**Team 4 – Product Backlog**

**Team members:** Pooja Tewari, Shivangi Chand, Siddharth Dhar, Sripath Mishra, Ethan Niu

**Project Coordinator:** Adam Johnston

**Problem Statement**

Organizing a party is not a simple task and takes a lot of planning. Everyone the host invites would inevitably bring gifts to the party but the host will have to give away the gift if it is a duplicate or it is something they don’t want to use. As a solution to this problem, and to streamline the party hosting process, we will make a web app that would make the process of planning a party easier with features like creating a guest list and sending invites, finding a place to host the party and catering services depending on the type of party it is and the host’s budget. Also, the host can make an Amazon wishlist and share it with everyone invited. Invited guests can decide what to gift from the wish list, they can RSVP through the web app and also get directions to the party location. Hence we aim at making a one-stop party app.

**Background Information**

**Audience**

People across the globe throw parties or events on various occasions. The experience can be further enhanced and streamlined if a guest can use an online platform that provides access to the guest list as well as various details about the party. With Fiesta, we aim to provide a strong online platform to notify and invite guests to a party or an event and also provide them with directions to the venue. In the same vein, we aim to make a smooth experience for the host by providing various features such as selecting a catering service, location or event just by a click of a button.

**Similar Platforms**

There are very few party hosting applications out there that would achieve the same functionality as our application. The closest platform to our application is Facebook’s party invite list. However, we would like to provide more functionality to our users than any other application out there that falls within the same domain.

**Limitations**

There are very few party hosting applications that people can make use of. The most notable one is Facebook, which permits users to host parties and also provides access to the invite list. However, there is nothing more to it than accessing the invite list and getting a list of the users attending the party. With Fiesta, we aim to provide a smooth experience for both the host as well as the guests. Fiesta will be a top-notch application that provides access to the invite list, permits the host to select a catering service, provides information about dress codes, cautions anyone under 21 to attend a party serving alcoholic drinks, provides a list of events at the party, sends out reminders about the event, and so much more.

**Environment**

As part of our development environment, the backend will be made using Node.js with Express. We will be using Postgres for creating a fast and scalable database. We will be using multiple NPM packages for securing and making the backend more efficient. We will be using React, HTML, CSS, and Javascript to make the frontend of the web app.

**Functional Requirements**

Green: All complete.

Yellow: Only frontend integration is left.

|  |  |  |  |
| --- | --- | --- | --- |
| **Backlog ID** | **Functional Requirement** | **Hours** | **Status** |
| 1 | As a user, I would like to be able to register for a Fiesta account. | 7 | Planned for Sprint 1 |
| 2 | As a user, I would like to be able to log in and manage my Fiesta account. | 7 | Planned for Sprint 1 |
| 3 | As a user, I would like my password to be reset if I forget it. | 7 | Planned for Sprint 1 |
| 4 | As a user, I would like to choose a place for my event. | 7 | Planned for Sprint 1 |
| 5 | As a user, I would like to choose a catering option for my event. | 7 | Planned for Sprint 1 |
| 6 | As a user, I would like to be able to easily access Fiesta across all of my web-enabled devices. | 7 | Planned for Sprint 1 |
| 7 | As a user, I would like to be able to contact providers of event hosting supplies. | 7 | Planned for Sprint 2 |
| 8 | As a user, I would like to be able to generate the list of people to be invited. | 7 | Planned for Sprint 1 |
| 9 | As a user, I would like to be able to manage the guest list to the event. | 7 | Planned for Sprint 1 |
| 10 | As a user, I would like to be able to generate and send a wishlist of gifts. | 7 | Planned for Sprint 2 |
| 11 | As a user, I would like to be able to cross off an item on wishlist as a guest. | 7 | Planned for Sprint 2 |
| 12 | As a user, I would like to be able to get directions to the event place. | 4 | Planned for Sprint 2 |
| 13 | As a user, I would like to be able to order items or suggest items for the wishlist from Amazon. | 4 | Planned for Sprint 2 |
| 14 | As a user, I would like to be able to change event details if am the host. | 7 | Planned for Sprint 2 |
| 15 | As a user, I would like to be able to view the guest list, RSVP to the event and view the list of RSVP’d events. | 7 | Planned for Sprint 2 |
| 16 | As a user, I would like to be able to share images of the party. | 7 | Planned for Sprint 2 |
| 17 | As a user, I would like to be able to design cards for invites. | 10 | Planned for Sprint 1 |
| 18 | As a user, I would like to be able to create a checklist of tasks for the event and assign people to the tasks. | 7 | Planned for Sprint 2 |
| 19 | As a user, I would like to be able to ask questions regarding the event to the host. | 7 | Planned for Sprint 2 |
| 20 | As a user, I would like to be able to answer the questions of the guest. | 7 | Planned for Sprint 2 |
| 21 | As a user, I would like to get suggestions for the event venue according to the venue budget. | 7 | Planned for Sprint 2 |
| 22 | As a user, I would like to get suggestions for the catering for the event according to the catering budget. | 7 | Planned for Sprint 2 |
| 23 | As a user, I would like to have a way to connect to the developers for providing feedback. | 7 | Planned for Sprint 1 |
| 24 | As a Software Developer, I would like to have a continuous development and continuous integration pipeline. | 3 | Planned for Sprint 1 |
| 25 | As a Software Developer, I would like to have a Docker container for local testing. | 4 | Planned for Sprint 1 |

