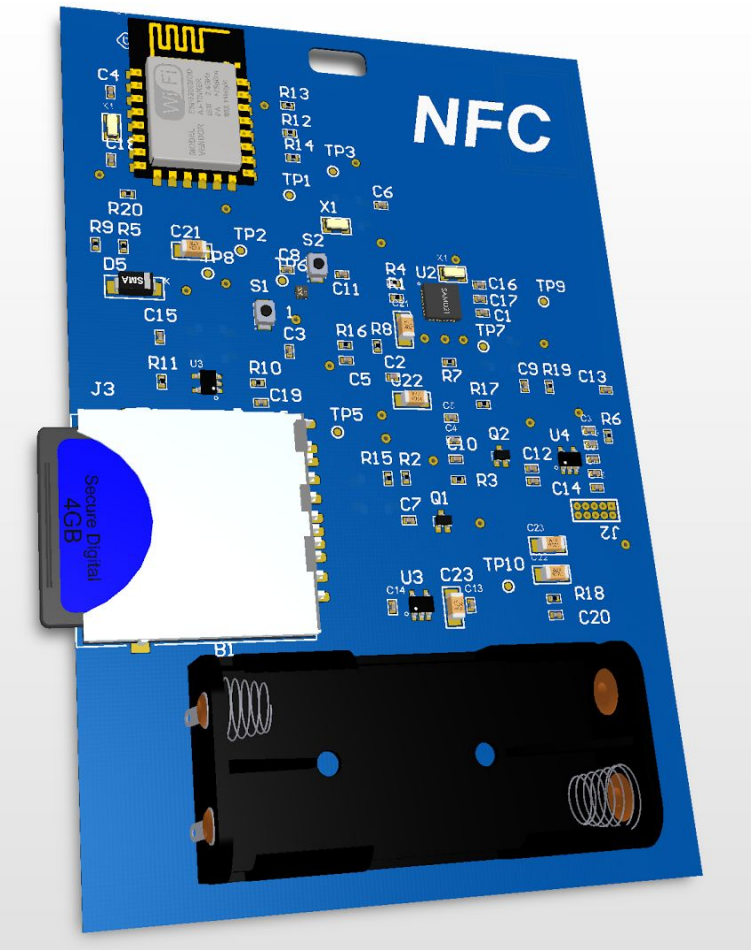
TouchBadge

An Internet Connected Lanyard for Conferences



Computer Science and Engineering Department, University of Nevada, Reno

Team 11: Evan Grill, Jon Weatherspoon, Cody Worsnop

Instructors: Sergiu Dascalu, Devrin Lee

External Advisors: Daniel deLaveaga and Elizabeth Hitchcock of Breadware, Inc.

November 15, 2017

Table of Contents

Abstract

Conferences exist to cater to almost every industry, and some companies hold conferences entirely dedicated to their own customers. Theses events are prime locations for networking with other people in the industry as well as potential customers. For vendors at these events, conference badges are an important tool as they allow for the collection of attendee data for future contact. TouchBadge looks to bring that kind of experience to every conference attendee.

Working with hardware partner Breadware, Inc., we are developing TouchBadge, a smart conference badge. TouchBadge allows the transfer of contact information between attendees wirelessly and without an external reader. Once obtained, contacts are stored in the cloud for later retrieval and are exportable to the user's customer relationship management (CRM) software of their choice.

Introduction

TouchBadge is a smart conference badge. Designed with hardware partner Breadware, Inc., it consists of an electronic badge equipped with near-field communication (NFC) and Bluetooth protocols to allow for interaction between badges as well as smartphones for syncing to the TouchBadge web service. The goal of TouchBadge is to reduce the time and effort it takes to share information in order to enhance the networking experience of conference attendees.

Since the project specification, the design has shifted slightly. The main focus is still present, but the hardware interaction has been delegated to the hardware design partner, Breadware, Inc. The team has decided on a web service centered design with interactions made through the web-based attendee application portal (AAP) and conference organizer application portal (COAP), or the companion mobile application, being developed for Android and iOS.

The web service will stand as a common point of communication for all of the various access interfaces available to users. Additionally, it will handle the database management keeping the interactions in and out as clean as possible. Security will be handled by requiring the use of Secure Sockets Layer (SSL) for the communications. Additional security will be provided in the form of an authentication token unique to each user, which will be required for all interactions and can be revoked by the security team in case of compromise.

The mobile application will be the primary point of interaction for the attendees. It will provide the ability for the user to activate or reactivate (in the case of failure) a TouchBadge device. Additionally, it will serve as the initiator for syncing the TouchBadge to the user's mobile device.

The AAP will allow the user easy access to all of their obtained contacts, separated by event, and provide the ability to export data to other platforms (e.g. CRMs, Outlook, etc.). The COAP will allow the conference organizer to create and manage conferences as well as manage the list of registered attendees.

High-Level and Medium-Level Design

System Level Design

The TouchBadge software architecture will contain several different design patterns in various different subsystem levels. Figure 1 illustrates the overall system architecture on the system level and provides dataflow connections.

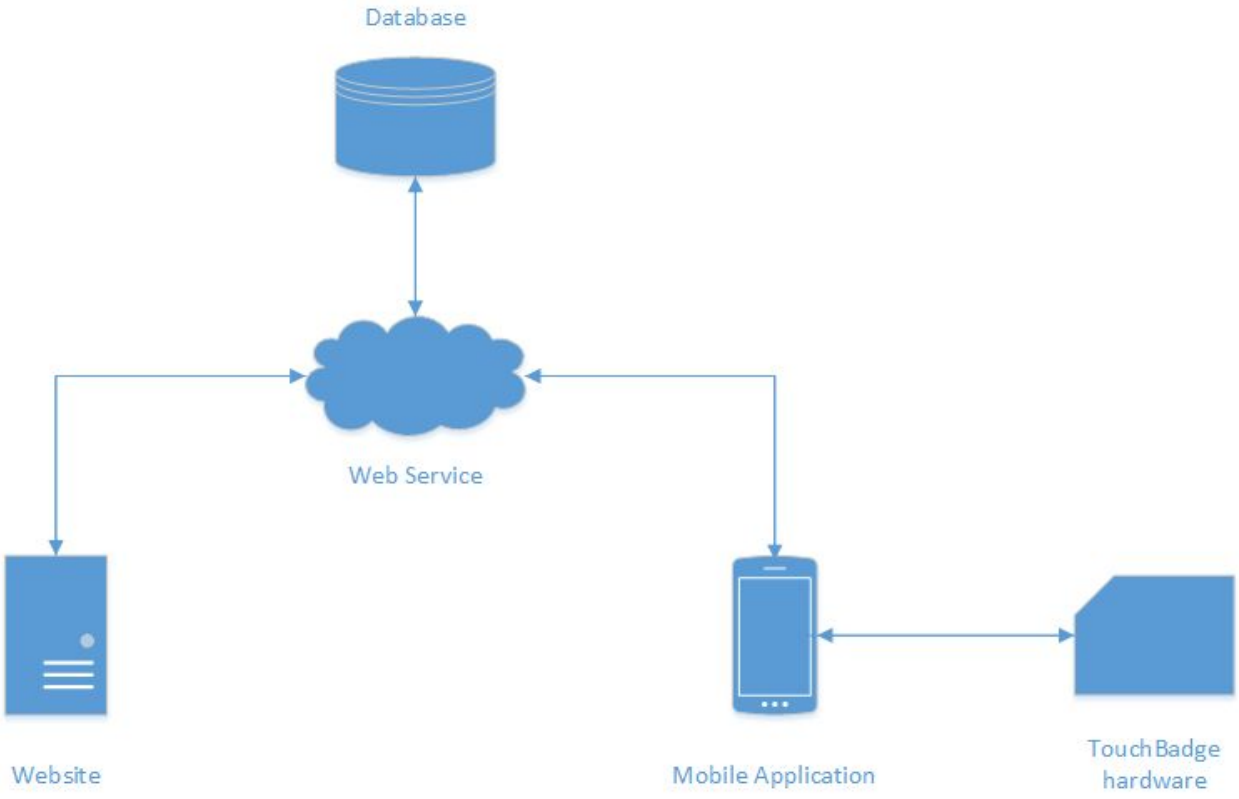


Figure 1: A system level overview of the various systems involved in making the TouchBadge function.

