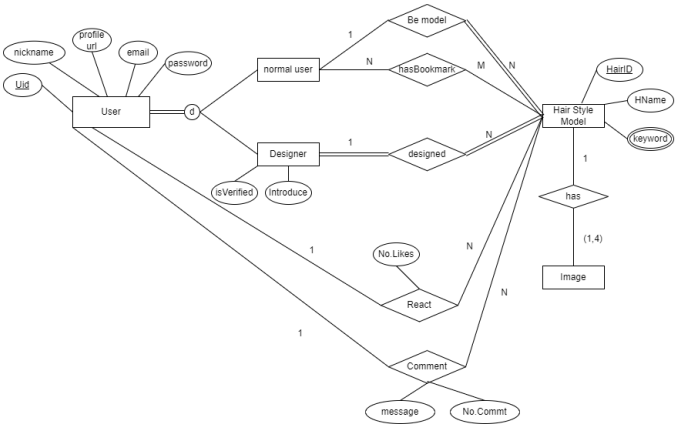
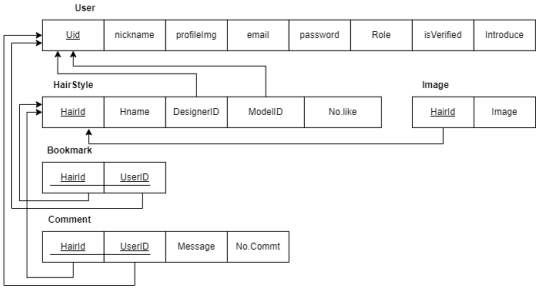
Hair Heist Backend Description

Relational Database Design

Enhanced Entity Diagram & Relational Mapping





General Description

Entity	Description

User	Used to authentication and management the account of the all users in the system.
Normal user	For those user who aren't hair designers, just have demand for finding suitable hairstyle and designer for themselves.
Designer	User who is the hair designer. This user can upload images, and introduction about themselves
Hairstyle	Hairstyles that be uploaded by the designer.
Bookmark	A list of Hair styles that interest normal user.
Comment	Comments of all user in for a specific hairstyle.
Images	Each hairstyle has 1-4 images to describe it. In relational database, just storing the links of images. The source would be store in the cloud.

Detail Description

Feature : User Information

User Model

→ for Authentication

- `String` email : email for sign-in
- `String` password : password for sign-in

- `String` uid

  (unique, primary key) → uuid.v4

  uuid

  For the creation of RFC4122 UUIDs Complete - Support for RFC4122 version 1, 3, 4, and 5 UUIDs Cross-platform - Support for ... Secure - Cryptographically-strong random values

  https://www.npmjs.com/package/uuid

- `String` nickName

  : user’s name (2 - 8 characters will be enough)

- `String` profileImgUrl

  : It’s just url generated from Firebase storage, and it will be handled(create new url, or update new image to storage) by client side.

- `String[]` bookmarks

  : array that store hair style doc id

▼ `Object` designerInfo(nested in User table)

> 💡 → I think this pattern is for NO-sql, maybe u can modify this to be suitable to SQL.(like, divide this from user table, and make other table named “Designer” and connect with foreign key)

: default value is null. Users can add this information anytime after sign-up.

- `String` designerId(secondary key, I don’t know clearly, is this necessary?)

- `Boolean` isVerified

- `String` licenseImgUrl

  : license image url by firebase storage (for verify designer)

- `String` designerImgUrl

  : profile image for designer account

- `String` Introduce

  : Self-introduction about their career.

Methods

- createNewUser(UserModel value) ⇒ void

: create new user to db with new uid (using uuid.v4)

- deleteUserByUid(String uid) ⇒ void

: delete exist user by given uid

- logIn(String email, String password) ⇒ User

: Login to our service and then return current user’s infomation

- fetchUserByUid(String uid) ⇒ UserModel

  : fetching user information by given uid

- createNewDesigner(DesignerInfo value) ⇒ void

  : create new designer info to db

- verifyDesignerByUid(String uid) ⇒ void

  : change isVerified status of designer(given uid)

💡 How to verify?? How to check the license?

Feature : HairStyle

Hair Style Model

- `String` hairId (primary key)
- `string` modelUid (foreign key) : user id of this haircut
- `string` designerUid (foreign key)

  : user id of this haircut's designer
- `string` hairName
- `string[]` keywords : used for searching, classification

  → user can type their own keyword, or they can just use given keywords by default.
- `string[]` array of hair style image urls (1 - 4 images)

→ for community features

- `Number` likes : like rating by users
- `String[]` comments : array for comment ids

Comment Model

-
- `String` hairId (primary key)
- `String` commentId (primary key)
- `String` author (user id)
- `String` message

: comment body

Methods

- createNewHairStyle(HairStyleModel value) ⇒ void
- deleteExistHairStyleById(String hairId) ⇒ void
- updateExistHairStyleById(String hairId) ⇒ void
- fetchHairStyleById(String hairId) ⇒ HairStyleModel
- fetchHairStylesByUid(String userId) ⇒ Array of HairStyleModel
- fetchHairStylesByDesigner(String designerId) ⇒ Array of HairStyleModel
- fetchHairStylesByKeyword(String keyword) ⇒ Array of HairStyleModel
- fetchComments(String hairId) ⇒ Array of Comment Model
- saveToBookmark(String userId, String hairid) ⇒ void
- removeFromBookmark(String userId, String hairId) ⇒ void

Case Tools

- **Nodejs- Backend framework**

1. Provides its APIs directly to developers.
2. Contains a lot of built-in modules to work with the database, handle security …

- **MySQL- Database**

1. Search, run queries, and visualize our relational database.
2. Build model and let the query be the job of sequelize

- **Xampp - Database Server**

1. Easy to use
2. Highly request by online teaching tutorial.

- **Github - Version control and collaboration tools**

1. Easy to managing both group and individual projects.
2. Keeping an online backup of our work.

- **Diagrams.net - Graph sketching**

1. Easy to produce diagrams and other visualizations.
2. Good support for real-time collaboration when used with Google Drive.
3. Completely free service.