

**SPORTS MANAGEMENT**

By Implementing Agile Methodology and refactoring

**PRT 453** [**PROCESS DEVELOPMENT METHODOLOGIES**](https://online.cdu.edu.au/webapps/blackboard/execute/announcement?method=search&context=course_entry&course_id=_41604_1&handle=announcements_entry&mode=view)

**Product Owner**

Mr. Narendra Vellela

**Team**

Ms. Kanchan Kaushal

Mr. Muhammad Ali Malik

Mr. Suraj Desai

Table of Contents

[DETAILS of Project: 3](#_Toc513196192)

[Sprint Team Members: 3](#_Toc513196193)

[Product Backlog 4](#_Toc513196194)

[Sprint -1 4](#_Toc513196195)

[Finished User Stories Details: 5](#_Toc513196196)

[Details: 6](#_Toc513196197)

[Class AccountController 6](#_Toc513196198)

[In events Controllers: 7](#_Toc513196199)

[Sprint 2: 9](#_Toc513196200)

[Planning 9](#_Toc513196201)

[2.1 User Story: 9](#_Toc513196202)

[Class Diagram Before/After : 10](#_Toc513196203)

[Risk Assessment of the Changes: 10](#_Toc513196204)

[Test Analysis 10](#_Toc513196205)

[2.2 User Story 10](#_Toc513196206)

[Class Diagram Before/After : 11](#_Toc513196207)

[Risk Assessment of the Changes: 11](#_Toc513196208)

[Test Analysis 11](#_Toc513196209)

[2.3 User Story 11](#_Toc513196210)

[Class Diagram Before/After : 11](#_Toc513196211)

[Risk Assessment of the Changes: 12](#_Toc513196212)

[Sprint 4: 12](#_Toc513196213)

[Planning: 12](#_Toc513196214)

# DETAILS of Project:

Project Name: Sports Management

Sprint End Date: April 19, April 2018

Context

Start Date & Day of Sprint: April 13, 2018

Last Day of Sprint: April 19, 2018

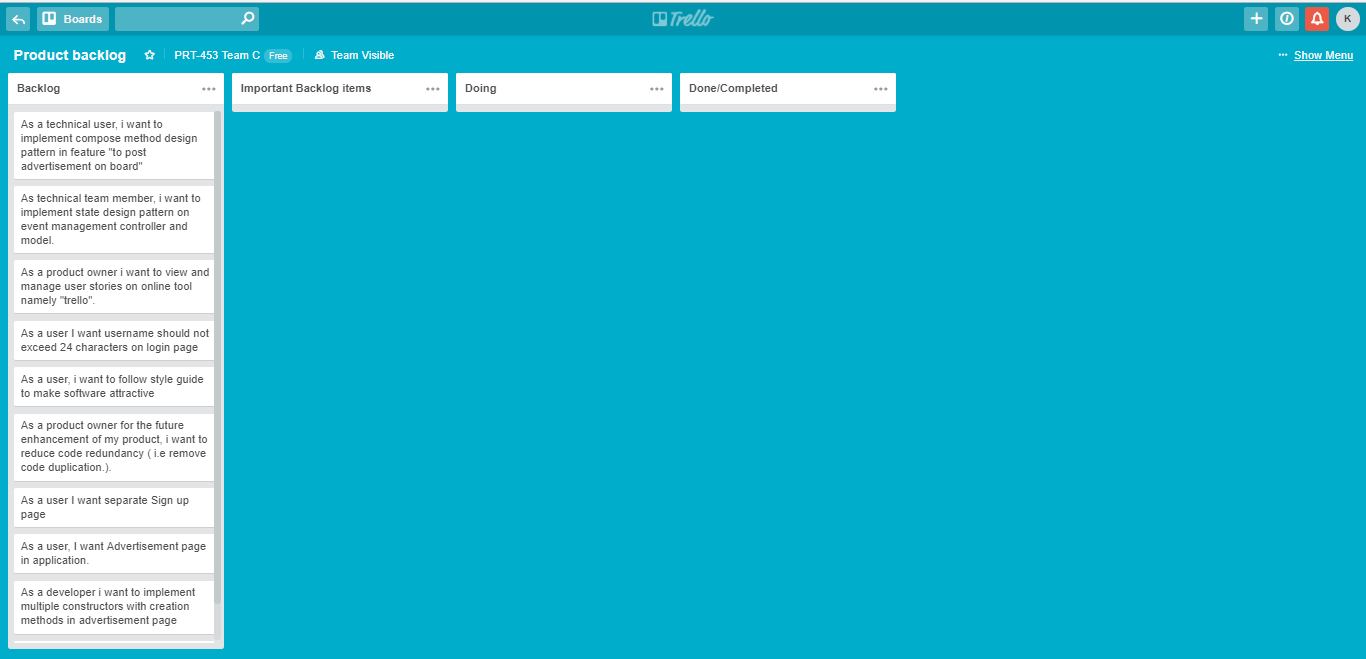
Working Days in Sprint: 7

## Sprint Team Members:

The following listed persons participated in the sprint. Also listed their expected number of days to work and the number of days they did work.

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Name** | **Planned Days to Work** |
| 1 | Kanchan Kaushal | 3 days in a week |
| 2 | Suraj Desai | 3 days in a week |
| 3 | Muhammad Ali Malik | 3 days in a week |

## Product Backlog



# Sprint -1

|  |  |  |  |
| --- | --- | --- | --- |
| Story Number | User Stories | Estimated Hours | Assign to |
| 1 | As a user I want advertisement Page in application by using multiple constructors with creation methods | 8 | Kanchan, Ali, Suraj |
| 2 | As a product owner I want to view and manage user stories on On line Software Trello | 2 | Ali,Suraj |
| 3 | As a product owner I want to reduce code redundancy and remove extra comments from code | 2 | Kanchan, Ali |
| 4 | As a user I want username should not exceed 24 characters on login page | 3 | Suraj, Kanchan |
| 5 | As a user, I want to follow style guide to make software attractive | 4 | Ali,suraj |

Sprint Summary:

In first sprint meeting Team agrees to maintain Project information on online tool called. All team members register into trello. Backlog has been identified and uploaded on Trello. Task prioritisation is performed by all three team members by using poker card method.

Contents and Assessment

Point Planned – 19

Point Earned- 4

|  |  |  |  |
| --- | --- | --- | --- |
| Task Number | Detail | Points | Result |
| 1 | As a user I want advertisement Page in application by using multiple constructors with creation methods | 8 | Not finished |
| 2 | As a product owner I want to view and manage user stories on On line Software Trello | 2 | Finished |
| 3 | As a product owner I want to reduce code redundancy and remove extra comments from code | 2 | Finished |
| 4 | As a user I want username should not exceed 24 characters on login page | 3 | Not finished |
| 5 | As a user, i want to follow style guide to make software attractive | 4 | Not finished |

## Finished User Stories Details:

Task 2

As a product owner I want to view and manage user stories on On line Software Trello.

Detail:

Team members register on Trello and also send link to Product Owner to Join. The product backlog and current status of sprint 1 is been uploaded to trello.

Task 3

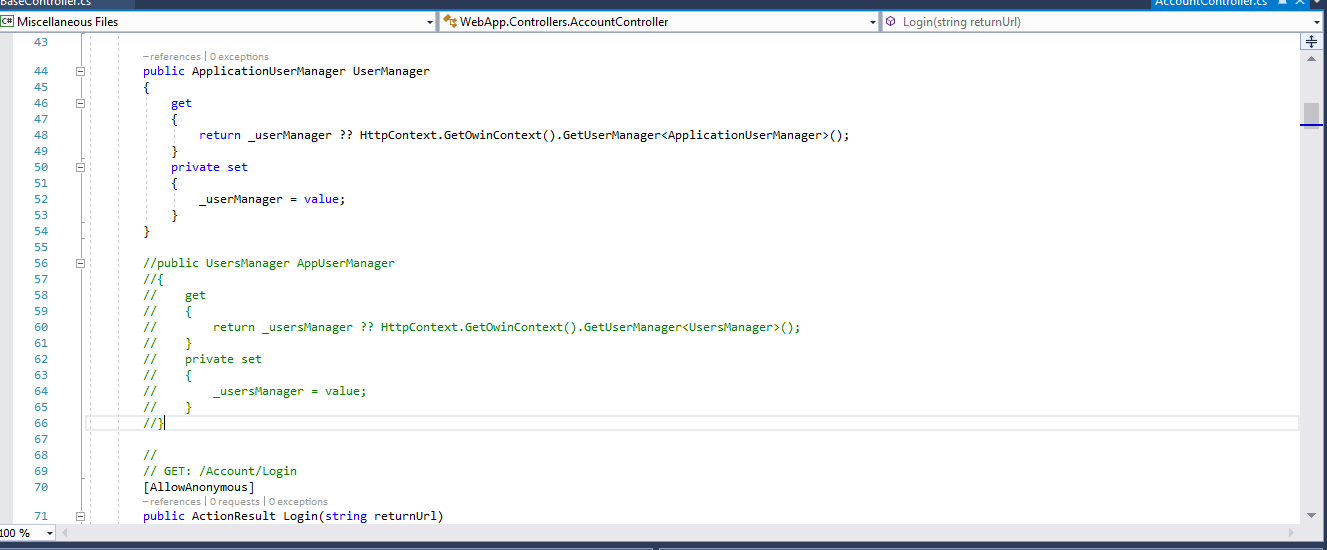
As a product owner I want to reduce code redundancy and remove extra comments from code

Details:

By analysing MVC application it has been found that unnecessary comments in controllers’ classes. Even code itself written in comment form. Two of them are discuss below.

Class AccountController contains code that is no more in use and stay as comment form.

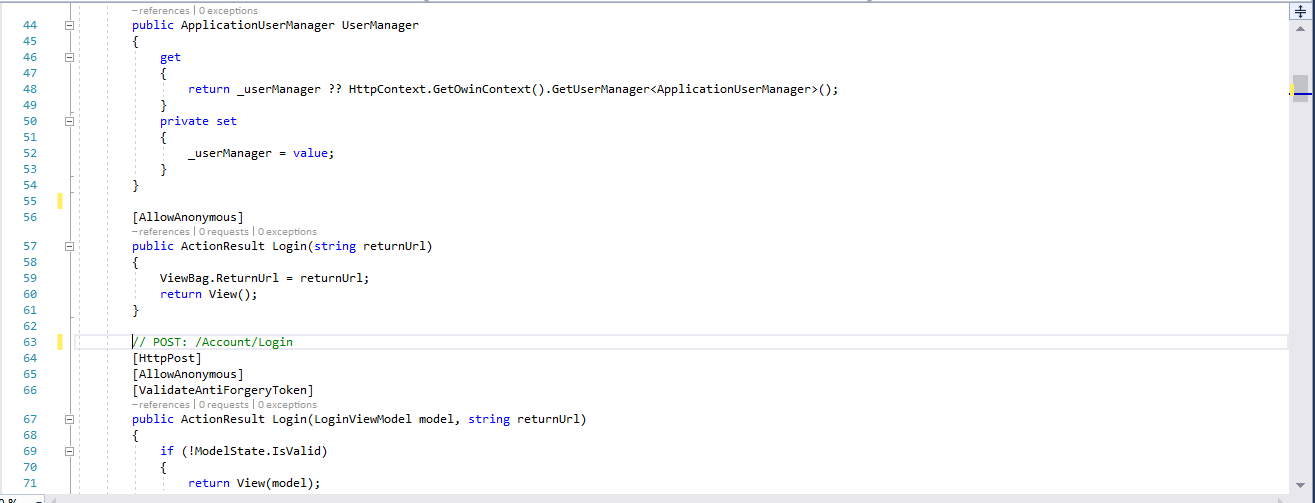
Before Code:



After code:

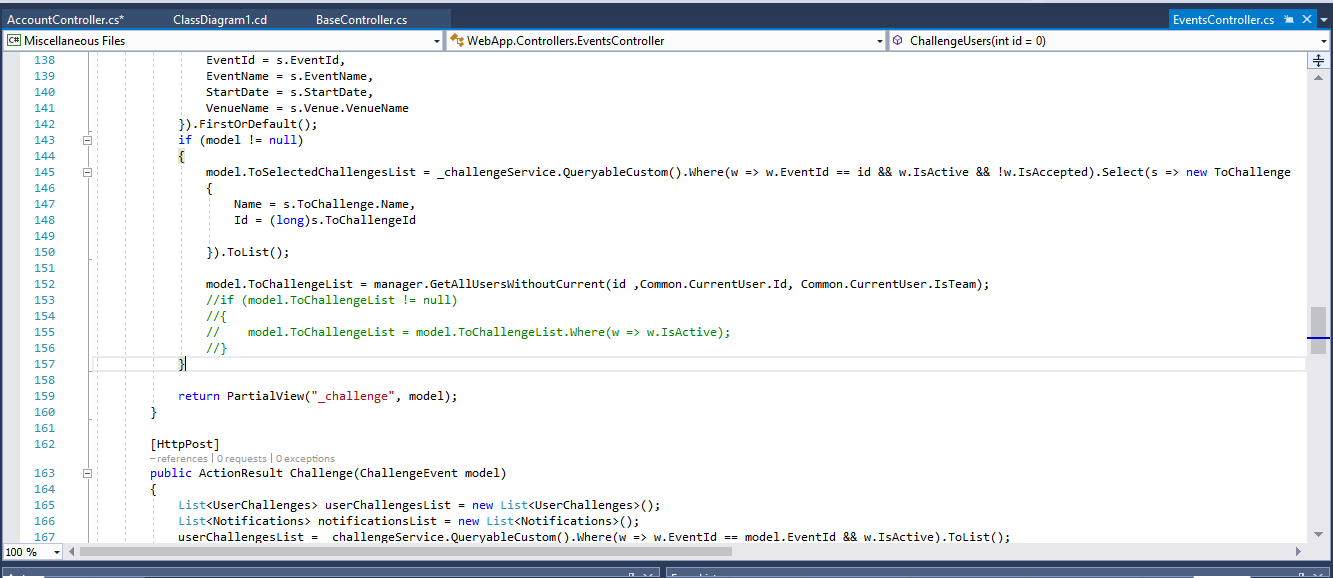
Extra code is removed from comments but keeping necessary comments Like shown in screenshot

// POST: /Account/Login



In events Controllers: also removed code written in comments that is no more in use

Code Before:



Code After:

Removed the code that is in comment.



Class Diagram:

There is no change in class diagram after implementing this task.

Risk Assessment:

Deleted code may be useful in future:

That code was put in comments because client has given the change request. According to developers if that code keep in comments that can be used in future if client can demand to reverse the changes. Then, deleted code has to write again.

But on the other side client is satisfy with new change.

Sprint Review and Accomplishment:

The sprint review was held on April 19, 2018 and attended by all team members but Product owner was not there. Key decisions from the review were:

Complexity of the user stories were not identified correctly

Team have to work more hard to complete the task within

A Sprint Document has been written for record purpose

# Sprint 2:

## Planning

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO. | User Story | How to implement | Assign to |
| 1 | As a User I want to Create the authentication check before generating any ticket so that user cannot make unwanted change in the ticket generation process. | I want to implement Proxy Pattern to fulfill user story | Ali |
| 2 | As a user I want In case any input value in application is incompetent/blank, than help message should be different according to option selected. | Planning to implement it with Prototype Pattern so that can be useful in future. | Kanchan |
| 3 | Create the option to recover forget Password. | I want to implement this with Adapter Pattern | Suraj |

## 2.1 User Story:

In case any input value in application is incompetent/blank, than help message should be different according to option selected.

Pattern Implemented and justification for changes:

To implement user story, Prototype pattern of creational design pattern is best suits.  Because it helps create the deeply copied objects using the existing objects created in the program. It minimizes the need of writing the repetitive code of creating. Then it copies the object state to the new object. The prototype design patter indeed does some complex job of creating the deep copy of the object.

In the Application, I have declared **Abstract basic class** which specifies **Clone ()** method. If any other class needs “polymorphic constructor “, can derive itself from abstract Basic class and implement clone method to create similar copies of the object in **Class Venue Controller** . No need to create a new object in the standard way by using “new” keyword.

In Future if application will be extended, this abstract basic class will be useful to avoid the repeatability of code and create the same method as in Abstract class.

## Class Diagram Before/After :

Please follow link <https://github.com/Suraj35/prt453>

code snapshot of the skeleton of the classes/methods :

Please follow link <https://github.com/Suraj35/prt453>

## Risk Assessment of the Changes:

The subclass of Basic Abstract class must implement Clone() method. But Implementing Clone () Method when these subclasses have objects that don't support copying.

## Test Analysis

|  |  |
| --- | --- |
| **Test Case number** | TC2.1 |
| **Test case summary** | Add Venue in Venue Arrangement screen |
| **Prerequisite** | Existed user account |
| **Tools used** | Windows 10, Visual Studio 2017 |
| **Steps to stimulate** | Open Venue Arrangement Screen  Go to venue field  Add incompetent venue.  Leave the field blank |
| **Expected Output** | On writing wrong venue, should generate message “Venue does not exist” |
| **Actual Output** | Venue does not exist |
| **Result** | Pass |

## 2.2 User Story

Create the authentication check before generating any ticket so that user cannot make unwanted change in the ticket generation process.

Pattern Implemented and justification for changes:

To implement user story, Proxy pattern of structural design pattern can be useful. In this Application, the Class Ticket Controller is wrapper to Class Abstract Base Controller. The Class Base controller cannot be accessed directly because it is a Abstract Class. The Ticket Controller will check the authentication of user to generate ticket then generate ticket by implementing method from Class Base Controller.

It helps to minimize the code duplicity by creating same method in different classes. It provide the way to add extra authentication in existing feature.

## Class Diagram Before/After :

Please follow link <https://github.com/Suraj35/prt453>

code snapshot of the skeleton of the classes/methods :

Please follow link <https://github.com/Suraj35/prt453>

## Risk Assessment of the Changes:

It is necessary the class inherit Class Abstract Base implement methods of it.

## Test Analysis

|  |  |
| --- | --- |
| **Test Case number** | TC2.2 |
| **Test case summary** | Create ticket in false way |
| **Prerequisite** | User Profile |
| **Tools used** | Windows 10, Visual Studio 2017 |
| **Steps to stimulate** | Login as normal user  Fill the false details to generate ticket  View the error message |
| **Expected Output** | By providing false details to system, should generate error message |
| **Actual Output** | Error Message |
| **Result** | Pass |

## 2.3 User Story

Create the option to recover forget Password.

The best Pattern to implement it, is Adpater Pattern of Structural Pattern. In the application, Class Login ViewModel() is not compatible to newly created Class Forget Viewmodel(), so, Class ForgetPassword () is created as adapter class.

It helps to implement new feature in existing code.

## Class Diagram Before/After :

Please follow link <https://github.com/Suraj35/prt453>

code snapshot of the skeleton of the classes/methods :

Please follow link <https://github.com/Suraj35/prt453>

## Risk Assessment of the Changes:

Sometimes it is hard to override the Adaptee class behaviour.

|  |  |
| --- | --- |
| **Test Case number** | TC2.3 |
| **Test case summary** | Check the functionality of forget password |
| **Prerequisite** | Existing User Profile |
| **Tools used** | Windows 10, Visual Studio 2017 |
| **Steps to stimulate** | Login as normal user  Click on forget password option. |
| **Expected Output** | Forget Password will generate message to email |
| **Actual Output** | Email Notification |
| **Result** | Fail |

# Sprint 4:

## Planning:

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO. | User Story | How to implement | Assign to |
| 1 | As a User I want to introduce new features: Book ticket to without creating user profile | There already different login authorization existing. I have to create another authorization where user does not need to provide Id and Password.so, I am planning to implement **factory pattern t**o deal with the problem of creating objects without having to specify the exact class of the object that will be created.  Class Diagram: | Kanchan |
| 2. | As a user I want view Advertisement without having any user authorisation. | For that I have to provide the advertisement page link on the login page. I will implement this with Adapter Pattern.  Class Diagram: | Ali |
| 2 | As a user I want different Login Authorization: Captain, Team member, Admin User, | At the moment only one class for two login it will make the class complex if implement all the logins in one class.  Builder pattern is best suit for this.  Sample Class Diagram: | Kanchan |
| 3 | As a user I want make the Application response in 3 second now and in future after any extention. | This is not a bigger project. But in future if it will extended. To keep in mind I am implementing **singleton pattern** that restricts the instantiation of a **class** to one object. That will also makes the Application fast responsive.  Sample Class Diagram: | Suraj |

User Story

User want to introduce new features in find user by ID and Delete Existing Password.

