

**Computer Science and Engineering**

**Sharit**

**Software Design Description**

**Version 2.0**

Document Number: SDD-002

Team B6

Allen Zheng (az1010)

Hui Huang (hh1128)

Kenneth Liang (kl1792)

Warlon Zeng (wz634)

**REVIEW AND APPROVALS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Printed Name and Title** | **Function (Author, Reviewer, Approval)** | **Date** | **Signature** |
| Allen Zheng | Author | December 15, 2016 | Allen Zheng |
| Hui Huang | Author | December 15, 2016 | Hui Huang |
| Kenneth Liang | Author | December 15, 2016 | Kenneth Liang |
| Warlon Zeng | Author | December 15, 2016 | Warlon Zeng |
|  |  |  |  |
|  |  |  |  |

**REVISION LEVEL**

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision Number** | **Purpose** |
| October 25, 2016 | Version 1.0 | Initial Release |
| December 15, 2016 | Version 2.0 | Second Release |
|  |  |  |
|  |  |  |

**Table of Contents**

[1. INTRODUCTION](#_2fqumh7huito) x

[1.1 Purpose](#_j0u63spk9woj)

[1.2 Scope](#_hqw1g8mrxua4)

[1.3 Identification](#_d84rdhbsd2k)

[1.4 Document Summary](#_6bm2j58cwimy)

[1.5 System Overview](#_chxp8a7pr3lj)

[1.6 Document Overview](#_kvep8lejnxtt)

[2. REFERENCE DOCUMENTS](#_pghv6vumkzre) x

[3. SYSTEM WIDE DESIGN DECISIONS](#_da1x3vm2tryo) x

[3.1 Software Component Architectural Design](#_x9yk4xrwnbsw)

[3.2 Software Architecture General Description](#_ij2pfog3y9w6)

[3.3 Software Item Components](#_4b2astwtl57a)

[3.4 Component Interface Identification](#_qbsxhasxe0we)

[3.5 Software Component Concept of Execution](#_ltlxdkerl3iv)

[4. SOFTWARE ITEM DETAILED DESIGN](#_he9nosz5m8f0) x

[4.1 Structure](#_x2w2yt1z03gh)

[4.1.1 Software Unit Detailed Design](#_v9j0mrr32bkj)

[4.2 Static Relationship of Software Unit](#_pruo3ldu9nhc)

[4.2.1 Runtime Object Instances](#_gkpsg32yl8b5)

[4.3 Behavior](#_bj5nukkd5rqf)

[4.3.1 Interaction Diagrams](#_jnmve3vnq3kz)

[4.3.2 Collaboration Diagrams](#_9z4m3u3k79m9)

[4.3.3 Activity Diagrams](#_r9ti47vkmn5t)

[4.4 Concept of Execution](#_kn4329izl7c7)

[4.5 Interface Design](#_m0smb6nxu5r4)

[4.5.1 Interface Identification and Diagrams](#_vmn36labuga9)

[4.5.2 Unique Identifier of Interface](#_p7elni8y9esc)

[5. IMPLEMENTATION ARCHITECTURE](#_u7bjeynzzgw9) x

[5.1 All Active and Passive Classes Assigned to Components](#_z4pbt4i4tqfg)

[5.2 Diagrams of Physical Packaging of Logical Components](#_87ln6yy2juoq)

[6. DEPLOYMENT ARCHITECTURE](#_yxuk479glsyy) x

[6.1 Physical Deployment Architecture Diagram](#_1wl3tndeyuek)

[7. DICTIONARIES](#_mubz64ubno2w) x

[8. SOFTWARE ITEM COMPUTER RESOURCE UTILIZATION](#_4tschkk4d3rl) x

[9. REQUIREMENTS TRACEABILITY](#_drpnlcja0rhs) x

[9.1 Software Component-Level Requirements Traceability](#_5ko57pxeuo44)

[10. SYSTEM DESIGN TESTING](#_5s282587mzni) x

[11. RATIONALE](#_sulcpn63913l) x

[12. NOTES](#_4zk27mp4oqns) x

[13. APPENDICES](#_mn8tqdxonij5) x

[13.1 Dictionaries](#_ur06cfx0bzy6)

[13.2 UML Diagrams](#_9fgkapj58k9i)

[13.3 Schedule Tracking](#_1q50nsi26v7d)

[13.4 Defect Tracking](#_22r4ks61ssky)

[13.5 Gantt Chart/Microsoft Project Schedule](#_au5flkt39fmm)

# **1. INTRODUCTION**

## **1.1 Purpose**

The purpose of this document is to specify the overall design of Sharit and to characterize its architecture. This document is intended for the development team and management.

## **1.2 Scope**

The project is a website that facilitates conversations and file sharing between people from a common organization. The layout will be a clean and simple format for easy intuitive understanding. Organizations will have a domain that will be split into smaller pieces, such as the subdomain and topics. Users will have permissions regarding which organization domain and subsequent subdomains they may access.

