User Stories

Actors

Spectator - Guest of web application who either hasn't logged into account or doesn't own an account. They have restricted access to features offered by application.

Regular User - Client who have an account and are currently logged into account. They have ability to utilize features offered by application.

Administrator - Regular user who has special privileges and assigned special roles. They have authority to act upon a troubled scenario.

Story #1

Name: Account Registration
Actors: Spectator
Triggers/Preconditions: Using features, not logged in and has no existing account
Actions/Postconditions: Display sign up form, create new account
Acceptance Tests: *Username*, presence: true, length: {minimum: 5}, restrictions: {a-z, A-Z, 0-9, -, and _}; *firstname,* presence: true; last*name,* presence: true; *email*, presence: true, restrictions: {a-z, A-Z, 0-9, ., @, and _}, format: {end with @___.__}; *password*, presence: true, length: {minimum: 6}, restrictions: {one each of a-z, A-Z and 0-9}
assert *username* doesn't exist in database
Required Behavior: Display error message for acceptance test failure; new account created for acceptance test success
Relevance: Increase members
Iteration: 1

Account Registration

Given a spectator is browsing web application, when they are trying to access features while currently aren't logged in and has no existing account, then a sign up form appears which permits ability to create a new account.

Story #2

Name: Registration Verification
Actors: Spectator
Triggers/Preconditions: Successful Registration
Actions/Postconditions: Send verification e-mail
Acceptance Tests: -
Required Behavior: -
Relevance: Increase members
Iteration: 1

Registration Verification

Given a spectator is browsing web application, when they successfully register a new account with application, then a verification e-mail is sent to spectator.

Story #3

Name: Account Login
Actors: Spectator
Triggers/Preconditions: Using features, not logged in and has existing account
Actions/Postconditions: Display sign in form, log into account
Acceptance Tests: *Username*, presence: true; *password*, presence: true
assert *e-mail* exists in database
assert *e-mail* and *password* match in database
Required Behavior: Display error message for acceptance test failure; logged into account for acceptance test success
Relevance: Increase productivity
Iteration: 1

Account Login

Given a spectator is browsing web application, when they are trying to access features while currently aren't logged in and has existing account, then a sign in form appears which permits ability to log into account.

Story #4

Name: Account Home Page
Actors: Spectator, regular user
Triggers/Preconditions: Log into account, successful credentials submitted
Actions/Postconditions: Render home page
Acceptance Tests: -
Required Behavior: -
Relevance: Convenience
Iteration: 1

Account Home Page

Given a spectator trying to access their account, when they are logging in and submitted successful credentials, then the home page corresponding to their account is rendered.

Story #5

Name: Account Logout
Actors: Regular User
Triggers/Preconditions: End session
Actions/Postconditions: Logged out
Acceptance Tests: -
Required Behavior: -
Relevance: Security
Iteration: 1

Account Logout

Given a regular user who is currently signed into their account, when they want to end their session, then they are logged out of their account.

Story #6

<u>Name</u>: Account Monitoring
<u>Actors</u>: Administrator
<u>Triggers/Preconditions</u>: Modifying regular users
<u>Actions/Postconditions</u>: Display remove regular users form and add users form, update regular users
<u>Acceptance Tests</u>: *Username*, presence: true, length: {minimum: 5}, restrictions: {a-z, A-Z, 0-9, -, and _}; *firstname,* presence: true; last*name,* presence: true; *email*, presence: true, restrictions: {a-z, A-Z, 0-9, ., @, and _}, format: {end with @___.__}; *password*, presence: true, length: {minimum: 6}, restrictions: {one each of a-z, A-Z and 0-9}
assert *username* doesn't exist in database
<u>Required Behavior</u>: Display error message for acceptance test failure; inform administrator of action change for acceptance test success
<u>Relevance</u>: Convenience, security
<u>Iteration</u>: 1

Account Monitoring

Given an administrator currently logged into their account, when they want to remove or add regular users, then privileges are granted.

Story #7

<u>Name</u>: Secure Password
<u>Actors</u>: Spectator
<u>Triggers/Preconditions</u>: Account registration, successful credentials submitted
<u>Actions/Postconditions</u>: Create new account
<u>Acceptance Tests</u>: -
<u>Required Behavior</u>: -
<u>Relevance</u>: Security
<u>Iteration</u>: 1

Secure Password

Given a spectator is browsing web application, when they successfully register an account, then hashed and salted password is placed in database.

Velocity

Convention

User stories can be used to measure progress. This is done by assigning integer values called Story Points to user stories in accordance to their level of difficulty. By accumulating total story points we are able to dictate rate of delivery called Velocity.

Story Points - 1 means straight forward
2 means intermediate
3 means complex

Velocity - Average number of story points per iteration.

Velocity of Iteration #1

<u>Stories Implemented</u>

#1	Account Registration	Story Points: 2
#2	Registration Verification	Story Points: 1
#3	Account Login	Story Points: 1
#4	Account Home Page	Story Points: 1
#5	Account Logout	Story Points: 1
#6	Account Monitoring	Story Points: 2
#7	Secure Password	Story Points: 1

DineRoulette

Velocity of iteration #1 is 9.