SEG 4105

Deliverable # 3

Mamadou Diallo 300203858

Tia El Masry 300160596

Weekly meeting minutes:

Week #1:

Objective: Introduce the project and discuss the overall goal of the recommendation algorithm.

Agenda:

- Overview of the project's purpose and scope.
- Discuss the importance of user preference algorithms in enhancing user experience.
- Initial brainstorming on how to implement user preference collection and analysis.
- Define milestones and goals for the project.

Expected Outcome: Clear understanding of the project objectives and a preliminary plan for the algorithm's implementation.

Week #2:

Objective: Dive into the specifics of the user preferences algorithm.

Agenda:

- Discuss different methods to capture user preferences (e.g., surveys, initial user profile setup).
- Outline the algorithm's logic and flow.
- Assign tasks for coding the initial version of the user preferences algorithm.

Expected Outcome: A detailed plan for the user preferences algorithm and task assignments for development.

Week #3:

Objective: Combine user history data with user preferences.

Agenda:

- Review the progress on the user preferences algorithm.
- Discuss methods to track and integrate user history (e.g., watch duration).
- Plan the integration of user history with user preferences in the recommendation logic.

Expected Outcome: An integrated approach to user preferences and history in the recommendation algorithm.

Week #4:

Objective: Focus on integrating and testing the algorithm with the database.
Agenda:

- Review the integrated algorithm for user history and preferences.
- Discuss database structure and integration.
- Plan and assign testing procedures.

Expected Outcome: A functioning algorithm integrated with the database, ready for testing.

Week #5:

Objective: Focus on implementing the database with Docker to ensure a robust and scalable environment.
Agenda:

- Discuss the current state and structure of the database.
- Introduce Docker and its benefits for the project (e.g., containerization, consistency across environments).
- Plan and assign tasks for integrating the database with Docker.
- Outline steps for initial testing of the database within the Docker environment.

Expected Outcome: A clear plan for database implementation using Docker, with tasks assigned for setup and testing.

Week #6:

Objective: Refine the recommendation algorithm to include user interaction data and test the backend using Postman.
Agenda:

- Review progress on Docker database implementation.
- Discuss the integration of user interaction data (e.g., clicks on descriptions) into the recommendation algorithm.
- Introduce Postman and its role in testing the backend API without the frontend.
- Plan and assign tasks for creating mock product data in WordPress, which will serve as the source for videos and categories.
- Outline steps for testing the updated recommendation algorithm using Postman.

Expected Outcome: A refined recommendation algorithm that incorporates user interaction data, and a clear testing strategy using Postman and mock WordPress data.

Evolution of Scope Maps:

Here we will outline the things that were done during the 6 weeks:
Week 1: Introduction and Project Overview

- **Prepare Project Introduction Materials:** Develop a presentation or document outlining the project's scope and objectives.
- **Brainstorm User Preference Collection Methods:** List possible ways to gather user preferences (e.g., user profiles, sign-up form).
- **Outline Algorithm Logic:** Draft a basic flowchart or outline for the user preferences algorithm.
- **Set Milestones and Goals:** Define key milestones and deliverables for the project.
- **Organize Team Meeting:** Schedule and prepare for the introductory team meeting.

Week 2: User Preferences Algorithm Design

- **Detail Algorithm Design:** Develop a detailed plan for the user preferences algorithm, including data flow and logic.
- **Assign Development Tasks:** Divide the algorithm development tasks among team members.
- **Research Best Practices:** Gather information on best practices for preference algorithms.
- **Prepare Progress Report:** Compile a report on the week's progress for the team meeting.

Week 3: Integrating User History and Preferences

- **Review Preference Algorithm Progress:** Assess the development status of the user preferences algorithm.
- **Plan User History Integration:** Outline how user history (like watch duration) will be integrated.
- **Assign Integration Tasks:** Distribute tasks related to integrating user history and preferences.
- **Update Progress Report:** Update the progress report for the team meeting.

Week 4: Database Integration and Testing

- **Integrate Database:** Begin integrating the recommendation algorithm with the database.
- **Test Integration:** Conduct initial tests for database integration.
- **Debug and Optimize:** Identify and resolve any issues found during testing.
- **Prepare Integration Report:** Prepare a report detailing the integration and testing outcomes for the meeting.

Week 5: Database Implementation and Docker Integration

- **Review Database Structure:** Ensure the current database structure supports the recommendation algorithm.
- **Setup Docker: I**nstall Docker and create containers for the database.
- **Migrate and Test Database in Docker:** Migrate the database and test its functionality in Docker.
- **Document Docker Implementation:** Document the Docker setup process and any issues encountered.