Students can use this service to post notes for the underclassmen and receive feedback on their work. Students will be able to collaborate and create notebooks that can be shared with other students in the class. This will benefit those who may not have been able to attend school for that day or those who want to use this platform to study for an upcoming exam. The key to our platform is organization of uploaded files; we want people to be able to look up the file they want and quickly access it or ask the owner of the file to grant permission to access.

Upon release, students that are part of the NYU can access the site with the email and creating a password. Teachers can also access the site with their NYU credentials.

## **1.3 Identification**

This document is the Software Design Description for Sharit. This is revision 1.0, number 1. This document will be released on 10/25/16.

## **1.4 Document Summary**

The software design description (SDD) aims at defining, in detail, the core functionalities of the Sharit system, its implementation, design, class architecture, deployment methods and definitions of components.

## **1.5 System Overview**

Students can use this service to post notes for the underclassmen and receive feedback on their work. Students will be able to collaborate and create notebooks that can be shared with other students in the class. This will benefit those who may not have been able to attend school for that day or those who want to use this platform to study for an upcoming exam. The key to our platform is organization of uploaded files; we want people to be able to look up the file they want and quickly access it or ask the owner of the file to grant permission to access.

The project is a website that facilitates conversations and file sharing between people from a common organization. The layout will be a clean and simple format for easy intuitive understanding. Organizations will have a domain that will be split into smaller pieces, such as the subdomain and topics. Users will will have permissions regarding which organization domain and subsequent subdomains they may access.

## **1.6 Document Overview**

The document will cover the system architecture of Sharit, software class interaction, system testing, and deployment architecture. See the table of contents for more details.

# **2. REFERENCE DOCUMENTS**

Team A6 System Requirements Specification, Version 2.0, March 23, 2016

Team A6 System Analysis Specification, Version 1.0, April 18, 2016

Team B6 Project Proposal, Version 1.0, September 20, 2016

Team B6 Software Project Management Plan, Version 2.0, September 27, 2016

Team B6 Requirements/Analysis Specification, Version 1.0, October 4, 2016

# **3. SYSTEM WIDE DESIGN DECISIONS**

## **3.1 Software Component Architectural Design**

The application server will handle the data between the frontend and the backend. The backend includes the database server and the file server. The frontend is the application server itself. In general, the application server will be in charge of displaying information and handling data changes that it will send to the backend as necessary. Since the application server is the main point of access for the user, it is important that it is able to handle a large load proportional to the number of users.

Software Component Architecture.png

## **3.2 Software Architecture General Description**

The user interface is the main component of Sharit. It is the front-end and will be the point of access for the user. It is composed of the settings, navigation bar, and the main view. The settings is where the user can sign in and change account information and website settings. The navigation bar will be used to navigate the domains and subdomains.The main view is used to display the threads. From the threads, users can download shared files. The main view can also be used to create new threads with attached files.

## **3.3 Software Item Components**

|  |  |
| --- | --- |
| **Component** | **Description** |
| Settings | This is where the use can change account information and site settings. Future site functionality, such as searching, will also be located here. |
| Account Manager | This is used to manage user accounts. Login and account changes will be handled by this component. |
| User Interface | The front-view where the user will interact with the site. |
| Navigation Bar | Handles navigation between domains and subdomains that the user has permission to access. |
| Main View | Component that will show available threads and the comments and files in each thread. |
| Domain Manager | Handles domain and subdomain authentication for users. |
| Thread Manager | Handles organizing threads, comments, and files. Future functionality may include sorting threads. |
| File Manager | Handles file uploads and downloads. |

## **3.4 Component Interface Identification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Interface 1** | **Interface 2** | **Interface 3** |
| User Interface | Navigation Bar | Main View | Settings |
| Navigation Bar | Domain Manager | Manager | Navigation |
| Main View | Thread Manager | Manager | View |
| Settings | Account Manager | Manager | Account |
| Thread Manager | File Manager | Manager | File |

## **3.5 Software Component Concept of Execution**

|  |  |  |
| --- | --- | --- |
| **Component** | **Goal** | **Execution** |
| Settings | The user wants to change accounts settings or site settings | The user presses the Profile link or Settings link on the settings. |
| Account Manager | The user wants to change account settings. | The user changes the desired information and saves the changes. |
| User Interface | The user wants to use Sharit. | The user navigates the website through the User Interface. |
| Navigation Bar | The user wants to access a domain or subdomain. | The user clicks the domain or subdomain he wishes to access. |
| Main View | The user wants to access a thread and read comments or download files. | The user clicks on the links to access a thread and download a file. |
| Domain Manager | The user wants to know the what domains or subdomains are accessible. | The domain manager lets the navigation bar know which domains are accessible by the user. |
| Thread Manager | The user wants to access a thread. | The thread manager lets the main view know which threads are in this domain and what each comment contains. |

# 

# 

# **4. SOFTWARE ITEM DETAILED DESIGN**

## **4.1 Structure**

All data will be stored in a Postgres database server. Each transaction to the database will follow ACID (atomicity, consistency, isolation, durability) to ensure that the database is never left in an inconsistent state and that all committed changes will remain on the server. This will minimize the amount of rollbacks in the database due to errors and improve performance. Files will be stored on a file server.

