# 

# Abstract

Max 500 words / 1 page. Brief summary of contents and report structure. How work was performed and main outcomes.

Table of Contents

Abstract 1

1.0 Introduction and Objectives 4

2.0 Background 5

3.0 Product Backlog 6

4.0 Overall Design 7

4.1 Design Details 7

4.2 Implementation Details 7

5.0 Sprint Documentation 8

5.1 Sprint 1 8

5.2 Sprint 2 8

5.3 Sprint 3 8

5.4 Sprint 4 8

6.0 Project Review 9

7.0 Conclusions and Summary 10

8.0 References 11

9.0 Appendices 12

9.1 Appendix A: 12

9.2 Appendix B: 12

**INSERT CUSTOM TABLE OF FIGURES HERE WHEN FINISHED.**

# Introduction and Objectives

Max 2 pages. What report is about. Project objectives, constraints, scope and delimitations. Approach used to complete work. Structure of report. Set the scene for rest of report (context).

# Background

Max 5 pages. Describe initial project management approach and plan. Discuss solution approaches considered. Describe other similar systems and possible solutions you have looked at. Describe tools considered for development and for SCRUM. Provide reason for choice of tools (Salami made us). Pictues of project mockup.

The requirements volatility was quite low throughout the project, with minimal changes to the requirements by the product owner after the initial specification.

Talk about pivotal tracker.

# Product Backlog

Describe analysis and grouping of user stories provided. Modifications made to user stoires before sprints started. Use pictures as appropriate.

List entire product backlog and groupings. Product backlog after every sprint ( minus stories completed).

Burndown chart! How product backlog changed over time.

# Overall Design

Max 10 pages

## 4.1 Design Details

*Relevant design details. Uml, er diagram etc. provide reasons for choices. Report changes to user stores due to the design process. So many ER diagrams + relations*

*UI design, CSS and template design*

* Design Introduction
* Backend design
  + Submissions
    - Creation
    - Deletion
    - Modification
  + Comments/improvements
  + Voting
  + Reward system
* UI design
  + Goals
  + Design change over time
  + Final design

After initially reading through and understanding the requirements for the project, each user story was created and placed in a sub-group: admin, user accounts, submissions, voting, commenting or rewards.

It was decided quite early on in the project to implement the website using the Python web-framework, Django. This would allow for easy to use and powerful admin features, along with well-defined ways to implement all the functionality deemed necessary for the project to work (ADD MORE REASONS WHY USED DJANGO). Django’s model system seemed very applicable for creating objects such as users, submissions and comments which were central to the overall design.

Figure x shows the UML diagram for the design of the project.

## Back-end

One of the major design issues that was central to the project as a whole was the submission. The submission model needed to store information about a user-generated idea including the title, author, category, the idea itself and links to external webpages. The majority of the project features in some way interact with the submission, be that viewing, commenting on, suggesting improvements or voting. As well, submissions had to be both editable and removable by the correctly privileged users/admin.

We wanted a user to be able to create an account, storing basic personal and contact information that could be used to identify the user and for them to participate in posting submissions and commenting/voting on submissions. A user would have to be currently logged onto the system to be able to submit, comment and vote on submissions. However, submissions could be viewed by anyone of the general public without requiring a registered account.

Comments and improvements were added for users to be able to give and receive feedback on ideas that were submitted. Improvements added onto the idea of comments but were designed for providing constructive criticism on how to improve or alter the current submitted idea.

## User Interface

The interface design went through numerous iterations before eventually settling on the current design.

The main aims of the interface design were to create a sleek, modern interface which was easy to navigate and to use.

