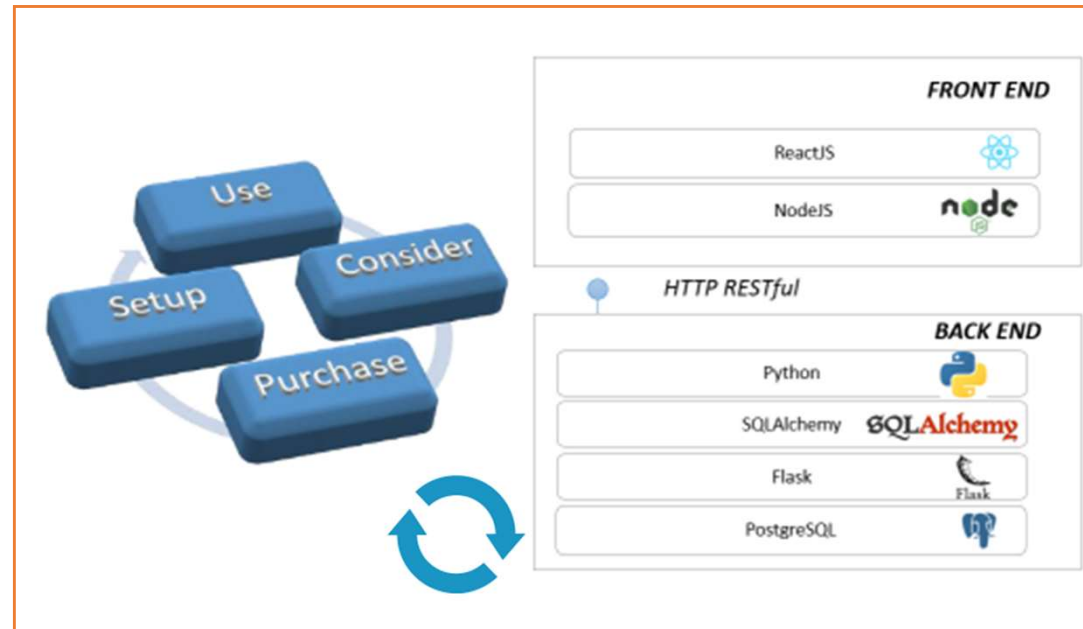


Retail Store Online | Demonstrator

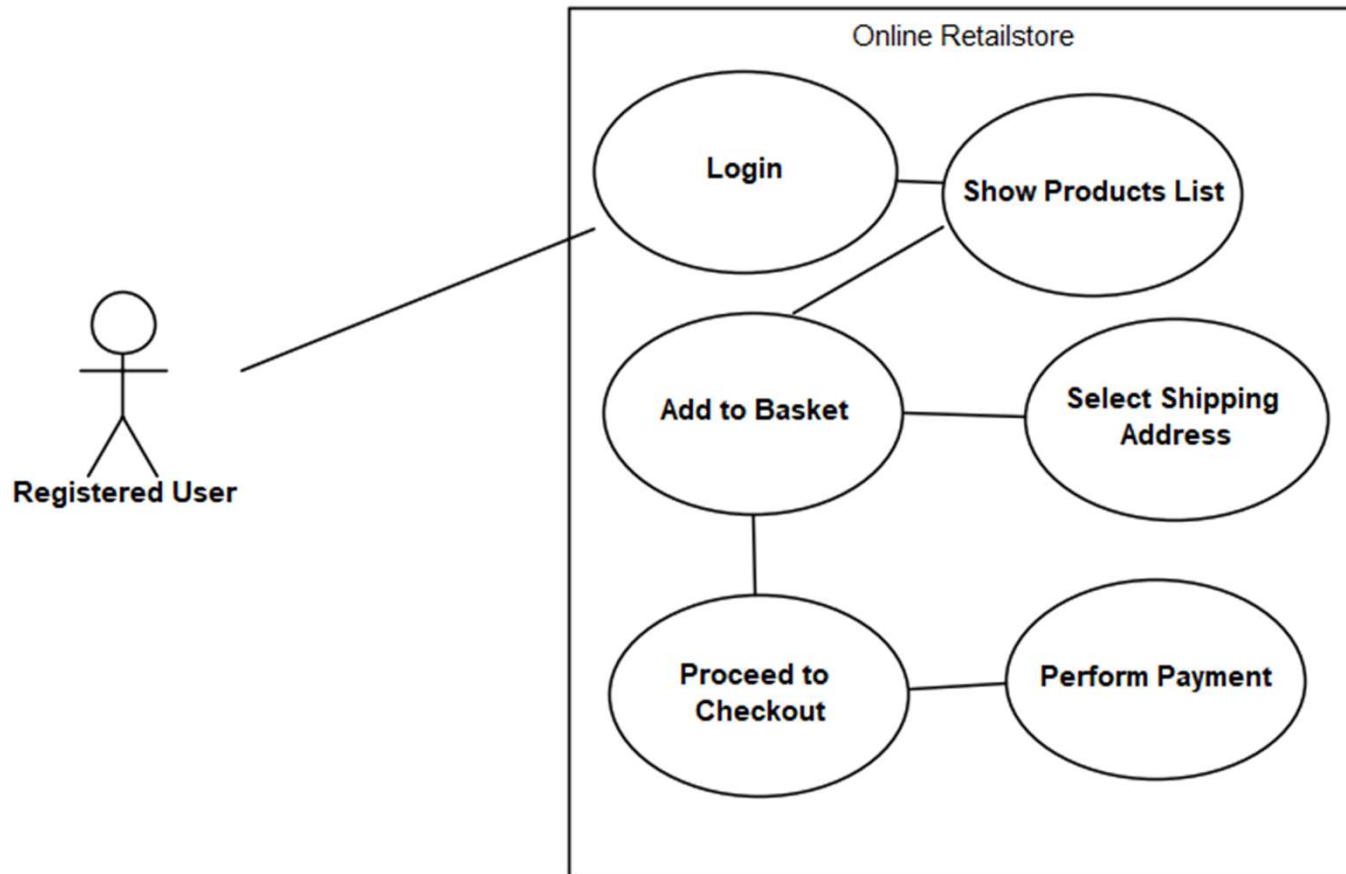
Presentation



Agenda

Overview | Architecture | Getting Started -API | ReactJS | Summary

Overview | Use case/Context



First set of Features

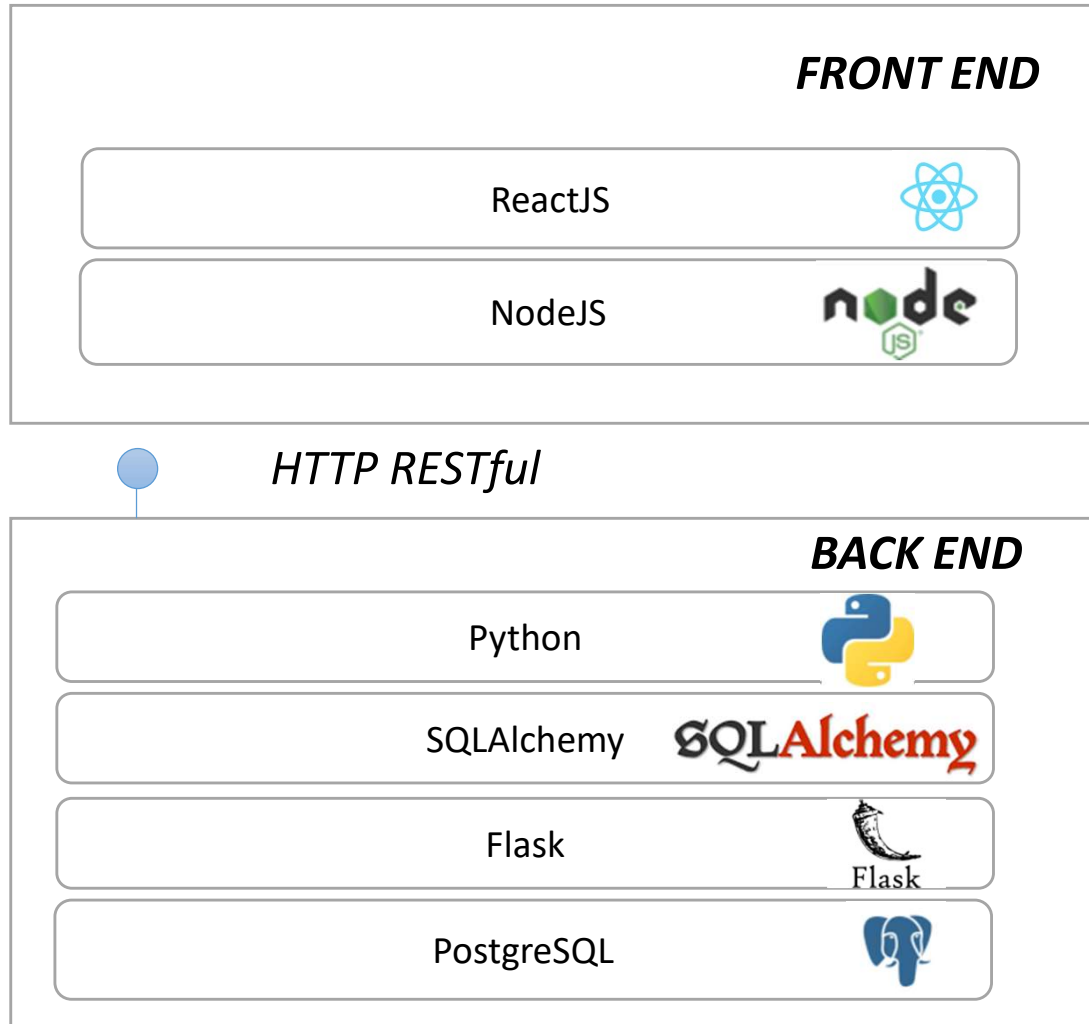
1. User should be able to **login and see a list of products**, select from there, select one of the defined addresses and do payment and see the order in order list.
2. Delivery happens separate as Shipped by the Seller and once Delivered it is updated in the System.

Assumptions:

User information including Address, Product Types, Products List are available. Everything available in Products List is available in Store in infinite numbers (Not checking availability at the moment).

Payment is dummy interface. Inventory and price history is not handled now. Price is fixed at the moment.

