1.Project Management Plan:

A. The Gantt chart attached in the file shows how the tasks have been broken down and which resources are allocated to certain tasks.

B. Our group had decided that we would not have a team lead and that the tasks will be divided among ourselves as displayed in the Gantt chart. The Group would have a compulsory meeting once a week on Mondays 3:30 – 5:30, in which all the members would meet and discuss the progress made within the week. If any of the group members felt it was necessary to meet to clarify something the members would set a time to meet each other and resolve the matter. The group members created a Slack and a GitHub accounts to share.

C. The team work distribution was divided in this manner:

Afrah Abdulmajid	→	Construct the Gantt chart
		Documentation for state of the prototype implementation
Afrah Abdulmajid and Jay Patel -	→	1. Responsible for Requirement Gathering
		2. High level design and low-level design
		3. Filling the DocStrings for the implementation.
Zayed Imam	→	Developing the source code and implementing the Prototype
Ahmed M.M Ellisy	→	Responsible for documenting the test suites
		Creating test suites and running the tests
Tirth Patel	→	Responsible for packing the distribution so it can Installed as a package

D. Ranking
Afrah Abdulmajid : 19
Jay D Patel: 17.25
Ahmed Ellisy: 15
Tirth Patel: 11.5
Zayed Imam: 18.5

The rankings were done using based on the following:

I. individual effort
II. value of contribution to project
III. quality of individual work-product
IV. teamwork

The points given to each student for each ranking aspects of the project is out of 5 and so the total is out of 20. The average of the rankings placed by the group members becomes the rank of the individual.

2. Introduction

Project Description

The client is the instructor for COMP2005 course at the Computer Science Department, Memorial University of Newfoundland
The project description presented by the client is as follows:
The COMP2005 course involves use of tools and environments which students have to "pick-up" along the way, with limited instruction. This presents a challenge for students at different levels of experience with software development. I would like them to have a collaborative tool for exchanging information on materials, tutorials, tools, organization and other helpful resources or links. It should be along the lines of other software development forums such as Stack Overflow.
The project team is a group of up to 5 students registered in COMP2005 for the Winter 2018 semester.

Methodology

Accepted techniques for requirements gathering include:

- Client interview
- User group dicussion
- User stories
- UML use case analysis

In their first assignment, each student is tasked with applying a selection of techniques and producing three possible requirements for this project. At least one of the three

requirements produced is to be original and added to the requirements list in this document.

Functional Requirements

1. User Authentication:

1.1 Create a User

desc: Users can create an account by signing up with a unique username, email address and Password and that would be used to sign-up **[M]**

valid: Derived from Use Case

verif: Run the application and click “Signup” button, a Signup form should appear

1.2 User Sign-in

desc: User can sign into the application using the unique username and password created after the user has created an account **[M]**

valid: Derived from Use case

verif: Run the application and click “Login” button, a Login form should appear

2. User Topics:

2.1 Create a topic thread

desc: Authorized User can create a topic to be discussed among other Users in the System **[M]**

valid: Derived from Use Case

verif: Login to System and click on “topic” .

2.2 Share Important Information with regards to topics created

desc: User can attach files of any form with regards to the topics that has been created **[C]**

valid: Derived Use Case

verif: Log into System, select a topic you would like to add to, and select “Post” and post an attachment file.

2.3 Users posting comments

desc: User can post comments on to topic threads in response to the topic created Or in response to other user’s comments. **[S]**

valid: Derived from Use Case

verif: log into the system, then select the Topic and select “Post” and post the comment.

2.4 Users can delete a topic thread

desc: Authenticated Users can delete a topic that the user created by the same user that is logged in. **[M]**

valid: derived from the Use case.

verif: Login to the application and select the topic you would like to delete and Click “Delete”

3. User Subscription:

3.1 Subscribe to a thread

desc: User can subscribe to a topic thread to be up to date with information with regards to the thread. **[M]**

valid: Client Interview/ Use case

verif: Login to System and select a topic thread to subscribe to by clicking “Subscribe”. .

