INDIAN INSTITUTE OF INFORMATION TECHNOLOGY NAGPUR

Software Architecture Mini Project

Emotion Based Music Player

Raj Aryan	BT19CSE043
Sarthak Gupta	BT19CSE054
Aditi Sahu	BT19CSE057
Aseem Ranjan	BT19CSE085
Krish Rustagi	BT19CSE089
Naveen Rathore	BT19CSE117

Introduction

Emotion based music player is an approach that helps the user to automatically play songs according to the emotions of the user. It recognizes the facial emotions of the user and plays the songs according to their emotion. The emotions are recognized using machine learning algorithms. The human face is an important organ of an individual ‘s body and it especially plays an important role in extraction of an individual ‘s behaviours and emotional state. The camera in the user's device captures the image of the user. It then extracts the facial features of the user from the captured image. Facial expression is categorized into four: joy, sadness, anger and fear.

According to the emotion, the music will be played from the predefined directories.

Specific Requirements

The system will be composed of server-side components and client-side components. The server-side component will manage the database operations (managing and categorizing songs) and algorithms that produce recommendation playlists. The client-side components will be graphical interfaces that are integrated into corresponding larger systems.

The following section will describe the software requirements in detail as subsections which are interface requirements, functional and non-functional requirements.

- **Interface Requirements**

  Since there is just one type of user, the application will have only the user interface. Basically, the user interface will direct the user through steps and will display the recommendations found by our algorithm.

Recommendation Subsystem

This Subsystem is the core of the project, because our algorithm will work in the background of it. This interface will be used by both customers and suppliers. Customers cannot do many operations, but their feedback is very important to create a relevant recommendation system. Users can only provide feedback operations. Suppliers can do more operations, and these are to index new items (more songs) into the songs database collection, monitor user feedbacks, view recommendations. View recommendations operation can be used after our algorithm gives the result.

- **Functional Requirements**

  In this section, we will explain the major functions of the Recommendation System.

Client

The client-side of the system will be an application with a user interface that is integrated into a music listening website or application. This application gathers the information from users, captures images of the users, and provides the connection with the server. This application is the client-side interface of the Music Recommender, so it does not include the functionalities of the host music environment such as playing music etc.

- **Observing User**

The system will interact with the user through the camera, and send these obtained back to the server.

- **Display Recommendations**

The application must display the recommendations that are obtained from the server to the user and details of the current song being played in a proper way by providing a GUI.

- **Requesting Recommendations**

After gathering information about the mood of the user, the client-side application will ask the user to approve the mood which has been detected by our algorithms. If the user disagrees with results, the user is allowed to request recommendations manually, and interact with the server to receive recommendations.

Server

The server-side system will hold the entire song collection, and must include all functionality to perform operations on this database, receive requests from the clients, add/remove songs, create and send playlist recommendations etc.

- **Handle Recommendation Requests**

The server application shall obtain and handle requests based on mood for recommendations.

- **Data Storing**

The server application shall be able to store the song database and also allow to add/remove songs and make changes to the playlist recommendations.

Non-Functional Requirements

The non-functional requirements of the system are explained below as performance requirements.

Performance Requirements

➢ **Accuracy**

Since we will give priority to the accuracy of the software, the performance of the Music Recommender will be based on its accuracy on recommendations.

➢ **Failure Handling**

System components may fail independently of others. Therefore, system components must be built so they can handle failure of other components they depend on.

➢ **Security**

Since the application will be capturing images of the user and collecting data about the behaviour of the user, sensitive information should be kept safe.