Overview | Architecture



The Front End is build using ReactJS



Service/API Testing is done using Postman

The Services/Back End is built using Python with Flask microframework serving HTTP RESTful services.

Architecture | Back end

```
#route returning Products list
@cross_origin(**api_v2_cors_config)
def getProductsList():
    product = ProductController()
    return product.getAllProducts()
```

Routes

ProductController

```
+ __init__(var)
+ getAllProducts(var)
+ getAllProductsByType(var, var)
+ obj_dict(var, var)
```

Controllers

PaymentType

db.Model

```
+ __tablename__: var = "payment_methods"
+ code: var = db.Column(db.In...)
+ description: var = db.Column(db.St...)
+ id: var = db.Column(db.In...)
+ name: var = db.Column(db.St...)
```

User

db.Model

```
+ __tablename__: var = "user"
+ admin: var = db.Column(db.Bo...)
+ email: var = db.Column(db.St...)
+ id: var = db.Column(db.In...)
+ password_hash: var = db.Column(db.St...)
+ public_id: var = db.Column(db.St...)
+ registered_on: var = db.Column(db.Da...)
+ username: var = db.Column(db.St...)
```

```
+ password(var)
+ password(var, var)
```

Product

db.Model

```
+ __tablename__: var = "products"
+ description: var = db.Column(db.St...)
+ id: var = db.Column(db.In...)
+ name: var = db.Column(db.St...)
+ producttype: var = db.relationship...
+ producttype_id: var = db.Column(db.In...)
```

```
+ __repr__(var)
+ to_json(var)
+ toJSON(var)
```

ProductType

db.Model

```
+ __tablename__: var = "product_types"
+ code: var = db.Column(db.St...)
+ description: var = db.Column(db.St...)
+ id: var = db.Column(db.In...)
+ name: var = db.Column(db.St...)
```

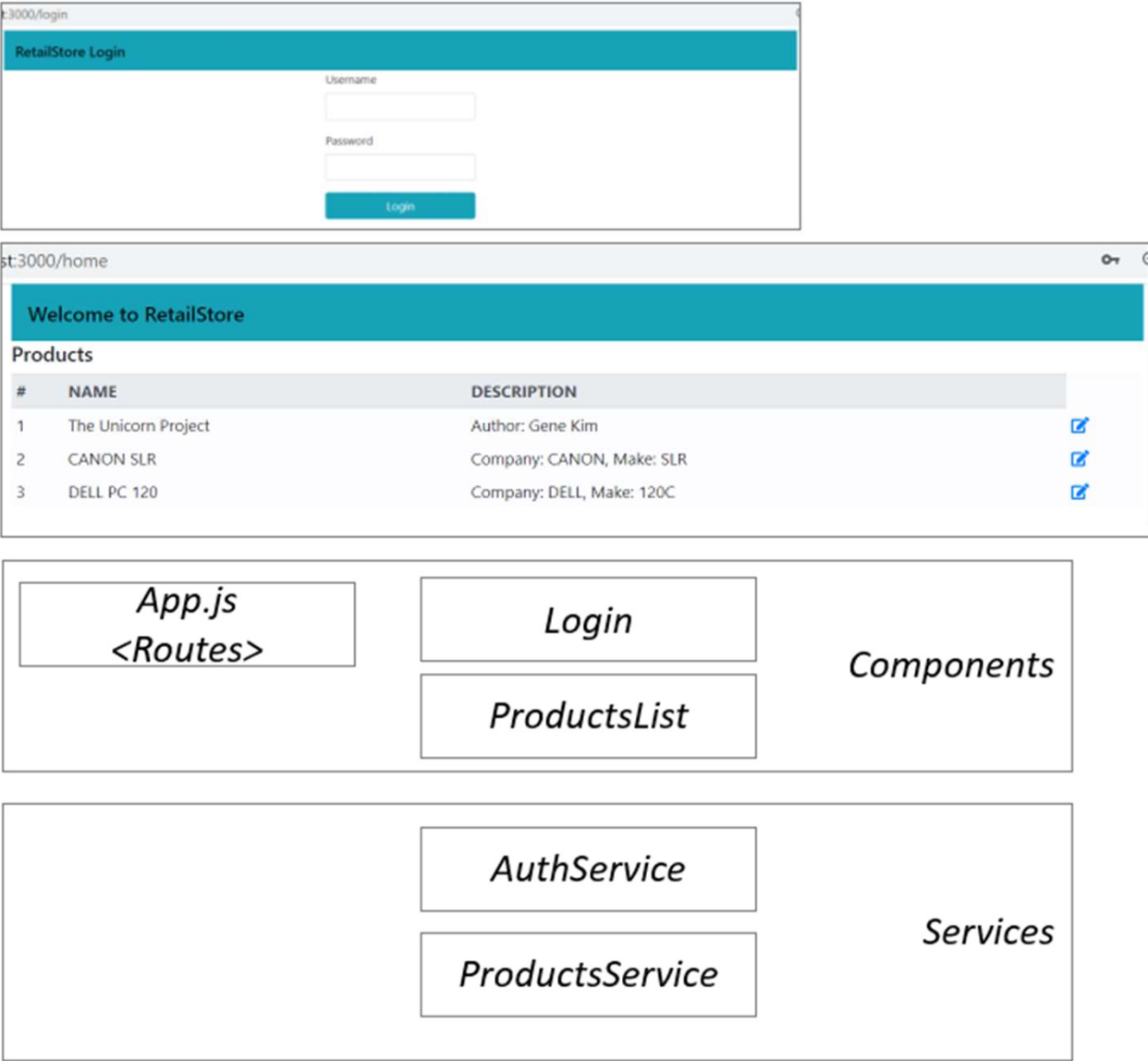
Model

Routes define the service endpoints. They are linked to a URL in manage.py using `add_url_rule` `app.add_url_rule('/api/producttypes', view_func=routes.getProductTypesList)`

Controllers expose the methods which can be exposed as service end points. They are invoked from route.