3.2 Unsubscribe to a thread

Desc: User can choose to unsubscribe to a topic thread. When the user unsubscribes the user will not receive any notifications with regards to the the topic thread created.

valid: Derived from the use case.

verif: Login to the System and select the topic user is subscribed to, and click “Unsubscribe”.

4. User Notification:

4.1 Turn On/off User Notification

desc: User can choose whether the User would like to receive notifications or not. **[C]**

valid: Client Interview/Use case

verif: Log into system and choose to turn on notification when creating account.

4.2 Notification for posts on topic thread

desc: Users can receive notification through email when there is a new post on threads User is subscribed to **[S]**

valid: Client Interview/ Use case

verif: After post has been posted, Log in to email and check email for notification.

5. User Search:

5.1 Search engine for Users

desc: Users can search other users that have users in the system**[W]**

valid: Use case/ Client Interview

verif: Login to system and search for user in the system

6. Group Discussion:

6.1 Create group discussion

desc: Users can create group discussions with specific users to openly discuss amongst each other.**[M]**

valid: Derived from the Use case

verif: log into system and create group discussion by clicking “Groups”

6.2 Set the privacy for group discussion

desc: Users can decide if they would like to keep the discussion open to the public or to keep it private amongst the people added into the group. **[M]**

valid: Derived from Use case

verif: Create a group discussion and set the privacy of the group to “Private” or “Public”.

7.0 User Archives:

7.1 Users can save discussions that have been created.

desc: Users would be able to save a topic thread after a topic thread has been Deleted and still view it. **[M]**

valid: Derived from the Use case

verif: Login into system and Select a topic and click “Archive” button to Archive the discussion

8.0 User Posts:

8.1 Users can create posts that can be viewed by the all the users

desc: All authenticated users should be able to create a post that can be viewed by all the users who are logged in and the user should be able to view all the posts posted by other users.

valid: Derived from the Use case

verif: Log in to the system and click on the button “Post” to create a post

Usability:

9.0 Navigation

9.1 Creating an account using the System

desc: The Users are going to be able to click a button to that is going to guide the User to create a new account **[M]**

valid: Derived from the deliverables of the project

verif: Click on the button *Create account*

9.2 Text fields to input the right information

desc: The system is going to to have text fields that require email addresses when creating Users **[M]**

valid: Derived from the deliverables of the project.

verif: Go to create account and write the email address

10. User Activity:

10.1 User Dashoard:

desc: Users will be able to see their latest activity on the application upon Login, and all the topics the user has created if there are any.

valid: Derived from the Use Case

verif: Login to the application, and the DashBoard should appear

Design Constraints:

11. Flask frame-work is used
    desc: Use of the flask web development framework imposes design restrictions regarding how web services are structured and how mobile devices are supported. Much of the project architecture is pre-determined by the flask Framework.
    valid: Imposed by the context of comp2005 project
    verif: Examine the implementation code

12. Flask-SQLAlchemy is used
    desc: Use of the SQL-Alchemy as a the database that stores all the instances of the Application.
    valid: Discussions in the group meeting
    verfi: Examine the implementation code

User Story

User can access the web app created by signing into the app, the user can sign in directly if an account is already created and they have the login credentials

Title: User Authentication
Primary Actor: User
Stakeholders and Interests:
User - wants access to the web app.

Precondition: User must sign up for an account.
Trigger: User is notified when successfully logged in.
Main success scenario:

1. SuD displays an option whether to login or signup.
2. Existing user can login directly into the account.
3. New user signs up for an account to obtain login credentials
4. SuD enters the new user data into a database.

5. New/existing user logs in the account created after successfully obtaining the credentials.
6. SuD authenticates the information provided by the user upon login.
7. SuD displays the main page upon successful login.

Extensions:

3a. New user signs up for an account to obtain login credentials.

3a1. SuD will display the form for signup.

