SW Engineering CSC648-848

Summer 2022

Milestone 4

PlayDate — by Team 03 (the "Babysitters")		
Name	**Job**	**Email**
Soujanya Ravindra Nayak	Team Lead	soujanyaravindra@gmail.com
Margaret De La Torre	Frontend Lead	mdelato1@mail.sfsu.edu
Andy Cho	Backend Lead	andrewyunejo@gmail.com
Martin Salvatierra	Frontend Team	martysaljhost@gmail.com
Qin Geng	Frontend Team	gengqin50@gmail.com
William Plachno	Git Master	wjplachno@gmail.com
Victor Callejas	Frontend Team	vcallejas@mail.sfsu.edu

Date	Version
07/28/2022	M4 V1
07/28/2022	M3 V2
07/19/2022	M3V1
07/19/2022	M2V2
07/19/2022	M1V2
07/07/2022	M2V1
06/21/2022	M1V1

Table of contents

1. Product Summary

PlayDate

Social Media has become a big part of society where adults can find like-minded friends. But what about kids and pets? They, too, need someone of their kind with whom they can spend quality time. Parents try to find playdates for their children and pets, but they are always worried about child safety. There are some applications in the market that try to solve the issue of finding playdates and activities but none ensures guaranteed safety from unauthorized users. How much do we really know about people whom we invite to be our kids' playdates?

"PlayDate" is aiming to help parents solve their pain points while looking for activities and companions for their children and pets. Parents can register in PlayDate and can either join or create events for their kids and pets. All the users of PlayDate need to undergo background verification to be able to create any event or sign up for other user's events. This way PlayDate ensures the safety of your kids and pets.

Functions:

1. General users can view public events on PlayDate that are available to all users.
2. General users shall also be able to search for public events based on location.
3. General users shall be able to register on PlayDate by completing the signup page.
4. General users shall be able to request assistance from playdate support staff by filling the contact form and providing their name, email id and description of assistance.
5. General users shall be able to use chat bot to get directions on using our application.
6. Registered user shall be able to use their login credentials and login to PlayDate application.
7. Registered users shall be able to view and edit their profile where they can add dependents and update their personal information.
8. Registered users need to upload verification documents on profile to become verified users. Hence the registered user shall be able to upload photo identity as verification document on their profile.
9. Registered users shall be able to view public events.
10. Registered users shall be able to log out from their account on PlayDate.
11. Registered users who are not verified shall not be able to create events or create groups. They shall also not be able to join groups or RSVP to events created by other users.
12. Registered users shall be able to view events created by other PlayDate users under Events on PlayDate.
13. Registered user who are verified, can register to many events that they are interested in.
14. Registered users who are verified shall be able to create their own events.
15. Registered users who are verified shall be able to delete the events created by them.
16. Registered users can view all of their events that they created and RSVP'd on My Events
17. Registered users shall be able to create many groups to form small communities of similar interests and become a group admin.

18. Registered users shall be able to search the groups based on interests, location, group name.
19. Registered users shall also be able to join many groups created by other users.
20. Registered users shall be able to view all groups that they are part of, on My Groups.
21. Registered users shall be able to cancel their signup to any event
22. Each group user can create an event for their group, which will be available only to that group's users.
23. Group users can RSVP to the group events.
24. Group admin who created the group shall be able to delete any event from the group.
25. Group users shall be able to post and comment in the group to share their thoughts with their group.
26. Group users shall be able to leave the group if they are no longer interested in that group.
27. Group admin shall be able to remove any user from their group.

Uniqueness:

Unlike adults, kids need an extra safe environment. Hence the unique feature of our application lies in its business logic of user verification which is incorporated to ensure safety from scammers. Post registering on our application, users need to mandatorily upload their identity document to their profile, which will be analyzed by PlayDate backend admins to permit the user to use all the functionalities of our application. If the user does not complete the verification process they will be restricted from core functions like create and join events and groups provided by PlayDate.

Application url: http://34.83.255.32:8000/

2. Usability Test

2.1. Usability Test Plan

2.1.1. Purpose

As an application for parents of children and pets, “PlayDate” pays much attention to user’s and their families’ security. One of our superior features is “User Verification”, which will be tested in this section. “User Verification” will be tested since we want to make sure this superior feature is working well to compete with our competitors. There are 5 major functions related to this superior feature to be tested: upload identification image, sign up for events, join groups, create events, and create groups.

2.1.2. Problem Statement and Objective

- Upload Identification Image: we will test if a user can easily upload the identification photo for verification. A verified user can obtain more privileges, like sign up events and create groups, and it’s important for them to upload the identification photo successfully.
- Sign Up For Event: After the verification, we will test if a verified user can successfully sign up for events on “PlayDate”. This “sign up” function is a core privilege on our website and it’s important for users to successfully implement it.
- Join Groups: After the verification, we will test if a verified user can successfully join a group on “PlayDate”. This “join group” function is a core privilege on our website and it’s important for users to successfully implement it.
- Create Event: After the verification, we will test if a verified user can successfully create an event on “PlayDate”. This “create event” function is a core privilege on our website and it’s important for users to successfully implement it.
- Create Groups: After the verification, we will test if a verified user can successfully create a group on “PlayDate”. This “create group” function is a core privilege on our website and it’s important for users to successfully implement it.

2.1.3. User Profile

Users for the Usability test are registered users who haven’t been verified yet.

2.1.4. Method (test design)

Usability test with metrics of Effectiveness, Efficiency, and Satisfaction.

2.1.5. Task list

- Upload Identification Photo
- Sign Up For Event
- Join Groups
- Create Event
- Create Groups

2.1.6. Test Environment

Test host server website with Google Chrome Incognito on Mac.

2.1.7. Test Monitor Role

Front-end team member

2.1.8. Evaluation measures

Effectiveness & Efficiency form and Satisfaction Questionnaire

2.2. Usability Test Table for Effectiveness & Efficiency

# pages	Test Case	% completed	errors	comments	% time to complete	# steps	Amount of time
1 or 3	Upload Identification Photo	100%	No errors	The acceptable photo formats are: apng, avif, gif, jpeg, jpg, png, webp. If an unverified user tries to create an event or a group, the user will be redirected to the verification step.	0	3	30 seconds
2	Sign Up Event	95%	No errors. But after signing up for an event, the event page format changed.	Only unverified users can sign up for and view details of an event.	5	2	30 seconds
2	Join Group	95%	When unverified users joined a group, “404 error” popped up.	No errors for verified users joining a group. When unverified users join a group, it should show a page rather than 404.	5	2	30 seconds
2	Create Event	98%	No errors. But Individual event page need more styling, the time of events are all “midnight” right now	Only verified users can create an event, unverified users shall be redirected to the verification step when trying to create an event.	5	2	1 minute
2	Create Group	98%	No errors, but need more styling. Couldn’t see my group picture in the group page. “Create Group” button out of page border.	Only verified users can create a group, unverified users shall be redirected to the verification step when trying to create a group. Both groups and individual group pages need more styling.	5	2	1 minute

2.3. Task Description

Task1	Description
Task	Upload Identification Photo http://34.83.255.32:8000/profile/
Machine State	Host Server Running
Successful Completion Criteria	Identification photo uploaded and “Awaiting approval” message showed up.
Benchmark	Completed in 30 seconds

Task2	Description
Task	Sign Up Event http://34.83.255.32:8000/events/members-events/
Machine State	Host Server Running
Successful Completion Criteria	Sign up for an event and can view this event’s individual page.
Benchmark	Completed in 30 seconds

Task3	Description
Task	Join Group http://34.83.255.32:8000/groups/
Machine State	Host Server Running
Successful Completion Criteria	Join a group and can view this group’s individual page.
Benchmark	Completed in 30 seconds

Task4	Description
Task	Create Event http://34.83.255.32:8000/events/my-events/
Machine State	Host Server Running
Successful Completion Criteria	Create an event and can view this event's individual page.
Benchmark	Completed in 1 minute

Task5	Description
Task	Create Group http://34.83.255.32:8000/groups/myGroup/
Machine State	Host Server Running
Successful Completion Criteria	Create a group, can view this group's individual page, and is the admin of this group.
Benchmark	Completed in 1 minute

2.4.User Satisfaction

	Strongly disagree				Strongly agree
1. The port for uploading the identification photo is easy to find.	1	2	3	4	5
2. You can upload different formats of images.	1	2	3	4	5
3. The photo uploading process is easy to implement.	1	2	3	4	5
4. You can review info of all events.	1	2	3	4	5
5. It's easy to sign up for an event.	1	2	3	4	5
6. You can easily get the info of your signed up event.	1	2	3	4	5
7. You can review info of all groups.	1	2	3	4	5
8. It's easy to join a group.	1	2	3	4	5
9. You can easily get the info of groups you joined.	1	2	3	4	5
10. It's easy to create an event.	1	2	3	4	5
11. You can enter details about the event you want to create.	1	2	3	4	5
12. You can see your event on our website after you created it.	1	2	3	4	5
13. It's easy to create a group.	1	2	3	4	5
14. You can enter details about the group you want to create.	1	2	3	4	5
15. You can see your group on our website after you created it.	1	2	3	4	5

16. You are satisfied with the look and feel of the user interface that you used.	Strongly disagree				Strongly agree
	1	2	3	4	5
17. Comments or Advices:					

Average Agreement on User Satisfaction Survey
Survey repository:
https://drive.google.com/file/d/1iH16FyUmyE06jC6T-B_r-9TRNeol26e0/view?usp=sharing

Average Agreement Table:

Statement number	Average User Response
1	4
2	4
3	5
4	4
5	4
6	4
7	3
8	4
9	3
10	4
11	4
12	3
13	4
14	4
15	4
16	4

3. QA Test Plan

HW and SW setup:

Testing hardware 1 - MAcBook Pro 2019

Software 1 - MacOs Monterey Version 12.4

Testing hardware 2 - MacBook Air 2017

Software 2 - macOS Monterey Version 12.3.1

App link: http://34.83.255.32:8000/

TEST 1: Usability

2.1 Users shall receive online help from support for any assistance on the application.

Test Plan Outline:

This test will be performed on the home page where the chat bot is located on the bottom right. The test environment for 2.1 QA test is performed on Chrome and Safari. The action should take a minute or two to complete, this includes the typing and submission process. The risks that the test may run into is if the user submits repeated text or if the automated message lags to reply or simply does not work.

Test Number	Test Title	Test Description	Test input	Expected correct output	Pass/Fail
1	Test typical	Input a typical expected user input.	“hi”	Thanks for the message. Our team is offline. 💤 We will contact you back as soon as we can. 🙂	Pass
2	Test symbols	Test is non typical user input such as numbers and symbols.	Enter, “1!@$@2^2^3433&#%&”, into the input field for the chat help bot.	Auto reply saying something like “Thanks for the message. Our team is offline. 💤 We will contact you back as soon as we can. 🙂”	Fail
3	Test alt language	Test using other languages in input..	Enter, “私は助けが必要です”, into the input field for the chat help bot.	Auto reply saying something or auto message saying that they will reply when they can.	Pass

TEST 2: Security

4.2 - Information should be securely transmitted to the database server without any changes in information.

Test objectives:

This test is to make sure that user input is correctly sent properly to the database and we also want to make sure that password input is type in securely. This test should take about one minute to perform input and about another minute to check the database. A risk of this test is if the test fails we might have to fix data in the backend if incorrect tables are affected.

