# Comp4 Computing Project

# Web-based teaching portal

## Table of Contents

# ANALYSIS

## Background of problem

In the modern era, teachers and students need to have a communication between each other, whether it is giving back the homework, writing a feedback for student or just asking a question from a teacher. Most used type of communication between students and teachers on the internet is e-mail. But the problem with e-mail is that it can become quite cluttered when you manage multiple classes as a teacher.

Also e-mail is quite limited, there is no way to make up to date information pages. Current system is based on sending e-mails to a pre-defined list of students which then they can check their e-mail account for new e-mails.

Recently the school has installed a web-portal called “[redacted]” that helps with managing prep work and other useful features.

## Description of the current systems and Research

At the moment, some teachers are telling the prep on the end of the lesson, which is annoying for students, because they have to either memorise or write the given prep somewhere, but it's also annoying for teachers, because they need to write down the prep somewhere too so that it would be asked. This is both frustrating and time consuming for both students and teachers.

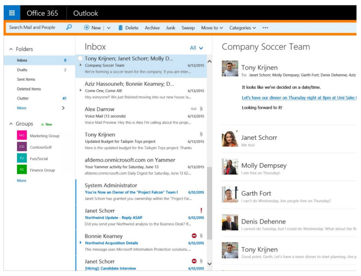


*Figure 1. Outlook Web App*

Some teachers are also using e-mails to send and receive prep. But the problem with that is it could become really messy if there are multiple classes, and it's difficult to manage those classes as well. Teachers hope that the new system could improve the management of classes and it would be less time-consuming for them and for students.

There has been a recent change to the school system management: school has installed a web-portal [redacted], which helps with sending and receiving prep work and showing useful information like timetables to the students and teachers. But the problem is that not everyone likes it, because its interface is slow and confusing, and there are some bugs.

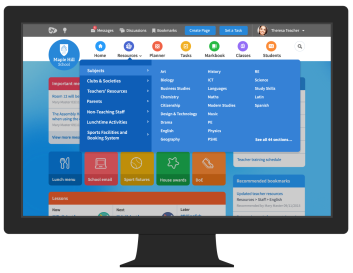


*Figure 2. [redacted]*

## Identification of prospective user

This is where my project can help **students** and **teachers**. Those groups will mostly be using my system on daily basis.

## Users of the new system

New system will be based on web pages, so anyone could use it if they have internet access. No installation or any external software is required for teachers and students, only web browser and a server where the school system is stored and running.

Everything will be stored in one server, every part of the system could be managed easily. Students, teachers and administrators will have different permissions, so, for example, students can't add assignments, only teachers can, or only administrators have access to admin panel.

School web system will need to be available at any computer or mobile.

## Interview

**Question: What about [redacted] that you don’t like? What can be improved/added?**

The temperamental nature of the system so far is a major issue. Both students and staff are anxious that it’s a system that crashes and doesn’t allow people access to their work. This needs to improve as confidence in systems is everything.

There should be improvements made regarding loading powerpoints on to academic pages as presently if there are a high number the system cannot handle it – PP are very important tools to teach students.

**Question: Do you think that using website for doing all school work is more beneficial and fun than sending emails to students?**
I think I would say that emails add a sense of security at the moment as it’s a well-used system.

If the IT dept. could guarantee 100% then I would be more than happy using the website.

- [redacted]

## Identification of user needs

- The system needs to be able to hold all information about each user, like
  - user's login details
  - user's personal details (like DoB, gender, phone number)
  - user's account creation date
  - permissions
  - what classes does he participate
  - what classes does he teaches, etc.
- The system needs to be able to store information about assignment, like
  - name
  - full text
  - what teacher has created the assignment
  - what class this assigned to
- The system needs to be able to show all information about assignment quickly

## Objectives

1. Program should have log-in system, a method to register users, have activation system, log-out system
2. User interface should show all assignments
3. User inputs must be validated to avoid erroneous or incorrect data.
4. Permissions
   4.1. Unlogged users can't access main part of system
   4.2. Students can't access some of parts of system
   4.3. Teachers can access most of parts of system except admin panel
   4.4. Administrators can access everything
5. There is teacher assigned to classes/groups and students are assigned to classes/groups
6. Teacher can send assignment to students
7. Students can upload their finished work to the system
   7.1. Students can upload text
   7.2. Students can upload files
8. Teacher can submit feedback to a student
9. Users can reset submitted work or delete assignment
10. Timetable available for a student (timeline)

11. Administrators can populate database with data (Excel file, etc.)

## Data sources and destinations

Current system – E-mail :

| What is it | Source | Destination |
|---|---|---|
| Task set for students | Teacher's input on email program | Students emails |
| Finished task from students | Student's input on email program | Teachers emails |

New system:

| What is it | Source | Destination |
|---|---|---|
| Task set for students | Teacher's input on web portal/Upload | Assignments db table/uploaded file to the server |
| Finished task from students | Student's input on web portal/Upload | Assignments db table/uploaded file to the server |
| Messages to students/teachers | Students/Teachers | Notices db table |

**Data volumes**

I will be storing hundreds of user records and also assignments and other features that are related in a database. Each day, teachers will be setting assignments to the class (usually 10-15 or more students). Teachers will either set the whole task by text in assignments description or they could upload a file, and the file's size may vary from few kilobytes to few megabytes.

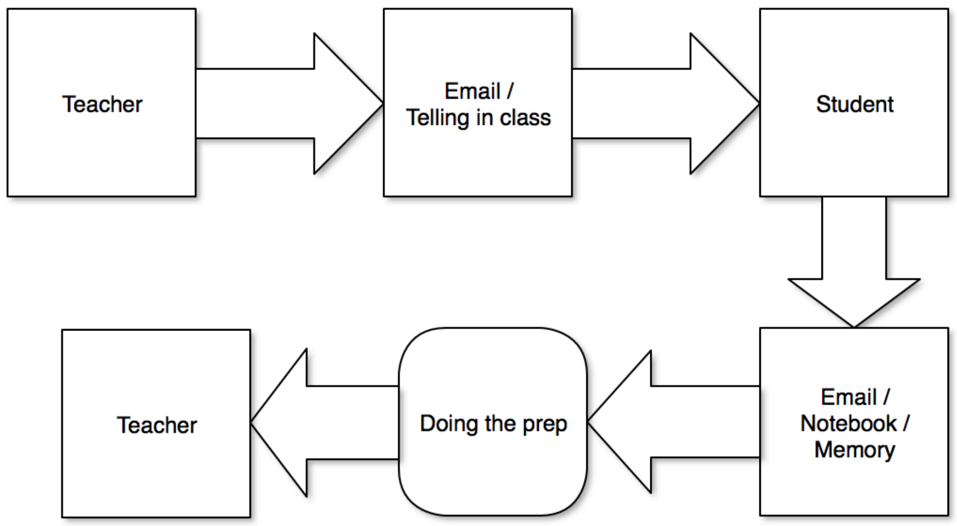
**Data Dictionary**

| Field Name | Field Purpose | Field Type | Field Size | Example Data | Validation |
|---|---|---|---|---|---|

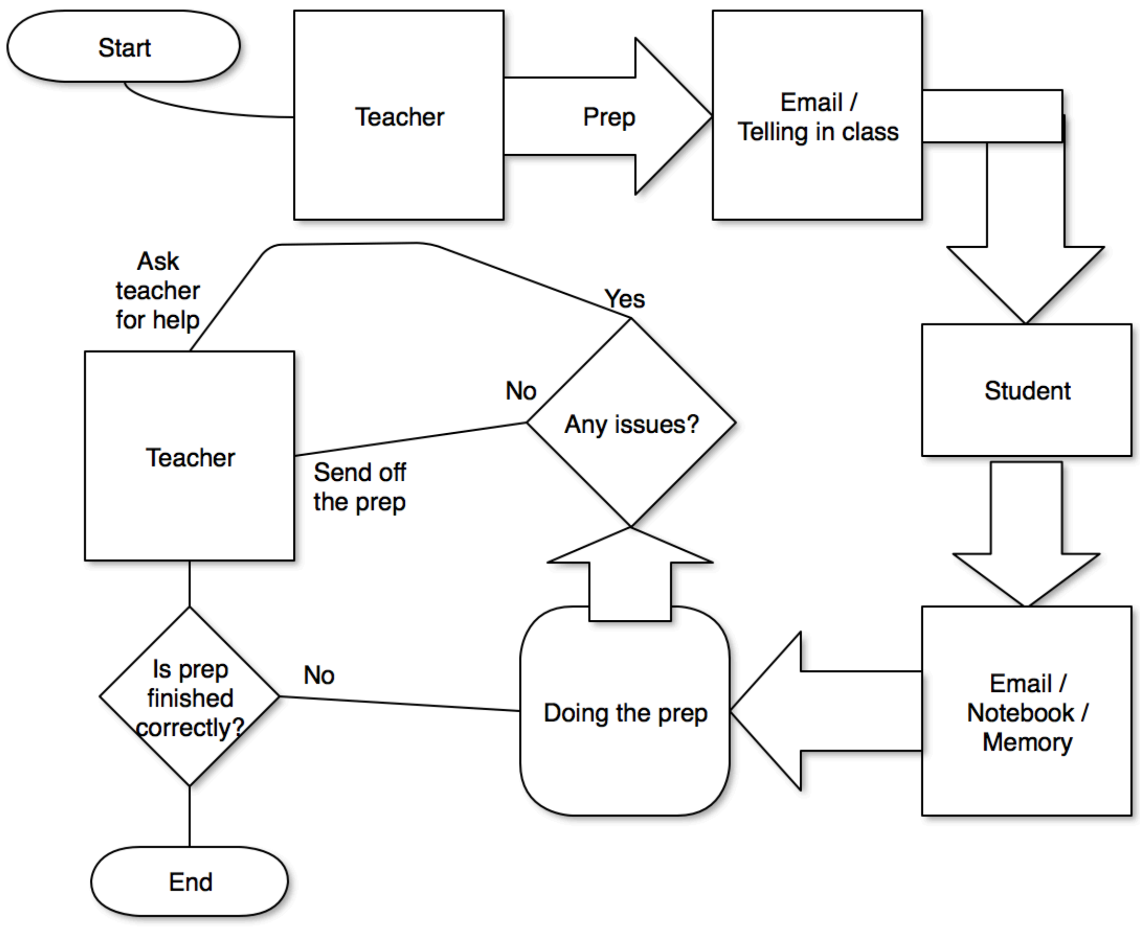
| | | | | | |
|---|---|---|---|---|---|
| Username | Stores the login information identifier | String | 30 | peter.robinson, peter@gmail.com | Not blank |
| Password | Stores the password for login identifier | String (encrypted hash) | 120 | pbkdf2:sha1:1000$TCWFmccG$98917247c942eca71b84eda2f8a42592bf178fd5 | Not blank |
| First Name | Stores first name of an user | String | 30 | Peter | Check if it's only text (no special symbols allowed) |
| Last Name | Stores last name of an user | String | 30 | Robinson | Check if it's only text (no special symbols allowed) |
| Date of Birth | Stores the date of birth of an user | DateField | 10 | 10/10/1997 | Only validate by format %d/%m/%Y |
| Activated | Stores if the user is activated or not (did user put in his personal info) | Boolean | 1 (true/false) | True | Only boolean |

Database will be based on SQLite. It is easy and lightweight choice, but it is also robust and can handle lots of queries without problems. SQLAlchemy package will be used for connections between the server and the database. It simplifies the communication and you don't have to write raw SQL.

## Data flow of the current system



1. Teacher opens up email program and sends email with description of the prep and files attached (if any) // Teacher tells the prep after lesson
2. Student gets email // Student writes down the prep
3. Student is doing prep, if there are any issues, student emails the teacher
4. Teacher gets either written prep or prep in digital form



## Potential solutions

1. Windows-based school system application

Students login to school computers and do all their work there. Application would have a GUI where students upload their work and it gets sent to teacher's computers. Authentication will be tied to school system's Windows login system (Active Directory), and all application's data will be on Windows Server.

**Advantages:**
Authentication is unified, so no need for an extra authentication system if the school is already using one
All files are stored in school network; fast access

**Disadvantages:**
Users don't have portability, they can't access system from outside the school network
System would be tied to one operative system (Windows)
No access from mobile devices

2. Standalone applications

Different applications for each major operating system (Windows, Mac OS, Linux). Cross-platform programming language and GUI will be used which will work on all platforms. Those applications will be able to connect to a server which sends back necessary data (like assignments, etc.)

**Advantages:**
Faster interface (because it would be native application)
Takes advantages of operating system (push notifications?)
Security (application can identify computers that are using the system)

**Disadvantages:**
Application needs to be downloaded before using
Application needs to be tested everytime on every platform
No access from mobile devices

3. Web-based portal

Users can open a webpage, log-in and use the system. Mobile users can also use the system because of responsive design of the webpage that adapts to the screen resolution.

**Advantages:**
Every device with web browser can access the system
Interface can be easily edited

**Disadvantages:**
Limited resources
No native possibilities

**Chosen solution**

I have chosen web-based portal because it is the easiest and accessible solution. Almost every device has a web browser built-in and it makes accessing school web portal much easier.

I have chosen to use Python as a main programming language because I have a lot of experience in it, and there are a lot of useful libraries that can help with my project.

## Design

## Overall System Design

We are going to use IPSO table to show possible inputs/outputs.

| **Inputs** | **Processes** | **Storage** | **Outputs** |
|---|---|---|---|
| User register from Excel file | First name<br>Last name<br>Username<br>Email<br>Password<br>DoB<br>Gender<br>Phone<br>Nationality | Database table: User | Registered users |
| User login | Login<br>Checking password's hash against db | Database table: User | Success/Failure message |
| Assignment submit | Text<br>Upload file | Database table: Assignment<br>Uploaded files goes to storage folder | Success/Failure message<br>Assignment text |

## Modular design

- Main menu
  - Assignments
    - Add assignment
    - Remove assignment
    - Mark assignment
    - View assignment
  - Notices
    - Add notice
  - Admin view

- Add user from form
- Add user from excel file
- Manage users
- Populate

## Data Dictionary

| Field Name | Field Purpose | Field Type | Field Size | Example Data | Validation |
|---|---|---|---|---|---|
| Username | Stores the login information identifier | String | 30 | peter.robinson, peter@gmail.com | Not blank |
| Password | Stores the password for login identifier | String (encrypted hash) | 120 | pbkdf2:sha1:1000$TCWFmccG$98917247c942eca71b84eda2f8a42592bf178fd5 | Not blank |
| First Name | Stores first name of an user | String | 30 | Peter | Check if it's only text (no special symbols allowed) |
| Last Name | Stores last name of an user | String | 30 | Robinson | Check if it's only text (no special symbols allowed) |
| Date of Birth | Stores the date of birth of an user | DateField | 10 | 10/10/1997 | Only validate by format %d/%m/%Y |
| Activated | Stores if the user is activated or not (did user put in his personal info) | Boolean | 1 (true/false) | True | Only boolean |

## Definition of record structure

Database management will be based on SQLAlchemy. It is Object Relational Mapper (ORM) that simplifies managing database, and it is used for connections between the server and the database. It simplifies the communication and there is no need to write raw SQL.

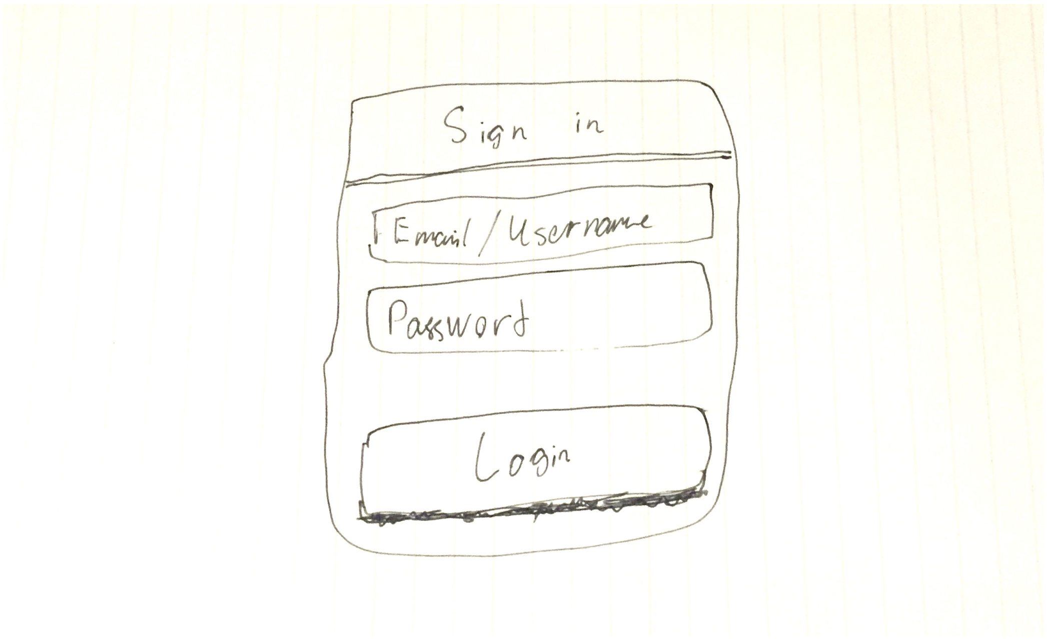
Database will be based on SQLite. It is easy and lightweight choice, but it is also robust and can handle lots of queries without problems. SQLAlchemy package will be used for connections between the server and the database. It simplifies the communication and you don't have to write raw SQL.

## Validation

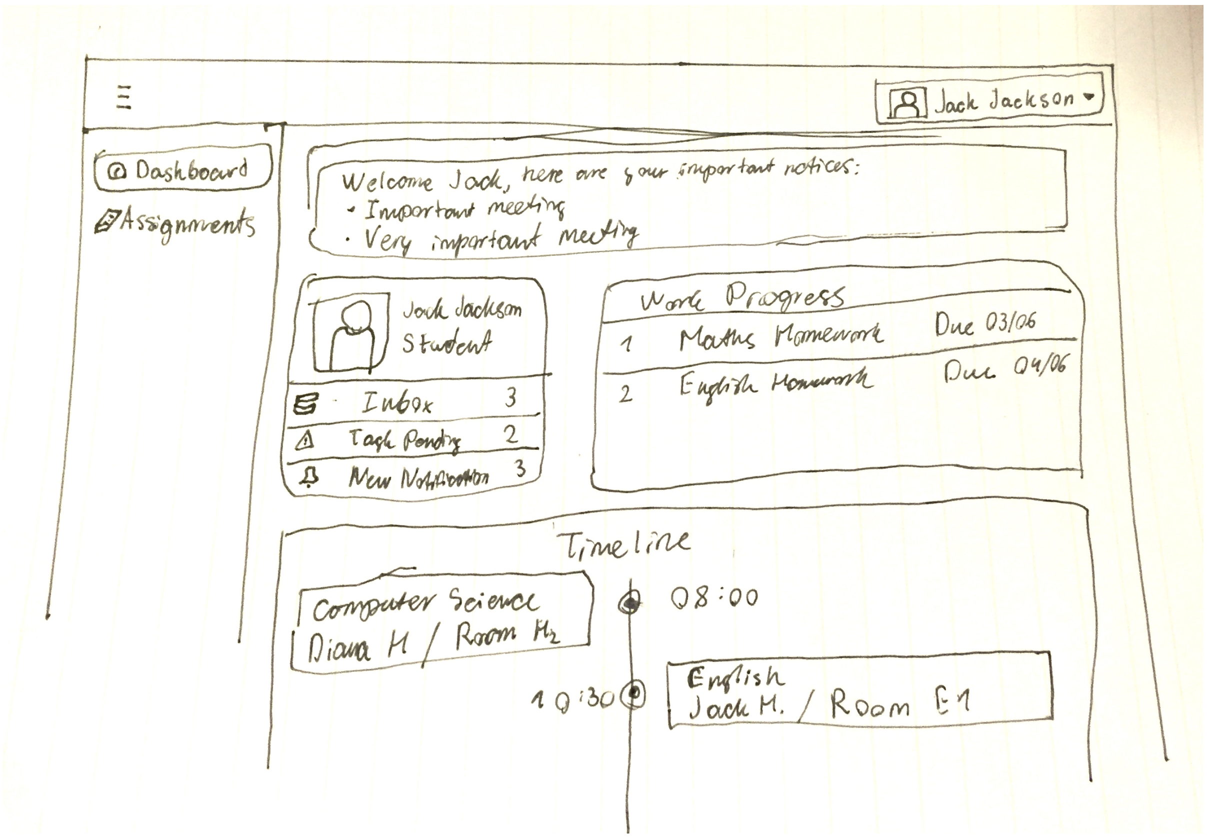
| **Field name** | **Validation checks** | **Description** | **Error message** | **Data** | **Caught** |
|---|---|---|---|---|---|
| Login – Username/Email | Presence, Length | We can't check email because it can be username | Please enter valid login. | test@test.com, test123 | Yes |
| Password | Presence, Length, no spaces, only valid set of characters | Make sure that password is in certain set of valid char | Please enter a password | Qwe, ewq | Yes |
| Date of Birth | Datatype – Date (DD-MM-YY) | Make sure it is correct date | Please insert valid date | 10-10-1997, 12-12-2004 | Yes |
| Gender | Lookup/List | Make sure the correct gender is chosen | Please insert correct gender | Male, Female | Yes |
| Subject | Llist | Make sure the correct subject is chosen | Please choose the subject | Computer Science, Biology | Yes |