### **4.1.1 Software Unit Detailed Design**

ClassDiagram2.png

## **4.2 Static Relationship of Software Unit**

### **4.2.1 Runtime Object Instances**

SDD 4.2.1.png

## **4.3 Behavior**

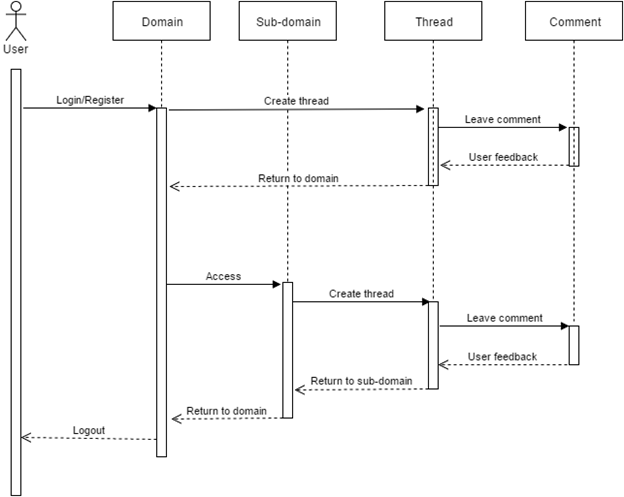
Sharit is broken up into domains and subdomains (subsharits). Domains hold subdomains. In each subdomain, there are threads. In threads, there are comments and optionally, files attached to these threads. A domain will represent an organization. For the minimum viable product only the NYU organization will be present. In the NYU domain, the subdomains will be various classes.

In order to use Sharit, a user must register and login. He may then request access to join a domain and subsequent subdomains. Once he has access he may view all threads in a subdomain, post comments, or create new threads. He may also start his own subdomain for other users in his domain to join. Regarding threads and comments, the user will be able to upvote or downvote based on relevancy and usefulness of the thread/comment. Finally, he may download files attached to the threads. In the future, files may also be attached to comments.

Domain moderators have the power to delete subdomains, threads, and comments as they see fit. Subdomain moderators have similar powers, but on a subdomain level. As of now, moderators have not been implemented, but will be in the future.

Administrators are all-powerful users. They have all the powers of a domain moderator in addition to the power of creating or deleting a domain. In Sharit, administrators are employees of the company.

### **4.3.1 Interaction Diagrams**



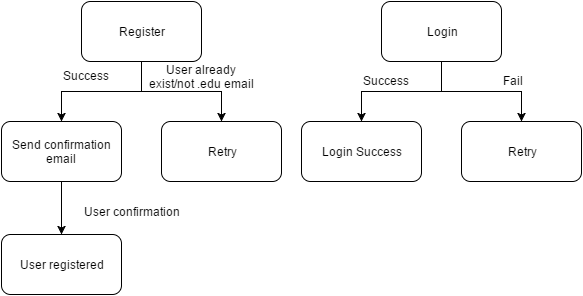
### **4.3.2 Collaboration Diagrams**

This is a very simplified collaboration diagram as there would be too many actions to fit everything on one clean diagram.

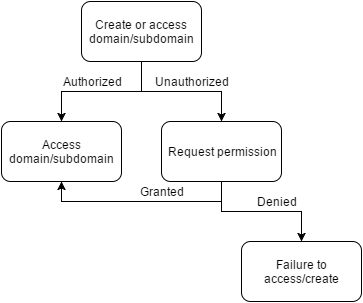
Collaboration Diagram.png

### **4.3.3 Activity Diagrams**

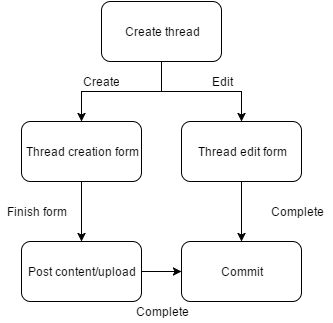
The user tries to login or register into Sharit.



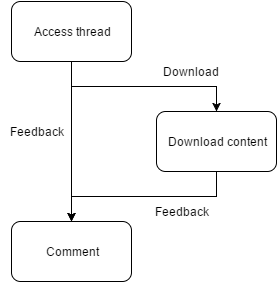
The user tries to access or create a domain or a subdomain.



The user tries to create a thread after gaining access to a domain or subdomain.



