**Breakdown of phases**

Phase 1: Web development

The section of this phase comprises of creating a webpage, which people can log on to, and view their posts, and create new posts.

Phase 2: Twitter search

This phase contains the twitter searching capabilities, which are helped by the tweepy API for python. This allows the user to search for tweets of specific content, then return them in a list.

Phase 3: Google sentiment analysis

This phase comprises of using each tweet to get an average sentiment for that tweet, then use all the sentiments to have an average sentiment, which is visible to the user. The way the sentiment is calculated is through google cloud language API.

**Timeline of phases**

September 2020 – October 2020: Whole of Phase 1.

October 2020 – December 2020: Whole of Phase 2.

January 2021 – March 2021: Integrating Phase 1 and Phase 2

March 2021 - : Whole of Phase 3, with integration

**Documentation**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parent Class | Function definition | Function Description | Parameters | Returns |
| Flaskr.auth | @bp.route('/register', methods=('GET', 'POST')) def register() -> Response | This function registers a new user based on information provided by register.html |  | Returns:  the template for register.html |
| “ | @bp.route('/login', methods=('GET', 'POST')) def login() -> Response | This function checks to see if information provided by login.html is the same as a user, then redirects to blog.html |  | Returns:  the template for login.html |
| “ | @bp.before\_app\_request def load\_logged\_in\_user() -> Optional[Any] | This function loads a user without the user having to log on if the user has previously logged on on this computer |  |  |
| “ | @bp.route('/logout') def logout() -> Response | This function logs the user out for the computer the user was using, then redirects them to index.htm |  | Returns:  the redirect URL for index.html |
| “ | def login\_required(view: Any) -> (kwargs: Dict[str, Any]) -> Response | This function determines whether the user has previously logged in on this device. :param view: :return: wrapped\_view | Params:  view – | Returns:  wrapped\_view |
|  |  |  |  |  |
| Model.py | def work(search\_term: Any) -> float | This Function searches twitter for tweets | Params:  search\_term – The search term passed in to search | Returns:  the average sentiment |
| “ | def workSentiment(data: {\_\_iter\_\_},  totalint: Any) -> float | This function divides the tweets into individual strings, which then the sentiment of is calculated | Params:  data – the tweets data gathered  totalint – the total number of tweets | Returns:  the mean sentiment |
|  | def sample\_analyze\_sentiment(text\_content: Any) -> Any | This function Analyzes Sentiment in a String | Params:  text\_content – The string text content to analyze | Returns:  sentiment gathered from the string passed in |
|  |  |  |  |  |
| Db.py | def get\_db() -> Connection | This function returns the database, and if no database is found, creates its own database |  | Returns:  the database |
|  | def close\_db(e: Any = None) -> None | This function closes the database | Params:  e – None |  |
|  | def init\_db() -> None | Initializes the database |  |  |
|  | @click.command('init-db') @with\_appcontext def init\_db\_command() -> Optional[Any] | Clear the existing data and create new tables. |  |  |
|  | def init\_app(app: {teardown\_appcontext, cli}) -> None | This function closes the database, then puts init\_db\_command() on the command list | Params:  app – current app being run |  |
|  |  |  |  |  |
| \_\_init\_\_.py | def create\_app(test\_config: Any = None) -> Flask | This function creates the app | Params:  test\_config – configuration of the launcher | Returns:  the app created |
|  | def newsearch(search\_term: Any) -> None | This function refers to the work function in model.py | Params:  search\_term – the search term to pass through |  |
|  |  |  |  |  |
| Flaskr/blog.py | @bp.route('/') def index() -> Any | This function finds all posts, and renders them on index.html |  | Returns:  template for index.html |
|  | @bp.route('/create', methods=('GET', 'POST')) @login\_required def create() -> Response | This function oversees the creation of a blog post, using create.html |  | Returns:  template for create.html |
|  | def get\_post(id: Any,  check\_author: bool = True) -> Optional[Any] | This function finds a post based on the post id provided, and checks the author id against the post id | Params:  id – id of the post to find  check\_author – author of the post | Returns:  the post |
|  | @bp.route('/<int:id>/update', methods=('GET', 'POST')) @login\_required def update(id: Any) -> Response | Updates an existing post, using update.html | Params:  id – the id of the post to be updated | Returns:  the template for the updated post |
|  | @bp.route('/<int:id>/delete', methods=('POST',)) @login\_required def delete(id: Any) -> Response | deletes a post based on the post id provided | Params:  id – the id of the post to be deleted | Returns:  a redirect to /index |
|  |  |  |  |  |

**Testing**

* Finding a tweet that does not exist does not break the program, but returns 0 if no tweets are found
* Finding a tweet over max character count does not break the program, as no tweets are returned, therefore the average sentiment will be 0
* It is impossible to get to a create blog webpage for a user who has not signed in, as they will be automatically redirected to the login page.