June 2021

Wedoo: An Event Planning App

Final Report

AnneMarie O'Loughlin 20086619
Higher Diploma in Science in Computer Science
Supervisor: Mary Lyng

Contents

Declaration

I declare that the work which follows is my own, and that any quotations from any sources (e.g., books, journals, the internet) are clearly identified as such by the use of 'single quotation marks', for shorter excerpt and identified italics for longer quotations. All quotations and paraphrases are accompanied by (date, author) in the text and a fuller citation is the bibliography. I have not submitted the work represented in this report in any other course of study leading to an academic award.

Student…… Date …..22/05/2021……..

1 INTRODUCTION

1.1 BACKGROUND

Having a finance background, I wanted to create an app which could do basic financial planning and process payments. I wanted to make something which they average person could use to help them budget for their life events. I decided to make an event planning app, which will be targeted at wedding planning.

My app will hopefully make the event planning process much easier for both the hosts and the guests attending. I wanted to take make a clean and easy-to-use application which will take a lot of the work out of planning events.

My app is similar to “The Knot”. My app has some extra guestlist functionality that The Knot does not have. Currently couples using The Knot must have all the guests’ information before sending the invites. Wedoo removes the step of contacting each guest one by one to gather information, it allows the user to select guests and email an information request where guests can enter their own information.

The Knot has a lot of menus and different functionality. This makes it a lot more difficult to navigate. I wanted my app to focus on user experience. I made Wedoo easy to navigate by limiting the amount of clutter on each page and using accordions to only show the information when selected. My app also allows the couple to assign table numbers to the guests. The wedding couple can limit the number of guests to a table and filter by table number.

Although my app will be targeted at weddings, users can use Wedoo to plan any event; big or small. Users can choose what part of the functionality they want to use and what they want to exclude.

1.2 PROJECT AIMS

I hope to achieve a fully functioning guestlist management system. Hosts planning an event can write the guestlist, confirm addresses for the invitations, track RSVPs and finally prepare a table plan. Hosts can upload an Excel version of the guestlist to the app. They can choose to add an email for each guest, this will allow the hosts to send Save the Date emails and request the guest add their home address. There will be a function where you can send an email requesting electronic RSVP through the app. Each guest will have their own unique link to the app. Hosts can choose to present further information on this page.

I would like to add functionality to the app to allow guests to send a monetary gift electronically. Currently, during weddings, guests give money in a card which is not very secure as a card could easily be misplaced or stolen. An electronic solution which would allow the couple to receive the money directly into their account would be much safer. I will use PayPal Sandbox system to implement a payment option for the guests.

I would also like to add a budget management function which would allow hosts to estimate the costs of each component of the event and manage their funds better.

The final part of my application will be a to-do list for the hosts to keep everything on track. Hosts can be assigned tasks and set the status for each task.

2 PROJECT MANAGEMENT

2.1 METHODOLOGY

I have decided to use Agile methodology for my project. From my experience Agile is widely used in tech companies. I like the approach of planning my project in short sprints to keep me on schedule.

2.2 PROJECT SCHEDULE

2.2.1 SPRINT 1, JANUARY 2021:

- **Research payment processing applications**

  My main focus for this project was to create an application which could accept payments easily and securely. I wanted to use a reputable payment processing system as I know people would feel uncomfortable entering their credit card information into an unknown application. I chose PayPal as it is a well-known brand which is trusted by many users. This gives the users both the option to log into their existing PayPal account to complete the payment and the option to enter their credit card details.

- **Create a basic application which will be linked to the payment application.**

  Using PayPal Developer sample code and my Sandbox credentials, I created a basic application with built in PayPal payment buttons and input fields for credit card details.

- **Research Excel upload into the application**

  Users may find it easier to create an excel file for the guestlist. Although it is not as user friendly as the application UI, it would allow the users to quickly tab through the columns and enter the relevant data. I discovered a Node library SheetJS js-xlsx which includes functions such as xlsx.readfile and xlsx.utilities.sheet_to_json which would let me upload and parse the Excel file uploaded by the user.

- **Plan the UI and how a user may use the application.**

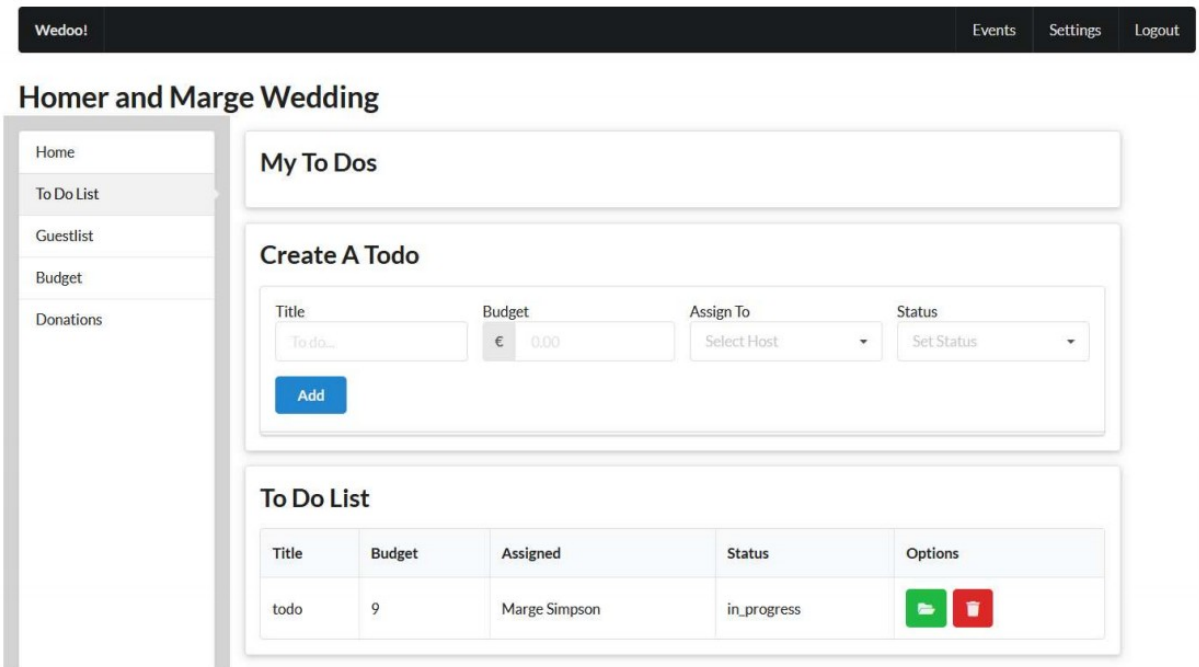
  The UI will mainly be used by the event host. The host will be able to create and manage an event using the UI and their log in information. However, it is more difficult to decide how the guests will RSVP and add their information. The two potential solutions to this are: guests will need to create an account with the exact email the host had provided, or guests will enter a unique id to log in. With the first option, all users would be able to create an account and the application would present all events where their email is included in the guestlist. The problem with this is users may be able to see events they have not yet been invited to. The hosts may not have finalised the guestlist and this could cause an issue.