Database

A backend database shall be implemented to provide storage for all of the software systems. These will include: accounts created on the mobile application and / or website holding LinkedIn information, business relations, event attendance and more. A realtime connection will be established as to allow for these properties to update in realtime. The database itself shall be a SQL database implementation stored on AZURE. This will allow for easy integration between all the platforms developed and the backend.

Web Service
A web service shall be implemented as an interface for the mobile application and website to communicate with the database. The web service shall manage all of the incoming and outgoing dataflow between all devices.

Website
The website shall be implemented using ASP.NET in conjunction with the Bootstrap framework to provide a responsive web design. The website shall be able to create new user accounts and plan new conferences / events as well as view current events and contacts. This information will then be pushed to the web service and stored in the database. The website will be able to receive data about a particular user when requested. This data will be communicated through the web service.

Mobile Application
The mobile application shall be implemented using the Xamarin framework with C# and XAML languages. Because it is implemented with Xamarin, it will be built to both the iPhone and Android devices natively. This creates a complete user experience over other traditional approaches such as Ionic. It will share data between the database using the web service as well as communicate directly with the TouchBadge.

TouchBadge
The TouchBadge shall use NFC protocols to communicate directly with other TouchBadge devices. During such communication, both devices will participate in data validation, and once complete, user's personal contact information will be stored on the other device and vice versa. The TouchBadge shall use Bluetooth communication protocols to communicate with and sync to the user's mobile device when paired using the companion mobile application. In such cases, the mobile application will validate sent data, and will alert the TouchBadge to resend any data in the case of corruption.

Program Units

Mobile Application

The mobile application shall be implemented using the model-view-viewmodel (MVVM) design pattern and will therefore have three stages of separation per object. Due to this, only the models of these objects will be listed below. The following are predicted objects to be used within the mobile application; however, there will be much more added during actual development.

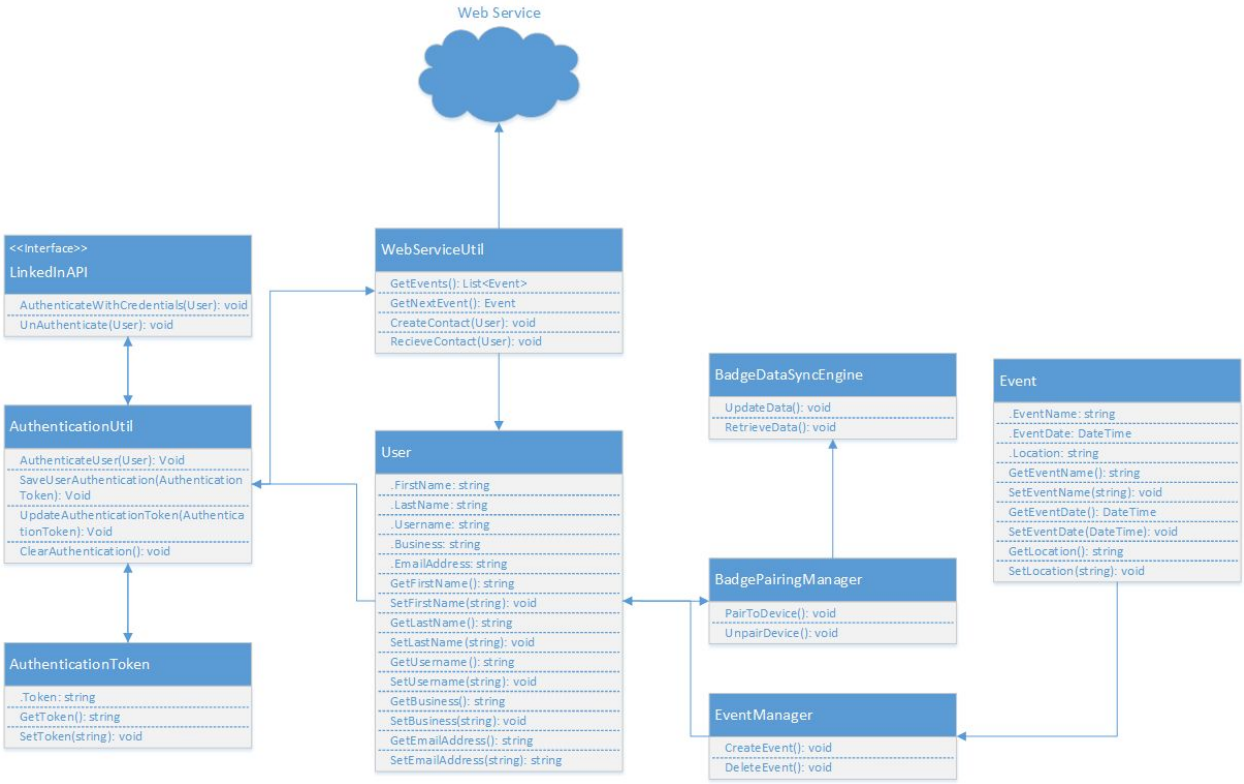


Figure 2: The class diagram presenting the relationships between the object for the mobile application. This diagram stands as an overview for the application, as there will be more added later on during the implementation phase.

Figure 3: The WebServiceUtil object. The WebServiceUtil object is designed as an interface between the mobile application and the backend for data communication purposes.

Purpose

The purpose of the web service utility is to provide a common interface between the mobile application and web service. All data flow out of the mobile application shall exist through the web service utility.

Methods

GetEvents	The GetEvents method shall retrieve a list of all the events the user currently has scheduled.
GetNextEvent	To simplify certain operations, GetNextEvent shall retrieve the next event and return one event.
CreateContact	The CreateContact method shall create a new contact in the database for the calling user.
ReceiveContact	The ReceiveContact method shall return a list of all contacts that belong to the calling user

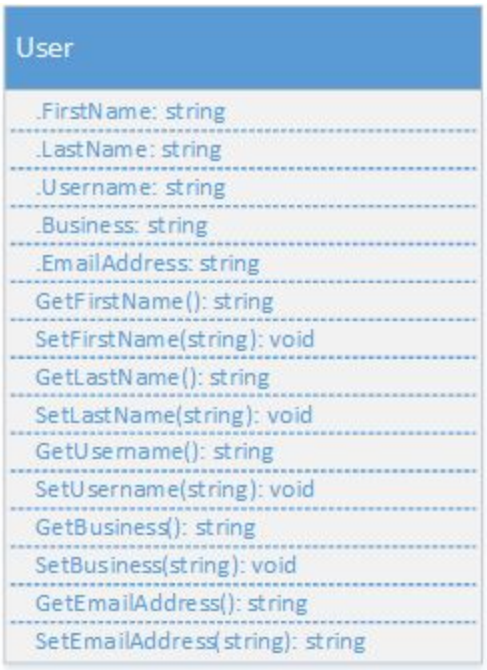


Figure 4: The User object, which is designed to encapsulate all useful user information.

Purpose

The purpose of the user object is to correctly and efficiently model every aspect of the user. It will contain user properties and will later implement more advanced methods.

Methods

GetFirstName	The getter for the first name field
SetFirstName	The setter for the first name field
GetLastName	The getter for the last name field
SetLastName	The setter for the last name field
GetUsername	The getter for the username field
SetUsername	The setter for the username field
GetBusiness	The getter for the business name field
SetBusiness	The setter for the business name field
GetEmailAddress	The getter for the email address field
SetEmailAddress	The setter for the email address field

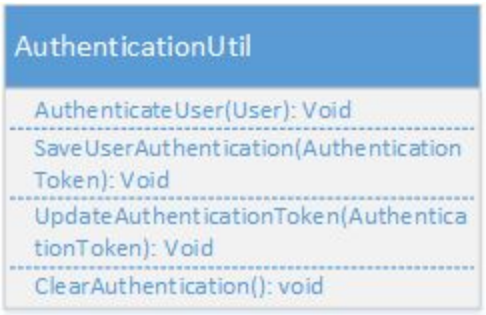


Figure 5: The AuthenticationUtil object, which will allow the user to authenticate to TouchBadge servers quickly and simply, while still maintaining security.

Purpose

The purpose of the Authentication utility is to provide an interface for authenticating the user against the LinkedIn API. It will take a user object and use its credentials to attempt to verify. It will also keep the authentication token object as a reference of the authentication to re-login the user when the app is opened later on.

