Aggielp

A website catering your appetite at TAMU

CSE606 Project

12/03/2013

Name & UIN	Percentage & Contributions
Tian Lan 922009685	20% Responsible for front-end design, integration of the team code and implementation of personal page, shop detail page and comment section of the index page
Yujia Liu 322006679	20% Responsible for the preliminary investigation and the implementation of rating and presenting top shops on the index page
Zhe Shen 322005893	20% Responsible for the testing of the whole project and for organizing sprint meetings, writing the backlogs and helping write the report
Yanming Zhou 223003032	20% Responsible for database design and the implementation of the search section
Longhua You 423000667	20% Responsible for the login, register and rating section of the website

Contents

1. Introduction

1.1 Topic

In order to serve TAMU students' appetites for cuisines, we decided to build a website named Aggielp. Basically, it is a website that provides functions such as cataloging, searching, viewing, commenting, rating, and saving favorites restaurant for users in the Bryan/College Station area. The agile development method was used throughout this project and we apply MVC design patterns for the construction of our project.

1.2 Programming language

PHP, HTML, CSS, JavaScript, SQL.

1.3 Programming environment

OS: Mac OS X & Windows

Server URL: http://198.58.119.124/aggielp

Database: MySQL

1.4 Methodology

Agile framework: Scrum (Sprint, Backlogs, PO, Team)

Design pattern: MVC (Codeigniter Framework)

Front end design framework: Bootstrap, jQuery.

1.5 Expectation

1.5.1 We expect to get familiar with Scrum, which is a popular agile method adopted in many development process. We also expect every team member to know the detail of our project by taking turns to act as a Product Owner.

1.5.2 We expect to master the use of MVC architecture model.

1.5.3 We expect every team member to learn and master the adept use of common programming languages and tools, such as PHP, SQL, HTML/CSS, etc.

2. Project detail

2.1. Project description

2.1.1 Perspective: Aggielp is a website aimed at providing users with restaurant information in Bryan/College Station area.

2.1.2 Functions: Basic functions a typical user of Aggielp would expect it to provide:

- Create an account and assign a password for themselves to save favorite restaurants, view history comments, etc. in Aggielp;
- Search the names for restaurants in Bryan/College Station area that are documented in Aggielp;
- View the basic information such as name, menu, etc. of any restaurant in Aggielp;
- Rate any restaurant documented in Aggielp;
- Post and view all his/her own comments for any restaurant in Aggielp;
- View the basic information of other users, such as account name, etc. in Aggielp.

2.1.1 User characteristics: Aggielp is mainly aimed at young consumers, especially university students at TAMU, in Bryan/College Station area.

2.1.2 User interfaces: Logical characteristics of user interfaces: We will deploy Aggielp on a remote server, and therefore it will have regular web interface for user interactions displaying:

- Logo and name of Aggielp;
- Username, password and basic information Textfields;
- Restaurant information;
- User information;
- Textfields for searching, viewing, and rating;
- Other basic information.

2.1.3 Hardware/Software interface: Aggielp aims at being able to be parsed and displayed on modern web browsers such as Internet Explorer (7.0 and higher), Mozilla Firefox (20.0 and higher), and Google Chrome (28.1 and higher). It does not require specific hardware to function normally.

2.2 Project implementation

2.2.1 Agile framework: Scrum

During our development, we adopted the Scrum as the agile method. Scrum is a Framework that enables iterative and incremental product development, allows us to get things done at the right time, maximizing the value of what is delivered.

The Scrum framework consists of three main roles: Product Owner (PO), Team and Scrum Manager (SM). The nature of a SM indicates that it is the person who is responsible for the Scrum process and ensures everybody plays by the rules. So we decided to obey the rules and overlook this position this time, because we need more people for development. As a result, for each sprint, four of us were selected to be in the Team and one person was selected as the Product Owner.

The reason for changing PO each sprint is that we expect every team member to get familiar with our product, which is the primary responsibility of a PO. As known, the Product Owner represents

the interests of everyone with a stake in the project (Stakeholder) and he is responsible for the final product. The PO is to elicit product requirements, manage the product backlog and manage the release plan. We met every Monday to do the sprint planning for the whole week and the PO of that week is to conduct the meeting, which is divided into two parts. First, the PO presents the User Stories. Second, when the Team members think they have enough Stories to start the Sprint, they begin breaking it down in Tasks to fill the Sprint Backlog (normally 3 to 4 days of work, than inspect & adapt).

