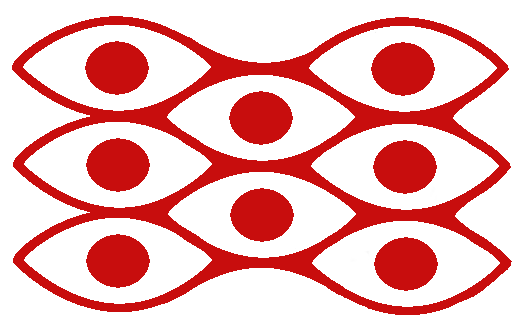
Department of Computer Science and Engineering  
The University of Texas at Arlington



Team: Team Argus

Project: Lynx- PixelSense Secure Transfer System

System Test Plan

Team Members:   
*Shamikul Amin*

*Brandon Deen*

*Brian Hasty*

*Keyur Patel*

*Seth Skocelas*

# Table of Contents

[Table of Contents 2](#_Toc415129419)

[Document Revision History 4](#_Toc415129420)

[Table of Figures 5](#_Toc415129421)

[List of Tables 6](#_Toc415129422)

[1. Introduction 7](#_Toc415129423)

[1.1 Document Overview 7](#_Toc415129424)

[1.2 Product Overview 7](#_Toc415129425)

[1.3 Project Scope 7](#_Toc415129426)

[2. Test References 11](#_Toc415129427)

[2.1 System Requirements Specification 11](#_Toc415129428)

[2.2 Architectural Design Specification 19](#_Toc415129429)

[PixelSense Super Layer 19](#_Toc415129430)

[Tablet/Lynx Super Layer 20](#_Toc415129431)

[Transfer Super Layer 20](#_Toc415129432)

[2.3 Detailed Design Specification 25](#_Toc415129433)

[3. Test Items 34](#_Toc415129434)

[3.1 Hardware Tests 34](#_Toc415129435)

[3.2 Unit Tests 35](#_Toc415129436)

[3.3 Component Tests 42](#_Toc415129437)

[3.4 Integration Tests 46](#_Toc415129438)

[3.5 System Validation 47](#_Toc415129439)

[4. Risks 48](#_Toc415129440)

[4.1 Risk Table 48](#_Toc415129441)

[5. Features To Be Tested 49](#_Toc415129442)

[5.1 Customer Requirements 49](#_Toc415129443)

[5.2 Packaging Requirements 53](#_Toc415129444)

[5.3 Performance Requirements 54](#_Toc415129445)

[5.4 Safety Requirements 55](#_Toc415129446)

[5.5 Maintenance and Support Requirements 56](#_Toc415129447)

[5.6 Other Requirements 57](#_Toc415129448)

[6. Features Not To Be Tested 58](#_Toc415129449)

[6.1 Customer Requirements 58](#_Toc415129450)

[7. Testing Approaches 59](#_Toc415129451)

[7.1 Strategy 59](#_Toc415129452)

[7.2 Metrics 60](#_Toc415129453)

[8. Item Pass/Fail Criteria 61](#_Toc415129454)

[8.1 Hardware Tests 61](#_Toc415129455)

[8.2 Unit Tests 61](#_Toc415129456)

[8.3 Component Tests 64](#_Toc415129457)

[8.4 Integration Tests 68](#_Toc415129458)

[8.5 Validation Tests 69](#_Toc415129459)

[9. Test Deliverables 70](#_Toc415129460)

[9.1 Deliverables 70](#_Toc415129461)

[10. Test Schedule 72](#_Toc415129462)

[11. Approval 73](#_Toc415129463)

# Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Revision Date | Description | Rationale |
| 1.0 |  | System Test Plan Draft | Initial Creation of the Document |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Table of Figures

[Figure 1‑1: System Test Diagram 10](#_Toc415158345)

[Figure 2‑1: Layer Definitions 19](#_Toc415158346)

[Figure 2‑2: Architecture Diagram 22](#_Toc415158347)

[Figure 2‑5: Detailed Design Diagram 25](#_Toc415158348)

# List of Tables

[Table 2‑1: Customer Requirements 12](#_Toc415158351)

[Table 2‑2: SDK Requirements 14](#_Toc415158352)

[Table 2‑3: Packaging Requirements 15](#_Toc415158353)

[Table 2‑4: Performance Requirements 16](#_Toc415158354)

[Table 2‑5: Safety Requirements 17](#_Toc415158355)

[Table 2‑6: Maintenance and Support Requirements 18](#_Toc415158356)

[Table 2‑7: Other Requirements 18](#_Toc415158357)

[Table 2‑8: Producer-Consumer Relationship (ADS) 23](#_Toc415158358)

[Table 2‑9: Requirements Traceability Matrix (ADS) 24](#_Toc415158359)

[Figure 2‑10: Producer-Consumer Matrix (DDS) 32](#_Toc415158360)

[Figure 2‑11: Requirements Traceability Matrix (DDS) 33](#_Toc415158361)

[Table 3‑1: Hardware Tests 35](#_Toc415158362)

[Table 3‑2: PixelSense App UI Unit Tests 35](#_Toc415158363)

[Table 3‑3: PixelSense Event Handler Unit Tests 35](#_Toc415158364)

[Table 3‑4: PixelSense Data Formatter Unit Tests 36](#_Toc415158365)

[Table 3‑5: PixelSense Data Analyzer Unit Tests 36](#_Toc415158366)

[Table 3‑6: PixelSense Graphical Processing Unit Tests 36](#_Toc415158367)

[Table 3‑7: PixelSense Computational Processing Unit Tests 37](#_Toc415158368)

[Table 3‑8: PixelSense Transfer Manager Unit Tests 37](#_Toc415158369)

[Table 3‑9: IR Receiver Unit Tests 37](#_Toc415158370)

[Table 3‑10: PixelSense Cluster Emitter Unit Tests 38](#_Toc415158371)

[Table 3‑11: PixelSense Transfer Formatter Unit Tests 38](#_Toc415158372)

[Table 3‑12: LED Receiver Unit Tests 38](#_Toc415158373)

[Table 3‑13: LED Transmitter Unit Tests 38](#_Toc415158374)

[Table 3‑14: Lynx Transfer Data Formatter Unit Tests 39](#_Toc415158375)

[Table 3‑15: Lynx Transfer Manager Unit Tests 39](#_Toc415158376)

[Table 3‑16: Android App UI Unit Tests 39](#_Toc415158377)

[Table 3‑17: Tablet Event Handler Unit Tests 40](#_Toc415158378)

[Table 3‑18: Tablet Presentation Data Formatter Unit Tests 40](#_Toc415158379)

[Table 3‑19: Tablet Data Analyzer Unit Tests 40](#_Toc415158380)

[Table 3‑20: Tablet Graphical Data Processing Unit Tests 41](#_Toc415158381)

[Table 3‑21: Tablet Computational Data Processing Unit Tests 41](#_Toc415158382)

[Table 3‑22: Request Manager Unit Tests 42](#_Toc415158383)

[Table 3‑23: Storage Converter Unit Tests 42](#_Toc415158384)

[Table 3‑24: Storage Manager Unit Tests 42](#_Toc415158385)

[Table 3‑25: PixelSense Presentation Layer Component Tests 43](#_Toc415158386)

[Table 3‑26: PixelSense Processing Layer Component Tests 44](#_Toc415158387)

[Table 3‑27: SDK Layer Component Tests 44](#_Toc415158388)

[Table 3‑28: PixelSense Transfer Layer Component Tests 45](#_Toc415158389)

[Table 3‑29: Lynx Transfer Layer Component Tests 45](#_Toc415158390)

[Table 3‑30: Tablet Presentation Layer Component Tests 46](#_Toc415158391)

[Table 3‑31: Tablet Data Processing Layer Component Tests 46](#_Toc415158392)

[Table 3‑32: Data Storage Component Tests 47](#_Toc415158393)

[Table 3‑33: PixelSense Super Layer Integration Tests 47](#_Toc415158394)

[Table 3‑34: SDK Super Layer Integration Tests 48](#_Toc415158395)

[Table 3‑35: Lynx Super Layer Integration Tests 49](#_Toc415158396)

[Table 3‑36: System Validation Tests 49](#_Toc415158397)

[Table 4‑1: Risk Table 50](#_Toc415158398)

[Table 7‑1: Test Metrics 62](#_Toc415158399)

[Table 8‑1: Hardware Test Criteria 63](#_Toc415158400)

[Table 8‑2: Unit Test Criteria 66](#_Toc415158401)

[Table 8‑3: Component Tests 70](#_Toc415158402)

[Table 8‑4: Integration Test Criteria 71](#_Toc415158403)

[Table 8‑5: Validation Test Criteria 71](#_Toc415158404)

[Table 10‑1: Test Schedule 74](#_Toc415158405)

[Table 11‑1: Approvals 75](#_Toc415158406)

# 1. Introduction

## 1.1 Document Overview

The System Test Plan is intended to describe the details of how the Lynx-PixelSense Secure Transfer System will be tested in order to ensure the delivery of a quality product. The testing plan will be detailed and shall include: unit, component, and integration tests. Any testing considerations from previous documents created, such as System Requirements, Architecture Design, or Detailed Design, shall be included and covered in greater detail.

## 1.2 Product Overview

The PixelSense Secure Transfer System is a tablet case with a series of sensors allowing secure transmission between the PixelSense table and the device (the Lynx) connected to the tablet. The benefit of this transfer method is that communication cannot be intercepted due to physical limitation, thus making the connection secure. When the Lynx is placed on the table with compatible software loaded, the table will be able to recognize the Lynx, and be able to transfer data when needed. Included with the Lynx will be an SDK that will allow users to develop their own applications for the both the tablet and the PixelSense table in order to use it however they see fit. With the demonstration application we’re developing for the table, a casino game, the Lynx will provide a way to securely store casino chips, and also be used as a secondary screen when playing the game. For example, if you were playing poker, you can display the cards on the tablet where no other player can see them as opposed to showing them on the table.

Since we will be developing an SDK for the Lynx, it can be used for many other purposes in tandem with the PixelSense table. Below are some ideas that could be implemented using the SDK we create:

* A health information tracker that stores information on the device, and can only be displayed when the device is placed on the PixelSense table. A doctor can then add or remove information using the table as he/she sees fit.
* An enhanced chess game where the transfer device serves as a chess piece that can store information regarding player habits.
* A degree plan tracker that can store advising information about a student, and when the device is placed on the table, the student’s degree plan and academic information would be displayed for the advisor, which the advisor could edit as he/she sees fit.

## 1.3 Project Scope

For this project, we’ll be delivering the Lynx, a device that connects to the an Android 4.0 tablet in order to transfer small amounts of data optically between the tablet and the PixelSense table it is placed on, provided that compatible software is loaded on both devices. Included with the Lynx will be an SDK that will allow the users to develop their own applications for the both the tablet and the PixelSense table in order to use it however they see fit.

For demonstration purposes, our group will be developing a casino game on the PixelSense table, and our secure transfer system attached to a tablet will serve a poker chip counter, securely storing your chips so you can move from table to table in order to play different casino games.

## 1.4 System Test Plan

The diagram below shows how the system has been divided into separate testing modules. Hardware, unit, component, integration, and system validation testing and the layers, sub layers, and modules those tests will encompass.