3a2. User will enter the requested details in the form.

3a3. SuD will take those details and transfer them to a database.

3a4. SuD will display the login page.

User Story:

The Client would like to have a platform that allows communication between students and professors,with the platform created the user would be able to create topics and students and professors will be able to actively participate in the thread that has been created for the specific topic as long as they have an account created on the platform. With in a topic Users can attach files connected to the topic thread created, comment on posts with the topic thread and delete a topic thread once only by the creator of the thread.

Title: Topic Threads
Primary Actor: User
Stakeholders and Interests:

User - Authenticated User would like to create a topic thread.

Precondition: User has to have login credentials, and must be logged in.
Main success scenario:

1. SuD provides the option to post a topic thread.
2. User chooses a topic or a question to pose to the other users.
3. User inputs the title of the topic thread to be created.
4. SuD gives special privileges to the User that creates the topic thread
5. SuD creates a link that will connects the topic thread to the topic discussion page.

6. SuD will allow users to post comments, post attachments, and share links.
7. SuD displays the topic thread created as link for the other users to join.

Extensions:

4a. SuD gives special privileges to the User that creates the topic thread
4a1. The Moderator of the topic thread is assigned.
4a2. Topic Moderator will be able to delete irrelevant posts.
4a3. Topic Moderator can invites user to join the thread.
4a4. Topic Moderator can delete the topic.

Explanation:

This Functional Requirements is there to help the users communicate with other users who share a common interest or users who would like to pose a question that needs to be clarified or resolved by other users within the application. Once a topic has been created it will be visible to anyone that has an account within the web application and so all users can share ideas or give suggestions.

It is important to have an efficient topic thread that does not slow down when there are many users attempting to post and share inside the thread, we would like to have a system that does not lag or crash with the incoming traffic.

User Story:

In the platform that was created for students and professors to communicate, the users of the application can create a post or a topic that can be viewed by all the users. Once the post/topic has been created, the user that created the post/topic is the only user that can make changes to the post and delete the post if need be.

Title: Editing of Postings and topics
Primary Actor: User
Stakeholders and interests:

User: User would like to make changes to the post/topic created, or delete it.

Preconditions: The User that is making changes to the post has to be the user that created the post.

Main Success Scenario:

1. User can choose the post he/she would like to edit/delete.
2. SuD provides the user with options for the action the User would like to perform.
3. User performs the changes to the post.

4. SuD displays the changes done to the profile.

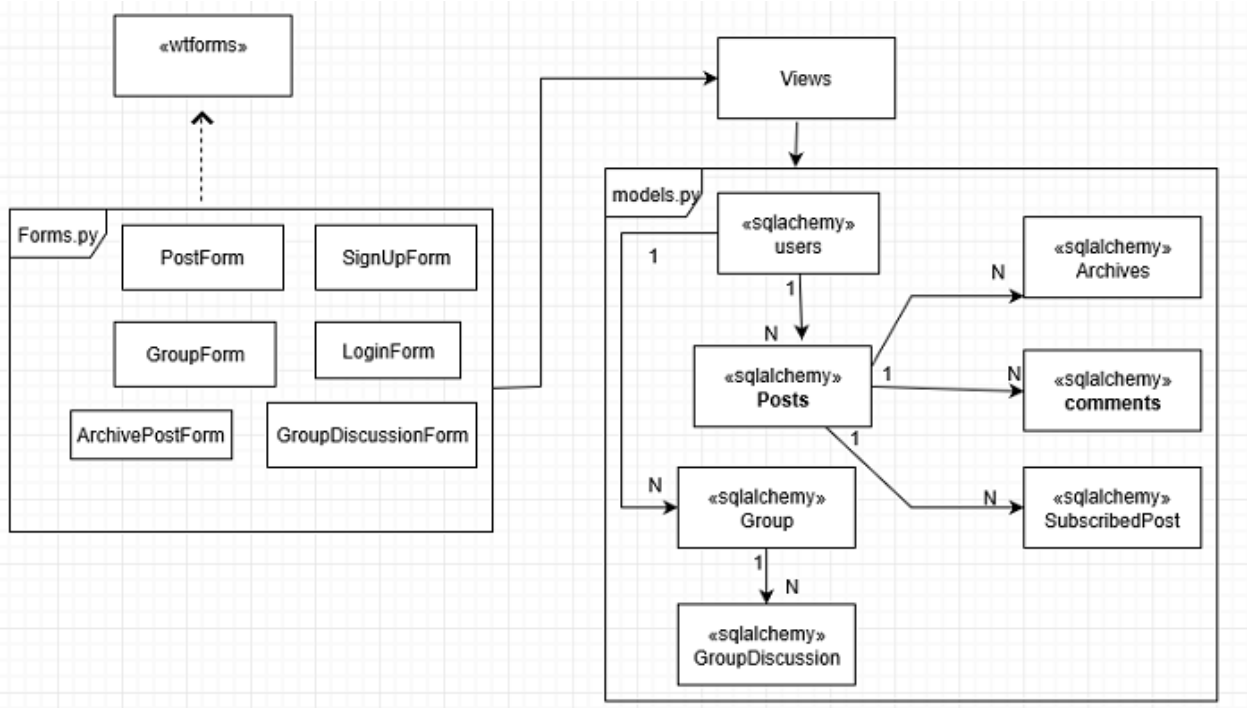
Extensions:

3a1. User chooses to edit the post that he/she has created.
3a2. SuD displays the post that is being edited and allows the user to make changes to the post.
3a3. SuD updates the post.

3b1. User chooses to delete the post that he/she has created.
3b2. SuD deletes the post from the list of postings that is viewed.

Explanation:
The editing and deleting of post allows users to be able to change the minds about the contents their posts/topic or even make changes to a post/topic that has been posted by the user. For example, when a post/topic that has been posted is grammatically incorrect, user can change to post/topic to fix it, instead of creating a whole new post/topic. As well as when a post/topic that has been created and is no longer relevant the user can just delete it.

High Level Design document:



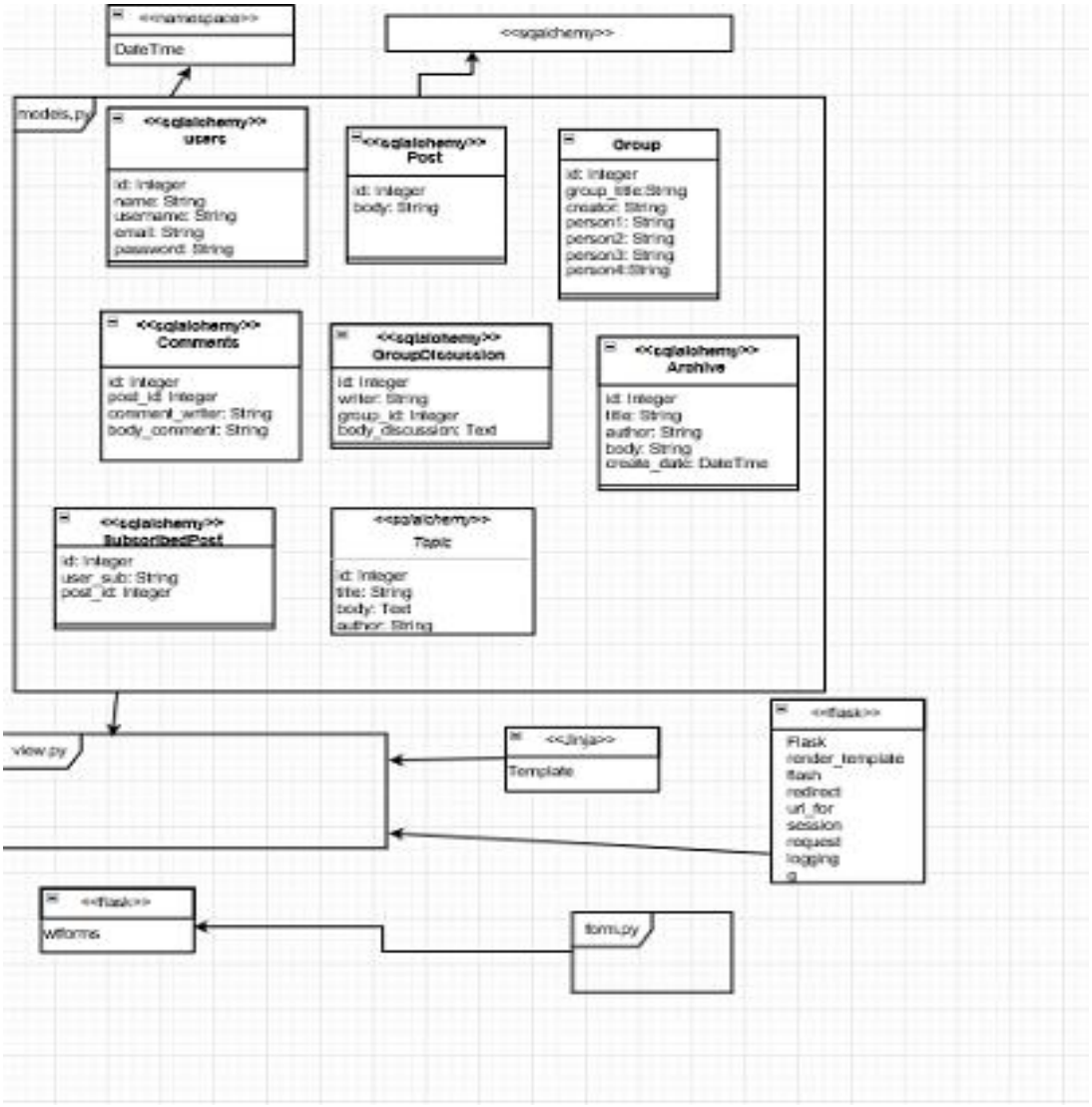
Notes:

The Architecture we have implemented follows the MVC Pattern , where we have a Model, which represents the objects that has the data of the objects created, given by the models.py module.

The view represents the visualization of the data within the model, given by forms.py module in the project.

