DateNight Product Backlog

Team 2

Problem Statement

Dating today can be quite a hassle, whether it be figuring out a good place to go eat in your area or finding events that would be fun and memorable but also affordable. Enter DateNight, your helper for figuring out and planning your date night. It allows users to search for date ideas in their area based on criteria that they provide. Currently, there are many ways to research and plan a date, but none of these methods are as streamlined and efficient as DateNight.

Background Information

Relationships are hard, and dating is no exception. Often times planning a date can prove to be just as difficult, if not more difficult, than going on the date itself. There are many factors that tie into planning a date including timing, overall time allotment, traveling distance, overall expenses, and many others that can complicate the process and leave couples unenthusiastic. DateNight aims to alleviate the stresses of this process by helping users find local restaurants, events, and activities that fit their schedule, budget, and interests.

Environment

Our front-end will be designed in-house and implemented using HTML, CSS, and JavaScript. We will be using React as our front-end framework of choice. The back-end will be written in JavaScript running on a Node.js environment with a custom Express API. For our database, we are going to use PostgreSQL. We will also be using external libraries like Google Maps APIs for finding places and displaying a map and Google Auth for authenticating our users.

Functional Requirements

#	Functional Requirement	Hours	Status
1	As a user, I would like to optionally register for an account using Google.	6	Planned for Sprint #
2	As a user, I would like to log into my account if I have one.	5	Planned for Sprint #
3	As a user, I would like to log out of my account if I am	6	Planned for

	logged in.		Sprint #
4	As a user, I would like to be able to enter a location to search within.	3	**Planned for Sprint #1**
5	As a user, I would like to be able to choose a price point to search by.	3	**Planned for Sprint #1**
6	As a user, I would like to be able to choose a type of date to search for.	3	**Planned for Sprint #1**
7	As a user, I would like to be able to choose a time duration minimum/maximum.	3	**Planned for Sprint #1**
8	As a user, I would like to be able to submit my search parameters.	3	**Planned for Sprint #1**
9	As a developer, I would like to have a detailed plan for creating and implementing our date search algorithm and API.	5	**Planned for Sprint #1**
10	As a developer, I would like an algorithm and API to search for date events based on the search parameters.	6	**Planned for Sprint #1**
11	As a user, I would like to view my search results on a Google Maps style map with pins.	5	**Planned for Sprint #1**
12	As a user, I would like to be able to click on a pin to get more information about one of the results.	3	**Planned for Sprint #1**
13	As a user, I would like to be able to use a map pin to redirect to that event's website.	2	**Planned for Sprint #1**
14	As a user, I would like to see movie showtimes if I click on a theater.	4	Planned for Sprint #2
15	As a user, I would like to view my search results in a list-style view.	4	**Planned for Sprint #1**
16	As a user, I would like to be able to see price range for each result in the list-style view.	2	**Planned for Sprint #1**
17	As a user, I would like to be able to see reviews for each result in the list-style view.	2	**Planned for Sprint #1**
18	As a user, I would like to be able to see hours for each result in the list-style view.	2	**Planned for Sprint #1**

19	As a user, I would like to be able to see contact info like a website and phone number for each result in the list-style view.	2	**Planned for Sprint #1**
20	As a user, I would like to be able to sort the list-style view by price range.	1	**Planned for Sprint #1**
21	As a user, I would like to be able to sort based on rating.	1	**Planned for Sprint #1**
22	As a user, I would like to be able to start a date plan.	2	**Planned for Sprint #1**
23	As a user, I would like to be able to add and remove things from a date plan.	2	**Planned for Sprint #1**
24	As a user, I would like to be able to save a date plan to my account (account required).	3	Planned for Sprint #2
25	As a user, I would like to be able to view my previously saved date plans (account required).	3	Planned for Sprint #2
26	As a user, I would like to be able to rate completed plans privately for future reference (account required).	3	Planned for Sprint #2
27	As a user, I would like to be able to delete a date plan (account required).	2	Planned for Sprint #2
28	As a user, I would like to be able to share a date plan via a link.	4	Planned for Sprint #2
29	As a user, I would like to view another person's date plan via a link.	3	Planned for Sprint #2
30	As a user, I would like to be able to submit a location that would make a good date event.	4	Planned for Sprint #2
31	As a user, I would like to be able to view crowdsourced date events on the map.	3	Planned for Sprint #2
32	As a user, I would like to be able to add time to the crowdsourced date events.	2	Planned for Sprint #2

Non-Functional Requirements

Easy Development: To ensure quality development, all developers should have an environment that promotes clean code. Both the front-end and back-end of our

application will be suited with a linter, easy-to-use build and testing scripts, and editor settings.

Usability: One of the biggest factors that go into making a successful web application is the user-friendliness. Navigation should be self-explanatory and the user interface should be simple and clean.

Scalability: Our app will store a lot of information about crowdsourced events in the database. The database needs to be able to organize all of this information easily and traversed efficiently no matter how much data there is. We also need to be able to scale the database storage size up and down to handle a user base of one to a user base of one thousand.

Security: DateNight takes user data very seriously. With an account system in place, we will need to ensure all API endpoints that potentially reveal or modify sensitive data will be properly safeguarded by authentication checks and passwords for accounts should not be stored in plain text. Authentication should be through JWT.

