iCrew:
Quality Assurance Plan
version 1.0

Cru iOS App
Computer Science Department
California Polytechnic State University
San Luis Obispo, CA USA

December 2, 2015

Contents

Credits

Name	Date	Role	Version
Eric Tran	December 2, 2015	Document Owner and Creator of Test Case 2 and 3	1.0
Jordan Tang	December 2, 2015	Author of Introduction and Creator of Test Cases 1, 4, 5, 6	1.0
Daniel Lee	December 2, 2015	Co-Author of Approach	1.0
Tammy Kong	December 2, 2015	Co-Author of Approach	1.0
Mariel Sanchez	December 2, 2015	Author of Key Participants	1.0

Revision History

Name	Date	Reason for Changes	Version

1 Introduction

This document outlines our quality assurance plan for the Cru Central Coast application. The application will allow users to find rides to events, sync events to their calendar, find ministry teams and community groups, view resources, and apply to summer missions. We will be trying to implement all the use cases outlined in our SRS document, and will be using them as a basis for our test cases.

2 Approach

2.1 Process

We plan to be testing continuously through each iteration to make sure our work aligns with the customer specifications and use cases. We will have an initial suite of test cases which will cover how we expect the application to behave based on the use cases. Before release, we will run the current build against the test suite in Jenkins in order to validate the iteration. The history of test runs should be recorded in Jenkins, and failed test cases should be resolved immediately.

2.2 Tools

The tool used for testing our solution is Jenkins. With Jenkins, we will be able to do continuous testing automatically and efficiently. All members of the team will be expected to use Jenkins.

3 Key Participants

In terms of running tests, each group member should run the test suite after finishing a major component of the application. This is to ensure that each piece for the iteration still aligns with our use cases.

4 Test Cases

4.1 Acceptance Criteria

Test Case ID:	1	
Requirement:	UC-1 Join a Community Group (See SRS v3.0)	
Created By:	Jordan Tang	
Last Updated By:	Jordan Tang	
Date Created:	December 2, 2015	
Date Last Updated:	December 2, 2015	
Preconditions:	Patron has selected "Get Involved" on the navigation bar	
Actions:	**Step**	**Expected Output**
	Patron selects "Join a Community Group"	System displays a questionnaire for the patron to fill out
	Patron fills out all fields in the questionnaire	
	Patron selects a question to fill out	System will expand to a larger text area for the patrons answer
	Patron confirms question answer	System returns to questionnaire
	Patron finishes questionnaire and submits	System will display community groups that align with the patrons questionnaire
	Patron selects community groups they wish to join	System will display the list of community group leaders contact information to the community groups they selected.

Test Case ID:	2	
Requirement:	UC-5 Offer a Ride (See SRS v3.0)	
Created By:	Eric Tran	
Last Updated By:	Eric Tran	
Date Created:	December 2, 2015	
Date Last Updated:	December 2, 2015	
Preconditions:	Patron has selected "Events" on the navigation bar	
Actions:	**Step**	**Expected Output**
	Patron selects "Rides"	System switches to display ride options including "Status, "Offer, and "Request"
	Patron selects "Offer"	System displays a questionnaire for patron to fill out
	Patron selects a question to fill out	System expands to a larger text area for the patrons answer
	Patron confirms question answer	System returns to questionnaire
	Patron finishes questionnaire and submits	System proceeds to a screen confirming patron is now offering rides

Test Case ID:	3	
Requirement:	UC-6 Request a Ride (See SRS v3.0)	
Created By:	Eric Tran	
Last Updated By:	Eric Tran	
Date Created:	December 2, 2015	
Date Last Updated:	December 2, 2015	
Preconditions:	Patron has selected “Events” on the navigation bar	
Actions:	**Step**	**Expected Output**
	Patron selects “Rides”	System switches to display ride options including “Status, “Offer, and “Request”
	Patron selects “Request”	System displays a questionnaire for patron to fill out
	Patron selects a question to fill out	System expands to a larger text area for the patrons answer
	Patron confirms question answer	System returns to questionnaire
	Patron finishes questionnaire and submits	System proceeds to a screen informing the patron about the driver they have been paired with, displaying the drivers contact information in the process

Test Case ID:	4	
Requirement:	UC-9 Sync Calendar to iOS (See SRS v3.0)	
Created By:	Jordan Tang	
Last Updated By:	Jordan Tang	
Date Created:	December 2, 2015	
Date Last Updated:	December 2, 2015	
Preconditions:	Patron has selected the “Events” tab	
Actions:	**Step**	**Expected Output**
	Patron selects the event they wish to sync	System displays the extended display for that event
	Patron presses the calendar button in the bottom left	System displays a notification that the event has been synced

Test Case ID:	5	
Requirement:	UC-4 Apply to a Summer Missions Trip (See SRS v3.0)	
Created By:	Jordan Tang	
Last Updated By:	Jordan Tang	
Date Created:	December 2, 2015	
Date Last Updated:	December 2, 2015	
Preconditions:	Patron has selected the "Summer Missions" tab	
Actions:	**Step**	**Expected Output**
	Patron selects Summer Mission they wish to apply to	System displays extended information for the Summer Mission with an Apply button on the bottom
	Patron presses the Apply button	System opens the native browser the the summer mission application

Test Case ID:	6	
Requirement:	UC-13 Viewing Events (See SRS v3.0)	
Created By:	Jordan Tang	
Last Updated By:	Jordan Tang	
Date Created:	December 2, 2015	
Date Last Updated:	December 2, 2015	
Preconditions:	Patron is on a tab other than the Events tab	
Actions:	**Step**	**Expected Output**
	Patron selects the Events tab on the navigation bar	System displays a list of all events, queried from the database
	Patron selects "Fall Retreat"	System shows extended information for Fall Retreat

4.2 System Tests

The functionality we need from the system is as follows:

- Sync to the native and Google calendar
- Fetch YouTube videos from the Cru YouTube channel
- Fetch information from the database including necessary phone numbers, event information, and summer mission information

We will be testing these manually by seeing if the necessary information appears on the app. Since we are simply grabbing information from the respective services, we can observe when an API is not working properly, as the necessary screens won‘t be populated with information.

4.3 Non-functional Tests

We will determine if our system meets its non-functional requirements by manually performing tasks and checking visually whether or not the requirements were fulfilled.

A Glossary

B Issues List