CS 486C - Team Badgers
Software Design Document

Northern Arizona University

February 15, 2021

Sponsor: Dave Hopfensperger & Glenn Austin

Team Mentor: Sambashiva Kethireddy

Members:

Abdulrahman Alamoudi
Tristan Marcus
Logan Ritter
Robel Tegegne
Yuanbo Xu

Overview:

The purpose of this document is to provide a blueprint for our product in order to avoid major issues during the development process. In this document we go over our Implementation and Architectural overviews as well as module and interface designs and our implementation plan.

Table of Contents

1. **Introduction** 3

2. **Implementation Overview** 5

   2.1. **Technologies** 5

   2.2. **Main Components** 6

3. **Architectural Overview** 8

4. **Module and Interface Descriptions** 12

   4.1. **Badge Delivery System** 13

   4.2. **Kudos Social Currency** 14

   4.3. **Badge Printing** 16

   4.4. **Email Signature Generator** 17

   4.5. **User Authentication/Log in System** 18

5. **Implementation Plan** 20

6. **Conclusion** 22

1. Introduction

Since the beginning of March 2020, the world has been dealing with the SARS-Cov-2 pandemic, better known as Covid-19 or Coronavirus. The virus has affected many people's lives causing a massive change in work environments. Most people, even those who have not caught the disease themselves, have been impacted. Millions of people have lost their jobs due to the inability to attend work because their business is not considered essential and the state government is requiring the business to be closed. There are not a lot of places that are capable of migrating their employees to a virtual workspace, but even those who can are stumbling upon a new set of issues. The majority of companies that were able to operate virtually are finding that many employees are struggling with motivation, recognition, and many other things that they use to find daily in the office. Since virtual employees are now abruptly working on a laptop in their home “office”, they lack the social interaction that was constantly present in the workplace of old. In most cases productivity is down due to the lack of oversight, depression is up due to the lack of social interaction, and motivation is down due to the lack of recognition.

Our sponsor for this project is State Farm and more specifically Dave Hopfensperger and Glenn Austin who represent the Enterprise Technology Department of State Farm. In Enterprise Technology there are hundreds of teams and around 6,000 employees that work on a wide variety of things to keep the company running smoothly while also keeping State Farm up to date in this ever growing world of technology. State Farm has always been a great place to work for employees primarily because of the respect and recognition they show their employees. In the office there was constant praise and recognition for everyone who produced quality work State Farm is known for. This came in the form of verbal praise, buying your coworker a coffee or

donuts, and even sometimes ribbons to be displayed on your desk. Now since the majority of Enterprise Technology is working at home virtually due to the Covid19 Pandemic, it is not possible to award an employee with anything more than an email, or praise on a skype call. Mr. Hopfensperger and Mr. Austin has proposed that this product can solve a lot of the social problems presented by the virtual workspace nature.

The Product that our clients have envisioned is a single page application that can provide and perform multiple tasks that works like a social media platform. The main functional requirement that our clients wish to see on the product is a badging system where employees can earn and send badges based on milestones and skills they achieve with the company. They would also like us to have some form of social currency which can be dispersed for smaller achievements which then can be redeemed for rewards set by managers. There are many other functionalities we wish to implement to enhance the user experience and create a well rounded social media platform. We are fortunate that there were no environmental restrictions set on our team giving us developmental freedom.

2. Implementation Overview

The product that our clients with State Farm have tasked us to build is a single page web based internal social media platform that can award their employees with badges and rewards for continuing to provide quality work from their offices, into this new online nature. The team has decided that the product would excel best as a single page application that has a multitude of pop up models to perform all the tasks within the application. We would like our product to look similar to Facebook with a waterfall feed down the middle of the application, and functions to the left and right of the feed, so that it follows familiar design patterns. Building our product this way will provide simplicity and eliminate confusing navigation which will cut the learning curve of the application tremendously.

2.1 Technologies

In order to implement the envisioned product the team has selected a few softwares and tools based on their familiarity as well as how helpful they will be for our product.

- ReactJS - Starting with the front-end UI we have decided to use ReactJs as it provides an easy to use front end framework. With ReactJs we can incorporate HTML CSS and JavaScript without any hassle.
- MongoDB - For our database we have decided to use MongoDB because of its ease of use and the team's familiarity with it. MongoDB will store and update all the data from every user and will be readily available for our front-end to use and manipulate.

- NodeJS (Express) - In order to have communication between our front-end and our database we have chosen NodeJS with the express framework as our back-end language, to perform all our functionality and data updating.
- Gimp - Lastly for unique website and badge design our team has decided to use Gimp, which is a graphic design application. Using Gimp we will be able to create customized badges and unique images for the application.

With these technologies we will be able to implement our key features successfully.