Use Cases

Case 1	**Register an account**
Action	**System Response**
Choose “Log in using Google”	Redirected to Google login screen
Choose account to log in with that is not registered	Redirected to Google sign up screen
Choose account to log in with that is registered	Redirected to DateNight homepage

Case 2	**Login to account**
Action	**System Response**
Click on sign up button in navbar	Route moves you to the login url, and the login page shows up
Type in username and password in corresponding fields and click on the login button	If success, redirect back to the home screen with username showing where login button would be

	If failure, then red text shows up demonstrating failure

Case 3	**Logout of account**
Action	**System Response**
Click on the logout button under the username dropdown	Redirect back to the home screen with login button replacing the username dropdown

Case 4	**Enter a location**
Action	**System Response**
Click on location box	Window focuses on location box
Type in location such as city or zip code	Form has location completed inside it

Case 5	**Choose a price point**
Action	**System Response**
Click on checkbox with desired price point (cheap, moderately priced, expensive)	Appropriate check box is filled in corresponding to user selection

Case 6	**Choose type of date**
Action	**System Response**
Choose a type of date from the dropdown	Field populates with a type of date from the dropdown
As a user, I would like to be able to choose a type of date to search for.	

Case 7	**Choose time duration**
Action	**System Response**
Enter amount of time for the minimum	Minimum amount of time field populates

amount of estimated time a date can take.	with user inputted time.
Enter amount of time for the maximum amount of estimated time a date can take.	Maximum amount of time field populates with user inputted time.

Case 8	**Submit search parameters**
Action	**System Response**
Click on submit button	Frontend sends data that was entered into the form to the backend
	Backend authenticates data and confirms that the data entered was correct

Case 9	**Plan for search API**
Action	**System Response**
Create and standardize plan for API methods so all developers are on the same page	
Create an ER diagram to map out relationships for different data in the databases	

Case 10	**Access search API**
Action	**System Response**
User submits query parameters via UI	Frontend calls search API according to query parameters
	Backend evaluates database and google apis and formulates response
	Frontend receives response and displays information to user

Case 11	**View pins on Google Maps**

Action	**System Response**
User prompts map view of events	Frontend calls google API to display event locations on map as pins
	Frontend calls API to display specific event information

Case 12	**Click on pin**
Action	**System Response**
User selects a pin	Frontend calls backend API to obtain specific event information
	Backend API returns information to frontend and data is displayed to user

Case 13	**Redirect from pin**
Action	**System Response**
Click on the pin	Frontend calls backend API to obtain specific event information
	Backend API returns information to frontend and data is displayed to user
Click on link for the external website	Redirect to website

Case 14	**View movie showtimes**
Action	**System Response**
Click on the pin	Frontend calls API to list the showtimes for that day in a view.

Case 15	**View list of results**
Action	**System Response**
User submits search	System returns search results

	System displays results as map
	System displays results as a list on the side as well

Case 16	**View price range for list item**
Action	**System Response**
User submits search	System returns search results
	System displays results as map
	System displays results as a list on the side as well
User clicks on list item	System displays more info about the list item including price range

Case 17	**View reviews for list item**
Action	**System Response**
User submits search	System returns search results
	System displays results as map
	System displays results as a list on the side as well
User clicks on list item	System displays more info about the list item including reviews

Case 18	**View hours for list item**
Action	**System Response**
User submits search	System returns search results
	System displays results as map
	System displays results as a list on the side as well
User clicks on list item	System displays more info about the list

	item including hours

Case 19	**View contact info for list item**
Action	**System Response**
User submits search	System returns search results
	System displays results as map
	System displays results as a list on the side as well
User clicks on list item	System displays more info about the list item including contact info

Case 20	**Sort list by price**
Action	**System Response**
User clicks on sort button	The button will present sorting options
User clicks on sort button based on price	The results on the page will be sorted accordingly

Case 21	**Sort list by rating**
Action	**System Response**
User clicks on sort button	The button will present sorting options
User clicks on sort button based on rating	The results on the page will be sorted accordingly

Case 22	**Start a date plan**
Action	**System Response**
Click on the plan button in the nav bar	System redirects to the date plan screen

Case 23	**Add and remove events to date plan**
Action	**System Response**
Navigate to a date event	System pulls up the screen for an event
Click on the “add to plan” button	System adds the event to the date plan.
Click on the “remove” button	System removes the event from the plan.

Case 24	**Save date plan to account**
Action	**System Response**
While logged in, click on the plan button in the nav bar	System redirects to the date plan screen
Click on the save button in the screen	System posts to the api to save the plan to the user account. The api saves the plan to the users account in the DB

Case 25	**View saved date plans**
Action	**System Response**
While logged in, click on the account button	System redirects to the account page
Click on the saved plans button	System shows the saved plans

Case 26	**Rate a date plan**
Action	**System Response**
While logged in, click on the account button	System redirects to the account page
Click on the saved plans button	System shows the saved plans
Click on a number of stars on the saved plans	System sends a post to the api. Api saves the information in the database under the account

Case 27	**Delete date plan**
Action	**System Response**
User opens their saved list of date plans	Frontend displays users list of date plans
User selects and deletes an unwanted plan	Frontend calls backend to remove the date plan from the user account
	Frontend indicates success by removing the given date plan from the list

Case 28	**Share date plan**
Action	**System Response**
Click share	Frontend displays link with date plan ID

Case 29	**View other user's date plan**
Action	**System Response**
Click on link	Frontend displays the other user's date plan

Case 30	**Submit location**
Action	**System Response**
User prompts feature via a "add location" style button	Frontend prompts user with a custom interface and input box
User enters location and submits	Frontend calls backend to verify that location exists and is valid
	Backend rejects location data and frontend notifies user
User alters location and submits	Frontend calls backend to verify that location exists and is valid
	Backend accepts location data and frontend notifies user

Case 31	**View locations**
Action	**System Response**
User navigates to the page to view date events on the map	A fetch is made to the API to retrieve the proper information of date events
	The UI displays the date events

Case 32	**Add time to locations**
Action	**System Response**
Click on the event to edit	System pulls up the detail view for the event
Click on the clock icon	System pulls up an input
Type in a time	System posts to the API. API saves the time for the event in the database