With the second option, I do not want the users to be required to enter long unique IDs, if unique ids are present in the invites there may be room for error when sending these out. I decided that the unique ID would be a better solution. The system will create a short 5-character event IDs and guest IDs. Users will need the IDs to RSVP or enter any information.

2.2.2 Sprint 2, February 2021:

- **Create a basic application for the to do list.**

  I started my application development with the To Do list. I first created a basic to do model which would just accept the title and the budget amount. I created the input fields on the front end to accept a string and a number. I added a route to the to dos Controller which added a new to do object and printed the information to the console.

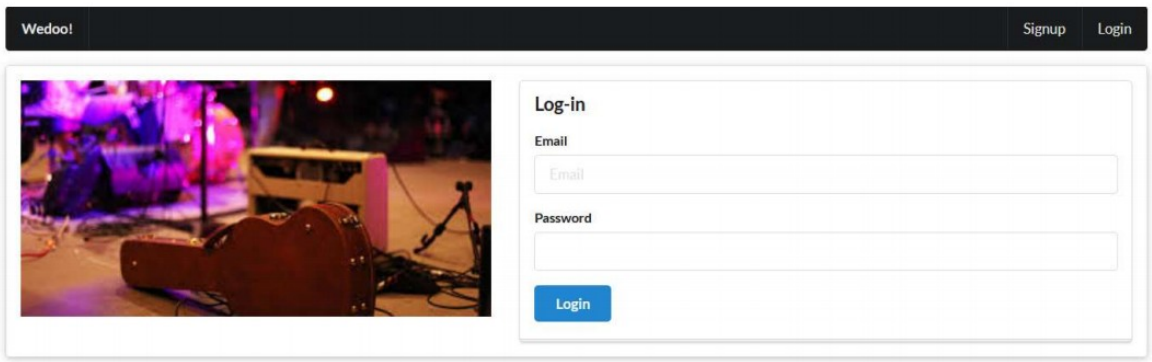


- **Link the application to MongoDB.**

  Once I had a basic application which created To Do objects. The next step was to get the information into MongoDB. I created the new database on my MongoDB account and included the information in the dotenv file. I used Mongoose to connect to the DB. I used the Mongo DB Atlas to test if the information I was sending was being stored in my database.

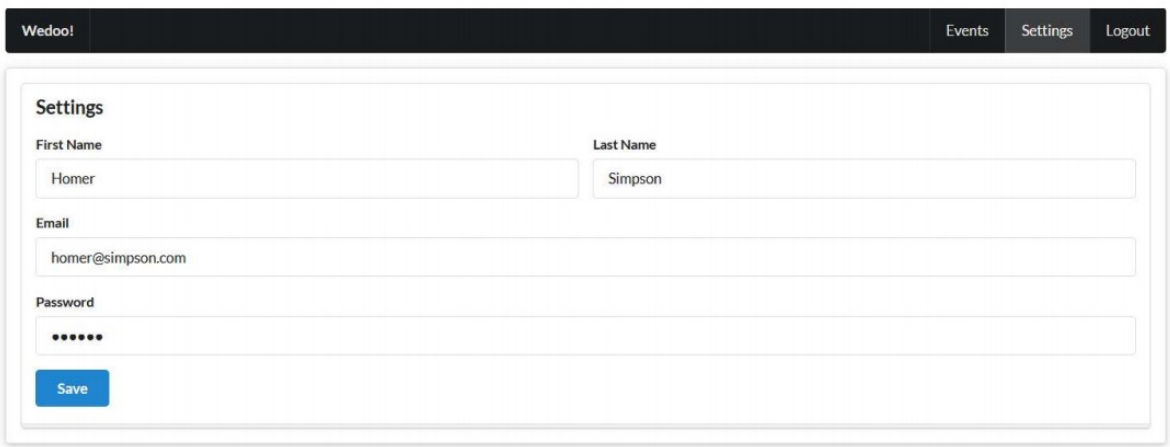
- **Add user login with validation.**

My to dos and events would need to be associated with Users. My next step was to add the user model and allow the user login. I created the landing page and menu bar with options to Sign Up and Login. I added a User model including first name, last name, email address and password. I included an accounts controller which would allow the user to create an account and validate login. If the login details were correct the user was brought to the main page.



- **Create a user settings page.**

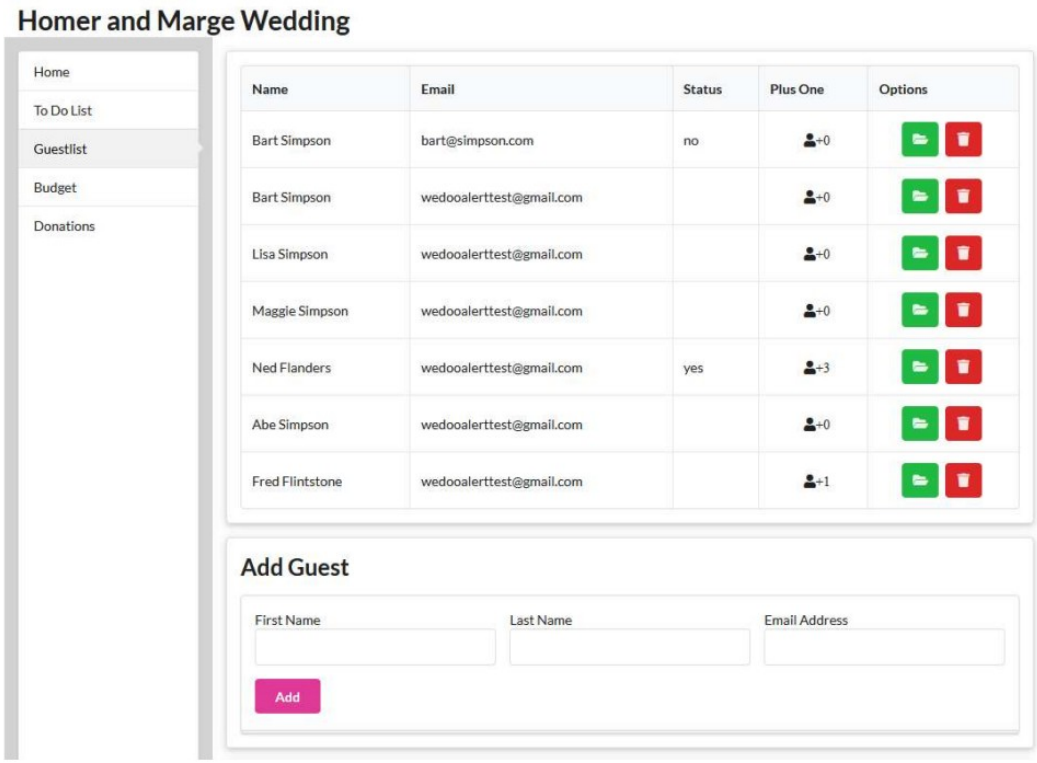
  Once logged in the user would need to be able to edit their information. I created the menu bar and included a settings button. This would take the user to a screen where they could update their information.