2.2 Main Components

While the team hopes to implement as many features and functions as possible in the time we have available, to make this product usable and functional our team has decided on five main features that the product could not go without. These features include:

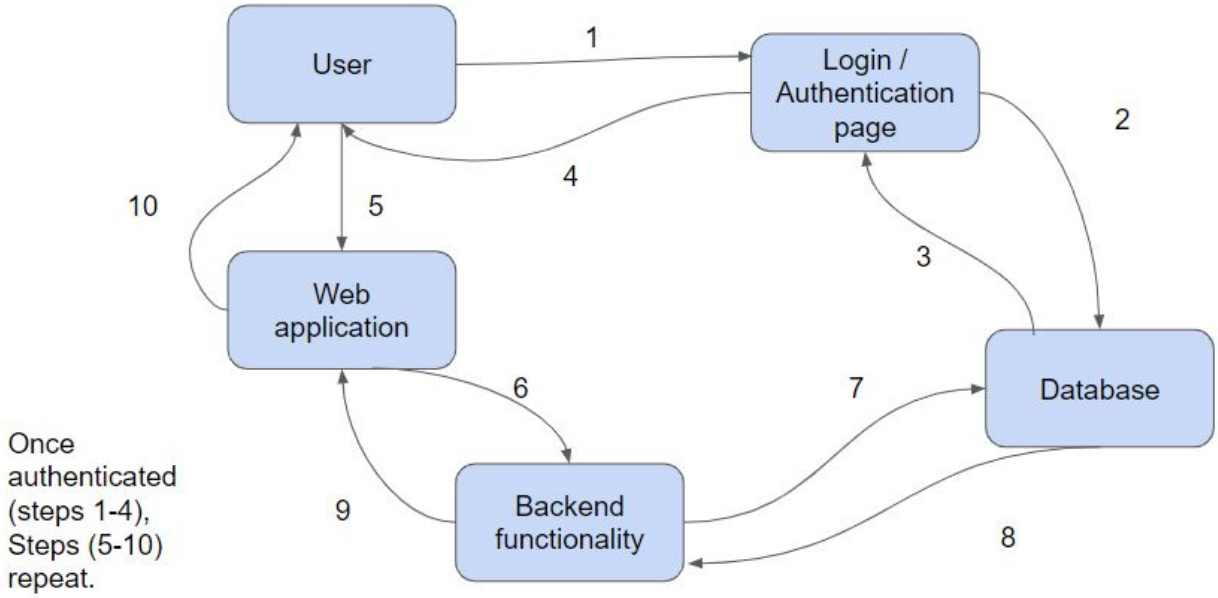
- Badge Delivery System - This is the core feature of the product. Our badge delivery system will allow users to send, receive and earn badges for various milestones and accomplishments that can be achieved while working at State Farm. Badges are meant for larger milestones.
- Kudos Social Currency - We will have an internal social currency called Kudos that will allow employees to send their coworkers Kudos to be redeemed for rewards set by their managers. Kudos are meant for the day to day accomplishment.
- Badge Printing - We would like for every user to be able to download and print any of the badges that they have earned. This will allow a user to show off and be proud of their accomplishments.

- Email Signature Generator - Users will be able to generate a fully customizable email signature incorporating their earned badges and Kudos into the signature. Doing this will allow them to show off their accomplishments to their fellow employees.
- User Authentication/ Log in system - Every user will be able to create their own secure account. This will help provide security to the application.

Using these technologies to implement our five main features we will be able to develop a unique and interactive social media application for State Farm.

3. Architectural Overview

Alongside the technologies used, we will need a strong architectural plan to reduce road bumps in the future and to get a good grasp on how our product will work computationally. Each of our components for this product will need to communicate with one another for this to be successful.



(**Figure 1 -** Basic Communication flow between each of our components)

In Figure 1, we can see how a user would interact with our product and the order in which each component will be used. The flow in which our users would interact with our product would be as follows (Each step number is associated with the number in Figure 1):