The user wants to download content from a thread or leave feedback



## **4.4 Concept of Execution**

|  |  |  |
| --- | --- | --- |
| **Component Name** | **Motive** | **Expected Outcome** |
| Menu | Allows users to modify settings as well as maintain all site settings. | The user successfully creates an account and can update account settings and site settings. (Database insert and update) |
| Create Thread | User can create a thread and set permissions for it. | The user successfully creates a thread with information. (Database insert) |
| Feedback | User can leave comments and feedback on threads. | The user successfully leaves a comment or feedback on a thread.(Database insert |
| Download Content | User can download content that is in a thread. | The user successfully accesses a thread and downloads content (Database select) |
| Request Permission | User requests permission from thread owner to access thread. | User sends a message to the thread owner(Database select) |
| Grant Permission | User grants access to a thread | Thread owner receives a message and can accept or deny the request(Database update,insert) |

## **4.5 Interface Design**

### 4.5.1 Interface Identification and Diagrams

User Interface:

1. Create account
   1. INSERT INTO user(username, firstName, lastname, password, email, dob) VALUES (...)
   2. SELECT user\_id FROM user WHERE “some\_username” = username and “some\_password” = password

Navigation Bar:

1. View domains
   1. SELECT \* FROM domain
2. View subdomains
   1. SELECT \* FROM subdomain

Main View:

1. SELECT \* FROM thread
2. SELECT \* FROM comment

Settings:

1. Change account settings
   1. UPDATE user SET password = new\_val WHERE id = user\_id
   2. UPDATE user SET email = new\_val WHERE id = user\_id

Thread Manager:

1. Create thread
   1. INSERT INTO thread(title, user, datePosted) VALUES (...)
2. View thread
   1. SELECT \* FROM thread WHERE thread\_id = domain\_id
3. Delete thread
   1. DELETE FROM thread WHERE thread\_id = domain\_id
4. Create comment
   1. INSERT INTO comment(user, comment, datePosted) VALUES (...)
5. View comment
   1. SELECT \* FROM comment WHERE thread\_id = comment\_id
6. Delete comment
   1. DELETE FROM comment WHERE thread\_id = comment\_id

### **4.5.2 Unique Identifier of Interface**

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Interface 1** | **Interface 2** | **Interface 3** |
| User Interface | Navigation Bar | Main View | Settings |
| Navigation Bar | Domain Manager | Manager | Navigation |
| Main View | Thread Manager | Manager | View |
| Settings | Account Manager | Manager | Account |
| Thread Manager | File Manager | Manager | File |

# **5. IMPLEMENTATION ARCHITECTURE**

## **5.1 All Active and Passive Classes Assigned to Components**

Active and passive classes are not required in this project.

## **5.2 Diagrams of Physical Packaging of Logical Components**

Diagrams of physical packaging of logical components are not required in this project.

# **6. DEPLOYMENT ARCHITECTURE**

## **6.1 Physical Deployment Architecture Diagram**

SDD6.png

# **7. DICTIONARIES**

See appendix section 13.1 for dictionaries.

# **8. SOFTWARE ITEM COMPUTER RESOURCE UTILIZATION**

The Sharit file server will require 100TB of disk storage for user uploads, threads, user information, domains and sub-domain, and posts. This server will need to be periodically upgraded as more files get uploaded and the system grows. To ensure that the user will have seamless upload/download speeds, the system should have a minimum of 500Megabits/s upload and download. To support many concurrent users, the main application server will require 100GB of RAM, increasing with more traffic.

# **9. REQUIREMENTS TRACEABILITY**

## 9.1 Software Component-Level Requirements Traceability

The functional and nonfunctional requirements will be thoroughly traced throughout the life of the software engineering process. Each specification will be corresponded with an artifact in each of the workflows and be traced. Requirements tracing is done to avoid litigation and also mitigates liabilities. Keeping track of requirements will make sure everything is implemented and that the product meets specifications.

# 10. SYSTEM DESIGN TESTING

The document, code, system design and architecture will go through many rigorous reviews to ensure that there are no defects and that the system is most optimal. There will be several walkthroughs and inspections by various members of the team to review all the information related to the Sharit. Each team member has a thorough understanding of all the system specifications and documentation. All faults that are found will be reported to all the other members of the team for verification. Once all team members have inspected the code and documentation, faults will be promptly corrected by the person in charge of that piece of work. Once the defects have been found, another team member will inspect the correction for further defects.

All aspects of the software development life cycle will be documented and monitored in each phase and workflow. Quality control will be handled by the SQA. Reviews will happen at regular intervals and at the end of each workflow. Each team member will perform validation checks once the software is complete before the system is finalized.

# **11. RATIONALE**

Finding information relevant to coursework can drastically improve one’s grades. NYU-Poly has historically incurred an usually low graduation rate (50~60%). Information, pertaining previous semester student notes, tests, and possibly homeworks, should be upheld as samples for students currently taking the course to help them review. In the requirements, we aim to answer general questions posted by students that not even google.com, NYUClasses, and instructors could not provide. This is most often phrased “what’s the format” or “is there a sample midterm?”. To uphold integrity of information, we provide and limit information and registration to students with a nyu.edu email address.

In the future, if the project expands outside the NYU community, Sharit can be a creative solution for an easier file sharing system while also fostering a sense of community within a group. It intentions are to be simplistic in nature while boasting required functionalities to fulfill its purpose. Clear organization will allow intuitive understanding to allow the user to quickly execute their actions.