Week 6: Interaction-Based Recommendation Refinement and Testing with Postman

- **Refine Recommendation Algorithm:** Update the algorithm with user interaction data.
- **Learn and Setup Postman:** Familiarize the team with Postman for API testing.
- **Create Mock Data in WordPress:** Develop and upload mock product data.
- **Conduct Postman Testing:** Test the backend API with Postman.
- **Compile Testing Results:** Document the test results and any bugs for discussion in the team meeting.

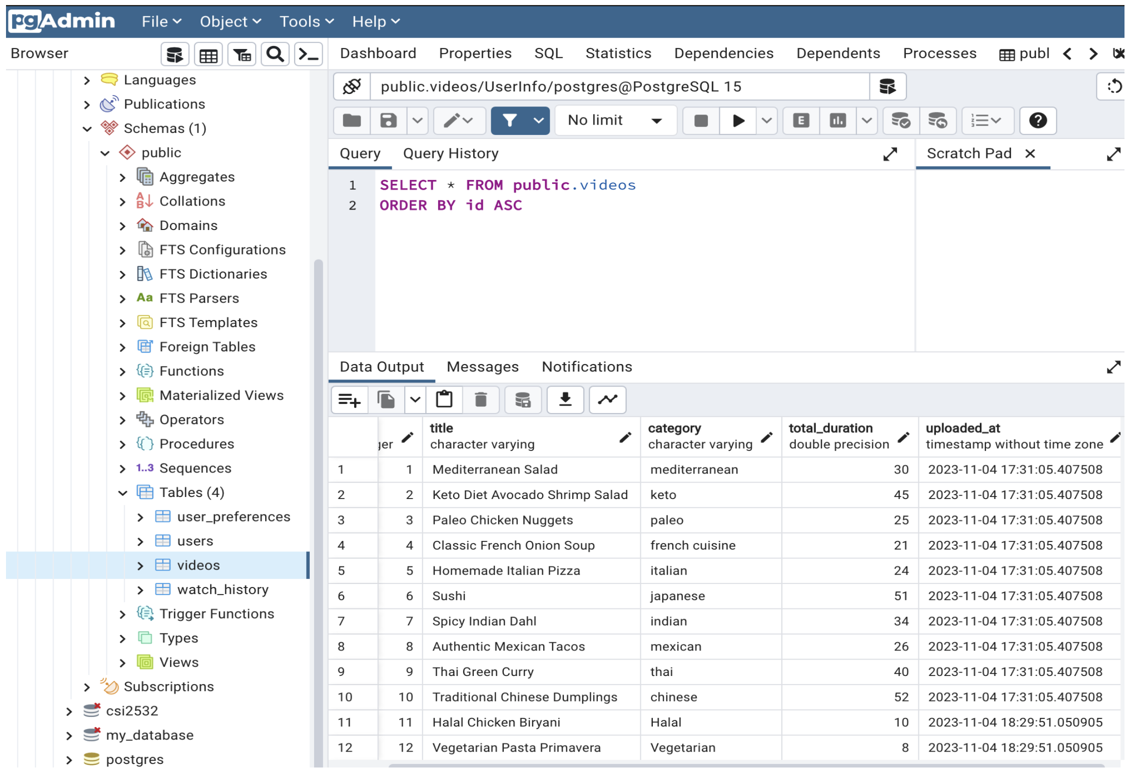
Here are the remaining tasks that were not done throughout the cycle:
Remaining Tasks:

- **Implement Click Interaction Tracking**
  - Task: Develop functionality to capture and analyze user clicks on items.
  - Details: Ensure accurate logging of clicks, integrate this data into the recommendation algorithm, and maintain user privacy and data security.
- **Populate WordPress Content via Docker**
  - Task: Set up and manage WordPress content within Docker containers, ensuring seamless integration with the app.
  - Details: Create a robust system for content update and synchronization, test content loading and real-time updates, and train content managers.

During those 6 weeks, we had to develop that algorithm. To ensure it was working, we had to use Postman to run the tests in the backend. As our project is an existing app, we are working with a startup where each person has his task. As I was assigned to the backend, the best way to make sure it was working was through Postman. Later on, a frontend will be implemented which will take the user preference by filling a form when a person creates an account. When we have

no data on the person, it will just render the trending videos. We set up a database in PGAdmin, created some users, and populated the database to see if it would work.
For example, this first image is the available categories, it is based on a video that will be uploaded by the Chefs, one person may like something but that category may not be in the database. It will just return what's available through short videos.

