USER SETUP/ SET UP STEPS	Notes	completion
Setup Steps		
1. Download and extract the zip file from the starter code repo, open in VS Code.	flask tutorial 1	done
2. Create your own Github repo with a Python gitignore and your own README, then push the code up.	flask tutorial 1; need to come back to, initially left off readme since one came with starter files	
3. Create a MySQL Database with an appropriate name for the project.	flask tutorial 1	done
4. Change the username, password and db_name in the .env file to your MySQL username/password and the database you just created.	done: note, make sure that password is typed in without the <>, this held up last time.	done
NOTE: Some special characters in the password (especially @ and /) must be url escaped. For a @, use %40. For a /, use %2F	none	done
5. Create your virtual environment with terminal commands:	flask tutorial 1	done
pipenv install	flask tutorial 1	done
pipenv shell	flask tutorial 1	done
6. Once the venv is created, you can start the app at any time with flask run	done. NOTE::: make sure this is active	
7. Create your database model(s) in app.py with the required properties, then run:	done	
flask db init (Creates tables)	done	
flask db migrate -m "Init" (Creates migration)	done-note-- if things don't migrate correctly, note any needs for saves.	
flask db upgrade (Runs migration)	done	
8. Create Marshmallow Schema in app.py	flask tutorial 3	
Continue on to create Resource classes and Routes, making sure to test each endpoint in Postma		

TASKS/ USER STORIES	Notes	completion
(5 points): As a developer, I want to make good, consistent commits.	**in progress**	
(2.5 points): As a developer, I want to create an Entity Relationship Diagram that will accurately show the necessary properties on the Product model. Be sure to include a screenshot of your ERD in your GitHub repository!	**done: note- bigint for primary key vs int as specified. May need to get clarity if this matters.**	
(5 points): As a developer, I want to build a REST web API in Flask, so that I can make HTTP requests interact with the data set.	**Once the parameters of the database are defined in flask,then create transactions in Postman, effects should be reflected in MySQL.**	

2.5 points): As a developer, I want to create a Product model
Property names must be in snake_case and match the following exactly!
id - Integer
name - String
description - String
price - Float
inventory_quantity - Integer
	class Product(db.Modeclass Product(db.Model):
	id = db.Column(db.Integer, primary_key=True)
	name = db.Column(db.String(255), nullable=False)
	description = db.Column(db.String(255), nullable=False)
	price = db.Column(db.Float)
	inventory_quantity = db.Column(db.Integer)
	
	def __repr__(self):
	return f'{self.id} {self.name} {self.description} {self.price} {self.inventory_quantity} '	

(5 points): As a developer, I want my API to serve content on the following url paths: Paths must match these exactly! '127.0.0.1:5000/api/products/' '127.0.0.1:5000/api/products/<int:pk>'	Flask Tutorial 4: This http is from flask. Must run flask. Then can send requests from Postman. It will show in flask if a connection was made. Products is the main table, where int is used as the identifier such as id.	
(5 points): As a developer, I want to create a GET endpoint that responds with a 200 success status code and all of the products within the Product table.	Flask Tutorial 4: Get All Products in postman will get all in the table. In postman, it shows a status of 200 OK,	
(5 points): As a developer, I want to create a GET by id endpoint that does the following things: Accepts a value from the request's URL (The id of the product to retrieve). Returns a 200 status code. (Explicitly return this, not just allow it to default) Responds with the product in the database that has the id that was sent through the URL.	Flask Tutorial 6: IN postman api/products/7 - the seven would be the item number. Example of this query requested 3 which was the wireless mouse	
(5 points): As a developer, I want to create a POST endpoint that does the following things: Accepts a body object from the request in the form of a Product model. Adds the new product to the database. Returns a 201 status code. Responds with the newly created product object.	Flask Tutorial 5: need to create post function that links flask to mysql using postman -- added item 7, the red drinking apparatus.. note, must have flask side set up before trying to post	
(5 points): As a developer, I want to create a PUT endpoint that does the following things: Accepts a value from the request's URL (The id of the product to be updated). Accepts a body object from the request in the form of a Product model. Finds the product in the Product table and updates that product with the properties that were sent in the request's body. Returns a 200 status code. (Explicitly return this, not just allow it to default) Responds with the newly updated product object.	Flask Tutorial 6? For this example changed price of colored pencils. Last put was changing the price to 8.59.	

(5 points): As a developer, I want to create a DELETE endpoint that does the following things: cepts a value from the request's URL. Deletes the Product from the database. Returns a 204 status code (NO CONTENT).	**Flask Tutorial 6: in this example deleted item id 3; it shows the 204 content. However, if I run a get all again, it is still there….. After working through this and help ticket. I added a db.session.commit(). And it worked.**	
(5 points): As a developer, I want to use Postman to make a POST, PUT, DELETE, and both GET requests (get by id and get all) request to my REST web API, save it to a collection, and then export it as a JSON from Postman. Be sure to include the exported JSON file in your project folder and push it to GitHub	**All of the code that is done in flask(VS CODE) will get activated through Postman where results show up in MySQL; this had its own class ProductResource**	
Bonus Stories (5 points): As a developer, I want to add the ability to add an image link to each product. (Link to picture on the internet, this column will just be a simple String representing the URL of the image, you do NOT need to add an actual image file.)	**will come back to if time before next project**	**done**

CHECKLIST	**Notes**	**completion**

END RESULT	**Notes**	**completion**
The result of your Products API backend will be the execution of requests made in Postman. You must test your Products API by executing each request you create in Postman.	Done.	