NAME: IHEMEBIRI CHINASA MARTINS

STUDENT NO: 2981843

MODULE: MSCC-CPA

ASSIGNMENT 03

BRACKET 8

addpost.py:

- **GET method:** This method renders a simple form to the logged in user, the form consists of a text field for accepting the caption of a post and an file upload for accepting the image file from the user, the image file is filtered to only accept png and jpeg in the html template of this class. To upload an image this class uses blobstore.create_upload_url('/upload') which routes to the uploadhandler class

edituser.py:

- **GET method:** The get method here simply renders a html page that includes a text field that accepts a text from the user and uses this to update the user's username for the app. A logged in user can also check his profile details on this page as it renders the user's current username and email

- **POST method:** When the update button is clicked this method simply accepts the text input from the user, a check is performed to see if there is indeed input from the user, if there is, this method gets the logged in current user and updates their username field in the MyUser model

followlist.py:

- **GET method:** This page is only accessed from a user's profile page after the followers or following count link is clicked. It displays the number of followers or following that the user has and displays the followers or followings as a list on the page. To do this the get method gets the key of the user being viewed from the URL. The key is also concatenated with "-following" or "-followers" depending on which is trying to be viewed, the get method splits the key from the text using .split('-') and depending on which was chosen from the profile page it either renders the users following list or followers list in the html

Main.py:

- **GET method:** This is the first page a user visits after they are logged in, if a user isn't logged in this page redirects them to the guest page and requests that they log in, if a new user logs in this method creates a new object for that user in the MyUser model.
  After a user is logged in, this method iterates through the users following list and appends all the posts of its users into another list then iterates through the logged in users posts as well and appends it to the same list. Then using python's lambda function the lists with all the posts are then sorted using their timestamp field in reverse chronological order and put into a new list called posts
  Using list slicing the top 50 elements in this list are then passed through template values to the html i.e. posts[:50] and renders this on the timeline of a logged in user

- **POST method:** This method is triggered when the comment button is clicked when a user wants to add a comment to a post, the method gets the ID of a post through the URL passed in the form, the users comment string passed through the comment text field, the current date time using pythons datetime function and also gets the ID of the logged in user which represents the user that's trying to add a new comment. This method using these values to create a new comment object and the appends this to the structured property comments of the Post model, after the new comment is added the page refreshes and displays the new comments added to the post

profilepage.py:

- **GET method:** This method is used to display the profile of a logged in user or of another user and does this in various steps, first it gets the current user that is logged into the app and then also checks the URL for a key from another page that represents another user that isn't logged but is trying to be viewed by the logged in user. If the logged in user is the same as the user key in the URL then this page displays the profile of the logged in user, however if the logged in user's key differs from the user's key in the URL this would mean that the logged in user is attempting to view the profile of another user there for the profile page of that other user will be displayed along with a follow or unfollow button and using python's lambda and the time stamp of each post object the profile page will display posts in reverse chronological order. This method also displays the number of followers and following the user has as links which when clicked goes to a new page and displays a list

- **POST method:** The post method in this class is responsible for following and unfollowing of users. When the follow button is clicked the post method appends that user to the following list of the logged in user and appends the logged in user to that users followers list, vice versa for when the unfollow button is clicked. After this is done the page is refreshed

search.py:

- **GET method:** This method simply displays a text field to the user along with a search button, the page is responsible for searching the MyUser model for username of users of the app

- **POST method:** After the search button is clicked the user input is taken and this post method is triggered, if the user did not put any text then all the users of the app is displayed, if there is a text for searching, then this method first queries the MyUser model and gets all users, it then uses the text to check against each username of MyUser objects, it performs this check using pythons IN comparison statement, this enables partial matching of usernames just like the app works in real life, for example, if a user searches for “er”
  Then any user in the MyUser model whose username contains that string will be returned e.g “Peter”, “Casper” etc and displayed to the searcher

viewcomments.py:

- **GET method:** This method renders a single post that has more than five comments, on the previous page any post with more than five comments has a view more button which when clicked navigates the user to this page and displays the post along with all the comments that it has

uploadhandler.py:

- **POST method:** This method is responsible for creating a new post for the logged in user whenever triggered. The method gets the current user and then creates a new post object. It gets the caption, the current date and time, the logged in user and the image file that was uploaded and uses this to create a new Post object for the logged in user. It handles the image file upload into the blobstore and stores the file uploads blobkey as a key property in the post object for use later when the image is to be displayed

BRACKET 9

Data structures used:

The only data structure used in this application is python's lists

Lists are used in this application because they are iteratable and are mostly sent from python into jija2 html template where they are iterated over and display various things

- In **main.py**, a list called timeline is used to hold post objects of the logged in user as well as posts of the user he/she follows, this list is then sorted using the timestamp field for each post in reverse chronological order and put into a new list called posts which is then sliced[:50] to only contain the top 50 posts and is then sent to the jinja template to be displayed on the homepage of the logged in user
- In **profilepage.py,** a list is used to hold the posts the user that is being viewed, this list is sorted in reverse chronological order and sent to the html template to be rendered
- In **search.py**, a list called search_list[] holds all the users that matches completely or partially the search query of a logged in user and this list is sent to the html for render

Models:

Comments.py:

- This model contains all the attributes associated with a comment
- This model has two datatypes i.e String property and key property
- The attributes contained in this model are text(String) this text contains the actual comment made by a user , comment_time(String) this holds the time and date when the comment was made, comment_user(Key) this holds the key of the user that made the comment

MyUser.py:

- This model contains all the attributes associated with a user and it's used to store a user object in the database
- This model has two datatypes i.e String property and key property
- The attributes contained in this model are the name(string) this represents the user name for that user, user_posts(key) this field holds the key of the posts created by the user, following_list(Key) this holds the list of the users that are being followed by this user, followers_list(key) this holds the key of all users that are following that user
- The following_list, followers_list and user_posts attributes are set to repeated true so they can store multiple keys of multiple users and posts

Post.py:

- This model contains all the attributes associated with a single post
- This model has four datatypes i.e String property, key property, blobkey property and structured property
- The attributes contained in this model are user(Key) this holds the key of the user that created the post, blob(blob key) this holds the blob key of image file that is uploaded and associated with a particular post , caption(String) this holds the string caption made by the creator of the post, time(String) this holds the time a post was made and used to sort all posts in reverse chronological order , comments(Structured property) this holds a comment object that is associated with a single post, a comment has various attributes that is why it had to be put into a new model
- The comment structured property is set to repeated true because a post can have numerous comments

BRACKET 10

- Bootstrap img-fluid class is used to make all the displayed images responsive for smaller screens, the images are all set to a certain size irrespective of their actual size to give a more uniform display of all images

- On the main page Username's are all set as hyperlinks so that when clicked the user can quickly navigate to that user's profile page and view all their posts

- Horizontal lines are used to signify the start and end of a single post along with its comments this would help the customer visually to separate one post from the other

- A comment icon is used to toggle a text field for adding comments, this helps make the page less bulky by only showing the text field when the icon is clicked on other wise it remains hidden from the user

- A like icon is set to each post though it does not do anything its purpose is to give the app a more realistic feel of the actual Instagram app

- The date and time of each post is shown with the post, so a user knows when it was made and how recent the post is