Joke Generator – Database Layer

In our project, as users and the jokes they generate will eventually form a large dataset, a reliable and scalable database solution was needed. MSSQL meets this need with its high-performance query engine, data integrity mechanisms, and advanced security features. Additionally, MSSQL provides a solid foundation for long-term system stability, managing user histories, and analyzing large volumes of data.

It would be possible to keep such a project simple, without using a database, just to show a quick joke. However, using a database makes significant contributions to the project for both the user and the developer. Let's take a look at these advantages and disadvantages.

◆ Advantages of Using a Database

- From the Project's Perspective:
  - **Data Persistence:** Generated jokes are saved and remain even if the system shuts down.
  - **History Management:** Users' past jokes can be listed and statistics can be kept.
  - **Scalability:** The system can adapt as the number of users and data grows.
  - **Data Analysis:** It becomes possible to analyze which types of jokes are generated more often.
  - **Data Integrity:** The relationship between users and jokes is consistently maintained.
- From the User's Perspective:
  - **Access to History:** Users can revisit the jokes they previously created.
  - **Personalization:** The system can offer suggestions based on past preferences.
  - **Sense of Security:** Knowing their data won't be lost improves the user experience.

◆ Disadvantages of Using a Database

- From the Project's Perspective:
  - **Extra Development Load:** Tables must be created, and connections must be set up.
  - **Server Requirement:** A server or service is needed to host the database.
  - **Maintenance Needs:** Concerns like data security, backups, and performance must be managed.
- From the User's Perspective:
  - **Registration Requirement:** Collecting user data may be necessary to store history.
  - **Privacy Concerns:** Users may be uncomfortable with their content being stored (privacy issues).

Database Structure

❖ Users Table(Users)

This table stores information about the people using the application. It includes each user's unique ID, username, email address, and registration date. This helps manage user identities in an organized way..The structure of the table is as shown in the screenshots

	Column Name	Data Type	Allow Nulls
🔑	UserId	int	☐
	Username	varchar(50)	☐
	Email	varchar(100)	☐
	CreatedAt	datetime	☑
			☐

```
SELECT TOP (1000) [UserId]
      ,[Username]
      ,[Email]
      ,[CreatedAt]
  FROM [JokeGenerator].[dbo].[Users]
```

❖ Jokes Table(Jokes)

This table stores the jokes created by users. It contains information about which user wrote which joke and when. It's used to keep a record of user history and to enable content analysis..The structure of the table is as shown in the screenshots

	Column Name	Data Type	Allow Nulls
🔑	JokeId	int	☐
	UserId	int	☐
	[Content]	text	☐
▶	CreatedAt	datetime	☑
			☐

```
SELECT TOP (1000) [JokeId]
      ,[UserId]
      ,[Content]
      ,[CreatedAt]
  FROM [JokeGenerator].[dbo].[Jokes]
```

❖ Relationship (User – Jokes)

In this structure, the UserId column in the Jokes table is related to the UserId column in the Users table. This relationship creates a clear connection between each joke and the user who created it.

Thanks to this foreign key relationship:

- We can easily track which joke belongs to which user.
- Data integrity is preserved by ensuring that every joke is linked to a valid user.
- It becomes easier to perform user-specific queries, such as listing all jokes by a particular user or analyzing joke history.

Risk Table – Project Risks and Mitigations

Risk	Description	Impact	Likelihood	Mitigation Strategy
Data Loss	Database may crash or data may be accidentally deleted	High	Medium	Regular backups, use of transaction safety
Unauthorized Access	User data could be exposed if authentication isn't implemented correctly	High	Medium	Implement secure login, hash passwords
Scalability Issues	If user base grows rapidly, performance may degrade	Medium	Medium	Optimize queries, consider indexing, scale DB
User Privacy Concerns	Storing user-generated jokes may raise privacy issues	Medium	High	Clearly state privacy policy, allow opt-out
Server Downtime	Database or app might go offline due to server issues	Medium	Low	Use reliable hosting, set up monitoring & alerts
SQL Injection	Poorly secured queries could be exploited	High	Low	Always use parameterized queries or ORM
Password Storage Issues	If passwords are stored insecurely, they may be exposed in data breaches	High	Medium	Store hashed & salted passwords using secure algorithms
Nonsensical AI Output	AI-generated jokes may be irrelevant, inappropriate, or lack logical structure	Medium	Medium	Apply content filters and implement basic output validation rules