Methods

AuthenticateUser	The AuthenticateUser method shall take a user object to authenticate with the LinkedIn API. Upon success the user shall be able to enter the application
SaveUserAuthentication	The authentication data will be savable so the user can open the app in another instance and be auto-loggedin with LinkedIn. This prevents the user from entering credentials every subsequent time they want to use the app.
UpdateAuthenticationToken	The update authentication token method shall refresh the token if the application is still active on the phone.
ClearAuthentication	The clear authentication method shall delete the authentication token if user logs out of the application

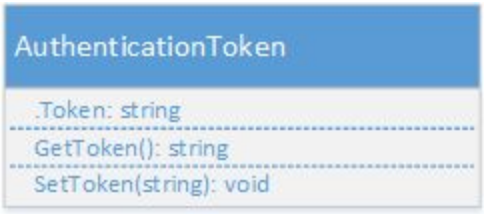


Figure 6: The AuthenticationToken object. TouchBadge authentication tokens will allow users to easily and securely maintain their authentication session with TouchBadge servers for better ease-of-use.

Purpose

The purpose of the Authentication token is to simplify the authentication token storage process. Typically when an API is authenticated with, the token is a string to keep the authentication during the session. This object will make that process easier by keeping the token and any artifacts associated with it in one place.

Methods

Get Token	The Get Token method will retrieve the last stored token. This method will be used to re-authenticate with the LinkedIn API
Set Token	The Set Token method will set the latest token into storage. This will be useful when re-authenticating with the LinkedIn API

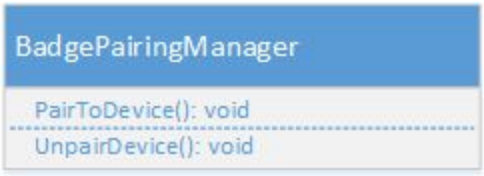


Figure 7: The BadgePairingManager object. The TouchBadge requires users to pair their device with the companion mobile application in order to unlock full usability.

Purpose

The purpose of the Badge Pairing Manager is to provide a common interface for connecting to the badge hardware. It will implement two simple to call methods that will either pair or unpair the user's mobile device with the hardware. It will take into account signal strength of the badge and let the user choose what to pair to.

Methods

PairToDevice	The Pair To Device method shall let the user pair their mobile device to the touch hardware.
UnpairDevice	The Unpair Device method shall let the user disassociate the with the TouchBadge hardware.

Figure 8: The BadgeDataSyncEngine object, which is designed to update shared contact information on TouchBadge servers in real time. The BadgeDataSyncEngine object will ensure the reliability and integrity of user data by ensuring a backup always exists.

Purpose

The purpose of the BadgeDataSyncEngine is to continually push data to the cloud for access on the website or at another time. When the user is synced with their phone, once a contact is shared it will push all information to the SQL backend. This then will be replicated across all platforms.

Methods

UpdateData	The Update Data method shall push all unsynced data to the cloud. If connection is currently unavailable, the data shall be added to a waiting queue until connection is resumed.
RetrieveData	The Retrieve Data method shall fetch all unsynced data from the cloud.

Figure 9: The EventManager object, which will allow privileged users to create and delete events that they manage.

Purpose

The purpose of the EventManager is to keep all event related tasks under one object. It shall implement create event and delete event methods. When the user is in the calendar of the mobile application, they shall be able to add events as well as remove them.

Methods

CreateEvent	The Create Event method shall let users add events while viewing the event calendar. Essentially, the user is registering for an event that is already existing.
DeleteEvent	The Delete Event method shall let users remove events from the event calendar. This is unregistering the user from specific events.

Figure 10: The Event object. Used to model conferences and events in addition to all their associated properties.

Purpose

The purpose of the Event object is to allow easy modeling of any events. It shall hold all the properties of events and their associations.

Methods

GetEventName	Gets the event name field
SetEventName	Sets the event name field
GetEventDate	Gets the event date field
SetEventDate	Sets the event date field
GetLocation	Get event location field
SetLocation	Set event location field

Notes On Program Units

As the mobile application and website will both be communicating with the web service using frameworks native to Microsoft, Team 11 expects to be able to reuse some of the code from the application in the website. As such, we have left out the design of the web for the purposes of this document. Team 11 also acknowledges the lack of design for the web service as this exceeds the requirements for this document.

Database Tables

The database shall consist primarily of four tables. Figure 11 below illustrates the relation between these tables.

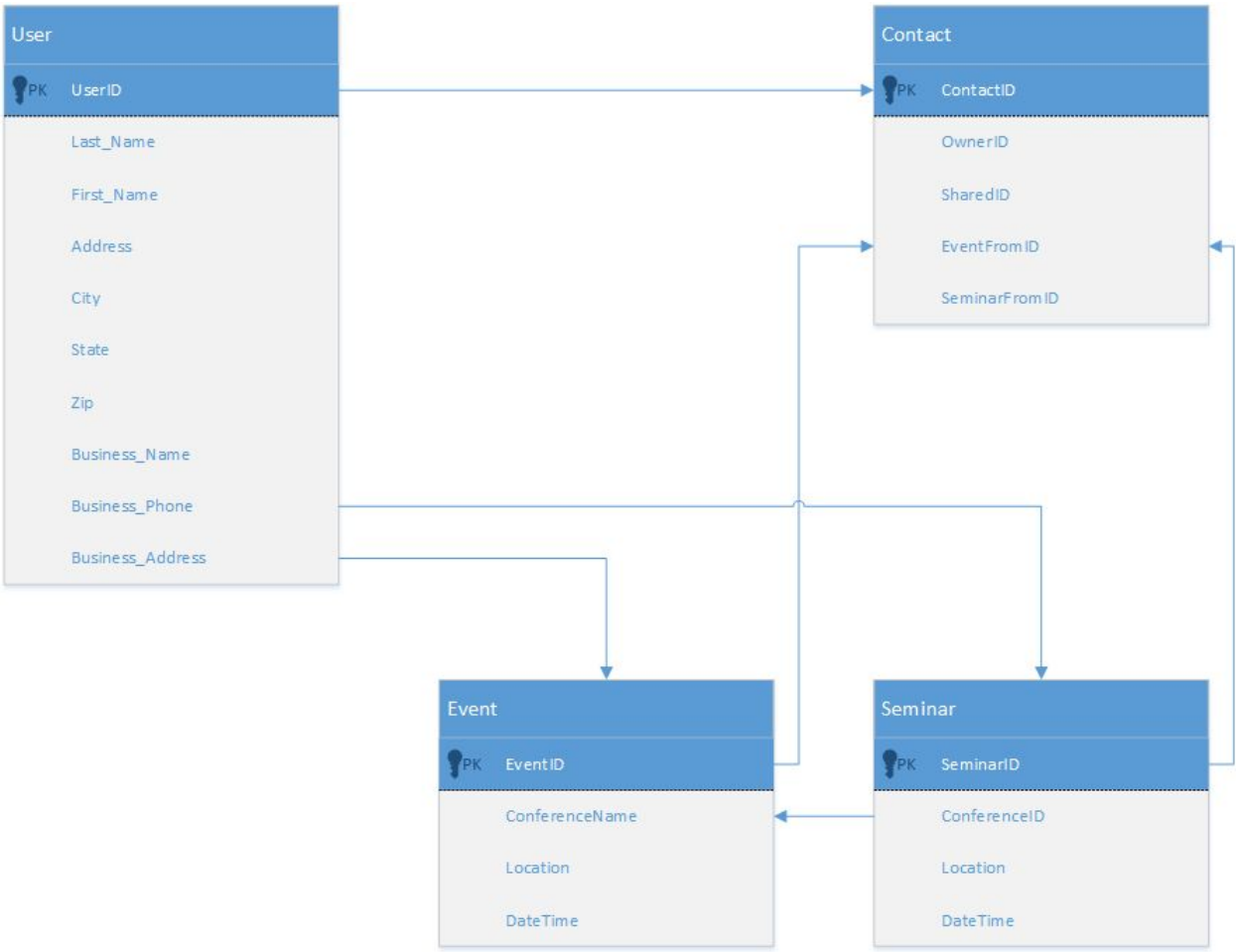


Figure 11: The database table diagram. This diagram presents a general outline of how the database will be designed.

User

The user table shall store all relevant information pertaining to the user object. This includes first name, last name, address, city, state, zip, business name, business phone, and business address. It has a one to many relationship with the contact table, event table, and seminar table.

Contact
The contact table shall store all relevant information pertaining to the contacts shared between users. It includes OwnerID, SharedID, ConferenceID, and SeminarID. The purpose of the planned fields is to allow a user to track their contacts by the conference and seminar they met them at..

