



โครงการ

เรื่อง ร้านขายต้นไม้

จัดทำโดย

นายนิกิต้า	โหมัส	รหัสนักศึกษา 632110342
นางสาวพิจักขณา	จันทับหลวง	รหัสนักศึกษา 632110348
นายภูมิพัฒน์	ธีระวัฒน์	รหัสนักศึกษา 632110350
นางสาวธัญชนก	ดวงทิพย์	รหัสนักศึกษา 632110363

เสนอ

อาจารย์ ศรัณยา ศิริพฤกษ์พงษ์

วิชา 960231 (M6 Backebd) Digital Industry Infrastructure 2

ภาคการศึกษาที่ 1 ปีการศึกษา 2564

วิทยาลัยศิลปะ สื่อ และเทคโนโลยี มหาวิทยาลัยเชียงใหม่

Api

Postman interface showing a GET request to `localhost:8080/api/products`. The response is a JSON array of two product objects.

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body

```
1 {
2   {
3     "id": 2,
4     "name": "Mammillaria",
5     "imageUrl": "/assets/images/cactus2.jpeg",
6     "description": "ขนาด: 6.5 cm (4นิ้วเฉพาะต้นไม้ในแบบมาตรฐาน)",
7     "rating": 4,
8     "price": 399,
9     "quantity": 20
10  },
11  {
12    "id": 3,
13    "name": "Astrophytum",
14    "imageUrl": "/assets/images/cactus3.jpeg",
15    "description": "ขนาด: 6.6 cm (4นิ้วเฉพาะต้นไม้ในแบบมาตรฐาน)",
16    "rating": 5,
17    "price": 1299,
18    "quantity": 30
19  }
20 }
```

Status: 200 OK Time: 54 ms Size: 2.6 KB

Postman interface showing a GET request to `localhost:8080/api/product?id=2`. The response is a JSON object for a single product.

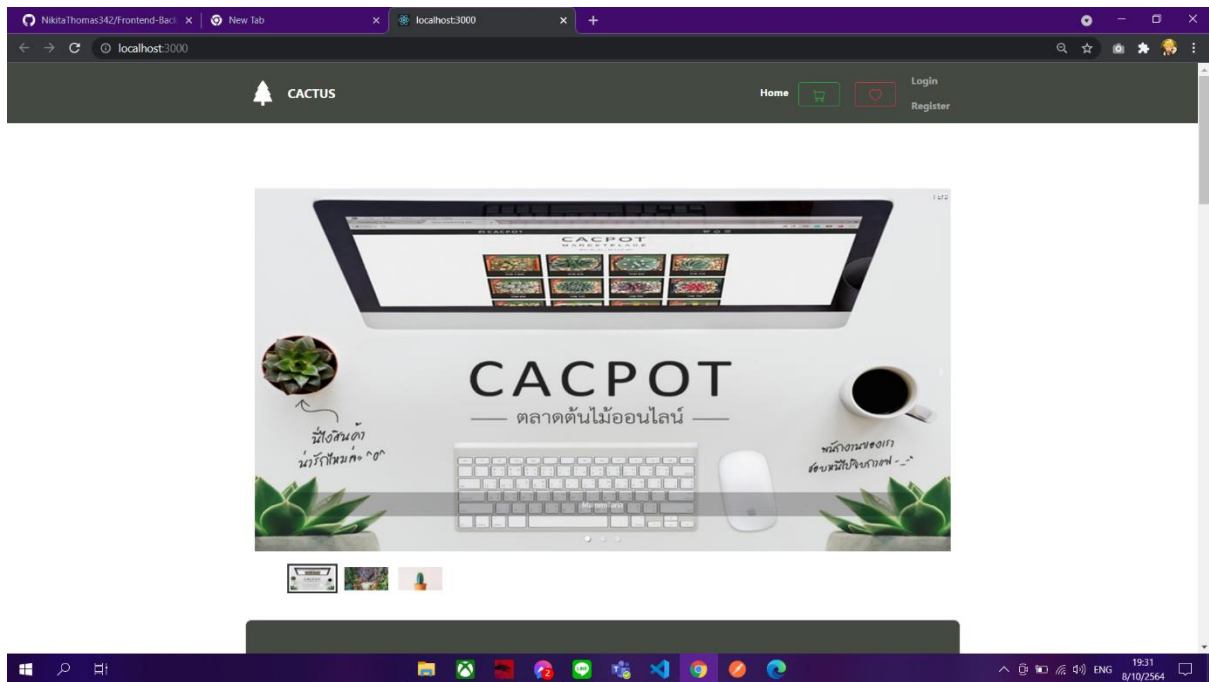
Query Params

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> id	2	
Key	Value	Description

Body

```
1 {
2   {
3     "id": 2,
4     "name": "Mammillaria",
5     "imageUrl": "/assets/images/cactus2.jpeg",
6     "description": "ขนาด: 6.5 cm (4นิ้วเฉพาะต้นไม้ในแบบมาตรฐาน)",
7     "rating": 4,
8     "price": 399,
9     "quantity": 20
10  }
11 }
```

Status: 200 OK Time: 26 ms Size: 569 B



Home:

```

router.get('/api/products', async (req, res) => {
  try{
    let products = new Promise((resolve,reject)=>{
      connection.query('SELECT * FROM product',(err,result)=>{
        if(err){
          return reject(err)
        }
        resolve(results)
      })
    })
    res.json(products)
  } catch(err) {
    console.log(err)
    res.sendStatus(500)
  }
})

router.delete('/api/deleteproduct', async (req,res) => {
  try{
    await new Promise((resolve,reject) => {
      connection.query('DELETE FROM product WHERE id = (?)',[req.query.id] ,(err,result)=>{
        if(err){
          return reject(err)
        }
        resolve(results)
      })
    })
    res.json('success')
  }catch(err){
    console.log(err)
    res.sendStatus(500)
  }
})