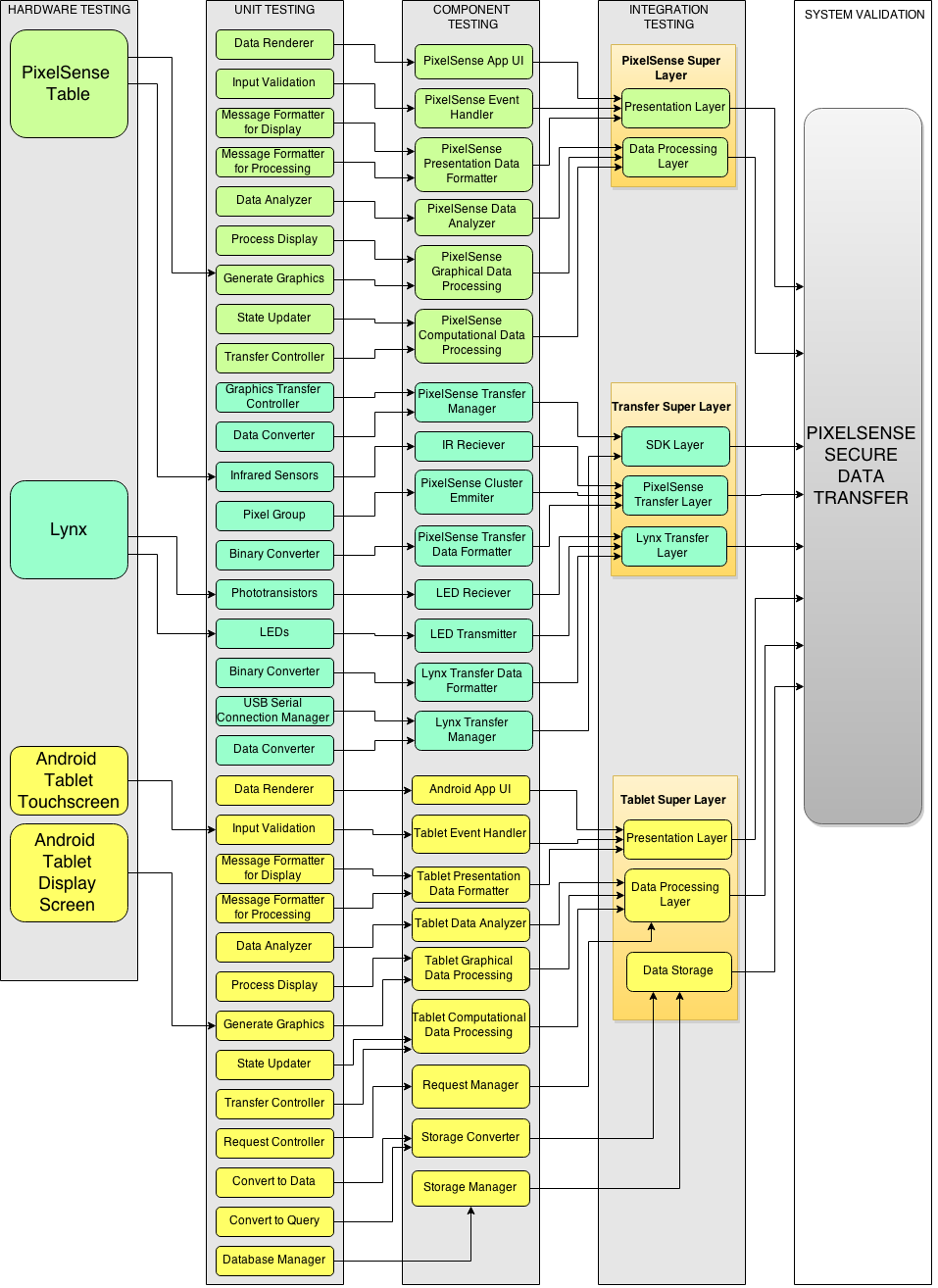


Figure 1‑1: System Test Diagram

# 2. Test References

This section provides the references used to develop the test plan. The documents referenced include: System Requirements Specification, Architecture Design Specification and Detailed Design Specification. In the creation of these documents, the requirements, layers and modules that need to be tested were also created.

## 2.1 System Requirements Specification

The System Requirements Specification details the requirements of the Lynx-PixelSense Secure Transfer system. This section will list all of the requirements, note that not all requirements need to be tested and some cannot be tested.

### 2.1.1 Customer Requirements

The following section contains the customer requirements agreed upon by Team Argus and Dr. Gergely Zaruba. These requirements cover the creation of the device, the SDK, and the casino software associated with this project.

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Requirement** | **Description** | **Priority** |
| 3.1 | The product shall be able to send data optically. | The product must be able to send specifically timed flashes of light as bits of information and send them in a way a receiving unit can interpret it. | Very High  5 |
| 3.2 | The product shall be able to read data optically | The product must be able to receive specifically timed flashes of light as bits of information and interpret them correctly, storing the data if necessary. | Very High  5 |
| 3.3 | All work done by the product involving optical communication will be compiled into a well-documented library. | A suite of functions designed to handle optical communication between capable hardware will be available in a portable SDK library. The details of this requirement is elaborated on in the next section. | Very High  5 |
| 3.4 | The device must have a serial port to transfer data collected | The Lynx will have a serial port to transfer data optically collected to the Android device connected. |  |
| 3.5 | The device and table’s communications should be error tolerant | All data sent should be received intact, and be identical to the original message. The Lynx should detect errors in information received. | High  4 |
| 3.6 | Communication between both devices must sustain a reliable connection | All data sent should be received intact, and be identical to the original message. The protocol will have a system to detect missing or corrupt data, and replace the missing pieces with the correct information. | High  4 |
| 3.7 | Software will be built to showcase the optical transfer protocol and the built device | A program will be supplied to demonstrate the capabilities of the optical communication SDK, and Lynx. The program will play a game of Blackjack, with the device holding chip information. | Very High  5 |

Table 2‑1: Customer Requirements

### 2.1.2 SDK Requirements

The following section contains the customer requirements agreed upon by Team Argus and Dr. Gergely Zaruba that cover the creation of the SDK associated with this project. Unless specified otherwise, the functions mentioned in these requirements will be a part of all three subsystems of our project. (The Lynx device, the PixelSense table, and Android platform)

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Requirement** | **Description** | **Priority** |
| 4.1 | The system will support Android API 17 and Surface 2.0 platforms | An SDK will be provided for both Android API 17 (4.2 and up) and Microsoft Surface 2.0 (PixelSense) platforms. An SDK will also be provided for the Lynx device itself. | Very High  5 |
| 4.2 | The System will support customizable light sequences | The Device, Android and PixelSense SDK will provide functions that gives ability to customize the sequence of light transmitted and to set what happens when those light sequences occurs. | Moderate  3 |
| 4.3 | The system shall allow encoding of data | The SDK will provide function(s) that allow the user to encode the data being transmitted. | High  4 |
| 4.4 | The system shall allow the expansion of the array | The Device and PixelSense SDK functions provided will allow the user to set the size of the array of transmitters they want to use and support that functionality | Low  2 |
| 4.5 | The system shall support a 4x4 array of transmitters and receivers | The Device and PixelSense SDK will provide support for a 4x4 array to transmit and receive data to and from the Surface table. | High  4 |
| 4.6 | The system shall authenticate the Lynx device when plugged into the tablet | The Android and Device SDK will provide function(s) that will allow the Android application to authenticate that a valid Lynx device is plugged into the Android tablet. | High  4 |
| 4.7 | The system shall allow the Lynx to send and receive data | The Device and Android SDK will provide functions that will allow the Android application developed with it to send and receive data using the Lynx device. | Very High  5 |
| 4.8 | The system shall allow the PixelSense to send and receive data | The PixelSense SDK will provide functions that will allow the PixelSense application developed with it to send to and receive data from the Lynx device. | Very High  5 |
| 4.9 | The system shall be able to determine the orientation of the Lynx | The PixelSense SDK functions will provide the ability to determine how the Lynx is oriented on the Surface table. These functions can leverage existing functionality present in the development kit for PixelSense. | Very High  5 |
| 4.10 | The system shall notify the Android device if the Lynx is on the table | The Device and Android SDK will provide information to the user via the Android tablet connected to the Lynx when the Lynx is on the Surface table and ready to transmit or receive data. | Very High  5 |
| 4.11 | The system shall notify the table that the Lynx is on it | The Device and PixelSense SDK will provide information to the user via the Surface table that the Lynx is on the Surface table and ready to transmit or receive data. | Very High  5 |
| 4.12 | The system shall authenticate that a valid Lynx device is placed on the PixelSense table | The PixelSense SDK will provide functions to authenticate that a valid Lynx device is on the PixelSense table and is capable of sending and receiving data. | High  5 |
| 4.13 | The system shall support multiple devices | The PixelSense SDK will provide functions to detect and support multiple Lynx devices on the table | Moderate  3 |

Table 2‑2: SDK Requirements

### 2.1.3 Packaging Requirements

In this section, we’ll cover the requirements for how we will deliver our product to the customer. The Secure Transfer device will be a self-contained entity (the Lynx) already assembled for the user and ready for use. Since this will be used with a tablet, the Lynx will need to be able to attach/detach from the tablet. Our SDK will be loaded on a CD, which can be used on Windows computers to develop for both Android and Surface SDKs. The Casino software we develop will be available on a USB drive to be loaded on a PixelSense Table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Requirement** | **Description** | **Priority** |
| 5.1 | Attachable to tablet | The Lynx must be attachable and detachable physically from the tablet it is connected to. This is referring to a mechanism so the Lynx can physically stay on the tablet it is connected to, not the serial connection with the cord plugged into the tablet. | Moderate  3 |
| 5.2 | Fully Assembled | The Lynx will be assembled and ready for use. | High  4 |
| 5.3 | Lynx as a tablet case | The Lynx will serve as a tablet case for the tablet we chose to develop for. | Low  2 |
| 5.4 | SDK on USB Flash Drive | The Lynx’ SDK for the PixelSense table, the Secure Transfer Device, and the connected device will be provided on a USB flash drive for use on Windows. | High  4 |
| 5.5 | PixelSense Casino Software | The PixelSense Casino Software will be provided on a USB Flash Drive and the software can be installed on the table. | High  4 |
| 5.6 | PC Casino Companion Software | The PC Casino Companion Software will be packaged as an executable file on an USB Flash Drive that can be used on compatible Windows PCs. | High  4 |
| 5.7 | Android Casino Software | The Android Casino Software will be packaged in an APK on an USB Flash Drive that can be installed on compatible Android tablets. | High  4 |

Table 2‑3: Packaging Requirements

### 2.1.4 Performance Requirements

In this section we will go over the necessary performance requirements for the entire system as well as the individual components. The different aspects of this product mean that we will have different performance requirements over each component. The components that will be covered are as follows: optical transfer, data translation, connection times, authentication times, software boot times and data read/write times.

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Requirement** | **Description** | **Priority** |
| 6.1 | Minimum Data transfer rate | Data transfer must be consistent at minimum of 200 bits/sec | High  4 |
| 6.2 | Data translation | Converting optical data to binary data for later readability should be at most 1 second. | Moderate  3 |
| 6.3 | Authentication time | The Lynx should not take too long to authenticate its integrity with the PixelSense. Time should at most 5 seconds. | Moderate  3 |
| 6.4 | Software Boot times | Software should not take a long time to boot and load data. Ideally this should be less than 10 seconds. | High  4 |
| 6.5 | Data Read/Write times | Reading and writing data to the Lynx should be at most 1 second. | High  4 |
| 6.6 | Battery Life | Battery Life on the Lynx should last about half a casino day which is about 8 hours. | High  4 |
| 6.7 | Overall Connection Times | Connection times to the Lynx, tablet and DB should take no longer than 5 seconds. | Moderate  3 |

Table 2‑4: Performance Requirements

### 2.1.5 Safety Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Requirement** | **Description** | **Priority** |
| 7.1 | No sharp edges | The lynx must not have any sharp edges | High  4 |
| 7.2 | Protective Case Around Device | Lynx should have a protective case to prevent exposure to liquids | High  4 |

Table 2‑5: Safety Requirements

### 2.1.6 Maintenance and Support Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Requirement** | **Description** | **Priority** |
| 8.1 | User Manual | A user manual shall include a step by step guide to use the SDK with device & software on Surface Table. | High  4 |
| 8.2 | Software Installation | Software shall be provided to interact between device and PixelSense Table. Software shall not require additional permission to install software. Software shall work on any Windows 7 Operating System on the PixelSense Hardware. | Very High  5 |
| 8.3 | Source Code / SDK | Source code shall be provided to the customer as final deliverable. Basic Software Development Kit with usage guide shall be provided in order to use it further for PixelSense Development. | Moderate  3 |
| 8.4 | Hardware Support | Device shall be ready to use with PixelSense Secure transfer system. Device shall include Android Operating System with PixelSense Application. If required, customer can replace device with same application in future. | Very High  5 |

Table 2‑6: Maintenance and Support Requirements

### 2.1.7 Other Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Requirement** | **Description** | **Priority** |
| 9.1 | American English Standard | The PixelSense Secure Transfer shall use American English as the default language for any text or audio speech. | High  4 |
| 9.2 | User Friendly Interface | The PixelSense Secure Transfer shall have a user interface that can be learned in under 2 hours, and includes guidance to aid the user during use of the application. | High  4 |

Table 2‑7: Other Requirements

## 2.2 Architectural Design Specification

The purpose of this section is to detail the architecture design for Lynx, the PixelSense Secure Transfer System. This document will describe in detail the meta-architecture, layers and subsystems of the all the components of the system, showing the guiding principles we will use to drive this project. We will also detail any OS dependencies as well as the testing considerations we have made while describing this architecture.