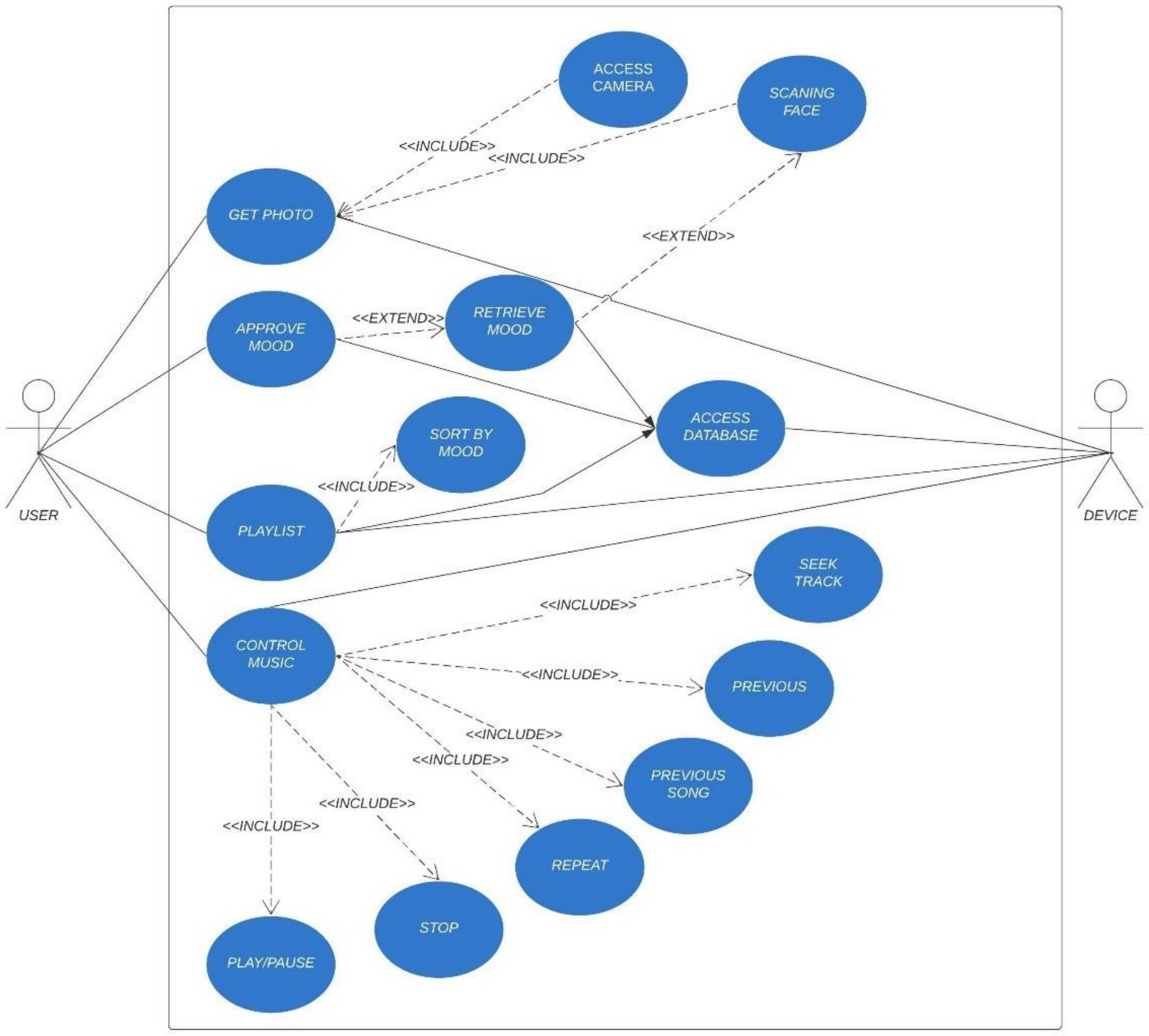
System Design and UML Diagrams

The system architecture of the Emotion based music system majorly comprise of three modules:

- Emotion Module for detecting the emotion of the user.
- Music Classification Module to segregate songs into different four mood classes.
- Recommendation Module to suggest appropriate playlist to the user according to the mood detected by the emotion module.

The system design and flow of the system is being depicted through the following UML diagram.

