# Re-twitter

## SOFTWARE REQUIREMENTS

### Dependencies:

1. bcrypt: ^3.0.2
2. express: ^4.16.4
3. jsonwebtoken: ^8.3.0
4. mongoose: ^5.3.4

### devDependencies:

1. body-parser: ^1.18.3
2. nodemon: ^1.18.4

## DATABASE

1. MongoDB Atlas Used
2. Mongoose used as MongoDB Client

## MODELS

### Schema of user

(models/user.js)
Each user will have

- User ID assigned by system to every user at the time of signup
- Unique username chosen by him (String)
- Unique Email ID (String)
- List of followers (initially list of followers is empty) (Array of String)

### Schema of tweet

(models/tweets.js)
Each Tweet will have

- Tweet ID generated by system at the time of posting a tweet
- Content posted by user (String)
- Author of the tweet (String)

## BASIC and EXTENDED FUNCTIONALITIES (routes)

### Basic Functionalities implemented:

1. Signup: User can register into the system
2. Login: user can login into the system using the credentials created

## Extended Functionalities implemented

1. Tests: Tests for Basic Functionalities (Login and signup)
2. Read tweets: User can read all the tweets
3. Create tweet: User can post a tweet
4. Delete a tweet: User can delete a tweet
5. Follow another user: User can follow another user
6. Unfollow another user: user can unfollow another user
7. View Self-Profile: User can view their own information

**For extended functionalities user have to login first**

## PREREQUISITES FOR FUNCTIONALITIES

### Signup:

(localhost:8080/user/signup) - POST

- Username (Has to be unique)
- Email (Has to unregistered email and should follow regex pattern for email)
- Pasword

### Login:

(localhost:8080/user/login) - POST

- Username (created at the time of signup)
- Password

### Read Tweets:

(localhost:8080/tweets) - GET

- Login Token

### Create tweet:

(localhost:8080/tweets) - POST

- Login token
- Data which follows tweet schema (content, author)

## Delete tweet:

(localhost:8080/tweets/:tweetId) - DELETE

- Login token
- Tweet ID generated by system at the time of creating a tweet

## Follow another user:

(localhost:8080/user/:userId) - PATCH

- Login token
- User ID of the user to be followed

## Unfollow another user:

(localhost:8080/user/:userId) - DELETE

- Login token
- User ID of the user to be unfollowed

## View Self-Profile:

(localhost:8080/user/me)

- Login Token

**NOTE:** Login token is always passed as Authorization header as "Bearer <login_token>"

## TESTING

1. Api is tested using Postman Chrome Extension
2. Sample tests for Login are written and can be run on postman.
3. For testing raw input is given in Json format
4. For Login- JWT token needs to be added in Header - Authorization as "Bearer <login_token>"