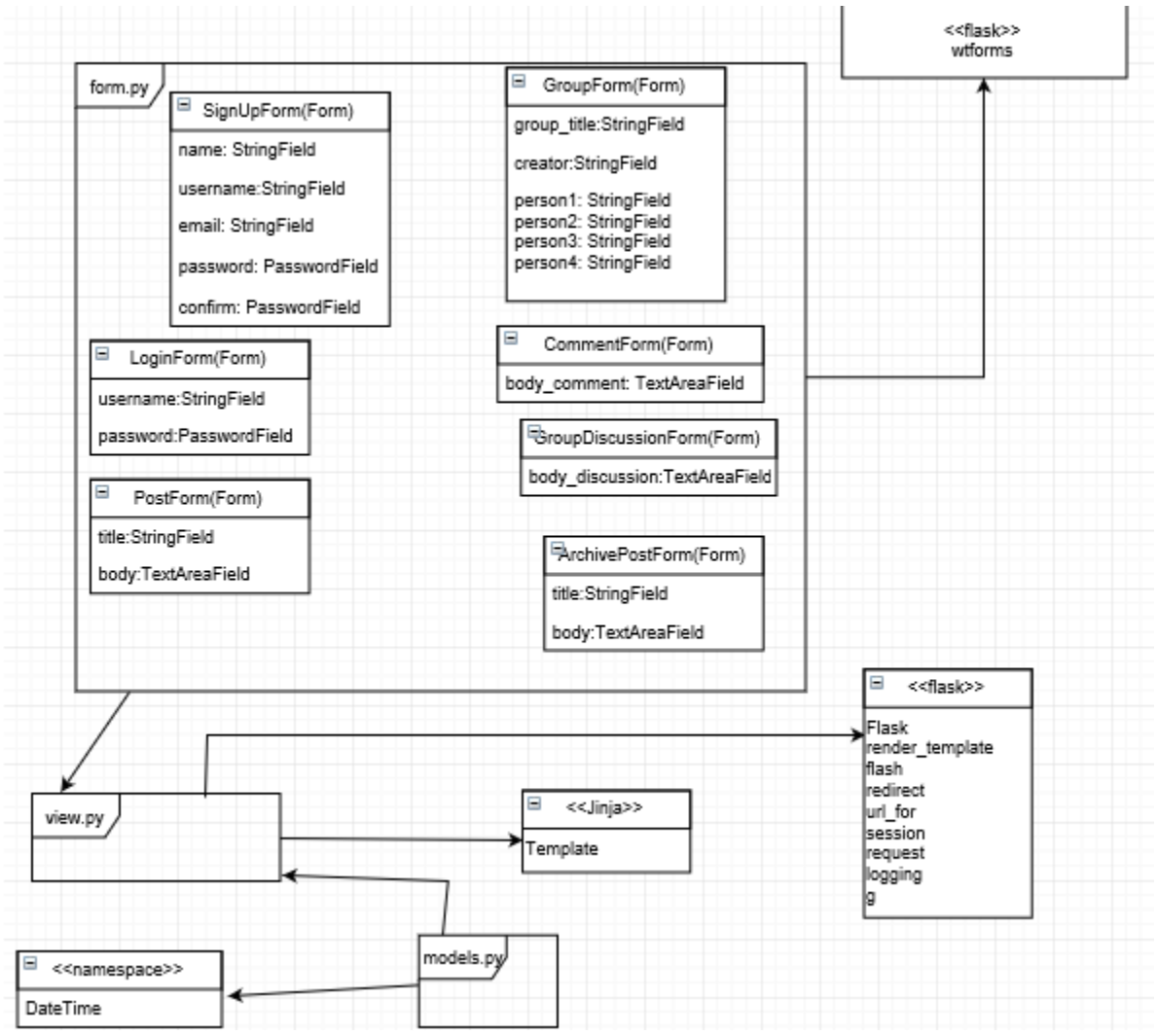
The controller is given by the view.py file, which acts on the model and view. It controls the data that flows into the model's objects and updates the view depending on the data that is being