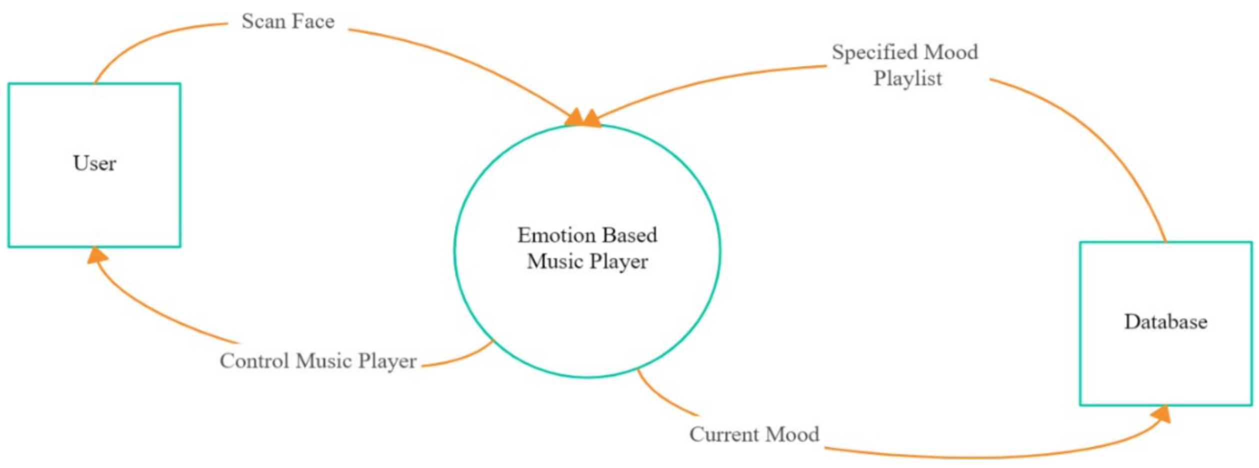
- **Use Case Diagram:** Use case diagram of the recommendation system and the other subsystems are revealed in the diagram below.



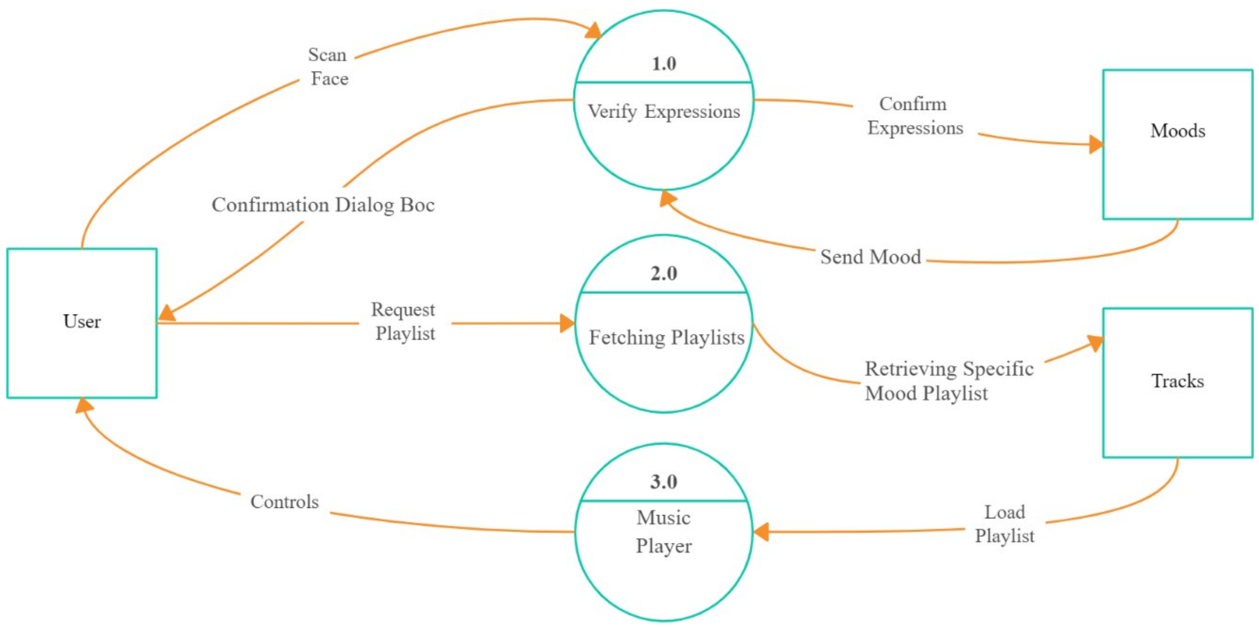
Data Flow Diagram

This section illustrates the design and functional phase of the application. The user can access their customized play-lists and play songs based on their emotions. The following illustrations depicts the overview of the data flow of the application.

