

Florida International University
Knight Foundation School of Computing and Information Sciences

Software Engineering Focus

Final Deliverable

Project Title: NextCloud App: Sync CSV Userlist

Team Members: Hunter Sanchez, Adam Kobke, Augusto Rodriguez

Product Owner(s): Masoud Sadjadi

Instructor: Masoud Sadjadi

The MIT License (MIT)
Copyright (c) 2016 *Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Abstract

This comprehensive documentation serves as a detailed guide to the NextCloud App: Sync CSV Userlist, a robust system designed to simplify and streamline the process of managing FIU student accounts and groups within a Nextcloud environment. Developed to cater to the specific needs of FIU professors, the application provides a user-friendly interface that allows the user to efficiently upload student rosters, and automatically create or update user accounts on the Nextcloud platform.

Starting with an overview of the current system's limitations, the documentation outlines the purpose of the new system, highlighting its enhanced functionality and improved user experience. User stories offer insights into the system's capabilities and its evolution. The project plan section details the hardware and software resources, emphasizing the technology stack and tools utilized in the development process.

In the system design section, the documentation delves into the architectural patterns that lay the foundation for a modular and scalable application. It then proceeds with a thorough system and subsystem decomposition, explaining how the application is structured into distinct, cohesive units that interact seamlessly. The deployment diagram provided offers a visual representation of the application's runtime configuration, ensuring clarity in understanding the deployment architecture.

The application's codebase is explained through an exploration of design patterns, which articulates the reusable solutions employed to address common problems encountered during development. Additionally, system validation reports validate the functionality, performance, security, and usability of the application, ensuring it meets the predefined requirements and quality standards.

Overall, this documentation encapsulates the essence of the Nextcloud User Management Application, from conception to deployment, and serves as a vital reference for developers, stakeholders, and users alike. It ensures that all aspects of the system are well-documented and easily accessible for future maintenance, updates, and enhancements.

Table of Contents

INTRODUCTION	6
CURRENT SYSTEM	6
PURPOSE OF NEW SYSTEM	6
USER STORIES	
IMPLEMENTED USER STORIES	7
PROJECT PLAN	
HARDWARE AND SOFTWARE RESOURCES	9
SPRINTS PLAN	10
<i>Sprint 1</i>	10
<i>Sprint 2</i>	10
<i>Sprint 3</i>	10
<i>Sprint 4</i>	10
<i>Sprint 5</i>	10
<i>Sprint 6</i>	10
SYSTEM DESIGN	
ARCHITECTURAL PATTERNS	11
SYSTEM AND SUBSYSTEM DECOMPOSITION	12
DEPLOYMENT DIAGRAM	18
DESIGN PATTERNS	14
SYSTEM VALIDATION	16
APPENDIX	17
APPENDIX A - UML DIAGRAMS	17
<i>Static UML Diagrams</i>	17
<i>Dynamic UML Diagrams</i>	17
APPENDIX B - USER INTERFACE DESIGN	19
APPENDIX C - USER MANUALS, INSTALLATION/MAINTENANCE DOCUMENT, SHORTCOMINGS/WISHLIST DOCUMENT AND OTHER DOCUMENTS	20
.....	0

INTRODUCTION

NextCloud, a file hosting and collaboration platform, offers a solution for educational institutions like FIU to manage digital resources and collaboration. However, FIU professors face a significant challenge with this system - the manual addition of students. Each student must be individually added to NextCloud, a process that is not only prone to human error but also remarkably inefficient. Mistakes such as misspelled names or incorrect user details during data entry can lead to complications in user identification and access control.

Current System

There is currently no system for automatically adding / syncing users to NextCloud. All users must be added manually.

Purpose of New System

To address the issue of having to manually add users to NextCloud by offering an automated solution to the student integration process.

USER STORIES

The following section provides the detailed user stories that were implemented in this iteration of the NextCloud: Sync CSV Userlist project. These user stories served as the basis for the implementation of the project's features. This section also shows the user stories that are to be considered for future development.

Implemented User Stories

Sprint 1

- User Story 1: Learn how to use basic Nextcloud features.
- User Story 2: Finish setting up a development environment.

Sprint 2

- User Story 1: As a user I want to be able to specify the location of a csv file on my nextcloud server
- User Story 2: As a user, I want to be able to read csv data into nextcloud
- User Story 3: as a user, i want a tab to access the sync user data app
- User story 4: as a user i want to view the data being imported in nextcloud

Sprint 3

- User Story 1: As a user I want to be able to specify the location of a csv file on my nextcloud server.
- User Story 2: As a user I want new CSV data to be able to update existing users.
- User Story 3: As a user I want to be able to see which groups in the csv file don't exist on nextcloud.
- User Story 4: As a user I want to have a mapping on course codes to more readable names.
- User Story 5: As a user I want to be able to handle users with multiple majors.

Sprint 4

- User Story 1: As a user I want to be able to update groups based on a given mapping.
- User Story 2: As a user, I want to be able to select my mappings csv file through Nextcloud.
- User Story 3: As a user I want to be able to use xlsx files from PantherSoft to update the users/groups.
- User Story 4: As a user I want to be able to use csv files from canvas to update the users.

- User Story 5: As a user, I want my mappings and files to remain loaded on the app when the user navigates to a different menu.

Sprint 5

- User Story 1: As a user, I want to be able to load my mappings from a default file in nextcloud.
- User Story 2: As a user, I want my users information to update properly based on matching usernames.
- User Story 3: As a user, I want my user to be able to select the input file type from a list of different file types (canvas, panthersoft, cap2 grouplist, default roster) .
- User Story 4: As a user, I want the application to work for a variety of different cases.
- User Story 5: As a user, I want to see detailed information about the results of my sync.

Sprint 6

- User Story 1: As a user I want to be sure all groups are created correctly.
- User Story 2: As a user, I want to be sure all users are created correctly.
- User Story 3: As a user I want to see a log of each individual change to nextcloud (users added, groups created, etc.).
- User Story 4: As a user I want to see a pleasant user interface.
- User Story 5: As a user, I want both Nextcloud file uploads and local file uploads to work properly.

Pending User Stories

None

PROJECT PLAN

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

Hardware:

Computer

Software

NextCloud

Web hosting service (Linode, AWS, Google Cloud)

Development IDE (VSCode, JetBrains)

Sprints Plan

Sprint 1

During Sprint 1, we will learn about Nextcloud features and setup our environment. The velocity of the team was estimated to be 2 User Stories, worth 3 hours a day.

Sprint 2

During Sprint 2, we will develop a feature that allows the user to import CSV data into Nextcloud. The velocity of the team was estimated to be 4 User Stories, worth 3 hours a day.

Sprint 3

During Sprint 3, we will work on adding features involving syncing CSV data with Nextcloud. The velocity of the team was estimated to be 5 User Stories, worth 3 hours a day.

Sprint 4

During Sprint 4, we will work on features to allow for more diverse files and data to be uploaded. The velocity of the team was estimated to be 5 User Stories, worth 3 hours a day.

Sprint 5

During Sprint 5, we will work on ensuring that the quality of the data extracted from the input files is adequate, as well as adding new mapping features. The velocity of the team was estimated to be 5 User Stories, worth 3 hours a day.

Sprint 6

During Sprint 6, we will work on ensuring the information is added to Nextcloud, and that there are no bugs in the application. The velocity of the team was estimated to be 5 User Stories, worth 3 hours a day.

Sprint 7

During Sprint 7, we will work on developing our final presentation materials.

SYSTEM DESIGN

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

Architectural Patterns

Overview

The Nextcloud Sync CSV Userlist application follows several architectural patterns, primarily aligning with the Model-View-Controller (MVC) architecture in its backend (PHP) and a Client-Server architecture overall. The application facilitates user and group management in a Nextcloud environment, processing user list files, and interacting with the Nextcloud API.

1. Client-Server Architecture

- **Client (Frontend) - JavaScript (script.js):**
 - The JavaScript file manages client-side logic, including user interface interactions, file processing, and asynchronous communication with the server.
 - The use of asynchronous functions (**async**) and event listeners enables a dynamic, event-driven user interface.
 - Functions like **processContent**, **handleCSVContent**, and **handleXLSXContent** enable file processing capabilities, preparing data for server-side interaction.
- **Server (Backend) - PHP (PageController.php):**
 - The PHP file represents the server-side component, handling HTTP requests and interacting with the Nextcloud API.
 - Functions like **createUser** and **getAllGroups** are facilitated by typical CRUD (Create, Read, Update, Delete) operations related to user and group management in Nextcloud.
 - The server component is responsible for the logic and data manipulation, abstracting these details from the client.

2. Model-View-Controller (MVC) Pattern

- **Controller (PageController.php):**
 - The **PageController** class extends a base controller.
 - Functions in this controller handle incoming requests, process them, and return appropriate responses, either as rendered templates or JSON data.
- **Model and View:**
 - The MVC pattern promotes separation of concerns, enhancing maintainability and scalability.

3. Asynchronous Programming and Event-Driven Model

- The use of asynchronous JavaScript (**async/await**) for handling file processing and server communication facilitates non-blocking operations, essential for responsive web applications.
- The application likely employs an event-driven model on the client-side, reacting to user actions like file uploads, format selections, etc.

4. Service-Oriented Architecture (SOA) / API-Driven Design

- Interaction with the Nextcloud API from the PHP backend aligns with SOA principles, where the backend acts as a client to the Nextcloud service.
- The API-driven approach in the backend facilitates modularity and separation of the web application from the Nextcloud service, allowing for flexible development and potential integration with other services or APIs.

Conclusion

The Sync CSV Userlist application's architectural design combines Client-Server, MVC, and SOA/API-Driven patterns, resulting in a separation of concerns, modular design, and maintainable code structure. This architecture supports scalability and adaptability to changing requirements or integration with additional services in the future.

System and Subsystem Decomposition

System Decomposition

1. Frontend System (Client-Side - JavaScript)

- **Primary Role:** User Interface Interaction and Presentation Layer
- **Components:**
 - **User Input Handlers:** Functions that capture and process user inputs (e.g., csv file uploads, format selections).
 - **File Processing Utilities:** Functions that parse and process user list files.
 - **Asynchronous Data Handlers:** Functions that handle asynchronous operations, primarily communicating with the backend.
 - **DOM Manipulators:** Scripts that dynamically modify the Document Object Model (DOM) based on user actions or data received from the backend.
- **Interactions:**
 - Interacts with the backend via HTTP requests (AJAX).
 - Processes user inputs and updates the UI accordingly.

2. Backend System (Server-Side - PHP)

- **Primary Role:** Business Logic Processor and Data Management Layer
- **Components:**
 - **Request Handlers (PageController):** Functions within **PageController.php** that handle various HTTP requests from the frontend.
 - **Nextcloud API Interactors:** Components that interact with Nextcloud's API for CRUD operations on users and groups.
- **Interactions:**
 - Receives requests from the frontend.
 - Performs necessary operations on Nextcloud and sends back responses.

Subsystem Decomposition

1. Frontend Subsystems

- **File Format Handlers:** Subsystems for handling different file formats like CSV and XLSX.

- **Group and User Data Processors:** Dedicated subsystems for processing group and user information extracted from files.
- **UI Update and Notification System:** Subsystem for updating the user interface in response to processing results or backend responses.

2. Backend Subsystems

- **User Management Subsystem:** Handles all operations related to user management, such as creation and updates.
- **Group Management Subsystem:** Manages group-related operations, including group creation, renaming, and deletion.
- **Data Validation Subsystem:** Ensures the integrity and validity of data being processed or sent to Nextcloud.

Communication and Data Flow

- The frontend and backend systems communicate over HTTP, with the frontend sending requests and the backend sending responses (JSON or HTML).
- Data flow typically starts from the frontend, where user inputs are captured and processed. The processed data is then sent to the backend for further actions.
- The backend interacts with Nextcloud's API, performing necessary operations and relaying results back to the frontend.

Design Patterns

Structural Patterns

1. Module Pattern:

- The JavaScript file encapsulates functionality into modules. This is enabled by the functions that are self-contained with specific responsibilities, such as handling CSV content and XLSX content.

2. Facade Pattern:

- The PageController acts as a facade for the Nextcloud API, providing a simplified interface for the frontend to interact with the backend. The frontend does not need to know about the complexities of the API interactions, which are handled by the backend PHP functions.

3. Composite Pattern:

- The UI elements such as the buttons and input fields follow a Composite pattern, since they are treated uniformly with other UI components.

Behavioral Patterns

1. Observer Pattern:

- The use of event listeners in the JavaScript file created an Observer pattern. UI elements like buttons and input fields are "observed" for user actions (events), and corresponding handler functions are "notified" to respond to those actions.

2. Command Pattern:

- Actions in the UI, such as uploading a user list or group mappings, can be associated with the Command pattern since there are JavaScript functions that encapsulate these actions as objects.

3. Strategy Pattern:

- Different methods for processing the user list (**processDefaultCSV**, **processCanvasCSV**, etc.) involves the Strategy pattern. Each function represents a different strategy for parsing and handling the file content, and the appropriate strategy is selected based on the file format.

SYSTEM VALIDATION

Validation Strategies and Methodologies

1. Functional Testing:

- **Purpose:** To ensure that the application functions as expected and all user requirements are met. This includes testing the file upload process, prefixes, mappings, user/group management functionalities, and UI interactions.
- **Methods:**
 - Unit Testing of individual functions in **script.js** and **PageController.php**.
 - Integration Testing to verify that different parts of the application work together.
 - System Testing of the application as a whole, in an environment that mimics production.

2. User Interface Testing:

- **Purpose:** To verify that the UI is intuitive, responsive, and provides the correct output for user actions.
- **Methods:**
 - Usability Testing to assess the ease of use and user satisfaction.

3. Performance Testing:

- **Purpose:** To confirm that the application performs well under expected loads.
- **Methods:**
 - Load Testing to simulate the processing of large user list files.
 - Stress Testing to evaluate system behavior under peak loads, especially when interacting with the Nextcloud API.

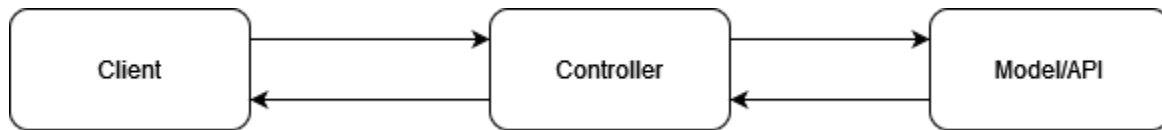
Validation Criteria

1. **Correctness:** The application accurately processes user list files and manages users/groups in Nextcloud as specified.
2. **Reliability:** The application performs reliably without errors under normal operating conditions.
3. **Usability:** The UI is user-friendly, and the application is easy to navigate and operate.
4. **Performance:** The application has acceptable response times and handles multiple simultaneous operations efficiently.
5. **Maintainability:** The code is well-documented, and the system is easy to update and maintain.

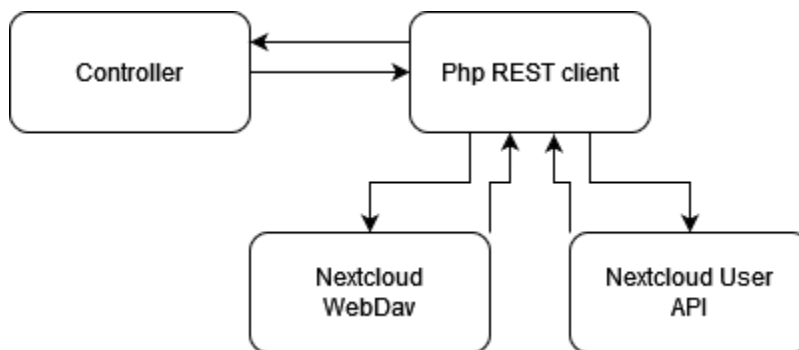
APPENDIX

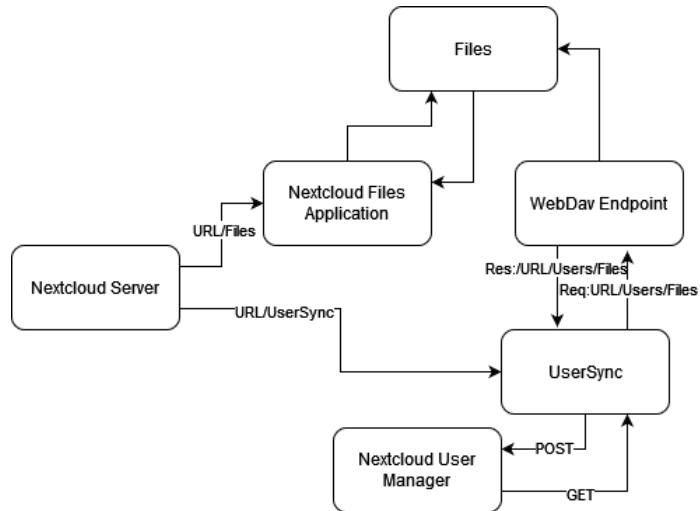
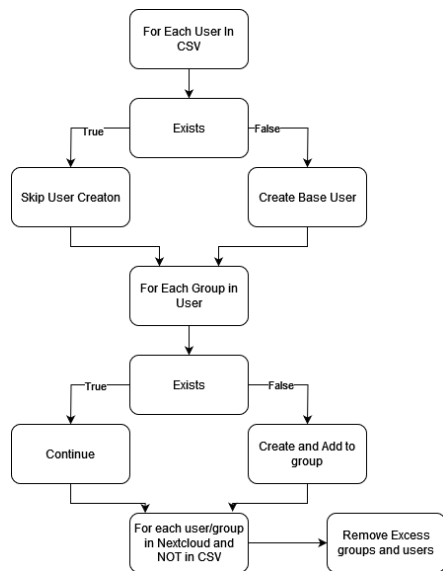
Appendix A - UML Diagrams

MVC

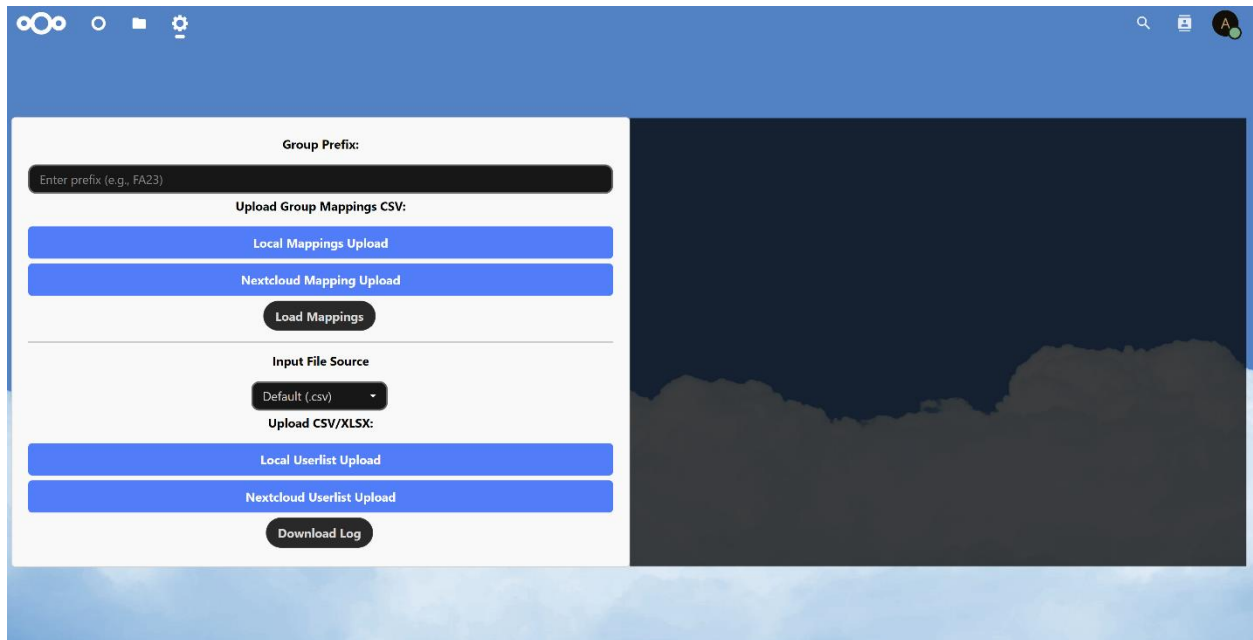


General backend structure



Main deployment diagram**Main UserSync Dataflow**

Appendix B - User Interface Design



Appendix C - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents

UserSync

Easy Nextcloud user management

UserSync is the easiest way to manage and sync Nextcloud users from FIU student rosters.

With CSV files generated from Canvas, or XLSX files generated from PantherSoft and personal Capstone roster files, we can create, modify, and update users and groups based on updated CSV/XLSX data.

Table Of Contents

Click on a link to jump to the section of interest

- [Developers](#)
- [Installation](#)
- [Usage](#)
- [Limitations](#)

Developers

This application was developed for CIS4951 final project under the leadership of Prof. Masoud Sadjadi

Lead Developer - [Hunter Sanchez](#) Developer - [Adam Kobke](#) Developer - [Augusto Rodriguez](#)

Installation

The installation of UserSync is quite simple if you follow the steps provided

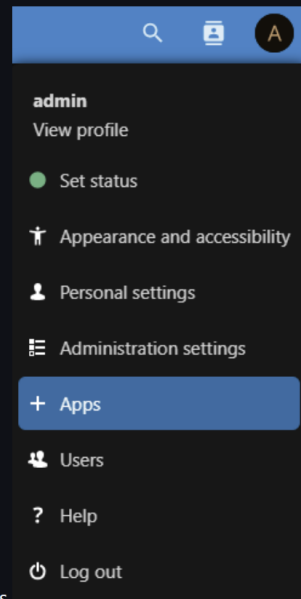
1. SSH Into the server that is running Nextcloud. This can be done in the MacOS terminal, Windows Powershell, or external programs such as PuTTY. Then cd into the directory hosting nextcloud. This directory should be called something like `Nextcloud-docker-dev`
2. Once in the directory the file structure should look something like this

```
kalesy@TwinkPad:~/nextcloud-docker-dev$ ls
Makefile  bootstrap.sh  docker          example.env  package-lock.json  renovate.json  workspace
README.md data          docker-compose.yml  node_modules  package.json       scripts
kalesy@TwinkPad:~/nextcloud-docker-dev$
```
3. Next cd into the application files for Nextcloud. These should be located at `.../Nextcloud-docker-dev/workspace/server/apps-extra`. This is the directory that Nextcloud stores your applications at.
4. In the apps-extra directory run the command `git clone https://github.com/Akobke/usersync.git` to download the application to the server
5. cd into the now created usersync folder and run the following command `npm i` This will install all of the required dependencies to run the application

In Nextcloud configuration

In this section we will go over the configuration required in your nextcloud instance

1. First navigate to your files application and create a new file named `UserSyncConfig`. This name is **!!CASE SENSITIVE!!**. In this file you can upload your group name mappings, and a plain .txt file with your group prefix, examples of which will be provided in the [usage](#) section.



2. Now press on your user icon in the top right of the page and navigate to apps
3. Inside of apps, navigate to the UserSync option and click on it, a options window should open on the right
4. Locate the field that says `Limit to groups`, and select which groups you would like to access the app. We recomend only administrators or highly trusted individuals

Usage

Setting Defaults

The usage of UserSync is quite straight forward, if you dont already have your desired group mappings you can get example mappings [here](#) If you dont have any mappings that is fine, you can upload a one time mapping in the application. To upload your mappings for default use, simply navigate to your `UserSyncConfig` file and upload your mappings **Be sure to rename your mappings to `groupmapping.csv`**. For your group prefix, once again you can change this on a case by case basis in the user interface of the application, or you can upload a default prefix in the `UserSyncConfig` file and name it to `groupprefix.txt`

For the group CSV, it follows a simple format (fake data)

Course	-	Group	SID	Username	First Name	Middle Name	Last Name	Groups
cis:4951	1	u01	6256456	msje056	John	William	Smith	engineering - computer science - bs
cis:4951	1	u01	6304986	jfkwe206	Sarah	Houghs	Robins	engineering - computer science - bs
cis:4951	1	u01	609483	lakej676	Robert	Downy	Junior	engineering - computer science - bs/mathematics - mn

Groups are seperated with the `/` as a delimiter

Application Fields

In the application you can see a few fields, all of which I will explain

- **Group Prefix** This field is where you define the group prefix if you would like to override the default you specify in `UserSyncConfig`
- **Group Mapping CSV** This is where you would upload your group mapping CSV. These can either be uploaded locally, or uploaded and accessed through NextCloud
- **Input File Source** This is where you define the formatting of the CSV/xlsx file (based on the source), the options currently supported are
 - Default (.csv file - custom to our CIS course, colon delimited)

course	number	-	unit	PID	email	first	last	-	major
cis	4951	1	u01	6543210	jdoe012@fiu.edu	john	doe	-	engineering - computer science - bs
cis	4951	1	u01	6543211	jdoe013@fiu.edu	jane	doe	-	engineering - computer science - bs/mathematics - mn

- Canvas (.csv file, comma delimited)

Last	Other Information(Semi colon delimited)...
Doe	John;000000;6543210;jdoe012;...
Doe	Jane;000000;6543211;jdoe013;...

- PantherSoft (.xlsx file, comma delimited)

-	PID	email	Name	Graded	-	Major	Level
-	6543210	jdoe012@fiu.edu	Doe,John	Graded	2	Engineering - Computer Science - BS	College Senior
-	6543211	jdoe013@fiu.edu	Doe,Jane	Graded	2	Engineering - Computer Science - BS/Mathematics MN	College Senior

- Cap2 Groups (.xlsx file - custom to our CIS course, comma delimited)

-	-	-	-	-	-	-	email	Name	Field	Course	Extra Info
-	-	-	-	-	-	-	jdoe012@fiu.edu	Doe,John	Computer Science	Capstone II	Usersync App
-	-	-	-	-	-	-	jdoe013@fiu.edu	Doe,Jane	Computer Science	Capstone II	Different App

- CSV Upload This is where you upload the CSV containing the user data
- Download Log Downloads all the log information (on the right side) locally to a .txt file.

Application Usage

Using the application if both defaults are provided is simple 1. Upload the CSV/XLSX file with the Local File Upload button (or Nextcloud File Upload), and wait for the users to be created (up to 5 minutes for large groups > 700)

- To use a custom prefix, simply specify the prefix in the **Group Prefix** field
- To use custom mappings, simply local upload, or choose a file from nextcloud, and press the **Load Mappings** button
- If you dont provide any custom mappings, the group names are going to be exactly as they are displayed in the original CSV file
- If you would like to simply update user groups or information, simply upload a new user CSV file
- If you would like to rename groups, simply reupload a new mappings CSV and press **Load Mappings**

Limitations

Unfortunately this application has a few limitations that I would like to address First and foremost, if you are not Prof. Masoud Sadjadi, this application is likely of no use to you.

Group Renaming

- Any groups that are created by UserSync can not be renamed from the application itself if you would like to maintain the ability to remap them at a later date. This is due to how NextCloud handles groups and was not really up to us

Auto Mapping Uploading

- Auto Mapping uploading unfortunately won't be a feature. Due to how WebDav and Nextcloud handle same origin HTTP requests this simply was not feasible to do without a proxy server

Default User Passwords

- Users created with UserSync will have the default password of `password` and are able to keep the default password without changing it. This can be a security issue, and it is **STRONGLY** urged to make your users change their passwords as soon as they log in

File formatting

- The application assumes that the formatting of each file (Default, Canvas, Panthersoft, or CAP Groups) remains the same over time. Any changes to the file type (.csv to .xlsx or .xlsx to .csv), or the file structure could impact the ability of the application to process those files