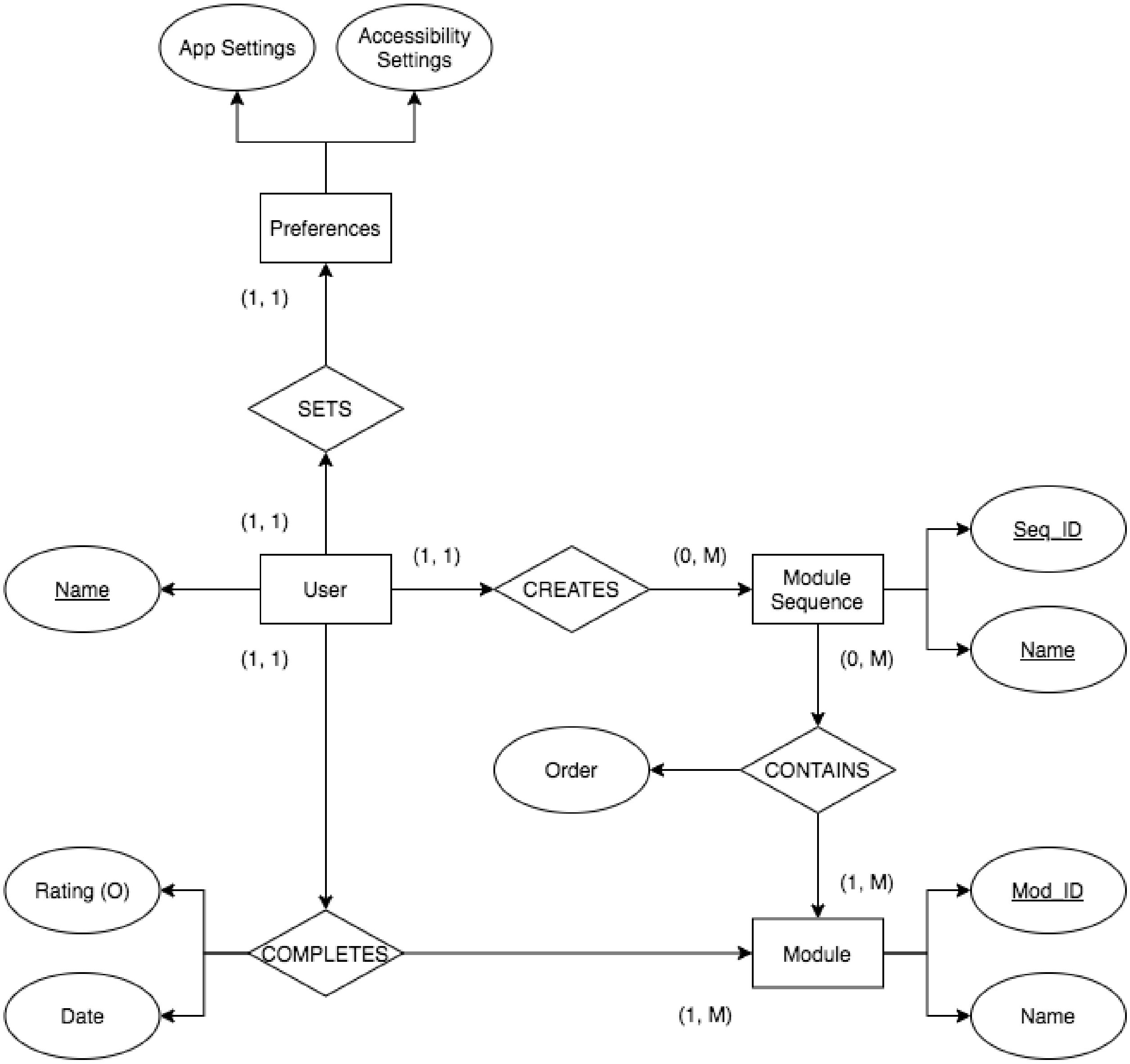
From the project proposal:

> “Multiple users are allowed to create an account and will be identified by a unique name. The user will be able to create a customized sequence of modules in the database that can be launched at any time. To do this, a module entity will also exist in order to keep track of each one’s unique ID. Furthermore, a set of preferences will be associated with each user.”

Entity-Relationship Diagram:



*This diagram uses look-across notation

Relational Model:

Key:

- Primary Key
- **Candidate Key**
- *Foreign Key*

Step 1: Convert Strong Entities.