### 2.2.1 Layer Overview

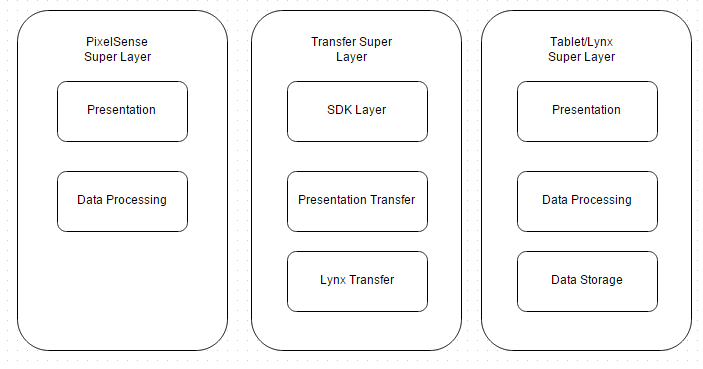


Figure 2‑1: Layer Definitions

### 2.2.2 PixelSense Super Layer

The purpose of this super Layer is to serve the applications on PixelSense Table. The main function is to get the data from the user on the PixelSense table and process it to forward it to transfer layer. This super layer also serves the purpose of displaying the data back to user after it is being transferred back from different super layer. Having a super layer is useful showing the different parts of both systems as well as showing them as one system as a whole.

#### 2.2.2.1 Presentation Layer

The purpose of the PixelSense presentation layer is to present some form of information to the user. It also accepts input from the user which would trigger the corresponding events. This layer level will serve as main source of input from the user to the system as well as presentation to the user from the system. Information sent to this layer is initially formatted and then transferred for the processing. Similarly, information received by this device is formatted for the presentation

#### 2.2.2.2 Data Processing Layer

The purpose of this layer is to accept the data from the Presentation Layer and process it. It also serves the purpose of sending data back to the presentation layer after processing. This Layer contains a data formatter which analyzes the incoming or outgoing data and formats according to the need of receiving layer. After analysis, if the data is non-graphical it is being processed and sent to PixelSense Transfer Manager which forwards data to transfer Layer.

### 2.2.3 Tablet/Lynx Super Layer

The purpose of this super layer is to enclose and distinguish between the three internal layers. The main functionality is to show the dataflow between the three internal layers, and between the Lynx and the tablet

#### 2.2.3.1 Presentation Layer

The purpose of this layer is to present the data on the Lynx to the user, and handle the user’s inputs. The layer will display the data on the Lynx, a log of recent transactions, and buttons the user can interact with via the tablet’s touch screen. Each interaction from the user is handled, formatted and transferred to the Data Processing Layer

#### 2.2.3.2 Data Processing Layer

The purpose of this layer is to process the data from the Presentation layer, the Data Storage Layer, and the Transfer Super Layer. This layer will accept an input of data from one of the previously stated layers, process that data, and then transfer it to the receiving layer of the transaction.

#### 2.2.3.3 Data Storage Layer

The purpose of this layer is to manage and store transaction logs. This layer will receive a request from the Data Processing layer, then send that request to the Data Storage Layer which will return the relevant information to the Data Processing Layer.

### 2.2.4 Transfer Super Layer

The purpose of this super layer is to enclose and distinguish between the two internal layers. The main function is to show the transfer between the PixelSense table and the tablet/lynx device as well as the dataflow between the two internal layers. Having a super layer is useful showing the different parts of both systems as well as showing them as one system as a whole

#### 2.2.4.1 PixelSense Transfer Layer

The purpose of this layer is to transfer (send/receive) strictly through the PixelSense side. Here the layer will receive data from the PixelSense data processing layer, format the data and emit it through some pixel cluster configuration. Going the other way, the table will detect certain light sequences through its IR receiver, format it to binary data and then send it off to the PixelSense data processing layer for interpretation.

#### 2.2.4.2 Lynx Transfer Layer

The purpose of this layer is to transfer (send/receive) strictly through the Lynx side. Here the layer will be receiving data from the tablet’s processing layer, it will format it and then emit it through a certain light sequence via LEDs. Going the other way, the Lynx will detect light sequences through LED light detection. It will then format it to binary data and transfer it to the tablet’s data processing layer.

#### 2.2.4.3 SDK Layer

The purpose of this layer is to convert the computational data from both the tablet side and PixelSense side to data to a protocol where is can be sent over optics as well as converting the data received to something the PixelSense and tablet will both understand

### 2.2.2 Subsystem Overview

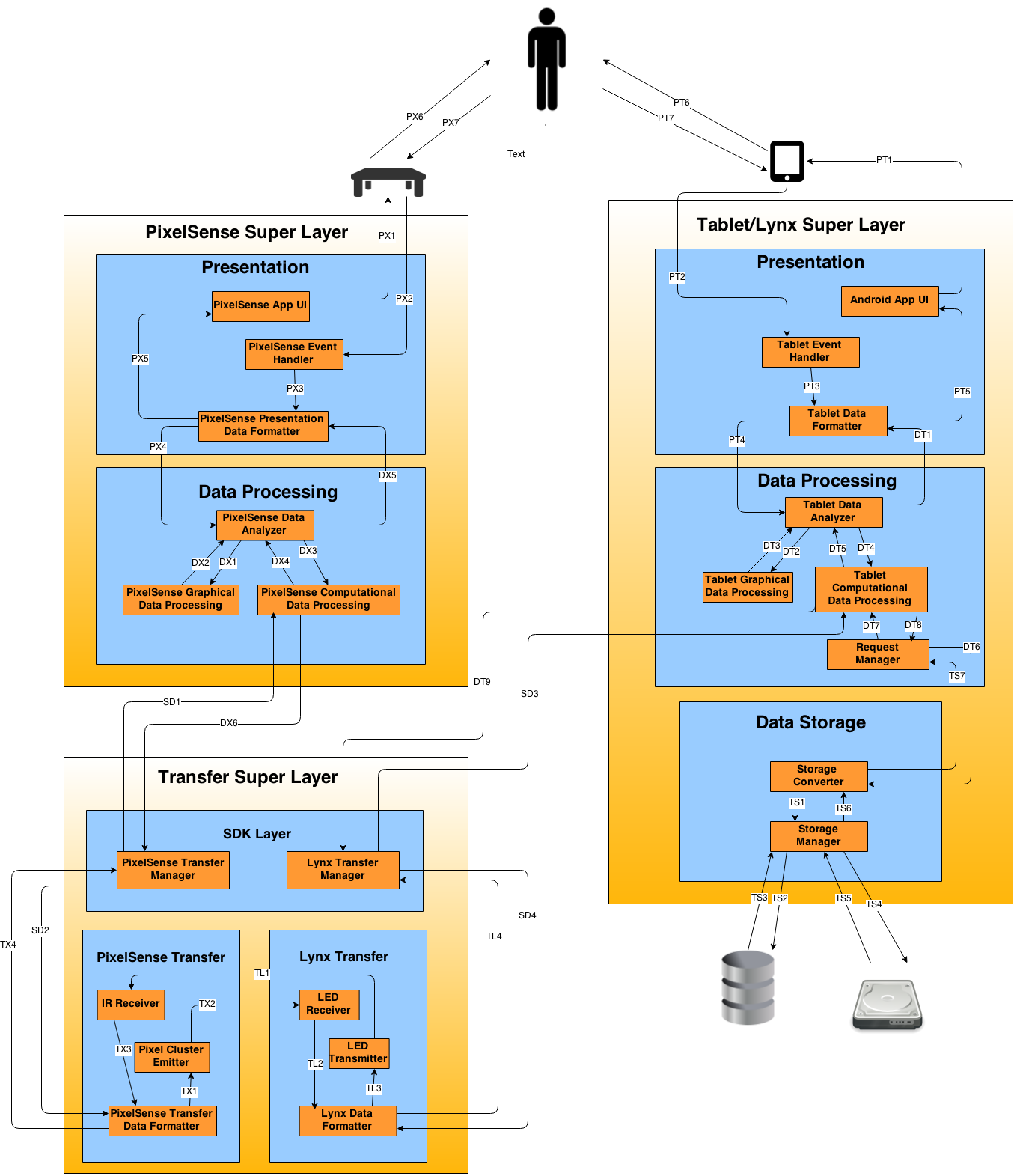


Figure 2‑2: Architecture Diagram

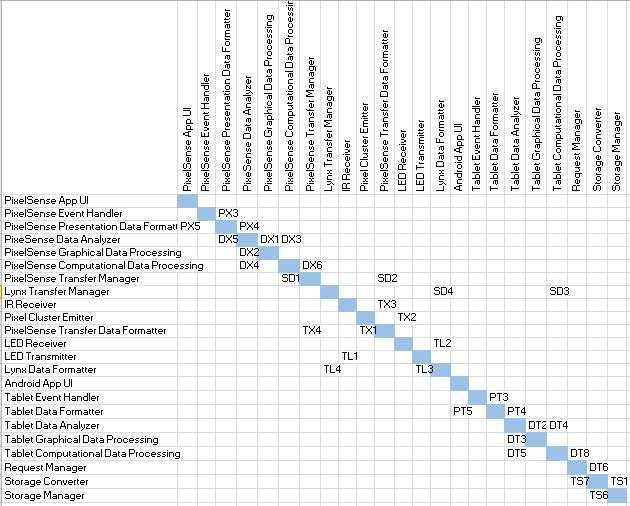


Table 2‑8: Producer-Consumer Relationship (ADS)

### 2.2.3 Requirements Mapping

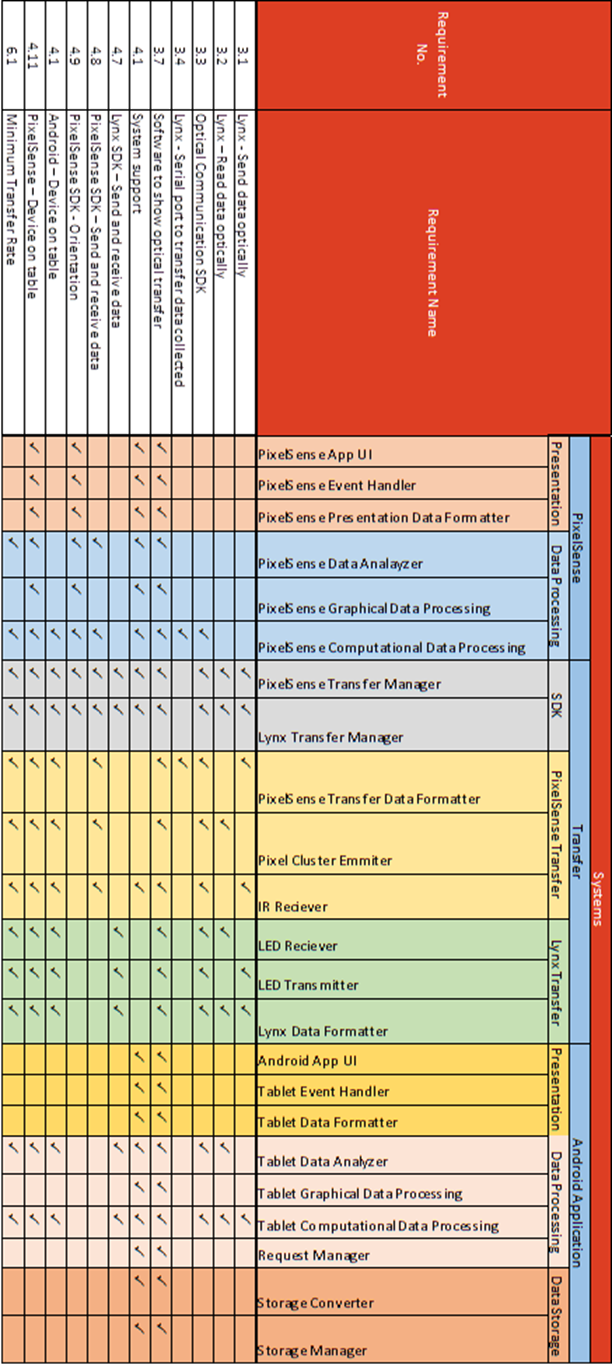


Table 2‑9: Requirements Traceability Matrix (ADS)

## 2.3 Detailed Design Specification

The purpose of this section is to go over the detailed design for the Lynx-PixelSense Secure Transfer System. This document will describe in detail the meta-architecture, layers and subsystems of all the components of the system, showing the guiding principles we will use to drive this project. We will also detail any OS dependencies as well as the testing considerations we have made while describing this architecture.

