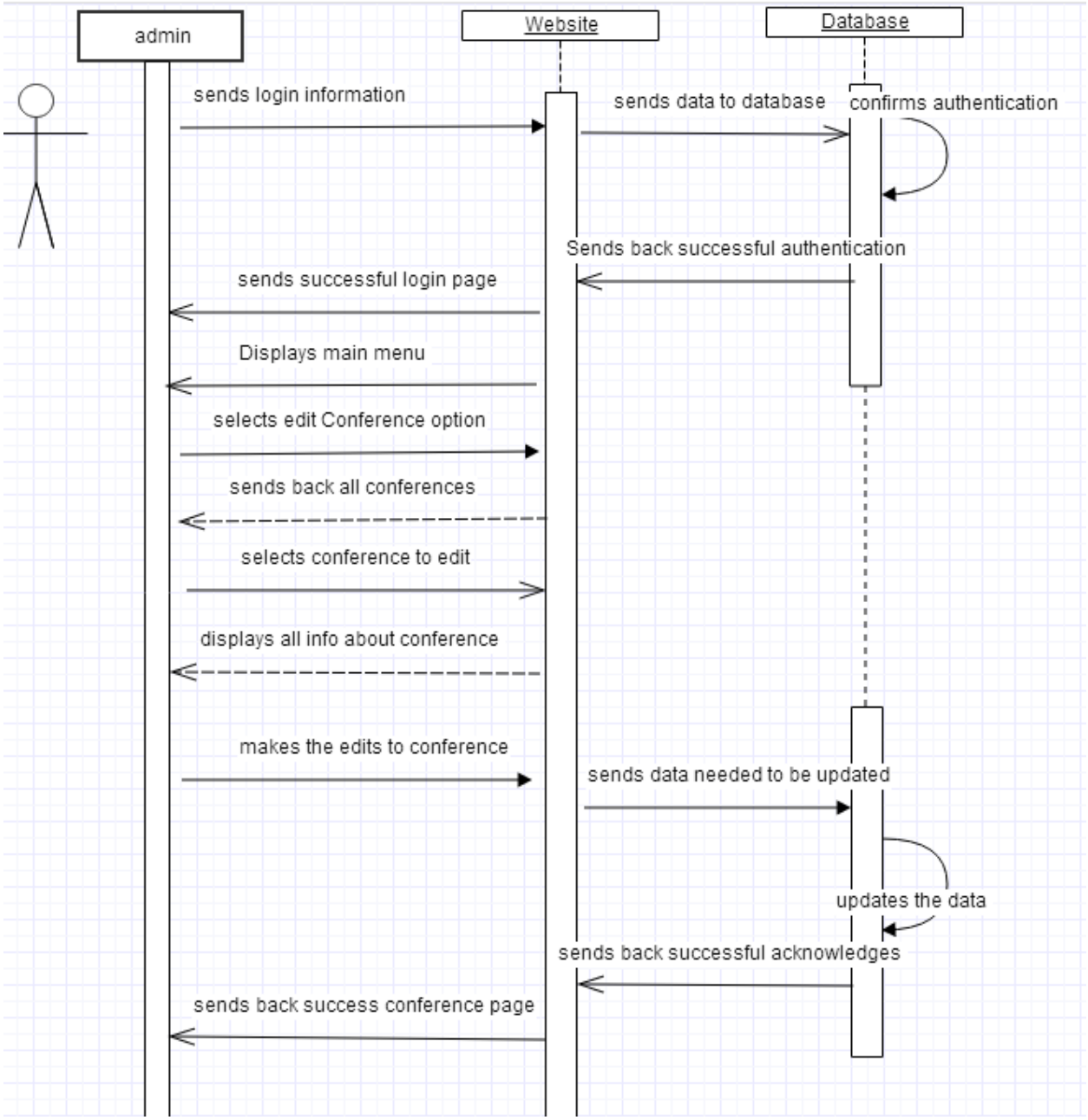
If a User makes changes and his or her contact information is found to be valid, the User will be told that the information has successfully been updated. It will then automatically go back to the Settings menu.

