Group 8, Cluster C, SEGP 1, Andrew Nguyen, Qun Xu, Kamal Arieff Ahmad Faizel

Final Report

Past Project Showcase and Engagement Website

Table of Contents

1 Introduction

1.1 Background

Apart of the Software Engineering Degree at The University of Adelaide is that fourth year students undertake group projects with direct involvement with industry clients. This engagement between the faculty and industry while important, poses many difficulties such as relations, project compatibility, time, communications and more.

1.2 Aim

To eliminate and or reduce the difficulties involved by providing an accessible system for industry to view projects completed by students, allow industry to submit project proposals along with a single platform for all communications regarding their corresponding proposal.

2 Specification

2.1 Project Gallery

The project gallery page showcases all of the past projects in a grid-styled gallery with pictures and the title linked to each projects. When the pictures or the title of the projects is clicked, the user is redirected to the specific project page. There is also a working search field for users to enter keywords to search for the specific projects. Besides that, there is also a filter function for users to filter the projects according to these specifications:

- Year
- Category

2.2 Specific project page

The user is redirected to the specific project page when it is clicked from the project gallery page. The details of the past project can be seen on this page. The details shown are:

- Project title
- Project summary
- Finished year of the project
- Category of project
- Last updated date of the project
- Project image
- Client name
- Client image
- Client summary
- Client link
- Student details

2.3 Specific project page edit

The specific project page is viewed differently depending on the session of the user. The view for course coordinators is different from the public user's view. When the course coordinator is logged in, there is an edit and delete button for the course coordinator to edit or delete the specific project.

Course coordinators can edit the details of the project directly from the specific project page. Once in the edit page, course coordinators can edit almost all the details that were described in previous section. Those details are:

- Project title
- Project summary
- Finished year of the project
- Category of project
- Project image
- Client name
- Client image
- Client summary
- Client link
- Student details
    - Student ID
    - Student name
    - Student email
    - Student course

Besides that, there is also an option to add more students or delete the existing students.

2.4 Adding or deleting students

When adding more students, the course coordinator is redirected to the add student page. This page is a distinctly separate page from the add project page. The add student page displays the text field for the student attributes which are:

- Student ID
- Student name
- Student email
- Student course

This will then be saved to the students table which each student linked to the group projects via a student project ID which is auto generated. The action of deleting a student is done by removing the student attributes from the students table. This will not create a database issue as the student project ID will also be deleted as one of the attributes of the student.

2.5 Specific project page delete

Course coordinators are also authorized to delete the specific project page. Upon clicking the delete button, a prompt will appear to confirm the action of the course coordinators. Deletion of a project will remove the project entry from it's respective table.

2.6 Past Projects Management Systems

A Past Projects Management Systems is also implemented to enable course coordinators to manage multiple projects at once. The course coordinators can add multiple projects one-by-one, edit multiple projects which works with the use of drop down boxes with JavaScript, and delete multiple projects.

2.7 Changes made to the original requirements

No changes were made to the original requirements as the drawn up requirements from Semester 1 were strictly followed and delivered the product according to it.

3 Design

As the fundamental aim of the project is to advertise software engineering group past projects, which has been done in last ten years, to industry partner or general visitors, this project mainly focuses on how past project records' are displayed to visitors, how past project records' relevant fetch behaviors (e.g., search past projects, filter past projects, click and redirect to detailed past project show page, etc.) are operated by visitors, and how past projects are edited or maintained by web administrator, and so on. The architectural design details will be introduced in the following section part.

3.1 Project Architecture

3.1.1 Models Structure

With the consideration of focus points, three models of the project are implemented, which are StudentPastProject, Students and PastProjectManagement models, and an extra model was also implemented called Contact which is used for the contact page only. Applied with the four models, nearly all functionalities of the project needs can be achieved.

The main three models mentioned above are the core data for the project, and all front-end interface displays or operations are executed based on them. Some models' records are initially stored in the Database and during the server-running period, the controller for each model can read or modify the already stored records, even create new records for models, and the views as the models' interfaces clearly demonstrate or show up the models' records at the same time.

From the schema of models structure (can be found in Appendix 9.2), the three models are correlative: a PastProject has many Students who have undertaken this PastProject, and Students belong to one PastProject as well, and the top level model PastProjectManagement handles the multiple modify actions for PastProjects.

The create, read, update and delete behaviors of Students are performed indirectly from the modify of PastProjects, and there is no any isolate Student can be modified without the introduce of PastProjects, which means adding, destroying or updating a new Pastproject record will add, delete or update the correlative Student as well. In the project, it is unexpected to modify a Student independently, and all Students must have worked for a correlative PastProject, so there is no point to permit modifying Student alone.

A PastProject can perform modify actions itself, and PastProjectManagement can also perform modify actions for PastProjects, the difference between the two modify behaviors are the numbers of PastProjects to be modified: PastProjectManagement is capable of editing multiple PastProjects simultaneously, while PastProject cannot do that without the help of a parent model, and this is also one of reasons for introducing an extra management model for the project.

