

# FULL STACK



## Setting the Project Structure and Sync with GitHub

# You Already Know

Before we begin, let's see what we have covered till now:

- Agile
- Git
- SQL
- HTML or CSS
- JavaScript
- Angular



## Agile

An iterative approach to manage the development of a software project

## Git and GitHub

A distributed version control system that helps handle software projects

## SQL or MySQL

Relational Database Management System to store data in a structured way using tables



## HTML or CSS

Design interactive web pages

## JavaScript

Programming language for the web pages

## Angular

A platform and framework by Google to create single page web applications using HTML and TypeScript



# A Day in the Life of an Automation Test Engineer

As an Automation Test Engineer, our key role is to test both client and server software with the latest test automation tools.

We shall be testing food delivery application built in Angular, Node as the front end with Spring Boot, Java, and MySQL/MongoDB as the backend.

We will clone the developed Angular and Java projects and sync them to our Git repo. Further, we will create the database in MySQL and necessary tables required for the project to run.





# Learning Objectives

By the end of this lesson, you will be able to:

- 👁️ Clone the code of Angular and Java Projects using Git on GitHub
- 👁️ Push the code of Angular and Java Projects using Git on GitHub
- 👁️ Create a database in MySQL
- 👁️ Create tables required for the project in MySQL



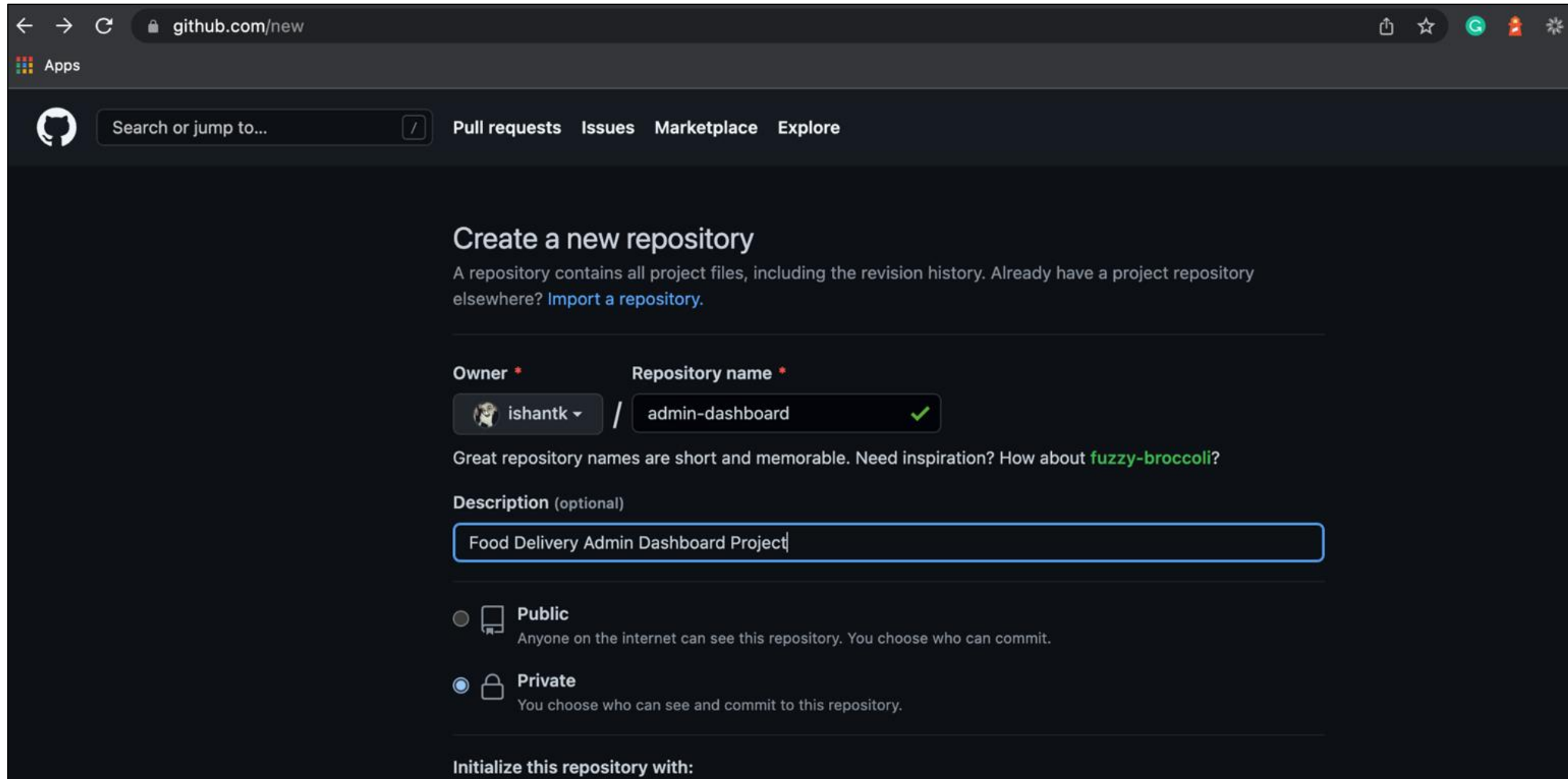
# FULL STACK

## Task 1: Sync Angular Admin DashBoard



# Create a New Repository on Github

Log in to your GitHub account and create a new repository for Admin Dashboard Project:



The screenshot shows the GitHub 'Create a new repository' page. The browser address bar displays 'github.com/new'. The page header includes the GitHub logo, a search bar, and navigation links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main content area is titled 'Create a new repository' and includes a sub-header explaining that a repository contains all project files and revision history. Below this, there are two input