* + for each user story and business rule indicates the extent to which the code is written, everything that can break related to the code is unit tested, and you have verified via the user interface that the story can be carried out or that the business rule is enforced in a way that the user can verify.

**User Stories Enforced**

1. As a Park Manager I want to submit a new job request.
   * As a Park Manager you will be asked whether you want to add a Job through a method in the ParkManager class called addJob(Park). The method will require a Park and will return a boolean that will confirm whether or not the job has been added. The method addJob(Park) is tested in the class ParkManagerTest in the method testGetAllUpcomingJobs().
2. As a Volunteer I want to view upcoming jobs I can sign up for.
   * As a Volunteer you are given the option to view many jobs that have all been checked before hand through a method in the Volunteer class called canSignUpForJob(Job). This method will check each job to make sure they are eligible for signup, therefore the volunteer can choose anyone of them to sign up for. The method canSignUpForJob(Job) is tested in the class VolunteerTest by the method testCanSignUpForJob().
3. As a Volunteer I want to volunteer for a job.
   * As a Volunteer you can volunteer for a job through a method in the Volunteer class called addJob(Job). This method will require the volunteer to give a job, and then it will proceed to add the job to their personal job list and return a boolean that will confirm whether the job has been added. The method addJob(Job) is tested in the class VolunteerTest by the method testAddJobFails().
4. As a Volunteer I want to view the jobs I am signed up for.
   * As a Volunteer you can view your upcoming jobs through a method in the Volunteer class called getJob(), which returns a list of jobs that you have volunteered for. Testing for getJob() happens in the class VolunteerTest.
5. As a Park Manager I want to view upcoming jobs in the parks that I manage.
   * As a Park Manager you can view upcoming jobs after entering the Park that you manage. Then you can access the Park class which has a method called getJobs(), this method will return a list of upcoming jobs that are for the Park that you mange. Testing for getJobs() was not necessary.
6. As a Park Manager I want to view the Volunteers for a job in the parks that I manage.
   * As a Park Manager you can view the list of volunteers for each of your jobs by inputting the park you manage, the job you want to view the list for and receiving the list of volunteers. The Job class has a method called getAllVolunteers(), this method will return a list of volunteers depending on the job you want the list for. Testing for getAllVolunteers() was not necessary.
7. As an Administrator, I want to search volunteers by last name.
   * As an Administrator you can search volunteers by last name through a method in the the Handler class called getVolunteerByLastName(String), by entering a last name which the method will take as a parameter. Testing for getVolunteerByLastName() happened in the testGetVolunteerByLastName() method in the HandlerTest class.

**Business Rules Enforced**

1. A job may not be added if the total number of pending jobs is currently 30.
   * The Handler class has a global variable that is set to 30 and is used in the method isPendingJobsFull() which checks to make sure that the number of pending jobs does not exceed 30; isPendingJobsFull() returns a boolean that will tell the Park Manager whether they can or cannot add the job. The method isPendingJobsFull() is tested by the method testIsPendingJobsFull() in the class HandlerTest.
2. A job may not be added if the total number of pending jobs during that week (3 days on either side of the job days) is currently 5. In other words, during any consecutive 7 day period there can be no more than 5 jobs.
   * The Handler class has a global variable set to help with the boundaries around pending jobs. The method isJobScheduleAvailable(Date, Date) in the Handler class, determines whether the start date and end date of a job has a boundary of three days from the current day, and makes sure there isn’t more than 5 jobs within one week. The method isJobScheduleAvailable() is tested by the method testIsJobScheduleAvailable() in the class HandlerTest.
3. A Volunteer may not sign up for a work category on a job if the maximum number of Volunteers for that work category has already been reached.
   * Within the Job class the methods isLightVolunteersFull(), isMediumVolunteersFull(), and isHeavyVolunteersFull(), all check whether there is room for more volunteers in that work category. If there isn’t based off of the max set by the park manager than a false will be returned and the volunteer will be told that that work category is full. The JobTest class tests all these methods.
4. A job may not be scheduled that lasts more than two days.
   * Within the Job class the method isJobScheduleValid(Date, Date) takes in a start date and an end date for a job and ensures that there is only a max of two days between them. The class JobTest tests the method isJobScheduleValis(Date, Date) through the metho
5. A job may not be added that is in the past or more than three months in the future.
   * Within the Job class the method isJobScheduleValid(Date, Date) takes a start date and an end date as well as generates the current date in the method. With all this information it checks to makes sure that the start date of a job is within 90 days in the future and does not take place before the current date.
6. A Volunteer may not sign up for a job that has passed.
   * The isJobScheduleValid(Date, Date) method is the Job class returns a boolean so if it returns a false it will not allow the volunteer to sign up for the job because the job has most likely passed.
7. A Volunteer may not sign up for two jobs on the same day.
   * The Handler class has a method called canSignUpForJob(Job) that checks to make sure that you are not signing up for a job that is happening within the same start date and end date of another job you are signed up for.
8. A Park Manager can create jobs only for those parks that he/she manages.
   * Park Managers can only add jobs to the park that they manage because the park they are set to is automatically put into the the Park class constructor and then the addJob(Job) method can be used to add the new job.