**Group 20 Smart Recipe Recommender Project Write-Up**

#### Algorithm and Processes

The backend of our Smart Recipe Recommender and Meal Planner utilizes a modular and scalable design, incorporating multiple key algorithms and methodologies to ensure secure, efficient, and accurate functionality.

1. **Authentication and Security**

The `auth.py` module implements a secure authentication mechanism using JSON Web Tokens (JWT). The algorithm:

- Leverages the HS256 algorithm for token signing, ensuring both performance and security.

- Incorporates password hashing using bcrypt, a proven cryptographic hashing algorithm.

- Validates user credentials and securely encodes user identity in JWT tokens, which expire after a predefined duration for enhanced security.

This approach was chosen for its combination of performance and the robust security needed for managing sensitive user data.

2. **Database and Persistence**

The `db.py` module uses SQLAlchemy, a powerful ORM, to manage interactions with the database. The configuration:

- Establishes a connection to a database defined by the `DATABASE\_URL` from a `.env` file for secure and flexible configuration.

- Implements session management to ensure efficient and reliable query execution.

SQLAlchemy was chosen for its ease of use, compatibility with various databases, and the ability to define complex relationships between data models.

3. **Data Validation and Input Processing**

The `pipe\_a\_text\_into\_csv.py` script ensures robust user input validation for ingredient entries:

- Validates ingredient names using regular expressions, ensuring only valid characters are accepted.

- Confirms units against a predefined set of valid options (e.g., "grams", "cups").

- Validates quantities as positive integers with reasonable limits to avoid processing errors or unrealistic data.

This detailed validation ensures data integrity, preventing errors during further processing or recommendation generation.

4. **User Preferences Management**

The `generate\_preferences.py` script generates JSON files containing predefined dietary preferences, including nutritional goals for diets such as vegetarian, keto, and bulking. The data structure is:

- Flexible for adding new dietary types or modifying nutritional goals.

- Easily parsed by other modules, enabling customized recommendations.

JSON was chosen for its simplicity, readability, and compatibility with web and API-based systems.

5. **API Design**

The `main.py` script defines the FastAPI-based backend:

- Utilizes routers to modularize functionality into ingredients, recipes, meal plans, and recipe agent features.

- Integrates CORS middleware, enabling cross-origin requests and ensuring usability across different frontend platforms.

- Exposes a custom OpenAPI schema for documentation, enhancing API usability for developers.

FastAPI was chosen for its high performance, modern features, and automatic documentation generation.

**#### Reason for Algorithm Choices**

The algorithms and frameworks were selected to address the following project requirements:

- **Scalability**: Using SQLAlchemy and FastAPI ensures that the system can handle increasing data and user loads with minimal performance degradation.

- **Security**: Incorporating JWT and bcrypt protects sensitive user information and prevents unauthorized access.

- **Maintainability**: Modular design with clearly defined responsibilities makes the codebase easier to extend and debug.

- **User-Centric Design**: Validation logic and preference generation ensure that user inputs are meaningful and aligned with their dietary goals.

- **Interoperability**: JSON-based preferences and standardized APIs allow seamless integration with other systems, such as frontend applications.

This combination of algorithms and processes ensures that the system is robust, user-friendly, and secure while remaining adaptable to future enhancements.