```

```

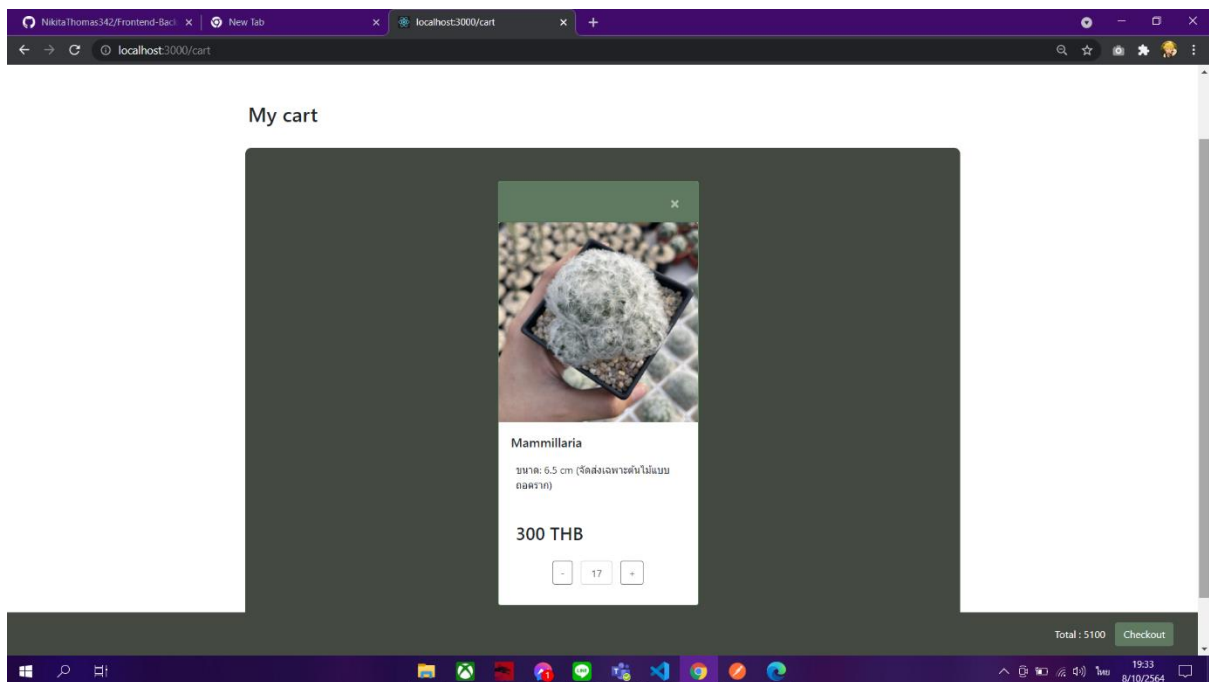
    router.post('/api/addcart', async (req,res) => {
    try{
        let body = req.body
        let user_id = body.user_id
        let name = body.name
        let imageURL = body.imageURL
        let description = body.description
        let rating = body.rating
        let price = body.price
        let quantity = 1

        await new Promise((resolve,reject)=>{
            connection.query('INSERT INTO cart (user_id,name,imageURL,description,rating,price,quantity) values
            (?,?,?,?,?,?,?)',[user_id,name,imageURL,description,rating,price,quantity] ,(err,results)=>{
                if(err){
                    return reject(err)
                }
                resolve(results)
            })
        })
        res.json('success')
    }catch(err){
        console.log(err)
        res.sendStatus(500)
    }
    })

    router.post('/api/addfavorite', async (req,res) => {
    try{
        let body = req.body
        let user_id = body.user_id
        let name = body.name
        let imageURL = body.imageURL
        let description = body.description
        let rating = body.rating
        let price = body.price
        let quantity = body.quantity

        await new Promise((resolve,reject)=>{
            connection.query('INSERT INTO favorite (user_id,name,imageURL,description,rating,price,quantity) values
            (?,?,?,?,?,?,?)',[user_id,name,imageURL,description,rating,price,quantity] ,(err,results)=>{
                if(err){
                    return reject(err)
                }
                resolve(results)
            })
        })
        res.json('success')
    }catch(err){
        console.log(err)
        res.sendStatus(500)
    }
    })

```



Cart:

```
router.get('/api/getcart', async (req,res) => {
  try{
    let products = await new Promise((resolve,reject)=>{
      connection.query("SELECT * FROM cart WHERE user_id = (?)",[req.query.id] ,(err,results)=>{
        if(err){
          return reject(err)
        }
        resolve(results)
      })
    })
    res.json(products)
  }catch(err){
    console.log(err)
    res.sendStatus(500)
  }
})

router.delete('/api/deletcart', async (req,res) => {
  try{
    await new Promise((resolve,reject) => {
      connection.query("DELETE FROM cart WHERE id = (?)",[req.query.id] ,(err,results)=>{
        if(err){
          return reject(err)
        }
      }
    })
  }
})
```

```

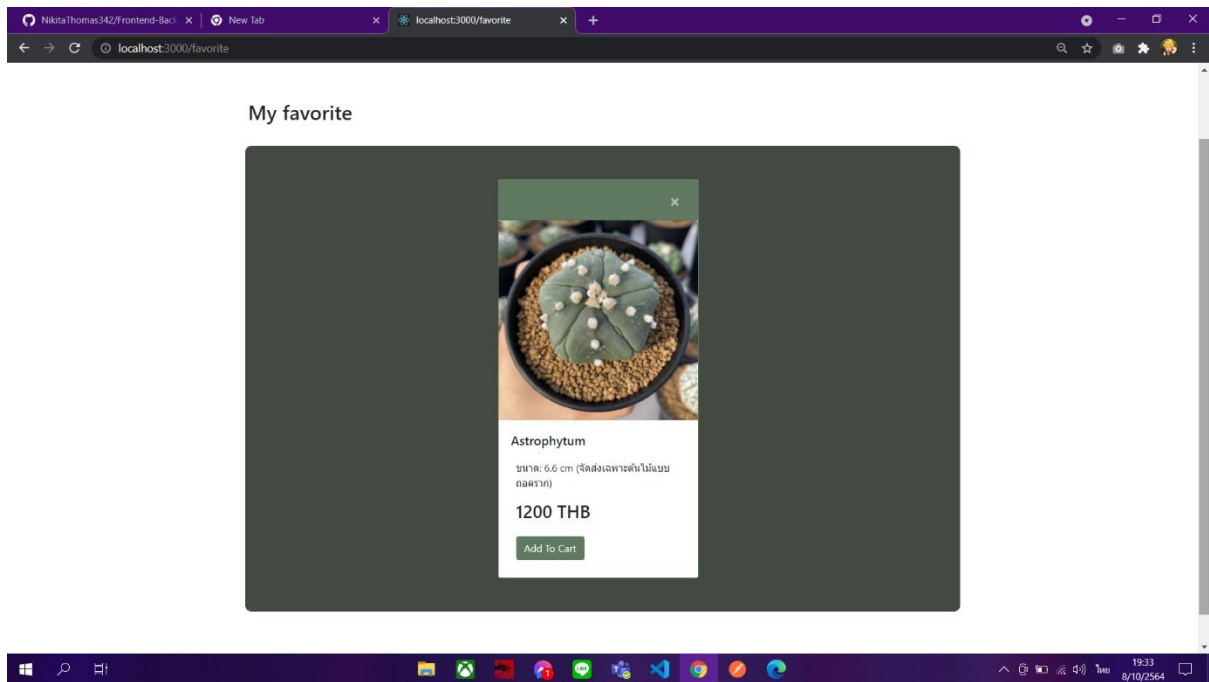
        resolve(results)
      })
    })
    res.json('success')
  } catch (err) {
    console.log(err)
    res.sendStatus(500)
  }
})

router.post('/api/updatecart', async (req, res) => {
  try {
    let body = req.body
    let id = body.id
    let quantity = body.quantity

    await new Promise((resolve, reject) => {
      connection.query('UPDATE cart SET quantity = (?) WHERE id = (?)', [quantity, id], (err, results) => {
        if (err) {
          return reject(err)
        }
        resolve(results)
      })
    })

    res.json('success')
  } catch (err) {
    console.log(err)
    res.sendStatus(500)
  }
})

```



Favorite:

```
router.post('/api/addcart', async (req,res) => {
  try{
    let body = req.body
    let user_id = body.user_id
    let name = body.name
    let imageURL = body.imageURL
    let description = body.description
    let rating = body.rating
    let price = body.price
    let quantity = 1

    await new Promise((resolve,reject)=>{
      connection.query("INSERT INTO cart (user_id,name,imageURL,description,rating,price,quantity) values
      (?,?,?,?,?,?,?)",[user_id,name,imageURL,description,rating,price,quantity] ,(err,results)=>{
        if(err){
          return reject(err)
        }
        resolve(results)
      })
    })
    res.json('success')
  }catch(err){
    console.log(err)
    res.sendStatus(500)
  }
})
```

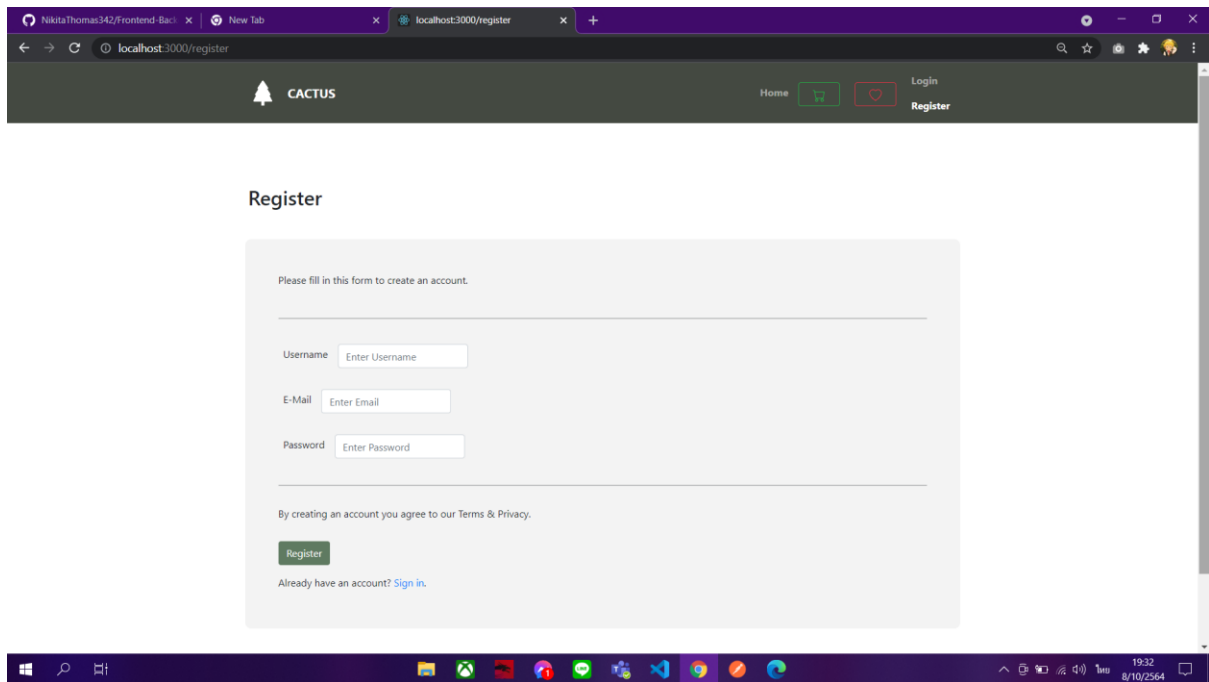
```

    })

    router.get('/api/getfavorite', async (req,res) => {
    try{
        let products = await new Promise((resolve,reject)=>{
            connection.query('SELECT * FROM favorite WHERE user_id = (?)',[req.query.id] ,(err,results)=>{
                if(err){
                    return reject(err)
                }
                resolve(results)
            })
        })
        res.json(products)
    }catch(err){
        console.log(err)
        res.sendStatus(500)
    }
    })

    router.delete('/api/deletefavorite', async (req,res) => {
    try{
        await new Promise((resolve,reject) => {
            connection.query('DELETE FROM favorite WHERE id = (?)',[req.query.id] ,(err,results)=>{
                if(err){
                    return reject(err)
                }
                resolve(results)
            })
        })
        res.json('success')
    }catch(err){
        console.log(err)
        res.sendStatus(500)
    }
    })

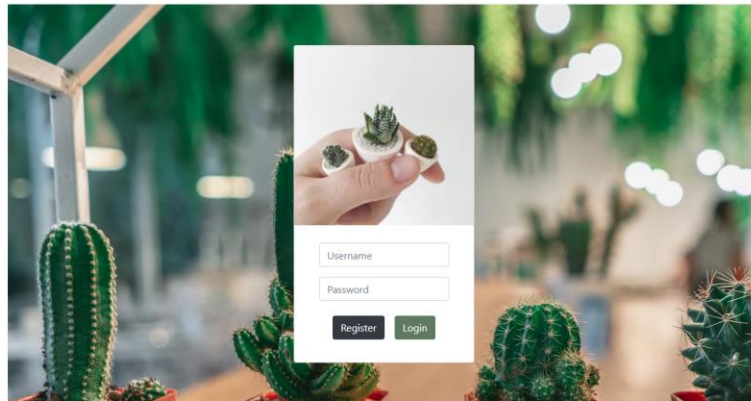
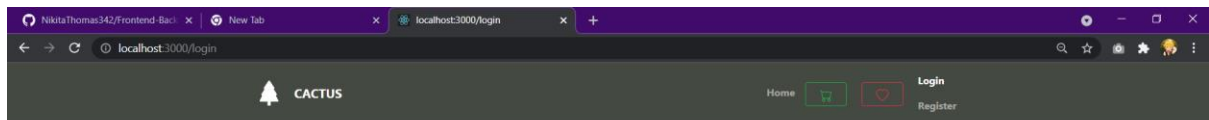
```

Register:

```
router.post('/api/register', async (req,res) => {
  try{
    let body = req.body
    let username = body.username
    let password = body.password
    let email = body.email
    var timestamp = new Date()
    timestamp.toISOString().slice(0, 19).replace('T', ' ')

    await new Promise((resolve,reject)=>{
      connection.query('INSERT INTO users (username,password,email,created,updated) values (?,?,?,?,?)',
        [username,password,email,timestamp,timestamp],
        (err,result)=>{
          if(err){
            return reject(err)
          }
          resolve()
        })
    })
    res.json('success')
  }catch(err){
    console.log(err)
    res.sendStatus(500)
  }
})
```



