1. Steps to take:   
   - gather requirements for crucial improvements in the most wanted areas and prioritize them (fields where crucial bugs emerged on production, the most important and required improvements in UI and UX making the application more appealing for the end user)  
   - prepare timeline and Test Plan putting the current planned improvements on the map (so that both we as the company and Client are on the same page)  
   Tools to introduce:   
   - platform for gathering test cases, User Stories, defects (TestLink, VSTS, TFS, Jira)  
   - platform for knowledge sharing (depending on complexity: Confluence, Jira, Sharepoint – if the Client persists: emails exchanged between Client or Third Party Platform – implied that the effort on our side is bigger then for just viewing or analyzing info gathered on many email messages)  
   - presence of any kind of functional or system documentation is always a plus (additionally if it can be stored in one place and maintained throughout the whole cooperation period); Also functional documentation – at least partially - can serve as manual for the end (business) users (as a result some problems with communication on the Client side can be mitigated) especially if team actions result in not only repairing app but also making it bigger and more capable (not just in the context of already present defects)
2. Test cases (as a general rule things like preferred browser is not mentioned at the level of single test case scenario as it is something well known especially when development and testing team is already established – team should be informed which browser – when we are taking web apps into consideration - is the default web browser for end users and therefore on that browser manual – and if possible also automated – testing should take place):   
   1. Registering for the app as the new user (signing up) with the automatic logon upon successful Sign Up.   
   Preconditions: User has not yet registered to the application (already registered data: username cannot be used again). User is not logged to the application (if he/she was logged in previously, action of log out needs to follow or time required for session expiring needs to pass – for the mentioned application 15 minutes is sufficient).   
   *Step 1: Actions to take:*   
   As a user from preconditions navigate to the app available under the following link: <https://yelpcamp-dem0.herokuapp.com/>.   
   *Step 1: Expected results:*   
   Main page of the app is displayed.   
   *Step 2: Actions to take:*   
   Check if “Welcome to YelpCamp” message is displayed and button: “View All Campgrounds” is present.   
   *Step 2: Expected results:*   
   All the mentioned elements are present.  
   *Step 3: Actions to take:*  
   Check if background images change – check if overall 5 different background images are loaded (they should fade in after every 10 seconds).   
   *Step 3: Expected results:*  
   Background images change. Overall number of background images is 5 and each of them fades in every 10 seconds.   
   *Step 4: Actions to take:*  
   Click on “View All Campgrounds” button.   
   *Step 4: Expected results:*   
   Home page with  
   Navbar   
   and Welcome to YelpCamp section   
   and at least one campground added is displayed for the user.   
   *Step 5: Actions to take:*   
   Check if Navbar contains:   
   on the left side: YelpCamp, then Home (highlighted as active) and   
   on the rights side (in the order from left to right) “Login” and “Sign Up”.   
   *Step 5: Expected results:*   
   Navbar contains the mentioned elements (positioned as described).  
   *Step 6: Actions to take:*Click on “Sign up” element located in the top right corner of the viewed page.  *Step 6: Expected results:*Signing up section is displayed for the user.   
   *Step 7: Actions to take:*Check if the link for the current section is the following one: https://yelpcamp-dem0.herokuapp.com/register.  *Step 7: Expected results:*The link is exactly as described.   
   *Step 8 Actions to take:*Check if Navbar is visible (containing the same elements as described in step 5 of this test case scenario, but this time “Sign Up” element should be highlighted).  *Step 8: Expected results:*Navbar is visible and it contains all the elements mentioned in the step 5 and this time “Sign Up” is highlighted.   
   *Step 9 Actions to take:*Check if “Sign Up” header is visible below the Navbar.   
   *Step 9: Expected results:*“Sign Up” header is visible below the Navbar.   
   *Step 10: Actions to take:*Check if the following fields with tooltip (message displayed within the field which disappears as soon as user clicks on it) are present below Sign Up header:  
   “username”  
   “password” (located under username).   
   *Step 10: Expected results:*The mentioned fields are located precisely as described.   
   *Step 11: Actions to take:*Check if button “Sign Up!” is located below password field and if the “Go Back” link is located below “Sign Up!” button.   
    *Step 11: Expected results:*The mentioned: button and link are located precisely as described.  
   *Step 12: Actions to take:*Enter unique username (unique so it was not used before) in the field “username“ – for example: Testuser2. Enter password: for example: 123456. Click on Sign Up! button. Side note: please make sure all those entered details are noted as they will be crucial not only for the purposes of this test case scenario but also as user’s credentials whenever login functionality is checked.  
   *Step 12: Expected results:*User is redirected to the home page again.   
   *Step 13: Actions to take:*Check if the URL of the displayed section is the following:   
   <https://yelpcamp-dem0.herokuapp.com/campgrounds>   
   *Step 13: Expected results:*URL of the displayed section is as described.   
   *Step 14: Actions to take:*Check if the following message below Navbar is displayed: “Welcome to YelpCamp Testuser2” where Testuser2 is the actual value entered in “username” field in step 12 of this test case scenario.  
   *Step 14: Expected results:*The described message is present.  
    *Step 15: Actions to take:*Check if the on the rights side of the Navbar instead of Login you can find: “Signed In As Testuser2” and instead of Sign Up you can find “Logout” element (on the top right corner of the Navbar).   
   *Step 15: Expected results:*The mentioned elements are present in the Navbar in the described places.  
     
