DIYup

Team 206
Phyo Htut (phutut@mail.sfsu.edu) [Team Lead]
Eduardo Ramos (eramos4@mail.sfsu.edu) [QA/UI/UX]
Myles Pedronan (mpedrona@mail.sfsu.edu) [Backend Lead]
Antonio Carmona (acarmona@mail.sfsu.edu) [DB/Git Master]
Jason Wei (zwei@mail.sfsu.edu) [Frontend Lead]

Software Engineering
CSC 648/848 Section 4
Fall 2019

Version Table

Date	Milestone	Version
10/3/19	Milestone 1	1.0.0

Table of Contents

Executive Summary

With DIYup, guests and creators alike can browse and upload their own step-by-step DIY projects. Creator or not, our site makes it easy for anyone to pick up a variety of projects and follow step-by-step instructions with pictures to easily complete their project. Users can easily browse through the hottest projects on our site and sort depending on the difficulty of the project or on the category of the project to suit their interests and level of building knowledge. Becoming a registered user comes with its own perks. As a registered user, users can become authors and upload their own projects, share their experience making a project from our site, rate, share and save their favorite projects. As an author, authors can view and respond to comments to posts they've made and edit their original post in case of any typos or other errors. With site administrators there will be a team looking at reported posts and making sure users are following our site's rules and guidelines. Any posts that are reported and found to be violating our sites rules will be deleted by site administrators. This is to ensure that our users are having the most enjoyable experience. Our team consists of five students with a variety of different areas of expertise. Each of us plays a part in working on our strengths to make the entire building of the site as seamless as possible.

Use Cases

Defining Actors

Consider the task of finding actors for a website which provides DIY tutorials. The sites connects the authors and viewers.

Potential actors are:

Guest - A guest represents any user who is not logged into the site.

Registered user - A registered user represents any user who is logged into the site.

Author - An author represents a registered user who posted a tutorial on the site.

Admin - An admin represents person who manages the site.

Actor: Author

An author recently has a new DIY idea for making a rotating closet. The author opens the site and creates a new tutorial for his idea. The author sets the tutorial's category to "Woodworking" and gives the project a difficulty. The author adds a material list and tools list to his tutorial. After the author finishes writing the tutorial, he includes detailed instructions with pictures for each step and submits the tutorial to the website. After a few days, the author's rotating closet tutorial gets good ratings by other users and is shown on the "hot" page.

Actor: Registered User

Before a Registered User decides what he wants to try for his first DIY project, he logs into our site and checks the tutorials that are shown in the "hot" page. The registered user looks into a project on how to make a DIY Food Storage Shelf. The registered user checks the material and tools lists and reads through all the steps. After looking over the category and difficulty of the tutorial, he feels confident

enough to make it himself, so he adds it to his favorite list. The Registered User gets all of the necessary materials and tools from his local hardware store, follows all the steps that the author posted, and completes the project. He returns to the tutorial, leaves a good rating and al difficulty score for the tutorial.

Actor: Guest

One Guest recently heard about the site from his friend and he wants to try to make his own DIY project. The guest looks into the hot page, but he finds that most tutorials are too hard for a beginner, so he sorts them by level of difficulty. The guest believes that the “Fireflies in a Jar” tutorial was cool and easy for a beginner. The guest made his own DIY Jar within a few hours by following the steps. In order to share his picture, he had to sign up a new account. The guest signs up for an account and posts a picture of his project in the comments section of the original tutorial.

Actor: Author

A registered user forgets his account’s password. To retrieve the password, he clicks ‘forgot password’. After the user fills his information and new password, his password is reset. The user goes to his post to view other user’s comments and from user’s comments, the author realizes that he has a typo in his post. The registered user edits his post and comments to apologize for the confusion.

Actors: Admin, Registered User

One Registered User, A, comments under a DIY tutorial advertising another DIY website. Another Registered User, B, reports the comment. When the admin is checking if the site has any security issues, the admin sees the report from Registered User, B. After the admin viewed the report, he believed that the comment is against the rules and deletes Registered User, A’s, comment.

