Chapel Checker System Design

Jonathan Manos
Travis Pullen

Link to slides: https://docs.google.com/presentation/d/1_2MlwIvhUyuW6WtK---Ue_MwLdEPZZp0QIejWp8Ry8Q/edit?usp=sharing

Introduction

Goals for the Chapel Checking System performance and operation:

- Increase productivity of checkout at bigger chapel events
- Streamline the upload process of attendance information

Goals in the system development of Chapel Checking System:

- Develop the system in C# for Windows devices
- Research alternative platforms for chapel checking system implementation

Current vs Proposed Software Architecture

Current System Architecture:

- Pocket PCs
- Barcode Scanner
- Physically Transfer Information

Proposed System Architecture:

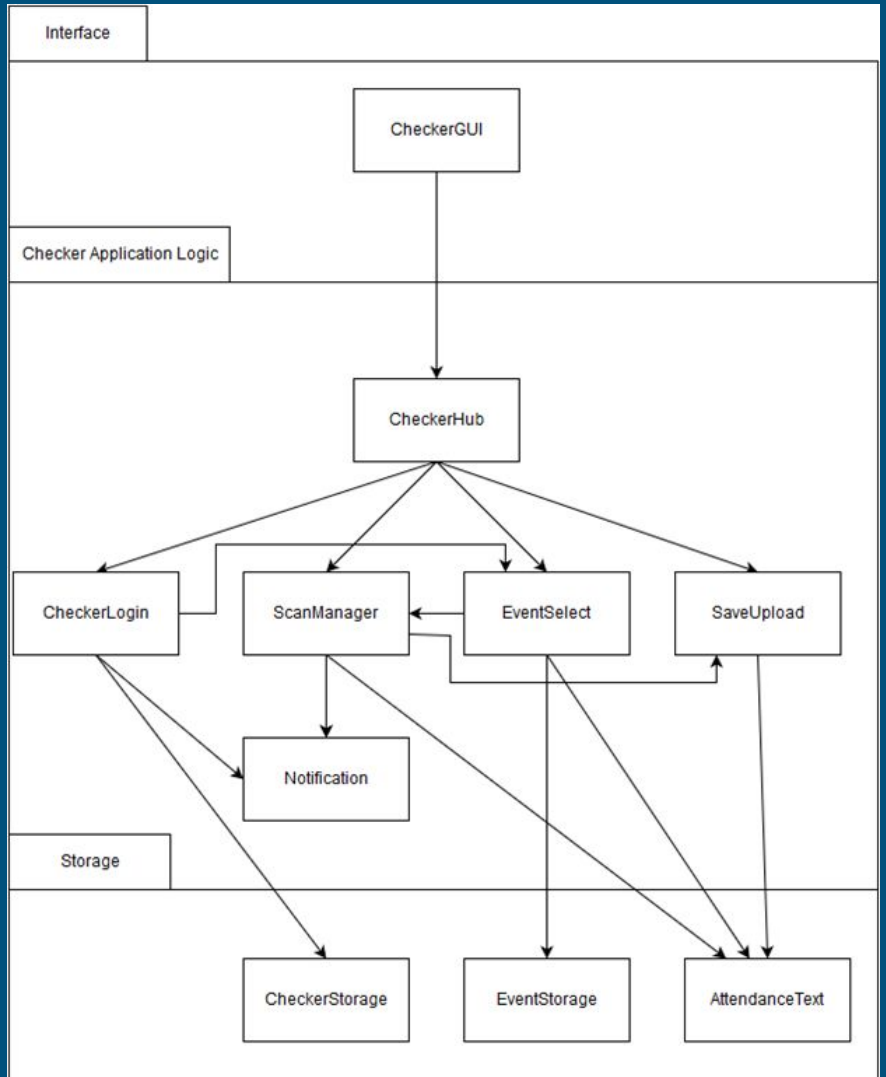
- Windows Device
- RFID Prox Reader
- Wi-fi connection to GordonNet

Subsystem Decomposition

No need for multiple subsystems.

Chapel Checking System can be decomposed into three parts:

1. Interface
2. Checker Application Logic
3. Storage


Interface
CheckerGUI
Checker Application Logic
CheckerHub
CheckerLogin
ScanManager
EventSelect
SaveUpload
Notification
Storage
CheckerStorage
EventStorage
AttendanceText

Hardware/Software Mapping

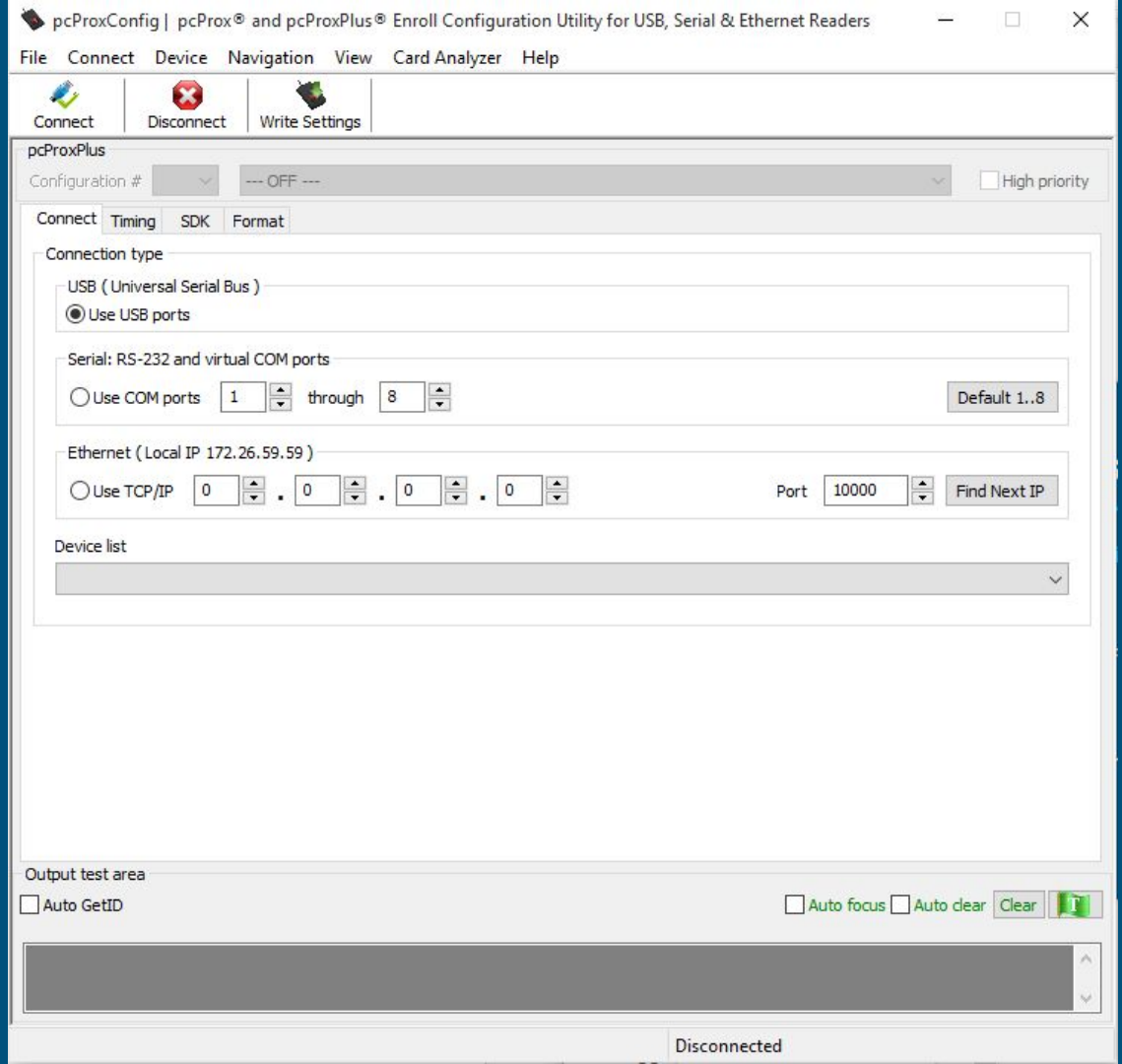
Main Hardware for the System:

- RFID Prox Reader USB Attachment

Software application that comes with device:

- pcProxConfig

Comes with SDK which comes with an API and examples.

