

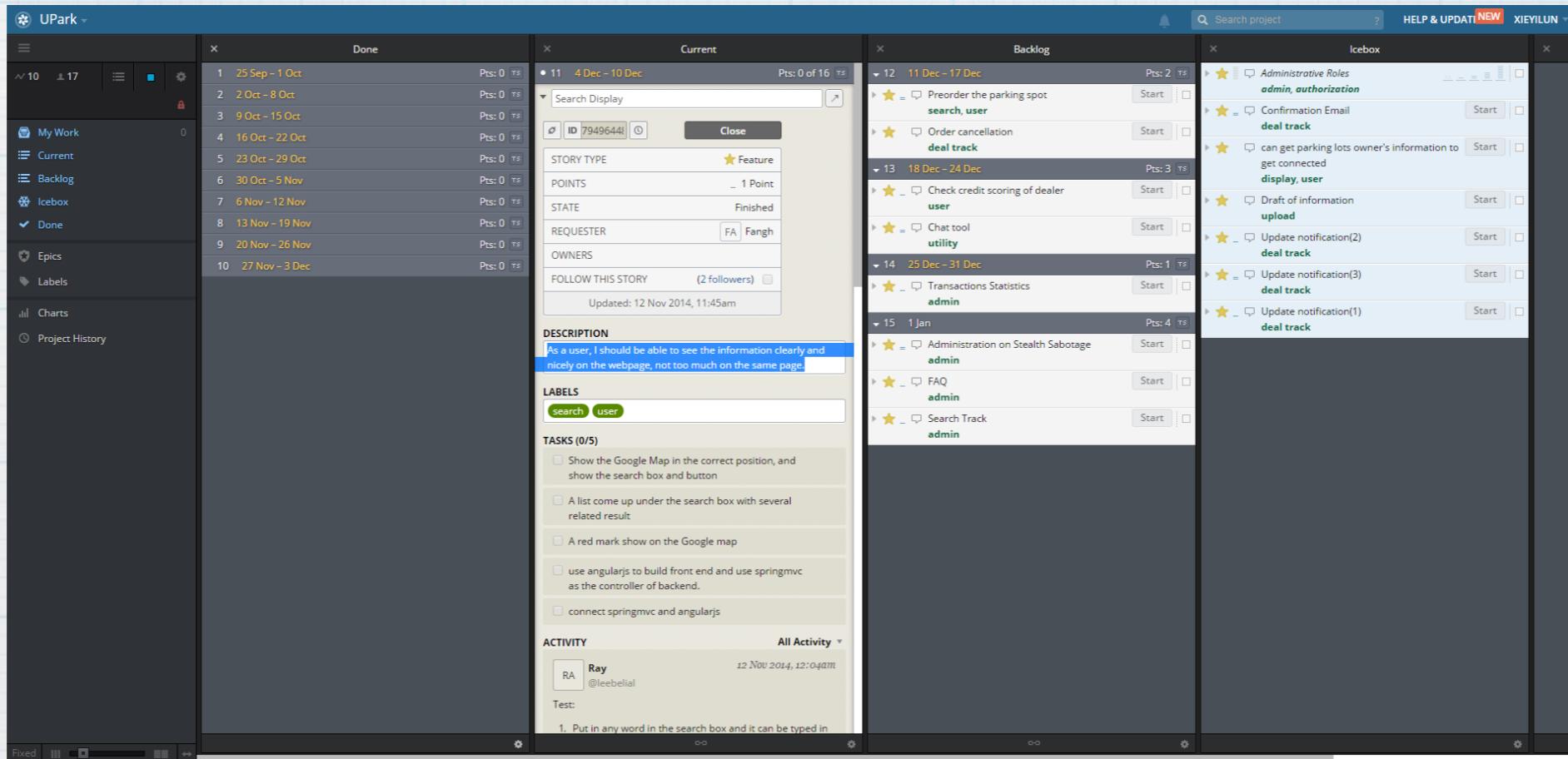
# U-PARK.INFO

Find your perfect parking space

- \* **Customer:** Aim to people who have spare parking places or need to find a parking space in advance.
- \* **Purpose:** UPark is a information provider for users who want to rent their idle parking spaces to earn some money or find available parking places around specific area in advance.

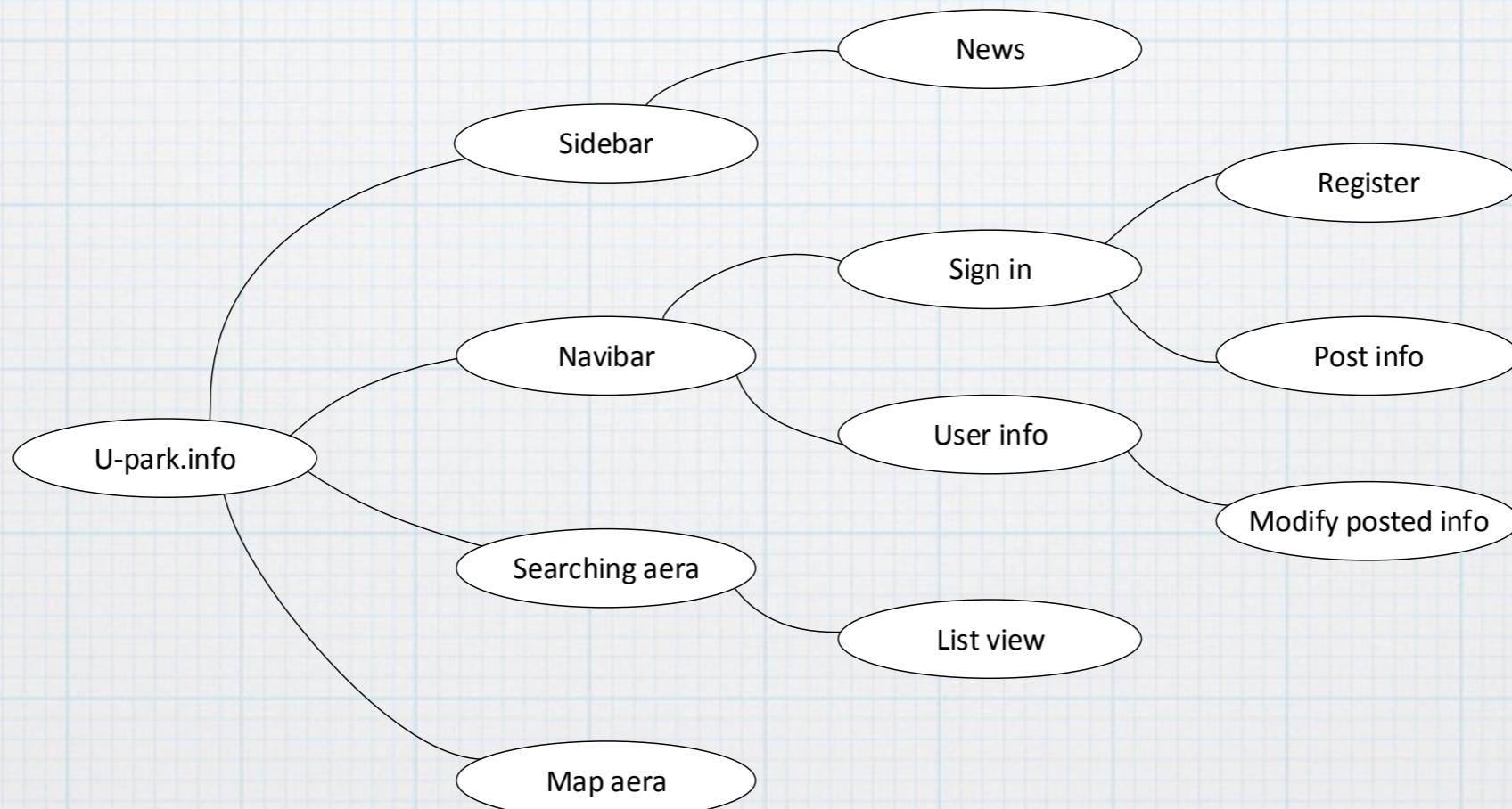
# Requirement Control

- \* We use pivotal track to track and handle the change of requirement.



