

**Computer Science and Engineering**

**ItWorks**

**Software Design Description(SDD)**

**Version 2.0**

Document Number: SDD-002

Project Team Number: B28

Project Team Members:

|  |  |
| --- | --- |
| Rong Fang | rsf288 |
| Tian Lin | tcl344 |
| Lisa Frankel | lhf238 |

**REVIEW AND APPROVALS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Printed Name and Title** | **Function (Author, Reviewer, Approval)** | **Date** | **Signature** |
| Tian Lin | Author | April 25, 2018 |  |
| Rong Fang | Author | April 25, 2018 |  |
| Lisa Frankel | Author | April 25, 2018 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**REVISION LEVEL**

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision Number** | **Purpose** |
| March 7, 2018 | Version 1.0 | Initial Release |
| April 25, 2018 | Version 2.0 | Final Release |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Contents**

1. **INTRODUCTION……………………………………………………………………………….….……5**
   1. PURPOSE………………………………………………………………………….………….…..5
   2. SCOPE……………..…………………….………………………..………………..……………5
   3. IDENTIFICATION …………………….……………………………. ……………..……….......5
   4. DOCUMENT SUMMARY.…………………………………………….……………...………....5
   5. SYSTEM OVERVIEW.………………………………………………..………………..……….5
   6. DOCUMENT OVERVIEW…………………………………………..………………..………….5
2. **REFERENCE DOCUMENTS**.……………………………..……………………………..….…….…..6
3. **SYSTEM WIDE DESIGN DECISIONS……………………………………………..…………………6**
   1. SOFTWARE COMPONENT ARCHICTETURAL DESIGN…………………………………6
   2. SOFTWARE ARCHITECTURE GENERAL DESCRIPTION…………………………………6
   3. SOFTWARE ITEM COMPONENTS……………………………………………………………6
   4. COMPONENT INTERFACE IDENTIFICATION………………………………………………7
   5. SOFTWARE COMPONENT CONCEPT OF ECXCUTION…………………………………7
4. **SOFTWARE ITEM DETAILED DESIGN……………………………………………………………7**
   1. STRUCTURE………………………………………………………………………………...…...7
      1. *Software Unit Detailed Design*………………………………………………………7
   2. STATIC RELATIONSHIP OF SOFTWARE UNIT……………………………………………8
      1. *Run-time Object Instance*……………………………………………………………..8
   3. BEHAVIOR.…………………………………………………………………………….…..…....8
      1. *Sequence Interaction Diagrams*………………………………………………*…*……8
      2. *Collaboration Diagrams…*…………………………………………………………*…14*
      3. *Activity Diagrams…*………………………………………………………………*…14*
   4. CONCEPT OF EXECUTION ………………………………………………..….…………….19
   5. INTERFACE DESIGN…………………………………………………………………………20
      1. *Interface Identification and Diagrams*………………………………………………20
      2. *Unique Identifier of Interface*…………………………………………………………20
5. **IMPLEMENTATION ARCHITECTURE(NOT REQUIRED)…………………………………...…..20**
   1. ALL ACTIVE AND PASSIVE CLASSES ASSIGNED TO COMPONENTS………………20
   2. DIAGRAMS OF PHYSICAL PACKAGING OF LOGICAL COMPONENTS………………20
6. **DEPLOYMENT ARCHITECTURE…………………………………………………………………..21**
   1. PHYSICAL DEPLOYMENT ARCHITECTURE DIAGRAM………………………………....21
7. **DICTIONARIES………………………………………………………………………………………..21**
8. **SOFTWARE ITEM COMPUTER RESOURCE UTILIZATION……………………………………21**
9. **REQUIREMENT TRACEABILITY…………………………………………………………………...22**
   1. SOFTWARE COMPONENT-LEVEL REQUIREMENTS TRACEABILITY………………...22
10. **SYSTEM DEISGIN TESTING……..……………..……………..……………..……………………22**
11. **RATIONALE……..……………..……………..……………..…………..……………..……………22**
12. **NOTES…………..……………..……………..……………..……………..……………..…………..22**
13. **APPENDICES…………….………………………………………….…………..…………………..22**
    1. DICTIONARIES…………………………………………………………………………..22
    2. UML DIAGRAMS, IF NOT INCLUDED IN THE BODY OF THE DOCUMENT…….26
    3. SCHEDULE TRACKING………………………………………………………………….26
    4. DEFECT TRACKING…………………………………………………………………….29
14. **INTRODUCTION**
    1. **PURPOSE**

The purpose of this document is to define the contents of the software design, system architecture and detailed design of the application. The document communicates overall quantitative and qualitative system characteristics to SQA team, operations management, technical support, training, and operators.

* 1. **SCOPE**

This document will describe software-only products, and will not describe hardware products.

* 1. **IDENTIFICATION**

Software Design Document: Version 2.0, Document Number: SDD – 002

* 1. **DOCUMENTATION**

The purpose of this SDD is to provide a detailed description of the software design of ItWorks. This SDD will explain the purpose, as well as scope, system wide design decisions, software item detailed design, implementation of the architecture, deployment of the architecture, and software item computer resource utilization for the project.

The intended audience of this SDD is the developers of the system, project management, corporate executives, and the SQA team.

* 1. **DOCUMENT SUMMARY**

ItWorks is a mobile/web based application that values user input. When conventional solutions don’t work, there is no good way to seek alternative solutions. Users can post advice and search for answers. ItWorks will not only enable its users to look for answers, but also use a feedback system that lets users know if the answers they find are effective. Additionally, there will be a reputation system, in which other users vote on whether a piece of advice is useful or not. This makes the application more engaging.

It is a crowdsourcing platform that allows users to access tried-and-tested, yet unconventional methods of solving problems that come about in day-to-day life.

* 1. **SYSTEM OVERVIEW**
* Section 2 contains reference documents
* Section 3 goes over the system wide design decisions
* Section 4 covers the software item detailed design
* Section 5 covers the implementation architecture
* Section 6 is the deployment architecture
* Section 7 references section 13.1 with our dictionaries
* Section 8 is the software item computer resource utilization
* Section 9 explains the requirements traceability
* Section 10 explains the plan for system design testing
* Sections 11, 12, and 13 are the rationale, notes, and appendices

1. **REFERENCE DOCUMENTS**

Team A28, ItWorks Project Proposal, V 1.0, 9/21/2017

Team A28, ItWorks Initial SRS, V 001, 10/17/2017

Team A28, ItWorks Final SRS, V 002, 10/28/2017

Team A28, ItWorks SPMP, V 1.0, 10/14/2017

Team A28, ItWorks SAS, V 1.0, 11/21/2017

Team B28, ItWorks Project Proposal V 2.0, 2/7/2018

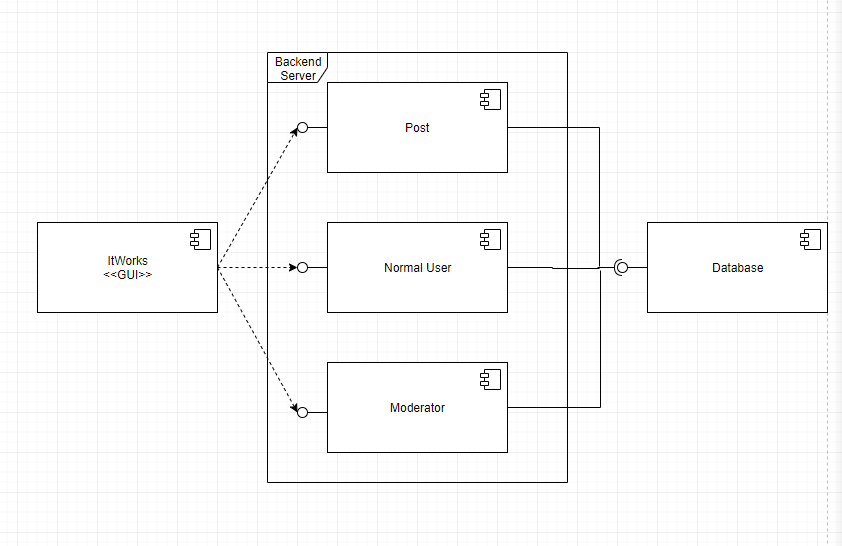
Team B28, ItWorks RAS, V 1.0,  2/10/2018

Team B28, ItWorks SPMP V 2.0, 2/15/2018

Team B28, ItWorks SDD 1.0, 3/11/2018

1. **SYSTEM WIDE DESIGN DECISIONS**
   1. **SOFTWARE COMPONENT ARCHITECTURAL DESIGN**

*Component Architecture Diagram*



* 1. **SOFTWARE ARCHITECTURE GENERAL DESCRIPTION**

The client using the GUI connects to the ItWorks server, with which the client may send and receive data. The ItWorks server connects to the database, with which the server may send and receive data.

* 1. **SOFTWARE ITEM COMPONENTS**

ItWorks GUI – The website interface that the client/users interacts with.