## User Interface Design

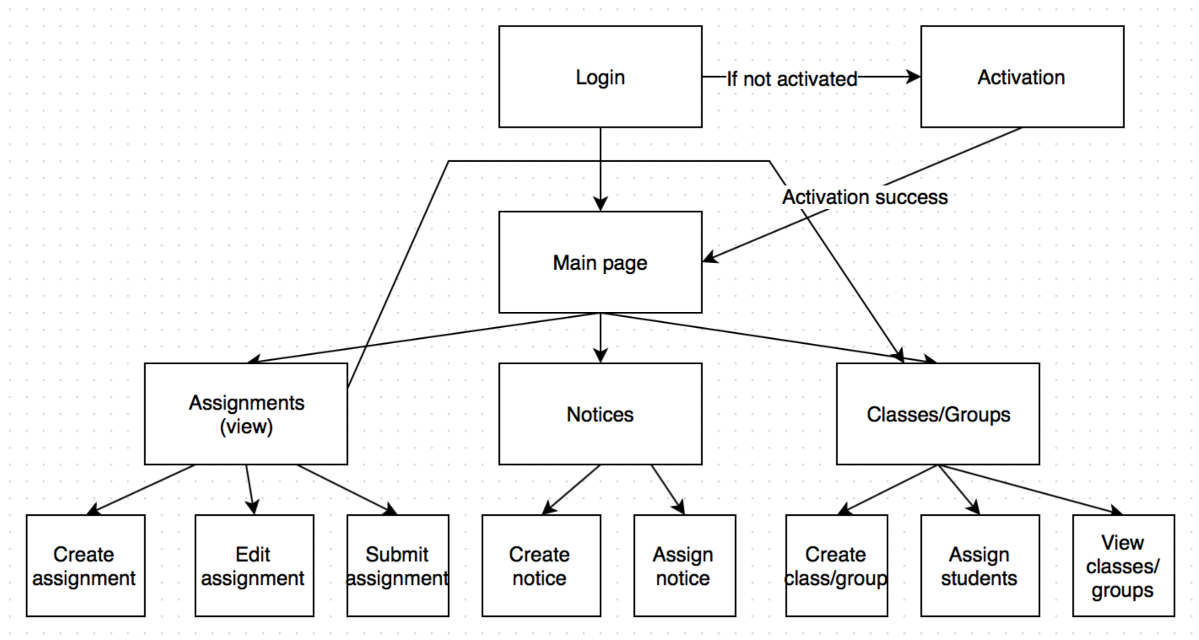
### Login page



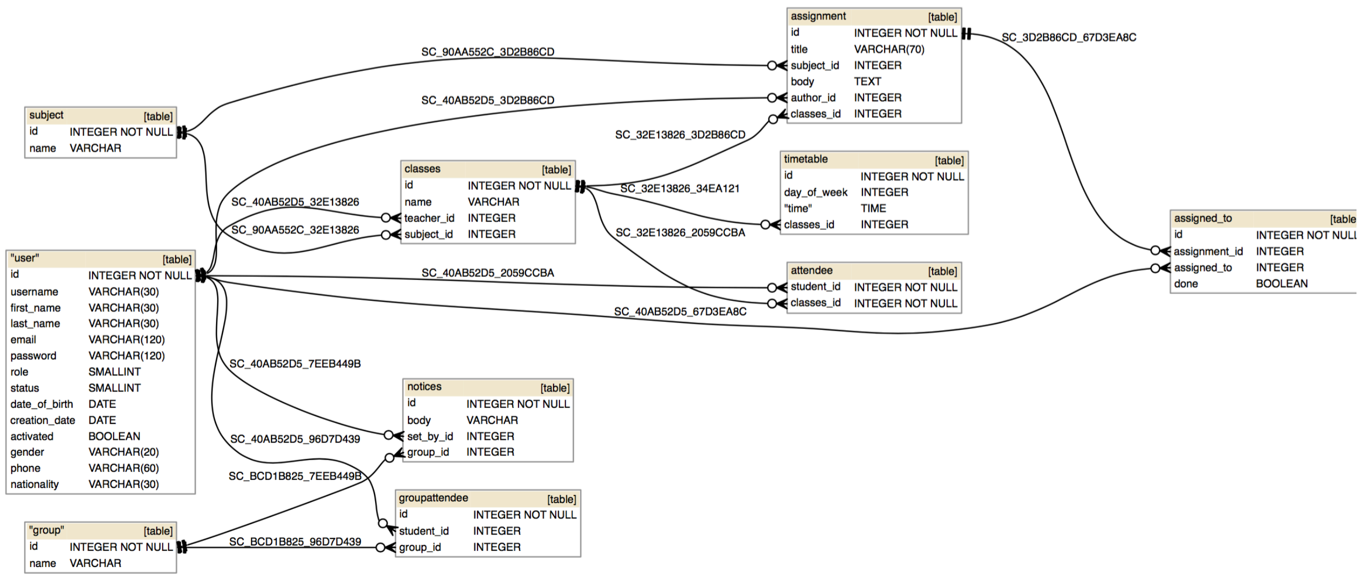
### Main dashboard



## System Flowcharts



## Entity-relationship diagram



## Storage Requirements

Software will be installed on a server and client will access the software by using web browser and entering server's address, so there is no need to install software on end user's computer.

On a server's side: the whole program will take a few megabytes of memory space, but the required size will increase as users upload more files and database records added, so a good amount of storage memory is required.

## Proposed Algorithms for complex structures

**Login form**

This code takes the username OR email and password input from an user which is received from the form and then it checks against the record in the database.

**Pseudo Code**

```
If Form.Valid():
Login = Form.GetLogin
Password = Form.GetPassword
User = DB.Query(username=Login)
If User not exists:
User = DB.Query(email=Login)
If User exists and check_password_hash(User.password, Password):
        Session['user_id'] = User.id
        ShowMessage('Welcome')
Else:
        ShowMessage('Wrong login or password')
```

**Serving uploaded files**

This code takes ID of required file and checks for database for entry of this file.

**Pseudo Code**

```
uid = Request.get(«uid»)
file = DB.AssignedTo.Query(submitted_file_id=uid).first()
```

```
if file exists:
    folder = GetAbsolutePath() + «/uploads/» # our uploads file
    return send_file_from_directory(folder, uid, as_attachment=True)
else:
    abort(404) # sends 404 error
```

**Add assignment**

This page sends possible subject list and student list to the user and processes data.

**Pseudo Code**

```
Form = AddAssignment()
Subject_choices = Subject.query.all().map(subject.id, String(subject))
Form.subject.choices = Subject_choices

Student_choices = User.query.all().map(student.id, String(student))
Form.student.choices = Student_choices

If form.IsValid():
    Assignment = Assignment(FormData)
    DB.Session.Add(Assignment)
    DB.Session.Commit()

    For students in form.students.data:
        Assigned_to = AssignedTo(assignment_id=assignment.id,
assigned_to=student)
        DB.Session.Add(Assigned_to)
        DB.Session.Commit()
```

## Security and Integrity of Data

There are a lot of sensitive data that will be stored in a database, so there should be some protection in case where system gets hacked.

All passwords for user in database are stored in encrypted form, using PBKDF2 (Password-Based Key Derivation Function 2) and the encryption algorithm is SHA1. Hashes are generated using «generate_password_hash» function from Werkzeug library. To check if the hash is valid for inputted password, «check_password_hash» function is used from the same library.

For uploading files, to prevent hackers to access data from different directories (using XSS), «secure_filename» function is used from Werkzeug library to sanitize the uploaded filename.

Also to prevent XSS attacks from users when submitting data (for example, assignment text), all inputted data is going through «escape» function which replaces special characters like “&”, “<”, “>” and (”) to HTML-safe sequences.

The main database for whole project is stored in «app.db» file.

## System Security

System Security is also an important part of my project. User needs a login and password in order to access most parts of the system. Without login and password, user gets redirected to login form and asked to enter their credentials.

## Modules that will be designed

Populate – this is where sample data gets added to the database and where database can be recreated
Config – this is where I put configuration settings for my application. I have borrowed _basedir function, upload folder variables and SQLAlchemy connection settings from the sample application from Internet
__init__ - starting point of the program. I took function for generating secret key for this
views – all views (URL endpoints) are located here.
Forms – all forms are located here
Modules – all structure for a database is located here
Decorators – all decorators that will help me create permissions are located here. I took a snippet of code for a decorator from official Flask website

## Software used

Backend:

- Python 2/3
- Flask (web-framework)
- Flask-Login (simplifies login management)
- Flask-Upload (for managing uploads)

- Flask-Admin (admin panel)
- SQLAlchemy (used to connect to a database)
- Nginx (serving static content like images, scripts etc) - optional
- UWSGI (used to connect Flask with Nginx) - optional
- Supervisor (for keeping server online) - optional

Frontend:

- Bootstrap 3
- Flatlab CSS
- Javascript
- Jquery
- CKEditor
- Jquery plugins

## System Testing

| Test No. | Purpose of Test | Test Data | Expected Outcome | Actual Outcome | Comments/ Actions | Screenshot Ref. |
|---|---|---|---|---|---|---|
| **1** | Testing Login form | Login: user1@user.com Password: 1 *Normal* | Logs in (test account), goes to activate page | As expected | Reference to Objective 1 | **Screenshot 1, 1.2** |
| **2** | Testing Login Form | Login: user1@user.com Password: *None* *Extreme* | Gives error message | As expected | Reference to Objective 1 | **Screenshot 2** |
| **3** | Testing Login Form | Login: user1@user.com Password: 123456 (wrong password) *Extreme* | Gives error message about password being wrong | As expected | Reference to Objective 1 | **Screenshot 3** |
| **4** | Testing Activate Form | E-mail: not-an-email Password: *None* *Extreme* | Gives error message about invalid error address | As expected | Reference to Objective 1 | **Screenshot 4** |

| | | | | | | |
|---|---|---|---|---|---|---|
| **5** | Testing Activate Form | E-mail: test@example.com Password: 123 *Normal* | Activates account and redirects to home page | As expected | Reference to Objective 1 | **Screenshot 5** |
| **6** | Testing Activate Form – trying to login by email which already exists in database | E-mail: 123@123.com Password: 123 *Extreme* | Gives error about email already been used | As expected | Reference to Objective 1 | **Screenshot 6** |
| **7** | Testing Activate Form – succesful log-in | E-mail: 12345@12345.com Password: 12345 *Normal* | Shows the message about successfull activation and redirects to main page<br>- Shows «Student» title | As expected | Reference to Objective 1 | Screenshot 7 |
| **8** | Testing Student permissions – going to admin panel | URL: http://127.0.0.1:5000/admin/ *Extreme* | Won't allow administrative panels | As expected | Reference to Objective 4.2 | **Screenshot 8** |

| | | | | | | |
|---|---|---|---|---|---|---|
| **9** | Testing interface - logout | URL: http://127.0.0.1:5000/users/me/ Clicking on profile button at the top right of interface<br>Clicking on «Logout» button<br>*Normal* | Shows «Logout» button<br>Log outs successfully and shows message about logout | As expected | Reference to Objective 1 | Screenshot 9, 9.1 |
| **10** | Testing access without login | URL: http://127.0.0.1:5000/users/me/<br>*Extreme* | Redirects to login page and shows message about signing in | As expected | Reference to Objective 4.1 | Screenshot 10 |
| **11** | Testing admin access | URL: http://127.0.0.1:5000/users/login/<br>Email/Username: admin<br>Password: admin<br>*Normal* | Redirects to main page and shows «Admin» status and few additional options on the menu | As expected | Reference to Objective 4.4 | Screenshot 11 |
| **12** | Testing Excel file upload | URL: http://127.0.0.1:5000/users/upload_excel/<br>*Normal* | Uploads a file to a server and it parses and creates records for users given | Error: FileNotFoundError: [Errno 2] No such file or directory: '/uploads/excel.xlsx'<br>**Fixed** by changing the value of UPLOAD_FOLDER from «/uploads/» to «uploads». os.path.join function was working incorrectly when two | Reference to Objective 11 | Screenshot 12 |

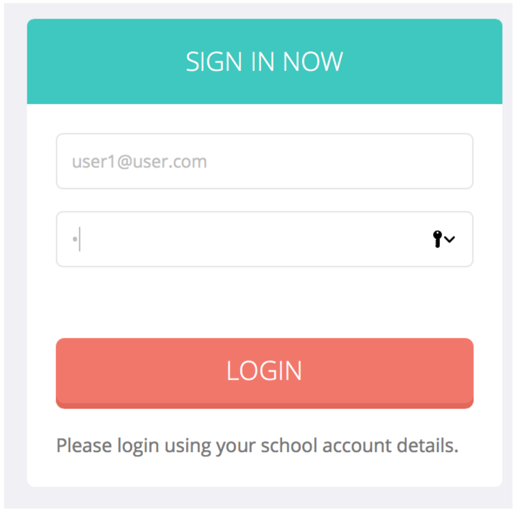
| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | slashes were on the sides of variable. | | |
| **13** | Testing Admin Panel access | Main page<br>Clicking on «Admin Panel» on sidebar<br>*Normal* | Redirects to admin panel | As expected | Reference to Objective 4.4 | Screenshot 13 |
| **14** | Adding timetable entry | Going to «Timetable» page<br>Adding entry:<br>User: Admin Admin<br>Classes: Computer Science \| Ivan Arnold<br>Day of Week: Monday<br>Time: 08:00<br>*Normal* | Adds entry | As expected | Reference to Objective 11 | Screenshot 14 |
| **15** | Testing timetable | Going back to main page<br>*Normal* | Shows the timetable entry on the timeline | Error:<br>jinja2.exceptions.UndefinedError<br>jinja2.exceptions.UndefinedError: 'datetime.time object' has no attribute 'time'<br>*<span class="timeline-date">{{ item.time.time() }}</span>*<br>**Fixed** by changing «item.time.time()» to «item.time». This is because «time» item is already passed, no need to convert it again by using .time(). | Reference to Objective 10 | Screenshot 15 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **16** | Adding timetable entry | Going to «Timetable» page on admin panel All fields are empty *Extreme* | Shows error about required fields | Record successfully added **Fixed** by making fields not nullable. | Reference to Objective 11 | Screenshot 16 |
| **17** | Adding few more timetable entries and showing timeline | On «Timetable» page on admin panel, adding few more timetable entries and testing timeline | Shows all added timetable entries in sorted way | As expected | Reference to Objective 10, 11 I set up so that all timetable entries will show, no matter which day of week. It will make testing much easier to do. | Screenshot 17 |
| **18** | Removing assignment | On «Assignments» page, clicking on Trash button on «Maths Homework» assignment | Shows modal about confirmation of deleting | As expected | Reference to Objective 9 | Screenshot 18 |
| **19** | Removing assignment | Clicking on «Delete» button | Shows message about deletion and deletes assignment «Maths Homework» | As expected | Reference to Objective 9 | Screenshot 19 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **20** | Adding new assignment | Clicking on “Add Assignment” button<br>Title: Biology Prep<br>Body: Please do task 1 on page 2.<br>Checkbox checked on “Biology”<br>Students: All selected<br>Deadline: 23-04-16 00:00<br>Text Required: Yes<br>File Required: Yes<br>Clicking on “Submit” | Shows message «Assignment successfully added»<br>Shows new assignment on assignments page | As expected | Reference to Objective 6 | Screenshot 20 |
| **21** | Checking assignment entry | Clicking on new «Biology Prep» entry | Shows two panels with all information at the left and submission of work at the right | As expected | Reference to Objective 6 | Screenshot 21 |
| **22** | Checking assignment entry as another user | Login in as another user: 123@123.com; 123 (Riley Baker) | Shows assignment «Biology Prep» at the main page | As expected | Reference to Objective 7 | Screenshot 22 |
| **23** | Submitting assignment | Clicking on assignment and submitting text and file | Shows message «Your assignment has been submitted.» and | As expected | Reference to Objective 7.1, 7.2 | Screenshot 23, 23.1 |

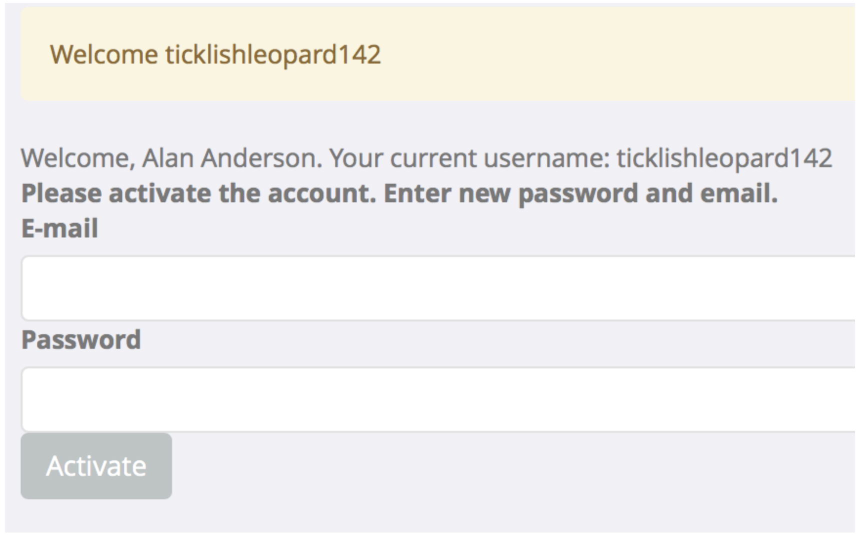
| | | | | | | |
|---|---|---|---|---|---|---|
| | | | panel changes to button with resetting the assignment | | | |
| **24** | Teachers view | Login in as another user: admin@admin.com, admin (Teacher) Clicking on «Biology Prep» assignment, clicking on «Teachers view», scrolling down to «Riley Baker» submission | Shows submitted text and file attached with button «Give Feedback» | As expected | Reference to Objective 8 | Screenshot 24 |
| **25** | Downloading attached file | Clicking on «766a20bc-ccf2-4759-ad70-7673589f6e00.gif» | Downloads file | Error: 404 Not Found<br>**Fixed** by removing «os.getcwd()» from «folder = os.getcwd() + app.config['UPLOAD_FOLDER']». | Reference to Objective 7.2 | Screenshot 25 |
| **26** | Giving feedback | Clicking on «Give Feedback» button, entering feedback text “Good Work!” and clicking on «Submit» | Shows message «Feedback successfully given.»<br>On «Riley Baker» assignment it shows that assignment has | As expected | Reference to Objective 8 | Screenshot 26 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | already been given | | | |
| **27** | Looking at feedback from another user | Login in as another user: 123@123.com; 123 (Riley Baker) Going to «Biology Prep» assignment. | Shows feedback from assignment from «Admin Admin» that says «Good work!» | As expected | Reference to Objective 8 | Screenshot 27 |
| **28** | Resetting assignment | Clicking on “Reset your submitted assignment” button | Shows text and file upload panels instead of button | As expected | Reference to Objective 9 | Screenshot 28 |
| **29** | Creating a class | Login as administrator, going to admin panel, clicking on «Group» tab, entering data: Name: CHEM1 | Shows «Record was successfully saved» and shows newly added record on the list | As expected | Reference of Objective 5 | Screenshot 29 |
| **30** | Adding a notice to a group | Going to main menu, clicking on «Add Notice», | | | | |



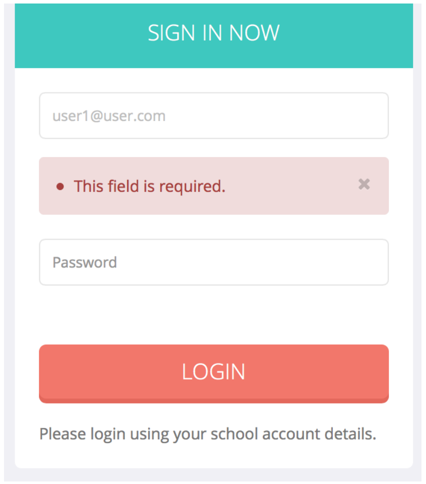
Please login using your school account details.

