DM

Combining the Power of Online Crowdsourcing With Creative Writing User Interfaces

MASTER DISSERTATION

Igor Diogo Silva Sousa
MASTER IN INFORMATICS ENGINEERING

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

January | 2016

Combining the Power of Online Crowdsourcing With Creative Writing User Interfaces

MASTER DISSERTATION

Igor Diogo Silva Sousa
MASTER IN INFORMATICS ENGINEERING

SUPERVISOR
Pedro Filipe Pereira Campos

CO-SUPERVISOR
Frederica Margarida Camacho Gonçalves

Abstract

The goal of this work was to provide professional and amateur writers with a new way of enhancing their productivity and mental well-being, by helping them overcoming writers block and being able to achieve a state of optimal experience while writing. Our approach is based on bringing together different components to create what we call a *creative moment*. A creative moment is composed by an image, a text, a mood, a location and a color. The color presented in the creative moment varied according to the mood that was associated to the creative moment. With the creative moments we hoped that our users could have a way to easily trigger their creativity and have a kick start in their work. The prototyping of a web crowdsourcing platform, named CreativeWall, and a Microsoft Word Add-In, that was used on the user study performed, is described and their implementations are discussed.

The user study reveals that our approach does have a positive influence in the productivity of the participants when compared with another existing approach. The study also revealed that our approach can ease the process of achieving a state of optimal experience by enhancing one of the dimensions presented on the Flow Theory. At the end we present what we consider would be some possible future developments for the concept created during the development of this work.

Acknowledgements

At first, I would like to thank my co-advisor Frederica Gonçalves without whom, this work would not produce the results it did. Many thanks for being always available for answering any questions, and helping me with everything I needed, from advising about the writing of this document to helping me with the user study and the definition of the concepts. Despite all the setbacks she has never given up and always tried to see the brightest side of things.

I would like to thank my advisor Pedro Campos for helping me define the concept of this thesis, for helping me with the related research and for always being available for clarifying any questions that I've had along the way.

I would like to thank all the people who helped me by participating in the user study which was an important part of this work.

I would also like to thank my friends and my girlfriend for always being there to listen to my problems and concerns.

Finally, I would also like to thank my family for supporting me all the way through my college by helping me with everything I needed.

Contents

Figures

Tables

1. Introduction

This first section describes all the motivations and goals of this work, as well as all the contributions that we expect to give to the community interested in the purpose of this thesis. This community should include everyone that is interested in writing and wants to share their writings in a way that can help other writers.

1.1. Overview

Sometimes writing might not be as easy as it looks, there are millions of subjects a writer can write about, thousands of ways to transmit the same idea, hundreds of tools to use to help them write. Deciding what to write about can sometimes be really tricky. The Internet provides lots of tools that can help on this matter, but it's really hard to find one tool that really helps without distracting writers from their goal. Search engines like Google can help you find ideas on what to write about but it takes a lot of time to view the thousands of results that are presented, even though they are shown in an optimized way. Social networks like Facebook or Pinterest can help you as well with groups that are created with the objective of helping writers, but they lack categorization and can be really distracting. The best way to solve this issue is to build a tool that is categorized, fast, usable and that can be integrated with the software that you usually use.

The best way to gather enough data for this tool is to take advantage of crowdsourcing. This way we can create a community who is interested in writing and give them the opportunity to share their creative writings. These writings will then be used by the tool to help other writers with their work.

One of the problems that usually drive writers away from their goals is writer's block. Writer's block is a problem that affects every writer more than once in their careers and is characterized as the inability to write, despite the desire and ability to do so. One of the methods used to avoid the writer's block is using writing prompts, which consists on a small text that is supposed to help writers have ideas on what to write. This method has been used for a long time now and has been proven to work in the majority of cases. We decided to take this method and refine it by adding some more components that can help the writer have the creativity he needs. By using images, moods and locations we create a more visual perspective of a moment that can bring creativity to the writer.

1.2. Goals and Motivations

The motivation for this work is the fact that most writers suffer and see their production reduced due to issues with creativity and not being able to overcome writer's block right away. According to Rose M. [1] writer's block can last for whole days and become a real source of frustration when people are anxious about deadlines and really need to get the writing done. This fact is one of the triggers of this work as it is a topic that has not yet been researched deeply enough, and current solutions for this problem seem to be somewhat scarce and ineffective.

One of the goals of this work is to develop an approach that can help writers overcome writer's block and enhance their productivity. With the user study and functional prototypes that serve as outputs for this work we will be able to understand whether or not our approach can become a viable solution for the writer's block problem. We hope we can create a solution that can be integrated with other existing tools, such as word processors (e.g. MSWord or Haven).

Another big motivator was the fact that to be productive, a writer must be in a state of deep concentration, as happens in any other activity where some kind of skill is required. To facilitate the achieving of this state of concentration and help the writer have an optimal experience while writing, a tool that is focused on this aspect is required. And so, another one of our goals is to build a tool that is proven to be helpful in this aspect.

1.3. Contribution

We have multiple goals to achieve with this work and with this give contribution in more than one area of research. With this work we expect to contribute with:

- A new approach that helps professional and amateur writers overcome writer's block;
- A new way to help professional and amateur writers enhance their productivity and mental well-being, by helping them have an optimal experience;
- An API that can be used to retrieve data in order to implement our approach for any word processor software in the market in the form of a plugin;
- A crowdsourcing platform where writers can keep their creative writings and share them with a community with the same interest creating an interactive way for users to develop their abilities in writing;
- A Microsoft Word Add-In that helps writers trigger creativity while writing (to be proven by the user studies);

1.4. Thesis Outline

This thesis is divided into six chapters, which are described below:

Chapter two, the state of the art, describes all technologies, concepts and studies that served as a base for this work. The first concept described is crowdsourcing. An attempt to define the concept of crowdsourcing is shown, as well as some big projects that have been developed in this area. The second aspect discussed in this chapter is about color, moods and emotions. Their relations, how color can lead to an emotion and how an emotion can influence creativity. This served as a base for a feature presented in our prototypes described in chapter three. Also, some concepts on creative writing are presented in this chapter. The concept of writers block and one possible solution for it are discussed as well. Then, the Flow Theory is discussed and the all its dimensions analyzed briefly. Another aspect presented in this chapter is usability and how can usability be evaluated. We then use this knowledge to perform a usability evaluation on one of our prototypes. Finally, social desirability bias is characterized. This is a problem that could victim our user study and so there were some aspects that had to be avoided.

Chapter three describes in detail each of prototypes that were implemented. All its features, implementations, design decisions, etc. Also, the usability evaluation is explained and its results reported. Another aspect worth of mentioning in this chapter is the software modeling that was performed for one of the prototypes. This software modeling includes requisites analysis and a use case diagram. Finally, all the technologies used in the implementation of the prototypes are briefly described. The most important parts of the code implemented are attached to this work in the appendices.

Chapter four presents the user study performed using the prototypes described in chapter three. In this user study, eleven participants were asked to write three texts using different contexts and different tools. All the procedures, settings and participant details are described in detail.

Chapter five focus on the conclusions taken from the user studies. In this chapter all the quantitative and qualitative results are analyzed and compared with the initial assumptions.

Chapter six describes the future work that should be undertook for further development on results and on the quality of the prototypes developed during this work. This includes the redesign of the prototypes to make them accordingly to the usability evaluation.

2. State of the Art

In this section we will discuss the theoretical aspects of this work, taking as a reference articles that were published to date and show some kind of proof to support their findings. The topics discussed are crowdsourcing, moods, colors and emotions, creative writing, flow theory, usability and social desirability bias. These topics are the background for this work and helped a lot in bringing the work together.

2.1. Crowdsourcing

Crowdsourcing is a concept that, although it is very powerful and effective, it is quite recent and still doesn't have a solid theoretical knowledge base that allows it to have a clear definition [2]. According to Howe J. [3] crowdsourcing can be defined as "the act of taking a task traditionally performed by a designated agent and outsourcing it by making an open call to an undefined but large group of people". Another example of a definition comes from Brabham [4], who says it can be defined as "a new web-based business model that harnesses the creative solutions of a distributed network of individuals through what amounts to an open call for proposals". As for Kleeman et al. [5] crowdsourcing can simply be defined as "the outsourcing of tasks to the general internet public". These are all valid definitions but there is no consensus regarding what would be a definitive and complete definition. An effort for finding this kind of definition has been made by E. Estellés-Arolas et al. [2], and as a result they state that "Crowdsourcing is a type of participative online activity in which an individual, an institution, a non-profit organization, or company proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task. The undertaking of the task, of variable complexity and modularity, and in which the crowd should participate bringing their work, money, knowledge and/or experience, always entails mutual benefit. The user will receive the satisfaction of a given type of need, be it economic, social recognition, self-esteem, or the development of individual skills, while the crowdsourcer will obtain and utilize to their advantage that what the user has brought to the venture, whose form will depend on the type of activity undertaken". According to the authors, this is a complete definition that covers all crowdsourcing types that have been identified in their study.

According to Kleeman et al. [5] there are various types of crowdsourcing:

- Participation of consumers in product development and configuration – calls for participation in product design and configuration;

- Product design – product is designed solely by the users;
- Competitive bids on specifically defined tasks or problems – users are paid for completing tasks or solving problems;
- Permanent open calls – same as previous, but there are no specific tasks or problems;
- Community reporting – users that belong to a community of registered users are asked to report to the platform about topics of interest;
- Product rating by consumers and consumer profiling – publishing of consumers knowledge and opinions about products;
- Customer-to-costumer support – creation of discussions where multiple users create and reply to a variety of topics of interest.

For the purpose of this work we will only focus on two types, product design and product rating by consumers and consumer profiling, which are the ones that most suit our needs. As for the product design, it is clear what the idea is, the users design the product entirely and it is then sold or used by the entity in order to deliver it to the end user. Usually if these products are sold, the user who builds it receives a percentage of the profits. Otherwise, the users must be given some motivation that makes them keep creating products. In the product rating by consumers and consumer profiling, it's all about getting users to rate products in order to guarantee quality and create user profiles that help in the marketing area, making users consume a bigger quantity of products.

2.1.1. Amazon Mechanical Turk

AMT (Amazon Mechanical Turk) is a crowdsourcing system used on a wide variety of areas (literature, natural language processing, human-computer interaction, etc.) [6]. The users who work on AMT (called Turkers) are paid small amounts of money for completing HIT (Human Intelligence Tasks) which are small tasks posted by employers. The HITs are usually composed by surveys about image categorization, text translation, etc.. Each HIT can yield around 0.10$ to a Turker.

In 2008, Rion Snow et al.[7], did a very interesting experiment using AMT to determine whether non-expert labelers could provide reliable natural language annotations. To do this, they chose five natural language understanding tasks that they felt were sufficiently natural and learnable for non-experts. For each one of the tasks, they already had labels created by experts, which served as a base of comparison among expert and non-expert labelers. The results of this experiment were very positive, referring that for the majority of tasks only a small number of AMT workers annotations were needed to equal the annotations of the experts meaning that AMT workers can produce a larger amount of correct annotations per hour.

The above example shows the power of AMT and their Turkers, but the problem is that although it works for some cases, there are others that are not so positive. For example, Aniket Kittur [8] states that if the HIT design is not appropriate there can be cases of gaming and low

quality responses that generate very bad results from the HITs. To prevent this, Kittur proposes the use of simple questions (e.g. number of references in a document, number of images, etc.) in order to obligate the user to spend some time on the task, reducing the response gambling percentage.

2.1.2. Galaxy Zoo

Galaxy Zoo is a crowdsourcing platform where users are asked to classify galaxies based on their shape, number of rings, etc.. A counting made in 2009 showed that over 200.000 volunteers have classified more than 100 million galaxies and in 2010, more than 50 research projects were developed with base on the results recorded to the date. From these 50 research projects resulted 16 articles accepted or submitted to peer-reviewed journals [9]. According to the same reference, the biggest motivation for users to do this work was the interest in Astronomy, which may induce the idea that the users related to the topic of the platform are more interested in participating in such activities.

2.1.3. Innocentive

Innocentive is a crowdsourcing platform that offers money prizes to users (Solvers) who can solve challenges posted by companies (Seekers). These challenges aim to generate new ideas for innovative solutions to problems that arise every day. Usually, what the solvers must do is to submit a written document describing the solution and wait for a result. At the end of the deadline, the Seekers review all the submitted answers in order to choose the ones that suit their needs the most. After that they pay the prize to the solver. These prizes can go up to millions of dollars, depending on what the Seeker is willing to pay for the solution of the challenge. Innocentive then takes a commission called “finder’s fee” that equals 40% of the prize the solver gets [10]. According to the records registered on the Innocentive’s website, they have over 365000 solvers registered from over 200 countries, over 40 million dollars posted and have a success rate of 85% in finding solutions for premium challenges.

2.1.4. Conclusions

The previous examples are very interesting because they show different forms of crowdsourcing and give us an idea of the power this concept really has and the vast amount of areas where it can be applied. It’s interesting to see that the motivation for the users to perform such work does not always come from financial profit, sometimes it comes from intellectual profit or even just from the fun that a user can have when doing something from his area of interest. As for this work, the motivation we are looking to give the crowd is the joy of having a community with the same interests and that can help evaluating the pictures and texts that the users create on our platform.

2.2. Moods, Colors and Emotions

By examining the current state of the art on this subject we try to get base knowledge on how to use moods and colors to generate emotions onto an individual in order to enhance his/her creativity. To do this, we must examine some research works made in these areas to acquire results that can be applied in our own studies. First we will relate moods and emotions to understand how it is possible to enhance creativity through these two factors. After that, we will study the relation between color and emotions and wrap everything up on a third conclusive phase which gives us a mapping between color and emotion that aims to embed these emotions into the users.

2.2.1. Moods and Emotions

As Matthijs Baas et al. [11] states, there are differences between moods and emotions. According to the reference, moods are long lasting while emotions are more related towards a specific stimulus, for example, an emotion would be a person feeling happy because he/she found some money on the ground. This situation forces an emotion onto the person, the emotion of happiness. On the other hand, a mood is something that is more general, for example, a person feeling happy because he/she just feels great. With this we can conclude that there really are differences in terms of intensity of feelings, being that emotion is generally stronger than a mood. Another definition of mood states that moods are the accumulation of emotions and other affective events [12].

Moods can have multiple dimensions, but only three of those dimensions are proven to be related to creativity. They are hedonic tone, activation level and regulatory focus. We will now describe briefly each of these dimensions, relating them with creativity.

2.2.1.1. Hedonic Tone

The hedonic tone, or valence, simply put, describes whether the mood is positive or negative [11] [13] (e.g. happiness has a positive tone while anger has a negative tone). This dimension is usually related to creativity as some studies refer [14]. The same studies state that moods with a positive tone help a subject produce more original word associations which means that there might be a boost in creativity.

2.2.1.2. Activation Level

Activation relates to whether or not the mood can generate active behaviors in the subject (e.g. calm is a deactivating mood while fear is activating). According to Carsten De Dreu et al. [13] activation is a necessary precondition for creativity to come by while hedonic tone determines the route through which creative fluency and originality is achieved. They also argue that activating moods are more likely to generate creativity than deactivating moods.

2.2.1.3. Regulatory Focus

Regulatory focus refers to the motivation an individual has to complete a task. According to studies made in this area, regulatory focus plays an important role in triggering creativity [15]. There are two types of regulatory focus, the promotion focus and prevention focus. Promotion focus comes from the desire of accomplishing something, while prevention focus comes from the will of securing something [11]. A good example of this would be the attackers and goalkeepers in a football game. The objective of the attackers is to score goals and that's their motivation for playing (promotion focus). On the other hand, goalkeepers want to prevent goals, and that's their motivation for playing (prevention focus).

2.2.1.4. Hedonic Tone, Activation Level, Regulatory Focus and Creativity

Table 1 shows a small list of moods categorizing them with the hedonic tone, activation level and regulatory focus, according to Baas, et al. [11].

Positive Hedonic Tone		**Negative Hedonic Tone**		
Deactivating	Activating	Deactivating	Activating	
Prevention Focused	Promotion Focused	Promotion Focused	Prevention Focused	Promotion Focused
Calm Serene Relaxed	Happy Upbeat Elated	Sad Discouraged Disappointed	Uneasy Tense Fear Disgust	Angry Frustrated

Table 1 - List of moods according to the dimensions they fit in

The studies appoint that the moods that actually affect creativity are the activating, promotion focused moods (happy, upbeat, elated, angry and frustrated) while the activating, prevention focused moods (uneasy, tense, fear and disgust) tend to diminish creativity. Also, moods that are deactivating usually don't have any effect on the creativity of an individual, meaning that only activating moods are related with higher levels of creativity [11].

Even though moods and emotions are different, as stated before, they share the dimensions that were identified in the above conclusions and so they should be considered the same in terms of hedonic tone, activation level and regulatory focus, which gives us an idea of what emotions can actually work on enhancing creativity and help us developing our platform.

2.2.2. Color and Emotions

Color can lead to feelings, and that can be proved by several studies in this area [16] [17]. A good example of this statement is a cloudy day. A cloudy day has a predominant color, which is gray, and gray is related to sadness, so people usually feel sad on cloudy days. If you look outside through a yellow window you can see that a feeling of warmness comes to you and

everything feels a bit happier. This is described by Goethe [18] in his book called "Theory of Colours". Even though the book was published more than a 100 years ago, many people still follow some of the theories described in it. And it's curious to see that the relation between color and emotions (or feelings) was already object of research at that time.

Naz Kaya [16] el al. [17] performed user studies in order to be able to map colors to emotions and the result were somewhat similar to the statements made in the "Theory of Colours".

2.2.3. Conclusions

After considering the statements above, a list of moods and a mapping to colors was produced. Table 2 shows the mapping between color and emotion.

Happy	Yellow	
Hopeful	Green	
Excited	Orange	
Energetic	Light Blue	
Loved	Pink	
Fearful	Black	
Angry	Red	

Table 2 - Moods mapped into colors

Even though fearful and angry are not considered to be creativity enhancing moods, they were included in the list in order to give users some variety to choose from. As future work an increase to the number of moods should be granted in order to include other creativity enhancing moods.

2.3. Creative Writing

Creative writing can be found in a huge variety of areas (e.g. journalism, science fiction, etc.) and it can be expressed through different forms (e.g. prose, poetry, etc.) [19]. There are also lots of techniques that can be applied in order for the writer to be able to write efficiently and with quality. Usually, writers try to take their creativity from anything that they get a glimpse of, even dreams or television news reports, in a way in which they can imagine new characters and situations that can be included in their writings. According to some writers, a good idea is to keep diaries of everything you see, ear or think, that can actually serve a purpose for your writings [19]. Another one of the main principles for being productive and efficient is the creation of habits. Writers tend to develop habits that can't be broken, like writing at night when everyone else is asleep or having a place specifically to write [20].

Even though all the previous techniques are applied, sometimes, and everyone who writes regularly knows this, it just seems like we can't write a single line. This condition is usually called writer's block and we will discuss the term in the next section.

2.3.1. Writer's Block

Writers refer to writer's block as the inability to write, despite the desire and ability to do so. There are different reasons for this situation to happen. Aspects such as stress, fear or simple problems with organization or prioritization can be the cause for it [20]. There are currently some possible solutions to this issue and each writer seems to have its own way to deal with it. Some writers state that if you keep writing everything that comes to mind, things will start flowing and you will be able to write, eventually. Other writers claim that writing prompts might help. We will analyze the second option next, the writing prompts.