Server - Backend server which interacts with database and the front end GUI

Post – An object created with content from user that will be stored on database.

Normal User – a default user of application with base level permissions

Moderator – a user of application that can perform administrative functions such as removing posts from other users.

Database – storage for user and post data

* 1. **COMPONENT INTERFACE IDENTIFICATION**

The interface from the GUI to the backend server is called GuiToBackend.

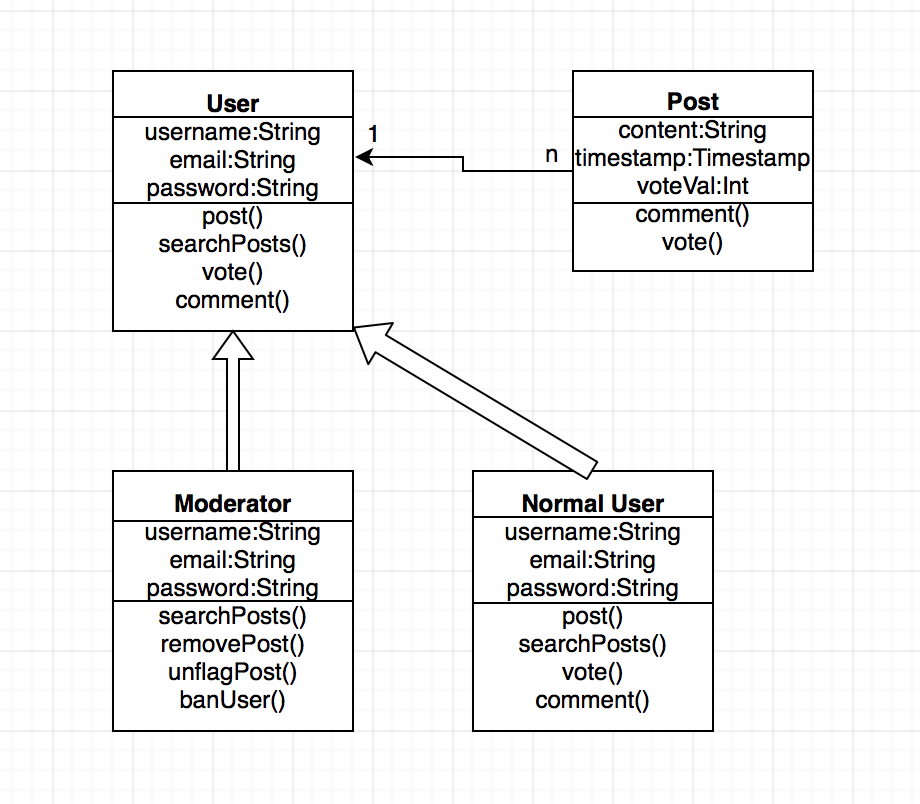
The interface from the backend server to the database is called BackendToDatabase.

* 1. **SOFTWARE COMPONENT CONCEPT OF EXECUTION**

The server will be launched through the AWS platform. User will access the frontend GUI by entering the website domain in their computer device.

1. **SOFTWARE ITEM DETAILED DESIGN**
   1. **STRUCTURE**
      1. ***Software Unit Detailed Design***

*Class Diagram*



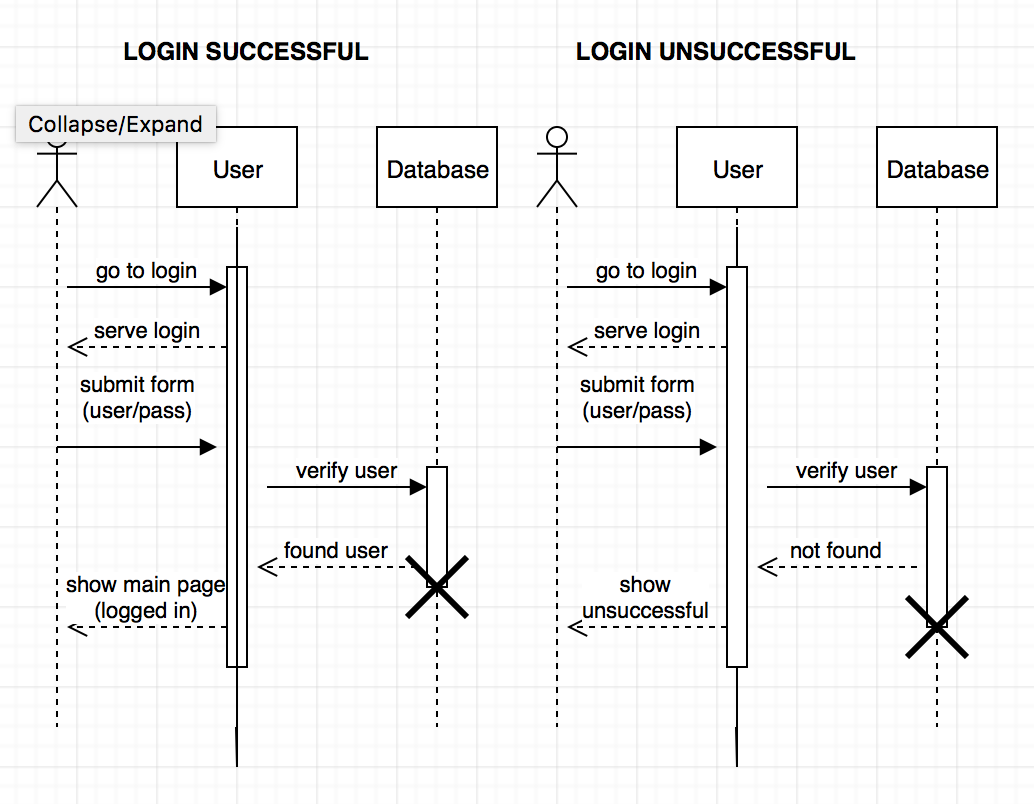
* 1. **STATIC RELATIONSHIP OF SOFTWARE UNIT**
     1. ***Run-time Object Instances***

Everytime when a user logs in, a new thread which contains the user object is created for as long as he is logged in. Users can only create one post at a time. The thread responsible for the user will create the post object and store it in the database. The user object will terminate along with its thread when user logs out.

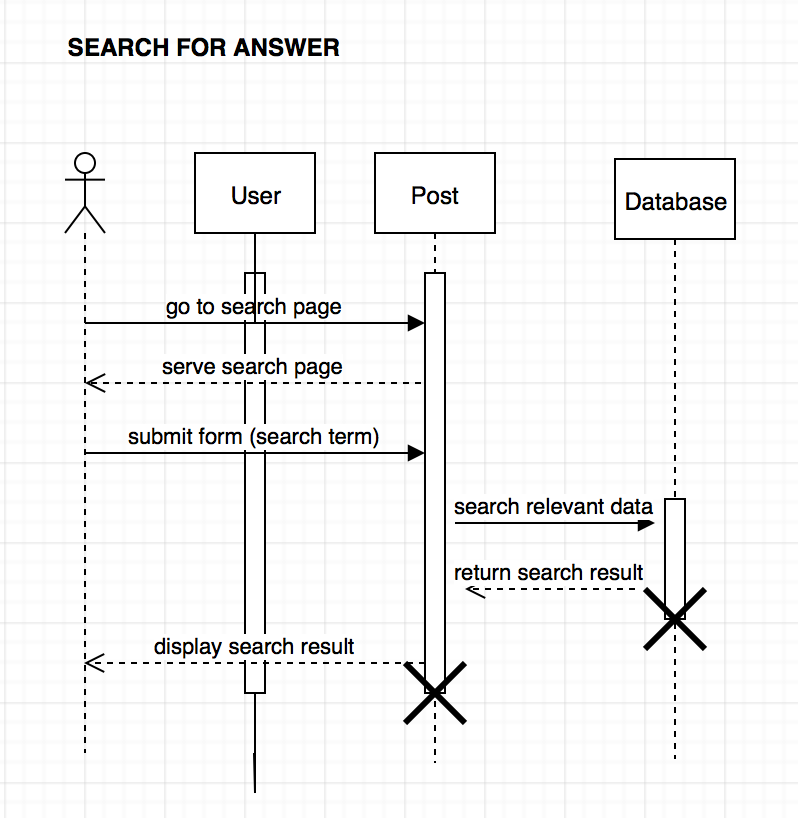
A post object will be created when a post is being fetched and displayed. The object stays for as long as the user stays on the page.

