Important:

- The assets (files) should not be stored in the database, use the filesystem instead!
- Try to keep the final solution KISS.

“User” entity requirements:

1. Storing of user’s email address. Keep in mind that this information must be unique otherwise we won’t be able to successfully identify an individual user.
2. Storing of user’s password. Don’t forget about the security implications.
3. A flag used for determining whether this account is active or not.
4. **Important:** New user accounts should not be automatically activated. They should be activated only after email confirmation. (instead of actually sending the email, you can save it to the filesystem - mail() function maybe won’t work locally).

“Asset” entity requirements:

1. Every asset must be connected to a user and can belong only to one user (the one that uploaded it).
2. Since we’re not storing asset directly in the database, we must store a “reference” to it.
3. There will be a requirement to store some meta-data about the asset itself. Keep this in mind while designing the “Asset” entity.
4. We want to store the number of times an asset was “downloaded”.
5. A flag used for determining whether the asset is public or private.
6. Make sure to protect private assets.

A couple of notes:

- While designing the underlying tables, try to use the appropriate data type and size for fields in question.
- Is the field optional (nullable)?
- Don’t forget to add appropriate constraints (e.g. unique key) were necessary.
- Define foreign keys were necessary.

- Data must be stored in UTF-8.
- Using PDO to interact with the database is strongly encouraged, but not mandatory.

The requirements stated above must be fulfilled, but try to keep the final solution as simple as possible. Do not over-engineer.

An example of a possible frontend implementation: