Service Layer API EndPoints Documentation

By: Peter Tran

Title	Create User
URL	/user
Method	POST
URL Params	No parameters

Data Params	{
	"profileType": "string",
	"firstName": "string",
	"lastName": "string",
	"email": "string",
	"password": "string"
	}

Success Response	Code: 200 Content: { “id” : 12 }
Error Response	Code: 422 Un-processable Entry Content: { error : "Email invalid" }

Sample Call	$.ajax({
	url: "/users",
	dataType: "json",
	data : {
	profileType : “customer”,
	firstName : "Peter",
	lastName : "Tran",
	email : "abc@abc.com",
	password : "abc"
	}
	},
	type : "POST",
	success : function(r) {
	console.log(r);
	}
	});

Notes	This will create a new user in the profile table when users register using the web form. It will return their newly created profileId that the database will increment up so that the website can auto-log into their profile.

Title	Show All Orders
URL	/orders/{ profileId }
Method	GET
URL Params	Required: profileId = [integer]
Data Params	No parameters

Success Response	Code: 200
	Content:
	[
	{
	"orderId": 1,
	"date": “2019-07-11T18:25:43.511Z”,
	"total": 13.75
	},
	{
	"orderId": 2,
	"date": “2019-07-12T18:25:43.511Z”,
	"total": 11.63
	}
	]

Error Response	Code: 404 NOT FOUND Content: { error : "User doesn't exist" }

Sample Call	$.ajax({
	url: "/orders/13",
	dataType: "json",
	type : "GET",
	success : function(r) {
	console.log(r);
	}
	});

Notes	This will provide a list of past orders when customers log into their profile.

Title	Get the Address
URL	/address/{ addressId }
Method	GET
URL Params	Required: addressId = [integer]
Data Params	No parameters

Success Response	Code: 200
	Content:
	{
	"FirstName": “Peter”,
	"LastName": “Tran”,
	"Address1": “123 Fake Street”,
	"Address2": “Apt. 1013”,
	"City": “St. Louis”,
	"State": “MO”,
	"ZipCode": “12345”
	}

Error Response	Code: 404 NOT FOUND Content: { error : "Address doesn't exist" }

Sample Call	$.ajax({
	url: "/address/13",
	dataType: "json",
	type : "GET",
	success : function(r) {
	console.log(r);
	}
	});

Notes	This will return a specific address when customers log into their profile. The website will make two GET calls, one for their billing address and the other for their delivery address.

Title	Update the Address
URL	/address/{ addressId }
Method	PUT
URL Params	Required: addressId = [integer]

Data Params	{
	"FirstName": “string”,
	"LastName": “string”,
	"Address1": “string”,
	"Address2": “string”,
	"City": “string”,
	"State": “string”,
	"ZipCode": “string”
	}

Success Response	Code: 200 Content: { "message": “Address Updated” }
Error Response	Code: 404 NOT FOUND Content: { error : "Address doesn't exist" } Code: 422 Un-processable Entry Content: { error : "Address invalid" }

Sample Call	$.ajax({
	url: "/users",
	dataType: "json",
	data : {
	"FirstName": “Peter”,
	"LastName": “Tran”,
	"Address1": “123 Fake Street”,
	"Address2": “Apt. 1013”,
	"City": “St. Louis”,
	"State": “MO”,
	"ZipCode": “12345”
	}
	},
	type : "PUT",
	success : function(r) {
	console.log(r);
	}
	});

Notes	This will update a specific address in the database when customers log into their profile and click the update address button on the address card.

Title	Log into Profile
URL	/profile
Method	GET
URL Params	No parameters

Data Params	{
	"email": “string”,
	"password": “string”
	}

Success Response	Code: 200
	Content:
	{
	"ProfileId": “123”,
	"FirstName": “Peter”,
	"LastName": “Tran”,
	"BillingAddressId": 123,
	"DeliveryAddressId": 123,
	"RestaurantId": 123,
	}

Error Response	Code: 404 NOT FOUND Content: { error : "User doesn't exist" } Code: 400 Bad Request Content: { error : " Invalid username/password supplied" }

Sample Call	$.ajax({
	url: "/users",
	dataType: "json",
	data : {
	email : "abc@abc.com",
	password : "abc"
	}
	},
	type : "GET",
	success : function(r) {
	console.log(r);
	}
	});

Notes	Users will log into their profile using email and password. If successful, they will be returned profile information so that their profile can be loaded.

