Agile User Story

**VR Arcade Booker**

Kendall Fowler for Omar Bakraky

CIS2232 Intercession 20192020AYR

2020-05-18

**Table of Contents**

[Core Business Need 3](#_Toc43913337)

[User Stories 3](#_Toc43913338)

[User Story #1: Add Bookings 3](#_Toc43913339)

[User Story #2: List Upcoming Bookings 3](#_Toc43913340)

[User Story #3: Edit and Manipulate Bookings 3](#_Toc43913341)

[User Story #4: Delete Bookings 3](#_Toc43913342)

[User Story #5: Find/Report on Bookings 3](#_Toc43913343)

[User Story#3: Save Bookings to log file 3](#_Toc43913344)

[VR Bookings Attributes: 3](#_Toc43913345)

[Database Table Design: 4](#_Toc43913346)

[What is an Agile User Story? 5](#_Toc43913347)

[The Characteristics of a User Story 5](#_Toc43913348)

[What to Avoid When Writing User Stories 6](#_Toc43913349)

[Examples of User Stories: Good and Bad 7](#_Toc43913350)

[User Story with Acceptance Criteria and Acceptance Testing 8](#_Toc43913351)

# 

# Core Business Need

This project will serve as a basic booking service for Virtual Reality Arcade. The user will be able to add a room booking and share with friends.

# User Stories

## User Story #1: Add Bookings

I want to be able to add details of a booking to the Virtual Reality booking system.

## User Story #2: List Upcoming Bookings

I want to be able to view bookings and see who has what booked.

## User Story #3: Edit and Manipulate Bookings

I want to be able to change bookings by editing them.

## User Story #4: Delete Bookings

I want to be able to remove bookings from the system.

## User Story #5: Find/Report on Bookings

I want to be able to find bookings and/or pull a report for them based on booking date.

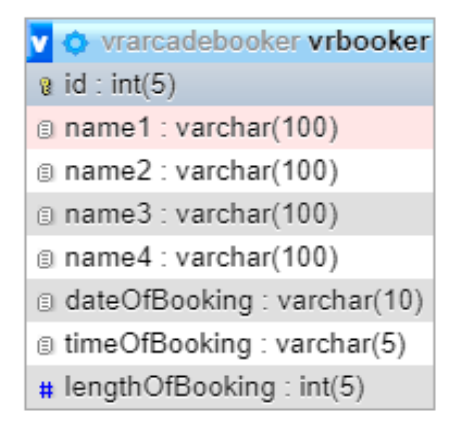
## User Story#3: Save Bookings to log file

The application must save list of bookings to a file as an archive.

# VR Bookings Attributes:

* id
* name1
* name2
* name3
* name4
* dateOfBooking
* timeOfBooking
* lengthOfBooking

# Database Table Design:



# What is an Agile User Story?

A user story is an agile project management tool used to define product or system functionality and the associated benefit of the functionality. In an Agile environment, projects are commonly comprised of a large number of user stories representing various levels of system/product user. The user story describes the type of system/product user, what functionality they want, and why they want it (or why it’s beneficial). The format of a user story is:

As a <*type of user*>, I want to <*feature/function*> so that <*reason/benefit*>.

The purpose of the user story is to encourage collaboration among the project team for each defined function of the system or product being developed. Agile project management places emphasis on collaboration rather than formal documentation. Because of this, each user story created by the team represents an opportunity for the team to collaborate, decide which user stories will be worked on in the next iteration, further define the functionality (during iteration planning), and accomplish the work.

While user stories represent a simplified approach to defining functionality, the challenge for the project team is developing user stories with the appropriate level of detail.

# The Characteristics of a User Story

All user stories should be developed with the expectation that once completed; the functionality defined in the user story will add value to the final product. If it does not add value to the finished product then it should be avoided.

User Stories should be developed and written collaboratively by the project team. This development must also include input and feedback from the ultimate customer or product owner. Remember that agile methodology emphasizes collaboration over formal documentation. The user stories represent a tool to facilitate team communication and collaboration instead of a formal specification or requirement. Not only does this collaboration encourage all the team to contribute to the project, but only through the eyes of the customer or product owner do we know for certain what does or does not add value.

Finding the appropriate level of detail for user stories is often a challenge for the team. User stories should be general enough to provide a description of the functionality and the benefit while also allowing for innovation and creativity for developing a solution. They should not be so detailed as to lock the team into only one way of accomplishing the solution.