Event
The event table shall store all relevant information pertaining to the events in the system. It includes conference name, location, and datetime. Users will be able to sign up for specific events and a history of past events for every user shall also be maintained.

Seminar
The seminar table shall store all relevant information pertaining to the seminars during events. It includes ConferenceID, location, and date time to allow for a relationship to exist between seminars to events in the database. Users will be able to see a history of attended seminars and who they met during what seminar.

Detailed Design

Hardware Statechart

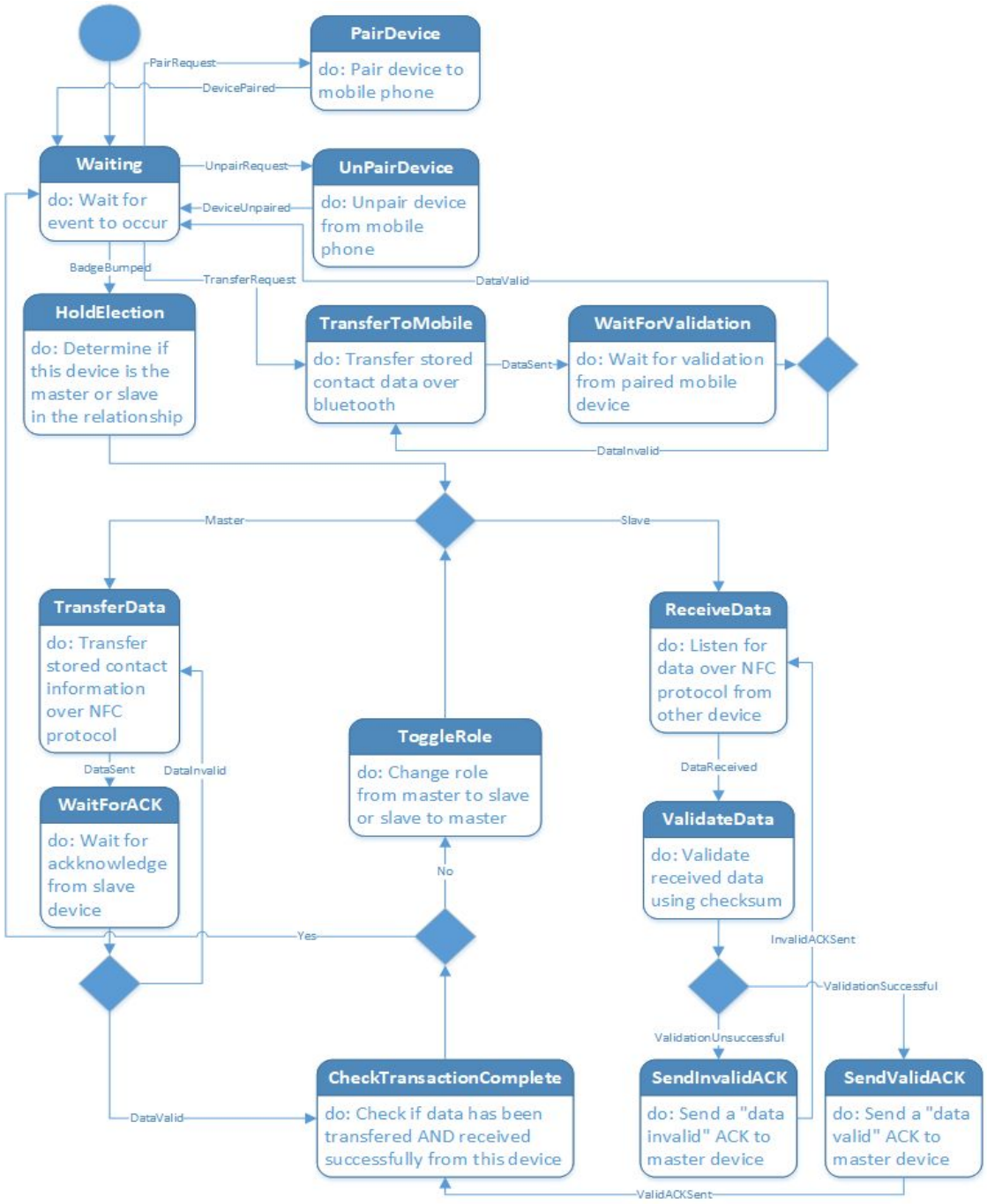


Figure 12: The hardware statechart describing all possible states the badge can take on. The statechart describes the data flow between hardware devices as well as between the TouchBadge and the companion mobile application.

Roles

An important note is that each badge will take on a slave or master role. If the badge is in a slave state then it will listen for data from the master device. Once this state is complete, it will take on the master role and send its own contact information.

Data Checking

Valid data checking has also been built into the hardware to ensure that contacts are fully and reliably sent. If a contact has been sent but it has been corrupted, the badges will attempt to resend the contacts. If successful, then the badge will return to the waiting state. If it fails, however, an alert message will appear on the companion app alerting the user to a possible problem with the badge.

Website Statechart

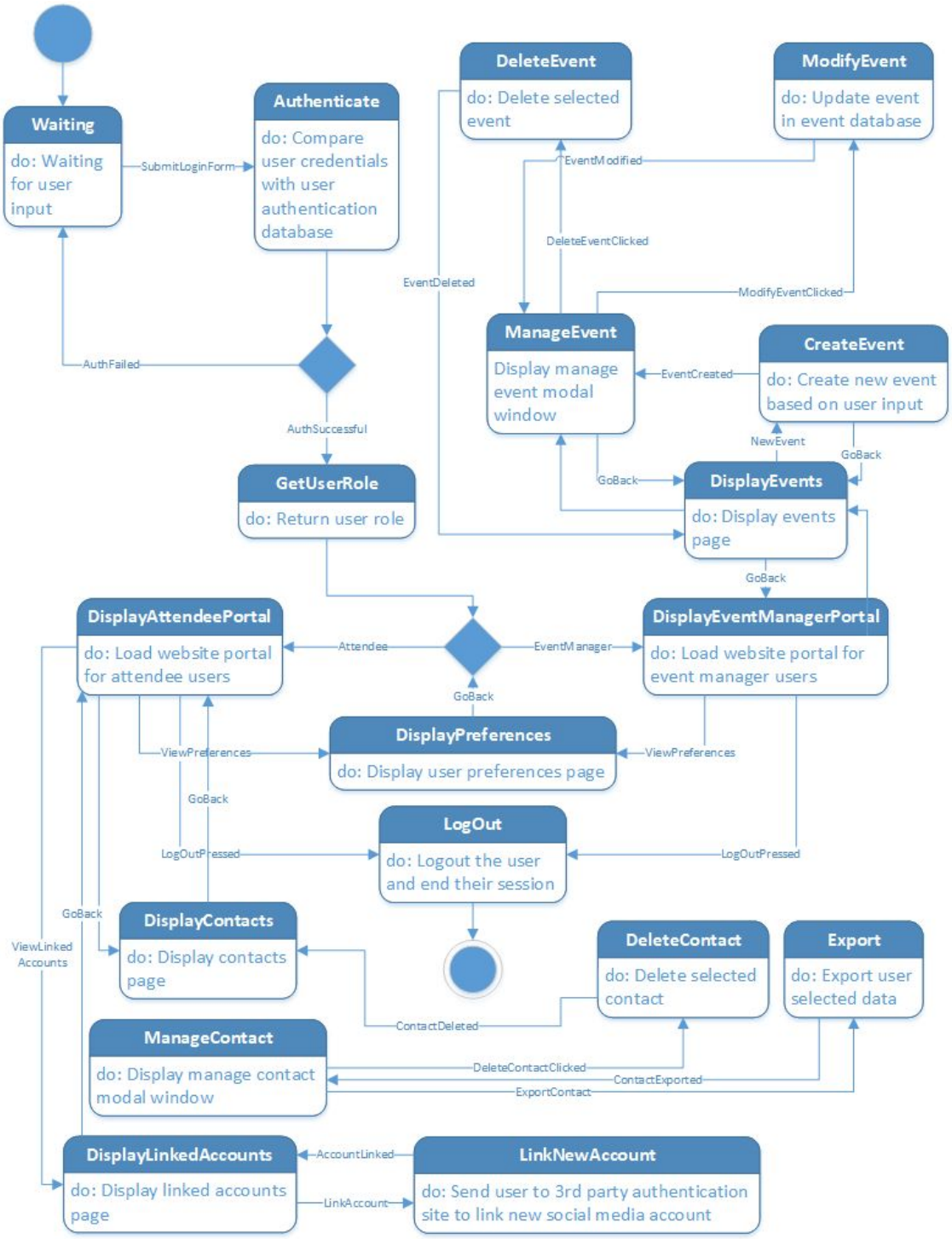


Figure 13: The website statechart describing all possible states the website can take on. The website statechart explains the data flow for users with either the attendee or event manager roles.