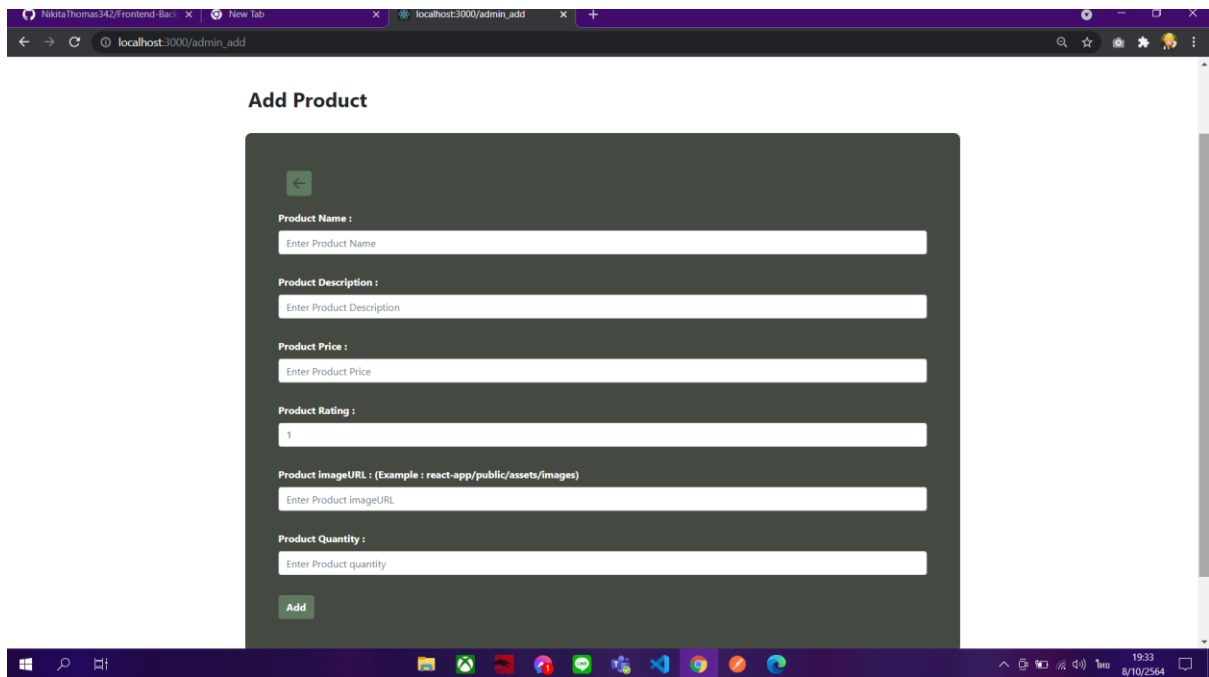
Cactus Shop @copyright



Login:

```
router.get('/api/login', async (req,res) => {
  try{
    var timestamp = new Date()
    timestamp.toISOString().slice(0, 19).replace('T', ' ')

    let user = new Promise((resolve,reject) => {
      connection.query('SELECT * FROM users WHERE username = (?)',
        [req.query.username],
        (err,result)=>{
          if(err){
            return reject(err)
          }
          resolve(result)
        })
    })
    res.json(user)
  }catch(err){
    console.log(err)
    res.sendStatus(500)
  }
})
```



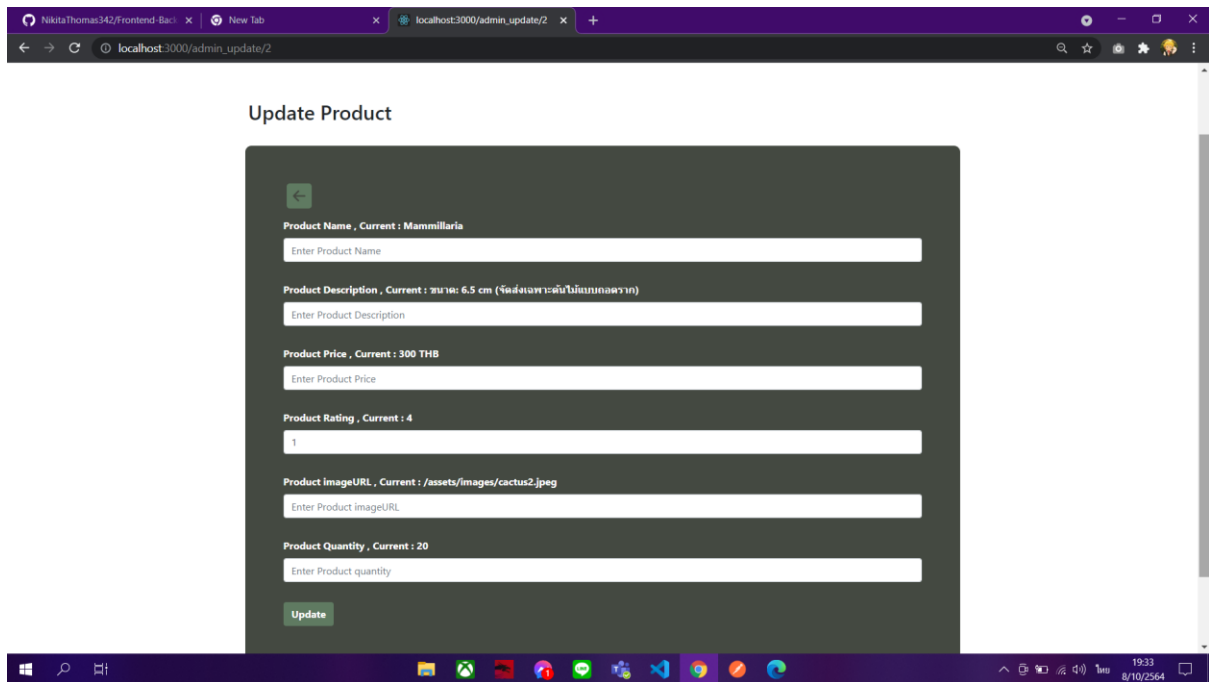
Admin_Add:

```

    router.post('/api/addproduct', async (req,res) => {
    try{
      let body = req.body
      let name = body.name
      let imageURL = body.imageURL
      let description = body.description
      let rating = body.rating
      let price = body.price
      let quantity = body.quantity

      await new Promise((resolve,reject)=>{
        connection.query('INSERT INTO product (name,imageURL,description,rating,price,quantity) VALUES
        (?,?,,?,?),[name,imageURL,description,rating,price,quantity]',(err,result)=>{
          if(err){
            return reject(err)
          }
          resolve(result)
        })
      })
      res.json('success')
    }catch(err){
      console.log(err)
      res.sendStatus(500)
    }
  })

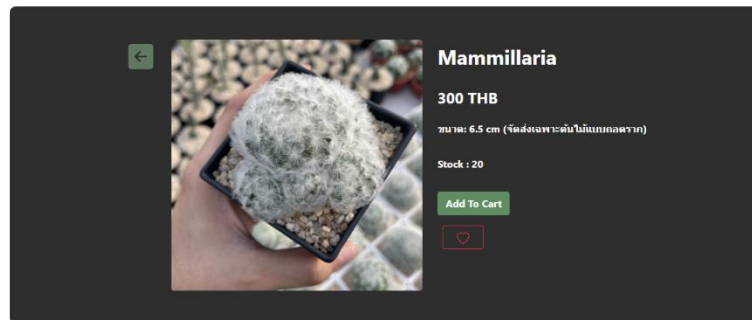
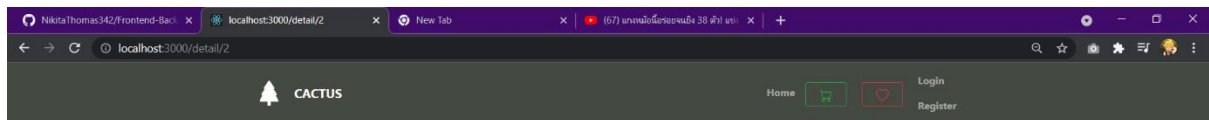
```



Admin_Update:

```
router.post('/api/updateproduct', async (req,res) => {
  try{
    let body = req.body
    let id = body.id
    let name = body.name
    let imageURL = body.imageURL
    let description = body.description
    let rating = body.rating
    let price = body.price
    let quantity = body.quantity

    await new Promise((resolve,reject)=>{
      connection.query('UPDATE product SET name = (?),imageURL = (?),description = (?),rating = (?),price = (?),quantity
= (?) WHERE id = (?)',[name,imageURL,description,rating,price,quantity,id] ,(err,results)=>{
        if(err){
          return reject(err)
        }
        resolve(results)
      })
    })
    res.json('success')
  }catch(err){
    console.log(err)
    res.sendStatus(500)
  }
})
```



Cactus Shop @copyright



Detail:

```

router.get('/api/productid', async (req,res) => {
  try{
    let product = new Promise((resolve,reject)=>{
      connection.query("SELECT * FROM product WHERE id=(?)",[id] ,(err,results)=>{
        if(err){
          return reject(err)
        }
        resolve(results)
      })
    })
    res.json(product)
  }catch(err){
    console.log(err)
    res.sendStatus(500)
  }
})

router.post('/api/addcart', async (req,res) => {
  try{
    let body = req.body
    let user_id = body.user_id
    let name = body.name
    let imageURL = body.imageURL
    let description = body.description
    let rating = body.rating
    let price = body.price
    let quantity = 1
  
