



Hat-Tip Database Design

07.11.2021

—

Tye Hickman

Maryville University

SWDV 610 Capstone

Database

The Hat-Tip database will mainly be concerned with document storage. Since the model of what users will be entering and retrieving is very predictable, storing each 'Entry' from a user as a JSON formatted document in MongoDB makes sense.

User profiles can also be stored as JSON formatted documents for the time being. As development progresses, managing user authentication may become integrated in a cloud provider workflow like AWS.

Mongoose will be used to create the TypeScript models in our code that will be sent to MongoDB.

Users

First we have our notion of a user:

```
var User = mongoose.model('User', {  
  email: String,  
  password: String,  
  userId: String,  
  // profileImage: BinData, TODO: figure out BinData data types...  
  Stretch Feature??  
  profileName: String,  
  profilePrompt: String,  
});
```

A stretch feature for our MVP would be to include profile images for users. This will require storing the image as binary data in a MongoDB document which will require research and vetting to find the best practice for the application.

Journal

A user creates an entry from the user interface. These entries are organized in the notion of a Journal object which when called from our user interface, will return a subset of the most recent Journal Entry objects to help populate the UI with fewer service calls.


```
var Journal = mongoose.model('Journal',{
  journalId: String,
  userId: User.userId, // Relates to User collection
  consecutiveDays: Number,
  JournalEntries: [JournalEntry] //Subset of 10-20 Journal entries
  ordered by date to return to UI.
})
```

Here we see a relationship to the User collection as well as to the JournalEntry collection we will discuss below.

Journal Entry

A Journal consists of many entries. While our Journal object will mainly be concerned with reducing the number of service calls loading the User Interface, it also serves the purpose of further identifying entries and associating them with a user.

```
var JournalEntry = mongoose.model('JournalEntry', {
  entryId: String,
  journalId: Journal.journalId, //relates to the Journal collection
  userId: User.userId, // Relates to User collection
  dateCreated: Date,
  entryTitle: String,
  entryPrompt: String,
  entryBody: String
})
```



The Journal Entry collection relates both to our Users collection and to our Journal collection.

Relationships

Here is a quick example of Hat-Tip relational data:

- Users have a one-to-one relationship to Journals
- Journals have a one-to-many relationship with JournalEntry
- Users have an implicit one-to-many relationship with JournalEntry but JournalEntry should be called under the scope of the Journal collection.