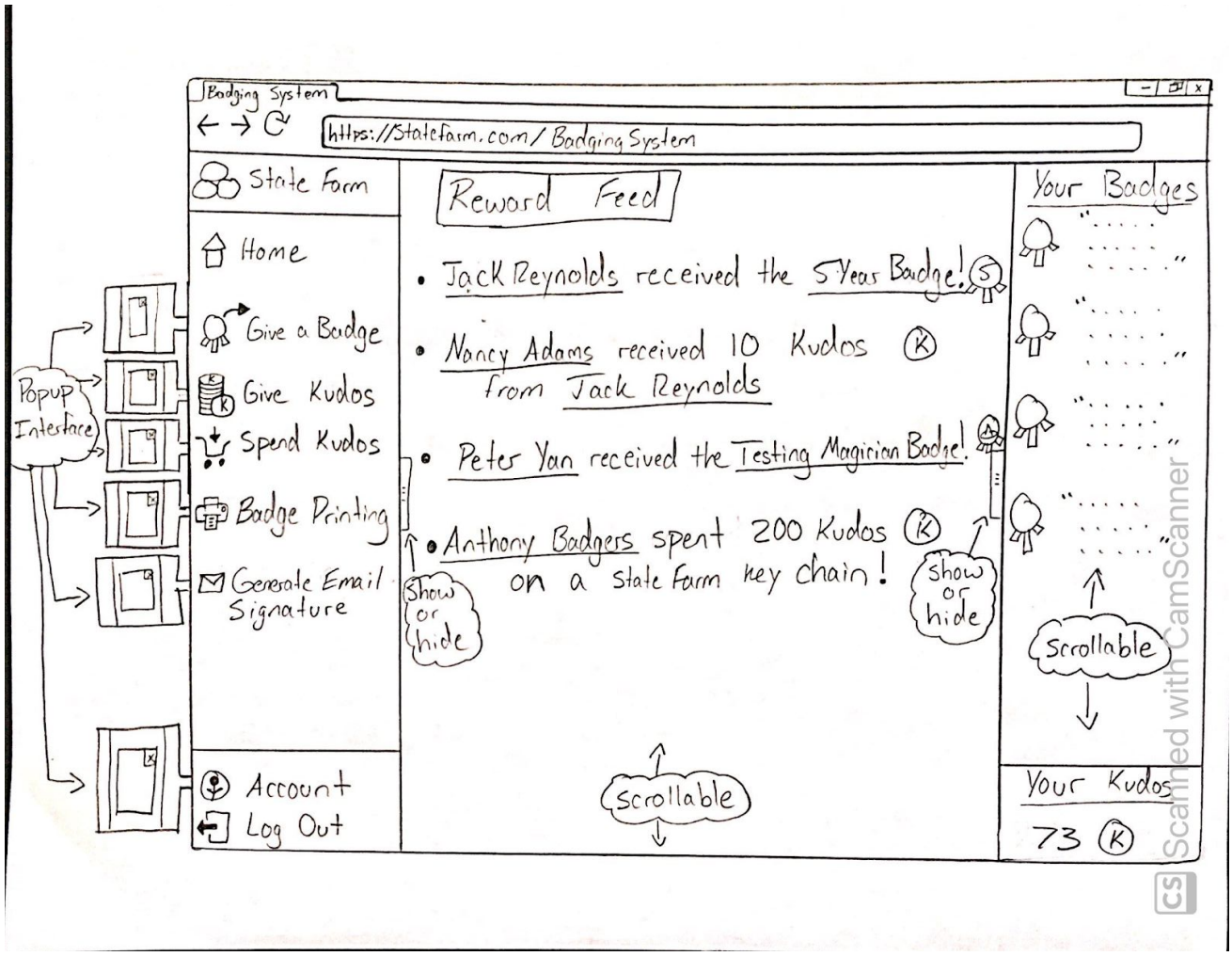
1. A user opens their browser and goes to our web application and is greeted by our login page.
2. Given the user already has an account, the login page will take the information provided by the user, and verify with the database that the user has an account and is authenticated

3. The database will confirm back to the login page that the user is authenticated
4. Then the user will be sent to the web applications homepage.
5. Once in the homepage the user performs one of the various functions.
6. The web application calls the backend functions associated with the front-end function performed by the user.
7. The back-end the performs the function, then updates/pulls from the database
8. The database sends confirmation to the backend
9. The back-end then updates the front end with what it needs to show the user
10. Finally the front end provides the user with updated information, or with the product of the function they chose to perform. (ie. a confirmation that a badge was sent, the reward they redeemed with kudos, etc.)

While it is vital that all of our components are communicating properly, each of the components has its own responsibilities that it needs to perform. For each of our components we will discuss the main responsibilities and features.

- Login web page - This web page is separate from our application and its main responsibility is to provide extra security and force users to make an account with their State Farm/Google email. A key feature for this page is that it will use Google authentication after an account is created. The login page will communicate with the database to assure a user is authenticated and the right information is sent to the homepage when the user logs in.
- Web Application - This is the front end single page application that the user will interact with. Its responsibility is to provide an easy to use UI and communicate with all the

back-end functions and display information to the user. All the five main features of our application will be accessible for use. Since this is a single page application, each of our functionalities will come up in pop up interfaces, or modals, for our user to interact with. Below is a sketch of our envisioned UI (Figure 2).



(Figure 2 - Prototype sketch of the front-end of our application)

- Back-end - The back-end will be responsible for communication between the database and the front end. It will also perform all the functionality of our application. This will be

