San Diego State University

SDS Verification Test Plan for: Kili Trekker System

Version: 1.0

Members: Aatusa Mehdiyan, Jenny Nguyen, Minh Nguyen, Jose Payan

GROUP 13

CS 250

Professor Bryan Donyanavard

November 2, 2021

Testing Goals

System Acceptance Testing: Complete System

- This testing will be performed to determine if the Kilitrekker System has met the requirement specifications and also evaluate the overall system's compliance with the park company's requirements. Another significant goal of this testing is to verify if it has met the required criteria for proper delivery to end users including current/potential park visitors, rangers, park staff, and tour guides. New acceptance tests should be conducted for each iteration during development for quality assurance and to confirm correctness.

Function/Integration Testing: Interfaces among integrated units

- This document will overview the testing of the multiple components of the Killi Trekker System and view if the components function together properly. We will analyze how the external systems interact with the main system and check for proper data flow and database changes between modules. Thorough testing of the interfaces will help formulate a successful user experience when using the Kilitrekker website.
- **Functional Testing:**
  - Testing the software in its entirety to confirm that each function, feature or module of the trekker is working as intended.
- **Integration Testing:**
  - Testing multiple modules after integrating them together to detect potential defects. Trekkers must communicate with different modules. We will be checking for performance, data flow and any changes that take place during their interactions.

Unit Testing: Single Code "unit" (e.g., method)

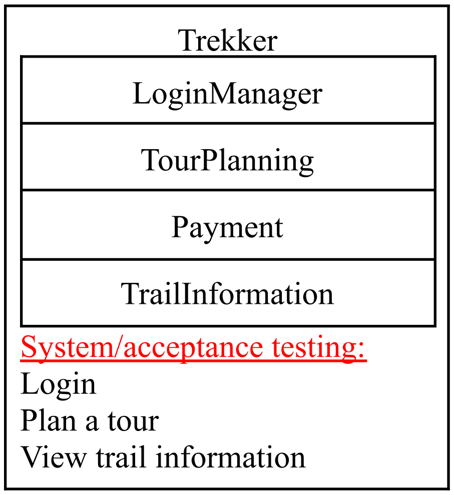
- **User Interface Test Cases:**
  - Check functionality of textboxes, buttons, dropdowns, internal and external links, and page scaling.
- **Input Data Validation:**
  - Input data validation will encompass a range of checks that may be adopted generally to the data which is entered into an application system. It will check and establish:
    - Mandatory Fields testing
    - Unique Field Values testing
    - Null value testing
    - Acceptance of allowable characters
    - Negative values testing
    - Field length specifications
    - Improbable Value testing
    - Garbage Value testing

Trekker Class

System Acceptance Testing: Complete System

Trekker Class contains the following main components:

- LoginManager
- TourPlanning
- TrailInformation
- Payment



LoginManager

System Acceptance Testing: Test objective for the LoginManager is to allow users to have their account information and settings stored into the system efficiently. They will be able to access and update their personal information to their preference and also view their Tour information/history. LoginManager module controls are:

- Login
- Logout
- Register

Function/Integration Testing:

- The Login Manager module will store the following information into the Killitrekker system's database after successful login. This module will communicate with the database, and the overall Killitrekker website. Login information in particular will connect with the TourPlanning and Payment modules to allow users to complete their tour bookings.
    - Int: rangerID
    - Int: parkStaffID
    - Int: guideID
    - Int:trekkerID
    - String: username
    - String: password
    - String: email

- Int: loginDate

- The expected fields are checked.
- A welcome message must appear after successful login and an error message must appear on the screen after an invalid login
- There must be stored site cookies for login fields.
- “Forgot Password”(“Reset Password”)/ “Forgot Username”/ “ForgotID” functionality options should be present
- Login attempts: disable for one hour after 3 unsuccessful login attempts
- Successful login will bring users to the Welcome page, Home page, or to the original page they were on.

Unit Testing:

- **Input data validation:** In order for users to successfully log-in, they must satisfy the required fields in order to further access the system’s other major components for security. Such testing will check for secure and reliable user interface and usability. Specifications include:
    - Valid E-mail address must exist
    - Passwords are case sensitive and must be alphanumeric. Length should be between 8 to 32 characters.
    - Text fields cannot exceed the signified lengths
    - Errors for Null inputs in email addresses including empty inputs
    - No Invalid characters

	isAdminLogin	MyAccount
Login --- String: Email String: Password int: ID bool: isAdmin --- void setIsAdmin(...) Unit testing: Access.setIsAdmin(...)	LoginTest --- bool {true, false}: loginTest if (loginTest=F) return readAccess; else if (loginTest=T) return fullAccess; else return 0; Web: connectedWeb --- void readAccess(...) void fullAccess(...) Function/Integration testing: LoginTest.readAccess(Login)	Login --- SettingsManager --- TourPlan --- Payment --- RegisterAccount System/acceptance testing: Access to website Read all park information Login Logout