Model holds Object Relational Mapping to Database. These are made serializable to JSON using `@dataclass` attribute and defining the attributes (like `id:int` in `ProductType`)

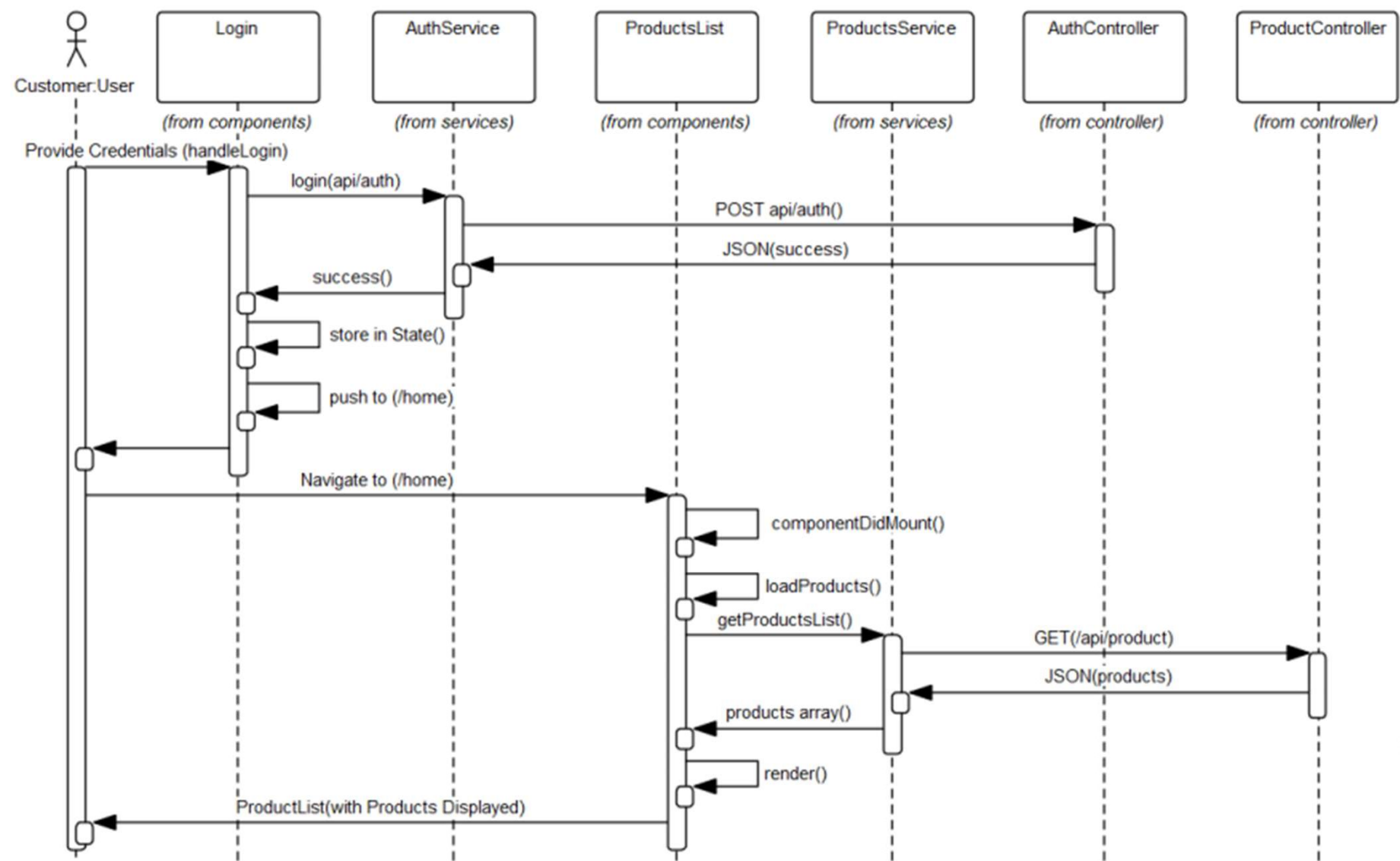
Architecture | Back end



*Components are the way react organizes elements
–Login, ProductsList are examples. They are
defined as routes in App.js*

