ADIPIT02 - OBJECT ORIENTED DESIGN AND PROGRAMMING

TOPIC: Traveler’s Journal

Report By: Aurora

- Ishan Shrestha (NP03A180083)
- Rajesh Kawan(NP03A180090)
- Smita Shrestha (NP03A180080)
- Subin Shrestha (NP03A180072)

Submitted To:

Mr. Raj Shrestha

Table of Contents

Table of Figures

1. Project Description

We have decided to create a social platform for travelers to share their travel experience. This gives travelers a platform to share their experience as well as their creative talent. Also recommend places to stay and eat or unexplored places that normal searches may not show. This platform gives future travelers a platform where they can read their fellow travelers experience on the place they are planning to visit. This gives future travelers much needed insight that is needed during the planning of the trip.

On our website, travelers can share a photo from their travel or write a blogpost summarizing their trip. We let the creators decide how they want to share their experience. This project also provides readers an option to follow their favorite travelers, like their post and leave the comment.

2. Methodology

For our project development, we have decided to use agile development framework Scrum to develop our project. Scrum is a light weight framework that allows you to work through complicated evolving problems. It is a high-speed approach; Sprints are central to the scrum workflow.

2.1 Advantages

Scrum is one of the most popular agile development framework. It makes complex software development process very easy and manageable. Scrum work flow gives retrospect at the end of a sprint to evaluate what worked during the sprint and what did not. It also provides an estimate on when software would be completed and when new features will be updated.

2.2 Disadvantages

The sprints can be difficult to plan. Not being able to properly plan and estimate what can be done with in a given sprint can throw the development process into chaos. Not being able to complete the task planned during a particular sprint can lead to team members being demoralized.