Note: For hours, we have taken an estimate of 3 hours to complete one backend route and 4 hours to complete its frontend counterpart.

**Non-Functional Requirements**

**Architecture and Performance**

In order to successfully develop Fiesta, our team will divide into two sub-teams: frontend and backend. The backend team will begin by creating a RESTful API (Representational State Transfer) written in Node.js. The Express framework will be used and our team will be incorporating Bluebird to help with versioning our API. Through using these frameworks, we will be able to create multiple instances of the server in just a single deployment. Our application will use a fast and scalable database management system called Postgres. We chose Postgres primarily as it has many additional features. One such feature is a vector which allows our app to attain faster search results. Our app will follow linting rules provided by Airbnb so that we can keep our code organized. Docker will be incorporated into our app for easy development of our API. It will allow us to make the development process independent of the local system. Our goal would be to send a response within 0.002 seconds for each endpoint when the database tables have at least 100,000 columns and a minimum of 100 users will be sending their requests simultaneously.

The frontend will be implemented in React Native, which will be in Typeset. Our app will incorporate elements that will allow faster page loading. These specifications would allow us to export our frontend to Android and iOS apps if the time permits. Multiple CSS files will be used for all styling as this would allow us to improve file organization. React Dom and path obscuration methods will be used to increase our security levels.

For both the frontend and the backend, CircleCI will be used to incorporate a CD/CI pipeline (continuous deployment/continuous integration). The hooks provided by CircleCI will be used to connect our GitHub repositories to it. This would allow us to fully build and test our code after we push our code onto the remote branch. Most than 95% of our test cases will be automated. This will be used as a deployment tool as well. Special rights will be given to the master branch so that whenever a commit is pushed to the master branch or a branch is merged to the master branch, after successful execution of all the test cases, the API will be deployed to Heroku while the client will be deployed to GitHub pages.

**Security**

Security is a primary concern for any software today. To ensure secure authentication, our backend team we will be using bcrypt to encrypt any password using the blowfish algorithm. JWT tokens will be used to authorize and securely authenticate the user for interacting with the API. Through this method, a token will be valid for only five minutes. Once the five minutes are over the user would have to log in again to get a new token. To improve usability the frontend of the application will manage the token and token renewal till the session is either open or till twelve hours have passed. We will require a minimum of two levels of permission in order to interact with the API. To check for token validation, a middleware will be implemented for all routes except the getToken route. We will be using official modules from npm for database protection. These modules will allow us to prevent SQL injection. We will not be using string manipulation to create the SQL statements. Instead, when we create SQL statements, the ‘?’ character will be used. We will be passing the values to the npm modules to handle the rest. We will be disabling CORS. This will prevent cross-site-scripting. Measures will be taken to prevent directory traversals, denial of service attacks, etc.

The minified code will be created by the front end team so that it will be hard for people to breach our security barrier. Pseudo paths will be used in the URL to prevent anyone from knowing the page type (.html, .php, etc) and other additional security features will also be used to ensure a secure experience for all of our users.