IMAGE OF THE VIDEOS IN PGADMIN:

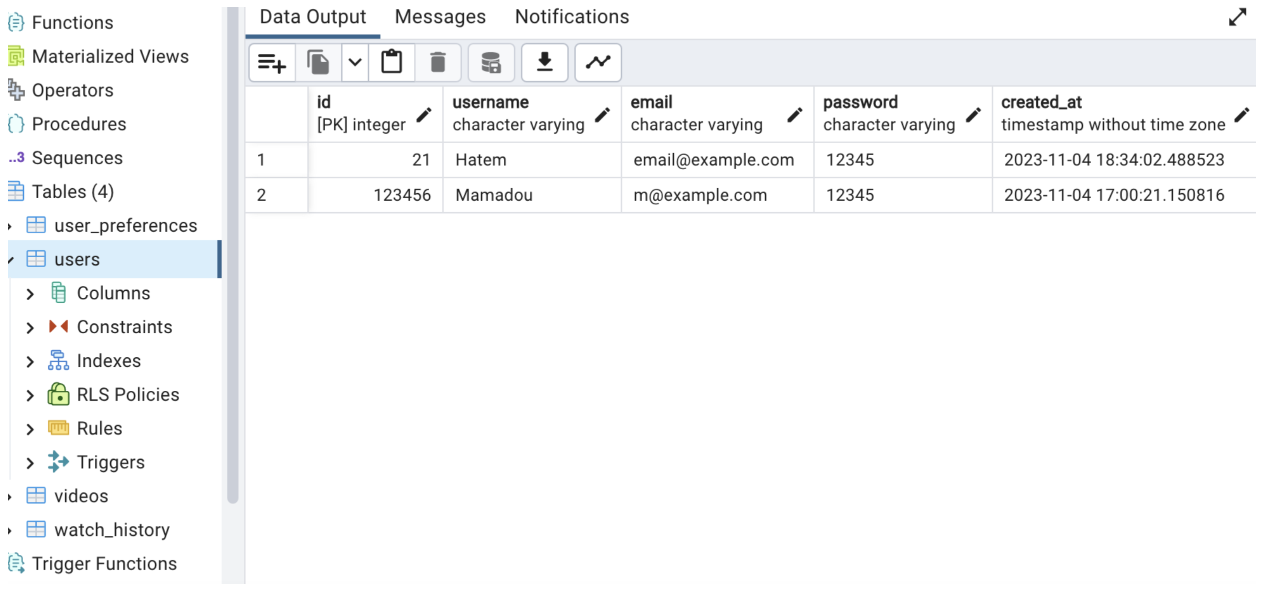


public.videos/UserInfo/postgres@PostgreSQL 15

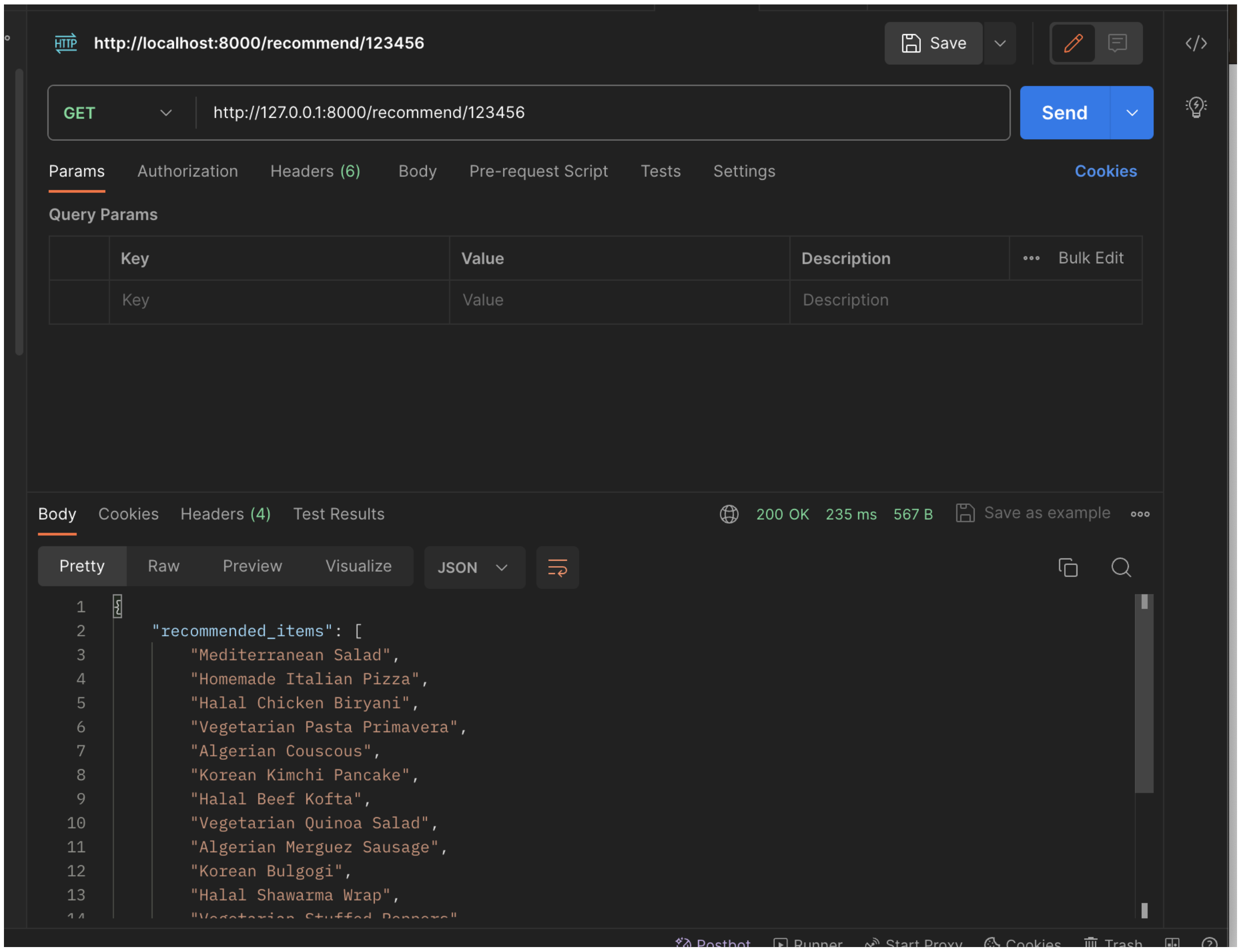
```
SELECT * FROM public.videos
ORDER BY id ASC
```

	jer	title character varying	category character varying	total_duration double precision	uploaded_at timestamp without time zone
1	1	Mediterranean Salad	mediterranean	30	2023-11-04 17:31:05.407508
2	2	Keto Diet Avocado Shrimp Salad	keto	45	2023-11-04 17:31:05.407508
3	3	Paleo Chicken Nuggets	paleo	25	2023-11-04 17:31:05.407508
4	4	Classic French Onion Soup	french cuisine	21	2023-11-04 17:31:05.407508
5	5	Homemade Italian Pizza	italian	24	2023-11-04 17:31:05.407508
6	6	Sushi	japanese	51	2023-11-04 17:31:05.407508
7	7	Spicy Indian Dahl	indian	34	2023-11-04 17:31:05.407508
8	8	Authentic Mexican Tacos	mexican	26	2023-11-04 17:31:05.407508
9	9	Thai Green Curry	thai	40	2023-11-04 17:31:05.407508
10	10	Traditional Chinese Dumplings	chinese	52	2023-11-04 17:31:05.407508
11	11	Halal Chicken Biryani	Halal	10	2023-11-04 18:29:51.050905
12	12	Vegetarian Pasta Primavera	Vegetarian	8	2023-11-04 18:29:51.050905

HERE WE ALSO HAVE THE USERS THAT WERE CREATED THROUGH POST METHOD:

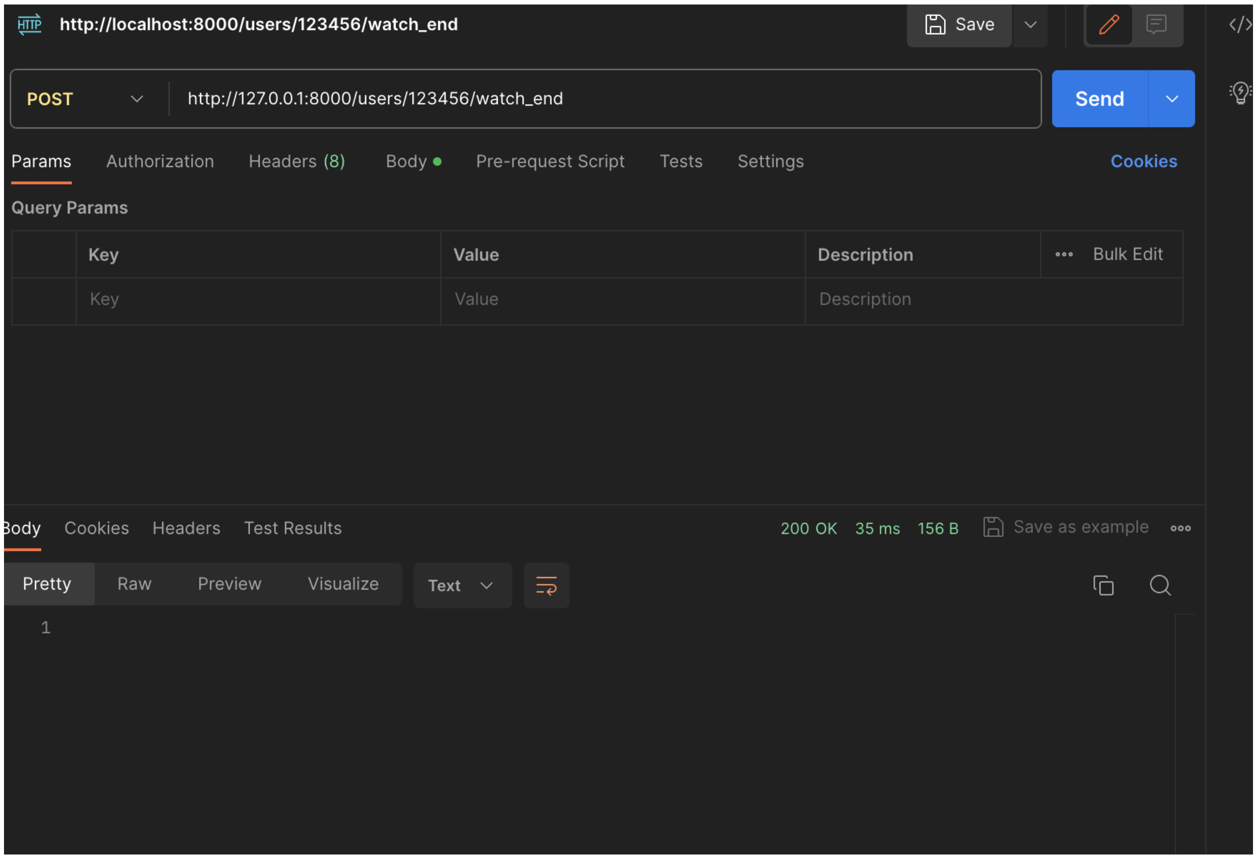


	id [PK] integer	username character varying	email character varying	password character varying	created_at timestamp without time zone
1	21	Hatem	email@example.com	12345	2023-11-04 18:34:02.488523
2	123456	Mamadou	m@example.com	12345	2023-11-04 17:00:21.150816



GET http://127.0.0.1:8000/recommend/123456

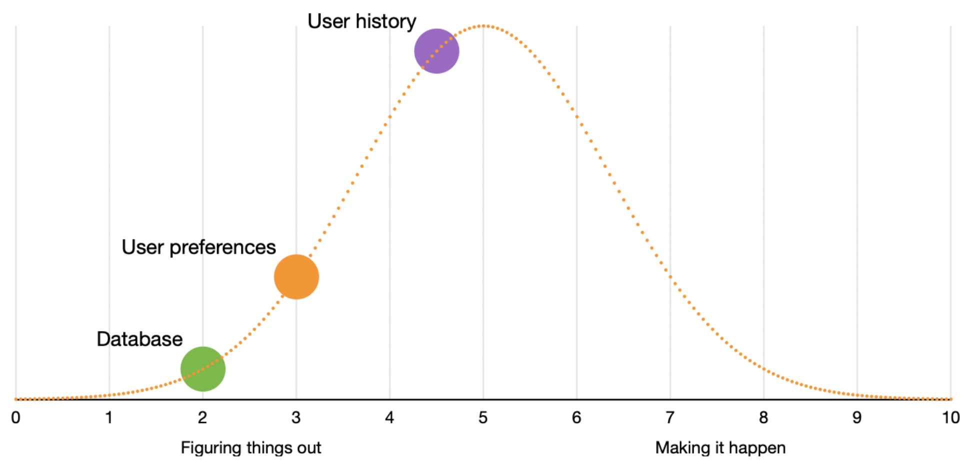
```
{
    "recommended_items": [
        "Mediterranean Salad",
        "Homemade Italian Pizza",
        "Halal Chicken Biryani",
        "Vegetarian Pasta Primavera",
        "Algerian Couscous",
        "Korean Kimchi Pancake",
        "Halal Beef Kofta",
        "Vegetarian Quinoa Salad",
        "Algerian Merguez Sausage",
        "Korean Bulgogi",
        "Halal Shawarma Wrap",
```



With the images from Postman, we can see how we created those elements with the user preference, and the watch history and we also used trending videos in case it was a new client that did not fill the form.