*Screenshot 2*


Welcome ticklishleopard142

Welcome, Alan Anderson. Your current username: ticklishleopard142
**Please activate the account. Enter new password and email.**
**E-mail**
**Password**

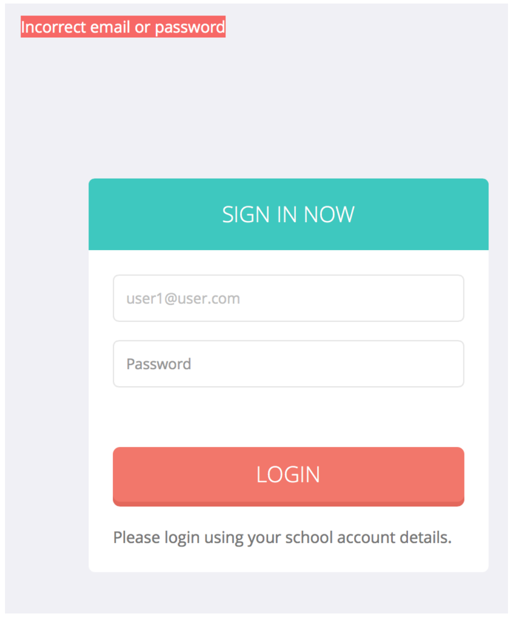
*Screenshot 1.2*



● This field is required.

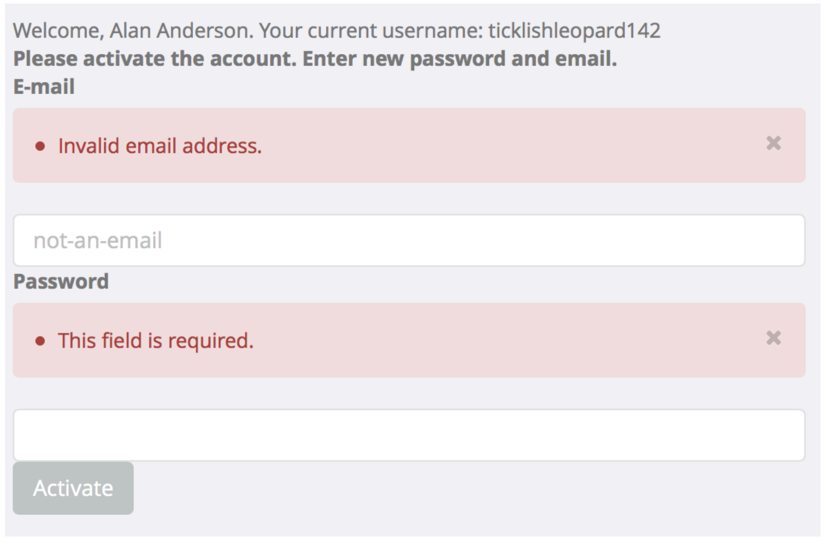
Please login using your school account details.

*Screenshot 2*


Incorrect email or password

Please login using your school account details.

*Screenshot 3*



Welcome, Alan Anderson. Your current username: ticklishleopard142

**Please activate the account. Enter new password and email.**

**E-mail**

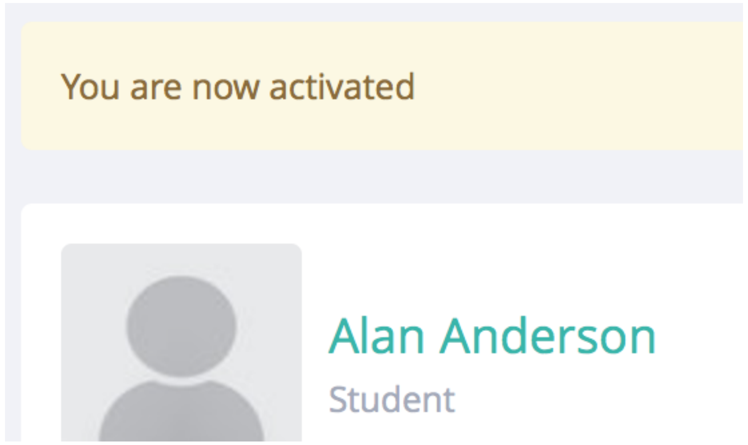
- Invalid email address.

not-an-email

**Password**

- This field is required.

*Screenshot 4*

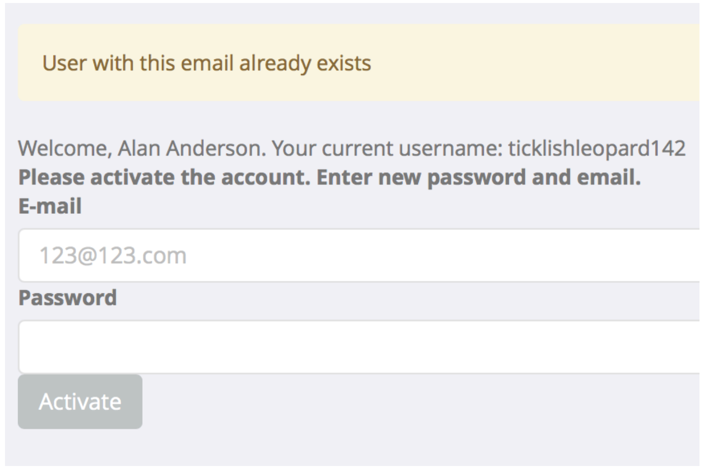


You are now activated

Alan Anderson

Student

*Screenshot 5*



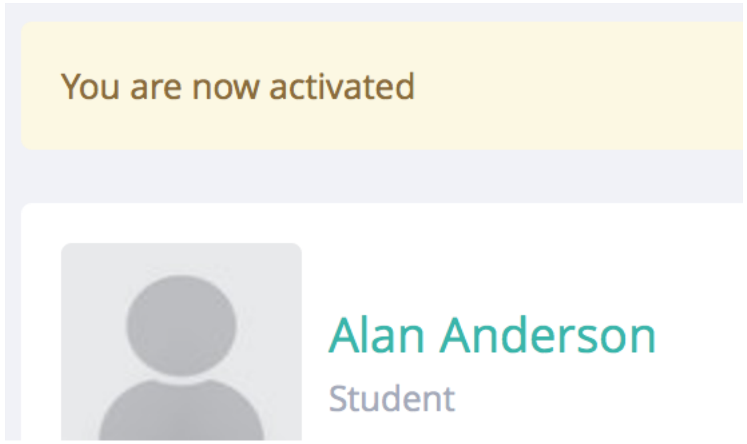
User with this email already exists

Welcome, Alan Anderson. Your current username: ticklishleopard142
**Please activate the account. Enter new password and email.**
**E-mail**

123@123.com

**Password**

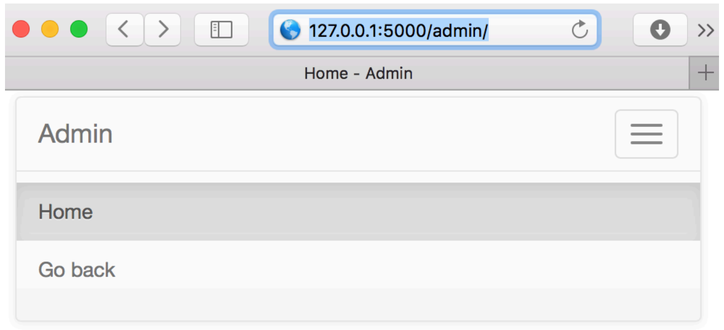
*Screenshot 6*



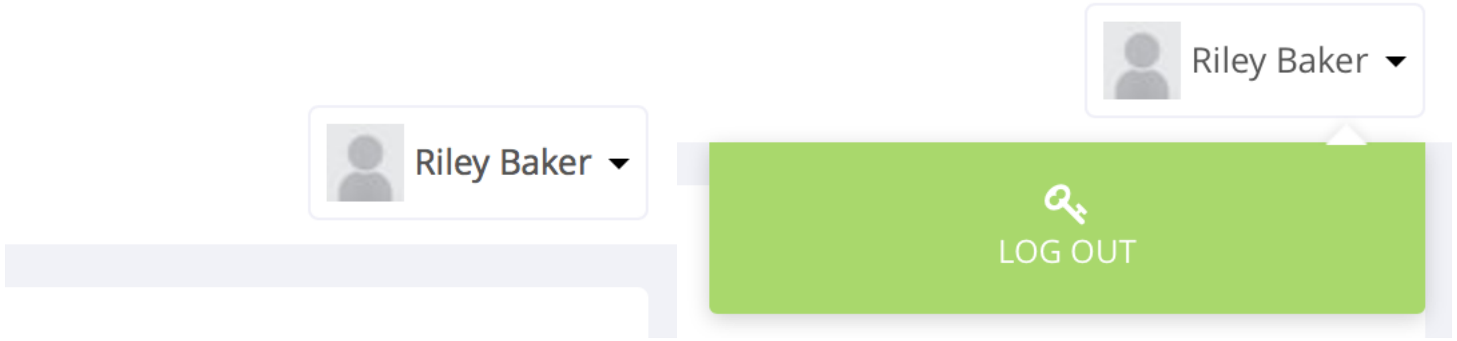
You are now activated

Alan Anderson
Student

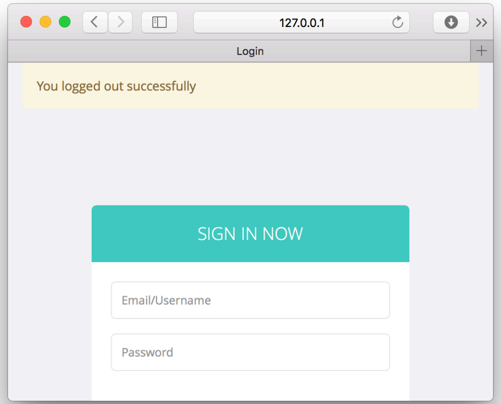
*Screenshot 7*



*Screenshot 8*

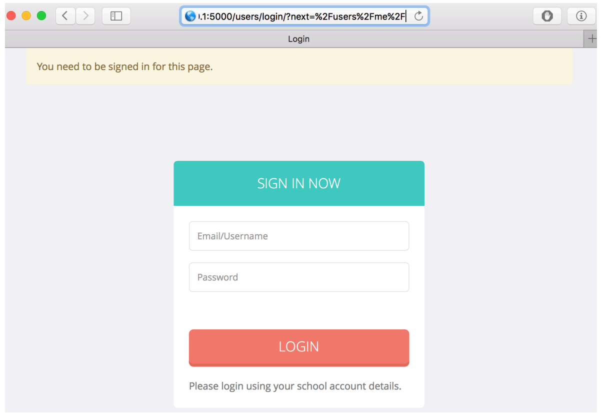


*Screenshot 9*



You logged out successfully

*Screenshot 9.1*

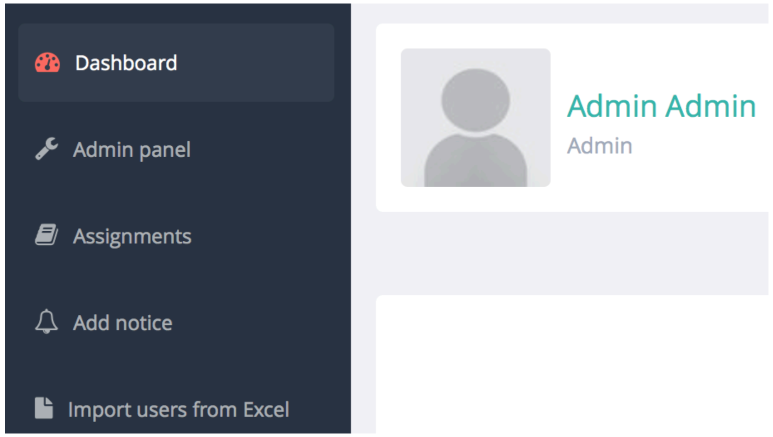


:1:5000/users/login/?next=%2Fusers%2Fme%2F

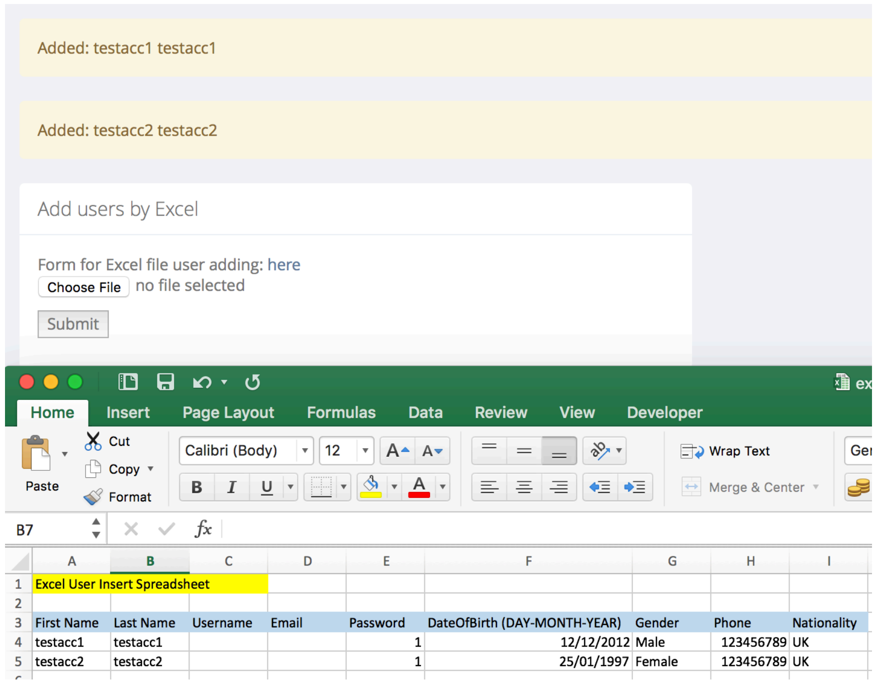
You need to be signed in for this page.

Please login using your school account details.

*Screenshot 10*



*Screenshot 11*



Added: testacc1 testacc1

Added: testacc2 testacc2

Add users by Excel

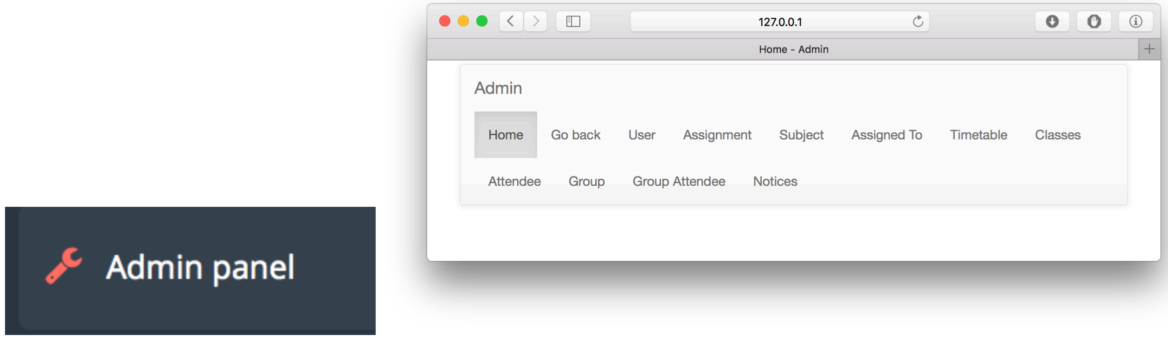
Form for Excel file user adding: here

Choose File no file selected

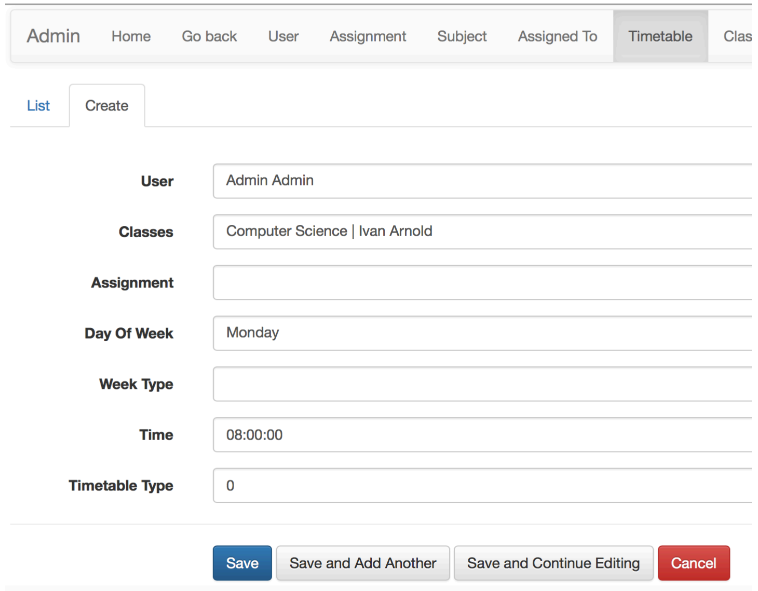
Submit

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Excel User Insert Spreadsheet | | | | | | | | |
| 2 | | | | | | | | | |
| 3 | First Name | Last Name | Username | Email | Password | DateOfBirth (DAY-MONTH-YEAR) | Gender | Phone | Nationality |
| 4 | testacc1 | testacc1 | | | 1 | 12/12/2012 | Male | 123456789 | UK |
| 5 | testacc2 | testacc2 | | | 1 | 25/01/1997 | Female | 123456789 | UK |

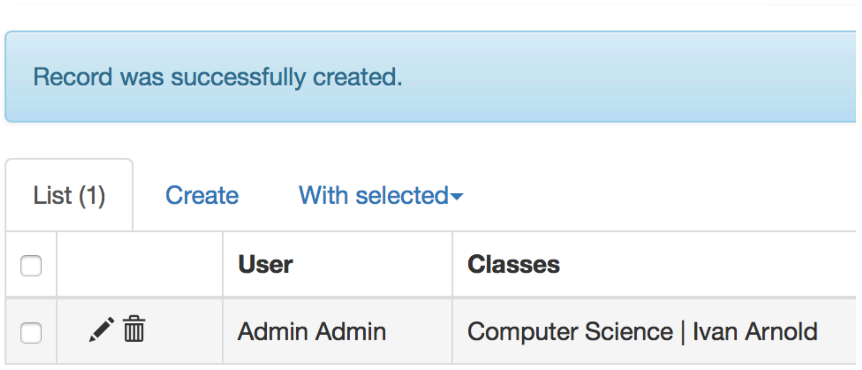
*Screenshot 12*



*Screenshot 13*

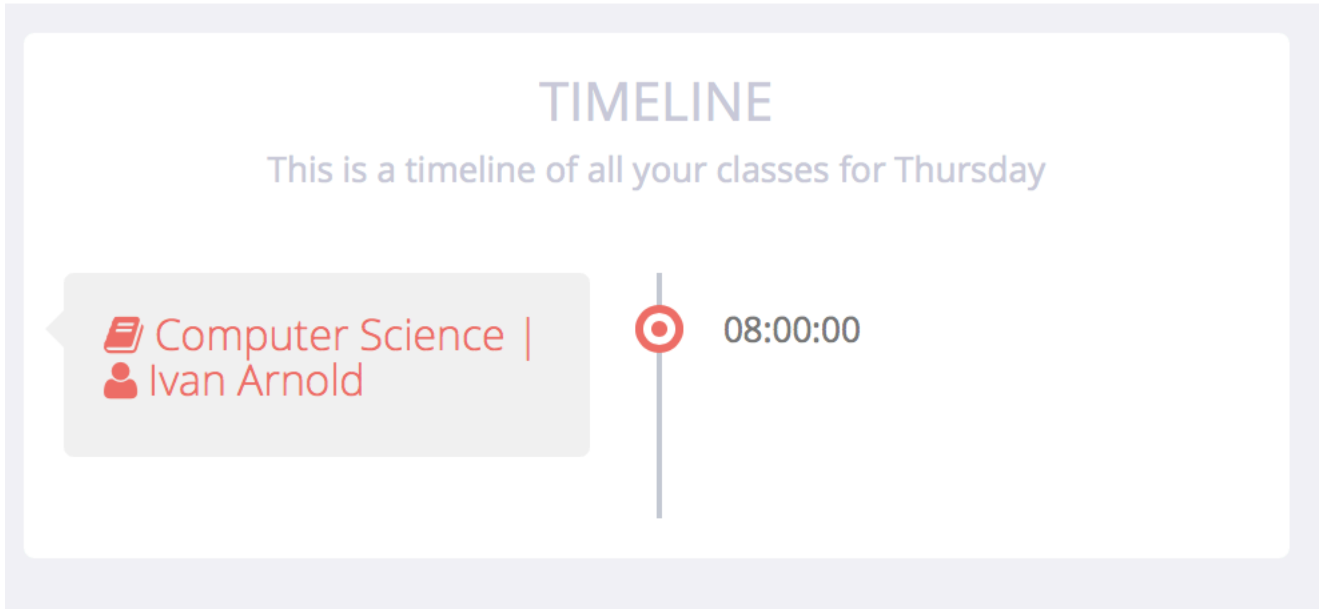


| | |
|---|---|
| User | Admin Admin |
| Classes | Computer Science \| Ivan Arnold |
| Assignment | |
| Day Of Week | Monday |
| Week Type | |
| Time | 08:00:00 |
| Timetable Type | 0 |


Record was successfully created.

| | | User | Classes |
|---|---|---|---|
| ☐ | | Admin Admin | Computer Science \| Ivan Arnold |

*Screenshot 14*



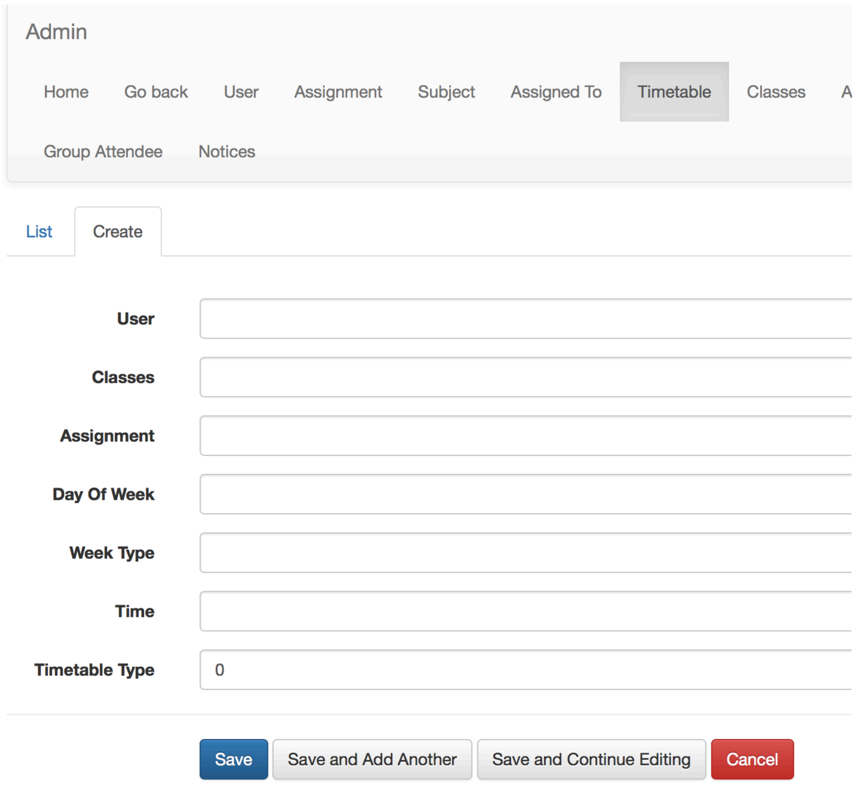
TIMELINE

This is a timeline of all your classes for Thursday

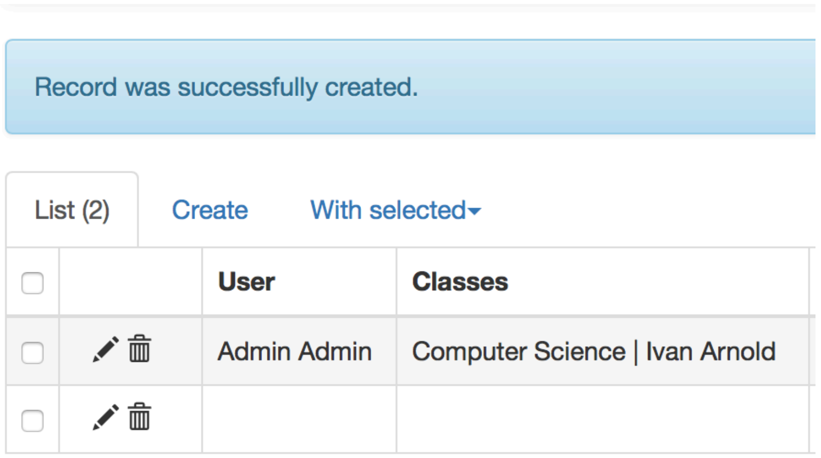
Computer Science | Ivan Arnold

08:00:00

*Screenshot 15*

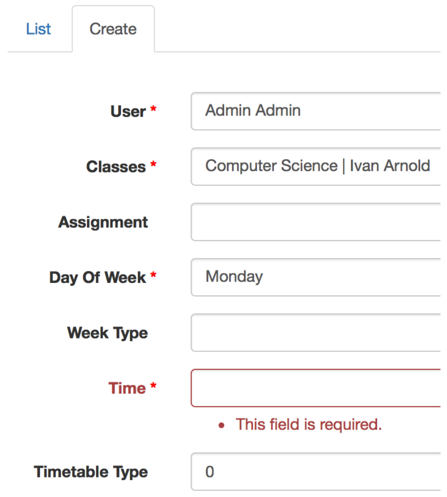


|  |  |
|---|---|
| User |  |
| Classes |  |
| Assignment |  |
| Day Of Week |  |
| Week Type |  |
| Time |  |
| Timetable Type | 0 |


Record was successfully created.

|  |  | User | Classes |
|---|---|---|---|
| ☐ | ✎ 🗑 | Admin Admin | Computer Science \| Ivan Arnold |
| ☐ | ✎ 🗑 |  |  |

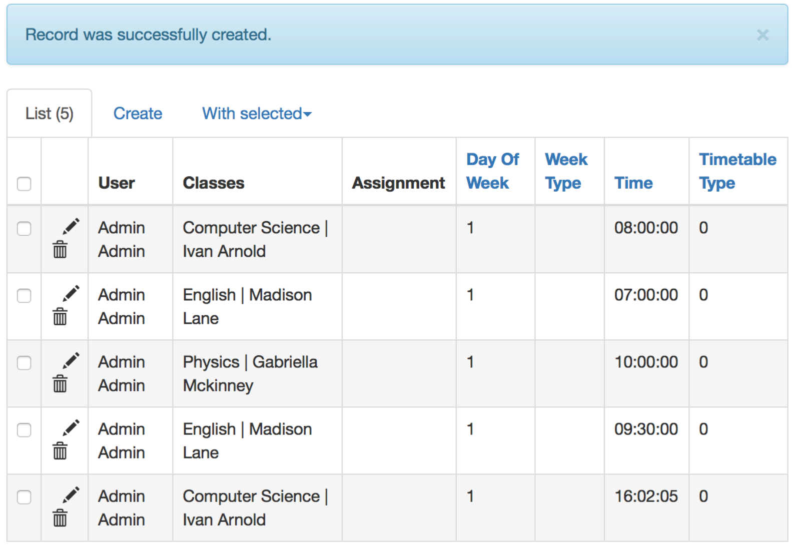
*Screenshot 16*



List | Create

| | |
|---|---|
| User * | Admin Admin |
| Classes * | Computer Science \| Ivan Arnold |
| Assignment | |
| Day Of Week * | Monday |
| Week Type | |
| Time * | |
| | • This field is required. |
| Timetable Type | 0 |

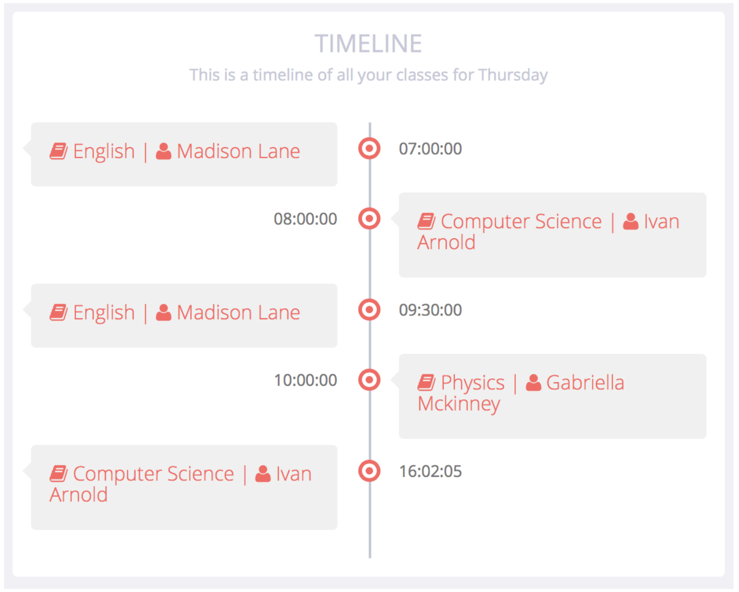
*Screenshot 16.1 (fixed)*



Record was successfully created.

List (5) | Create | With selected

| | | User | Classes | Assignment | Day Of Week | Week Type | Time | Timetable Type |
|---|---|---|---|---|---|---|---|---|
| | | Admin Admin | Computer Science \| Ivan Arnold | | 1 | | 08:00:00 | 0 |
| | | Admin Admin | English \| Madison Lane | | 1 | | 07:00:00 | 0 |
| | | Admin Admin | Physics \| Gabriella Mckinney | | 1 | | 10:00:00 | 0 |
| | | Admin Admin | English \| Madison Lane | | 1 | | 09:30:00 | 0 |
| | | Admin Admin | Computer Science \| Ivan Arnold | | 1 | | 16:02:05 | 0 |



TIMELINE

This is a timeline of all your classes for Thursday

07:00:00 English | Madison Lane

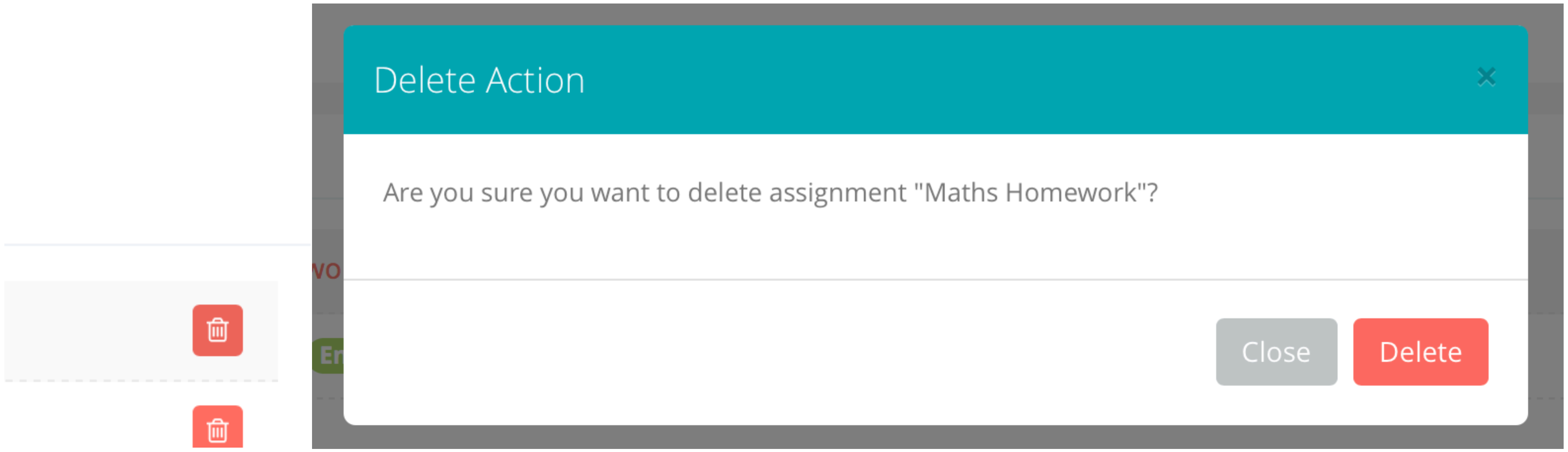
08:00:00 Computer Science | Ivan Arnold

09:30:00 English | Madison Lane

10:00:00 Physics | Gabriella Mckinney

16:02:05 Computer Science | Ivan Arnold

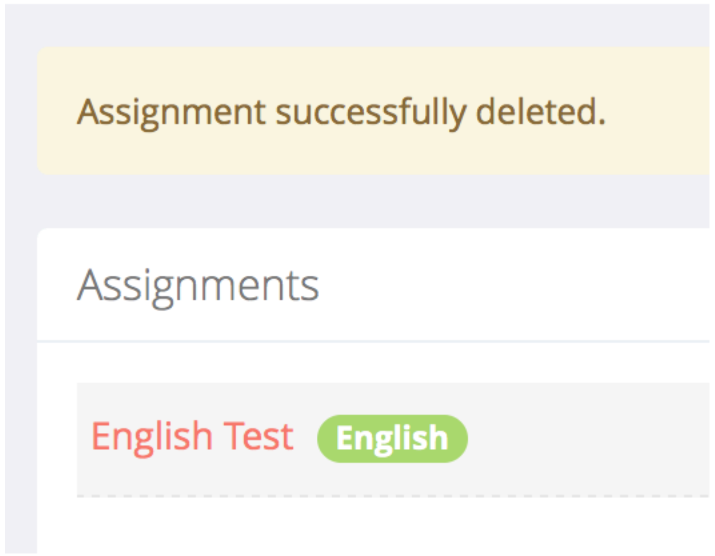
*Screenshot 17*



Delete Action

Are you sure you want to delete assignment "Maths Homework"?

*Screenshot 18*

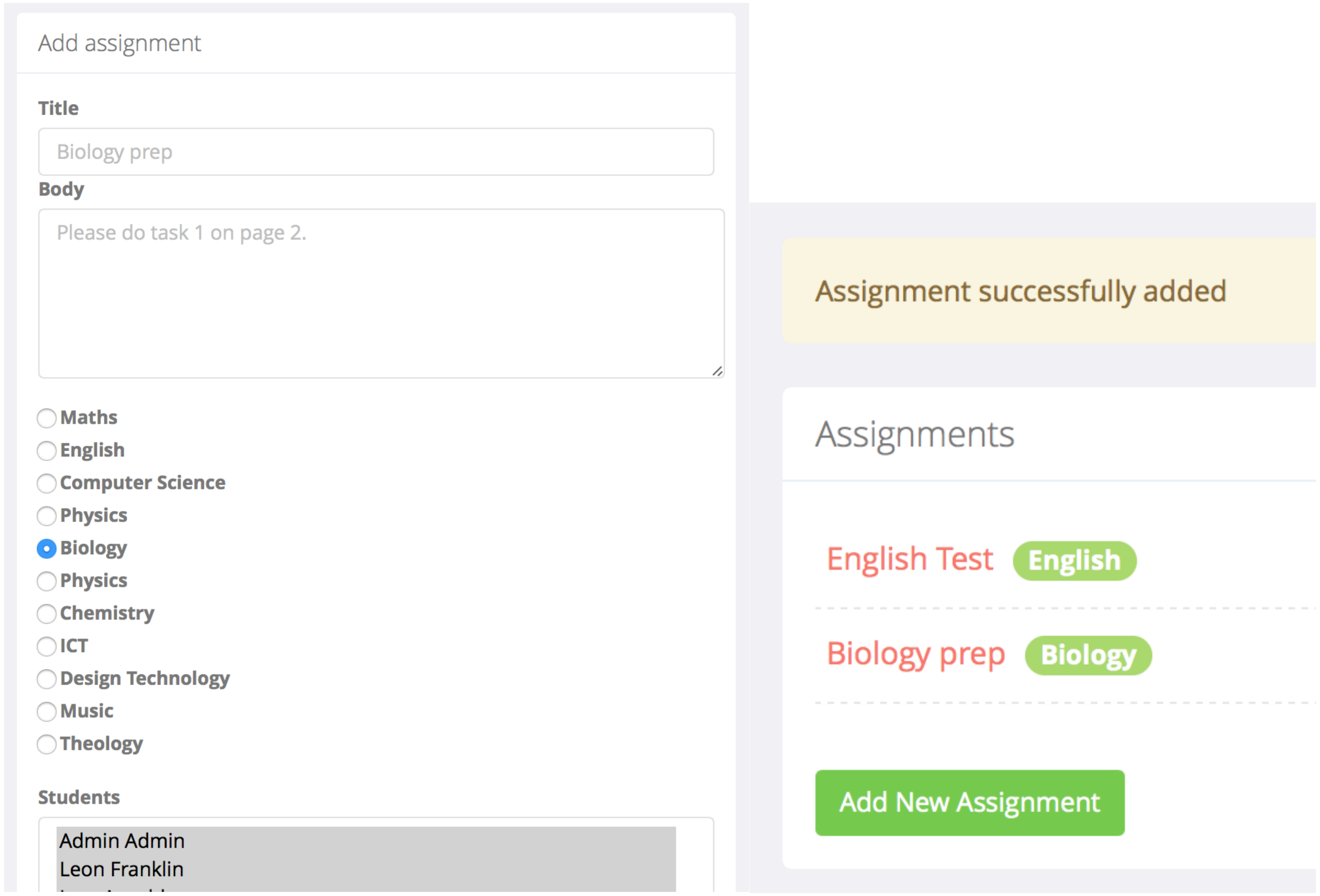


Assignment successfully deleted.

Assignments

English Test

*Screenshot 19*



Add assignment

Title

Biology prep

Body

Please do task 1 on page 2.

( ) Maths
( ) English
( ) Computer Science
( ) Physics
[✓ Biology]
( ) Physics
( ) Chemistry
( ) ICT
( ) Design Technology
( ) Music
( ) Theology

Students

Admin Admin
Leon Franklin

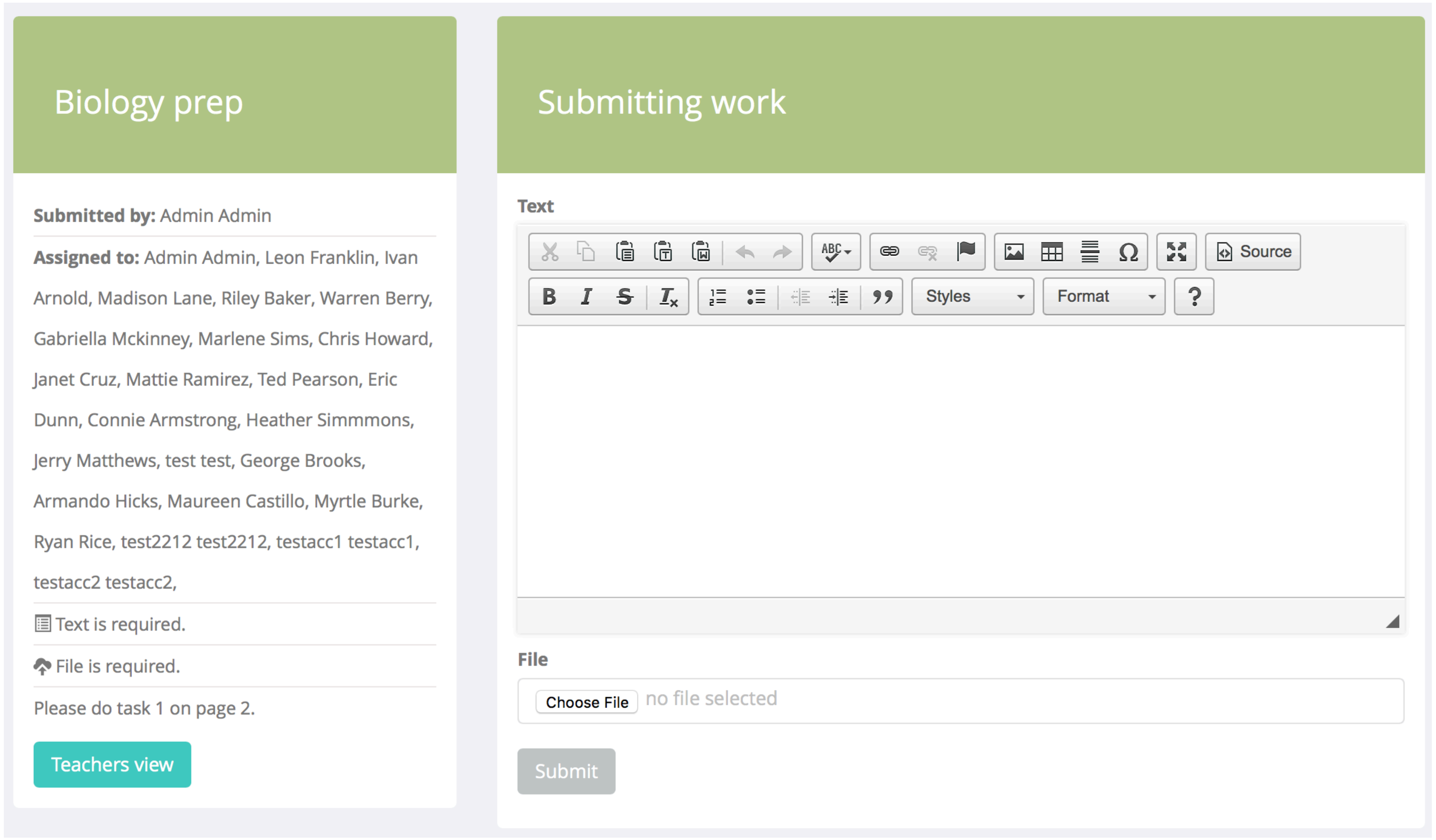
Assignment successfully added

Assignments

English Test English

Biology prep Biology

*Screenshot 20*



Biology prep

**Submitted by:** Admin Admin

**Assigned to:** Admin Admin, Leon Franklin, Ivan Arnold, Madison Lane, Riley Baker, Warren Berry, Gabriella Mckinney, Marlene Sims, Chris Howard, Janet Cruz, Mattie Ramirez, Ted Pearson, Eric Dunn, Connie Armstrong, Heather Simmmons, Jerry Matthews, test test, George Brooks, Armando Hicks, Maureen Castillo, Myrtle Burke, Ryan Rice, test2212 test2212, testacc1 testacc1, testacc2 testacc2,

Text is required.

File is required.

Please do task 1 on page 2.

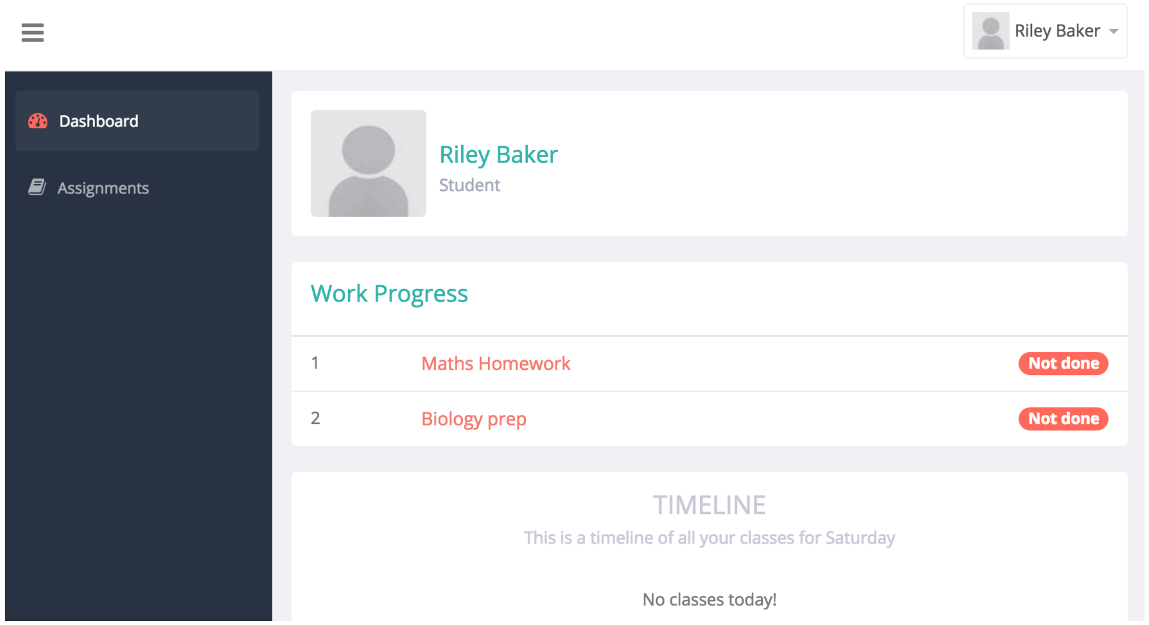
Submitting work

**Text**

**File**

no file selected

*Screenshot 21*



Riley Baker

Dashboard

Assignments

Riley Baker
Student

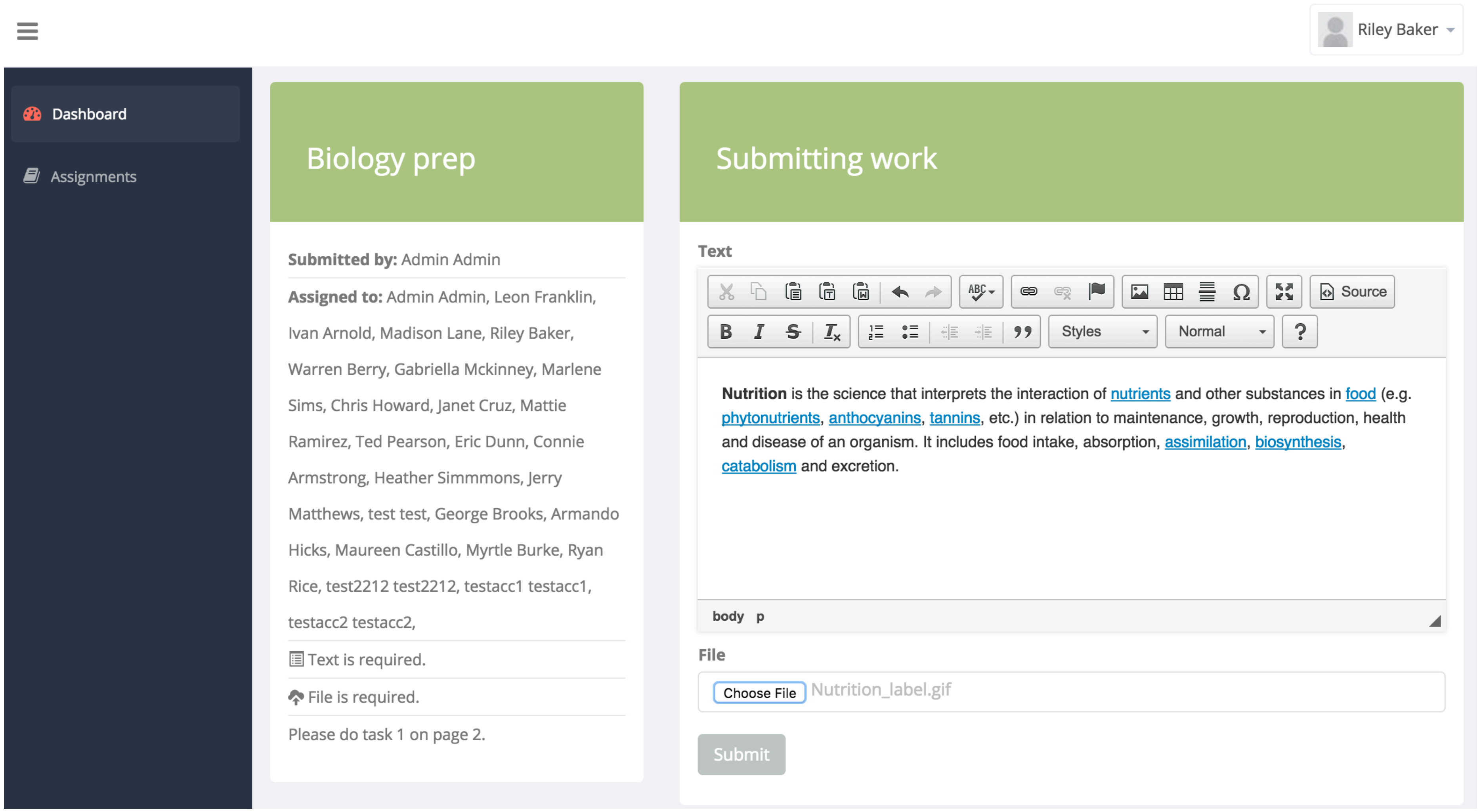
Work Progress

| | | |
|---|---|---|
| 1 | Maths Homework | Not done |
| 2 | Biology prep | Not done |

TIMELINE
This is a timeline of all your classes for Saturday

No classes today!

*Screenshot 22*



Biology prep

**Submitted by:** Admin Admin

**Assigned to:** Admin Admin, Leon Franklin, Ivan Arnold, Madison Lane, Riley Baker, Warren Berry, Gabriella Mckinney, Marlene Sims, Chris Howard, Janet Cruz, Mattie Ramirez, Ted Pearson, Eric Dunn, Connie Armstrong, Heather Simmmons, Jerry Matthews, test test, George Brooks, Armando Hicks, Maureen Castillo, Myrtle Burke, Ryan Rice, test2212 test2212, testacc1 testacc1, testacc2 testacc2,

Text is required.

File is required.

Please do task 1 on page 2.

Submitting work

Text

**Nutrition** is the science that interprets the interaction of nutrients and other substances in food (e.g. phytonutrients, anthocyanins, tannins, etc.) in relation to maintenance, growth, reproduction, health and disease of an organism. It includes food intake, absorption, assimilation, biosynthesis, catabolism and excretion.

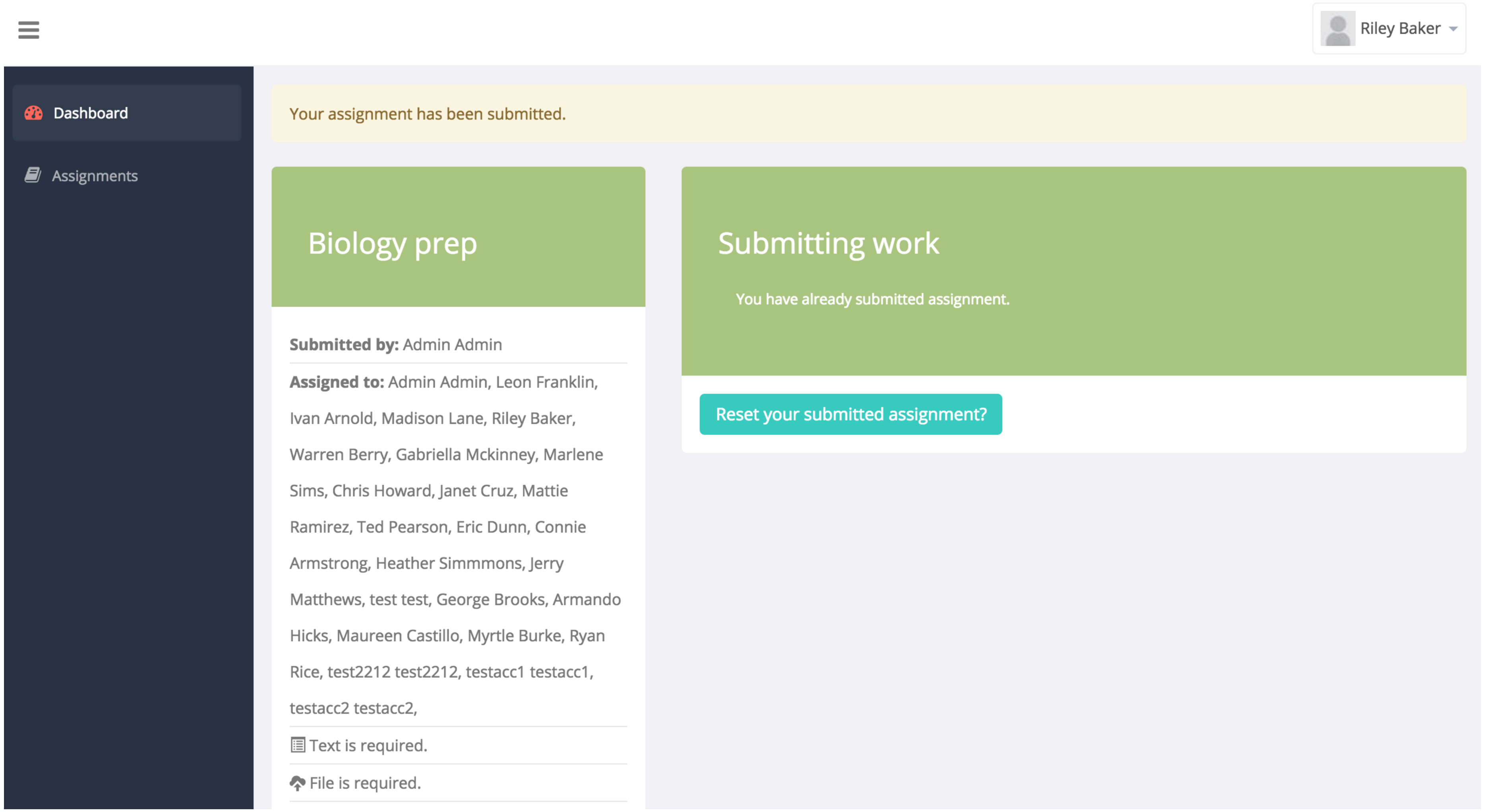
body p

File

Choose File Nutrition_label.gif

Submit

*Screenshot 23*



Your assignment has been submitted.

Biology prep

**Submitted by:** Admin Admin

**Assigned to:** Admin Admin, Leon Franklin, Ivan Arnold, Madison Lane, Riley Baker, Warren Berry, Gabriella Mckinney, Marlene Sims, Chris Howard, Janet Cruz, Mattie Ramirez, Ted Pearson, Eric Dunn, Connie Armstrong, Heather Simmmons, Jerry Matthews, test test, George Brooks, Armando Hicks, Maureen Castillo, Myrtle Burke, Ryan Rice, test2212 test2212, testacc1 testacc1, testacc2 testacc2,

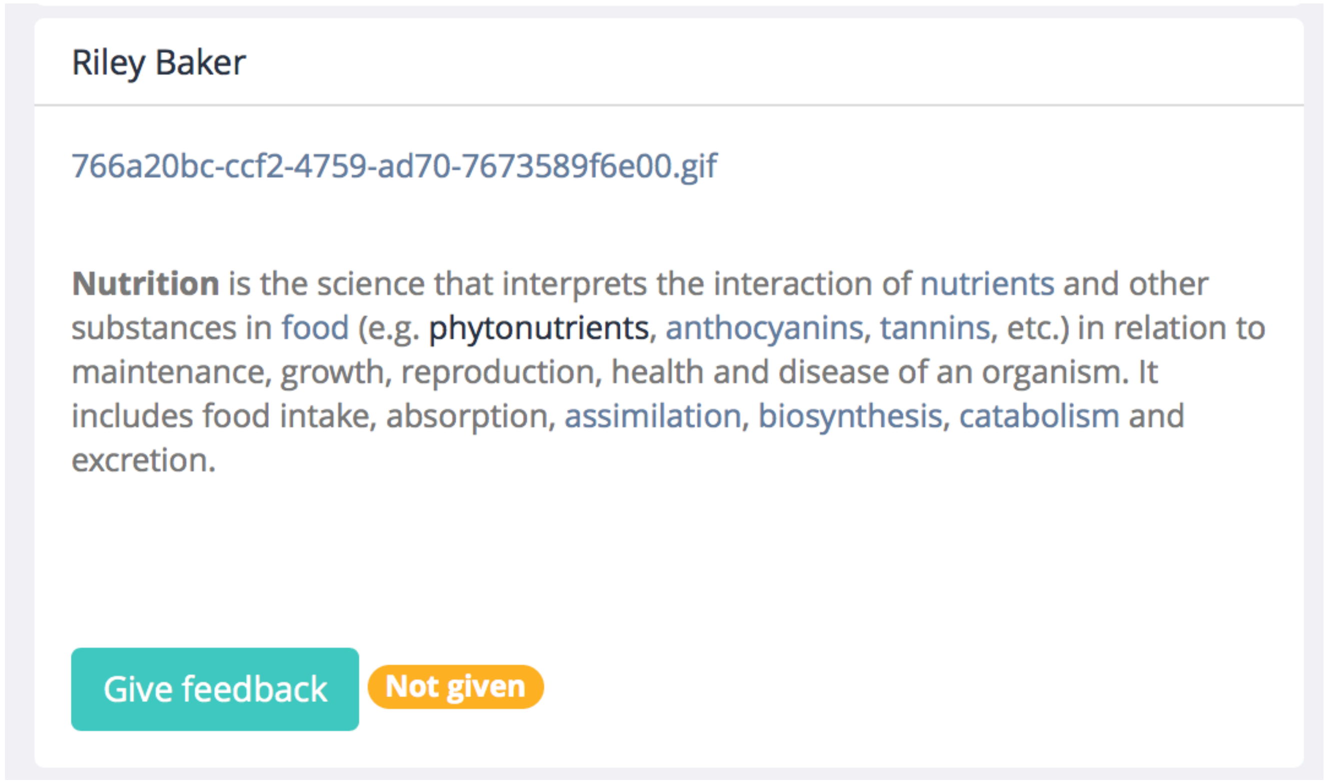
Text is required.

File is required.

Submitting work

You have already submitted assignment.

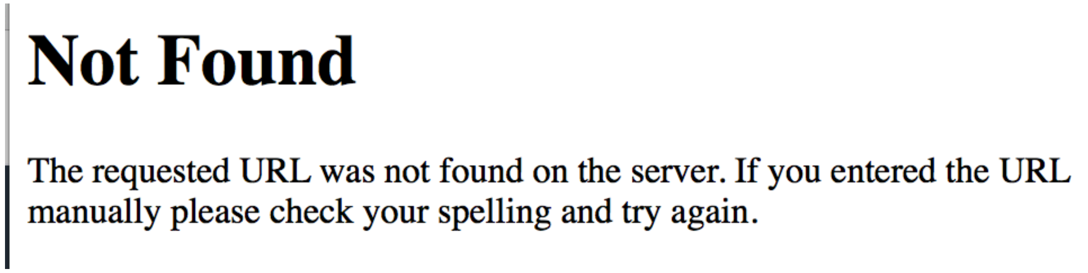
*Screenshot 23.1*


Riley Baker

766a20bc-ccf2-4759-ad70-7673589f6e00.gif

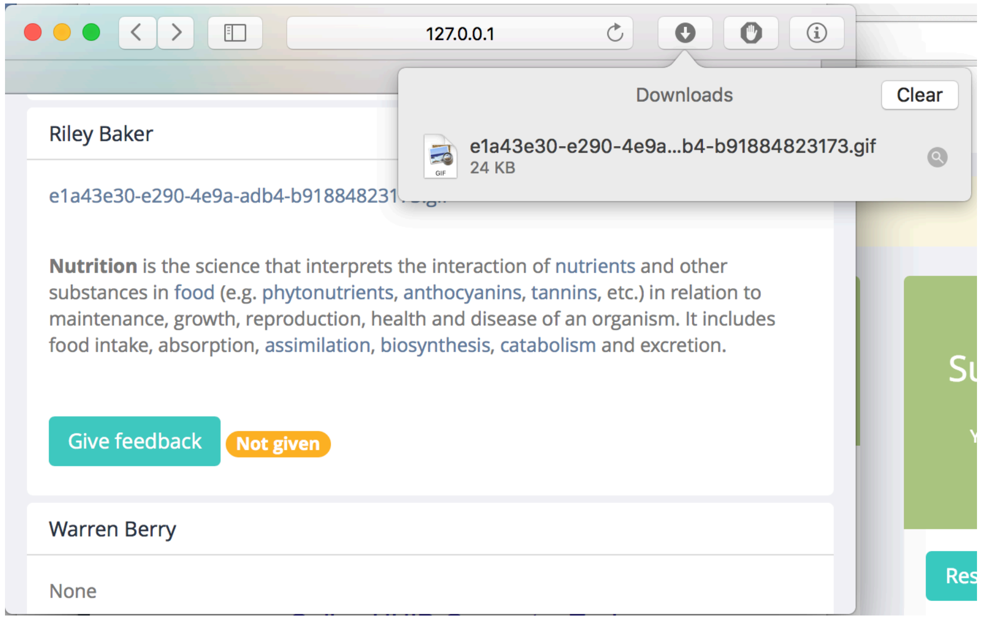
**Nutrition** is the science that interprets the interaction of nutrients and other substances in food (e.g. phytonutrients, anthocyanins, tannins, etc.) in relation to maintenance, growth, reproduction, health and disease of an organism. It includes food intake, absorption, assimilation, biosynthesis, catabolism and excretion.

*Screenshot 24*


Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

*Screenshot 25*



Riley Baker

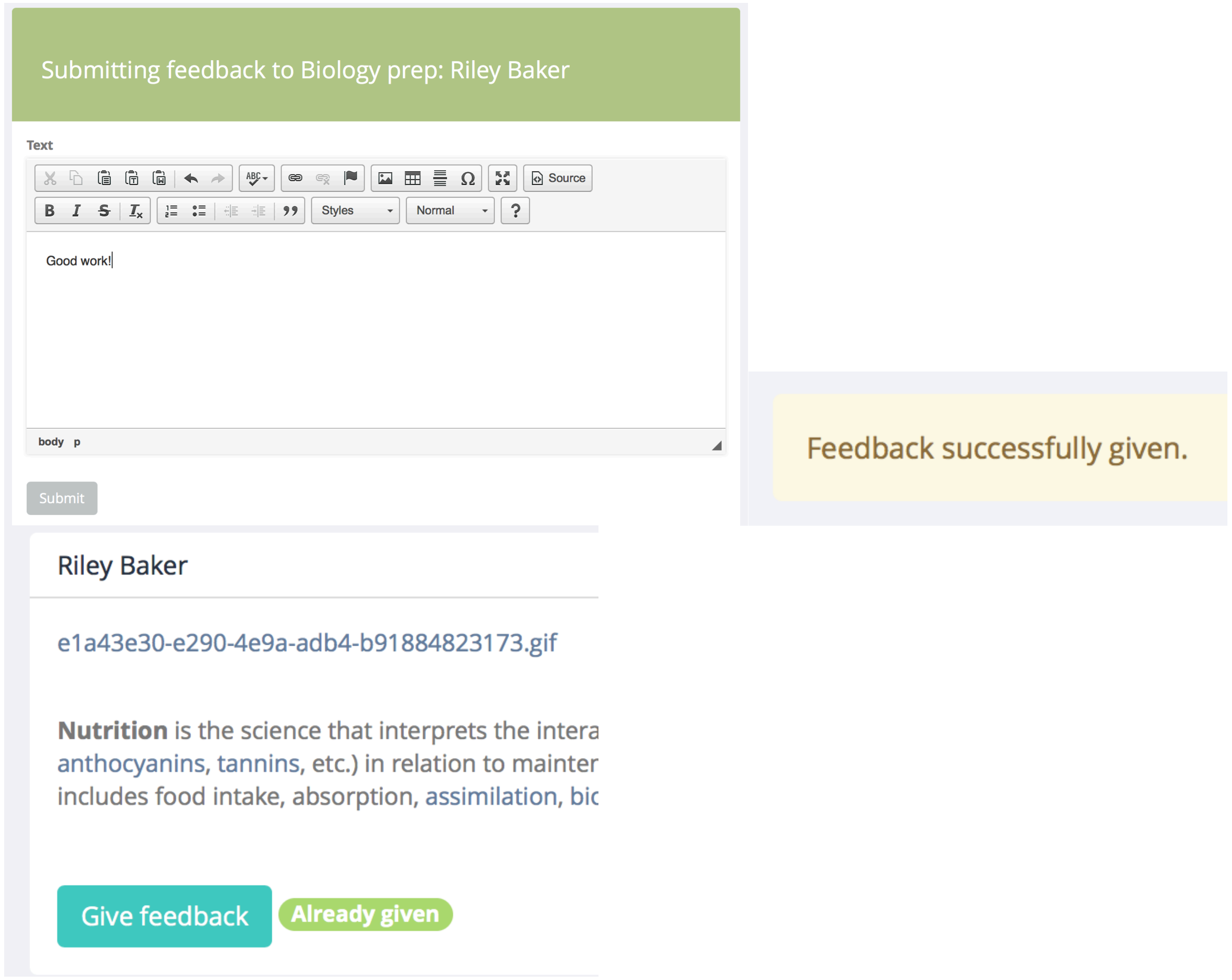
e1a43e30-e290-4e9a-adb4-b91884823173.gif

**Nutrition** is the science that interprets the interaction of nutrients and other substances in food (e.g. phytonutrients, anthocyanins, tannins, etc.) in relation to maintenance, growth, reproduction, health and disease of an organism. It includes food intake, absorption, assimilation, biosynthesis, catabolism and excretion.

Warren Berry

None

*Screenshot 25.1 (fixed)*



Submitting feedback to Biology prep: Riley Baker

Text

Good work!

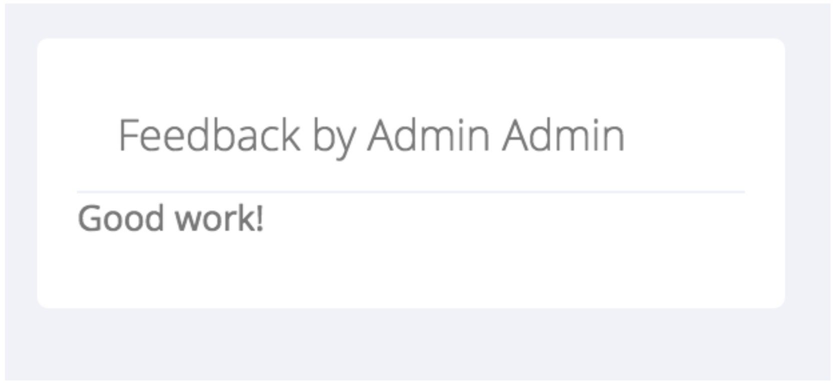
Feedback successfully given.

Riley Baker

e1a43e30-e290-4e9a-adb4-b91884823173.gif

**Nutrition** is the science that interprets the intera
anthocyanins, tannins, etc.) in relation to mainter
includes food intake, absorption, assimilation, bic

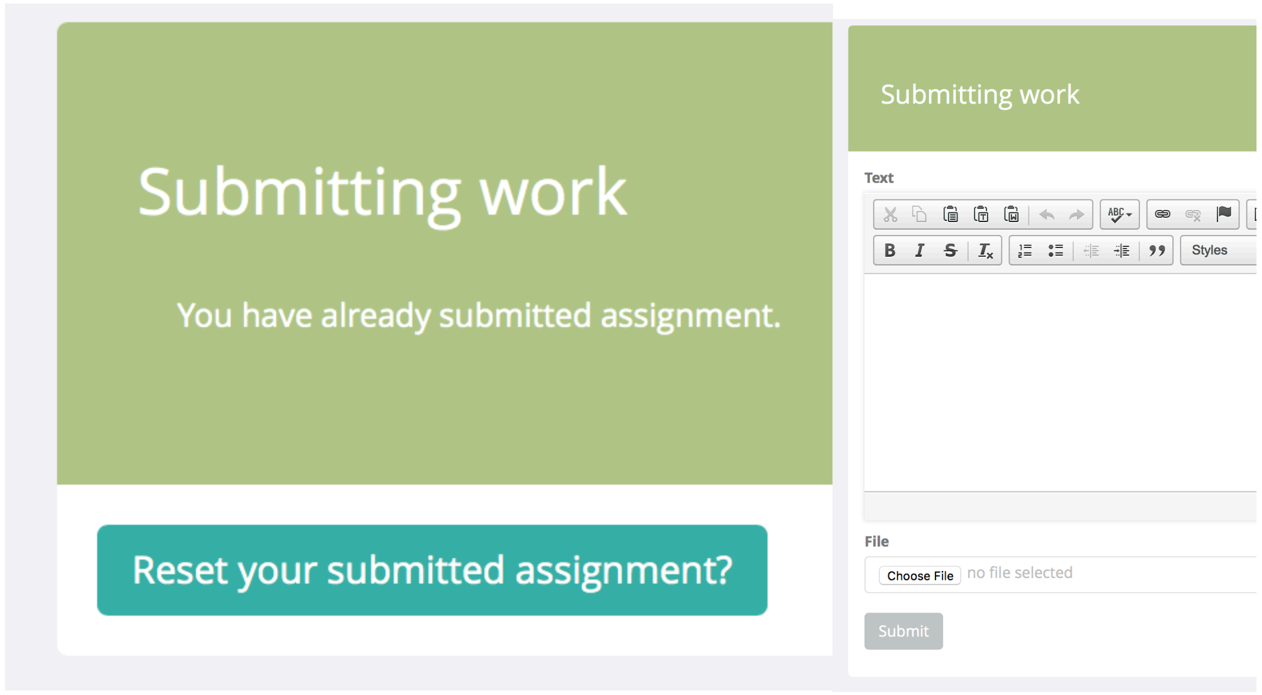
*Screenshot 26*



Feedback by Admin Admin

Good work!

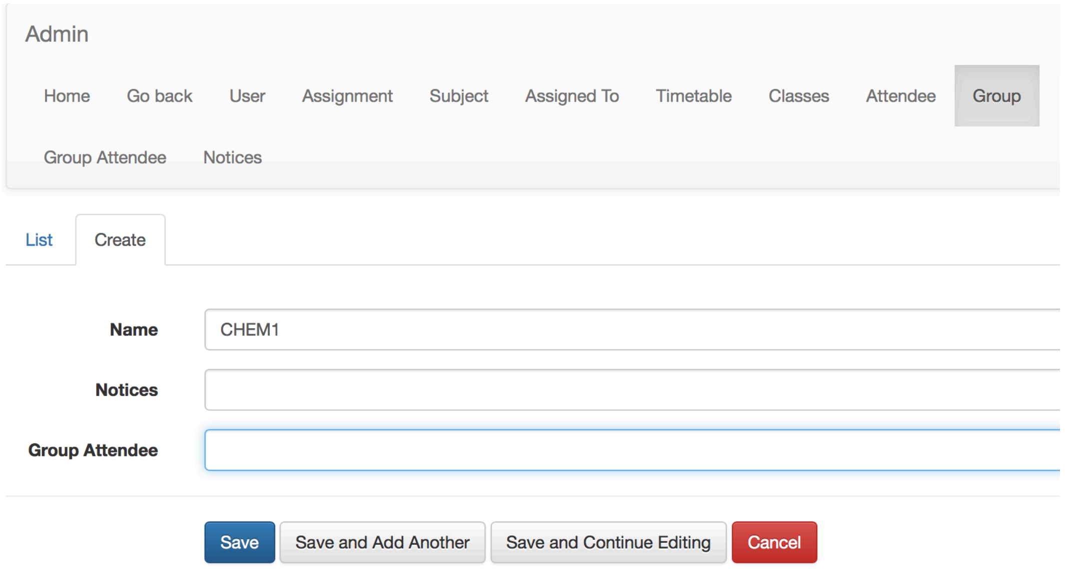
*Screenshot 27*



Submitting work

You have already submitted assignment.

*Screenshot 28*



Admin

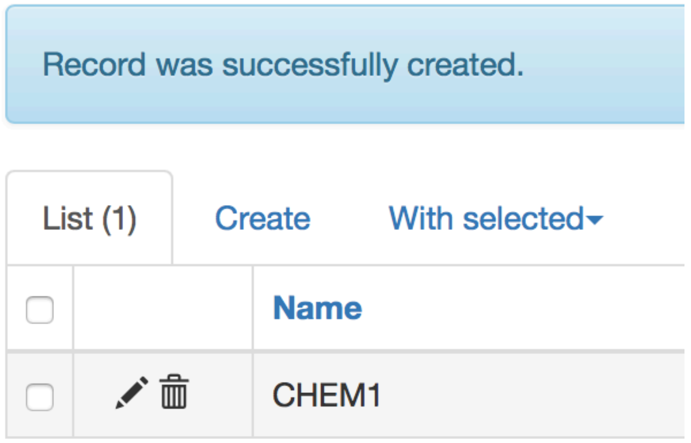
Home Go back User Assignment Subject Assigned To Timetable Classes Attendee Group

Group Attendee Notices

List Create

| | |
|---|---|
| **Name** | CHEM1 |
| **Notices** | |
| **Group Attendee** | |

Save | Save and Add Another | Save and Continue Editing | Cancel



Record was successfully created.

List (1) Create With selected

| | | **Name** |
|---|---|---|
| | | CHEM1 |

*Screenshot 29*

## Trace tables

**Testing Excel import**

**Description:**
Import algorithm reads the Excel file row by row from pre-defined range of cells and then sets fields from each cell in a database record.

**Code being tested:**

```
for row in
ws1.iter_rows('A4:I29'):
        '''
        Iterating through columns,
        numbers can be adjusted
        Each field is assigned by using cells in a row
        '''
        if row[0].value == "" or row[0].value == None:
            break
        first_name = row[0].value
        last_name = row[1].value
        username = row[2].value
        if username == None:
            username = first_name.lower() + "." + last_name.lower()
        email = row[3].value
        password = str(row[4].value)
        dateofbirth = row[5].value
        gender = row[6].value
        phone = row[7].value
        nationality = row[8].value
```

```
    # if user exists already, don't add him
    if User.query.filter_by(username=username).first():
        pass
    else:
        user = User(username=username, email=email,
password=generate_password_hash(password),
        first_name=first_name, last_name=last_name, gender=gender,
phone=phone,
        nationality=nationality)
        db.session.add(user)
        db.session.commit()
```

**Expected result:**
For each row in a spreadsheet, a record is made in the database, if it doesn't exists yet (checking by username).

| # | First_name | Last_name | Username | Email | Password Encrypted | DateofBirth | Gender | Phone | Nationality | OK? |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | Joseph | Rice | joseph123 | joseph@ mail.com | 123 | 12/12/2012 | Male | 123 456 789 | UK | Yes Commited |
| **2** | George | Brooks | george.brooks | george@ mail.com | 123 | 25/01/1997 | Male | 123 456 789 | UK | Yes Commited |
| **3** | Armando | Hicks | armando.hicks | armando@ mail.com | 123 | 10/10/1997 | Male | 123 456 789 | German | Yes Commited |

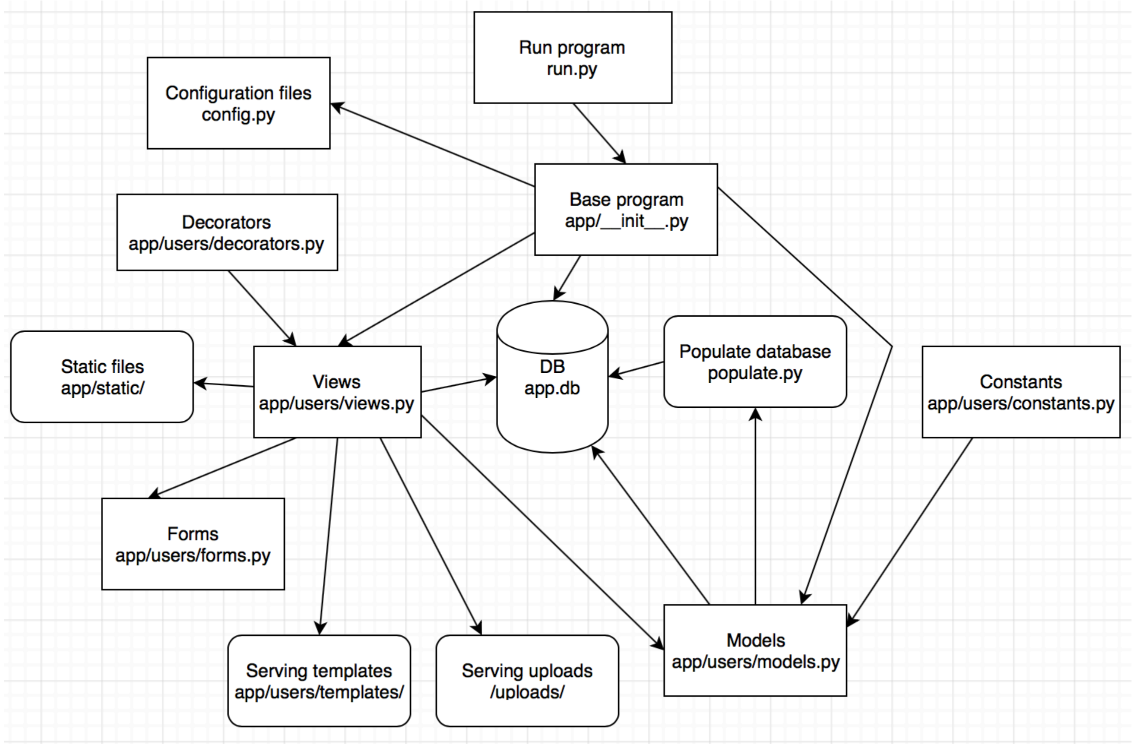
| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **4** | Haris | Duratovic | haris024 | haris@ mail.com | 123 | 04/06/1994 | Male | 123 456 789 | UK | No (already exists) Not commited |
| **5** | Myrtle | Burke | myrtle.burke | myrtle@ mail.com | 123 | 03/06/2004 | Female | 123 456 789 | UK | Yes Commited |
| **6** | Ryan | Rice | ryan.rice | ryan@ school.com | 123 | 31/12/1998 | Male | 123 456 789 | UK | Yes Commited |

## System Maintenance

### System Overview

The system is cross-platform; it can be run on Mac OS X, Linux (and other distros like Ubuntu) and Windows. The easiest way to install it on is Ubuntu, as it comes with pre-packaged Python 3 and it is easy to install modules.

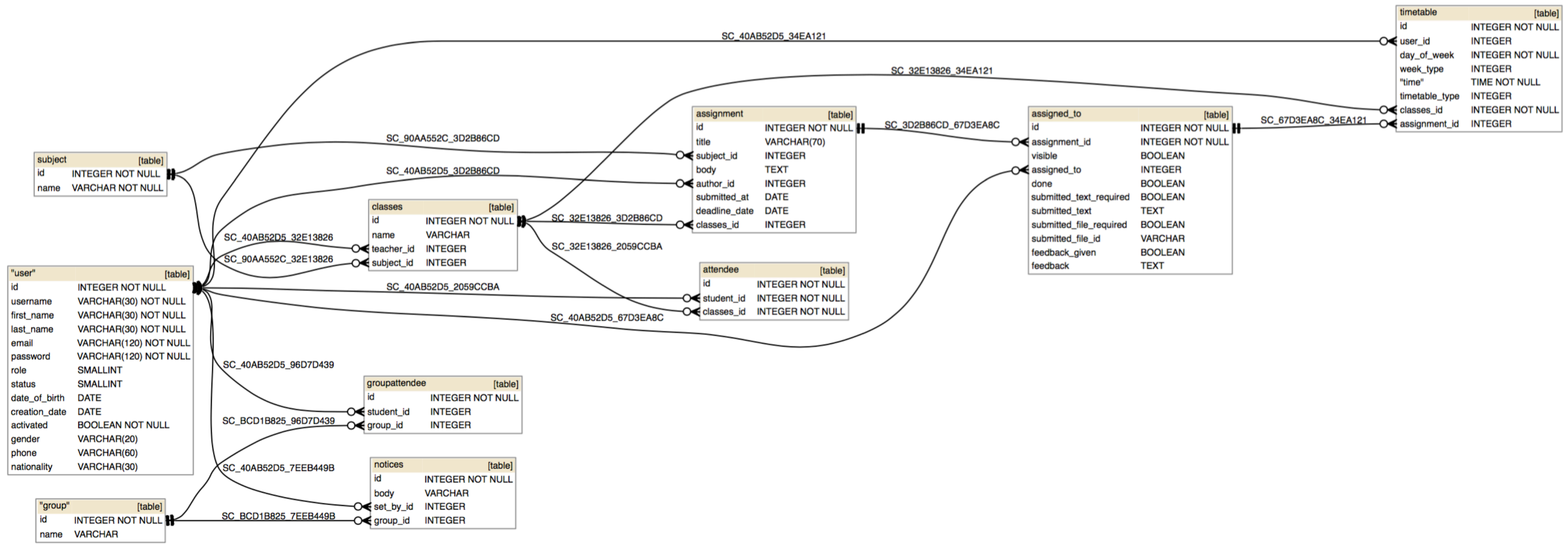
I have made a project based on a modular system. That means that different functions are in different files.
Here is a diagram showing all main code splitted into different files:



**run.py** – it is where program starts and it is where is should be started. By running «python run.py», the program starts.
**App/__init__.py** – base, where most dependencies are imported, database initialises and blueprints activates. Also there are settings for admin panel view and all models that are needed to be imported to admin panel are there.
**Config.py –** this is where configuration files are stored.
**Decorators.py –** decorators such as login_required
**Views.py –** all routes and views are stored here.
**Populate.py –** script where database can be rebuilt using new schema and populated with random data. Mostly used for testing
**Models.py –** all models for database are stored there.
**Forms.py –** all forms for WTForms are stored here.
**Constants.py –** constants for models.

**app.db –** main database file
**app/users/templates/** - location for all templates (HTML)
**app/static/ -** location for all static files (JS, CSS, images)
**/uploads/ -** location for file uploads (uploaded assignments)

## E-R diagram


timetable [table]
id INTEGER NOT NULL
user_id INTEGER
day_of_week INTEGER NOT NULL
week_type INTEGER
"time" TIME NOT NULL
timetable_type INTEGER
classes_id INTEGER NOT NULL
assignment_id INTEGER
SC_40AB52D5_34EA121
SC_32E13826_34EA121
assignment [table]
id INTEGER NOT NULL
title VARCHAR(70)
subject_id INTEGER
body TEXT
author_id INTEGER
submitted_at DATE
deadline_date DATE
classes_id INTEGER
SC_3D2B86CD_67D3EA8C
assigned_to [table]
id INTEGER NOT NULL
assignment_id INTEGER NOT NULL
visible BOOLEAN
assigned_to INTEGER
done BOOLEAN
submitted_text_required BOOLEAN
submitted_text TEXT
submitted_file_required BOOLEAN
submitted_file_id VARCHAR
feedback_given BOOLEAN
feedback TEXT
SC_67D3EA8C_34EA121
SC_90AA552C_3D2B86CD
SC_40AB52D5_3D2B86CD
subject [table]
id INTEGER NOT NULL
name VARCHAR NOT NULL
classes [table]
id INTEGER NOT NULL
name VARCHAR
teacher_id INTEGER
subject_id INTEGER
SC_32E13826_3D2B86CD
SC_32E13826_2059CCBA
SC_40AB52D5_32E13826
SC_90AA552C_32E13826
attendee [table]
id INTEGER NOT NULL
student_id INTEGER NOT NULL
classes_id INTEGER NOT NULL
"user" [table]
id INTEGER NOT NULL
username VARCHAR(30) NOT NULL
first_name VARCHAR(30) NOT NULL
last_name VARCHAR(30) NOT NULL
email VARCHAR(120) NOT NULL
password VARCHAR(120) NOT NULL
role SMALLINT
status SMALLINT
date_of_birth DATE
creation_date DATE
activated BOOLEAN NOT NULL
gender VARCHAR(20)
phone VARCHAR(60)
nationality VARCHAR(30)
SC_40AB52D5_2059CCBA
SC_40AB52D5_67D3EA8C
SC_40AB52D5_96D7D439
groupattendee [table]
id INTEGER NOT NULL
student_id INTEGER
group_id INTEGER
SC_BCD1B825_96D7D439
SC_40AB52D5_7EEB449B
notices [table]
id INTEGER NOT NULL
body VARCHAR
set_by_id INTEGER
group_id INTEGER
"group" [table]
id INTEGER NOT NULL
name VARCHAR
SC_BCD1B825_7EEB449B

**Adding assignment algorithm**

This piece of code takes subject and students choices from database and sends them to user, and if user sends request with a valid form, assignment is added.

