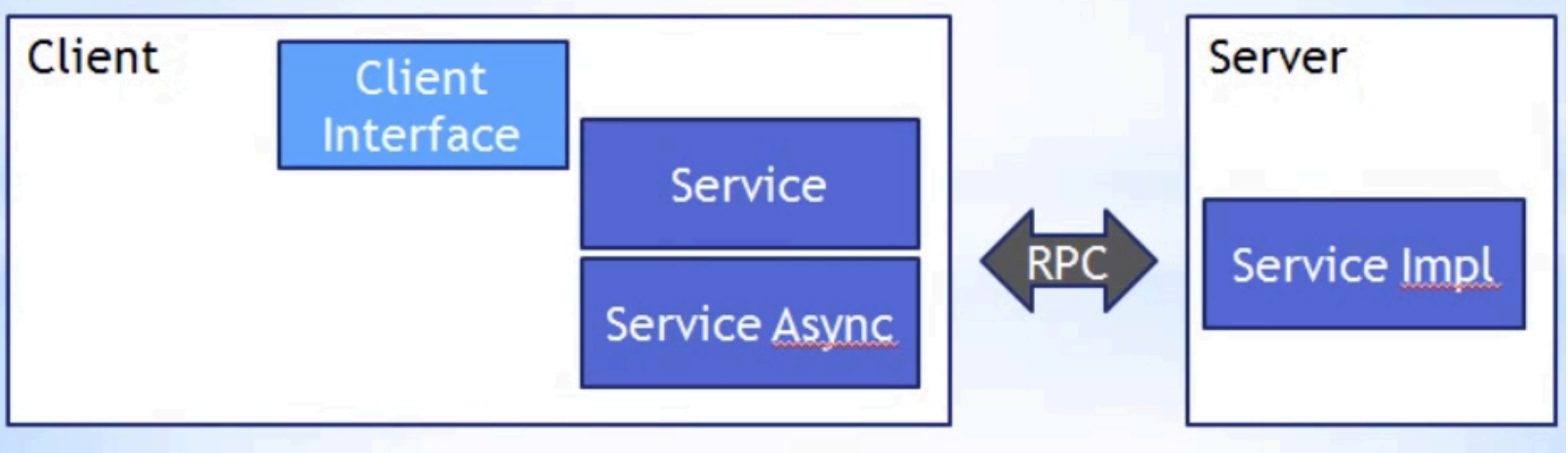
Healthy Eating Application

System Implementation Overview

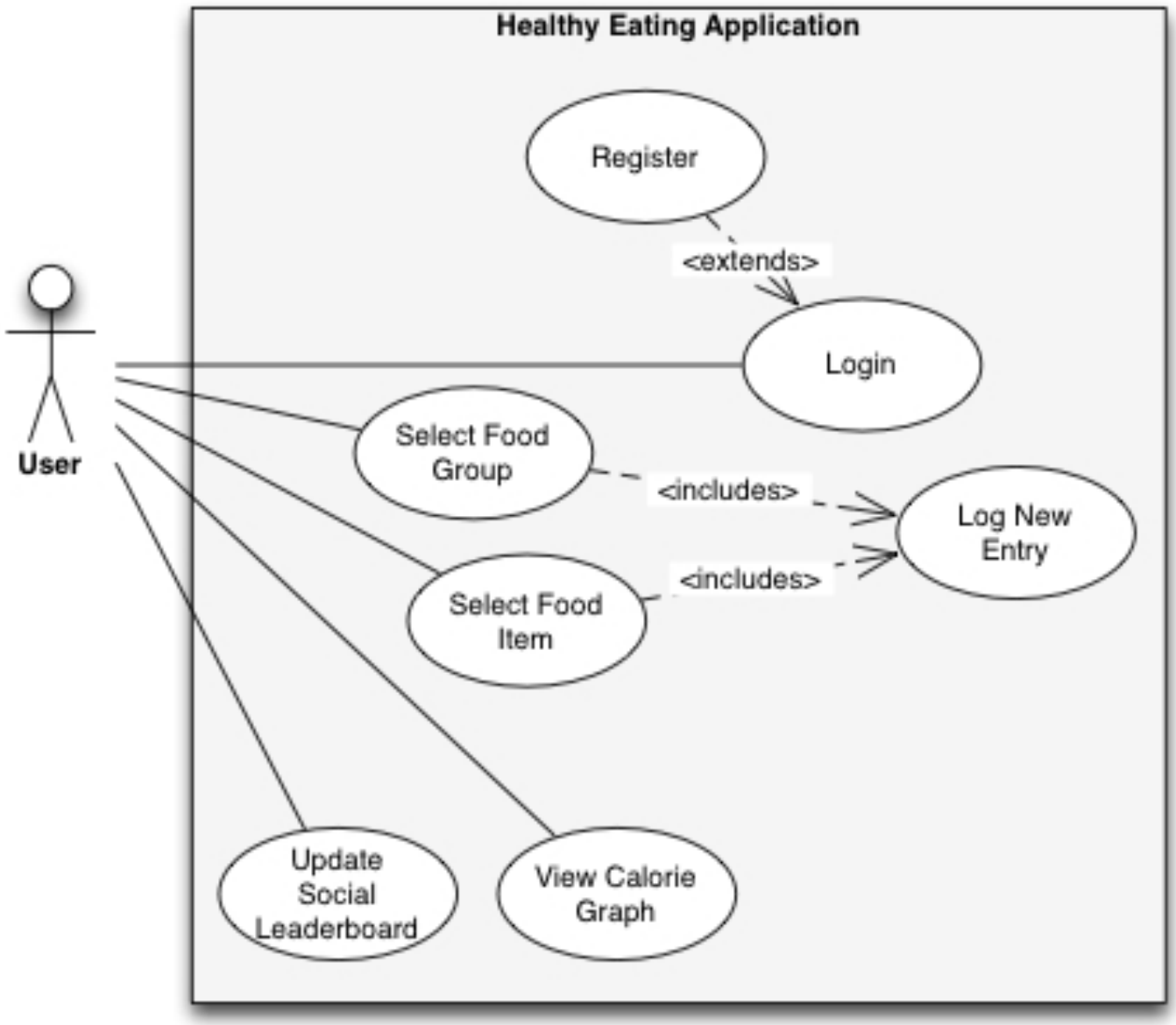
Rashna Razdan
Atulan Zaman Deep
Kevin Lee
Nadeem Ahmad

System Overview

System Scope



Key Use Cases



Underlying Technical Complexity

- Remote Procedure to communicate with MySQL database
- BCrypt framework
- Facebook API
- Google Web Toolkit Visualizations API
- Google Web Tookit Framework

Implementation Process

Process and Tools

- Use Case → Class
- Used GWT
- Eclipse
- SQL Workbench

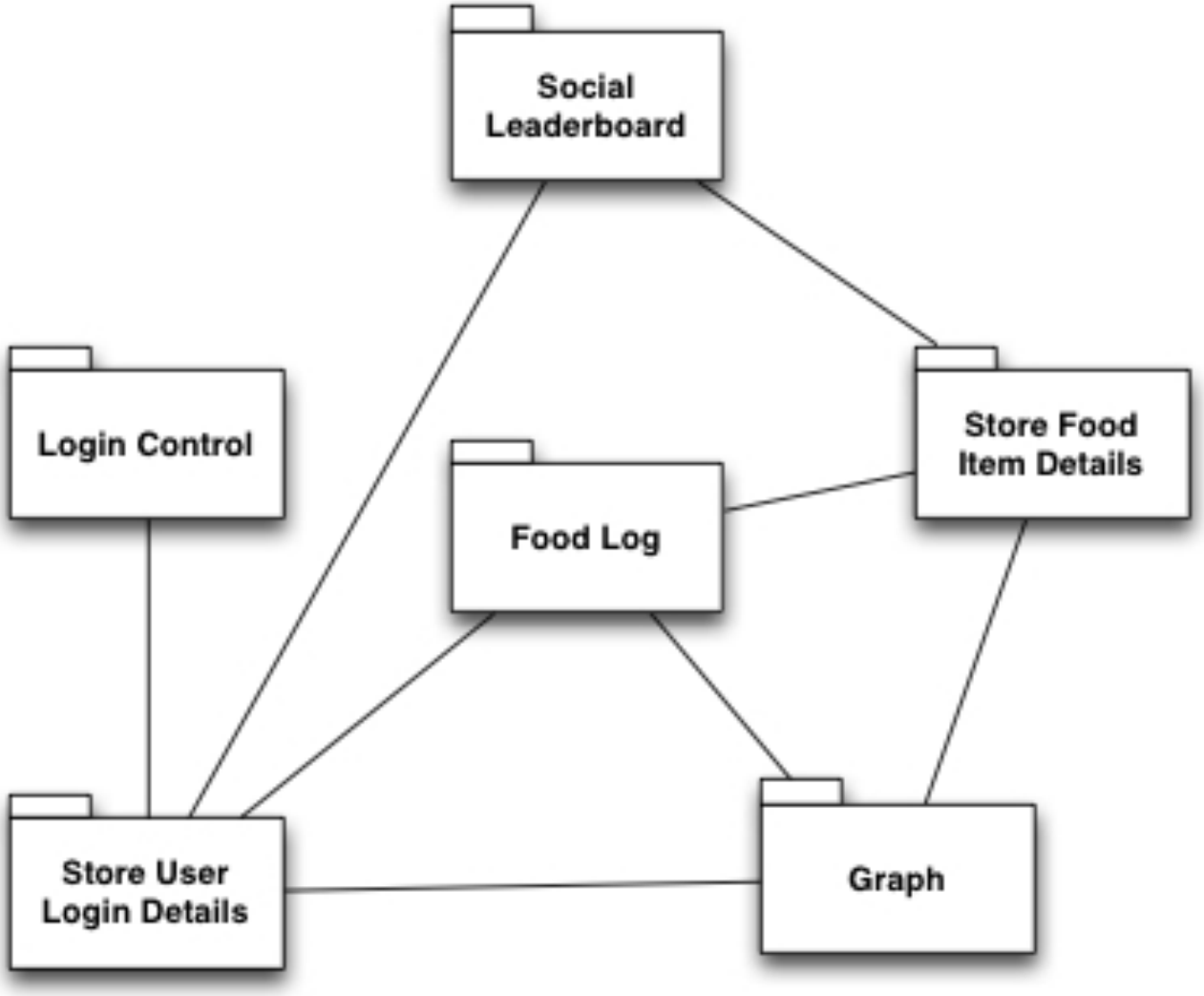
- Accomplishments {see use cases}
- Further Improvements

Subsystem Design

- Important subsystems:
  - Login Management (includes session management)
  - Social Media Management
  - User Interface
    - Login <<component>>
    - Food Log <<component>>
    - Calorie Graph <<component>>
    - Leader Board <<component>>
  - Healthy Eating Application Database

More information about components in following slide

Connectors Between Components



Control Flow Implementation

Notable Code Example (1 contd)

```
// Listen for mouse events on the login button.
loginButton.addClickHandler(new ClickHandler() {
@Override
public void onClick(ClickEvent event) {
      rpcLogin.authenticateUser(usernameBox.getText(),passwordBox.getText(), new LoginButtonCallback())
  }
});
```