2.3.2. Writing Prompts

Writing prompts are simple phrases meant to help writers trigger their creativity and start writing fluently without losing their time with a creative block, not being able to write a single line of text. One example of a writing prompt might be:

"After a short walk you can see the top of the towers of a big dark castle. Do you think someone still lives in there?"

This kind of short text might be the start of a story, and might help the writer initiate what could be a new best seller novel.

2.3.3. Conclusion

Even though some expert writers say that writing prompts are better suited for beginner writers (and this comes from an interview with an experienced writer performed at the beginning of this work), we think it might be an important tool for everyone who writes.

There are some software tools that give you writing prompts created by users but they lack categorization so all the prompts you get are about random topics, and that, in our opinion, seems to be the main problem with the platforms that currently exist.

In this thesis we try to find a new solution by coupling writing prompts with images and colors and creating what we call a creative moment. This creative moment can help a writer unblock his creativity and start writing. The fact that to write fluently you need to write, means that, sometimes, the hard part is to start writing and so our approach is designed to help with that hard part of writing, the very beginning.

2.4. The Flow Theory

Csikszentmihalyi [21] who has been studying the states of the optimal experience for more than 20 years was the author of the Flow Theory. According to him, what makes an experience genuinely satisfying is a state of consciousness called flow. The flow is a state of concentration in which the sense of time and emotional problems seem to disappear and the individual experiencing it feels in total control of the situation and in the peak of his ability.

Csikszentmihalyi identified eight major components that define a positive experience for an individual:

1. One is confronted with tasks which he as a chance of completing;
2. One must be able to concentrate on what he's doing;
3. The task has clear goals;
4. The task provides immediate feedback;
5. One acts with deep, but effortless involvement, that removes from awareness the worries and frustrations of everyday life;
6. One exercises a sense of control over their actions;
7. Concern for the self disappears, yet, paradoxically the sense of self emerges stronger after the flow experience is over;
8. The sense of duration of time is altered.

For the purpose of this work we will focus only on four out of the eight components listed above.

2.4.1. The merging of Action and Awareness

When a challenge takes all of an individual's relevant skills to complete the individual becomes so involved in what he is doing that the activity becomes spontaneous and the individual stops being aware of himself as separate from the actions he's performing. To be in the flow, disciplined mental activity is required and any lapse in concentration will break it. [21]

2.4.2. The Paradox of Control

When in the flow, an individual acts as if he is in control, even though the control may not be real. The feeling of control is generated by the lack of sense of worrying derived from the involvement in the activity at hand. [21]

For some activities the sense of control might be irrelevant, for example games of chance. People rely on random events for the success of the tasks, and therefore no control can actually be exercised from the individual. Also, this sense of control has a very important negative aspect, the addiction. Most of the activities that can make an individual feel in the flow, can become addictive because of this sense of control that people feel when performing them. Taking an example from the subject of this work, writing can become an addiction for some people. When writing people feel like their world is the one in their writings and not the real one and so it becomes so addictive that people forget to eat or sleep. [21]

2.4.3. The loss of Self-Consciousness

This is one of the most important and relevant aspects of the flow. When in the flow, people often lose consciousness of who they are, consequently being able to reduce the distractions that he/her will pay attention to. For example it is not relevant whether a climber's name is

John when he's climbing a mountain. This feeling can lead to self-transcendence and feeling like we're part of a bigger thing, like when driving a car, for example, often people feel like the steering wheel and the pedals are an extension of their body making it easier to perform the task of driving. [21]

During the flow, an individual doesn't realize that he is developing his skills as he has not consciousness of that but after the activity has ended the self-consciousness has time to resume and note that something has changed, that some skills have been improved. This makes us understand that a sense of loss of self-consciousness is needed in order to improve the self. [21]

2.4.4. The Transformation of Time

When an individual is immersed into an activity time seems to move differently than the normal conventional clock. According to Csikszentmihalyi it can either seem to go faster, but it can also seem to go slower. There are exceptions to this as to some activities time is crucial and having a different perception of time could mean the failure of the activity. Even though the time aspect of an activity is not very important for the enjoyment of the activity it contributes to the joy people feel during the flow. [21]

2.4.5. Conclusions

The reason why the flow theory will be important for this work is that in order for a writer to be able to be creative and write original contents they must be in a state of deep concentration, and for this the solution presented for enhancing creativity must immerse the writer into the flow state. In the user studies that is one of the dimensions that will be evaluated.

2.5. Usability

2.5.1. Characteristics

Usability sets five essential characteristics to which interface designers should always pay attention. These characteristics are:

- Learnability – refers to the speed with which the users are able to begin working with the system;
- Efficiency – measures the speed and quality with which the system enhances the user's productivity;
- Memorability – makes it possible for the user to return to using the system without having to relearn everything from scratch;
- Low error rate – prevents users from committing errors that are hard to rectify;
- Satisfaction – which makes the users experience pleasant.

It's hard to achieve the peak of all these characteristics without making sacrifices, and so there are tradeoffs that must be performed in order to get the most out of the characteristic that best suits the system that is being implemented [22].

2.5.2. Usability Evaluation Techniques

A study made in 1991 [25] compared four of the most used usability evaluation techniques. The techniques used for the comparison were empirical usability testing, heuristic evaluation, cognitive walkthrough and software guidelines. We will now describe each of these techniques.

2.5.2.1. Empirical Usability Testing

This kind of usability evaluation is performed by testing the interface. This is usually done by real users that are recruited to specifically do this kind of work. The problem with this technique, although it has been used worldwide for a long time, is that it can be really expensive as every single user has to be paid in order to perform the required work. Also, in some projects with tight deadlines, there is no time to make all the required arrangements for users to perform the tests [26].

The output of this method is a list of usability problems found by each of the users, reported by the responsible for the tests [23].

2.5.2.2. Heuristic Evaluation

Heuristic evaluation is a usability engineering method with the objective of finding usability problems in each phase of the development of a product. This method involves having a small set of evaluators examine the interface and judge it according to a set of usability principles. An example of a set of principles that is usually used in this technique are the ten heuristics designated by Nielsen [24].

Nielsen J. [25] designated ten heuristics which are used for heuristic evaluation. The ten heuristics are:

1. Visibility of system status – the user should always be informed about what's going on;
2. Match between system and the real world – the system should speak the users' language, with words, phrases and concepts that are familiar to the user;
3. User control and freedom – give the users an intuitive way to rollback their actions;
4. Consistency and standards – follow conventions;
5. Error prevention – try to minimize users errors by implementing features in a way that keeps the user from committing mistakes;
6. Recognition rather than recall – users should not have to remember things in order to be able to perform one action that they have performed before, they should recognize them instead;

7. Flexibility and efficiency of use – ways of accelerating the usage of the system should be implemented in order for the expert user to be able to perform frequent actions faster;
8. Aesthetic and minimalist design – only relevant information should be shown;
9. Help users recognize, diagnose and recover from errors – error messages should be intuitive and helpful, and not contain error codes;
10. Help and documentation – the system should provide help to the user whenever it is needed.

The evaluation starts by having each of the evaluators inspect the interface. After all evaluations are completed, they are aggregated. It is important that evaluations are done separately in order to ensure independent and unbiased evaluations. An observer can be used to record all findings and assist the evaluators in case any problem is found in the interface operation. Contrarily to the empirical tests, where the observer can't help the user with anything more than the strictly necessary, the observer must provide evaluators with hints on how to proceed or any information they need, especially if the evaluators don't have knowledge on the domain of the platform where the interface is implemented. This should only be done when the evaluator cannot proceed or has already commented on the usability problem in question [24].

The time for each session should be one or two hours, even though it can take longer for larger or more complex interfaces, which should be divided into smaller sessions [24].

During the session, the evaluator should go through the interface several times and inspect all the elements, comparing them with the selected set of principles (heuristics). The evaluator is free to add any other heuristics that seem fit [24].

The output of this method is a list of usability problems with reference to the usability principles that are infringed. All problems should be described in a very detailed way, and should be all described separately. In addition to the list of usability problems, a severity can be associated to better describe the problem. These severity ratings help the development team deciding which problems should be solved first [24].

2.5.2.3. Cognitive Walkthrough

Cognitive walkthrough is a usability evaluation method that evaluates the ease of exploratory learning on an interface based on a cognitive model of learning and use. This process has two phases, the preparatory phase and the analysis phase [26].

On the preparatory phase, the guideline of the task is defined. Aspects like interface to be used, users to perform the task and time to complete it are defined in this phase [26].

For the analysis phase, the users walk through the four steps of human-computer interaction developed by Lewis C. and Polson P. [26] [27]:

1. The user sets a goal to be completed within the system;
2. The user determines the currently available actions;
3. The user selects the action that he thinks will take him closer to his goal;
4. The user performs the action and evaluates the feedback given by the system.

This process can be used to help developers find actions that can lead to errors in safety-critical interfaces. By having the evaluator find these actions, developers can then change the interface in order to prevent this kind of interface misunderstandings. One of the strong points of this method is that it can be applied, not only by usability experts, but by novice users as well. The biggest adversity the novice users can find is the definition of the usage scenario, and if this is not set adequately the analysis can be ineffective [26].

The output of this method is a list of discrepancies between the user's expectations and the steps that are required to achieve a goal [23].

2.5.2.4. Guidelines

This method is the simplest one and requires no usability specialists to apply it. It can be applied by software developers just by following certain guidelines published by usability specialists. These guidelines are applied during the interface implementation and helps developers decide, for example, where to put contents on screen or how to organize the items of a menu [23].
The output of this method is the interface itself as usually there are no reports written during the application of this method [23].

2.5.2.5. Conclusions

The study made by Jeffries R. and Miller J. [23] shows that, heuristic evaluation found more usability problems than any other of the three techniques that were tested. According to them, heuristic evaluation was able to find more than the double of the usability problems when compared with each of the other techniques. In terms of severity, the empirical usability testing managed to find a higher average severity problems.

For the purpose of this work, a usability test will performed by the author to one of the prototypes. Taking into account the results presented in this state of the art and due to its superiority when compared to the other techniques, the heuristic evaluation will be the technique applied. Although it should be applied by a group of usability experts, in this case, only a demonstration will be made and taking that into account, there will be only one evaluator, even though he will is not a usability expert.

2.6. Social Desirability Bias

The Social Desirability Bias is described as the pervasive tendency of individuals to present themselves in the most favorable manner relative to prevailing social norms [26]. This is a

problem that affects self-reported surveys and requires due attention in order to minimize its effects. An important thing to do in the surveys to prevent social desirability bias is to keep the confidentiality of the individuals filling the survey [26]. This makes them feel like their responses won't be directly related to them and makes them feel more comfortable with giving negative responses.

In the user studies designed for the purpose of this work, these rules shall be applied as there must be some kind of assurance related to the answers given by the users.

2.7. Summary

In this chapter, a revision of all the existing technologies/approaches was made for everyone to understand all that has been done before in all the areas of interest in the scope of this work. All the state of the art presented in this chapter was somehow used in the building of the prototypes or in the definition of the concept for this work.

The crowdsourcing concept and its definition were discussed. A variety of crowdsourcing platform examples were analyzed and some theory about color, emotions and moods, creative writing and the flow theory was described in order to provide support to the concept of this work.

The last part of this chapter talks about usability and describes some evaluation methods that can be used in order to enhance user interfaces.

The next chapter will focus on the prototypes that were built, its features, implementation, design decisions and usability evaluation performed to one of the prototypes.

3. Prototypes

In the next chapter we will discuss all the prototype building, design decisions and technologies used along the way. We also show some illustrations about how the prototypes turned out to be, show all the features implemented and explain the reason why they exist. Also a usability evaluation and its results will be presented.

3.1. Creative Wall

3.1.1. Concept

CreativeWall is a crowdsourcing platform where its users can share their creative writings along with images, locations and moods, creating what we call a creative moment. This concept of creative moment came to life from the idea that people sometimes have ideas for creative writings when they are, for example, walking on the street and see something that triggers their creativity. That is the essence of the creative moment. They have an image that triggers an idea that is described by the text. A certain mood is also associated to that moment and it happens in a certain location, date and time. When brought together, all these aspects generate our creative moment. Figure 1 illustrates how a Creative Moment is shown on the platform.

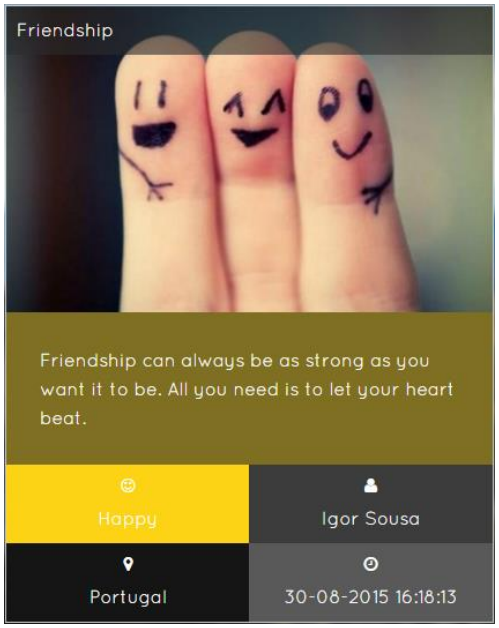

Figure 1 - Example of a Creative Moment

In this case the creative moment was captured in Portugal when the user was Happy. The moment has author, date and time information for a better knowledge about its details and the environment where it happened. With this kind of information, other users can recreate the moment mentally and maybe absorb some kind of creativity from it giving birth to their own ideas based on what they see and feel with the moment recreation.

3.1.2. Features

There are two main sections on the platform, the Creative Wall and My Posts. Figure 2 shows the top menu and the sections that are represented in it.

Figure 2 - Top menu showing both sections available

3.1.2.1. Creative Wall

In this section users can check out creative moments shared by other users. They can also report, flag the moments as well written, or rate them. The first option, the reporting of moments, allows users to report other users moments for offensive content, copyright violation, etc.. As for the second option, the well written flag, it allows users to flag the moments that have a correct syntax and semantics, and with this make them part of the moments that appear when a user selects the well written filter. The last option, the moment rating, allows users to rate a moment from 1 to 5 stars where 1 means very uncreative and 5 means very creative. By doing this users make shared moments more and more relevant. These three options are part of a very important component of a crowdsourcing platform, the quality control, and can only be accessed by users that are logged in the platform. Below we can see these three options and how the user interacts with it. Figure 3 shows the three options described above.

Figure 3 - Top bar of the moment showing the three options available

In this case the moment has already been rated, but is not checked as well written. Figure 4 shows how a user can interact with the system in order to rate a moment.

Figure 4 - Top bar of the moment showing the rating feature

We can see that the user rated the moment with two stars. This means that the moment was uncreative and should not appear on the top of the relevant moment's list.

Another important feature that was implemented is a filter system where users can insert the conditions that most suits them and the platform will look for the moments that match those conditions. Figure 5 shows the interface implemented for this feature.

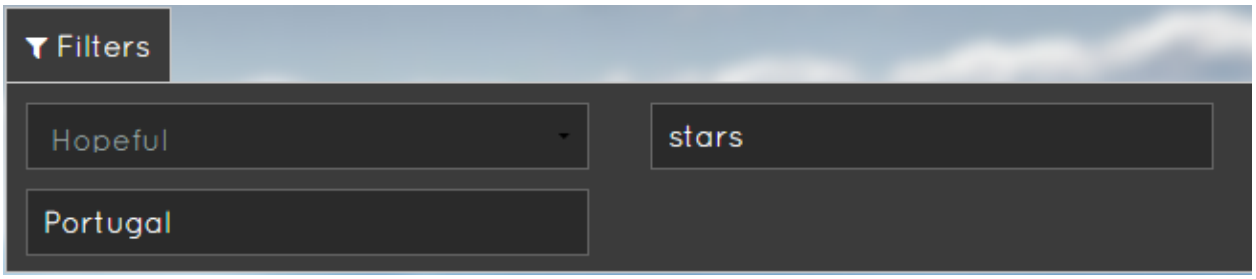

Figure 5 - Filters bar showing the three available filters

There are three fields on the filters bar, mood, location and tag. These three fields help users categorize their searches for faster finding of the moments they want. Each moment can be associated to a group of tags that can then be used for search purposes. There is also an option for sorting the results for a variety of options. Figure 6 shows the list of options that are available for sorting.

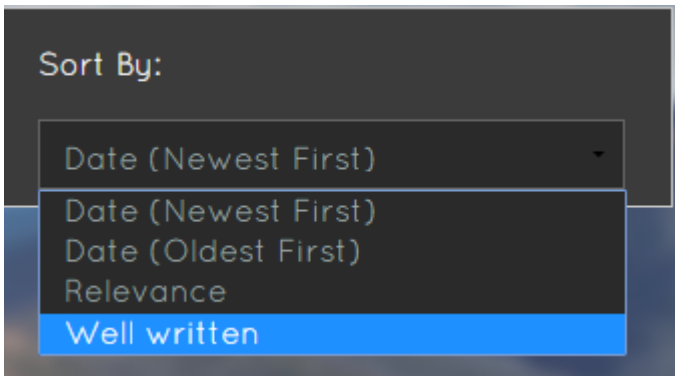

Figure 6 - Sort feature showing the options available

For the first two options, they are self-explanatory. On the relevance option, the moments that are presented first are the ones with a higher average rating. This average is calculated by dividing the sum of all the ratings by the number of ratings assigned to that moment. As for the well written, the first moments that are presented are the ones with a higher count of well

written flags. With this feature, it is possible to make sure that only quality content is shown and that the user does not have to pay extra precious attention to error check on moments shared by other users.

3.1.2.2. My Posts

In this second section, users can see all the moments that they have already created, share them, add new ones and edit or remove the existing ones. Users can choose whether or not they want to share their moments. They can use the platform just for saving their moments without making them available for other users to see. To share a moment, users just have to press the share icon and the moment is made available for every user. To edit or delete a moment, users just have to press the according icon. Figure 7 shows these three options and how the user interacts with them.

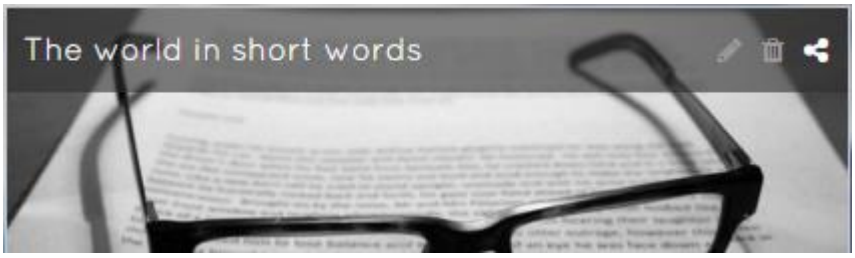

Figure 7 - Top bar of the moment showing the three options available

In this case, this moment is shared, as the icon for sharing is active. When a user deletes a moment a confirmation dialog box is presented to make sure that this is the desired action. As referred before, a user can also create a new moment in this section. When the user presses the button to add a new moment, a dialog box is presented with all the fields necessary to create the new moment. Figure 8 shows all the information the user can insert to create the new moment.

Figure 8 - Dialog box showing the possible fields for creating a new moment

All fields are required, except for the image. For the purpose of this work, and due to hosting costs, the image cannot be directly uploaded on the platform. It must be uploaded externally and the user must use the link generated on that external source to fill the image field presented on the dialog. The rest of the fields are self-explanatory. After a moment is created, a color is associated to it depending on the mood chosen. The relation between colors and moods is described in the state of the art chapter.

3.1.3. Software Modeling

3.1.3.1. Requirements analysis

Before the implementation of the prototype, a brief brainstorm resulted in a requirements analysis that helped defining the characteristics that the crowdsourcing platform should have.

The functional requirements describe all the functionalities to be implemented on the platform. Table 3 shows the functional requirements that were identified.

ID	Name	Description
1	**User Management**	
1.1	Sign up	Users must be able to sign up on the platform
1.2	Sign in	Users must be able to sign in on the platform
1.3	Password change	Users must be allowed to change their password
1.4	Password recovery	Users must be allowed to recover their lost password
2	**Content Management**	
2.1	Create content	Users must be able to create their original content
2.2	Edit content	Users must be able to edit their original content
2.3	Remove content	Users must be able to remove their original content
2.4	Share content	Users must be allowed to share their original content
2.5	Keep content without sharing it	Users must be allowed to not share their original content
3	**Search Management**	
3.1	Filter content	Users must be able to search for content shared by other users in order to find the content that has the most interest to them
3.2	Sort content	Users must be able to sort the content shared by other users in order to find the content that has the most interest to them
4	**Content Quality Management**	

4.1	Rating other users content	Users must be allowed to rate other users shared content in order to keep a quality control and allow users to search for quality content only
4.2	Mark other users content as well written	Users must be allowed to mark other users shared content as well written in order to keep a quality control and allow users to search for quality content only
4.3	Report other users content	Users must be allowed to report other users shared content in order to prevent users from infracting copyright rules and prevent offensive language

Table 3 - Functional requirements for the CreativeWall platform

As for the non-functional requirements, they represent the general characteristics that the platform must have in order to give the users a good experience. Table 4 shows the list of non-functional requirements that were identified.

Category	Description
Performance	The platform must respond quickly in order to allow users to find the content they need as quick as possible
Usability	The platform must have a usable interface in order to facilitate users navigation and usage of functionality
Security	The platform must be secure in order to keep users original content from being stolen
Portability	The platform must be usable in all kinds of devices with a browser installed (e.g. smartphones, tablets, computers, etc.)

Table 4 - Non-functional requirements for the CreativeWall platform

3.1.3.2. Use Case Diagram

The Use Case Diagram helps developers identify the roles that should be present on the platform as well as all the functionalities that should be related to them. These roles don't need to be implemented as they can be nothing more than abstractions for reference only.

Usually the administrator role is included in the use case diagram, but in this case, the platform's back-end will not be implemented and should be considered as future work. As a consequence, the administrator should also be considered future work.

This diagram follows the functional requirements analysis in order to make a clear definition of the effect that each of the roles has on a certain functionality.

Figure 9 shows the use case diagram that was created.

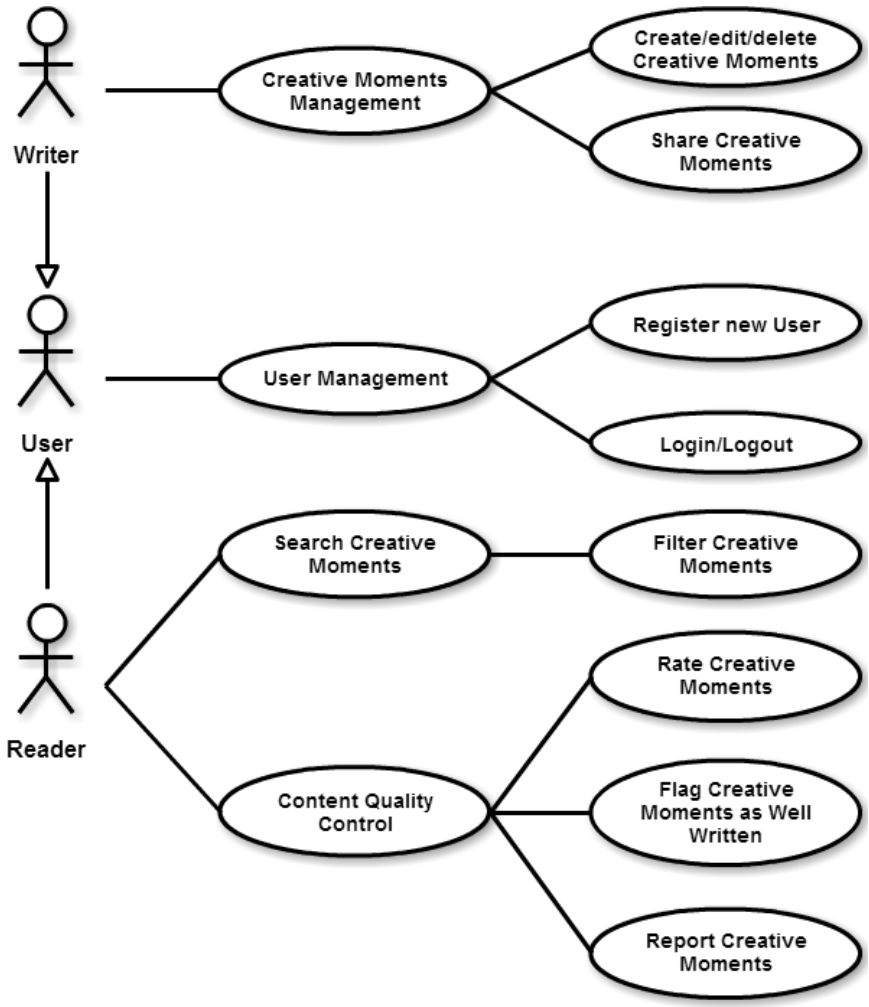

Figure 9 - Use Case Diagram related to the Creative Wall Platform

We should have three kinds of users:

- Normal User – can register and login on the platform;
- Writer – after performing login a User can become a writer and create new creative moments, edit/delete/share his previously created ones;
- Reader – after performing login a User can become a reader and search for creative moments. He can also perform the Content Quality Control tasks as rate/report/flag creative moments.

Writer and Reader are abstractions of the normal user and are mentioned for a better and clearer explanation of the structure.

3.1.3.3. Software Architecture

In order to build a usable prototype a decision about the architecture of the software was needed. One requirement that was important was the scalability of the whole software structure. It is important, in the context of our approach, to be able to provide data to any

word processor plugin or platform that wants to use the data created in our crowdsourcing platform, as long as they have previously applied for it. The solution is to build a client-server architecture. Having a centralized server (in our case the API works as both the server and an abstraction communication layer) it is possible to provide data to as many clients as we need. In the following image we describe the above statements in a more visual way so it is easier to understand.

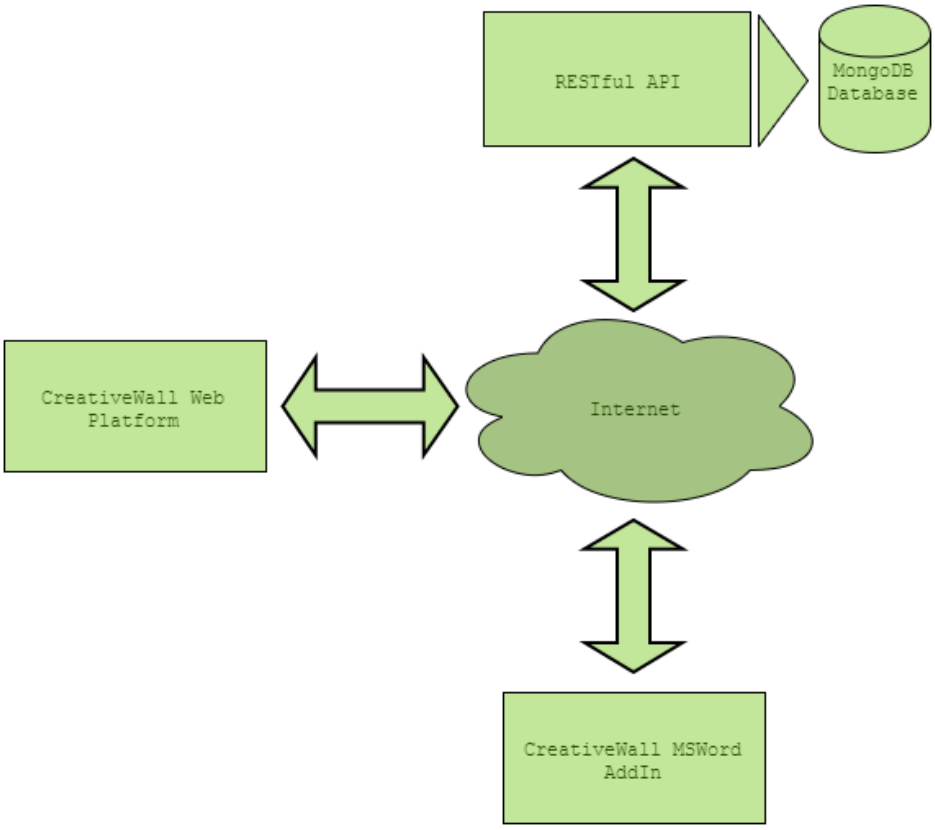

Figure 10 - Software architecture including the three platforms implemented

All connections should be made through HTTPS so the connection data (e.g. tokens or client credentials) is not exposed through package sniffing. For the purposes of this work we won't be making those HTTPS connections because this kind of features has high costs associated to it and it is not relevant for our user studies. This should be considered as future work.

All the clients have read-only permissions, except for the crowdsourcing platform that sends data to the API for it to be inserted on the database. Before being able to request any data from the API the clients must be registered and define a client ID and a client secret so they can be authenticated before establishing a connection. This protects the data from being accessed by unauthorized clients.

3.1.4. Technologies

The platform was implemented in PHP as the server-side technology and JavaScript, HTML and CSS as the client-side technologies. The reasons behind these choices were the previous experience in these technologies and the solidity that they have in the market. There was no need for a database technology as this platform has all of its data retrieved from the API and not from a direct connection to a database.

3.1.4.1. PHP

PHP (PHP: Hypertext Preprocessor) is one of the most used and solid server-side scripting languages in the range of languages available nowadays. It was created by Rasmus Lerdorf but has been developed by other teams since then. PHP is open-source and can be used without any additional cost. There are lots of frameworks developed in this language and since they are fundamental for today's development we chose to work with Yii Framework.

3.1.4.2. Yii Framework

Yii Framework was created by Qiang Xue in 2008 and is considered one of the most consistent, secure and professional frameworks available in the market. This framework is open-source and can be used without any additional cost as well. This framework follows an MVC structure and has a component called Gii to generate all the models, controllers and views. This framework is highly extensible and has lots of extensions available in the official website. Another strong point is the quantity of documentation available online. As it has been used by a huge amount of people almost all questions are already answered and have solutions available. All these facts contributed for the selection of the Yii Framework.

3.1.4.3. JavaScript

JavaScript was created by Brendan Eich and is basically mandatory in every web platform development nowadays. Together with HTML and CSS, JavaScript is a crucial part on information presentation. Being a client-side scripting language, JavaScript makes webpages more dynamic and customable. JavaScript can also be used as a server-side scripting language but in this case we won't be using this feature.

3.1.4.4. JQuery

JQuery is the most popular JavaScript front-end framework. It facilitates the use of JavaScript using selectors that access the actual DOM (Document Object Model). DOM is an object that represents an HTML element or tag. JQuery allows the user to manipulate DOMs in order to make animations, clone elements, append elements into other elements, etc.. JQuery is an open-source project with lots of documentation available online. It shows to be a very mature framework that is used every day worldwide.

3.1.4.5. HTML & CSS

HTML stands for HyperText Markup Language and is the most used language for information structuring in a web page. Was created by Berners-Lee in the late 90's. The language is based on tags and is simple to learn, although it takes a lot of experience to become an expert in it. HTML is a standard nowadays and it is used in almost every web page available online. Along with CSS it allows the developers to set a structure of rules and customize the web pages to their taste. CSS (Cascading Style Sheets) basically complements HTML. Without HTML, CSS does not do anything. It serves the purpose of customizing HTML with rules that are applied at runtime and change how the HTML looks.

3.1.4.6. Foundation Framework

There are some frameworks available that help you structure the HTML, CSS and JavaScript of a web page. The most popular frameworks are Bootstrap and Foundation. They both have a very robust grid system that allows developers to build a modern web page design without spending too much time with details. The framework that was chosen was the Foundation due to its robustness and easy customization through Sass. Sass is a CSS preprocessor in which the developer can set variables, build mixins (similar to functions in other languages), make loops, etc.. The Sass files, after compiled with an adequate compiler, generate CSS files that can later be used to customize web pages. Foundation is built with Sass and Foundation allows the user to overwrite the Sass rules in order to facilitate adaptation to any web page. Foundation is mobile friendly and that was another very important factor for choosing it. When it comes to mobile, Foundation shows a better quality than Bootstrap. Both frameworks also come with some components written in JavaScript that can be imported into any project and used to make it even more dynamic with less work and time spent.

3.1.5. Design

One of the priorities for the design was that it had to be responsive for usage in multiple devices. One of the most used web design concepts nowadays is the card based design. This kind of design shows information in cards which are a very simple and effective way of passing information to the user. Most of the today's social networks (e.g. Facebook or Pinterest) have a card based design along with a flexible grid.

3.1.6. Implementation

3.1.6.1. MVC Structure

To follow an MVC structure the first step was to decide the structure of the project. Figure 11 shows the MVC structure that was generated.

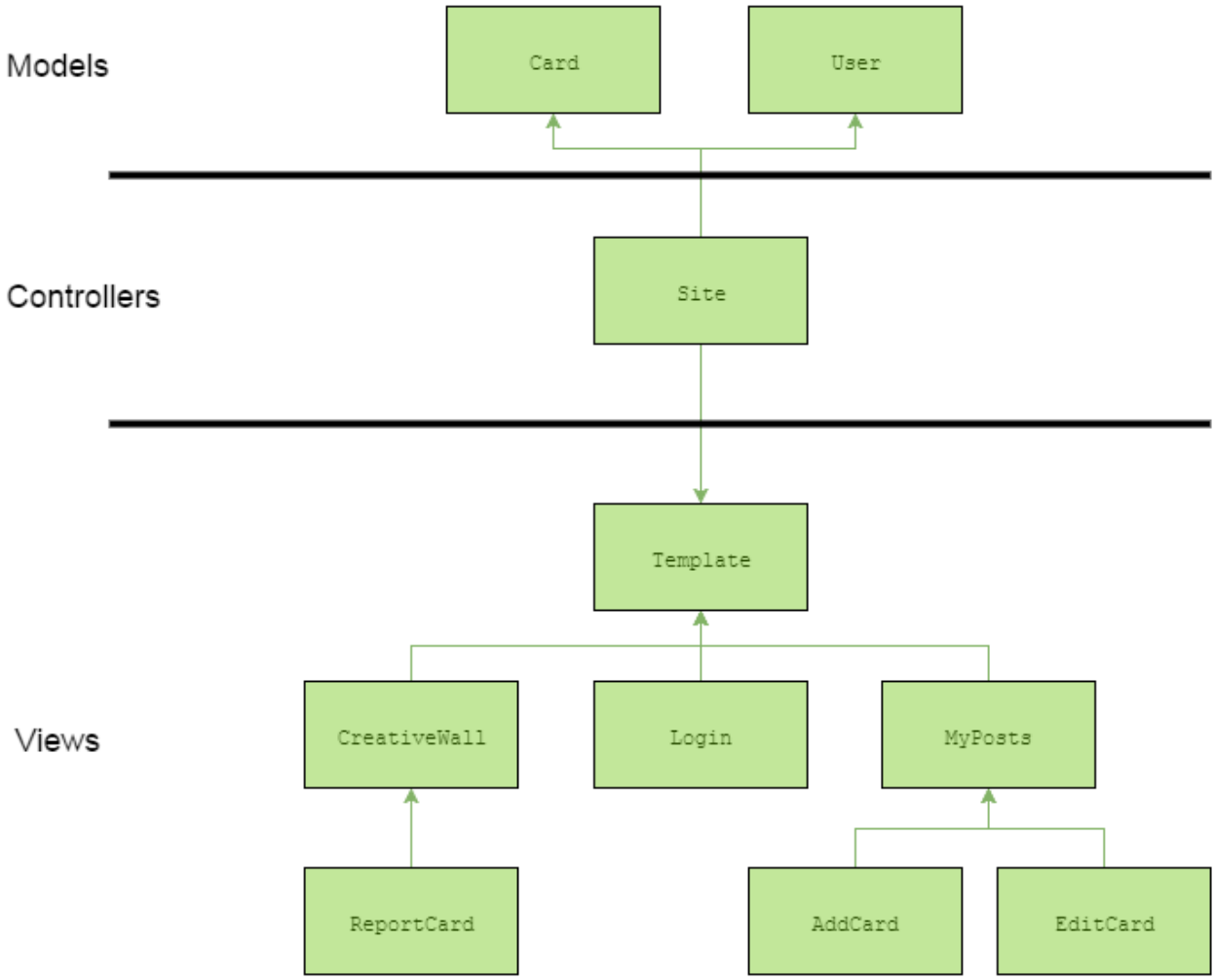

Figure 11 - MVC diagram with the structure of the project

As shown in the image, there was only one controller, for the purposes of the project there was no need for any more controllers as there would be only two models. One model was for the creative moment (called card because of the design adopted) and anything that was related to it. The other one for the User, for authentication and registration purposes. The models usually are responsible for the connection to the database, and all communications with it, but in this case all connections to the database are handled by the API and so no database connections were performed by the models.

The structure of the views follow a tree structure because some of them are only partial views (e.g. the AddCard view is not a full web page, but only a part of it). The template is the main view and the rest of the views are derived from the template, being that all of them are included inside the template. This tree structure guarantees robustness and reusability for all the views that compose the project.

3.1.6.2. HTTP Requests

All HTTP requests to the API were handled by an extension that was downloaded and installed into the Yii Framework. This extension is called EHttpClient. This extension was built by Antonio Ramirez, a developer from the Yii Framework Community. The extension is quite popular in the framework when it comes to HTTP/HTTPS requests and is, at the moment, completely stable. This extension allows developers to make any kind of HTTP request (post, get, etc.) which is very useful as we need different kinds of requests to fulfill all the API requirements in order to retrieve data from the database.

To use this extension, there must be an instantiation of the class in order to create an object that can be then used to make the requests. To minimize the waste of memory from the web server a design pattern called Singleton was implemented. In this design pattern an object is only created once, as the next calls for instantiation get the instance created before and not a new instance of the object. This prevents the object from being created over and over again on every single request the platform makes to the API.

3.1.6.3. Cookies

The login feature was implemented using a cookie that consisted of the token returned from the API to identify and authenticate a user in future request without having to pass the user credentials along with the request. When an HTTP request is send, there is an authentication header that goes along with the request so the user gets recognized by the API.

3.1.6.4. Infinite Scroll

An infinite scroll was also implemented. An infinite scroll allows users to scroll the page down until all the records were fetched from the database. In our case, the main reason to implement this feature was to prevent requests to the database that would fetch a high number of rows, although this would never actually happen for the purposes of this work as there are not a high enough number of records on the database. The infinite scroll was implemented using JQuery and it will only fetch 3 records at a time from the database. This will increase the number of requests to the database but as a tradeoff we have smaller requests that will not take too much time to be concluded and returned.

3.1.7. Functional Prototype

After all the implementation was done, a functional prototype could finally be tested. Figure 12 shows the final prototype and how the section My Posts looks. The design chosen was achieved and we felt very happy with all the design choices that were made.

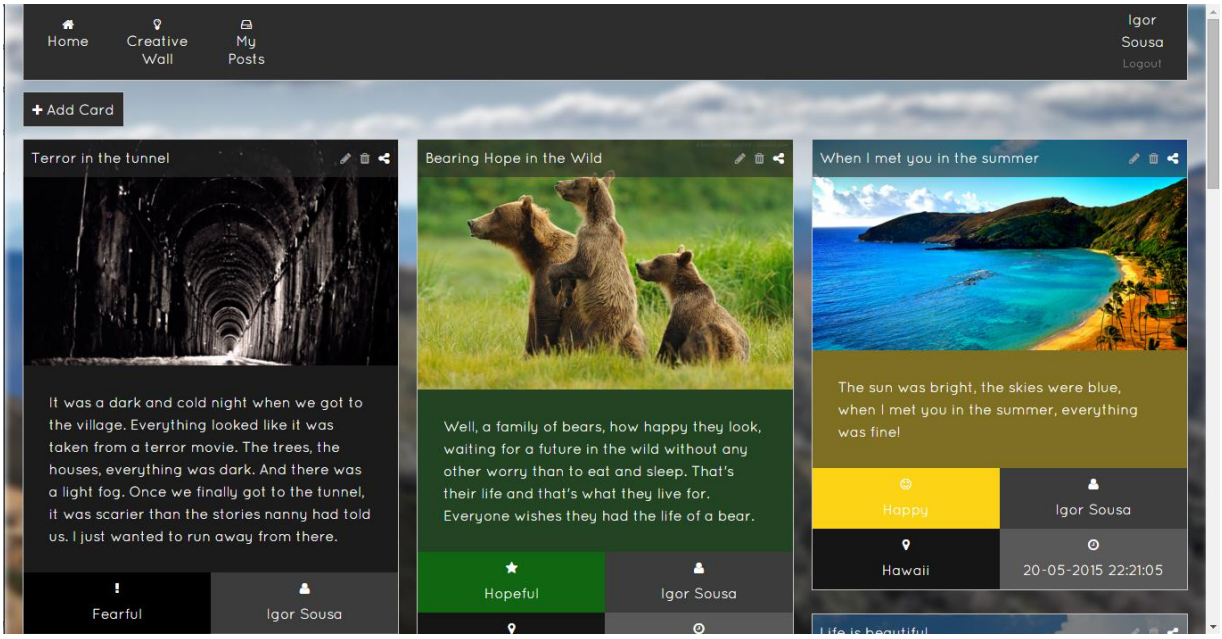

Figure 12 - Final Prototype of the CreativeWall platform

No specific tests were performed for the crowdsourcing platform as it acts only as a tool to populate the database with data to be used in the MSWord Add-In.

3.2. RESTful API

3.2.1. Concept

A RESTFul API is responsible for receiving HTTP requests such as POST, GET, PUT or DELETE and return protected resources according to the content of the request. A RESTful API supports create, read, update and delete (CRUD) operations and handles encryption of the sensible data and authentication of both users and client applications.

In this project, the need of an API came from the fact that multiple platforms will need access to the same resources, and so a communication layer between the clients (the platforms) and the database is important to keep a certain degree of security. The API is responsible for all the communication between the Client platforms and the database, being them for writing or reading information.

3.2.2. Technologies

3.2.2.1. NodeJS

NodeJS is a JavaScript runtime environment for server-side web applications and was created in 2009 by Ryan Dahl. It is an emerging technology and has gained a lot of popularity in these last 6 years being that it is already used by great companies like Microsoft or IBM. NodeJS is asynchronous and as so, is based on callbacks. It is event driven which makes it very flexible.

There are some frameworks that work with NodeJS. One of those frameworks is the Express Framework.

3.2.2.2. Express Framework

Express Framework is a minimal framework for interacting with NodeJS web servers. It is perfect for building APIs. It fully supports communications through HTTP and HTTPS and its routing system makes it much easier to build a reliable API. Express uses middleware's to extend its core features. Middleware can be defined has an extension that can be installed in Express. One of the middleware's that was used in this project was Passport. Passport makes it easier to implement authentication strategies like OAuth2.0. This is a very useful feature for an API because it grants an additional layer of security for connections through HTTP/HTTPS. Express Framework also has middleware to communicate with MongoDB.

3.2.2.3. MongoDB

MongoDB is a NoSQL database that is becoming very popular for its performance and ease of use. MongoDB has no schema associated which makes it much more flexible than conventional relational databases like SQL databases. MongoDB is based on collections instead of tables and documents instead of tables. Some of the advantages of using MongoDB is the ease of use with object oriented programming and the low costs associated with maintaining a database.

3.2.3. Implementation

3.2.3.1. Routing

The first phase of the implementation of the API was the routing for all the HTTP/HTTPS communications. It was necessary to define the appropriate response for all kinds of requests (GET, POST, DELETE and PUT). As referenced before, each one of these requests concerns a different type of action. A description of each type of request is listed below:

- GET – returns a resource or set of resources from the database to the client;
- POST – inserts a resource or set of resources into the database;
- DELETE – removes a resource or set of resources from the database;
- PUT – updates a resource or set of resources in the database.

3.2.3.2. Database Connection with Mongoose

An important middleware available on Express Framework is called Mongoose and it implements a layer between the NodeJS and the MongoDB. Mongoose defines a schema for the collections of the database. Although this kind of goes against the schema less concept, this schema serves the purpose of data type validation. Mongoose also supports the query system for MongoDB making querying very easy with asynchronous calls and callbacks to define the behavior of an event.

3.2.3.3. File Structure

The API is built with two kinds of files, models and controllers. It can't be considered an MVC because in the API there are no views, but there are models and controllers. Models are responsible for the communication with the database and the definition of mongoose schema for each collection. On the other hand, controllers deal with the data received from the requests and calls for models for taking the according action.

3.2.3.4. Authentication and Access to Protected Resources

For the authentication, we decided to use the OAuth 2.0 authentication protocol as it is becoming a standard for the new generation applications. This protocol is composed by four roles (Client, Resource Owner, Authorization Server and Resources Server) and Figure 13 shows how they communicate with each other.

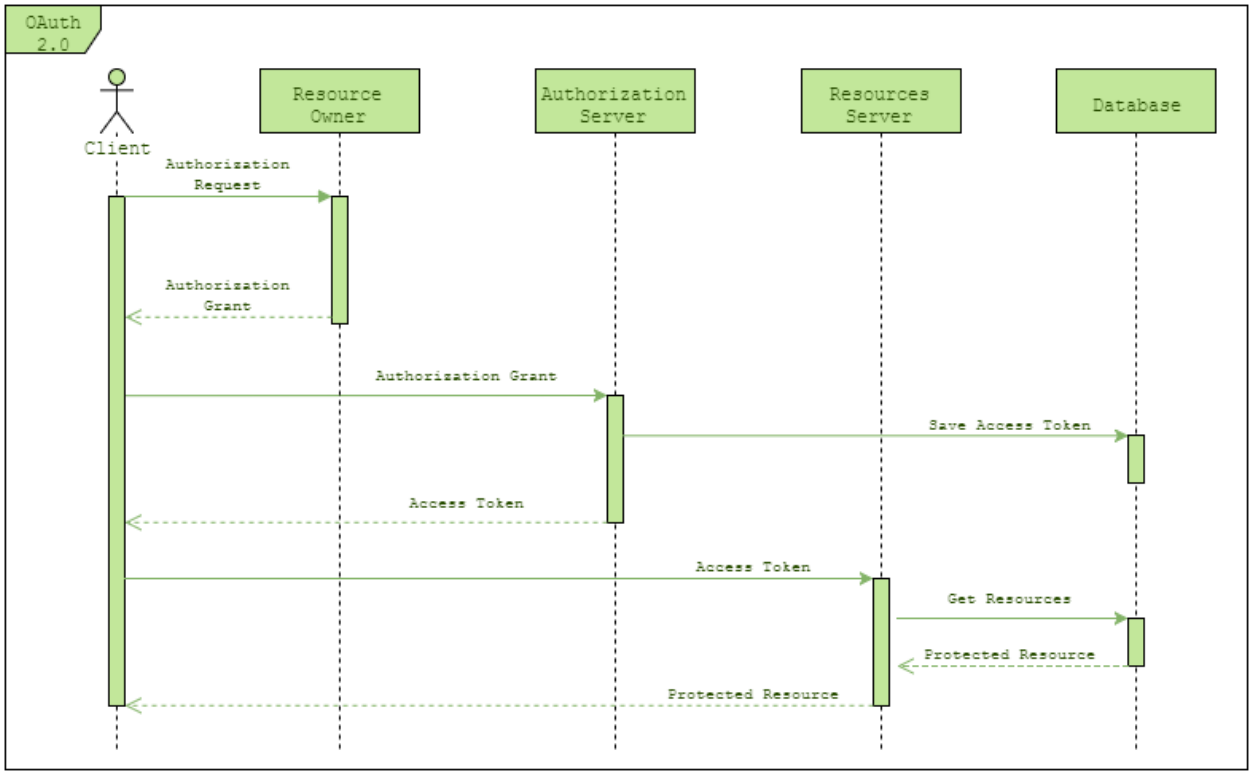

Figure 13 - Sequence diagram showing all the communication between all the components present in the API

The flow is simple, there's a request from the Client application to the Resource Owner and once the authorization is granted there's another request from the Client to the Authorization Server with the grant token provided by the Resource Owner. The Authorization Server responds with an access token that will be used by the Client to have access to the protected

resources provided by the Resources Server. As for the grant tokens, they can have one of four different types:

- Authorization Code – the Client redirects the Resource Owner to the Authorization Server so the Resource Owner credentials are never shared with the Client, after the Resource Owner is authenticated, an Authorization Code is generated and the Resource Owner is redirected to the Client;
- Implicit – the Client receives an access token directly without having to be authenticated by the Authentication Server;
- Resource Owner Password Credentials – the Client exchanges the Resource Owner credentials for an access token that can be used to access the protected resources;
- Client Credentials – the Client has his own credentials that can be exchanged by the access token.

We decided to use the Resource Owner Password Credentials grant token as it is indicated for when there is a high degree of trust between the Resource Owner and the Client (in this case they are built to work together).

Even though the Client has access to the Resource Owner credentials, they are never stored, instead they are exchanged by the access token. Furthermore, the Client has its own credentials which must be passed along with the Resource Owner credentials. These Client credentials include a Client Id and a Client Secret.

The access token generated is used as a Bearer Token, where the "bearer" uses the token to access the protected resources. This basically means that anyone that "bears" the token can access the protected resources, which is a risk. That's why, to use this kind of token, some criteria must be met.

- All the connections must be made in HTTPS;
- Access tokens have a short lifetime and they must not be passed in the URLs (GET method).

These rules must not be broken in a production environment as they might compromise the security of the protected resources. A refresh token system should also be implemented but for the purpose of this work it is not necessary as security is not crucial, but it should be considered a necessary future work.

3.2.3.5. Encryption

As for the encryption, the algorithm used is SHA-256 with a salt string of 24 bytes. Each user has a unique salt, which is a random group of characters that is used to give an additional level of encryption to the users' password. SHA (Secure Hash Algorithm) was developed by NSA and has a variety of versions (SHA-0, SHA-1 and SHA-2) developed to correct earlier versions potential issues that could compromise the security of the data encrypted with those versions.

3.2.4. Database

The database type chosen to support the data structure was the NoSql database, with the MongoDB system. This system is document-based and uses the JSON (JavaScript Object Notation) which is becoming a standard in web programing. MongoDB follows an object-oriented paradigm which facilitates the manipulation of records. The performance is also better than the traditional relational database system as there is no need for joining tables and performing complex queries.

Figure 14 shows the relationship between collections presented in the database.

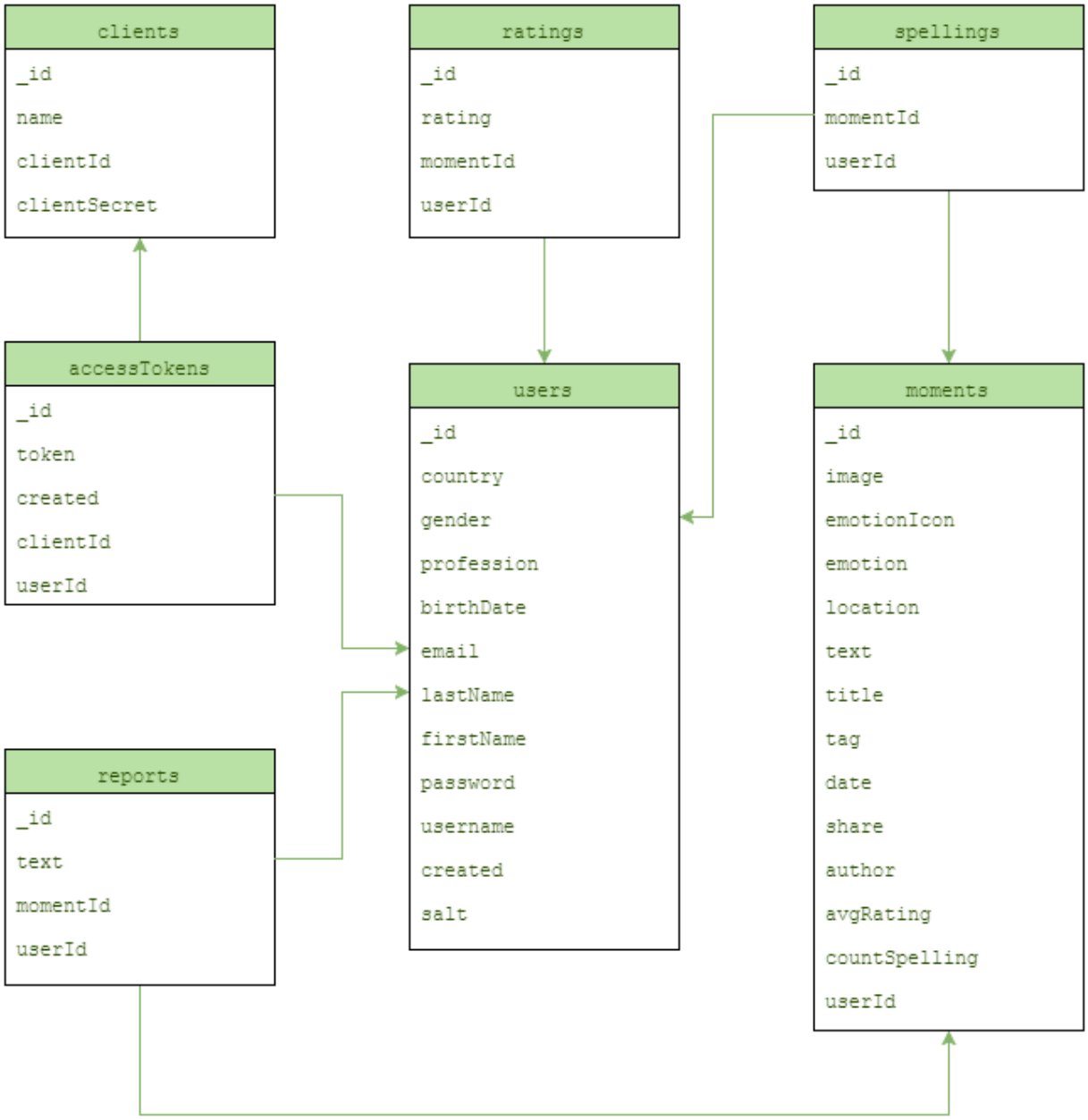

Figure 14 - Diagram representing the NoSQL database implemented

The reason for us representing this as a simple diagram is that the database is supposed to be schema less and so the diagram is just for reference in order to keep track of the logic of the database.

There are three main collections, clients, moments and users. All other collections have a relation with these three. The user collection stores all the information about one user. It includes a field called salt that is used for the encryption of the user's password.

The moments collection is used for storing all the creative moments data, including average rating and spelling count. The spelling is related to the well written flag that each user can set on other user's moments.

As for the clients collection, it stores all the information about a client (platform that uses the API in order to store or read data e.g. CreativeWall crowdsourcing platform). Each client has a clientId and a clientSecret that is used to authenticate each request that the API receives. The reports, ratings and spellings collections serve the purpose of storing all users actions in other user's creative moments (like rating a moment or reporting it for any reason). As for the accesstokens collection, it stores all the access tokens generated for each user enabling the API to check if the user already has an access token or not and generate tokens according to the user needs.

3.3. CreativeWall MSWord Add-In

3.3.1. Concept

CreativeWall MSWord Add-In is a plugin (called Add-In by Microsoft Office) for Microsoft Word that uses the data created in the CreativeWall platform in order to provide users with a way to trigger creativity while writing in the Microsoft Word application. After taking a look at the current state of the art in terms of existing plugins to enhance creativity, we couldn't find any plugin that attempts to trigger the user's creativity and overcome writer's block, and concluded that it would be a good addiction to the market.

This plugin requests creative moments from the API according to some filters chosen by the user and then shows those creative moments in Microsoft Word. By doing this, the user can use the creative moments for triggering creativity in the word processor tool itself without having to change applications and search for creative moments in the online platform.

3.3.2. Features

The features that were implemented help the user getting what he wants as fast as possible, and that was the main goal of the interface implemented. The plugin has a ribbon that contains a set of buttons and inputs which the user can use to interact with the plugin.

3.3.2.1. Ribbon

As described before, the ribbon contains a set of buttons and inputs that can be used to interact with the plugin. Figure 15 shows the ribbon and the elements available for interaction.

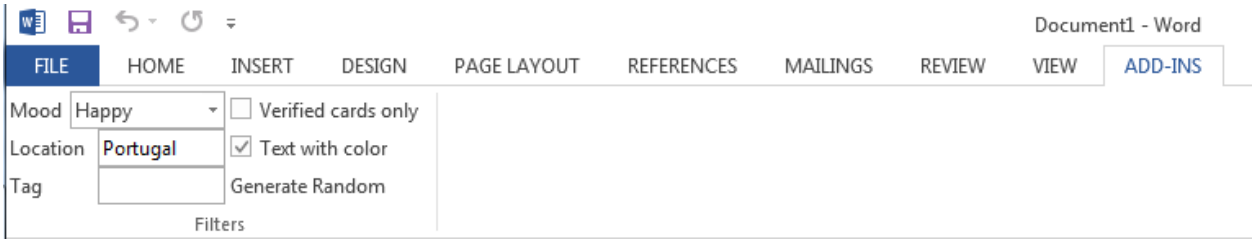

Figure 15 - CreativeWall Add-In ribbon with buttons and inputs available

The first three options, mood, location and tag are self-explainable, they apply filters to the results. The checkbox for "Verified cards only" will return cards that have average ratings over 4 stars and at least 5 well-written tags. This helps the user get data that has a minimum of quality and is not just random words with no meaning. This also helps the user getting creative moments that are written correctly. As for the option "Text with color", it defines if the creative moment should be shown with a gray background or with a background according to the color associated with the mood of the creative moment. The button "Generate Random" generates a random creative moment according to all the filters selected on the other options. If the user inserts a set of filters to which there are no creative moments associated, an error message is displayed. Figure 16 shows the error box that is shown.

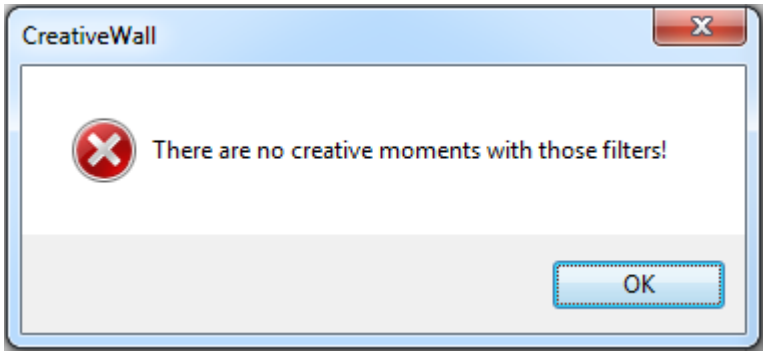

Figure 16 - Error box shown on the CreativeWall Add-In

3.3.2.2. Add-In Interface

The add-in creative moments interface is basically the same as the one in the CreativeWall web platform, it just has less fields. Each of the creative moments shown have a title, image, text, location and author. Figure 17 shows an example of a creative moment in the add-in.

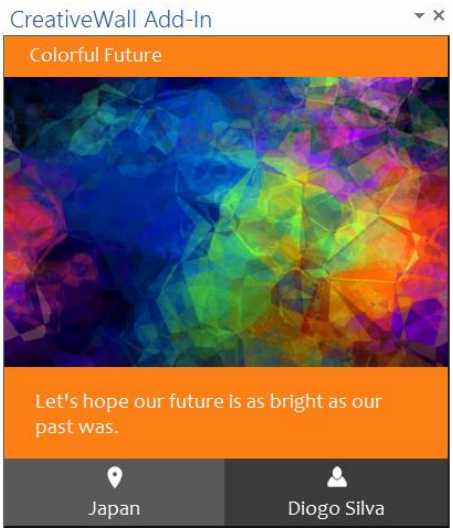

Figure 17 - Creative Moment in the Microsoft Word Add-In

3.3.3. Technologies

3.3.3.1. C#

The CreativeWall Add-In is implemented in C#, which is a very powerful high-level object-oriented programing language developed by Microsoft as part of the .NET framework. For the interface Windows Forms were used as they are easy to manipulate and make the design of a desktop interface very easy to implement.

3.3.4. Implementation

The communication between the add-in and the API is made through HTTP requests made with the HttpWebRequest library included in the C# project. This library allows applications built with C# to make HTTP requests to an address and receive a response. In this case, the response is a JSON string that is interpreted and converted into an object through JavaScriptSerializer library. These libraries make the communication with the API very easy to implement.

3.3.5. Functional Prototype

After all the implementation was done a final prototype, that is to be used in the user study, was achieved. This prototype has all the features described above and they are all usable. It can be used by users after running a setup to install all the registry entries needed to use it on Microsoft Word as an Add-in. This prototype is just one example of an application using the add-in, since it may be used by any other word processor tool that wishes to implement our approach. All they need to do is register for usage of the API and after they set a client id and a client secret, they can start making requests to the API in order to receive the desired data. Figure 18 shows the final prototype and how it is integrated on the word processor tool Microsoft Word.

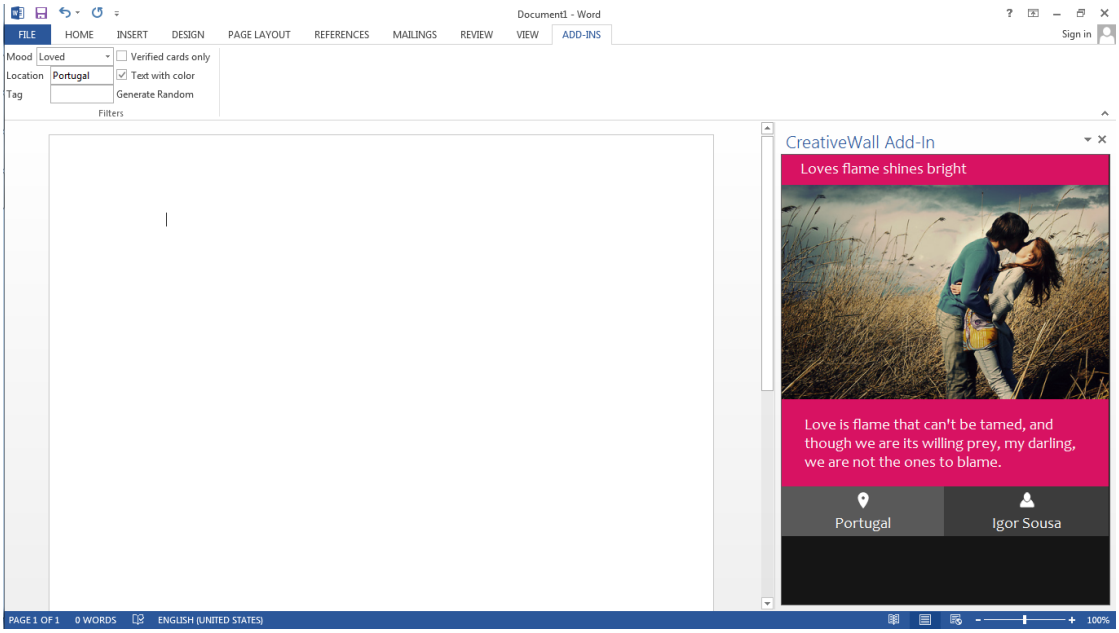

Figure 18 - Final prototype of the Microsoft Word Add-In

3.3.6. Usability Evaluation

After the functional prototype was ready a usability evaluation was made in order to understand if there was something that should be enhanced on future work. The evaluation method chosen was the heuristic evaluation as it has been proven, in the state of the art chapter, to give the best results when applied correctly. In this case, the method will be applied with only one evaluator and that evaluator will not be a usability specialist, it will be the author of the work. According to Nielsen [24] this evaluation is difficult to be made with only one evaluator as it would be difficult for one individual to find all the usability problems present on the interface. As this is just a demonstration there is no intention in finding all the usability problems, just a few will be enough to prove the point.

The heuristics chosen for this test were the ones developed by Nielsen [25]. For the evaluation, a set of tasks was defined in order for the evaluator to follow and check for usability problems. The list of tasks to be performed by the evaluator are listed below:

1. Filter a creative moment by mood;
2. Filter a creative moment by location;
3. Filter a creative moment by tag;
4. Filter a creative moment by verified cards only;
5. Include color in a creative moment;
6. Generate a new random creative moment applying the filters that are available.

After the evaluation was performed and the usability problems noted, a table that shows the results was made. Table 5 shows the results of the usability evaluation. The number of the heuristic is presented according to the number of the heuristic assigned in chapter 2.

ID	Problem	Component	Task	Heuristic	Severity
1	The system does not provide any feedback that shows that the information is being loaded	Filters panel	General	1	1
2	The system allows the user to enter any characters that he wants when those characters should be limited	Filters panel	General	2, 3	1
3	The generate random button doesn't look like a button	Filters panel	6	6	1
4	There is no way to undo an action and make sure the same creative moment as before is shown	Filters panel	General	3	2
5	There is no available documentation or help	Filters panel	General	10	2

Table 5 - Usability evaluation results

These usability problems should be considered as future work as they are not crucial to the user tests.

It's interesting to note that these usability problems were not considered on the development but were easily discovered after the usability evaluation, even though it was performed by a non-expert evaluator. This solidifies the idea that usability evaluations, being the method whichever one you choose, or have the chance to do, are really important when looking to provide the best user experience.

3.4. Summary

This chapter was divided into three parts, the CreativeWall prototype, the RESTful API prototype and the CreativeWall MSWord Add-In prototype. For each of these prototypes, all the features, technologies, modeling, implementations and design decisions were discussed showing why and how each of these aspects were defined the way they were.

The architecture of the prototypes is one of the most important points in this work as it was implemented with scalability in mind. The RESTful API can be used by any other word processor tool in order to retrieve the information stored in the database, as long as they are considered trusty clients.

A requirements analysis and consequent use case diagram were built for the CreativeWall platform. These modeling diagrams helped us understand the business logic and what features to implement.

Another aspect that was discussed was the usability evaluation that was performed to the CreativeWall MSWord Add-In prototype following the heuristic evaluation technique. This helped us understanding the importance of usability evaluation for the interface design. By undertaking this experience we were able to detect some of the problems presented in the interface that make the user experience less enjoyable.

On the next chapter the user study performed using the prototypes presented in this chapter is discussed. The methods used in the study, and the results are presented in order for us to understand what was done and what results did it produce.

4. User Study

In this chapter we discuss the study that was made with real users using the prototypes described in the previous chapter and explain all the proceedings that were followed. We also show the tools and tasks that were given to each of the participants. After that we analyzed the results and calculate the differences between each tool that was used.

RQ1: What is the influence of the Creative Wall Add-In UI on the participants' mental well-being and productivity, when compared to the Microsoft Word Simple UI?

4.1. Participants

The study involved 11 individuals (7 males and 4 females) aged between 20 and 32 years old. Every participant was a Software Engineer. All sessions were made separately, each one on a different time. The main prerequisites were to be interested in writing, writing quite regularly and have basic knowledge about computers and Internet. Table 6 shows the number of participants grouped by age and gender.

Age	20-24	5
	25-28	4
	29-32	2
Gender	Male	7
	Female	4
Number of Participants	11	

Table 6 - Data relative to the participants

4.2. Method

The experimental design was based on a within-subjects design in which each individual performed the three possible tasks. In this kind of experiment design, there is no control group as each member of the group serves as their own control. The individuals were assigned a random order of tasks in order to guarantee that no knowledge was passed from one task to the other as that would influence the results. The three tasks proposed were:

- Using Microsoft Office Word without any kind of add-in related to creativity to write a text based on a given context;

- Using Microsoft Office Word with the CreativeWall Add-In with a gray background to write a text based on a given context;
- Using Microsoft Office Word with the CreativeWall Add-In with a colorful background to write a text based on a given context.

The tasks were labeled as task A, B and C, respectively. All participants were presented with a context for each one of the tasks. For task A, the participants were given a context that was not produced by our crowdsourcing platform. This task was considered a baseline. Figure 19 shows the context given on the task A.

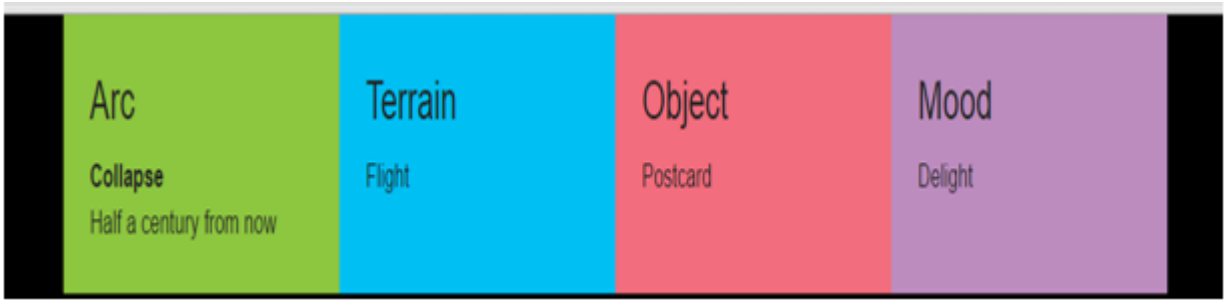

Figure 19 - Context given to the user on the first challenge

For task B, participants were given a context created on our crowdsourcing platform. This context is a creative moment that was created and shared by a user who was registered on the CreativeWall. Figure 20 shows the context that was presented for the task B. In order for this context to be presented in grey, the option “Text with color” on the MSWord Add-In has been turned off.

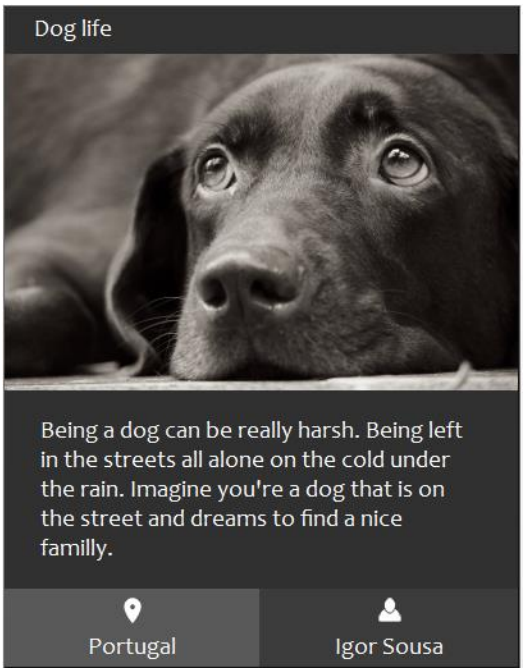

Figure 20 - Context given to the user on the second challenge

For task C, participants were given another context created on our crowdsourcing platform. Again, this context is a creative moment that was created by a registered user on the CreativeWall platform. Contrary to what happens on the second task, the option “Text with color” was turned on for this task. Figure 21 shows the context used for the task C.

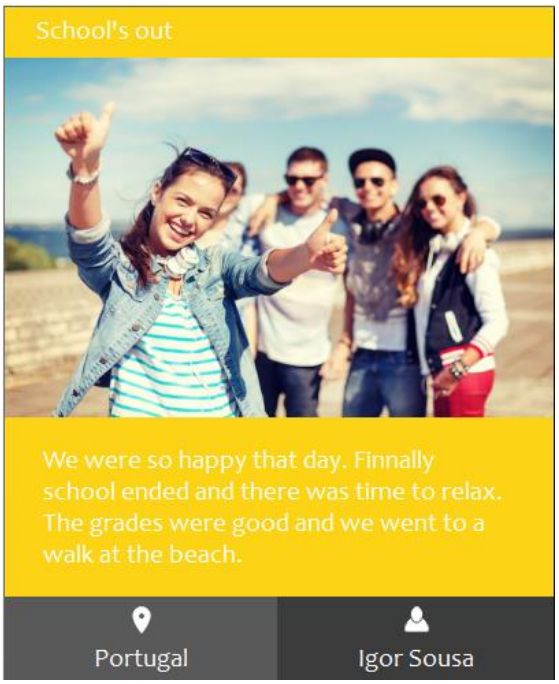

Figure 21 - Context given to the user on the third challenge

4.3. Setting

Before starting the study, every participant was introduced to the tools they were going to use and were informed about the rules of the task. During the task, the participant was left alone in a room with the computer at his disposal. After completing each task, participants were asked to fill out self-reported survey. In the end of the study, each participant was asked some general questions about their opinion about the whole experience.

4.4. Procedure

Participants were asked to write a short text using the tool designated to the task they were assigned to do. Each one of the task had a 10 minutes maximum duration. The time was monitored by the person responsible for guiding the participant through the tasks and after it reached the 10 minutes mark the participant was instructed to stop writing. The participant could ask to stop whenever he felt like he had finished writing. The text written in that time was then saved on the computer and the participant was asked to fill out the self-reported survey.

The survey contained some general questions like age, or gender and some Likert scale questions. Also a multiple choice question about how the user felt during the task was included.

The Likert scale questions were ranked from 1 (totally disagree) to 7 (totally agree). The first question was about whether the participant considered himself a creative person.

The other Likert scale questions were based on four dimensions of the Flow Theory:

1. Intense and focused concentration on the present moment;
2. Sense of personal control or agency over the situation or activity;
3. Loss of reflective self-consciousness;
4. Distortion of temporal experience.

For these dimensions questions such as "*I felt very concentrated during the challenge" or "I lost track of time during the challenge"* were included. Questions like "*Which tool did you enjoy using the most*?" or "*Is there any comment that you would like to add?*" were also asked after the three tasks were completed, in order to know the participant opinion about the whole experience.

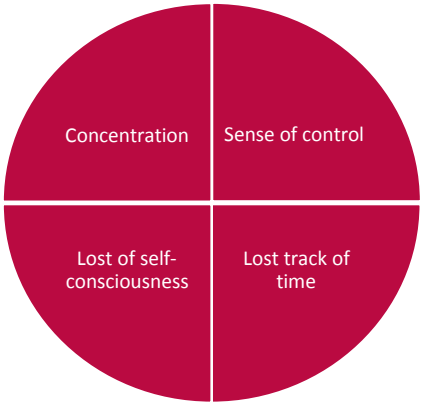

Figure 22 - Dimensions based on Flow Theory

4.5. Results

For the rest of this chapter we will refer to the tools used for the experience as shown in Table 7.

Tool	Description
Tool A	Microsoft Office Word without any creativity plugin installed
Tool B	Microsoft Office Word with the CreativeWall plugin installed with the colorful background deactivated
Tool C	Microsoft Office Word with the CreativeWall plugin installed with the colorful background activated

Table 7 - Description of the tools used

We evaluated the study from a perspective that triangulates the results, using the answers from the surveys, the statements of the interviews and the qualitative measurable information of the writing challenges.

4.5.1. Quantitative Results

To assess the reliability of our survey, we used Cronbach's alpha [29] as a measure. It was taken into account the polarity of the scale. Table 8 exhibits results of reliability (internal consistency) analyses for questions in each dimension of flow.

Flow Dimensions	Cronbach's Alpha
Concentration	.629
Sense of Control	.797
Lost Self-Consciousness	.672
Lost Track of Time	.633

Table 8 - Cronbach's alpha related to each of the dimensions

Results show that the number of test items can be considered with an acceptable consistency [29] in the scale used, from the survey, on seven-point Likert scales.

As mentioned before, the first question of the survey was about how creative the participant considered himself. The results for this question can be seen in Figure 23.

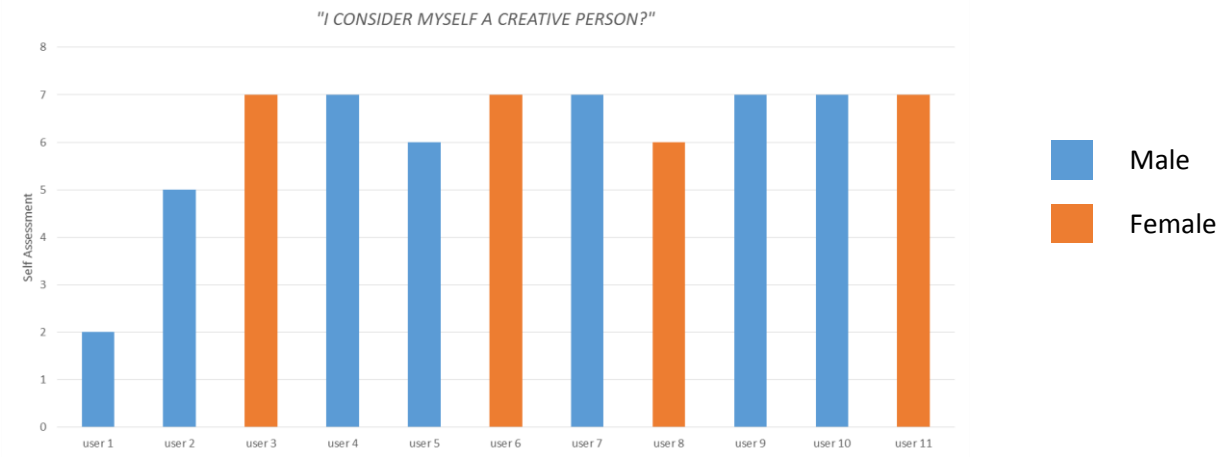

Figure 23 - Total count of the rating for the question "I consider myself a creative person"

As we can conclude from the chart, most of the participants considered themselves creative persons.

To assess the participants mental well-being we asked them to select up to three adjectives from the following list: animated, creative, distressed, fear, serious, angry, satisfied, frustrated, sad, astonished, depressed, bored, tired, happy, delighted, pacific and relaxed.

Figure 24 shows the total count for each of the adjectives presented on the above list. The words that were used the most were animated, creative, happy and pacific, being that creative was the most used for Tool B and Tool C, and serious was the most used for Tool A.

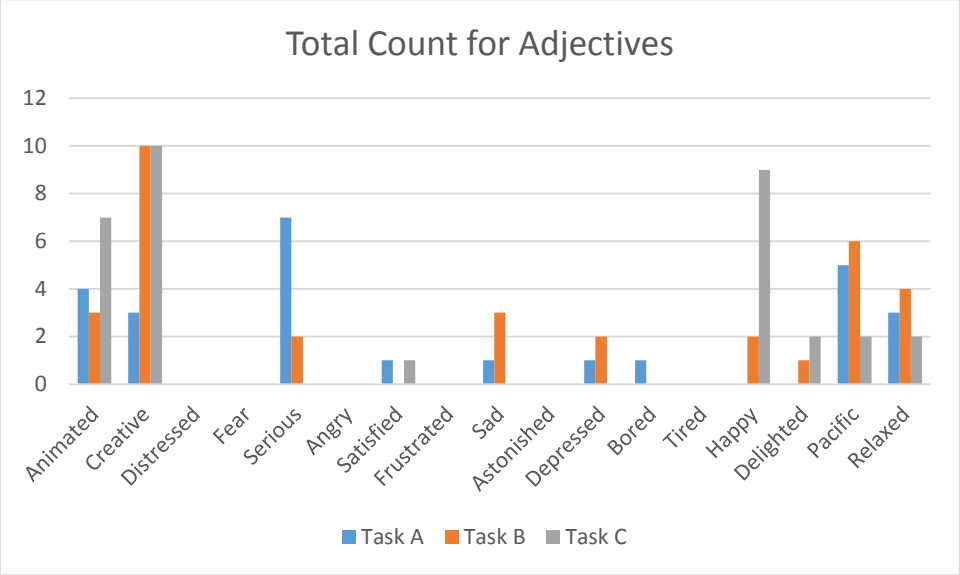

Figure 24 - Chart relating the tasks with the adjectives used in them

We proceeded using repeated measures such as Friedman's ANOVA [29][30] approach to testing differences between each condition.

The Flow dimensions Concentration (Fr (2) = 5.20, p>.05) and Lost Self-Consciousness (Fr (2) = 1.90, p>.05) did not have statistical significance when compared with each tool.

For the other dimensions Sense of Control (Fr (2) = 10.21, p<.05) and Lost Track of Time (Fr (2) = 17.43, p<.05) results were statically significant. Therefore the non-parametric Wilcoxon tests [29] [30] were used to display if there were any differences for each pair or tools, using Tool A to compare as a baseline.

Results showed that, participants using Tool A, when compared with participants using Tool B (T=0, z= -2.06, p<.025, r=-.44), for the dimension Sense of Control were not statistically significant. Also, participants using Tool A when compared to participants using Tool C, for the levels of Sense of Control, even though the value is in the border line, it was not statistically significant as well (T=0, z= -2.23, p<.025, r=-.48). We applied the effect size that gives us the magnitude of the effect investigated [1].

For the dimension Lost Track of Time, the results showed a significant difference between the participants using Tool A when compared to participants using Tool B (T=0, z= -2.71, p<.025, r=-.58). For the same dimension, the differences between participants using Tool A when

compared to participants using Tool C were also statistically significant (T=0, z= -2.72, p<.025, r=-.58).

Productivity was estimated as the average number of words produced. We used the Skeweness and Kurtosis and Kolmorov-Smirnov tests (p>0.05) to analyze the data normality. Figure 25 shows that the sampling distribution is normal. T-tests were used to compare the statistical significance of the samples using a 95% level of confidence.

Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
ToolA_NumberWords	,149	11	,200*	,934	11	,458
ToolB_NumberWords	,170	11	,200*	,946	11	,592
ToolC_NumberWords	,155	11	,200*	,950	11	,649

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

Figure 25 - Sampling distribution

When using Tool B participants wrote on average more words (M=186.1, SE=49.3) than when using Tool A (M=156.2, SE=54.5). For the same condition the difference was statistically significant (t(10)=-3.20, p<.05).

When using Tool C, participants wrote on average more words (M=185.7, SE=65.1) than when using Tool A (M=156.2, SE=54.5). The difference was statistically significant (t(10)=-2.23, p<.05) for the same condition.

4.5.2. Qualitative Results

Table 9 shows the time and sequence in which the users completed their tasks and the time spent in each one of the tasks.

User	Task sequence	Time spent on Task A (min)	Time spent on Task B (min)	Time spent on Task C (min)
User 1	A B C	10	10	10
User 2	C B A	9	10	10
User 3	B C A	8	8	10
User 4	A C B	9	10	9
User 5	C A B	10	9	8
User 6	B A C	8	9	9

User 7	A B C	7	9	8
User 8	C B A	8	10	9
User 9	B C A	9	10	9
User 10	A C B	8	10	10
User 11	C A B	10	10	10

Table 9 - Task sequence and time on which participants completed their tasks

As referred before, after all the tasks were completed, the participants were submitted to a semi-structured interview. The questions asked in the interview were:

- Which tool did you enjoy using the most?
- Do you have any comments to add to this experience?
- Did time limit your creativity?

On Table 10 we can see the answers to the questions listed above. All these answers were given directly to the person responsible for the study and written by that same person.

User	Answers
User 1	– The tool I liked the most was the C; – The color and the image encourage creativity. In fact, I felt some ease on task C. The image and color helped me idealize some ideas; – Time was a little bit limiter.
User 2	– The tool I liked the most was the B; – The text and the image helped me on the creative process. The color is a little bit distracting. The image and text really help on the creation of a story; – Time wasn't a limiter at all.
User 3	– I liked the B as well as the C, although the C is more complete because the color is really intuitive and helps us being more creative; – The image and the text were the aspects that most helped me with my creativity. Color helps as well but is not as decisive as the text and color. Color can also be a motive for distraction. A less intense color might have been better; – Time wasn't a limiter because I did not think about it.
User 4	– The tool I liked the most was the tool B for the fact that the text and the image were more creative;

	– Even though color is a very interesting component, the most important ones are the text and image. If the text is not creative then the color won't make a difference; – Time wasn't a limiter as I only reached the time limit in one of the tools.
User 5	– My favorite tool was the B, because it had a more attractive text; – Although tool C has the color, which helps as well, the text and image were not as easy to build something from it as tool B; – Time was not a limiter.
User 6	– The tool I enjoyed the most was the C; – Color helps the user remember of something. For example, the yellow resembles the joy of the summer. The task with the tool B was also very attractive because of the story. Tool A is very limited because it lacks the image, and the image is very important for creativity; – Time was not a limiter.
User 7	– My favorite tool was C, because it really made me feel happy and refreshed both because of the color and because of the image; – I think that in this case, color really helped, because the text was about summer and yellow resembles the summer. It did make me feel kind of happy. The gray color from the task B made sense with the subject. Task A was the one that made me feel less what I was writing; – Time was not a limiter.
User 8	– My favorite tool was C; – Color helped, but the most important thing was the text and the image; – Time was not a limiter.
User 9	– The best tool was C as it has a complement that could help with something; – Color did not helped all that much, but it's always good to have one more feature. It doesn't hurt anyway. The color in the B actually made the same effect as the one on the C; – Time was not a limiter.
User 10	– My favorite tool was C as the color made me feel happy and creative. The theme of the text and the image went really well with the color; – In my case, and for the kind of text that I've written, everything helped a little bit; – Time was not a limiter.
User 11	– The best tool was B because the text was more creative; – Color did not help me much, I think it was kind of distracting actually. The images and the text were very important; – Time was kind of a limiter as I would write a lot more if it was not for the 10 minutes limit.

Table 10 - Answers to the free opinion questions

Triangulating the results with the semi-structured interviews conducted after the writing challenges apparently suggest that, by unanimity of the answers registered, task A was the less enjoyed task, and the reason participants presented for this statement was the lack of images and texts to take as a reference. According to some participants, these two factors (images and texts) are the most important for their creativity to flow. For example: "*Tool A is very limited because it lacks the image, and the image is very important for creativity*." (User6).

Another interesting fact is that, for the majority of participants, color does not influence creativity when they don't find the text creative. "*Even though color is a very interesting component, the most important ones are the text and image. If the text is not creative then the color won't make a difference."* (User 4).

Some other participants refer to the color as being a source of distraction and state that the color chosen was too intense, making it hard to concentrate on any other component. "*Color helps as well but is not as decisive as the text and image. Color can also be a motive for distraction. A less intense color might have been better."* (User 3)

With this we assume that our initial idea that color would help triggering creativity through the emotion it generates on people can't be applied to every participant. Even though some participants enjoyed having the color together with the image and the text, and stated that it helped their creativity to flow, they are still a minority. "*I think that in this case, color really helped, because the text was about summer and yellow resembles the summer. It did make me feel kind of happy*." (User 7)

Also, most participants don't see time as being a limiter. If we take into account Table 7, we can see that participants usually do not write for the 10 minutes, which was the time limit for each task. This might indicate that even though the tools used for the experience helped the participants start writing, they are not as effective when it comes to keeping a constant creativity flux and therefore helping only on the first steps of the creative process.

4.6. Summary

In this chapter, the user study performed after the implementation of the prototypes was described and analyzed. The study included eleven participants and showed to have some statistically significant results in one of the dimensions of the flow theory and in the word count statistic. Also, we were able to take some considerations about the qualitative results that resulted from the semi-structured interviews made after each of the sessions.

5. Conclusions

The main goals of this work were to investigate if a creative writing support UI could enhance a user mental well-being and productivity, and to check whether color could influence creativity in any way. Although the number of participants in the tests was kind of limited, it was possible to develop some statistical results and take some conclusions from them. Even so, as future work, the idea of performing more studies for more solid results should be considered. Having a wider sample would be essential for having more reliable results.

As for the results obtained through the user study, we were able to conclude, through statistical evidence that participants performing the tasks with the Creative Wall Add-In lost track of time more often than participants using the simple Microsoft Word UI without any add-ins. Another aspect that was statistically relevant was the fact that users using the CreativeWall Add-In were able to produce more words, what could mean that participants using the Add-In can be more productive than participants that don't.

Also, taking into account the answers that results from the semi-structured survey performed after each of the sessions, it was clear that the tasks performed with the CreativeWall add-in were more enjoyable and that participants had no problems starting to write, which could mean that these tools help in the initial phase of the creative writing process. This could also mean, that, by consequence of the previous statements, this approach can be effective when trying to overcome writer's block.

About the color, the majority of participants stated that color does not influence their creativity, and can be somewhat distractive. With this kind of statements we can assume that color does not always influence the creativity of users.

6. Future Work

Taking into account, the conclusions made in the previous chapters, the color feature should be reviewed in order to try and take the most out of it. This includes reviewing the concept and reviewing the color itself. Trying to make color less intense would be one of the solutions for some of the problems presented in the user study chapter, namely the distraction caused by it.

One of the features that were talked about when we were defining the concept, was sound. It could be interesting to include environment sound. When a creative moment is recorded, an image, a location, a mood, etc. is present, and so is sound. Sound is all around us every day and maybe, it can be used by some people for enhancing creativity.

Another thing that could be included in this work were wearables that could be used to record the creative moments and then upload them into the database in order for them to be available in the crowdsourcing platforms as well as in the plugins. This could be a very interesting use for this approach as it could be seen in wearables as well.

The problems detected in the usability evaluation should also be solved, in case of intention to introduce the platform in the market. This would be very important to attract people and make them use these solutions. People are more and more interested in having a really good experience when using any kind of software and the usability is very important for any platform to survive in the market.

Also, as it was the intention from the beginning, having this approach implemented in various word processor tools would be a valuable aspect for users to want to try it. This would make it possible for users to try the approach in their favorite word processor interface and not having to change to a new one.

References

[1] M. Rose, "Writer's Block: The Cognitive Dimension. Studies in Writing & Rhetoric," Southern Illinois University Press, Carbondale, 1984.

[2] E. Estellés-Arolas and F. González-Ladrón-De-Guevara, "Towards an integrated crowdsourcing definition.," *J. Inf. Sci.,* vol. 38, no. 2, pp. 189-200, 2012.

[3] J. Howe, Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business (1 ed.), New York, NY, USA: Crown Publishing Group, 2008.

[4] D. C. Brabham, "Crowdsourcing as a Model for Problem Solving: An Introduction and Cases," *Convergence: The International Journal of Research into New Media Technologies,* vol. 14, no. 1, pp. 75-90, 2008.

[5] F. Kleemann, G. Voß and K. Rieder, "Un(der)paid Innovators: The Commercial Utiliza-tion of Consumer Work through Crowdsourcing," *Science, Technology & Innovation Studies,* vol. 4, no. 1, pp. 5-26, 2008.

[6] K. Crowston, "Amazon Mechanical Turk: A Research Tool for Organizations and Information Systems Scholars," in *IFIP Working Group 8.2 Conference: Shaping the Future of ICT Research: Methods and Approaches*, Tampa, FL, 2012.

[7] R. Snow, B. O'Connor, D. Jurafsky and A. Y. Ng, "Cheap and Fast — But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*, Stroudsburg, PA, USA, 2008.

[8] A. Kittur, "Crowdsourcing, Collaboration and Creativity," *XRDS,* vol. 17, no. 2, pp. 22-26, December 2010.

[9] J. Raddick, G. Bracey, P. L. Gay, C. J. Lintott, P. Murray, K. Schawinski, A. S. Szalay and J. Vandenberg, "Galaxy Zoo: Exploring the Motivations of Citizen Science Volunteers," *Astronomy Education Review,* vol. 9, no. 1, SEPTEMBER 2009.

[10] D. Cornelia, "If You Have a Problem, Ask Everyone," The New York Times, New York, 2008.

[11] M. Baas, C. K. W. De Dreu and B. A. Nijstad, "A Meta-Analysis of 25 Years of Mood–Creativity Research: Hedonic Tone, Activation, or Regulatory Focus?," *Psychological Bulletin,* vol. 134, no. 6, p. 779–806, 2008.

[12] A. de Rooij and S. Jones, "Mood and Creativity: An Appraisal Tendency Perspective," in *9th ACM Conference on Creativity & Cognition*, Sydney, Australia, 2013.

[13] C. K. W. De Dreu, M. Baas and B. A. Nijstad, "Hedonic Tone and Activation Level in the Mood–Creativity Link: Toward a Dual Pathway to Creativity Model," *Journal of Personality and Social Psychology,* vol. 94, no. 5, p. 739–756, 2008.

[14] Z. Ivcevic, M. A. Brackett and J. D. Mayer, "Emotional Intelligence and Emotional Creativity," *Journal of Personality,* vol. 2, no. 75, pp. 200-236, 2007.

[15] R. S. Friedman and J. Forster, "The Effects of Promotion and Prevention Cues on Creativity," *Journal of Personality and Social Psychology,* vol. 81, no. 6, pp. 1001-1013, 2001.

[16] N. Kaya and H. H. Epps, "Relationship between color and emotion: a study of college students," *College Student Journal,* vol. 38, no. 3, pp. 396-405, 2004.

[17] N. Kaya and H. H. Epps, "Color-emotion associations: Past experience and personal preference," in *Color and Paints, Interim Meeting of the International Color Association*, Porto Alegre, 2004.

[18] J. W. van Goethe, Theory of Colours, London: Mit Press, 1970.

[19] D. Morley, The Cambridge Introduction to Creative Writing, Cambridge: Cambridge University Press, 2007.

[20] M. Weiss, "Overcoming Writer's Block," Pennsylvania, 2010.

[21] M. Csikszentmihalyi, Flow: The Psychology of Optimal Experience, New York: Harper Perennial, 1991.

[22] A. Holzinger, "Usability engineering methods for software developers," *ACM,* vol. 48, no. 1, pp. 71-74, January 2005.

[23] R. Jeffries, J. R. Miller, C. Wharton and K. Uyeda, "User interface evaluation in the real world: a comparison of four techniques," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '91)*, New York, 1991.

[24] J. Nielsen, "Usability Inspection Methods," in *Conference Companion on Human Factors in Computing Systems (CHI '94)*, New York, 1994.

[25] J. Nielsen, "Nielsen Norman Group," 1 January 1995. [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/. [Accessed 10 January 2016].

[26] T. Hollingsed and G. D. Novick, "Usability Inspection Methods after 15 Years of Research and Practice," in *Proceedings of the 25th annual ACM international conference on Design of communication (SIGDOC '07)*, New York, 2007.

[27] P. Polson, C. Lewis, J. Rieman and C. Wharton, "Cognitive walkthroughs: A method for theory-based evaluation of user interfaces (tutorial)," in *Proceedings of the Conference on Human Factors in Computing Systems (SIGCHI 91)*, New Orleans, 1991.

[28] M. F. King and G. C. Bruner, "Social Desirability Bias: A Neglected Aspect of Validity Testing," *Psychology & Marketing,* vol. 17, no. 2, p. 79–103, 2000.

[29] H. Cooligan, Research Methods and Statistics in Psychology, New York: Routledge, 2013.

[30] A. Field, Discovering Statistics Using SPSS, London: SAGE Publications Ltd, 2009.

[31] J. Nielsen, "Introduction to Web design," in *CHI 98 Cconference Summary on Human Factors in Computing Systems*, New York, 1998.