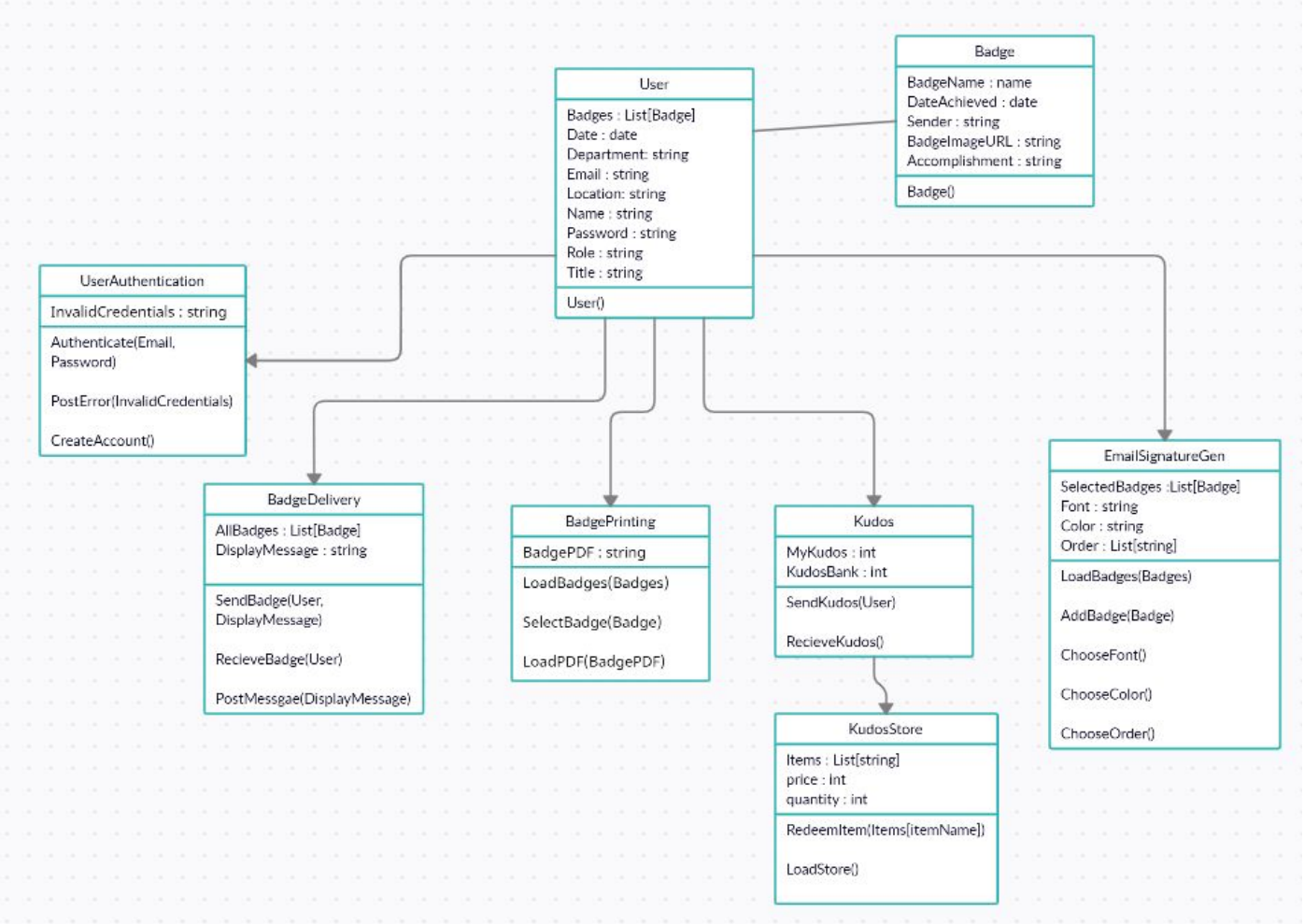
the backbone of our product.

- Database - The database is used to store all users information and progress. The database will provide and update information as needed. Communication will occur with the back-end which will push the data to the correct places on the front end.

With the given architectural plan we will be able to build the product as intended avoiding major setbacks and speed bumps that could appear in development.

4. Module and Interface Descriptions

As mentioned before our product will consist of five major components that will provide all the functionality of our application. In this section we will describe the responsibilities and service that each of our components provides as well as how they fit into the architecture. Those components are the badge delivery system, the Kudos system, badge printing, email signature integration, and user authentication. We will also be implementing a Badge object that provides the structure for each of our badges. We have created a UML diagram of our components in **Figure 3** below.

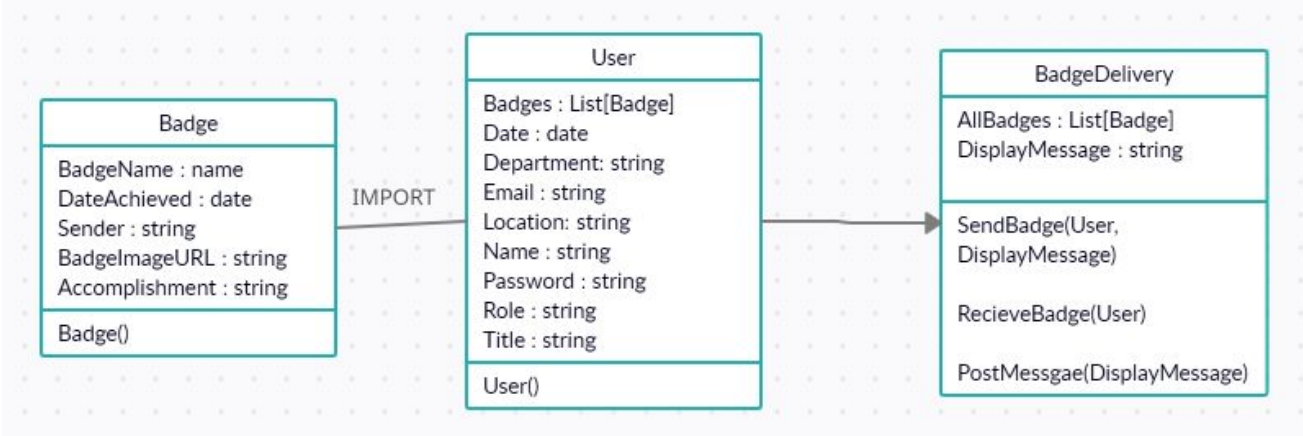


(**Figure 3 -** UML diagram of our system components)

