

Design Document

2021

## 

## Contents

[**Introduction**](#_n8se6ii2v8re) **2**

[**High Level Functionality**](#_zej1dlyo05nd) **2**

[**Architecture**](#_2wq5x3ynvdxf) **and system design 3**

[**Data storage**](#_tvh65fxf0wm7) **9**

[**Security Implementation**](#_3n6cqlsqcn6z) **11**

## 

# Introduction

Handin is a python based program that serves to simplify assessing students' coding assignments. Handin provides a way for students to submit assignments and get instant results on their work. Handin also reduces a lecturer's work load by autonomously assessing their students' assignments for them.

# High Level Functionality

Handin is composed of three user types: Lecturer, Student, Administrator.

Administrator:

The handin administrator creates modules by entering its module code and setting its access rights, as in who has access to alter the modules material, by setting a username for the lecturer and TA’s involved. The administrator also had to enter the lecturers and TA’s emails so that a handin thin client can be sent to them.

Lecturer:

Handin allows lecturers to register a module that only they, and other people they nominate, have access to to alter its material.

Lecturers can then add weekly or once off assignments for their students using a params file that will let handin know what commands to execute and what the expected output is.

Lecturers also include a class list that consists of the students taking the module.

Lecturers can check how their students are getting along through viewing their students scores for each assignment they have set.

Lecturers can do all this through running their handin thin client that would have been emailed to them once the administrator has set up their first module. From then on for multiple modules they can continue to use the same handin thin client by entering the code of the module they wish to alter.

Students:

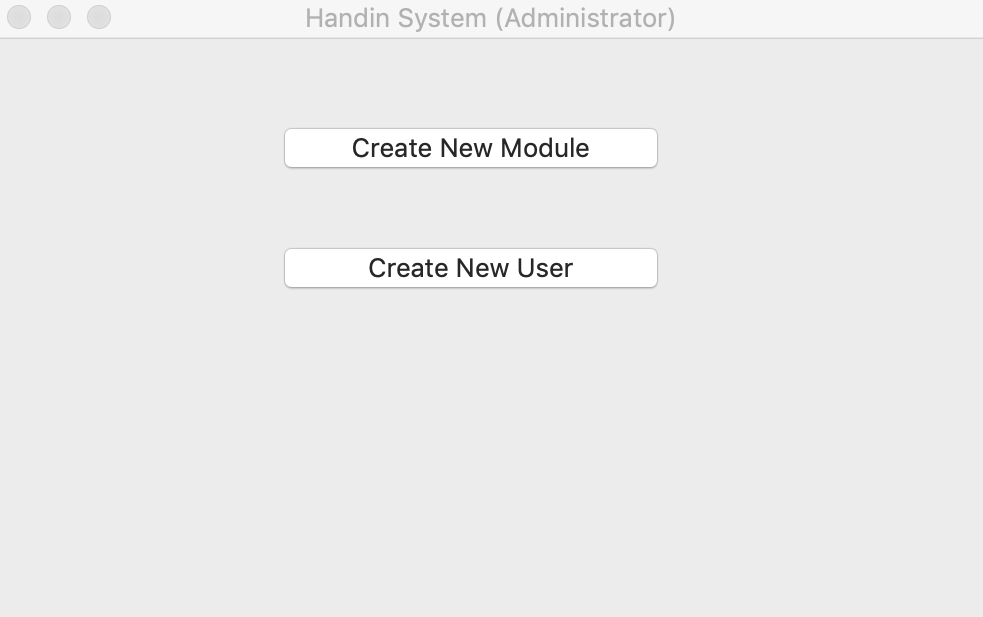
Handin allows students to register for their modules by entering their students id and email, they will then be sent out a generic handin thin client with their student id hard coded in.

Handin provides a way for students to submit their assignments without having to wait for their lecturers to correct them, instead they will get instant scores on their work. They will run the thin client and enter the module code and the code of the assignment they wish to hand in.

# Architecture and system design

*Administrator:*

The administrator runs the src/h4a.py program.



Administrator main window.

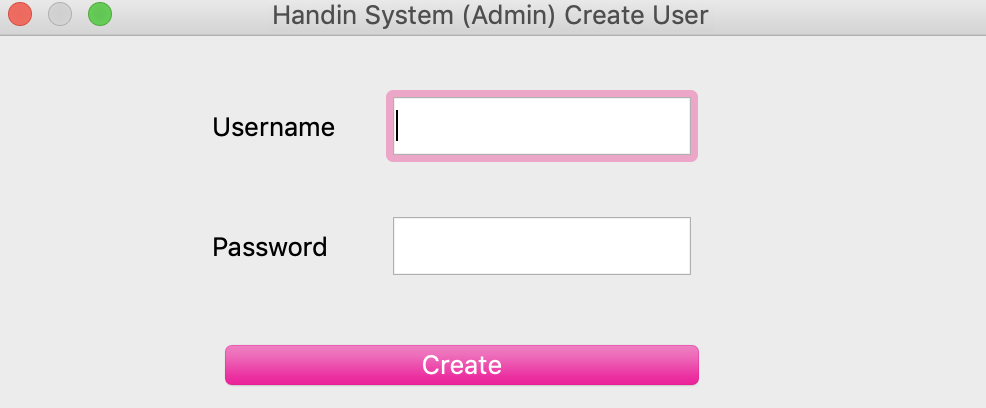
To add:

Add user to existing module access list.

Create a thin client.