- **Allow add/delete of guests.**

  I created a Guest tab on the main page which would allow the users to add a guest. I started with a basic guest model containing first name, last name, and email address. I

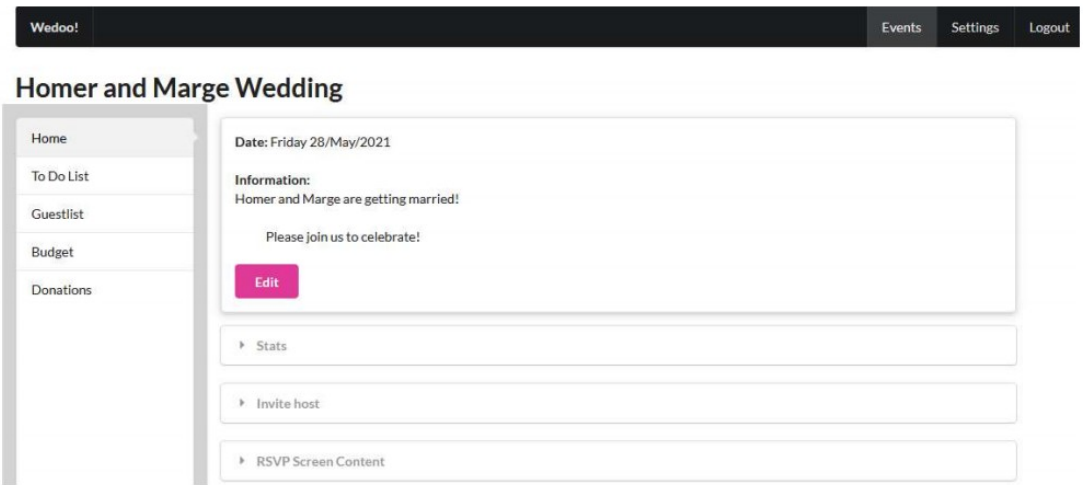
added a form where the host could enter the guest information. I added a delete button to the guestlist view where guests could be removed.



2.2.3 SPRINT 3, MARCH 2021:

- **Create an Event object.**

  To Dos and Guests should both be associated with an Event. Previously they were associated with the User, so I needed to create an Event Object which would contain an array of To Dos and Guests. I created an Event model which contained a title, info, and a data for the event. I used a calendar UI date picker in the form to allow users select the date. I created an event list table which would display the events. I added open and delete buttons to the event list table so users could click into the event.



- **Update the to do/guest creation methods to add new objects to an event.**

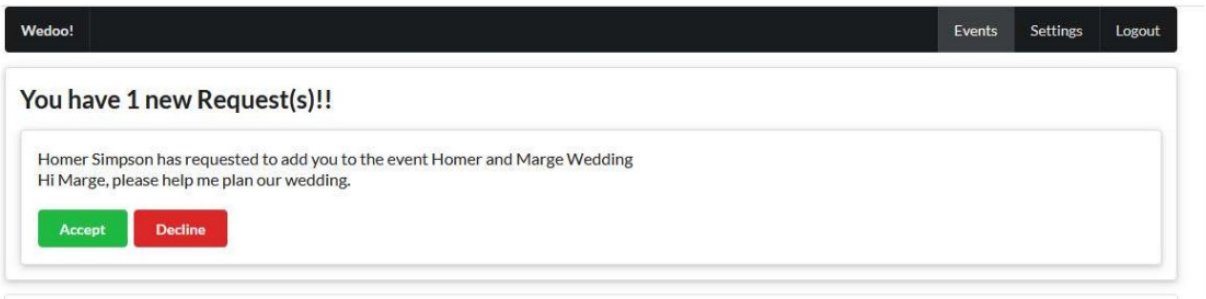
  When the event was created, I could then start allowing To Dos and Guests to be included in the Event. I updated the controllers for the add to do and add guest methods to search for the event and push the new object into the relevant list. I changed the routes for the to do and Guests to be in the event.

- **Create a new side menu for events.**

  When I added the to dos and the guests to the events the old menu tabs for the to do list and guestlist no longer worked, as they were not linked to the event object. To solve this, I created a vertical menu for the event which would allow users to select to do, Guestlist or the Event Homepage from the event page.

- **Create a request object which will send a request message to add another host to the event.**

  An important part of this application is allowing hosts to collaborate. To make collaboration easier I wanted to allow multiple hosts to make changes to the event. To allow one host add another host I decided to introduce a request model. One host can choose to add a host by entering their email and a request message. If the host already has an account or signs up with the same email, they will be able to see the request appear on their homepage. They can then choose to accept or reject the invitation. If they accept the event is then added to their list of events. Any host that is added has the exact same abilities as the original host. Eventually, I will add the functionality to allow emails to be sent to notify users that they have received a request.



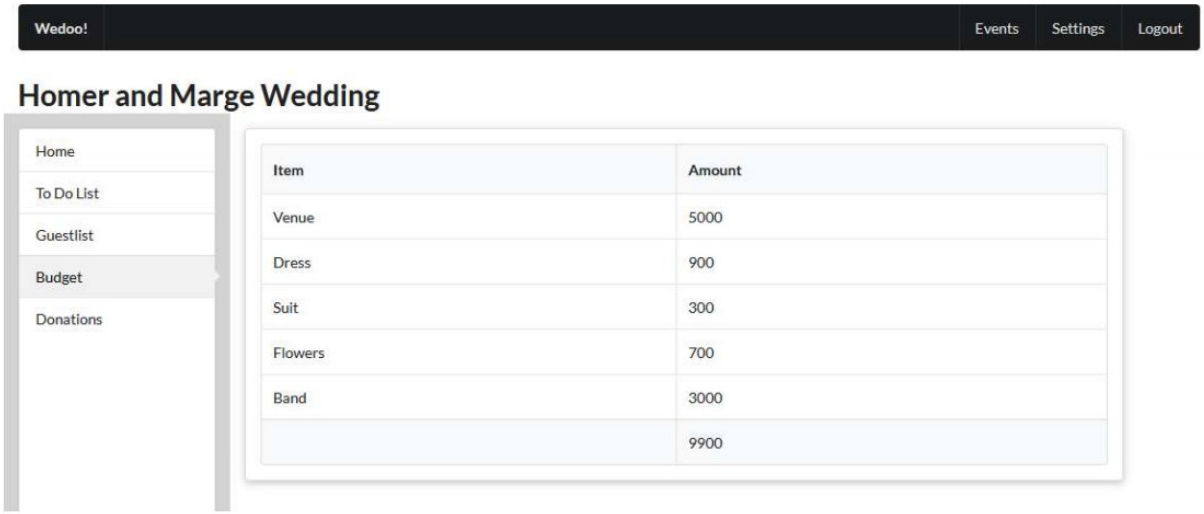
- **Research creating an email server and automatically sending emails from the applications.**

  I needed to research how my application will be able to send emails to users. As my application is a small project, it does not need to be a large email server. After some research I have found that it is possible to allow applications to send emails from your Gmail account. I created a new Gmail account to send emails and another Gmail account which I can use to test if emails are being received.