The main model that each of our components extends is the User model. This model holds all the information from the user such as their name, email, title, location, and more. This information is set when the user creates an account. The User also imports the Badge object and stores a list of all the badges the user has earned. The badge object holds information such as the badge name, the accomplishment achieved, the date achieved, and the badge image URL.

4.1 Badge Deliver System

Our badge delivery system will be the main component of our application. The system will be an extension of our User which imports the Badge object. Its main job is to provide badges to the user when they earn them. It is also responsible for displaying the badges earned by users in the waterfall feed. A user will be able to see their badges on the front-end on the right panel of the application, as well as the badges they can still earn. Badges can also be sent to other users depending on the badge. This part of our application will be hosted on the front-end but all the functionality will be done on the back-end which will use the database. Below is the UML diagram of our badge delivery system (**Figure 4**).

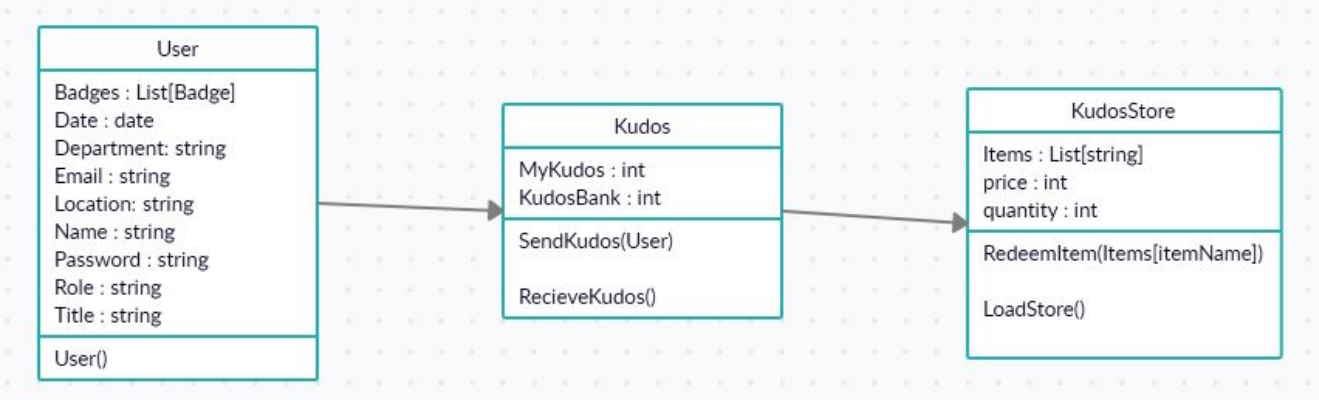


(**Figure 4 -** UML diagram of the badge delivery system)

The badge delivery system inherits all the attributes from User. Additionally it has a list of all the badges available and a message variable that constantly changes to the most recent message that came with the last badge sent. Some of the functions our system can perform are SendBadge(User, DisplayMessage), ReceiveBadge(), and PostMessage(). SendBadge(User, DisplayMessage) will take in the user you wish to send a badge to and the message you want to send with it. It will then call the ReceiveBadge(Badge) function on the receiver's side which will handle the updating of the receiver's badge list. Lastly it will call the PostMessage(DisplayMessage) function to add the message to the waterfall feed which all users can see on the homepage of the application.

4.2 Kudos Social Currency

The next component of our application is the Kudos system. This will be a second way for users to earn rewards and recognition from our application. This will work differently from the badges in the sense that Kudos will be given out on the daily by coworkers for many different reasons. Kudos are meant to be sent for smaller accomplishments where the badges are sent for larger milestones. Our Kudos system will be responsible for the sending and receiving of Kudos to every user. It will also be responsible for the Kudos store where managers can set rewards and employees can redeem them. Like the badge delivery system, the Kudos system will be hosted on the front end of the application with all the functionality being done in the back-end with help from the database. Similar to the rest of the components the Kudos system extends the User model. Below is the UML diagram of our Kudos system (**Figure 5**).