fields: 'Owner' (set to 'ishantk') and 'Repository name' (set to 'admin-dashboard' with a green checkmark). A suggestion for 'fuzzy-broccoli?' is shown below the repository name field. The 'Description (optional)' field contains the text 'Food Delivery Admin Dashboard Project'. At the bottom, there are two radio button options for repository visibility: 'Public' (unselected) and 'Private' (selected). The 'Private' option is described as 'You choose who can see and commit to this repository.'.

github.com/new

Apps

Search or jump to... / Pull requests Issues Marketplace Explore

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner \* Repository name \*

ishantk / admin-dashboard ✓

Great repository names are short and memorable. Need inspiration? How about [fuzzy-broccoli?](#)

Description (optional)

Food Delivery Admin Dashboard Project

☐ Public  
Anyone on the internet can see this repository. You choose who can commit.

☒ Private  
You choose who can see and commit to this repository.

Initialize this repository with:



# Git Clone Command

Simply navigate to any of your working directory on your terminal or shell window

Type the following command to clone the project from the repository:

- **git clone https://github.com/username/foodinc-admin-dashboard.git**

Change directory to foodinc-admin-dashboard using the command:

- **cd foodinc-admin-dashboard**

Check the files in the project on the master branch:

- **ls**

You will see the given file structure:

<b>README.md</b>	<b>package-lock.json</b>	<b>tsconfig.app.json</b>
<b>angular.json</b>	<b>package.json</b>	<b>tsconfig.json</b>
<b>karma.conf.js</b>	<b>src</b>	<b>tsconfig.spec.json</b>



# Git Remote Command

---

Rename the local repository current origin to upstream:

**git remote rename origin upstream**

Add the remote origin:

**git remote add origin <https://github.com/username/admin-dashboard.git>**



# Git Push Command

---

Finally, push the code to the Github account. Type the following command:

**git push origin master**

This command will update remote references using local ones while sending objects that are necessary to complete the given references.



