Entities:

User:

- Assumptions: Users can create accounts and save podcasts. Each user has a unique username and password.
- Reason for being an entity: Users need to manage their personal information, such as login details and saved podcasts.
- Cardinality: A user can save many podcasts (0..*), but each podcast is only saved by one user (1..1).

Podcast:

- Assumptions: A podcast is defined by its ID, the associated user, and its name. Each podcast can reference multiple books, people, and companies.
- Reason for being an entity: Podcasts need to reference other entities like books, people, and companies.
- Cardinality: A podcast must belong to exactly one user (1..1). It can reference many books, people, and companies (0..* for all).

Books:

- Assumptions: Books mentioned in podcasts are stored with information such as the name, Wikipedia link, author, average rating, and book description.
- Reason for being an entity: Since books may be referenced by multiple podcasts, it is better to make them an entity rather than an attribute of a podcast.
- Cardinality: Books can be referenced by many podcasts (0..*), but each reference will be associated with only one book.

People:

- Assumptions: People mentioned in podcasts are stored with their name, Wikipedia link, net worth, profession, and a description of who the person is.
- Reason for being an entity: People can be referenced by multiple podcasts so it is better to make them an entity rather than an attribute.
- Cardinality: People can be referenced by many podcasts (0..*), but each reference will be associated with only one person.

Companies:

- Assumptions: Companies mentioned in podcasts are stored with details like name, Wikipedia link, state, revenue, industry, and company description.
- Reason for being an entity: Companies can also be referenced by multiple podcasts so it is better to make them an entity rather than an attribute as well.

- Cardinality: Companies can be referenced by many podcasts (0..*), but each reference will be associated with only one company.

Relationships:

Saved:

- Assumptions: A user can save multiple podcasts, but a podcast is associated with only one user.
- Cardinality: 0..* on the podcast side (a user can save multiple podcasts), 0..1 on the user side (users can save podcasts but do not have to).

BookReference:

- Assumptions: A podcast can reference multiple books. Each book may also be referenced by multiple podcasts.
- Cardinality: Many-to-many relationship between Podcasts and Books. This relationship will use a composite key of PodcastID and BookID to uniquely identify which book is mentioned in which podcast.

PersonReference:

- Assumptions: A podcast can reference multiple people. Each person may also be referenced by multiple podcasts.
- Cardinality: Many-to-many relationship between Podcasts and People. This relationship will use a composite key of PodcastID and PersonID to uniquely identify which person is mentioned in which podcast.

CompanyReference:

- Assumptions: A podcast can reference multiple companies. Each company may also be referenced by multiple podcasts.
- Cardinality: Many-to-many relationship between Podcasts and Companies. This relationship will use a composite key of PodcastID and CompanyID to uniquely identify which company is mentioned in which podcast.

Normalization:

- We used BCNF Normalization. Initially, when we made our first iteration of the UML we had the PodcastsBook & Books (and the corresponding tables for Companies and People) as the same table. The issue with this is PodcastID is a partial dependency as it would not uniquely determine BookID. For example, a podcast could reference multiple books, but the podcast ID wouldn't uniquely identify a particular book. In our case,

PodcastID was not a superkey so the database wasn't normal. To fix this we created the PodcastBook, PodcastCompanies, and PodcastPeople to break up the many-to-many relations. These tables ensure the combination of PodcastID + BookID (or CompanyID, PersonID) forms a key that allows each entity to have its attributes without dependency on PodcastID alone.

Relational Schema:

User(UserID: INT [PK], Username: VARCHAR(255), Password: VARCHAR(255))

Podcast(PodcastID: INT [PK], UserID: INT [FK to User.UserID], PodcastName: VARCHAR(255), Link: VARCHAR(255))

PodcastBooks(PodcastID: INT [FK to Podcast.PodcastID], BookID: INT [FK to Books.BookID])

Books(BookID: INT [PK], BookName: VARCHAR(255), WikipediaLink: VARCHAR(255), Author: VARCHAR(255), AvgRating: DECIMAL(3,2), BookDescription: VARCHAR(255))

PodcastPeople(PodcastID: INT [FK to Podcast.PodcastID], PersonID: INT [FK to People.PersonID])

People(PersonID: INT [PK], PersonName: VARCHAR(255), WikipediaLink: VARCHAR(255), NetWorth: DECIMAL(15,2), Profession: VARCHAR(255), PeopleDescription: VARCHAR(255))

PodcastCompanies(PodcastID: INT [FK to Podcast.PodcastID], CompanyID: INT [FK to Companies.CompanyID])

Companies(CompanyID: INT [PK], CompanyName: VARCHAR(255), WikipediaLink: VARCHAR(255), State: VARCHAR(255), Revenue: DECIMAL(15,2), Industry: VARCHAR(255), CompanyDescription: VARCHAR(255))