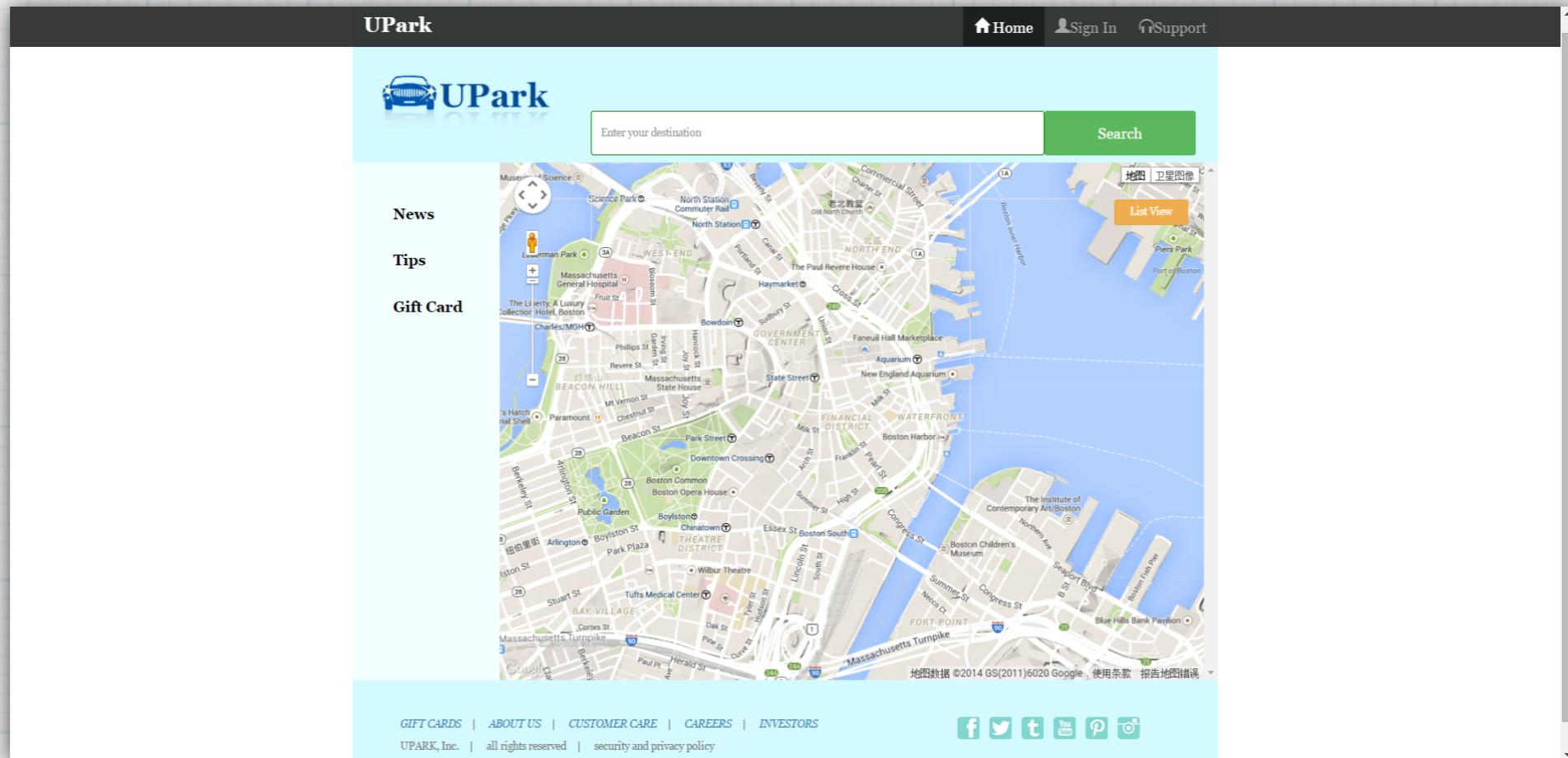
# Functional Requirement

- \* UI flow diagram

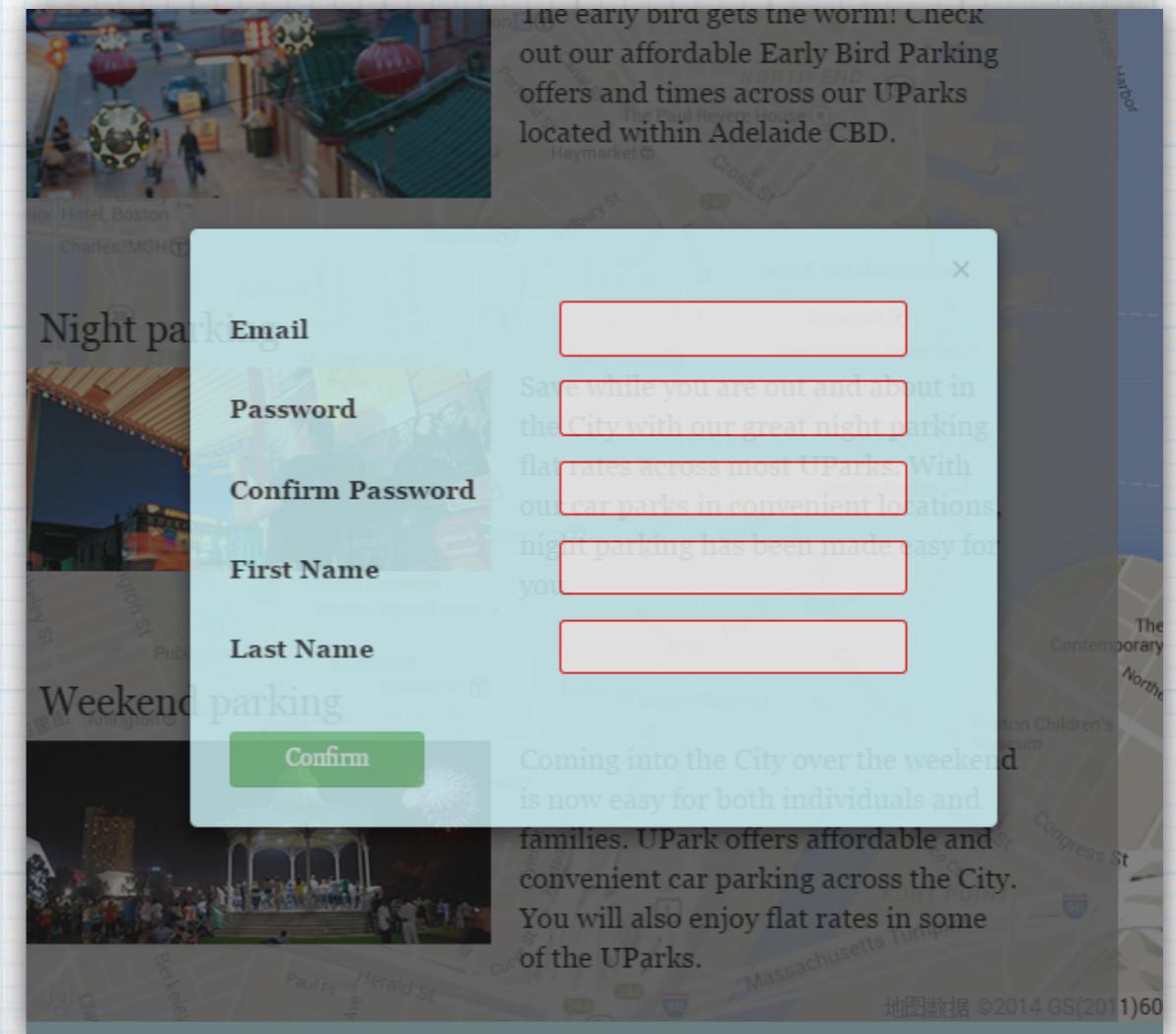
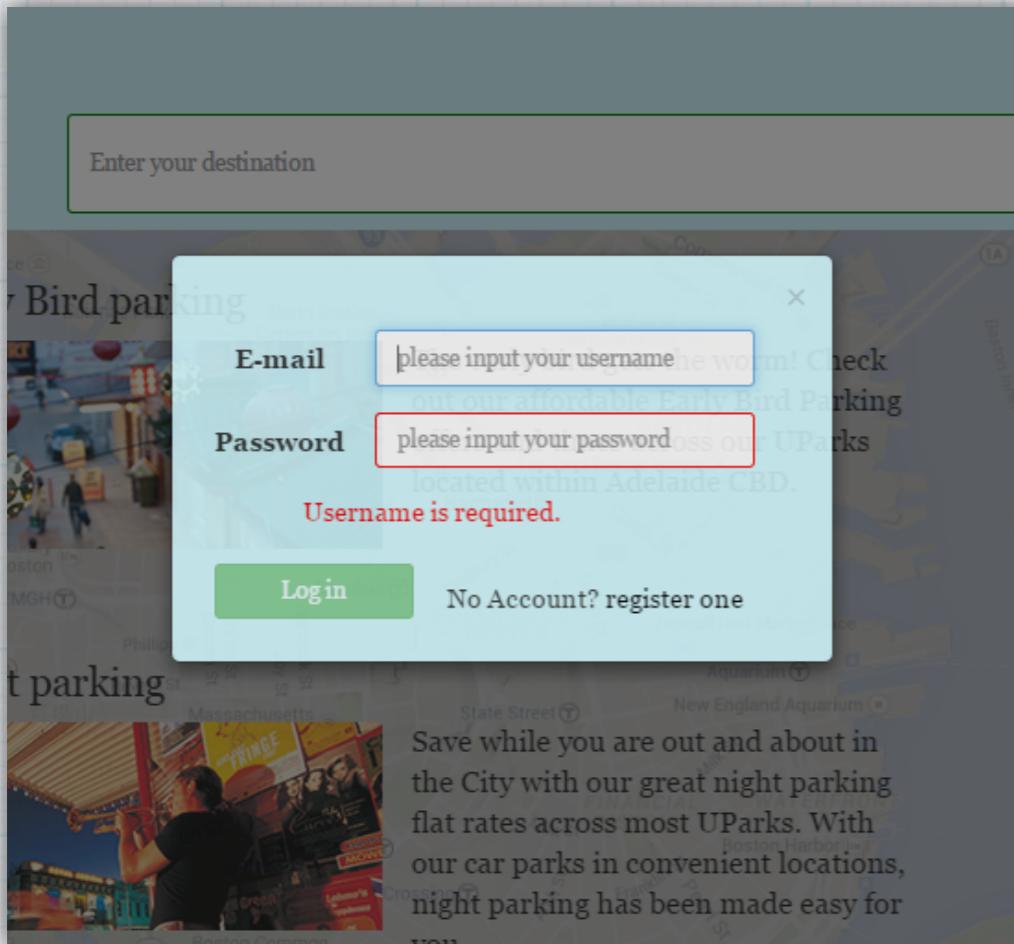


# Functional Requirement

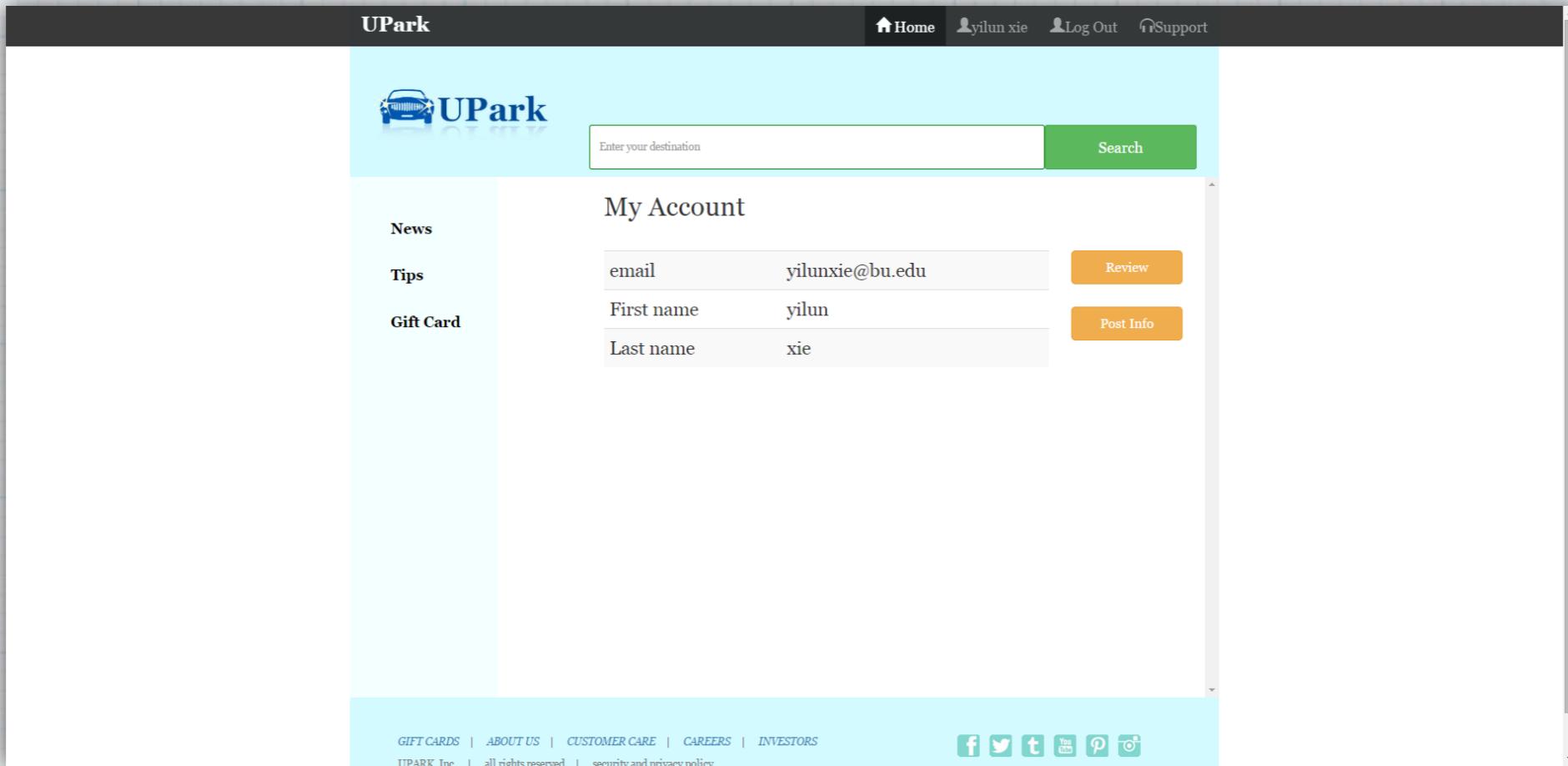
## \* UI Design



# Functional Requirement



# Functional Requirement



The screenshot shows the UPark website interface. At the top, there is a dark header bar with the 'UPark' logo on the left and navigation links for 'Home', 'yilun xie', 'Log Out', and 'Support' on the right.

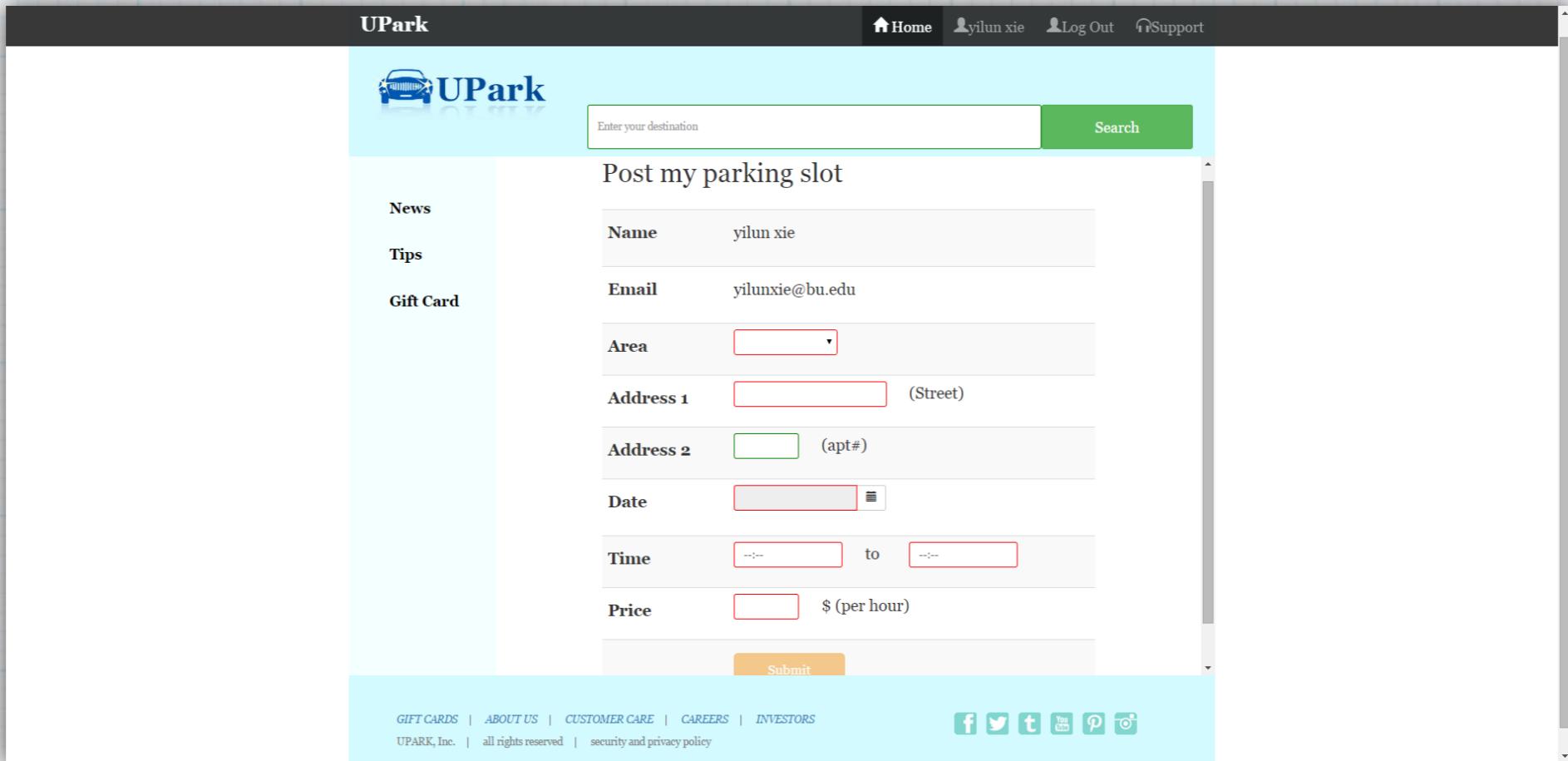
The main content area has a light blue header with the 'UPark' logo and a search bar containing the placeholder 'Enter your destination'. Below this is a green sidebar on the left with links for 'News', 'Tips', and 'Gift Card'.

The central part of the page displays 'My Account' information:

email	yilunxie@bu.edu	Review
First name	yilun	Post Info
Last name	xie	

At the bottom of the page, there is a light blue footer bar with links for 'GIFT CARDS', 'ABOUT US', 'CUSTOMER CARE', 'CAREERS', 'INVESTORS', and social media icons for Facebook, Twitter, YouTube, Pinterest, and Instagram.

# Functional Requirement



The screenshot shows the UPark website interface. At the top, there is a dark header bar with the 'UPark' logo, a user profile icon, and navigation links for 'Home', 'Log Out', and 'Support'. Below the header is a light blue navigation bar with the 'UPark' logo and a search bar containing the placeholder 'Enter your destination'. A green 'Search' button is located to the right of the search bar. The main content area features a title 'Post my parking slot' and a form for entering parking details. The form fields include:

- Name: yilun xie
- Email: yilunxie@bu.edu
- Area: (dropdown menu)
- Address 1: (Street) (text input field)
- Address 2: (apt#) (text input field)
- Date: (date input field)
- Time: (time input field) to (time input field)
- Price: (text input field) \$ (per hour)

At the bottom of the form is a yellow 'Submit' button. The footer of the page contains links for 'GIFT CARDS', 'ABOUT US', 'CUSTOMER CARE', 'CAREERS', and 'INVESTORS', along with copyright information: 'UPARK, Inc. | all rights reserved | security and privacy policy'. It also includes social media icons for Facebook, Twitter, YouTube, Pinterest, and Google+.

# Functional Requirement

The screenshot shows the UPark website interface. At the top, there is a dark header bar with the 'UPark' logo on the left and navigation links for 'Home', 'yilun xie', 'Log Out', and 'Support' on the right.

The main content area has a light blue header with the 'UPark' logo and a search bar containing the placeholder 'Enter your destination'. Below this is a green 'Search' button.

On the left side, there is a sidebar with three sections: 'News', 'Tips', and 'Gift Card'. The 'News' section is currently active, showing the heading 'My posted info' and a table of four rows. Each row contains information about a posted item, including address, date, time, price, and a delete icon ('x').

Address	Date	Time	Price	
15 Brighton Avenue #2, Boston, MA 02134	2015-01-06	15:00 20:00	\$15.00	x
20 Park Vale Avenue #3, Boston, MA 02134	2014-12-24	8:00 20:00	\$5.00	x
379 Washington Street, Boston, MA 02135	2015-01-03	16:00 22:00	\$20.00	x
Harvard University, 24 Quincy Street, Cambridge, MA 02138	2014-12-27	0:00 23:00	\$20.00	x

At the bottom of the page, there is a footer bar with links for 'GIFT CARDS', 'ABOUT US', 'CUSTOMER CARE', 'CAREERS', 'INVESTORS', 'UPARK Inc.', 'all rights reserved', and 'security and privacy policy'. To the right of these links are icons for social media platforms: Facebook, Twitter, Tumblr, YouTube, Pinterest, and Instagram.

# Functional Requirement

## \* User Story - Rental Information

\* As a renter, I should post the information of the price, available rental time, location, contact so that the tenant can get his wanted parking information.

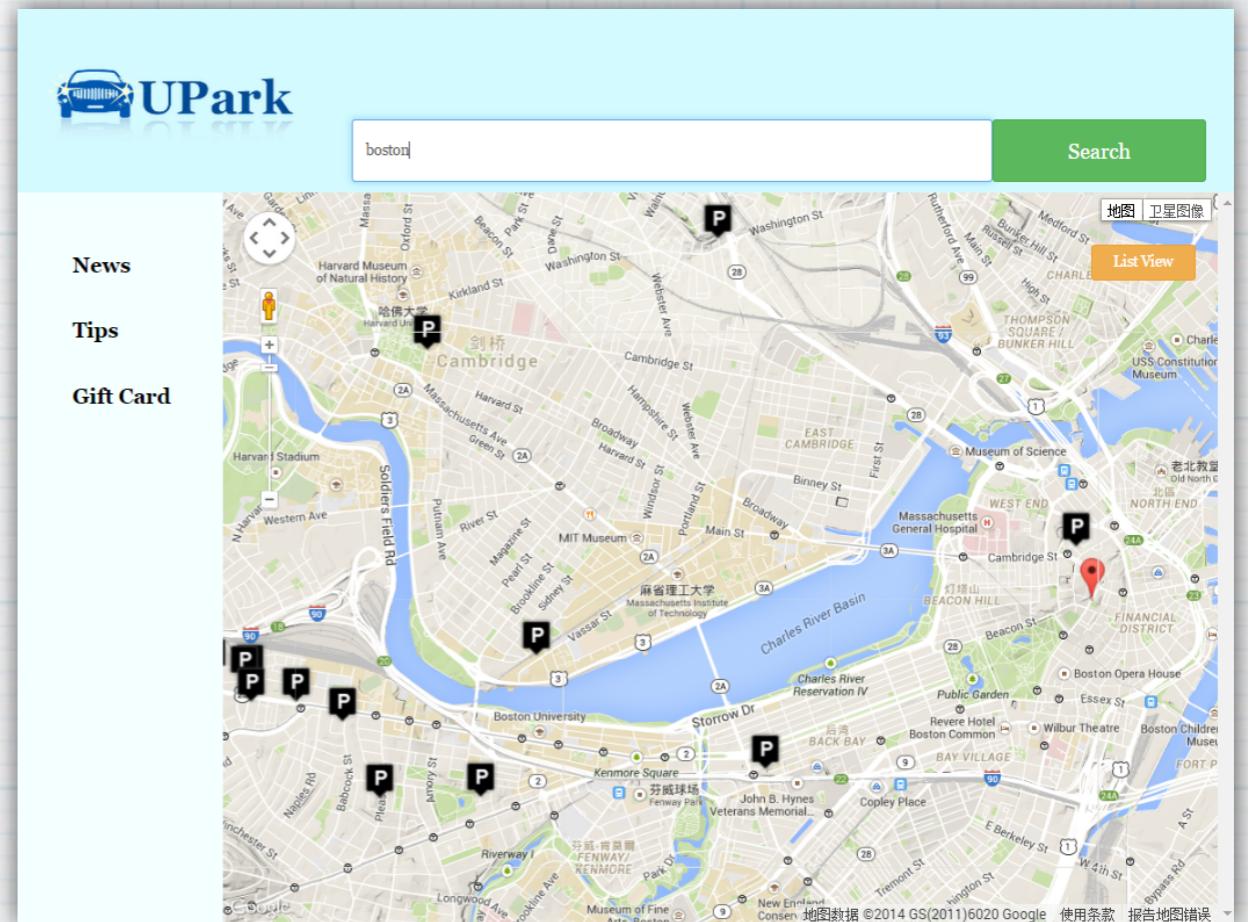
The screenshot shows the UPark website interface. At the top, there is a navigation bar with links for Home, User (yilun xie), Log Out, and Support. Below the navigation bar is the UPark logo and a search bar with the placeholder "Enter your destination". A green "Search" button is located to the right of the search bar. On the left side of the page, there is a sidebar with three categories: News, Tips, and Gift Card. The main content area is titled "Post my parking slot". It contains several input fields for posting a parking slot:

- Name: yilun xie
- Email: yilunxie@bu.edu
- Area: (dropdown menu)
- Address 1: (Street)
- Address 2: (apt#) (input field)
- Date: (input field with calendar icon)
- Time: (input field) to (input field)
- Price: \$ (per hour) (input field)

At the bottom of the form is a yellow "Submit" button. The footer of the website includes links for GIFT CARDS, ABOUT US, CUSTOMER CARE, CAREERS, INVESTORS, and UPARK, Inc., along with a note about security and privacy policy. Social media icons for Facebook, Twitter, YouTube, Pinterest, and Instagram are also present at the bottom right.

# Functional Requirement

- \* User Story - Result display
- \* As a user, I should be able to see the searching results on Google Map so it could be easier for me to find the location.



# Functional Requirement

## \* User Story – Information Review

- \* As a renter, I'll be able to review all the information that I provided about my parking space so I can make some changes if I change my mind.



The screenshot shows a web application interface for 'UPark'. At the top, there's a logo with a car icon and the text 'UPark'. Below the logo is a search bar containing the word 'boston' and a green 'Search' button. To the left of the main content area, there are three navigation links: 'News', 'Tips', and 'Gift Card'. The main content area is titled 'My posted info' and displays a table of five rows, each representing a posted parking space. The columns in the table are 'Address', 'Date', 'Time', and 'Price'. Each row includes a small 'x' icon at the end of the price column.

Address	Date	Time	Price	
15 Brighton Avenue #2, Boston, MA 02134	2015-01-06	15:00 20:00	\$15.00	x
20 Park Vale Avenue #3, Boston, MA 02134	2014-12-24	8:00 20:00	\$5.00	x
379 Washington Street, Boston, MA 02135	2015-01-03	16:00 22:00	\$20.00	x
Harvard University, 24 Quincy Street, Cambridge, MA 02138	2014-12-27	0:00 23:00	\$20.00	x

# Functional Requirement

## \* User Story – Information Review

- \* As a renter, I'll be able to review all the information that I provided about my parking space so I can make some changes if I change my mind.

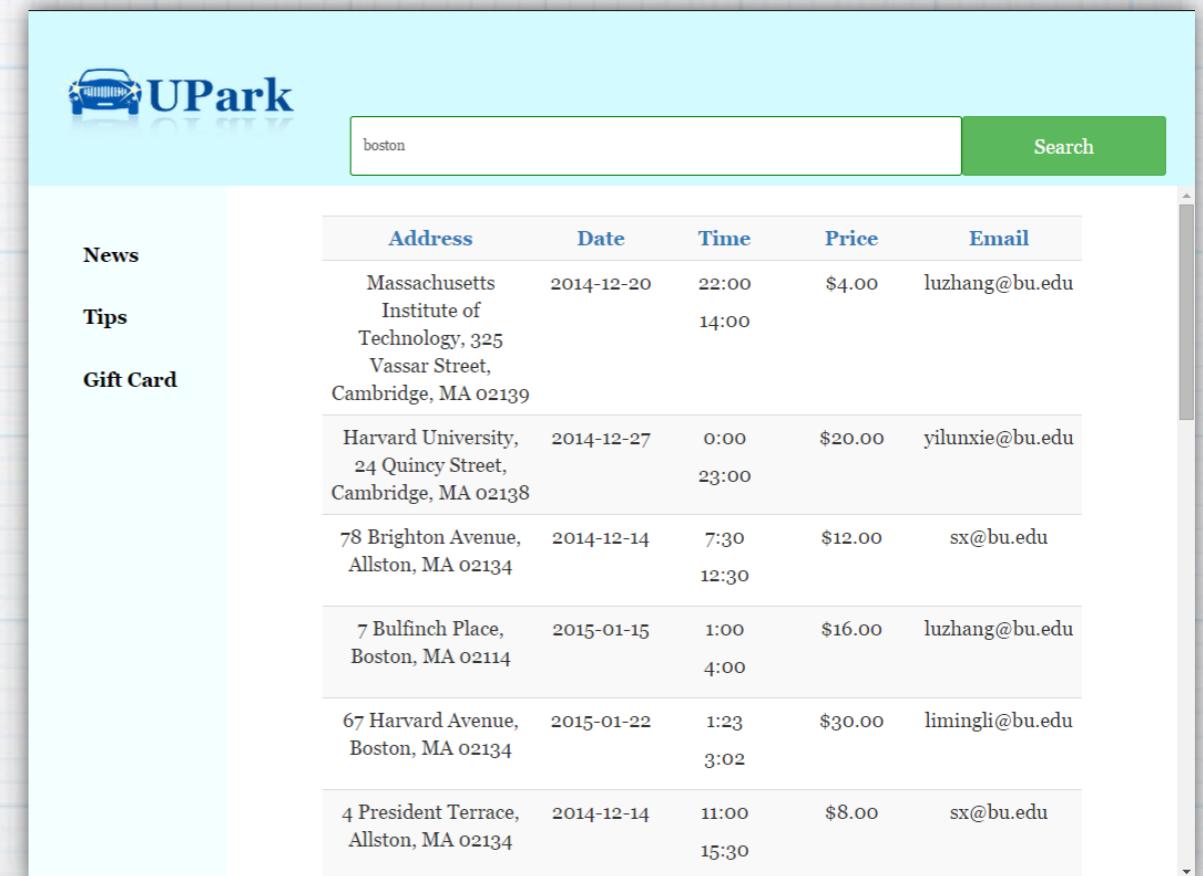


The screenshot shows a web application interface for 'UPark'. At the top, there's a logo with a car icon and the text 'UPark'. Below the logo is a search bar containing the word 'boston' and a green 'Search' button. To the left of the main content area, there are three navigation links: 'News', 'Tips', and 'Gift Card'. The main content area is titled 'My posted info' and displays a table of five rows, each representing a posted parking space. The columns in the table are 'Address', 'Date', 'Time', and 'Price'. Each row includes a small 'x' icon at the end of the 'Price' column.