Attendee Portal

The attendee portal itself then has three main features: display contacts, manage contacts, and display linked accounts. Under display contacts, the user will be able to see all contacts made while at specific events. Under manage contacts, the user will be able to export contacts to a CRM, or delete made contacts if they have no need to keep them. Finally, the user will be able to look at linked accounts. Under this view, the user will be able to login to CRM accounts, social media accounts, and any other professional networks.

Event Manager Portal

If the website redirects to the event manager portal, the user will have two main features: create event, and manage events. When the user creates an event, they will fill in all the appropriate information including time, location, seminars, attendees, cost, etc. Once submitted, the user shall be taken to the manage event view where they have the ability to modify and delete events linked to their account.

Notes

All users shall have the ability to view personal preferences about their own personal account. This view is common and shall be accessible from either portal.

Mobile Application Flow Chart

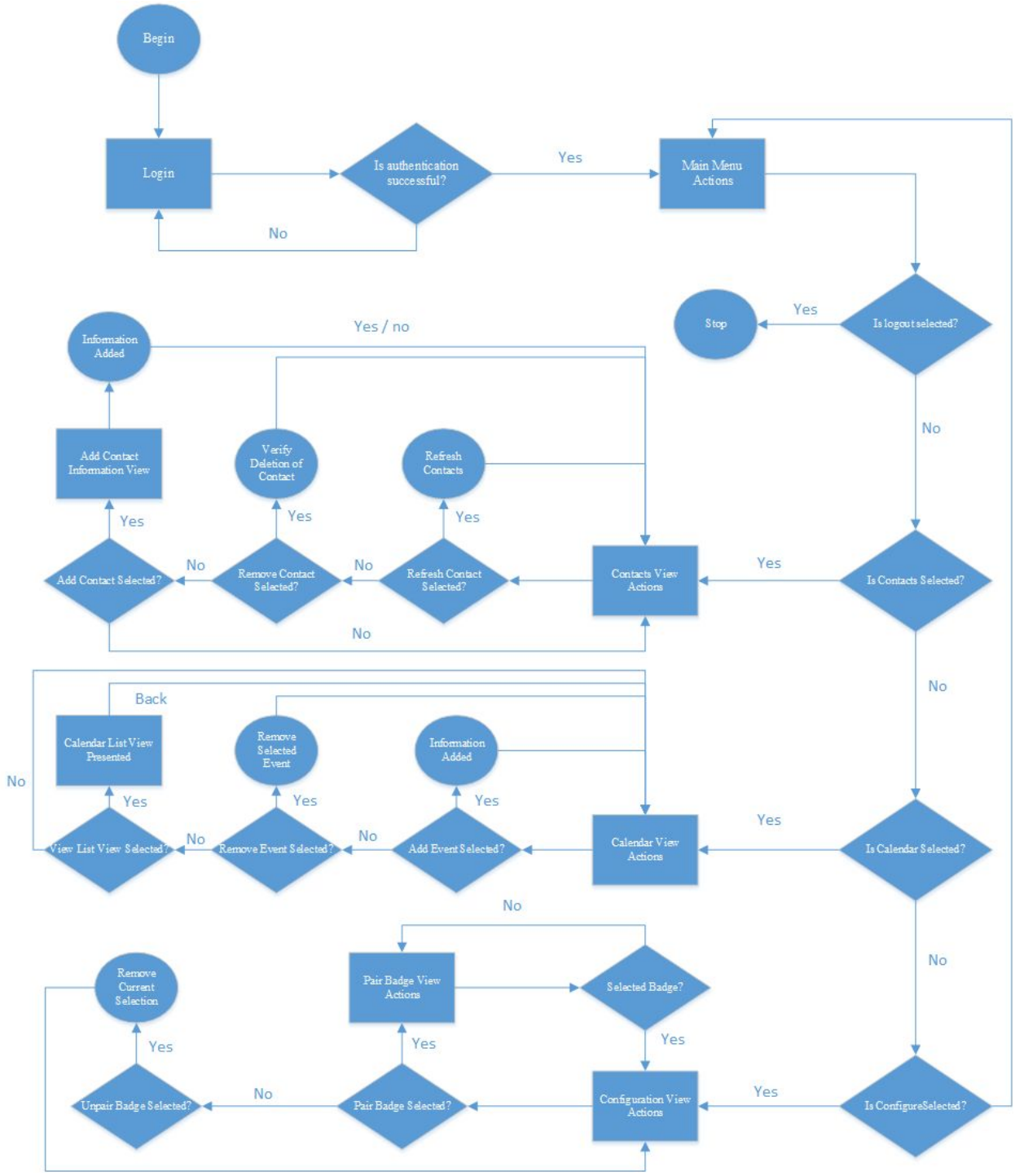


Figure 14: The mobile application flow chart diagram, which shows the features of the companion mobile application with respect to the TouchBadge web service as well as the TouchBadge hardware.

As figure 14 illustrates, the mobile application will have three main use cases. These use cases are the following: viewing contacts, viewing calendar, and configuring the badge.

Viewing Contacts
Viewing contacts in the mobile application will allow users to see all contacts they have made in one place. They will be able to sort the contacts by what events and seminars they met certain people at, as well as search for users by name or business. This will allow for a greater ease of use to the user.

Viewing Calendar
Viewing the calendar in the mobile application will allow users to view all events they are signed up for in one place. They can see past, present, and future events, and what seminar they are signed up for in each event. They will also be able to navigate to the contacts view by selecting an event and pressing “view shared contacts”.

Configuring Badge
The badge configuration view is an important part to the overall use of the mobile application. It will allow a user to sync their badge with their contact information in order to enable it to share contact information with others. Upon navigating to the view, the user will be prompted to login to LinkedIn and then pair the badge with their mobile device. Once paired, the device will automatically be sent the contact information from the mobile application and badge bumping will be enabled.

Notes
Depending on the length of implementation, Team 11 would like to also implement voucher views which would allow the user to access how many vouchers they have used during the conference, as well as maps. Maps in the mobile application would allow users to navigate the conference center and see where popular booths are located.

Initial Hardware Design

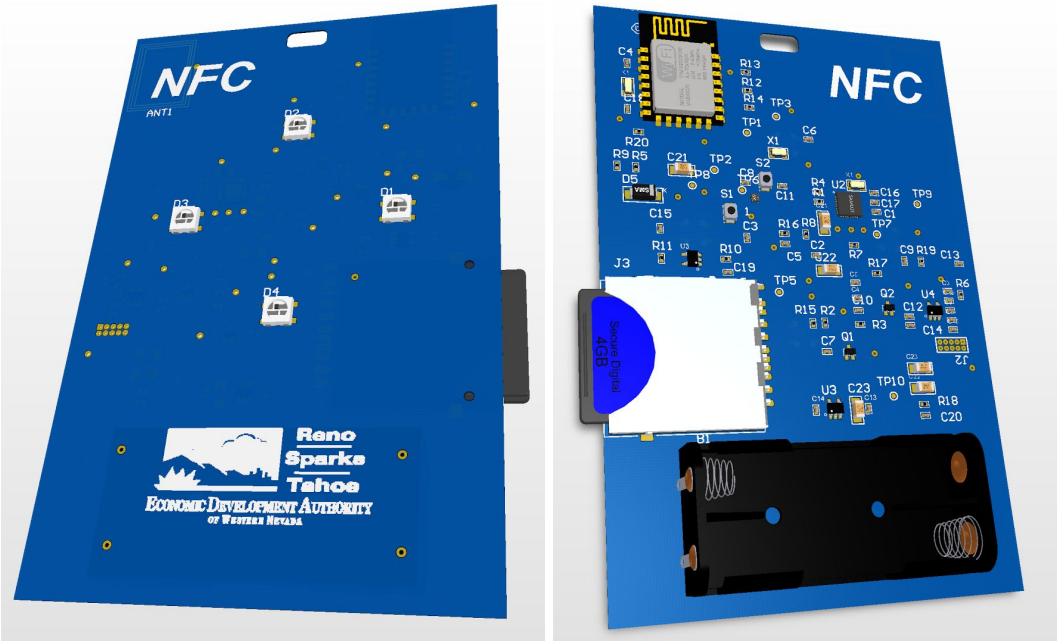


Figure 15: Initial hardware design provided by hardware partner Breadware, Inc.