Test Case #	Test Case Description	Test Steps/Data:	Expected Results	Actual Result	Pass/ Fail
1	User goes to the website and clicks the Register button to sign up. User types in their email, UserID, and password in the empty fields and click submit. Check response when valid information is entered	Email: user123@email.com UserID: 34567889 Password: pass999	Registration should be unsuccessful. Users must enter a valid password with a symbol for highest security. Users should not log in and the website will display an error page and require the user to try again.	Registration unsuccessful	Fail
2	Ranger goes to the website and clicks the Login/Register button to sign in. Ranger types in their email, rangerID, and password in the empty fields and click submit. Check response when valid information is entered	Email: ranger99@email.com RangerID: 09123442 Password: lNf9^Oti7^2h	Login should be successful and grant the ranger read/write permissions to the system.	Login successful	Pass
3	Check when required fields are not filled and no data is entered.	E-mail: ID: Password: Click-> Register	Page should display a mandatory symbol(*) next to required fields in red.	Registration unsuccessful	Fail
4	Check when the user inputs an invalid e-mail for log-in.	E-mail: eee@hotmail.com ID: 67896654 Password: Morty2018!	Page will refresh and highlight the "Email" field and note to the user, "Email does not exist within our system. Please try again. ⅔ login attempts left"	Login unsuccessful	Pass
5	Check when the user inputs invalid characters into text fields.	E-mail: yakuzadestroyer22@yahoo.com UserID: aasdklasdk Password: KiryuDameDane1&	Page will refresh and highlight the "UserID:" field. A red message will appear and tell the user, "Invalid characters. Please only enter numeric values only"	Login unsuccessful	Pass

System Acceptance Testing: Testing objective of the TourPlanning module is to ensure users are able to book their tours at the park without issue. Their payment information, tour dates, guide information, party size, and events will be customizable and displayed for them when they access their profile. Specified tour guides will be allowed to see the park visitor's selected info necessary for tour guide's planning preparation.

- Plan tour
- Choose tour dates
- Choose a trail to tour
- Add event
- Add tour guide
- Add person to party
- Pay for tour

Function/Integration Testing: TourPlanning module will have multiple components sending data to each other and will be stored personally into park visitors' accounts. Testing will ensure that user actions are fetched and selections load and update the information from other pages with no issue. TourPlanning connects to the Payment module to calculate payment amount for the tours and process payments successfully.

- List<TourGuides>: string guides
- List<TourInformation>: string tourInfo
- string: tourGuideName
- String: tourGuideAbout
- Bool: isAvailable
- Void chooseGuide(...)
- Void chooseDate(...)
- Void addPerson(...)
- Void addEvent(...)

Unit Testing:

- Links and booking modules are tested. This unit testing relies on users to interact with the components directly by clicking buttons and using dropdown menus to select specific values. Testing will oversee:
    - Successful button triggers on all devices such as desktop and mobile(func fetchUserAction(...))
    - UI buttons are displayed properly to user and accurately labeling
    - Ease of selection of tour dates through a pop-up calendar widget that allows users to pick the days they will be visiting
    - Clear use of menu choices of tour guides, events, and trails

	PlanTour	TourPlanning
Tour String: tourName Int: TourDates Void setTourDates(...)	TourList List<TourGuides> tourGuideNamesList App: connectTour void addEvent(...) void deletedTour(...)	AddTrailTour TourDates AddEvent ChooseGuide AddPerson TourPayment Refund
Unit testing: Tour.setTourDates(...)	Function/Integration testing: PlanTour.addTour(Tour n)	System/acceptance testing: Add a Trail Tour Delete tour dates Remove event Remove tour guide Remove person from party Refund for tour

Test Case #	Test Case Description	Test Steps/Data	Expected Results	Actual Result	Pass/Fail
1	User interface testing: Check functionality of trail condition buttons, links, trail dropdown options, and textboxes.	Click on the dropdown menu to select a trail. Ex) User selects and clicks the "Trail 1 Conditions" link.	Page redirects to Trail 1 information and displays Trail 1's camera, condition rating, and difficulty.	Page redirect and Trail 1 media display successful.	Pass
2	User interface testing: Check functionality of unavailable trails.	Click on the dropdown menu to select a trail. Ex) User selects and clicks the " Trail 7 Conditions" link.	Page redirects to Trail 7's camera, condition rating, and difficulty.	Page redirect and displays " Trail 7 Condition: Poor , Tour Unavailable ".	Fail
3	User interface testing:	User does not click on a trail	Page does not	Page does	Fail

	Check when user does not hit a trail selection	from the drop down menu.	redirect to another page. Page will highlight the user to select a trail from the dropdown menu.	not redirect and stays on the original page.	

