Night Owl Developers Documentation

Overview

- Project goal: To build a web application that functions as an “Airbnb for pets”, with features that facilitate interactions between pet owners and pet service providers.
- Application features:
    - Frontend: Service Page, Product Page, Media Page, Create Media Page, Account Page, Detail Service Page, Detail Product Page, Detail Media Page and Cart Page
    - Backend: DB tables setup, APIs for services, products and comments, authentication logic, user session logic, filtering and sorting, inserting and retrieving images, Cart System
- Team organization:
    - Frontend: Frank, Ivan, Jonathan, Meng
    - Backend: Gary, Jesse, Tyler

Frontend:

The frontend is built with React. Libraries used include Bootstrap. Below are the frontend features prioritized in Sprint 3.

Registration page: Users can register an account on the website by providing a username, password, email, and location.

Login page: Users can login to the website by providing their username and password.

Account page: After users are logged in, they can access their accounts page. On this page they can see options to edit their account information such as password, email, and location.

HomePage Search: Users can search for service providers by using Location, price range, pet breed, number of pets.

Media Page: Users can view a list of media posts.

Create Media Page: Users can send post requests to the backend to create media posts.

Product Page: Users can see a list of Product page cards based on selected parameters, filters and sort them accordingly.

Service Page:Users can see a list of Service page cards based on selected parameters, filters and sort them accordingly.

Detail Media Page: Detail Media Page shows the detail description, pictures and comments of the media page.

Detail Product Page: Detail Product Page shows the detail description, pictures and comments of the Product page.

Detail Service Page: Detail Service Page shows the detail description, pictures and comments of the Service page.

Cart Page: Cart Page shows the products and services currently in the user's cart.

Backend:

The backend server logic is built with Express and Node. Other Node modules used include Passport.

PostgreSQL is used as our database management system. Below are the tables currently used by our application:

users

uid	integer
username	varchar(30)
email	varchar(30)
password	varchar(30)
fname	varchar(30)
lname	varchar(30)
city	varchar(30)
phone_number	varchar(30)
avatar	varchar(200)
is_service_provider	boolean

providers

provider_id	integer
provider_name	varchar(200)
provider_phone	varchar(200)
provider_email	varchar(200)
provider_avatar	varchar(200)

services

service_id	integer
service_pic_url	varchar(200)[]
service_title	varchar(200)
service_detail	varchar(200)
service_facility	varchar(200)
location	varchar(200)
price_per_day	decimal
service_rating	decimal
service_pet_breed	varchar(200)
user_id	integer

products

product_id	integer
product_detail	varchar(200)
product_name	varchar(200)

product_origin	varchar(200)
product_category	varchar(200)
product_pet_breed	varchar(200)
product_price	decimal
product_type	varchar(200)
product_pic_url	varchar(200)[]
product_rating	decimal

comments

comment_id	integer
comment_type	varchar(200)
foreign_id	varchar(200)
comment_detail	varchar(200)
author_name	varchar(200)
author_profile_pic_url	varchar(200)
comment_time	varchar(200)

replies

reply_id	integer
cid	integer
reply_username	varchar(200)
reply_avatar_url	varchar(200)
reply_detail	varchar(200)
reply_time	varchar(200)

mediaPages

id	integer
author_id	integer
media_picture_irl	varchar(200)[]
media_title	varchar(50)
media_detail	varchar(200)
published_time	varchar(50)
number_of_likes	integer

product_cart_items

id	integer
user_id	integer
product_id	integer
quantity	integer
size	varchar

service_cart_items

id	integer
user_id	integer
service_id	integer
start_date	varchar(200)
end_date	varchar(200)
number_of_pets	integer

images

image_id	integer

image_name	varchar(200)
image_data	bytea

DB API

Server location: http://localhost:3001

User API

API Method	Description	Request Body Format
GET /api/user	Gets all rows of users from the database.	
GET /api/user/:uid	Gets the user with the given uid.	

POST /api/user	Create and insert a new user into the database.	{
		username: string,
		email: string,
		password: string,
		fname: string,
		lname: string,
		city: string,
		phone_number: int
		}

PUT /api/user/:uid	Edit the user's password, email, or location fields. Username cannot be changed. At least one of these fields must be in the request body.	{
		password:string,
		email: string,
		city: string,
		phone_number: int,
		avatar: string,
		Is_service_provider: bool
		}

Auth API

API Method	Description	Request Body Format
POST	Login as a user using the login	{

/api/auth/login	credentials.	username: string, password: string }
POST /api/auth/logout	Log out of the user's account.	
GET /api/auth/user	Check if current user is logged in, and if so, get the user details	

Providers API

API Method	**Description**	**Request Body Format**
GET /api/providers	Gets all rows of providers from the database.	
GET /api/providers/:provider_id	Get the provider with the given provider_id	

POST /api/providers	Create and insert a new provider into the database	{
		"provider_name":"value",
		"provider_phone":"value",
		"provider_email":"value",
		"provider_avatar":"value"
		}

PUT /api/providers/:provider_id	Edit the provider's information. Must contain at least one of the fields to edit.	{
		"provider_email":"value"
		}

DELETE /api/providers/:provider_id	Removes the given provider from the database.	

Services API

API Method	**Description**	**Request Body Format**
GET /api/services	Gets all rows of services from the database. Allows users to filter and sort on specific	Query params: - locations - pet_breeds

	columns using query params.	- minPrice - maxPrice - sortBy - sortDirection
GET /api/services/:service_id	Get the service with the given service_id	
GET /api/services/for_user/:user_id	Get all the services for the given user.	

POST /api/services	Create and insert a new service into the database	{
		“service_pic_url”:[”value1”, “value2”],
		“service_title”:”value”,
		“service_detail”:”value”,
		“service_facility”:[”value1”, “value2”]
		“location”:”value”,
		“price_per_day”:”value” ,
		“service_rating”:”value”,
		“service_pet_breed”:”value”,
		“user_id”:int // provider u id
		}

PUT /api/services/:service_id	Edits the service’s information. Must contain at least one of the fields to edit.	{
		“service_detail”:”value”
		}

DELETE /api/services/:service_id	Removes the given service from the database.	

Products API

API Method	**Description**	**Request Body Format**
GET /api/products	Gets all rows of products from the database. Allows users to filter and sort on specific columns using query params.	Query params: - categories - pet_breeds - minPrice - maxPrice - sortBy

		- sortdirection
GET /api/products/:product_id	Gets the product with the given product_id from the database.	

POST /api/products	Create and insert a new product into the database	{
		“product_detail”:”value”,
		“product_name”:”value”,
		“product_origin”:”value”,
		“product_price”:”value”,
		“product_type”:”value”,
		“product_pic_url”:”value”,
		“product_rating”:”value”
		}

PUT /api/products/product_id	Edits the product’s information. Must contain at least one of the fields to edit.	{
		“product_rating”:”value”
		}

DELETE /api/products/product_id	Deletes the product with the given product_id from the database.	

Comments API

API Method	**Description**	**Request Body Format**
GET /api/comments	Gets all comments belonging to a service, product, or media. comment_type can be [“product”, “service”, “media”] and foreign_id is an integer. For example, the request body to the right would get all comments belonging to the product with product_id of 5.	Query params: “comment_type”:”product”, “foreign_id”:5
POST /api/comments	Creates an inserts a new comment into the database	{ “comment_type”:”value”, “foreign_id”:”value”, “comment_detail”:”value”,

		“author_name”:”value”, “author_profile_pic_url”:”value”, “comment_time”:”value” }

PUT /api/comments/:comment_id	Edits the comment’s information. At least one field must be in the request body.	{
		“comment_detail”:”value”
		}

DELETE /api/comments/:comment_id	Deletes the comment with the given comment_id from the database.	

Replies API

API Method	**Description**	**Request Body Format**
GET /api/replies	Gets replies belonging to a comment. If cid is specified in query params, only replies for that comment will be returned. If nothing is specified, all replies are returned.	ie. GET /api/replies?cid=2
GET /api/replies/:reply_id	Gets the reply with the given reply_id from the database.	

POST /api/replies	Create and insert a new reply into the database.	{
		“cid”:int,
		“reply_username”:”user1”,
		“reply_avatar_url”:”url1.com”,
		“reply_detail”:”string”,
		“reply_time”:”2021-10-20 2:00PM”
		}

PUT /api/replies/:reply_id	Edit the reply’s information. At least one field must be in the request body.	{
		“reply_detail”:”string2”
		}

DELETE /api/replies/:repl	Deletes the reply with the given reply_id from the	

y_id	database.	

Media Pages API

API Method	**Description**	**Request Body Format**
GET /api/mediapages	Gets all rows of mediapages from the database.	
GET /api/mediapages/:id	Gets the media page with the given id.	
GET /api/mediapages/for_user/:user_id	Get the media pages for the given user.	

POST /api/mediapage	Create and insert a new mediapage into the database.	{
		author_id: int,
		media_picture_url: string,
		media_title: string,
		media_detail: string,
		published_time: string,
		number_of_likes: int
		}

PUT /api/mediapage/:id	Add a like to a mediapage in the database.	
DELETE /api/mediapage/:id	Delete a mediapage from the database.	

Cart API

API Method	**Description**	**Request Body Format**
GET /api/cart	Get all of the products and services in the currently logged-in user's cart.	

POST /api/cart/product	Add a product to the currently logged-in user's cart.	{
		product_id: int,
		quantity: int,
		size: string
		}

POST /api/cart/service	Add a service to the currently logged-in user's cart.	{
		service_id: int,
		start_date: string,
		end_date: string,
		number_of_pets: int
		}

DELETE /api/cart/product/:id	Delete a product from the currently logged-in user's cart.	
DELETE /api/cart/service/:id	Delete a service from the currently logged-in user's cart.	
DELETE /api/cart	Empty the currently logged-in user's cart	

Images API

API Method	**Description**	**Request Body Format**
GET /api/images	Gets the image with the specified name.	Query params must contain the key "image_name" with the value being the image's filename. (ie. corki.jpg)
GET /api/images/:image_id	Gets the image with the given id.	
POST /api/images	Insert an image into the database.	Must use form-data. KEY: img VALUE: file upload
DELETE/api/images/:image_id	Deletes the image with the given image_id from the	

	database.	

Payment API

POST /api/payment	Files a paypal payment request for the sum of total and redirects back to redirect.	{
		_total: int
		redirect: string
		}

Scripts

imageloader

Added a script to load a folder of images into the db. All images in the folder will be uploaded. The folder's location is /server/db/sample_images

Server must be running prior to running the script since it uses the server endpoints to insert images.

Prior to usage run:
`npm install --save form-data`

Usage Steps

1. run `npm run dev`
2. Start a new terminal, since the current one is running the server

3. run `npm run imageloader`