The TouchBadge hardware is being designed by our hardware partner, Breadware, Inc. Their initial design images are shown in Figure 15. The hardware fits into a form factor similar to existing conference badges, with a footprint of about 3 inches by 5 inches. The TouchBadge hardware includes both NFC and Bluetooth connectivity for communication between badges and with a smartphone. To signal the user of bump and syncing activity, the hardware includes an array of multicolor LEDs on the front-facing side of the device. To power the TouchBadge, the hardware includes a battery holder designed to hold “AA” sized batteries.

During the most recent design sessions, this initial hardware design has changed. In Figure 15 a Wifi chip is included, but will likely not be included in the final hardware design. The team found that Wifi connectivity requires a larger power draw than they were comfortable with and the connectivity was redundant with the inclusion of a smartphone to the user workflow. Additionally, the team has not yet decided if the hardware will contain a charging system to support rechargeable lithium-ion batteries. Testing will have to be done to determine the total power draw of the badge over the course of a conference and determine if standard alkaline or rechargeable nickel metal hydride (NiMH) batteries will provide enough power.

User Interface Design

TouchBadge Login Screen

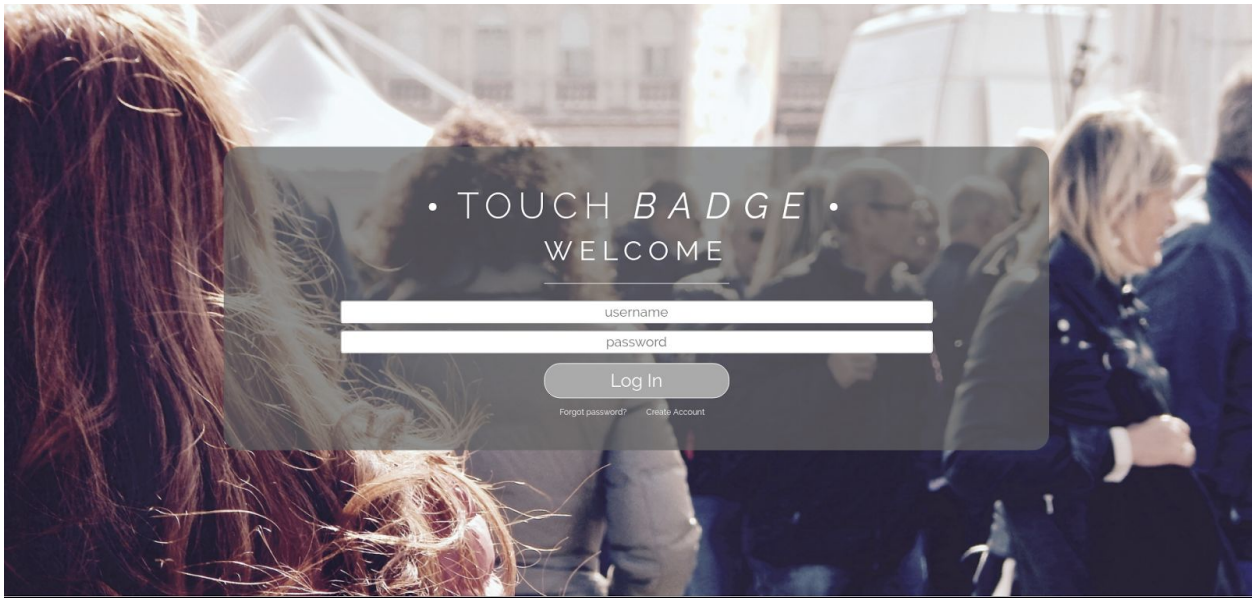


Figure 16: The website splash screen welcoming the user. This screen shall be presented to the user when they first hit the website. If they have already created an account they can log in here, if they have not they can simply click “Create Account” where they will be redirected to the creation page. The TouchBadge website will also be able to reset user credentials if they have been forgotten.

TouchBadge Dashboard

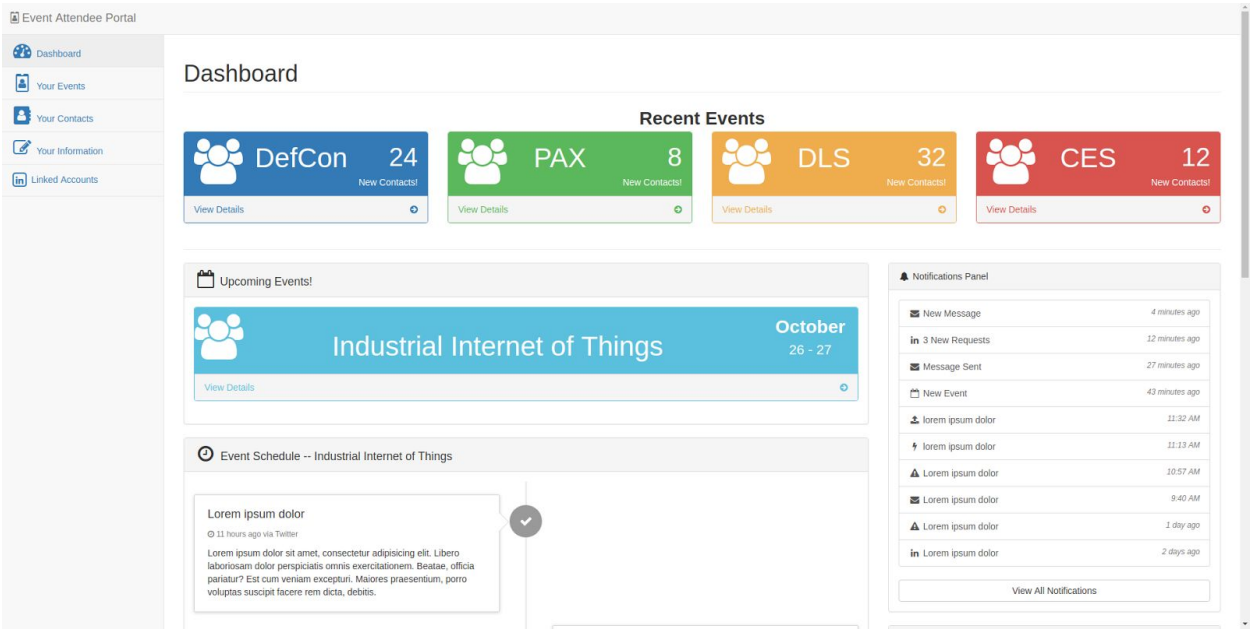


Figure 17: The main dashboard of the Touchbadge website. Here the user shall be able to have an "at a glance" overview of all past events, recent contacts, notifications, upcoming events, and more. It is designed to be a simple way for the user to interact with the site.

TouchBadge Mobile Application Home Screen

Figure 18: The TouchBadge companion app homescreen. Here the user shall be provided with an overview all everything happening. They can see their next event, if they have authenticated with a social network, and their contact information.

TouchBadge Mobile Application Contact Screen

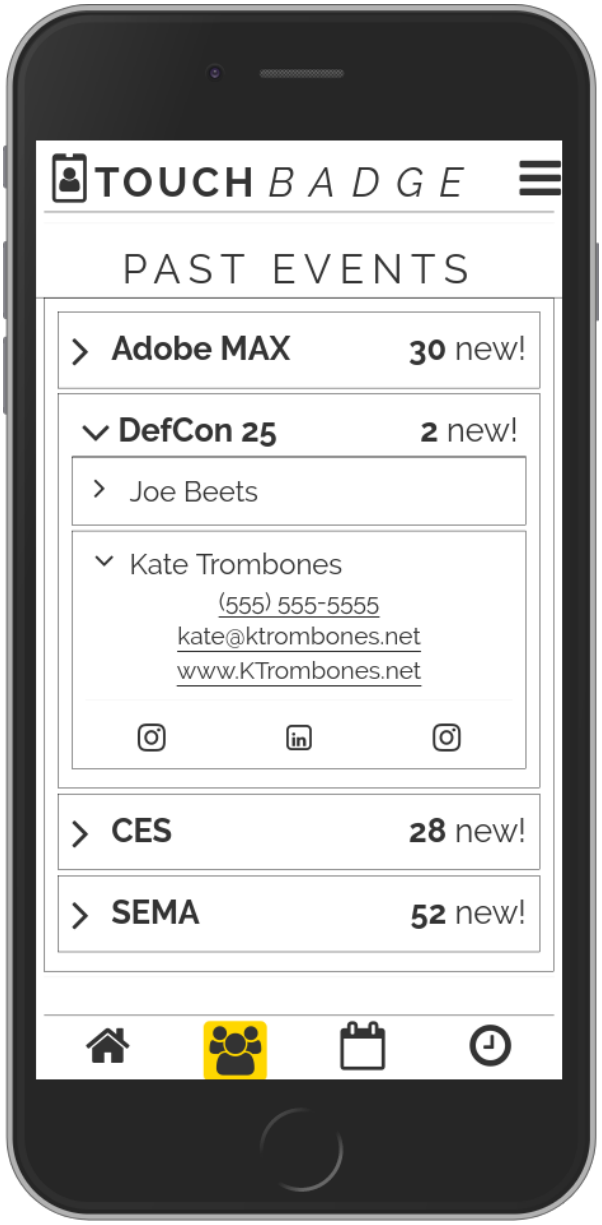


Figure 19: The TouchBadge companion app contact view. Here the user shall be able to see past events and the shared contacts they created at each event. On the contact themselves, they shall be able to view said contacts professional network, and essential contact information.

TouchBadge Mobile Application Configure Screen

Figure 20: The TouchBadge companion app configure view. Here, the user shall be able to easily pair with the badge hardware. The badge hardware will provide a key in the app to ensure they are connected to the right badge, and once connected shall be sent all the contact information of the user. Once done, the badge is ready to use.

TouchBadge Mobile Application Calendar Screen

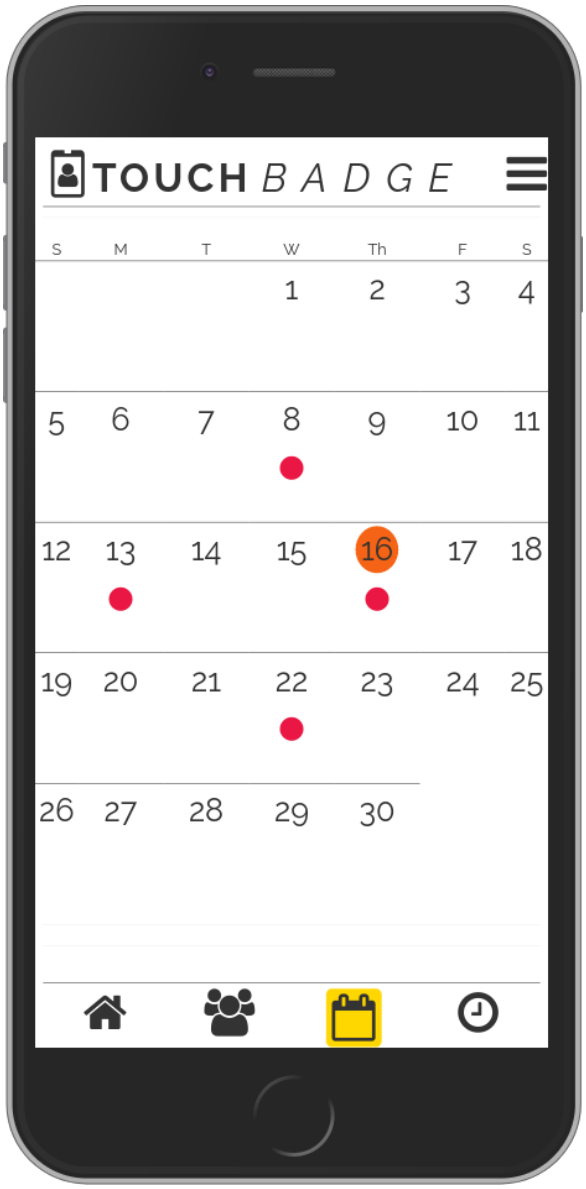


Figure 21: The TouchBadge companion app calendar view. The user shall be able to view past, present, and future event that they have signed up to attend. When the user selects the event, they will be able to see contacts shared, seminars, and room locations for seminars.

TouchBadge Mobile Application Calendar Detailed Screen

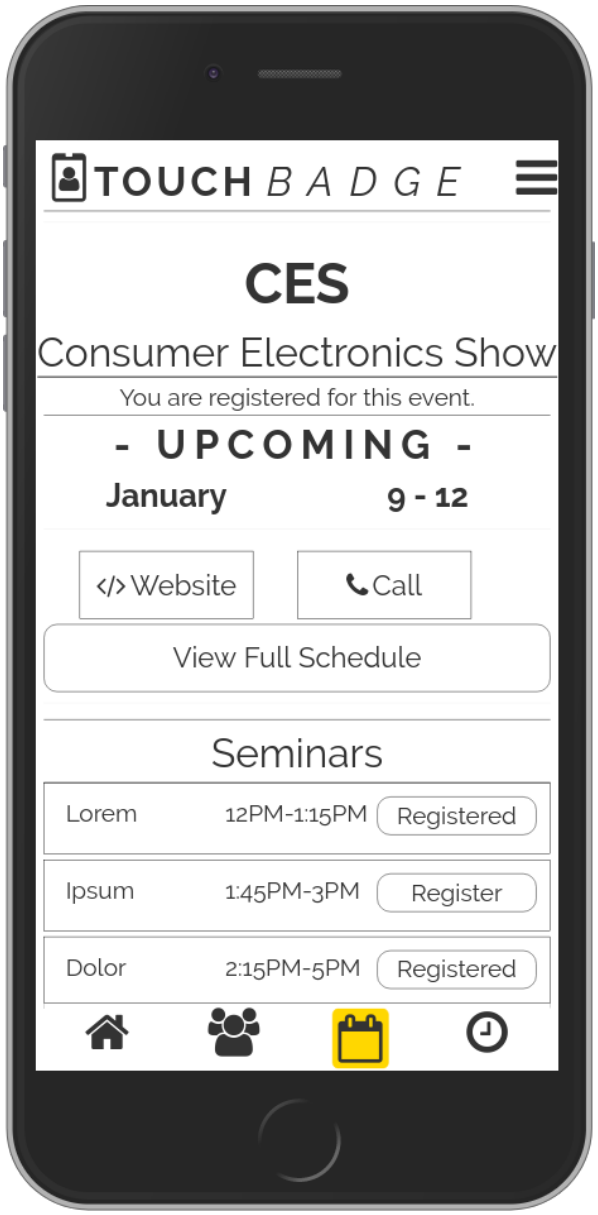


Figure 22: The TouchBadge companion app detailed calendar view. This subview is provided for the user to give more detail regarding a certain event. This highlights specific events, what they offer, and everything the user can experience while at the event.

TouchBadge Mobile Application Calendar Detailed Screen (2)

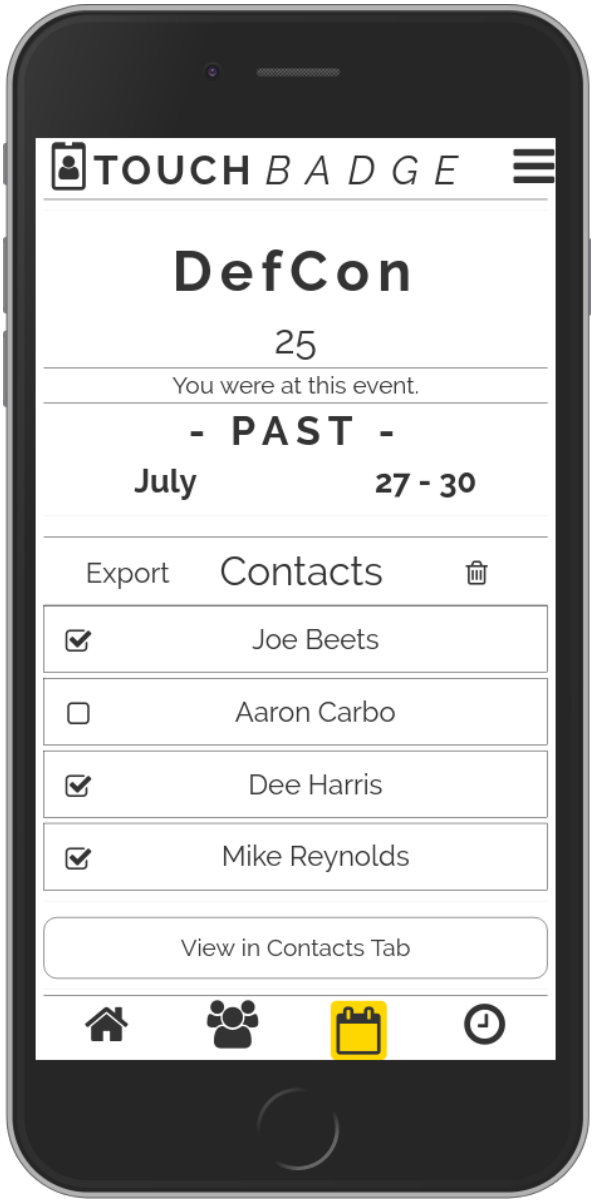


Figure 23: The TouchBadge companion app calendar detail view. This subview is provided for users to see shared contacts at events. The user shall be able to navigate here from the calendar view. This lets users see who they shared their contacts with and during what event.