# **12. NOTES**

Currently, Sharit is developed to be accessible only by those with an NYU email address. In the future access may be expanded to allow all email addresses.

# **13. APPENDICES**

## **13.1 Dictionaries**

**CLASSES**

|  |  |  |
| --- | --- | --- |
| **User** - represents a user in the system | | |
| username | The user’s username | String |
| firstName | The user’s first name | String |
| lastName | The user’s last name | String |
| password | The user’s password | String |
| email | The user’s email address | String |
| dateCreated | The timestamp the user was created | String |
| lastLogin | The last timestamp when the user logged in | String |
| privilege | The user’s privilege | int |
| dob | The user’s date of birth | String |
| organizations | A collection of the organizations the user is part in | Collection<String> |
| createAccount(username, firstName, lastName, password, email, dob) | Creates a new user | Boolean - returns true if successful |
| changePassword(newPassword) | Changes the user’s password | void |
| login(username, password) | Logs in the user | Boolean - returns true if successful |
| changeEmail(newEmail) | Changes the user’s email address | Void |

|  |  |  |
| --- | --- | --- |
| **Domain** - represents an organization’s domain | | |
| name | The domain’s name | String |
| description | A description of the domain | String |
| privilegeLevel | The user privilege required to access the domain | int |
| organization | The organization this domain belongs to | String |
| createDomain(name, description, privilegeLevel) | Creates a new domain with the passed in parameters | Boolean - returns true if successful |
| changeName(newName) | Changes the domain’s name | void |
| changeDescription(newDescription) | Changes the domain’s description | void |
| changePrivilegeLevel(newPrivilegeLevel) | Changes the domain’s access privilege level | void |

|  |  |  |
| --- | --- | --- |
| **Subdomain** - represents a subdomain, a mini-domain part of a domain | | |
| name | The subdomain’s name | String |
| description | A description of the subdomain | String |
| domain | The domain name this subdomain is part of | String |
| organization | The organization this subdomain belongs to | String |
| createSubdomain(name, description, domain, organization) | Creates a new subdomain with the passed in parameters | Boolean - returns true if successful |
| changeName(newName) | Changes the subdomain’s name | void |
| changeDescription (newDescription) | Changes the subdomain’s description | void |

|  |  |  |
| --- | --- | --- |
| **Thread** - represents a topic of discussion | | |
| title | The thread’s name | String |
| comments | A collection of all the Comments in the thread where the first comment is original post | Collection<Comment> |
| datePosted | The timestamp the thread was created | String |
| user | The user who created the thread | String |
| createThread(title, comment, subdomain) | Creates a new thread in the specified subdomain | Boolean - returns true if successful |

|  |  |  |
| --- | --- | --- |
| **Comment** - represents a user’s comment | | |
| user | The user who posted this comment | String |
| comment | The user’s comment | String |
| datePosted | The timestamp the thread was created | String |
| createComment(thread, comment) | Creates a new comment in the specified thread) | Boolean - returns true if successful |
| deleteComment() | Deletes the comment from the thread | void |

**RELATIONSHIPS**

|  |  |  |  |
| --- | --- | --- | --- |
| Class 1 | Class 2 | Description | Cardinality |
| User | Domain | A user can access a domain | Many-to-many |
| Domain | Subdomain | A subdomain is part of domain | One-to-many |
| Subdomain | Thread | A thread is part of a subdomain | One-to-many |
| Thread | Comment | A comment is part of a thread | One-to-many |

## **13.2 UML Diagrams**

Use Case Diagram.png

### 

Use case diagram:

|  |  |  |
| --- | --- | --- |
| **Register** | | |
| **Description** | Newcomers could register for an account on the website. | |
| **Pre-Conditions** | Registering for an account requires a @nyu.edu email address. | |
| **Flows** | **Basic or Normal Flows** | 1. Registrants registers for an account using @nyu.edu email address.  2. After registering, the server will send an email confirmation (click link to verify) to the registered email address.  3. Email address will be verified by server and registration complete. |
|  | **Alternative Flows** | 1. Registrant attempts to register with an invalid email address - not @nyu.edu  2. Registrant account creation denied |
| **Post Conditions** | Registrant becomes registered, functionality of the website is granted. | |
| **Special Requirements** | Email address used to register must be @nyu.edu. | |
| **Extension Points** | Invalid email addresses will result in rejection to account creation. | |

|  |  |  |
| --- | --- | --- |
| **Login** | | |
| **Description** | A user who has previously registered will be able to log in using the registered username and password. | |
| **Pre-Conditions** | The user must have been previously registered. | |
| **Flows** | **Basic or Normal Flows** | 1. User entered login information.  2. Website validates with database and the information is in the system.  3. The user is redirected to the homepage. |
|  | **Alternative Flows** | 1. User enters invalid credentials.  2. User is prompted to enter again.  3. Process repeats until valid credentials have been provided. |
| **Post Conditions** | The user is logged in and can use the website and its functionalities. | |
| **Special Requirements** | The system will create a session to store the user’s information. | |
| **Extension Points** | 1. User enters invalid credentials   * Prompt user to reenter credentials   2. User forgot the login information   * Has option to recover via email | |

|  |  |  |
| --- | --- | --- |
| **Edit account information** | | |
| **Description** | Users can edit their account information | |
| **Pre-Conditions** | Must login to edit account information | |
| **Flows** | **Basic or Normal Flows** | 1. User specifies which fields they desire to edit.  2. User then edit desired fields.  3. User then saves the information and server will acknowledge saved changes. |
|  | **Alternative Flows** | 1. User specifies which fields they desire to edit.  2. User provides invalid information or information that the server does not know how to process.  3. User edited fields will not be saved and error messages will be displayed. |
| **Post Conditions** | User account information is updated with saved changes. | |
| **Special Requirements** | All updated changes will be saved into the database. | |
| **Extension Points** | If a user tries to enter invalid information, such as wrong formatting, error message(s) pertaining to incorrect fields will be displayed. | |

|  |  |  |
| --- | --- | --- |
| **View other users’ profile** | | |
| **Description** | A user can view the information of other users | |
| **Pre-Conditions** | The user must be logged in and within the same domain | |
| **Flows** | **Basic or Normal Flows** | 1. User is redirected to the other user’s profile page  2. User is able to view basic information like name, school, and threads  3. User can see account activity |
|  | **Alternative Flows** | There are no alternate flows. |
| **Post Conditions** | The user is able to view another user’s basic information and account activity which includes: comments, threads and upvotes | |
| **Special Requirements** | The user must be logged in and within the same domain as the user they are trying to view. | |
| **Extension Points** | If the user tries to view the profile of another user who has deleted their account, an error message will be displayed. | |

|  |  |  |
| --- | --- | --- |
| **Access domain and subdomain** | | |
| **Description** | Head administrators and users can access domains that expand to further subdomains and threads. | |
| **Pre-Conditions** | The groups must be logged in and authorized. | |
| **Flows** | **Basic or Normal Flows** | 1. User chooses which domain to enter  2. User enters domain  3. User can see subdomains and, if have access to, its threads |
|  | **Alternative Flows** | 1. User does not have access to domain  2. User enters a domain they don’t have access to  3. Access denied |
| **Post Conditions** | User can view accessed subdomains and threads of entered domain. | |
| **Special Requirements** | User must have access to enter desired domain and subdomain | |
| **Extension Points** | If user tries to enter a domain they do not have permission to, they will not see its domain contents: subdomains and its threads. | |

|  |  |  |
| --- | --- | --- |
| **Create thread** | | |
| **Description** | Users can create threads within subdomains. | |
| **Pre-Conditions** | The user must be authorized within the subdomain and logged in. | |
| **Flows** | **Basic or Normal Flows** | 1. User is logged in and is authorized in the subdomain that they want to post to.  2. User browses to the subdomain  3. Use posts a thread |
|  | **Alternative Flows** | 1. User is logged in but is not authorized in the subdomain.  2. User browses to the subdomain.  3. User cannot post a thread. |
| **Post Conditions** | There is now an option to upload a file into the subdomain and leave comments. | |
| **Special Requirements** | User must be logged in and received permission from the administrator. | |
| **Extension Points** | If the user does not have authorization, they will not see the create thread button or be able to view any threads in the subdomain. | |

|  |  |  |
| --- | --- | --- |
| **Upload/Download** | | |
| **Description** | Users can upload and/or download file(s) to thread | |
| **Pre-Conditions** | The user must be authorized within the subdomain and logged in. | |
| **Flows** | **Basic or Normal Flows** | 1. User finds which thread to upload/download to/from  2. User uploads/downloads to/from a thread (comment-link)  3. User can post new thread (thread-link) |
|  | **Alternative Flows** | 1. User uploads something inappropriate  2. File(s) uploaded  3. File(s) deleted by moderator |
| **Post Conditions** | User will successfully download file from thread or upload file(s) to thread (comment-link) or make a new thread (thread-link). | |
| **Special Requirements** | User uploaded file(s) will be saved to database. | |
| **Extension Points** | If user tries to upload something inappropriate, the file(s) will be uploaded, but will be quickly brought down by moderators. | |

|  |  |  |
| --- | --- | --- |
| **Feedback** | | |
| **Description** | The user can leave a comment or upvote/downvote in a thread. | |
| **Pre-Conditions** | The user must be logged in and authorized within the subdomain. | |
| **Flows** | **Basic or Normal Flows** | 1. User navigates to a subdomain.  2. User finds a thread that they want to leave feedback for.  3. User leaves feedback. |
|  | **Alternative Flows** | 1. User navigates to a subdomain.  2. User does not have authorization within the subdomain.  3. User is unable to see threads or leave feedback. |
| **Post Conditions** | The feedback that the user leaves will be visible to other users. | |
| **Special Requirements** | User must be authorized within the subdomain | |
| **Extension Points** | There will be a character limit on the comments that the user leaves. | |