By default, there is only one record, which has been seeded into Database, for PastProjectManagement model with 'id=0'. The only one record of management model is enough to handle all nested child attributes: 'pastproject_attributes [student_attributes]'. Initially, five PastProject records are also seeded.

3.1.2 Dynamic behaviors

Depending on the 'heart' models that have been created, it is possible to perform dynamic behaviors for the project by calling models' controllers. The relevant controllers for each model have been automatically generated by the rails framework during the generation of model.

In PastProject controller, it is provided CRUD behaviors handle methods to help with the operations of modifying or reading a PastProject, and other helpful methods that are expected to meet specific requirements, such as 'managehandle' method (used for executing the further manage actions in management system page), 'stats' method (used for showing up the stats data for PastProject) and 'search' method (used for searching any PastProject) and 'index' method (used for displaying all PastProject lists).

In Student controller, the basic modify behavior methods are provided as needed, and in PastProjectManagement controller, the multiple edit methods for PastProject are created as well. The parent model's controller makes the editing of many PastProject records possible.

As the 'brains' of models, all the controllers above are capable of processing the models records and transmitting the necessary data to appointed view interfaces, which set up the communication bridges between models and views.

3.1.3 External Interfaces

For each method of controllers, it is provided the correlative embedded ruby view page. The view interfaces demonstrate the contents that is supposed to show for all visitors, including administrators. Some specific contents or feature interfaces are permitted for use only to administrators, such as add, edit or delete a past project, and multiple past projects management interfaces, as these core functions of maintaining PastProjects authorised for administrators, and all general visitors are not permitted to use these functions, even their interfaces would not show up to general visitors (Only when administrator sign in successfully, the views can be launched). As for the non-management feature interfaces, such as general information for SEGP, contact details, all PastProjects demonstration, etc., they are viewable to all users.

Inside each view interface, it might process some data provided by its controller method or all models, and output the data to viewers later. There may be data that interfaces can fetch outside of controller methods of themselves, except for the data that is directly passed from models or has been already stored in local sessions, for instance, user login session. Any models data passed from other controller methods that interfaces need to reveal for viewers is either transferred from methods of themselves, or session parameters. In addition, Rails routing configure all site addresses of view interfaces, so each accessible interface route has to be configured beforehand, if not, the site would return 404 not found page to viewers (e.g., any user cannot HTTP GET update interfaces, all update interfaces' routes are set to be POST/PATCH method, then there is no way to access them by GET method). By the setting of routes, all interfaces are connected to assigned controller methods.

As the 'eyes' of models, the external interfaces present exactly the same contents converted from embedded ruby view files. It can demonstrate arbitrary contents, which is needed, to viewers.

3.1.4 Third Party Components

Throughout the implementation stage, all functionality were broken down and assessed on how to best execute said function. Often times, gems or third party components achieved part of the desired function with the added benefit of being fully functional and tested by others. This allowed speedy implementation and were thus incorporated into the project.

Third party components included:
Bootstrap - achieve the homepage slider.
Google maps - embedded location map within the contacts page.
Chartkick - easily generate neat looking charts from models.
Rspec, Factorygirl, Capybara - improve efficiency and effectiveness of testing .
Mail_form gem - complete email post handling to deal with email enquiries.

3.2 Overall Design

To fulfill all functionalities mentioned in specification section, four models (see section 4.1) were created, and the overall design for the project was produced based on these models and the requirement document (see Group8 Project Requirements), which further made the implementation work simple and efficient.

In non-models features, a basic home page was created to advertise projects and present brief content about the whole project system. All visitors need to be able to freely switch each project image there. The information page was designed to present general information for Software Engineering Group Course with the further links of the site. The contact page was designed to display key contact details. In this page, a contact form was created that allows visitors to email enquires to course coordinators (currently set to email dummy address).

In models features, all previous completed projects need to be able to advertise to all site visitors, so a project portfolio page was designed, which would list all past projects with their essential details (image, title and summary), the filter and search features were also introduced to make users search or fetch the particular fields of past projects, then a further search results show page was also added in. Moreover, to make visitors get details of each past project, each essential component was required to be clickable to redirect to further information of relevant past project. Besides, it was added a completed projects management button at top right of this page that was used for administrator redirecting to models management page.

Since the detailed information show page for each past project was designed to state as much specific project contents as possible, it was presented to viewers the title and meta data (completed year, category and last updated date) at top-side, project image, whole summary and student members short profile, who undertook the projects, on the left side, and the industry client details (client name, banner image, short summary, and official site link) for the project on the right side. With this design, the detailed information for a past project can be fetched, and visitors can fully know the already completed projects with its working members and clients.

Furthermore, a stats feature (admin only) was also designed to show the statistics charts for all past project records: admin can view the distribution of all past projects by category, or the projects completed by year. The stats tool allow admins to gauge the amount of engagement or core data from the past projects in a glance, which is valuable as the system scales.