* 1. **BEHAVIOR**
     1. ***Sequence Interaction Diagrams***

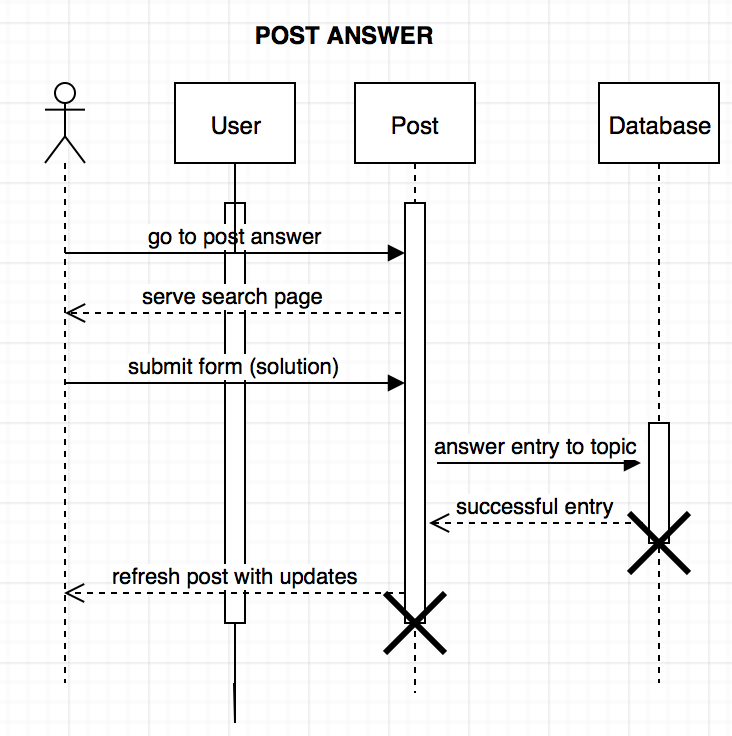
*Login Sequence (successful and unsuccessful)*