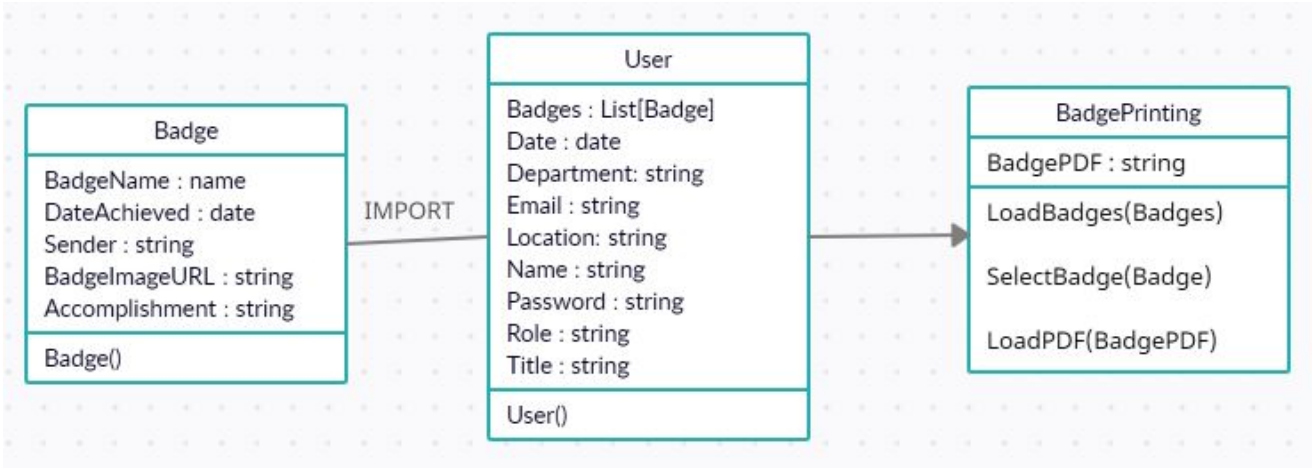
(**Figure 5 -** UML diagram of the Kudos system)

Starting with the Kudos themselves, there will be two different integers kept. The first is called MyKudos. MyKudos is for the users Kudos and will be where Kudos sent to them will go as well as the Kudos they will use to redeem rewards. The second is the KudosBank which is the kudos available to send to coworkers. This amount will be set at a certain number depending on the manager and will be renewed weekly. The only functions of the base Kudos class are SendKudos(User) which sends Kudos to the provided user and ReceiveKudos() which updates the receiver's MyKudos. For example, UserA has 1000 Kudos in their KudosBank and they use SendKudos(UserB) to send Kudos to UserB. Let's say they chose to send 100 Kudos, UserA's KudosBank will now have a balance of 900 and UserB's MyKudos will have a balance of 100, given they have not received Kudos before

Next we have the KudosStore which has a list of all the items that were set by the manager, the quantity of each item, and the cost of each item. The only functions that an employee can use from the KudosStore are LoadStore() which happens automatically when the page is loaded, and RedeemItem(items[itemname]) which redeems the provided item given the user has enough Kudos. An important detail is that each user's store will be different depending on what their team's manager sets for rewards.

4.3 Badge Printing

The next component of our application is the badge printing model. The goal of this component is to allow the user to access the PDF of every badge they have earned so they can download, print and display their badges in the real world. When a user chooses the badge printing model on our application the first thing that will happen is a popup with all their earned badges will appear. They will then select a badge then select the confirmation box. It will then open up the badge PDF in a new window for the user to do with it as they please. This component will be hosted on the front-end and will use the back-end to parse the database for the selected badges. Below is the UML diagram for our badge printing component (**Figure 6**).



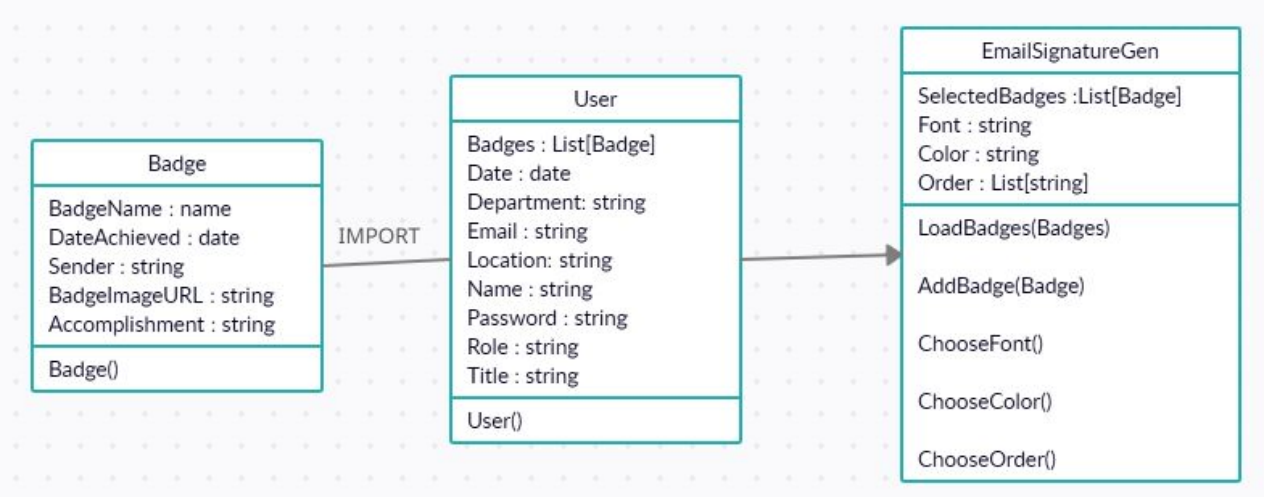
(**Figure 6 -** UML diagram of the badge printing system)