Test Number	Test Title	Test Description	Test input	Expected correct output	Pass/Fail
1	Password test	Test to see if a password is always hidden when entered.	Enter, "password123", into the password text box when creating an account.	"••••••••" SHA encrypted password. Database has an encrypted password.	Pass
2	Email test	Test to see if email was correctly sent to the database under auth_user table.	Enter "mail@email.com", in the email form when creating an account	In the database "mail@email.com ", is correctly placed into the email section on the auth_user table.	Pass
3	User name test	Test to see if the user name was correctly sent to the data under auth_user table.	Enter "MargaretDLT" under "user name" when creating an account.	In the database "MargaretDLT" was correctly placed under the "username" section of the auth_user table.	Pass

TEST 3: Compatibility
5.1 - Application should be supported on Mac via browsers of versions, Chrome >= 60, Safari >= 12

Test objectives:
The test requires that a standard input is inserted in the navigation search bar for the same results. The search result should work on compatible computers with chrome,firefox, and Safari. The test should take about a minute to verify that results are present for what is being looked for. The risks are that the user users an unsupported browser that fails to render a search.

Test Number	Test Title	Test Description	Test input	Expected correct output	Pass/Fail
1	Chrome	Test if app looks good and search function works well on Chrome.	Enter, “San Francisco”, into the search bar.	Shows propers list of events	Pass
2	Safari	Test if app looks good and search function works well on Safari	Enter, “ ”, into the search bar.	Display a message saying “No results found please try again”	Fail
3	Chrome	Test if app looks good and search function works well on Firefox	Enter, “ ”, into the search bar.	Display a message saying “No results found please try again”	Fail

TEST 4: Data Storage

7.2 - The application's back-end servers should only be accessible to authenticated backend admins.

Test objectives:

The test is to make sure that user input cannot affect the backend database. If a user were to input text that would be similar to backend code, then the backend should remain unaffected. This test should take about a minute to perform the input and then another minute to check the backend. The risks are that the backend might be affected by user input during this test.

Test Number	Test Title	Test Description	Test input	Expected correct output	Pass/Fail
1	Username login with SQL injection	User inputs text similar to a query in username form for login.	"5 OR 1=1" which corresponds to SELECT * FROM Users WHERE UserId = 105 OR 1=1;	User shouldn't be logged in	Pass
2	Username create	User inputs text similar to a query as their username when creating an account.	CREATE DATABASE injectionTest2;	"The username must consist of letters, digits, +, -, .,@,and _." , nothing was created in the database	Pass
3	Create event description	User inputs text similar to a query as their created event description.	CREATE DATABASE injectionTest3;	Uses "CREATE DATABASE injectionTest3;" as description and does not affect the database	Pass

TEST 5: Data Storage
7.1. The application's back-end servers should never display a customer's password.

Test objectives:
The test requires a new user to register an account using different sizes and characters. The password is encrypted by Django using the sha_256 algorithm. The action should take a minute or two to complete, this includes the typing and submission process. The risks that the test may run into are if the user submits more than what we tried of 170 characters and therefore might not be able to work. Another risk would be if the test fails we might be able to see other user passwords.

Test Number	Test Title	Test Description	Test input	Expected correct output	Pass/Fail
1	user1	Test a password that matches requirements.	"Password321!"	An encrypted password in the database on the "auth_user" table.	Pass
2	user2	Test a password that is really long.	"123456789qwertyuiopasdfghjklzxcvbn123456789qwertyuiopasdfghjklzxcvbn123456789qwertyuiopasdfghjklzxcvbn123456789qwertyuiopasdfghjklzxcvbn123456789qwertyuiopasdfghjklzxcvbn"	An encrypted password in the database on the "auth_user" table.	Pass
3	user3	Test a password with a bunch of characters	"aglas#$^#$^#@$"	An encrypted password in the database on the "auth_user" table.	Pass