- **Level 0 (Context Level)**

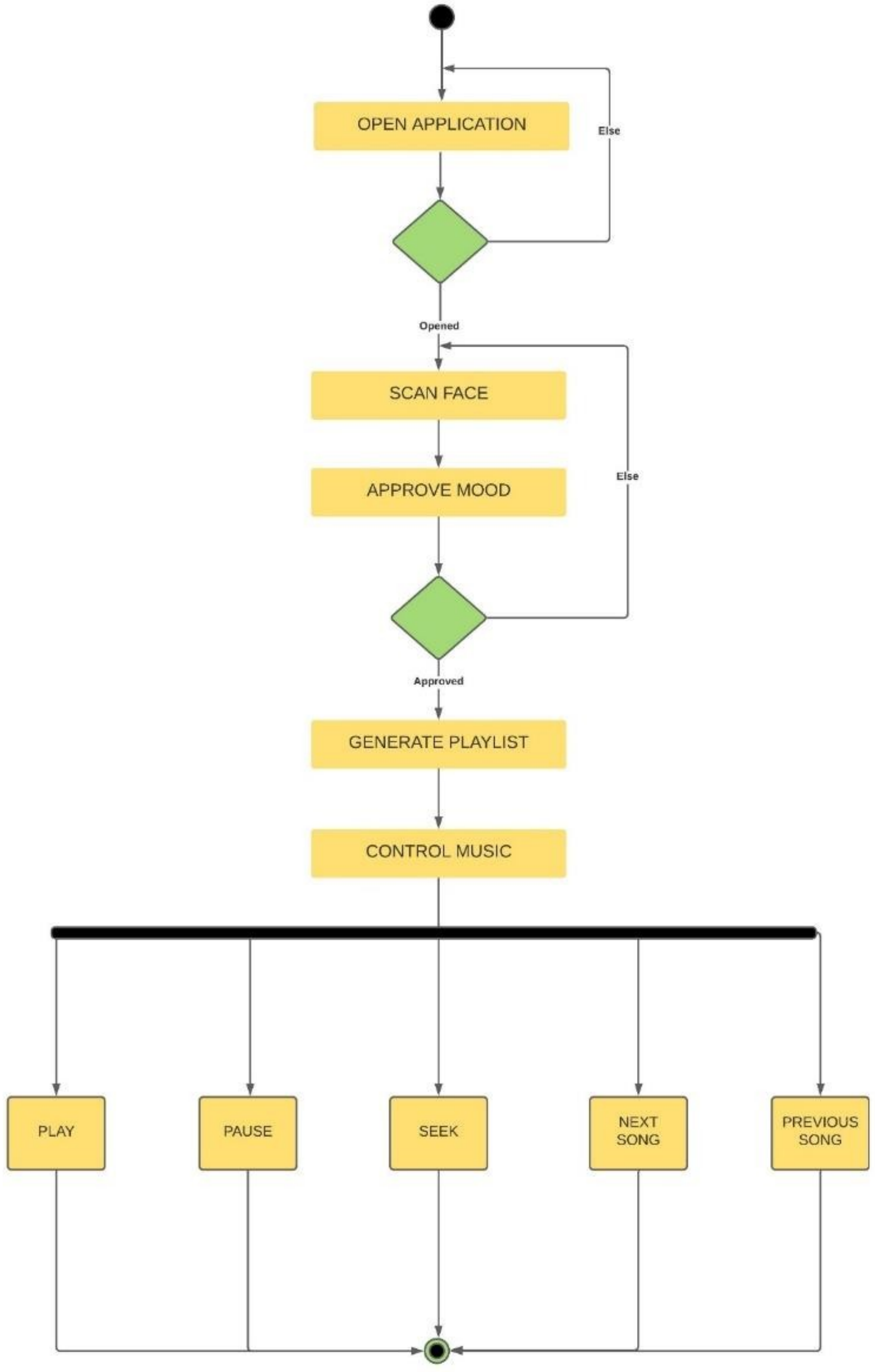


- **Level 1**



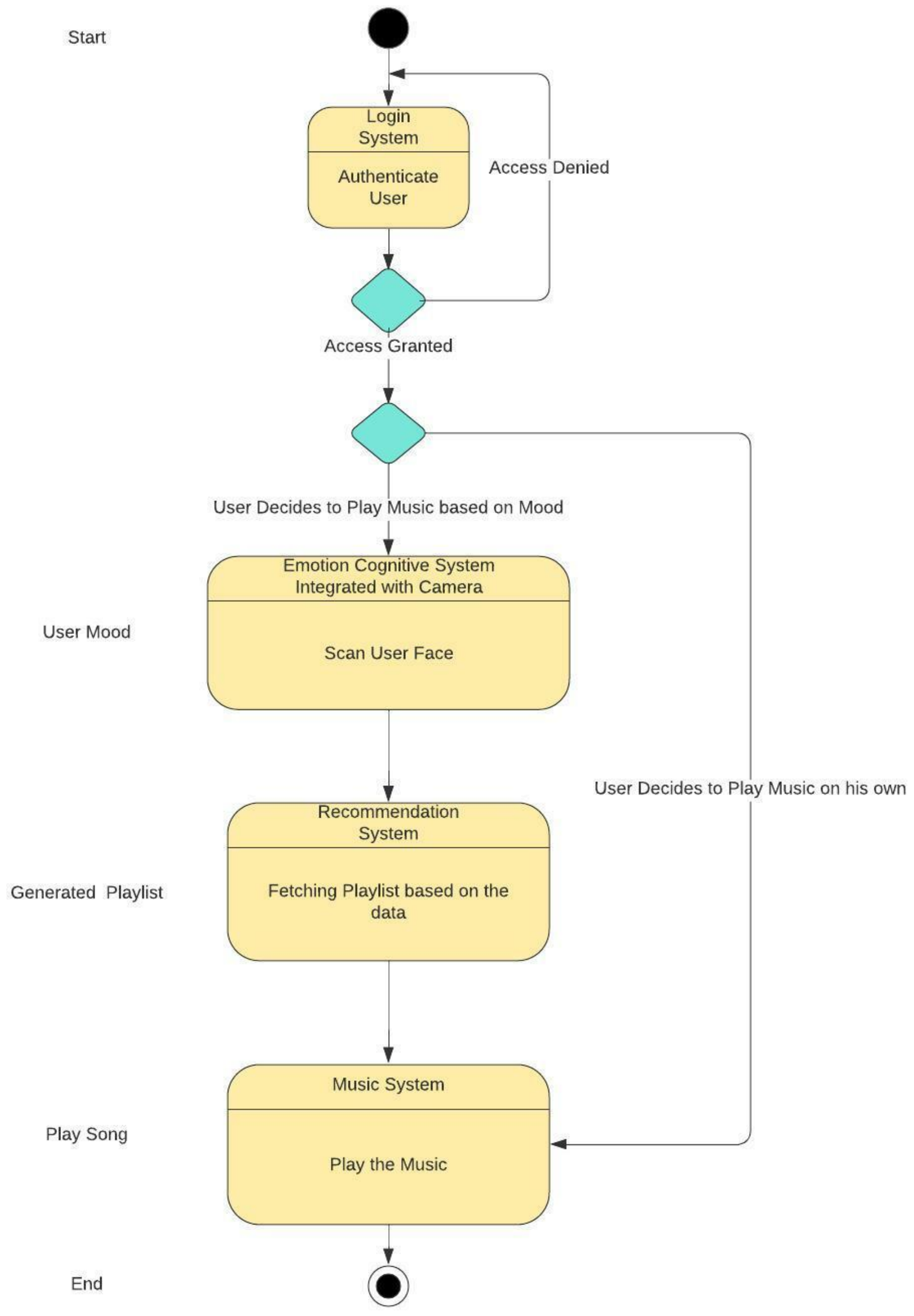
Activity Diagram

The illustration below depicts the overall system flow of the application.



