;

Project intermediate milestone:

Blackbox tests and plan of feature development

Team Members

Meet Nitinbhai Patel (B00899516)

Subject:

Software Development Concepts

Professor:

Mike McAllister

Blackbox Tests

In software testing, black box testing is important since it helps to validate the system's overall functionality. Customers' needs are used to conduct black box testing, which allows any missing or unanticipated requirements to be quickly discovered and resolved. Testing is done in a black box environment, with the end user in mind. From a customer's standpoint, the fundamental benefit of black box testing is that it handles both legitimate and incorrect inputs.

Various black box testing techniques.

1. Equivalence Class Partitioning

- Equivalence partitioning is a black box testing approach that splits a software unit's input data into data partitions from which test cases may be constructed.
- It decreases the number of test cases. An equivalence class is produced in equivalence class partitioning by the inputs for which the system's behavior is stated or anticipated to be similar.
- A collection of valid or incorrect states for input conditions is represented by an equivalence class.

• PersonIdentity addPerson(String name)

Input	Valid Equivalence class	Invalid Equivalence class	
Name	 Contains lower case letter Contains Upper case letter String length between 3 – 15 characters 	 Contains special character String length > 15 Contains Numbers Non-ASCII Character 	

Boolean recordAttributes(PersonIdentity person, Map<String, String> attributes)

o Examples of attributes are "date of birth", "gender", and "occupation"

Input	Valid Equivalence class	Invalid Equivalence class	
• Contains lower case letters other than "\"		Contains special character other than "\"Non-ASCII Character	
Gender	 Contain gender as "Male" or "Female" 	 Gender other than "Male' or "Female" 	

	Contains upper case lettersContains lower case letters	Contains numberContains special characterNon-ASCII Character
Occupations	 Contains lower case letter Contains Upper case letter String length between 3 – 15 characters 	 Contains special character String length > 15 Contains Numbers Non-ASCII Character

• Boolean recordReference(PersonIdentity person, String reference)

 Examples of a reference could be the location of a birth certificate or a web page that lists when someone graduated

Input	Valid Equivalence class	Invalid Equivalence class	
Reference	 Contains lower case letter Contains Upper case letter String length between 3 – 15 characters Contains Number 	 Contains special character String length > 15 Non-ASCII Character 	

• Boolean recordNote(PersonIdentity person, String note)

Input	ut Valid Equivalence class Invalid Equivalence class	
Note	 Contains lower case letter Contains Upper case letter Contains numbers 	Contains special characterNon-ASCII Character

• FileIdentifier addMediaFile(String fileLocation)

Input	Valid Equivalence class	Invalid Equivalence class
fileLocation	 Contains lower case letter Contains Upper case letter Contains numbers Contains Special Character 	Non-ASCII Character

Boolean recordMediaAttributes(FileIdentifier fileIdentifier, Map<String, String> attributes)

Examples of attributes are "year", "date", and "city"

Input	Valid Equivalence class	 Contains Upper case letter Contains Lower case Letter Contains Special character Contains less than 4 numeric characters Contains more than 4 numeric characters 	
Year	 Contains number Contains 4 numeric character 		
Date	Contains numbers	 Contains special character Contains Upper case letter Contains Lower case Letter Non-ASCII Character 	
City	 Contains lower case letter Contains Upper case letter String length between 3 – 15 characters 	 Contains special character String length > 15 Non-ASCII Character 	

• Boolean tagMedia(FileIdentifier, fileIdentifier, String tag)

Input	Valid Equivalence class	Invalid Equivalence class
tag	 Contains lower case letter Contains Upper case letter Contains numbers Contains Special Character 	Non-ASCII Character

2. Boundary Value Analysis

• Boundary value analysis is carried out by devising tests that put the edges of the input and output classes specified in the specification to the test.

We are testing following boundary values in boundary value analysis

- The extreme ends of the range
- Just beyond the ends
- Just before the ends

For Example, if an integer range is minimum to maximum, then the six values are

- Min − 1
- Min
- Min + 1
- Max − 1
- Max
- Max + 1

PersonIdentity addPerson(String name)

- o 1 character student's name
- o Must be within the length if defined

Boolean recordAttributes(PersonIdentity person, Map<String, String> attributes)

Examples of attributes are "date of birth", "gender", and "occupation"

Date of Birth

o Date must be between 1 and 31 and month must be between 1 and 12

Gender

Gender must be Male or Female

Occupation

- 1 character occupation
- Must be within the range if specified

Boolean recordReference(PersonIdentity person, String reference)

Examples of a reference could be the location of a birth certificate or a web page that lists when someone graduated

- 1 character reference
- o Must be within the range if specified
- Boolean recordNote(PersonIdentity person, String note)
 - 1 character note
 - o Must be within the range if specified
- Boolean recordMediaAttributes(FileIdentifier fileIdentifier, Map<String, String> attributes)

Examples of attributes are "year", "date", and "city"

date

O Date must be between 1 and 31 and month must be between 1 and 12

City

- o 1 character city name
- Boolean tagMedia(FileIdentifier, fileIdentifier, String tag)
 - o 1 character tag

3. Decision Table Testing

Methods	Conditions	Value returned
recordAttributes(), recordReference() recordNote(), recordMediaAttributes(), peopleInMedia(), tagMedia()	Passed appropriate not null String and not empty string	True
recordAttributes(), recordReference() recordNote(), recordMediaAttributes(), peopleInMedia(), tagMedia()	Passed null or empty string	False
recordChild()	If parent child relation was stored in database	True
recordChild()	Parent child relation doesn't allow to store in database	False
recordPartnering()	If relation between partner 1 and partner 2 was allowed to store in database	True
recordPartnering()	If relation between partner 1 and partner 2 doesn't allowed to store in database	False
recordDissolution()	If dissolution between partner 1 and partner 2 allowed to stored in database	True
recordDissolution()	If dissolution between partner 1 and partner 2 doesn't allowed stored in database	False

Other Black Box testing techniques are:

- State Transition Diagram
- Orthogonal Arrays
- All Pair technique

PLAN OF FEATURE DEVELOPMENT

System that links family tree information with an archive of pictures and the metadata of the pictures

From 17th Nov to 21th Nov

Family tree relation management

Storing the following individual's relation in the tree

- Parent/Child Relations
- Partnering relations
- Partnering dissolution

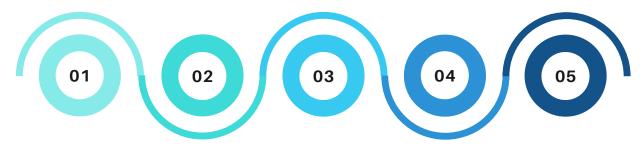
From 30th Nov to 5th DEC

Reporting Information as required

Part-1

The following information will be reported by this feature

- Location of an individual in the family tree
- Name of an individual based on iD
- Relation between two person
- Name of file associated with the file identifier



From 12th Nov to 16th Nov

Family tree management

Storing the following information about an individual into the family tree database

- Name
- Date and location of birth and death
- gender and Occupation
- references to the source material
- Notes on individual

From 22th Nov to 29th Nov

Media archive management

Storing the following information about media (Photo or Video) into a media archive database

- Filename
- The date on which the picture was taken
- Location of the picture where it was taken
- Tags and individuals seen in the picture

From 5th Dec to 10th DEC

Reporting Information as required Part 2

The following information will be reported by this feature

- List of descendants in the family tree within generations
- List of ancestors in the family tree within generations
- Set of media files linked to the given tag and Location between the provided start and end date
- List of specified person's immediate children

0

0