In addition to the design features above, the product still needs a modify mechanism to edit past projects, such as submitting a new past project record to the system, updating the past projects already exist in Database, and delete past project records. The features mentioned before are all reading behaviors, which fetch past project records data and represent them in some form later, the modify features are not involved, and there is no using limitation for users (all types of users can perform those features). The modify mechanism, however, is supposed to be executed for limited users only (administrators), and all past projects' modify need to be feasible using this mechanism, thus, a set of modifying methods was added into the system . After the admin logs in the system, the two modify link buttons are displayed at top-right side of an individual past project show page, admin is allowed to redirect to relevant edit or delete pages to execute modify work, and it is also allowed to create a new past project record in home page (which would show up a submit new button in home page after log in). Applied with this mechanism, the whole system of the project coordinates with each other: general visitors can only read records or use the permitted functionalities at front-end interfaces, while administrator is capable of adding, editing or removing records, and the previously modify work of admin would reflect on the reading behaviors of all viewers later.

During the last few sprints period, it was considered a more complex mechanism for admin's modify: the system should allow admin to maintain multiple past projects simultaneously, which improves the efficiency of managing past projects. Therefore, it was a management mechanism was created to help admin manage all past project records more efficiently and visually. By the addition of new parent-level model 'PastProjectManagement', admin is allowed to view and manage all past projects records into one integrated view page: admin can select the numbers of past projects and perform multiple edit and delete actions. The 'submit new' button stated before was also integrated to this page for adding new past project record.

For the users model' design, it was introduced from Group 7. The user model that Group 7 has implemented contain general visitors, industry and administrators (course coordinators) types, so the user type was validated in this group's project part, and show up relevant links after validation, and only allow administrators to perform manage features. All non-manage features are executable for all users.

As for other non-functional features for the project, including the navigation links to each specific view interface and site's page footer, as the whole industry engagement system, the cluster uses the same style design.

3.3 Other Design Choices

Initially, during the creation period of models, it was discussed the models that were necessary, and then it was figured out that three models (PastProject, Student and Client) in total were enough to demonstrate all features from project requirements. As the project developed, however, industry client information from past projects were hard to attain and was thus discarded. And instead each past project contains essential client information only.

The following list show other design choices that were initially envisioned but in the end were not feasible and did not make implementation:

- Allow visitors of the site to view student profile pages, which were to include a detailed information such as programming skills, academic levels, related working experience, and contact details. Initially, not knowing the full scope of the project and what resources were made available, it was envisioned that this project was to have access or be linked to Adelaide University's' database for student information. Allowing the administrator to simply add students to projects by id and with that all related information would follow through. However due to privacy and access this design feature was scraped.
- Again detailed industry client profile page was envisioned. The page was to include general contact details, achievements, client summary, evaluation and feedback of the program and more. However as the information attainable was limited and non feasible the design feature was abandoned and all information regarding industry client is strictly stored in the involved past projects table.

- Present to visitors greater details about each past project. It was considered that a specific past project should contain the timeline and feedback fields, and even derived implementation field after past project finished. All these fields content need to be presented in each specific past project show page. The timeline should be clickable and traceable to describe the extent of project finished: all users can click related project time point to fetch the rate of progress data with a relevant progress brief summary. The timeline is supposed to clearly show all time points as an integrated component to all viewers. The feedback field is supposed to provide testimonies or feedback about the completed project from industry clients. Due to the lack of more detailed information about past projects and time consuming for these fields, this design choice was abolished.

- Allow administrator to rearrange all past projects' showcase list. Before the implementation period for the project, it was supposed that admin could rearrange all past projects and reflect to projects portfolio page. While later it was found the rearrange action should never be allowed to perform. The Database itself allocates the primary order for each record, and it takes charge of maintaining records order that have been already stored before or just added in, so it is not supposed to reorder the records by primary key ('id' by default) anyway. The reasonable way that can achieve the similar action is to call 'order_by' method for all records, and it just changes the display order for records rather than overrides the original Database records order. However due to simplicity of the default arrangement, a decision was made to not implement this feature.

- Allow administrator to submit multiple new past projects records simultaneously. The creation behavior of past projects needs to submit multiple past projects at same time, which is similar with multiple edit and remove actions. While it does not influence the past projects records previously stored if multiple edit or delete action fails, the records keep unchanged, but the multiple create actions fail to influence the records that will be added. When admin submit a large number of new past project records simultaneously, all submit forms' fields are empty, if any part is filled wrong, then all other currently submitting records may disappear, which might force admin to re-enter large amount of contents. The situation like this is not supposed to occur for the whole system, so it was finally decided to allow admin to submit single past project record each time only, which is easier to handle the errors of submitting and is enough for admin to perform create action (submit the past project one by one if multiple records are required to insert).

- Demonstrate any new features for users. In project portfolio page and each specific show page, the two sides (right and left sides) are supposed to present more plentiful contents for the project. The contents of two sidebars can be the last submitted past project essentials or some helpful information that helps visitors to acquire more details about past projects. While this design was eventually discarded due to the lack of useful contents that can be filled at both sides.