Like the other components the BadgePrinting component extends the User model. The only unique attribute the BadgePrinting model has is called BadgePDF which is a string containing the URL link to the selected badge's PDF. The functions provided by this model are LoadBadges(Badges) which displays all the users earned badges, SelectBadge(Badge) which

grabs the users selected badge and sets the correct URL in the BadgePDF attribute, and lastly the LoadPDF(BadgePDF) which brings the user to the provided PDF link.

4.4 Email Signature Generator

Another component of our application is the email signature generator which allows users to generate a customized email signature using their earned badges. This will provide another way for the users to show off their accomplishments. Unlike the other components, the majority of the email signature generator functionality will take place on the front end. The only back-end use will be the loading of the badges to be selected for the signature. We want to give our users customizability by allowing them to choose the font, color, and order in which they would like the parts of their signature to be displayed. Once a signature has been generated it will be available to copy as a whole and saved by the user. Below is the UML diagram of our email signature generator component (**Figure 7**).

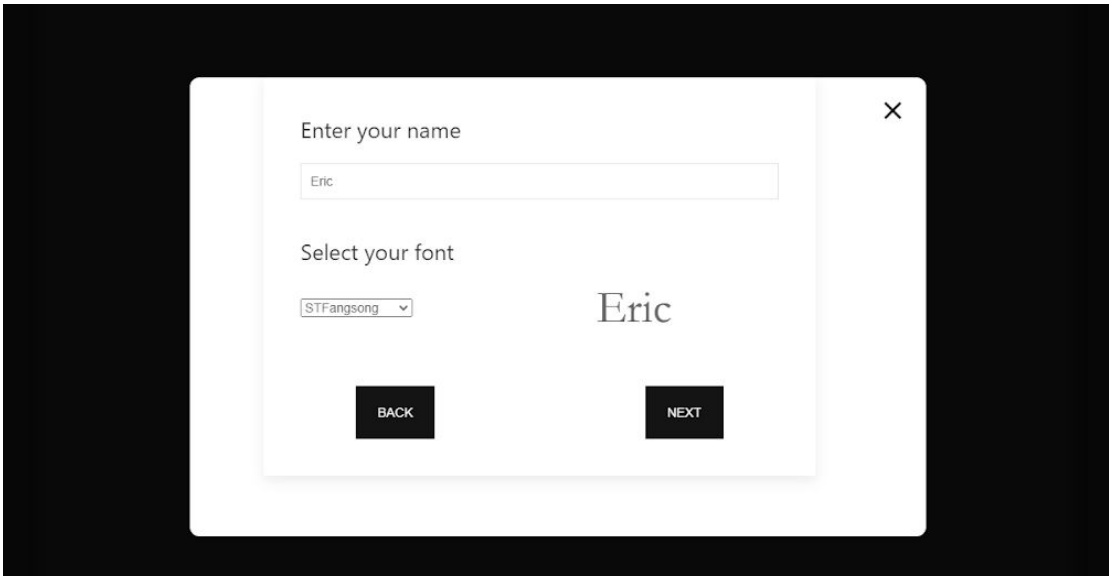


(**Figure 7 -** UML diagram of the email signature generator)

The EmailSignatureGen model extends the User model. It additionally provides four more attributes, SelectedBadges which is a list of the badges selected by the user, font, color, and

string. When a user chooses to generate an email signature the following functions will be executed. First LoadBadges(Badges) will load all the badges from the user to be selected. The user can then choose up to 3 badges and AddBadge(Badge) will add the selected badge to the SelectedBadges list. Finally the user will use the ChooseFont(), ChooseColor() and ChooseOrder(), to customize the signature. Now the signature would be ready to save and copy.

Below is an example of what our badge printing system looks like currently (**Figure 8**).