```

```

    await new Promise((resolve,reject)=>{
        connection.query('INSERT INTO cart (user_id,name,imageURL,description,rating,price,quantity) values
(?,?,?,?,?,?,?)',[user_id,name,imageURL,description,rating,price,quantity] ,(err,results)=>{
            if(err){
                return reject(err)
            }
            resolve(results)
        })
    })
    res.json('success')
} catch(err){
    console.log(err)
    res.sendStatus(500)
}
})

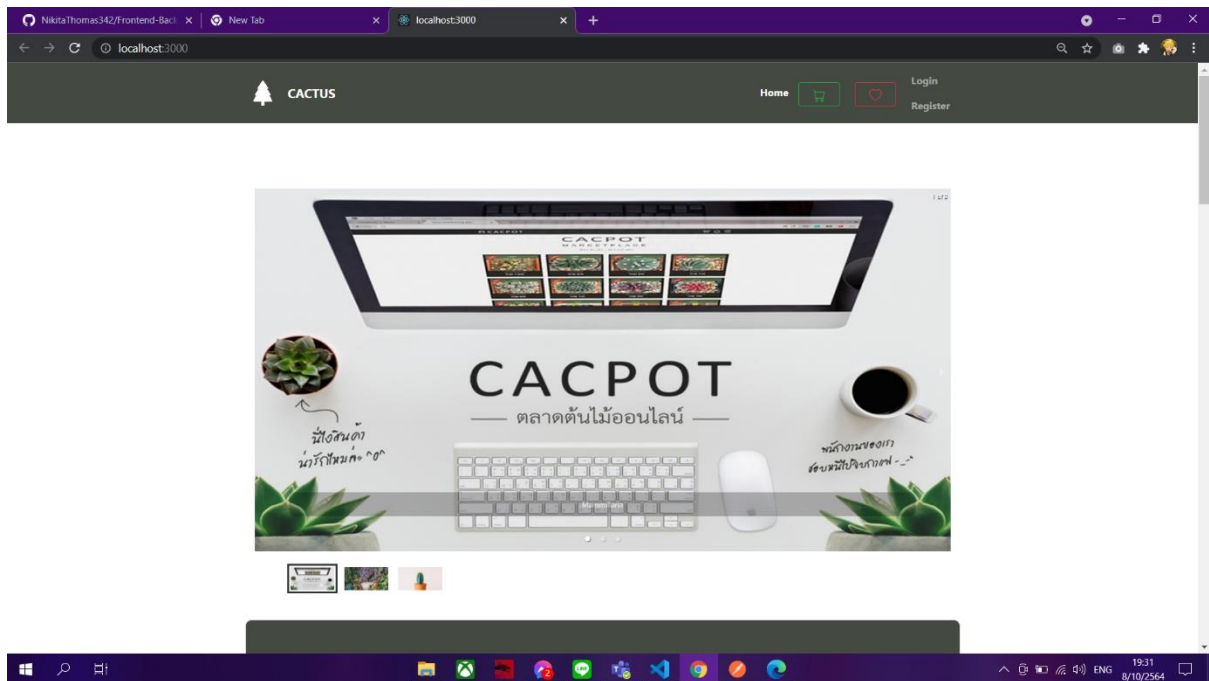
router.post('/api/addfavorite', async (req,res) => {
try{
    let body = req.body
    let user_id = body.user_id
    let name = body.name
    let imageURL = body.imageURL
    let description = body.description
    let rating = body.rating
    let price = body.price
    let quantity = body.quantity

    await new Promise((resolve,reject)=>{
        connection.query('INSERT INTO favorite (user_id,name,imageURL,description,rating,price,quantity) values
(?,?,?,?,?,?,?)',[user_id,name,imageURL,description,rating,price,quantity] ,(err,results)=>{
            if(err){
                return reject(err)
            }
            resolve(results)
        })
    })
    res.json('success')
} catch(err){
    console.log(err)
    res.sendStatus(500)
}
})

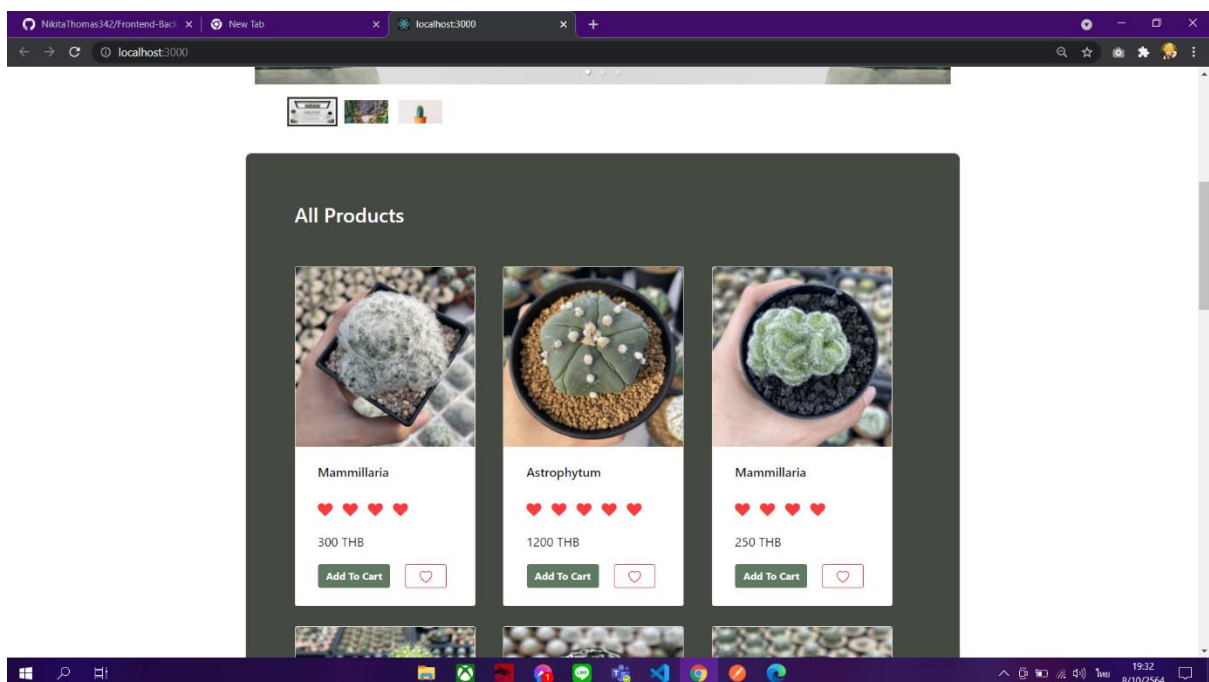
```