# FULL STACK

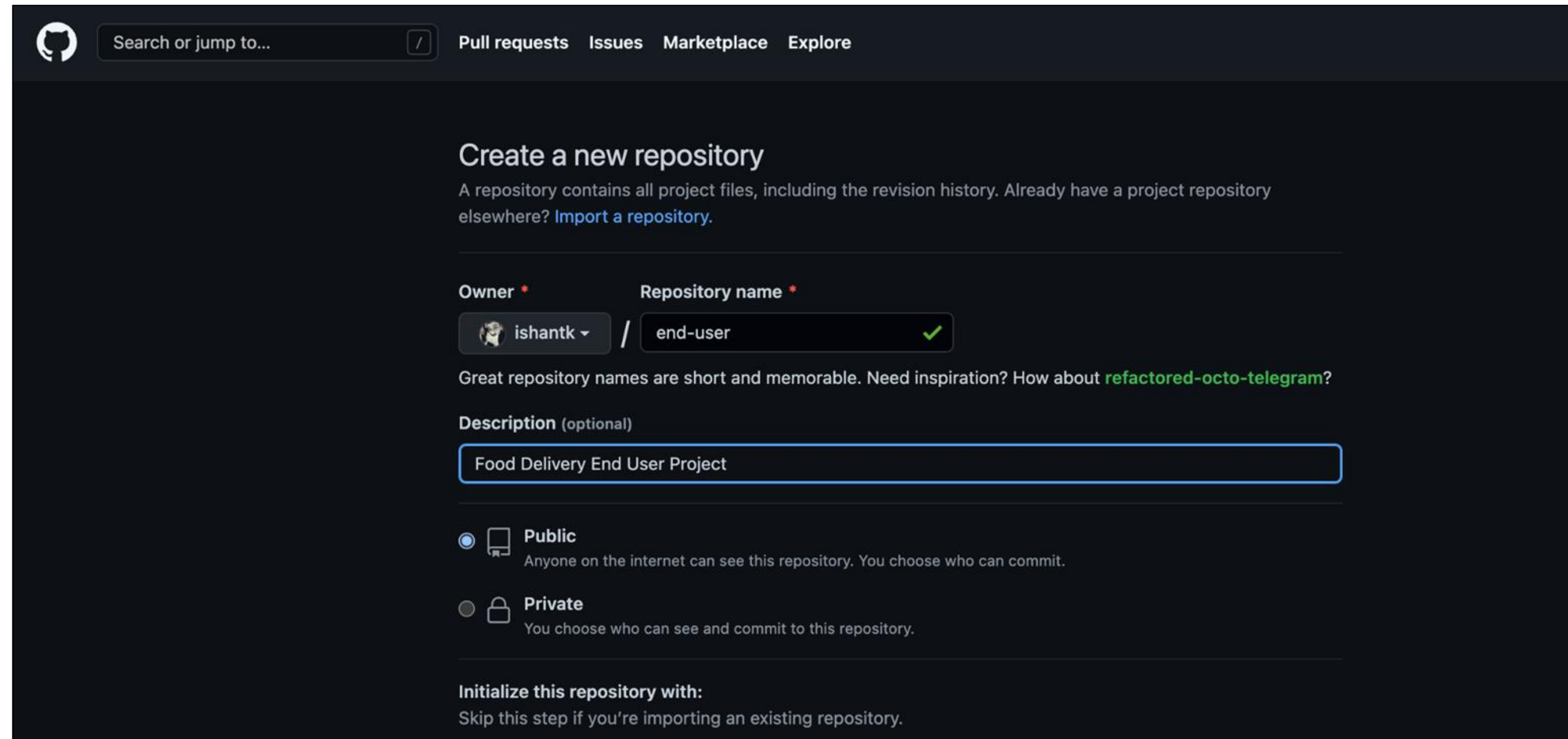
## Task 2: Sync Angular End User Web App





# Create New Repo on GitHub

Log in to your GitHub account and create a new Repository for end user web app project:



The screenshot shows the GitHub 'Create a new repository' page. The header includes the GitHub logo, a search bar, and navigation links for Pull requests, Issues, Marketplace, and Explore. The main heading is 'Create a new repository', followed by a subtext explaining that a repository contains all project files and revision history, with a link to 'Import a repository' if one already exists elsewhere.

The form fields are as follows:

- Owner:** A dropdown menu showing 'ishantk' with a small profile picture icon.
- Repository name:** A text input field containing 'end-user', marked as valid with a green checkmark.
- Description (optional):** A text input field containing 'Food Delivery End User Project'.

Below the description field, there are two radio button options for repository visibility:

- Public:** Selected by default. Description: 'Anyone on the internet can see this repository. You choose who can commit.'
- Private:** Description: 'You choose who can see and commit to this repository.'

At the bottom, there is a section titled 'Initialize this repository with:' with the instruction 'Skip this step if you're importing an existing repository.'

