DIYup

SW Engineering CSC 648/848 Spring 2019
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]

Milestone 4
12/9/19

Version Table

Date	Milestone	Version
12/12/19	Milestone 4	2.0.0
12/5/19	Milestone 4	1.0.0
11/23/19	Milestone 3	1.0.0
11/15/19	Milestone 2	2.0.0
10/17/19	Milestone 2	1.0.0
10/10/19	Milestone 1	2.0.0
10/3/19	Milestone 1	1.0.0

Table of Contents

Product Summary

Product Name: DIYup

Major Functions

Guest User

Priority 1:

1. Guest users shall be allowed to register for an account for each unique email.
2. Guest users shall be allowed to search for tutorials posted by registered users.
3. Guest users shall be allowed to view tutorials posted by registered users.
4. Guest users shall be allowed to view tutorials based on categories.

Registered User

Priority 1:

1. Registered users shall be allowed to log out or recover their account passwords.
2. Registered users shall be allowed to search for tutorials posted by registered users.
3. Guest users shall be allowed to view tutorials based on categories.
4. Registered users shall be allowed to post tutorials.
5. Registered users shall be allowed to include tools lists and materials lists with their posted tutorials.
6. Registered users shall be allowed to edit or delete their posted tutorials.

Admin

Priority 1:

1. Admin accounts shall be reserved to the developers.
2. Admin accounts shall be created on the server side.
3. Admins shall be allowed to have all of the abilities of registered users.
4. Admins shall be allowed to delete any tutorials.

DIYup allows users to experience DIY projects in an easy to use and seamless application by allowing more community involvement such as allowing users to rate their own projects as well as rate the difficulty of other user's projects. Through this community rating, users are able to better determine the difficulty of projects. Another user friendly feature DIYup uses is a parts and materials list in order to make recreating a project easy for users.

URL: http://ec2-54-153-68-76.us-west-1.compute.amazonaws.com/#/

Usability Test Plan

Test Objectives:

The objective of this usability test plan is to ensure that a new user can easily search for a tutorial. An easy user experience when searching for a tutorial of a specific category or title will allow for an enjoyable experience when using the website. The main purpose of the website is to provide tutorial content for users and allow them to quickly find the information needed. Through this simple user experience, DIYup hopes to become a quick resource for users needing to find a tutorial for their DIY projects. This test plan will help gather important feedback to whether the product is of value to users and what can be done to further improve the product.

Test Description:

System Setup: To test the search function, the user will search for various tutorials using either the search bar to look up tutorial titles or a filter to filter by category. Once the user has accomplished the task, they will be able to assess the accuracy, speed, and difficulty of using the function.

Starting Point: The user will begin by opening the browser of their choosing and navigate to the DIYup website as provided by the URL. The user will then use the search bar in the upper right of the application to search for tutorials by either keywords or the full title of the tutorial. If the user does not know the keywords or the full title, they can use the drop down filter to filter by categories that match their desired tutorial.

Intended Users: The intended users of the application are people of varying levels of expertise with building DIY projects looking for a project that suits their expertise and likings. By having a

wide range of ages and skill sets, this test can help simulate various users intending to use the application

URL: http://ec2-54-153-68-76.us-west-1.compute.amazonaws.com/#/

What is measured: This test is to assess the effectiveness of the application at allowing users to easily find a tutorial of a specific category or title. This test aims to answer, “Are our users satisfied with choosing our site to find a tutorial they are looking for”.

Usability Task Description:

TASK	**DESCRIPTION**
Task	Search for a tutorial using DIYup
Machine State	DIYup website loaded
Successful Completion Criteria	Tutorial matching user preferences found
Benchmark	Completed in 30 sec.

Questionnaire:

	Strongly Disagree				Strongly Agree
I was able to find a desired tutorial quickly	1	2	3	4	5
I thought the application was easy to use	1	2	3	4	5
I would use this web application frequently	1	2	3	4	5
I would need the support of a technical person to be able to use this web application	1	2	3	4	5

QA Test Plan

Test Objectives:

The objective of this QA test plan is to verify that returned tutorial search results properly reflect the search bar text input and category filter settings provided by users. A search using the title search bar should only return tutorials with titles that contain the user's provided search text and the category filter drop-down menu should filter out all tutorials that do not belong in the user's specified category.

Searching for DIY tutorials will likely be the most used function of our site (as writing and publishing tutorials takes more effort on the part of users), and as a result, the site's search function should provide fast and accurate results in response to user queries. This plan involves testing that the title search function and category filter work as expected both independently and together.