Address	Date	Time	Price	
15 Brighton Avenue #2, Boston, MA 02134	2015-01-06	15:00 20:00	\$15.00	x
20 Park Vale Avenue #3, Boston, MA 02134	2014-12-24	8:00 20:00	\$5.00	x
379 Washington Street, Boston, MA 02135	2015-01-03	16:00 22:00	\$20.00	x
Harvard University, 24 Quincy Street, Cambridge, MA 02138	2014-12-27	0:00 23:00	\$20.00	x

# Functional Requirement

- \* User Story – Result display
- \* As a user, I should be able to order the result in different ways so I don't need to deal with too much unnecessary information.



The screenshot shows a website for 'UPark' with a light blue header. On the left, there are three menu items: 'News', 'Tips', and 'Gift Card'. In the center, there is a search bar with the word 'boston' and a green 'Search' button. Below the search bar, the results are displayed in a table format. The table has columns for Address, Date, Time, Price, and Email. There are six rows of data, each representing a parking spot in Boston:

	Address	Date	Time	Price	Email
	Massachusetts Institute of Technology, 325 Vassar Street, Cambridge, MA 02139	2014-12-20	22:00 14:00	\$4.00	luzhang@bu.edu
	Harvard University, 24 Quincy Street, Cambridge, MA 02138	2014-12-27	0:00 23:00	\$20.00	yilunxie@bu.edu
	78 Brighton Avenue, Allston, MA 02134	2014-12-14	7:30 12:30	\$12.00	sx@bu.edu
	7 Bulfinch Place, Boston, MA 02114	2015-01-15	1:00 4:00	\$16.00	luzhang@bu.edu
	67 Harvard Avenue, Boston, MA 02134	2015-01-22	1:23 3:02	\$30.00	limingli@bu.edu
	4 President Terrace, Allston, MA 02134	2014-12-14	11:00 15:30	\$8.00	sx@bu.edu

# Non-Functional Requirement

- \* UCompatible with Chrome, IE9+ and other common web browsers
- \* Need screen resolution higher than 1366\*768
- \* Not compatible with mobile web browsers
- \* Security guaranteed by https protocol
- \* Java runtime environment needed

# Functional Requirement

- \* Account Management

- \* Sign in/Log out
  - \* Register

- \* Information Management

- \* User information modification
  - \* Parking å information modification

- \* Search Service

- \* Search with address
  - \* Show result in list with different order

# Techniques



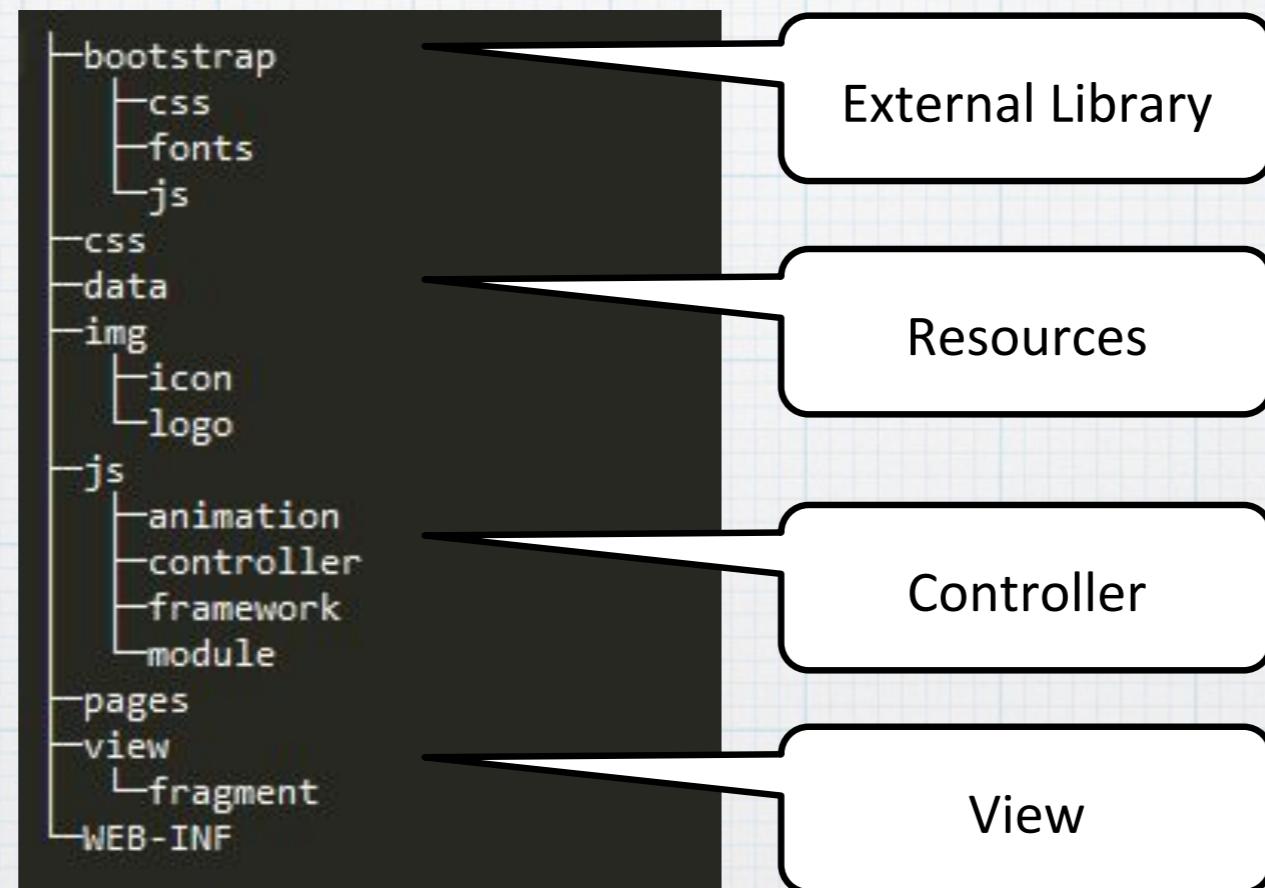
AngularJS



Google Maps Javascript API V3

# Directory Structure

# Standard Structure

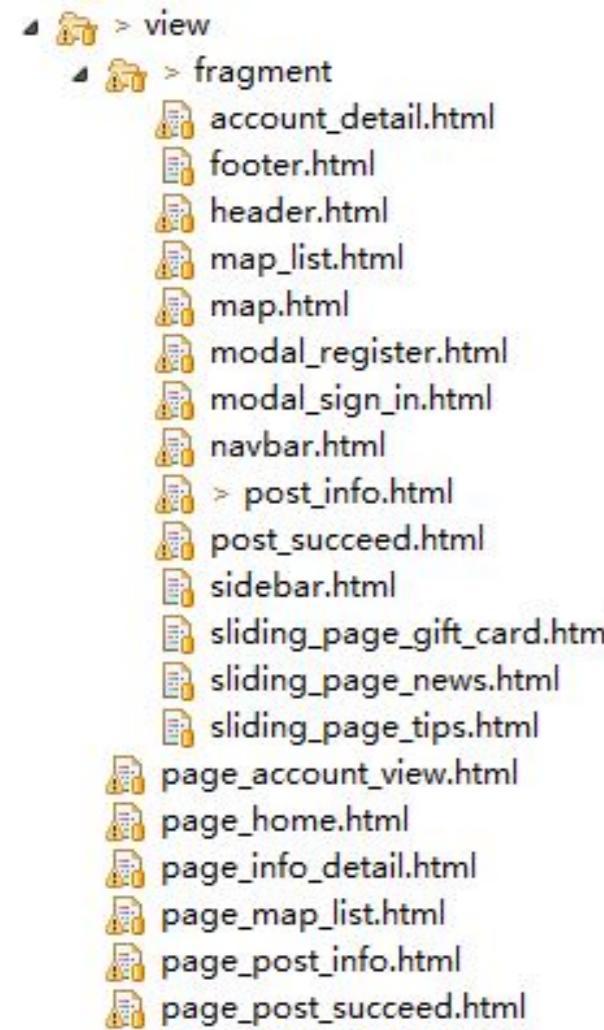


# Project Design Pattern

MWW = Model + View + Whatever

MVC or MWM

# View



UPark

Home Sign In Support

UPark

Enter your destination

Search

Map Satellite List View

News

Tips

Gift Card

Map data ©2014 Google | Terms of Use | Report a map error

GIFT CARDS | ABOUT US | CUSTOMER CARE | CAREERS |  
INVESTORS

f t y p o



E-mail  please input your username

Password  please input your password

**Log in** No Account? register one

Email

Password

Confirm Password

First Name

Last Name

**Confirm**

## My Account

email                   luzhang@bu.edu  
First name           Lu  
Last name           Zhang

Review

Post Info

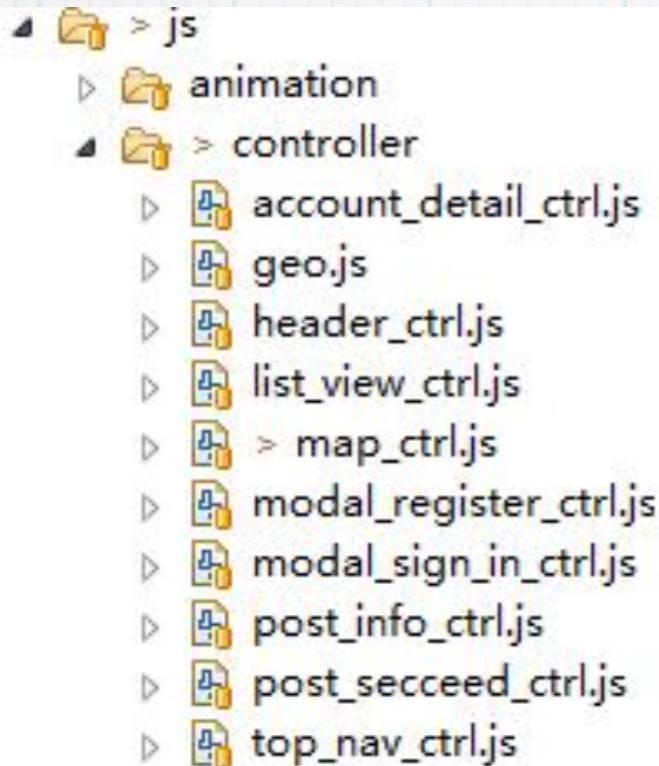
## My posted info

Address	Date	Time	Price	
12 Boston Street, Somerville, MA 02143	2014-12-06	15:00 16:00	\$5.00	x
10 Jamaicaway, Jamaica Plain, MA 02130	2014-12-07	12:00 15:00	\$10.00	x
344 Newbury Street, Boston, MA 02115	2014-12-10	9:00 12:00	\$12.00	x
Massachusetts Institute of Technology, 325 Vassar Street, Cambridge, MA 02139	2014-12-21	1:00 17:00	\$4.00	x
7 Bulfinch Place, Boston, MA 02114	2015-01-16	4:00 7:00	\$16.00	x

## Post my parking slot

Name                   Lu Zhang  
Email                   luzhang@bu.edu  
Area                     
Address 1            (Street)  
Address 2            (apt#)  
Date                      
Time                    to   
Price                    \$ (per hour)

