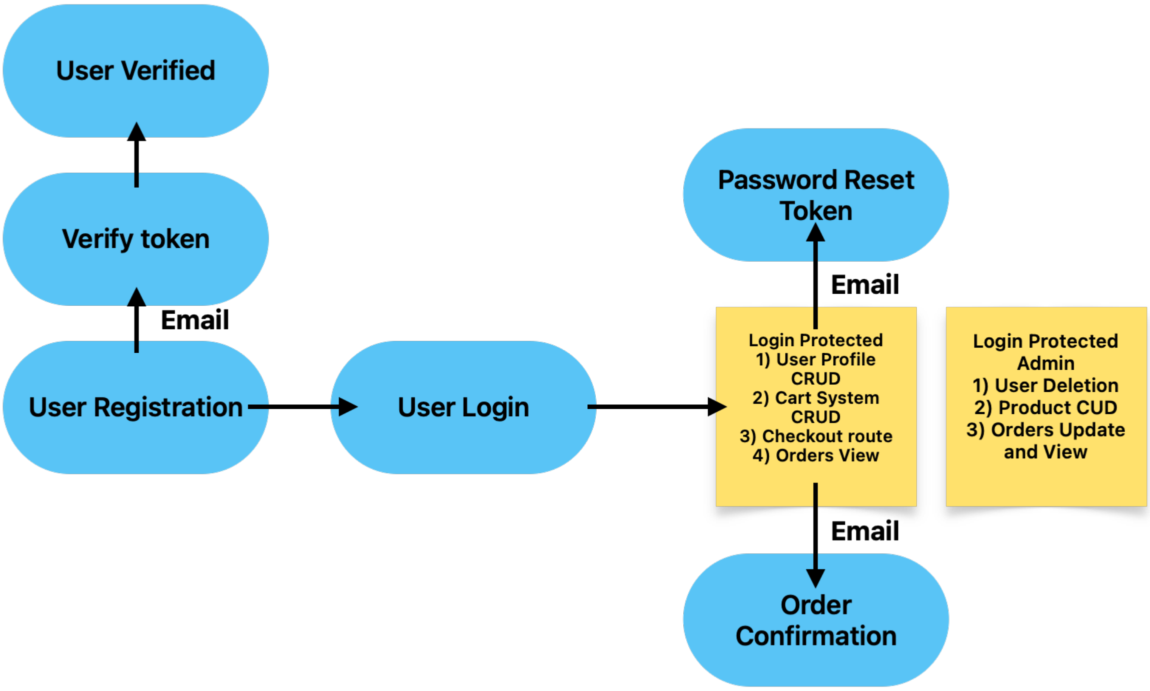
API Flowchart



1. **User Authentication**:
   - The user registers via the POST /auth/register endpoint.
   - The user logs in via the POST /auth/jwt/login endpoint and receives a JWT (JSON Web Token).
   - All subsequent requests must include this JWT in the Authorization header.
2. **User Management**:
   - The user's profile can be retrieved or updated via the GET /users/me and PATCH /users/me endpoints respectively.
   - Specific user profiles can be managed via the GET /users/{id}, PATCH /users/{id}, and DELETE /users/{id} endpoints.
3. **Product Catalog**:
   - All products can be listed via the GET /products endpoint.
   - New products can be added via the POST /products/add endpoint.
   - Existing products can be updated or removed via the PATCH /products/{product_id} and DELETE /products/{product_id} endpoints respectively.
4. **Shopping Cart**:
   - The cart can be viewed via the GET /cart/show endpoint.
   - Items can be added to the cart via the POST /cart/add endpoint.
   - Items can be removed from the cart via the DELETE /cart/remove/{item_id} endpoint.
   - A summary of the cart can be obtained via the GET /cart/summary endpoint.
5. **Checkout and Orders**:
   - The checkout process can be initiated via the POST /checkout endpoint.
   - New orders can be created via the POST /orders/create endpoint.
   - Existing orders can be viewed or edited via the POST /orders/view and PUT /orders/edit/{order_id} endpoints respectively.