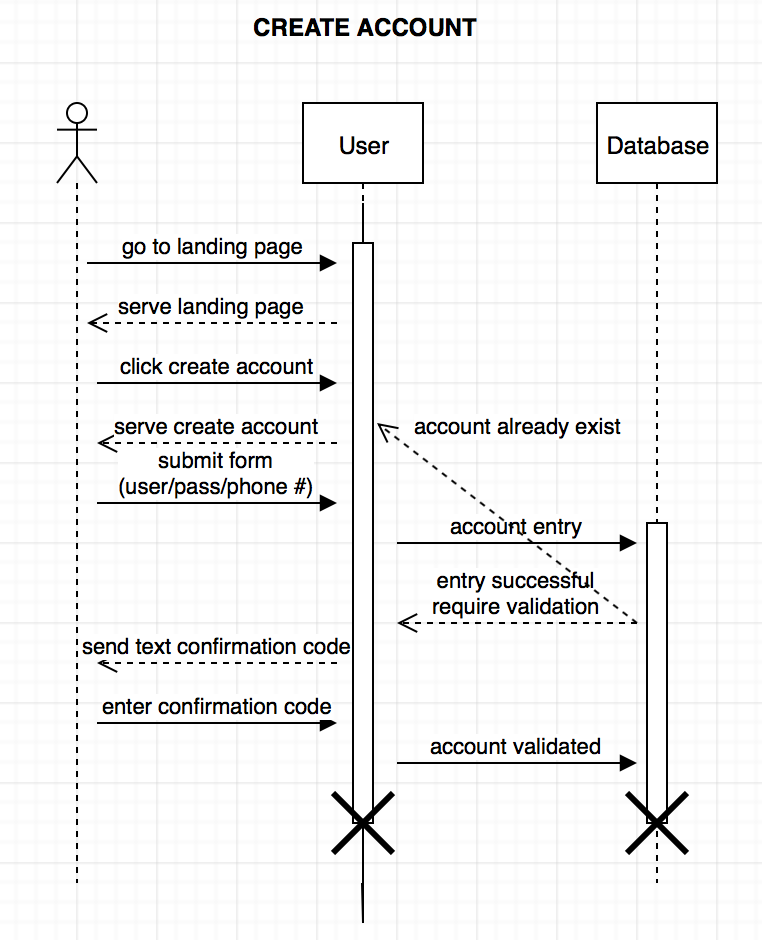
*Search Sequence*

**

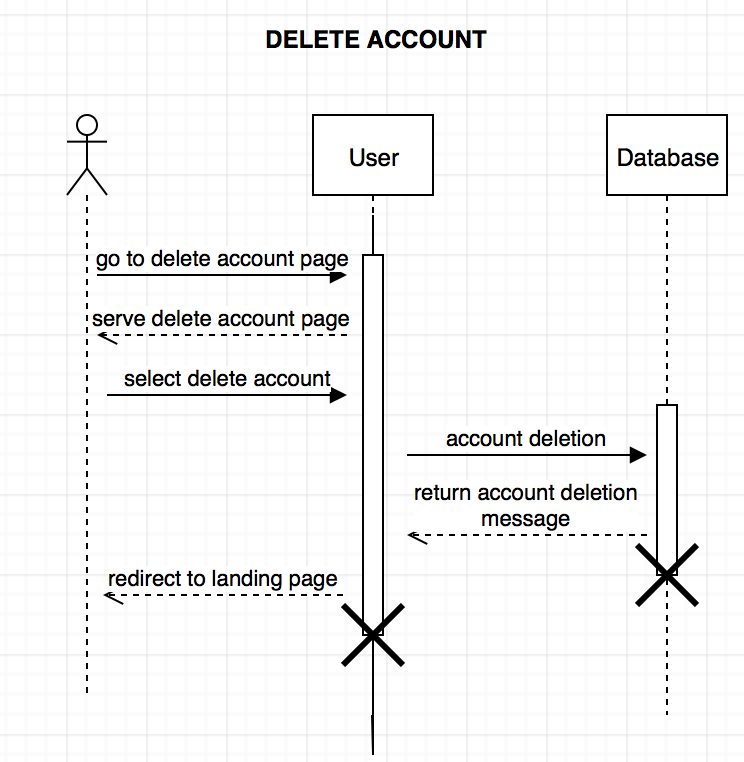
*Post Sequence*

**

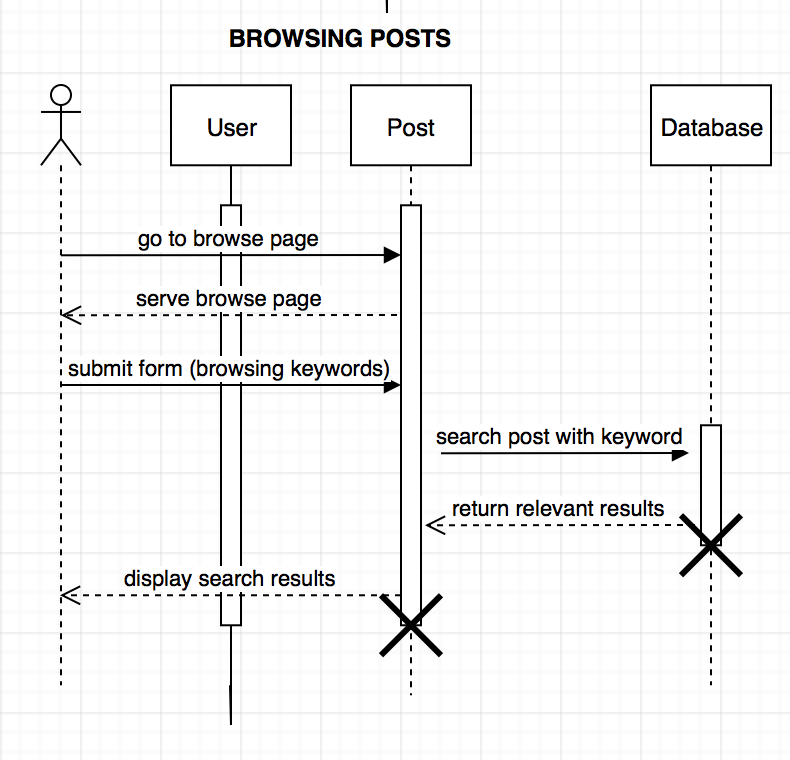
*Create Account Sequence*

**

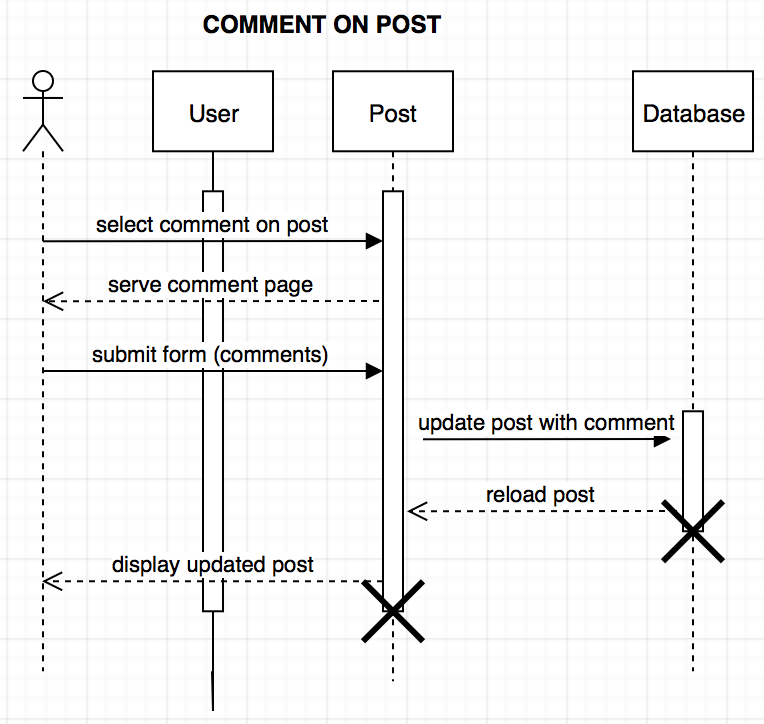
*Delete Account Sequence*

**

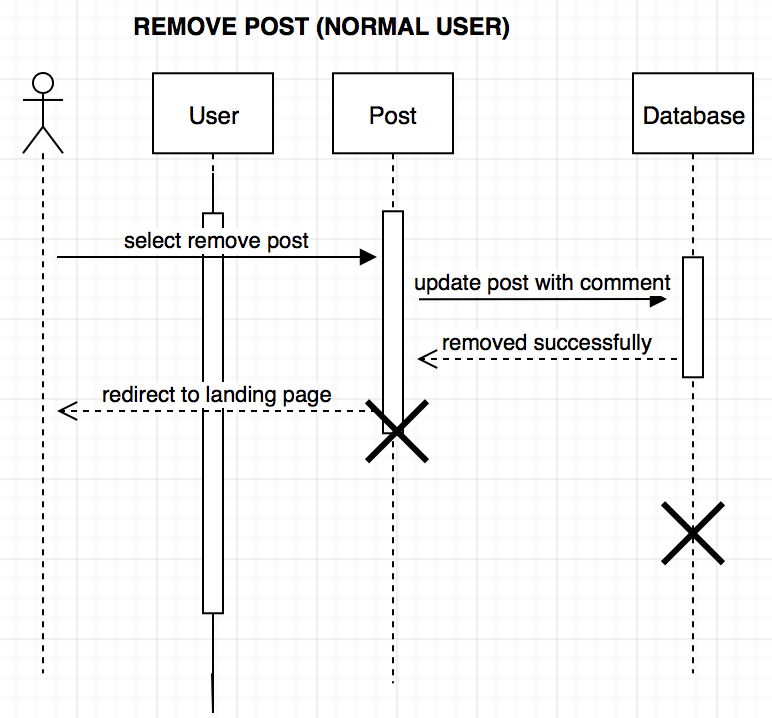
*Browse Sequence*

**

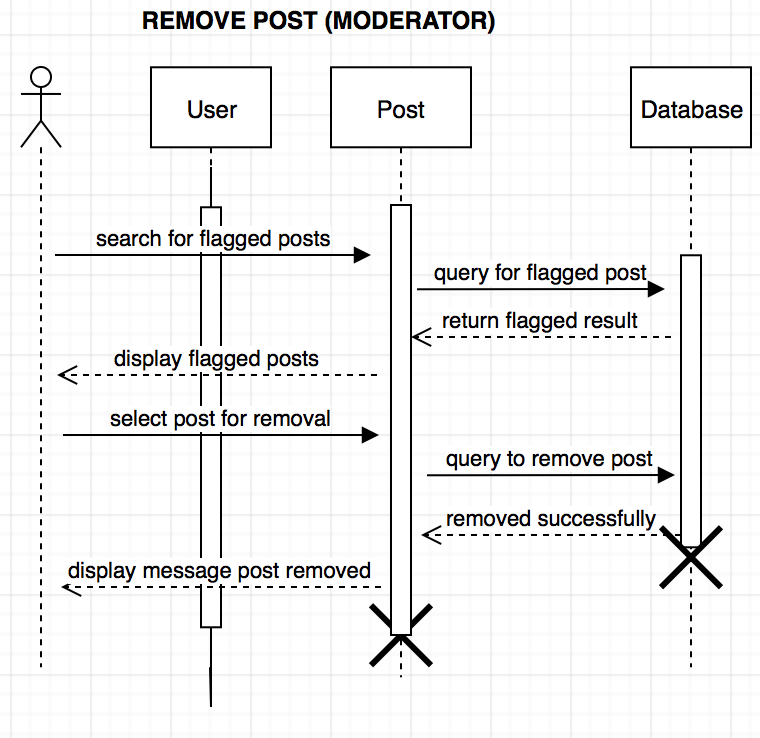
*Comment Sequence*

**

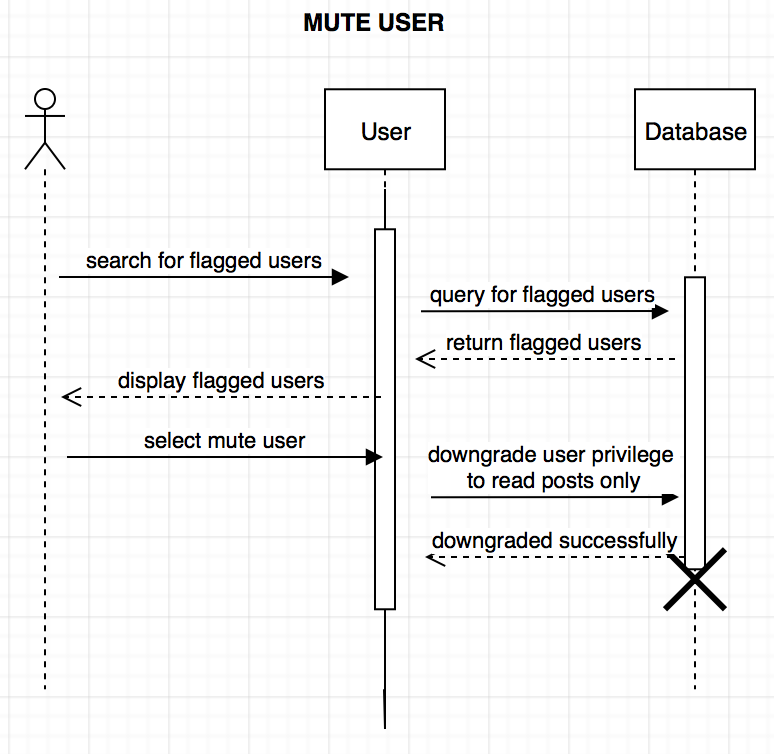