Web server

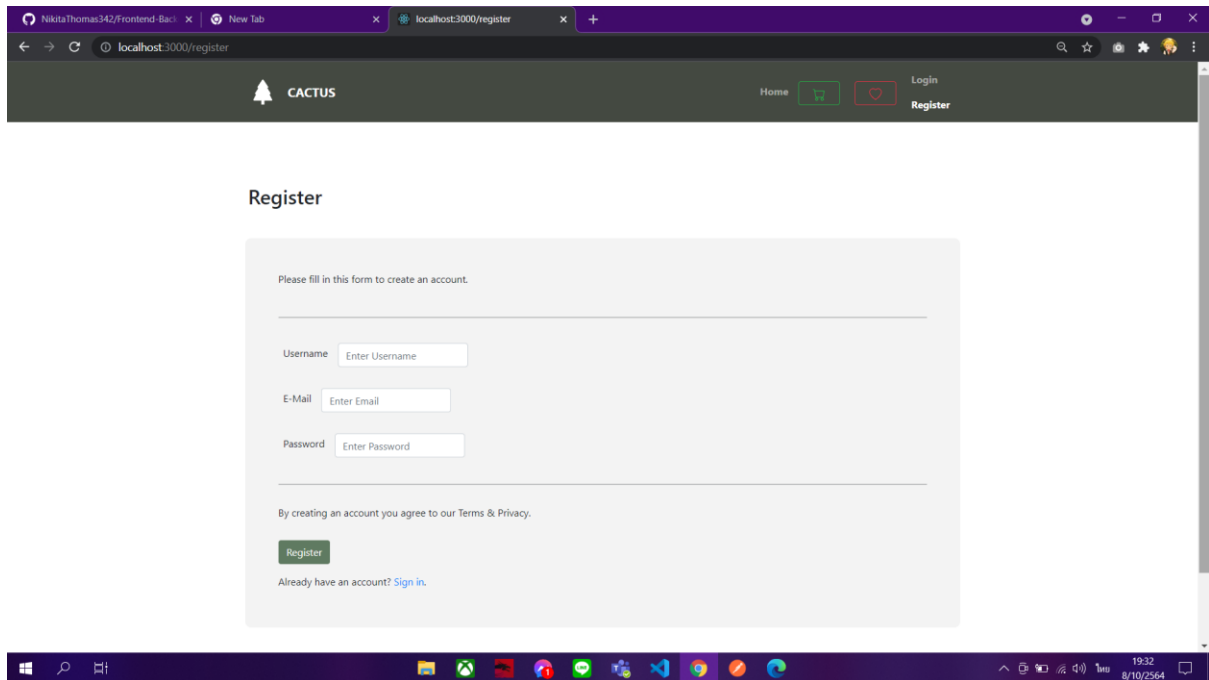
หน้า Home (ยังไม่ได้login)



หน้า Home products

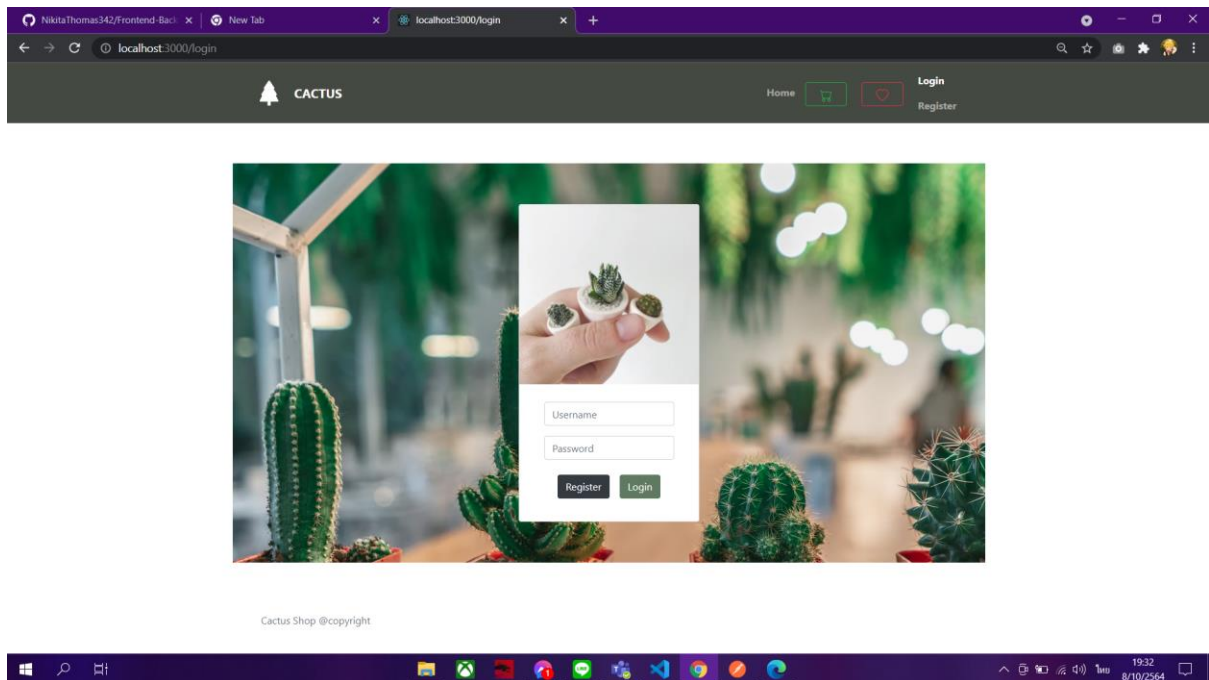


หน้า Register



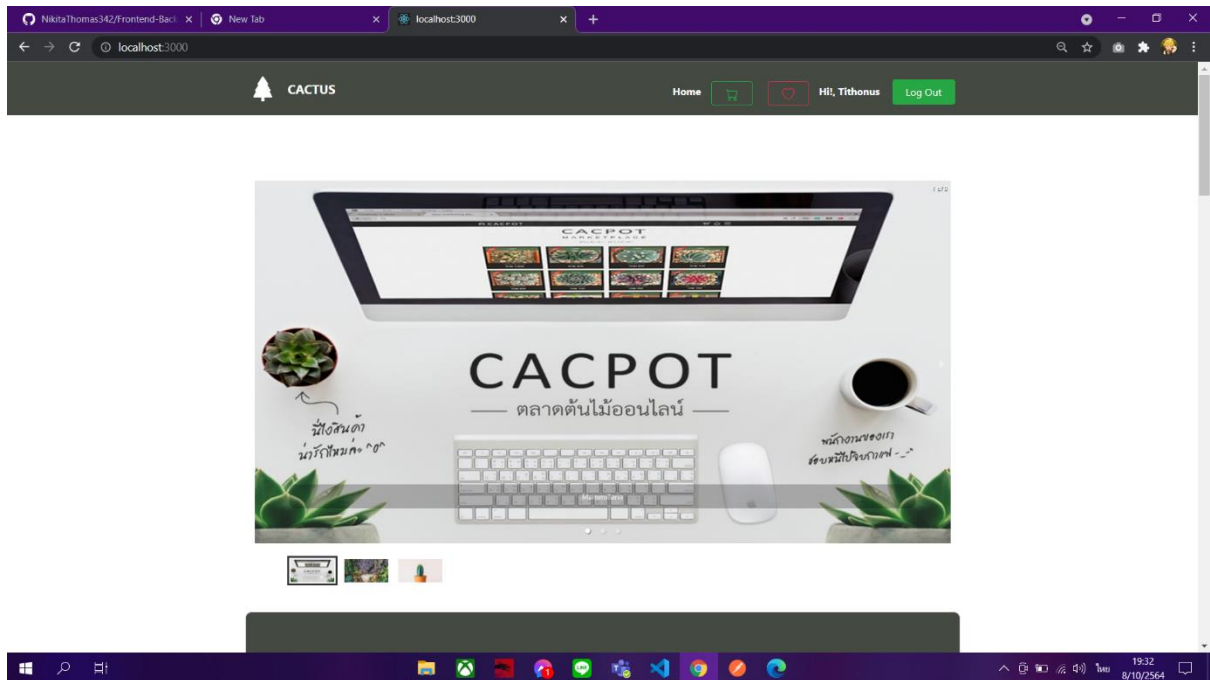
The screenshot shows a web browser window with the URL `localhost:3000/register`. The page has a dark header with the CACTUS logo and navigation links for Home, a shopping cart, a heart icon, and buttons for Login and Register. The main content area is titled "Register" and contains a form with the instruction "Please fill in this form to create an account." The form has three input fields: "Username" with placeholder text "Enter Username", "E-Mail" with placeholder text "Enter Email", and "Password" with placeholder text "Enter Password". Below the fields is a line of text: "By creating an account you agree to our Terms & Privacy." followed by a green "Register" button. At the bottom of the form, it says "Already have an account? [Sign in.](#)". The browser's taskbar at the bottom shows various application icons and the system clock indicating 19:32 on 8/10/2564.