Services make call to APIs and return response

Architecture | Interactions

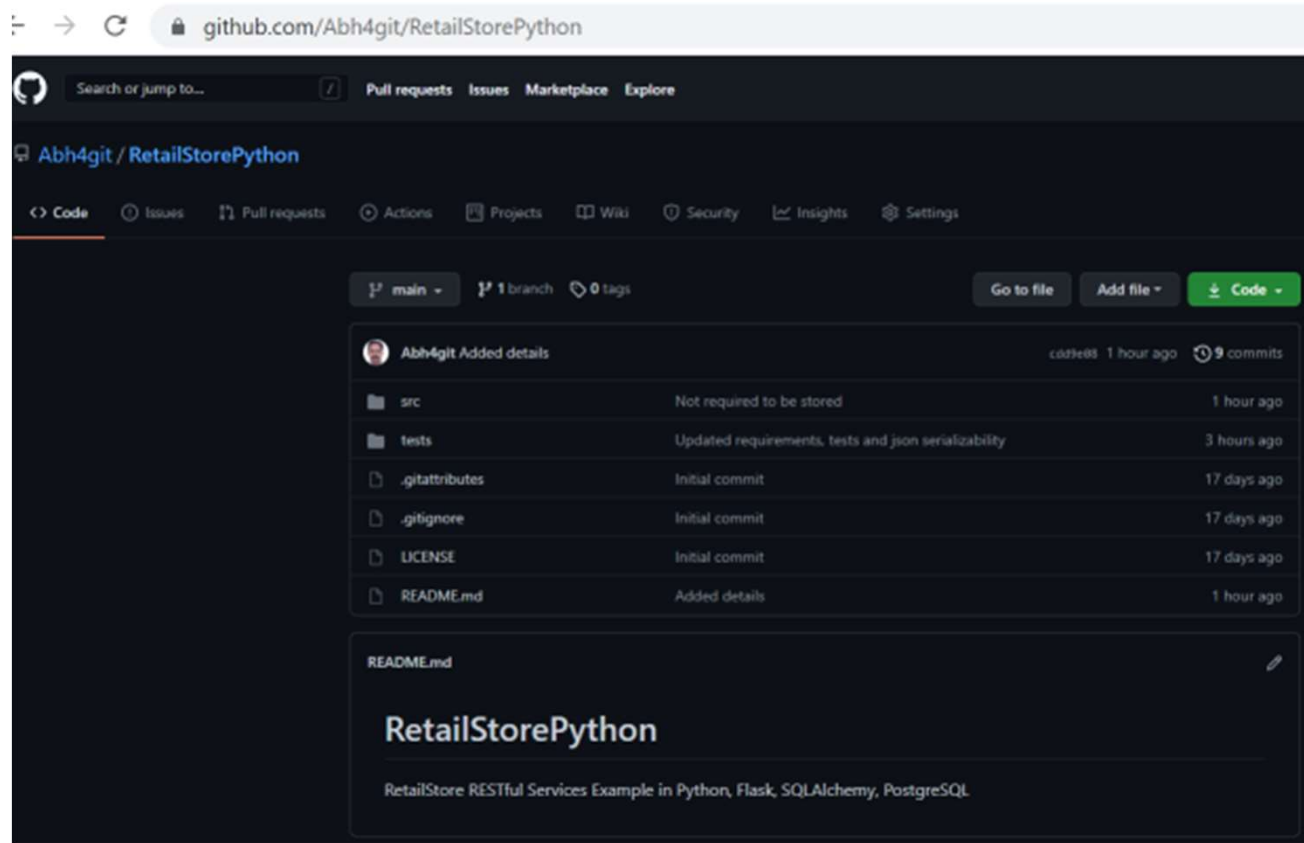


*the way react organizes elements
list are examples. They are
in App.js*

// to APIs and return response

Architecture – Sequence Diagram

Getting Started | **Back End**

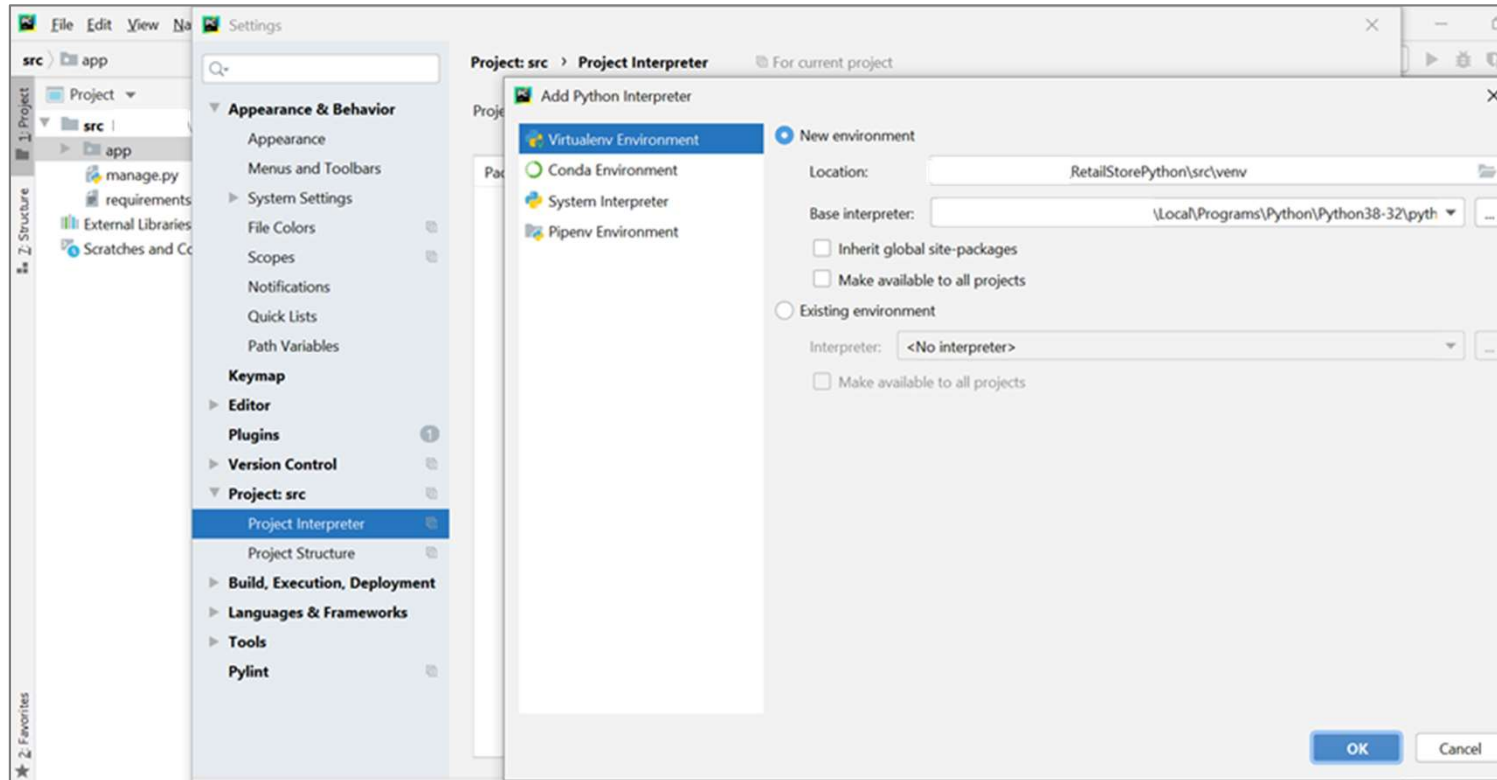


GitHub URL:

<https://github.com/Abh4git/RetailStorePython>

GitHub Repo

Getting Started | Back End



1. Open PyCharm in src folder and setup Interpreter and select Virtual Environment

2. Pip install as shown below:

```
src>pip install -r requirements.txt
```

3. Set FLASK_APP environment variable like below

src> **set FLASK_APP=manage.py** (In Linux use export)

Back End running

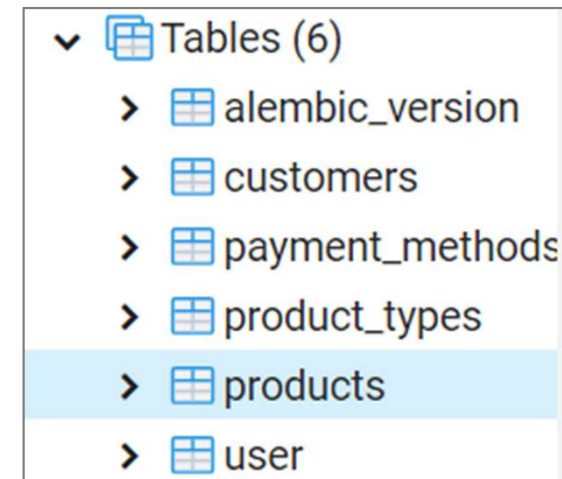
Getting Started | Back End - Database

```
age.py x config.py x app\_init_.py x producttypes.py x user.py x model\_init_.py x main\_init_.py x
basedir = os.path.abspath(os.path.dirname(__file__))
class Config:
    SECRET_KEY = os.getenv('SECRET_KEY', 'my_precious_secret_key')
    DEBUG = False
class DevelopmentConfig(Config):
    # uncomment the line below to use postgres
    SQLALCHEMY_DATABASE_URI = 'postgresql://localhost/retailstoredb?user=postgres&password=[REDACTED]'
    SQLALCHEMY_TRACK_MODIFICATIONS = True
```

```
Terminal: Local x +
(venv) D:\Abhilash\GitHub\RetailStorePython\src>flask db migrate -m "Added table ProductTypes"
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Will assume transactional DDL.
INFO [alembic.autogenerate.compare] Detected added table 'product_types'
INFO [alembic.ddl.postgresql] Detected sequence named 'user_id_seq' as owned by integer column 'user(id)', assum
Generating D:\Abhilash\GitHub\RetailStorePython\src\migrations\versions\be483fd19b64_added_table_producttypes.py

(venv) D:\Abhilash\GitHub\RetailStorePython\src>flask db upgrade
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Will assume transactional DDL.
INFO [alembic.runtime.migration] Running upgrade 28619cbdb4d0 -> be483fd19b64, Added table ProductTypes
```

1. Check your database URI is updated with your username and password
2. Run Db Init to start migration like below:
src> flask db init
3. Run DB Migrate
src> flask db migrate -m "Added tables"
4. Run DB Upgrade
src> flask db upgrade



*After step4,
you should
find all
tables in
database*

Getting Started | Back End - Database

Tables (6)

