CS673 Software Engineering
Team 1 - UNO
Plan.ly
Test Report

Team Member	Role(s)	Signature	Date
George Wright	Team Leader	*George Wright*	4/6/2021
Aysha Zenab Kenza	Security Leader	*Aysha Zenab Kenza*	4/7/2021
Vibhu Bhatia	Backup Team Leader	*Vibhu Bhatia*	4/7/2021
Karen Sommer	Design and Implementation Leader	*Karen Sommer*	4/7/2021
Zach Schandorf-Lartey	Configuration Leader	*Zach Schandorf-Lartey*	4/7/2021
Chris Kulig	Requirements Leader	*Christopher Kulig*	4/7/2021
Matt Dowding	Quality Assurance Leader	*Matthew Dowding*	4/7/2021

Revision history

Version	**Author**	**Date**	**Change**
1	Group	4/6/2021	Initial release
2	Group	4/27/2021	Release for iteration 3

Introduction

Test Summary

Tests Reports

Testing Metrics (table copied from SPPP)

References

Glossary

Introduction

This document provides an overview of the testing program and strategy being used by the Plan.ly development team. Plan.ly is being developed using an Agile management structure, with both developers and a QA lead carrying out the implementation of product testing. Our technical stack includes a Python backend and a ReactJS front-end, and all testing is being managed from the Python back-end. We are using the Python unittest framework for unit testing, and for end-to-end testing of the front-end, we are using the Python selenium module.

Developers are themselves responsible for unit tests covering the stories which they are assigned during team planning meetings. Our QA manager is taking the lead for the selenium end-to-end tests.

Test Summary

Unit Tests. For unit testing, we have used the Django built-in unit testing mechanism in tests.py. In this file, we have checks to verify that we can create and delete model records with the expected successes and validation exceptions being returned by the back-end. Example tests include:

- Checking to make sure that the id field of the User model cannot be assigned by a developer; only the back-end
- Testing that Event Invitee records can be created and tied to an event/user using the respective ID attributes of each.
- Testing behavior when task records are given extremely long names/descriptions (exceeding the validation requirements)
- Checking that no ValidationError is raised when an Event record is created with all required fields filled incorrectly.

System Tests. Our system testing approach relies upon Selenium, which we have imported through its Python module. The system testing is designed specifically around the requirements we have stored in PivotalTracker: in the spreadsheets referred to below, we specifically map each test case to the requirements it exercises. We aim to automate each and every single one of our test cases. By doing so, we will ensure they can be quickly run and serve as a valuable tool to quickly test changes to our app and ensure nothing has regressed. Samples of system test cases include (but are not limited to):

- Sign up for a new Plan.ly account and ensure you can log in
- Attempt to log in with credentials known to be invalid and ensure a warning is given
- Sign in, navigate to an Event page, and attempt to edit the event; ensure the details are updated on the event page once "Save" is clicked on the edit form

Tests Reports

In order to ensure adequate test coverage, our process is to map requirements from PivotalTracker to test cases using two spreadsheets:

1. End-to-End Test Cases inventories and specifies the procedure for a series of end-to-end test cases to exercise all functionality of the product. Additionally, it reports the results of each execution of the automated test cases in the columns to the right. https://docs.google.com/spreadsheets/d/1gZJcX6biXOf7Gv98G0zO-6l7JiUrrCSRTK-n2O4d__g/edit#gid=0

