Stage 2: Conceptual and Logical Database Design

logical design (relational schema) and description

Entities:

```
Company(
        CompanyID: LONG [PK],
        CompanyName: VARCHAR(50),
        Followers: INT
        CurrentStockPrice: REAL
        CurrentGrowthRate: REAL
        PredictedStockPrice: REAL
        PredictedGrowthRate: REAL
)
```

The company is uniquely identified by company id, and we need company names and followers for showing company's info, and we need to have current and predicted stock price and growth rate to display core features: predicting and checking stock information.

```
StocksByDate(
        CompanyID: LONG [PK],
        Date: DATE [PK]
        StockPrice: REAL,
        GrowthRate: REAL
)
```

The stock price and growth rate are uniquely determined by each company at each date. In other words, at the end of each day, we collect the stock price and growth rate and store them into this schema. Then we can use this in the model to predict later stock prices.

```
Users(
        UserId: LONG [PK],
        AccountName: VARCHAR(50)
        )
```

User is uniquely identified by user id, and it has an account name for displaying user's name or nicknames.

```
LoginInfo(
        UserID: LONG [PK],
        Password: VARCHAR(50),
        PhoneNumber: LONG,
        Email: VARCHAR(50)
)
```

The login info is uniquely identified by user id because it is for a user to login. It has phone numbers or email addresses because it is served as a login identification. The password is for a user to securely certify identity.

```
CompanyEmotionByDate(
        CompanyID: LONG [PK]
        Date: DATE [PK]
        Emotion: LONG [PK]
        Frequency: INT
)
```
A CompanyEmotionByDate is uniquely identified by its company id, date, and emotion. We also need frequency to store the amount of searching for a specific company, date, and emotion.

Relationships:

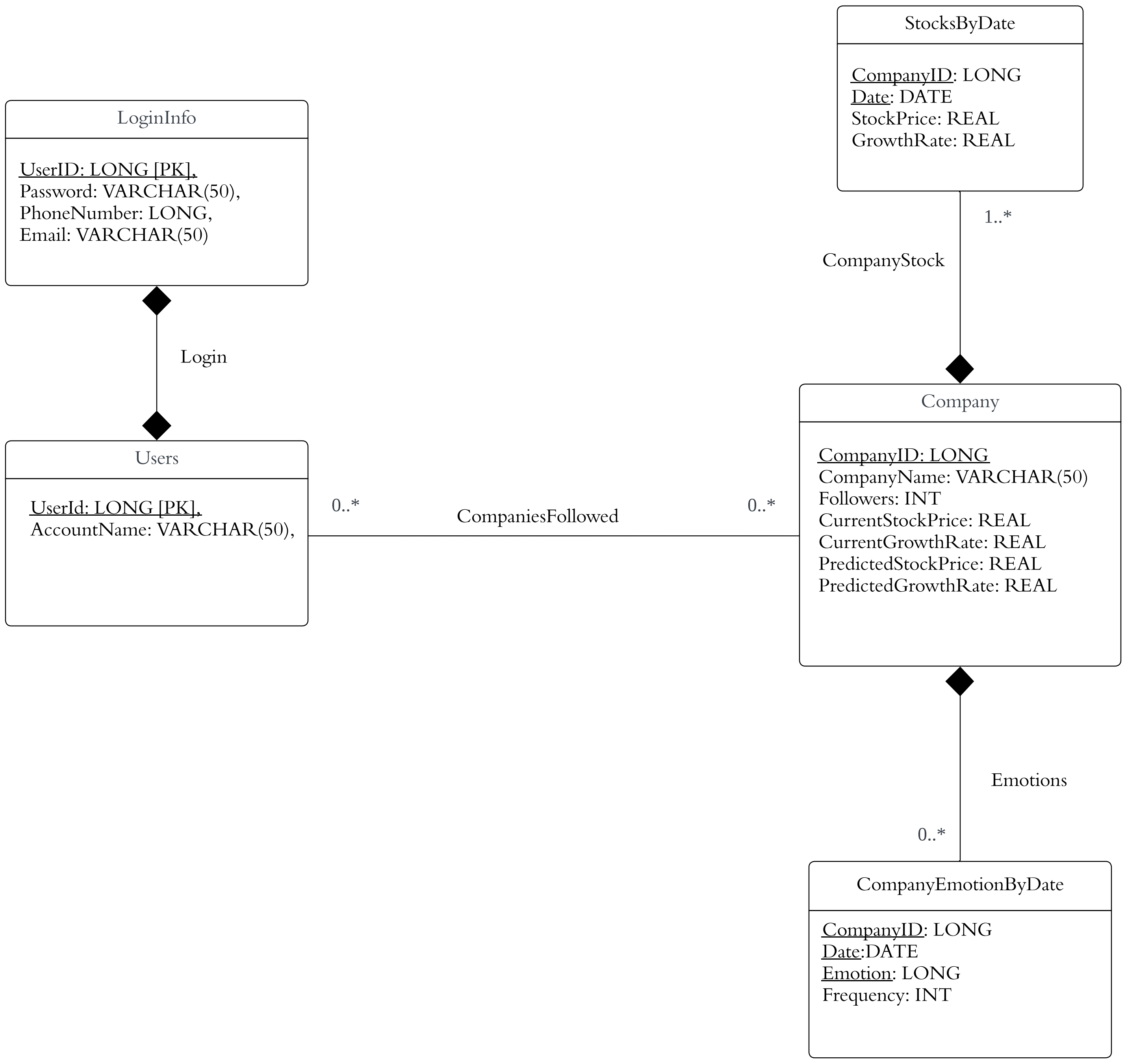
```
CompaniesFollowed (
        UserId: LONG [FK to Users.UserID]
        CompanyID: LONG [FK to Company.CompanyID]
)
```
The CompaniesFollowed relationship demonstrates users following status of any company.
We assume that there can be none to multiple users for each company they followed, and there can be none to multiple companies that are followed by each user.

```
Login(
        UserID: LONG [FK to LoginInfo.UserID]
        UserID: LONG [FK to Users.UserID]
)
```
The Login relationship demonstrates users login status on the app.
We assume that only one user can login to the app, and the app takes one user at each login.

```
CompanyStock (
        CompanyID: LONG [FK to StockByDate.CompanyID]
        Date: DATE [FK to StockByDate.Date]
        CompanyID: LONG [FK to Company.CompanyID]
)
```
The companyStock relationship demonstrates the relationship between companies and their stocks by date.
We assume that each company can have its multiple stocks information by date and that each stocksByDate information only refers to one company

```
Emotions (
        CompanyID: LONG [FK to Company.CompanyID]
        CompanyID: LONG [FK CompanyEmotionsByDate.CompanyID]
        Date: DATE [FK to CompanyEmotionsByDate.Date]
        Emotion: LONG [FK  CompanyEmotionsByDate.Emotion]
)
```

Emotions relationship demonstrates the relationship between the companies and their audience attitudes toward them by date.
We assume that each company has multiple attitudes from the audience by date and that each companyEmotionByDate is referring to only one company.


StocksByDate
CompanyID: LONG
Date: DATE
StockPrice: REAL
GrowthRate: REAL
1..*
CompanyStock
LoginInfo
UserID: LONG [PK],
Password: VARCHAR(50),
PhoneNumber: LONG,
Email: VARCHAR(50)
Login
Users
UserId: LONG [PK],
AccountName: VARCHAR(50),
0..*
CompaniesFollowed
0..*
Company
CompanyID: LONG
CompanyName: VARCHAR(50)
Followers: INT
CurrentStockPrice: REAL
CurrentGrowthRate: REAL
PredictedStockPrice: REAL
PredictedGrowthRate: REAL
Emotions
0..*
CompanyEmotionByDate
CompanyID: LONG
Date:DATE
Emotion: LONG
Frequency: INT