Appendices

Appendix A – RESTful API source code

Controllers

```
/**
 * authController code
 * This controller is responsible for applying the strategy for authentication
 * There are 3 strategies that are used. The Basic and ClientPassword strategies authenticate
the client,
 * while the Bearer authenticates the user and generates a token for later accesses
 */

var passport                = require('passport');
var BasicStrategy           = require('passport-http').BasicStrategy;
var ClientPasswordStrategy  = require('passport-oauth2-client-password').Strategy;
var BearerStrategy          = require('passport-http-bearer').Strategy;
var UserModel               = require('../models/user');
var ClientModel             = require('../models/client');
var AccessTokenModel        = require('../models/accessToken');

passport.use(new BasicStrategy(
    function(username, password, done) {
        ClientModel.findOne({ clientId: username }, function(err, client) {
            if (err) { return done(err); }
            if (!client) { return done(null, false); }
            if (client.clientSecret != password) { return done(null, false); }
            return done(null, client);
        });
    }
));

passport.use(new ClientPasswordStrategy(
    function(clientId, clientSecret, done) {
        ClientModel.findOne({ clientId: clientId }, function(err, client) {
            if (err) { return done(err); }
            if (!client) { return done(null, false); }
            if (client.clientSecret != clientSecret) { return done(null, false); }
            return done(null, client);
        });
    }
));

passport.use(new BearerStrategy(
    function(accessToken, done) {
        AccessTokenModel.findOne({ token: accessToken }, function(err, token) {
            if (err) { return done(err); }
            if (!token) { return done(null, false); }

            UserModel.findById(token.userId, function(err, user) {
                if (err) { return done(err); }
                if (!user) { return done(null, false, { message: 'Unknown user' }); }
                var info = { scope: '*' };
                done(null, user, info);
            });
        });
    }
));

exports.isBearerAuthenticated = passport.authenticate('bearer', { session: false });


/**
 * momentController code
 * This controller is responsible for everything that has to do with the creative
 * moments (creating, editing, rating, etc.).
 */
```

```
var AccessToken = require('../models/accessToken');
var Moment = require('../models/moment');

/**
 *  GET to retrieve creative moments
 */

exports.getMoments = function(req, res) {
    var auth = req.header("Authorization");
    var token = auth.split(" ")[1];
    AccessToken.findOne({ token: token }, function(err, accessToken){
        Moment.find({ userId: accessToken.userId }, {}, {}, function(err, moments){
            if (err)
                res.send(err)
            else
                res.send(moments)
        })
    })
};

/**
 *  GET to retrieve creative moments by id
 */

exports.getSingleMoment = function(req, res) {
    Moment.findById(req.params.id, function (err, moment) {
        if (err)
            res.send(err)
        else
            res.send(moment)
    });
};

/**
 * POST to add creative moments
 */
exports.postMoments = function(req, res) {
    var moment = new Moment ({
    });
    var auth = req.header("Authorization");
    var token = auth.split(" ")[1];
    AccessToken.findOne({ token: token }, function(err, accessToken) {
        moment.title = req.body.title;
        moment.text = req.body.text;
        moment.location = req.body.location;
        moment.emotion = req.body.emotion;
        moment.emotionIcon = req.body.emotionIcon;
        moment.image = req.body.image;
        moment.author = req.body.author;
        moment.userId = accessToken.userId;
        moment.sound = " ";
        moment.tag = req.body.tag.split(",");
        moment.save(function (err) {
            if (err)
                res.send(err)
            else
                res.send(req.body)
        });
    })
};

/**
 * DELETE to delete creative moments by id
 */
exports.deleteSingleMoment = function(req, res) {
    Moment.remove({ _id: req.body.id }, function (err) {
        if (err)
            res.send(err)
        else
            res.send(req.body)
    });
```

```
};

/**
 * PUT to update creative moments by id
 */
exports.putSingleMoment = function(req, res) {
    var params = new Object();
    params.title = req.body.title;
    params.text = req.body.text;
    params.emotion = req.body.emotion;
    params.emotionIcon = req.body.emotionIcon;
    params.location = req.body.location;
    params.image = req.body.image;
    params.tag = req.body.tag;
    Moment.findByIdAndUpdate(req.params.id, params, function (err) {
            if (err)
                res.send(err)
            else
                res.send(req.body)})
};

/**
 * PUT to update the rating of a creative moment
 */
exports.putRating = function(req, res) {
    var auth = req.header("Authorization");
    var token = auth.split(" ")[1];
    AccessToken.findOne({ token: token }, function(err, accessToken) {
        Moment.findById(req.params.id, function (err, moment) {
            if (err)
                res.send(err)
            else
            {
                moment.save(function (err) {
                    if (err)
                        res.send(err)
                    else
                        res.send(req.body)
                });
            }
        });
    });
};


/**
 * oauth2Controller code
 * This controller is responsible for implementing the oauth2 protocol and is called by the
 * authController in order to generate the token for later accesses.
 */

var oauth2orize         = require('oauth2orize');
var passport            = require('passport');
var crypto              = require('crypto');
var UserModel           = require('../models/user');
var ClientModel         = require('../models/client');
var AccessTokenModel    = require('../models/accessToken');
var RefreshTokenModel   = require('../models/refreshToken');

// create OAuth 2.0 server
var server = oauth2orize.createServer();

// Exchange username & password for access token.
server.exchange(oauth2orize.exchange.password(function(client, username, password, scope,
done) {
    UserModel.findOne({ username: username }, function(err, user) {
        if (err) { return done(err); }
        if (!user) { return done(null, false); }
        if (!user.verifyPassword(password)) { return done(null, false); }

        AccessTokenModel.remove({ userId: user._id, clientId: client.clientId }, function
```

```
(err) {
            if (err) return done(err);
        });

        var tokenValue = crypto.randomBytes(32).toString('base64');
        var token = new AccessTokenModel({ token: tokenValue, clientId: client.clientId,
userId: user._id });

        var info = { scope: '*' };

        token.save(function (err, token) {
            if (err) { return done(err); }
            done(null, tokenValue);
        });
    });
}));

// token endpoint
exports.token = [
    passport.authenticate(['basic', 'oauth2-client-password'], { session: false }),
    server.token(),
    server.errorHandler()
];


/**
 * ratingController code
 * This controller is responsible for managing the ratings given to the creative moment
 */

var Rating = require('../models/rating');
var AccessToken = require('../models/accessToken');
var Moment = require('../models/moment');
var bcrypt = require('bcrypt-nodejs');

/**
 * POST to insert a new rating related to a creative moment
 * An average rating is used in order to facilitate the calculation of the rating of a
creative moment
 */
exports.postRating = function(req, res) {
    var auth = req.header("Authorization");
    var token = auth.split(" ")[1];
    AccessToken.findOne({ token: token }, function(err, accessToken) {
        Moment.findOne({_id: req.params.id}, function (err, moment) {
            if (err)
                res.send(err);
            else
                if(moment != null)
                {
                    Rating.findOne({userId: accessToken.userId, momentId: req.params.id},
function (err, rating) {
                        if (err)
                            res.send(err);
                        else {
                            if (rating != null) {
                                rating.rating = req.body.rating;
                                rating.save(function (err) {
                                    if (err)
                                        res.send(err)
                                    else {
                                        Rating.find({momentId: req.params.id}, function (err,
rating) {
                                            var ratingCounter = 0;
                                            for(var i=0; i<rating.length; i++)
                                                ratingCounter += parseInt(rating[i].rating);
                                            var avgRating = ratingCounter/rating.length;
                                            Moment.findByIdAndUpdate(req.params.id,
                                                {
                                                    avgRating: avgRating
                                                }, function (err) {
```

```
                                                    if (err)
                                                        res.send(err)
                                                    else
                                                        res.send(req.body)})
                                            })
                                        }
                                    });
                                }
                                else {
                                    var ratingDOM = new Rating({});
                                    ratingDOM.userId = accessToken.userId;
                                    ratingDOM.momentId = req.params.id;
                                    ratingDOM.rating = req.body.rating;
                                    ratingDOM.save(function (err) {
                                        if (err)
                                            res.send(err)
                                        else {
                                            Rating.find({momentId: req.params.id}, function (err,
rating) {
                                                var ratingCounter = 0;
                                                for(var i=0; i<rating.length; i++)
                                                    ratingCounter += parseInt(rating[i].rating);
                                                var avgRating = ratingCounter/rating.length;
                                                Moment.findByIdAndUpdate(req.params.id,
                                                    {
                                                        avgRating: avgRating
                                                    }, function (err) {
                                                        if (err)
                                                            res.send(err)
                                                        else
                                                            res.send(req.body)})
                                            })
                                        }
                                    });
                                }
                            }
                        });
                    }
                    else
                        res.send("Moment not found");
            });
        });
};

/**
 * GET to retrieve a rating related to a creative moment
 */
exports.getRating = function(req, res) {
    var auth = req.header("Authorization");
    var token = auth.split(" ")[1];
    AccessToken.findOne({ token: token }, function(err, accessToken) {
        Rating.find({userId: accessToken.userId}, 'rating, momentId', function(err, rating){
            if(err)
                res.send(err);
            else
                res.send(rating);
        });
    });
};


/**
 * reportController code
 * This controller is responsible for managing the reports given to the creative moment
 */

var Rating = require('../models/rating');
var AccessToken = require('../models/accessToken');
var Moment = require('../models/moment');
var Report = require('../models/report');
```

```
/**
 * POST to insert a new report related to a creative moment
 */
exports.postReport = function(req, res) {
    var auth = req.header("Authorization");
    var token = auth.split(" ")[1];
    AccessToken.findOne({ token: token }, function(err, accessToken) {
        var report = new Report ({
        });
        report.userId = accessToken.userId;
        report.momentId = req.query.momentId;
        report.text = req.query.text;
        report.save(function (err) {
            if (err)
                res.send(err)
            else
                res.send(req.body)
        });
    });
};


/**
 * sharedMomentController code
 * This controller is responsible for everything that has to do with the creative moments
 * that have been shared.
 */

var SharedMoment = require('../models/shared_moment');
var Moment = require('../models/moment');
var Rating = require('../models/rating');
var Spelling = require('../models/spelling');
var AccessToken = require('../models/accessToken');
var async = require('async');

/**
 * GET to retrieve shared creative moments together with their rating and speeling flags
 */
function getSharedMomentsWithRatings(req, res, sort, filters, limit, skip) {
    Moment.find(filters, {}, {limit: limit, skip: skip, sort: sort}, function(err,
sharedmoments) {
        var returnArray = [];
        var ratingEnd = false;
        var ratingCount = 0;
        var spellingEnd = false;
        var spellingCount = 0;
        var auth = req.header("Authorization");
        if(typeof auth != 'undefined') {
            var token = auth.split(" ")[1];
            AccessToken.findOne({token: token}, function (err, accessToken) {
                // Async library is used to make synchronous requests because of the return of
the callback
                async.eachSeries(sharedmoments, function (file, callback) {
                    var objString = JSON.stringify(file);
                    var obj = JSON.parse(objString);
                    Rating.findOne({userId: accessToken.userId, momentId: obj._id}, 'rating',
function (err, rating) {
                        if (rating != null) {
                            obj.rating = rating.rating;
                        }
                        else {
                            obj.rating = "0";
                        }
                        if(++ratingCount == sharedmoments.length)
                            ratingEnd = true;
                    });
                    Spelling.findOne({userId: accessToken.userId, momentId: obj._id}, function
(err, spelling) {
                        if (spelling != null) {
                            obj.spelling = "true";
                        }
```

```
                        else {
                            obj.spelling = "false";
                        }
                        returnArray.push(obj);
                        if(++spellingCount == sharedmoments.length)
                            spellingEnd = true;
                        callback();
                    });
                }, function (err) {
                    if (err)
                        res.send(err);
                    else {
                        if(!spellingEnd || !ratingEnd){
                            setTimeout(function () {
                                if(!spellingEnd || !ratingEnd) {
                                    res.send("Request problem!");
                                }
                                else
                                    res.send(returnArray);
                            }, 2000);
                        }
                        else
                            res.send(returnArray);
                    }
                });
            });
        }
        else {
            res.send(sharedmoments);
        }
    });
}

/**
 * GET to retrieve shared creative moments
 */
exports.getSharedMoments = function(req, res) {
    console.log(req.connection.encrypted);
    var filters = new Object();
    filters.share = true;
    getSharedMomentsWithRatings(req, res, {date: -1}, filters, req.query.limit, 0);
};

/**
 * GET to retrieve shared creative moments in the form of an array
 */
exports.getArraySharedMoments = function(req, res) {
    var rawcards = req.params.cards;
    var cards = rawcards.split(";");
    var response = [];
    //console.log(splicedCards);
    for(var i=0; i<cards.length; i++)
    {
        Moment.find({_id: cards[i]}, function(err, moment){
            response = moment;
        });
    }
    res.send(response);
};

/**
 * POST to share a creative moment
 */
exports.postSharedMoments = function(req, res) {
    var sharedmoment = new SharedMoment ({
    });
    sharedmoment.momentId = req.body.id;
    sharedmoment.save(function (err) {
        if (err)
            res.send(err)
        else
```

```
            res.send(req.body)
    })
};

/**
 * DELETE to delete a shared moment
 */
exports.deleteSingleSharedMoment = function(req, res) {
    SharedMoment.remove({ momentId: req.body.id }, function (err) {
        if (err)
            res.send(err)
        else
            res.send(req.body)
    });
};

/**
 * PUT to mark a creative moment as not shared
 */
exports.putSharedMoments = function(req, res) {
    Moment.findById(req.params.id, function (err, moment) {
        Moment.findByIdAndUpdate(req.params.id,
            {
                share: ((moment.share=='true') ? false : true)
            }, function (err) {
                if (err)
                    res.send(err)
                else
                    res.send(req.body)
            });
    });
};

/**
 * GET to retrieve a creative moment applying the filters selected
 */
exports.getFilteredMoments = function(req, res) {
    var filters = new Object();
    var sort = {};
    filters.share = true;
    if(req.query.emotion != "")
        filters.emotion = req.query.emotion;
    if(req.query.location != "")
        filters.location = req.query.location;
    if(req.query.tag != "")
        filters.tag = req.query.tag;
    switch(req.query.sort) {
        case "dateNew": {
            sort = {date: -1};
            break;
        }
        case "dateOld": {
            sort = {date: 1};
            break;
        }
        case "relevance": {
            sort = {avgRating: -1};
            break;
        }
        case "spelling": {
            sort = {countSpelling: -1};
            break;
        }
    }
    getSharedMomentsWithRatings(req, res, sort, filters, req.query.limit, req.query.skip);
};

/**
 * GET to retrieve a random shared moment
 */
exports.getRandomSharedMoment = function(req, res) {
```

```
    Moment.count(function(err, count){
        if(err)
            res.send(err);
        else {
            var rand = Math.floor(Math.random() * count);
            Moment.findOne({share:true}).skip(rand).exec(function(err, moment) {
                if(err)
                    res.send(err);
                else
                    res.send(moment);
            });
        }
    })
};

/**
 * Function responsible for generating a random card and putting it in an array
 */
function generateRandomCard(req, res, count, filters)
{
    if(req.query.verify == "True")
    {
        var momentsArray = [];
        Moment.find(filters).exec(function(err, moments) {
            if(err)
                res.send(err);
            else
            {
                async.eachSeries(moments, function (moment, callback) {
                    Rating.count({momentId: moment._id}, function (err, rating) {
                        if (err)
                            res.send(err);
                        if (rating >= 15 && moment.avgRating >= 4 && moment.countSpelling >=
15)
                            momentsArray.push(moment);
                    });
                    callback();
                }, function(err) {
                    if (err)
                        res.send(err);
                    if(momentsArray.length != 0)
                    {
                        var rand = Math.floor(Math.random() * momentsArray.length);
                        res.send(momentsArray[rand]);
                    }
                    else
                    {
                        res.send("");
                    }
                });
            }
        });
    }
    else
    {
        var rand = Math.floor(Math.random() * count);
        Moment.findOne(filters).skip(rand).exec(function(err, moment) {
            if(err)
                res.send(err);
            else
                res.send(moment);
        });
    }
}

/**
 * GET to retrieve a random shared moment applying the filters selected
 */
exports.getRandomFilteredSharedMoment = function(req, res) {
    var filters = new Object();
    filters.share = true;
```

```
    if(req.query.emotion != "")
        filters.emotion = req.query.emotion;
    if(req.query.location != "")
        filters.location = req.query.location;
    if(req.query.tag != "")
        filters.tag = req.query.tag;
    Moment.count(filters,function(err, count){
        if(err)
            res.send(err);
        else {
            generateRandomCard(req, res, count, filters);
        }
    })
};


/**
 * spellingController code
 * This controller is responsible for managing the spelling flags given to the creative moment
 */

var Spelling = require('../models/spelling');
var AccessToken = require('../models/accessToken');
var Moment = require('../models/moment');

/**
 * POST to insert a new spelling flag related to a creative moment
 */
exports.postSpelling = function(req, res) {
    var auth = req.header("Authorization");
    var token = auth.split(" ")[1];
    AccessToken.findOne({ token: token }, function(err, accessToken) {
        Moment.findOne({_id: req.params.id}, function (err, moment) {
            if (err)
                res.send(err);
            else
                if(moment != null)
                {
                    Spelling.findOne({userId: accessToken.userId, momentId: req.params.id},
function (err, spelling) {
                        if (err)
                            res.send(err);
                        else if(spelling != null) {
                            res.send(req.body);
                        }
                        else {
                            var spellingDOM = new Spelling({});
                            spellingDOM.userId = accessToken.userId;
                            spellingDOM.momentId = req.params.id;
                            spellingDOM.save(function (err) {
                                if (err)
                                    res.send(err);
                                else {
                                    Spelling.count({momentId: req.params.id}, function(err,
spellingCount) {
                                        if(err)
                                            res.send(err);
                                        else
                                            Moment.findByIdAndUpdate(req.params.id,
                                                {
                                                    countSpelling: spellingCount
                                                }, function (err) {
                                                    if (err)
                                                        res.send(err)
                                                    else
                                                        res.send(req.body)})
                                    });
                                }
                            });
                        }
                    });
```

```
                }
                else
                    res.send("Moment not found");
        });
    });
};

/**
 * DELETE to remove a spelling flag related to a creative moment
 */
exports.deleteSpelling = function(req, res) {
    Spelling.remove({momentId: req.params.id}, function(err){
        if(err)
            res.send(err);
        else
            Spelling.count({momentId: req.params.id}, function(err, spellingCount) {
                if(err)
                    res.send(err);
                else
                    Moment.findByIdAndUpdate(req.params.id,
                        {
                            countSpelling: spellingCount
                        }, function (err) {
                            if (err)
                                res.send(err)
                            else
                                res.send(req.body)})
            });
    })
};


/**
 * userController code
 * This controller is responsible for everything that has to do with users
 */

var User = require('../models/user');
var AccessToken = require('../models/accessToken');

/**
 *  GET to retrieve users
 */

exports.getUsersByToken = function(req, res) {
    var auth = req.header("Authorization");
    var token = auth.split(" ")[1];
    AccessToken.findOne({ token: token }, function(err, accessToken){
        User.findById(accessToken.userId, function(err, user){
            if (err)
                res.send(err)
            else
                res.send(user)
        })
    })
};

/**
 *  GET to retrieve users by id
 */

exports.getSingleUsers = function(req, res) {
    User.findById(req.params.id, function (err, user) {
        if (err)
            res.send(err)
        else
            res.send(user)
    });
};

/**
```

```
 * POST to add users
 */
exports.postUsers = function(req, res) {
    var user = new User ({
    });
    user.username = req.body.username;
    user.password = user.encryptPassword(req.body.password);
    user.first_name = req.body.firstname;
    user.last_name = req.body.lastname;
    user.email = req.body.email;
    user.birth_date = req.body.birthdate;
    user.profession = req.body.profession;
    user.gender = req.body.gender;
    user.country = req.body.country;
    user.save(function (err) {
        if (err)
            res.send(err)
        else
            res.send(req.body)
    });
};

/**
 * DELETE to delete a user by id
 *
 */
exports.deleteSingleUsers = function(req, res) {
    User.remove({ _id: req.params.id }, function (err) {
        if (err)
            res.send(err)
        else
            res.send(req.body)
    });
};

/**
 * PUT to update a user by id
 */
exports.putSingleUsers = function(req, res) {
    User.findByIdAndUpdate(req.params.id,
        { username:req.body.username,
            password:req.body.password,
            auth_key:req.body.auth_key }, function (err) {
            if (err)
                res.send(err)
            else
                res.send(req.body)})
};

/**
 * DELETE to logout a user
 */
exports.logout = function(req, res) {
    var auth = req.header("Authorization");
    var token = auth.split(" ")[1];
    AccessToken.remove({token: token}, function (err) {
        if (err)
            res.send(err)
        else
            res.send(req.body)
    });
};
```