Payment

System Acceptance Testing: Testing objective of the Payment system component is to test that the payment processor appointed by a merchant can handle transactions from different channels. It will check and receive details to respective issuing banks or associations. It will also successfully accept payment information from users, perform and record transactions safely, and send such information to relevant parties.

- Select payment method
- Enter payment details
- Store payment details

Function/Integration Testing: In order for park visitors to complete their tour booking, the Payment component communicates with the TourPlanning component which will send payment funds to the Kilitrekker company bank account and also save and record transaction history details to the database. Payment details will also be externally sent out by email to customers.

- String: name
- String: billingAddress
- Int: cardNumber
- Int: securityCode
- Int: date
- Float: amountDue
- Void addPayment(...)
- Float calculatePaymentDue(...)
- Void paymentType(...)
- Bool paymentSuccessful(...)
- Void paymentDetails(...)
- Void refund(...)

Unit Testing:

- **Input data validation:** In order for users to successfully book and pay for their tour, they must satisfy the required payment fields and security measures. Specifications include:
  - Card information matches with billing and identity information
  - Card number field should be limited to 16 digits, no alphabetical letters or symbols are enterable.

- CV pins are required
- Card information is up to date and valid
- No Invalid characters in integer accepting fields including null inputs
- Fields cannot exceed the signified lengths

	PaymentProcess	MyPayment
Card	CardTest	SelectPaymentMethod

String: cardName
String: cardAddress
int: cardNum
int: cardCodeNum
int: cardDate
float: Payment
	bool cardTest
	if (cardTest = false) return 0;
	else return 1;
		EnterPaymentDetails

void setPayment	void approvedCard(...) void errorCard(...)	CancelPayment
		PrintPayment
		ViewPayment
		LoginManager
Unit testing: Card.setPayment(...)	Function/Integration testing: PaymentProcess.approvedCard(Card)	System/acceptance testing: Cancel a payment to get a refund View a payment of finalized order Print a receipt of payment Login logout

Test Case #	Test Case Description	Test Steps/Data	Expected Results	Actual Result	Pass/Fail
1	To pay for their order tour of services. They need to click the “Checkout” button, and enter shipping and payment details to the required fields. Then they click the “Submit" button to get a verified transaction.	Billing Name: John Smith Billing Address: 2312 Marlesta St. CA 92111 Card number: 2342 5234 4356 2345 CV: 345 Expiration: 12/21	Authorization to approve with correct card information input and successful message display on the payment page.	Payment successful	Pass
2	To pay for their order tour of services. They need to click the “Checkout” button, and enter shipping and payment details to	Billing Name: Jasmine Nguyen Billing Address: 2435 Convoy St. CA 92121 Card number: 1256 124 4566 120 CV: ??? Expiration: 12/21	Authorization to decline with incorrect card information input and failure message display.	Payment unsuccessful	Fail

	the required fields. But, they miss CV and wrong card numbers. Then they click the “Submit" button to get a verified transaction.		User will be directed back to the payment page to try again.		
3	To pay for their order tour of services. They need to click the “Checkout” button, and enter shipping and payment details to the required fields. But they delay entering the required fields process too long. The transaction request is not verified.	Billing Name: …. Billing Address: ... Card number: 2323 5234 000 000 CV:... Expiration: Payment process declares remaining time out on the page. Click the “yes" button to continue. Click “no” or run out of time countdown to end.	Authorization to decline and the session expired message display. Users will be directed back to the payment page to try again.	Payment unsuccessful	Fail

TrailI Condition InformationTracker

System Acceptance Testing: Test objective for the Trail Condition Tracker is to allow real time results of the trail conditions. The Tracker should efficiently update current weather conditions pulling information from the Killi Trek Website. A menu will display all trails and direct to the appropriate page for further evaluation.

- Approve Trail Condition
- Unapproved Trail Condition

Function/Integration Testing: The Trail Condition Tracker will confirm availability for safe trails for the Tour Planning Application. The application will be able to update real time with current trail conditions by accessing the Weather Application Interface database and Killitrekker website.