3. UML Diagrams

3.1 Use case Diagrams

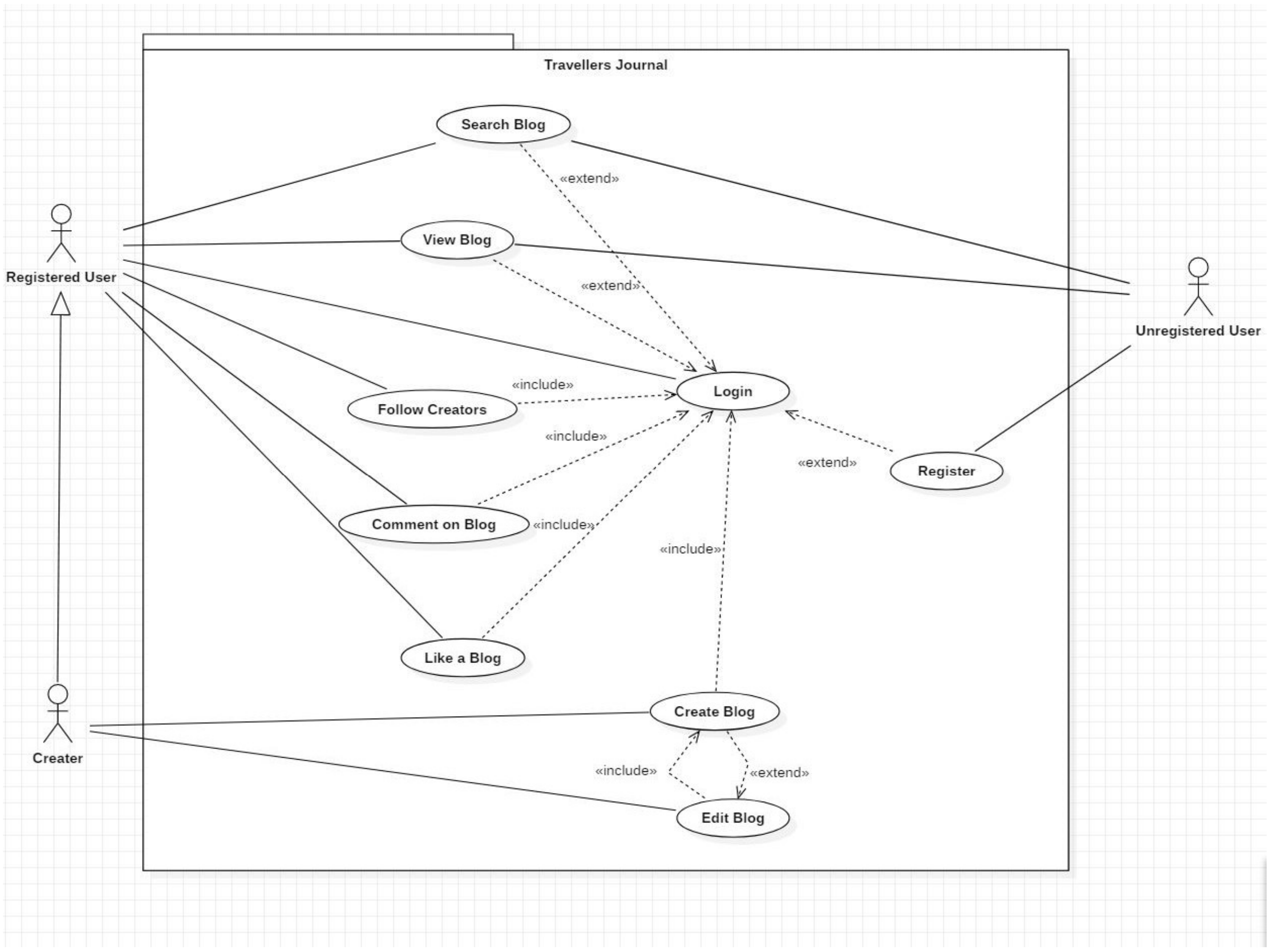

Figure 1: Use case Diagram

Description:

Above use case diagram show the flow of operation carried out in our system from our understanding on the system till now. System allows unregistered user to view and search for the content. If user wants to access other facilities within the system like creating blogs, comment, follow fellow user etc. Registered user gets access to creating and editing blogs which is viewed by other users (Travelers). While the system is idle, system manages as well as updates user details and also add new features if available.

3.2 Class Diagram

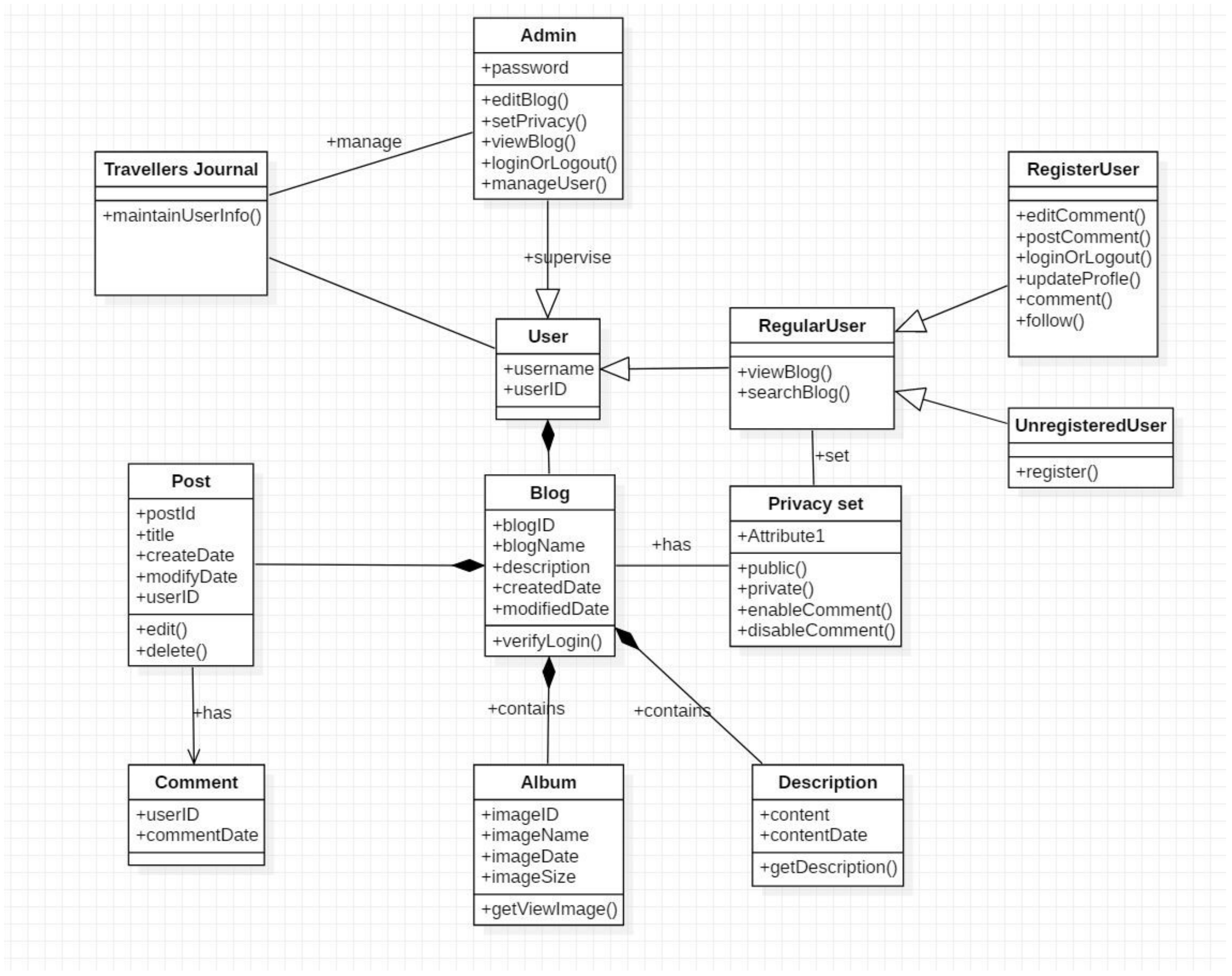

Figure 2: Class Diagram

Above class diagram show the necessary classes required in the system. Each classes represent each entities that includes user, system, admin and other. Admin class manages the whole system while user class interact with the system like viewing blogs, creating and editing etc. Blog class includes content like images and description which is interacted by the user depending upon security level which is included by privacy set class. System will continue to update or changes. So new classes with be included with introduction of new idea.