3.7 A user wants to create a new Conference



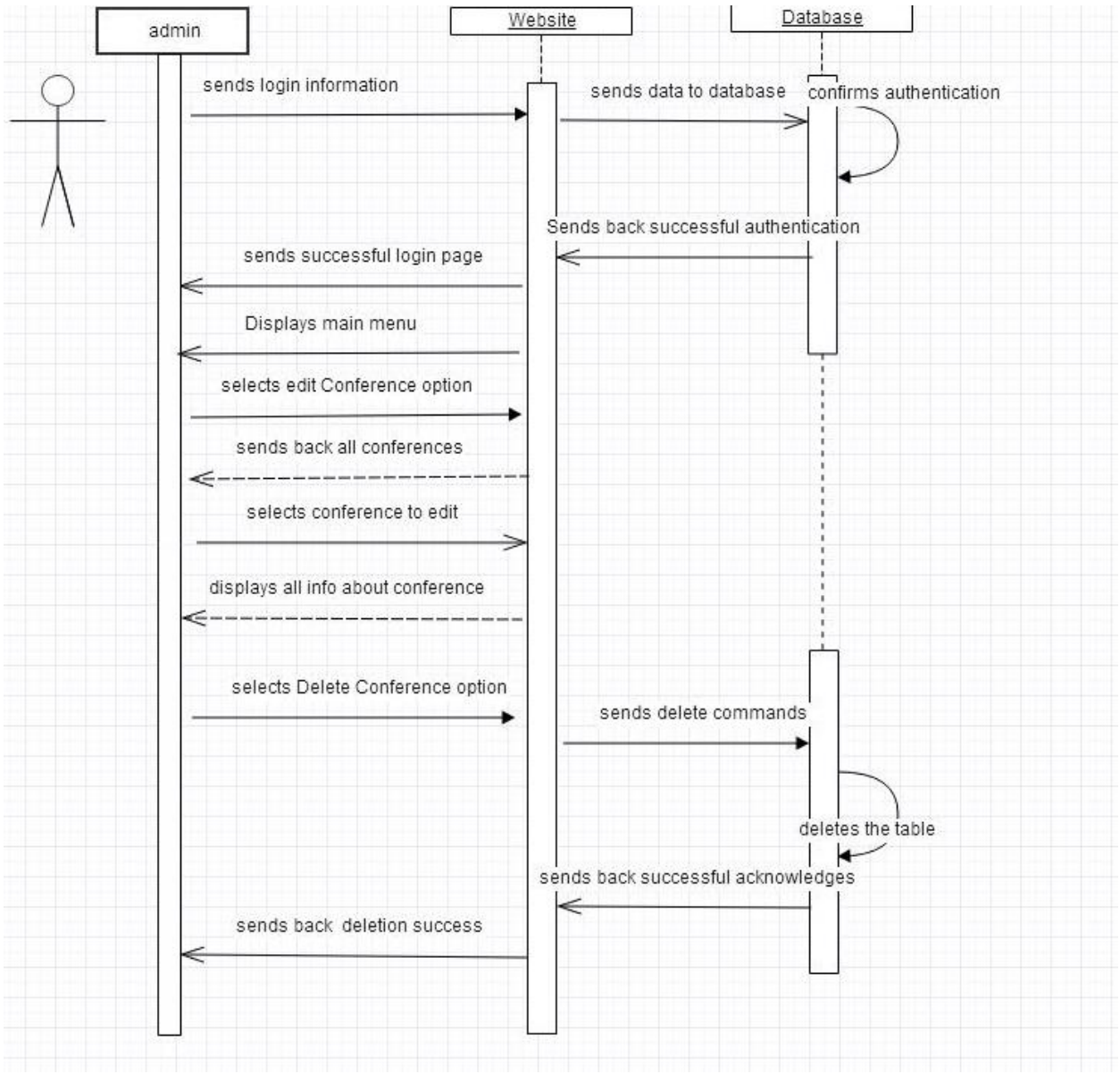
The figure above is the sequence diagram for creating a new conference. The summary of the diagram is that the admin user will first login and select the option to create a new conference. Then the user will insert the information needed to set up the new conference. Once the user has completed the info, they will submit it and the info would be sent to the database so the database can create the new table with the given information.