HW and SW Setup:

Three tutorials titled "Example Tutorial" will be added to the database with the same user account. Each "Example Tutorial" will have a different category attribute (i.e. one will belong to the category "Cooking", one will belong to the category "Electronics", and one will belong to the category "Craft"). After the three test tutorials are added to the database, the QA tests can all be performed from the site's homepage using only the tutorial title search bar and category filter.

URL: http://ec2-54-153-68-76.us-west-1.compute.amazonaws.com/#/

Feature to be Tested:

The site's search feature (title searching by text and category filtering via drop-down menu) will be tested to confirm that the expected search results are returned when searching for specific tutorials that are known to be present in the database.

QA Test Plan:

Number	Description	Test Input	Expected Output	Pass/Fail
1	Test “Electronics” tutorial category search filter	Tutorial category search filter set to “Electronics”	Out of the three added tutorials, only the “Example Tutorial” with the category “Electronics” is present in the search results.	Pass
2	Test search for tutorial title “Example” with “All” tutorial category filter search	Tutorial title search for “Example” and tutorial category filter set to “All”	The returned search results contain all three added tutorials titled “Example Tutorial”.	Pass
3	Test search for tutorial title “Example Tutorial” with “Cooking” tutorial category search filter	Tutorial title search for “Example Tutorial” and tutorial category filter set to “Cooking”	The returned search results contain only the added “Example Tutorial” that has the category “Cooking”	Pass

Code Review

Frontend

Style: ESLint VueJS

URL: https://eslint.vuejs.org/, https://eslint.org/

Comments.Vue

```
<template>

  <div class ="q-pa-md q-gutter-md">

    <strong>Add your comment</strong>

    <q-input

      filled

      type="textarea"

      v-model="reply"

      maxlength="250"

      placeholder="Leave your comments"

      style="max-width:500px;"

    />

        <!-- sendmes should be sendMes -->

    <q-btn

      @click.prevent="sendmes"

      label="send"

    />



    <q-separator />

    <div>
```

```
<q-list>
  <q-item
    v-for="(comment, ind) in comments"
    :key="ind"
  >

  <div class="row justify-start">
    <div class="q-py-md">
    <q-avatar>
  <!-- use align=”left” -->
      <q-img
        style="align: left"
        src="https://cdn.quasar.dev/img/avatar3.jpg"
      />
    </q-avatar>
    </div>

    <div class=" q-pa-md">
    <q-item-label lines="1">
  <!-- {{ something }} pad the template with space -->
      <p style="font-weight:bold; color:#027BE3">{{comment.user}}</p>
    </q-item-label>
    <q-item-label lines="2" style="padding-left:10px">
  <!-- {{ something }} pad the template with space -->
      {{comment.text}}
    </q-item-label>
```

```
    </div>
  </div>



  <div>
    <br>
  <!-- openreply should be openReply-->
    <q-btn
      @click.prevent="openreply"
      label="reply"
    />



  <!-- v-if=”var” -->
  <!-- replybtn should be replyBtn and replymes should be replyMes -->
    <q-input
      id="replybtn"
      v-if=replybtn
      v-model="replymes"
    />



  <!-- v-if=”var” -->
  <!-- replybtn should be replyBtn and sendreply should be sendReply -->
    <q-btn
      class="q-my-sm"
      v-if=replybtn
      label="send"
      @click.prevent="sendreply"
```

```
        />
      </div>
      </q-item>
    </q-list>
  </div>
 </div>
</template>


<script>
export default {
 data () {
  return {
    reply: '',
      // variables should be camel cased
    replymes: '',
    replybtn: false,
    comments: [
      {
        user: 'whoevercomment it',
        text: 'balabla'
      }
    ]
  }
 },
 methods: {
  sendmes () { // function name should be camel cased
```

```
    if (this.reply !== '') {
      this.comments.push({
        user: this.$q.localStorage.getItem('__diyup__username'),
        text: this.reply
      })
      this.reply = ''
    }
  },
  openreply () { // function name should be camel cased
      // variables should be camel cased
    this.replybtn = true
  },
  sendreply () { // function name should be camel cased
      // variables should be camel cased
    this.replybtn = false
    this.replymes = ''
  }
 }
}
</script>
```

From: Phyo Thein Htut <phtut@mail.sfsu.edu>
Sent: Thursday, December 5, 2019 5:19 PM
To: Zijie Wei <zwei@mail.sfsu.edu>
Subject: Re: Code review

Hi Jason,

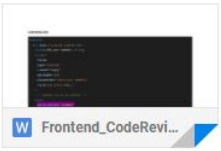
Overall the code looks but there are few things I want you to tweak.