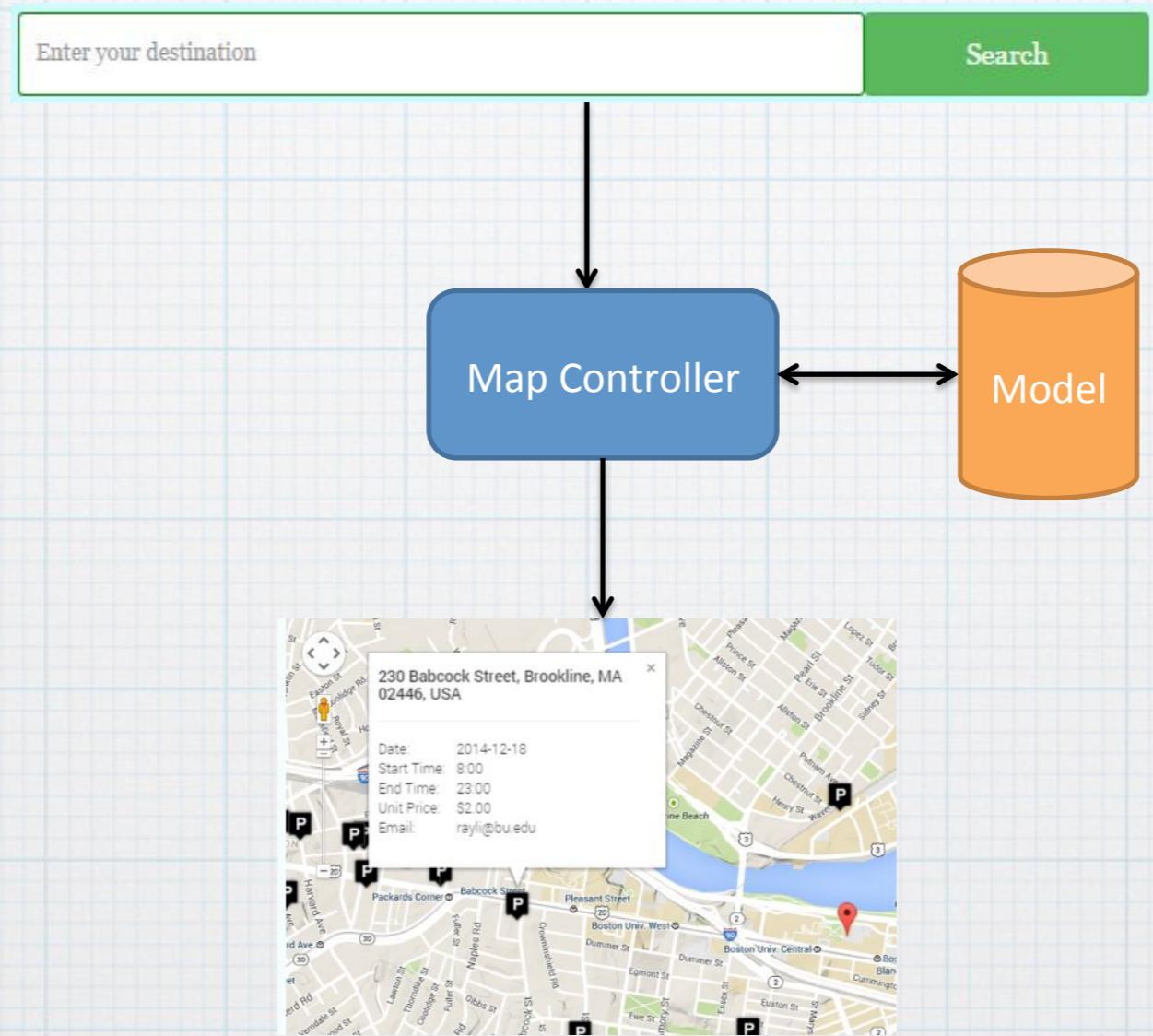
# Controller



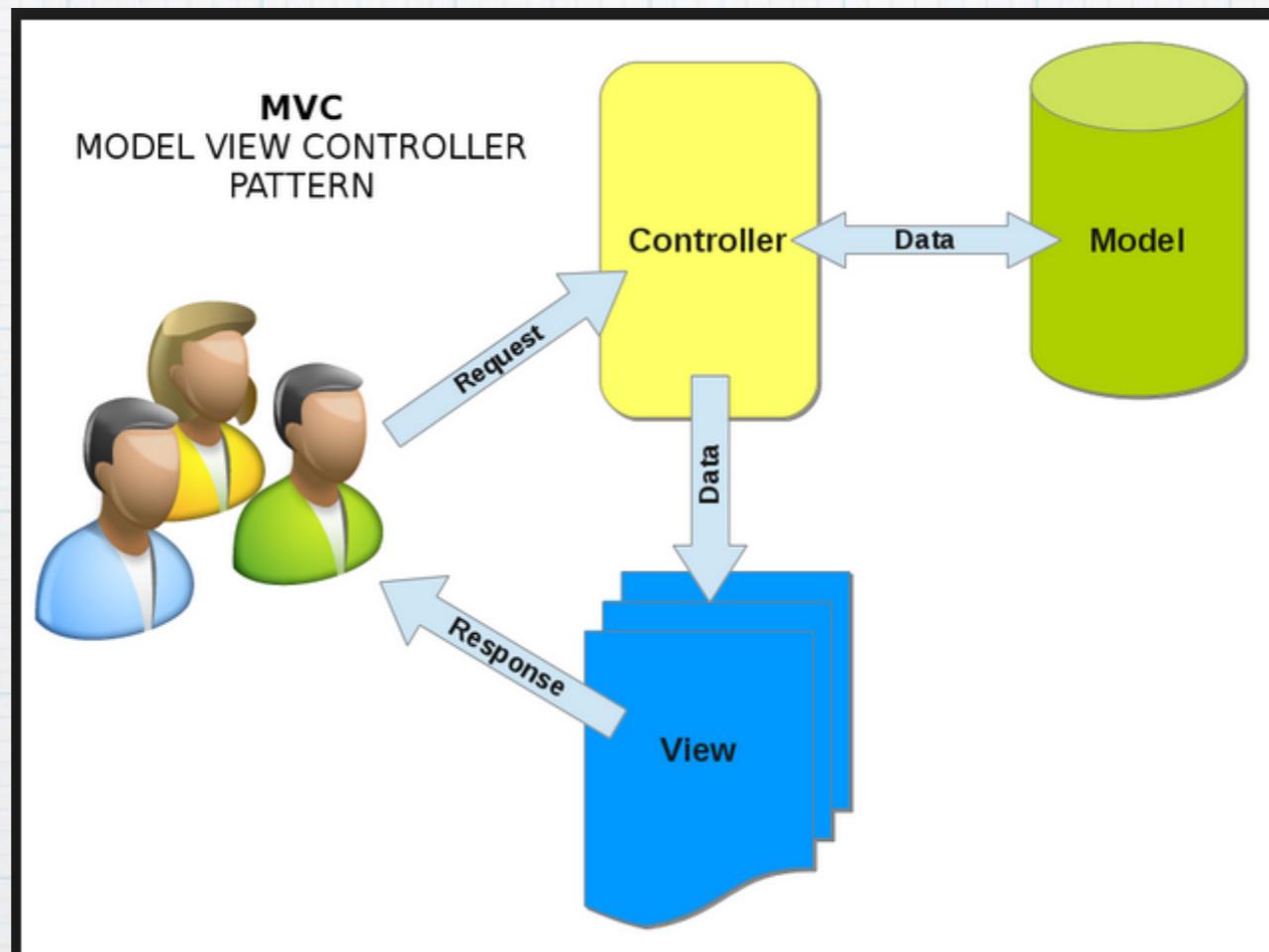
\$Scope

\$Resource

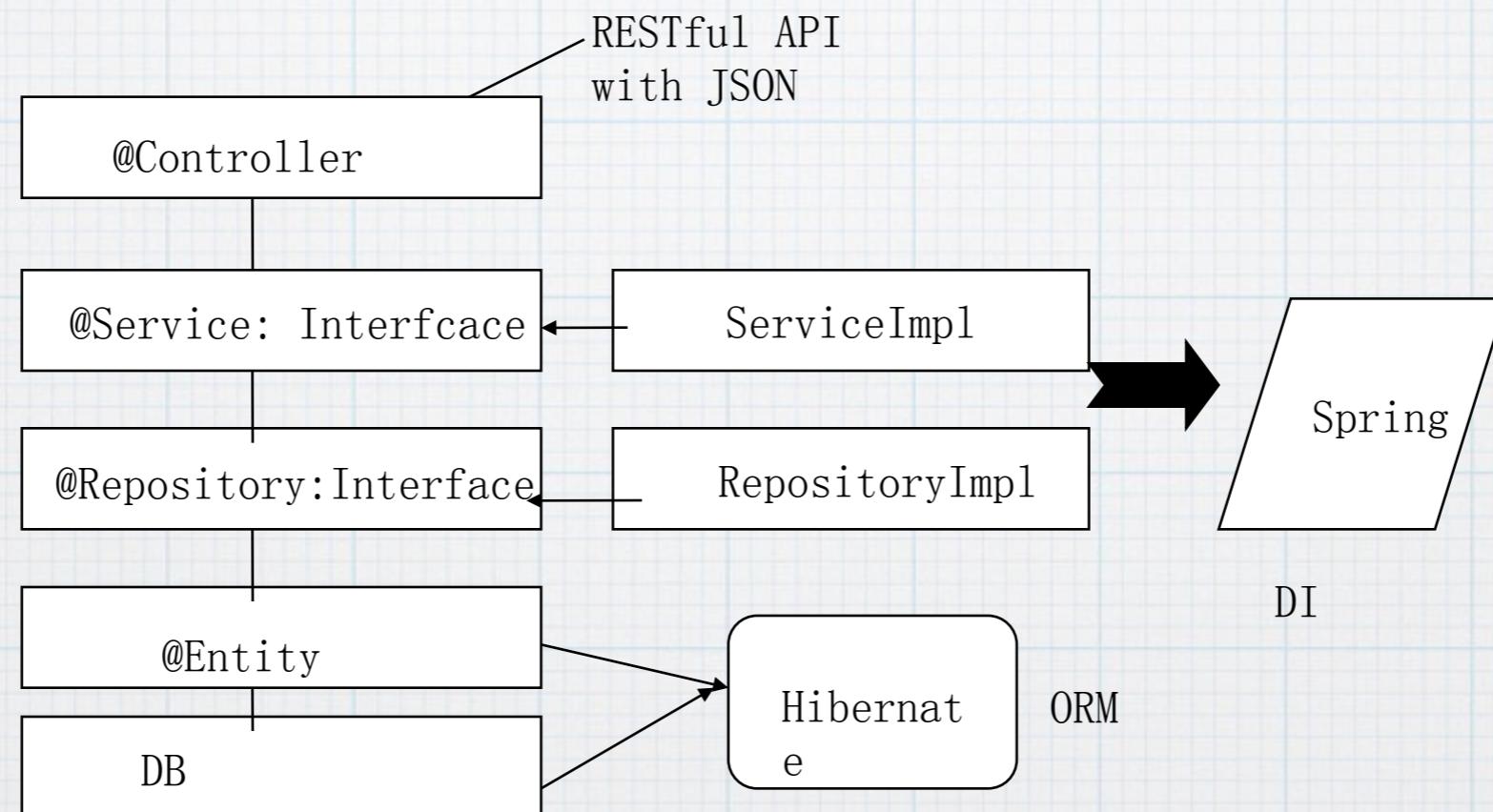
# Map Controller



# Spring MVC

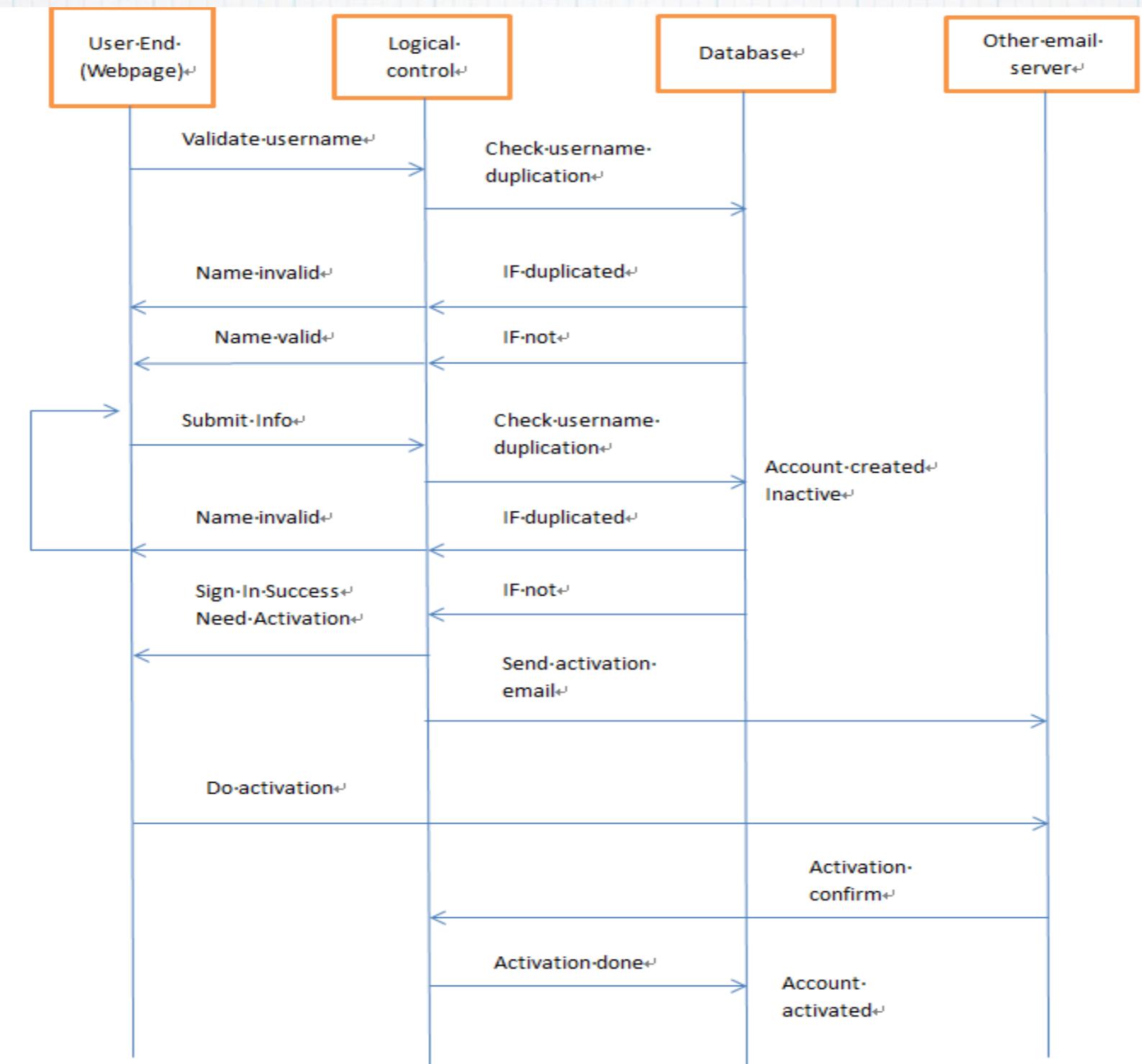


# Architecture

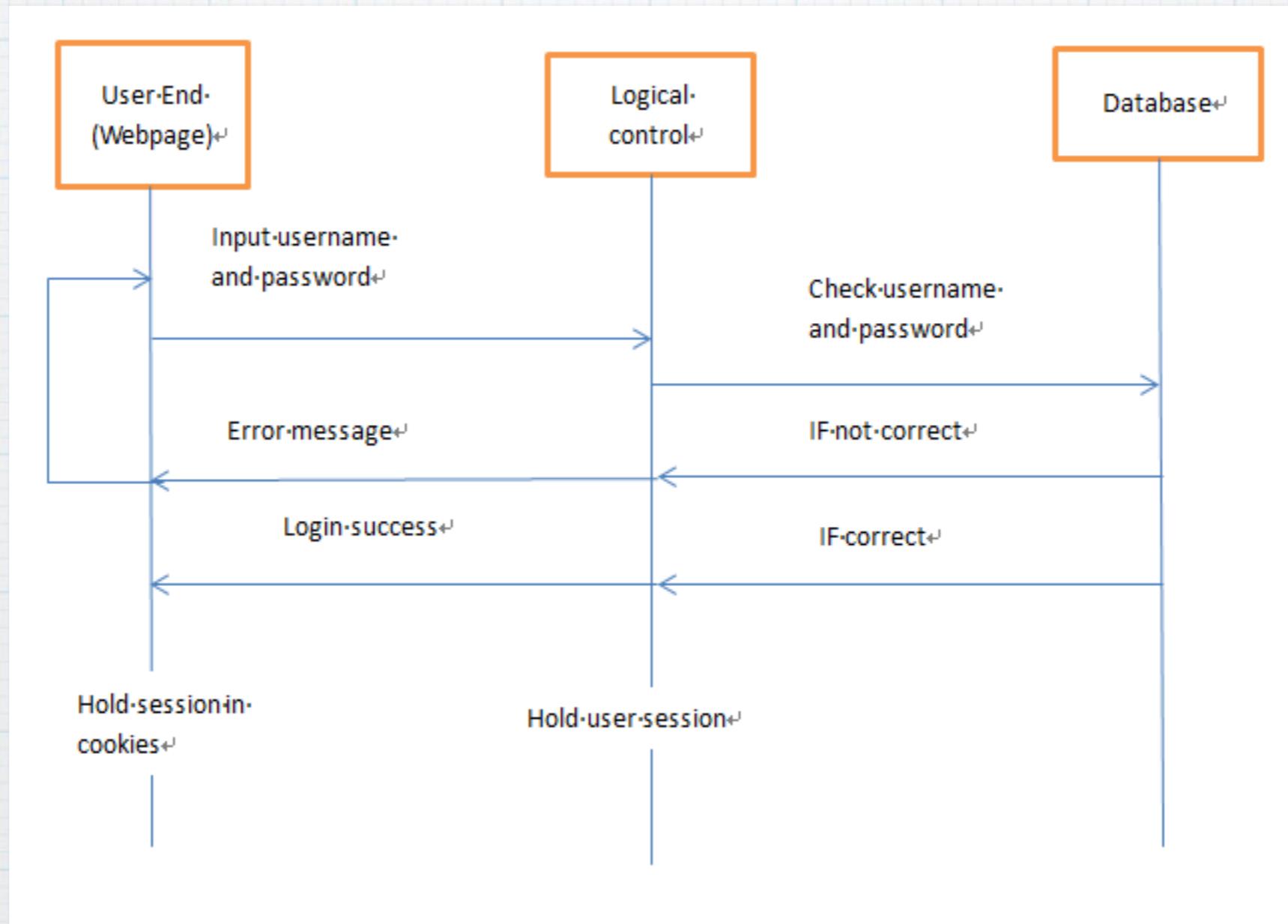


# Main Function Sequence Diagram

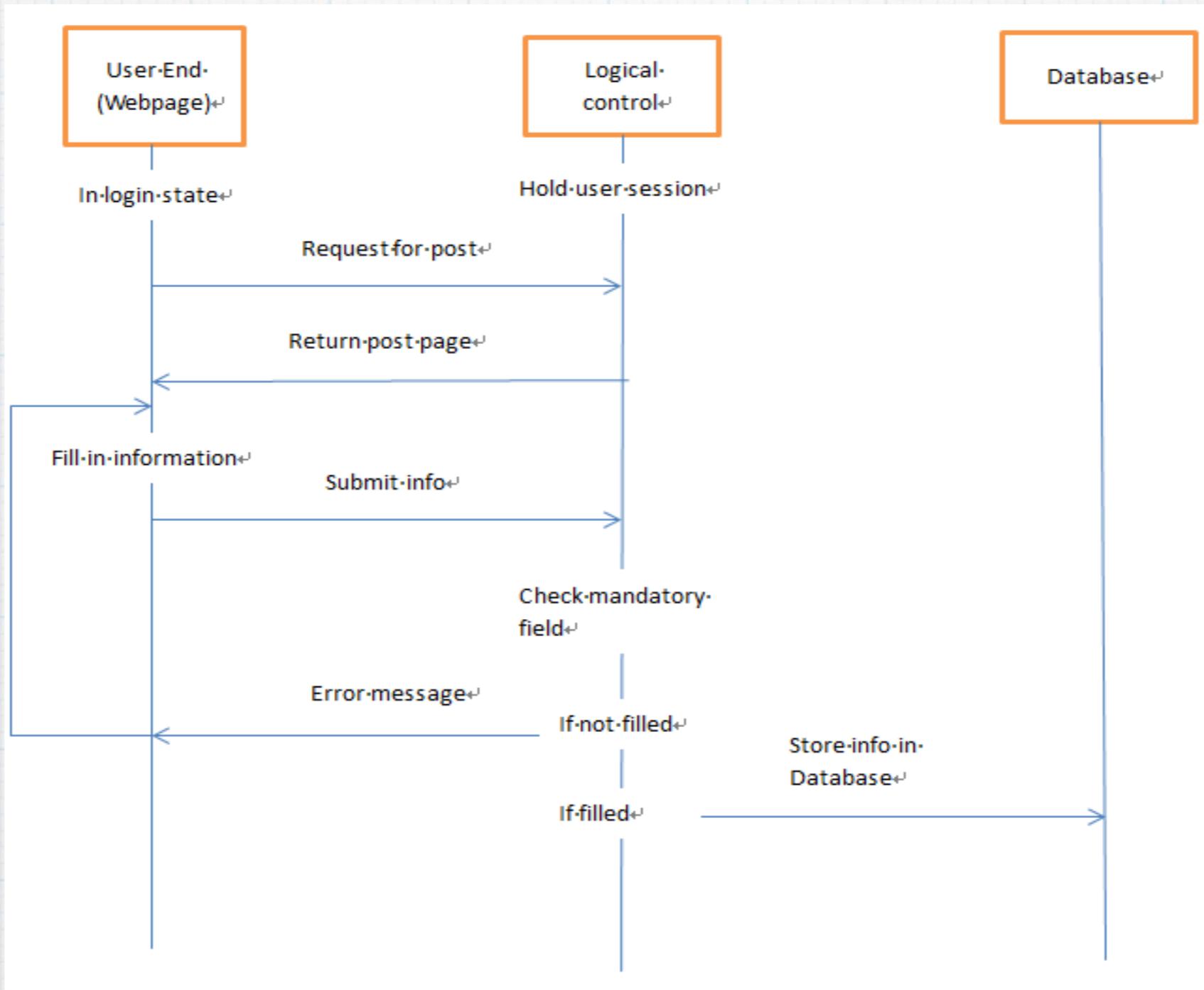
# Register



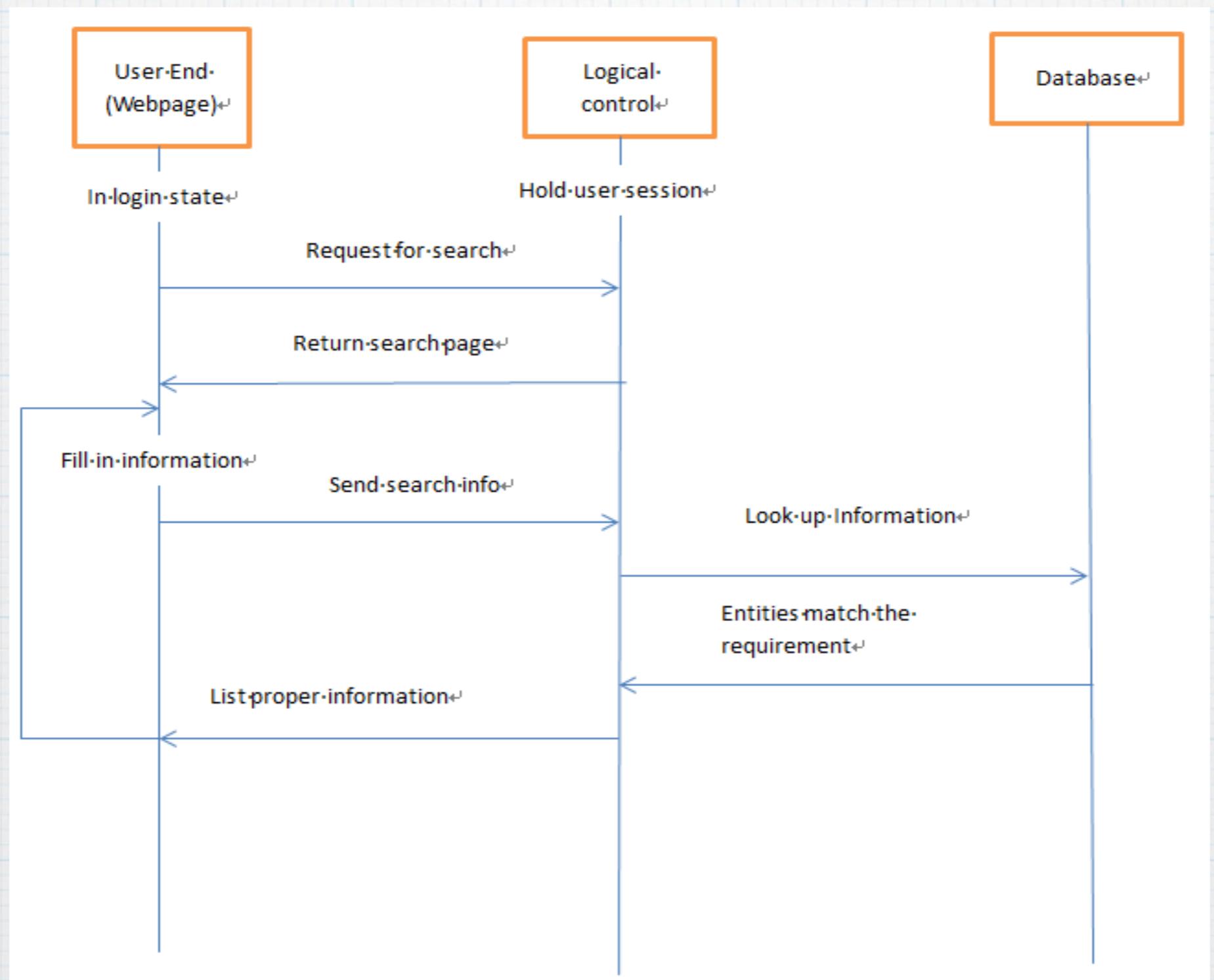
# Login



# Post

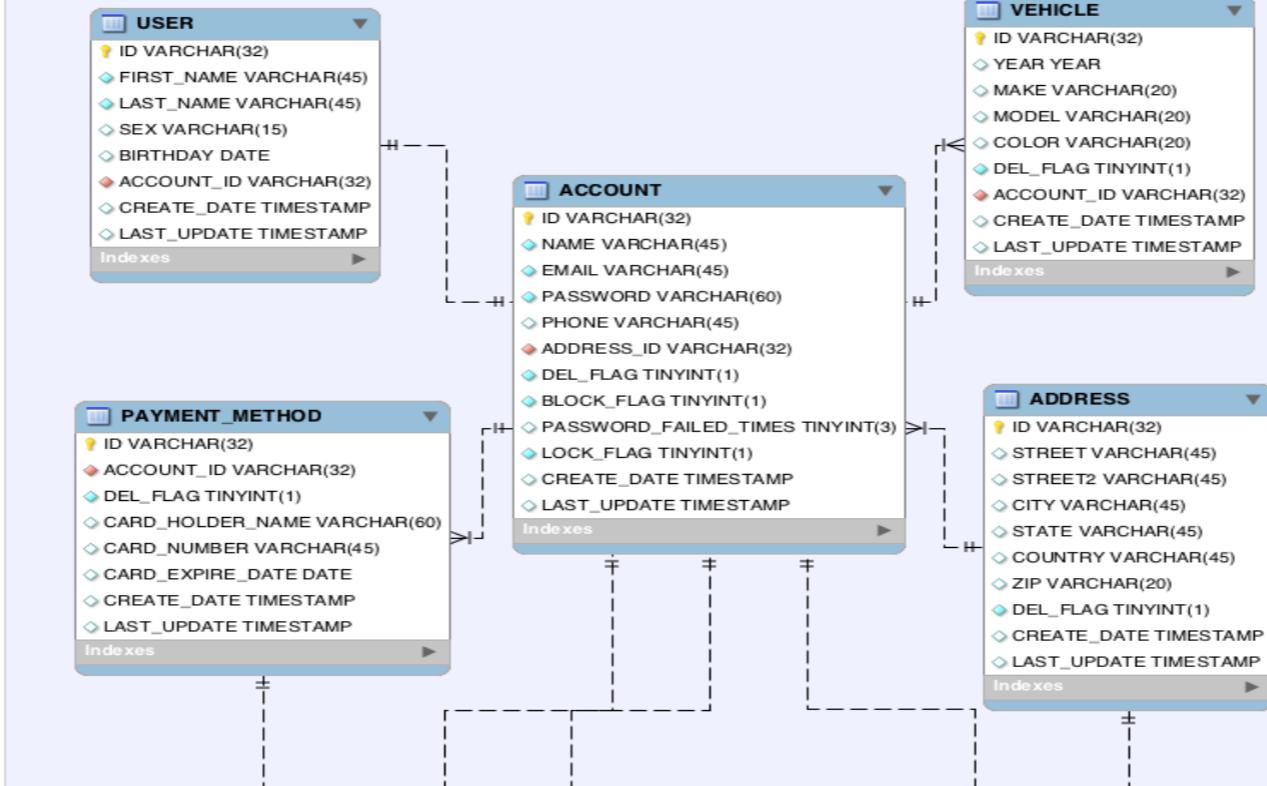


# Search

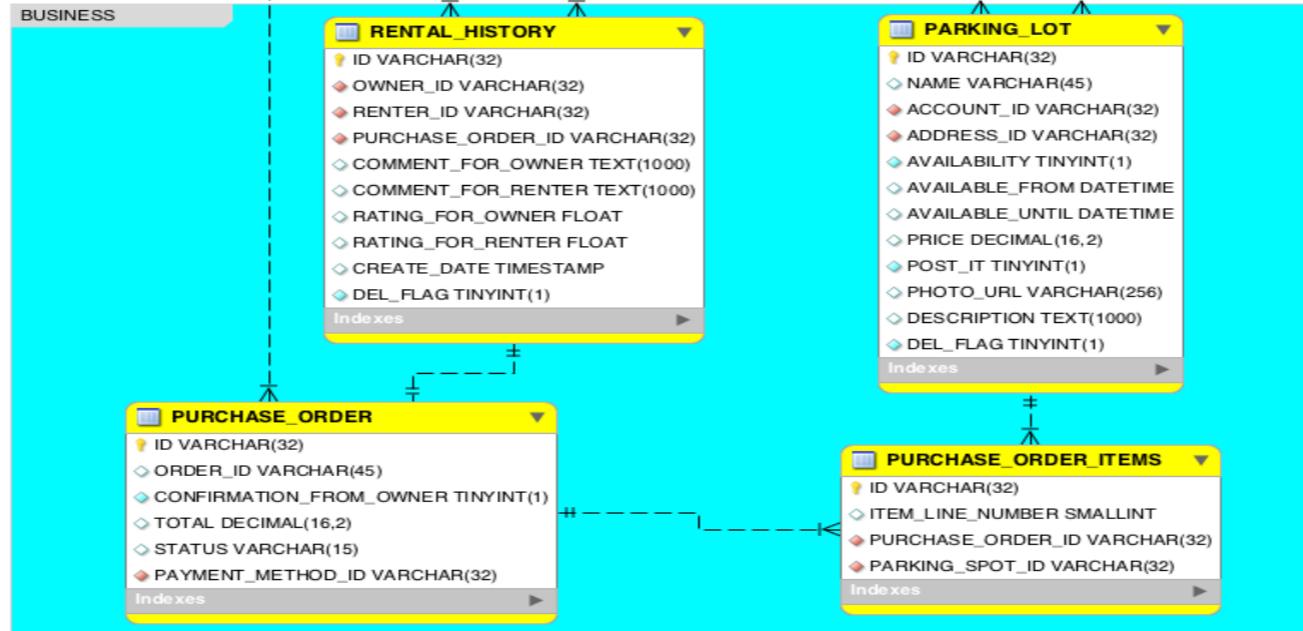


UPark Schema  
Created By: Liming Li, Feiyu  
Created On: 10/3/2014

CUSTOMER\_DATA



BUSINESS



# Quality Assurance

- \* Metrics
- \* Standard
- \* Inspection / review
- \* Testing
- \* Defects management
- \* QA Team: Feiyu Shi, Rui Li

# Metrics

- \* MProduct metrics
  - \* LOC: 14827
  - \* #defects per KLOC: 0.27
- \* Process Metrics
  - \* Project effort: 785 man-hours
  - \* Productivity: around 80 man-hours per main functionality
- \* More on next page...

# Metrics

Metric	Value
+ Abstractness	18.7%
+ Average Block Depth	0.84
+ Average Cyclomatic Complexity	1.41
+ Average Lines Of Code Per Method	7.95
+ Average Number of Constructors Per Type	0.00
+ Average Number of Fields Per Type	2.03
+ Average Number of Methods Per Type	4.25
+ Average Number of Parameters	0.62
+ Comments Ratio	1.8%
+ Efferent Couplings	30
+ Lines of Code	1,486
+ Number of Characters	59,245
+ Number of Comments	27
Number of Constructors	0
+ Number of Fields	69
+ Number of Lines	1,883
+ Number of Methods	136
Number of Packages	14
+ Number of Semicolons	930
+ Number of Types	32
+ Weighted Methods	193

Generated by Google CodePro

# Standard

- \* Meeting and Friday Coding Time
- \* Documentation: project plan, software design, requirement, weekly report, others on Git, database, spring framework, coding style, etc
- \* Coding style: checkstyle plugin for eclipse

# Coding Style

# Before

## After

The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** Shows the title "Java - upark/src/main/java/bu/edu/upark/controllers/PostController.java - Eclipse -".
- Left Side (Package Explorer):** Displays the project structure under "src/main/java".
  - bu.edu.upark.controllers:** Contains files: LoginController.java, LogoutController.java, PostController.java (selected), RegisterController.java, SearchController.java, SessionController.java, StartController.java.
  - bu.edu.upark.entities:** Contains files: ParkingInfo.java, ParkingInfoList.java, UserAccount.java, UserInput.java.
  - bu.edu.upark.repositories:** Contains files: ParkingInfoDAO.java, ParkingInfoDAOImpl.java, UserAccountDAO.java, UserAccountDAOImpl.java.