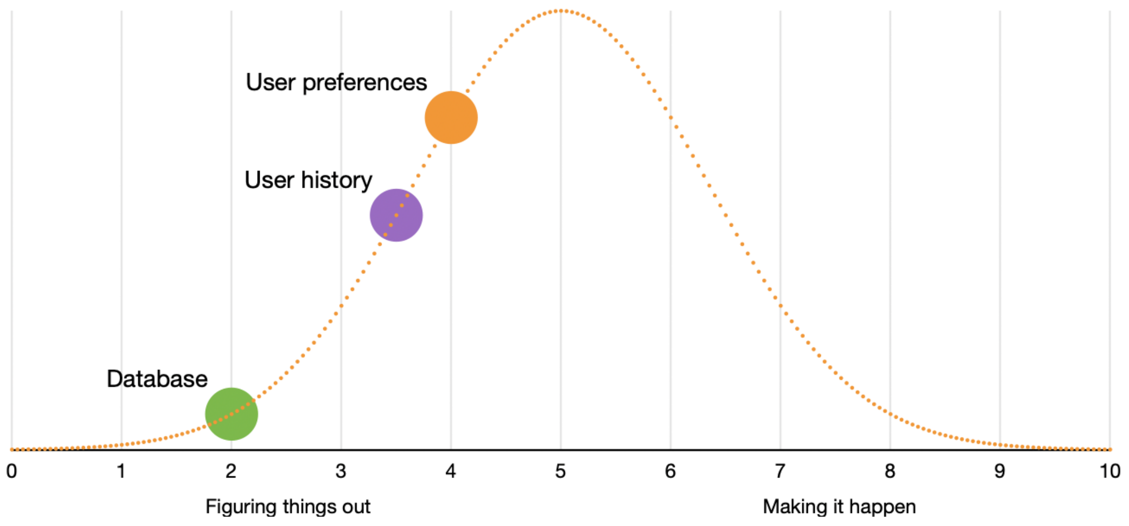
Evolution of Hill Charts:

Hill Chart #1



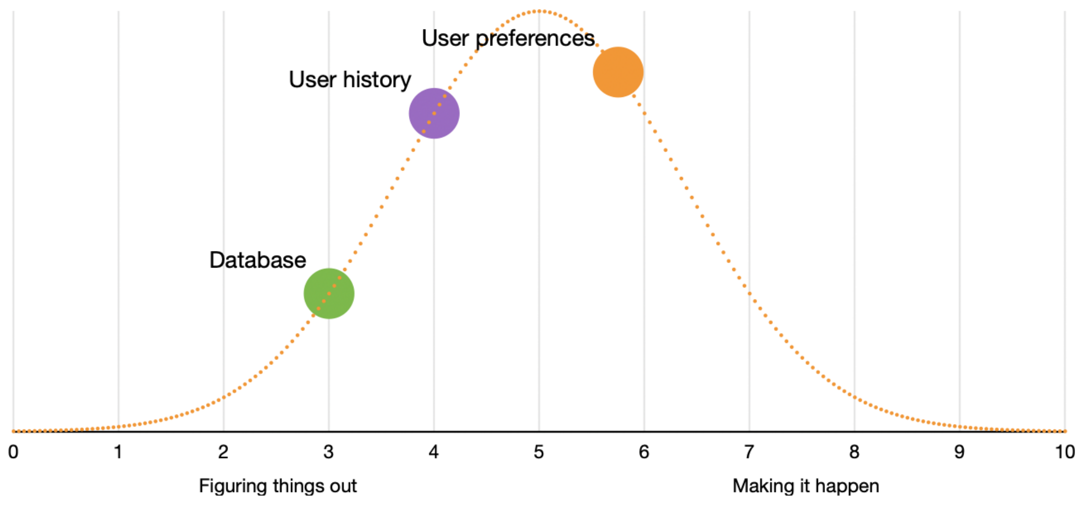
	Scope name	X	Notes
🟩	Database	2	
🟧	User preferences	3	
🟪	User history	4.5	
🟥			
⬛			
🟦			
🟩			
🟧			
🟪			