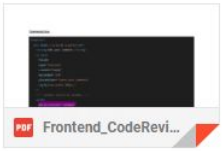
Please check the attachments for the feedbacks and please email me if you have any questions.

Best,
Phyo

From: Zijie Wei <zwei@mail.sfsu.edu>
Sent: Thursday, December 5, 2019 5:03 PM
To: Phyo Thein Htut <phtut@mail.sfsu.edu>
Subject: Code review

2 Attachments





Reply Forward

Backend

Style: PEP 8

URL: https://www.python.org/dev/peps/pep-0008/

Example of PEP 8:

```
Yes: import os
     import sys

No:  import sys, os
```

It's okay to say this though:

```
from subprocess import Popen, PIPE
```

LEGEND:

PURPLE: Code is over 80 characters. **BLUE : Please put comments**

tutorial_routes.py

```
@app.route('/api/tutorial/<username>/<tutorial_uuid>', methods=['GET'])
def get_one_tutorial(username, tutorial_uuid):
        """
        TELL ME SOMETHING ABOUT THIS FUNCTION AS WELL AS THE PARAMS AND ITS RETURN VAL
        """
    sql_query = "SELECT * FROM diyup.tutorials WHERE author_username=%s AND uuid=%s"
    cur = mysql.connection.cursor()
    cur.execute(sql_query, (username, tutorial_uuid))
    tutorial = cur.fetchone()

    if not tutorial:
        return jsonify({'message' : 'No tutorial found!'}), 400

    sql_query = "SELECT * FROM diyup.steps WHERE tutorial_uuid=%s"
    cur.execute(sql_query, (tutorial[0],))
    steps = cur.fetchall()

    output_steps = []

    tutorial_data = {}
    tutorial_data['uuid'] = tutorial[0]
    tutorial_data['author_username'] = tutorial[1]
    tutorial_data['title'] = tutorial[2]
```

```
tutorial_data['image'] = tutorial[3]

tutorial_data['category'] = tutorial[4]

tutorial_data['description'] = tutorial[5]

tutorial_data['author_difficulty'] = str(tutorial[6])

tutorial_data['viewer_difficulty'] = str(average_rating_type_for_tutorial('difficulty', tutorial[0]))

tutorial_data['rating'] = str(average_rating_type_for_tutorial('score', tutorial[0]))



for step in steps:

    step_data = {}

    step_data['index'] = step[1]

    step_data['content'] = step[2]

    step_data['image'] = step[3]

    output_steps.append(step_data)



tutorial_data['steps'] = output_steps



cur.close()



return jsonify({'tutorial' : tutorial_data}), 200
```

user_routes.py

```
@app.route('/api/login', methods=['POST'])

def login():

        “””

        TELL ME SOMETHING ABOUT THIS FUNCTION AND IT’S RETURN VAL

        “””

    data = request.get_json()



    username = data['username']

    password = data['password']



    if not username:

        return make_response('Could not verify auth', 401, {'WWW-Authenticate' : 'Basic realm="Login
required!"'})



    sql_query = "SELECT * FROM diyup.users WHERE username=%s"

    cur = mysql.connection.cursor()

    cur.execute(sql_query, (username,))

    user = cur.fetchone()

    cur.close()



    if not user:

        return make_response('Could not verify user', 401, {'WWW-Authenticate' : 'Basic realm="Login
required!"'})



    if check_password_hash(user[2], password):

        token = jwt.encode({'email_address' : user[0]}, app.config['SECRET_KEY'])
```

```
        return jsonify({'token' : token.decode('UTF-8')})



    return make_response('Could not verify', 401, {'WWW-Authenticate' : 'Basic realm="Login required!"'})
```