หน้า Login

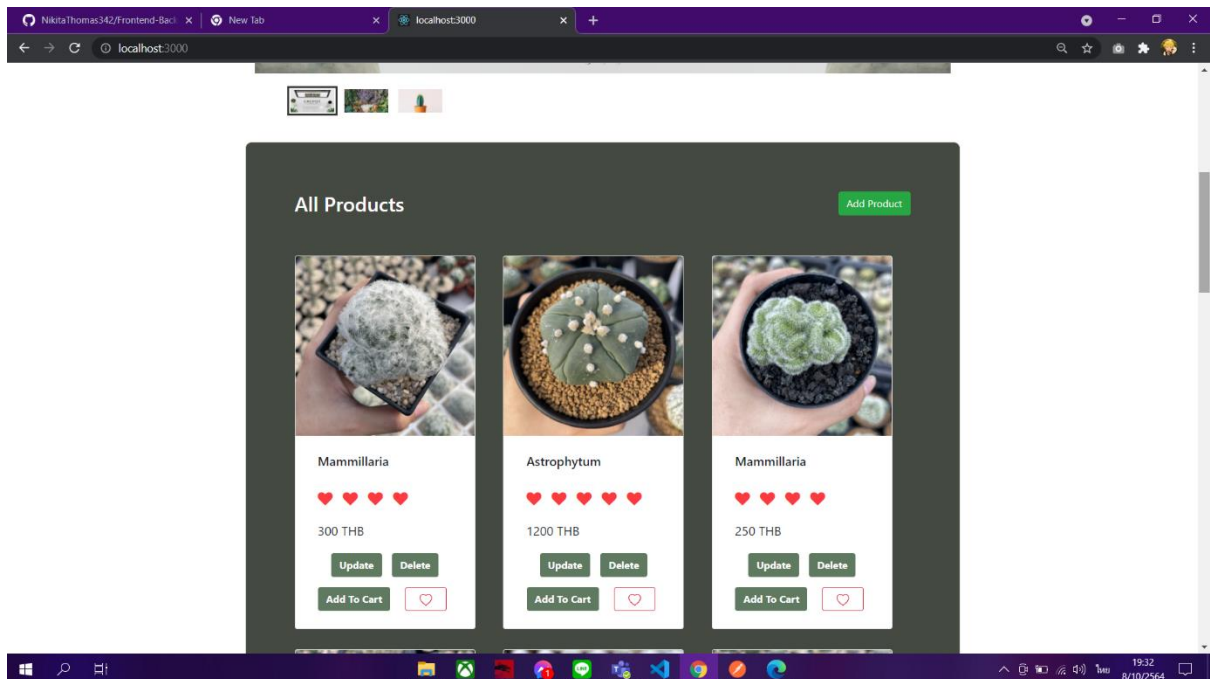


The screenshot shows a web browser window with the URL `localhost:3000/login`. The page features a background image of a hand holding small potted cacti in a greenhouse setting. Overlaid on this is a white login form with "Username" and "Password" input fields, and "Register" and "Login" buttons. The browser header is identical to the Register page. Below the form, the text "Cactus Shop @copyright" is visible. The browser's taskbar at the bottom shows the same application icons and system clock as the previous screenshot.

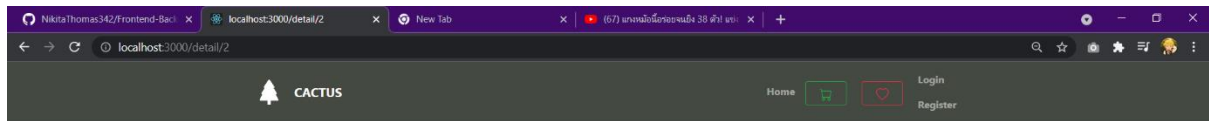
หน้า Home (loginแล้ว)



หน้า Home products (สำหรับadmin)