- View Trail Information
- List<TrailList>: trailList
- List<trailInformation> string trailInfo
- String : trailName “n”
- Void selectTrail(…)
- Void selectTrailHistory (. . .)
- View Trail Conditions
- Select a Trail Camera

Unit Testing:

- **Real Time Update of Trail Information :** For approval of trail availability , satisfied conditions must be met. As well as approved adminID for manual approval for moderate conditions.
    - Select a Trail : null Trails will not be accepted
    - Current Weather : "Good, Moderate, Warning: Risk" Real Time Weather update from the weather API ; Good: clear weather conditions ; Moderate: High or Low Temperature ,
    - View Camera : Select Type of Weather and Forecast
    - Approve Trail : input adminID and Password
    - Choose Green/ Red : Green approves ; Red marks trail as unavailable.

	TrailConditionTracker	Trail Information
Trail String : trailName bool : isSafe void setIsSafe(...) Unit testing: Trail.setIsSafe(...)	TrailList List<Trails> : trailList App: connectTrailCam AP : connectKiliTrekkerAPI selectTrail(n) Void getCurrentWeather(string "date") string viewCamera(currentTemp, trailLiveCam int n) Approve trail : enter adminID and Password Choose Green/Red : click green or red void addTrail(...) void deleteTrail(...) Function/Integration testing: TrailConditionTracker.addTrail(Trail n)	Choose trail View trail information View trail conditions Select a trail camera System/acceptance testing: Approve Trail Mark Trail as Unavailable Select Trail Weather Type Select

Test Case #	Test Case Description	Test Steps/Data	Expected Results	Actual Result	Pass/Fail
1	This tests the trails for	Select Trail : Trail 7	Int 1 ,	Green Light	Pass

	their condition and approval or denial for tour guide requirements. Accesses the database and updates the approved trails and information in real time and matches weather conditions. All good conditions are approved.	Current Weather : "Good" View Camera : Sunny , calm weather Approve Trail : "adminID" Green/Red : "Choose: GreenLight : isSafe else "RedLight: isNotSafe"	Display message "String : Green Light : isSafe"	"This Trail Conditions are Safe "	
2	This tests the trails for their condition and approval or denial for tour guide requirements. Accesses the database and updates the trails as unavailable and information in real time and matches weather conditions. All "Warning: Risky" conditions are unapproved.	Select trail: " trail 5" Current Weather : " Warning : Risk " View Camera : Raining , Windy Approve Trail : "Enter adminID" Green/Red : "Choose: GreenLight : isSafe else "RedLight: isNotSafe"	Int 0 , Display message " String : Red Light : isNotSafe"	Red Light "This Trail Conditions are not Safe"	Fail
3	This test approves an Administrators manual input to override moderate trail conditions with a valid adminID and Password. All invalid IDs and Passwords will not be able to make changes.	Select Trail : Trail 3 Current Weather : " Moderate: Risk " View Camera : "Cold , Windy" Approve Trail : "Enter adminID" Green/Red : "Choose: GreenLight : isSafe else "RedLight: isNotSafe"	Invalid ID	Error invalidID, Please enter correct ID and Password.	Error
4	Last Verification Step Method for Manual Approval. Ensures confidence in final review.	Select Trail from List : Trail 7 Current Weather : " Moderate: Risk " View Camera : "High Temp , Dry" Approve Trail : "Enter adminID" Green/Red : GreenLight : isSafe"	Are you sure you want to approve trail: "Yes Approve, No Set As	adminID: Approved Trail Thanks for Reviewing Trail 7	Pass

			Unapproved		
5	This test objectively test the Green/Red default set to "Auto" where the Trail Information Tracker automatically approves any trail with good weather conditions. Set trail as isSafe when weather conditions are "good" or moderate only.	Select Trail: 3 Current Weather : "Good" View Camera : " 73 Degrees, Sunny" Approve Trail : "admin1232 , password: passWord1232" Green/Red : Auto: Green	Auto: Green	Auto : Trail Approved	Pass
6	This test objectively test the Green/Red default set to "Auto" where the Trail Information Tracker automatically Unapprove's a trail and sets it as unavailable. Set trail as isNotSafe when weather conditions are "Moderate" and "Warning: risky".	Select Trail: 2 Current Weather : "Good" View Camera : " 73 Degrees, Sunny" Approve Trail : "admin1232 , password: passWord1232" Green/Red : Auto: Green	Auto: Red	Auto : Trail Unapproved	Fail

Noted Changes and Updates:

- Anti-Fraud measures/alerts against suspicious card transactions will be considered and integrated into the payment processor. <<void FraudAlert(...)>>
- Reorganization of the Trekker Trail component to update and show trail conditions and information at once and onto the same page.
- Automated Color-coded range grading of trail conditions that are consistently updated as data is retrieved from and analyzed by cameras, ranger reports, and weather conditions
- Tests