Title	Get the Order Details
URL	Profile/{ profileId }/order/details/{ OrderId }
Method	GET
URL Params	Required: OrderId = [integer] Optional: ProfileId – [integer]
Data Params	No parameters

Success Response	Code: 200
	Content:
	[
	{
	"FoodId": 1,
	"Price": 5.99,
	"CustomizedOrder": “No onions, no sauce”
	},
	{
	"FoodId": 2,
	"Price": 4.69,
	"CustomizedOrder": null
	}
	]

Error Response	Code: 404 NOT FOUND Content: { error : "OrderId doesn't exist" }

Sample Call	$.ajax({
	url: "/address/13",
	dataType: "json",
	type : "GET",
	success : function(r) {
	console.log(r);
	}
	});

Notes	On the user profile page, customers will have an order history page where they will be able to click to drill down to the details of the order. This GET method will query the database for a specific orderId and return all contents of the order.

Title	Create Order
URL	/order
Method	POST
URL Params	No parameters

Data Params	{
	"ProfileId": "integer",
	"RestaurantId": "integer",
	"Date": "string",
	"Total": "float",
	"Status": "string"
	}

Success Response	Code: 200 Content: { “OrderId” : 12 }
Error Response	Code: 422 Un-processable Entry Content: { error : "Invalid Parameters" }

Sample Call	$.ajax({
	url: "/users",
	dataType: "json",
	data : {
	ProfileId: 12,
	RestaurantId: 13,
	Date: "2019-07-11T18:25:43.511Z ",
	Total: 13.76",
	Status: "Restaurant Queue"
	}
	},
	type : "POST",
	success : function(r) {
	console.log(r);
	}
	});

Notes	When a customer submits their order, an entry will be posted to the orders table to keep record and track of the order. The database will increment up on OrderId’s and will return it.

Title	Get the Order Status
URL	/order/status/{ OrderId }
Method	GET
URL Params	Required: OrderId = [integer]
Data Params	No parameters

Success Response	Code: 200
	Content:
	{
	"ProfileId": 1,
	"RestaurantId": 3,
	"SubTotal": 7.99
	"Tax": 1
	"DeliveryFee": 2
	"Total": “10.99
	"EstimateDeliveryTime": 15
	"Status": “Delivery Queue”
	}

Error Response	Code: 404 NOT FOUND Content: { error : "OrderId doesn't exist" }

Sample Call	$.ajax({
	url: "/address/13",
	dataType: "json",
	type : "GET",
	success : function(r) {
	console.log(r);
	}
	});

Notes	Multiple pages will be utilizing this call. When the user creates a new order, they will be given an order confirmation with details from this call. The Restaurants and delivery staff will also have their dashboards that will utilize this API call to obtain the status of the orders in their queue.

Title	Update the Order Status
URL	/order/status/{ orderID }
Method	PUT
URL Params	Required: orderId = [string]

Data Params	{
	"Status": "Waiting for Delivery"
	}

Success Response	Code: 200 Content: { "message": “Order Status Updated” }
Error Response	Code: 404 NOT FOUND Content: { error : "Order doesn't exist" } Code: 422 Un-processable Entry Content: { error : "Order Status invalid" }

