Analysis Me

**Spring**

2016

Technical Report

Team 11

Enclosed in this document is the technical report of the <Scientific Social Network> sponsored by <Jet Propulsion Laboratory>.

# Table of Contents

1. **Introduction**
2. **Motivation**
3. **Related work**
4. **System design**
5. **System implementation**
6. **Experiments and analysis**
7. **Conclusions and future work**

**1. Introduction**

Nowadays, more and more people talk or share life experience via social networks like Facebook, Twitter, etc. These social networks have been taking increasing responsibility of our daily interactions and communications, which play a significant role in our daily life. This trend also affects the development of scientific researches. We can easily find that more and more scientific findings come from collaborations and most of them benefit from effective communication at work. If a scientist or a researcher would like to achieve success in his or her career at present, they must make the best of modern communication tools to share ideas, findings and collaborate with colleagues. As a result, more and more researchers begin to use online tools to manage researches and share working outcomes, which is why we would like to develop a scientific targeted workflow social network.

A scientific workflow system is a specialized workflow management system designed specifically to compose and execute a series of computational or data manipulation steps, or workflow, in a scientific application The reason we use workflows is that workflows can encapsulate the process of data manipulations and allow us to focus on the overall structure of the complex project, which could simplify and speed up the research with complex experiments and steps. In general, we often use web services to afford critical computation of the workflows, and use local scripts to perform data format conversion and some other less important procedures. Since workflows liberate scientists and researchers from arranging complex steps and stages, it could increase the efficiency of scientific researches and save a lot of time and money in trivial procedures for scientists.

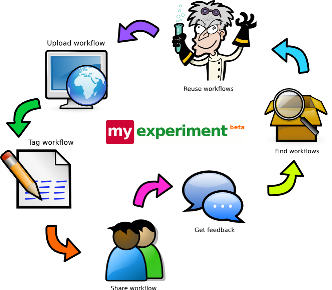


Figure 1.1

Design patterns are standard solutions designs for common problems in programming designs. The utilization of design patterns can simplify the structure of the program and involve more flexibility and extensibility to the program, which give us space to modify and maintain the workflow social network. Based on the program provided, we introduce eight design patterns to refactor the project. They are Abstract Factory, State, Builder, Composite, Adapter, Iterator, Facade and Template Method pattern.

**2. Motivation**

Most of the scientific social networks in the market at present ignore the needs of workflow management and only focus on improving the social interaction part of their product. Since the tremendous needs of scientific social network applications, we have the idea of develop such a web service to help researchers manage their experiments and researches, build social networks with peers.

The program provided has successfully realize a group of functionalities for the scientific social network. Our mission is to introduce some design patterns to refactor the project in reasonable ways. Design Patterns are standard repeatable solutions for common problems in software engineering. They are a kind of description or template for programmers to utilize in order to solve related problems in an elegant and reasonable way.

Design Patterns can help speed up the pace of development by providing programming paradigms. These paradigms give the standard solutions in software industry and introduce effective software designs. Since we must consider all of the issues that may not become visible until later in implementation and maintenance in the future, it is necessary to apply mature software designs in the industry to build our application. Reusing these designs could contribute to preventing subtle problems that may result in severe problems which may impede the pace of development. In addition, Design Patterns are a kind of standard language used among coders and developers. Applying design patterns could improve the readability of the program and help us describe the structure of the program in an easy way.

Design Patterns provide general solutions for common recurring problems that we may encounter in software development. They are not tied to a particular problem and can be applied in the similar scenarios. This attribute of design patterns helps programmers to take advantage of Design Patterns to refactor their code and build a more reasonable and extensible program. Furthermore, design patterns allow us to communicate via standard terminology and jargons, which could increase the efficiency of interaction remarkably and make the team more energetic. Finally, all the well-known Design Patterns can be improved over time, which could make our project more and more robust and extensible in the future.

**3. Related Work**

1). Set up Docker environment

2). Install IntelliJ IDE

3). Database design

4). Implement the Scientific Social Network main page in front-end server.

5). Implement API for form groups and interact with group members for users to connect with each other to the front-end server.

6). Implement a scientific workflow management system in front-end server

7). Designed to arrange a group of computational steps and workflows in a scientific application in back-end server.

8). Implement State Pattern for the AppState for the Application in user membership states.

9). Implement Iterator Pattern for the User to initialize the GroupUser for iterating all the users in a group.

10). Implement Builder Pattern to build a complex user class with user levels and user bases.

11). Implement Adapter Pattern to join functionalities of independent or incompatible interfaces for the UserInfo class to get the level and rights for different users.

12). Implements Composite Pattern to group different types of the comments based on the number of thumbs up as a single comment object.

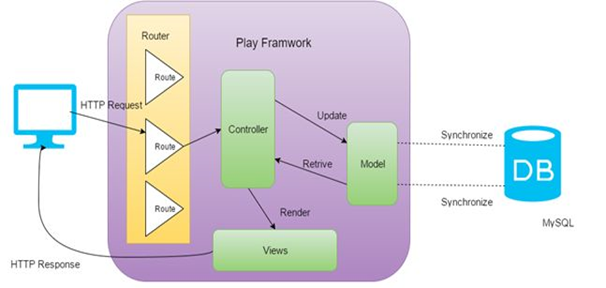
13). Implements Façade Pattern to implement the Feedback interface for the users to utilize the FeedbackMaker object to creat comments or suggestion.

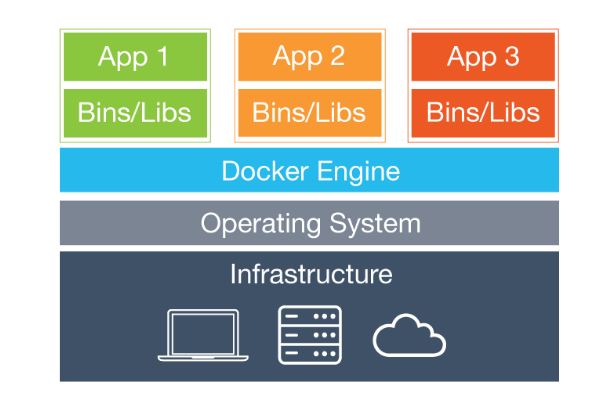
14). Implement Abstract Factory Pattern to implement the ModelFactory and RepositoryFactory to create the Model and Factory objects for the application.

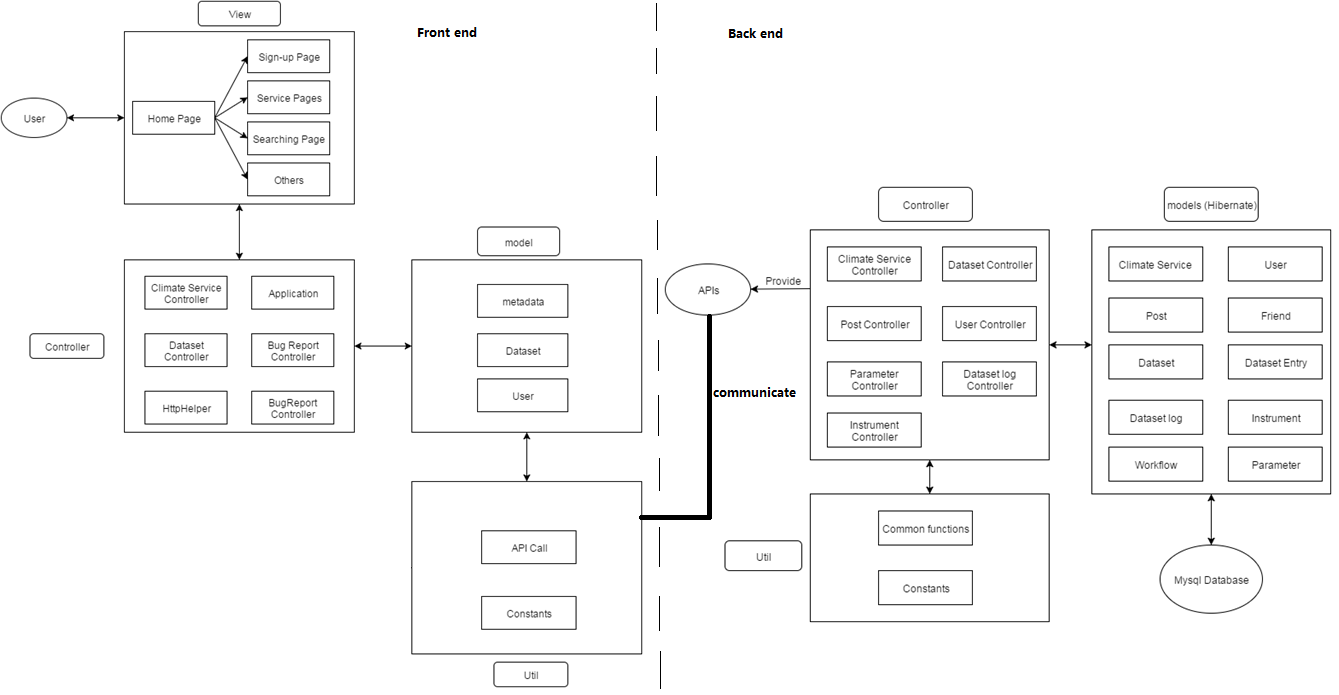
15). Implement Template Pattern to implement the Abstract Flow class and implement concrete template methods and abstract methods.