Data Definitions

- *Users*
    - **Registered User**

      A user of the site that is logged in (and has an account). A ***Registered User*** has a unique *email address,* a unique *username*, an *avatar* image, and *password*.

    - **Guest User**

      A user of the site that is not logged in (and may have an account). The information of a ***Guest User*** is not stored in any form.

    - **Admin**

      A site administrator. An ***Admin*** has a unique *email address*, a unique *username*, and *password*.

- *Entities*
    - **Tutorial**

      A tutorial posted by a ***Registered User***. A ***Tutorial*** is associated with the *email address* of a single ***Registered User*** and has a unique *ID*, a *title*, an *image*, a *category*, a *difficulty*, a list of *materials*, a list of *tools*, and a *rating*.

    - **Step**

      A single step of a ***Tutorial***. A ***Step*** is associated with the *ID* of a single ***Tutorial***, has a unique *ID*, some *content*, an *image*, and is associated with the *IDs* of the previous and next ***Steps*** in the ***Tutorial*** that it is associated with.

- **List**

  A list of items associated with a ***Tutorial***. A ***List*** is associated with the ID of a single ***Tutorial***, has a unique ID, a list of *contents*, and a list of *links*.

- **Comment**

  A comment posted on a ***Tutorial*** by a ***Registered User***. A ***Comment*** is associated with the *ID* of a single ***Tutorial*** and the *email address* of a single ***Registered User***. It has a unique *ID*, some content, an *image*, and is associated with the *IDs* of the previous and next ***Comments*** on the ***Tutorial*** that it is associated with.

- *Attributes*

  - **Category**

    Used to describe the main area of work that a ***Tutorial*** is associated with. Several examples include Woodworking and Electronics.

  - **Difficulty**

    Used to describe the amount of experience required in the specified **Category** associated with a ***Tutorial*** that is needed to complete the ***Tutorial***.

    - Author Difficulty

      The difficulty of a ***Tutorial*** set by its author (a ***Registered User***).

    - Viewer Difficulty

      The difficulty of a ***Tutorial*** set by ***Registered Users***.

- **Rating**

  Used to represent the quality of a ***Tutorial*** based on feedback from ***Registered Users***.

Functional Requirements

General

1. A help button shall be present on every page.
2. Tutorials with a large number of reports shall be automatically hidden.
3. Comments with a large number of reports shall be automatically hidden.
4. Comments shall have timestamps.
5. Edited tutorials shall have timestamps.
6. Edited comments shall have timestamps.

Guest

7. Guest users shall be allowed to view all the public contents.
8. Guest users shall not be allowed to rate posts.
9. Guest users shall not be able to comment on posts.
10. Guest users shall not be allowed to report posts.
11. Guest users shall not be allowed to report comments.
12. Guest users shall not be allowed to create posts.
13. Guest users shall not be allowed to delete posts.
14. Guest users shall not be allowed to add posts to personal favorite list.

User

1. User shall route Register User Page when Sign Up button is clicked.
2. User shall invoke Log In Modal when Log In button is clicked.

3. Registered users shall be allowed to Sign In.
4. Registered users shall be allowed to recover password.
5. Registered users shall only rate posts made by other registered users once logged in.
6. Registered users shall edit their own posts once logged in.
7. Registered users shall not edit posts by other registered users.
8. Registered users shall report posts by other registered users.
9. Registered users shall delete their own posts.
10. Registered users shall not delete posts made by other registered users.
11. Registered users shall be allowed to comment on posts.
12. Registered users shall be allowed to report improper comments.
13. Registered users shall be allowed to delete their own comments.
14. Registered users shall not be allowed to delete comments made by other registered users.
15. Registered users shall not be allowed to report their own comments.
16. Registered users shall be allowed to add posts to personal favorite list.
17. Registered users shall be allowed to remove posts from personal favorite list.