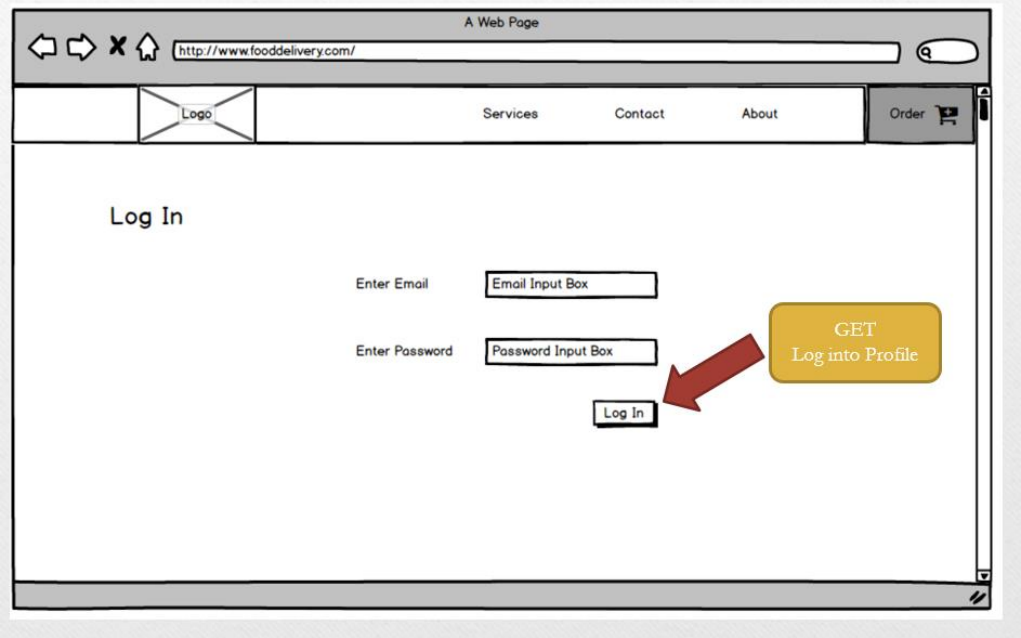
Sample Call	$.ajax({
	url: "/users",
	dataType: "json",
	data : {
	"Status": "Waiting for Delivery"
	}
	},
	type : "PUT",
	success : function(r) {
	console.log(r);
	}
	});

Notes	As an order flows through the process, the order status will be updated periodically by the Restaurant owners and Delivery staff as they mark their parts complete.

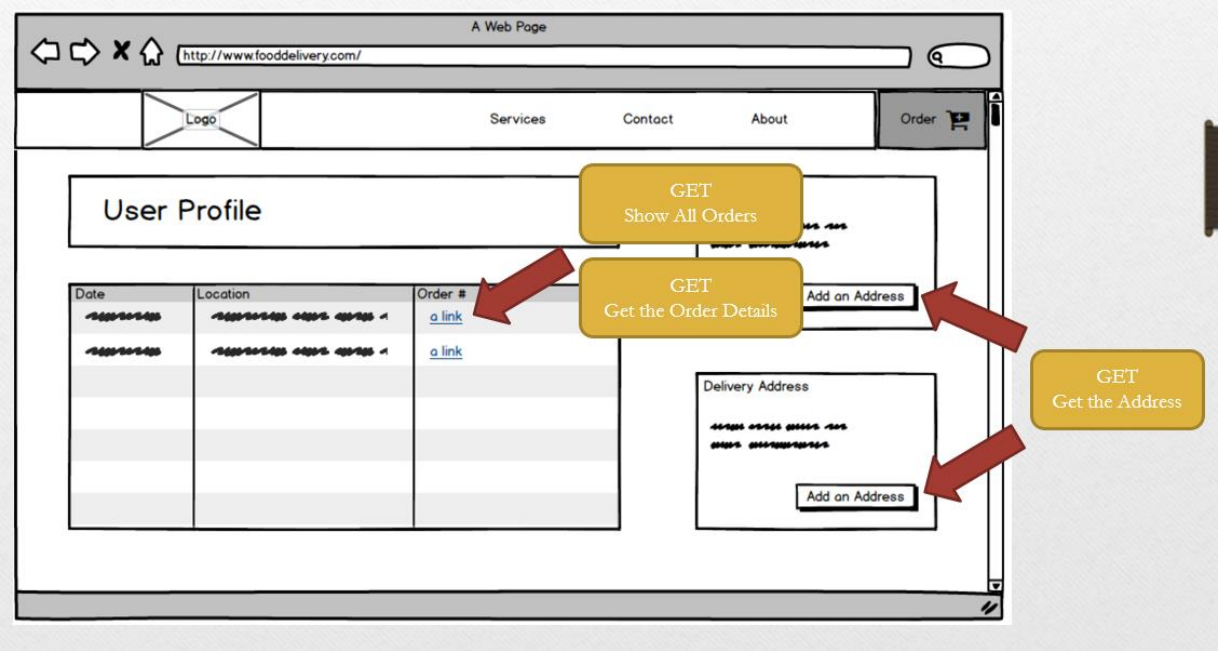
UI Interactions with API Endpoints:

The following images represents the user interface and how it interacts with the back end service layer API endpoints.