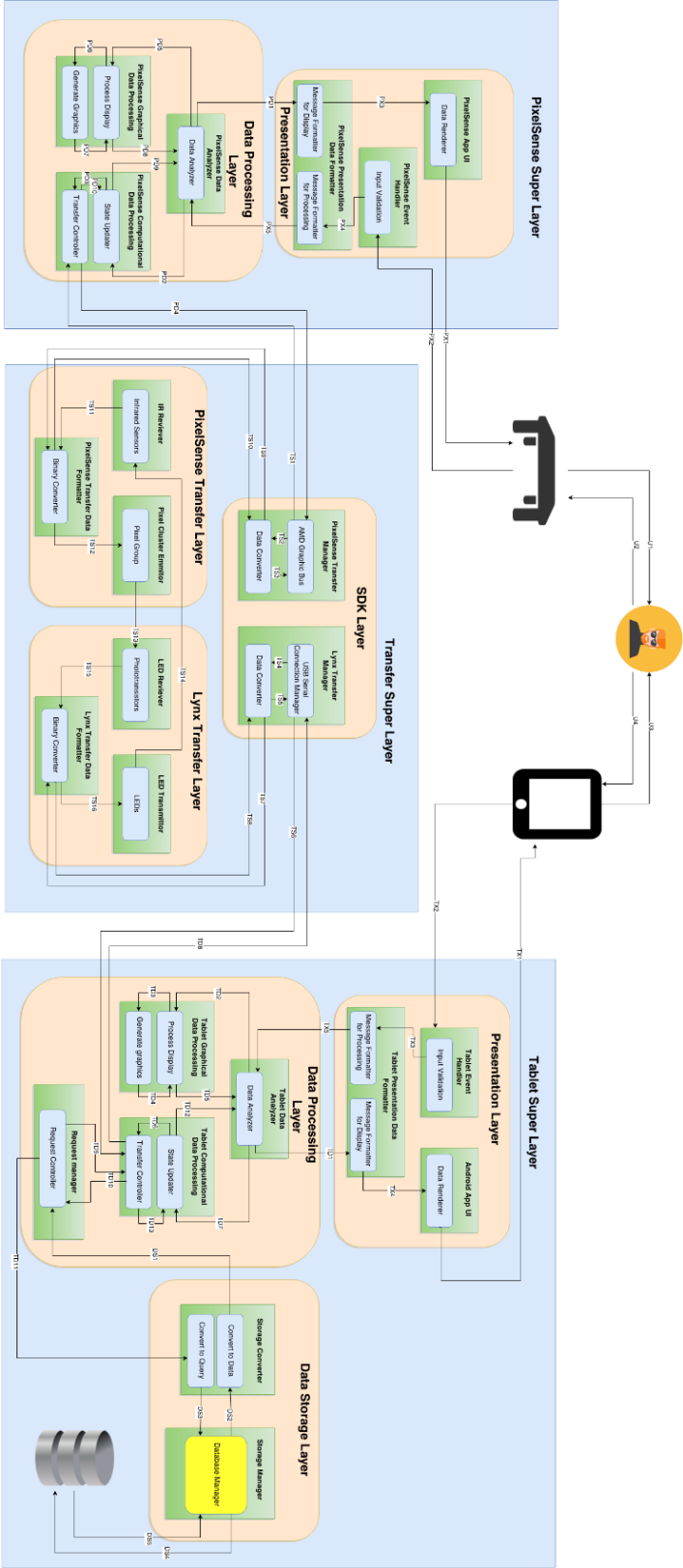


Figure 2‑5: Detailed Design Diagram

### 2.3.1 PixelSense Super Layer

#### 2.3.1.1 Presentation Layer

##### 2.3.1.1.1 PixelSense App UI

###### 2.3.1.1.1.1 Data Renderer

This module will render graphical as well as computational data content to PixelSense Table’s screen.

##### 2.3.1.1.2 PixelSense Event Handler

###### 2.3.1.1.2.1 Input Validation

Input Validation module is responsible for determining data type of the user input.

##### 2.3.1.1.3 PixelSense Presentation Data Formatter

###### 2.3.1.1.3.1 Message Formatter for Display

The Message Formatter for Display module will accept the analyzed data from Data Analyzer module, and it will format the data in such a way that it is ready for display on the table.

###### 2.3.1.1.3.2 Message Formatter for Processing

Message Formatter module for Processing will accept the validated input data from Event Handler subsystem, and it will format the data in such a way that it is ready for further processing. It then forwards formatted data to Data Analyzer for processing.

#### 2.3.1.2 Data Processing Layer

##### 2.3.1.2.1 PixelSense Data Analyzer

###### 2.3.1.2.1.1 Data Analyzer

Data Analyzer module interacts with Presentation layer. It receives formatted data from processing message formatter module, and identifies graphical and computational data. Both forms of data are then transferred to graphical and computational data formatter respectively. This module also sends back analyzed data to message formatter for display purposes.

##### 2.3.1.2.2 PixelSense Graphical Data Processing

###### 2.3.1.2.2.1 Process Display

Process Display module accepts graphical data from data analyzer module, and uses that data to create graphical content using the graphics generator module.

###### 2.3.1.2.2.2 Generate Graphics

Generate Graphics module is responsible for generating game related graphics using the graphical content like card faces, table cloth, chips images, etc.

##### 2.3.1.2.3 PixelSense Computational Data Processing

###### 2.3.1.2.3.1 State Updater

This module updates the state of input from the data analyzer, and makes it ready for the transfer controller to forward it to SDK layer. Data passed to this module is computational and no graphics are involved.

###### 2.3.1.2.3.2 Transfer Controller

This module is responsible for transferring computational data to the Graphics Transfer Controller inside SDK layer. This module also accepts the data coming from Lynx device and forwards it to State Updater for processing.

### 2.3.2 Tablet/Lynx Super Layer

#### 2.3.2.1 Presentation Layer

##### 2.3.2.1.1 Android App UI

###### 2.3.2.1.1.1 Data Renderer

This module will render data to be displayed to the user via the Tablet.

##### 2.3.2.1.2 Tablet Event Handler

###### 2.3.2.1.2.1Input Validation

This module will validate and interpret the inputs from the user, given through the Tablet’s touch screen.

##### 2.3.2.1.3 Tablet Data Formatter

###### 2.3.2.1.3.1 Message Formatter for Processing

This module formats data to be sent to the Data Processing Layer.

###### 2.3.2.1.3.2 Message Formatter for Display

This module formats data received from the data processing layer, to be displayed to the user.

#### 2.3.2.2 Data Processing Layer

##### 2.3.2.2.1 Tablet Data Analyzer

###### 2.3.2.2.1.1 Data Analyzer

This module analyzes the data it receives and determines which parts need graphical processing and which parts need general computational processing.

##### 2.3.2.2.2 Tablet Graphical Data Processing

###### 2.3.2.2.2.1 Process Display

This module processes data to be displayed on the tablet.

###### 2.3.2.2.2.2 Generate Graphics

This module generates the graphics to be displayed to the user on the tablet.

##### 2.3.2.2.3 Tablet Computational Data Processing

###### 2.3.2.2.3.1 State Updater

This module updates the state of the application, processes general data.

###### 2.3.2.2.3.2 Transfer Controller

This module transfers data to the Lynx (USB Serial Connection Manager), which will be sent to the table.

##### 2.3.2.2.4 Request Manager

###### 2.3.2.2.4.1 Request Controller

This module receives requests to store and retrieve data in the database.

#### 2.3.2.3 Storage Layer

##### 2.3.2.3.1 Storage Converter

###### 2.3.2.3.1.1 Convert to Data

This module converts the result of an executed query into relevant data as per the request received from the Request Controller Module.

###### 2.3.2.3.1.2 Convert to Query

This module converts a request into a query to be executed, and passes that query on to the Database Manager Module.

##### 2.3.2.3.2 Storage Manager

###### 2.3.2.3.2.1 Database Manager

This module connects to the database and executes the queries it receives, returning the result of that query to the Convert to Data module.

### 2.3.3 Transfer Super Layer

#### 2.3.3.1 SDK Layer

##### 2.3.3.1.1 PixelSense Transfer Manager

###### 2.3.3.1.1.1 Graphics Transfer Connector

The Graphics Transfer Connector module is responsible for transferring data back and forth from the PixelSense graphics module and the AMD Athlon processor.

###### 2.3.3.1.1.2 Data Converter

The Data Converter will convert the binary data to arrays of pixel location and value so that the graphics processor will know what pixels to update and where they should be updated. This will also work the other way and convert the incoming binary data from specific pixel points to raw computational data.

##### 2.3.3.1.2 Lynx Transfer Manager

###### 2.3.3.1.2.1 USB Serial Connection Manager

The USB Serial Connection Manager is responsible for transferring data back and forth between the Lynx microcontroller and the Android device over a serial connection with a baud rate of 9600.

###### 2.3.3.1.2.1 Data Converter

The Data Converter will convert the incoming data from the serial connection to binary and put it into an array so the Lynx microcontroller will know which LEDs to flash. This will also work the other way in that it will convert the raw binary input to a binary array which can be translated to English letters and words.

#### 2.3.3.2 PixelSense Transfer Layer

##### 2.3.3.2.1 IR Receiver

###### 2.3.3.2.1.1 Infrared Sensors

The Infrared Sensors will detect the Lynx device placed on the screen as well as detect the LEDs flashing for data transmission.

##### 2.3.3.2.2 IR Pixel Cluster Emitter

###### 2.3.3.2.2.1 Pixel Group

The Pixel Group module will flash a group of pixels on the screen proportional to a size of a phototransistor so that it can emulate flashing LEDs.

##### 2.3.3.2.3 PixelSense Transfer Data Formatter

###### 2.3.3.2.3.1 Binary Converter

The Binary Converter module will convert the incoming arrays to raw binary so the graphics processor can update the screen appropriately.

#### 2.3.3.3 Lynx Transfer Layer

##### 2.3.3.3.1 LED Receiver

###### 2.3.3.3.1.1 Phototransistors

The Phototransistor module is responsible to detect light and send the individual on/off states back to the Arduino.

##### 2.3.3.3.2 LED Transmitter

###### 2.3.3.3.2.1 LEDs

The LEDs will be an array of LEDs which will flash to transfer the incoming data.

##### 2.3.3.3.3 Lynx Transfer Data Formatter

###### 2.3.3.3.3.1 Binary Converter

The Binary Converter module will convert the incoming arrays to raw binary so the Arduino will know which LEDs to flash and in what specific order.

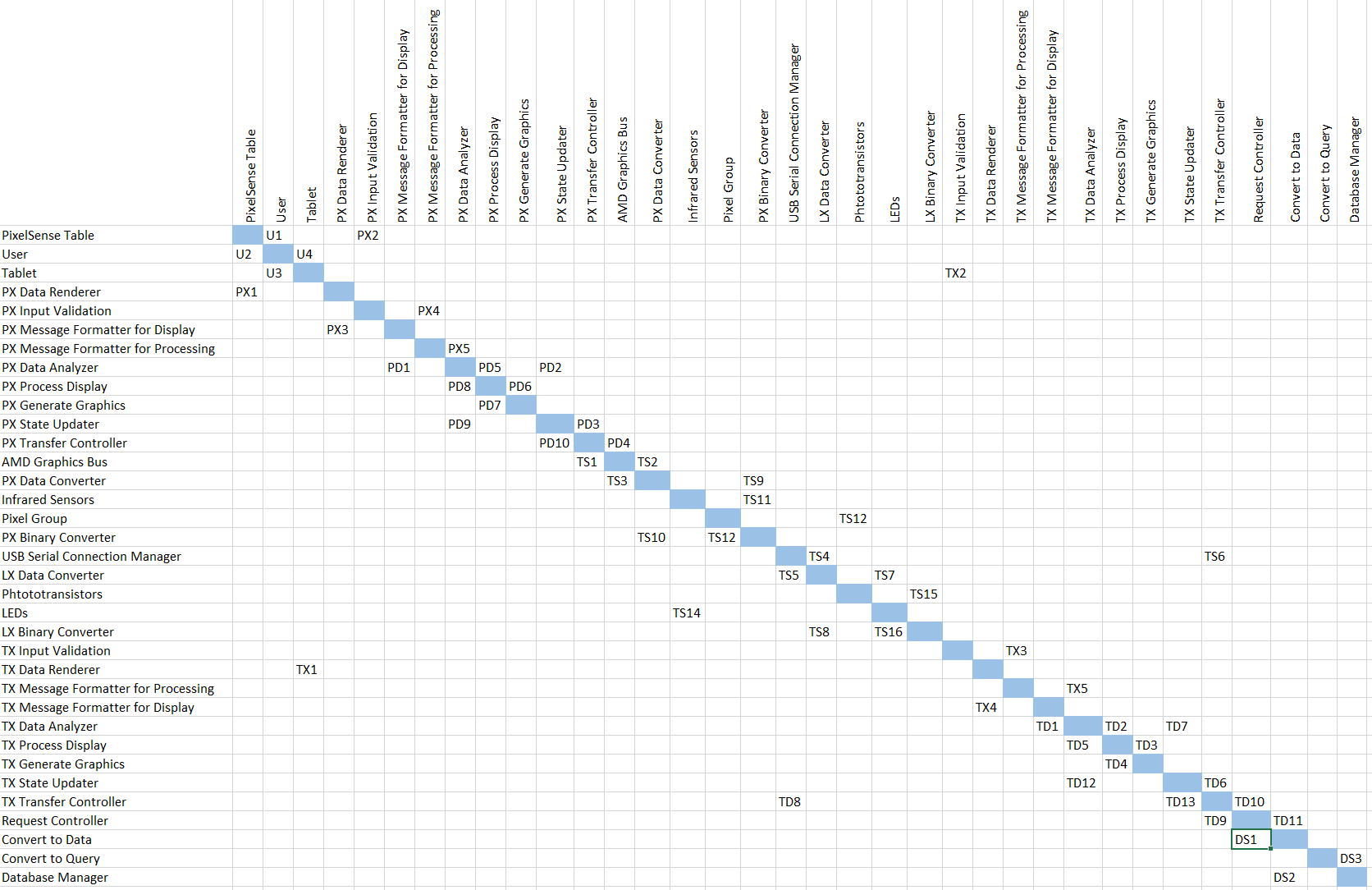


Figure 2‑10: Producer-Consumer Matrix (DDS)