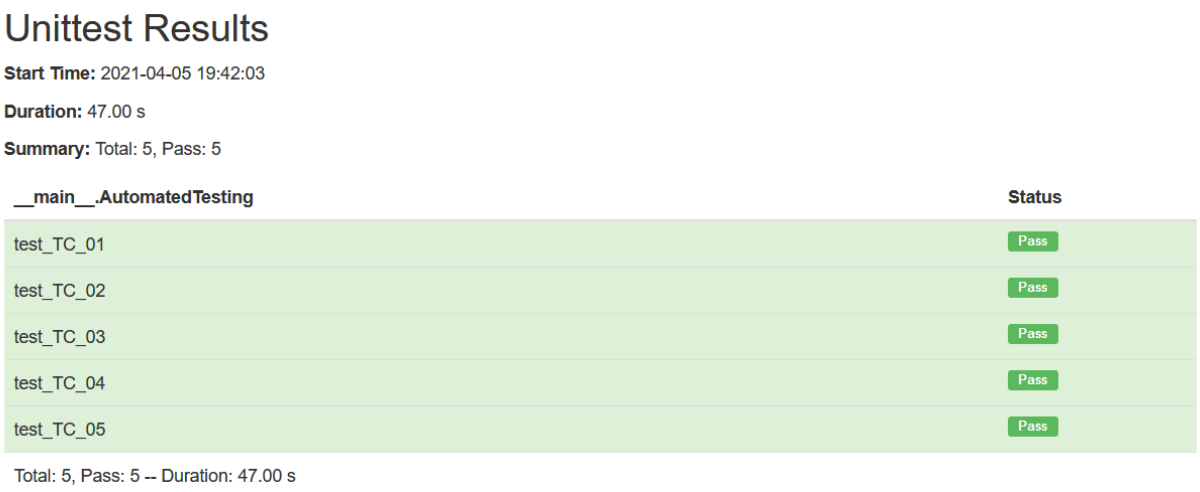
						Date: 2021-04-03 15:34:47		Date: 2021-04-04 18:58:05	
Test Case ID	Description	Type	Objective	Procedure	Expected Results	Results	Comments	Results	Comments
TC-01	Sign up with Plan.ly for the first time, using a valid email account	Positive Functionality	Ensure that Plan.ly allows new (unregistered) users to sign up using a valid email account	1. Open web browser and navigate to Plan.ly homepage 2. Click "signup" to initiate registration process 3. Choose the option to register with a Google account 4. Type in the credentials of a known email account to be used for testing 5. Click submit	1. Account is successfully created and user is redirected to the dashboard/feed page. 2. User receives an automated email (at the email address associated with the email account) confirming that the registration was succesful.	P	N/A	Error: Name Error	syntax error
TC-02	Log in to Plan.ly with a valid email account	Positive Functionality	Ensure that Plan.ly allows a previously registered user to sign in with their associated email account	1. Open web browser and navigate to Plan.ly homepage 2. Click "Sign In" button to initiate logging in 3. Choose same email account the user has previously signed up with 4. Click "submit" button	1. Successful log in with correct Google account 2. Redirect to user dashboard	P	N/A	P	N/A
TC-03	Logout from Plan.ly	Positive Functionality	Ensure that Plan.ly allows users to logout after they successfully logged in	1. Open web browser and navigate to Plan.ly homepage. 2. Login with known email account credentials that were previously used to register on Plan.ly 3. Once logged in, click logout	1. Successfully logged out of users account 2. Return to Plan.ly homepage	P	N/A	P	N/A
TC-04	Logout from Plan.ly and attempt to click back button	Negative Functionality	Ensure that a logged out user cannot use the back button to see restricted content	http://plan.ly/ register on Plan.ly 3. Once logged in, click logout 4. Click the "back" browser button	1. The user is not able to return to previous restricted content 2. User redirected to Plan.ly homepage	P	N/A	P	N/A
TC-05	Create a private event	Positive Functionality	Ensure that a registered user on Plan.ly can create an event	1. Open web browser and navigate to Plan.ly homepage. 2. Login with known email account credentials that were previously used to register on Plan.ly 3. Clcik on create an event 4. Verify that an event has been created	1. Successfully create an event for the user	P	N/A	Error: No Such Element Exception	failure after merge with updated dev

2. Requirement Testing Coverage then maps each test case to the requirements it covers (as exported from PivotalTracker). https://docs.google.com/spreadsheets/d/1FhZ3sMop-bVWqO1shsqFxP1SVxuPNs6RsYk

Cb2k_nfQ/edit#gid=0

Pivotal Export (4-5-2021)				Test cases					
Id	Title	Description	Labels	TC-01	TC-02	TC-03	TC-04	TC-05	TC-06
176879523	Log in	essential, user authentication/registration/profile	As a web site/user I want to be able to log into my account to securely gain access to my resources. -2H						
176949922	Sign up	essential, user authentication/registration/profile	As a website user I want to be able to create an account to be able to create so that I can log into the page as an authorized user and use the page functions as necessary.8.5H						
176879631	View my current private events	basic event management, essential	As an event owner, I want to view my current private events, so that I see how are they going - 4H						
176879628	Create a private event	basic event management, essential	As a website member(Signed User), I want to create a private event, so that I can share details like the description, location, and starting and ending date and time, of the event with the invitee. 6.5 H						
176879638	Update my private event	basic event management, desirable	As an event owner, I want to update a private event, so that I can modify the information(description, starting and ending date, and time) of the event. 7.2H						

3. Finally, we are using HtmlTestRunner Report Generator to produce an HTML-based output of each execution of our selenium automated test cases. A screenshot of its output is shown below.



Unittest Results

Start Time: 2021-04-05 19:42:03

Duration: 47.00 s

Summary: Total: 5, Pass: 5

__main__.AutomatedTesting	Status
test_TC_01	Pass
test_TC_02	Pass
test_TC_03	Pass
test_TC_04	Pass
test_TC_05	Pass

Total: 5, Pass: 5 -- Duration: 47.00 s

Testing Metrics (table copied from SPPP)

Metric ID	Metric Description	End of Iteration 0	End of Iteration 1	End of Iteration 2	End of Iteration 3
M-1	Testing coverage	0	4	13	24
M-2	Unit tests written	0	0	7	7
M-3	Unit tests passing	0	0	7	7

M-4	System test written	3	4	6	17
M-5	System test simulated	0	4	6	17
M-6	System tests passing	0	4	6	17

References

- Unittest
- Selenium
- Python
- ReactJS
- HTMLTestRunner

Glossary

- **Unit Tests.** Unit tests are designed to test small units of code to ensure that functions/methods/etc. are correctly performing the task they were designed to perform. Unit tests should especially focus on boundary conditions.
- **System Tests.** Testing the end to end functionality of the current iteration of software and making sure it works as intended. Stories are extracted from pivotal and end to end testing is created to ensure coverage of those stories. Selenium scripting is then written to ensure that there is automated testing coverage.
- **Acceptance Tests.** Testing that is performed to accept the software before moving the software application to a production environment. These tests will be done upon finishing a feature and merging into development.
- **Regression Tests.** Testing done to make sure none of the changes made over the course of the development process have caused new bugs nor resurfaced old bugs. These tests are done after updates to the development branch such as merging in a new feature. By creating simulated tests as development continues, regression testing can be carried out quickly and incorporated into the team's weekly processes.