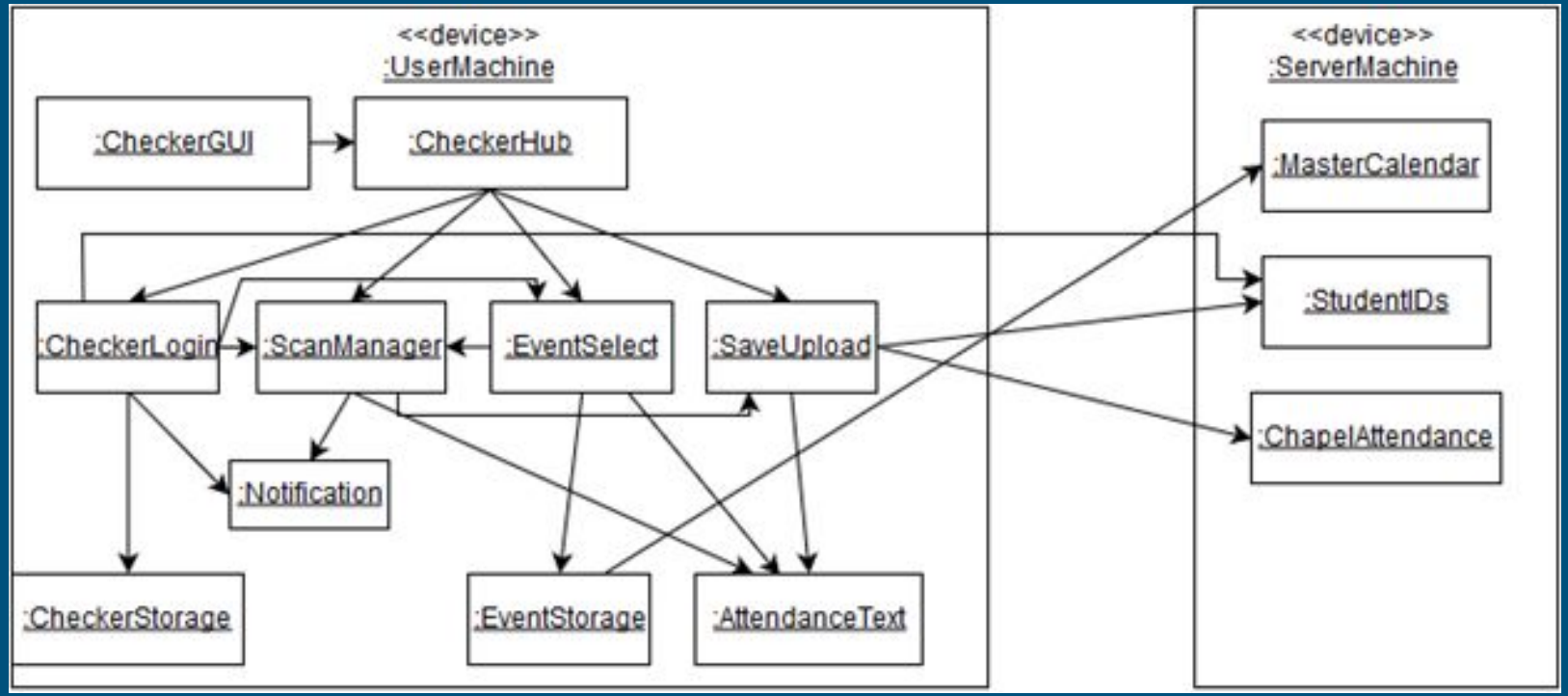

pcProxConfig | pcProx® and pcProxPlus® Enroll Configuration Utility for USB, Serial & Ethernet Readers
File
Connect
Device
Navigation
View
Card Analyzer
Help
Connect
Disconnect
Write Settings
pcProxPlus
Configuration #
--- OFF ---
High priority
Connect
Timing
SDK
Format
Connection type
USB (Universal Serial Bus)
Use USB ports
Serial: RS-232 and virtual COM ports
Use COM ports
1
through
8
Default 1..8
Ethernet (Local IP 172.26.59.59)
Use TCP/IP
0
0
0
0
Port
10000
Find Next IP
Device list
Output test area
Auto GetID
Auto focus
Auto clear
Clear
Disconnected

Hardware/Software Mapping

System mostly runs on the user system

Makes use of Gordon server in three areas:

1. Master Calendar
2. Student IDs
3. Student’s CLAW Credit (Go Gordon)


<<device>>
:UserMachine
:CheckerGUI
:CheckerHub
:CheckerLogin
:ScanManager
:EventSelect
:SaveUpload
:Notification
:CheckerStorage
:EventStorage
:AttendanceText
<<device>>
:ServerMachine
:MasterCalendar
:StudentIDs
:ChapelAttendance

Persistent Data Management

There are three areas of persistent data for the chapel checking system:

1. Approved Chapel Checkers ProxIDs Database
    a. Approved ProxIDs
2. CLAW Credit Events Database
    a. Event ID
    b. Name
    c. Date
    d. Time
3. Attendance Text Document from each event
    a. Checker’s ProxID
    b. Scanned ProxID
    c. datetimestamp
    d. No Credit Flag

Access Control and Security

Access:

Approved Chapel Checkers - Connects to Gordon list or local database

Security:

No Threats - only stores numbers of IDs and event information

Access Matrix:

Actors ↓ Objects →	Login	Event	Scan	SaveUpload
ChapelChecker	Sign in by scanning card	Select event from current date	Scan people's cards by tapping them	Click done to finalize event processing

Global Software Control

1. No connection issues
    a. If the computer is not connected to the internet it will store the numbers locally on a text file.
2. No synchronization and concurrency issues
    a. Operate individually and locally
    b. Utilize event-driven control
3. Scanning multiple cards at one time
    a. Only concurrency issue
    b. Can rescan card to make sure their information was read - system will check for duplicates

Boundary Conditions

System Start-up:

- Connects to Gordon Net
- Populates/Refreshes Event Database
- Checks for needed attendance text files needing upload

Errors - System Crash:

- Local Text File Saved - uploaded in background on next System Start-up

System Shut-down:

- Runs in background till event attendance uploaded successfully

The End

Questions?