3.3 Activity Diagram

3.3.1 Activity Diagram of creating a post:

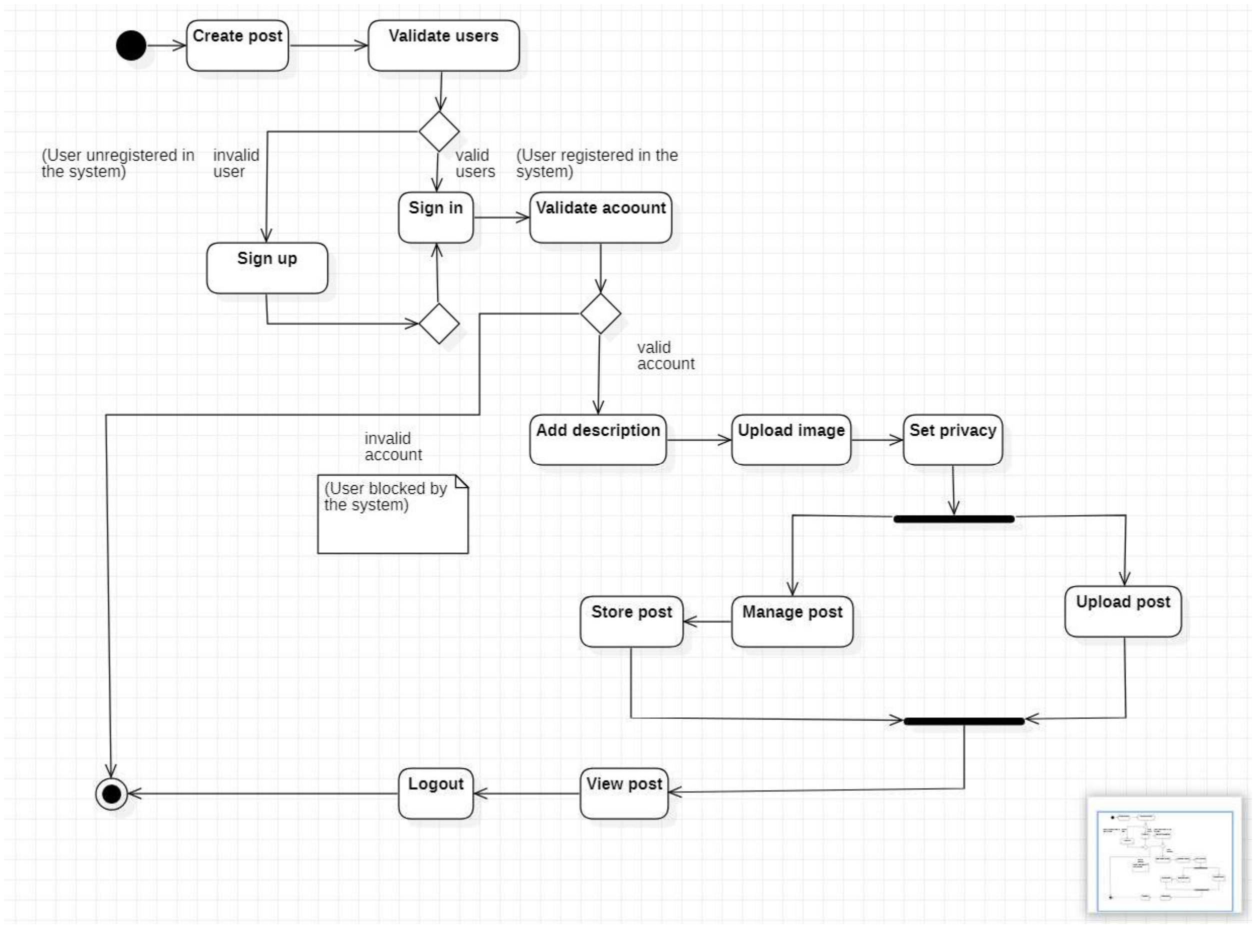

Figure 3: Activity diagram of create post (Use case)

Above activity diagram shows the flow of operation within Create post (Use case). For user to create post, user needs to perform every activities mentioned in above activity diagram.

For user to create post, user need to be registered. Else, user is requested to sign up and then created desirable post if user account is validated i.e. user is valid (**NOT Blocked**) by the system. Then, user adds necessary description, images for the post and set its privacy to be viewed by. As users uploads his/her post, post is then managed by the main system and also post content is stored in database. Uploaded post is then viewed by the user.