2.2.4 SPRINT 4, APRIL 2021:

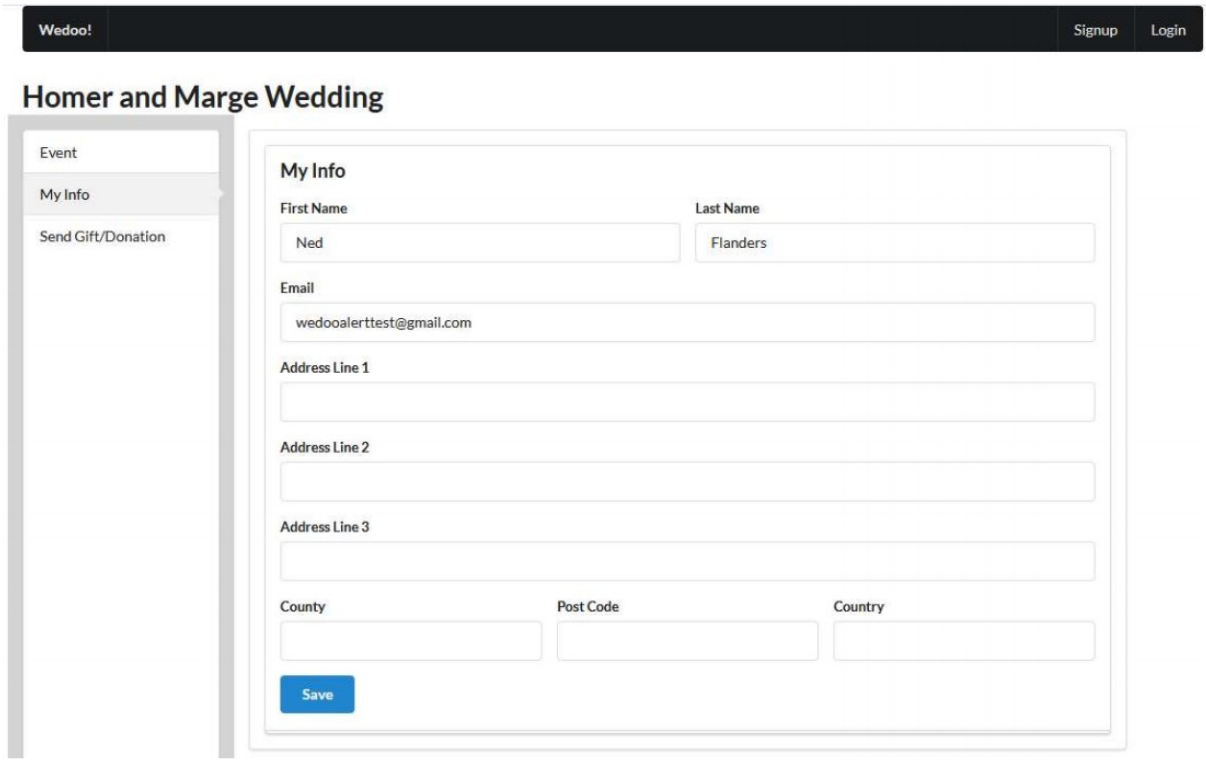
- **Build the Budget page of the application.**

  The budget was a key feature of my original idea. It is important for the hosts to keep an eye on the overall cost of the event. I decided to link the budget to the to do list, as most items in the to do list will also have a cost associated. Doing this will avoid the hosts having to enter certain items twice and I will not have to create another model to contain the budget items. The budget is a simple list of each To Do title and budget amount with a total sum. I created a separate item on the vertical menu for the budget view.



- **Add the views for the invited guest to add their information.**

  Up to this point the application only had the hosts' functionality. I wanted the app to allow guests to RSVP directly, rather than having the hosts manually update the guests' response. I used five-character event and guest IDs to allow guests log into the application. Once logged in the guest can see the event details. I added a page for the guests to update their information.

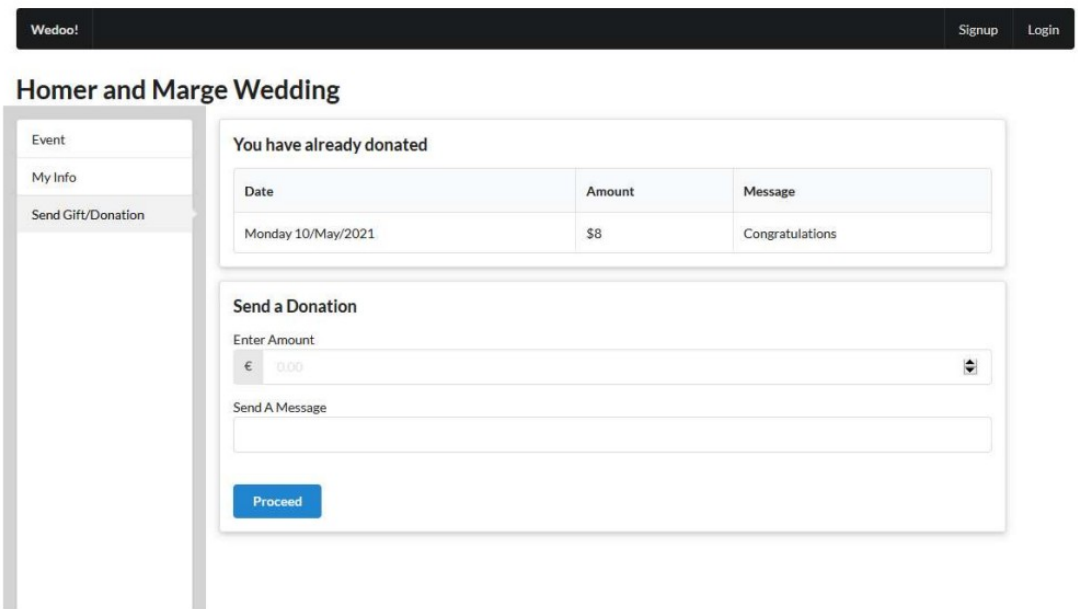


- **Build the "send a gift/donate" page which allows guests to send payment automatically to the hosts via PayPal.**

  Once the guest can log in and are taken to the homepage they can donate to the event. This function is to allow wedding guests to securely send a gift to the couple or to allow people to donate to a fundraiser. At first, I tried to combine the donation page and the confirmation page. I wanted users to enter the amount and message and to directly be able to proceed to payment. I had some trouble binding the input data in the form to the PayPal Create Order method. To avoid this, I instead created a separate confirmation page where a user can review their payment and process through PayPal. The on Approve method then processes the payment and returns an Order ID, the amount, and the message. This data is then combined into a payload and the application is updated.