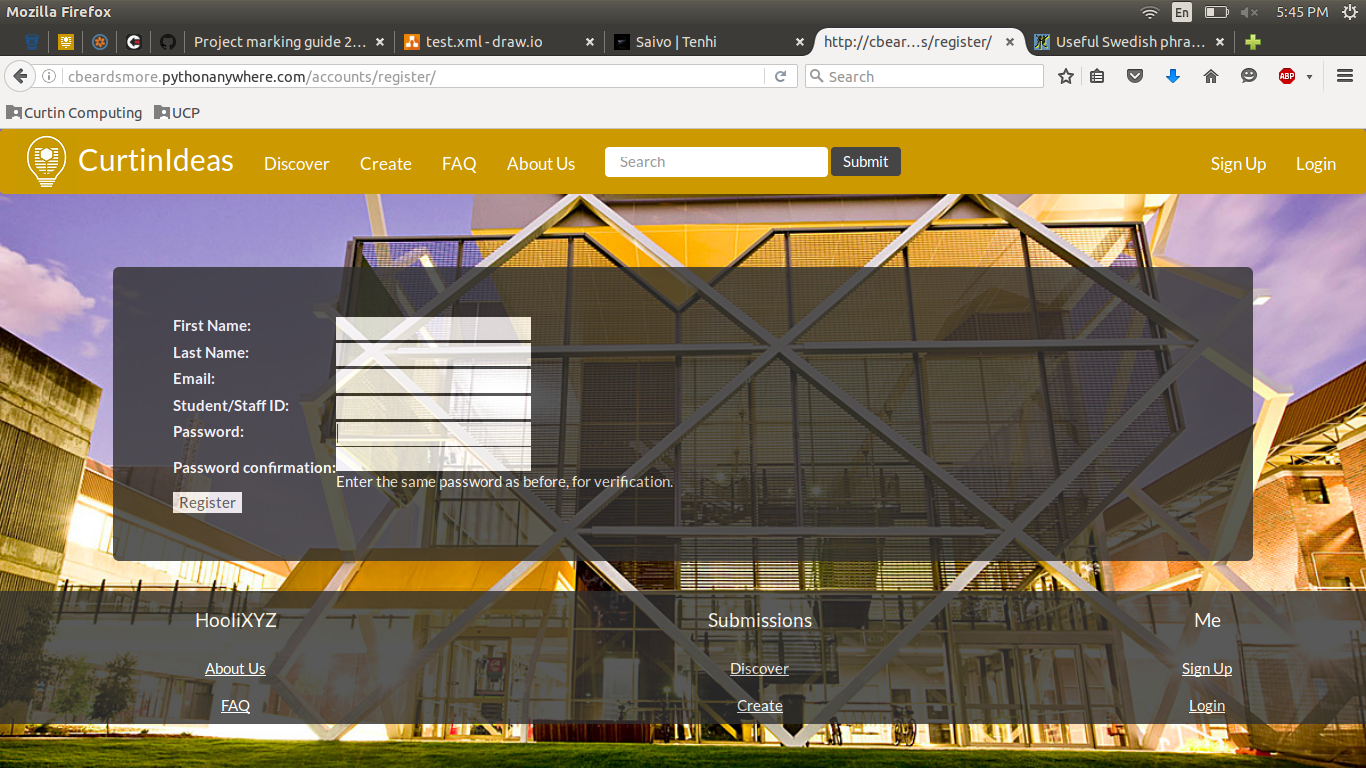
## 4.2 Implementation Details

Document the implementation including tasks, effort, features and task distribution (planned and actually completed). Screenshots of functionality. Changes to initial design. What testing was done? \*lol none.

As discussed previously, user stories were broken down into different groupings. The task of implementing the stories within these groupings was distributed to different group members, usually in pairs or small groups.

Accounts

The plan was to initially implement the administration and basic user account functionality to form the basis of the project. The tasks required to set up user accounts included: creating a user model which in Django is used to represent a database object with object fields and creating a login form for users to enter the required personal information. These tasks were given a high effort rating due to their integral nature to the functionality of the project. An account creation form was made to allow the user to input the required information such as name, email, student/staff id and password (Figure x). All the other information is automatically added to the created user object and the object is then stored in the database.

Figure x User creation form.

Once these foundational tasks were accomplished, more user account functionality was built in, like changing passwords and browsing other user profiles.

Over the life of the project, there were minimal changes to the account creation form and overall functionality, with only a few UI improvements being added later on. All of the planned functionality from the user stories was successfully implemented.