During the development stages, we scheduled our planning adaptively to the progress of the project, and our solutions also evolved along with the requirements. The PO is responsible for the Product Backlog, which contains the requirements for the product. The Product Backlog is an always changing, dynamically prioritized list of requirements. The PO breaks down the requirements into User Stories. We released working solutions (here it meant the website) instead of well-written documents after our sprint every week, and our website responded to the changes in the requirement analysis too.

After the PO has committed the User Stories for each sprint. Our Team members began to break down all the committed User Stories into Tasks, which were kept in the Sprint Backlog. Then every single task was assigned to a Team member, who was responsible for keeping track of that task. More specifically, every task has three states: To-Do, Doing and Down. At any given time, all tasks should be stay in any of the three states. All items on the Sprint Backlog should be developed, tested, documented and integrated in order to fulfill the Sprint Goal.

Team members updated the Burndown chart every sprint, which showed the amount of work remaining per Sprint. It is a very useful way of visualizing the correlation between work remaining at any point in time and the progress of the Team. It inspects their progress in relation to the Planning by using the Burndown Chart, and makes adjustments as necessary.

At the end of a Sprint, the Team evaluates the finished Sprint. They capture positive ways as a best practice, identify challenges and develop strategies for improvements.

2.2.2 Design pattern: MVC

Model-View-Controller is used as our architecture pattern. Our model consists of basic logic and data, and we implemented different views for the presentation of different data in different situations. Our controllers ensured the interactions with the users were accurately captured and transferred to the model layer and the views changed accordingly.

In our project, we use CodeIgniter to build the dynamic web sites with PHP. CodeIgniter is an open source rapid development web application framework. "Its goal is to enable developers to develop projects much faster than writing code from scratch, by providing a rich set of libraries for commonly needed tasks, as well as a simple interface and logical structure to access these libraries." CodeIgniter is based on the Model-View-Controller development pattern. MVC separates application logic from presentation. It permits the web pages to contain minimal scripting since the presentation is separate from the PHP scripting.

The Model represents the data structures. We have four tables in our database: Users, Shops, Menu and Review. The model classes contain functions to retrieve, insert and update information in the

database. We have three model classes to interact with the database: User, Shop and Comment. User model includes functions related to user activities. It interacts with the User table in the database. Shop model includes functions of rating shop and getting shop details, which interact with the Shop table in the database. Comment model includes functions of viewing and adding comments. It involves all three tables in the database.

The View is the information that is being presented to a user. The View in our project are web pages. In CodeIgniter, a view can also be a page fragment like a header or footer. Other view pages are presented as login and register pages, shop details pages, search pages, personal pages etc.

The Controller serves as an intermediary between the Model, the View and any other resources needed to process the HTTP request and generate a web page. We have a main controller which is the entrance of the website. The "index" function is loaded by default.

Figure 1. Directory structure of CodeIgniter

2.2.3 Database design:

There are 4 tables in the database: Users, Shops, Menu and Review. Here is the structure of the database:

- Table "Users" has an id as Primary key for identification, and the "username" and email attribute are unique. Field "passwd" stores users' password value after MD5.
- Table "Shops" stores shops's information, including shop's id, name, address and the menu it used. The menu id is a foreign key, referencing to the key attribute "id" in Menu table.
- Table "Menu" is consisted of three field: id, dished and price. The dished with same id means that they are in the same menu, which can be seen in Table Shops.

- Table "Review" can show users'c comments and rates on a certain shop. There are two foreign key in the table: sid(id for shops) and uid(id for users). By calculating the average of rating can be showed on the shop detail page.

