

**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
<b>Name</b>	<ul style="list-style-type: none"><li>• Contains lower case letter</li><li>• Contains Upper case letter</li><li>• String length between 3 – 15 characters</li></ul>	<ul style="list-style-type: none"><li>• Contains special character</li><li>• String length &gt; 15</li><li>• Contains Numbers</li><li>• Non-ASCII Character</li></ul>

- **Boolean recordAttributes(PersonIdentity person, Map<String, String> attributes )**
  - Examples of attributes are “date of birth”, “gender”, and “occupation”

Input	Valid Equivalence class	Invalid Equivalence class
<b>DOB</b>	<ul style="list-style-type: none"><li>• Contains numbers</li><li>• Contains lower case letters</li><li>• Contains upper case letters</li><li>• Contains special character “\”</li></ul>	<ul style="list-style-type: none"><li>• Contains special character other than “\”</li><li>• Non-ASCII Character</li></ul>
<b>Gender</b>	<ul style="list-style-type: none"><li>• Contain gender as “Male” or “Female”</li></ul>	<ul style="list-style-type: none"><li>• Gender other than “Male’ or “Female”</li></ul>

	<ul style="list-style-type: none"> <li>Contains upper case letters</li> <li>Contains lower case letters</li> </ul>	<ul style="list-style-type: none"> <li>Contains number</li> <li>Contains special character</li> <li>Non-ASCII Character</li> </ul>
<b>Occupations</b>	<ul style="list-style-type: none"> <li>Contains lower case letter</li> <li>Contains Upper case letter</li> <li>String length between 3 – 15 characters</li> </ul>	<ul style="list-style-type: none"> <li>Contains special character</li> <li>String length &gt; 15</li> <li>Contains Numbers</li> <li>Non-ASCII Character</li> </ul>

- **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
<b>Reference</b>	<ul style="list-style-type: none"> <li>Contains lower case letter</li> <li>Contains Upper case letter</li> <li>String length between 3 – 15 characters</li> <li>Contains Number</li> </ul>	<ul style="list-style-type: none"> <li>Contains special character</li> <li>String length &gt; 15</li> <li>Non-ASCII Character</li> </ul>

- **Boolean recordNote(PersonIdentity person, String note )**

Input	Valid Equivalence class	Invalid Equivalence class
<b>Note</b>	<ul style="list-style-type: none"> <li>Contains lower case letter</li> <li>Contains Upper case letter</li> <li>Contains numbers</li> </ul>	<ul style="list-style-type: none"> <li>Contains special character</li> <li>Non-ASCII Character</li> </ul>

- **FileIdentifier addMediaFile(String fileLocation )**

Input	Valid Equivalence class	Invalid Equivalence class
<b>fileLocation</b>	<ul style="list-style-type: none"> <li>Contains lower case letter</li> <li>Contains Upper case letter</li> <li>Contains numbers</li> <li>Contains Special Character</li> </ul>	<ul style="list-style-type: none"> <li>Non-ASCII Character</li> </ul>

- **Boolean recordMediaAttributes( FileIdentifier fileIdentifier, Map<String, String> attributes )**
  - Examples of attributes are “year”, “date”, and “city”

Input	Valid Equivalence class	Invalid Equivalence class
<b>Year</b>	<ul style="list-style-type: none"> <li>• Contains number</li> <li>• Contains 4 numeric character</li> </ul>	<ul style="list-style-type: none"> <li>• Contains Upper case letter</li> <li>• Contains Lower case Letter</li> <li>• Contains Special character</li> <li>• Contains less than 4 numeric characters</li> <li>• Contains more than 4 numeric characters</li> </ul>
<b>Date</b>	<ul style="list-style-type: none"> <li>• Contains numbers</li> </ul>	<ul style="list-style-type: none"> <li>• Contains special character</li> <li>• Contains Upper case letter</li> <li>• Contains Lower case Letter</li> <li>• Non-ASCII Character</li> </ul>
<b>City</b>	<ul style="list-style-type: none"> <li>• Contains lower case letter</li> <li>• Contains Upper case letter</li> <li>• String length between 3 – 15 characters</li> </ul>	<ul style="list-style-type: none"> <li>• Contains special character</li> <li>• String length &gt; 15</li> <li>• Non-ASCII Character</li> </ul>

- **Boolean tagMedia( FileIdentifier, fileIdentifier, String tag )**

Input	Valid Equivalence class	Invalid Equivalence class
<b>tag</b>	<ul style="list-style-type: none"> <li>• Contains lower case letter</li> <li>• Contains Upper case letter</li> <li>• Contains numbers</li> <li>• Contains Special Character</li> </ul>	<ul style="list-style-type: none"> <li>• Non-ASCII Character</li> </ul>

## 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

- o Gender must be Male or Female

Occupation

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

Examples of attributes are “year”, “date”, and “city”

date

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

City

- 1 character city name

- **Boolean tagMedia( FileIdentifier, fileIdentifier, String tag )**
  - 1 character tag

### 3. Decision Table Testing

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