Models

```
/**
 * accessToken Model code
 * This model is responsible for defining the schema for
 * the accessToken documents
 */
```

```
var mongoose = require ('mongoose');

var accessTokenSchema = new mongoose.Schema({
    userId: {
        type: String,
        required: true
    },
    clientId: {
        type: String,
        required: true
    },
    token: {
        type: String,
        unique: true,
        required: true
    },
    created: {
        type: Date,
        default: Date.now
    }
});

module.exports = mongoose.model('AccessToken', accessTokenSchema);


/**
 * client Model code
 * This model is responsible for defining the schema for
 * the client documents
 */

var mongoose = require ('mongoose');

var clientSchema = new mongoose.Schema({
    name: {
        type: String,
        unique: true,
        required: true
    },
    clientId: {
        type: String,
        unique: true,
        required: true
    },
    clientSecret: {
        type: String,
        required: true
    }
});

module.exports = mongoose.model('Client', clientSchema);


/**
 * moment Model code
 * This model is responsible for defining the schema for
 * the moment documents
 */

var mongoose = require ("mongoose");
var Rating = require('../models/rating');

var momentSchema = new mongoose.Schema({
    userId: {type: String, required: true},
    title: {type: String, required: true},
    text: {type: String, required: true},
    location: {type: String, required: true},
    emotion: {type: String, required: true},
    emotionIcon: {type: String, required: true},
    date: {type: Date, default: Date.now},
```

```
    image: {type: String, required: false},
    author: {type: String, required: true},
    sound: {type: String, required: true},
    share: {type: String, default: false},
    avgRating: {type: String, default: 0},
    countSpelling: {type: String, default: 0},
    tag: {type: [String], required: true}
});

module.exports = mongoose.model('Moment', momentSchema);


/**
 * rating Model code
 * This model is responsible for defining the schema for
 * the rating documents
 */

var mongoose = require ("mongoose");

var ratingSchema = new mongoose.Schema({
    userId: {type: String, required: true},
    momentId: {type: String, required: true},
    rating: {type: String, required: true}
});

module.exports = mongoose.model('Rating', ratingSchema);


/**
 * report Model code
 * This model is responsible for defining the schema for
 * the report documents
 */

var mongoose = require ("mongoose");

var reportSchema = new mongoose.Schema({
    userId: {type: String, required: true},
    momentId: {type: String, required: true},
    text: {type: String, required: true}
});

module.exports = mongoose.model('Report', reportSchema);


/**
 * spelling Model code
 * This model is responsible for defining the schema for
 * the spelling documents
 */

var mongoose = require ("mongoose");

var spellingSchema = new mongoose.Schema({
    userId: {type: String, required: true},
    momentId: {type: String, required: true}
});

module.exports = mongoose.model('Spelling', spellingSchema);


/**
 * user Model code
 * This model is responsible for defining the schema for
 * the user documents and encrypting the users password
 */

var mongoose            = require ('mongoose');
var crypto              = require('crypto');
```

```
var userSchema = new mongoose.Schema({
    username: {type: String, required: true, unique: true},
    password: {type: String, required: true},
    salt: {type: String, required: true, unique: true, default:
crypto.randomBytes(24).toString('hex')},
    first_name: {type: String, required: true},
    last_name: {type: String, required: true},
    email: {type: String, required: true},
    country: {type: String, required: true},
    gender: {type: String, required: true},
    birth_date: {type: String, required: true},
    profession: {type: String, required: true},
    created: {type: Date, default: Date.now}
});

userSchema.methods.encryptPassword = function(password) {
    return crypto.createHmac('sha256', this.salt).update(password).digest('hex');
};

userSchema.methods.verifyPassword = function(password) {
    return this.encryptPassword(password) === this.password;
};


module.exports = mongoose.model('User', userSchema);
```

App.js

```
/**
 * This file is responsible for managing the dependences, routings and MVC structure
 */

var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var passport = require('passport');
var mongoose = require ('mongoose');

var router = express.Router();

var uristring =
'mongodb://admin:MDBadmin@ds031271.mongolab.com:31271/heroku_app32978819?authMode=scram-sha1';
mongoose.connect(uristring, function (err, res) {
    if (err) {
        console.log ('ERROR connecting to: ' + uristring + '. ' + err);
    } else {
        console.log ('Succeeded connected to: ' + uristring);
    }
});

var routes = require('./routes/index');

var userController = require('./controllers/userController');
var sharedMomentController = require('./controllers/sharedMomentController');
var momentController = require('./controllers/momentController');
var ratingController = require('./controllers/ratingController');
var spellingController = require('./controllers/spellingController');
var reportController = require('./controllers/reportController');
var authController = require('./controllers/authController');
var oauth2 = require('./controllers/oauth2Controller');

var app = express();

app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');

app.use(logger('dev'));
```

```
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));
app.use(passport.initialize());

app.use('/', routes);

router.route('/users')
    .post(userController.postUsers)
    .get(authController.isBearerAuthenticated, userController.getUsersByToken);
router.route('/sharedMoments/:id')
    .put(authController.isBearerAuthenticated, sharedMomentController.putSharedMoments);
router.route('/sharedMoments')
    .get(sharedMomentController.getSharedMoments);
router.route('/randomSharedMoments')
    .get(sharedMomentController.getRandomSharedMoment);
router.route('/randomFilteredSharedMoments')
    .get(sharedMomentController.getRandomFilteredSharedMoment);
router.route('/filteredMoments')
    .get(sharedMomentController.getFilteredMoments);
router.route('/moments/:id')
    .get(authController.isBearerAuthenticated, momentController.getSingleMoment)
    .put(authController.isBearerAuthenticated, momentController.putSingleMoment);
router.route('/rating/:id')
    .post(authController.isBearerAuthenticated, ratingController.postRating);
router.route('/rating')
    .get(authController.isBearerAuthenticated, ratingController.getRating);
router.route('/spelling/:id')
    .post(authController.isBearerAuthenticated, spellingController.postSpelling)
    .delete(authController.isBearerAuthenticated, spellingController.deleteSpelling);
    router.route('/moments')
    .post(authController.isBearerAuthenticated, momentController.postMoments)
    .get(authController.isBearerAuthenticated, momentController.getMoments)
    .delete(authController.isBearerAuthenticated, momentController.deleteSingleMoment);
router.route('/users/:id')
    .get(authController.isBearerAuthenticated, userController.getSingleUsers)
    .delete(authController.isBearerAuthenticated, userController.deleteSingleUsers)
    .put(authController.isBearerAuthenticated, userController.putSingleUsers);
router.route('/oauth/token')
    .post(oauth2.token);
router.route('/logout')
    .delete(authController.isBearerAuthenticated, userController.logout);
router.route('/report')
    .get(authController.isBearerAuthenticated, reportController.postReport);

app.use('/', router);

module.exports = app;
```

Appendix B – CreativeWall source code

Controller

```
/**
 * sharedMomentController code
 * This controller is responsible for everything that has to do with the creative moments
 * that have been shared.
 */

var SharedMoment = require('../models/shared_moment');
var Moment = require('../models/moment');
var Rating = require('../models/rating');
var Spelling = require('../models/spelling');
var AccessToken = require('../models/accessToken');
var async = require('async');

/**
 * GET to retrieve shared creative moments together with their rating and speeling flags
 */
function getSharedMomentsWithRatings(req, res, sort, filters, limit, skip) {
    Moment.find(filters, {}, {limit: limit, skip: skip, sort: sort}, function(err,
sharedmoments) {
        var returnArray = [];
        var ratingEnd = false;
        var ratingCount = 0;
        var spellingEnd = false;
        var spellingCount = 0;
        var auth = req.header("Authorization");
        if(typeof auth != 'undefined') {
            var token = auth.split(" ")[1];
            AccessToken.findOne({token: token}, function (err, accessToken) {
                // Async library is used to make synchronous requests because of the return of
the callback
                async.eachSeries(sharedmoments, function (file, callback) {
                    var objString = JSON.stringify(file);
                    var obj = JSON.parse(objString);
                    Rating.findOne({userId: accessToken.userId, momentId: obj._id}, 'rating',
function (err, rating) {
                        if (rating != null) {
                            obj.rating = rating.rating;
                        }
                        else {
                            obj.rating = "0";
                        }
                        if(++ratingCount == sharedmoments.length)
                            ratingEnd = true;
                    });
                    Spelling.findOne({userId: accessToken.userId, momentId: obj._id}, function
(err, spelling) {
                        if (spelling != null) {
                            obj.spelling = "true";
                        }
                        else {
                            obj.spelling = "false";
                        }
                        returnArray.push(obj);
                        if(++spellingCount == sharedmoments.length)
                            spellingEnd = true;
                        callback();
                    });
                }, function (err) {
                    if (err)
                        res.send(err);
                    else {
                        if(!spellingEnd || !ratingEnd){
                            setTimeout(function () {
                                if(!spellingEnd || !ratingEnd) {
                                    res.send("Request problem!");
                                }
                                else
```

```
                                    res.send(returnArray);
                                }, 2000);
                            }
                            else
                                res.send(returnArray);
                        }
                    });
                });
            }
            else {
                res.send(sharedmoments);
            }
        });
}

/**
 * GET to retrieve shared creative moments
 */
exports.getSharedMoments = function(req, res) {
    console.log(req.connection.encrypted);
    var filters = new Object();
    filters.share = true;
    getSharedMomentsWithRatings(req, res, {date: -1}, filters, req.query.limit, 0);
};

/**
 * GET to retrieve shared creative moments in the form of an array
 */
exports.getArraySharedMoments = function(req, res) {
    var rawcards = req.params.cards;
    var cards = rawcards.split(";");
    var response = [];
    //console.log(splicedCards);
    for(var i=0; i<cards.length; i++)
    {
        Moment.find({_id: cards[i]}, function(err, moment){
            response = moment;
        });
    }
    res.send(response);
};

/**
 * POST to share a creative moment
 */
exports.postSharedMoments = function(req, res) {
    var sharedmoment = new SharedMoment ({
    });
    sharedmoment.momentId = req.body.id;
    sharedmoment.save(function (err) {
        if (err)
            res.send(err)
        else
            res.send(req.body)
    })
};

/**
 * DELETE to delete a shared moment
 */
exports.deleteSingleSharedMoment = function(req, res) {
    SharedMoment.remove({ momentId: req.body.id }, function (err) {
        if (err)
            res.send(err)
        else
            res.send(req.body)
    });
};

/**
 * PUT to mark a creative moment as not shared
```

```
 */
exports.putSharedMoments = function(req, res) {
    Moment.findById(req.params.id, function (err, moment) {
        Moment.findByIdAndUpdate(req.params.id,
            {
                share: ((moment.share=='true') ? false : true)
            }, function (err) {
                if (err)
                    res.send(err)
                else
                    res.send(req.body)
            });
    });
};

/**
 * GET to retrieve a creative moment applying the filters selected
 */
exports.getFilteredMoments = function(req, res) {
    var filters = new Object();
    var sort = {};
    filters.share = true;
    if(req.query.emotion != "")
        filters.emotion = req.query.emotion;
    if(req.query.location != "")
        filters.location = req.query.location;
    if(req.query.tag != "")
        filters.tag = req.query.tag;
    switch(req.query.sort) {
        case "dateNew": {
            sort = {date: -1};
            break;
        }
        case "dateOld": {
            sort = {date: 1};
            break;
        }
        case "relevance": {
            sort = {avgRating: -1};
            break;
        }
        case "spelling": {
            sort = {countSpelling: -1};
            break;
        }
    }
    getSharedMomentsWithRatings(req, res, sort, filters, req.query.limit, req.query.skip);
};

/**
 * GET to retrieve a random shared moment
 */
exports.getRandomSharedMoment = function(req, res) {
    Moment.count(function(err, count){
        if(err)
            res.send(err);
        else {
            var rand = Math.floor(Math.random() * count);
            Moment.findOne({share:true}).skip(rand).exec(function(err, moment) {
                if(err)
                    res.send(err);
                else
                    res.send(moment);
            });
        }
    })
};

/**
 * Function responsible for generating a random card and putting it in an array
 */
```

```
function generateRandomCard(req, res, count, filters)
{
    if(req.query.verify == "True")
    {
        var momentsArray = [];
        Moment.find(filters).exec(function(err, moments) {
            if(err)
                res.send(err);
            else
            {
                async.eachSeries(moments, function (moment, callback) {
                    Rating.count({momentId: moment._id}, function (err, rating) {
                        if (err)
                            res.send(err);
                        if (rating >= 15 && moment.avgRating >= 4 && moment.countSpelling >=
15)
                            momentsArray.push(moment);
                    });
                    callback();
                }, function(err) {
                    if (err)
                        res.send(err);
                    if(momentsArray.length != 0)
                    {
                        var rand = Math.floor(Math.random() * momentsArray.length);
                        res.send(momentsArray[rand]);
                    }
                    else
                    {
                        res.send("");
                    }
                });
            }
        });
    }
    else
    {
        var rand = Math.floor(Math.random() * count);
        Moment.findOne(filters).skip(rand).exec(function(err, moment) {
            if(err)
                res.send(err);
            else
                res.send(moment);
        });
    }
}

/**
 * GET to retrieve a random shared moment applying the filters selected
 */
exports.getRandomFilteredSharedMoment = function(req, res) {
    var filters = new Object();
    filters.share = true;
    if(req.query.emotion != "")
        filters.emotion = req.query.emotion;
    if(req.query.location != "")
        filters.location = req.query.location;
    if(req.query.tag != "")
        filters.tag = req.query.tag;
    Moment.count(filters,function(err, count){
        if(err)
            res.send(err);
        else {
            generateRandomCard(req, res, count, filters);
        }
    })
};
```

Models

```
/**
 * Model Request is responsible for performing http requests
 */
class Request extends CFormModel
{
    /**
     * Declares the validation rules.
     * The rules state that username and password are required,
     * and password needs to be authenticated.
     */
    public function rules()
    {
        return array(
        );
    }

    /**
     * Declares attribute labels.
     */
    public function attributeLabels()
    {
        return array(
        );
    }

    /**
     * Function responsible for performing all the httprequests to the api
     * @param array $parameters - parameters to be included in the request
     * @param string $type - type of request
     * @param string $url - url to which the request should be performed
     * @param string $headers - authentication headers to be included in the request
     * @return mixed - result of the requests converted from json to array
     */
    public function makeHttpRequest($parameters=array(), $type="POST", $url="", $headers="")
    {
        Yii::import('ext.ehttpclient.*');
        $httpclient = new EHttpClient($url,
            array(
                'maxredirects' => 0,
                'timeout' => 30
            ));

        if ($headers != "") {
            $httpclient->setHeaders("Authorization", $headers);
        }

        switch ($type) {
            case "POST":
                $httpclient->setParameterPost($parameters);
                $response = $httpclient->request("POST");
                break;
            case "GET":
                $httpclient->setParameterGet($parameters);
                $response = $httpclient->request("GET");
                break;
            case "PUT":
                $httpclient->setParameterPost($parameters);
                $response = $httpclient->request("PUT");
                break;
            case "DELETE":
                $httpclient->setParameterPost($parameters);
                $response = $httpclient->request("DELETE");
                break;
        }
        return json_decode($response->getBody(), true);
    }
}
```

```php
/**
 * Model Card is responsible for managing creative moments
 */
class Card extends CFormModel
{
    public $id;
    public $title;
    public $text;
    public $userId;
    public $date;
    public $location;
    public $emotion;
    public $emotionIcon;
    public $image;
    public $tag;
    public $report;

    /**
     * Declares the validation rules for each of the fields
     */
    public function rules()
    {
        return array(
            array('title, text, location, emotion', 'required', 'on'=>'add'),
            array('title, text, location, emotion', 'required', 'on'=>'edit'),
            array('report', 'required', 'on'=>'report')
        );
    }

    /**
     * Declares attribute labels.
     */
    public function attributeLabels()
    {
        return array(
            'id'=>'Id',
            'title'=>'Title',
            'text'=>'Text',
            'userId'=>'UserId',
            'date'=>'Date',
            'location'=>'Location',
            'emotion'=>'Emotion',
            'emotionIcon'=>'Emotion Icon',
            'image'=>'Image',
            'tag'=>'Tag (separated by ",")',
            'report'=>'Report Reason',
        );
    }
}


/**
 * Model User is responsible for managing user operations
 */
class User extends CFormModel
{
    public $username;
    public $email;
    public $password;
    public $retype;
    public $firstname;
    public $lastname;
    public $birthdate;
    public $profession;
    public $gender;
    public $country;

    /**
     * Declares the validation rules
     */
```

```
    public function rules()
    {
        return array(
            array('username, email, password, firstname, lastname, birthdate, gender, country,
retype, profession',
                'required', 'on'=>'register'),
            array('username, password', 'required', 'on'=>'login'),
            array('password', 'length', 'min'=>5, 'message'=>'Password is too short',
'on'=>'login'),
            array('password', 'length', 'min'=>5, 'message'=>'Password is too short',
'on'=>'register'),
            array('password', 'authenticate', 'on'=>'login'),
            array('email', 'email', 'on'=>'register'),
            array('firstname, lastname, country, profession', 'match',
                'pattern' => '/^[a-zA-Z\s]+$/',
                'message' => '{attribute} can only contain letters'),
            array('birthdate', 'type', 'type' => 'date', 'message' => '{attribute} is not a
valid date',
                'dateFormat' => 'dd-MM-yyyy'),
            array('retype', 'validatePassword', 'on'=>'register'),
            array('retype, password', 'safe')
        );
    }

    /**
     * Function responsible for validating the password
     * @param $attribute - attribute where the error should be shown
     */
    public function validatePassword($attribute)
    {
        if($this->password != $this->retype)
            $this->addError($attribute, 'Retype must be the same as Password');
    }

    /**
     * Declares attribute labels
     */
    public function attributeLabels()
    {
        return array(
            'username'=>'Username',
            'email'=>'Email',
            'password'=>'Password',
            'retype'=>'Retype Password',
            'firstname'=>'First Name',
            'lastname'=>'Last Name',
            'birthdate'=>'Date of Birth',
            'profession'=>'Profession',
            'gender'=>'Gender',
            'country'=>'Country'
        );
    }

    /**
     * Function responsible for authenticating the client and the user
     * @param $attribute - not used
     * @param $params - not used
     */
    public function authenticate($attribute,$params)
    {
        $request = new Request();
        $array = $request->makeHttpRequest(
            array(
                "grant_type" => "password",
                "client_id" => "webV1",
                "client_secret" => "web123456",
                "username" => $this->username,
                "password" => $this->password),
            "POST",
            "https://api-cw.herokuapp.com/oauth/token",
            "");
```

```
        if(isset($array["error"]) && $array["error"] == "invalid_grant")
            $this->addError('username','Incorrect username or password.');
        else {
            Yii::app()->request->cookies['_token'] = new CHttpCookie('_token',
$array["access_token"]);
            $array = $request->makeHttpRequest(
                array(),
                "GET",
                "https://api-cw.herokuapp.com/users",
                "Bearer ".Yii::app()->request->cookies['_token']->value);
            Yii::app()->user->setState('author', $array['first_name']."
".$array['last_name']);
            Yii::app()->user->setState('username', $array['username']);
        }
    }
}
```