Below is the ER diagram for the database:

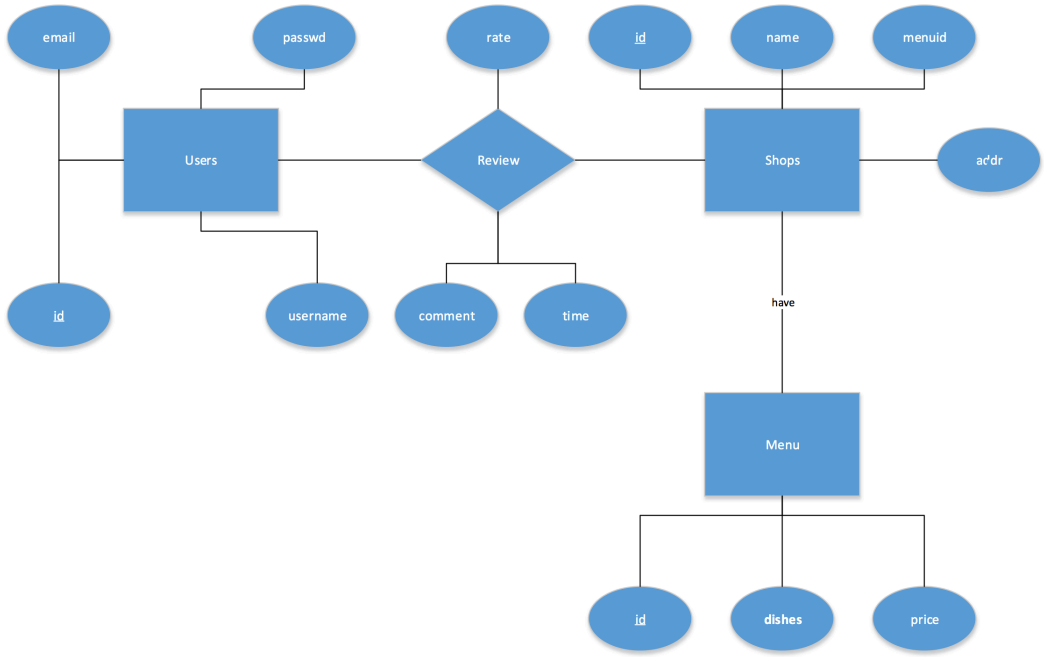


Figure 2. Database ER diagram

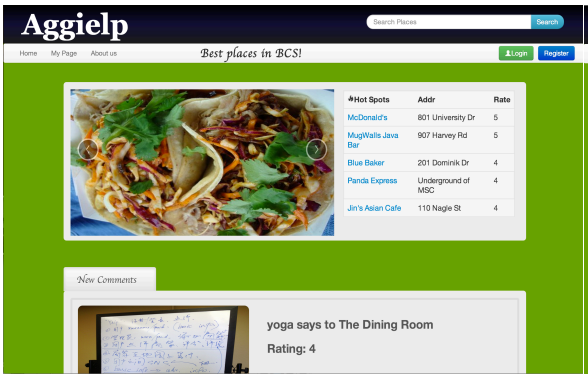
3. Conclusion

3.1 Discussion of results

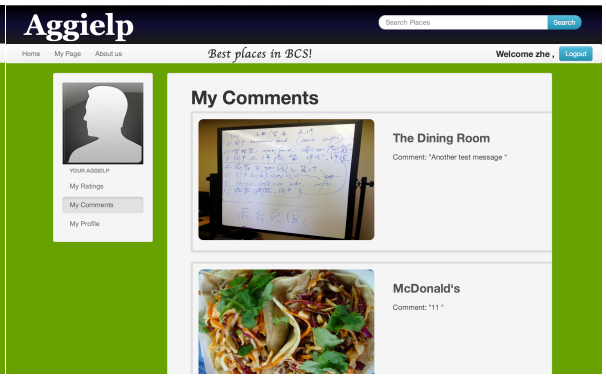
3.1.1 We succeeded in implementing the following features:

- Functions: A typical user of Aggielp is able to:
  - Create an account and assign a password for themselves to save favorite restaurants, view history comments and ratings, etc. in Aggielp;
  - Search the names for restaurants in Bryan/College Station area that are documented in Aggielp;
  - View the basic information such as name, menu, etc. of any restaurant in Aggielp;
  - Rate any restaurant documented in Aggielp;
  - Post and view all his/her own comments for any restaurant in Aggielp;
  - View the basic information of other users, such as account name, etc. in Aggielp.
- User interfaces: Aggielp has the following interface elements:
  - Logo and name of Aggielp;
  - Username, password and basic information Textfields;
  - Restaurant information;
  - User information;
  - Textfields for searching, viewing, and rating;
  - Other basic information.
- Hardware/Software interface: Aggielp can now smoothly run on modern web browsers such as Internet Explorer (7.0 and higher), Mozilla Firefox (20.0 and higher), and Google Chrome (28.1 and higher). It does not require specific hardware to function normally.