**Usability**

Usability is a primary aspect of any software that is created. Our team aims to create a user-friendly UI so that people of any age can easily operate the web app. The UI will allow any user to easily register and instantly start planning any parties/events required. To increase usability, we will be using React in our frontend as it allows for the application to be both smooth and fast. The UI will be responsive to user input and also allow the app to call APIs smoothly without freezing the browser. Our app will be compatible with most popular browsers such as Firefox and Chrome. We will focus on having a cleanly aligned UI that is visually appealing to the user so that they are more likely to use our app again. Our frontend team will make sure that elements are aligned properly and stay put when a browser is resized or when the application is used on any web-enabled device.

**Hosting/Deployment**

Heroku’s free version will be used to host the Fiesta API and our team will be deploying two instances of the server itself. Staging will be one instance and that will be used by our frontend team so that the code can be tested. The second deployment instance will be the one that is open to the public for usage, so it will be the most important one. We chose Heroku because it allows for a Postgres database with enough storage for our app. In case we need a database that requires a higher capacity, then our team will host it on Amazon Web Services. The CD/CI pipeline will be implemented and through that our CircleCI Integration will create a hook. This hook will then ensure that our master branch is being deployed in both the instances of our server. The frontend will be using environment variables to connect to the right deployment. We will also be using Github pages to host the frontend since the service is free. The hook that is created in CircleCI will make sure that the tested frontend is hosted through Github pages.

