Yelp – final project of INFO 6250 Web-Tool

1. Short summary of the project

A simple version of Yelp application. Project contains two parts User and Restaurant.

2. Summary of the Functionality Performance

MainController: Home page and log out.
LoginController : Log in with email and password.
SignUpController: Create a new account.
FindFriendsController: Search user by keyword of name/ Follow another user/ Stop following other user.
UserDetailController: Return user information including follower and people who follows the user, review and create pagination of searching result.
UserProfileController: Update first name, last name, email address, password of users.
RestaurantController: Return all restaurant information.
SearchRestaurantController: Search restaurants by the keyword of their name.
WriteReviewController: Add comments and scores of searched restaurants.

3. Technologies Used

Annotated Spring framework+ Spring Validator + Interceptor + Annotated Hibernate ROM + MySQL

4. User Roles and performed tasks for each role

User:

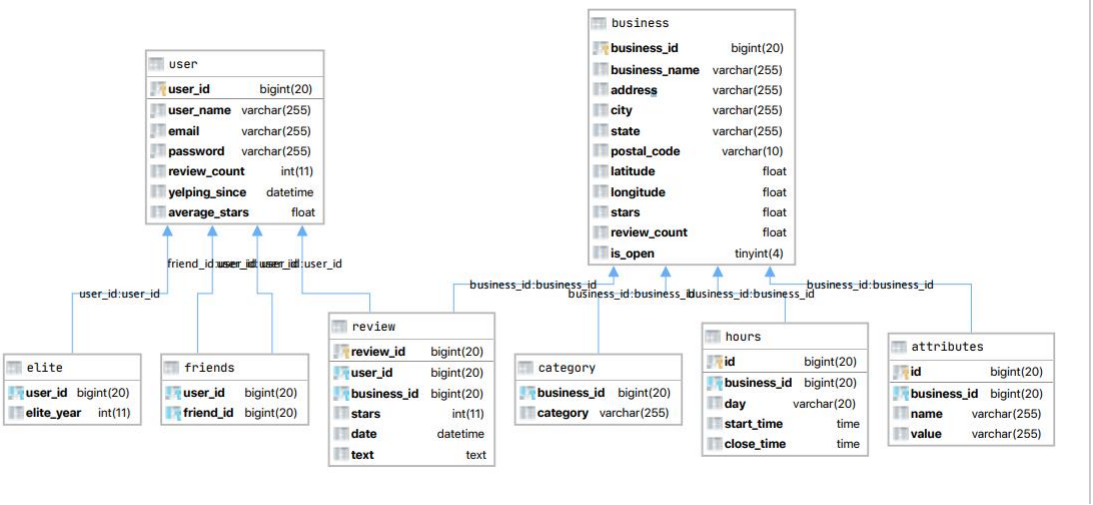
1. Users can create an account by *SignUp/SignIn.* An account must contain a first name, a last name, an **Unique** email address and a password.
2. Users can login in with their **Unique** email and password.
3. Users can check their profile information. Including historical reviews, rating distribution follower, people who follow you, join date, elite year.
4. Users can update some of their profile information. Including first name, last name, email address, password.
5. Users can search friends by keywords.
6. Users can follow/unfollowed the people they searched.
7. Users can search restaurants with their keywords.
8. Users can write a review for any restaurants if they have already got an account.

Restaurant:

1. Each restaurant has a name, an address, a list of category, a list of key-value pair attribute, reviews from users.
2. Each restaurant can be valued by users with attached comment.

3. Reviews of restaurants are visible to every user.

5. Data Model and Relationship



User:

private int id;
private String name;
private String email;
private String password;
private Integer reviewCount;
private Date dateJoined;
private Set<Integer> followList;
private Set<Integer> followerList;
private Set<Integer> elite;
private Double averageStars;
private List<Review> reviews;

Restaurant:

private int id;
private String name;
private String address;
private String city;
private String state;
private String postalCode;
private Double latitude;
private Double longitude;
private Double stars;
private Integer reviewCount;
private Short isOpen;
private List<Attribute> attributes;

private List<String> categories;
private List<Hours> hours;
private List<Review> reviews;

Review:

private int reviewId;

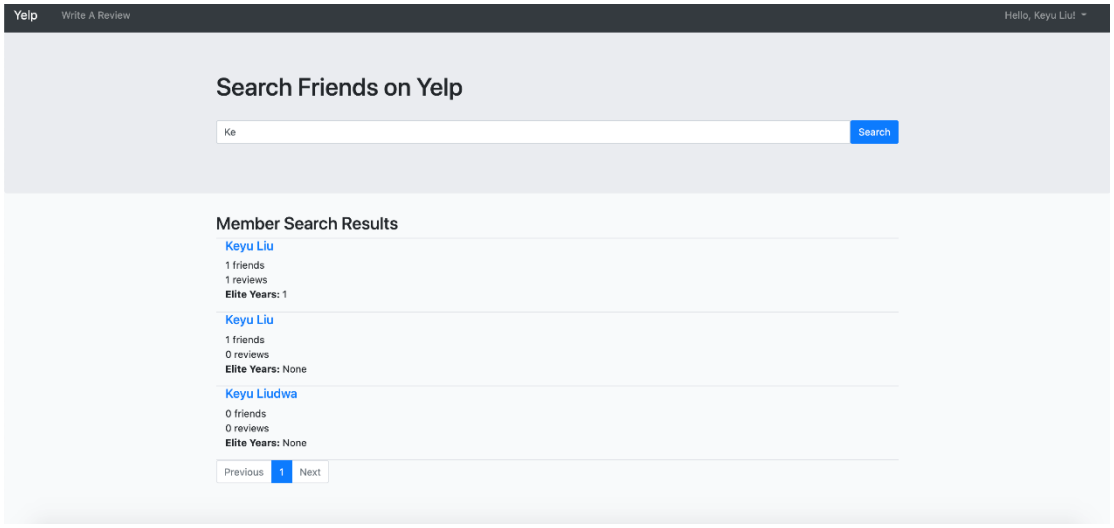
private User user;
private Restaurant restaurant;

private Integer stars;
private Date date;
private String text;

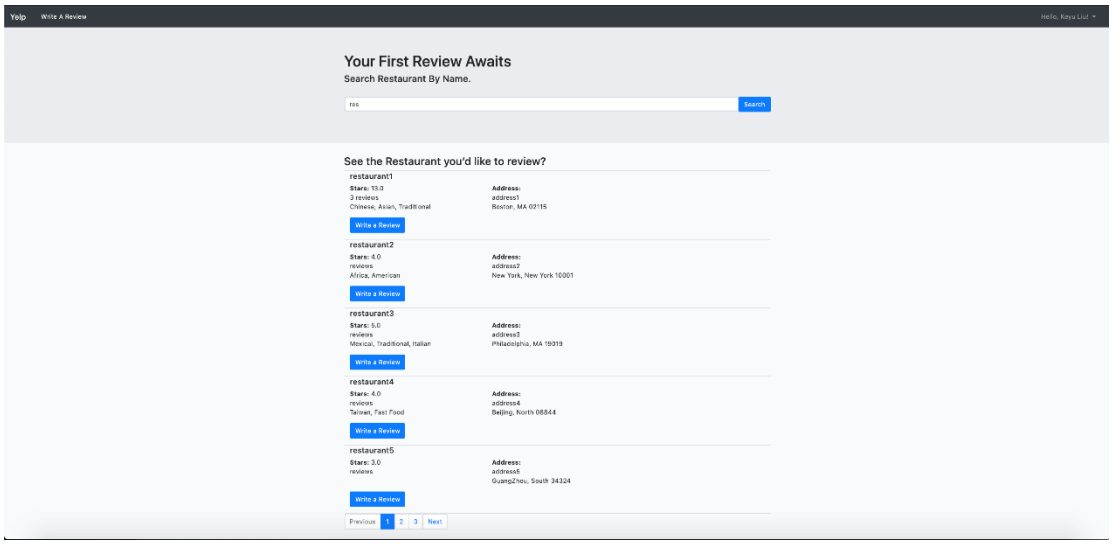
Attribute:

private int id;
private Restaurant restaurant;
private String name;
private String value;

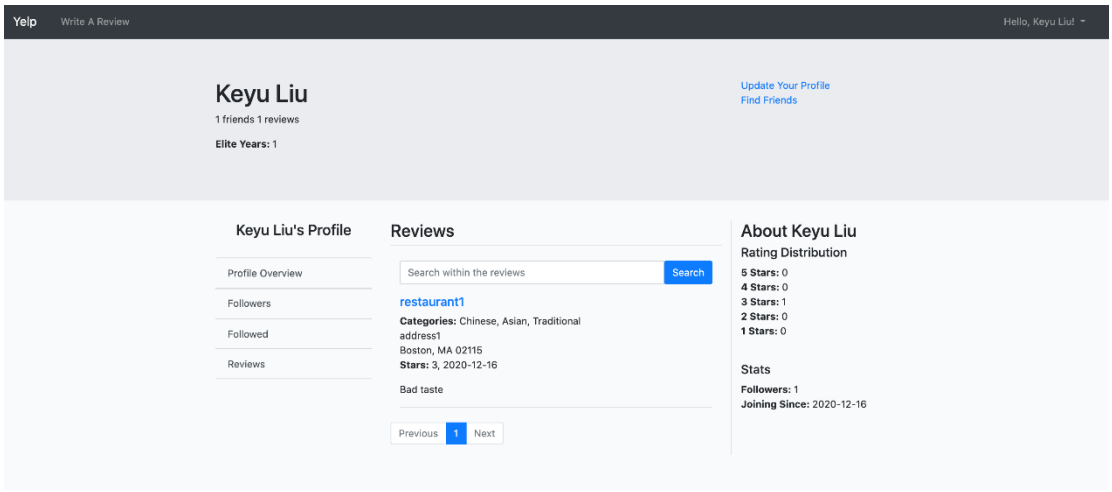
7. Screenshot of Main function



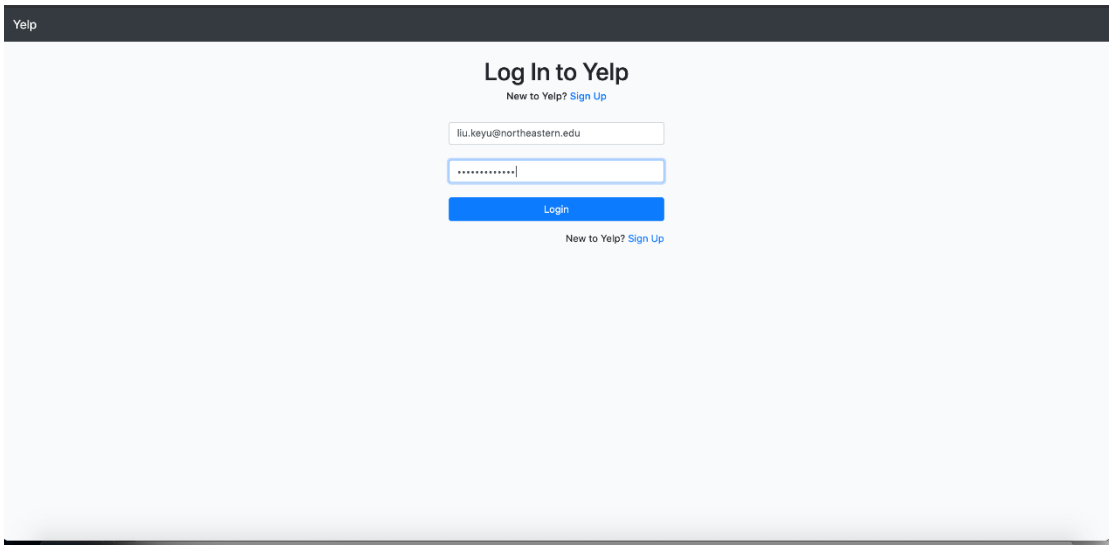
图表 1 Search Friends



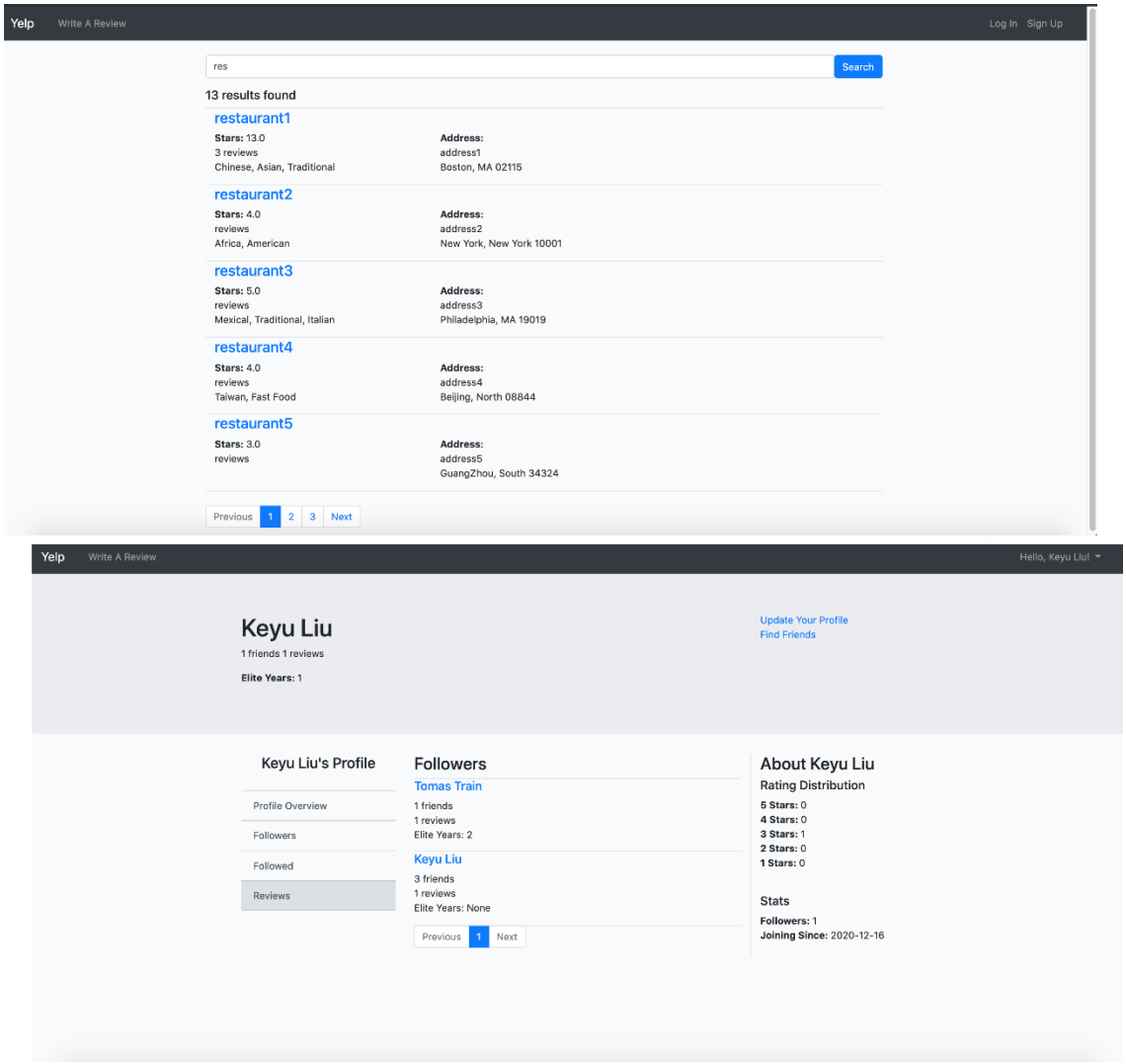
图表 2 Search Restaurants



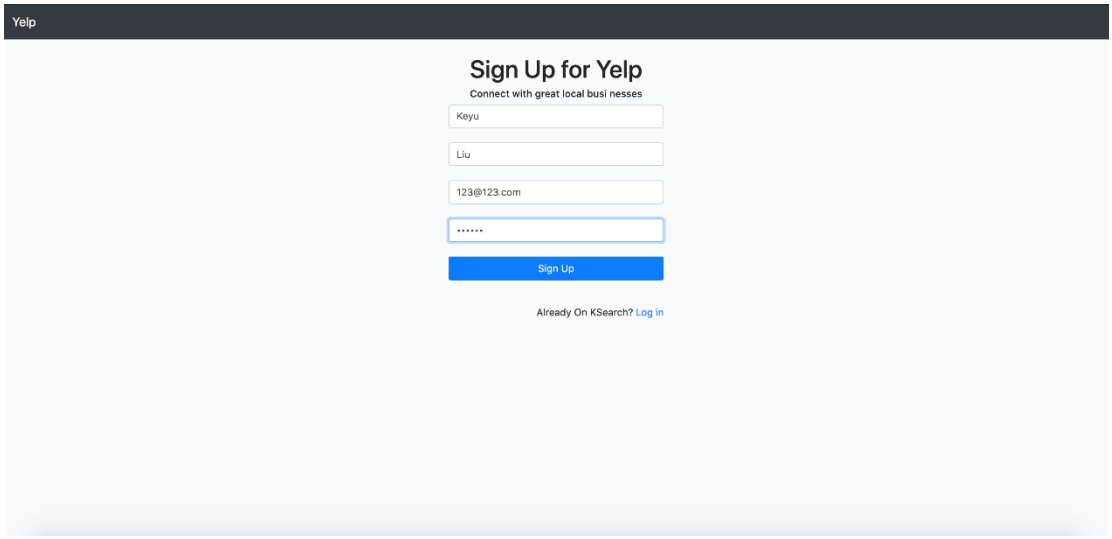
图表 3 User Profile



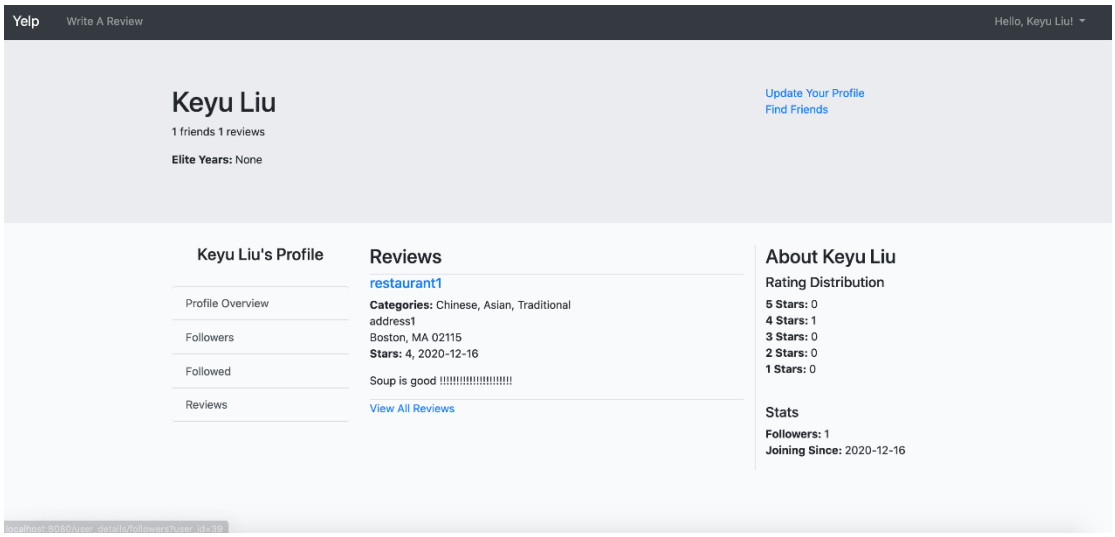
图表 4 Log in



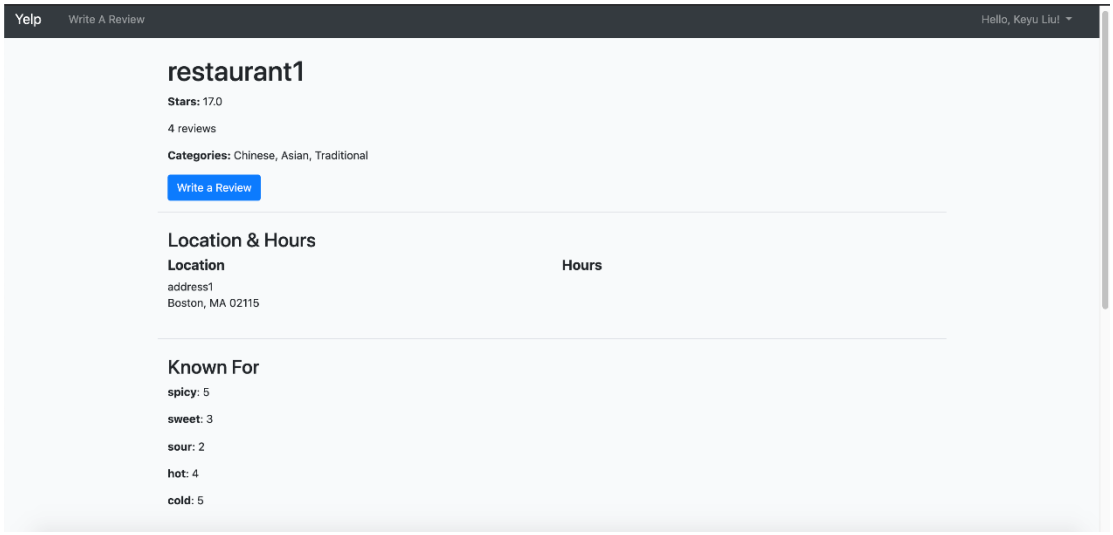
图表 5 User Follower



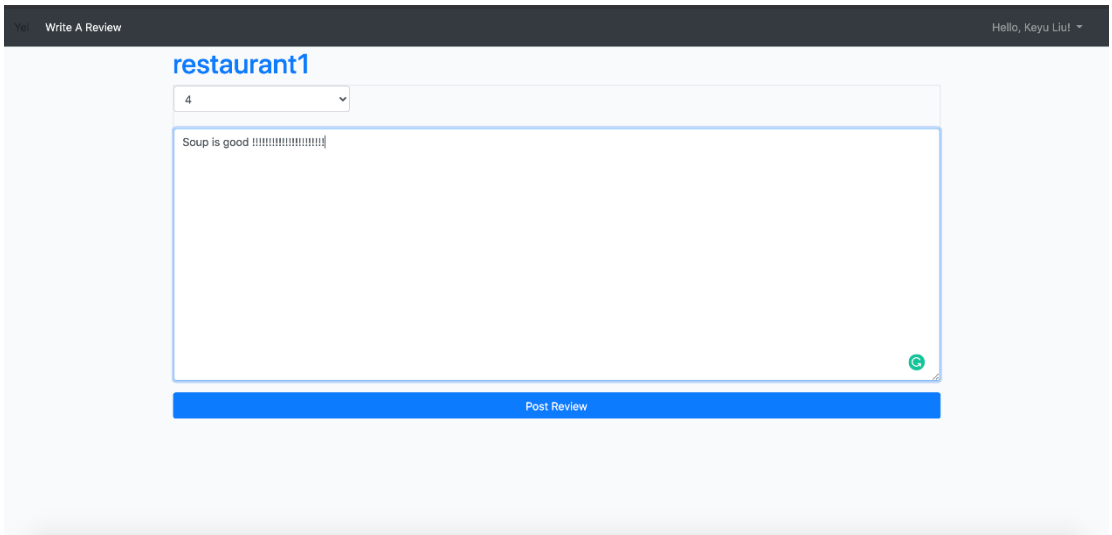
图表 6 User Sign Up



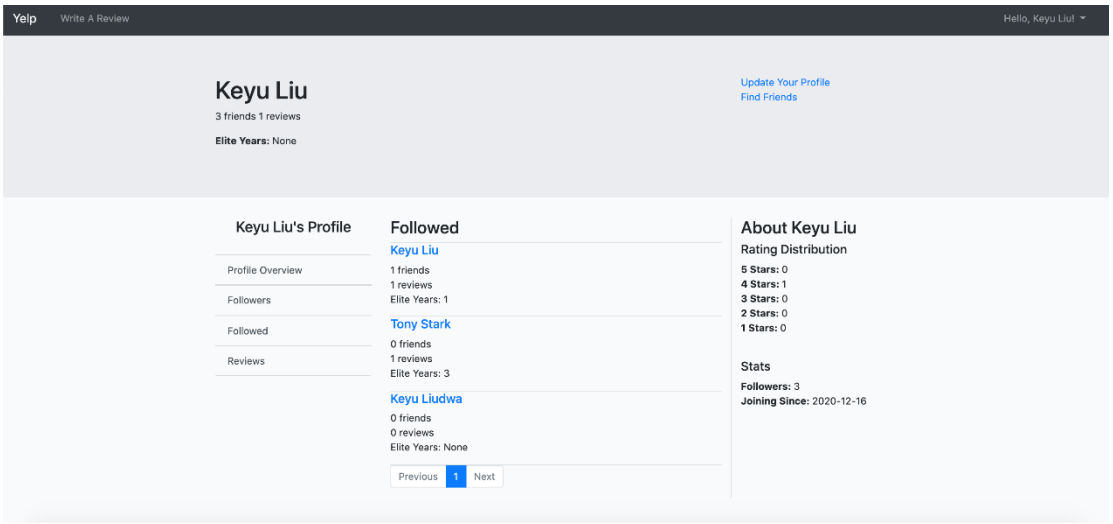
图表 7 User Reviews



图表 8 Restaurant Details



图表 9 User write review for Restaurant



图表 10 User Followed

8. CONTROLLER Source Codes

MainController.java

```
package
Controller;

                import javax.servlet.http.*;

                import org.springframework.stereotype.*;
                import org.springframework.web.bind.annotation.*;

                @Controller
                public class MainController {

                    @RequestMapping(path = "/main", method = RequestMethod.GET)
                    public String mainHandler(HttpServletRequest request) {
                        HttpSession session = request.getSession();
                        if (session.getAttribute("loggedInUser") == null) {
                            session.setAttribute("loggedInUser", null);
                        }
                        return "main";
                    }

                    @RequestMapping(path = "/logout", method = RequestMethod.GET)
                    public String dealWithLogout(HttpServletRequest request) {
                        HttpSession session = request.getSession();
                        session.setAttribute("loggedInUser", null);
                        return "main";
```

```
    }

}
```

LoginController.java

```
package
Controller;

import DAO.*;
import Model.Data.*;
import Model.Form.*;
import javax.servlet.http.*;
import org.springframework.stereotype.*;
import org.springframework.ui.*;
import org.springframework.validation.*;
import org.springframework.validation.annotation.*;
import org.springframework.web.bind.annotation.*;

@Controller
public class LoginController {

    @RequestMapping(path = "/login", method = RequestMethod.GET)
    public String getUserLogin(ModelMap model, UserLogin user) {
        model.addAttribute("userLogin", user);
        return "login";
    }

    @RequestMapping(path = "/login", method = RequestMethod.POST)
    public String dealWithLogin(@Validated
@ModelAttribute("userLogin") UserLogin userLogin, BindingResult
result, ModelMap model, HttpServletRequest request) {
        if (result.hasErrors()) {
            return "login";
        }

        HttpSession session = request.getSession();
        User loggedInUser = (User)
session.getAttribute("loggedInUser");
        if (loggedInUser != null) {
            model.addAttribute("failure", "Some user has logged in!
Please try again!");
            return "login";
        }
```

```
        UserDAO userDAO = new UserDAO();
        User user = userDAO.getUserByEmail(userLogin.getEmail());
        if (user == null) {
            model.addAttribute("failure", "User Not Found! Please
try again!");
            return "login";
        } else if
(!user.getPassword().equals(userLogin.getPassword())) {
            model.addAttribute("failure", "Password incorrect!
Please try again!");