Pattern Implemented and justification for changes:

User wants to introduce different functionality so, I have implemented **factory pattern in application t**o deal with the problem of creating objects without having to specify the exact class of the object that will be created. I have created new GeneralUsers class, whose object is created in Usermanager class by using of Usercreator class. It will help to fulfil users’ demand to add more feature to existed functions in application.

Class Diagram Before/After :

Please follow link <https://github.com/Suraj35/prt453/blob/master/Factory%20Pattern>

Code snapshot of the skeleton of the classes/methods :

Please follow link [https://github.com/Suraj35/prt453/blob/master/Factory%20Pattern](https://github.com/Suraj35/prt453/blob/master/Factory%20Pattern/before/before%20Pattern%20class.PNG)

Risk Assessment of the Changes:

It creates more code in the application. In our application 2 new classes are being introduced.

|  |  |
| --- | --- |
| **Test Case number** | TC3.1 |
| **Test case summary** | Check functionality of search user by ID and Delete Password button |
| **Prerequisite** | Existing User Profile |
| **Tools used** | Windows 10, Visual Studio 2017 |
| **Steps to stimulate** | Go to login page  Click on button “search User”  Write ID  Click OK button  Login as any User  Go to personal profile  Go to password  Delete the password  Add new Password |
| **Expected Output** | Search user by User ID  Delete Existing Password |
| **Actual Output** | 1. Able to search by User ID  2.Successfully delete password |
| **Result** | Pass |

User Story

User want different Login Authorization for Captain, Team member and Admin User

Pattern Implemented and justification for changes:

At the moment only one class user controller is for two logins. If the third login team captain will introduce into class user controller. It will make the class more complex. So, by implementing Builder pattern, two more classes created name as class Abstract Usertype and Class ConcreteUser1. It reduces the complexity of class. In future it will also easy to introduce more logins.

Class Diagram Before/After :

Please follow link <https://github.com/Suraj35/prt453/tree/master/builder%20Pattern>

Code snapshot of the skeleton of the classes/methods :

Please follow link <https://github.com/Suraj35/prt453/tree/master/builder%20Pattern>

Risk Assessment of the Changes:

It creates more code in the application. In our application 2 new classes are being introduced.

|  |  |
| --- | --- |
| **Test Case number** | TC3.2 |
| **Test case summary** | Check login authorization as: Team captain |
| **Prerequisite** | Existing User Profile |
| **Tools used** | Windows 10, Visual Studio 2017 |
| **Steps to stimulate** | Login as Team captain  Initiate an event, send requests for an event.  Login as Normal User  Accepts the invitation  Initiate new event |
| **Expected Output** | Only team captain will able to initiate event  Normal user cannot create new event. But can only accept/decline the invitations. |
| **Actual Output** | Normal user able to also create an event. |
| **Result** | Fail |

Sprint 4:

Sprint 4:

Planning:

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO. | User Story | How to implement | Assign to |
| 1 | As a User I want to add more section in Advertisement Page | At the moment, only one class is handling advertisement. But user wants to introduce different types of advertisements. To fulfil user’s demand, I am planning to implement **factory pattern t**o deal with the problem of creating objects without having to specify the exact class of the object that will be created.  Class Diagram: | Ali |
| 2 | As a user I want different Login Authorization: Captain, Team member, Admin User, | At the moment only one class for two login it will make the class complex if implement all the logins in one class.  Builder pattern is best suit for this.  Sample Class Diagram: | Kanchan |
| 3 | As a user I want make the Application response in 3 second now and in future after any extention. | This is not a bigger project. But in future if it will extended. To keep in mind I am implementing **singleton pattern** that restricts the instantiation of a **class** to one object. That will also makes the Application fast responsive.  Sample Class Diagram: | Suraj |