**4. System Design**

For simplification, the whole system is designed to be two parts, front end and back end, classified by its functionality. And use the Play Framework and Docker Framework as the enabling technologies.



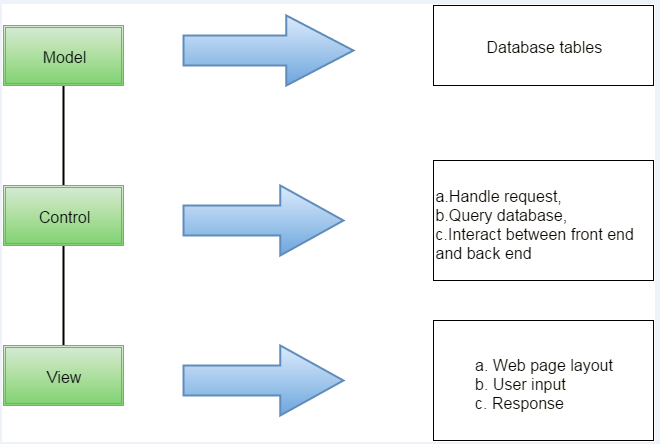




Front end is mainly about user interface, not only includes the graphical web page, but also includes some functionality such as make connection with other users with back end.

Back end receive data from front end and store them in to data base. When receiving requests, it fetches data from data base and give back to front end.

Both in front end and back end, we further divide the design into Model, Control, and View.



As illustrated in the graph, MVC design divide the work into three independent part.

Model layer is mainly about database structure design and CRUD method implementation. All database information are managed in this layer.

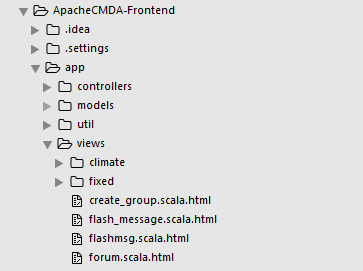
Control layer is the interaction part for front end and back end. It basically handle requests and query database for information.

View mainly deals with the interaction with users. Web page layout is designed in this part. Further, it receives the user inputs and send back the response after business logic.

**5. System Implementation**

1) Front-end server

Front-end server is for collecting user input information, which is built on Play framework. The following is the structure of our front-end system:



View part implementation:

In view part, to give a beautiful and comfortable user interface, Bootstrap and JQuery are used to provide dynamic response to users.

Controller:

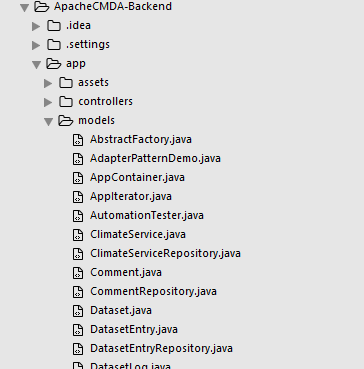
Controller is heart of the front-end server, which will pass the data collected from View part to Model part, and finally send the data to the database in back-end server. Controllers in this system is implemented with a number of specific actions, using java code.

Model and Util:

Model and Util are responsible for the communication java functions between front-end server and back-end server. Model contains the important data class definition for viewing and Util has API calling function for getting or posting data to back-end server.

2) Back-end server

Back-end server is for storing data collected from frond-end server, including dataset, climate services, user information and workflow. The structure is as follows:



Controller:

When the front-end server sends a request for some information through APIs, controller will access the Mysql database to create, retrieve, delete or update the data and response the front-end server accordingly.

Model:

We use Hibernate to implement our database access functions. Those models are mapped into tables in a Mysql database. The key database tables design is as follows:



Util:

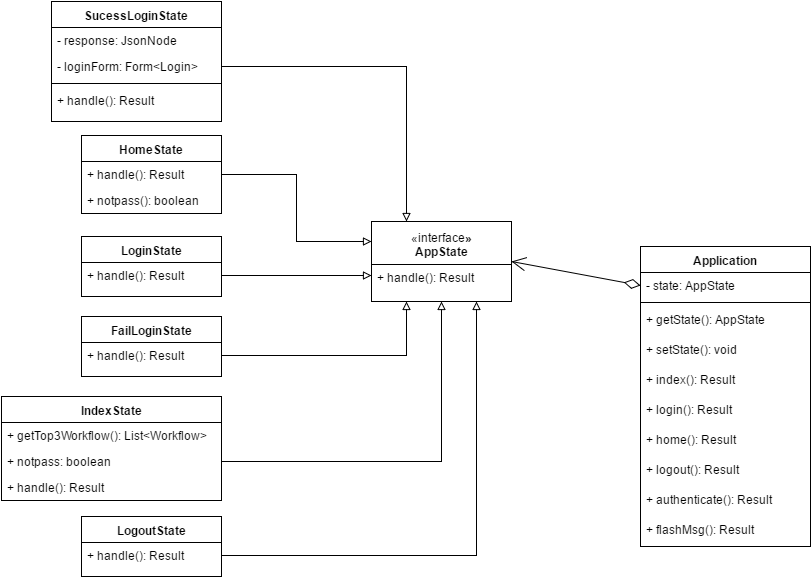
Provide some common functions and constants which might be used in controllers.

3) Design patterns

To make the code extensible and the program solution reusable, we utilize 8 design pattern in our code implementation,

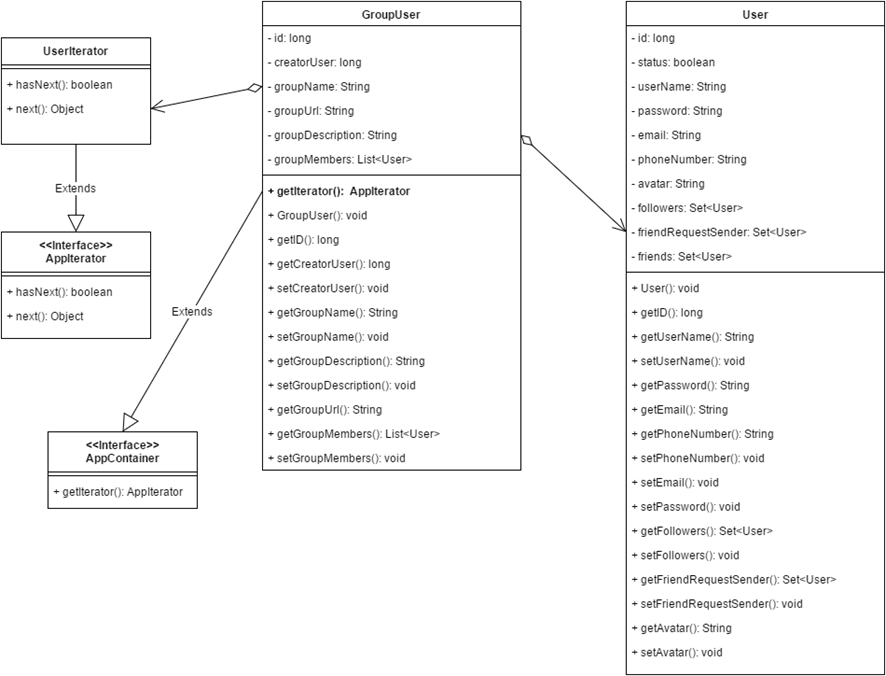
3.1) State Pattern

The behavior of an object depends on its state. The behavior need to change according to the run-time state. In common cases, the object may have numerous states. Below is the class diagram:



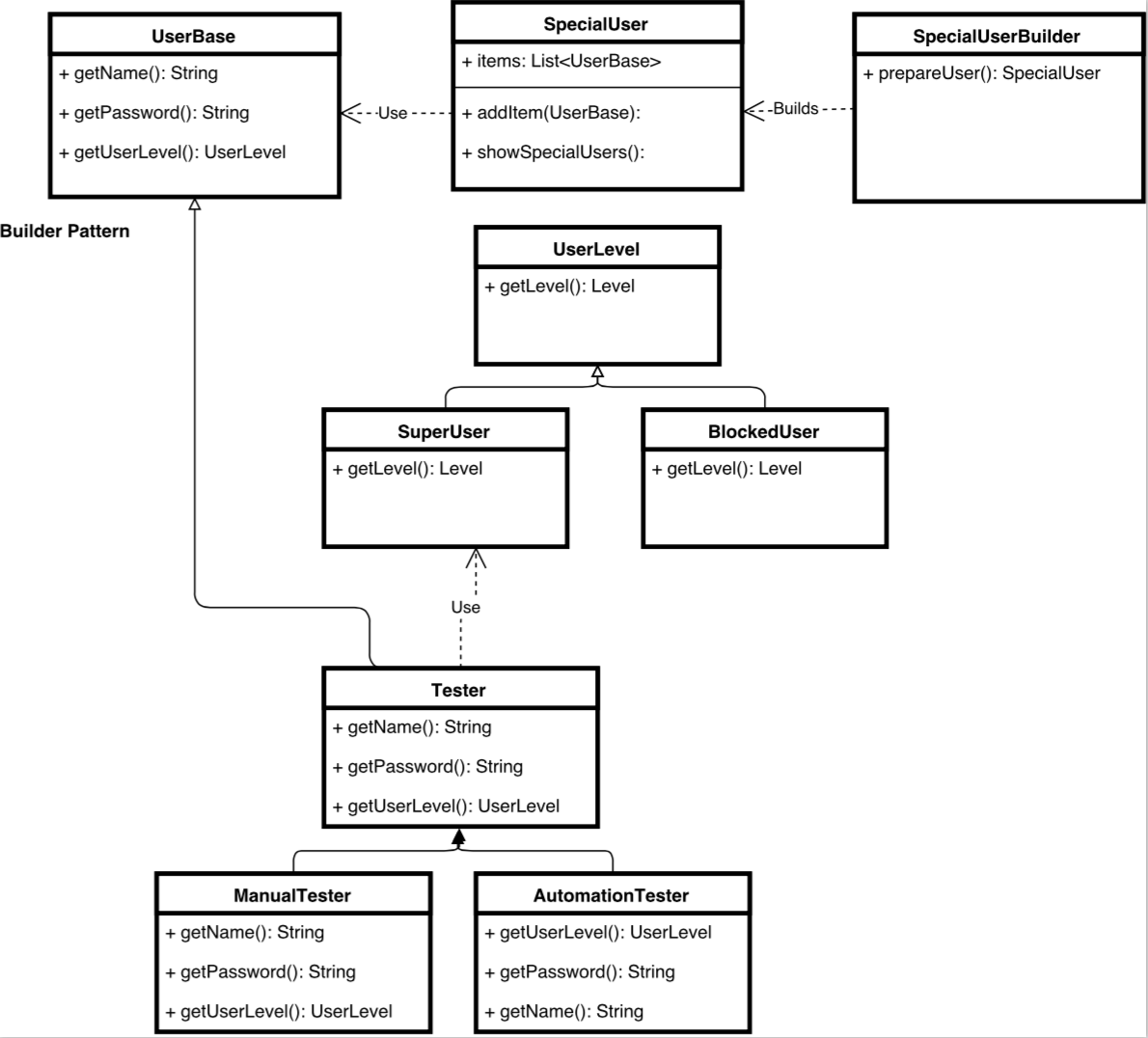
3.2) Iterator Pattern

Provide a way to access the elements of a collection sequentially without exposing its underlying representation. Use the iterator object to take the responsibility of traversing elements.



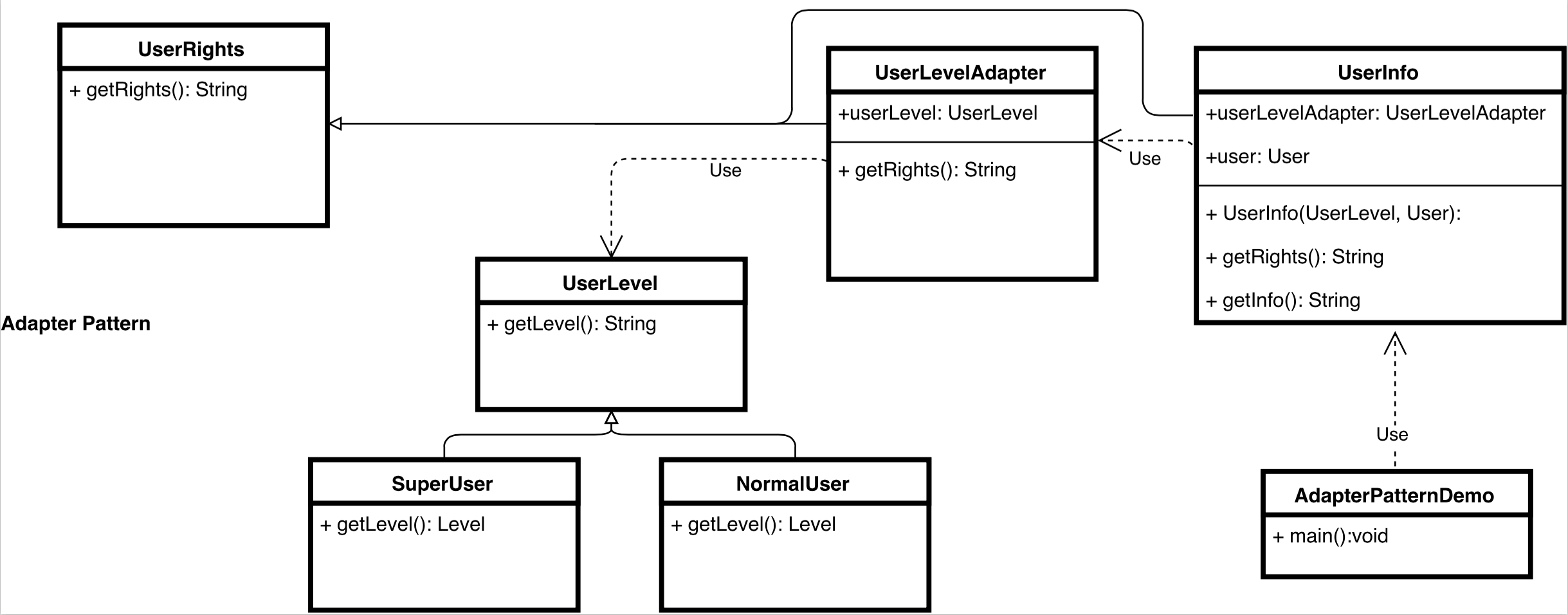
3.3) Builder Pattern

Builder Pattern is an object creation design pattern. It builds a complex object using simple objects and using a step by step approach.



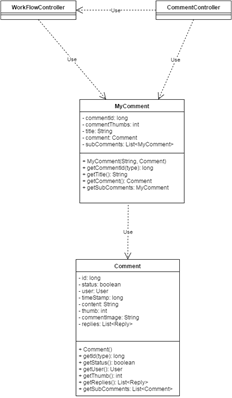
3.4) Adapter Pattern

Works as a bridge between two incompatible interfaces. This pattern involves a single class which is responsible to join functionalities of independent or incompatible interfaces.



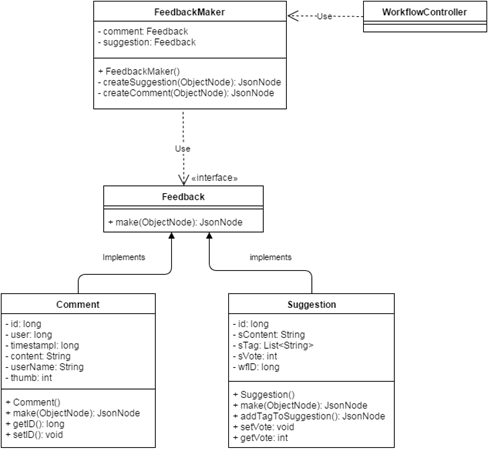
3.5) Composite Pattern

Composite pattern is used where we need to treat a group of objects in similar way as a single object. The key concept is that you can manipulate a single instance of the object just as you would manipulate a group of them.