3.3.2 Activity Diagram of viewing and reacting to post:

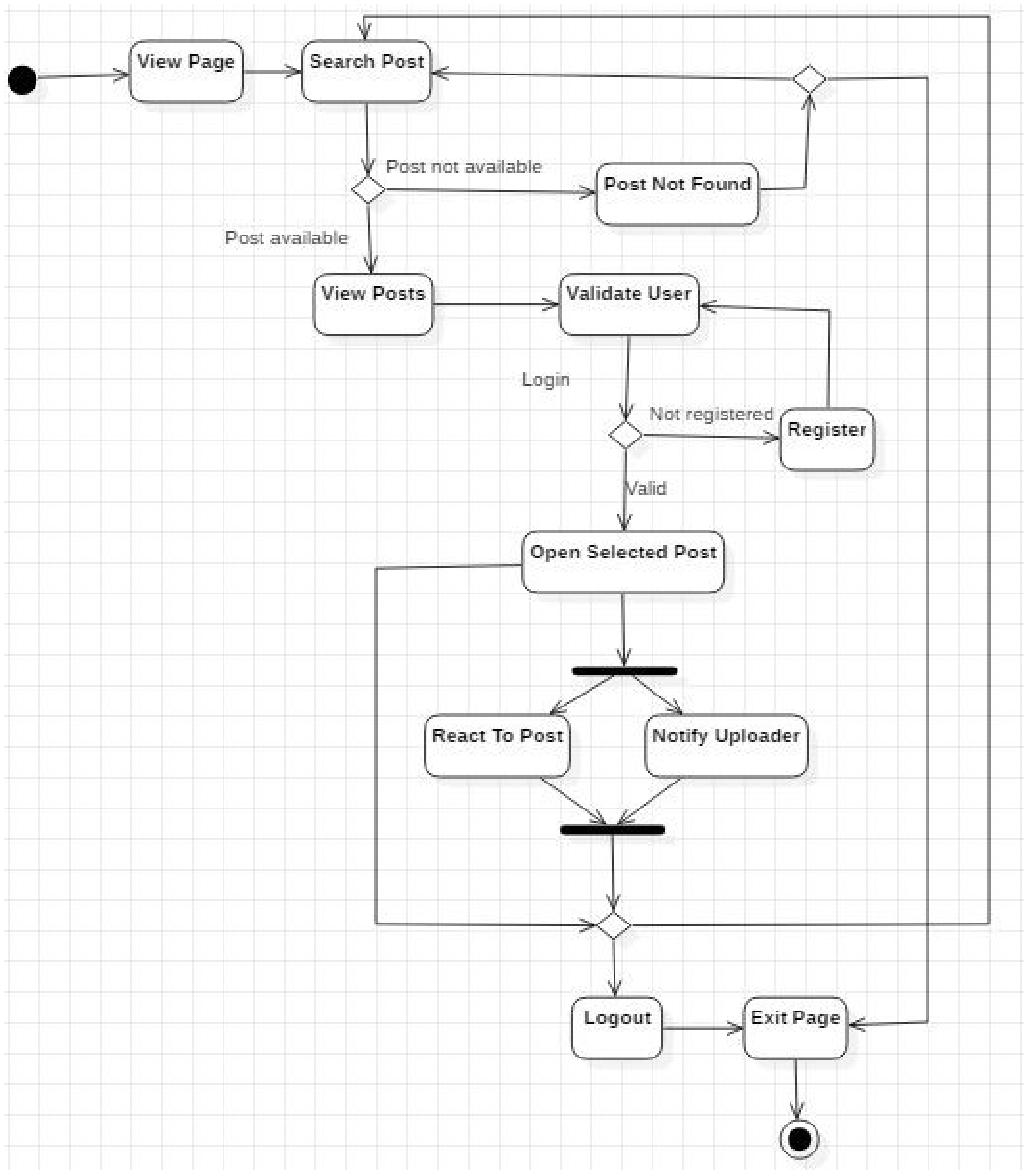

Figure 4: Activity Diagram for reacting to post

Above activity diagram explains the workflow of viewing a post and reacting to it. Each activity in the above diagram represents the steps. When one activity is completed, user switches to another activity. At first, the user is on the view page where number of posts will be displayed. Then user searches for the particular post that user wants to see. In case the relevant post is not found, the user has a choice to exit the page or loop back to search post. When the post is available, the user must login in order to react (like, comment, share) to the post. The user reacting to the post and notification message to the user who uploaded the post occur simultaneously. After reacting to the post, user can switch to search another post activity or logout and exit the page.