Hill Chart #2



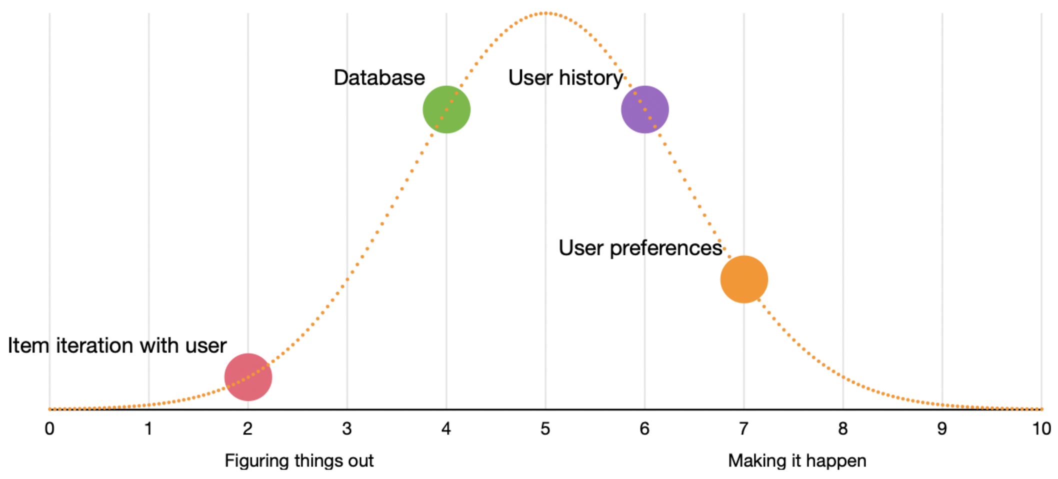
	Scope name	X	Notes
🟩	Database	2	
🟧	User preferences	4	
🟪	User history	3.5	
🟥			
⬛			
🟦			
🟩			
🟧			
🟪			

Hill Chart #3



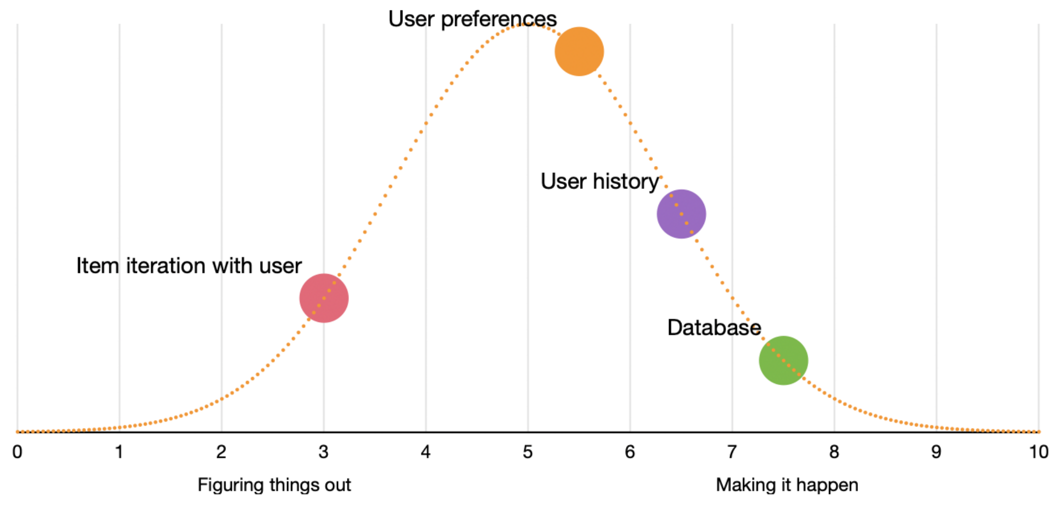
	Scope name	X	Notes
🟩	Database	3	
🟧	User preferences	5.75	
🟪	User history	4	
🟥			
⬛			
🟦			
🟩			
🟧			
🟪			