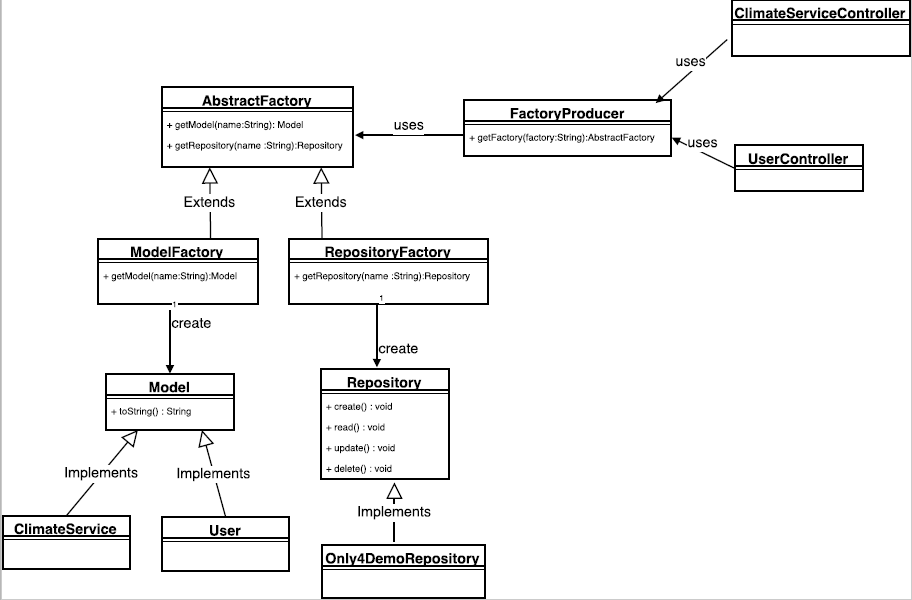
3.6) Facade Pattern

The purpose is to hide the complexities of the system. Facade pattern will provide an interface to the client and client can use the provided interface to access the system.



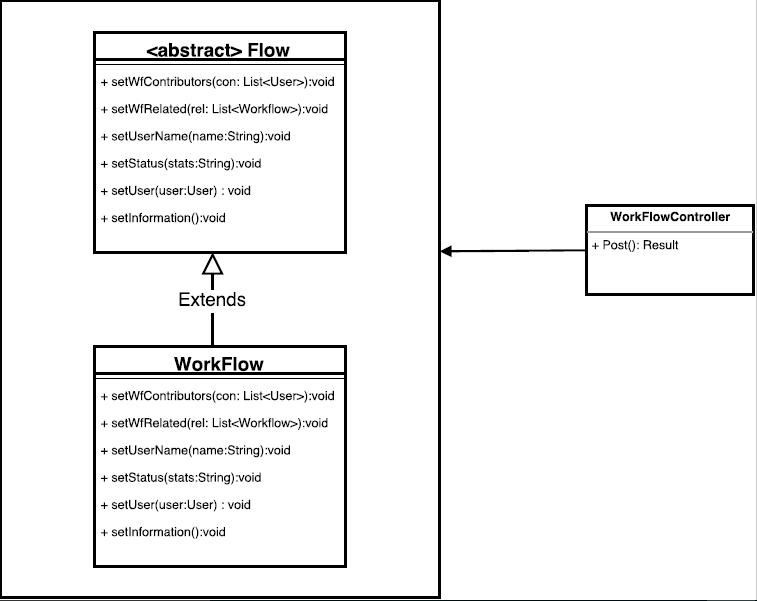
3.7) Abstract Factory

Abstract factory is a factory of factories. With abstract factory pattern, we can hide the details of the instantiation of classes. This gives us a lot of flexibility to decide what kind of class to generate.



3.8) Template Pattern

Template Pattern is to use an abstract class to expose defined ways to execute its method. It has a concrete template method and several abstract methods. Those abstract methods are overridden in its subclasses.



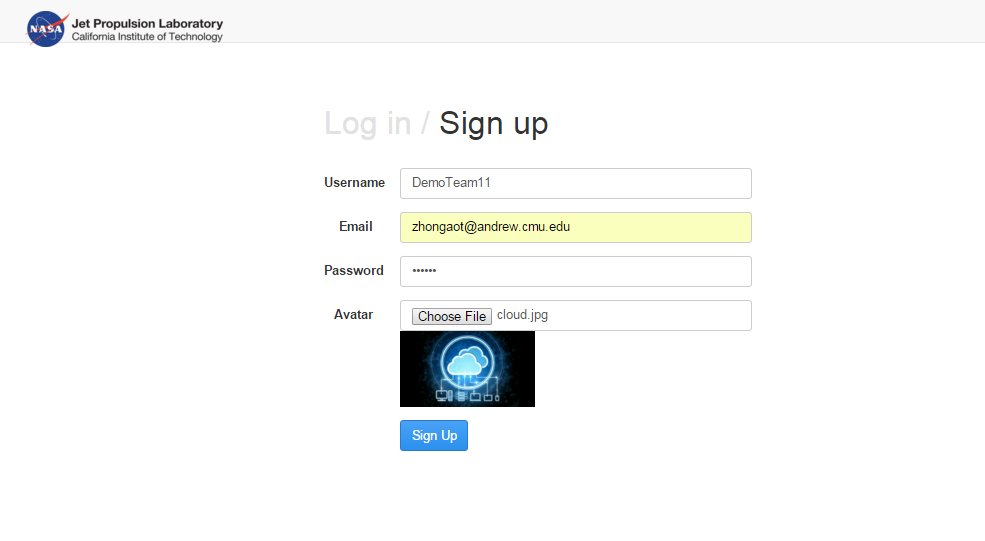
**6. Experiments and analysis**

We will test the following functions of our website and analyze the result:

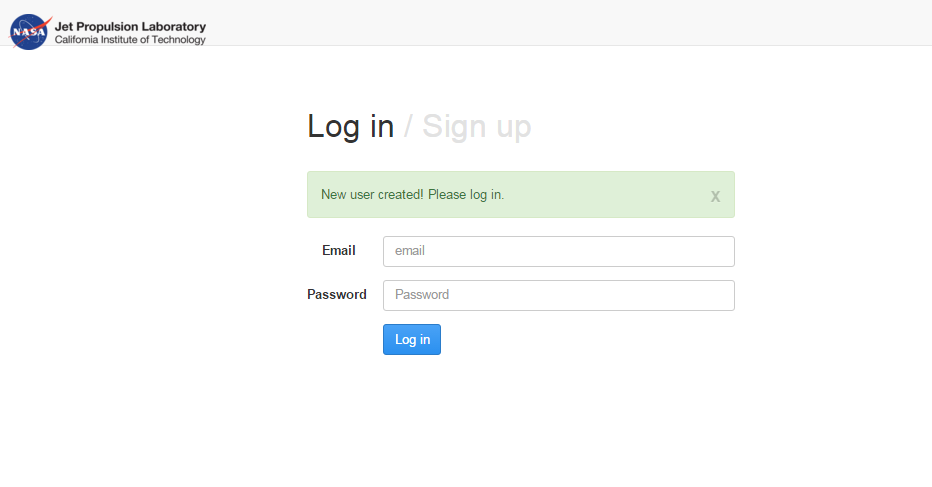
* Sign-up and login.
* Check the profile page for the user.
* Create a workflow.
* Check if the forum contains the workflow created in the previous step.
* Comment a workflow and test its thumbs-up action.
* Make a suggestion to a workflow and test its thumbs-up action.
* Switch a user and check if the data has been successfully updated in the backend server.