3.8 A user wants to edit information about conference



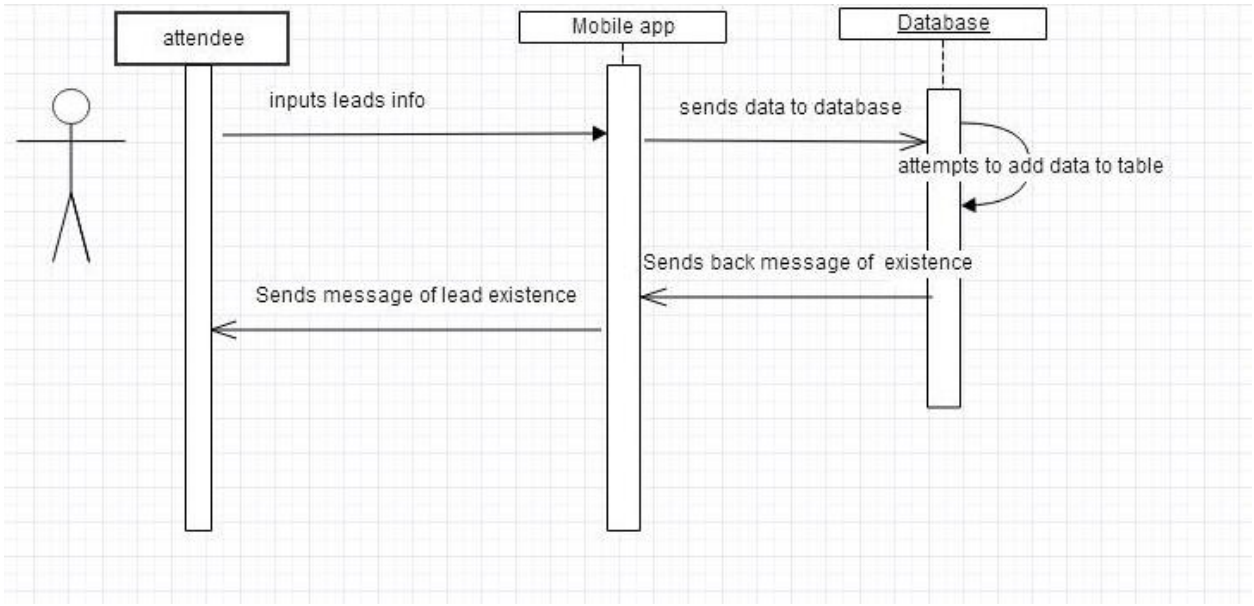
The figure above is the sequence diagram when the admin user wants to edit information on a conference that has been created already. The summary of the diagram is that the user first logs in and gets authenticated by the database. Then the user chooses the option to edit conferences and selects the conference. Once the user edits the information that is needed to be updated the webpage sends the data back to the database where the database updates the table with the given information.