**Use Cases**

|  |  |
| --- | --- |
| Case 1: As a user, I would like to be able to register for a Fiesta account. | |
| **Action:**  1. Choose ‘create account’ option  3. Fill out form  5. Submit form | **System Response:**  2. Account creation form appears  4. Process form  6. Indicate success or failure to the user  7. Form disappears |
| Case 2: As a user, I would like to be able to log in and manage my Fiesta account. | |
| **Action:**  1. Click ‘login’ button  3. Fill out form  4. Submit form | **System Response:**  2. Initiate the login process  5. Redirect to root |
| Case 3: As a user, I would like my password to be reset if I forget it. | |
| **Action:**  1. Choose ‘Forgot Password’ option  3. Fill out form  4. Submit form | **System Response:**  2. Initiate the password reset process  5. Redirect to login |
| Case 4: As a user, I would like to choose a place for my event. | |
| **Action:**  1. Click on ‘Location’  3. Select party location on map or input coordinates  5. Submit the party location | **System Response:**  2. Redirect user to party location page  4. Find location/coordinates on the map  6. Confirm to the user that location is selected |
| Case 5: As a user, I would like to choose a catering option for my event. | |
| **Action:**  1. Click on ‘Catering’  3. Select catering location on map or input coordinates  5. Submit the catering business location | **System Response:**  2. Redirect the user to the catering location page  4. Find location/coordinates on the map  6. Confirm to the user that catering location is selected |
| Case 6: As a user, I would like to be able to easily access Fiesta across all of my web-enabled devices. | |
| **Action:**  1. Log out on the initial device  2. Log in on a different device | **System Response:**  3. Load party plan linked to the account |
| Case 7: As a user, I would like to be able to contact providers of event hosting supplies. | |
| **Action:**  1. Click on ‘Suppliers’  4. Confirm event supplier contacted | **System Response:**  2. Query for nearby event hosting suppliers  3. Display supplier contact info to the user  5. Indicate that event supplies fulfilled |
| Case 8: As a user, I would like to be able to generate the list of people to be invited. | |
| **Action:**  1. Click on ‘Invitations’  3. Add guests by email address  4. Close dialog | **System Response:**  2. Show dialog for adding guests  5. Indicate that the guest list has been generated |
| Case 9: As a user, I would like to be able to manage the guest list to the event. | |
| **Action:**  1. Click on ‘Invitations’  3. Add/remove guests or edit guest email  4. Close dialog | **System Response:**  2. Show dialog for editing guest list  5. Indicate that the guest list has been saved |
| Case 10: As a user, I would like to be able to generate and send a wishlist of gifts. | |
| **Action:**  1. Click on ‘Gifts’  3. Import Amazon wishlist or add gifts manually  4. Close dialog | **System Response:**  2. Show dialog for adding wishlist  5. Indicate that wishlist has been generated |
| Case 11: As a user, I would like to be able to cross off an item on wishlist as a guest. | |
| **Action:**  1. Click on ‘Gifts’  3. Select a gift to check off the wishlist  4. Submit change and close dialog | **System Response:**  2. Show dialog for available gifts  5. Indicate that wishlist has been modified |
| Case 12: As a user, I would like to be able to get directions to the event place. | |
| **Action:**  1. Click on ‘Directions’  2. Choose transportation type | **System Response:**  3. Show user directions on a map (rideshare pricing if applicable) |
| Case 13: As a user, I would like to be able to order items or suggest items for the wishlist from Amazon. | |
| **Action:**  1. Click on ‘Gifts’  2. Choose ‘Find suggestions’  5. Order item (and check off wishlist) or add suggested an item to wishlist | **System Response:**  3. Query Amazon API for items similar to wishlist items  4. Show user suggestions and their prices  6. Indicate to the user that item ordered |
| Case 14: As a user, I would like to be able to change event details if I am the host. | |
| **Action:**  1. Click on a category to edit (Invitations, Catering, etc.)  3. Edit necessary details and close dialog | **System Response:**  2. Show dialog of editable options for the category  4. Indicate to the user that modifications are saved |
| Case 15: As a user, I would like to be able to view the guest list, RSVP to the event and view the list of RSVP’d events. | |
| **Action:**  1. Click on ‘Invitations’  3. Select the invitation user wants to RSVP for  5. The user selects from ‘Will Attend’, ‘Not Attend’, ‘Maybe’ | **System Response:**  2. Fetches all the invitations of the User  4. Displays the Invite  6. Indicate that response has been recorded |
| Case 16: As a user, I would like to be able to share images of the party. | |
| **Action:**  1. Click on ‘Share Images’  3. Upload images through select a file | **System Response:**  2. Display the share images dialog box  4. Indicates that the Images have been uploaded |
| Case 17: As a user, I would like to be able to design cards for invites. | |
| **Action:**  1. Click on ‘Create Invite’  3. Select a template  4. Modify the Template depending on the party details  5. Submit | **System Response:**  2. Displays all the available templates  4. Displays the Template  6. Indicates that the invite is ready |
| Case 18: As a user, I would like to be able to create a checklist of tasks for the event and assign people to the tasks. | |
| **Action:**  1. Click on ‘Create Checklist’  3. Enter all options  4. Submit | **System Response:**  2. Dialog appears  5. Record response  6. Dialog disappears |
| Case 19: As a user, I would like to be able to ask questions regarding the event to the host. | |
| **Action:**  1. Click on ‘Messages’  3. Select the host  5. Enter the Message  6. Click ‘Send’ | **System Response:**  2. Fetches all the hosts  4. Displays the conversation Dialog box  7. Indicate that the message has been sent |
| Case 20: As a user, I would like to be able to answer the questions of the guest. | |
| **Action:**  1. Click on ‘Messages’  3. Select a Message to be answered  5. Enter the message  6. Click ‘Send’ | **System Response:**  2. Fetches all the messages  4. Displays the conversation box  7. Indicate Message has been sent |
| Case 21: As a user, I would like to get suggestions for the event venue according to the venue budget. | |
| **Action:**  1. Click on Event Planner  3. Click on Budget Limit and input a numeric value in the text area. | **System Response:**  2. A display dialogue box with various planning features.  4. Displays suggestions within the budget. |
| Case 22: As a user, I would like to get suggestions for the catering for the event according to the catering budget. | |
| **Action:**  1. Click on Event Planner  3. Click on Budget Limit and input a numeric value in the text area. | **System Response:**  2. A display dialogue box with various planning features.  4. Displays catering services within the budget. |
| Case 23: As a user, I would like to have a way to connect to the developers for providing feedback | |
| **Action:**  1. Fill the Feedback TextArea and the “Send” button is pressed.  3. Click ‘send’ | **System Response:**  2. When the button has been pressed, the feedback is sent back to the database |
| Case 24: As a Software Developer, I would like to have a continuous development and continuous integration pipeline. | |
| **Action:**  1. Make a Git push of my commits to the remote. | **System Response:**  2. The Circle Ci Takes the commit and Runs all the Unit test cases.  3. The Circle Ci Takes the commit and Runs all the Integration test case.  4. The Circle Ci Takes the commit and Runs The linter.  5. The Circle Ci pushes the commit (if made to master) to the testing domain of the web app. |
| Case 25: As a Software Developer, I would like to have a Docker container for local testing. | |
| **Action:**  1. The developer types “Docker Compose up” in the terminal. | **System Response:**  2. The Docker takes the current code and creates a container.  3. The Docker runs all the test cases (both unit and integration).  4. The Docker opens the front-end Web app and runs the backend locally. |