<u>Pseudo code</u>

```
Form = AddAssignmentForm

Subject_choices = MakeTuple(DB.Query.Subjects.all())
Form.subject.choies = Subject_choices
Students_choices = MakeTuple(DB.Query.Students.all())
Form.students.choices = Students_choices

If form.isValid():
    user = DB.Query.filter_by(id=user_id).first()
    assignment = Assignment(title=form.title.data, subject_id=form.subject.data, body=form.body.data, author=user)
    DB.Session.add(assignment)
    DB.sesssion.commit()

    For each student in form:
        DB.Session.Add(AssignedTo())
        DB.Session.Commit()
    Flash("Assignment successfully added")
    Return RedirectTo(Users.Assignments)
Return render(users.assignments.html)
```

Real code

```
@mod.route('/add_assignment/', methods=['GET', 'POST'])
@requires_admin
def add_assignment():
    """
    Adding assignment
    """

    form = AddAssignment(coerce=int) # wtforms expects int

    # adding options for the form subjects
    subject_choices = Subject.query.all()
    subject_dict = [(subject.id, str(subject)) for subject in subject_choices]
    form.subject.choices = subject_dict

    # adding options for the form users
    students_choices = User.query.all()
    students_dict = [(student.id, str(student)) for student in students_choices]
    form.students.choices = students_dict

    if form.validate_on_submit():
        user = User.query.filter_by(id=g.user.id).first()
        assignment = Assignment(title=form.title.data, subject_id=form.subject.data, body=form.body.data, author=user)
        db.session.add(assignment)
        db.session.commit()

        for student in form.students.data:
            assigned_to = AssignedTo(assignment_id=assignment.id, assigned_to=student,
submitted_file_required=form.file_required.data,
                submitted_text_required=form.text_required.data)
            db.session.add(assigned_to)
            db.session.commit()

        flash("Assignment successfully added")
        return redirect(url_for("users.assignments"))
        ##



    return render_template('users/add_assignment.html', user=g.user, form=form)
```

## User manual

**Introduction to School Web portal:**

This program is made for better communication between teachers and students, and also giving necessary information to the student. Teachers can send assignments to students, and students can send their work back to teacher, and teacher can give them feedback. Students can also see lots of information like notices, their classes and what time/teacher.

### Installation guide

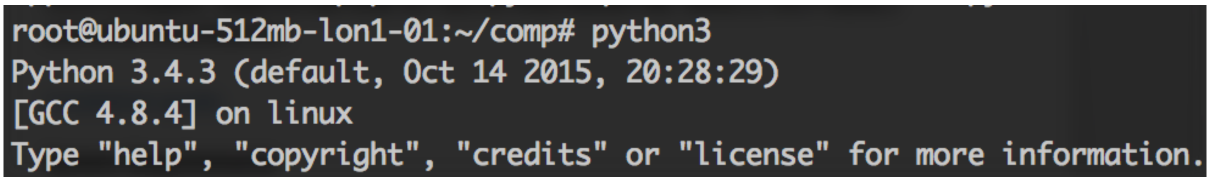
**System Requirements**

- Linux-based operating system
- Python installed on the computer
- No less than 50MB of memory storage
- Correct permissions set so that files can be uploaded onto /uploads folder

**Installation for Linux (Ubuntu)**

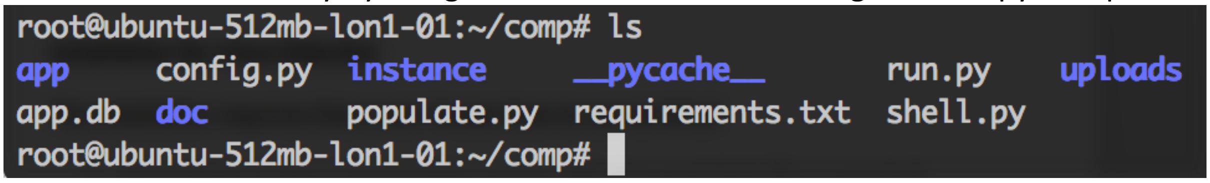
The installation requires that a user knows how to use terminal

1. Check that you have appropriate version of Python installed. The supported version is Python 3.x. If you type «python3» in console, this should pop up:



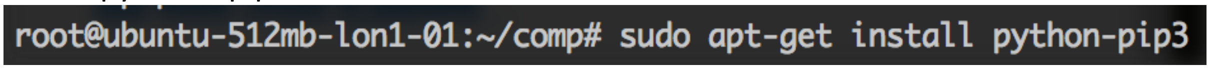
```
root@ubuntu-512mb-lon1-01:~/comp# python3
Python 3.4.3 (default, Oct 14 2015, 20:28:29)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

2. Go to folder with program by using «cd» commands and check that you're in correct directory by using «ls» command and checking for «run.py» script.



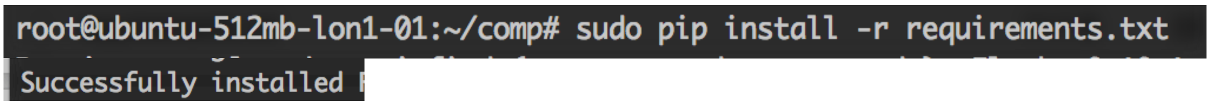
```
root@ubuntu-512mb-lon1-01:~/comp# ls
app     config.py  instance     __pycache__       run.py    uploads
app.db  doc        populate.py  requirements.txt  shell.py
root@ubuntu-512mb-lon1-01:~/comp# 
```

3. Then, the installation of required modules is needed. «PIP» is used to install required modules. Type «sudo apt-get update» and then «sudo apt-get install python3-pip»



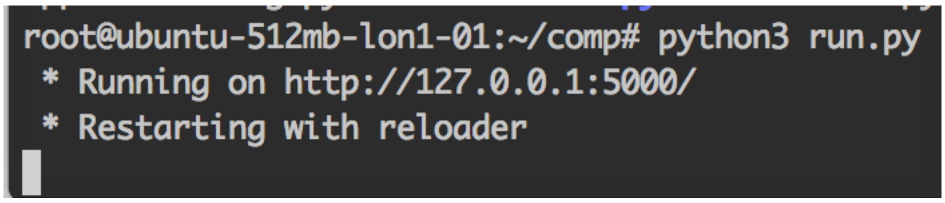
```
root@ubuntu-512mb-lon1-01:~/comp# sudo apt-get install python-pip3
```

4. After that, type in «pip install –r requirements.txt». It will take some time to download modules.



```
root@ubuntu-512mb-lon1-01:~/comp# sudo pip install -r requirements.txt
Successfully installed
```

5. Type in «python3 run.py». Now the app is successfully installed and running.



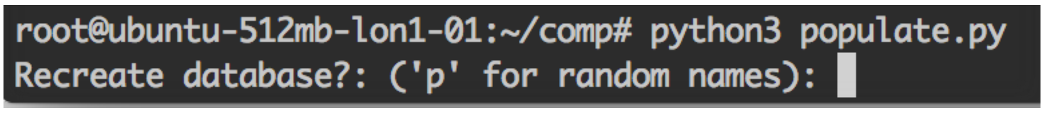
```
root@ubuntu-512mb-lon1-01:~/comp# python3 run.py
 * Running on http://127.0.0.1:5000/
 * Restarting with reloader
```

**Keeping program alive**

To keep our program working after closing the terminal window, we can use program called Screen (https://www.gnu.org/software/screen/), which keeps terminal screen working after closing, or we can use Supervisor (http://supervisord.org), which can be configured so that our program can be remotely stopped or started.

**Database management**

We can use «python3 populate.py» command for rebuilding database and adding dummy users to it. After using it, this message pops up:



```
root@ubuntu-512mb-lon1-01:~/comp# python3 populate.py
Recreate database?: ('p' for random names):
```

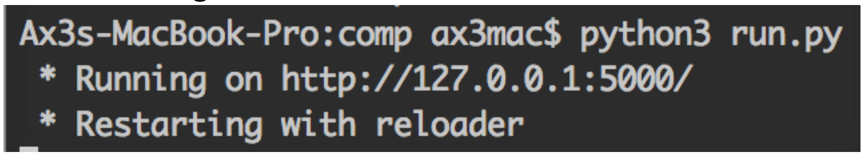
If you enter «y», database will be deleted and new one will be made. Also admin user will be added. If you press «p», a few fake users will be also added.

**Configuration**

In order to adapt the program to the environment of your school, it is useful to change a few settings. In «run.py» script, we can change debug state and which port it is going to be used. By default it is set up at «5000», if you want users to access without specifying port, you should choose port «80». Make sure that it is available for program to use (i.e. not occupied by other program).

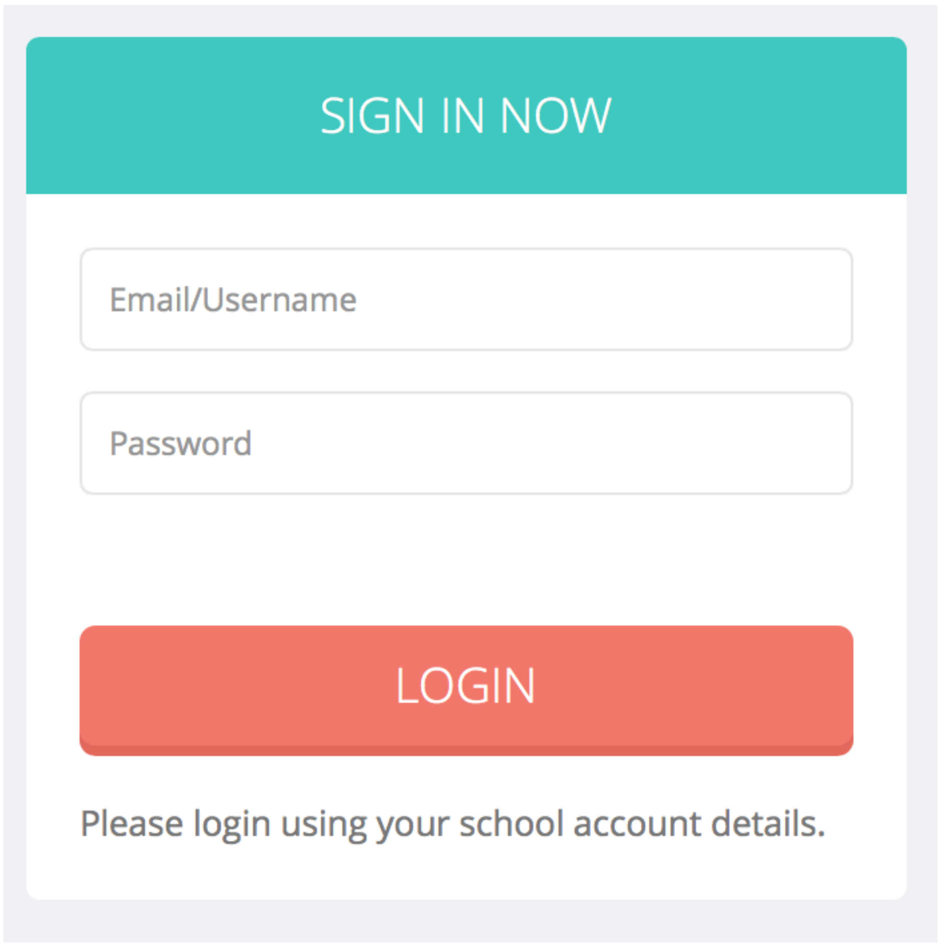
**Running program**

After setting up database, type «python3 run.py» in order to run the program. With default settings it should look like this:



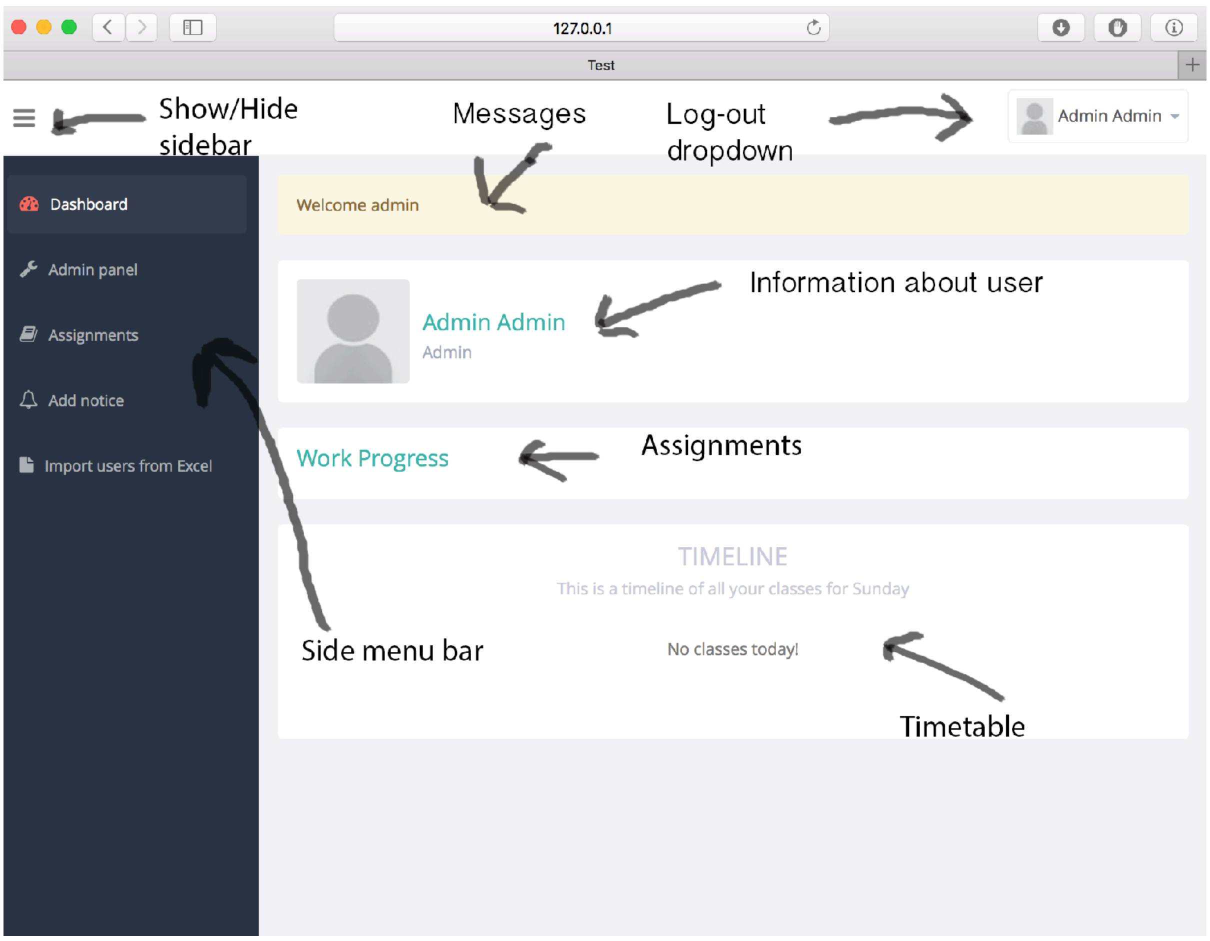
```
Ax3s-MacBook-Pro:comp ax3mac$ python3 run.py
 * Running on http://127.0.0.1:5000/
 * Restarting with reloader
```

After that, go to http://127.0.0.1:5000/users/login/ (login page). Change port and IP address appropriately if you changed them in configuration. This window should pop up.



Please login using your school account details.

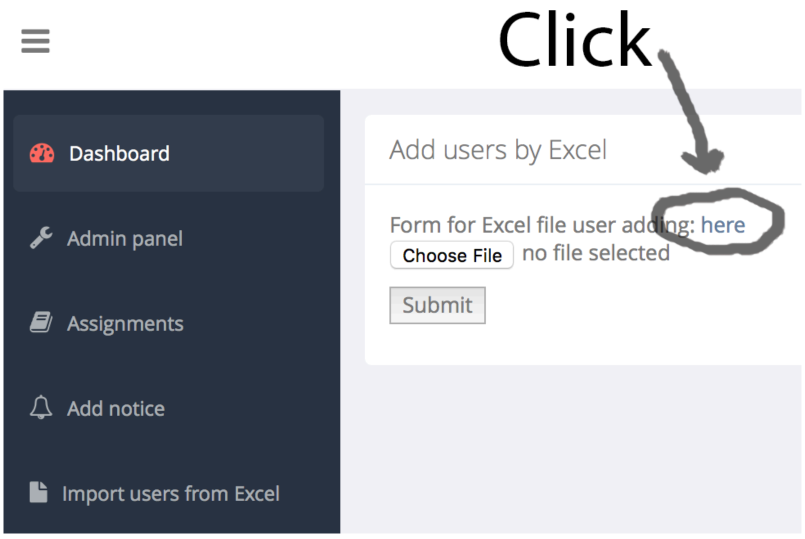
Default login/password for administrator is “admin/admin”.



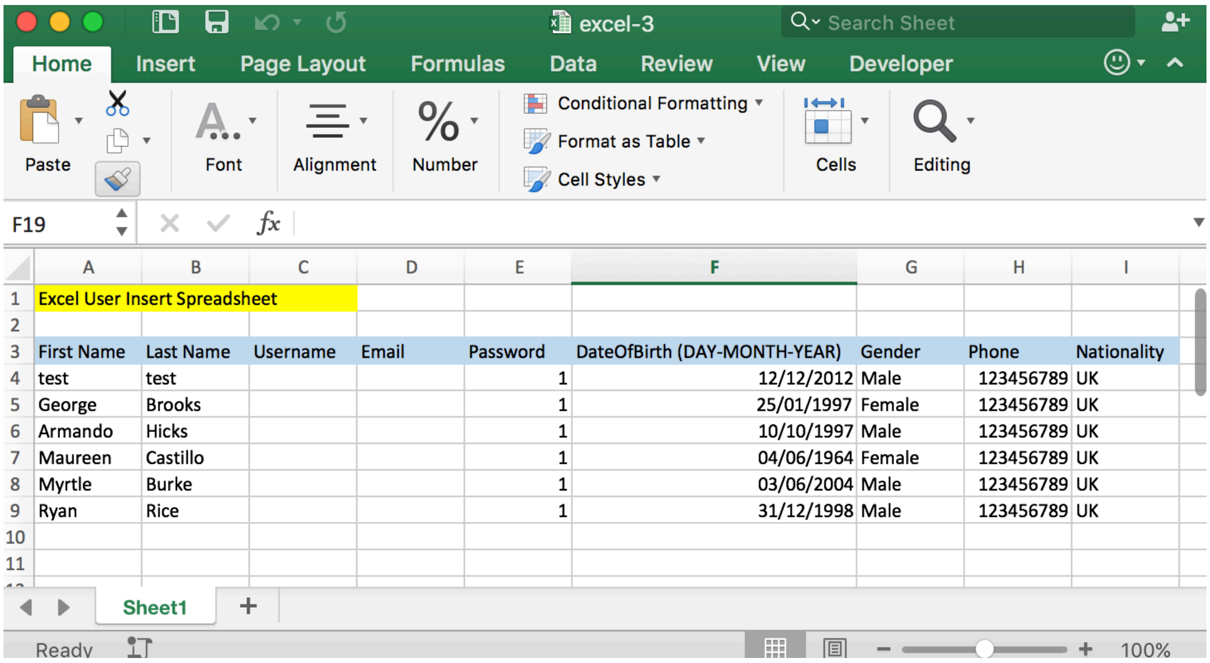
Here is the main dashboard of the website after log-in.

**Adding users from Excel file**

It is relatively easy to add users from excel file. Click on «Import users from Excel» at the sidebar, and click on hyperlink where it says «Form for Excel file user adding».



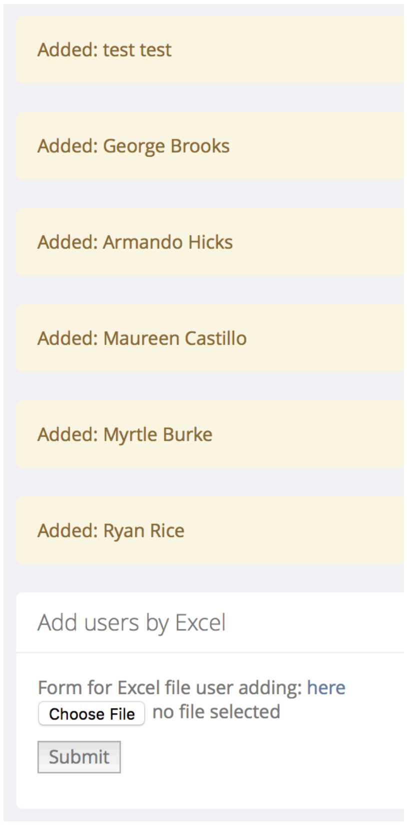
There is already data inside Excel file, so if you want, clear it and fill it with your own data.



| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Excel User Insert Spreadsheet | | | | | | | | |
| 2 | | | | | | | | | |
| 3 | First Name | Last Name | Username | Email | Password | DateOfBirth (DAY-MONTH-YEAR) | Gender | Phone | Nationality |
| 4 | test | test | | | 1 | 12/12/2012 | Male | 123456789 | UK |
| 5 | George | Brooks | | | 1 | 25/01/1997 | Female | 123456789 | UK |
| 6 | Armando | Hicks | | | 1 | 10/10/1997 | Male | 123456789 | UK |
| 7 | Maureen | Castillo | | | 1 | 04/06/1964 | Female | 123456789 | UK |
| 8 | Myrtle | Burke | | | 1 | 03/06/2004 | Male | 123456789 | UK |
| 9 | Ryan | Rice | | | 1 | 31/12/1998 | Male | 123456789 | UK |
| 10 | | | | | | | | | |
| 11 | | | | | | | | | |

Username will be generated from first name and last name, and email will be set up when user logins in for first time.

When you finished editing Excel file, upload it back to the website. All users that are added will be shown here.



Added: test test

Added: George Brooks

Added: Armando Hicks

Added: Maureen Castillo

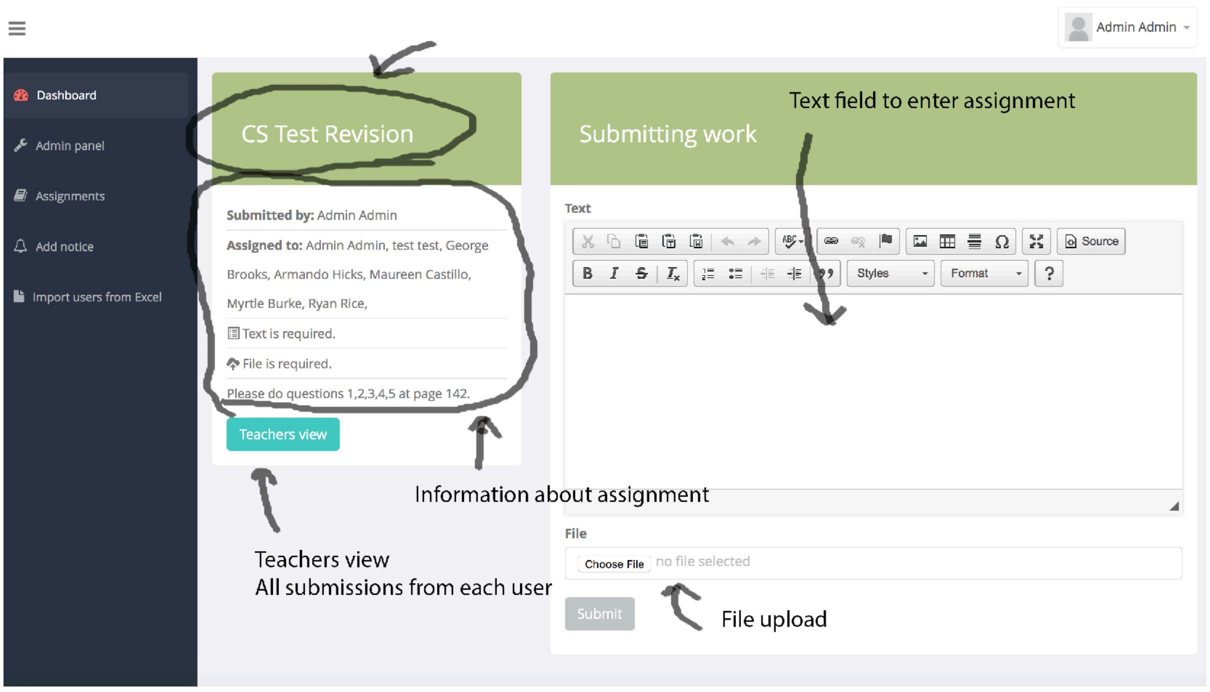
Added: Myrtle Burke

Added: Ryan Rice

Add users by Excel

Form for Excel file user adding: here

## Assignments view



**Submitted by:** Admin Admin

**Assigned to:** Admin Admin, test test, George Brooks, Armando Hicks, Maureen Castillo, Myrtle Burke, Ryan Rice,

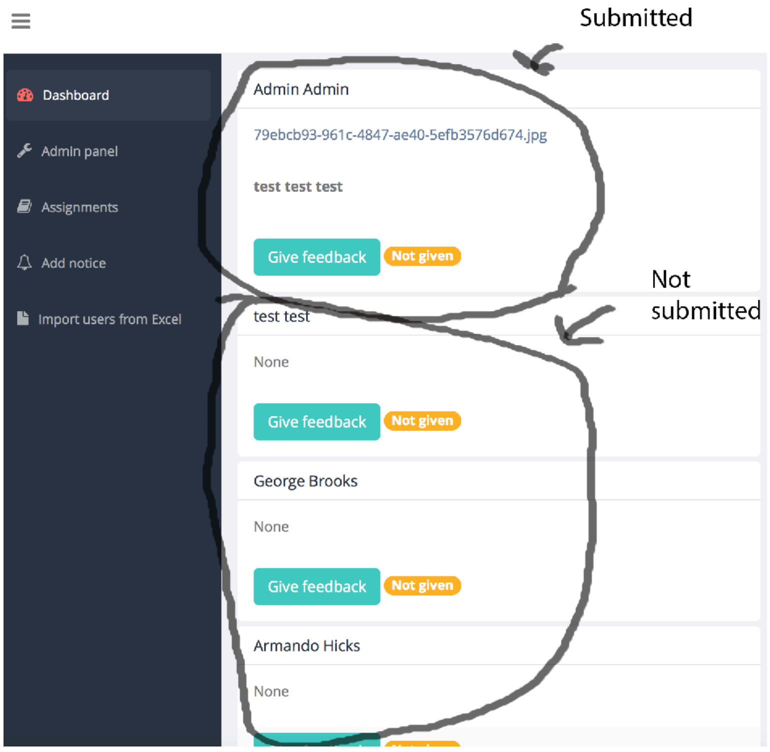
Text is required.

File is required.

Please do questions 1,2,3,4,5 at page 142.

Here is the assignments view, where all information about assignment and submission form is showed.

## Feedback view



Admin Admin

79ebcb93-961c-4847-ae40-5efb3576d674.jpg

test test test

test test

None

George Brooks

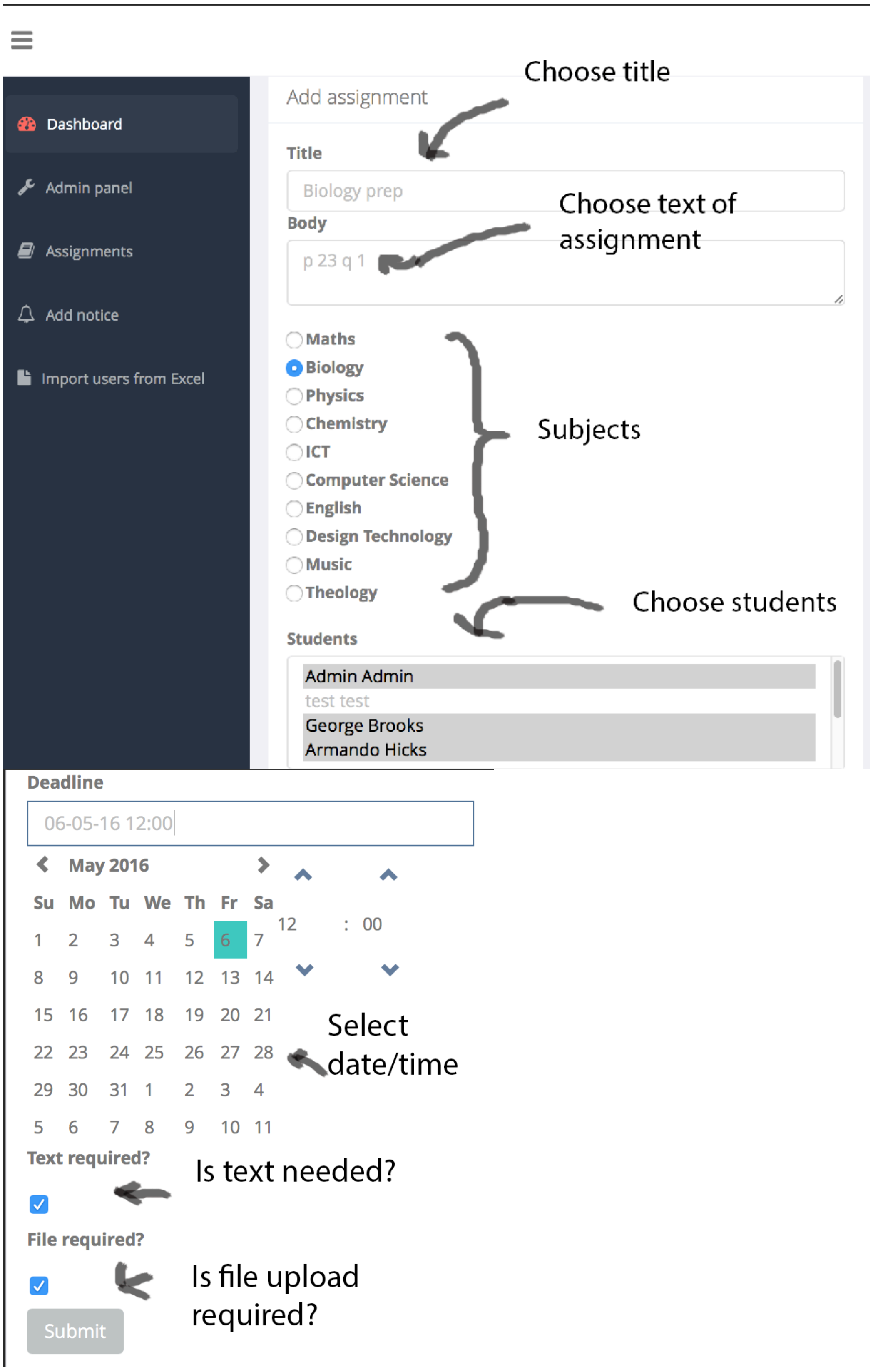
None

Armando Hicks

None

Here is all submitted data from each user.

## Adding assignment



Add assignment

Title: Biology prep

Body: p 23 q 1

( ) Maths
(●) Biology
( ) Physics
( ) Chemistry
( ) ICT
( ) Computer Science
( ) English
( ) Design Technology
( ) Music
( ) Theology

Students
Admin Admin
test test
George Brooks
Armando Hicks

Deadline: 06-05-16 12:00

May 2016

| Su | Mo | Tu | We | Th | Fr | Sa |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 | 30 | 31 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |

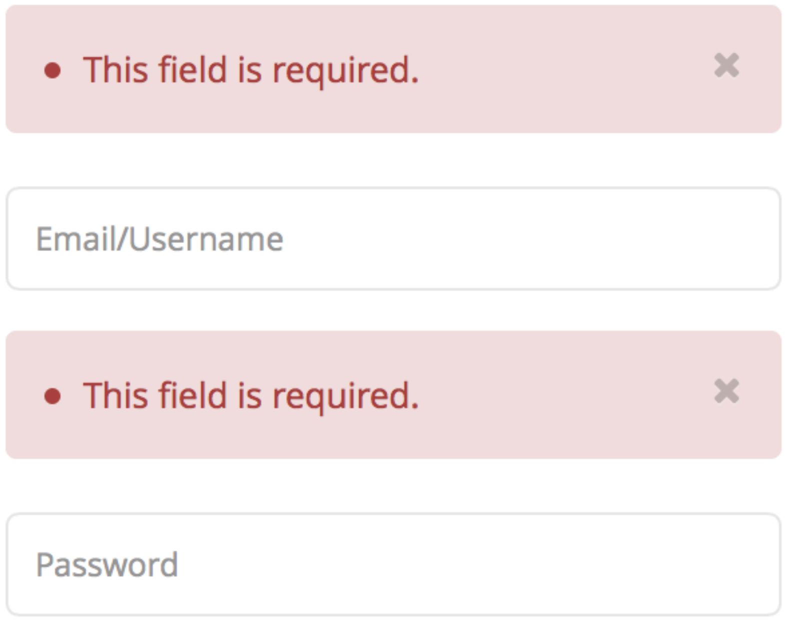
12 : 00

Text required? ☑

File required? ☑

## Error handling

In every form, there is error handling. For example, presence check, this is where the entered value is empty:



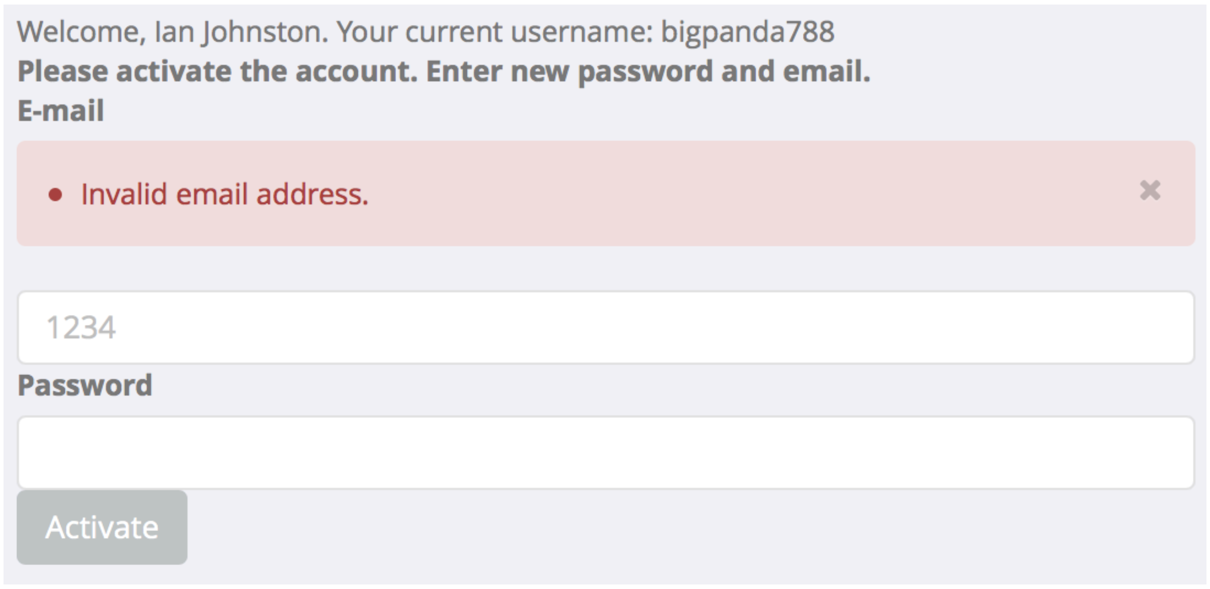
- This field is required.

Email/Username

- This field is required.

Password

Invalid email:



Welcome, Ian Johnston. Your current username: bigpanda788

**Please activate the account. Enter new password and email.**

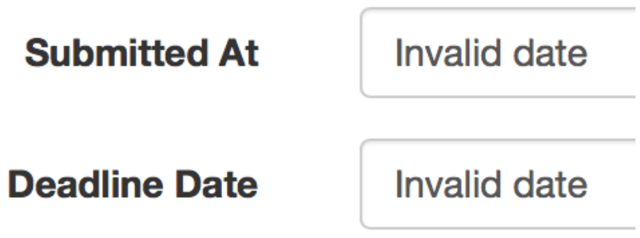
**E-mail**

- Invalid email address.

1234

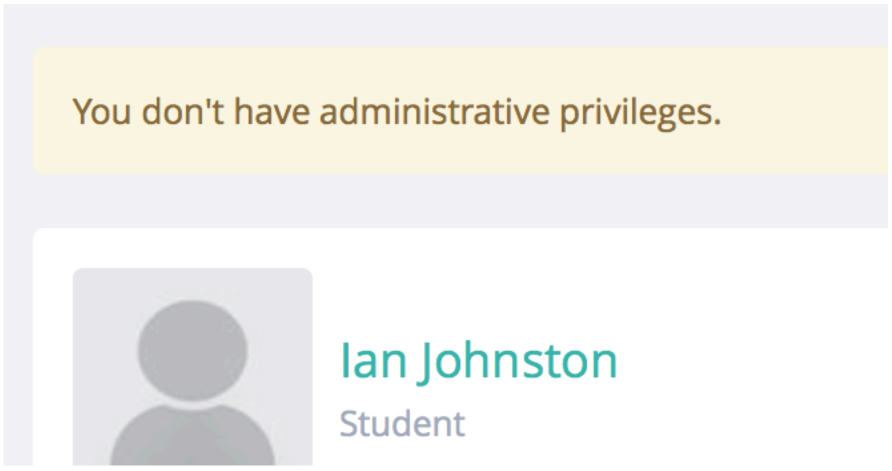
**Password**

When incorrect date is given, this message shows up:



| | |
|---|---|
| **Submitted At** | Invalid date |
| **Deadline Date** | Invalid date |

If student tries to access sections of website that are only available for teachers and administrators:



You don't have administrative privileges.

Ian Johnston
Student

## Appraisal

**Feedback letter**

*Overall I found Nazar's system very effective and I was hugely impressed with the final management system for schools. I did have concerns about the project – I did wonder about the effectiveness and user friendly nature as it's vital in a time challenged role that the system allowed easy access and navigation.*

*The system seems to meet all the objectives and I was certainly overwhelmed when I saw the marriage between detail and the user-friendly nature of the layout and front screen. I was particular struck how Nazar enabled me to directly contact members of my house and even manage sub groups like prefects and tutor staff. Navigation was simple yet hugely effective and I felt very comfortable with control panel and the options given.*

*But, to make it the main system for the school, more work has to be made, for example, I would like separate paging system for each school subject, where I can upload my own material for students.*

*Overall, I was able to work very quickly on it and its usefulness as a data controller and tool for assessment was brilliant.*

*Well done Nazar and what an impressive piece of work.*

*[redacted]*

**Analyse of feedback**

The feedback I was given was mostly positive, especially about design and usability.

| Objective | Met? | Comment |
|---|---|---|
| **1. Program should have log-in system, a method to register users, have activation system, log-out system** | Yes | This was relatively easy to do, but managing database and tables and hashing/checking password was the hardest part |
| **2. User interface should show all assignments** | Yes | Positive feedback |
| **3. User inputs must be validated to avoid erroneous or incorrect data.** | Yes | This was achieved both with manual checking and forms with validation |
| **4. Permissions**<br>**4.1. Unlogged users can't access main part of system**<br>**4.2. Students can't access some of parts of system**<br>**4.3. Teachers can access most of parts of system except admin panel**<br>**4.4. Administrators can access everything** | Yes | - |
| **5. There is teacher assigned to classes/groups and** | Yes | Positive feedback |

| | | |
|---|---|---|
| **students are assigned to classes/groups** | | |
| **6. Teacher can send assignment to students** | Yes | Positive feedback |
| **7. Students can upload their finished work to the system**<br>**7.1. Students can upload text**<br>**7.2. Students can upload files** | Yes | Positive feedback given |
| **8. Teacher can submit feedback to a student** | Yes | Positive feedback |
| **9. Users can reset submitted work or delete assignment** | Yes | - |
| **10. Timetable available for a student (timeline)** | Yes | - |
| **11. Administrators can populate database with data (Excel file, etc.)** | Yes | Positive feedback |

**Extensions**

There are lots of ways in which the project can be improved:

- Ability to add customised pages with information on it (for example, related to a subject)
- Ability to see classes and who are on those classes
- Integration with existing authentication systems (like Microsoft Active Directory)
- Ability to see other users' profiles

- Integration with e-mail systems (notifications about new assignments, feedback given, etc)
- Messaging system (real-time)
- Online user help

**Reflection**

When I was starting the project, I was confident that I will do something web-based and written on Python, because on those areas I have most experience, and they are most interesting areas as well. I had loads of ideas in my head, like real-time games, web portals, etc. But I chose to do this project, because I thought that it will be optimal in terms of experience and time it will take for me to finish the project. When I was programming my project, I haven't encountered any problems.