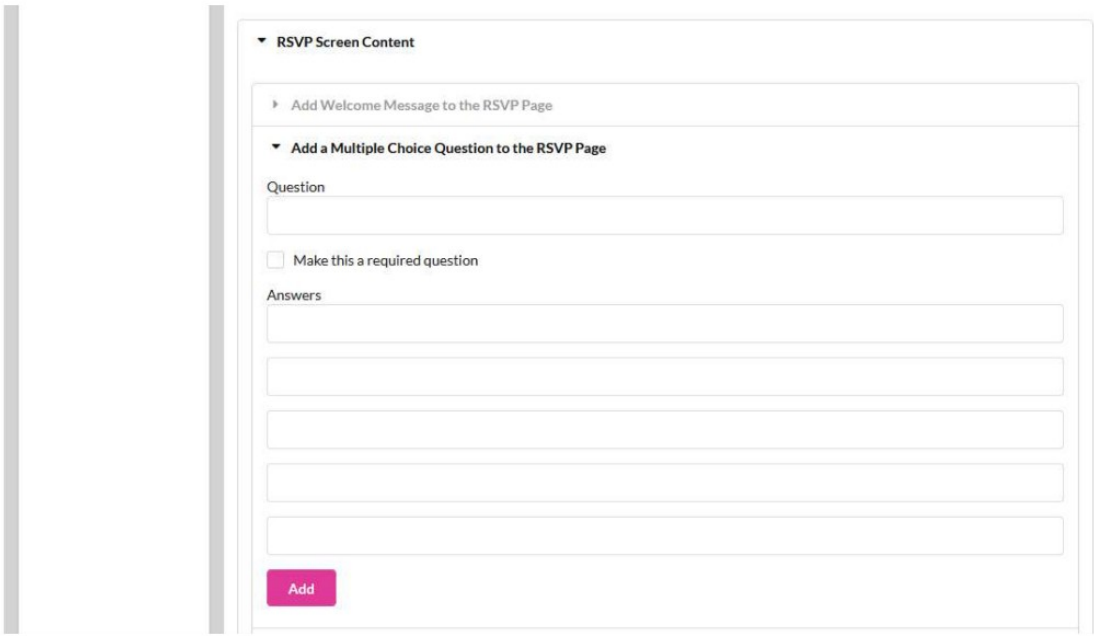
```
<script>
    paypal.Buttons({
        createOrder: function(data, actions) {
            // This function sets up the details of the transaction, including the amount and line item details.
            return actions.order.create({
                purchase_units: [{
                    amount: {
                        currency_code: "USD",
                        value: {{donation}}
                    },
                    description: 'WEDOO: Donation to {{event.title}}',
                }]
            });
        },
        onApprove: function(data, actions) {
            // This function captures the funds from the transaction.
            return actions.order.capture().then(function(details) {
                // This function shows a transaction success message to your buyer.
                alert('Transaction completed by ' + details.payer.name.given_name);
                const responsePromise =  fetch('/guest/{{guest._id}}/paypal-transaction-complete', {
                    method: 'post',
                    headers: {
                        'content-type': 'application/json'
                    },
                    body: JSON.stringify({
                        orderID: data.orderID,

                        message: '{{message}}',
                        donation: '{{donation}}'
                    })
                });
                responsePromise.then(function (responseFromServer){
                  location.href = '/guestdonation/{{guest._id}}';
                })
            });
        }
    }).render('#paypal-button-container');
    //This function displays Smart Payment Buttons on your web page.
</script>
</body>
```



- **Hosts can add questions.**

  This is a function I wanted to add to give the hosts the flexibility to request additional information from their guests. I have been invited to weddings where the hosts needed to know what main course I would order. Although this is not a usual occurrence, I wanted to be able to account for it. I built the option for the host to add multiple choice questions with up to 5 choices. The hosts can also decide whether the question will be required or not.



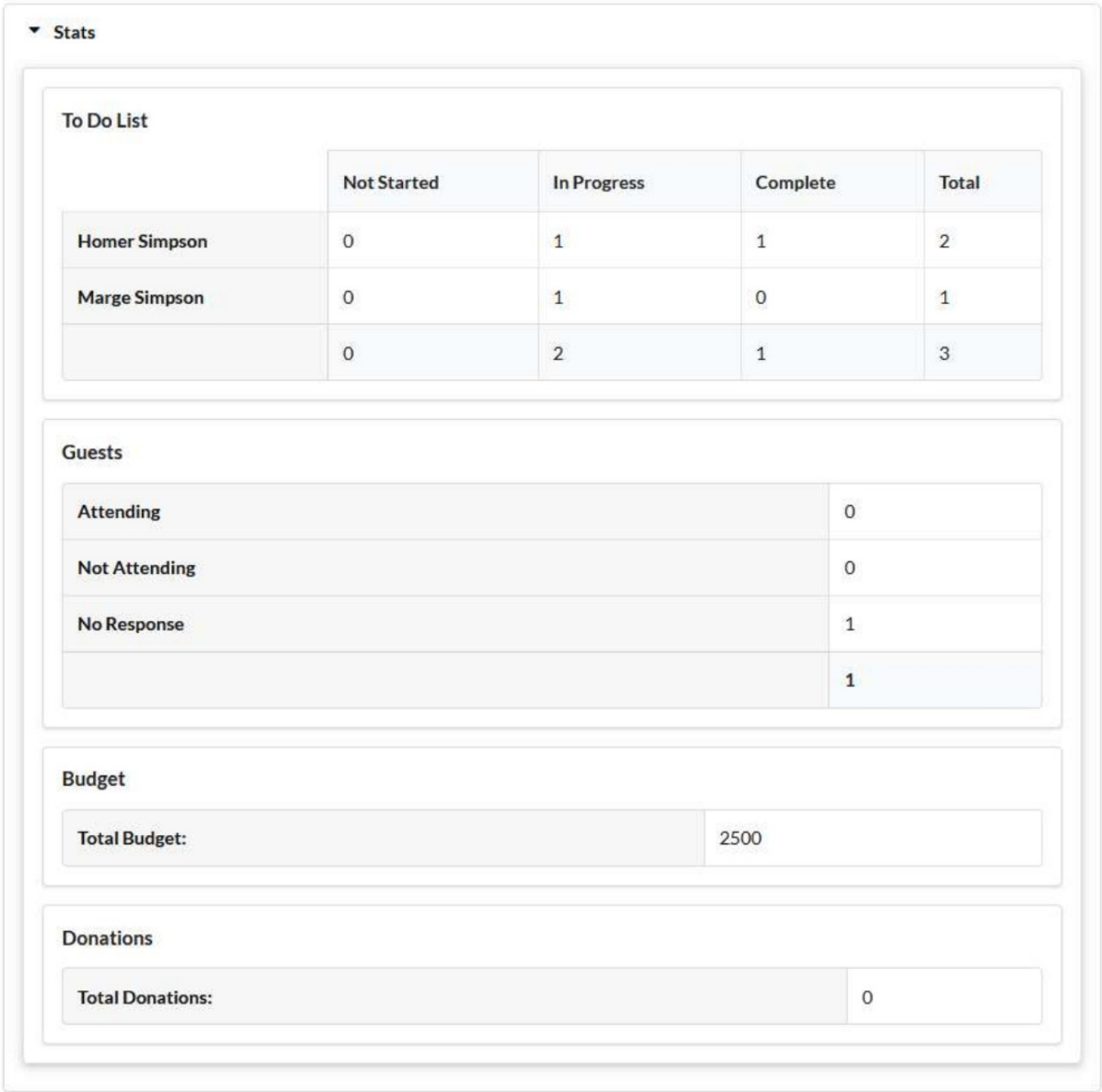
2.2.5 Sprint 5, May 2021:

- **Add the Excel Upload functionality**

  A Key feature of my application was the ability to upload the guestlist using an Excel file. The current UI only allows guests to be added one by one with many clicks required. An Excel file seemed like a good option as it is widely used, and users can quickly enter the guest information with a few clicks. I wanted the file to be as user friendly as possible, so I added an instructions tab and made the headers clear. I used the SheetJS js-xlsx library to read the file. This made it easy for me split the file into sheets and ignore the instructions sheet. I used the library to convert my data from excel to json. Once the data was in json format it was easy for me to access the correct data for each guest. I faced an issue trying to figure out how to add guests as a plus one of another guest. I decided to use a map to record the excel guest number and the corresponding guest ID. This was originally part of the loop through the data set. However, if the plus one appeared before the original guest it would cause an error. To solve this issue, I had to create a separate loop so that once all the guests have been created the system loops back over the data and if the guest is a plus one the map finds both the original guest and the plus one guest. The plus one information is updated for the original guest.