- User(ID, **Name**)
- Module(ID, Name)
- ModuleSequence(ID, **Name**)
- Preferences({setting_1}, {setting_2}, … , {setting_n})

Step 2: Convert Weak Entities.

- None

Step 3: Binary 1:1 Relationships

- User SETS Preferences
  - Total participation for both entities, so the relations will be merged.
  - User(ID, **Name**, {setting_1}, … , {setting_n})

Step 4: Binary 1:M Relationships

- User CREATES Module Sequence
  - The relationship has no attributes, so a foreign key will be used.
  - Module Sequence has the greater cardinality, so it gets the foreign key.
  - ModuleSequence(*usrID*, ID, **Name**)
- User COMPLETES Module
  - The relationship has attributes, so it will be made into a LOOKUP table.
  - CompletedModule(*usrID*, *modID*, ID, Rating (optional), Date)

Step 5: Binary M:M Relationships

- Module Sequence CONTAINS Module
  - SequenceOrder(*seqID*, *modID*, Order)

Step 6: Multi-Valued Attributes

- None

Step 7: N-ary Relationships

- None

Step 8: Final Model

- User(ID, **Name**, {setting_1}, … , {setting_n})
- Module(ID, Name)
- CompletedModule(*usrID*, *modID*, ID, Rating (optional), Date)
- ModuleSequence(*usrID*, ID, **Name**)
- SequenceOrder(*seqID*, *modID*, Order)

Relational Algebra Test Queries:

Legend:

- σ (select)
- π (project)
- $\bowtie$ (join)
- * (natural join)
- $_{\text{grouping}} G_{\text{function}}$ (aggregate function)

List all of User 1’s module sequences:

R <- $\sigma_{\text{usrID = 1}}$ (ModuleSequence)

List all of the modules in the module sequence 5:

R <- $\sigma_{\text{seqID = 5}}$ (SequenceOrder)

Find the name of User 1’s most completed module.

M <- $\sigma_{\text{usrID = 1}}$ (CompletedModule)

C(modID, count) <- $_{\text{modID}} G_{\text{count(ID)}}$ (M)

R <- π_{Name} ($G_{\text{max(count)}}$ (C))

Find the average rating for each of User 1’s module sequences:

Seq <- $\sigma_{\text{usrID = 1}}$ (ModuleSequence)

M <- $\pi_{\text{seqID, modID}}$ (SequenceOrder * π_{ID} Seq)

CM <- M $\bowtie_{\text{modID = modID} \wedge \text{usrID = 1}}$ CompletedModule

R <- $_{\text{seqID}} G_{\text{average(Rating)}}$ (CM)

Database Schema:

User(ID, **Name**, {setting_1}, … , {setting_n})

```
    CREATE TABLE User(
            ID INTEGER PRIMARY KEY AUTOINCREMENT,
            Name VARCHAR(30) UNIQUE,
            setting_1 INTEGER NOT NULL DEFAULT 0,
            …
            setting_n INTEGER NOT NULL DEFAULT 0)
```

Module(ID, Name)

```
    CREATE TABLE Module(
            ID INTEGER PRIMARY KEY AUTOINCREMENT,
            Name text)
```

CompletedModule(*usrID*, *modID*, ID, Rating (optional), Date)

```
    CREATE TABLE CompletedModule(
            usrID INTEGER REFERENCES User(ID) ON DELETE CASCADE,
            modID INTEGER REFERENCES Module(ID),
            ID INTEGER PRIMARY KEY AUTOINCREMENT,
            Rating INTEGER,
            Date TEXT NOT NULL DEFAULT CURRENT_DATE)
```

ModuleSequence(*usrID*, ID, **Name**)

```
    CREATE TABLE ModuleSequence(
            usrID INTEGER REFERENCES User(ID) ON DELETE CASCADE,
            ID INTEGER PRIMARY KEY AUTOINCREMENT,
            Name VARCHAR(30) UNIQUE)
```

SequenceOrder(*seqID*, *modID*, Order)

```
    CREATE TABLE SequenceOrder(
            seqID INTEGER REFERENCES ModuleSequence(ID)
                    ON DELETE CASCADE,
            modID INTEGER REFERENCES Module(ID) ON DELETE CASCADE,
            modOrder INTEGER NOT NULL,
            PRIMARY KEY (seqID, modOrder))
```