comment_routes.py

```
@app.route('/api/comments/<tutorial_uuid>/create/<reply_comment_id>', methods=['POST'])

@token_required

def reply_to_tutorial_comment(current_user, tutorial_uuid, reply_comment_id):

        “””

        TELL ME SOMETHING ABT THIS FUNCTION AND ITS VARIABLES AS WELL AS ITS RETURN VAL

        “””

    data = request.get_json()



    cur = mysql.connection.cursor()



    cur.execute("SELECT * FROM diyup.comments WHERE tutorial_uuid=%s AND reply_to=%s", (tutorial_uuid,
reply_comment_id,))

    uuid = cur.fetchall()



    if not uuid:

        return jsonify({'message' : 'No tutorial ID found!'}), 400



    cur.execute("SELECT * FROM diyup.comments ORDER BY id DESC LIMIT 1")

    index = cur.fetchone()

    id = int(index[0]) + 1



    content = data['content']

    image = data['image']

    reply_to = reply_comment_id

    edited = 0

    date = time.ctime(1574039538)
```

```
    timestamp = date



    cur.execute("INSERT INTO diyup.comments(comments.tutorial_uuid, username, content, created, timestamp,
edited, image, reply_to) VALUES(%s, %s, %s, %s, %s, %s, %s, %s)",

                (tutorial_uuid, current_user[1], content, date, timestamp, edited, image, reply_to,))



    mysql.connection.commit()

    cur.close()



    return jsonify({'message' : 'Reply created!', 'comment id' : id}), 201
```

Antonio Damaso Carmona
Hi Phyo, Here are some snippets of backend code for review (one route regarding tutorials, one route regarding users, and one route regarding comments).

Phyo Thein Htut
to Antonio, me

Hi Myles and Antonio,

I have reviewed the code and there is not much issues rather than some of the issues I have highlighted.

Please check the PDF and let me know if you have more questions.

Overall, the code was great.

Best,
Phyo

From: Antonio Damaso Carmona <acarmona@mail.sfsu.edu>
Sent: Thursday, December 5, 2019 1:31 PM
To: Phyo Thein Htut <phtut@mail.sfsu.edu>
Cc: Myles Pedronan <myles.pedronan@gmail.com>
Subject: Backend Code Review

•••

2 Attachments

Reply | Reply all | Forward

Self-check on Best Practices for Security

- **Major Assets**

    - User email addresses are stored in the database and are never exposed to users other than those that have accounts associated with them (users are all represented to other users by their usernames).
    - Cookies information are all Same Site Strict enabled to make sure no sensitive information stored on cookies are transferred unintentionally by the developers.
    - All sensitive information (email and password), will be encrypted before leaving client and sever.

- **Password Encryption**

    - User passwords stored in the database are being encrypted using SHA-256.

- **Input Data Validation**

    - For Email input field, there will be soft check performed such that the input contains at most one ‘@’, at least one ‘.’, and no special characters other than ‘-’, ‘_’, and ‘.’.
    - Username must be non-spaced string.
    - As for title search input, all special characters will be stripped and replaced with an empty character before being searched.
    - In the backend, all data before entering the database are being sanitized by MYSQLdb python library. The following are the input params affected by the the sanitization:
        i. Email
        ii. Username
        iii. Title
        iv. Password
        v. Title
        vi. Description
        vii. Steps
        viii. Materials

Self-check: Adherence to Original Non-functional Specs

Performance & Reliability Requirements:

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

Storage & Media Requirements:

User images shall be uploaded to Imgur.	**ON TRACK**
Images uploaded by users shall be stored in the database as Imgur links.	**ON TRACK**
Users shall be allowed to upload JPEG and PNG format image files.	**ON TRACK**
The maximum uploaded image file size shall be 2MB.	**ON TRACK**

Privacy & Security Requirements:

Data that the site shall collect includes the email addresses of registered users and pictures uploaded by registered users.	**ON TRACK**
The email addresses provided by registered users shall only be visible to their associated users and DIYup admins.	**ON TRACK**
Users shall be required to validate their email addresses before logging in for the first time.	**ON TRACK**
User passwords shall be stored in the database after being hashed.	**DONE**

Compatibility Requirements:

The site shall be compatible with Chrome 51 and above.	**DONE**
The site shall be compatible with Firefox 54 and above.	**DONE**
The site shall be compatible with Microsoft Edge 14.0 and above.	**DONE**
The site shall be compatible with Safari 10 and above.	**DONE**

The site shall be compatible with Opera 38 and above.	**DONE**

Marketing Requirements:

The site's logo shall be present somewhere on every page of the site.	**DONE**
The name of the site shall always be stylized as "DIYup" when referenced.	**DONE**

Coding Standards:

No line of code shall have more than 80 characters.	**ON TRACK**
No functions shall have more than 80 lines of codes.	**ON TRACK**
Helper functions shall never be declared before the primary function.	**DONE**
Variables should be self-explanatory.	**ON TRACK**
No variables shall be unused.	**DONE (Frontend) | (Backend)**
No functions shall be unused.	**DONE (Frontend) | (Backend)**
All functions must be thoroughly explained.	**ON TRACK**
All error messages shall be verbose.	**ON TRACK**