     
   Other test case scenario concerning action of Login will be covered in automated test cases (due to limitation of time).   
     
   Automation: done with Cypress.   
   Here is the instruction how to proceed with running automated tests.   
   Make sure that Node.js is installed on your local machine.   
   Recommended IDE is Visual Studio Code.   
   Create a folder in your local hard drive – for example CypressAssignment. Right click on that folder and select Open in Code.   
   As a result Visual Studio Code should open that folder. If the view for terminal is not opened by the default – please from the upper menu click on Terminal – New Terminal. From the lower view of Visual Studio Code click on Terminal and make sure it points to the location you have opened. If it doesn’t move up (one folder higher) via writing “cd ..” (omit quotation marks, please) into the terminal and pressing Enter key or by writing “cd [here goes your full path to the folder for example: D:\materials\CypressAssignment]”. You can list the elements of the currently accessed folder via the command “ls”. Once you are in the right folder via terminal (and Node.js is already installed on your machine) type in “npm init” and click on Enter. You will be prompted a few questions like the author, keywords and so on. Whatever you select here is of minor importance, but just to be precise you will be asked about:   
   package name – confirm the default by pressing Enter,   
   version 1.0.0 – confirm as well,  
   description – leave empty,   
   entry point: (index.js) – again just confirm;   
   test command – leave empty,   
   git repository – leave empty as well,   
   keywords – leave empty,   
   author – it is actually me but you can write your name,   
   license (ISC) – leave as it is,   
   Is this Ok? - type in ‘yes’ and press Enter.   
   Now while being in the same folder type in: npm install cypress – it should take a while for the package being downloaded and installed. When installation is finished you should see message like: + cypress@6.2.1

added 217 packages in 37.284s (the version and number of package as well as time that your computer needed required may vary but the structure of the message should be the same).   
Now from the folder which was chosen for the installation find and open package.json file and make sure your script section looks exactly like this (it will be required for running Cypress Test Runner):  
  
 "scripts": {

"cypress:open": "cypress open",

"cy:open": "cypress open",

"cy:run": "cypress run"

},  
  
Type in “npm run cypress:open” in your terminal and press Enter. After a few seconds depending on your machine’s capabilities you should see a new window opened (that is the CypressTestRunner) and a new folder in your main folder chosen for installation (named: Cypress) should appear.

Make sure you download file: assignment.spec.js (sent via email) and put in into: [FolderWherePackageWasInstalled \cypress\integration\examples].   
Now from the view of your Cypress Test Runner make sure you expand Integration Tests folder and from there a file: assignment.spec.js should be double clicked.   
And now Cypress should give you a view of the automated actions on the application.   
Any questions – please let me know (preferably via email).

1. Imagine you are responsible for testing a web application. Your team is preparing a release of a new feature. Just before the release you spot a bug. What next steps you would take to resolve the situation. Take into consideration different scenarios and circumstances.  
     
   Depending on severity and priority of the spotted bug – if the bug messes with functionality already working fine on PROD I get in touch with Project Manager as soon as possible. In this particular situation I inform about the bug explaining potential consequences it can have on the current system and its users. If release cannot be postponed then I already prepare bug description informing DEV team that fixing this bug will be top priority plus will result in a hotfix on PROD (of course after a bug is verified successfully on TEST environment by me). If it can be moved to a later time/date – then I am making the team aware – especially DEV team – it is the important part for the release to be successful. If on the other hand severity and/or priority of the spotted bug is not high I simply report the bug expecting it to be deployed during the next release (if it is included into release scope – usually that is the Project Manager/ Product Owner or Analyst’s decision to make).