The code above gives an example of the RPC call mechanism used to communicate with the server. The Facade design pattern was used to communicate with the server. An interface for all methods that talked to the server was created as shown below.

```
package com.google.gwt.sample.healthyeatingapp.client;

import com.google.gwt.user.client.rpc.AsyncCallback;

//Asynchronous interface specifies the same methods you have written into the synchronous interface, with two
//differences: The return type is always void and every method has an additional AsyncCallback parameter
public interface DBConnectionServiceAsync {
    public void authenticateUser(String username, String password, AsyncCallback<User> callback);
    public void authenticateFacebookUser(String firstName, String lastName, AsyncCallback<User> callback);
    public void logout(AsyncCallback callback);

    public void register(String newusername, String newpassword, String newfirstname, String newlastname, AsyncCallback callback);
    public void GetFriendsPoints(String username, AsyncCallback<Points> callback);
    public void getUserCalories(String username, AsyncCallback<String> callback);


    public void IsFriendUser(String firstName, String lastName, AsyncCallback<FriendUser> callback);
    public void GetFoodNames(AsyncCallback<String> callback);
    public void QueryFoodLog(String userName, String date, AsyncCallback<String> callback);
    public void getUserName(AsyncCallback<String> callback);
    public void InsertFoodLog(String userID, String foodName, String date, int calories, AsyncCallback callback);
}
```

Notable Code Example (1 cont.)

```
private class LoginButtonCallback implements AsyncCallback {

    @Override
    public void onFailure(Throwable caught){
        System.out.println("fail login");
    }

    @Override
    public void onSuccess(Object result) {

        userLoginTrack = (User)result;

        if(userLoginTrack == null){
            loadLoginAgain();
        }

        else if (userLoginTrack.getLoggedIn()){

            loadHomepage();

             //cookie stuff  *************************
             String sessionID = userLoginTrack.getSessionId();
             //set session cookie for 1 day expiry.
             final long DURATION = 1000 * 60 * 60 * 24 * 1;
             Date expires = new Date(System.currentTimeMillis() + DURATION);
             Cookies.setCookie("sid", sessionID, expires, null, "/", false);
             Cookies.setCookie("healthy_app_user", userLoginTrack.getUserName());
             //*****************************************************
        }
        else{
            loadLoginAgain();
        }
```

Every RPC Callback had an OnFailure() and an OnSuccess() method in order to seamlessly handle any unexpected and boundary cases that might cause errors while running server side code

Notable Code Example (2)

The code below highlights one of the application's 'novel' features: social media integration.

Code to get Friends list from Facebook. OnSuccess(), the JSONObject that is returned is parsed

Code that 're-paints' the graph every time a food item is inserted into Food Log. onLoadCallBack Runnable is called at every onSuccess for InsertFoodLog async method call

```
FacebookGraph.getStaticObject().RequestAuthorizationAndQueryGraph("me/friends", new Callback<JSONObject, Throwable>(){...}

//Logout code that logs out the user from the app as well as Facebook in case user was also logged in to FB
private class LogoutButtonCallback implements AsyncCallback {
            @Override
            public void onSuccess(Object result) {

                Cookies.removeCookie("JSESSIONID");
                Cookies.removeCookie("sid");
                Cookies.removeCookie("healthy_app_user");
                String token = FacebookGraph.getStaticObject().getToken();
                System.out.println(token);
                if(token!=null){
                    Window.Location.replace("https://www.facebook.com/logout.php?next=http://localhost:8888/"
                                            +"HealthyEatingApp.html?gwt.codesvr=127.0.0.1:9997&access_token=" + token);
                }
                FacebookGraph.getStaticObject().resetToken();
                Auth.get().clearAllTokens();
                loadLogin();
            }
        }
```

Main Design Patterns Used

- Composite (for widget creation)
- Singleton (for RootLayoutPanel)
- Facade (for server side method invocation)

Nonfunctional Requirements

- Robustness
  - OnFailure() and OnSuccess() methods implemented for every RPC call
  - Error checking and prevention of duplicate registered users [see register() and authenticateUser() in DBConnectionServiceImpl.java]
  - Passwords stored in hashed format via Bcrypt framework
- Performance
  - Asynchronous callbacks used in order to communicate with the server
- Portability
  - GWT produces cross browser compatible