|  |  |  |
| --- | --- | --- |
| **Search for threads, subdomains, users** | | |
| **Description** | Head and domain administrators, as well as users, can search for existing threads, subdomains, and/or users | |
| **Pre-Conditions** | Must be logged in and have authorization. | |
| **Flows** | **Basic or Normal Flows** | 1. Specify which thread, subdomain, and/or user to search for  2. Searches for specified item  3. Item found and displayed |
|  | **Alternative Flows** | 1. User chooses which thread/subdomain/user to search for  2. Specified item does not exist  3. Error message displayed |
| **Post Conditions** | Request of item will be delivered if it exists and request party has access to. | |
| **Special Requirements** | In order to use the search functionality, head and domain administrator, as well as user, will require access beforehand. | |
| **Extension Points** | If search functionality is used without proper access rights, error message will be displayed | |

|  |  |  |
| --- | --- | --- |
| **Grant/revoke domains, subdomains access/privilege** | | |
| **Description** | Administrators will be able to grant or revoke permission/access to users or other administrators | |
| **Pre-Conditions** | The administrator must be registered in the database | |
| **Flows** | **Basic or Normal Flows** | 1. User requests privilege from administrators.  2. Administrator receives the request.  3. Administrator grants request. |
|  | **Alternative Flows** | 1. User requests privilege from administrators.  2. Administrator receives the request.  3. Administrator denies request. |
| **Post Conditions** | The user or other administrators now have privilege/access to a domain or subdomain. | |
| **Special Requirements** | Authorization must be granted from a current administrator. | |
| **Extension Points** | 1. Access to a subdomain grants access to all threads in that sub-domain  2. Privilege to to a domain or subdomain means you are the administrator | |

|  |  |  |
| --- | --- | --- |
| **Create domains, sub-domains** | | |
| **Description** | Administrators can create domains and subdomains. | |
| **Pre-Conditions** | Administrators must be logged in. | |
| **Flows** | **Basic or Normal Flows** | 1. Administrator logs in.  2. Administrator creates a domain.  3. Domain is now visible. |
|  | **Alternative Flows** | 1. Administrator logs in.  2. Administrator creates a subdomain.  3. Subdomain is now visible. |
| **Post Conditions** | The domain or subdomain can be accessed by those who have permission, | |
| **Special Requirements** | Only head administrator can create domains. Domain administrator can create subdomains. | |
| **Extension Points** | Head administrator has the most power, followed by domain administrator, followed by subdomain administrator. | |

|  |  |  |
| --- | --- | --- |
| **Delete domains, subdomains, threads** | | |
| **Description** | Administrators can delete domains, subdomains, and threads. | |
| **Pre-Conditions** | Administrators must be logged in. | |
| **Flows** | **Basic or Normal Flows** | 1. Administrator logs in  2. Administrator deletes domain |
|  | **Alternative Flows** | 1. Administrator logs in  2. Administrator deletes subdomain |
|  |  | 1. Administrator logs in  2. Administrator deletes a thread |
| **Post Conditions** | The domain/subdomain/thread is deleted. | |
| **Special Requirements** | Only head administrators and domain administrators can delete domains. Only domain administrators and subdomain administrators can delete subdomains. | |
| **Extension Points** | Deleting a domain deletes the domain and all subdomains within. Deleting a subdomain deletes the subdomain and all threads within. | |