3.4 Sequence Diagram

3.4.1 Sequence Diagram of creating a post:

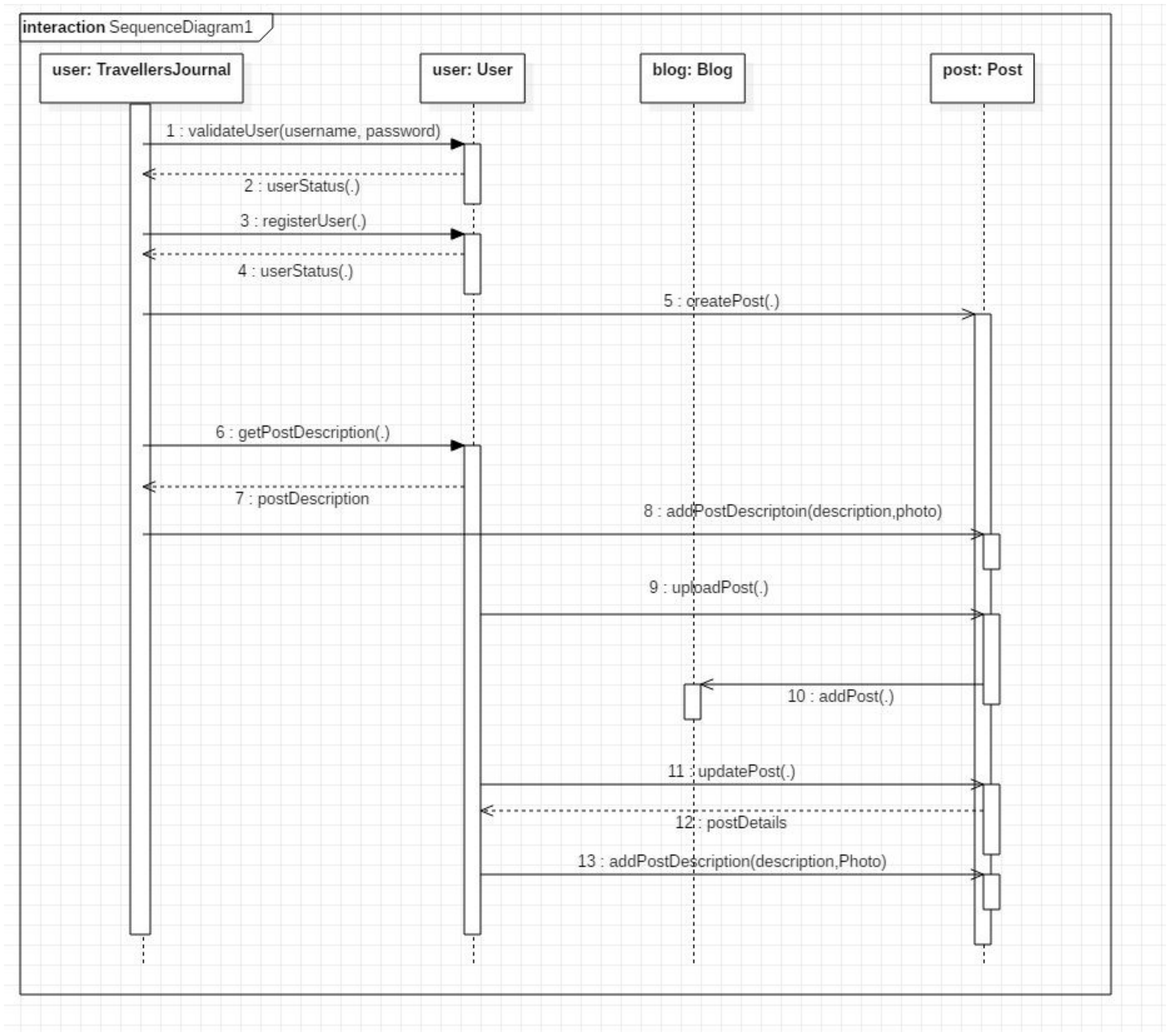

Figure 5: Sequence diagram of Creating a Post

Above Sequence diagram shows the interaction of objects when a user creates a post. Each life line represents the class of which the object is created to perform the operation requested by the user. When a user sends a request to create a post, traveler’s journal creates an instance of it and checks weather the user is logged in and is a valid user. It then sends request to the post class to create its instance to create a post. It then takes all the post description from the user and sends it to post class. When the user sends the request to upload the upload the post, its requests the post class to upload it and then the post class adds the created post to the user’s blog. After this as per the requirement, user can send request to post class in order to update the post.