            return "login";
        }

        session.setAttribute("loggedInUser", user);

        return "redirect:/main";
    }
}
```

SignUpController.java

```
package
Controller;

import DAO.*;
import Model.Data.*;
import Model.Form.*;
import org.springframework.stereotype.*;
import org.springframework.ui.*;
import org.springframework.validation.*;
import org.springframework.validation.annotation.*;
import org.springframework.web.bind.annotation.*;

@Controller
public class SignUpController {

    @RequestMapping(path = "/signup", method = RequestMethod.GET)
    public String getUserSignUp(ModelMap model, UserSignUp user) {
        model.addAttribute("userSignUp", user);
        return "signup";
    }

    @RequestMapping(path = "/signup", method = RequestMethod.POST)
```

```
    public String dealWithSignUp(@Validated
@ModelAttribute("userSignUp") UserSignUp userSignUp, BindingResult
bindingResult, ModelMap model) {
        if (bindingResult.hasErrors()) {
            return "signup";
        }

        UserDAO userDAO = new UserDAO();
        boolean existed =
userDAO.findExistedEmail(userSignUp.getEmail());
        System.out.println("Email existed? " + existed);
        if (existed) {
            model.addAttribute("failure", "Your Email has existed!
Please try again!");
            return "signup";
        }

        User user = new User(userSignUp);
        boolean result = userDAO.addUser(user);
        System.out.println("Add User Result: " + result);
        if (!result) {
            model.addAttribute("failure", "Sign Up Failed! Please
try again!");
            return "signup";
        }

        return "redirect:/login";
    }
}
```

UserDetailsController.java

```
package
Controller;

import java.util.*;
import javax.servlet.http.*;

import DAO.*;
import Model.Data.*;
import org.springframework.stereotype.*;
import org.springframework.ui.*;
import org.springframework.web.bind.annotation.*;
```

```
@Controller
@RequestMapping("/user_details")
public class UserDetailsController {

    @RequestMapping(path = "", method = RequestMethod.GET)
    public String getUserDetails(@RequestParam("user_id") int
userId,  HttpServletRequest request, ModelMap model) {
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("loggedInUser");
        if (user == null || user.getId() != userId) {
            UserDAO userDAO = new UserDAO();
            user = userDAO.getUserById(userId);
        }

        ReviewDAO reviewDAO = new ReviewDAO();
        List<Review> reviews =
reviewDAO.getReviewsByUserLimited(userId, 3);
        List<Integer> ratingDistribution =
user.countUserRatingDistribution();

        model.addAttribute("user", user);
        model.addAttribute("showReviews", reviews);
        model.addAttribute("ratingDistribution",
ratingDistribution);

        return "userdetails";
    }

    @RequestMapping(path = "/followers", method =
RequestMethod.GET)
    public String getUserFollowers(@RequestParam("user_id") int
userId, HttpServletRequest request, ModelMap model) {
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("loggedInUser");
        UserDAO userDAO = new UserDAO();
        if (user == null || user.getId() != userId) {
            user = userDAO.getUserById(userId);
        }

        int resultCount = user.getFollowerList().size();
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
```

```
        }

        List<User> followerList =
userDAO.getFollowers(user.getId(), 1, eachPageCount);

        model.addAttribute("user", user);
        model.addAttribute("followerList", followerList);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", 1);
        model.addAttribute("ratingDistribution",
user.countUserRatingDistribution());

        return "userdetails";
    }

    @RequestMapping(path = "/followers_pagination", method =
RequestMethod.GET)
    public String getUserFollowers(@RequestParam("user_id") int
userId, @RequestParam("page") int page,
                                   HttpServletRequest request,
ModelMap model) {
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("loggedInUser");
        UserDAO userDAO = new UserDAO();
        if (user == null || user.getId() != userId) {
            user = userDAO.getUserById(userId);
        }

        int resultCount = user.getFollowerList().size();
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }

        List<User> followerList =
userDAO.getFollowers(user.getId(), page, eachPageCount);

        model.addAttribute("user", user);
        model.addAttribute("followerList", followerList);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", page);
```

```
        model.addAttribute("ratingDistribution",
user.countUserRatingDistribution());

        return "userdetails";
    }

    @RequestMapping(path = "/follows", method = RequestMethod.GET)
    public String getUserFollows(@RequestParam("user_id") int
userId, HttpServletRequest request, ModelMap model) {
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("loggedInUser");
        UserDAO userDAO = new UserDAO();
        if (user == null || user.getId() != userId) {
            user = userDAO.getUserById(userId);
        }

        int resultCount = user.getFollowList().size();
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }

        List<User> followList = userDAO.getFollows(user.getId(), 1,
eachPageCount);

        model.addAttribute("user", user);
        model.addAttribute("followList", followList);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", 1);
        model.addAttribute("ratingDistribution",
user.countUserRatingDistribution());

        return "userdetails";
    }

    @RequestMapping(path = "/follows_pagination", method =
RequestMethod.GET)
    public String getUserFollows(@RequestParam("user_id") int
userId, @RequestParam("page") int page,
                                 HttpServletRequest request, ModelMap
model) {
        HttpSession session = request.getSession();
```

```
        User user = (User) session.getAttribute("loggedInUser");
        UserDAO userDAO = new UserDAO();
        if (user == null || user.getId() != userId) {
            user = userDAO.getUserById(userId);
        }

        int resultCount = user.getFollowList().size();
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }

        List<User> followList = userDAO.getFollows(user.getId(),
page, eachPageCount);

        model.addAttribute("user", user);
        model.addAttribute("followList", followList);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", page);
        model.addAttribute("ratingDistribution",
user.countUserRatingDistribution());

        return "userdetails";
    }

    @RequestMapping(path = "/reviews", method = {RequestMethod.GET,
RequestMethod.POST})
    public String getUserReviews(@RequestParam("user_id") int
userId, HttpServletRequest request, ModelMap model) {
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("loggedInUser");
        UserDAO userDAO = new UserDAO();
        if (user == null || user.getId() != userId) {
            user = userDAO.getUserById(userId);
        }

        ReviewDAO reviewDAO = new ReviewDAO();
        String keyword = request.getParameter("keyword");
        int resultCount;
        if (keyword == null || keyword.equals("")) {
            resultCount = user.getReviews().size();
        } else {
```

```
            resultCount = reviewDAO.getReviewCountByUserId(userId,
keyword);
        }
        int eachPageCount = 3;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Review> reviewList =
reviewDAO.getReviewsByUserId(userId, keyword, 1, eachPageCount);


        model.addAttribute("user", user);
        model.addAttribute("reviewList", reviewList);
        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", 1);
        model.addAttribute("ratingDistribution",
user.countUserRatingDistribution());

        return "userdetails";
    }

    @RequestMapping(path = "/reviews_search", method =
RequestMethod.GET)
    public String getUserReviews(@RequestParam("user_id") int
userId, @RequestParam("keyword") String keyword,
                                 @RequestParam("page") int page,
HttpServletRequest request, ModelMap model) {
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("loggedInUser");
        UserDAO userDAO = new UserDAO();
        if (user == null || user.getId() != userId) {
            user = userDAO.getUserById(userId);
        }

        ReviewDAO reviewDAO = new ReviewDAO();
        int resultCount;
        if (keyword == null || keyword.equals("")) {
            resultCount = user.getReviews().size();
        } else {
            resultCount = reviewDAO.getReviewCountByUserId(userId,
keyword);
```

```
        }
        int eachPageCount = 3;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Review> reviewList =
reviewDAO.getReviewsByUserId(userId, keyword, page,
eachPageCount);

        model.addAttribute("user", user);
        model.addAttribute("reviewList", reviewList);
        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", page);
        model.addAttribute("ratingDistribution",
user.countUserRatingDistribution());

        return "userdetails";
    }


}
```