# Git Clone Command

Simply navigate to any of your working directory on your terminal or shell window

Type the command to clone the project from the repository:

**git clone https://github.com/username/foodinc-end-user.git**

Change the directory to foodinc-end-user:

**cd foodinc-end-user**

Check the files in the project on the master branch:

**ls**

You will see the following file structure:

<b>README.md</b>	<b>package-lock.json</b>	<b>tsconfig.app.json</b>
<b>angular.json</b>	<b>package.json</b>	<b>tsconfig.json</b>
<b> karma.conf.js</b>	<b>src</b>	<b>tsconfig.spec.json</b>



# Git Remote Command

---

Rename the local repository current origin to upstream:

**git remote rename origin upstream**

Add the remote origin:

**git remote add origin <https://github.com/username/end-user.git>**



# Git Push Command

---

Finally push the code to github account. Type the following command:

**git push origin master**

This command will update remote references using local ones while sending objects necessary to complete the given references.





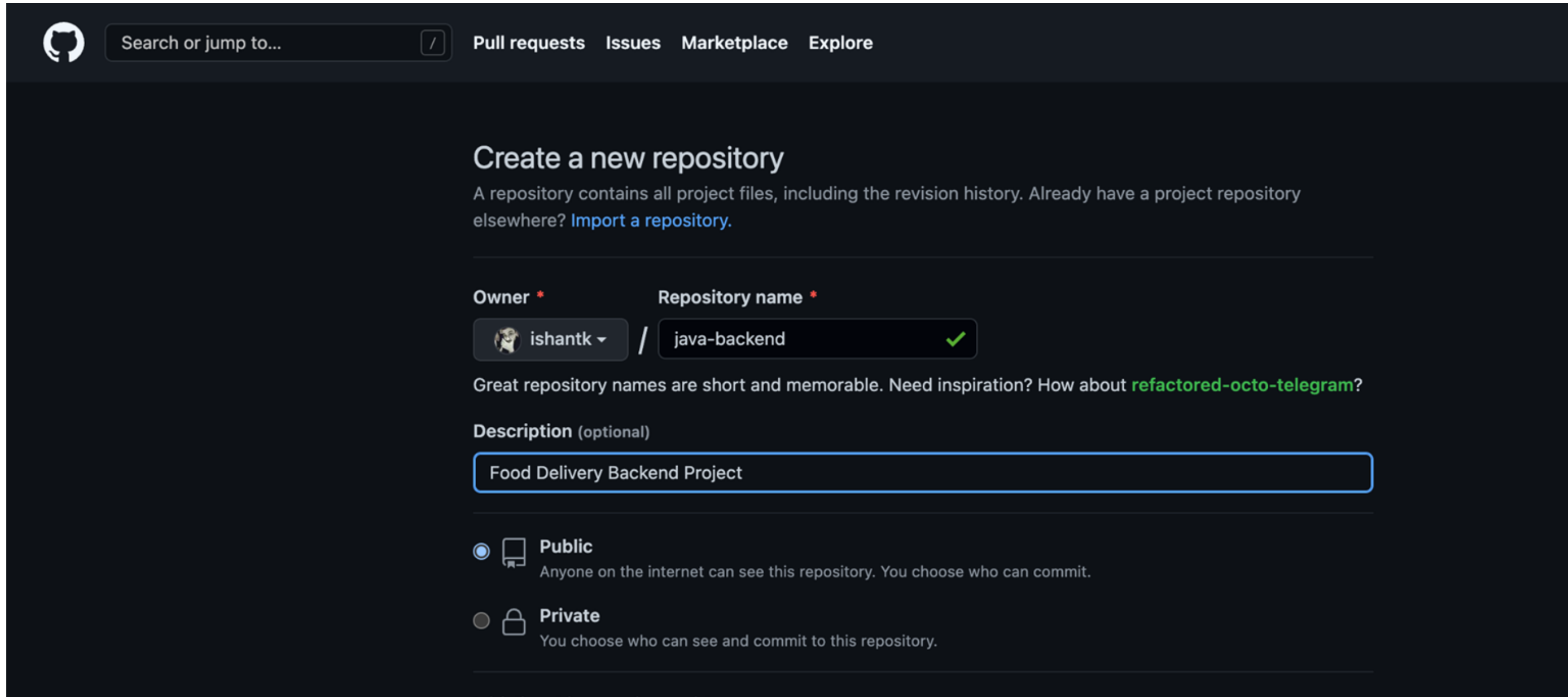
# FULL STACK

## Task 3: Sync Java Backend



# Create a New Repository on GitHub

Log in to your GitHub account and create a new Repository for end user web app project:



The screenshot shows the GitHub 'Create a new repository' page. At the top, there's a navigation bar with the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main heading is 'Create a new repository', followed by a subtext: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'

The form fields are as follows:

- Owner \***: A dropdown menu showing 'ishantk' with a small profile picture icon.
- Repository name \***: A text input field containing 'java-backend' with a green checkmark icon to its right.
- Description (optional)**: A text input field containing 'Food Delivery Backend Project'.
- Visibility**: Two radio buttons are present. The 'Public' option is selected, indicated by a blue dot. Below it, the text says 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option is unselected, indicated by a grey dot. Below it, the text says 'You choose who can see and commit to this repository.'

# Git Clone Command

---

Simply navigate to any of your working directory on you terminal or shell window

Type the following command to clone the project from the repository:

**git clone https://github.com/username/foodinc-java-backend.git**

Change the directory to foodinc-java-backend:

**cd foodinc-java-backend**

Check the files in the project on the master branch:

**ls**

You will see the following file structure:

**mvnw            pom.xml**

**mvnw.cmd    src**



# Git Remote Command

---

Rename the local repository current origin to upstream:

**git remote rename origin upstream**

Add the remote origin:

**git remote add origin <https://github.com/username/java-backend.git>**





# Git Push Command

---

Finally, push the code to GitHub account. Type the given command:

**git push origin master**

This command will update remote references using local ones while sending objects necessary to complete the given references.



## Task 4: Set up the Database Structure



# Creating DataBase in MySQL

In MySQL CLI, use the given command to create and work with database:

## **create database foodie;**

- This command creates the database foodie

## **use foodie;**

- This command changes the current selection of the database to foodie.

## **show tables;**

- This command will list all the tables in the database. As of now, no Table is available so it will return an empty set.



# FULL STACK

## Creating Tables

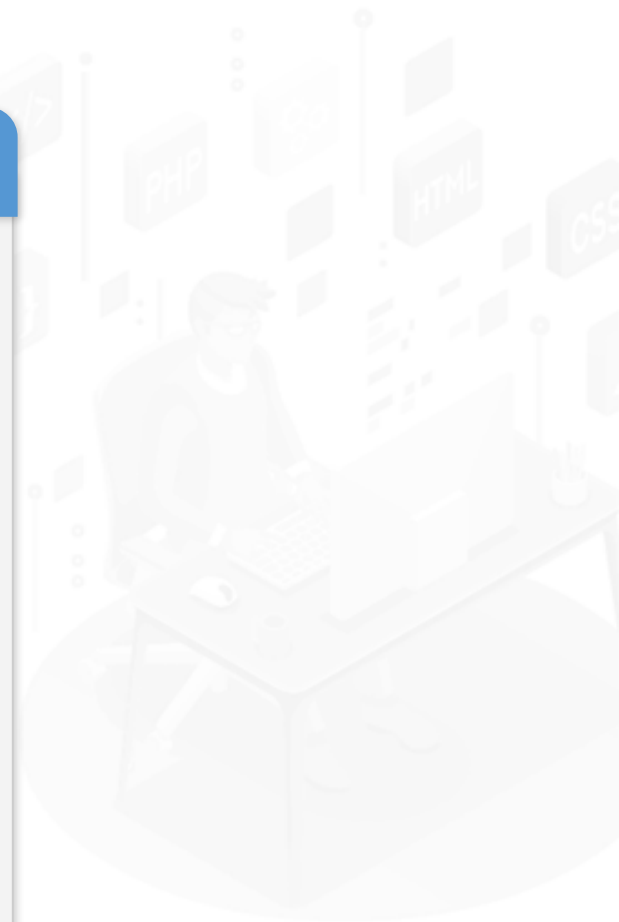


# Create Admins Table

You must make sure the database is selected as foodie, or you can execute the command to make the DB selection for foodie as:

**use foodie ;**

```
CREATE TABLE `admins` (  
  `admin_id` int NOT NULL,  
  `added_on` datetime(6) DEFAULT NULL,  
  `email` varchar(255) DEFAULT NULL,  
  `full_name` varchar(255) DEFAULT NULL,  
  `login_type` int DEFAULT NULL,  
  `password` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`admin_id`)  
) ;
```