3.9 A user wants to Delete a conference



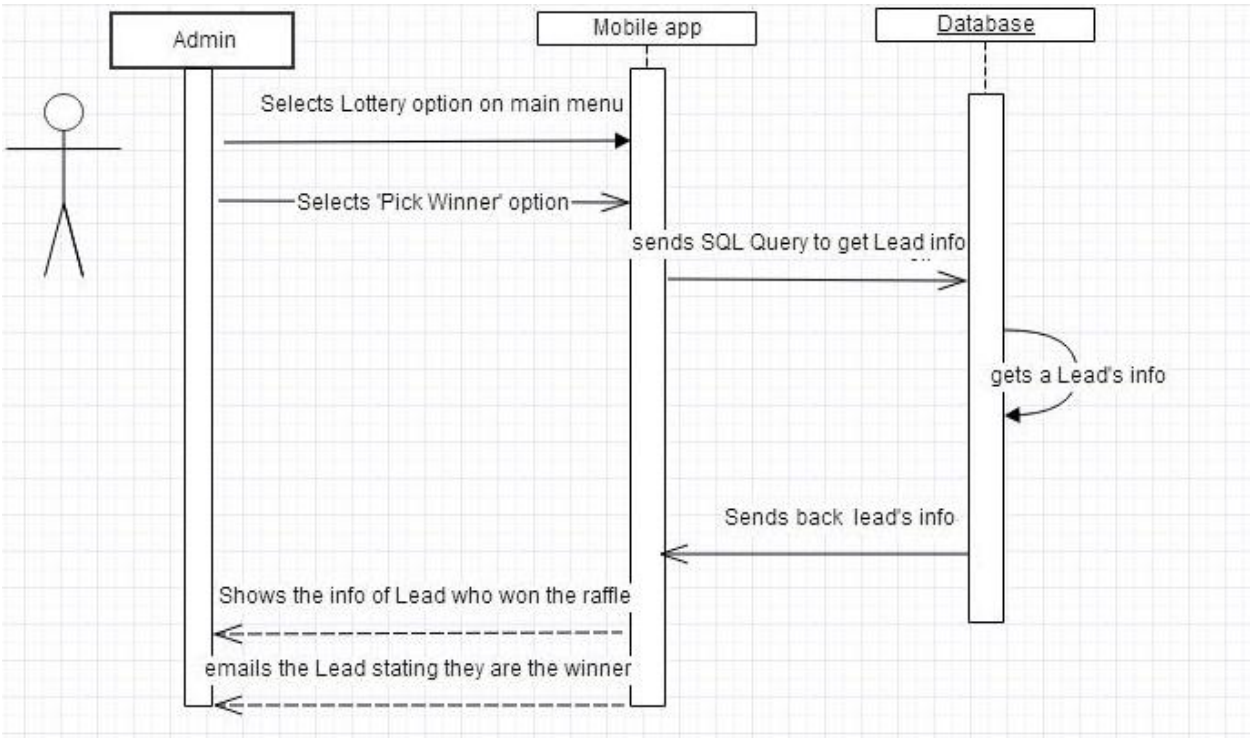
The figure above is the sequence diagram for when the admin user wants to delete a conference from the database. The user will first have to log in. Then once the user has successfully logged in the user will select the edit conference option and select the conference of choosing. In the edit page there will be a delete conference option where once selected the webpage will send SQL queries to the database the correlate with deletion of the conference. Once the database has successfully deletes the table it will send back an acknowledgment to the website and the website will show the user a deletion success text.