## **13.3 Schedule Tracking**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Artifact or Deliverable** | **Whom** | **Estimated** | **Actual** | **Difference** |
| Initial SRS | Allen Zheng | 3 hr | 4 hr | 1 hr |
|  | Hui Huang | 3 hr | 5 hr | 2 hr |
|  | Kenneth Liang | 3 hr | 7 hr | 4 hr |
|  | Warlon Zeng | 3 hr | 5 hr | 2 hr |
|  | Summary | 12 hr | 21 hr | 9 hr |
| SRS 2.0 | Allen Zheng | 2 hr | 2 hr | 0 hr |
|  | Hui Huang | 2 hr | 4 hr | 2 hr |
|  | Kenneth Liang | 2 hr | 3 hr | 1 hr |
|  | Warlon Zeng | 2 hr | 4 hr | 2 hr |
|  | Summary | 8 hr | 13 hr | 5 hr |
| Initial SPMP | Allen Zheng | 3 hr | 3 hr | 0 hr |
|  | Hui Huang | 3 hr | 5 hr | 2 hr |
|  | Kenneth Liang | 3 hr | 4 hr | 1 hr |
|  | Warlon Zeng | 3 hr | 4 hr | 1 hr |
|  | Summary | 12 hr | 16 hr | 4 hr |
| SPMP 2.0 | Allen Zheng | 2 hr | 1 hr | 1 hr |
|  | Hui Huang | 2 hr | 2 hr | 0 hr |
|  | Kenneth Liang | 2 hr | 1 hr | 1 hr |
|  | Warlon Zeng | 2 hr | 2 hr | 0 hr |
|  | Summary | 8 hr | 6 hr | 2 hr |
| Initial SAS | Allen Zheng | 3 hr | 4 hr | 1 hr |
|  | Hui Huang | 3 hr | 5 hr | 2 hr |
|  | Kenneth Liang | 3 hr | 3 hr | 0 hr |
|  | Warlon Zeng | 3 hr | 2 hr | 1 hr |
|  | Summary | 12 hr | 14 hr | 2 hr |
| Initial RAS | Allen Zheng | 3 hr | 3 hr | 0 hr |
|  | Hui Huang | 3 hr | 4 hr | 1 hr |
|  | Kenneth Liang | 3 hr | 5 hr | 2 hr |
|  | Warlon Zeng | 3 hr | 3 hr | 0 hr |
|  | Summary | 12 hr | 15 hr | 3 hr |
| Initial SDD | Allen Zheng | 4 hr | 3 hr | 1 hr |
|  | Hui Huang | 4 hr | 3 hr | 1 hr |
|  | Kenneth Liang | 4 hr | 4 hr | 0 hr |
|  | Warlon Zeng | 4 hr | 3 hr | 1 hr |
|  | Summary | 16 hr | 13 hr | 3 hr |
| SDD 2.0 | Allen Zheng | 2 hr | 1 hr | 1 hr |
|  | Hui Huang | 2 hr | 1 hr | 1 hr |
|  | Kenneth Liang | 2 hr | 2 hr | 0 hr |
|  | Warlon Zeng | 2 hr | 1 hr | 1 hr |
|  | Summary | 8 hr | 5 hr | 3 hr |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| **Whom** | **Estimated** | **Actual** | **Difference** |
| Allen Zheng | 22 hr | 21 hr | 1 hr |
| Hui Huang | 22 hr | 29 hr | 7 hr |
| Kenneth Liang | 22 hr | 29 hr | 7 hr |
| Warlon Zeng | 22 hr | 24 hr | 2 hr |
| Summary | 88 hr | 103 hr | 17 hr |

## **13.4 Defect Tracking**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Artifact or Deliverable** | **Whom** | **Estimated** | **Actual** | **Difference** |
| Initial SRS | Allen Zheng | 5 | 3 | 2 |
|  | Hui Huang | 5 | 2 | 3 |
|  | Kenneth Liang | 5 | 4 | 1 |
|  | Warlon Zeng | 5 | 2 | 3 |
|  | Summary | 20 | 11 | 9 |
| SRS 2.0 | Allen Zheng | 4 | 2 | 2 |
|  | Hui Huang | 4 | 5 | 1 |
|  | Kenneth Liang | 4 | 3 | 1 |
|  | Warlon Zeng | 4 | 2 | 2 |
|  | Summary | 16 | 12 | 6 |
| Initial SPMP | Allen Zheng | 5 | 4 | 1 |
|  | Hui Huang | 5 | 5 | 0 |
|  | Kenneth Liang | 5 | 6 | 1 |
|  | Warlon Zeng | 5 | 3 | 2 |
|  | Summary | 20 | 18 | 4 |
| SPMP 2.0 | Allen Zheng | 3 | 4 | 1 |
|  | Hui Huang | 3 | 3 | 0 |
|  | Kenneth Liang | 3 | 4 | 1 |
|  | Warlon Zeng | 3 | 2 | 1 |
|  | Summary | 12 | 13 | 3 |
| Initial SAS | Allen Zheng | 7 | 3 | 4 |
|  | Hui Huang | 7 | 8 | 1 |
|  | Kenneth Liang | 7 | 5 | 2 |
|  | Warlon Zeng | 7 | 4 | 3 |
|  | Summary | 28 | 20 | 10 |
| Initial RAS | Allen Zheng | 6 | 5 | 1 |
|  | Hui Huang | 6 | 7 | 1 |
|  | Kenneth Liang | 6 | 9 | 3 |
|  | Warlon Zeng | 6 | 4 | 2 |
|  | Summary | 24 | 25 | 7 |
| Initial SDD | Allen Zheng | 8 | 6 | 2 |
|  | Hui Huang | 8 | 9 | 1 |
|  | Kenneth Liang | 8 | 10 | 2 |
|  | Warlon Zeng | 8 | 7 | 1 |
|  | Summary | 32 | 32 | 6 |
| SDD 2.0 | Allen Zheng | 2 | 1 | 1 |
|  | Hui Huang | 2 | 1 | 1 |
|  | Kenneth Liang | 2 | 3 | 1 |
|  | Warlon Zeng | 2 | 1 | 1 |
|  | Summary | 8 | 6 | 4 |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| **Whom** | **Estimated** | **Actual** | **Difference** |
| Allen Zheng | 40 | 28 | 12 |
| Hui Huang | 40 | 40 | 0 |
| Kenneth Liang | 40 | 44 | 4 |
| Warlon Zeng | 40 | 25 | 15 |
| Summary | 160 | 137 | 31 |

## **13.5 Gantt Chart/Microsoft Project Schedule**