Low level design for models.py



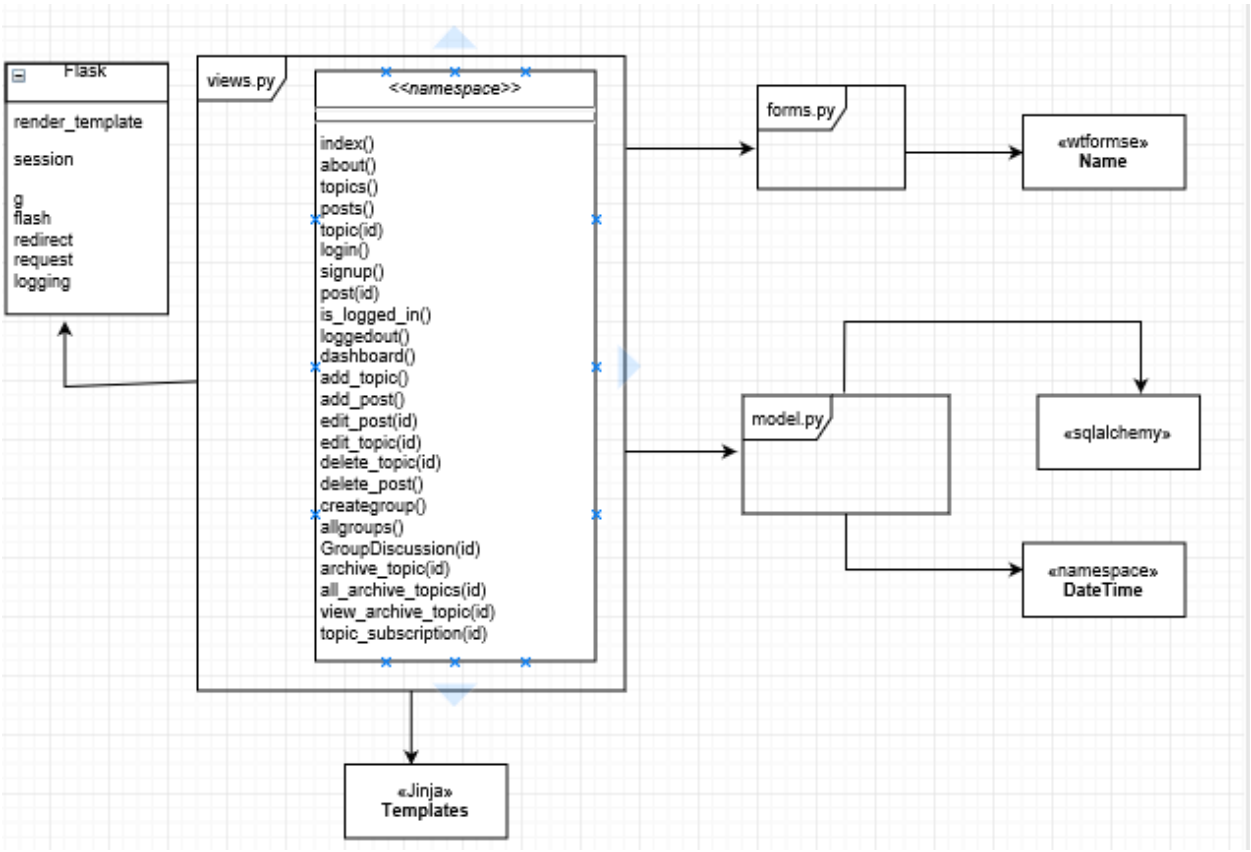
This is the Low-Level design for the Models.py folder. In it objects are created as tables in the SQL-Alchemy database and the objects attributes are given by the columns that are created by in the tables.

Low-level design for form.py



Here we have the forms that are going to be displayed to the user, forms.py inherits from the wtforms.Forms, and it creates forms with the variables within the interface.

Low – level design for views.py



6

A . User guide

The main project deliverable was implemented successfully. The functional requirements outlined for the group project were satisfied and the breakdown of each functional requirement is as follows:

Requirement:

1. Archive feature which was approved by the instructor and this feature basically archived the posts and stored them in the archived column in the database and it will remain in the database forever

2.User authentication was implemented and for login it would validate the username and password from the database and signup would create the user and add it to the database and then the user can login in using the credentials

3.Creation and editing of their own posting by authenticated users this feature allowed the registered users to create a new post or edit their own post

4. User subscription to topics – as we were not sure of what means of notification the project required we just flashed a message when the user selected subscription

5. Creation of discussion groups**********

The project deliverables were completed as per the description

Gantt chart with all the breakdown of the project with start and end times as well as the breakdown of all the tasks among the group members

The requirements file was also completed with a prototype and its full implementation

The design document consisted of the high-level design and low-level design of each module or package

We tested the coded with 15 suites to test the flask app created and all the tests ran well without any error

Using setup.py we created a package which needs to be installed in the environment.

B. Project review

For the current project, our entire project runs mainly on sql-alchemy, for the next implementation the modules should be separated so that the dependencies between the modules is reduced. Currently, the whole project would not be able to run without any of the modules, and if any of the modules are changed, the remaining other modules will have to be updated. The next implementation we would also like to implement better security features for the passwords, it would be best to encrypt the passwords before the its stored in the database of your choosing.

Also, the features that we were not able to implement within the requirements list (the features in the red blocks) would have to be implemented in the next implementation