*Remove Post (Normal User) Sequence*

**

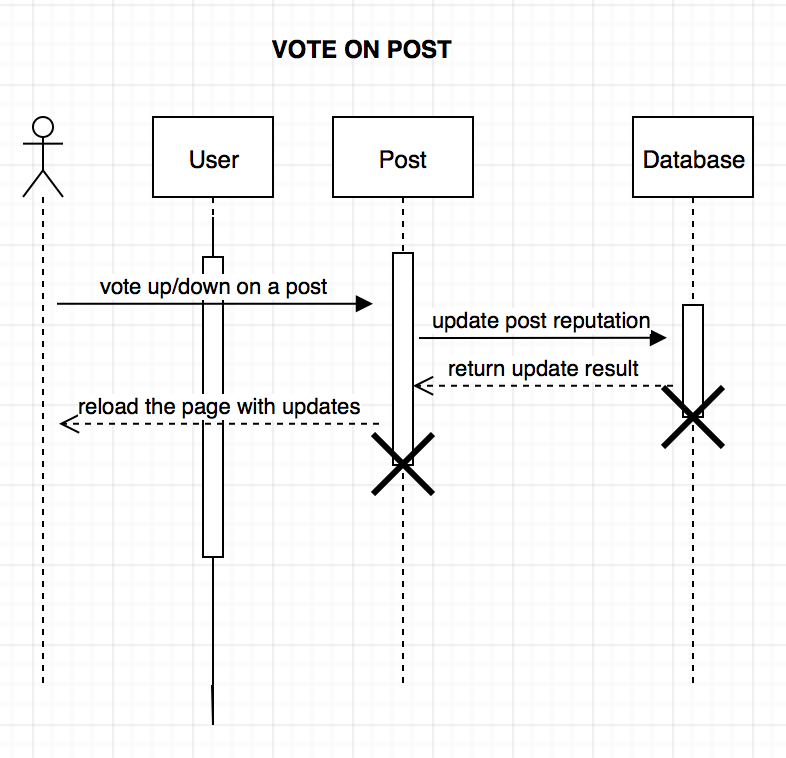
*Remove Post (Moderator) Sequence*

**

*Mute User Sequence*

**

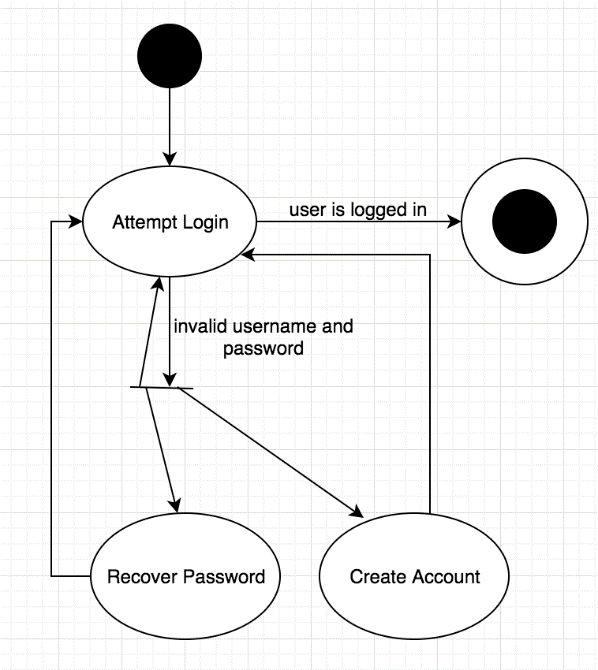
*Vote Sequence*

**

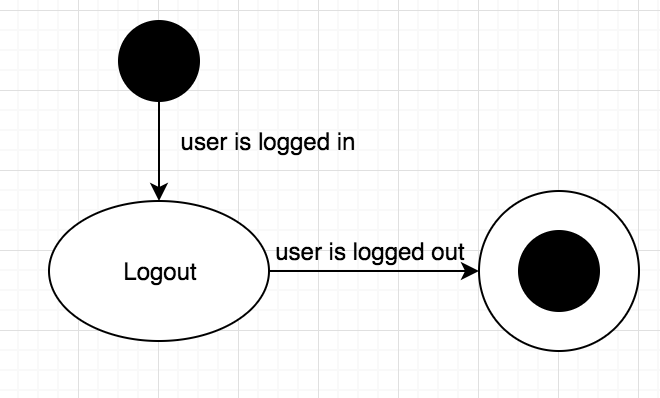
* + 1. ***Collaboration Diagrams***
    2. ***Activity Diagrams***

A scenario (work flow) diagram will be provided for each scenario. The scenarios are:

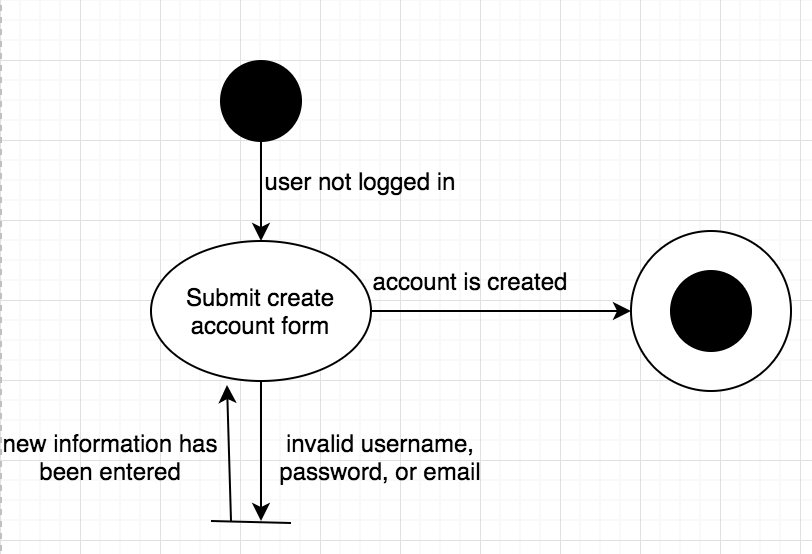
Login



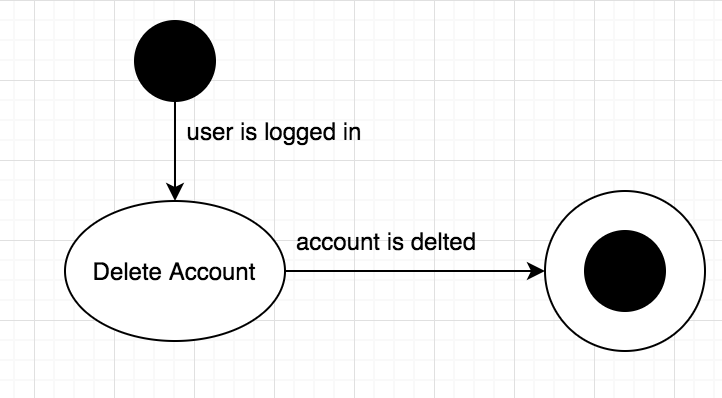
Logout



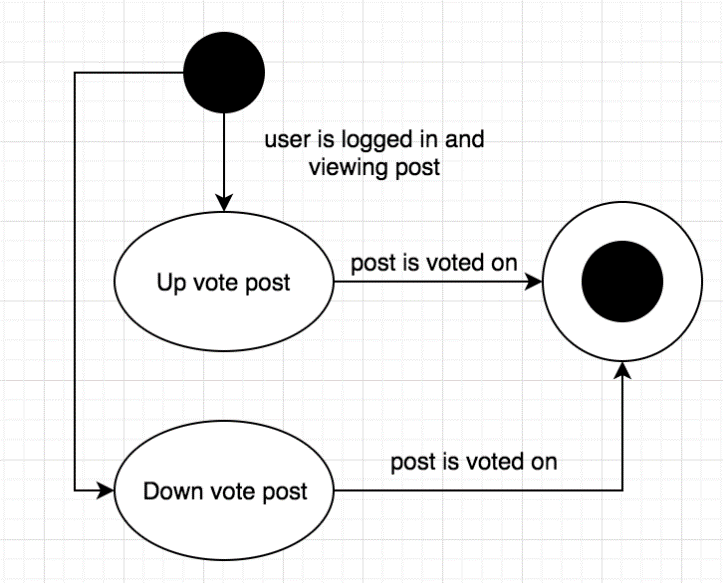
Create Account



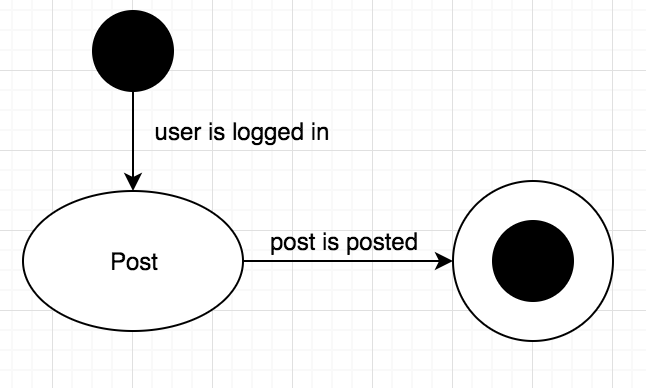
Delete Account



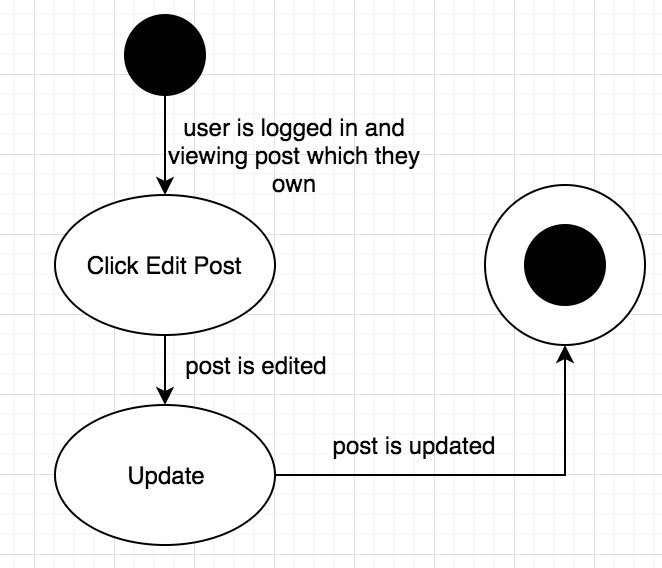
Vote on Post



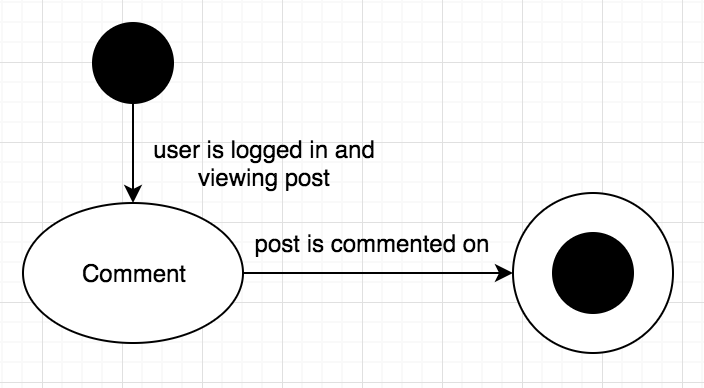
Create Post



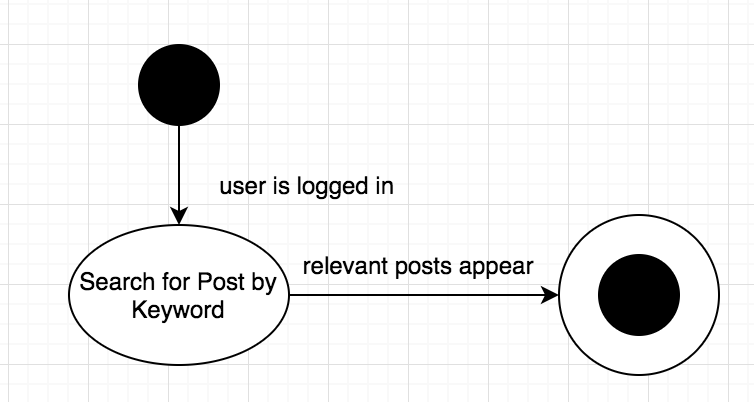
Edit Post



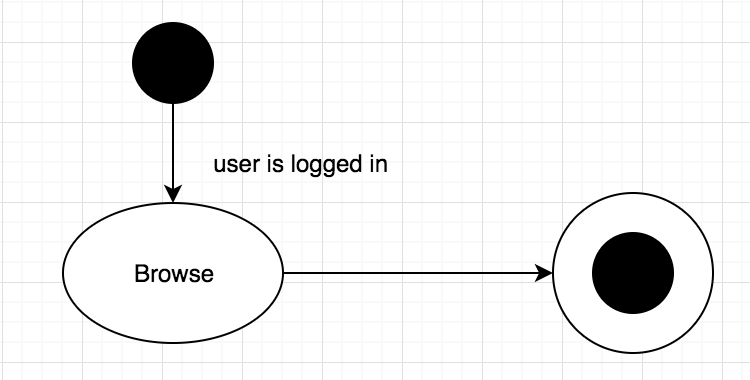
Comment



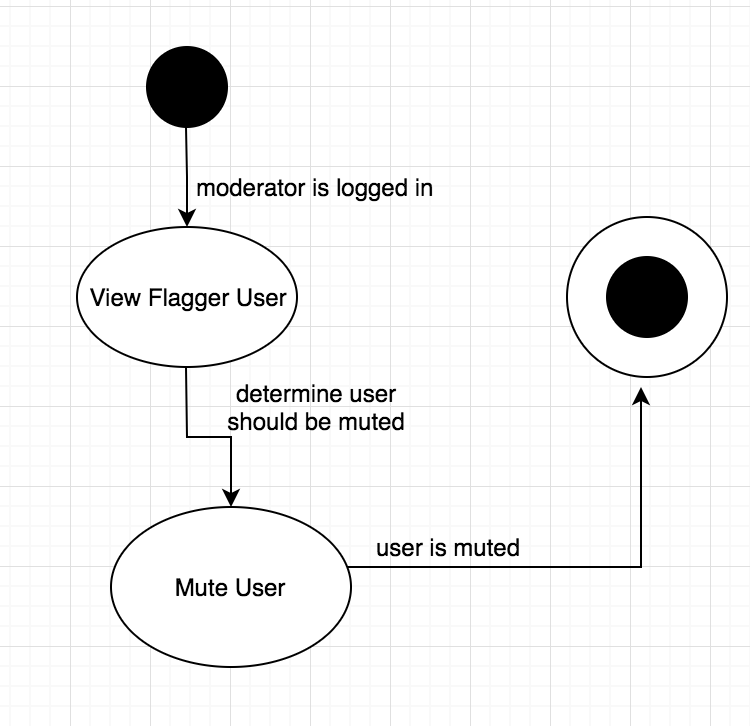
Search



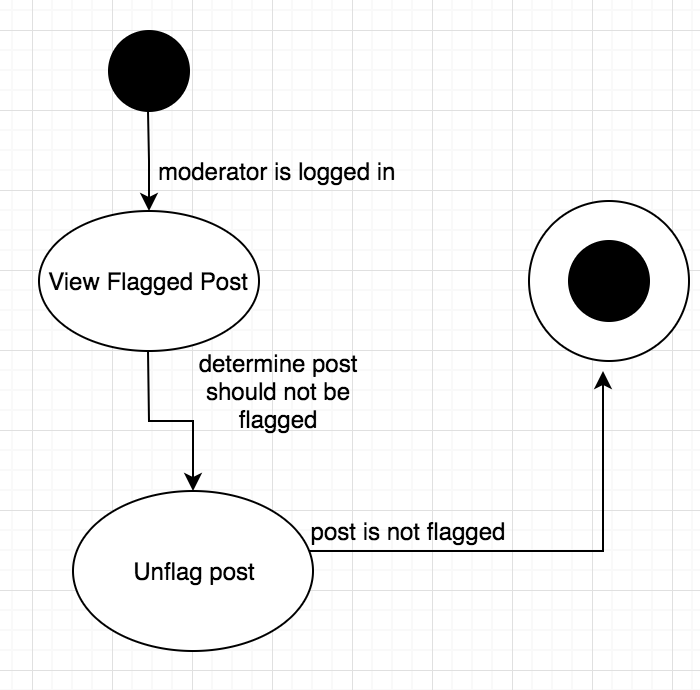
Browse



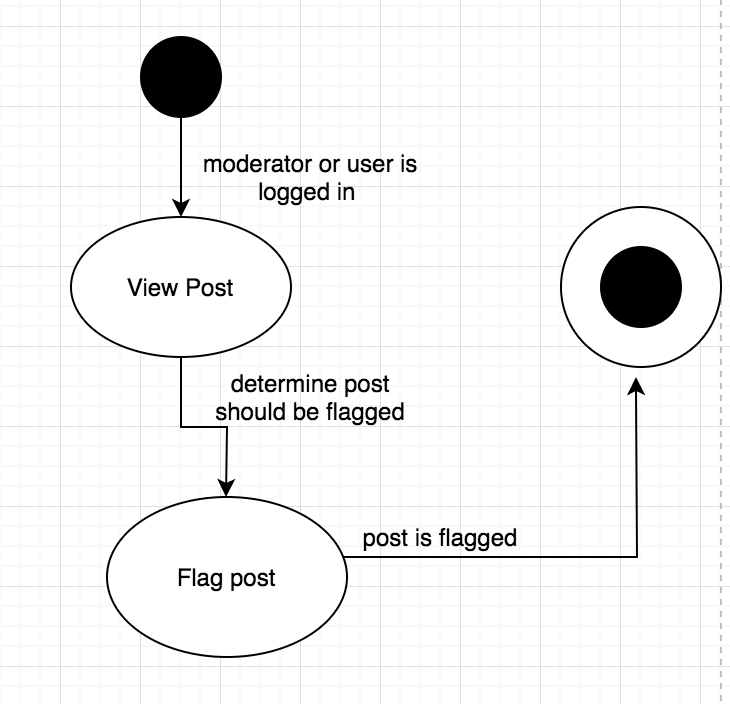
Mute User



Flag post



Unflag post



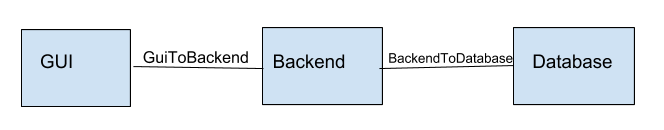
* 1. **CONCEPT OF EXECUTION**

Database object will be the first object to be initiated by the frontend and will remain active as long as the server is online.

Users will access the GUI through entering the ItWorks website on their browser. The GUI will query the backend server and instantiates a user object when the user logs in, the user object will communicate with the DB object and forward the SQL query to the database to conduct logins/logouts. The user object terminates when the user logs out.

The frontend(GUI) will query the backend server and instantiates a post object when it detects the user object which attempts the initiation exists and is valid. The post object will be created only when the user object attempts to create a new post or retrieve an existing post. The user object is able to update the post by commenting/voting on it. The post object will terminate when there are no users interacting with the same post object.

all user and post objects will be managed by individual threads and will be executed concurrently.