Hill Chart #4



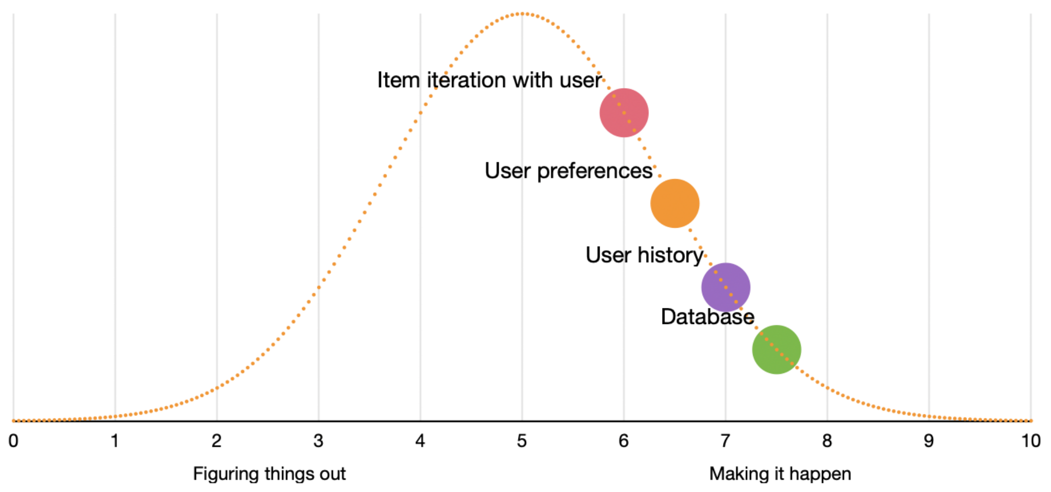
	Scope name	X	Notes
🟩	Database	4	
🟧	User preferences	7	
🟪	User history	6	
🟥	Item iteration with user	2	
⬛			
🟦			
🟩			
🟧			
🟪			

Hill Chart #5



	Scope name	X	Notes
🟩	Database	7.5	
🟧	User preferences	5.5	
🟪	User history	6.5	
🟥	Item iteration with user	3	
⬛			
🟦			
🟩			
🟧			
🟪			

Hill Chart #6



	Scope name	X	Notes
🟩	Database	7.5	
🟧	User preferences	6.5	
🟪	User history	7	
🟥	Item iteration with user	6	
⬛			
🟦			
🟩			
🟧			
🟪			

Detailed Overview of Delivered Features:

Introduction:

- **Purpose**: This section documents the key features delivered in the video recommendation service, showcasing how each contributes to enhancing user experience and system functionality.

- **Value**: Highlight the significance of these features in providing a personalized, efficient, and user-friendly video discovery platform.

Core Features:

- **User Preferences and History Tracking**
    - **Description**: This feature captures individual user preferences (e.g., genre, artist, video length) and watch history. It's crucial for tailoring video recommendations to each user's unique taste.
    - **Implementation**: Built with FastAPI for backend processing, using SQLAlchemy for data management. The system records user interactions and viewing patterns in a PostgreSQL database.
- **Video Recommendation Engine**
    - **Description**: The engine analyzes user data to suggest videos. It utilizes a combination of collaborative filtering, content-based filtering, and user interaction data to enhance recommendation accuracy.
    - **Implementation**: Implemented using Python algorithms, potentially integrated with machine learning libraries (e.g., TensorFlow, scikit-learn) for advanced data analysis.
- **Database Integration**
    - **Description**: Robust database integration to manage extensive user data, video metadata, and interaction logs. Ensures fast, reliable access and updates to data.
    - **Implementation**: Utilizes PostgreSQL for its reliability and scalability. Integrated via SQLAlchemy ORM for efficient data querying and manipulation.
- **API Endpoints**
    - **Description**: Comprehensive set of RESTful API endpoints, enabling operations like starting/stopping video watch, setting preferences, and fetching recommendations.
    - **Implementation**: Developed with FastAPI, ensuring high performance and easy scalability. Security measures like JWT for secure API access.

Additional Features:

- **Category-Based Filtering**
    - **Description**: Allows users to filter videos by categories. This feature enhances user navigation and discovery of videos aligned with their interests.

    - **Implementation**: Integrated with a category management system, possibly linked with an external API for dynamic category updates.
- **Docker Deployment**
    - **Description**: Utilizes Docker for deploying the application, ensuring consistency across different environments and simplifying deployment processes.
    - **Implementation**: Includes Dockerfile for container setup and Docker Compose for managing multi-container applications.

Future Deliverables

- **Upcoming Features**: Outline planned enhancements like improved machine learning algorithms for recommendations, additional social features, etc.
- **Timeline**: Provide an estimated timeline or version number for these upcoming features.