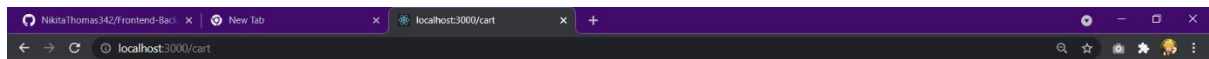
หน้า Detail



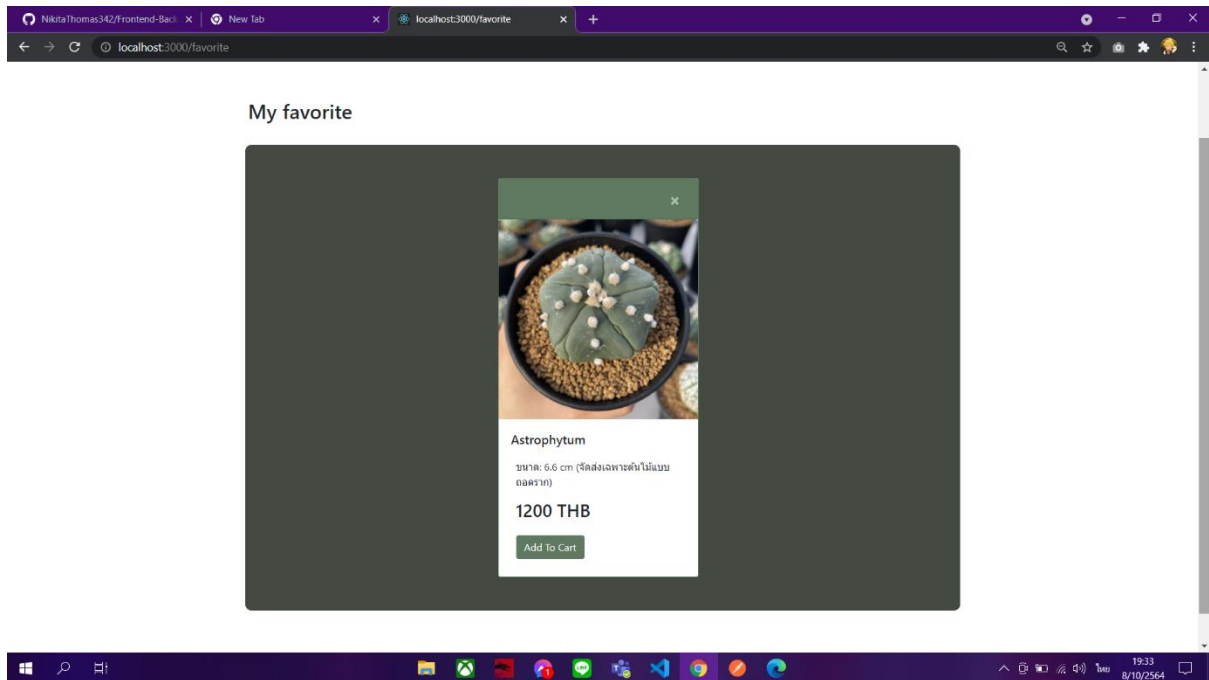
Cactus Shop @copyright



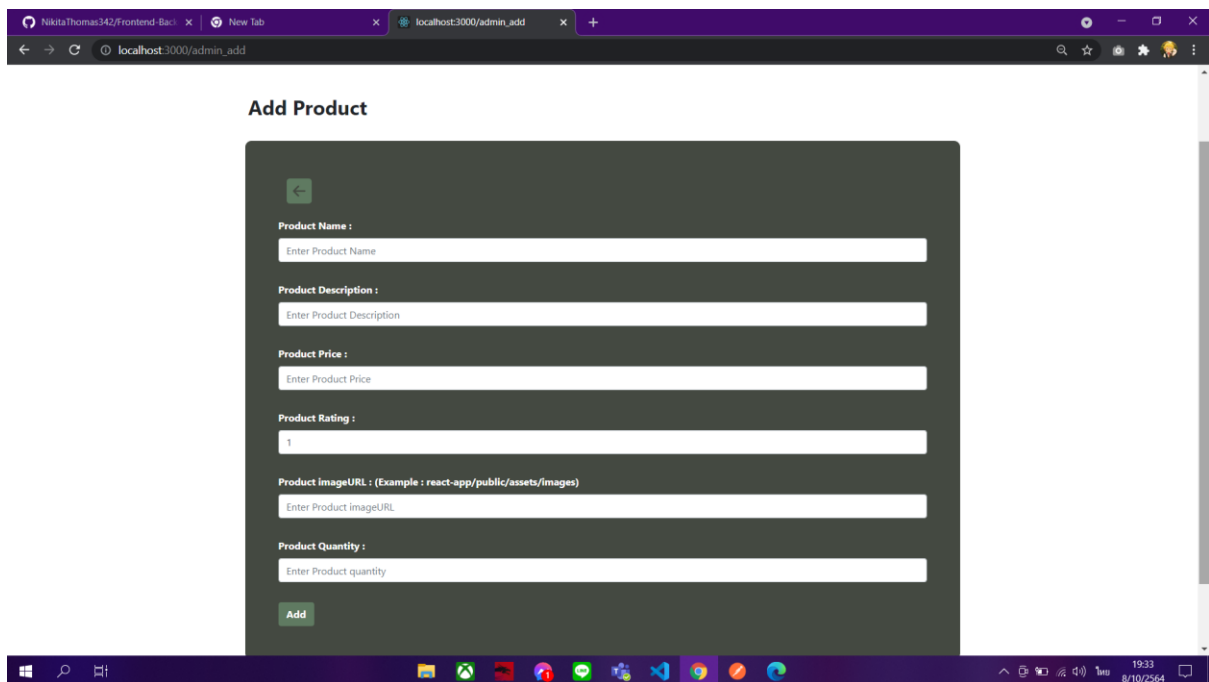
หน้า Cart



หน้า Favorite



หน้า Add products



หน้า Update products

Update Product

←

Product Name , Current : Mammillaria

Enter Product Name

Product Description , Current : ขนาด: 6.5 cm (จัดส่งเฉพาะ: ต้นไม้แบบกึ่งพุ่ม)

Enter Product Description

Product Price , Current : 300 THB

Enter Product Price

Product Rating , Current : 4

1

Product ImageURL , Current : /assets/images/cactus2.jpeg

Enter Product ImageURL

Product Quantity , Current : 20

Enter Product quantity

Update

Database Design

Product

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'plant_store' database with tables, views, stored procedures, and functions. The 'product' table is selected. The 'Field Types' pane shows the table structure:

#	Field	Schema	Table	Type	Character Set	Display Size	Precision	Scale
1	id	plant_store	product	INT UNSIGNED	binary	10	2	0
2	name	plant_store	product	VARCHAR	utf8mb4	200	11	0
3	imageUrl	plant_store	product	VARCHAR	utf8mb4	500	27	0
4	description	plant_store	product	VARCHAR	utf8mb4	1000	101	0
5	rating	plant_store	product	INT	binary	11	1	0
6	price	plant_store	product	INT	binary	11	4	0
7	quantity	plant_store	product	INT	binary	11	2	0

The 'Output' pane shows the query result for 'product 1 x' with the message '14 row(s) returned' and a duration of '0.000 sec / 0.000 sec'.

Cart

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'plant_store' database with tables, views, stored procedures, and functions. The 'cart' table is selected. The 'Field Types' pane shows the table structure:

#	Field	Schema	Table	Type	Character Set	Display Size	Precision	Scale
1	id	plant_store	cart	INT UNSIGNED	binary	10	2	0
2	user_id	plant_store	cart	INT	binary	11	1	0
3	name	plant_store	cart	VARCHAR	utf8mb4	200	11	0
4	imageUrl	plant_store	cart	VARCHAR	utf8mb4	500	27	0
5	description	plant_store	cart	VARCHAR	utf8mb4	1000	101	0
6	rating	plant_store	cart	INT	binary	11	1	0
7	price	plant_store	cart	INT	binary	11	4	0
8	quantity	plant_store	cart	INT	binary	11	2	0

The 'Output' pane shows the query result for 'cart 2 x' with the message '4 row(s) returned' and a duration of '0.000 sec / 0.000 sec'.

Users

The screenshot shows the MySQL Workbench interface. The 'Schemas' pane on the left shows the 'plant_store' database selected. The 'Query' pane shows a query: `SELECT * from users`. The 'Field Types' pane displays the structure of the 'users' table:

#	Field	Schema	Table	Type	Character Set	Display Size	Precision	Scale
1	id	plant_store	users	INT UNSIGNED	binary	10		0
2	username	plant_store	users	VARCHAR	utf8mb4	80	8	0
3	password	plant_store	users	VARCHAR	utf8mb4	200	11	0
4	email	plant_store	users	VARCHAR	utf8mb4	50	20	0
5	state	plant_store	users	TINYINT	binary	1	1	0
6	created	plant_store	users	DATETIME	binary	19	19	0
7	updated	plant_store	users	DATETIME	binary	19	19	0
8	last_login	plant_store	users	DATETIME	binary	19	0	0

The 'Output' pane shows the results of the query, which is empty. The 'Action Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	18:52:12	SELECT * from product LIMIT 0, 1000	14 row(s) returned	0.000 sec / 0.000 sec
2	18:52:55	SELECT * from cart LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
3	18:53:10	SELECT * from users LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
4	18:53:24	SELECT * from favorite LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
5	18:53:36	SELECT * from users LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec

Favorite

The screenshot shows the MySQL Workbench interface. The 'Schemas' pane on the left shows the 'plant_store' database selected. The 'Query' pane shows a query: `SELECT * from favorite`. The 'Field Types' pane displays the structure of the 'favorite' table:

#	Field	Schema	Table	Type	Character Set	Display Size	Precision	Scale
1	id	plant_store	favorite	INT UNSIGNED	binary	10	1	0
2	user_id	plant_store	favorite	INT	binary	11	1	0
3	name	plant_store	favorite	VARCHAR	utf8mb4	200	11	0
4	imageUrl	plant_store	favorite	VARCHAR	utf8mb4	500	27	0
5	description	plant_store	favorite	VARCHAR	utf8mb4	1000	101	0
6	rating	plant_store	favorite	INT	binary	11	1	0
7	price	plant_store	favorite	INT	binary	11	4	0
8	quantity	plant_store	favorite	INT	binary	11	2	0

The 'Output' pane shows the results of the query, which is empty. The 'Action Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	18:52:12	SELECT * from product LIMIT 0, 1000	14 row(s) returned	0.000 sec / 0.000 sec
2	18:52:55	SELECT * from cart LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
3	18:53:10	SELECT * from users LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
4	18:53:24	SELECT * from favorite LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
5	18:53:36	SELECT * from users LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
6	18:53:47	SELECT * from favorite LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec