**Design Document**

# Introduction

This project is a dynamic web app based on a memo/post system. I decided to do this because I had previously worked on a group project on a basic dynamic web app and felt I could make something somewhat better or useful. The App itself is available online at <http://g00325746.pythonanywhere.com/> .

The function of the app is to allow users to post quick notices for others to view it is ordered by publish time . Implementation-wise it could be setup for separate companies or groups to only allow them to view and add to the notices.

# Layout

When the user opens the page they will be presented with the home page on this page there will be a google sign in button top nav and a side nav option to the other pages as-well as a view at the current posts they will have to come back to the home page to see if something was posted while they were at the home. Under the new post section they will be presented with a form to make a new post and finally the about page just gives some information.

# System Requirements

To access ,browsers it work on include Google Chrome, Microsoft Edge and Firefox. With nothing extra needed. The site used for hosting provides python functionality and from their console I was able to install all the necessary frameworks and other imports needed. The app itself is not

To run locally you will need to download the project folder from <https://github.com/MichaelGeraghty/3rd-Year-Project.git>

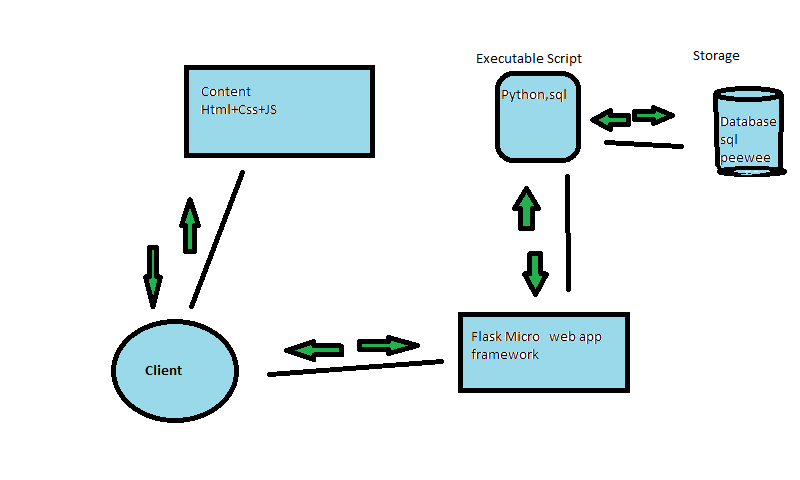
Have Python 3.6 Installed then from the console in the folder location install flask and the few other imports its mentioned in the git repository but things like “pip install Flask”, “pip install peewee” etc the console will notify you of any smaller imports if they were not picked up.

# Technology Used and Why

Seeing as it is a dynamic web app The main technologies used here were python and the flask framework specifically. I used these because I had some prior experience with them. I also find python to be quite straightforward at times as-well the ease at which it is to make a flask web app. I used peewee for the database peewee is a small ORM(Object relational mapping) tool it is built with mysql and sqlite in mind and it works really well with a flask app. Also the size of the database needed is not extremely large and this is more than sufficient. It also is all programmed into the app and models files , its structured simply using a class like structure. The database files can then be edited added to removed in any program that can interact with them eg. MySql .

I then used pythonanywhere.com for hosting because github and others would only host static pages . Also pythonanywhere allowed me to install anything extra needed for the site through their console system.

# Architecture of the Solution



This diagram is in reference to a typical web app architecture used for blogs. Content is fed to the user as or if needed the the python scripts are being fed as called and database is retrieving data and taking data.

# Limitations

At the moment there is a google sign in but have not implemented the auto complete of filling in the users name when they post they have to enter a name themselves. There is no deletion of a post currently also but edits can be made by an admin with access to the database.

# Known Bugs

If run locally the google sign in returns an error because it is not a fan of using localhost but works perfectly fine online.

# Features

This is a multi-user system . Multiple users may actively post to the site and be able to read other users posts in the same session. Users can navigate using buttons on top nav bar which is easier for mouse users or if they have there hand placed there. If users are viewing this from a handheld device there is a button in the top left for navigation also. There is one data entry screen on the site, google provide a data entry screen for a google sign in. The purpose of the database currently is to store and retrieve the posts it contains an id which is a primary key field, date which is a date time field, title ,text and content which are all text fields.

# Recommendations for Future Development

Memo pages will be setup under different groups who have their own unique id’s and passwords to join groups where they can leave posts that they can only see.

When a user signs in their name associated with their email will be used. Currently that info is accessible in console but I am yet to implement it fully.

The option to delete a post through the site itself.

# Conclusions

In conclusion at the start of this project I set out to improve my ability with developing dynamic web apps, to implement an external login like google, to use a database and to have it hosted online with full functionality. Knowing that if I had spent a little more time earlier on it I could have implemented the current recommendations for future development.