4 Methodology

4.1 Github

4.1.1 Codebase

The usage of GitHub was crucial for the development of the project. GitHub is a source control management system that is simple to use yet very powerful to manage the code. GitHub provides all group members as well as the cluster members to have the same codebase to work on. This means that the same layout across all the websites can be used for all group members but different layout for individual group work to satisfy the individual group's requirements. Besides that, the authentication system that was developed by Group 7 can also be easily integrated into the project's system as well.

4.1.2 Documentation needs

In addition to offering the repository to manage the source code, GitHub also offers Wiki for documentation needs and Issues page to submit new issues for the cluster to collaborate on. Since Agile method with weekly cluster meetings was used as the software development strategy, GitHub Wiki allows the Agile method to be used to its maximum benefit. During each meeting, the Scrum Master would ask the members what they have accomplished in the previous week, any problems that they encountered and is the problem solved, and what will they be doing next. The Scrum Master will then write the goals that need to be achieved by next week's meeting in the GitHub Wiki page. In addition to that, the Scrum Master will check the issues page and ask the assignee if they have completed the task assigned to them. If the task is completed, the Scrum Master will close the issue. However, if the task is not yet completed, then the issue is marked with a new milestone, which is by the next cluster meeting.

4.1.3 Github Issues Page

In between weekly meetings, the group and cluster members are encouraged to submit any issues to the GitHub Issues page if they encounter any bugs or thought of any new enhancements, etc. Those issues will then be assigned to the member who is responsible for the particular feature and will be tagged as such. If the bug needs to be resolved by a specific milestone, a milestone can also be added to the issue as well. GitHub also has the ability to create branches for the cluster to use. Each branch that is branched off from the master is dedicated to work on a specific feature of the project. Most of the time, the branch is created and developed by the person who is assigned on the GitHub Issues page. However, other members of the group are encouraged to help with the implementation of the feature.

4.1.4 Resolving conflicts

Even though branches provide a way to simplify the work of the cluster, it also added a few difficulties as well particularly with conflicts. In order to solve conflicts, a high amount of communication is required to solve these conflicts by determining which version is the correct one. This is done during the cluster meeting where Group 7 will branch to the master branch first then it is Group 8's turn.

4.2 Requirements gathering

4.2.1 Schedules and Meetings

Requirements gathering for the project was done in the mid semester of Semester 1. At the time, Group 8 consisted of 4 members (Louis Griffith dropped out in Semester 2). A fortnightly meeting was set up to gather requirements for the project which took place in the Lady Symon meeting room. Each meeting usually took about 2 to 3 hours to conduct. Meetings were scheduled via each group members' Gmail student accounts. The possible users that were considered to be using the system were:

- Course coordinators
- General users
- Industry partners

Methods of gathering requirements were done in two ways which were:

- User Stories
- Conventional method of gathering requirements

4.2.2 User Stories

The User Stories method of gathering requirements was done on a piece of paper instead of a 3" x 5" card. Group members would then fill in blanks which are:

- As a [user]
- I want [the name of the feature]
- So that [goal to be achieved]

For example, if the user is a general viewer, a user story is created as such:

- As a **public user**
- I want **to search through the available projects**
- So that **I can quickly find the specific project that I am looking for**

This method is used for the other possible users as well.

4.2.3 Conventional method of gathering requirements

The conventional way of gathering requirements was also used which is coming up with the requirements in a brainstorming meeting.

4.2.4 Changes to the original requirements

There is a slight modification to the original requirements as the requirements changed along the course of the implementation phase as well extensive planning with Group 7 as a cluster.

4.3 Testing

Testing of the project was done using rspec rails, capybara and factorygirl gems. Rspec is a testing tool for ruby on rails that is easy to use and provides human readable test cases. Paired with capybara which allows extensive browser behaviour testing and factorygirl for automated generation of models, the combination enables thorough testing of all core aspects of the project.

The methodology of testing a new feature was to generate use cases of how the new feature would be used and simulating the action into each test case. The simulation of a test case involved three main steps, generating the correct state of the web app, next to perform the action and lastly check the result with the expected behaviour.

Example: Logout test case

1. Initial state: User is logged in

```
describe "login" do
let(:user) { FactoryGirl.create(:user) }
before { log_in user }
```

2. Action: click logout button

```
describe "followed by logout" do
before { click_link "Logout"}
```

3. Check: user is indeed logged out

```
it { should have_link('Login') }
```

5 Group work

5.1 Qun

5.1.1 Individual Past Project Page

Each specific past project page provides the most detailed contents about this past project. The related view layout was involved as well.

5.1.1.1 Individual Past Project Page Show

A specific past project show page demonstrates the title of project, deliverable year, particular category, last updated time of modifying, representative project publicity image, the project's summary in detail, the short profiles of group members who worked for the project, industry client banner or logo, client name, client essential summary, and the further site link for knowing greater details about a client. As the main parts of show page, project summary and group members can be expanded or collapsed as modules.