### 2.3.2 Requirements Traceability

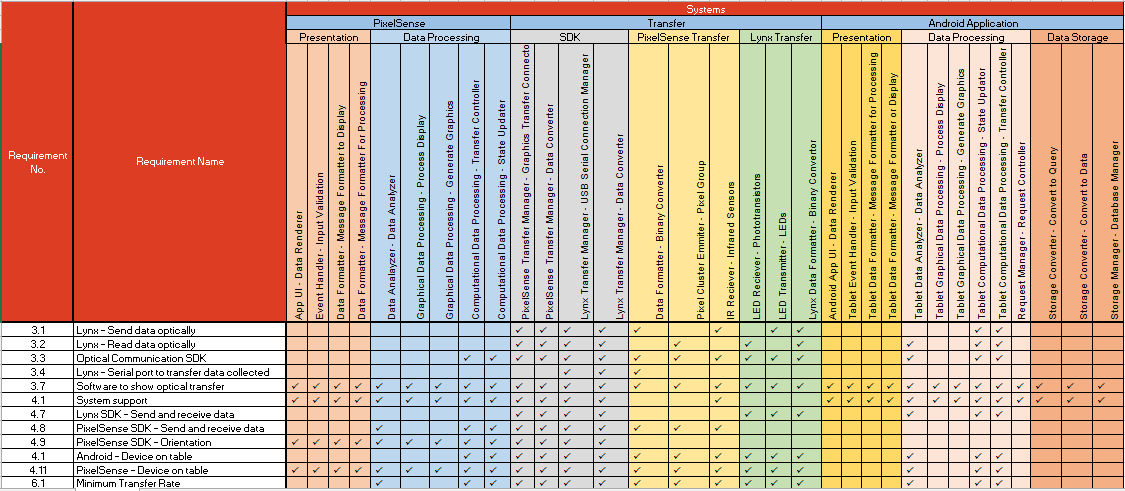


Figure 2‑11: Requirements Traceability Matrix (DDS)

# 3. Test Items

This section will describe how the PixelSense/Lynx system will be tested. Individual modules will be tested in a closed environment during the Unit Testing phase. Units will be gradually combined and tested together during component testing. Final integration testing will test each separate system environment (PixelSense and Lynx) as one entire unit. System Verification will use a series test cases to assess how well the systems interact with each other, and that all system requirements are being met. The pass/fail criteria for these tests will be mentioned in further detail in section 8.

## 3.1 Hardware Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| H1 | PixelSense Table (Infrared Sensors) | String in a binary sequence through light | The table will display the transmitted text on the screen | Use the lynx device to send an arbitrary string in binary through its IR LEDs and then check if the table outputs the same message. | Very High 5 |
| H2 | PixelSense Table (Table Screen) | Graphics vectors/shapes | The table will display the graphic vectors/shapes | We will pass a shape (square/rectangle) to the draw command and see if the table will display it | Very High 5 |
| H3 | Lynx (Phototransistors) | Light beam | Detect light beam as a logical 1 | We will shine a light on the phototransistor and if it detects a light it should output 3.3v or be “On” | Very High 5 |
| H4 | Lynx (LEDs) | 3.3V signal | LED should turn on. | We will tell the microcontroller to duty cycle 1 pin every 100ms and see if the LED turns on and off | Very High 5 |
| H5 | Android Tablet Touchscreen | Human finger touch | Screen will output that touch is detected | We will create a button in the application and when the button is a touched a “Toast” notification should appear displaying the touch location | Very High 5 |
| H6 | Android Tablet Display Screen | Turn device on. | The screen should display the boot sequence and the Android home screen after boot is finished | We will press the power button and wait and see if the tablet will boot into the Android desktop. | Very High 5 |

Table 3‑1: Hardware Tests

## 3.2 Unit Tests

### 3.2.1 PixelSense App UI Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| PAU1 | Data Renderer | Formatted string messages, images | Displayed String Message, Images | Emulate input data or supply input from a tester, and record output of module to file for later review | Moderate 3 |

Table 3‑2: PixelSense App UI Unit Tests

#### 3.2.2 PixelSense Event Handler Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| PEH1 | Input Validation | User Event | Input String Message | Emulate user input (button presses) and compare results with expected output | Moderate 3 |

Table 3‑3: PixelSense Event Handler Unit Tests

#### 3.2.3 PixelSense Presentation Data Formatter Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| PPF1 | Message Formatter for Display | Analyzed string data, images | Formatted string data for display, images | Emulated transaction string data and blackjack images and compare results with expected output | Moderate 3 |
| PPF2 | Message Formatter for Processing | Validated input string data | Formatted string data for processing | Emulate user input data (button presses) and compare results with expected output | Moderate 3 |

Table 3‑4: PixelSense Data Formatter Unit Tests

#### 3.2.4 PixelSense Data Analyzer Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| PDA1 | Data Analyzer | Formatted string data for processing, return input from processing | Formatted strings for display | Emulate input data (button presses and bets) and compare results with expected output | Very High 5 |

Table 3‑5: PixelSense Data Analyzer Unit Tests

#### 3.2.5 PixelSense Graphical Processing Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| PGP1 | Process Display | Graphics format strings | Images to display | Emulate input data (button presses and bets) and compare results with expected output | Moderate 3 |
| PGP2 | Generate Graphics | Requests for specific images | images | Emulate input data (card request) and compare results with expected output | Low 2 |

Table 3‑6: PixelSense Graphical Processing Unit Tests

#### 3.2.6 PixelSense Computational Data Processing Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| PCP1 | State Updater | Formatted data for computation, requests for optical data | State change results | Emulate input data (application commands: hit, bet, stay) and compare results with expected output | Very High 5 |
| PCP2 | Transfer Controller | Requests for optical data | Results of successful optic transfer | Emulate input data (transfer data: header, data) and compare results with expected output | Very High 5 |

Table 3‑7: PixelSense Computational Processing Unit Tests

#### 3.2.7 PixelSense Transfer Manager Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| PTM1 | Graphics Transfer Controller | Requests for transferred data, converted data from data converter | Function call to data converter polling for necessary information, successful requests from converter | Emulate input data (transfer data) and compare results with expected output | Very High 5 |
| PTM2 | Data Converter | Graphics locations and values, raw binary pixel changes from IR | Array of pixel locations and IR values | Emulate input data (pixel location, values and changes) and compare results with expected output | Very High 5 |

Table 3‑8: PixelSense Transfer Manager Unit Tests

#### 3.2.8 IR Receiver Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| IRR1 | Infrared Sensors | Raw Touch data, pixel location information | Touch meta information, binary data for pixel placement | Emulate input data (PixelSense touches, pixel locations) and compare results with expected output | Very High 5 |

Table 3‑9: IR Receiver Unit Tests

#### 3.2.9 PixelSense Cluster Emitter Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| PCE1 | Pixel Group | Binary pixel information | Light sequence | Emulate input data (binary pixel information) and compare results with expected output | Very High 5 |

Table 3‑10: PixelSense Cluster Emitter Unit Tests

#### 3.2.10 PixelSense Transfer Formatter Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| PTF1 | Binary Converter | Raw touch data, pixel location information | Raw touch data, pixel location information | Emulate input data (touch data, pixel locations) and compare results with expected output | Very High 5 |

Table 3‑11: PixelSense Transfer Formatter Unit Tests

#### 3.2.11 LED Receiver Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| LR1 | Phototransistors | Light sequence | Binary states | Emulate input data (light sequence) and compare results with expected output | Very High 5 |

Table 3‑12: LED Receiver Unit Tests

#### 3.2.12 LED Transmitter Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| LT1 | LEDs | Binary sequence | Light sequence | Emulate input data (binary sequence) and compare results with expected output | Very High 5 |

Table 3‑13: LED Transmitter Unit Tests

#### 3.2.13 Lynx Transfer Data Formatter Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| LTF1 | Binary Converter | Light sequence, ASCII Array | Binary sequences, ASCII Array | Emulate input data (light sequence and ASCII array) and compare results with expected output | Very High 5 |

Table 3‑14: Lynx Transfer Data Formatter Unit Tests

#### 3.2.14 Lynx Transfer Manager Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| LTM1 | USB Serial Connection Manager | Byte Array, ASCII Characters | Received Byte Array, ASCII Characters | Emulate input data (byte array, ascii characters) and compare results with expected output | Very High 5 |
| LTM2 | Data Converter | Raw Binary values, Byte Array | ASCII Character sequence, Byte Array | Emulate input data (binary values and byte array) and compare results with expected output | Very High 5 |

Table 3‑15: Lynx Transfer Manager Unit Tests

#### 3.2.15 Android App UI Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| AAU1 | Data Renderer | Formatted string messages, images | Displayed String Message, Images | Emulate input data (button presses and bet amounts), and record output of module to file for later review | Moderate 3 |

Table 3‑16: Android App UI Unit Tests

#### 3.2.16 Tablet Event Handler Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| TEH1 | Input Validation | User Input | Event ID | Emulate input data (button presses and bet amounts), and record output of module to file for later review | Moderate 3 |

Table 3‑17: Tablet Event Handler Unit Tests

#### 3.2.17 Tablet Presentation Data Formatter Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| TPF1 | Message Formatter for Display | Formatted String for Display, images | Formatted User Input for Rendering | Emulated transaction string data and blackjack images and compare results with expected output | Moderate 3 |
| TPF2 | Message Formatter for Processing | Event ID | Formatted User Input for Processing | Emulate user input data (button presses) and compare results with expected output | Moderate 3 |

Table 3‑18: Tablet Presentation Data Formatter Unit Tests

#### 3.2.18 Tablet Data Analyzer Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| TDA1 | Data Analyzer | Returned values from state and graphics, formatted input strings | Formatted Data and update requests, split to corresponding lower modules, final combined state update ready for render | Emulate input data (button presses: hit, bet, stay) and compare results with expected output | Very High 5 |

Table 3‑19: Tablet Data Analyzer Unit Tests

#### 3.2.19 Tablet Graphical Data Processing Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| TGP1 | Process Display | raw request for graphic | Request for image or graphic, returned generated graphic | Emulate input data (request for image) and compare results with expected output | Moderate 3 |
| TGP2 | Generate Graphics | Request for graphic (ID) | graphic | Emulate input data (graphic id) and compare results with expected output | Low  2 |

Table 3‑20: Tablet Graphical Data Processing Unit Tests

#### 3.2.20 Tablet Computational Data Processing Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| TCP1 | State Updater | Data to be processed from user input | Data request for transfer controller, returned value of new state | Emulate input data (blackjack commands and bet amounts) and compare results with expected output | Very High 5 |
| TCP2 | Transfer Controller | Transferred data requests, data to be transferred, retrieved data, | Request to store data, processed data, data to be transferred | Emulate input data (blackjack commands and bet amounts) and compare results with expected output | Very High 5 |

Table 3‑21: Tablet Computational Data Processing Unit Tests

#### 3.2.21 Request Manager Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| RM1 | Request Controller | Request as strings | Returned requests | Emulate input data (request for stored data) and compare results with expected output | Very High 5 |