Admin

1. Admin accounts are reserved to the developers.
2. Admin accounts are created on the server side.
3. Admin shall delete any post by the users.
4. Admin shall view any post by the users.
5. Admin shall comment on any post by the users.
6. Admin shall delete any comments.

Non-Functional Requirements

Performance Requirements:

- The site's UI shall respond visually within 5 seconds.
- The site shall have an hourly uptime percentage of at least 80%.
- All database tables shall use indexes to speed up queries.

Storage Requirements:

- User images shall be uploaded to and retrieved from Imgur.

Marketing Requirements:

- The site's logo shall be present somewhere on every page.

Privacy Requirements:

- The only personal data that the site shall collect is the email address of a registered user.

Security Requirements:

- Users shall be required to validate their email addresses before logging in for the first time.

Coding Standards:

- No line of code shall have more than 80 characters.
- No functions shall have more than 80 lines of codes.
- Helper functions shall never be declared before the primary function.
- Variables should be self-explanatory.
- No variables shall be unused.
- No functions shall be unused.

- All functions must be thoroughly explained.
- All error messages shall be verbose.

Competitive Analysis

Feature Company	Makezine	Instructables	DIY Network	wikiHow	DIYup
Community Rating	-	-	-	+	+
Save Favorites to User Profile	-	+	-	-	+
Comments from registered users	-	+	-	-	+
Difficulty Rating	+	-	+	-	++
Parts/Tools Lists	+	-	+	+	++
Project Prices	+	-	+	-	-

With DIYup we plan to provide the most useful way for users to find projects that they enjoy and can build according to their building expertise. Like our competitors, we provide a large category of projects to suit their interests, but put a higher emphasis in user experience by allowing more community involvement. Users are able to add a community rating that can help better rate the difficulty of the project and authors can include a parts list to help aid users wanting to build the project. Guests and users visiting our site can find something that they could enjoy building without any hassle.

High-Level System Architecture and Technologies Used

- Sever: AWS micro 1 CPU 1GB RAM
- Operating System: Ubuntu 16.0.4
- Database: MySQL
- Web-Server: Apache
- Server-Side Language: Python
- Additional Technologies:
    - Frontend Framework: Quasar
        - Vue
    - Web Framework: Flask
    - IDE: VSCode

Team

- *Phyo Htut (Team Lead)*
    - As a team lead, Phyo maintains the morale and welfare of all the team members as well as software deployment onto the cloud. He also serves as a satellite frontend and backend engineer to support Jason Wei and Myles Pedronan.
- *Antonio Carmona (Database and Git Master)*
    - As a database and git master, Antonio is in charge of version control as well as data flow for the software. He also serves as a satellite backend engineer to support Myles Pedronan.
- *Myles Pedronan (Backend Lead and Document Master)*
    - As a backend lead, Myles is in charge of creating RESTful APIs so that the client side can retrieve server side data from the server. He also serves as a document master, which means he edits and reviews all the documents of the group.
- *Eduardo Ramos (UI/UX and QA Tester)*
    - As UI/UX and QA tester for the software, Eduardo creates the layout of the program as well as making sure all the components are functionally correct when the events are triggered. If bugs were found, he will report the bugs to Phyo (Team Lead) and Jason (Frontend Lead) in order for the bugs to get resolved.
- *Jason Wei (Frontend Lead)*
    - As a frontend lead, all the client side behaviors and software layouts are implemented and managed by Jason. He works closely with Eduardo and Phyo in order to improve the looks and feels of the program.

Checklist

Item	Status
Team found a time slot to meet outside of class	**DONE**
Github master chosen	**DONE**
Document master chosen	**DONE**
Backend Lead chosen	**DONE**
Frontend Lead chosen	**DONE**
UI/UX and QA chosen	**DONE**
Team decided and agreed together on using the listed software tools and deployment server	**DONE**
Team ready and able to use the chosen backend and frontend framework	**DONE**
Team lead ensured that all team members read the final M1 and agree/understand it before submission	**DONE**
Github organized as discussed in class (eg. master branch, development branch, folder and milestone documents etc.)	**DONE**