  - bu.edu.upark.services:** Contains files: LoginService.java, LoginServiceImpl.java, PostService.java, PostServiceImpl.java, RegisterService.java, RegisterServiceImpl.java, SearchService.java.
  - bu.edu.upark.utils:** Contains files: GoogleMapUtils.java, SearchUtils.java, SearchUtilsTest.java.
  - Test:** Contains file: log4j.properties.
- Right Side (Editor):** Shows the content of PostController.java.

```
1 package bu.edu.upark.controllers;
2
3
4 import java.util.List;
5
6 import bu.edu.upark.entities.*;
7
8 import javax.servlet.http.HttpServletRequest;
9
10
11
12
13
14 import javax.servlet.http.HttpSession;
15
16 import bu.edu.upark.services.*;
17 import net.sf.json.JSONObject;
18
19 import org.hibernate.Session;
20 import org.hibernate.SessionFactory;
21 import org.hibernate.cfg.AnnotationConfiguration;
22 import org.hibernate.cfg.Configuration;
23 import org.springframework.beans.factory.annotation.Autowired;
24 import org.springframework.context.annotation.ComponentScan;
25 import org.springframework.stereotype.Controller;
26 import org.springframework.ui.ModelMap;
27 import org.springframework.web.bind.annotation.PathVariable;
28 import org.springframework.web.bind.annotation.RequestBody;
29 import org.springframework.web.bind.annotation.RequestMapping;
```
- Bottom Bar:** Shows tabs for Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying output from Javadoc Generation:

```
<terminated> Javadoc Generation
Generating /Users/fangh/Desktop/java api/allclasses-frame.html...
Generating /Users/fangh/Desktop/java api/allclasses-noframe.html...
Generating /Users/fangh/Desktop/java api/index.html...
Generating /Users/fangh/Desktop/java api/help-doc.html...
2 errors
2 warnings
```

The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** Shows the title "Java - upark/src/main/java/bu/edu/upark/utils/SearchUtils.java - Eclipse - /Users/".
- Left Sidebar (Package Explorer):** Displays the project structure with several packages and their contents. A red box highlights the entire sidebar area.
- Right Side (Editor Area):** Shows the Java code for `SearchController.java`. The code includes methods for getting formatted addresses and southwest bounds from JSON objects. A red box highlights the code area.
- Bottom Right (Tooltips):** A tooltip for the `java.lang.NullPointerException` class is displayed, stating: "Thrown when an application attempts to use null in a context where an object is required." It lists several causes:
  - Calling the instance method of a null object.
  - Accessing or modifying the field of a null object.
  - Taking the length of null as if it were an array.
  - Accessing or modifying the slots of null as if it were a throwable value.