3.1.2 Screenshot examples:



index page



personal page

3.3 Key learnings

3.2.1 Scrum:

With the guidance of Scrum, we completed our website project successfully and meanwhile we learned about the following key notions of this agile framework:

Firstly, a Scrum project mostly starts with a vision of the product or system to be developed. The vision might be vague at first but It will become clearer as the project moves forward. Out of this Vision the Product Owner is writing the Product Backlog. So the PO is truly an important role in the development process.

Secondly, we understand better the cooperation between the Team and the PO as we went through all the sprints in about 2 month. At the start of iteration, the Product Owner is presenting the prioritized Product Backlog to the team, which selects what it believes it can turn into an increment of potentially shippable functionality by the end of the Sprint. Doing so, the Sprint Backlog is created. After, the team is left alone to develop the features they have chosen. The team now takes a deeper look at the requirements, considers the available technology, and evaluates its own skills and capabilities. It then collectively determines how to build the functionality, modifying its approach daily as it encounters new complexities, difficulties, and surprises. The team figures out what needs to be done and selects the best way to do it. This creative process is the core of the Scrum productivity. At the end of the Sprint, the team presents the increment of functionality to the Product Owner so he can on one side inspect the functionality and on the other side make timely adaptations to the project.

Lastly, self-organizing Teams perform tasks faster and with higher quality. High levels of self-motivation are achieved and are the reason why Scrum allows teams to reach higher productivity faster.

3.2.2 MVC:

The model-view-controller (MVC) pattern is an architectural pattern used primarily in creating Graphic User Interfaces (GUIs). Model View Controller has been widely adopted as an architecture for World Wide Web applications in all major programming languages. Several commercial and noncommercial application frameworks have been created that enforce the pattern. As in our project, we use the framework of CodeIgniter to enforce MVC pattern.

The major premise of the pattern is based on modularity and it is to separate three different aspects: the data (Model), the visual representation of the data (View), and the interface between the view and the model (Controller). The primary idea behind keeping these three components separate is so that each one is as independent of the others as possible, and changes made to one will not affect changes made to the others. In this way, for instance, the GUI can be updated with a new look or visual style without having to change the data model or the controller.

MVC pattern allows people to modify the code without changing much else. Different developers have different strengths and weaknesses, so team building around MVC is easier. A View Team that is responsible for great views, a Model Team that knows a lot about data, and a Controller Team that see the big picture of application flow, handing requests, working with the model, and selecting the most appropriate next view for that client. One of the great advantages of the Model-View-Controller Pattern is the ability to reuse the application's logic (which is implemented in the model) when implementing a different view.

Appendix

Project Website:

http://198.58.119.124/aggielp

Scrum backlog examples:

Backlog1:Everyone can view the homepage

Everyone can view the homepage

User story

As a user

I want to view the homepage

So that I can decide what I do next

Acceptance Criteria:

1. **User is not logged in**

   This user could be any person
   They are able to view the homepage

2. **User is logged in**

   This user could be a registered user
   They are able to view the homepage

Validations:

User behavior rules

1. **User is not logged in**
   This kind of user cannot enter the person page
   This kind of user cannot rate or comment shops
   This kind of user cannot join the campaign
   1) Visitor
      Visitor need to register and login, otherwise he/she have no action expect look
   2) Registered user
      Need to login, otherwise he/she have no action expect browsing

Wireframes

Not the final version

Please note these are representative wireframes and not the final design

Backlog2:Logout

User Log Out

User story

As a user

I want to log out

So that I can leave the website or log in with another username

Acceptance Criteria

1. **User is not logged in**

   The not-logged-in-user is not able to see the log out button.

2. **User is already logged in**

   The already-logged-user is able to see the log out button and able to log out and re-login.

Validations

1. **User-logged-in validation rules**

   See whether the user is in a session, if exists, he/she is logged in. If not, system should direct the user to the login page.

Wireframes

Not the final version

Please note these are representative wireframes and not the final design