5.1.1.2 Individual Past Project Page Edit

The edit button at top-right side of show page can be clicked and then redirected to past project edit page. The edit project page is capable of editing all details of a past project record, including the create, remove, update of nested students attributes.

5.1.1.3 Individual Past Project Page Delete

The delete button at top-right side of show page can be clicked and then redirected to past project delete page. The delete page prompts administrator to perform remove action or not, the action will be executed if the confirm button applied.

5.1.2 Students CRUD Actions

It is allowed to edit any student record during editing its parent-level past project only. A student record can be added, updated, or deleted in group edit section of any kinds of past project edit page.

5.1.2.1 Students Create

When the add button of group edit section is applied, the view is redirected to student creation page, and it is allowed to add all student details fields there, and admin can choose a course name of student's attribute that already exists from selection lists or add a new course by typing manually. Once add submit is finished, a student create success page will be showed up, or a reminder information appears if any errors occur. The addition of a student record will be directly presented to group edit section as well.

5.1.2.2 Students Edit

When the edit button of group edit section is applied, the view is redirected to student edit page, and it is allowed to modify all student details fields there. The updating of a student record will be directly presented to group edit section as well. Otherwise, it is also allowed to modify student details from input fields of group edit section directly.

5.1.2.3 Students Remove

When the remove button of group edit section is applied, it will be prompted to continue or not, once confirmed, a student record will be deleted and the view will be redirected to the same edit page but with the new student records set (without deleted record).

5.1.3 Multiple Past Projects Management

The multiple past projects manage system is integrated to an isolated view page, in the manage page, admin can filter or search past projects and fetch the particular sets of projects in a table presented in that page. By selecting numbers of past projects table items, admin is capable of editing all chose projects one by one, editing simultaneously, or removing simultaneously.

5.1.3.1 Past Projects Edit Separately

The numbers of past projects can be edited in sequence in edit-one-by-one interface. Admin can choose any individual past project to edit, and move to next or previous past project record in the already selected lists to edit. Admin can stop the multiple records modify and back to manage system page any time, which lets admin freely handle the modification process.

5.1.3.2 Multiple Past Projects Edit Simultaneously

The numbers of past projects can be edited simultaneously, too. After ‘edit’ button is clicked, admin can modify and perform updating for all chose records. By selecting each tab modulo, it is allowed to expand or collapse the edit section, and choose other records’ edit section as well. Once modification is done, the multiple edit page pass all records parameters as nested attributes of PastProjectManagement model to multiple update method which further finishes the updating task.

5.1.3.3 Multiple Past Projects Delete

With the selections of records that admin has made, admin can apply remove button to delete the chose past projects simultaneously. After the action is confirmed by admin, the manage view page will be refreshed with the new records set (without the previous deleted records in the list now).

5.1.4 Past Projects Filter

In project portfolio page, it is provided a filter bar that is used to filter the projects by particular categories or production year. The options of category and year can be applied separately or combined to perform. When options are selected, by clicking filter button, the options regarded as parameters would be passed to the same page, and the page would be refreshed to display the particular sets of records that users have applied. If filter is applied without any option, then the page just shows all records to viewers. The filter feature is also used in past project management page, both of them have the same mechanism to make filtering happen.

5.1.5 Past Projects Search

In the portfolio page, the searching bar is provided for all users to search any field contents about past projects that they want. The searching function supports general searching about attributes fields of all past projects records, except for project summary field and any child-level student attributes. The users normally will not search any record by a mass of summary or any parts of summary, because users at least know parts of metadata (e.g., title, year, ID, category) that they are going to search rather than randomly search a summary part, which may not always happen in actual scenarios, and makes the searching method hard to handle and implement either. Therefore, the project summary field is not supported for searching. The student attributes searching by general keyword is not allowed either, because the searching for project nature attributes and child-level student attributes is involved different searching methods in controllers (all nature attributes searching is executed by PastProject controller itself and its helper method inside of model, while student attributes searching is transferred to Student controller and executed, and finally reversely fetch the parent-level search results), the students attributes cannot be tracked with the searching by general keyword (the general keyword searching may need to enhance). By selecting a particular field of search keyword, all visitors can search by relevant fields and fetch the results. In addition, the searching method supports fuzzy searching: any searching word that is more than 3 letters can be executed and return found results during searching, while students metadata attributes (id, name and email) are not supported with this scenario. Once the search button is applied, the interface either return the searching results list of past project records (with each one contains essentials information and further link to each result) or return not found page if no record can be fetched. The searching feature is also used in past project management page, both of them have the same mechanism to make searching results found.

5.2 Kamal

5.2.1 Submission of past projects page

Implementing the submission of past projects page which would include the use of dynamic forms as well as JQuery.