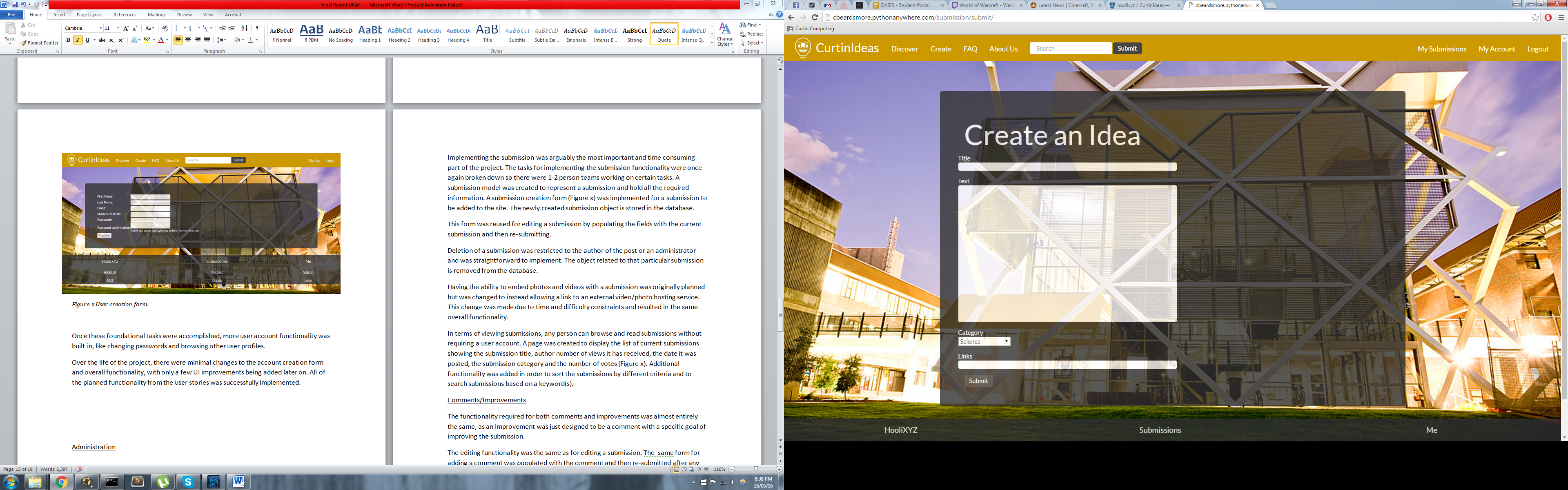


Figure x Submission creation form.

Administration

The default Django administration platform was used to allow administrators to perform tasks such as creating a user, editing and removing submissions.

Submissions

Implementing the submission was arguably the most important and time consuming part of the project. The tasks for implementing the submission functionality were once again broken down so there were 1-2 person teams working on certain tasks. These tasks included: creating a submission model, creating a submission form, updating the database with submission objects, adding edit and remove buttons, creating a page to list submissions and adding a sorting function.

The submission model was created to represent a submission and hold all the required information. The next task was to create a submission form (Figure x) for a submission to be added to the site. The newly created submission object is stored in the database.

This form was reused for editing a submission by populating the fields with the current submission and then re-submitting.

Deletion of a submission was restricted to the author of the post or an administrator and was straightforward to implement. The object related to that particular submission is removed from the database.

Having the ability to embed photos and videos with a submission was originally planned but was changed to instead allowing a link to an external video/photo hosting service. This change was made due to time and difficulty constraints and resulted in the same overall functionality.

In terms of viewing submissions, any person can browse and read submissions without requiring a user account. A page was created to display the list of current submissions showing the submission title, author number of views it has received, the date it was posted, the submission category and the number of votes (Figure x). Additional functionality was added in order to sort the submissions by different criteria and to search submissions based on a keyword(s).

Comments/Improvements

Each submission had the ability to receive comments and improvements from other student/staff users. The task breakdown for being able to add a comment included creating a comment model, creating a form to handle the comment input and linking the comment object to the target submission object. At this stage of the project, a lot of similar functionality had already been added for other sections and so the efforts given for the implementation reflected this. To declare whether a comment is also an improvement, a tick box was added to indicate if the user intends for their comment to be an improvement.

The editing and deleting functionality were the same as that for a editing and deleting a submission. The form used for adding a comment was populated with the comment and then re-submitted after any edits were completed.



Figure x The list of submissions showing the voting arrows and voting totals on the left and the sorting criteria at the top.

Voting

The voting functionality was implemented for both submissions and comments. The tasks that were assigned included: adding clickable arrow images to represent the up-voting and down-voting features (Figure x), adding and updating the overall total by subtracting the total down-votes from the total up-votes and greying out a user’s ability to vote multiple times on the same submission/comment. Voting was restricted to users with accounts and only one vote per user per submission/comment. Voting was initially implemented as two fields; up-vote and down-vote, in the submission and comment model before being moved to their own models, one for submission voting and one for comment voting. The decision to move the voting functionality to a separate model (CONNOR WHY??).

Rewards

The reward system was the final feature to be implemented after all the other features were completed and working. As previously discussed, the reward system was designed to ‘reward’ users for creating submissions and voting/commenting on other users submissions. The plan was to implement a points system to keep track of how active a user is by allocating points based on submissions, comments and votes. The allocation of points was set to 5 for posting a submission, 3 for commenting on a submission and 1 for voting on a submission. Initially, a user would start at the base level with 0 points and every 20 points gained, a new ‘level’ was reached. This was implemented by adding two extra fields to each user object to track points and their current level.

# Sprint Documentation

## 5.1 Sprint 1

## 5.2 Sprint 2

## 5.3 Sprint 3

## 5.4 Sprint 4

# Project Review

Max 5 pages. Post-mortem discussion on product development and SCRUM process. What went right, what went wrong, what was learned (so wanky). Suggestions for improving project experience (for future students).

# Conclusions and Summary

Max 2 pages. Summary of what report has covered. Describe what was achieved in product development. Describe what was gained in project experience. Suggestions to product owner to consider in future releases. Expansions!

# References

# 9.0 Appendices

So many screenshots. Pivotal analytics and csv of everything. Complete product backlog, when stories where started and completed etc.

## 9.1 Appendix A:

## 9.2 Appendix B:

|  |  |  |
| --- | --- | --- |
|  |  | |
|  | |  |