3.10 The user adds a Lead that is already in the database



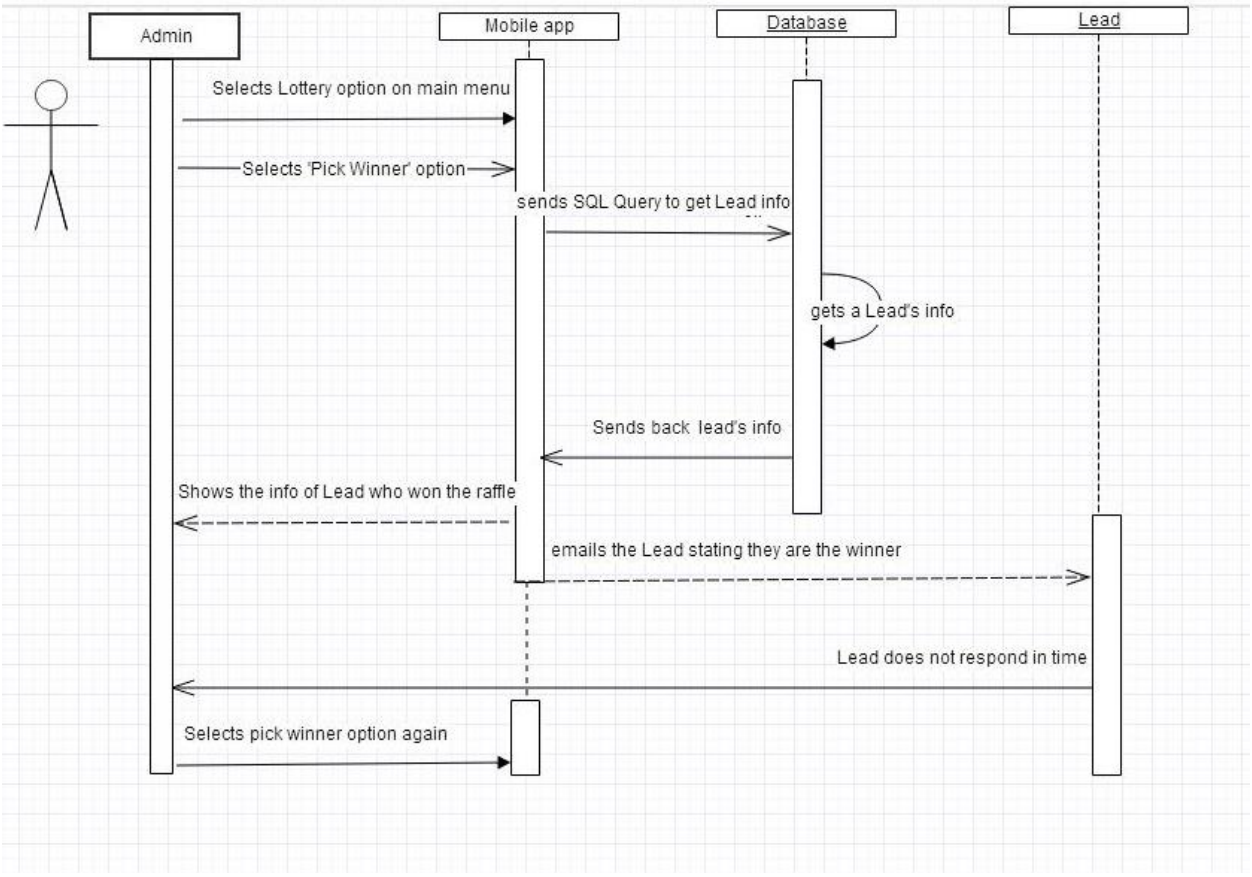
This figure shows the sequence diagram for when attendee inputs leads information and the lead already exist in the leads database. The user will use the mobile app and will either manually enter the information or take a picture of the leads business card or take a QR code and then send the data over to the database. In this scenario the lead already exist so the database will not add a new row to the table and it will send a message of existence to the app.

3.11 The User wants to select a winner for the Lottery



This figure shows the sequence diagram for when an admin wants to randomly select a winner from new leads they got at the conference. The admin user will select the Lottery option from the main menu. The mobile app will randomly choose a lead as a winner from the database and display the user the information about the lead. Also the app will email the lead telling them that they have won the lottery and gives instruction on how to claim prize and the time limit.

3.12 The Lead does not respond in time when a user selects a winner from lottery



This figure shows the sequence diagram for when an admin wants to randomly select a winner from new leads they got at the conference. The admin user will select the Lottery option from the main menu. The mobile app will randomly choose a lead as a winner from the database and display the user the information about the lead. Also the app will email the lead telling them that they have won the lottery and gives instruction on how to claim prize and the time limit. In this scenario the lead does not respond in time therefore the admin user selects a new lottery winner and the rest repeats.

4. UI Mockups

4.1. A User Creates a New NovaConnect Account

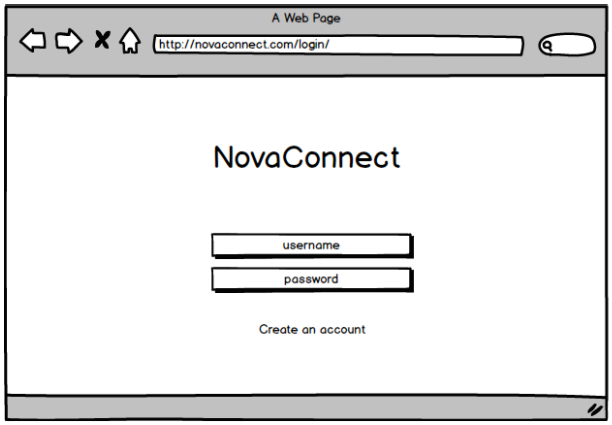


figure 4.1.1.