3.4.2 Sequence diagram of View post (Use case):

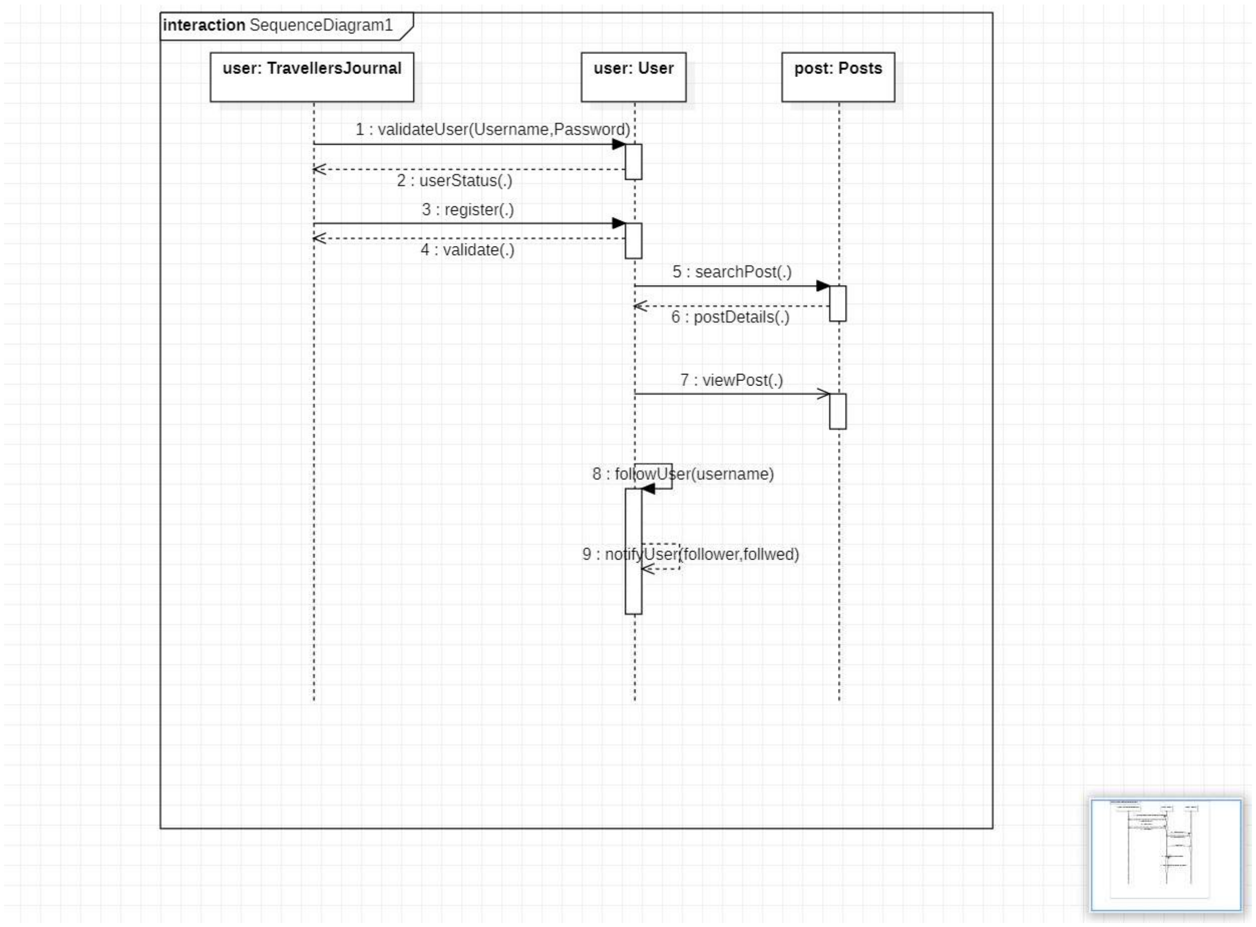

Figure 6: sequence diagram of view post

The main objective of the sequence diagram is to show how the objects in the system interact. For the view post usecase, we have three objects in the above diagram. They are: TravellersJournal, User and Posts. Firstly, the user must validate their id in order the view the selected page. In case of invalid id, the user must register. After entering valid id, the details of the post is viewed by the user where user can react to the post and also follow the user that uploaded the post.