# Create Users Table

**use foodie ;**

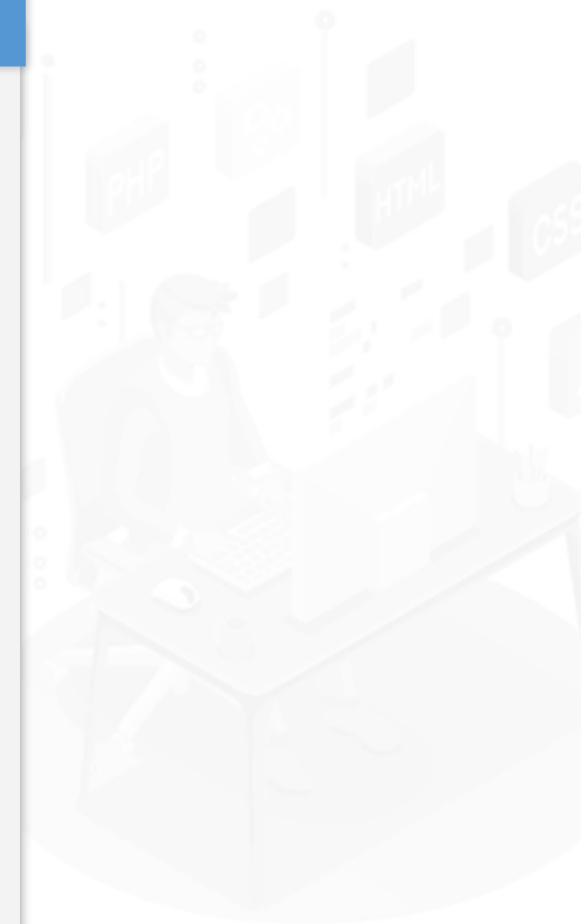
```
CREATE TABLE `users` (  
  `user_id` int NOT NULL,  
  `added_on` datetime(6) DEFAULT NULL,  
  `contact` bigint DEFAULT NULL,  
  `email` varchar(255) DEFAULT NULL,  
  `full_name` varchar(255) DEFAULT NULL,  
  `image` varchar(255) DEFAULT NULL,  
  `password` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`user_id`)  
);
```



# Create User Address Table

**use foodie ;**

```
CREATE TABLE `user_address` (  
  `address_id` int NOT NULL,  
  `address_tag` int DEFAULT NULL,  
  `city` varchar(255) DEFAULT NULL,  
  `country` varchar(255) DEFAULT NULL,  
  `pincode` int DEFAULT NULL,  
  `state` varchar(255) DEFAULT NULL,  
  `street` varchar(255) DEFAULT NULL,  
  `user_id` int DEFAULT NULL,  
  PRIMARY KEY (`address_id`)  
);
```



# Create Restaurants Table

**use foodie ;**

```
CREATE TABLE `restaurants` (  
  `restaurant_id` int NOT NULL,  
  `added_on` datetime(6) DEFAULT NULL,  
  `address` varchar(255) DEFAULT NULL,  
  `contact` bigint DEFAULT NULL,  
  `description` varchar(255) DEFAULT NULL,  
  `email` varchar(255) DEFAULT NULL,  
  `name` varchar(255) DEFAULT NULL,  
  `rating` int DEFAULT NULL,  
  `thumbnail_image` int DEFAULT NULL,  
  `status` int DEFAULT NULL,  
  PRIMARY KEY (`restaurant_id`));
```

# Generate Dish Table

**use foodie ;**

```
CREATE TABLE `dishes` (  
  `dish_id` int NOT NULL,  
  `added_on` datetime(6) DEFAULT NULL,  
  `description` varchar(255) DEFAULT NULL,  
  `name` varchar(255) DEFAULT NULL,  
  `price` int DEFAULT NULL,  
  `rating` int DEFAULT NULL,  
  `restaurant_address` varchar(255) DEFAULT NULL,  
  `restaurant_id` int DEFAULT NULL,  
  `restaurant_name` varchar(255) DEFAULT NULL,  
  `thumbnail_image` int DEFAULT NULL,  
  `status` int DEFAULT NULL,  
  PRIMARY KEY (`dish_id`));
```