5.2.2 Project gallery page

Implementation of the project gallery page which would show the pictures of the projects in a grid-like view. The title of the project was also described in each of the project picture. Each picture is linked to the specific project page that would provide more details for viewers to see.

5.2.3 Project models and migrations

Created the models StudentProject and Student with the relevant attributes and validations. Throughout the development, more attributes and validations were needed to be implemented as well. Handling of migrations for the two models to avoid any errors with the schema.rb file. Removed the Client model as it was unnecessary for the purpose of the project due to the insufficient amount of information needed to make the model reliable.

5.3 Andrew

Responsible for the design and implementation of the home page image carousel, the email enquiries form, the statistical tool for administrators and rspec testing for group 8.

6. Final product

6.1 Final Produced Product

Finally, a web application was produced with the core of previously past projects showcase mechanism to industry partners and general visitors, general information presentation to users, general contact details illustration, and the extra management system for administrator users to maintain all past projects or their attaching students records. The product, of course, is produced based on Rails framework. The following lists introduce the macroscopical product that have been produced in front-end interface point of view:

- A project portfolio to showcase the completed projects undertaken by students to the general population, especially the industry partners, with the filter and search functionality inside
- A specific past project show page with greater details, which demonstrates to industry partners or other visitors about the completed project details content
- An integrated project management page which allow admin to handle and maintain all completed projects details simultaneously
- The separate manage pages that allow admin to modify an individual record easily
- A statistic page that presents the stats charts of past projects to help admin to know the rough submitting condition of completed projects
- A home page that advertises the projects image and other useful contents to all visitors
- A contact page that allows visitors to get the contact details or make general enquiries to faculty
- An information page that introduces the general information about SEGP courses

6.2 Implementation of work

As it was shown in the requirements document of Group 8, an implementation of an advertising platform was completed for industry partners or general visitors to gain wider exposure and awareness about the past projects that have been undertaken by a group of students of University of Adelaide. Besides that, a set of management functionalities where administrators or course coordinators can manage the past projects was also completely implemented. The functionalities are as follows:

- Course coordinators can edit the details of the existing projects one by one or modify multiple projects simultaneously
- Course coordinators can edit the students that partaken in their respective project
- Course coordinators can delete an existing project (i.e. remove the data from the table)
- Course coordinators can delete an existing student tied to the project (i.e. remove the data from the table)
- Course coordinators can add new projects through the submit page or through the past projects management system
- Course coordinators can add students to an existing past projects via a different add student page

A filter function was also implemented to accommodate all visitors to easily filter the project list according to the two of the projects attributes which are:

- Year
- Category

A search function was also implemented to accommodate all visitors to search for specific project quickly. Visitors can search for a project either generally or with a specific attribute in mind such as title or category.

6.3 Unmanaged Functionality

As it was already mentioned in Design section, there are some functionalities that have not been managed, the reasons why discarding these contents involve the lack of information in some content fields, the difficulty of gathering requirements, the original design gap, the divergence between group members, the work assigned to another group and the limited time working for the project. The detailed functionality that have not been implemented shows as below.

6.3.1 User Model

In the whole cluster management system, it is introduced Users model from group7 rather than creating a new model to manage all users type, all users' maintenance is supposed to be group 7's work. During the validation of records management, checking the appropriate users session type is provided or not, if it is not provided or an invalid user type, then the view interfaces are transferred to group 7's log in section, only the users signed in as ‘course coordinator’ are allowed to reach all the records management functions.

6.3.2 Past Project Timeline Attribute

Like the illustration in Design section, the timeline field for past project model has not been implemented. Only the limited resources about past project records were given, and it might be not possible for group to fetch more detailed contents, such as past project timeline attribute. Otherwise, the presentation way for timeline filed cannot be figured out during designing and later implementing time. It seems to be difficult to present a reasonable timeline filed in project showcase page, so finally this part of contents was not be implemented.

6.3.3 Project Feedback Attribute

The feedback field of a past project was not implemented as well. Due to the lack of feedback contents provided in past project records, it was thought that the feedback field was not supposed to implement.

6.3.4 Other Unconsidered Functionality

There are some functionality or features that were not considered during the designing period, such as providing a separate model to denote industry clients details, adding a sidebar at portfolio or show page to present more plentiful information about completed projects, adding a communication mechanism for visitors that can be used to communicate between past project undertaken members and industry clients or other users. It was thought these features might need to be implemented during the last few sprint periods, while the time seems to be not enough to add some functionality like those that have not been considered so far. So finally they are not implemented.

6.4 Difficulties Encountered And The Solution

Throughout the process of developing the software , problems were encountered even in the requirements gathering phase until maintenance phase. The difficulties and the solutions are listed in the following section.

6.4.1 JQuery

During the planning phase, it was expected to create mockups with JQuery embedded into some parts of the website. However, none of group members had any experience with JQuery. To overcome that, it had to learn how JQuery works in Ruby on Rails and how to code JQuery as well. Due to the fact that group members are beginners in JQuery language, JQuery is only applied to some of the pages in the website such as the submitting a new past project form.