Ashte API Endpoints

Authentication Endpoints:

- **POST /auth/jwt/login**: Authenticate and receive JWT.
  - **Input** : username : str, password : str
  - **Output** : Access Token Bearer type
- **POST /auth/jwt/logout**: Terminate the current JWT session.
  - **Input** : Gets access token
  - **Output** : “User successfully logged out”
- **POST /auth/register**: Register a new user account.
  - **Input** :

```
{
  "email": "user@example.com",
  "password": "string",
  "is_active": true,
  "is_superuser": false,
  "is_verified": false,
  "first_name": "string",
  "last_name": "string"
}
```

  - **Output** :

```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "email": "user@example.com",
  "is_active": true,
  "is_superuser": false,
  "is_verified": false,
  "first_name": "string",
  "last_name": "string"
}
```

- **POST /auth/forgot-password**: Request a password reset.

  - **Input :**

```
{
  "email": "user@example.com"
}
```

- **POST /auth/reset-password**: Reset the password using a token.

  - **Input** :

```
{
  "token": "string",
  "password": "string"
}
```

- **POST /auth/request-verify-token**: Request a token for account verification.

  - **Input** :

```
{
  "email": "user@example.com"
}
```

- **POST /auth/verify**: Verify user account with a token.
  - **Input** :

```
{
  "token": "string"
}
```

  - **Output** :

```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "email": "user@example.com",
  "is_active": true,
  "is_superuser": false,
  "is_verified": false,
  "first_name": "string",
  "last_name": "string"
}
```

User Management:

- **GET /users/me**: Retrieve current user’s profile.
- **PATCH /users/me**: Update current user’s profile.
  - **Input** :

```
{
  "password": "string",
  "email": "user@example.com",
  "is_active": true,
  "is_superuser": true,
  "is_verified": true,
  "first_name": "string",
  "last_name": "string"
}
```

- **GET /users/{id}**: Get a specific user’s profile by ID. (Visible only to superuser)
  - **Input** : id : str
- **PATCH /users/{id}**: Update a specific user’s profile by ID. (Visible only to superuser)
  - **Input** : id : str

```
{
  "password": "string",
  "email": "user@example.com",
  "is_active": true,
  "is_superuser": true,
  "is_verified": true,
  "first_name": "string",
  "last_name": "string"
}
```

- **DELETE /users/{id}**: Delete a specific user’s profile by ID. (Visible only to superuser)
  - **Input** : id : str

Product Catalog:

- **GET /products**: List all products.

  Only for Superuser

- **POST /products/add**: Add a new product.
  - **Input** :

```
{
  "title": "string",
  "description": "string",
  "price": 0,
  "release_date": "2023-12-25",
  "quantity": 0,
  "is_visible": false
}
```

- **PATCH /products/{product_id}**: Update an existing product.
  - **Input** : same Input schema as add product
- **DELETE /products/{product_id}**: Remove a product from the catalog.
  - **Input** : product_id : int

Shopping Cart:

- **GET /cart/show**: Display items in the cart.
- **POST /cart/add**: Add an item to the cart.
  - **Input** : JSON List of product_ids and quantities
- **DELETE /cart/remove/{item_id}**: Remove an item from the cart.
  - **Input** : cart item_id
- **GET /cart/summary**: Get a summary of the cart contents.
  - **Input** : cart_id

Checkout and Orders:

- **POST /checkout**: Initiate the checkout process.
  - **Input** :

```
{
  "cart_id": 0,
  "amount": 0,
  "email": "string",
  "name": "string",
  "phone_no": 0,
  "address": "string",
  "postal_code": "string"
}
```

- **POST /orders/create**: Create a new order.
- **POST /orders/view**: View an existing order.

- **PUT /orders/edit/{order_id}**: Edit an existing order. (Visible only to superuser)
  - **Input** : order_id : int

```
{
  "shipped": true,
  "paid": true
}
```

Miscellaneous:

- **GET /authenticated-route**: Access a route that requires authentication.

Note – Things yet to be done:

1) Routes are not yet tested for production
2) Shift to production database
3) SMTP mail server configuration

Ethereal Mail Config –
(It uses faker.js to emulate the smtp server but it's not a real smtp server , every mail sent and received will be stuck on it's inbox, perfect for testing)

All the mails are visible in ethereal mail.com

```
conf = ConnectionConfig(
   MAIL_USERNAME="creola.nicolas@ethereal.email",
   MAIL_PASSWORD="9V4bh27eZK1rbnDDc4",
   MAIL_FROM="creola.nicolas@ethereal.email",
   MAIL_PORT=587,
   MAIL_SERVER="smtp.ethereal.email",
   MAIL_STARTTLS = True,
   MAIL_SSL_TLS = False
)
```

Author : Kaustubh M (Github repo- https://github.com/Sirczechs-Kaustubh/Ashte-fastapi)