Creating a user:

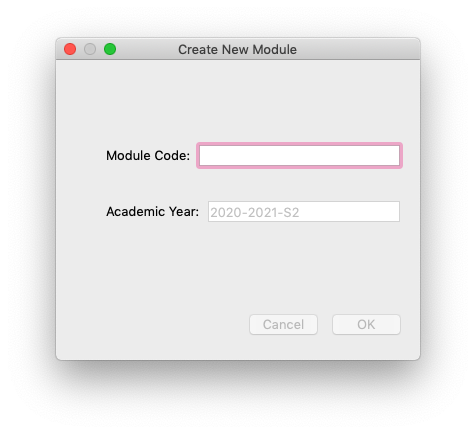
To create a new user the administrator clicks the “Create New User” button.



The Create User window appears with the required fields. The password will be hashed before it's stored.

Creating a module:

To create a module the administrator clicks the “Create New Module” button.



The “Create New Module” window appears with fields for the required information.

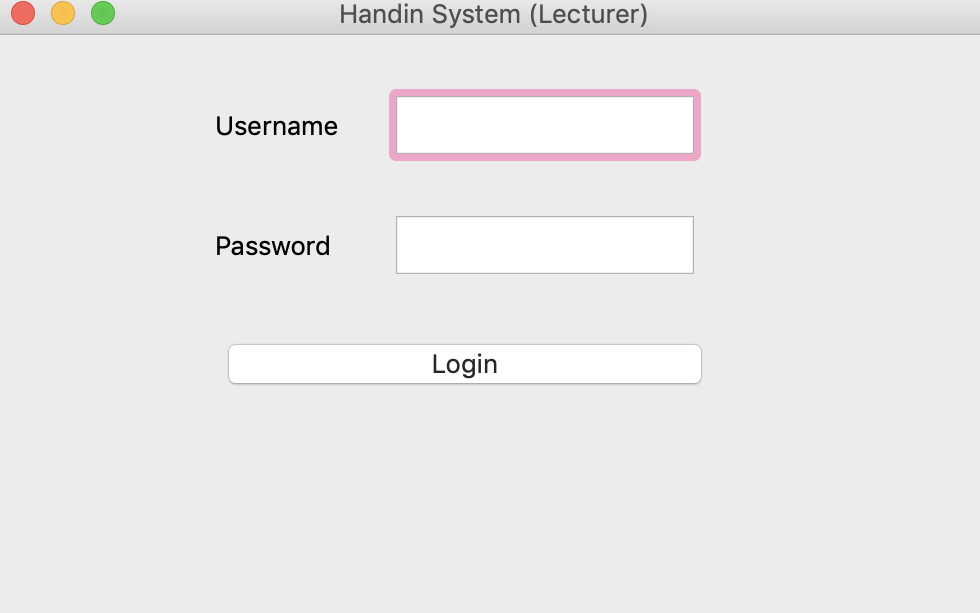
Files interacted with:

* ui/impl/handin\_admin\_main\_window.py contains the python code of the administrator main window ui.
* ui/impl/create\_new\_module\_main\_window.py contains the python code of the create new module window ui.
* ui/impl/create\_new\_user\_dialog.py contains the python code of the create new user window
* src/password\_security.py contains the code to hash the password.
* src/const.py contains variables and methods used across users.

*Lecturer:*

The lecturer runs the h4l.py program. First thing a lecturer needs to do is login to get access to other features.

Login:



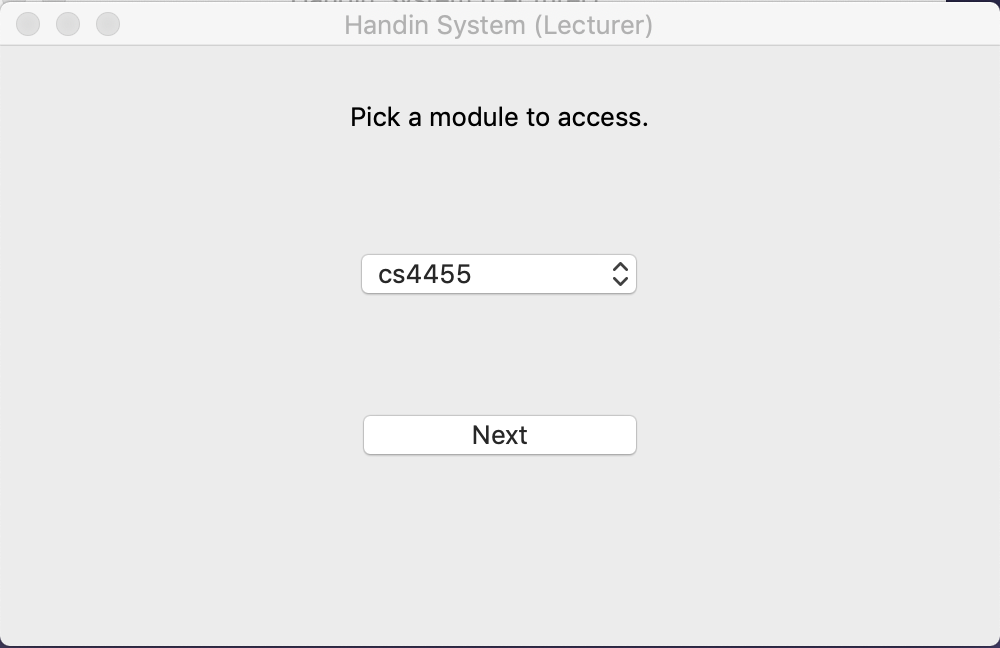
User enters their login credentials and if correct they are able to progress on to the next window.

Files interacted with:

* ui/impl/handin\_lecturer\_login.py contains the python code for the login window.
* src/password\_security.py contains the code to check the password against hashed password.

Pick module:

After a lecturer logs in they must pick the module they would like to work on.



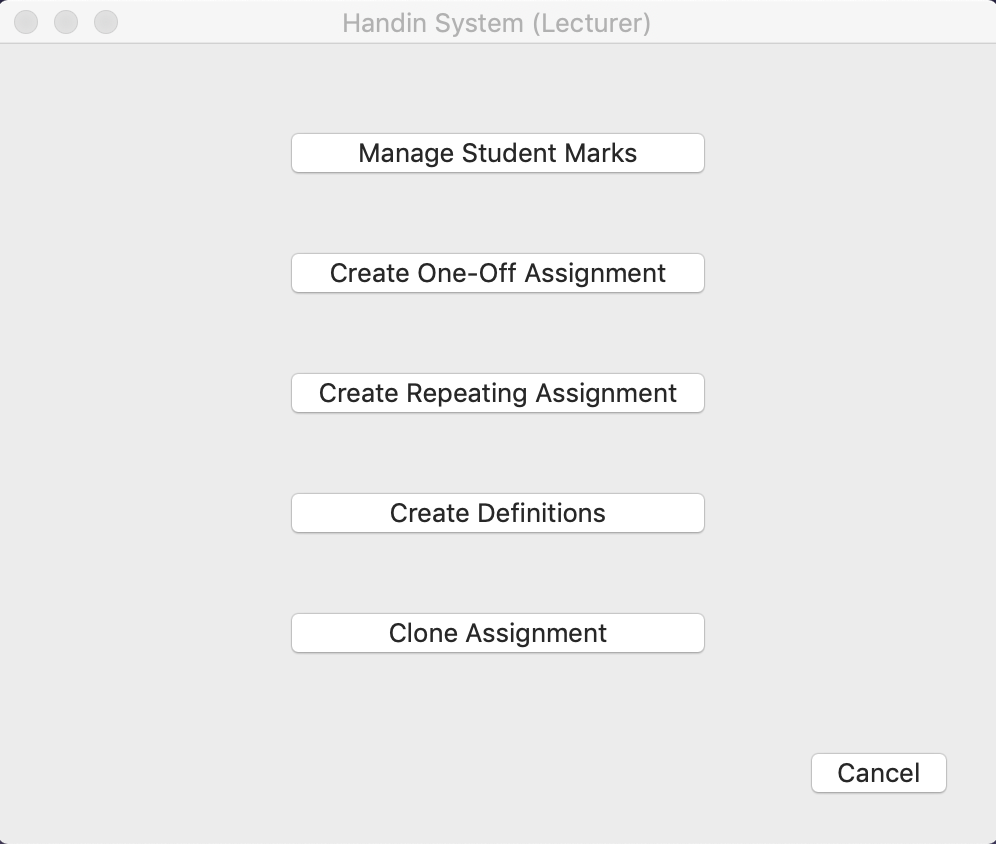
The user is presented with a combo box that only contains the modules that they have access to.

Files interacted with:

* ui/impl/pick\_module\_dialog.py contains the python code for pick module window.

Module functions:

After the lecturer has logged in they choose to carry out any of the following functions on the main dialog to the module they previously choose.

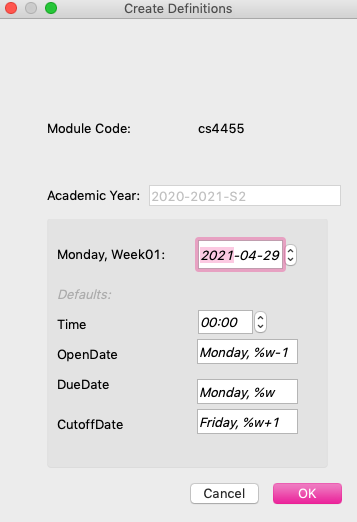


Lecturer main module dialog.

To add:

Class list.

Setting up a module:

After a module has been created by the administrator it is up to the lecturer to continue with the setup of that module.The lecturer clicks the “Create Definitions” button.

The definitions are used for weekly recurring assignments.

Monday, week01 lets handin know what the date of the first week of the module is. It will then calculate the dates of all the weeks to follow.

Time tells handin at what time of the due day to start penalising students submissions.

openDate tells handin what day submission for each week's assignment opens.

DueDate tells handin what day assignments must be submitted, for each week’s assignments, after that day students submissions will be penalised.

cutoffDate tells handin what day to stop accepting students submissions.

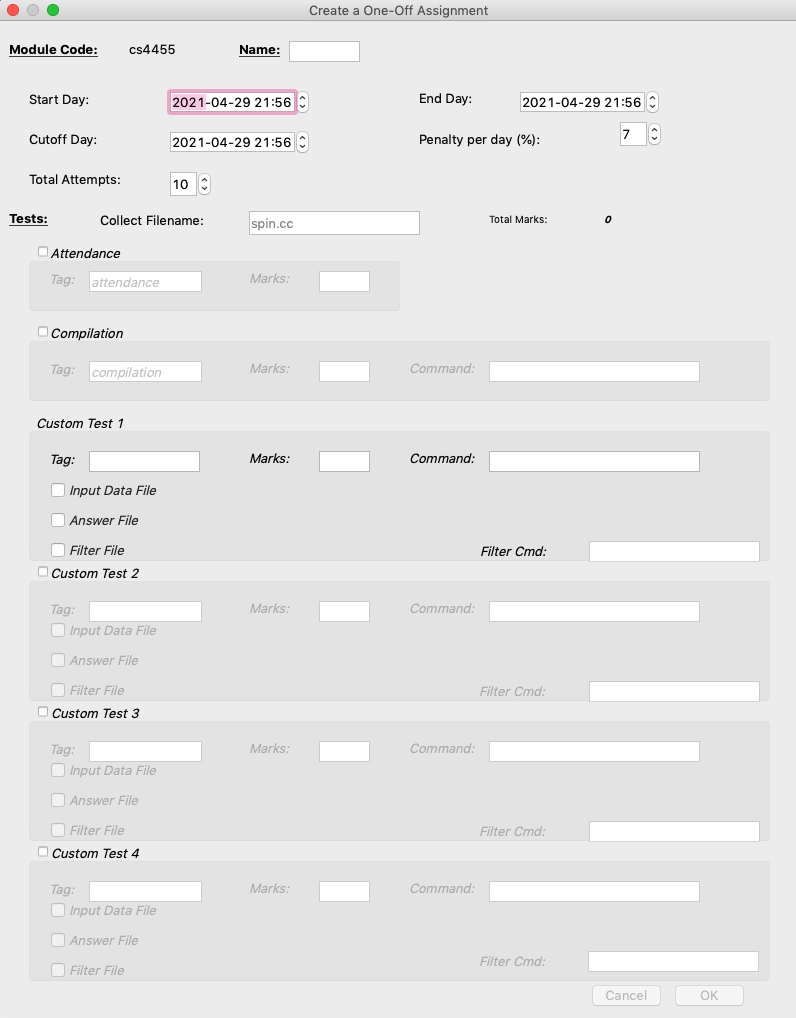
To add: penalty per day and total attempts.

Files interacted with:

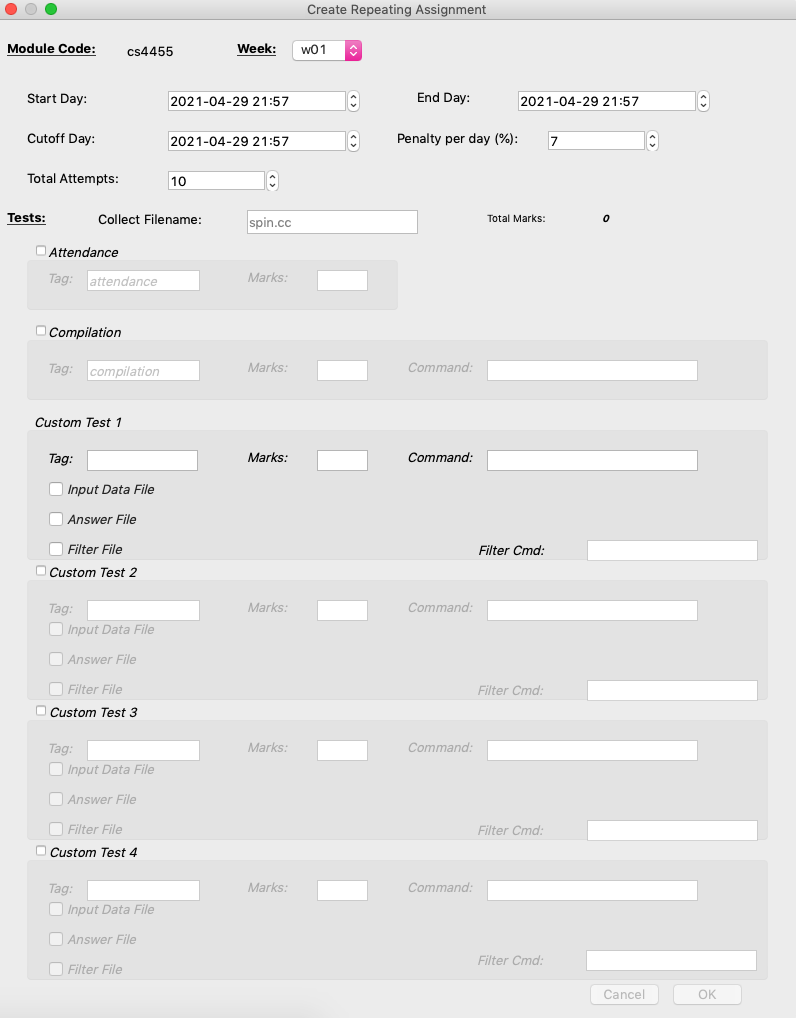
* ui/impl/handin\_lecturer\_mian\_window.py contains the python code of the lecturer main window ui.
* ui/impl/create\_definitions.py contains the python code of the create definitions window ui.
* src/const.py contains variables and methods used across users.

Creating assignments:

A lecturer can create three ways a lecturer can create assignments: a one-off assignment, a repeating assignment, or clone a previous assignment.



To create a one-off assignment the lecturer clicks the “Create One-off Assignment” button and is presented with the one-off assignment dialog.

To create a repeating assignment the lecturer clicks the “Create Repeating Assignment” button is presented with the repeating assignment dialog.

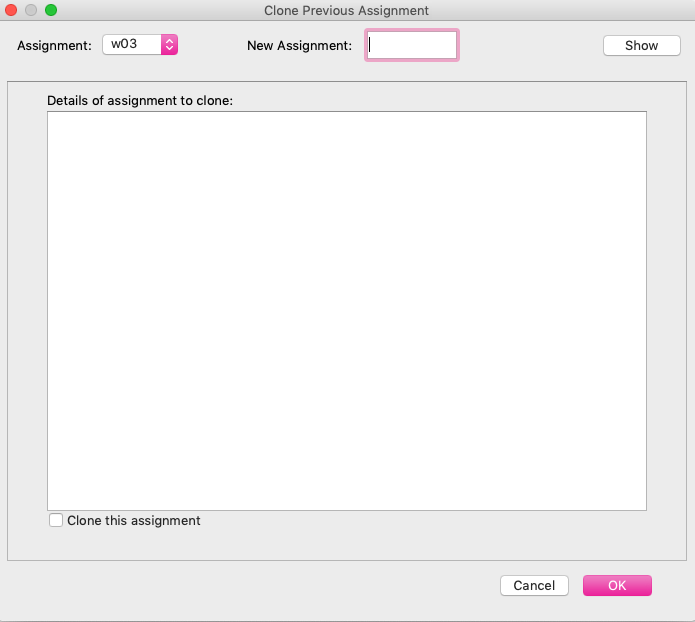
To create either type of assignment the lecturer has to enter the file name handin should expect to get from the students submission and also the tests they would like handin to run on the file. There are options to give marks for attendance (the file being presented), compilation (that the file compiles with no errors) and then up to 4 custom tests. The lecturer sets the marks for these.

For each custom test the lecturer can specify a tag and a command to run. They can also upload an input data file (for data needed to test the program), an output data file (for the expected output of the students program) and a filter file specified along with a filter command.

For one off assignment the lecturer has to specify the code of the module and then enter a name.

For repeating assignments the lecture has to specify the module code and then specify the week.

Start day, end day, cutoff day, penalty per day and total attempts are presented from the definitions file for the lecturer to review or override for certain circumstances.



To clone an assignment the lecturer clicks the “Clone Assignment” button and is presented with the clone assignment window.

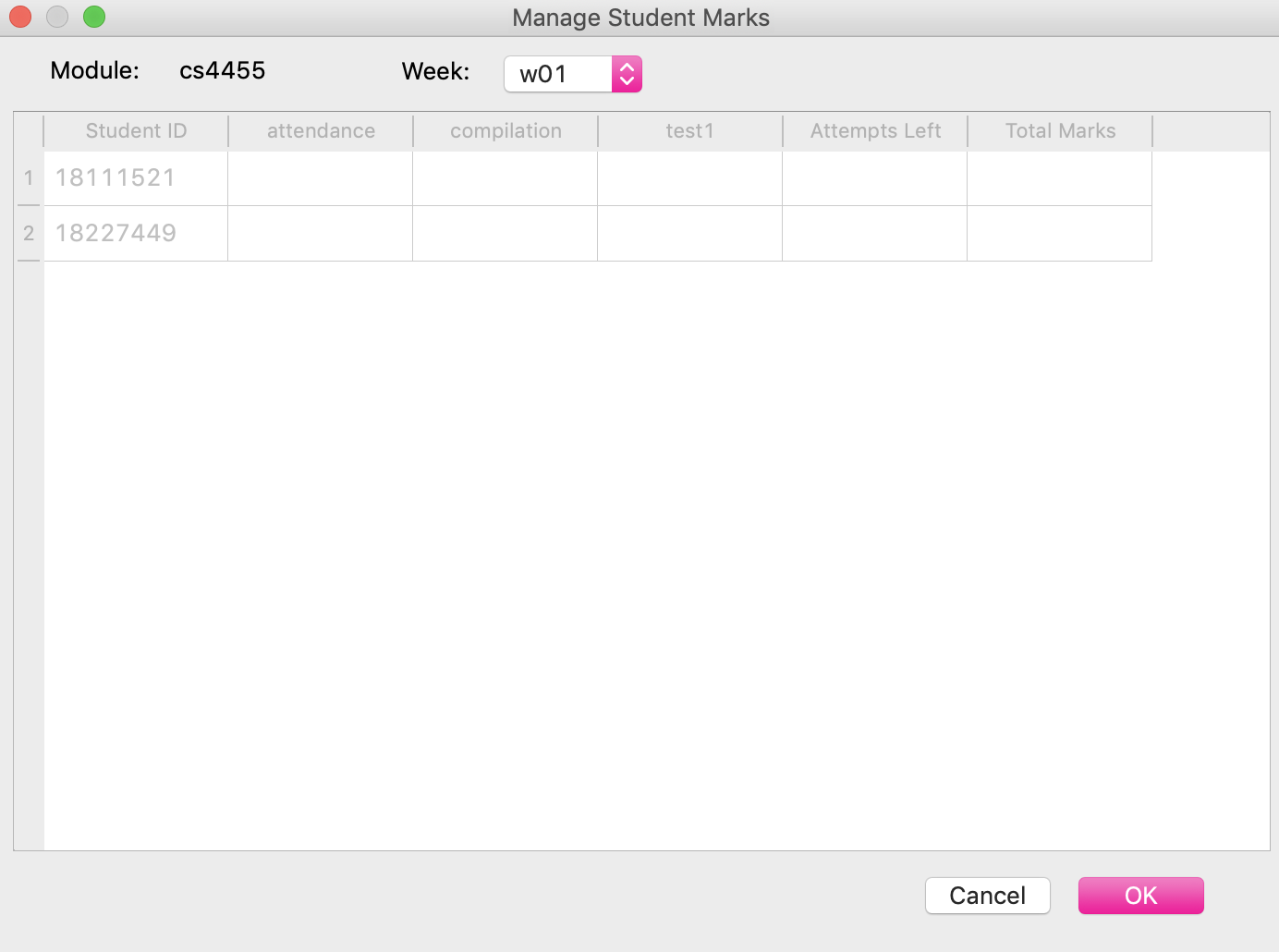
The lecturer picks an assignment using the combo box. Can clicks show to see the params file of the assignment they are about to clone and check the clone box to confirm they want to clone it.

Files interacted with:

* ui/impl/handin\_lecturer\_mian\_window.py contains the python code of the lecturer main window ui.
* ui/impl/createOneOffAssignment\_dialog.py contains the python code of the create one off assignment window ui.
* ui/impl/create\_repeat\_assignments\_dialog.py contains the python code of the create repeat assignment window.
* ui/impl/clone\_previous\_assignment\_dialog.py contains the python code of the clone assignment window.
* src/const.py contains variables and methods used across users.

Managing students marks:

A lecturer can view the marks a student got on their assignments through running h4l.py.



To check student marks the lecturer clicks the “Manage Student Marks” button.

The lecturer chooses the week of the assignment they would like to check. The marks will be presented in the window.

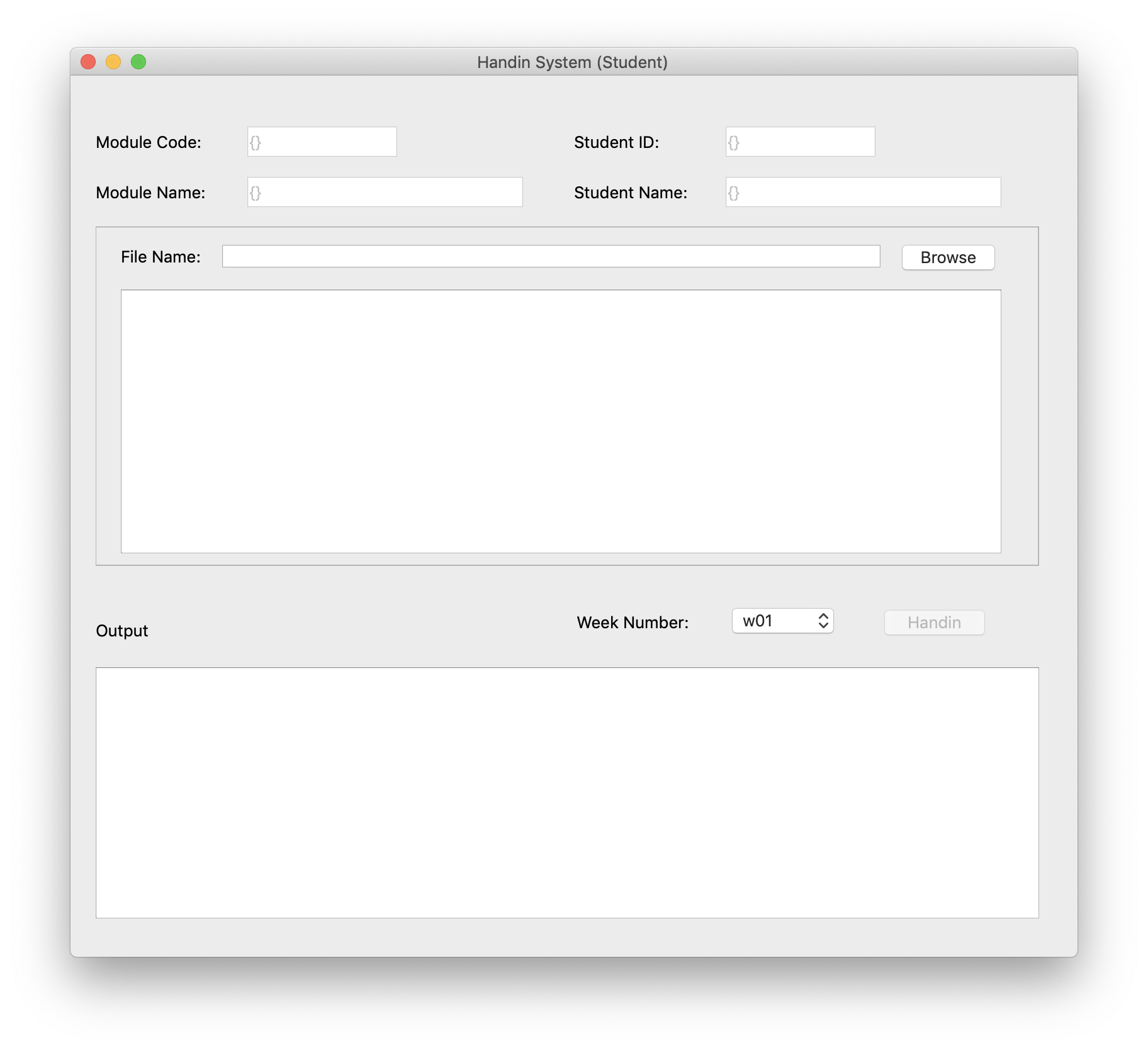
Files interacted with:

* ui/impl/manage\_student\_marks\_dialog.py contains the python code of the manage student marks window ui.
* src/const.py contains variables and methods used across users.

*Student:*

Assignment submission:

A student can submit an assignment by running h4s.py.



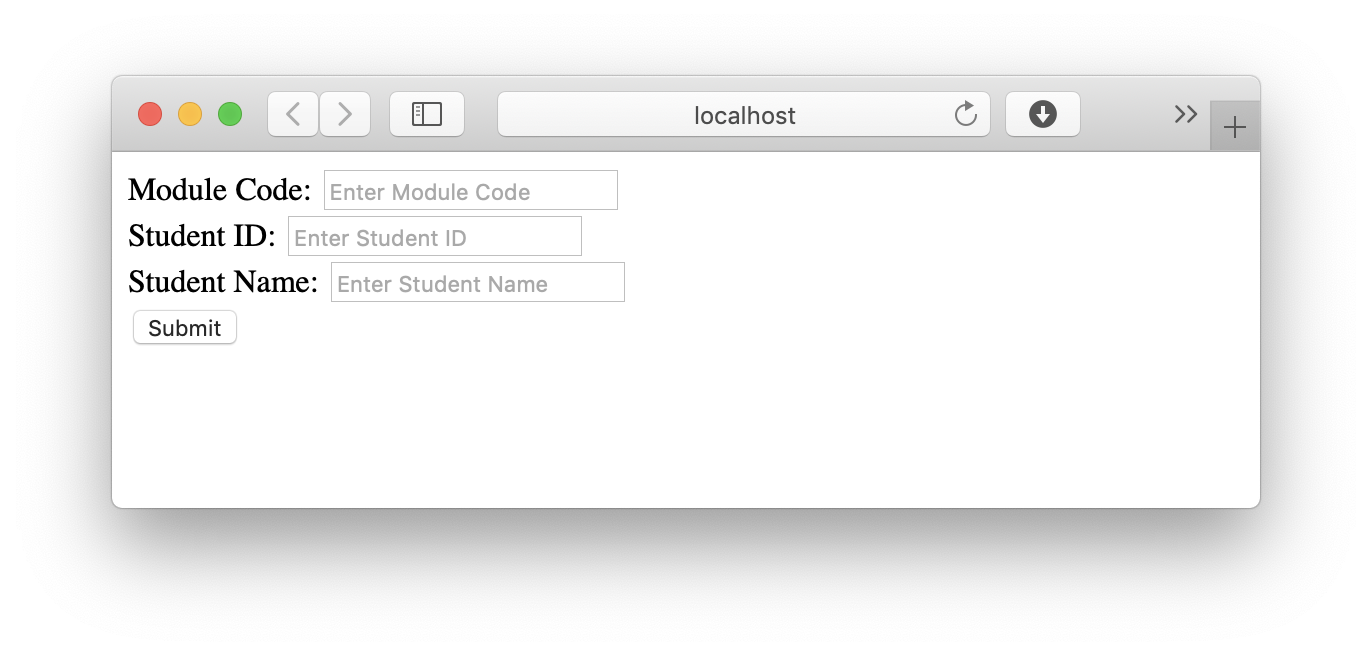
The student enters the module code.

The student's id and name are hard coded from the thin client.

The student then picks their file they’d like to submit and receive their results in the output window.

Registration:

A student registers online to receive a handin thin client.



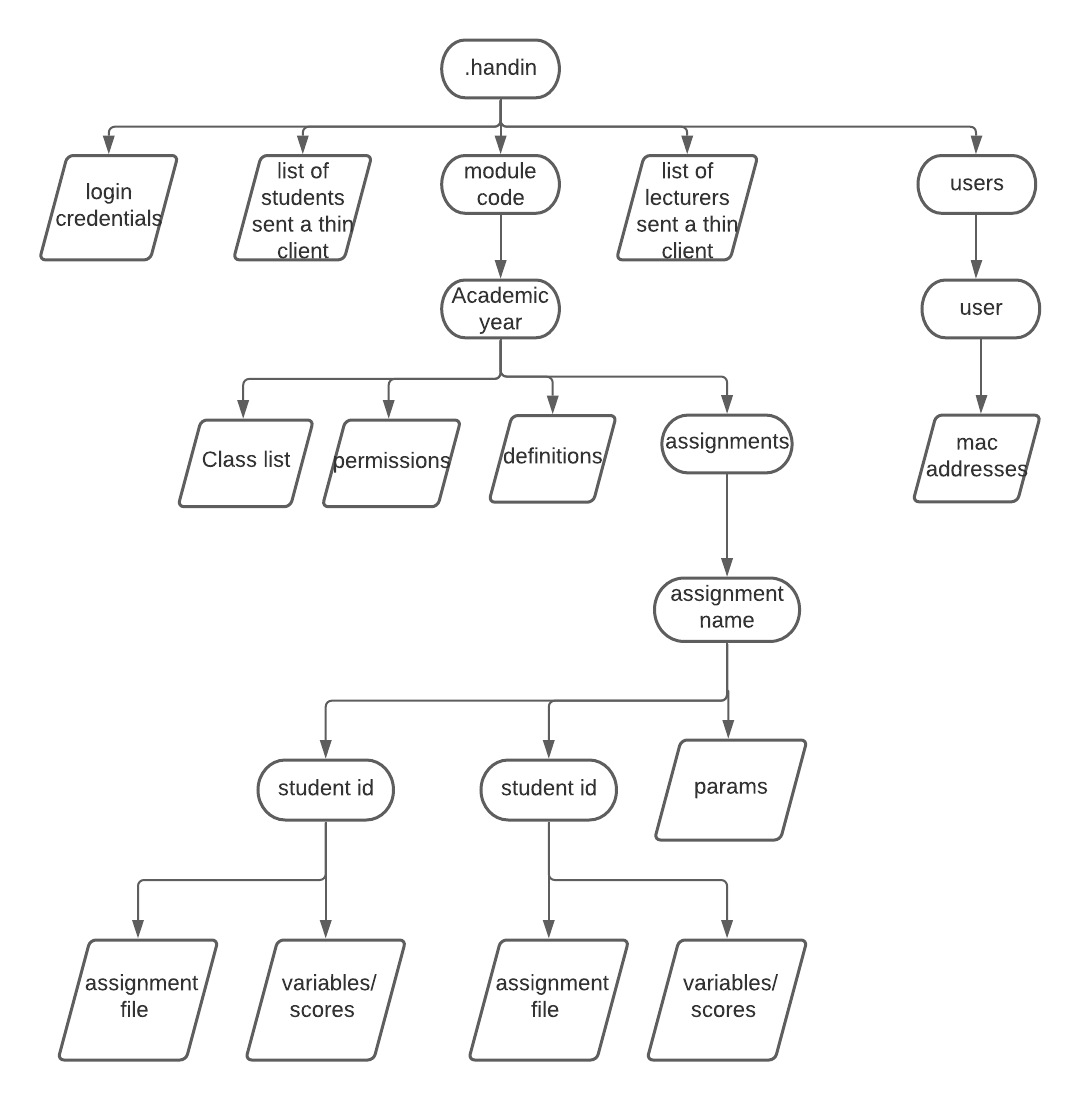
The students enter the module they wish to register for, their student id and name to register.

To add: email.

*Required Python Module:*

1. Python 3.x (Student, Lecturer)
2. PyQt5 (Student, Lecturer)
3. PyYAML (Lecturer)
4. getmac (Lecturer)
5. passlib (Lecturer)

# Data storage

Handin will store its data by the means of a subdirectory system illustrated by the following diagram.

The base directory of the sub directory system in .handin, all data will be stored in this directory.

*/.handin*

Under .handin there will be two files:

1. A file to store the emails of the students who have already been sent out a thin client.
2. A file to store the emails of the lecturers who have already been sent out a thin client.

This is so we don't send the same person multiple thin clients for multiple modules.

Each module will have its own directory under .handin, where data specific to that module will be stored.

1. A file containing login credentials (usernames and hashed passwords) of users.

*/.handin/<module\_code>*

Under each module directory, each academic year will get its own directory with data specific to that academic year stored under it.

*/.handin/<module\_code>/<academic\_year>*

Under each academic year directory there will be three files:

1. A class list to store the students currently taking the module.
2. A permissions file of who is allowed to alter the data in this module. The username the administrator set for the lecturer will be used for this.
3. A definitions file consisting of the start date, end date, due date, cutoff date, penalty per day, time and number of attempts for the assignments of the module.

There will also be a directory for each assignment of the module.

*/.handin/<module\_code>/<academic\_year>/<assignment\_name>*

Under each assignment directory there will be a params file consisting of the assignment commands information the lecturer entered when creating the assignment.

There will be a directory for each student that took the assignment.

Any files uploaded by the lecturer such as input data and output data files will be stored here.

*/.handin/<module\_code>/<academic\_year>/<assignment\_name>/<student\_id>*

Under each student directory there will be two files:

1. The assignment file they submitted with their work.
2. A variables/scores file with the scores handin gave the student for their work.

/.handin/users

Under the users directory each user will get their own directory.

/.handin/users/user

Under the user directory the a file containing the mac addresses that have accessed the users account is saved.

.

# 

# Security Implementation

To ensure handin is safe for its users, we will implement the following security measures:

Students security:

Upon registering students will receive a thin client through their email. This thin client will have hard coded into it the students id.

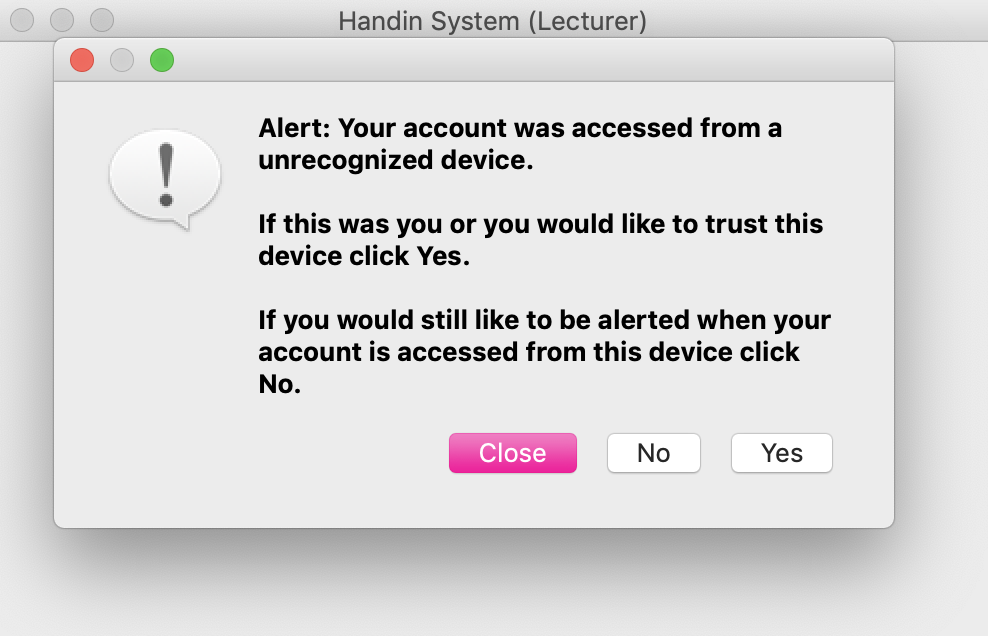
This will ensure that the student can only submit work under their own id and not someone else's. Addressing 3 security concerns:

1. If someone were to write malicious code to try and harm the handin server.
2. If someone were to submit a bad assignment under another student's id to harm their grades.
3. If someone were to help other students cheat and submit their work multiple times under different ids.

Lecturers security:

Upon registering lecturers will be emailed a thin client with their username. The first time they run this thin client, handin will make note of the lecturer's MAC address. From then on each time the thin client is used handin will compare the MAC address to the first one it had recorded. If it is different the next time the lecturer logs in from their recognized mac address they will be alerted to this and can choose to “trust” this mac address, so they won’t be alerted of it again, or not so that they are still alerted when they log in from it.

Addressing 1 security concern:

1. If the lecturer's thin client were to be compromised and used to access and alter the module information.

Module security:

Upon the registration of a module the handin admin will create a username for the lecturer of the module, and any TA’s upon the request of the lecturer.

Once the lecturer receives their thin client which they can use to log in. This will let handin know if this lecturer has access to a certain module or not by checking a permissions file in the modules directory that lists the usernames allowed to alter the module information. Addressing 2 security concerns:

1. If students were to access the module information and be able to see the assignments and their answers.
2. If students were to access their grades and alter them.

Handin server security:

Handin will make use of setuid when running students code. A user with as little permissions as possible will be created and used to test and run the code that has been submitted to handin. Addressing 1 security issue:

1. If a student were to submit malicious code to damage the data stored by handin.