6.4.2 Ruby on Rails

During the planning phase of the project, each group member was assigned to learn Ruby on Rails which was in the Semester 1 by sourcing out online tutorials such as Rails For Zombies. Even though the group had adequate study in Rails, it was still not enough in order to develop this project.

Along the development of the project, more advanced materials were needed to be mastered in order to achieve the desired website that were specified in the requirements document. The solution for this was to cooperate with the other group in the cluster who were more experienced. Tutorials available online as well as Stackoverflow was also very helpful in solving the problems encountered while developing. This ultimately also introduced difficulties in coding standards and style due to individuality and learning process.

6.4.3 Difficulties in gathering requirements

Difficulties also occurred during the requirements gathering phase as the topic assigned to the group as well as the cluster were unclear. The inability to know what the clients require of the type of product features also came as a difficulty as well. For example, what sort of details would an industry partner want to see in the project portfolio page. Furthermore, implementation of additional features would also be required to figure out so that the product would function more than just a portfolio page. The solution was to set up frequent meetings among the group members to gather those requirements so that the group has a clear vision of what needs to be done in order to achieve the project goals.

7 Product Evaluation

Group 8 product satisfied most of the requirements that were drawn up in Semester 1 so it is working to its expectations. Since the product satisfy all the requirements, it is determined that the product excels in doing what it is intended to be done.

However, the product would be much more easier to be developed if more gems were used instead of developing it from scratch. For example, the bootstrap gem offers a painless way to deal with the design of the web pages instead of coding with raw css files. By using the bootstrap gem, it would be easier to integrate both of the group's designs in the cluster so that a natural unified look is implemented in the cluster's design.

The product could be used for a system like the University of Adelaide if it were to undergo extreme validation from key stakeholders and rigorous testing. This would attract attention of industry partners to submit projects as well and certainly ease the course coordinators in using internal relations to source out projects.

8 Project Reflection

Upon reflection of the project, Github was a highlight as a tool that really worked well for the group and the cluster as a whole. Prior to using Github, Google drive was used as the central file storage and collaboration platform and was appropriate at the time due to solely producing documentation. As the implementation stage commenced and code was produced a transition was made to Github to support the codebase. Elements that the cluster utilised well were the branching and merging capabilities of Github to allow for smooth integration of work, shared wiki which tracked the clusters progress and steered the direction of the project and issues tracker that pushed for collaboration between members.

What didn’t work well was validation of the project requirements. Having no clients with the exception of Kevin that gave minimal feedback, the product functionality lacked input from practical entities and consequently were designed internally within the group.

Another big issue early on was Ruby on Rails because the language was foreign to all members of the group, this was overcome by each member learning the language in their own time and using online sources. This caused discrepancies between coding styles and standards due to each member sourcing out different tutorials. This is acknowledged as a norm in the collaborative nature of a software project however a suggested improvement to narrow the gap is to have all members source the same initial tutorial to build a similar foundation for the project. For example have each member run through (https://www.railstutorial.org/) by Michael Hartl and then sourcing more advanced knowledge individualy.