- alembic_version
- customers
- payment_methods
- product_types
- Columns
- Constraints
- Indexes

Data Output					Explain	Messages	Notifications
	id		name	description	code		
	[PK] integer		character varying (255)	character varying (255)	character varying (255)		
1		1	BOOK	Includes eBooks and Physical...	BKCODE123		
2		2	CAMERA	All types of Camera	CAMERA234		
3		3	COMPUTER	All Type of Computer	COMP456		
4		4	COMPUTER ACCESSORIES	Mouse, Keyboard etc	COMPAC234		



1. Add entries in product_types table

Tables (6)

- alembic_version
- customers
- payment_methods
- product_types
- products
- Columns
- Constraints

Data Output					Explain	Messages	Notifications
	id		name	description	producttype_id		
	[PK] integer		character varying (255)	character varying (255)	integer		
1		1	The Unicorn Project	Author: Gene Kim			1
2		2	CANON SLR	Company: CANON, Make: SLR			2
3		3	DELL PC 120	Company: DELL, Make: 120C			3

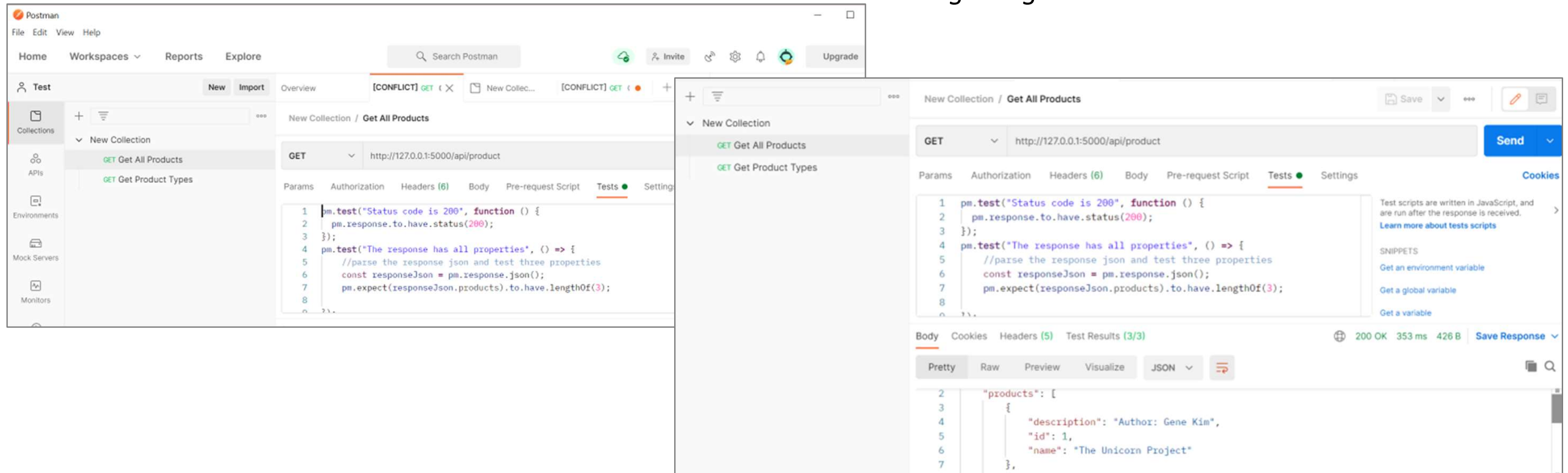
2. Add entries in products table

Getting Started | Running Back End and Testing it

```
(venv) D:\Abhilash\GitHub\RetailStorePython\src>flask run
* Serving Flask app "manage.py"
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

1. Start execution from terminal
src> flask run

Testing using Postman



The Test results show if the service is running fine.

Getting Started | **Front End**

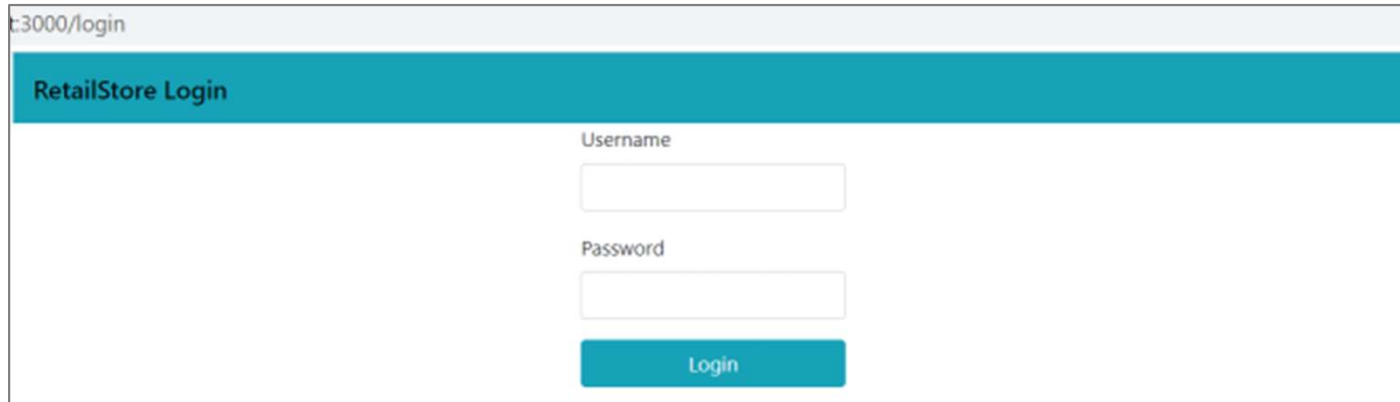
Pre-requisite: NodeJs need to be installed:

1. *Install dependencies*