- **Create the summary view of the event in the event home page.**

  I wanted a summary view which would show the hosts the total outstanding items. They could easily see how many To Dos were incomplete, how many people had responded or how much they have spent. I created a stats summary view to display this information. For most information I needed to do some calculations and then pass parameters back to handlebars to display. However, I ran into some problems with trying to get the summary to display the hosts and their respective number of To Dos in each of the three categories; Not Started, In Progress and Complete. To Solve this issue, I decided to create a Stat Model for each statistic. Each Host had their own Stat object to contain their information. I created a CalculateStats Util which calculated all the required information for the summary page.



Stats

To Do List

	Not Started	In Progress	Complete	Total
Homer Simpson	0	1	1	2
Marge Simpson	0	1	0	1
	0	2	1	3

Guests

Attending	0
Not Attending	0
No Response	1
	1

Budget

Total Budget:	2500

Donations

Total Donations:	0

- **Add RSVP functionality.**

  I wanted to allow guests to RSVP for themselves and any plus ones. I wanted the app to be flexible enough to allow one person to respond on behalf of their whole family. I used handlebars to display each plus one and a radio checkbox option for attending or not attending. I then added the guest id to the name tag. I had to get the object keys from the payload and extract the guest id to decipher which guest would be updated with which response.

  I ran into an issue getting the updated RSVP status displaying on the guest page. To get the values populating correctly I had to move the script section inside the each statement and use the guest id to identify which value should be selected.

```
{{#each guest.plusOne}}
    <label>{{firstName}} {{lastName}}</label>
    <div class="field">
        <div class="ui radio checkbox">
            <input type="radio" name="rsvpstatus.{{_id}}" value="yes">
            <label>Yes, I'm attending!</label>
        </div>
    </div>
    <div class="field">
        <div class="ui radio checkbox">
            <input type="radio" name="rsvpstatus.{{_id}}" value="no">
            <label>No, can't make it</label>
        </div>
    </div>
    <script>
        $('input[name="rsvpstatus.{{_id}}"][value="{{rsvpStatus}}"]'). attr("checked", "checked");
    </script>
{{/each}}
```

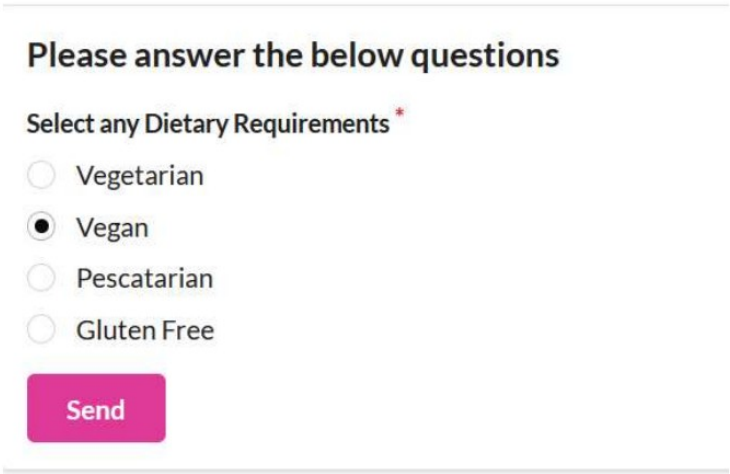
- **Guests can answer questions.**

  My intention with the questions was to allow hosts to find out more information from their guests than just their rsvp status. They can use this to find out about dietary requirements or maybe accommodation requirements. In some cases, questions will only need to be answered by the main guest, other times the hosts will need the information from the plus one also. To simplify the application, I allow the main guest to submit their response to the questions on the main page. If they also want to answer on behalf of their plus ones there is a button which will take them to that answer page too.

  As my answer model was contained in my guest model rather than being in the question model, it was more complicated to add the answers to the guest and display the answer to each question. I created a helper for dealing with the questions. I included the question id in the payload attribute name, I used this id to search the guests answers array for an answer with the same question id. If an answer was not returned, a new answer was created.

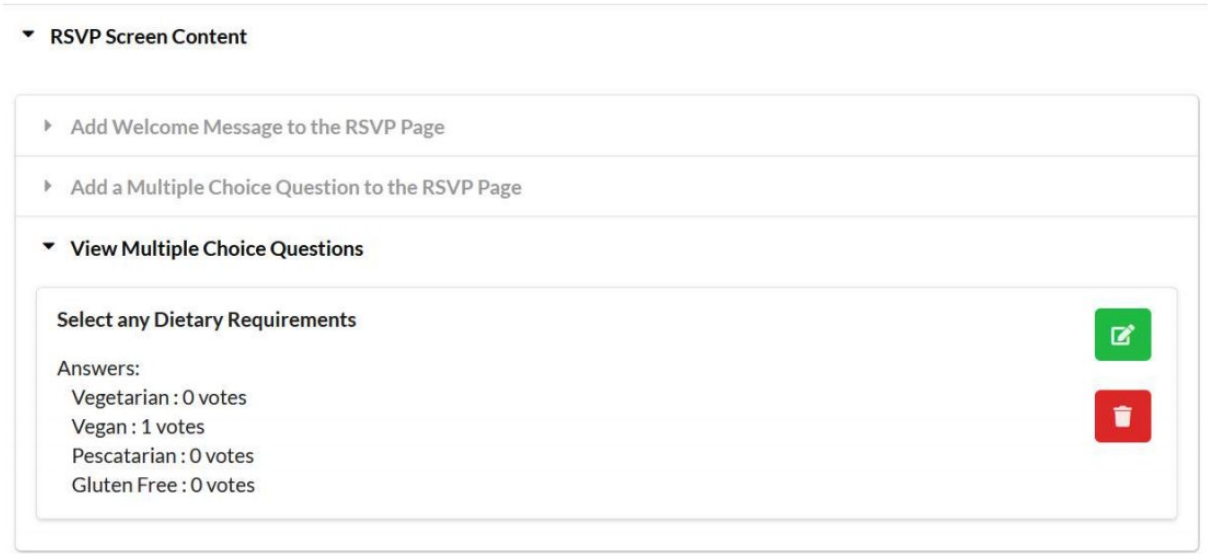
I had trouble showing the selected answer box checked after the user had submitted their answer. At first, I had tried to solve this using handlebars. I created a method which would pass the question id and the guest id. Using these variables, the method would return the answer the guest had input for that question. I then wrote script which would mark that value as checked. However, Handlebars would not work with the asynchronous method and only a promise would be returned.

After that had failed, I decided I needed to pass the answer back with the questions. I decided to add a new field called selected to my questions which would set the selected answer for each question on the list. I then could use the selected property to find the answer and check the radio button which matched the selected value.



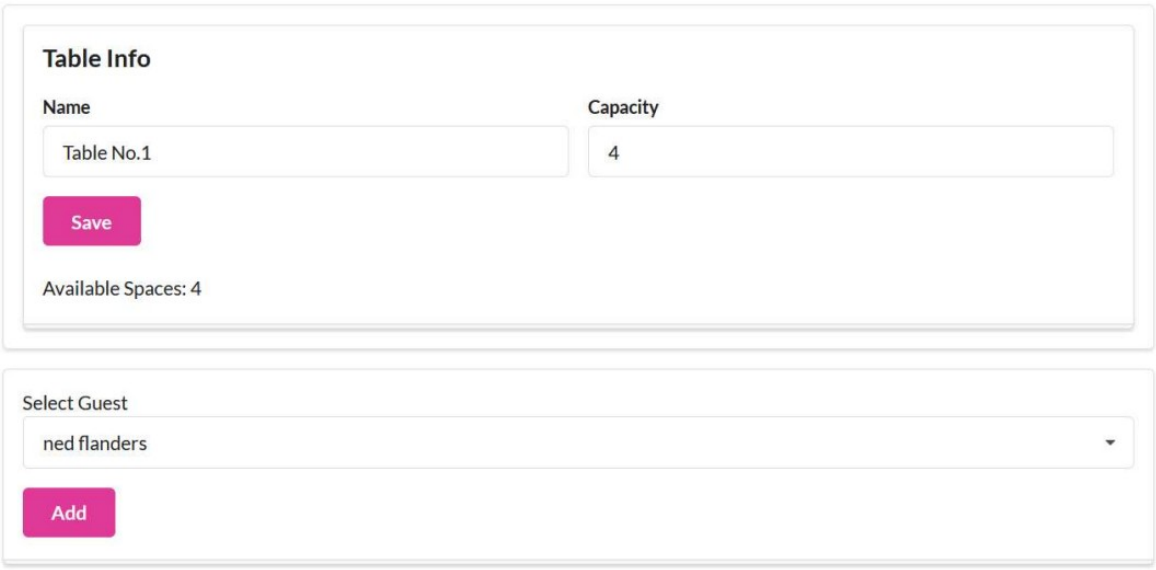
- **Hosts can view answers.**

To make it possible for the hosts to view the answers I decided to temporarily amend the answers to each question to be the answer followed by the count of votes that answer has got. This meant I did not have to pass another parameter.



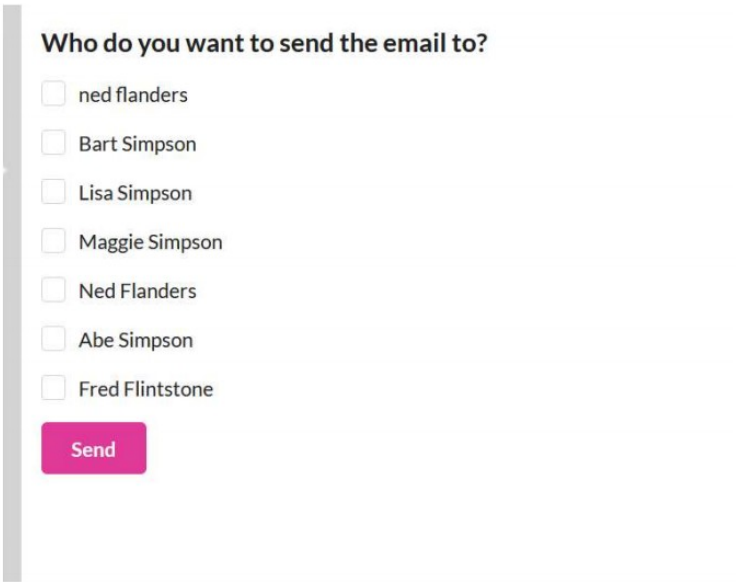
- **Table plan**

  I began the table plan by creating a table model which included table number, name, capacity, and a guest array. The host creates tables by selecting the number of tables and the capacity. The application then creates that number of empty tables. The host may then click on the tables and select guests to add. The add guest dropdown only displays the guests which have not yet been assigned a table.



- **Email Info/RSVP Requests**

  I added two buttons to the top of the guestlist view to allow the user to send info requests and RSVP requests. On clicking these buttons, a list of guests is displayed. This list only includes main guests, it does not include plus ones. In the RSVP list, only the guests who do not have an RSVP status are included. The user can check as many of these guests as they like and hit send. This then sends an email with a link to the guests.



- **Deployment**

  I deployed my application on Heroku. I first created a blank Heroku app, I then committed my code to git and pushed it onto my Heroku repo.

3 Design and Analysis

3.1 Technologies

- **JavaScript**: JavaScript is widely used for creating web applications. I feel much most comfortable coding in JavaScript.
- **Node**: Node is widely used for web application development. It is fast and works well for real-web applications.
- **Mongoose**: I chose to use a NoSQL database. Mongoose makes it much easier to parse my database.
- **Handlebars**: Handlebars allows me to write helpers. I find the helpers make it easy to change what the user sees based on the data the server has returned. This meant I could write conditional statements in the front end rather than the back end.
- **Semantic**: Semantic makes it much easier to add different elements to my UI. I used it to make my application look more professional.

3.2 Tools

- **Webstorm**: I chose to use Webstorm as my IDE. I like how Webstorm allows me to easily commit to Git.
- **MongoDB**: MongoDB Atlas allows me to store my NoSQL database on the cloud.
- **Hapi – Joi, Boom**: Hapi framework allows me to add different elements to my application more easily. Joi will allow me to easily validate the data input. Boom makes my error messages look more professional.
- **Heroku**: I deployed my application on Heroku. I find Heroku more straightforward to deploy my application.
- **GitHub**: GitHub is a reliable and widely used version control system. It integrates seamlessly with Webstorm.
- **PayPal Sandbox**: After much research on how to process payments I found PayPal was much clearer on how I could integrate it in my system. This will give users a choice to add their credit card information or to login to their PayPal account. Users may feel more secure paying through PayPal.

4 User Experience

4.1 User Stories

4.1.1 User Login

User can login to the application. The application will cross check the user email/password combination against the information stored in the application.

On unsuccessful login an error message will appear on screen.

On successful login the user will be brought to their home screen where they can see the events they are hosting. The user may also see events they have been invited to if they have chosen to add them.

4.1.2 User Signup

User may sign up to my application using the “Start Here” or “Sign Up” buttons on the home page. Users enter a unique email, their information, and a password to sign up.

4.1.3 Create Event

Any user may create an event. They can provide a title, date, and description for the event.

4.1.4 Edit Event

Hosts may edit the event information, title, or date. They may also add questions which the guests can respond to when completing the RSVP section.

4.1.5 Delete Event

A host may choose to delete the event. Once the event is deleted it cannot be restored.