# Generate Orders Table

```
CREATE TABLE `orders` (  
  `order_id` int NOT NULL,  
  `address` varchar(255) DEFAULT NULL,  
  `contact` bigint DEFAULT NULL,  
  `email` varchar(255) DEFAULT NULL,  
  `items_sub_total` double DEFAULT NULL,  
  `name` varchar(255) DEFAULT NULL,  
  `order_date` datetime(6) DEFAULT NULL,  
  `order_status` int DEFAULT NULL,  
  `payment_method` int DEFAULT NULL,  
  `payment_method_title` varchar(255) DEFAULT NULL,  
  `payment_status` int DEFAULT NULL,  
  `payment_status_title` varchar(255) DEFAULT NULL,  
  `shipment_charges` double DEFAULT NULL,  
  `total_amount` double DEFAULT NULL,  
  `total_items` int DEFAULT NULL,  
  `user_id` int DEFAULT NULL,  
  PRIMARY KEY (`order_id`)  
);
```



# Generate Order Items Table

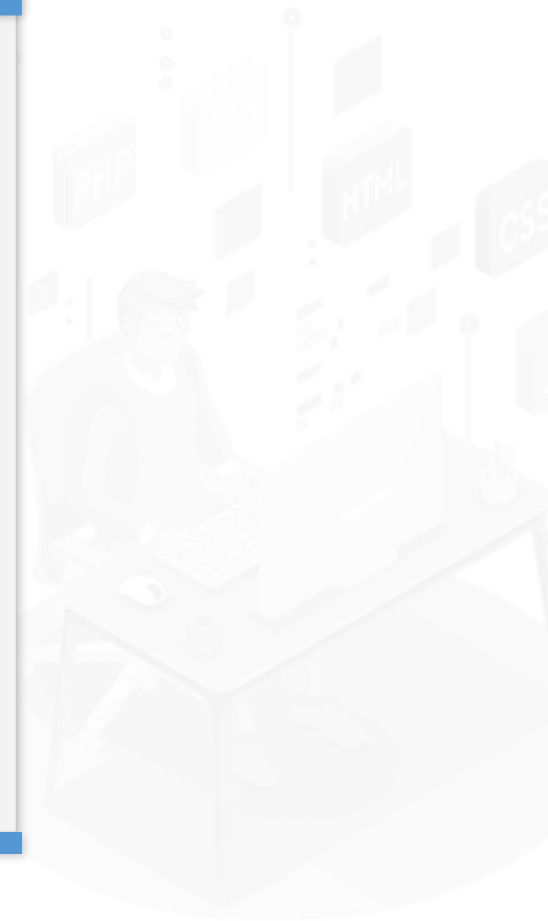
```
CREATE TABLE `order_items` (  
  `order_item_id` int NOT NULL,  
  `order_id` int DEFAULT NULL,  
  `price` int DEFAULT NULL,  
  `product_category` varchar(255) DEFAULT NULL,  
  `product_code` varchar(255) DEFAULT NULL,  
  `product_description` varchar(255) DEFAULT NULL,  
  `product_id` int DEFAULT NULL,  
  `product_img` varchar(255) DEFAULT NULL,  
  `product_title` varchar(255) DEFAULT NULL,  
  `quantity` int DEFAULT NULL,  
  `total_price` int DEFAULT NULL,  
  PRIMARY KEY (`order_item_id`)  
);
```



# Generate Cart Table

**use foodie ;**

```
CREATE TABLE `cart` (  
  `cart_id` int NOT NULL,  
  `dish_id` int DEFAULT NULL,  
  `quantity` int DEFAULT NULL,  
  `user_id` int DEFAULT NULL,  
  PRIMARY KEY (`cart_id`)  
);
```



## Key Takeaways

- 🕒 Users can perform hands-on tasks with Git commands.
- 🕒 They can create a MySQL database.
- 🕒 They can create tables for the project.
- 🕒 They can push the code to a GitHub account.



## Before the Next Class

Since you have successfully completed this session, in the next discussion, you should know how to:

- Execute HTML and CSS
- Review Angular CLI
- Configure dependencies in Angular
- Brush up Protractor





# What Next?

In the next class, we will:

- Build and execute Angular admin dashboard
- Build and execute Angular end user web app
- Configure dependencies for Angular projects
- Configure Protractor for Angular projects