TouchBadge Mobile Application Voucher Screen

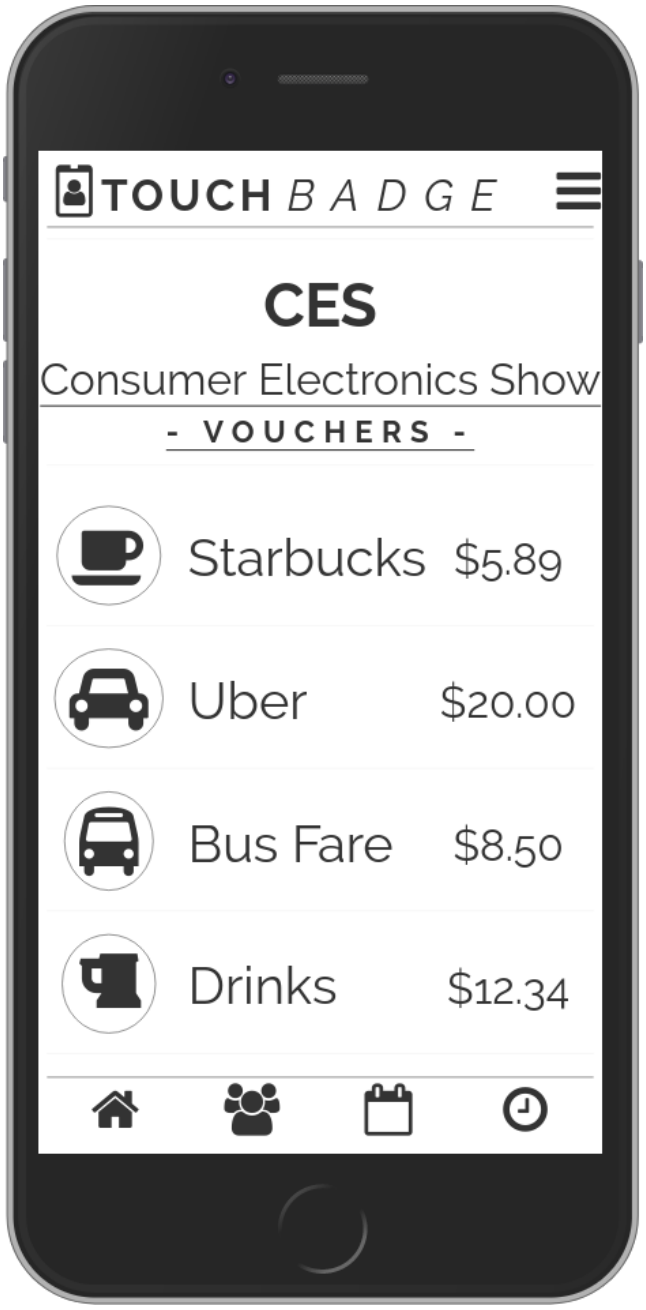


Figure 24: The TouchBadge voucher page. Here, the user will be able to view their vouchers provided to them by event managers for a specific event. The user can view their remaining credits for each voucher, and by tapping on a voucher, the user will be able to redeem credits for items.

Glossary

Application Programming Interface (API)
Externally accessible interface for a web service. Typically requires a "key" managed by the creators of the service to restrict access.

ASP.NET
an open source web framework for building modern web apps and services with .NET. ASP.NET creates websites based on HTML5, CSS, and JavaScript that are simple, fast, and can scale to millions of users.

Attendee Administration Portal (AAP)
The attendee-facing website that allows users to register, enter personal contact information, retrieve/export contacts shared with them, etc. Provides some troubleshooting options for their badge (initial registration, resynchronization).

Authentication Token
A unique string used to keep authentication data with the LinkedIn API. Used to re-authenticate with the api as to allow the user the ability to stay logged in.

AZURE
See "Microsoft Azure".

Badge
The lanyard device carried by each event attendee. Contains hardware necessary for interbadge communication, configuration, and export of data.

Bluetooth
A standard for high-bandwidth wireless connectivity. Typical communications range is up to 10 meters.

Bootstrap Framework
A CSS framework designed to easily create responsive, elegant front-end website solutions.

Bump
The process for interbadge communication. When two badges come into close proximity (1 cm) they transfer to each other the contact information for their wearer. The badges in turn will signal to the user that this interaction has occurred.

Conference Organizer Administration Portal (COAP)
The organizer-facing website that allows the event organizer to create and modify event details, see registered attendees, obtain demographic statistics on attendees, etc. Provides interface for event staff to register badges prior to the event or change out badges in the event of failure.

Conference
See "Event".

Contact
Personal information about an event attendee including name, address, phone number, email address, etc. Passed to another attendee or presenter through a bump interaction.

Customer Relationship Management (CRM) System
System designed to intake current and potential customer information and track relationships to improve sales. Leading implementations are SalesForce.com, SAP AG, Oracle, and Microsoft Dynamics. (Columbus)

Event
Occasion in which badge technology is used. Includes conventions, conferences, trade shows, etc. Used synonymously with "conference" in this document.

Extensible Application Markup Language (XAML)
XML based language, designed by Microsoft, used as a structured description for application views. XAML is a major part of Microsoft's .NET framework.

Lanyard
Neck-worn accessory that includes the strap and badge in combination.

Light Emitting Diode (LED)
Specialized diode design that emits light when powered. Available in single color or multicolor array form factors. Used to signal TouchBadge users of bumping and syncing activity.

Microsoft Azure
Microsoft's open, flexible, and reliable cloud computing software designed for the implementation of enterprise-grade solutions. In the context of TouchBadge, Microsoft's Azure platform will be used for web-hosting as well as the implementation of the backend database and web service.

Model View View Model (MVVM)
a software architectural pattern. MVVM facilitates a separation of development of the graphical user interface – be it via a markup language or GUI code – from development of the business logic or back-end logic (the data model).

Near-Field Communications (NFC)
A standard for short-range communication between devices. Allows for powered two-way communication or low-power single direction communication.

Secure Sockets Layer (SSL)
The standard security protocol for establishing secure links between backend web servers and client-side web browsers.

Structured Query Language (SQL)
a domain-specific language used in programming and designed for managing data held in a relational database management system

Synchronization (syncing)
Process for communication between badge and smartphone. Utilizes Bluetooth communication protocol to transfer obtained contacts to smartphone for transfer to web service.

TouchBadge
The title for the complete smart badge solution. Includes badges, websites, and app.

Voucher
An offer provided by the event organizer in collaboration with a third-party that is accessible through the badge. Examples include free or discounted food/drink or transportation.

Xamarin
A cross-platform development framework that will allow for robust, native mobile applications made for IOS and Android. If desired, Team 11 could also used this framework to develop for Windows phones as well.

Contributions of Team Members

Evan Grill

Over the course of the project design, Evan worked on the following items:

- Abstract - **0.5 hours**
- Introduction - **1.5 hours**
- Glossary - **1 hours**
- Editing and Revision - **2 hours**
- Initial Hardware Design **- 1 hours**

Jon Weatherspoon

Over the course of the project design, Jon worked on the following items:

- High-level and Medium-level Design **- 2 hours**
- User interface design **- 6 hours**
- Editing and Revision - **1 hour**

Cody Worsnop

Over the course of the project design, Cody worked on the following items:

- Coverpage **- 0.5 hours.**
- Table of Contents **- 0.5 hours.**
- High-Level and Medium-Level Design **- 5 hours**
- Detailed Design **- 1.5 hours**
- Glossary **- 0.5 hours**