4.1.6 Add a Host

Once a user has created an event, they may add another host using the “Add Host” button. This will create a request which will send link to the email address. That person will need to sign up and accept to have host rights. Once a host has been added, they may not be removed.

4.1.7 Add To Do

Hosts can click the To Do Option on the side menu to create a To Do. To Dos can be assigned to any of the event hosts. To Dos can have a cost associated with them which will feed into the event budget. To Do must have a status.

4.1.8 Add Guest

Hosts can add guests to the event. Guests are not notified that they have been added. If the host chooses to send them a request, they will then be allowed access to the application.

4.1.9 Delete Guest

Hosts may delete guests. Other hosts are not notified if a guest is deleted.

4.1.10 Upload Guestlist

Hosts may upload a excel guestlist to the application.

4.1.11 Delete all guests

If a host decides to remove the existing guests and upload a whole new guestlist they may do so using the Delete All Guests button

4.1.12 Add Tables

A host may decide to add tables to the event. They can select the number of tables they want to add and the capacity of those tables.

4.1.13 Assign guests to tables

A host can view the table and choose guests to add to each table. The dropdown menu for the guests only displays guests who are not assigned a table. Hosts can no longer assign guests to a table once it has reached capacity.

4.1.14 Remove guests from tables

A host may also remove guests from the table they were assigned and reassign them to a different table.

4.1.15 Host Edit Guest

Hosts may Edit Guest information. They can assign a table number, they can change their RSVP status and they can assign them a status. They may also add and remove plus one information.

4.1.16 Guest Edit Guest

Guests can edit their own information. They can RSVP for themselves and any plus ones. They may add/update their address or answer any questions the hosts have added.

4.1.17 Guest RSVP

Guests will be provided with a unique code which they may enter to RSVP to their invite.

4.1.18 Guest Send Gift

Guests may choose to send a monetary gift using the PayPal function on the application. They may choose an amount and pay using credit card or PayPal. This money is automatically sent to the Hosts' PayPal Account. Money sent cannot be reversed.

4.1.19 Budget View

Hosts may view the event budget. This is a total of all the budgeted To Do costs. Hosts may choose to estimate how much they expect to receive from the gifts/donations and offset this from their budget.

4.1.20 User settings

Users may update their information/password in the settings page.

5 Conclusion

5.1 Reflection

My application has achieved a lot of what I set out to do. I did encounter some unexpected issues and in certain areas the functionality is different to how I expected. There were a lot of new technologies I used in this application which I had not used before such as email servers, PayPal payments and excel uploads.

Some of the challenges I found were in the rsvp and question sections. To solve these problems, I had to pass the ID in the keys with the payload. I learned how to Object keys can be used for such instances.

In the Questions I had to identify the answer for the question and then check the correct box. This was the first time I had to create a temporary parameter in an object. This allowed me to temporarily add a selected field to the question object and pass it into the front end.

This project was a good learning experience and I completed most of what I had planned to do. There are some further functionalities that I would like to add to my application to make it feel more professional and complete.

5.2 Key Skills

5.2.1 Email Server Setup

The application was my first attempt at setting up an email server. I used Gmail as my email account as I was already familiar with. After some research I found a way to get my emails sending from my Gmail account without any issues.

5.2.2 Excel Upload and Parse

I began by researching how I was going to upload my excel file and have it parsed. I found the xlsx library and that made it easy for me to parse the file into what I wanted.

5.2.3 Handlebars

I used a lot of handlebars in my application. I had not much experience with registering handlebars helpers prior to working on this app. I found it useful for hiding sections when there was no information to include. I also found it useful for formatting my dates.

5.2.4 MongoDB

MongoDB was my main database for my application. I found using the MongoDB Atlas was good for checking if my objects where updated. I used mongoose to access my database. I could create methods like the findByGuestId or the findByShortId to query my database.

5.3 Future Work

5.3.1 Security

Security has not yet been a focus for me in building this application. Going forward security should be a main priority. As this could potentially be storing a lot of sensitive information it is important that this is an application people feel they can trust. At a minimum, all information should be encrypted in my database. I would need to restrict the number of IP addresses which can access the database.

5.3.2 PayPal Transactions

At the moment, I use PayPal only to accept donations from the guests. This means that all donations made to the hosts are not sent directly to the hosts' account. They instead are paid into Wedoo's account. In future, I need to build the functionality which will allow the hosts to choose to extract all the funds as one large lump sum or to have each donation in turn redirected immediately to the host's account.

To do this I would need to allow hosts to save their account information. Allowing all hosts to update the account information and transfer the funds will lead to security issues. I would need to create an audit trail which would highlight when account information was changed and who changed it. I also need to consider allowing hosts to enable two step approval for account changes and funds transfer on the application.

5.3.3 Hosts Notifications

At the moment, the hosts are not notified of any changes in the application. It is possible that there may be a last-minute RSVP or a donation after the wedding that the hosts may not see. Some hosts may like to be notified when guests RSVP or when someone sends them a donation. In future, it would be good include notification preferences in the user settings.

5.3.4 Filter, Search and Sort

My application does not include any filter, sort, or search functionality at the moment. If you were looking for one guest from a list of hundreds it could be very difficult to find. There should be an option for hosts to search for guests, to dos, events, and donations. To dos should be able to filter based on the assigned host and status. Donations should be able to sort new to old or high to low.

5.3.5 Export to Excel

A lot of people like to use Excel to view their data. If a host is updating the guestlist, they might find it easier to export to excel to make their changed and then re-upload. This also allows the host to do use any Excel tools or formulas to visualise their guestlist. They can create pivot tables to further analyse their data.

5.3.6 Duplicate Checks

There are currently no duplicate checks in the application. If the same Excel file is uploaded twice the hosts will need to manually delete all the duplicated guests. The application should have a built-in functionality to detect if there may be a duplication. If the system suspects duplication it should provide a popup and allow the host to confirm if the users are the same. It could then be made possible for the host to do their guest updates on the excel file and then upload to the app.

5.3.7 Pop-ups for delete confirmation

Events, Guests and To Dos can all be deleted in one click. As with most applications, I believe Wedoo should ask the user to confirm before deleting any information. I also would like to include pop-ups for log out confirmation.

5.3.8 Hosts can edit RSVP status

I have not yet built the screen where the host can change a guests RSVP status. There is a work around for the moment where the host can get the Event Id and Guest Id, then log in as the guest using these credentials. It would be far more user-friendly if the host could directly update the guest rsvp status.

5.3.9 HTML Emails

My email notifications at the moment are plain text email alerts. In future I would like to make these more professional looking. I would like to upgrade these to a HTML format.

6 PROJECT URLS:

Deployed Application: https://aqueous-falls-19394.herokuapp.com/

GitHub Repository: https://github.com/aoloughlin92/hdip-final-project

7 WORKS CITED

(n.d.). Retrieved from https://www.npmjs.com/package/js-xlsx

(2019). Retrieved from morioh.com: https://morioh.com/p/ca75996654d1

PayPal Developer Docs. (n.d.). Retrieved from developer.paypal.com/: https://developer.paypal.com/docs/checkout/integrate/

Semantic UI. (n.d.). Retrieved from https://semantic-ui.com