* 1. **INTERFACE DESIGN**
     1. ***Interface Identification and Diagrams***
     2. ***Unique Identifier of Interface***

Name: GuiToBackend

Type of interface: golang cgo wrapper reads C++ code with extern C linkage

Data type: mix of alphanumeric

Unit of measurement: # of instance of objects

Security constraint: encapsulation (OOP)

Sources: Backend

Recipients: GUI

Name: BackendToDatabase

Type of interface: C++ to database connection

Data type: mix of alphanumeric

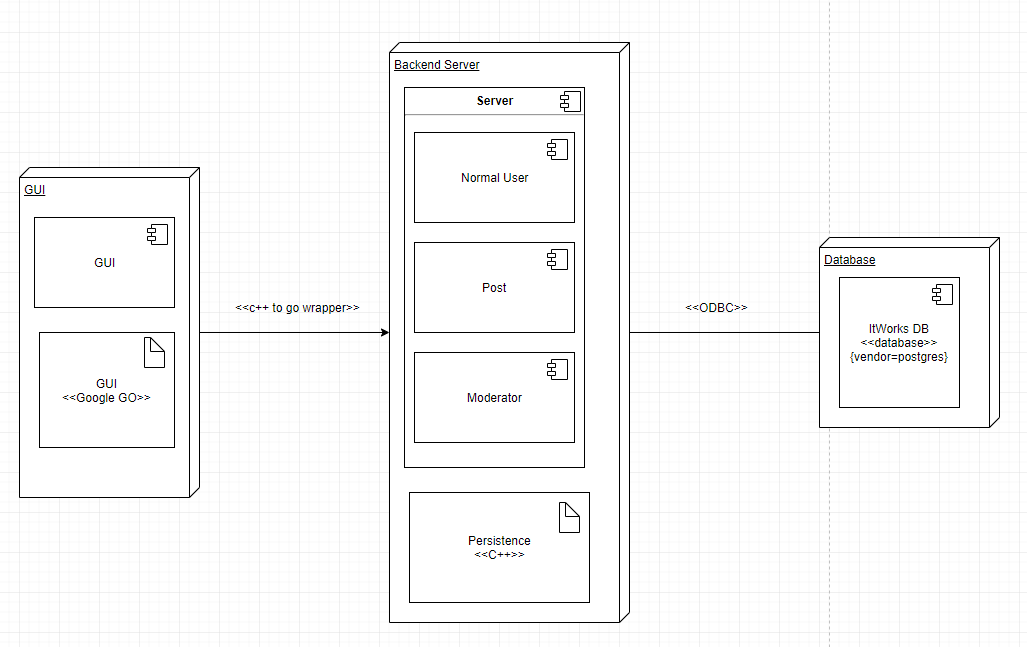
Unit of measurement: # of connections / # of queries

Security constraint: encapsulation (OOP)

Source: Database

Recipient: Backend

1. **IMPLEMENTATION ARCHITECTURE(NOT REQUIRED)**
   1. **All ACTIVE AND PASSIVE CLASSES ASSIGNED TO COMPONENTS**
   2. **DIAGRAMS OF PHYSICAL PACKAGING OF LOGICAL COMPONENTS**
2. **DEPLOYMENT ARCHITECTURE**
   1. **PHYSICAL DEPLOYMENT ARCHITECTURE DIAGRAM**



1. **DICTIONARIES**

The dictionary can be found in section 13.1 of the SDD.

1. **SOFTWARE ITEM COMPUTER RESOURCE UTILIZATION**

GUI-Backend Server resource usage

Typical usage: 100-300 simultaneous web users

Worst case usage: 1000+ simultaneous web users

Special considerations: each user connected will generate exactly 1 fixed size user object and minimum 0 and at most N post objects. Each post object will contain exactly one comment of variable size string and will be stored in a collection of posts.

Units of measurement: # of Instances of objects

Database resource Usage

Typical usage: 150 character per string on average and 1000 retrievals of post data per minute

Worst case usage: exceeding ⅔ of the SSD maximum read speed

Special considerations: User queries consume very little resources. Post queries can be very large string size which will require some data transfer time per post object.

Units of measurement: # of queries/minute, total size data transfer per post

1. **REQUIREMENT TRACEABILITY**
   1. **SOFTWARE COMPONENT-LEVEL REQUIREMENTS TRACEABILITY**

Each software component-level requirement will be traceable forwards and backwards. In software design documents, there will be section numbers so that it can be traced in the future. When we code any object we will have in the header, the requirement that it supports, so that it can be traced back to the requirements.

1. **SYSTEM DESIGN TESTING**

The development team will generate inputs to test the system. These inputs will consist of users doing certain actions that will test our software. We have no required simulators. Please refer to section 4 to see the software units, behavior, and interactions that will be tested. There will be various stages of design testing done: self checks, peer review, walkthroughs and inspections.

1. **RATIONALE**
2. **NOTES**
3. **APPENDICES**
   1. **DICTIONARIES**

*Classes Dictionary Table*

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Description | Methods | Attributes |
| User | Abstract class for a user of the system | searchPosts()  post()  vote()  comment() | username:String  email:String  password:String |
| Moderator | Subclass of user, moderator is a user of the system with special privileges | searchPosts()  removePost()  unflagPost()  banUser() | username:String  email:String  password:String |
| Normal User | Subclass of user, normal user is a standard user of the system | searchPosts()  post()  vote()  comment() | username:String  email:String  password:String |
| Post | Class for a post | comment()  vote() | content:String  timestamp:Timestamp  voteVal:int |

*Methods Dictionary Table*

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Class | Arguments |
| searchPosts() | Allows user to search through posts in the system that match a keyword or phrase | User, Moderator, Regular User | query:String |
| post() | Allows a user to post content to the site | User, Regular User | content:String  username:String |
| vote() | Allows a user to either up vote or down vote a post | User, Regular User | UpOrDown:bool  post:Post |
| comment() | Allows a user to comment on a post | User, Regular User | comment:String  post:Post |
| removePost() | Allows a moderator to remove a post | Moderator | post:Post |
| unflagPost() | Allows a moderator to unflag a post | Moderator | post:Post |
| banUser() | Allows a moderator to ban a user | Moderator | user:User |

*Attribute Dictionary Table*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Description | Simple/Complex | Type | Size | R/W |
| email | User’s email address | Simple | string | variable |  |
| password | User’s password | Simple | string | variable |  |
| username | User’s username | Simple | string | variable |  |
| content | Post content | Simple | string | variable |  |
| timestamp | A timestamp for a post | Simple | Timestamp | fixed |  |
| voteVal | A post’s score (based on votes) | Simple | int | variable |  |

*Relationship Dictionary Table*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Description | From Class | To class | Optional/Mandatory | Cardinality |
| OwnedBy | Each post is owned by a user | User | Post | Mandatory | A user can have many posts, and a post must have 1 and only 1 owner(user) |

*Key Events Dictionary Table*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Description | Motive | Action | Pre-Conditions | Post-Conditions | State Change |
| Login | user logs into their account | login | Go to login page  Enter name and password  Submit form | User is not logged in | User is logged in | Logged In |
| Search | user searches for answer through the application | Search for answer | Go to search page  Input question into search bar on page  Submit form | User is logged in | User gets results | None |
| Post | user posts a solution to a problem | Post answer | Go to post page  Input solution into post form on page  Submit form | User is logged in | User’s post is added to site | None |
| Create Account | User creates an account | Create an account | Go to landing page  Click create account  Enter username, name, password, and phone number into form  Submit form  Enter text confirmation code | User does not have account | User has an account | Redirected to login page |
| Delete Account | User deletes account and all associated data | Delete an account | Go to delete account page  Select delete account button  Submit form | User has an account | User deletes their account | Redirected to login page |
| Browse | Search through all existing posts | Browse through posts | Go to browse page  Enter browse keywords in form  Submit form | User is logged in | Content appears | None |
| Comment | User comments on a post | Comment on a post | Select “comment on post”  Input comment string  Submit form | User is logged in | User’s comment appears under post | None |
| Remove Post | Remove a post from the site | Remove a post | Select remove this post | User is logged in and post they want to remove is their own OR user is a moderator | Post is removed from the site | None |
| Mute Account | Users with frequent flagged posts will only be allowed to read posts | Mute an account | Look for flagged users  Select the flagged user  Edit account privilege to the applied user account to read post only | User has been flagged | User can no longer post | None |
| Vote on post | User up-votes or down-votes post depending on how good they think the solution is | Vote on a post | Click up or down arrow next to post | User is logged in and viewing a post | User has altered the posts vote value | None |
| Maintain App | Developer maintains application, handles any errors and updates site | Maintain the application | Understand error or update  Implement bug fix or update  Push code | There is an update, error or bug. | New code is implemented | Dependent on code that is implemented |

* 1. **UML DIAGRAMS, IF NOT INCLUDED IN THE BODY OF THE DOCUMENT**

Identified in body documents.

