## My motivation

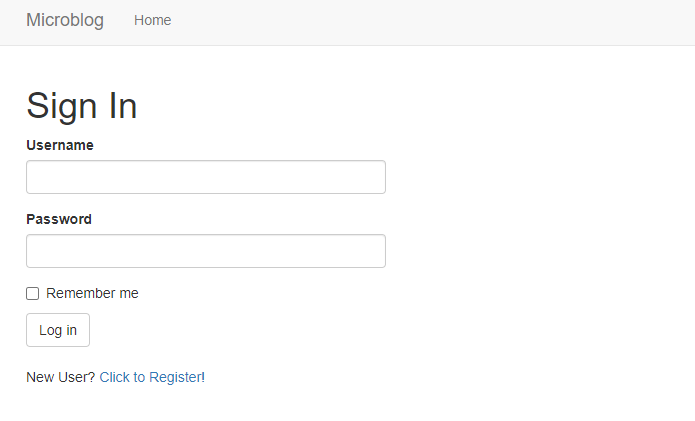
I wanted to build a personal project which has the following elements:

* Use web frameworks like Flask or Django – these can be used to write any web application or REST APIs
* Has backend code written in Python – my language preference
* Implements the web UI using the front-end trifecta – HTML, CSS and JavaScript – so that I have exposure how to contribute to any front-end project

## Why did I build this project?

1. *Web Application* - I had written backend code for REST APIs earlier but never owned a full-stack web application
2. *Flask was a better fit* – Flask is minimalistic and allows more flexibility than Django. It has had support of external libraries, routing to HTTP webpages is easy and has a large community presence. In terms of performance, any application in Flask is as fast as an application in Django.
3. *Miguel Grinberg’s tutorial­* – I came across a very extensive tutorial by Miguel Grinberg and decided to build it. Credits to Miguel Grinberg added below
4. *Microblog* – the idea of the microblog fulfilled a lot my requirements. I got to implement “*twitter like”* application where users follow each other and see their posts, a process to authenticate users and create new accounts, a page to customize a user’s profile etc.

## Features

* *Sign In* page for users to specify credentials and login. There’s an option for new users to create a new account. Users won’t be allowed to view any other page until they are authenticated.  
  
* *Home* page displays list of all posts created by you and by the people who you follow. It also provides you a text box to submit your post.
* *Explore users* page displays list of all users who are registered in the system. Any user can be clicked to be navigated to their personal *user* page
* *User* page displays the bio, photo/avatar of the user. It displays the number of followers/following, and lists the posts created by the user. The button to follow or unfollow the user can be seen here.
  + The *User* page of the logged user has an additional option to edit the bio

## Imported Extensions/Packages

*FRONT END*

* **Templates** were used to render the HTML pages
* **Jinja2 template engine**help gather data from backend and display it on HTML page
* **Flask-Bootstrap** extension extends support to **Bootstrap** **CSS framework** which provides a consistent visual template for all webpages
  + **Popover** component provides small overlay of content, for housing secondary information
* **Flask-Moment** extension extends support to **Moment.js** Javascript library which helps display time/date in local time-zones

*BACKEND*

* **Flask-WTF** extension provides means to use any web form (writing post form, password form etc.)
* **Flask-SQLALchemy** extension provides a way to use SQLAlchemy (Object Relational Manager) to manage tables and SQL
* **Flask-login** extension manages user’s logged-in state i.e. so that for example users can log in to the application and then navigate to different pages while the application "remembers" that the user is logged in
* **Flask-Babel** extension provides support to translate text from multiple languages
* **Werkzeug** package generates hashed passwords to be stored in the database. It also checks if entered password matches the stored hashed password without displaying the password anywhere in clear text
* Default profile avatar for a user is sourced from **https://www.gravatar.com**

*MAINTENANCE*

* **unittest** for unit testing

## Code Structure

* /app/routes: view functions for all the web pages and redirects
* /app/models: class definitions to design and manage database tables
* /app/forms: class definitions for all forms in the application
* /app/errors: error handling

## Credits

1. Miguel Grinberg’s Flask Mega Tutorial <https://www.linkedin.com/in/miguelgrinberg/>

## Suggested improvements

1. Implement NoSQL as it's more efficient for horizontal scaling if we must support large number of users