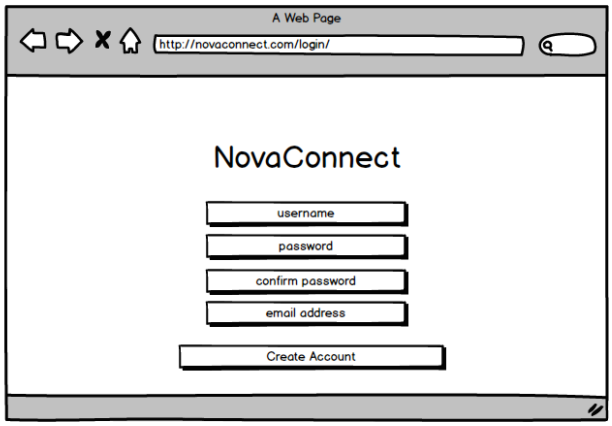


figure 4.1.2

4.1.1. The Login Page. Here the User has the ability to create a new account

4.1.2. The Account creation prompt.

4.2. A User Logins Into a NovaConnect Account

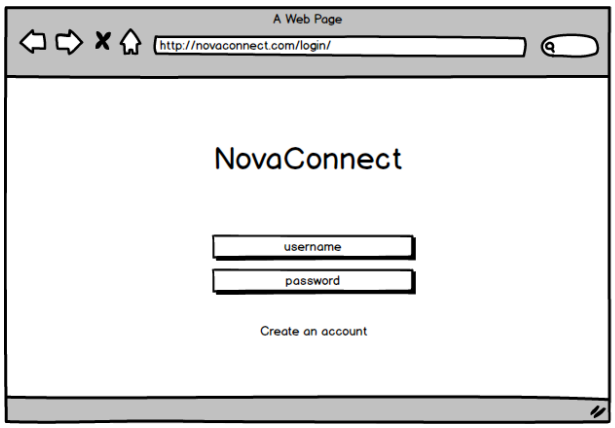


figure 4.2.1.

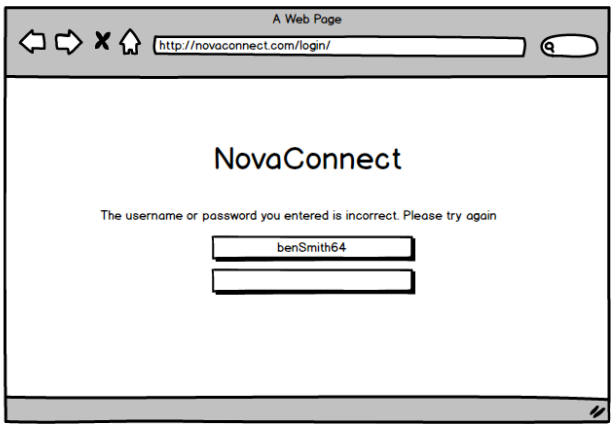


figure 4.2.2.

4.2.1. The Login Page. Here the User will input his or her username and password

4.2.2. An example of a case where the User cannot log in.

4.3. Examples of a QR Code Read, Business Card, and Manual Entry

figure 4.3.1.

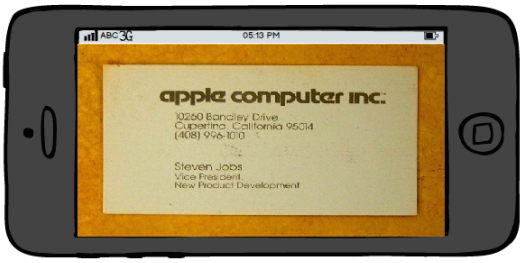


figure 4.3.2.

figure 4.3.3.

4.3.1. An example of a QR code being read, after the user has selected the Camera tab.

4.3.2. An example of a business card being read, after the user has selected the Camera tab.

4.3.3. An example of a manual entry of a lead. This is also prompted after a QR Code is read or a business card is read.

4.4. A User Selects the Settings Tab

figure 4.4.1.