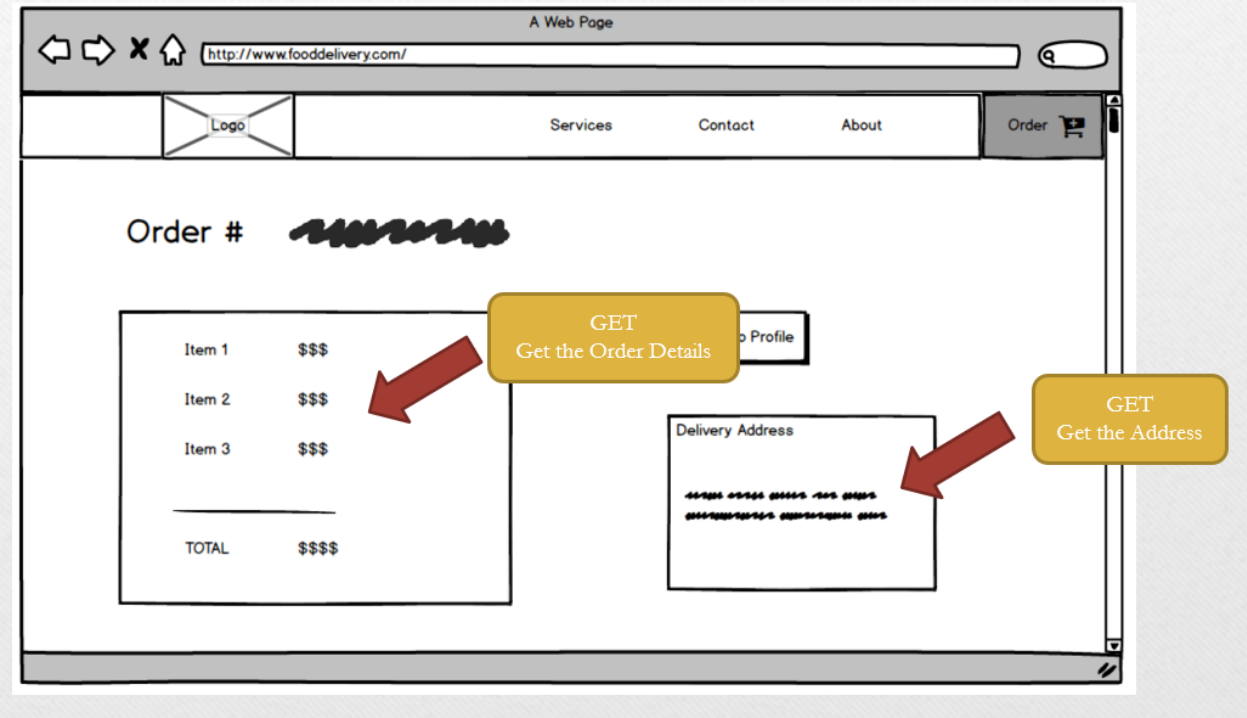
Customer Login Screen



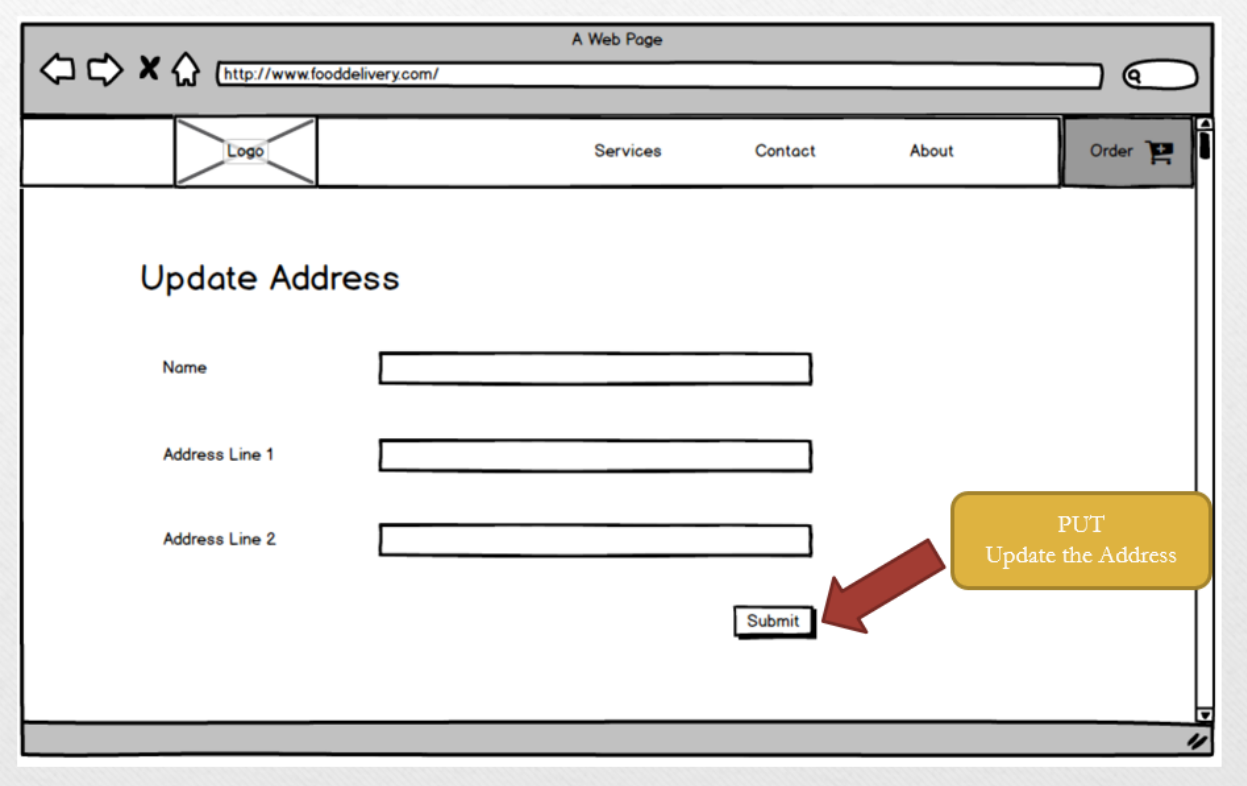