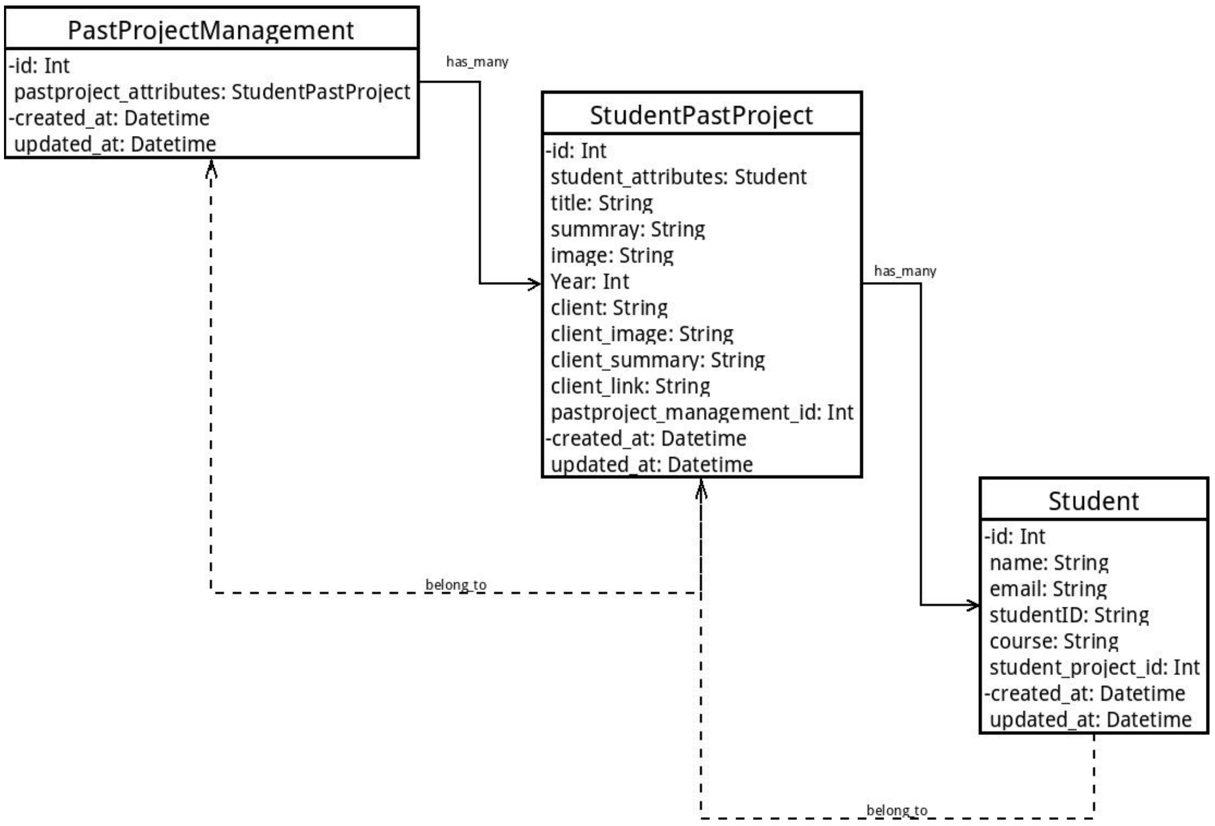
9 Appendix

9.1 Group Activities Logbook

WEEK	MEETING TIME	ALLOCATED WORK	WORK MEMBER	COMPLETED DATE
6	02/09/2014	Move all code from Server to GitHub	All members	08/09/2014
6	02/09/2014	Move all documentation and Collaboration from Google Docs to Github	All members	08/09/2014
6	02/09/2014	Move to a single code base	All members	08/09/2014
7	09/09/2014	gather past project info	Andrew	15/09/2014
7	09/09/2014	finalise portfolio overview	Kamal, Qun	13/09/2014
8	16/09/2014	Merge branches within group	Andrew	16/09/2014
8	16/09/2014	Overview	Kamal	06/10/2014
8	16/09/2014	Specific Project Selection	Andrew	06/10/2014
8	16/09/2014	Link Overview/Specific Project with Database	Qun	05/10/2014
9	07/10/2014	Compile Project Metadata/photos	Andrew	13/10/2014
9	07/10/2014	Adding location map to contacts tab	Andrew	13/10/2014
9	07/10/2014	Home Page Slider	Andrew	13/10/2014
9	07/10/2014	Design Past Project database (Good DB design - separate members/group model etc.)	Kamal	13/10/2014
9	07/10/2014	Display Individual Project View (Linkable from overview)	Qun	13/10/2014
9	07/10/2014	Group Merge	Kamal	14/10/2014
10	14/10/2014	Home page decoration and fix relevant bugs	Andrew	19/10/2014
10	14/10/2014	Add contact form	Andrew	19/10/2014
10	14/10/2014	Validate past projects submission	Kamal	18/10/2014
10	14/10/2014	Allow user to modify dynamic students data in create page(and make the view pretty)	Kamal	19/10/2014
10	14/10/2014	Fix bugs of edit individual project view page	Qun	17/10/2014
10	14/10/2014	Modify other CRUD behaviors of pastProjects	Qun	20/10/2014
10	14/10/2014	Add search bar/filter features to project model	Qun	21/10/2014
10	14/10/2014	Make testing suites and add more features	All members	28/10/2014
10	14/10/2014	Group Merge	Kamal	21/10/2014
11	21/10/2014	Project presentation prepare	All members	23/10/2014

11	21/10/2014	Stats feature implementation	Andrew	27/10/2014
11	21/10/2014	Multiple edit records feature implementation	Qun	27/10/2014
11	21/10/2014	Finalise past projects submission page	Kamal	27/10/2014
12	28/10/2014	App debugging	All members	03/11/2014
12	28/10/2014	Final report draft	All members	03/11/2014
12	28/10/2014	Home page modify	Andrew	01/11/2014
12	28/10/2014	Add new management model for multiple modify records	Qun	01/11/2014
13	04/11/2014	Finish final report	All members	06/11/2014
13	04/11/2014	Submit final products	All members	06/11/2014
13	04/11/2014	Project final check	All members	06/11/2014

9.2 Model Schema


PastProjectManagement
-id: Int
pastproject_attributes: StudentPastProject
-created_at: Datetime
updated_at: Datetime
has_many
StudentPastProject
-id: Int
student_attributes: Student
title: String
summray: String
image: String
Year: Int
client: String
client_image: String
client_summary: String
client_link: String
pastproject_management_id: Int
-created_at: Datetime
updated_at: Datetime
has_many
Student
-id: Int
name: String
email: String
studentID: String
course: String
student_project_id: Int
-created_at: Datetime
updated_at: Datetime
belong_to
belong_to