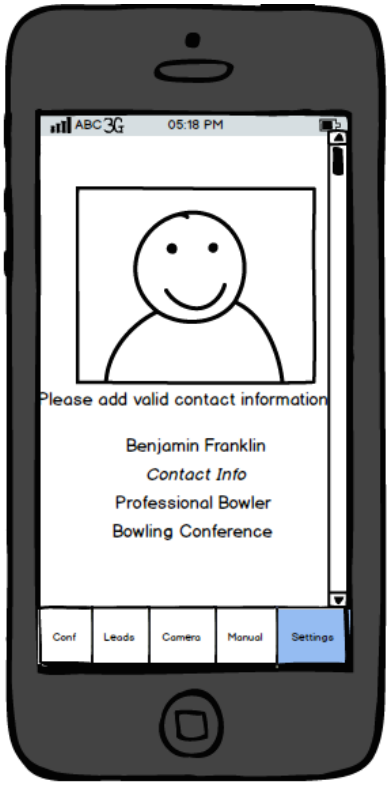
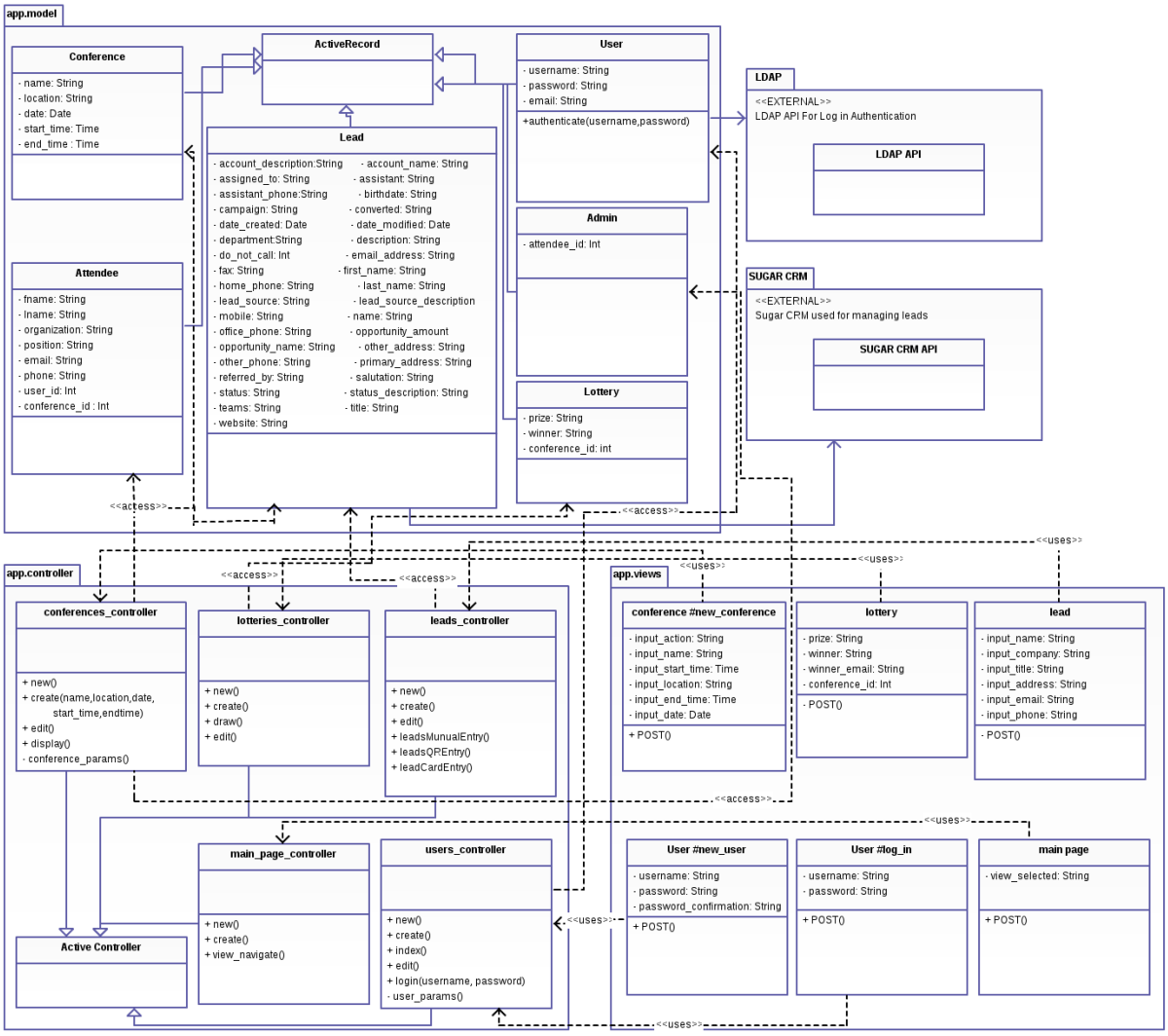