Table 3‑22: Request Manager Unit Tests

#### 3.2.22 Storage Converter Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| SC1 | Convert to Data | Result of query | Translated data | Emulate input data (raw query result) and compare results with expected output | Moderate 3 |
| SC2 | Convert to Query | Requests for query | Queries to database | Emulate input data (raw data request) and compare with expected output | Moderate 3 |

Table 3‑23: Storage Converter Unit Tests

#### 3.2.23 Storage Manager Modules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Module/Unit | Input | Output | Test Method | Priority |
| SM1 | Database Manager | SQLite query, result of query | SQLite query, result of query | Emulate input data (SQLite query) and compare results with expected output | Moderate 3 |

Table 3‑24: Storage Manager Unit Tests

## 3.3 Component Tests

#### 3.3.1 PixelSense Presentation Layer Components

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Subsystem | Input | Output | Test Method | Priority |
| PPL1 | PixelSense App UI | Messages to display | Messages through the GUI | Pass in message to display and compare results with expected output | Moderate 3 |
| PPL2 | PixelSense Event Handler | User Input | Event IDs and data to be changed in state | Pass in user input (application commands) and compare results with expected output | Moderate 3 |
| PPL3 | PixelSense Presentation Data Formatter | Event IDs and data to be changed in state, messages to display in the App UI | Returned messages to display in App UI, formatted data to data analyzer | Pass in event id’s and blackjack application data (current cards…) and compare results with expected output | Moderate 3 |

Table 3‑25: PixelSense Presentation Layer Component Tests

#### 3.3.2 PixelSense Processing Layer Components

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Subsystem | Input | Output | Test Method | Priority |
| PDL1 | PixelSense Data Analyzer | Formatted data from Presentation Layer, returned values from lower processing | Split data(formatted data for state, formatted data for display graphics), Returned updated state for display | Pass in simulated pre-processed data (new transaction, image, transferred data) and compare results with expected output | Very High 5 |
| PDL2 | PixelSense Graphical Data Processing | Raw data for graphics | Returned graphics | Pass in raw graphical data and compare results with expected output | Moderate 3 |
| PDL3 | PixelSense Computational Data Processing | Raw data for state update | Returned state | Pass in raw blackjack application data and compare results with expected output | Very High 5 |

Table 3‑26: PixelSense Processing Layer Component Tests

#### 3.3.3 SDK Layer Components

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Subsystem | Input | Output | Test Method | Priority |
| SDK1 | PixelSense Transfer Manager | Requests for optic data | Returned optic data result | Pass in request for data and compare results with expected output | Very High 5 |
| SDK2 | Lynx Transfer Manager | Requests for optic data | Returned optic data result | Pass in request for data and compare results with expected output | Very High 5 |

Table 3‑27: SDK Layer Component Tests

#### 3.3.4 PixelSense Transfer Layer Components

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Subsystem | Input | Output | Test Method | Priority |
| PTL1 | IR Receiver | Light sequence | Byte array for processing | Pass in a simulated light sequence and compare results with expected output | Very High 5 |
| PTL2 | PixelSense Cluster Emitter | Byte array for processing | Light sequence | Pass in byte array and compare results with expected output | Very High 5 |
| PTL3 | PixelSense Transfer Data Formatter | Optic data requests, received byte arrays | Byte arrays to transfer, returned received data | Pass in request for data and compare results with expected output | Very High 5 |

Table 3‑28: PixelSense Transfer Layer Component Tests

#### 3.3.5 Lynx Transfer Layer Components

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Subsystem | Input | Output | Test Method | Priority |
| LTL1 | LED Receiver | Light sequence | Byte array for processing | Pass in light sequence and compare results with expected output | Very High 5 |
| LTL2 | LED Transmitter | Byte array for processing | Light sequence | Pass in byte array and compare results with expected output | Very High 5 |
| LTL3 | Lynx Transfer Data Formatter | Byte array for processing | Light sequence | Pass in byte array and compare results with expected output | Very High 5 |

Table 3‑29: Lynx Transfer Layer Component Tests

#### 3.3.6 Tablet Presentation Layer Components

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Subsystem | Input | Output | Test Method | Priority |
| TPL1 | Android App UI | Messages to display | Messages through the GUI | Pass in message string and compare results with expected output | Moderate 3 |
| TPL2 | Tablet Event Handler | User Input | Event IDs and data to be changed in state | Pass in simulated user inputs and compare results with expected output | Moderate 3 |
| TPL3 | Tablet Presentation Data Formatter | Event IDs and data to be changed in state, messages to display in the App UI | Returned messages to display in App UI, formatted data to data analyzer | Pass in event id’s and blackjack application data and compare results with expected output | Moderate 3 |

Table 3‑30: Tablet Presentation Layer Component Tests

#### 3.3.7 Tablet Data Processing Layer Components

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Subsystem | Input | Output | Test Method | Priority |
| TDL1 | Tablet Data Analyzer | Formatted data from Presentation Layer, returned values from lower processing | Split data(formatted data for state, formatted data for display graphics), Returned updated state for display | Pass in formatted user inputs, and simulate processed data then compare results with expected output | Very High 5 |
| TDL2 | Tablet Graphical Data Processing | Raw data for graphics | Returned graphics | Pass in raw graphical data and compare results with expected output | Moderate 3 |
| TDL3 | Tablet Computational Data Processing | Raw data for state update | Returned state | Pass in raw blackjack application data and compare results with expected output | Very High 5 |
| TDL4 | Request Manager | Requests for data storage, requests for data from storage | Returned storage | Pass in request to store or retrieve data and compare results with expected output | Moderate 3 |

Table 3‑31: Tablet Data Processing Layer Component Tests

#### 3.3.8 Data Storage Components

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Subsystem | Input | Output | Test Method | Priority |
| DS1 | Storage Converter | Requests for data, returned string query results | Queries to database, requested data | Pass in request for data or raw query results and compare results with expected output | Moderate 3 |
| DS2 | Storage Manager | Queries | Query results | Pass in SQLite queries and compare results with expected output | Moderate 3 |

Table 3‑32: Data Storage Component Tests

## 3.4 Integration Tests

#### 3.4.1 PixelSense Super Layer Integration

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Functional Unit | Input | Output | Test Method | Priority |
| PSL1 | Presentation Layer | User Input, updated state to display | Displayed GUI | Pass in simulated user inputs or simulated processed data and compare results with expected output | Moderate 3 |
| PSL2 | Data Processing Layer | User Event data, received optic data | Sent optic data, final state to render | Pass in user event data or simulated optic data and compare results with expected output | Very High 5 |

Table 3‑33: PixelSense Super Layer Integration Tests

#### 3.4.2 Transfer Super Layer Integration

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Functional Unit | Input | Output | Test Method | Priority |
| TSL1 | SDK Layer | Optic data requests | Optic data transfer results | Pass in request for optic data and compare results with expected output | Very High 5 |
| TSL2 | PixelSense Transfer Layer | Data to send optically, light sequences | Light sequences, received and translated data | Pass in data to send optically or simulated light sequence and compare results with expected output | Very High 5 |
| TSL3 | Lynx Transfer Layer | Data to send optically, light sequences | Light sequences, received and translated data | Pass in data to send optically or simulated light sequence and compare results with expected output | Very High 5 |

Table 3‑34: SDK Super Layer Integration Tests

#### 3.4.3 Lynx Super Layer Integration

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Functional Unit | Input | Output | Test Method | Priority |
| LSL1 | Presentation Layer | User Input, updated state to display | Displayed GUI | Pass in simulated user input or updated application data and compare results with expected output | Moderate 3 |
| LSL2 | Data Processing Layer | User Event data, received optic data | Sent optic data, final state to render | Pass in user event or simulated optic data and compare results with expected output | Very High 5 |
| LSL3 | Data Storage | Requests to store or retrieve data | Retrieved data | Pass in request to store or retrieve data and compare results with expected output | Moderate 3 |

Table 3‑35: Lynx Super Layer Integration Tests

## 3.5 System Validation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Functional Unit | Input | Output | Test Method | Priority |
| V1 | Verify Lynx Device | ID transmitted through light | Acknowledgement response | The ID of the lynx device will be sent to the table where it will be authenticated and an acknowledgment response will be returned. | Very High 5 |
| V2 | Data Transfer among Devices (Lynx to PixelSense Table) | Data transmitted through light | The plain text of the data that was sent | The Lynx device will send a binary sequence of a string through the IR LEDs and the table will display that string on the table display. | Very High 5 |
| V3 | Data Transfer among Devices (PixelSense Table to Lynx) | Data transmitted through light flashes on the screen. | The plain text of the data that was sent | The PixelSense table will send a binary sequence of a string through the screen and the tablet will display that string on the tablet display. | Very High 5 |

Table 3‑36: System Validation Tests

# 4. Risks

The purpose of this section is to identify risks that may affect the testing process and the outcome of the tests.

## 4.1 Risk Table

Each identified risk has an assigned severity rating, 1-5 with 1 being very low and 5 being very high.

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Impact** | **Severity** | **Management Plan** |
| Teensy 3.1 not functioning properly | Transfer and receiving tests will be stopped. Could affect the delivery of final product. | Very High - 5 | Ensure the Teensy 3.1 is functioning properly on arrival and before testing. |
| PixelSense not functioning properly | Transfer, receiving and application tests will be stopped, may affect the delivery of final product. | Very High – 5 | Ensure the Samsung SUR40 PixelSense table is functioning properly on arrival and before testing. |
| Android Tablet not functioning properly | Transfer, receiving and application tests will be stopped, may affect the delivery of the final product. | Very High – 5 | Ensure the Android tablet is functioning properly on arrival and before testing. |
| LED’s on Lynx not functioning properly | Optical transfer tests will be stopped, unlikely to affect the delivery of final product. | Moderate – 3 | Ensure LED’s are functioning properly on arrival and before testing. (additional LED’s have been ordered to decrease severity |
| Photo Receiver’s on Lynx not functioning properly | Optical receiving tests will be stopped, unlikely to affect the delivery of final product | Moderate – 3 | Ensure Photo Receivers are functioning properly on arrival and before testing. (additional LED’s have been ordered to decrease severity |

Table 4‑1: Risk Table

# 5. Features To Be Tested

This section highlights how each requirement from the original SRD will be addressed during the testing process. Each will be described according to their original definition, and a testing process for each requirement.

## 5.1 Customer Requirements

### 5.1.1 The product shall be able to send data optically

**Description:** The product must be able to send specifically timed flashes of light as bits of information and send them in a way a receiving unit can interpret it.

**Test Approach:** This requirement will be tested by sending optical data from Tablet to PixelSense and vice versa.

### 5.1.2 The product shall be able to read data optically

**Description:** The product must be able to receive specifically timed flashes of light as bits of information and interpret them correctly, storing the data if necessary.

**Test Approach:** This requirement will be tested by sending optical data from Tablet to PixelSense then displaying the data on the receiving device to show correctness, as well as from PixelSense to Tablet.

### 5.1.3 All work done by the product involving optical communication will be compiled into a well-documented library

**Description:** A suite of functions designed to handle optical communication between capable hardware will be available in a portable SDK library. The details of this requirement is elaborated on in the next section.

**Test Approach:** One team member will impose a guideline for proper documentation of all SDK code. Adherence to these guidelines will establish the necessary pass/fail criteria

### 5.1.4 The device must have a serial port to transfer data collected

**Description:** The Lynx will have a serial port to transfer data optically collected to the Android device connected.

**Test Approach:** This feature will be tested in initial Hardware tests. Success hinges on data to successfully being pulled from the Lynx to the tablet, and vice-versa, with no error.

### 5.1.5 The device and table’s communication should be error tolerant

**Description:** All data sent should be received intact, and be identical to the original message. The Lynx should detect errors in information received.

**Test Approach:** This requirement will be tested along with the Transfer Super Layer integration test. Its success will also validate this requirement.

### 5.1.6 Communication between both devices must sustain a reliable connection

**Description:** All data sent should be received intact, and be identical to the original message. The protocol will have a system to detect missing or corrupt data, and replace the missing pieces with the correct information.

**Test Approach:** This requirement will be tested along with the System Validation tests. Passing will require for a session of Blackjack play to occur with little (<10%) error in state while using the Lynx

### 5.1.7 Software will be built to showcase the optical transfer protocol and the built device

**Description:** Software will be built to showcase the optical transfer protocol and the built device

**Test Approach:** This requirement will be tested along with the PixelSense and Tablet Super Layer integration tests. Their success will also validate this requirement.

### 5.1.8 The system will support customizable light sequences

**Description:** The Device, Android and PixelSense SDK will provide functions that gives ability to customize the sequence of light transmitted and to set what happens when those light sequences occurs.

**Test Approach:** This requirement will be tested along with the modules of the Transfer Super Layer. This requirement will be met as soon as two different packets (with unique contents) are successfully sent through the Super Layer.