*src> **npm install***

2. *Start program*

*src> **npm start***



A screenshot of a web browser window showing the 'RetailStore Login' page. The browser's address bar displays 't:3000/login'. The page has a teal header with the text 'RetailStore Login'. Below the header, there are two input fields: 'Username' and 'Password'. A teal 'Login' button is positioned below the password field.

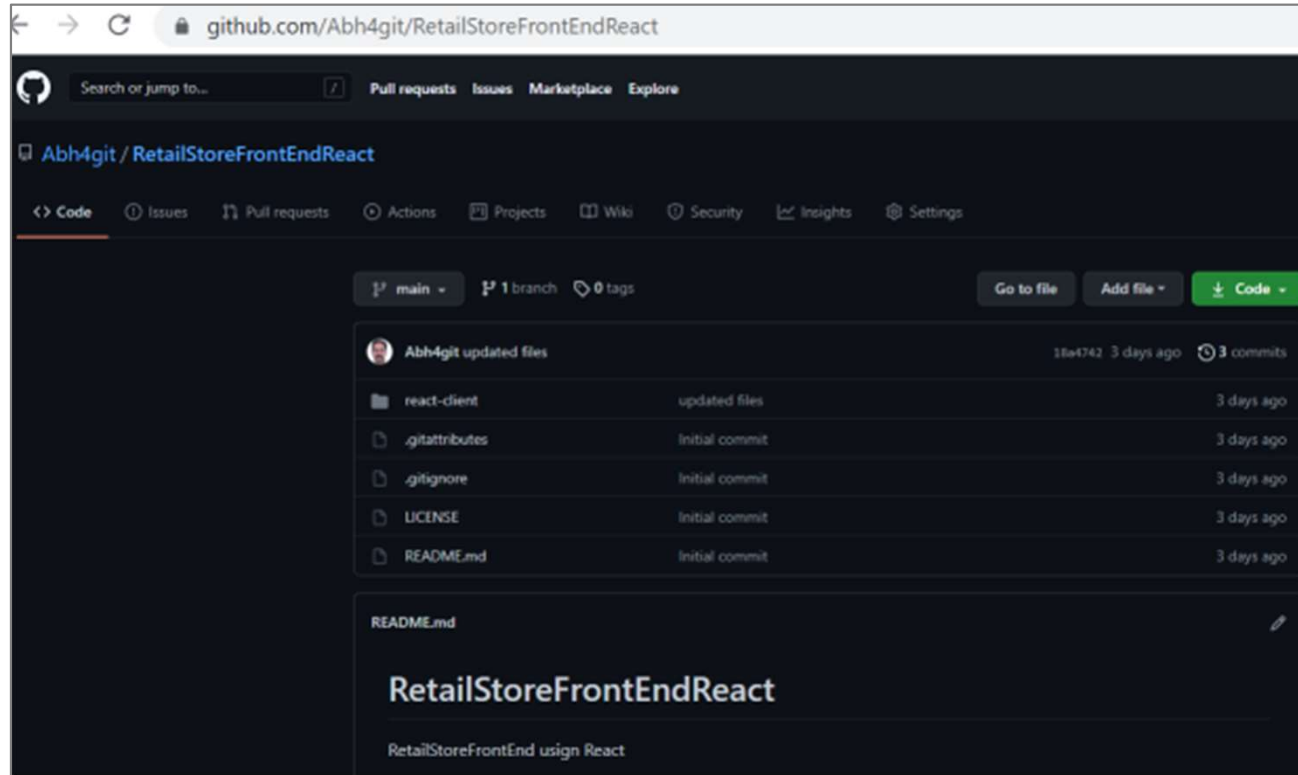


A screenshot of a web browser window showing the 'RetailStore' home page. The browser's address bar displays 'st:3000/home'. The page has a teal header with the text 'Welcome to RetailStore'. Below the header, there is a section titled 'Products' which contains a table with three columns: '#', 'NAME', and 'DESCRIPTION'. The table lists three products: 'The Unicorn Project' by Gene Kim, 'CANON SLR' by CANON, and 'DELL PC 120' by DELL. Each product row has a blue edit icon (pencil) to its right.

#	NAME	DESCRIPTION	
1	The Unicorn Project	Author: Gene Kim	
2	CANON SLR	Company: CANON, Make: SLR	
3	DELL PC 120	Company: DELL, Make: 120C	

Executing the Front End

Getting Started | **Front End**



GitHub URL:

<https://github.com/Abh4git/RetailStoreFrontEndReact>

GitHub Repo

Summary | **Architecture, Design and Getting it running**

- 1. Back End using Python, Flask, SQLAlchemy and PostgreSQL*
- 2. Front End using NodeJS, ReactJS*
- 3. Testing using Postman*
- 4. Step by step approach explained.*

Thank you!