State Diagram

The following diagram depicts the transition between several states that the application goes through during the entire process.



Quality Attributes

- **Performance Scenario:**
  → Song loading with no delay after emotion detection.
  → Emotion detection taking an average latency of 2 secs.

Portion of Scenario	Possible Values
User	General Public
Stimulus	Song load; Emotion detection
Environment	Normal
Artifact	Emotion detection system; song loader
Response	Song play with no delay
Response Measure	Average latency of 2 seconds in emotion detection; No delay in song loading

- **Usability Scenario**
  → Go through of the Application when the user registers First time.
  → User should not have to login each time (Session management)
  → Minimization of number of clicks required to play the music
  → User should have option to cancel the detection of emotion even in the process
  → Minimization of time taken in fetching the data (use of cache and preferences)
  → Use of symbols from existing music gadgets
  → Notification System or Mini Player System for play control
  → Shuffle and repeat options for user preferences
  → Shortcut keys should be valid for the application (play pause)
  → Use of async functions in the implementation such that app does not become unresponsive
  → UI should be dynamic and should not feel unresponsive (use of progress bars while processing)
  → Play buttons and options should be grouped together
  → Contact Form for feedback for user to connect

Portion of Scenario	Possible Values
User	End User
Stimulus	To use the system efficiently and adapt and configure the system

Environment	Runtime
Artifact	Emotion detection system and Song System with which user is interacting
Response	System to anticipate required or needed features for end users
Response Measure	Satisfaction of User, Process Times ,Number of system tasks completed

- **Availability Scenario:**

Portion of Scenario	**Possible Values**
User	Music Player Monitor
Stimulus	System errors, infrastructure problems, malicious attacks, and system load.
Artifact	Processor: Intel Core i3 Persistent storage: 4 GB RAM Communication channels: Webcam, Speakers
Environment	Normal operation

Response	Prevention from any fault Detect: ● Log the fault ● Notify to appropriate entities Recover: ● Disable the fault causing events ● Become temporarily unavailable
Response Measure	Time interval when the system must be available Time to detect and repair the fault Time in which system is in degraded mode.

- **Security Scenario**

→ Security is a measure of the system's ability to resist unauthorized usage while still providing its services to legitimate users.
→ Security can be characterized as a system providing confidentiality, assurance, nonrepudiation, availability, integrity and authorization.

- ➢ Confidentiality – privacy, no unauthorized access
- ➢ Assurance – the parties to a transaction are who they purport to be.
- ➢ Non-repudiation – cannot deny existence of executed transaction
- ➢ Availability – no denial of service
- ➢ Integrity – information and services delivered as intended and expected
- ➢ Authorization – grant users privileges to perform tasks

Portion of Scenario	**Possible Values**
User	Authorized user
Stimulus	Reduce availability to system services. Tries to display data. Access system services.
Artifact	Data within the system

Environment	Online
Response	Blocks access to data and/or services. Allows access to data and/or services. Grants or withdraws permission to access data and/or services. Encryption Logging
Response Measure	Time/effort/resources required to circumvent security measures with probability of success.

- **Modifiability Scenario:**

→ Improve web-interface in < 4person-weeks.
→ Improve test cases scenarios to reduce bugs.

Portion of Scenario	**Possible Values**
User	Administrator
Stimulus	Web interface
Environment	Online
Artifact	1. Add test cases 2. Change the web interface to interact with the user.
Response	1. Require regular changes in the web interface 2. Reducing bugs
Response Measure	Attain 0 crashes;