In the creation of their user stories, many agile practitioners subscribe to the INVEST acronym created by Bill Wake which states that user stories should be:

**I**ndependent – user stories should not be sequential or locked into a specific order. The team should be able to develop the user stories in any sequence.

**N**egotiable – user stories should be flexible and without too much detail. Details will be added later through team collaboration.

**V**aluable – the user stories should add value to the final product.

**E**stimable – the team must be able to use the user stories to estimate/approximate work.

**S**mall – large or vague user stories are difficult to estimate. User stories should be able to be designed, built, and tested within a single iteration.

**T**estable – the user story should be able to be tested with some type of acceptance criteria or other test (even if it has not yet been defined)

Acceptance criteria, while not formally a part of the user story format, is a crucial component of user stories. As user stories evolve toward iteration planning, the team should collaboratively discuss the acceptance criteria for satisfying each user story. Acceptance criteria are used by the team to understand when the intent of the user story has been met. Additionally, acceptance criteria are often used by the product/system testers to develop the acceptance tests for the user story.

Acceptance tests are developed to test the functionality of each user story. While an acceptance criteria ensures that a certain functionality is included in an iteration, an acceptance test ensures that functionality performs as expected.

# What to Avoid When Writing User Stories

User stories provide an effective tool to encourage communication and collaboration among the team. However, there are several common mistakes that teams make when developing their user stories which often reduce their effectiveness. Some of these mistakes include:

Not including the customer/product owner – the customer/product owner is a critical member of the team in an agile environment. They’re the ultimate authority on what does or does not add value to the product.

Too much detail – user stories should be broad enough to allow for flexibility and a collaborative effort to further define the user story during iteration planning when the team agrees to incorporate it into the next iteration. User stories with too much detail may lock the team into only one way to develop a solution which discourages innovation and creativity.

Lack of acceptance criteria – as user stories evolve, the team must discuss and capture acceptance criteria. Without acceptance criteria, the team will have no ability of knowing when they’ve satisfied the user story. Additionally, a lack of acceptance criteria will result in a lack of acceptance testing which hampers the team’s ability to ensure functionality of the product performs as expected.

# Examples of User Stories: Good and Bad

Below are examples of user stories which include too much detail, not enough detail, and an appropriate amount of detail.

**Too Much Detail:**

As an end user, I want the ability to access the human resource database to generate a staffing report with employee names, dates of birth, social security numbers, addresses, and telephone numbers so that I can periodically update the company’s staffing contact list.

Note that the above example goes into listing details about what the report should include. Including these items in the user story may prevent or discourage the team from including other important content in reports that will be built. This level of detail should be defined later during iteration planning.

**Too Broad:**

As an end user, I want access to the human resource database so that I can periodically update the company’s staffing contact list.

Note that the above example mentions nothing about the ability to generate reports from the human resource database. Only access to the database is mentioned. By leaving out the key information about generating a report, the team might overlook this important functionality and only grant access to the database when the iteration is performed.

**Appropriate Level of Detail:**

As an end user, I want to access the human resource database to generate reports so that I can periodically update the company’s staffing contact list.

The above example ensures that when iteration planning begins, the team can plan for both access and report generating while further defining what information the reports will consist of.

# User Story with Acceptance Criteria and Acceptance Testing

Below is an example of a single user story with both the acceptance criteria and acceptance testing information included.

**User Story:**

As an end user, I want to access the human resource database to generate reports so that I can periodically update the company’s staffing contact list.

**Acceptance Criteria:**

* Ability to gain access to human resource database
* Ability to generate a report which includes
  + Employee names
  + Dates of birth
  + Social security numbers
  + Addresses
  + Telephone numbers
* Ability to use report data to update staffing contact list

**Acceptance Testing:**

* Database is accessible internally but not from outside company firewall
* MS Access report can be generated from database based on query inputs
* MS Access report data can be exported to MS Excel spreadsheet
* Staffing contact list updates contain only most recent data from the database

Sponsor Acceptance

Approved by the Project Sponsor:

Date:

<Project Sponsor>

<Project Sponsor Title>

This free Agile User Story Template is brought to you by [www.ProjectManagementDocs.com](http://www.ProjectManagementDocs.com)