4. Code Review

1. Coding Style

- Indentation: We are using Python and it uses indentation to indicate control structures, so correct indentation is required. By doing this, the need for bracketing with curly braces is eliminated.
- Header comments: Having an introductory header comment is very important. Django organizes into 'applications'. Our project has the home app, the events app, and the groups app, each of which have a set of standardized files which share names. Having a comment at the top is the perfect place to mention which file you are looking at. It is also good practice to provide a sense of the purpose of that file and what the big ideas are that have an effect on the organization of the file itself.
- Camel case: We use camel case styling for definition and variable names.
- Code management:
    - Our project is divided into multiple applications like: playdate, home, events, groups for structured management.
    - Each application has templates and views.
    - All the frontend code is stored in the templates folder of each application
    - All the backend logic is written in views.py for each application.
    - All the database classes are defined in models.py of respective applications which can be imported into other applications when needed.
    - All the forms are defined in forms.py
    - Navigation tabs, footers, and other reusable html snippets that are common to multiple pages are stored globally in the playdate application's templates folder and are included in other applications, without the need to re-write the headers and footers.
    - All the css, javascript, and media files are stored in static folders for access from all html files.

2. Peer review

Repository for Code Review

https://github.com/andrewyjo/CSC648-Team03-CodeReview

Peer review by other team

Team 01 Code Review of Team 03

Part 1

home/views.py:

- Excellent header providing a short but detailed description of the particular file
- Good organization of imports

- Plenty of informative inline comments
- Consider separating parts of the code with white space rather than simply comments to make it less crowded and easier on the eyes to read
- Good use of standard naming conventions

home/HTML/:

- Good headers
- Consider having minimal inline style. Inline style avoids content and design from being separated. Use CSS files instead. For ex: everything related to font, color, align, etc should be in CSS file not inline, makes the code very crowded
- Consider separating divs with white space to make it less crowded and easier on the eyes to read
- Good use of standard naming conventions
- Very few informative comments
- Consider adding <footer> to a separate view
- Consider putting functions to a separate file

Part 2

events/views.py:

- Excellent header providing a short but detailed description of the particular file
- Good organization of imports
- Utilizes white space well, seems less packed, and is simpler to read.
- Plenty of informative inline comments
- Good use of standard naming conventions

Peer review by other team member

Part 1:

Home/html

- Would want footer as a separate file for better organization.
- Easy to understand and follow

Home/views

- Header, helpful to know which page we are in to edit
- Imports organized and easy to find
- Informative comments
- You would want to reduce the number of if statements. Make it more modular by adding functions. Want to make it easier to debug and modify.

Part2:

events/views

Organization:

- Header, helpful to know which page you are in for editing.
- Imports organized
- Inline comments informative
- Probably same as previous feedback, to reduce number of if statements if possible and make it modular
- Email logic as a separate function for contact support is an idea if possible

5. Self-check on best practices for Security

5.1. List of major assets we are protecting

- Passwords: Passwords are encrypted before storing them on the database.
- Database:
    - Input data validation: Text inputs are protected against SQL injections using the ORM model of Django by validating them.
    - Image validation: Image uploads are validated for proper file type to avoid user uploading harmful files.
- User identity:
    - Users have unique login credentials to login into the application.
    - Each user is assigned a role that differentiates backend admins from users of the application to avoid superior privileges to users of the application.

5.2 Password Encryption:

In Django we use the PBKDF2 algorithm with a SHA256 hash, a password stretching mechanism recommended by NIST.

Password encryption algorithm:

In django we use make_password to encrypt the password.

```
    user.password = make_password(password)
    user.save(using=self._db)
    return user
```

It validates if the password field is empty then returns an unusable string.
If the password is not abiding by password requirements, it raises an error.
If the password is valid then it calls PBKDF2PasswordHasher. This definition uses the pbkdf2_sha256 algorithm which executes 320000 iterations to generate a hash of the password.

```
def make_password(password, salt=None, hasher="default"):
    if password is None:
        return UNUSABLE_PASSWORD_PREFIX + get_random_string(
            UNUSABLE_PASSWORD_SUFFIX_LENGTH
        )
    if not isinstance(password, (bytes, str)):
        raise TypeError(
            "Password must be a string or bytes, got %s." % type(password).__qualname__
        )
    hasher = get_hasher(hasher)
    salt = salt or hasher.salt()
```

```
return hasher.encode(password, salt)
```