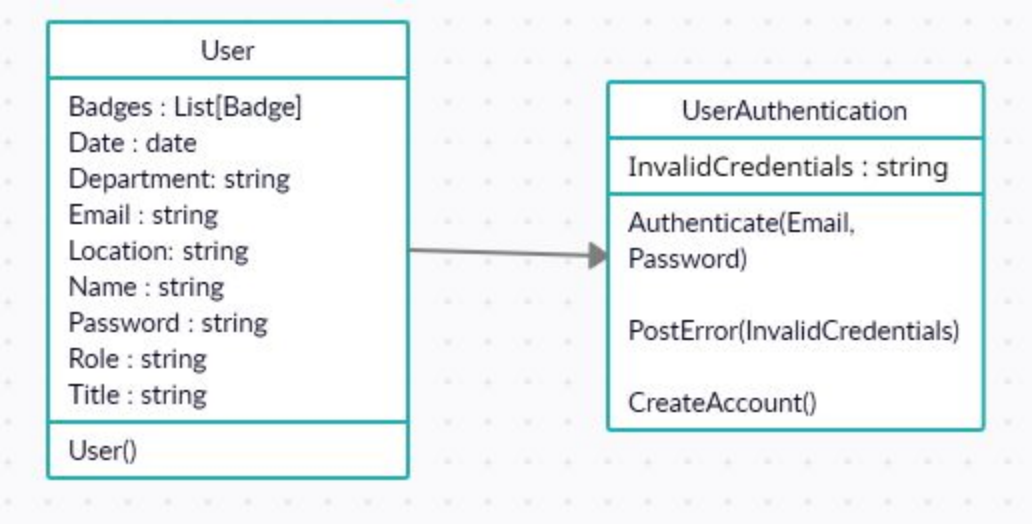


(**Figure 8 -** Badge Printing PopUp example)

4.5 User Authentication/Log in system

The last main component of our application is the user authentication system. The sole responsibility of this component is to allow users to create accounts and provide an extra layer of security by forcing users to log in to their accounts every time the application is opened. This is the only component that will be separate from the single page application as we do not want to give someone access to the application if they have not been authenticated. This component will

be hosted on the frontend and will speak directly to the database to authenticate users. Below is the UML diagram for our user authentication system (**Figure 9**).



(**Figure 9 -** UML diagram of the user authentication system)

The UserAuthentication model extends the User model. The only attributes used from the User model by the UserAuthentication model are the Email and Password attributes given the user already has an account. Once a user enters in their information the Authenticate(Email, Password) function will be used. If they already have an account and they typed in their information correctly they will be directed to the application homepage. If they enter in the wrong information the PostError(InvalidCredentials) function will be called and the predetermined message, InvalidCredentials, will be displayed to the user. Lastly if the user is new and needs to create an account, the CreateAccount() function will be used which will allow a user to enter in their information and a new User will be created. For more security we will be using a token authenticator. For example when a user creates a new account a random unique token will be generated for their account and every time they log in, the token will be compared to the one in the database to authenticate them.

5. Implementation Plan

To ensure our application is developed correctly and in the time we are given, the team has come up with a plan to ensure smooth implementation. The team has divided implementation into eight parts which we will implement in order. The reason we are choosing to do this is for a few reasons. The main reason being that we need to implement foundation, security and the more important aspects of our application first so that further development will be easier and we will have a better understanding of our own system. We also want to make sure that we are implementing the key aspects of our product before we spend time adding features that we do not necessarily need, and the reason for this way of development is the amount of time we have in the semester. The eight parts of implementation are listed in order of importance and in the order in which we will develop below.

1. Foundation and setup - This includes developing the base of our application both on the front-end and the back-end, as well ensuring that everything works properly before development.
2. Connecting components - Next we will Link all of our components together, such as the front-end to back-end and back-end to database.
3. Security and User Authentication - Before further development we need to make sure we implement the authentication system of our application.
4. Badge Delivery System - Now that we have implemented a secure foundation and we have a fully functional application we can begin development on the most important feature of the application.

5. Kudos System - The implementation of the Kudos system will run parallel to the Badge Delivery but with a lower priority.
6. Email Signature Generator - After we have implemented the Badge Delivery System and the Kudos System we will begin implementing the email signature generator.
7. Badge Printing - Lastly we will implement badge printing as our last main component.
8. Enhancement - Given we finish all previous steps before our semester is over the team will work to enhance all current features as well as add further features as time allows.

Below is the Gnatt chart which shows the envisioned times at which we plan to have each step of our plan completed.

(**Figure 10 -** Implementation plan Gnatt chart)

6. Conclusion

Many companies, since the beginning of March, were forced to switch to a virtual workspace due to the covid-19 pandemic. With this new workspace a new set of problems have been introduced. Those include a lack of recognition and thus their enthusiasm and incentive at work drops sharply. Although some companies have tried mechanisms to provide the missing acknowledgement and feedback, they were only partially effective. State Farm on the other hand has challenged our team to build a gamified web based application to award and incentivise their employees in the form of badges and Kudos.

In this document we went over the plans to our solution to tackle this project and develop a highly functional social media web application. First we went over our implementation overview where we introduced our five main components of our application. They are the badge delivery system, the kudos social currency system, the email signature generator, badge printing, and user authentication. Then we discussed our architectural overview where we introduced our technologies and how they interact with one another. Next we went into detail with our five components. For each component we went over the responsibilities and services the component provides as well as how it uses our technologies. Lastly we went over our team's plan for implementation for the semester.

At the moment the team is currently finishing up work on the security and user authentication of our application. We have already finished the foundation and setup, as well as the integration of components. Shortly we will begin developing the Badge Delivery System as well as the Kudos system.

This document will help guide the team to successfully creating this product. The document will serve as a blueprint for which the team can use throughout the development process. Also this document has helped plan ahead to prevent any major setbacks that could occur due to a lack of planning.