UserProfileController.java

```
package
Controller;

import java.util.*;
import javax.servlet.http.*;

import DAO.*;
import Model.Data.*;
import Model.Form.*;
import org.springframework.stereotype.*;
import org.springframework.ui.*;
import org.springframework.validation.*;
import org.springframework.validation.annotation.*;
import org.springframework.web.bind.annotation.*;

@Controller
public class UserProfileController {
```

```
    @RequestMapping(path = "/profile", method = RequestMethod.GET)
    public String modifyProfile(ModelMap model, UserSignUp user) {
        model.addAttribute("userSignUp", user);
        return "profile";
    }

    @RequestMapping(path = "/profile", method = RequestMethod.POST)
    public String dealWithSignUp(@Validated
@ModelAttribute("userSignUp") UserSignUp userSignUp, BindingResult
bindingResult, HttpServletRequest request, ModelMap model) {
        if (bindingResult.hasErrors()) {
            return "profile";
        }

        HttpSession session = request.getSession();
        User loggedInUser = (User)
session.getAttribute("loggedInUser");

        UserDAO userDAO = new UserDAO();
        if (!loggedInUser.getEmail().equals(userSignUp.getEmail()))
{
            boolean existed =
userDAO.findExistedEmail(userSignUp.getEmail());
            System.out.println("Email existed? " + existed);
            if (existed) {
                model.addAttribute("failure", "Your Email has
existed! Please try again!");
                return "profile";
            }
        }

        loggedInUser.setName(userSignUp.getFirstName() + " " +
userSignUp.getLastName());
        loggedInUser.setPassword(userSignUp.getPassword());
        loggedInUser.setEmail(userSignUp.getEmail());

        boolean result = userDAO.updateUser(loggedInUser);
        System.out.println("Modify User Result: " + result);
        if (!result) {
            model.addAttribute("failure", "Modify Profile Failed!
Please try again!");
            return "profile";
        }
```

```
            session.setAttribute("loggedInUser", loggedInUser);

            return "redirect:/user_details?user_id=" +
        loggedInUser.getId();
        }
    }
```

FindFriendsController.java

```
package
Controller;

        import java.util.*;
        import javax.servlet.http.*;

        import DAO.*;
        import Model.Data.*;
        import org.springframework.stereotype.*;
        import org.springframework.ui.*;
        import org.springframework.web.bind.annotation.*;

        @Controller
        public class FindFriendsController {

            @RequestMapping(path = "/search_friends", method =
        RequestMethod.GET)
            public String findFriends() {
                return "findfriends";
            }

            @RequestMapping(path = "/find_friends", method =
        RequestMethod.POST)
            public String searchUser(HttpServletRequest request, ModelMap
        model) {
                UserDAO userDAO = new UserDAO();
                String keyword = request.getParameter("keyword");
                if (keyword == null) {
                    keyword = "";
                }
                int resultCount = userDAO.getUserCount(keyword);
                int eachPageCount = 5;
                int pageCount = resultCount / eachPageCount;
                if (resultCount % eachPageCount != 0) {
                    pageCount++;
```

```
        }
        List<User> resultList = userDAO.getUsers(keyword, 1,
eachPageCount);

        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", 1);
        model.addAttribute("searchResult", resultList);

        return "findfriends";
    }

    @RequestMapping(path = "/find_friends", method =
RequestMethod.GET)
    public String searchUser(@RequestParam("keyword") String
keyword, @RequestParam("page") int page, ModelMap model) {
        UserDAO userDAO = new UserDAO();

        int resultCount = userDAO.getUserCount(keyword);
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<User> resultList = userDAO.getUsers(keyword, page,
eachPageCount);

        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", page);
        model.addAttribute("searchResult", resultList);

        return "findfriends";
    }

    @RequestMapping(path = "/follow", method = RequestMethod.GET)
    public String followUser(@RequestParam("user_id") int userId,
HttpServletRequest request, ModelMap model) {
        UserDAO userDAO = new UserDAO();
        HttpSession session = request.getSession();
        User loggedInUser = (User)
session.getAttribute("loggedInUser");
```

```
        if (loggedInUser == null) {
            model.addAttribute("login", "Please login to follow
user!");
            return "userdetails";
        }

        boolean result = userDAO.followUser(loggedInUser.getId(),
userId);
        if (!result) {
            model.addAttribute("followFailure", "Follow Failed!
Please try again!");
            return "userdetails";
        }

        loggedInUser.getFollowList().add(userId);
        session.setAttribute("loggedInUser", loggedInUser);
        return "redirect:/user_details?user_id=" + userId;
    }

    @RequestMapping(path = "/stopfollow", method =
RequestMethod.GET)
    public String stopFollowUser(@RequestParam("user_id") int
userId, HttpServletRequest request, ModelMap model) {
        UserDAO userDAO = new UserDAO();
        HttpSession session = request.getSession();
        User loggedInUser = (User)
session.getAttribute("loggedInUser");

        boolean result = userDAO.unfollowUser(loggedInUser.getId(),
userId);
        if (!result) {
            model.addAttribute("removeFailure", "Remove Failed!
Please try again!");
            return "userdetails";
        }

        loggedInUser.getFollowList().remove(userId);
        session.setAttribute("loggedInUser", loggedInUser);
        return "redirect:/user_details?user_id=" + userId;
    }
}
```

RestaurantController.java

```
package
Controller;

                import DAO.*;
                import Model.Data.*;
                import java.util.*;
                import javax.servlet.http.*;


                import org.springframework.stereotype.*;
                import org.springframework.ui.*;
                import org.springframework.web.bind.annotation.*;

                @Controller
                @RequestMapping("/restaurant")
                public class RestaurantController {

                    @RequestMapping(path = "/{restaurantId:[\\d]+}", method =
                {RequestMethod.POST, RequestMethod.GET})
                    public String showRestaurant(@PathVariable("restaurantId") int
                restaurantId,
                                                 HttpServletRequest request, ModelMap
                model) {
                        RestaurantDAO restaurantDAO = new RestaurantDAO();
                        ReviewDAO reviewDAO = new ReviewDAO();
                        Restaurant restaurant =
                restaurantDAO.getRestaurant(restaurantId);

                        String keyword = request.getParameter("keyword");
                        int resultCount;
                        if (keyword == null || keyword.equals("")) {
                            resultCount = restaurant.getReviews().size();
                        } else {
                            resultCount =
                reviewDAO.getReviewCountByRestaurantId(restaurantId, keyword);
                        }
                        int eachPageCount = 3;
                        int pageCount = resultCount / eachPageCount;
                        if (resultCount % eachPageCount != 0) {
                            pageCount++;
                        }
                        List<Review> reviews =
                reviewDAO.getReviewsByRestaurantId(restaurantId, keyword, 1,
                eachPageCount);
```

```
        model.addAttribute("restaurant", restaurant);
        model.addAttribute("reviews", reviews);
        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", 1);
        return "restaurant";
    }

    @RequestMapping(path = "/{restaurantId:[\\d]+}/search_review",
method = RequestMethod.GET)
    public String showRestaurant(@PathVariable("restaurantId") int
restaurantId, @RequestParam("keyword") String keyword,
                                 @RequestParam("page") int page,
ModelMap model) {
        RestaurantDAO restaurantDAO = new RestaurantDAO();
        ReviewDAO reviewDAO = new ReviewDAO();
        Restaurant restaurant =
restaurantDAO.getRestaurant(restaurantId);

        int resultCount;
        if (keyword == null || keyword.equals("")) {
            resultCount = restaurant.getReviews().size();
        } else {
            resultCount =
reviewDAO.getReviewCountByRestaurantId(restaurantId, keyword);
        }
        int eachPageCount = 3;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Review> reviews =
reviewDAO.getReviewsByRestaurantId(restaurantId, keyword, page,
eachPageCount);
        model.addAttribute("restaurant", restaurant);
        model.addAttribute("reviews", reviews);
        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", page);
        return "restaurant";
    }
```

```
}
```