User Profile Page



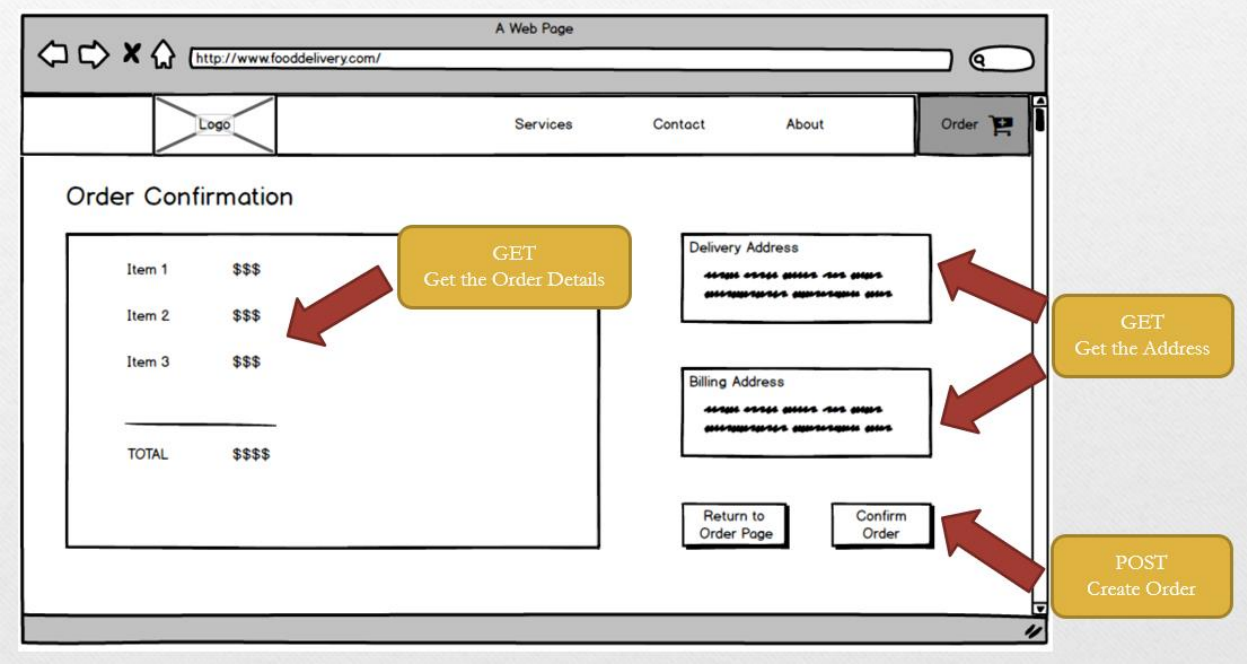
Past Order Details Page



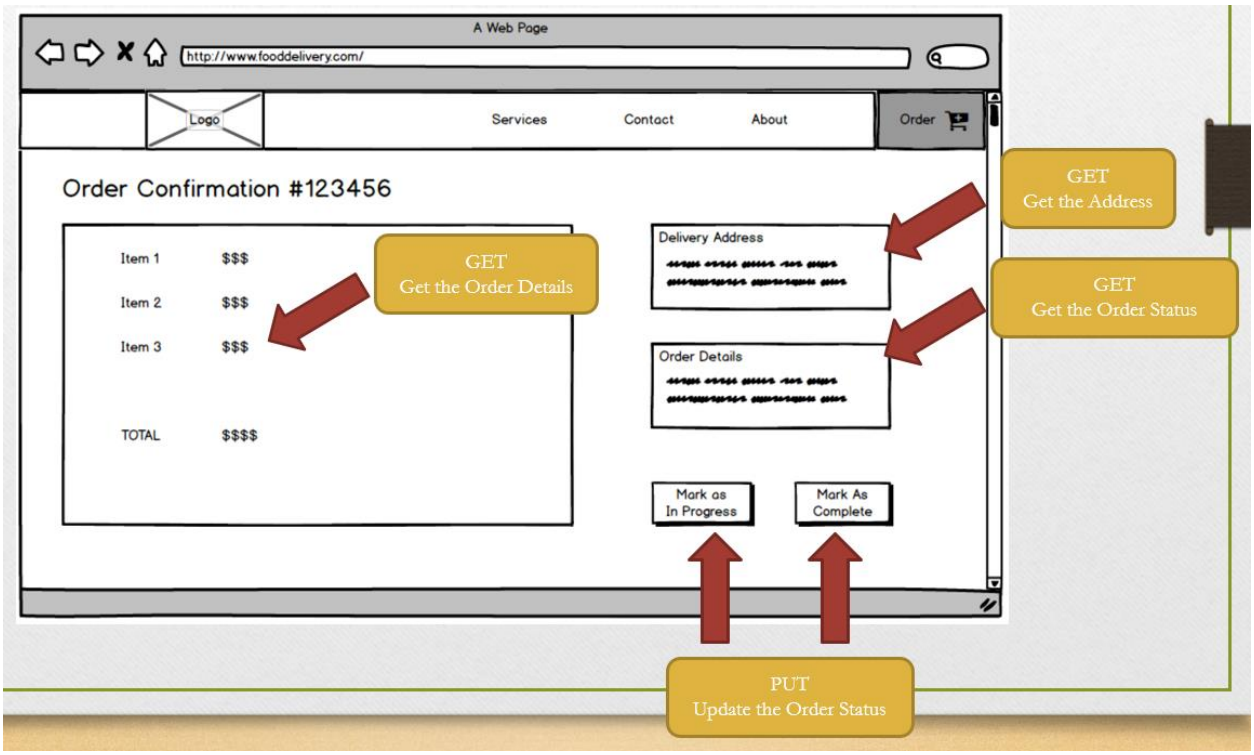
Update Address Web Form



Order Confirmation Page



Restaurant Dashboard Page



Delivery Staff Dashboard Page