In database table password is stored as below:

	id	password	last_login	is_superuser	username
▶	1	pbkdf2_sha256$260000$C02XC61bvVa8nMwK…	2022-07-28 09:03:04.723318	0	andy
	2	pbkdf2_sha256$260000$Fzday2kSLylZNLrLBE…	2022-07-28 13:57:04.849691	1	playdateadmin
	3	pbkdf2_sha256$260000$R47T7JRzA1aTKG3qr…	2022-07-28 04:38:22.178469	0	soujanya
	4	pbkdf2_sha256$260000$v1WgRmy9RpWcng8…	2022-07-28 03:54:42.355237	0	andy2
	5	pbkdf2_sha256$260000$qf5IRHk1l4ekUurwdYc…	2022-07-28 16:23:08.033585	0	wjplachno
	6	pbkdf2_sha256$260000$zh1l8JD8nh5BbIGu6C…	2022-07-28 04:53:08.174907	0	anna45
	7	pbkdf2_sha256$260000$PYNjslEVzS3KoevNV…	2022-07-28 05:03:05.273630	0	sana4536
	8	pbkdf2_sha256$260000$2RutipNnuZTTwEHQ…	2022-07-28 14:06:20.267441	0	qin6
	9	pbkdf2_sha256$260000$apDaUqSgxqmUi7Wm…	2022-07-28 13:58:19.063756	0	qin7
	NULL	NULL	NULL	NULL	NULL

5.3 Input data validation:

SQL Injections are a big security hazard. When you read in input from the client and try to stick it in the database, a nefarious user can set variables so that, instead of just the variables going into the database getting modified, the SQL query itself gets modified. This can lead to automatic password verification and data security breaches.

Thankfully, the Django framework was designed with SQL Injection protection in mind. Django provides the Object Relational Model layer with functionality for both defining the database as well as making database calls secure. Whenever a sql query gets run by the ORM, it double checks all variables for extra SQL. If it finds it, it removes the sql from the variable and only uses the portion not involved in the SQL attack.

While there are ways to side-step the ORM, our application never does. As such, we will be protected from SQL injection consistently.

We also protect ourselves by validating images that get uploaded. We require that all images uploaded to our server be less than 6.5MB, and that they are of a particular format. We allow images of .apng, .avif, .gif, .jpeg, .jpg, .png, and .webp format. We do these checks both client-side (using javascript inside of playdate/templates/imageUpload.html) as well as server-sde using custom validators on forms.

5.4 Search validation

Steps taken to validate search:

1. Firstly the input is validated for empty input. If input is empty search is not performed

```
{% if submitbutton == 'Search' and request.GET.q != '' %}
```

```
{% if results %}

<div class="result-body">
```

2. Search keywords are used in the query. We use the Django ORM model to make queries to the database. ORM removes SQL parts from the user input before checking for results from MySQL.

```
if request.method == 'GET':
    query = request.GET.get('q')
    filter = request.GET.get('category')

    submitbutton = request.GET.get('submit')
    print(query)

    if query is not None:
        if filter == 'All':
            # query databse to check if matching city, zipcode, or street

            lookups = Q(address__city__icontains=query) | Q(address__zipcode__icontains=query) | Q(
                address__country__icontains=query) | Q(address__street__icontains=query)

            results = Publicevent.objects.filter(lookups)

        elif filter == 'Kids':
            # query databse to check if matching city, zipcode, or street

            lookups = Q(address__city__icontains=query) | Q(address__zipcode__icontains=query) | Q(
                address__country__icontains=query) | Q(address__street__icontains=query)

            results = Publicevent.objects.filter(
                lookups).filter(Q(category__icontains='kids'))
            print(filter)
        else:
            lookups = Q(address__city__icontains=query) | Q(address__zipcode__icontains=query) | Q(
                address__country__icontains=query) | Q(address__street__icontains=query)

            results = Publicevent.objects.filter(
                lookups).filter(Q(category__icontains='pets'))
            print(filter)

        context = {'results': results,
```

```
                'submitbutton': submitbutton}
        return render(request, 'events/events.html', context)
```