***1) Sign-up and login experiment:***

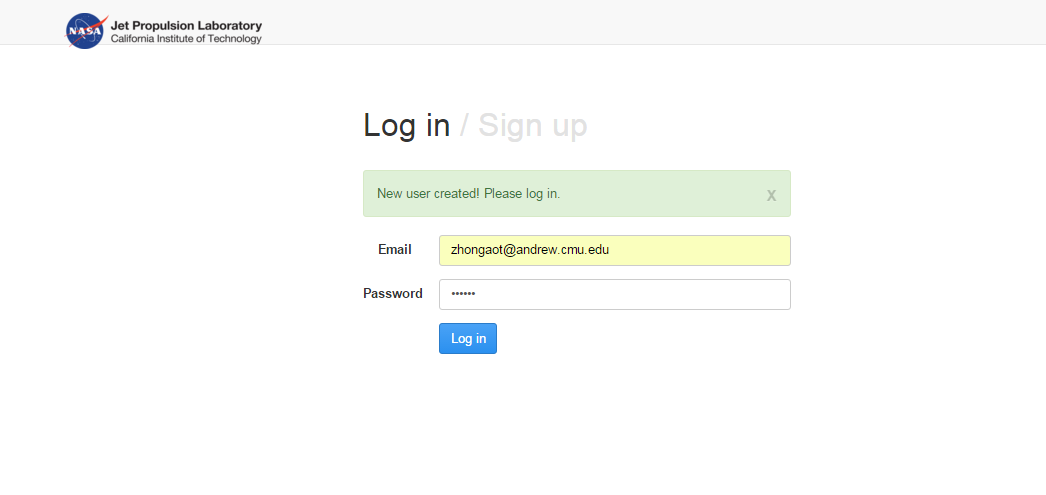
Sign-up page:



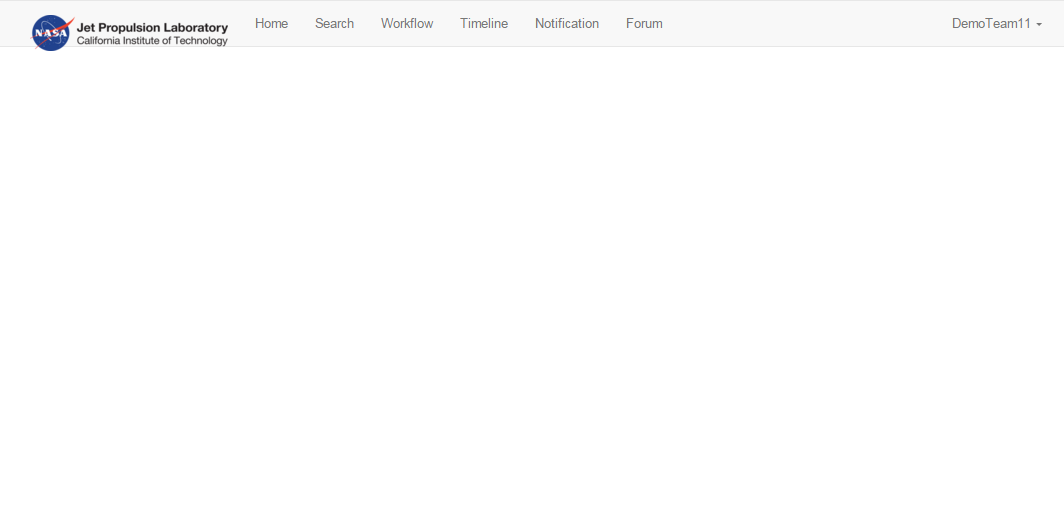
Registered successfully:



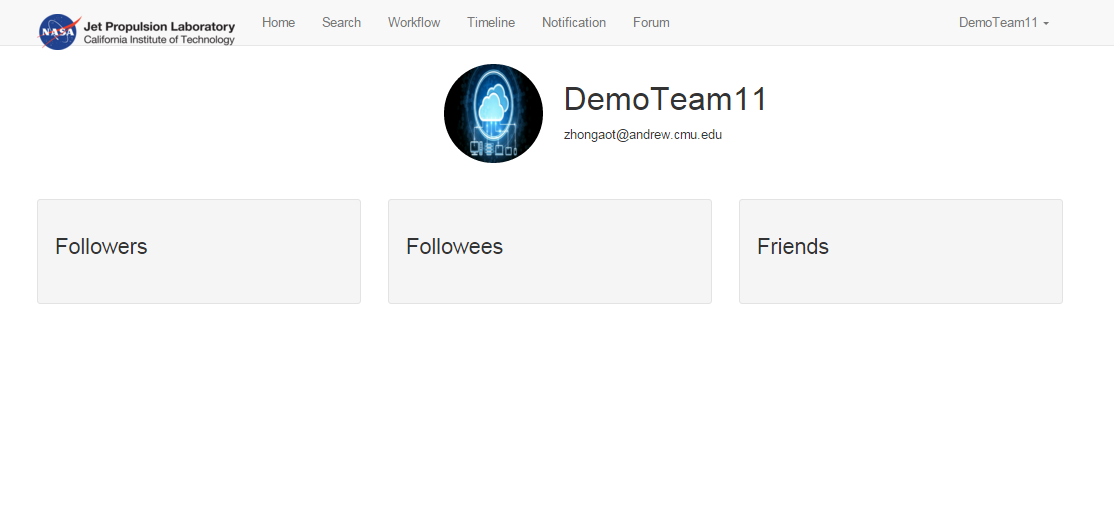
Login test:



After successfully login the website:

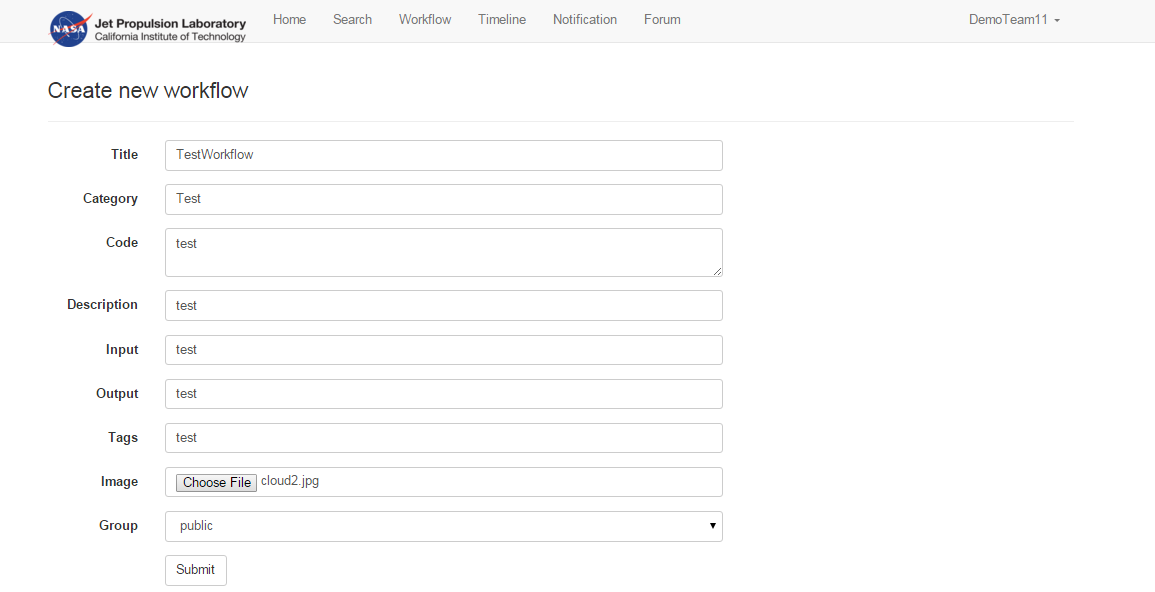


***2) Experiment of checking the profile page for the user.***

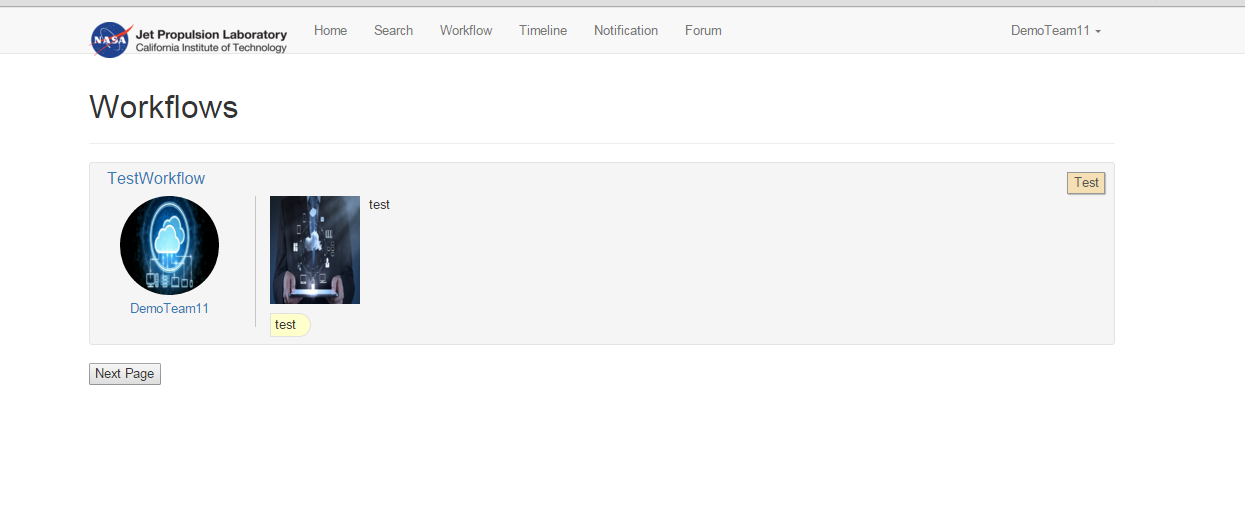


***3) Experiment - Create a workflow.***

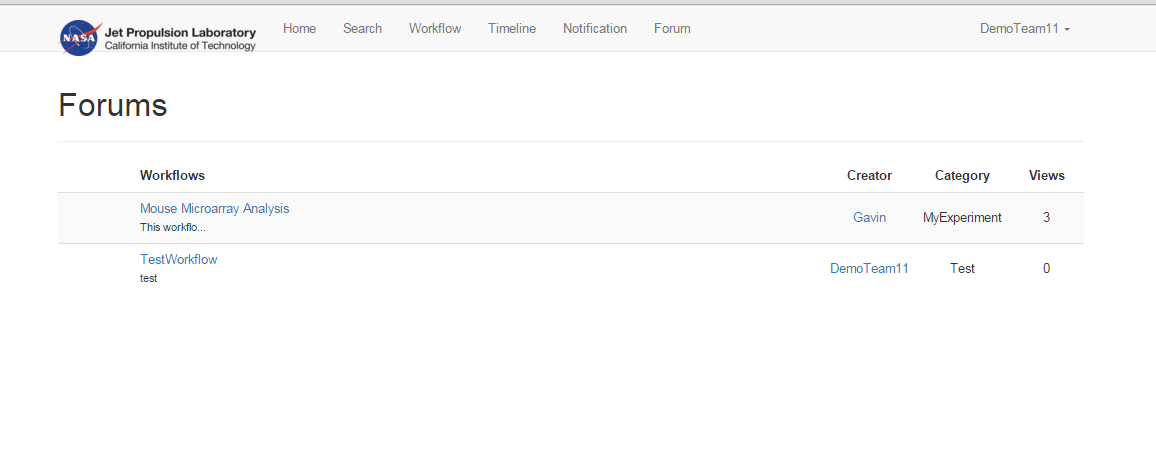
Workflow creating page:



Check if the the workflow created in the previous step:

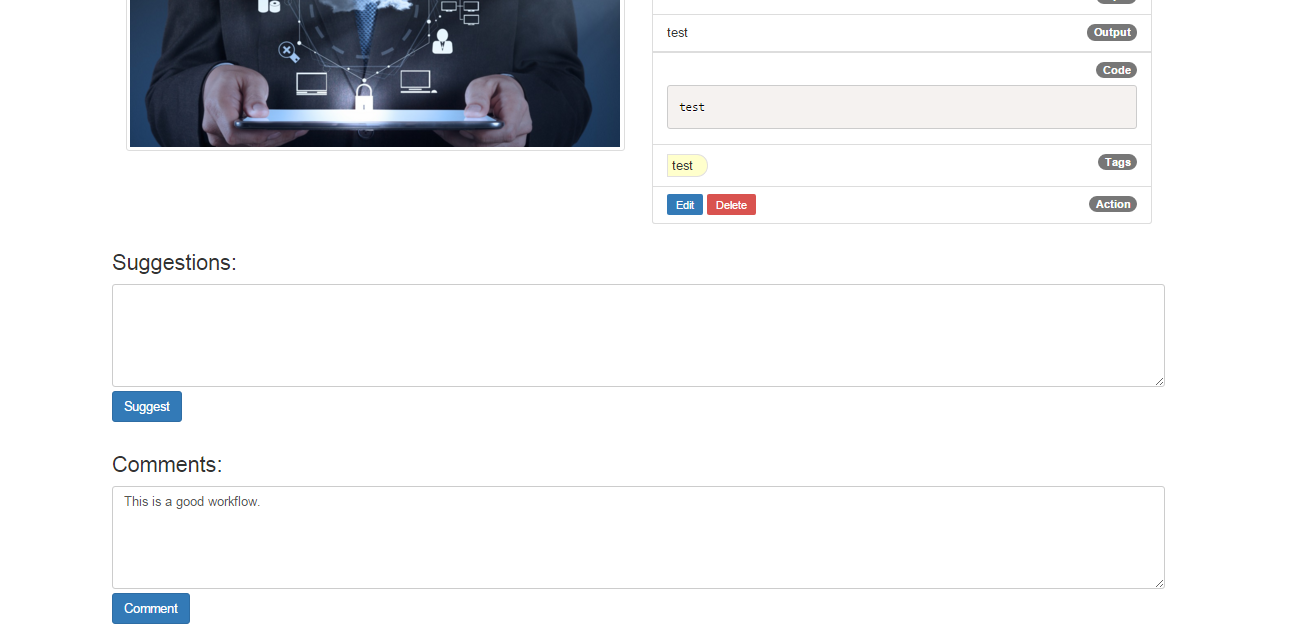


***4) Experiment – check if the forum contains the workflow created in the previous step.***

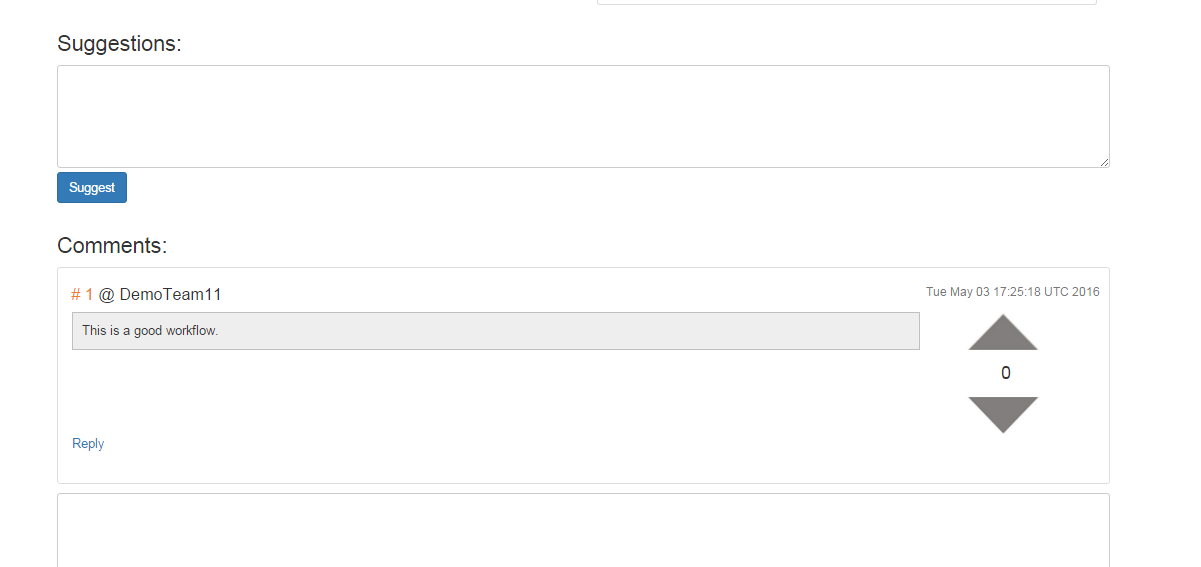


***5) Experiment – Comment a workflow and test its thumbs-up action.***

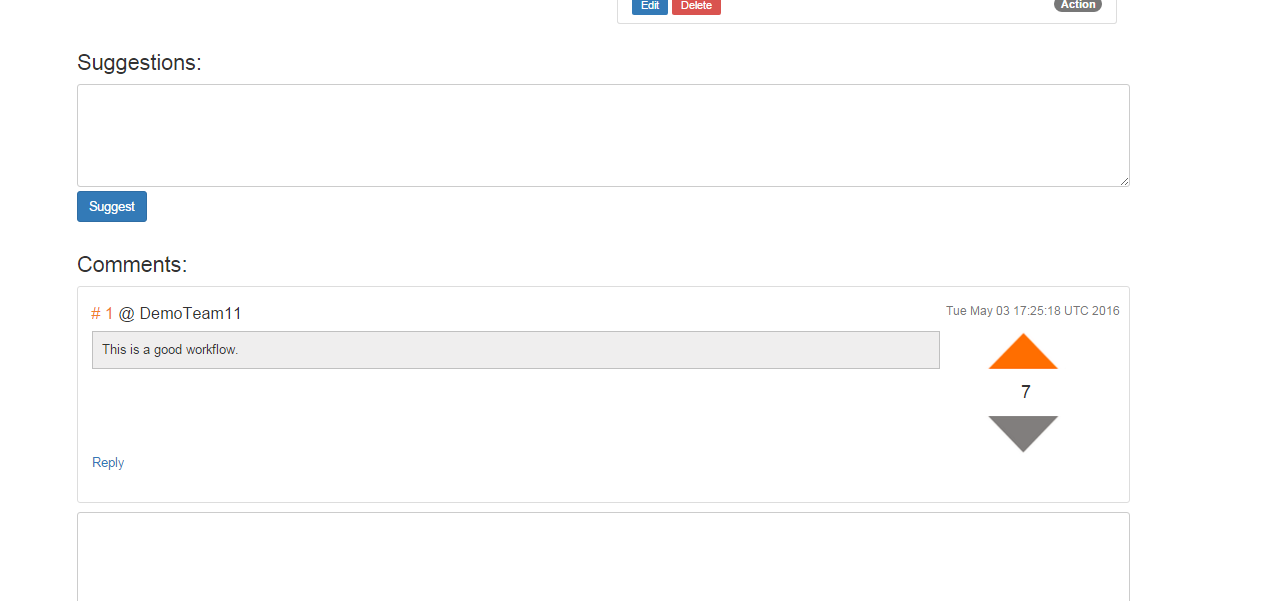
Comment a workflow:



After successfully comment a workflow:

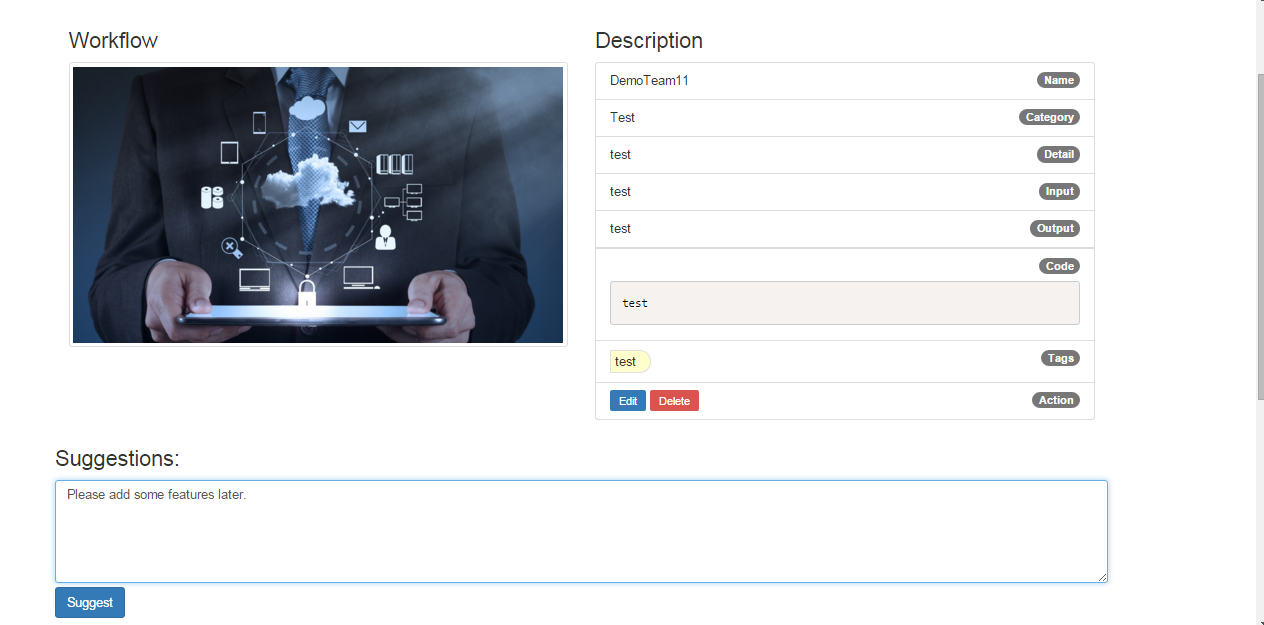


Test the thumbs-up function for a comment:

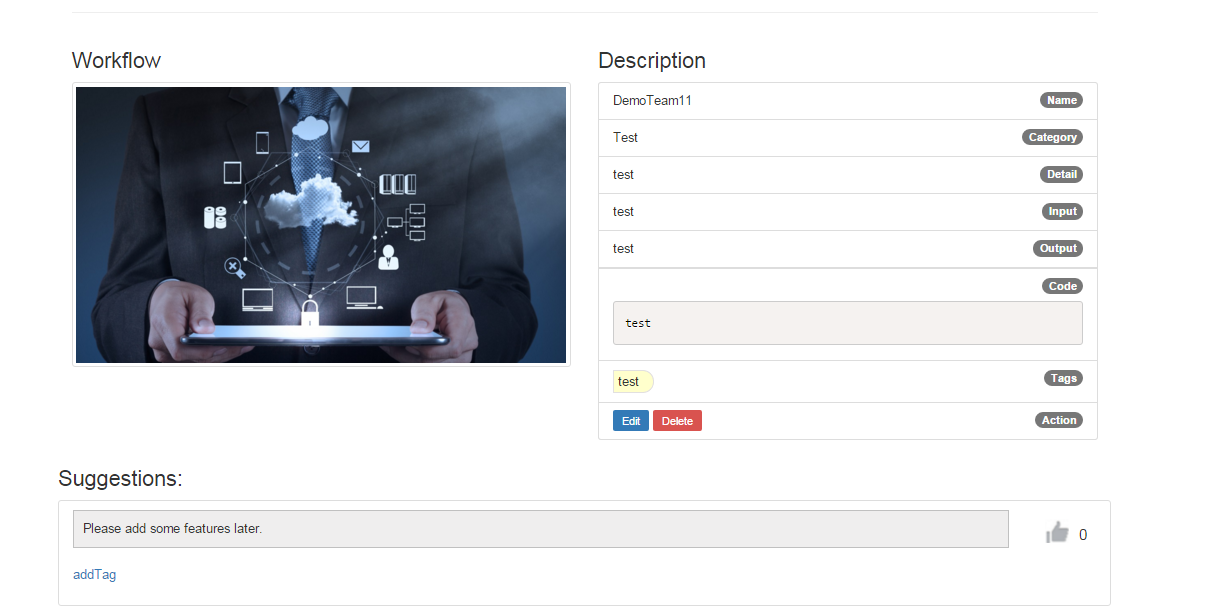


***6) Experiment – Make a suggestion to a workflow and test its thumbs-up action.***

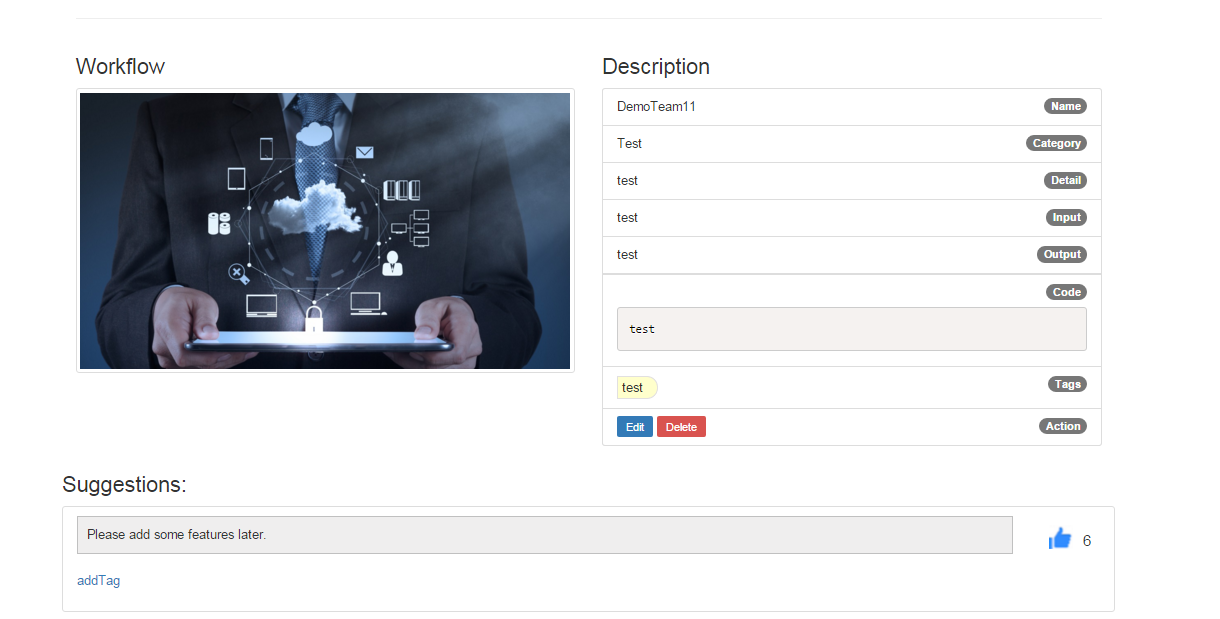
Make a suggestion:



After successfully make a suggestion:

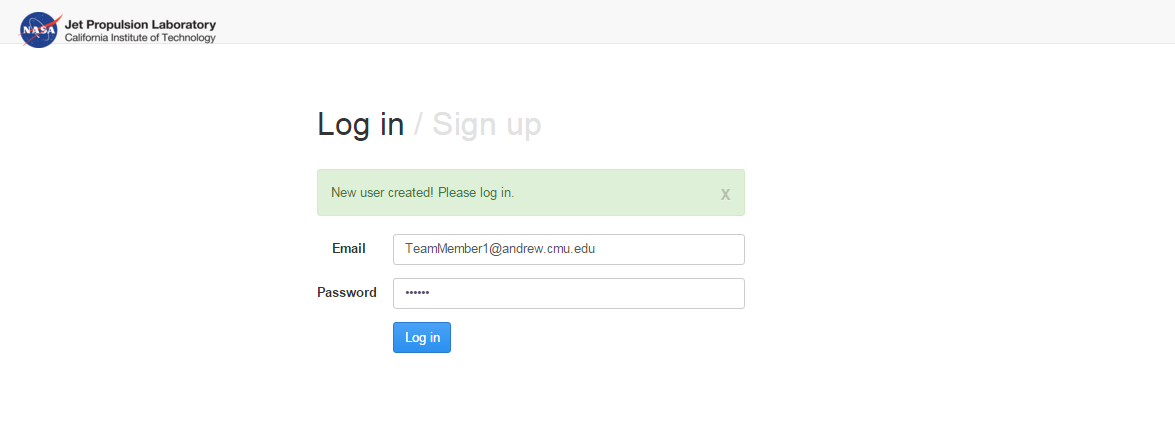


Test the thumbs-up function for a suggestion:

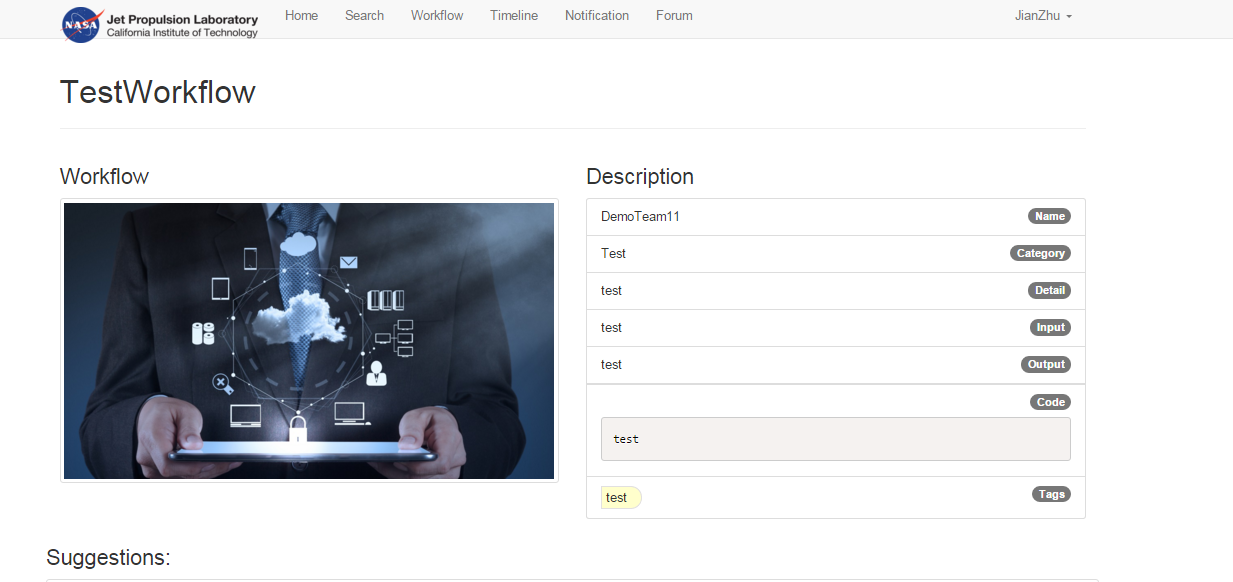


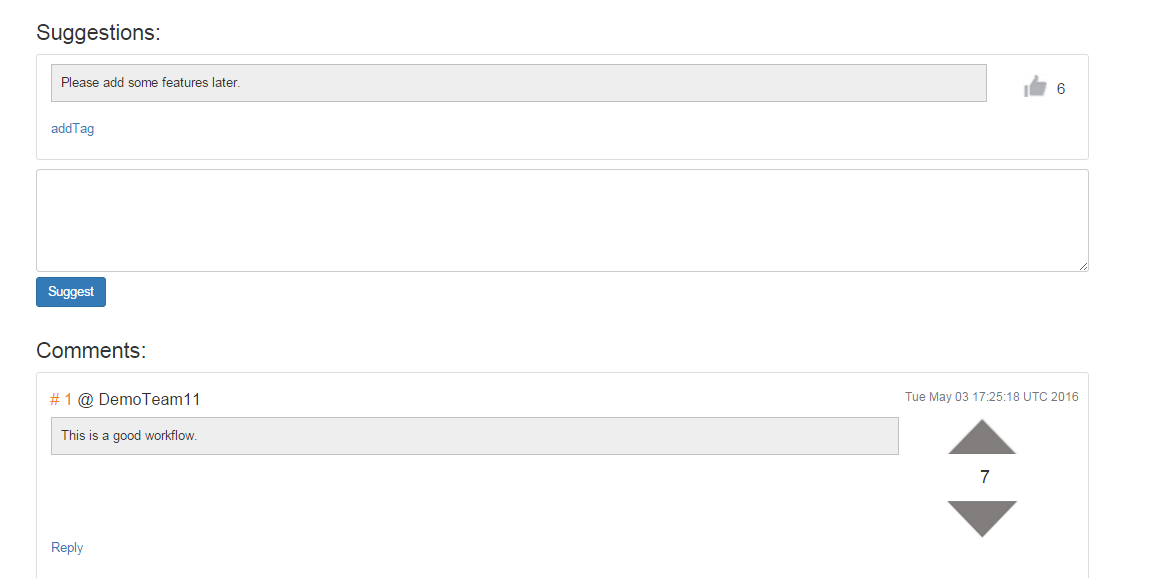
***7) Experiment – Switch a user and check if the data has been successfully updated in the backend server.***

Switch a user:



Check the workflow page created by the previous user:





**7. Conclusions and future work**

***Conclusions***

For this project, we have implemented 8 design patterns as follows:

* State pattern
* Iterator pattern
* Builder pattern
* Adapter pattern
* Composite pattern
* Façade pattern
* Abstract factory pattern
* Template pattern

***Future work***

To improve the application in future work, there are three possible ways:

* Implement filter pattern for searching function.
* User can mention their friends in comment and suggestion area with “@”.
* User can mention a specific workflow in comment and suggestion area with hashtag.
* Users can effectively and efficiently manage the workflows that they have posted.