* 1. **SCHEDULE TRACKING**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who | Estimated (hrs) | Actual (hrs) | Difference (hrs) |
| Initial SRS | Tian Lin | 12 | 10 | 2 less |
|  | Rong Fang | 12 | 10 | 2 less |
|  | Lisa Frankel | 12 | 9 | 3 less |
|  | Alex Huang | 12 hrs | 9 hrs | 3 less hrs |
|  | Summary for entire team | 48 | 38 | 10 less |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who | Estimated (hrs) | Actual (hrs) | Difference (hrs) |
| Final SRS | Tian Lin | 2 | 2 | 0 |
|  | Rong Fang | 2 | 3 | 1 more |
|  | Lisa Frankel | 2 | 3 | 1 more |
|  | Alex Huang | 2 hrs | 0 hrs | 2 less hrs |
|  | Summary for entire team | 8 | 8 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Who | Estimated (hrs) | Actual (hrs) | Difference (hrs) |
| SPMP | Tian Lin | 12 | 11 | 1 less |
|  | Rong Fang | 12 | 10 | 2 less |
|  | Lisa Frankel | 12 | 10 | 2 less |
|  | Alex Huang | 12 | 9 | 3 less |
|  | Summary for entire team | 48 | 40 | 8 less |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who | Estimated (hrs) | Actual (hrs) | Difference (hrs) |
| SAS | Tian Lin | 12 hrs | 10 hrs | 2 less hrs |
|  | Rong Fang | 12 hrs | 10 hrs | 2 less hrs |
|  | Alex Huang | 12 hrs | 10 hrs | 2 less hrs |
|  | Lisa Frankel | 12 hrs | 10 hrs | 2 less hrs |
|  | Summary for entire team | 48 hrs | 40 hrs | 8 hrs less |
| Artifact or  Deliverable | Who | Estimated (hrs) | Actual (hrs) | Difference (hrs) |
| Initial RAS | Tian Lin | 4 hrs | 2 hrs | 2 less hrs |
|  | Rong Fang | 4 hrs | 2 hrs | 2 less hrs |
|  | Lisa Frankel | 4 hrs | 2 hrs | 2 less hrs |
|  | Summary for entire team | 12 hrs | 6 hrs | 6 less hrs |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who | Estimated (hrs) | Actual (hrs) | Difference (hrs) |
| SPMP 2.0 | Tian Lin | 4 hrs | 3 hrs | 1 less hrs |
|  | Rong Fang | 4 hrs | 3 hrs | 1 less hrs |
|  | Lisa Frankel | 4 hrs | 3 hrs | 1 less hrs |
|  | Summary for entire team | 12 hrs | 9 hrs | 9 less hrs |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who | Estimated (hrs) | Actual (hrs) | Difference (hrs) |
| SDD 1.0 | Tian Lin | 8 hrs | 6 hrs | 2 less hrs |
|  | Rong Fang | 8 hrs | 6 hrs | 2 less hrs |
|  | Lisa Frankel | 8 hrs | 6 hrs | 2 less hrs |
|  | Summary for entire team | 24 hrs | 18 hrs | 6 less hrs |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who | Estimated (hrs) | Actual (hrs) | Difference (hrs) |
| SDD 2.0 | Tian Lin | 15 hrs | 16 hrs | 1 more |
|  | Rong Fang | 15 hrs | 17 hrs | 2 more |
|  | Lisa Frankel | 15 hrs | 17 hrs | 2 more |
|  | Summary for entire team | 45 hrs | 50 hrs | 5 more |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| Who | Estimated (hrs) | Actual (hrs) | Difference (hrs) |
| Tian | 69 | 59 | 10 less |
| Rong | 69 | 61 | 8 less |
| Lisa | 69 | 61 | 8 less |
| Alex | 38 | 28 | 10 less |
| Total | 245 | 209 | 36 less |

* 1. **DEFECT TRACKING**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who | Estimated (# of defects found) | Actual (# of defects found) | Difference (# of defects found) |
| Initial SRS | Tian Lin | 9 | 2 | 7 |
|  | Rong Fang | 9 | 1 | 8 |
|  | Lisa Frankel | 9 | 1 | 8 |
|  | Alex Huang | 9 | 2 | 7 |
|  | Summary for entire team | 36 | 6 | 30 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who | Estimated (# of defects found) | Actual (# of defects found) | Difference (# of defects found) |
| Final SRS | Tian Lin | 5 | 1 | 4 less |
|  | Rong Fang | 5 | 1 | 5 less |
|  | Lisa Frankel | 5 | 0 | 5 less |
|  | Alex Huang | 5 | 0 | 5 less |
|  | Summary for entire team | 20 | 2 | 28 less |
| Artifact or  Deliverable | Who | Estimated (# of defects found) | Actual (# of defects found) | Difference (# of defects found) |
| SPMP | Tian Lin | 9 | 7 | 2 less |
|  | Rong Fang | 9 | 5 | 4 less |
|  | Lisa Frankel | 9 | 3 | 6 less |
|  | Summary for entire team | 36 | 18 | 18 less |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Artifact or*  *Deliverable* | *Who* | *Estimated (# of defects found)* | *Actual (# of defects found)* | *Difference (# of defects found)* |
| *SAS* | *Tian Lin* | *9* | *3* | *6* |
|  | *Rong Fang* | *9* | *2* | *7* |
|  | *Alex Huang* | *9* | *1* | *8* |
|  | *Lisa Frankel* | *9* | *2* | *7* |
|  | *Summary for entire team* | *36* | *8* | *28* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who | Estimated (# of defects found) | Actual (# of defects found) | Difference (# of defects found) |
| RAS | Tian Lin | 5 | 4 | 1 |
|  | Rong Fang | 5 | 3 | 2 |
|  | Lisa Frankel | 5 | 1 | 4 |
|  | Summary for entire team | 15 | 8 | 7 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who | Estimated (# of defects found) | Actual (# of defects found) | Difference (# of defects found) |
| SPMP 2.0 | Tian Lin | 6 | 3 | 3 |
|  | Rong Fang | 6 | 3 | 3 |
|  | Lisa Frankel | 6 | 2 | 4 |
|  | Summary for entire team | 18 | 8 | 10 |

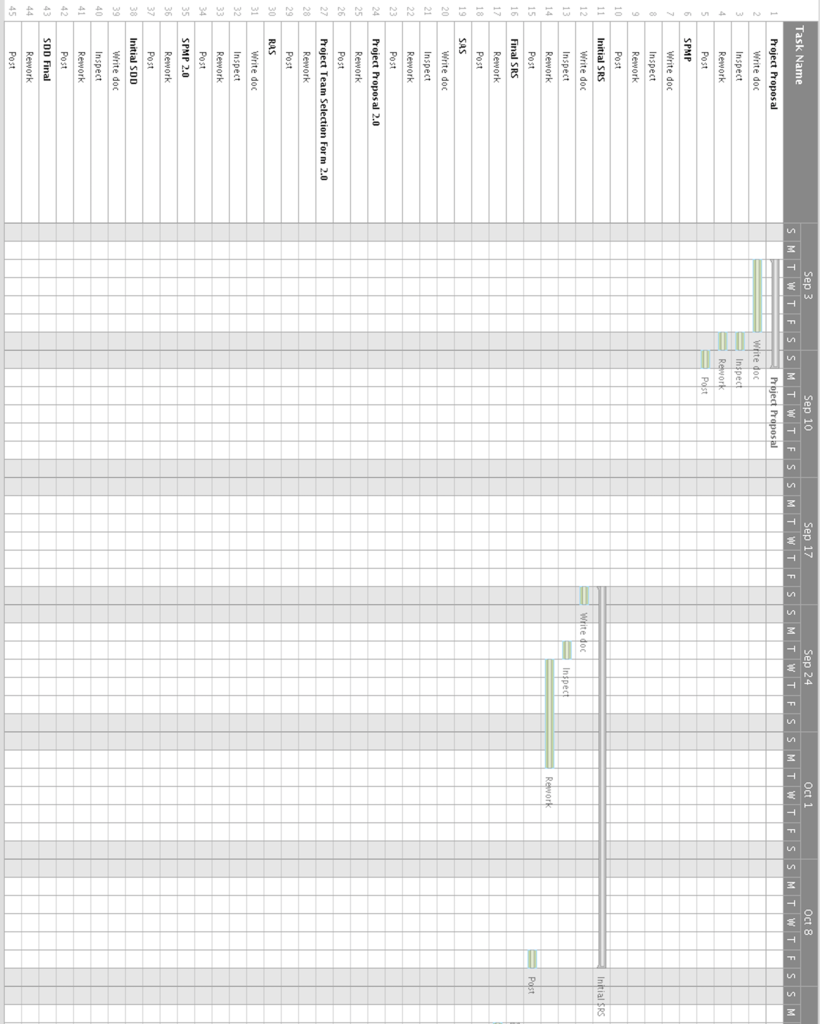
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who | Estimated (# of defects found) | Actual (# of defects found) | Difference (# of defects found) |
| SDD 1.0 | Tian Lin | 9 | 6 | 3 |
|  | Rong Fang | 9 | 5 | 4 |
|  | Lisa Frankel | 9 | 4 | 5 |
|  | Summary for entire team | 27 | 15 | 12 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or  Deliverable | Who | Estimated (# of defects found) | Actual (# of defects found) | Difference (# of defects found) |
| SDD 2.0 | Tian Lin | 6 | 4 | 2 |
|  | Rong Fang | 6 | 3 | 3 |
|  | Lisa Frankel | 6 | 3 | 3 |
|  | Summary for entire team | 18 | 10 | 8 |

***Cumulative***

|  |  |  |  |
| --- | --- | --- | --- |
| Who | Estimated (# of defects found) | Actual (# of defects found) | Difference (# of defects found) |
| Tian Lin | 58 | 30 | 28 less |
| Rong Fang | 58 | 23 | 35 less |
| Lisa Frankel | 58 | 15 | 43 less |
| Alex Huang | 32 | 7 | 25 less |
| Team Summary | 206 | 75 | 131 less |

**GANTT CHART**

****

**GANTT CHART**