3.5 Swinmlane activity diagram of Create post (Use case):

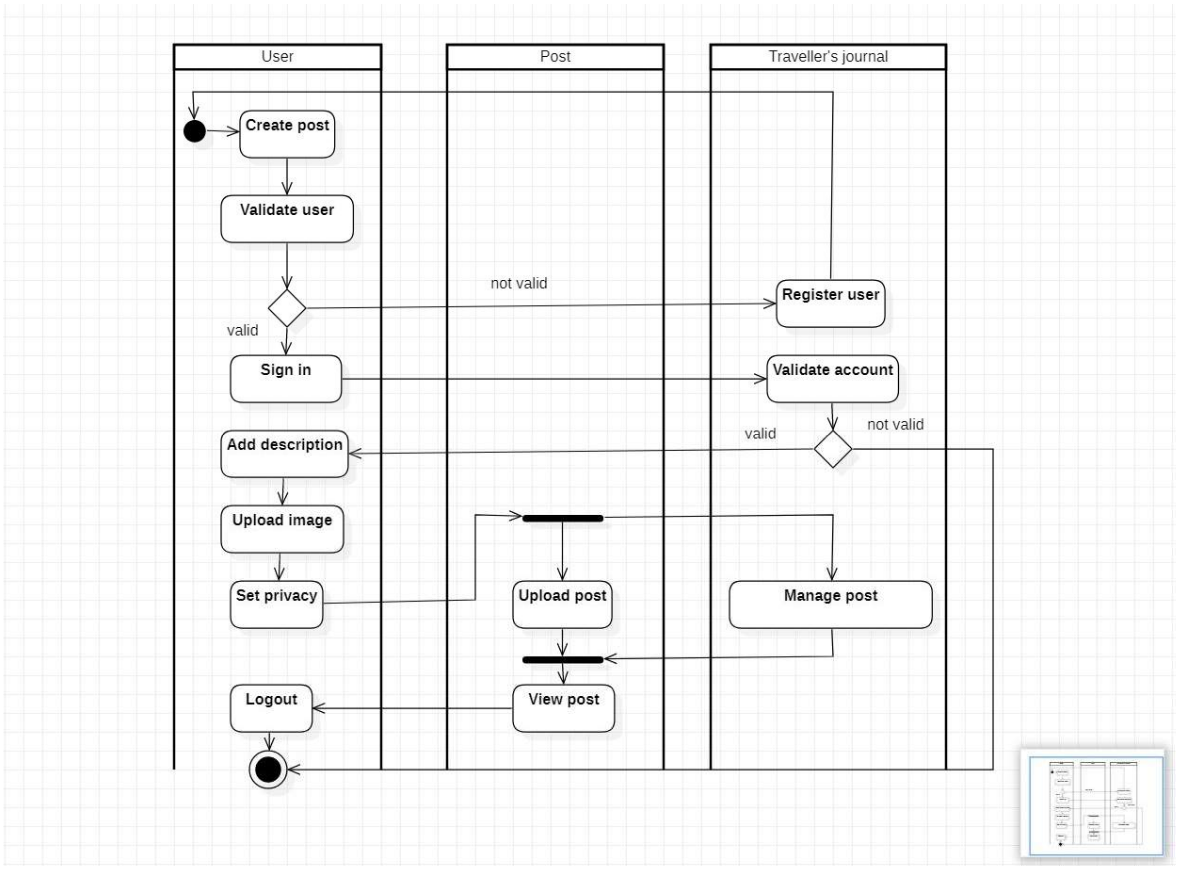

Figure 7: Swimlane activity diagram of Create post (Use case)

Above swimlane activity diagram shows the flow of operation within Create post (Use case). For user to create post, user needs to perform every activity mentioned in above swimlane activity diagram.

For user to create post, user need to be validated which is carried out within the User class. If not valid, user is requested to sign up or register and sign in. Then, user is validated in Traveler's journal class, if user account is validated i.e. user is valid (**NOT Blocked**) by the system. Then, user adds necessary description, images for the post and set its privacy to be viewed by in User class. Then user request upload method of Post class and uploads his/her post. Simultaneously, the post is managed by the Traveler's journal class and then post is displayed by Post class.

4. Software Architecture

Software Architecture is the abstraction of the system that aids in understanding how the system will behave and how each component of the system will interact with each other. Correct software architecture design pattern can help and improve software development process associated with large and complex software system. Software architecture model emphases on the architectural elements of data, processing and connection, highlights their relation and properties and also their constrains. (Perry & Wolf, 1992)

For our project we have decided to focus in using 3-layer pattern also known as Model-View-Controller pattern. Model-View-Controller is very common and a well-known method to architect an interactive, dynamic and flexible web application in a partition-independent pattern. (Leff & Rayfield, 2001)