SearchRestaurantController.java

```
package
Controller;

import DAO.*;
import Model.Data.*;
import java.util.*;
import javax.servlet.http.*;


import org.springframework.stereotype.*;
import org.springframework.ui.*;
import org.springframework.web.bind.annotation.*;

@Controller
public class SearchRestaurantController {

    @RequestMapping(path = "/search", method = RequestMethod.POST)
    public String searchRestaurant(HttpServletRequest request,
ModelMap model) {
        RestaurantDAO restaurantDAO = new RestaurantDAO();
        String keyword = request.getParameter("keyword");
        if (keyword == null) {
            keyword = "";
        }
        int resultCount =
restaurantDAO.getRestaurantCount(keyword);
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Restaurant> resultList =
restaurantDAO.getRestaurantes(keyword, 1, eachPageCount);

        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", 1);
        model.addAttribute("searchResult", resultList);
        return "search";
```

```
    }

    @RequestMapping(path = "/search", method = RequestMethod.GET)
    public String searchRestaurant(@RequestParam("keyword") String
keyword, @RequestParam("page") int page, ModelMap model) {
        RestaurantDAO restaurantDAO = new RestaurantDAO();
        int resultCount =
restaurantDAO.getRestaurantCount(keyword);
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Restaurant> resultList =
restaurantDAO.getRestaurantes(keyword, page, eachPageCount);

        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", page);
        model.addAttribute("searchResult", resultList);
        return "search";
    }


}
```

WriteReviewController.java

```
package
Controller;

import DAO.*;
import Model.Data.*;
import Model.Form.*;
import java.util.*;
import javax.servlet.http.*;
import org.springframework.stereotype.*;
import org.springframework.ui.*;
import org.springframework.validation.*;
import org.springframework.validation.annotation.*;
import org.springframework.web.bind.annotation.*;

@Controller
```

```
@RequestMapping("/writeareview")
public class WriteReviewController {

    @RequestMapping(path = "", method = RequestMethod.GET)
    public String showView() {
        return "writeareview";
    }

    @RequestMapping(path = "/search", method = RequestMethod.POST)
    public String searchRestaurant(HttpServletRequest request,
ModelMap model) {
        RestaurantDAO restaurantDAO = new RestaurantDAO();
        String keyword = request.getParameter("keyword");
        if (keyword == null) {
            keyword = "";
        }
        int resultCount =
restaurantDAO.getRestaurantCount(keyword);

        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Restaurant> resultList =
restaurantDAO.getRestaurantes(keyword, 1, eachPageCount);

        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", 1);
        model.addAttribute("searchResult", resultList);

        return "writeareview";
    }

    @RequestMapping(path = "/search", method = RequestMethod.GET)
    public String searchRestaurant(@RequestParam("keyword") String
keyword, @RequestParam("page") int page, ModelMap model) {
        RestaurantDAO restaurantDAO = new RestaurantDAO();
        if (keyword == null) {
            keyword = "";
        }
```

```
        int resultCount =
restaurantDAO.getRestaurantCount(keyword);

        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Restaurant> resultList =
restaurantDAO.getRestaurantes(keyword, page, eachPageCount);

        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", page);
        model.addAttribute("searchResult", resultList);

        return "writeareview";
    }

    @RequestMapping(path = "/restaurant/{restaurantId:[\\d]+}",
method = RequestMethod.GET)
    public String writeReview(@PathVariable("restaurantId") int
restaurantId, ModelMap model, StarsComment review) {
        RestaurantDAO restaurantDAO = new RestaurantDAO();
        Restaurant restaurant =
restaurantDAO.getRestaurant(restaurantId);
        model.addAttribute("restaurant", restaurant);
        model.addAttribute("starsComment", review);
        return "rate";
    }

    @RequestMapping(path = "/restaurant/{restaurantId:[\\d]+}",
method = RequestMethod.POST)
    public String submitReview(@PathVariable("restaurantId") int
restaurantId, ModelMap model, HttpServletRequest request,
                               @Validated
@ModelAttribute("starsComment") StarsComment starsComment,
BindingResult result) {
        RestaurantDAO restaurantDAO = new RestaurantDAO();
        Restaurant restaurant =
restaurantDAO.getRestaurant(restaurantId);
        model.addAttribute("restaurant", restaurant);
```

```
        if (result.hasErrors()) {
            return "rate";
        }

        HttpSession session = request.getSession();
        if (session.getAttribute("loggedInUser") == null) {
            model.addAttribute("failure", true);
            return "rate";
        }

        User user = (User) session.getAttribute("loggedInUser");
        ReviewDAO reviewDAO = new ReviewDAO();
        if (!reviewDAO.addReview(user.getId(), restaurantId,
starsComment)) {
            model.addAttribute("postFailure", true);
            return "rate";
        }

        return "redirect:/restaurant/" + restaurant.getId() +
"?page=1";
    }

}
```