**Client Application:** This is an app consisting UI. It will be on every print zone PC. It will start up with windows. Printed driver must be installed with this client app.

1. **Use case name:** Take user data

**Actor:** System

**Brief overview:** This app will take user data like user ID, User name, page left, total printed page, pending file names.

**Type:** Primary.

**Related Use case:** null.

|  |  |
| --- | --- |
| Actor action | System response |
| 1. User will login to windows with username and password, as usual. | **2.** System will take user NSU ID from windows user name, then flush location [a, b] (3.2) where file get stored temporarily and make API call to server app. |
| 3. Server app will bring User name, page left, total printed page count, pending files name (if present but printing not started yet and also from online), max file limit. | **4.** System will store those data on local (variables). |
| 5. | **6.** |

**Alternative course of action:** null

1. **Use case name:** Show user credentials.

**Actor:** System

**Brief overview:** This UI activity.

**Type:** Secondary.

**Related Use case:** Take user data (1), Show all print queue files (7). Catch files from own printer driver (9).

|  |  |
| --- | --- |
| Actor action | System response |
| 1. Show all stored data (got from 1.4) on UI. | **2.** Wait for any user activity like button click, print activity on printer driver (9). |
| 3. | **4.** |

**Alternative course of action:** null

1. **Use case name:** Print from any file.

**Actor:** User

**Brief overview:** There will be a printer driver installed on every client PC with this app, named “NSU\_RFID\_Printer”. That will be available just like other normal printers. User will print via this printer with their desired formatting (like page orientation, copies, page number etc.). It will just convert that to be printed file into desired orientation PDF file and store it to a temp location (without user intervention, location will be selected by us developer). To keep an abstraction, Client app will Move that pdf file into another hidden location.

**Type:** Primary.

**Related Use case:** Show all print queue files (7), Send file to server (10).

|  |  |
| --- | --- |
| Actor action | System response |
| 1. User will print any file using “NSU\_RFID\_Printer”. That printer driver will store it on a primary location[a] like other than Client app location. | **2.** System will check file limit and page limit and cut that file from that temp location, and store it into another dedicated location inside Client app installation folder, that will be known as print queue location[b]. That file will be renamed with ID\_filename and sent to server, with API call to notify server. |
| 3. | **4.** From that location, queued files (7) will be shown on UI. |

Location a:

Location b:%client\_app\_installationfolder%/print\_queue

**Alternative course of action:** null

1. **Use case name:** Drop PDF files.

**Actor:** User

**Brief overview:** User can drop their desired pdf file into a box (on UI), that will also be added into print queue after checking.

**Type:** Primary.

**Related Use case:** Show all print queue files (7), Send file to server (10).

|  |  |
| --- | --- |
| Actor action | System response |
| 1. User will drag and drop a finalized pdf file on a dedicated box in UI. | **2.** System will catch that file, checks either it is pdf or not, then copy it into Location[b] (3.2) |
| 3. | **4.** Also checks file limit and page limit before adding to queue. |

**Alternative course of action:** null

1. **Use case name:** Delete files from print queue.

**Actor:** User

**Brief overview:** User can delete print queued files before printing.

**Type:** Secondary.

**Related Use case:** Show all print queue files (7), Request for update data (11).

|  |  |
| --- | --- |
| Actor action | System response |
| 1. User will check on checkbox to select files and press delete button. | **2.** System will delete those file/files from queue and update data and UI. |
| 3. | **4.** Also system will request for update on server PC. (11) |

**Alternative course of action:** null

1. **Use case name:** Refresh.

**Actor:** User

**Brief overview:** User can refresh to repopulate their UI with updated data.

**Type:** Secondary.

**Related Use case:** Take user data (1).

|  |  |
| --- | --- |
| Actor action | System response |
| 1. User will press refresh button. | **2.** System does the whole thing described on 1, except cleaning location[a,b] (3.2) |
| 3. | **4.** |

**Alternative course of action:** null

1. **Use case name:** Show all print queue files.

**Actor:** System

**Brief overview:** User can view their queued files.

**Type:** Secondary.

**Related Use case:** null.

|  |  |
| --- | --- |
| Actor action | System response |
| 1. During log in System will get all file name from server, if file available online, system will request server to make those available offline. | **2.** System will show them on UI as a flow layout, with individual page count. |
| 3. | **4.** |

**Alternative course of action:** null

1. **Use case name:** Block user from printing

**Actor:** System

**Brief overview:** If user crosses page limit or cross max file count, they can print before clearing it.

**Type:** Secondary.

**Related Use case:** Catch file from own printer driver (9),

|  |  |
| --- | --- |
| Actor action | System response |
| 1. If user crosses max pending file (nonprinted) limit. | **2.** System will not receive any new file from driver (9) or drag drop (4). It will show warning to print those first. |
| 3. If user gets out of page. | **4.** System will not receive any new file from driver (9) or drag drop (4). It will show warning to refill page. |
| 5. | **6.** |

**Alternative course of action:** null

1. **Use case name:** Catch file from own printer driver.

**Actor:** System

**Brief overview:** While a file gets printed from any application, it will catch that file.

**Type:** Secondary.

**Related Use case:** Print from any file (3), Block user from printing (8).

|  |  |
| --- | --- |
| Actor action | System response |
| 1. User prints file using our driver. | **2.** Refer to 3.2. If any print command crosses any limit, system will not receive that file and show warning to UI (8). |
| 3. | **4.** |

**Alternative course of action:** null

1. **Use case name:** Send file to server.

**Actor:** System

**Brief overview:** After adding a file into queue, it will be sent to server for printing.

**Type:** Secondary.

**Related Use case:** Print from any file (3), Drop PDF files (4).

|  |  |
| --- | --- |
| Actor action | System response |
| 1. After adding a file into print queue. | **2.** System will send that file to server to be printed and make API call to notify Server. |
| 3. | **4.** |

**Alternative course of action:** null

1. **Use case name:** Request for update data.

**Actor:** System

**Brief overview:** For delete operation or new file insertion, it requires to notify the server.

**Type:**

**Related Use case:** Print from any file (3), Drop PDF files (4), Delete files from print queue (5), Send file to server (10).

|  |  |
| --- | --- |
| Actor action | System response |
| 1. File gets deleted from queue. (5) | **2.** System makes API call to abort file from printing and delete. |
| 3. New file gets inserted to queue. (3, 4) | **4.** System sends that file to server and makes API call for update. (10) |

**Alternative course of action:** null