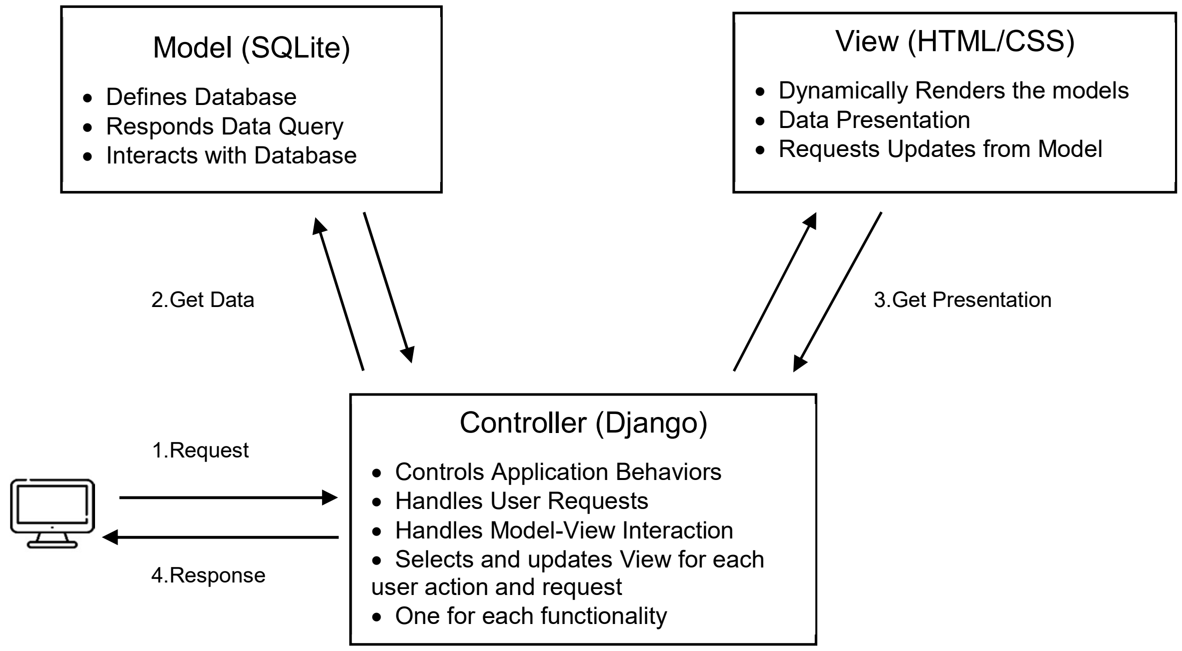
Model (SQLite)
Defines Database
Responds Data Query
Interacts with Database
View (HTML/CSS)
Dynamically Renders the models
Data Presentation
Requests Updates from Model
2.Get Data
3.Get Presentation
Controller (Django)
Controls Application Behaviors
Handles User Requests
Handles Model-View Interaction
Selects and updates View for each user action and request
One for each functionality
1.Request
4.Response

Figure 8: Model-View-Controller Design Pattern (Eltaeib & Palleboina, 2015)

The above diagram depicts our plan for using Model-View-Controller Pattern.

Model (SQLite):

We will be using SQLite as our model to create and interact with the database. In our project our primary data will be our user information and post details that our users will post. It will handle every query to provide information to view and controller elements in response for queries. It will also execute all the data logic for the management of database. (Eltaeib & Palleboina, 2015)

View (HTML and CSS):

We will be using HTML and CSS along with bootstrap as design tools for our web application. It will be able to retrieve data from model through controller and will also be able to dynamically display data. It will also provide input forms and controller for the user to interact with our application (Eltaeib & Palleboina, 2015)

Controller (Django)

We will be using python framework Django as our controller to manage and control both model and view. It will receive incoming request and routes them to appropriate handler and also handles actions from such requests. It will also provide response to the user depending upon the request made by user. It will also dynamically update the view as per the actions performed by the user. We will also be creating separate controller for each of the features in our application as it will emphasize low coupling and will reduce inter module dependency. (Eltaeib & Palleboina, 2015)

4.1 Advantages of Using MVC pattern

1. Fast web development as multiple developers can simultaneously work on each component.
2. Easy update and debug as MVC pattern as it has multiple partition for each functionality.
3. MVC pattern emphasizes low coupling and allows for efficient code reuse.
4. Due to Multiple level of partitions, this makes it easy for the application to interact with any third-party APIs.
5. MVC is a very flexible design pattern with manageable and extensible structure with distinct separation of concepts between different system processes or tasks.
   (Leff & Rayfield, 2001) (Rajagopal, et al., 2009) (Geeks for Geeks, 2019)

4.2 Disadvantages of Using MVC pattern

1. As MVC design pattern has multiple level of abstraction, it can be difficult to manage if documentation is not done properly.
2. MVC design pattern encourages developers to create partition very early in the development. Any changes in partitioning can considerably change the application's implementation.
   (Leff & Rayfield, 2001) (Geeks for Geeks, 2019)

5. References

Perry, D. E. & Wolf, A. L., 1992. Foundations for the Study of Software Architecture. *SOFTWARE ENGINEERING NOTES,* 17(4), pp. 40-52.

Eltaeib, P. & Palleboina, S., 2015. A Study of MVC – A Software Design Pattern or Web: Application Development on J2ee Architecture. *ournal of Multidisciplinary Engineering Science and Technology (JMEST),* 4 April, 2(4), pp. 666-667.

Geeks for Geeks, 2019. *geeksforgeeks.org.* [Online]
Available at: https://www.geeksforgeeks.org/mvc-design-pattern/
[Accessed 26 December 2019].

Leff, A. & Rayfield, J. T., 2001. Web-application development using the Model/View/Controller design pattern. *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference,* pp. 118-217.

Rajagopal, S., Ganapathi, P. & Kombiah, I., 2009. A Survey of Design Pattern Based Web Applications. *JOURNAL OF OBJECT TECHNOLOGY,* 8(2), pp. 61-70.