### 5.1.9 The system shall allow encoding of data

**Description:** The SDK will provide function(s) that allow the user to encode the data being transmitted.

**Test Approach:** This will be tested by encoding data and transmitting the data from both Tablet to PixelSense and vice versa.

### 5.1.10 The system shall support a 4x4 array of transmitters and receivers

**Description:** The Device and PixelSense SDK will provide support for a 4x4 array to transmit and receive data to and from the Surface table.

**Test Approach:** This requirement will be satisfied with the successful test of the Transfer Super Layer, and the completion of the Lynx prototype. Their success will validate this requirement

### 5.1.11 The system shall authenticate the Lynx device when plugged into the tablet

**Description:** The Android and Device SDK will provide function(s) that will allow the Android application to authenticate that a valid Lynx device is plugged into the Android tablet.

**Test Approach:** This feature will be tested as an authentication function of the SDK definition. The feature is successfully implemented with the Lynx and Table exchanging information about the Lynx’s unique identification number, and noting it for the state on the table.

### 5.1.12 The system shall allow the Lynx to send and receive data

**Description:** The Device and Android SDK will provide functions that will allow the Android application developed with it to send and receive data using the Lynx device.

**Test Approach:** This requirement will be tested by sending and receiving data from the Lynx device to the PixelSense and Android tablet.

### 5.1.13 The system shall allow the PixelSense to send and receive data

**Description:** The PixelSense SDK will provide functions that will allow the PixelSense application developed with it to send to and receive data from the Lynx device.

**Test Approach:** This requirement will be tested by sending and receiving data from the PixelSense to the Lynx device.

### 5.1.14 The system shall be able to determine the orientation of the Lynx

**Description:** The PixelSense SDK functions will provide the ability to determine how the Lynx is oriented on the Surface table. These functions can leverage existing functionality present in the development kit for PixelSense.

**Test Approach:** This requirement will be tested by using a unique set of images that will verify the location and orientation of the Lynx.

### 5.1.15 The system shall notify the Android device if the Lynx is on the table

**Description:** The Device and Android SDK will provide information to the user via the Android tablet connected to the Lynx when the Lynx is on the Surface table and ready to transmit or receive data.

**Test Approach:** This feature will be tested as an authentication function of the SDK definition. The feature is successfully implemented with the Lynx and Table exchanging information about the Lynx’s unique identification number, and noting it for the state on the table.

### 5.1.16 The system shall notify the table that the Lynx is on it

**Description:** The Device and PixelSense SDK will provide information to the user via the Surface table that the Lynx is on the Surface table and ready to transmit or receive data.

**Test Approach:** This feature will be tested as an authentication function of the SDK definition. The feature is successfully implemented with the Lynx and Table exchanging information about the Lynx’s unique identification number, and noting it for the state on the table.

### 5.1.17 The system shall authenticate that a valid Lynx device is placed on the PixelSense table.

**Description:** The PixelSense SDK will provide functions to authenticate that a valid Lynx device is on the PixelSense table and is capable of sending and receiving data.

**Test Approach:** This feature will be tested as an authentication function of the SDK definition. The feature is successfully implemented with the Lynx and Table exchanging information about the Lynx’s unique identification number, and noting it for the state on the table.

## 5.2 Packaging Requirements

### 5.2.1 Attachable to Tablet

**Description:** The Lynx must be attachable and detachable physically from the tablet it is connected to. This is referring to a mechanism so the Lynx can physically stay on the tablet it is connected to, not the serial connection with the cord plugged into the tablet.

**Test Approach:** The feature will be tested by constructing a casing that is designed to fit our tablet, and holding the tablet at different orientations. If the lynx remains attached, the test is successful.

### 5.2.2 Fully Assembled

**Description:** The Lynx will be assembled and ready for use.

**Test Approach:** This feature will be tested during final validation testing. The lynx will fail if it is in two discrete pieces, or only connected through wire.

### 5.2.3 Lynx as a Tablet Case

**Description:** The Lynx will serve as a tablet case for the tablet we chose to develop for.

**Test Approach:** This feature will be tested by creating a tablet case the also holds all of the Lynx hardware. If any components do not fit together the test will fail.

### 5.2.4 SDK on USB Flash Drive

**Description:** The Lynx’ SDK for the PixelSense table, the Secure Transfer Device, and the connected device will be provided on a USB flash drive for use on Windows.

**Test Approach:** The SDK will be loaded to a flash drive and the test will attempt to recover and use the SDK after transfer. Successful use of the code after transfer will pass.

### 5.2.5 PixelSense Casino Software

**Description:** The PixelSense Casino Software will be provided on a USB Flash Drive and the software can be installed on the table.

**Test Approach:** The PixelSense Casino software will be loaded onto a flash drive and the test will attempt to recover and use the SDK after transfer. Unsuccessful use of the code after transfer will fail.

### 5.2.6 PC Casino Companion Software

**Description:** The PC Casino Companion Software will be packaged as an executable file on an USB Flash Drive that can be used on compatible Windows PCs.

**Test Approach:** The PC Casino software will be loaded onto a flash drive and the test will attempt to recover and use the SDK after transfer. Unsuccessful use of the code after transfer will fail.

### 5.2.7 Android Casino Software

**Description:** The Android Casino Software will be packaged in an APK on an USB Flash Drive that can be installed on compatible Android tablets.

**Test Approach:** The Android Casino software will be loaded onto a flash drive and the test will attempt to recover and use the SDK after transfer. Unsuccessful use of the code after transfer will fail.

## 5.3 Performance Requirements

### 5.3.1 Minimum Data Transfer Rate

**Description:** Data transfer must be consistent at minimum of 200 bits/sec.

**Test Approach:** During recording of module output, a time stamp will be supplied to the file. The test will pass if the transfer of data is 200 bits/sec or above relative to these timestamps

### 5.3.2 Data Translation

**Description:** Converting optical data to binary data for later readability should be at most 1 second.

**Test Approach:** During recording of module output, a time stamp will be supplied to the file. The test will pass if the conversation of data is less than 1 second relative to these timestamps

### 5.3.3 Authentication Time

**Description:** The Lynx should not take too long to authenticate its integrity with the PixelSense. Time should at most 5 seconds.

**Test Approach:** During recording of module output, a time stamp will be supplied to the file. The test will pass if authentication is less than 5 seconds relative to these timestamps

### 5.3.4 Software Boot Times

**Description:** Software should not take a long time to boot and load data. Ideally this should be less than 10 seconds.

**Test Approach:** Boot time will be timed by a team member. Success is if boot times consistently finish under 10 seconds

### 5.3.5 Data Read/Write Times

**Description:** Reading and writing data to the Lynx should be at most 1 second.

**Test Approach:** During recording of module output, a time stamp will be supplied to the file. The test will pass if reading and writing to the Lynx is less than 1 second relative to these timestamps

### 5.3.6 Battery Life

**Description:** Battery Life on the Lynx should last about half a casino day which is about 8 hours.

**Test Approach:** During System Validation, one scenario will let the Lynx run for an extended period of time, measuring battery capacity. The test is successful if the battery remains useful through a day’s worth of use

### 5.3.7 Overall Connection Times

**Description:** Connection times to the Lynx, tablet and DB should take no longer than 5 seconds.

**Test Approach:** During recording of module output, a time stamp will be supplied to the file. The test will pass if DB requests are less than 5 seconds relative to these timestamps

## 5.4 Safety Requirements

### 5.4.1 No sharp edges

**Description:** The Lynx must not have any sharp edges.

**Test Approach:** This feature will be tested through system validation. If anyone is cut or injured during play, the test fails

### 5.4.2 Protective Case around Device

**Description:** Lynx should have a protective case to prevent exposure to liquids

**Test Approach:** This feature will be tested through system validation. If the Lynx is not covered by a protective case, and does not stay covered through regular play, this test fails.

## 5.5 Maintenance and Support Requirements

### 5.5.1 User Manual

**Description:** A user manual shall include a step by step guide to use the SDK with device & software on Surface Table.

**Test Approach:** A team member will set criteria for the structure of the manual. Failing to adhere to these criteria fails the test.

### 5.5.2 Software Installation

**Description:** Software shall be provided to interact between device and PixelSense Table. Software shall not require additional permission to install software. Software shall work on any Windows 7 Operating System on the PixelSense Hardware.

**Test Approach:** The test passes if the software can be installed on a new device.

### 5.5.3 Source Code / SDK

**Description:** Source code shall be provided to the customer as final deliverable. Basic Software Development Kit with usage guide shall be provided in order to use it further for PixelSense Development.

**Test Approach:** The test passes if source code for the SDK portion of this system is delivered to the client without compilation in addition to the final product.

### 5.5.4 Hardware Support

**Description:** The device shall be ready to use with PixelSense Secure transfer system. The device shall include Android Operating System with PixelSense Application. If required, customer can replace device with same application in future.

**Test Approach:** The test fails if the Lynx does not function at the end of development

## 5.6 Other Requirements

### 5.6.1 American English Standard

**Description:** The PixelSense Secure Transfer shall use American English as the default language for any text or audio speech.

**Test Approach:** Any code or documentation that is not written in English will fail the test.

### 5.6.1 User Friendly Interface

**Description:** The PixelSense Secure Transfer shall have a user interface that can be learned in under 2 hours, and includes guidance to aid the user during use of the application.

**Test Approach:** If a new user, from a different team, cannot comfortably use the system after 2 hours of use, the test fails.

# 6. Features Not To Be Tested

Some requirements will not receive a specific testing strategy, and the reasons for their exclusion are listed below.

## 6.1 Customer Requirements

### 6.1.1 The system will support Android API 17 and Surface 2.0 platform

**Description:** The system will support Android API 17 and Surface 2.0 platforms

**Reasoning:** Since the system is being built using the Surface 2.0 platform, and Android API17, the requirement will be met without any need for testing.

### 6.1.2 The system shall allow the expansion of the array

**Description:** The Device and PixelSense SDK functions provided will allow the user to set the size of the array of transmitters they want to use and support that functionality.

**Reasoning:** The team is focused on building one Lynx with one preconfigured array. Code will be implemented to allow for unique dimensions, but will not be tested.

### 6.1.3 The system shall support multiple devices

**Description:** The PixelSense SDK will provide functions to detect and support multiple Lynx devices on the table.

**Test Approach:** The team is focused on building one Lynx. Code will be implemented to allow for multiple lynx, but will not be tested.

# 7. Testing Approaches

## 7.1 Strategy

Due to the design of the system, the testing process can be streamlined to use a handful of testing modules for every unit in the system. Every unit expects a particularly formatted piece of data, and outputs a different piece of formatted data. This is true for every unit a module communicates with. Our approach would feed in appropriately formatted data to a unit under testing, and then logs the output, along with any alerts or exceptions, for visual confirmation. Passing a test would be rely on parity with an expected correct result per units. Since the system is a two-way pipeline of information, to and from the table and tablet, component and integration testing can be accomplished by chaining together unit tests.

### 7.1.1 Unit Testing

Unit Testing will be done upon the completion of a module. The designated developer responsible for testing the unit (someone who did not construct the module, preferably) will develop an appropriate file of test input. This input should include normal and extreme cases with the intent of breaking the module. This input file will be fed into the unit through a static function call in the module’s code. Additionally, two calls will be added to the end of the module’s code, one call to log the input received (the test file), as well as the intended output for each line of input. A successful test will output data that conforms to the data format agreed upon by the team, or appropriate error returns on bad data tests.

### 7.1.2 Component Testing

Component Testing is the combination of two or more successive modules in the pipeline chain. Regardless of the number of modules in a component, the “first” module in the chain needs to make a call for input. Every module must make the calls to output. This effectively documents the progression of data between the modules of a chain. Passing a test is the same criteria for Unit testing, but the intended format is consistent with the last module in the chain.

### 7.1.3 Integration Testing

Integration Testing will be conducted exactly like component testing, but on a larger scale

### 7.1.4 Regression Testing

Since every module is logging its output, regression testing will be implicitly performed at each step above unit testing.

### 7.1.5 System Validation Testing

As the system is developed, tests to verify whether the system as a whole is acceptable will be conducted. These tests will include user use cases and cover a variety of lower level test cases.