figure 4.4.2.

4.4.1. A User selects the Settings icon.

4.4.2. A case of a User trying to change their Personal Settings without having valid contact information.

5. Class Diagram



The figure above is the class diagram. This UML depicted our application's models, controllers, views relationship/interaction.

6. Testing

6.1. Testing Login functionality

```
def TestLogin()
        u = User.new('Andrew','david','david@david.com')
        result = User.authenticate(u.username,u.password)
        assert_equal(result, true)
end
```

6.2. Testing of a Unique Lead

```
def TestLeadUniqueness() #Checks if lead is already in database or not
        result = !Lead.find_by_id(this.lead_id).nil? #prototyped
        assert_equal(result, true)
end
```

6.3. Tests if Lead is Valid

```
def TestValidLead()
        l = Lead.new(124)
        l_entry = Lead.find_by_id(l.id)
        name_result = l_entry.name.length > 2 || l_entry.name.length < 32
        company_result = l_entry.lead_source.length > 2 || l_entry.lead_source.length <
        24
        title_result = l_entry.title.length > 3 || l_entry.title.length < 20
        address_result = l_entry.primary_address > 10 || l_entry.primary_address < 40
        email_result = send_email(l_entry.email_address)
        phone_result = verify_phone_regex(l_entry.mobile)
        result = name_result && company_result && title_result && address_result &&
        email_result && phone_result
        assert_equal(result,true)
end
```

6.4. Tests if User is Valid

```
def TestValidUser()
        u = User.new('Ernesto','cojulun','edanc')
                email_result = send_email(u.email) # send email to the address u.email
        as a simple
        # method to validate email
        username_result = u.username.length > 4 || u.username.length < 12
        password_result = u.password.length > 4 || u.password_length < 16
        assert_equal(email_result && username_result && password_result, true)
```

```
end
```

6.5. Tests if Conference is Valid

```
def TestValidConference()
        conf = Conference.new(4)
        c = Conference.find_by_id(conf.id)
        name_result = c.name > 4 || c.name < 20
        date_result = valid_date_format(c.date)?
        time_result = valid_time_format(c.start_time) && valid_time_format(c.end_time)
        && c.end_time - c.start_time > 0
        result = name_result && date_result && time_result
        assert_equal(result, true)
end
```

6.6. Tests if Lottery is Valid

```
def TestValidLottery()
        l = Lottery.new(5)
        l_entry = Lottery.find_by_id(l.id)
        result = !l_entry.puzzle.nil? && !l_entry.winner.nil? && !l_entry.puzzle.empty?
        && !l_entry.winner.empty?
        conf_result = Conference.new(l_entry.conference_id) #checks if the
        corresponding #conference id exists in the database
        result = result && conf_result
        assert_equal(result, true)
end
```

6.7. Tests if LDAP is Valid

```
def TestLDAP()
        result = false
        conn = LDAP::Conn.new(host ='localhost', port=LDAP_PORT)
        conn.bind(root, localhost, localdomain, secret)
        result = true
        conn.unbind
        assert_equal(result, true)
end
```