- Bottom Bottom (Console/Log):** Shows the terminal output of Tomcat 7.0 Server at localhost, indicating successful deployment and startup of the application.

# Inspection/Review

- \* MCode review: developers review each other's
- \* Story Review/Test: once each functionality's done, reviewed and tested by QAs
- \* Iteration Review: peer reviewing each other's performance

# Testing

- \* Unit Test

- \*JUnit, by developer

- \* Integration Test

- \*Grouped functionalities (Log in/out, search / display, registration/ log in, etc)

- \*Selenium, manual

- \*Test cases

# Testing

- \* Regression Test

- \*Test after each iteration and before deployment

- \* Acceptance Test

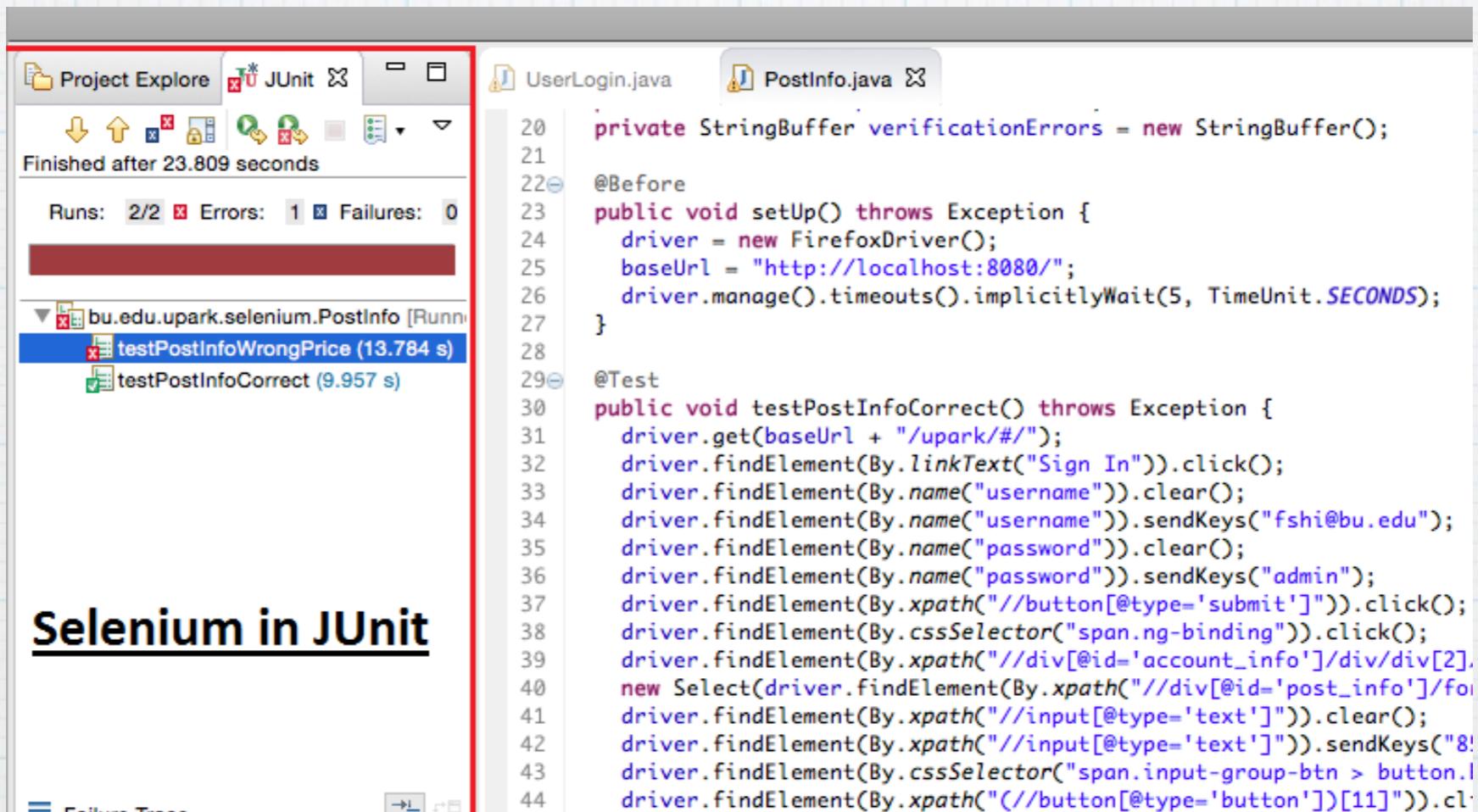
- \*Final round before “release” to check if requirements are met (last minute features)

# Testing

The screenshot shows a Java IDE interface with a red border around the left panel. In the top left, it says "Runs: 6/6 Errors: 0 Failures: 0". Below that is a green progress bar. Underneath the progress bar is a tree view of test cases under "bu.edu.upark.utils.SearchUtilsTest": "getZipcodeTest (0.001 s)", "getFormattedAddressTest (0.000 s)", "getSouthwestTest (0.000 s)", "getNortheastTest (0.000 s)", "getNewSouthWestTest (0.000 s)", and "getNewNorthEestTest (0.000 s)". At the bottom left, there's a "Failure Trace" button and a "JUnit Test" button.

```
4 import net.sf.json.JSONObject;
5
6
7 import org.junit.After;
8 import org.junit.AfterClass;
9 import org.junit.Before;
10 import org.junit.BeforeClass;
11 import org.junit.Test;
12
13 public class SearchUtilsTest {
14
15     public static JSONObject data;
16     public static double[] southwest;
17     public static double[] northeast;
18
19
20     @BeforeClass
21     public static void before() {
22         System.out.println("global");
23         String path = "./src/test.json";
24         String str = SearchUtils.ReadFile(path);
25         data = JSONObject.fromObject(str);
26         southwest = SearchUtils.getSouthWest(data);
27         northeast = SearchUtils.getNorthEast(data);
28     }
29
30     @AfterClass
31     public static void after() {
32         System.out.println("global desotry");
33     }
34 }
```

# Testing



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "bu.edu.upark.selenium.PostInfo".
- JUnit View:** Displays test results:
  - Runs: 2/2
  - Errors: 1
  - Failures: 0A progress bar indicates the total time taken: **Finished after 23.809 seconds**.
- UserLogin.java:** An empty Java file.
- PostInfo.java:** A Java file containing Selenium test code. The code uses WebDriver to interact with a Firefox browser, navigate to a sign-in page, enter credentials, and perform a dropdown selection. It then interacts with a form element and a button.