## 7.2 Metrics

|  |  |  |  |
| --- | --- | --- | --- |
| Priority | Description | Pass Criteria | Fail Criteria |
| Very High 5 | Essential to the core requirements of the system. Nothing can be delivered if not completed | Tests are successful >90% of the time | Tests fail >10% of the time |
| High  4 | Essential to the core requirements of the system. Nothing can be delivered if not completed | Test are successful >80% of the time | Tests fail >20% of the time |
| Moderate  3 | Semi-essential to the completion of the product. Either make the system easier to use /facilitate testing/showcasing of project | Tests are successful >70% of the time | Tests fail >30% of the time |
| Low  2 | Non-essential to the core requirements of the system. Strictly about ease of use, or feature stubs for further refinement | Tests are successful >60% of the time | Tests fail >40% of the time |
| Very Low 1 | Non-essential to the core requirements of the system. Strictly about ease of use, or feature stubs for further refinement | Tests are successful >50% of the time | Tests fail >50% of the time |

Table 7‑1: Test Metrics

# 8. Item Pass/Fail Criteria

Each test will have a specific criteria for passing and failing. These conditions are for each individual test, and not an aggregate over every individual test per module/component/layer.

## 8.1 Hardware Tests

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Hardware | Pass Criteria | Fail Criteria |
| H1 | PixelSense Table (Infrared Sensors) | Table successfully detects the infrared lights. | Table does not detect the infrared lights. |
| H2 | PixelSense Table (Table Screen) | Table will display the appropriate content to the user. | Table does not display or incorrectly displays the content. |
| H3 | Lynx (Phototransistors) | Phototransistors will detect lights being directed at it. | Phototransistors fail to distinguish light and no light. |
| H4 | Lynx (LEDs) | LEDs will flash the appropriate binary sequence. | LEDs do not flash or flashes an incorrect sequence.. |
| H5 | Android Tablet Touchscreen | Tablet will detect touch and will execute an action appropriately. | Tablet does not detect touch or performs and undesired operation on a touch event. |
| H6 | Android Tablet Display Screen | Tablet will display the appropriate content to the user. | Tablet does not display or incorrectly displays the content. |

Table 8‑1: Hardware Test Criteria

## 8.2 Unit Tests

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Unit/Module | Pass Criteria | Fail Criteria |
| PAU1 | Data Renderer | All data is rendered properly and matches the expected output | Any data is not rendered properly and does not match the expected output |
| PEH1 | Input Validation | Input data is properly validated and matches the expected output | Input data is not properly handled and does not match expected output |
| PPF1 | Message Formatter for Display | Message is properly formatted to be displayed and matches the expected output | Message is not properly formatted to be displayed and does not match the expected output |
| PPF2 | Message Formatter for Processing | Message is properly formatted for processing and matches the expected output | Message is not properly formatted for processing and doesn’t match the expected output |
| PDA1 | Data Analyzer | Data is analyzed and passed to the proper module and matches the expected output | Actual output does not matches the expected output |
| PGP1 | Process Display | Display is processed and matches the expected output | Actual output does not matches the expected output |
| PGP2 | Generate Graphics | Graphics are generated and matches the expected output | Actual output does not matches the expected output |
| PCP1 | State Updater | The state of the application is updated and matches the expected output | Actual output does not matches the expected output |
| PCP2 | Transfer Controller | Data to be transferred is handled properly and matches the expected output | Actual output does not matches the expected output |
| PTM1 | Graphics Transfer Controller | Graphics to be transferred are handled properly and matches the expected output | Actual output does not matches the expected output |
| PTM2 | Data Converter | Data is converted and matches the expected output | Actual output does not matches the expected output |
| IRR1 | Infrared Sensors | Infrared Sensors read data properly and matches the expected output | Actual output does not matches the expected output |
| PCE1 | Pixel Group | Pixel Group flashes the proper data and matches the expected output | Actual output does not matches the expected output |
| PTF1 | Binary Converter | Data is converted to binary and matches the expected output | Actual output does not matches the expected output |
| LR1 | Phototransistors | Phototransistors properly read data and matches the expected output | Actual output does not matches the expected output |
| LT1 | LEDs | LED’s properly flash and matches the expected output | Actual output does not matches the expected output |
| LTF1 | Binary Converter | Data is converted properly and matches the expected output | Actual output does not matches the expected output |
| LTM1 | USB Serial Connection Manager | Serial Connection between Lynx and Android functions properly and matches the expected output | Actual output does not matches the expected output |
| LTM2 | Data Converter | Data is converted properly and matches the expected output | Actual output does not matches the expected output |
| AAU1 | Data Renderer | Data is rendered properly and matches the expected output | Actual output does not matches the expected output |
| TEH1 | Input Validation | Input if properly validated and matches the expected output | Actual output does not matches the expected output |
| TPF1 | Message Formatter for Display | Message is formatted for display and matches the expected output | Actual output does not matches the expected output |
| TPF2 | Message Formatter for Processing | Message is formatted for processing and matches the expected output | Actual output does not matches the expected output |
| TDA1 | Data Analyzer | Data is analyzed and passed to proper module and matches the expected output | Actual output does not matches the expected output |
| TGP1 | Process Display | Display is processed and matches the expected output | Actual output does not matches the expected output |
| TGP2 | Generate Graphics | Graphics are properly generated and matches the expected output | Actual output does not matches the expected output |
| TCP1 | State Updater | State is updated according to current blackjack state and matches the expected output | Actual output does not matches the expected output |
| TCP2 | Transfer Controller | Data to be transferred or received is handled and matches the expected output | Actual output does not matches the expected output |
| RM1 | Request Controller | Request for data is properly handled and matches the expected output | Actual output does not matches the expected output |
| SC1 | Convert to Data | Query result is converted to data and matches the expected output | Actual output does not matches the expected output |
| SC2 | Convert to Query | Request is converted to SQLite query and matches the expected output | Actual output does not matches the expected output |
| SM1 | Database Manager | Query is able to execute and matches the expected output | Actual output does not matches the expected output |

Table 8‑2: Unit Test Criteria

## 8.3 Component Tests

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Subsystems | Pass Criteria | Fail Criteria |
| PPL1 | PixelSense App UI | PixelSense App UI takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| PPL2 | PixelSense Event Handler | PixelSense Event Handler takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| PPL3 | PixelSense Presentation Data Formatter | PixelSense Presentation Data Formatter takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| PDL1 | PixelSense Data Analyzer | PixelSense Data Analyzer takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| PDL2 | PixelSense Graphical Data Processing | PixelSense Graphical Data Processing takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| PDL3 | PixelSense Computational Data Processing | PixelSense Computational Data Processing takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| SDK1 | PixelSense Transfer Manager | PixelSense Transfer Manager takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| SDK2 | Lynx Transfer Manager | Lynx Transfer Manager takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| PTL1 | IR Receiver | IR Receiver takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| PTL2 | PixelSense Cluster Emitter | PixelSense Cluster Emitter takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| PTL3 | PixelSense Transfer Data Formatter | PixelSense Transfer Data Formatter takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| LTL1 | LED Receiver | LED Receiver takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| LTL2 | LED Transmitter | LED Transmitter takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| LTL3 | Lynx Transfer Data Formatter | Lynx Transfer Data Formatter takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| TPL1 | Android App UI | Android App UI takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| TPL2 | Tablet Event Handler | Tablet Event Handler takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| TPL3 | Tablet Presentation Data Formatter | Tablet Presentation Data Formatter takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| TDL1 | Tablet Data Analyzer | Tablet Data Analyzer takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| TDL2 | Tablet Graphical Data Processing | Tablet Graphical Data Processing takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| TDL3 | Tablet Computational Data Processing | Tablet Computational Data Processing takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| TDL4 | Request Manager | Request Manager takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| DS1 | Storage Converter | Storage converter takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |
| DS2 | Storage Manager | Storage Manager takes in the proper input and matches the expected output | Test output does not conform to expected output given emulated input over a chain of modules |

Table 8‑3: Component Tests

## 8.4 Integration Tests

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Systems | Pass Criteria | Fail Criteria |
| PSL1 | Presentation Layer | Presentation Layer takes in the proper input and matches the expected output | Test output does not conform to expected output given input over a chain of subcomponents |
| PSL2 | Data Processing Layer | Data Processing Layer takes in the proper input and matches the expected output | Test output does not conform to expected output given input over a chain of subcomponents |
| TSL1 | SDK Layer | SDK Layer takes in the proper input and matches the expected output | Test output does not conform to expected output given input over a chain of subcomponents |
| TSL2 | PixelSense Transfer Layer | PixelSense Transfer Layer takes in the proper input and matches the expected output | Test output does not conform to expected output given input over a chain of subcomponents |
| TSL3 | Lynx Transfer Layer | Lynx Transfer Layer takes in the proper input and matches the expected output | Test output does not conform to expected output given input over a chain of subcomponents |
| LSL1 | Presentation Layer | Presentation Layer takes in the proper input and matches the expected output | Test output does not conform to expected output given input over a chain of subcomponents |
| LSL2 | Data Processing Layer | Data Processing Layer takes in the proper input and matches the expected output | Test output does not conform to expected output given input over a chain of subcomponents |
| LSL3 | Data Storage | Data Storage Layer takes in the proper input and matches the expected output | Test output does not conform to expected output given input over a chain of subcomponents |

Table 8‑4: Integration Test Criteria

## 8.5 System Validation Tests

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Test Case | Pass Criteria | Fail Criteria |
| V1 | Verify Lynx Device | The PixelSense table successfully authenticates the Lynx device. | The PixelSense table fails to or incorrectly verifies the Lynx device. |
| V2 | Data Transfer among Devices (Lynx to PixelSense Table) | The plain text data is successfully displayed onto the table screen. | The table does not or incorrectly displays the data sent over. |
| V3 | Data Transfer among Devices (PixelSense Table to Lynx) | The plain text data is successfully displayed onto the tablet screen. | The tablet does not or incorrectly displays the data sent over. |

Table 8‑5: Validation Test Criteria

# 9. Test Deliverables

All testing deliverables listed below will be documented, recorded, and included in the final product.

## 9.1 Deliverables

### 9.1.1 System Test Plan

The current document explains how the development team (Argus), will plans to test the system’s hardware and software. Specifically see section 7.

### 9.1.2 Test Case Specifications

For all test cases, the following criteria will be used for their documentation:

* **Test Case ID**: each test case will be given a unique id number
* **Description**: a brief description of the test case, including how it is performed and why it is performed.
* **Inputs**: the set of inputs needed for the given test case
* **Expected** **Output**: the expected results of the test case using the provided inputs

### 9.1.3 Test Case Results

For all test cases, the following criteria will be used to document their results:

* **Test Case ID**: the unique id number for the test case
* **Date of Test**: the date of the test case was performed
* **Name of Tester**: the name of the person performing the test case
* **Inputs**: the set of inputs used for the test case
* **Expected** **Output**: the expected output of the test case
* **Actual Output**: the actual output of the test case
* **Test Result**: if the actual and expected outputs match then the test case Passed, if not the test case Failed
* **Tester Comments**: any comments the tester can provide about the test case
* **Bugs**: if any bugs are identified as a result of a test case the following criteria will be used in their documentation.
  + **Bug ID**: each identified bug receives a unique id number
  + **Bug Description**: a description of the bug

### 9.1.4 Bugs and Defects

For all bugs identified, the following criteria will be used for their documentation:

* **Bug ID**: each identified bug receives a unique id number
* **Test Case ID**: the unique id number for the test case
* **Date of Test**: the date of the test case was performed
* **Name of Tester**: the name of the person performing the test case
* **Inputs**: the set of inputs used for the test case
* **Expected** **Output**: the expected output of the test case
* **Actual Output**: the actual output of the test case
* **Bug Description**: a description of the bug
* **Tester Comments**: any comments the tester can provide about the bug

# 10. Test Schedule

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Work Breakdown Structure** | **Task** | **Planned Start** | **Planned Finish** | **Resource Name** |
| 1.2.5.1 | Hardware Test | 01 March 2015 | 17 April 2015 | Seth Skocelas |
| 1.2.5.2 | Unit Test | 20 March 2015 | 17 April 2015 | Brian Hasty |
| 1.2.5.3 | Component Test | 24 March 2015 | 17 April 2015 | Brian Hasty |
| 1.2.5.4 | Integration Test | 10 April 2015 | 24 April 2015 | Brian Hasty |
| 1.2.5.5 | System Validation Test | 17 April 2015 | 29 April 2015 | Brian Hasty |

Table 10‑1: Test Schedule

# 11. Approval

|  |  |  |
| --- | --- | --- |
| **Name** | **Signature** | **Date** |
| Manfred Huber |  |  |
| Gergeley Zaruba |  |  |
| Seth Skocelas |  |  |
| Shamikul Amin |  |  |
| Brandon Deen |  |  |
| Brian Hasty |  |  |
| Keyur Patel |  |  |

Table 11‑1: Approvals