3. If the search keywords do not match any results from the database, then we return no results found.

```
{% else %}
  <!-- input not found -->
  <div class="grid-container">
    <div class="gal-detail thumb" style="font-size: 130%;">
      <Header>No results found please try again</Header>
    </div>
  </div>
```

6. Self-check: Adherence to original Non-functional Requirements

1. Privacy:
    1.1. Every general user shall sign PlayDate Terms of Service and Privacy when they register. - Done
    1.2. Group admin shall sign an agreement to be responsible for appropriate data sharing in groups. - On Track
    1.3. Users violating PlayDate Terms of Service should be warned for the first time and removed for the second time. - Issue, this non functional requirement is dependent on the functional requirement 2.50 which is part of priority 2.

2. Usability:
    2.1. Users shall receive online help from support for any assistance on the application. - Done
3. Response time:
    3.1. General User shall receive an update on his/her background verification in less than 24 hours of registering on PlayDate. - Done
    3.2. Support staff shall respond to users within 12 hours of raising request. - Issue, implemented by the business and not in the application.
    3.3. Support staff shall send a notification to the backend admin within 30 minutes of receiving reports on inappropriate posts or technical issues. - Issue, implemented by the business and not in the application.
4. Security:
    4.1. PlayDate org should do background-checking when a general user registers on PlayDate.- Done
    4.2. Information should be securely transmitted to the database server without any changes in information. - Done
5. Compatibility:
    5.1. Application should be supported on Mac via browsers of versions, Chrome >= 60, Safari >= 12 - Done
    5.2. Application shall be supported on Windows via browsers of versions, Chrome >= 60 - Done
6. Business need
    6.1. Application should use googleapis to find nearby places to visit. - Issue, it is dependent on a functional requirement which is not implemented in priority 1.
7. Data Storage
    7.1. The application’s back-end servers should never display a customer’s Password. - Done
    7.2. The application’s back-end servers should only be accessible to authenticated backend admins. - Done
    7.3. User data should be deleted from the backend servers if the user deletes their account. - On Track
8. Legal, marketing, copyright
    8.1. Application should display the disclaimers, copyright of the advertisers

while advertising on the application. - Issue, currently there are no advertisements on the application.
8.2. Application should use only open source images in the web application. - Done

7. List of Contributions

Name	**Role**	**Contribution**
Soujanya Ravindra Nayak	Team Lead Document Contributor Backend Team	● Organized meetings and divided tasks among team and followed up on the tasks ● Revised M3 V2 with feedback on M3 V1 ● Contributed to summary and QA testing ● Implemented backend of events for our superior feature ● Collected user survey
Margaret De La Torre	Front-end lead Document Contributor	● Performed QA testing ● Implemented frontend based on feedback of M3 ● Collected user survey
Martin Salvatierra	Front-end Document Contributor	● Performed QA testing ● Implemented frontend based on feedback of M3 ● Collected user survey
Andy Cho	Back-end lead Document Contributor	● Contributed to code review ● Implemented backend of groups for our superior feature
Qin Geng	Front-end Document Contributor	● Performed Usability testing ● Implemented frontend based on feedback of M3 ● Collected user survey
William Plachno	Git Master Backend Team Document Contributor	● Contributed to QA testing ● Contributed to Self check on security ● Implemented backend of use verification for our superior feature ● Contribute to code review ● Collected user survey
Victor	Front-end Document Contributor	● Performed usability testing ● Contributed to code review ● Implemented frontend based on feedback of M3