**Selenium in JUnit**

```
private StringBuffer verificationErrors = new StringBuffer();

@Before
public void setUp() throws Exception {
    driver = new FirefoxDriver();
    baseUrl = "http://localhost:8080/";
    driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
}

@Test
public void testPostInfoCorrect() throws Exception {
    driver.get(baseUrl + "/upark/#/");
    driver.findElement(By.linkText("Sign In")).click();
    driver.findElement(By.name("username")).clear();
    driver.findElement(By.name("username")).sendKeys("fshi@bu.edu");
    driver.findElement(By.name("password")).clear();
    driver.findElement(By.name("password")).sendKeys("admin");
    driver.findElement(By.xpath("//button[@type='submit']")).click();
    driver.findElement(By.cssSelector("span.ng-binding")).click();
    driver.findElement(By.xpath("//div[@id='account_info']/div/div[2].new Select(driver.findElement(By.xpath("//div[@id='post_info']/fo
driver.findElement(By.xpath("//input[@type='text']")).clear();
driver.findElement(By.xpath("//input[@type='text']")).sendKeys("8");
driver.findElement(By.cssSelector("span.input-group-btn > button.l
driver.findElement(By.xpath("//button[@type='button'])[11]")).cl
```

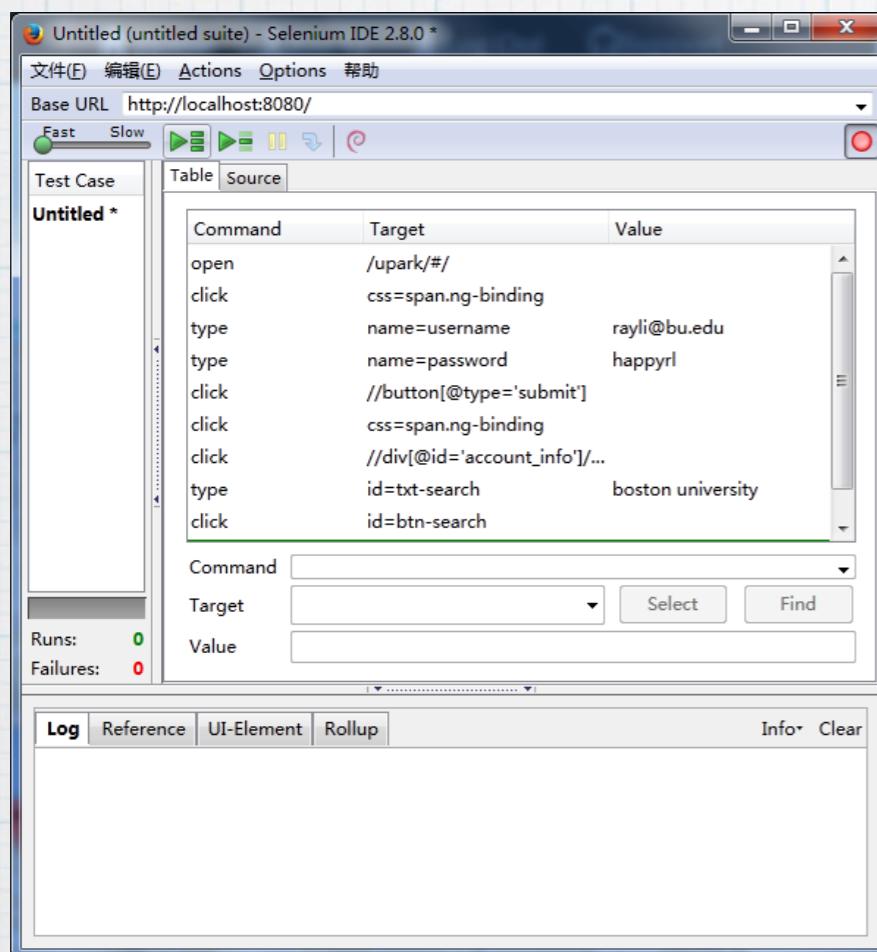
# Test Cases

- \* MLog in/out: 4
  - \* Selenium: 4 (test illegal inputs)
- \* Account registration: 5
  - \* Selenium: 5(test illegal inputs)
- \* Information posting: 4
  - \* Selenium: 4 (test illegal inputs)
- \* Search locations: 2
  - \* Manual: 2 (test info display)
- \* Result: at least 4 bugs reported

# Defect Management

- \* Degree: low, medium, major
- \* Defect follow-up: report to developer immediately; follow-up frequency depends on degree
- \* So far, all defects should be fixed.

# Selenium



The screenshot shows the Eclipse Java EE IDE with the title "Java EE - upark/src/test/java/bu/edu/upark/selenium/UserLogin.java - Eclipse". The Project Explorer view shows the project structure:

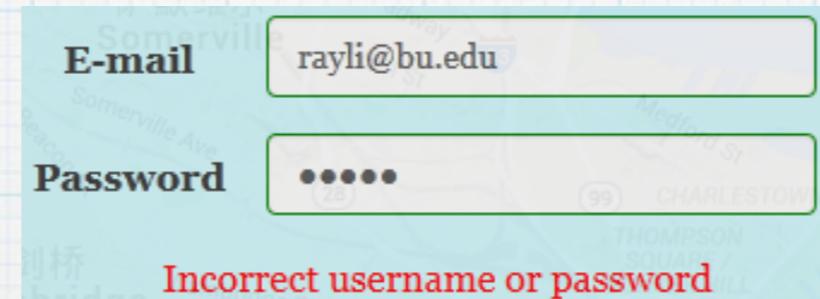
- upark [UPark selenium 12]
  - JAX-WS Web Services
  - Deployment Descriptor: upark
  - Java Resources
    - src/main/java
    - src/main/resources
      - hibernate.cfg.xml
    - src/test/java
      - bu.edu.upark.selenium
        - PostInfo.java
        - PostInfo
        - TestSearch.java
        - TestSearch
        - UserLogin.java
        - UserLogin
        - UserRegistration.java
        - UserRegistration
    - Libraries
    - JavaScript Resources
    - Deployed Resources
  - src
  - target
  - pom.xml
  - Procfile
  - upark.war

The UserLogin.java file contains the following code:

```
1 package bu.edu.upark.selenium;
2
3 import java.util.List;
4
5 /**
6  * Test case to test user login function.
7  * @author Feiyu Shi
8  * @since Nov. 21, 2014
9  */
10 public class UserLogin {
11     private WebDriver driver;
12     private String baseUrl;
13     private boolean acceptNextAlert = true;
14     private StringBuffer verificationErrors = new StringBuffer();
15
16     @Before
17     public void setUp() throws Exception {
18         driver = new FirefoxDriver();
19         baseUrl = "http://localhost:8080";
20         driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
21     }
22
23     @Test
24     public void testUserLoginCorrect() throws Exception {
25         driver.get(baseUrl + "/upark/#/");
26         driver.findElement(By.cssSelector("span.ng-binding")).click();
27         driver.findElement(By.name("username")).clear();
28         driver.findElement(By.name("username")).sendKeys("fshi@bu.edu");
29         driver.findElement(By.name("password")).clear();
30         driver.findElement(By.name("password")).sendKeys("admin");
31         driver.findElement(By.xpath("//button[@type='submit']")).click();
32         List<WebElement> list = driver.findElements(By.xpath(
33             "//[contains(text(),'" + "Incorrect username or password" + "')])");
34         assertFalse(list.size() > 0);
35     }
36
37     @Test
38     public void testUserLoginWrongPassword() throws Exception {
39         driver.get(baseUrl + "/upark/#/");
40     }
41 }
```

Selenium IDE exported as Junit 4

# Verification



```
List<WebElement> list = driver.findElements(By.xpath(  
        "//*[contains(text(),'" + "Incorrect username or password" + "')])");  
assertFalse(list.size() > 0);
```

Video of Automatic Test

# Risk Management

# General Cause of Risks

- \* Lack of Information
- \* Lack of Control
- \* Lack of Time

# An example for a failure

By the end of Iteration2, we were behind course schedule

We should set up most develop and test tasks

# Consider

Risk Identification

- \* Scope and Quality
- \* Resources
- \* Schedule

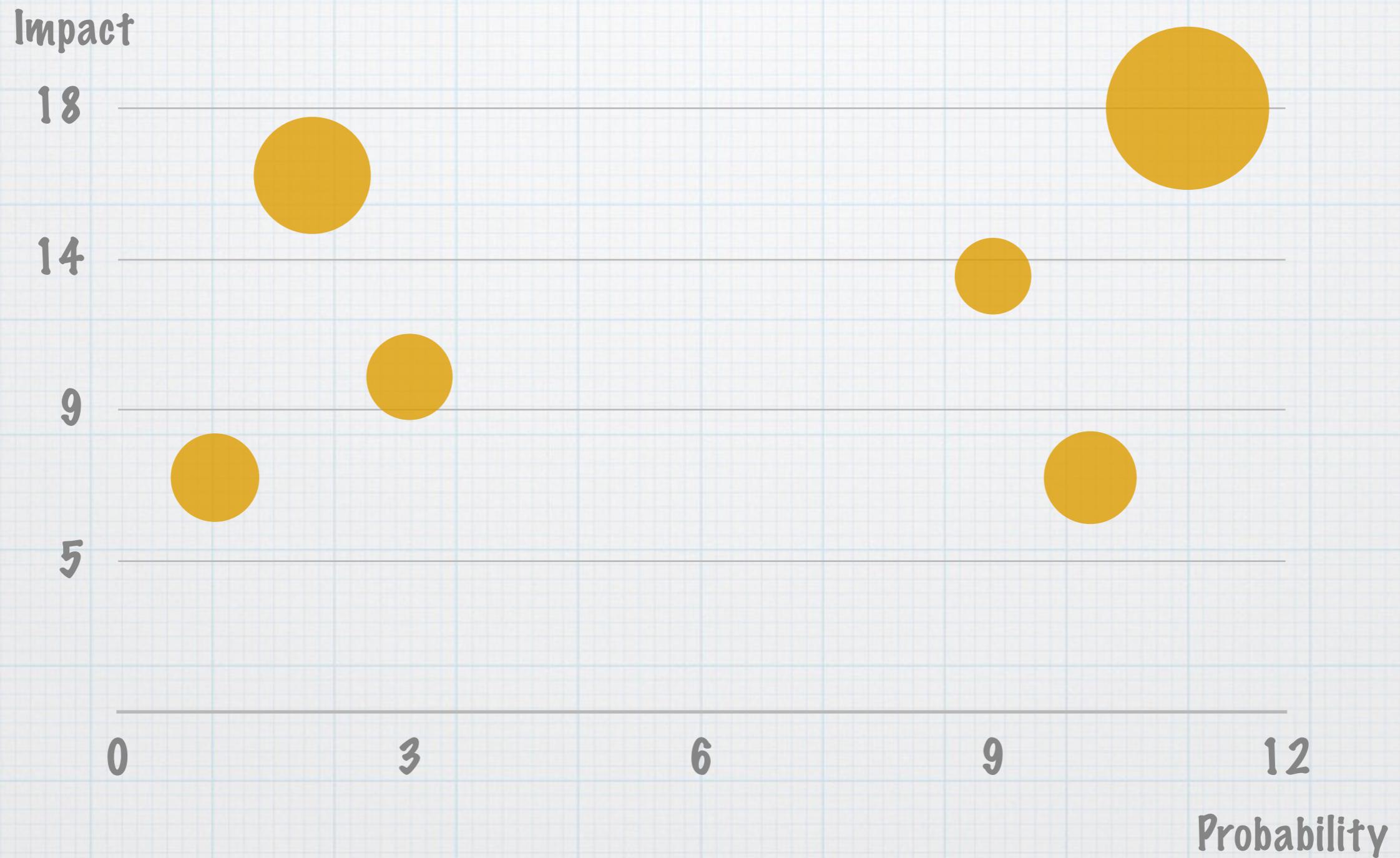
# Scope and Quality Risks Remedies

	Remedies
Unrealistic goals	do research, estimate, communicate
Complexity of task	Decompose complex tasks into smaller tasks
New tools and techniques	spend time and early use
..."	"..."

# Schedule Risks Remedies

	Remedies
Delays in critical path	Most experienced team members for these tasks
Complexity of task	Early Feedback

# Risk Matrix



**DEMO**