Documentation for HCI Project

Nick Lee
Roxana Nadrag
Naomi Pentrel
Vlad Victor Ungureanu
Tom Wiesing

Contents

0.1 Roles

- Vlad Previous Courses, Robotics Pages, TA Landing and Grading pages

- Tom Login, Logout, Submit, Profiles pages

- Nick Initial layout (landing pages) and Documentation

- Roxana Current courses, Python Lab Pages, TA Grading pages

- Naomi Documentation

GitHub link:

`https://github.com/tkw1536/HCI_Grader}{HCI-Grader`

Click here : HCI-Grader

1 Personas

This section contains our 2 personas.

1.1 The Student: Alice Larson

Alice is a first year student, majoring in Applied Computational Mathematics at Jacobs University. She comes from California all the way to Germany for her undergraduate studies because she wants to expand her life experience. She discovered her passion for Mathematics and Computer Science in highschool and is trying to be better in these fields.

Alice is taking both Python Lab and C Lab this semester because she wants to gain as much knowledge about programming as possible . She is also taking Calculus as part of her mandatory courses.

Alice is a very organized student, always doing her assignments on time. She is using jGrader for all of the above courses to submit homework and also check for deadlines and previous assignments.

In her free time, Alice enjoys running and dancing. She is also passionate about painting and design and would like to take a Design class some day.

- **Name: Alice Larson**

- Student, Applied Computational Mathematics

- **Age:** 20

- **Motto:** I'm a visual person

1.2 The TA: Filip Nikolche

He studies Computer Science at Jacobs University. He transferred from University of Hamburg after his first year. He is now in his 3rd year and is writing his thesis with the Computer Networks research group. Because he wants to focus on his thesis, he decided not to take any courses this semester. His thesis topic subject is "Large Scale Measurement of Broadband Performance". However he wants to earn some extra money, so he is TAing 3 courses via jGrader, Python Lab, C Lab and Calculus I.

He was an intern at Microsoft last summer. His hobbies are reading SF books and tech articles.
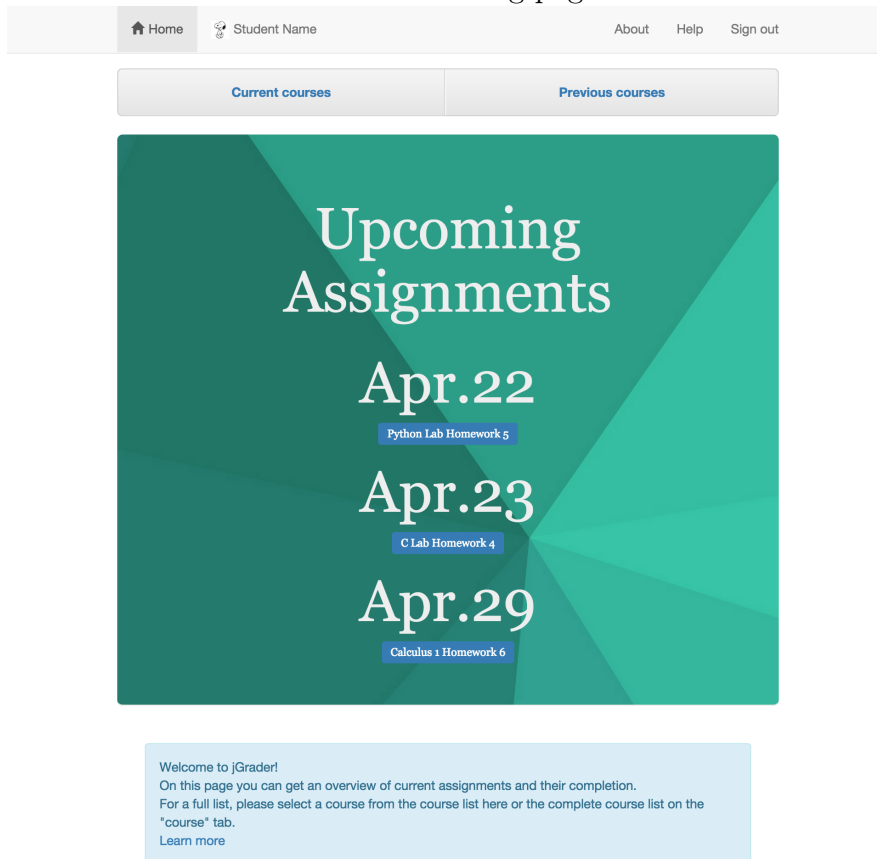

- **Name: Filip Nikolche**
- TA, Computer Science
- **Age:** 22
- **Motto:** I'm definitely a tech geek

2   Scenarios

This section contains our 2 scenarios.

2.1   Scenario 1 - Homework Submission

Alice is taking Python this year which she uses our grading platform for. When she is is on the student landing page.

From there she realizes that the next assignment due is for Python Lab. Alice clicks on the *Current courses* to have an overview of all her courses.

Current courses	Previous courses

Enrolled Courses

Python Lab

Grade so far:

48%

C Lab

Grade so far:

57%

Calculus I

Grade so far:

39%

Since she still has to do her Python homework assignment she navigates to the Python Lab.

The course page layout is designed to provide an easy to use and comprehensive dashboard for each course. It contains current assignments, previous assignments, tools to view further information, request extensions & excuses, and an overview of the course grading breakdown, instructors, credits, and other important information.

On the top of the page Alice can choose the current homework screen or the previous homework Assignments screen. On the current homework screen the most frequently accessed elements are at the top of the screen

to minimize scrolling and allow for easy accessibility. Hence the first information is when the deadline for the current assignment it and right below it is a huge submit button that cannot be missed. Apart from this it also has buttons to view the current assignment, to request and extension, and to request an excuse. The colors for the buttons are green, yellow, and red respectively which underlines how crucial the respective action is. E.g. requesting an excuse is a more crucial action than requesting an extension.

Below the buttons that allow Alice to perform actions, the course page offers information about the course and the grading criteria. Being naturally curious about the grades she received on previous homework assignments, she goes to that section and can see her previous submissions and the grades she received for them in a nicely formatted table.

Going back to the current homework page, Alice wants to now submit her current homework assignment. For that she clicks on the submit button and is redirected to the submission page.

2.2 Scenario 2 - TA Grading

Our TA, Filip Nikolche, wants to grade assignments for a course he is TAing. He signs into jGrader and selects the course (Python) from the list of courses on the landing page. He then selects assignment 3 because it is not fully graded, and he begins working on it. He downloads the submissions, reads them in the text editor of his choice on his computer, and he enters his grades and comments into the page. At the end of his grading session, he updates all grade slots to ensure that the grades are saved, and he signs out.

3 Use cases

This section contains our 4 use cases.

3.1 File Submission

- **Agents:**

  1. Student

- **Purpose:** Students should be able to submit files.

- **Procedure:**

  1. Student visits authenticated landing page.
  2. This page shows the next upcoming assignments. Clicking on one of them leads to the submission pages.
  3. Alternatively, the student can click a "Current courses" pages which lists all courses.
  4. The student can then click on an individual course which lists all current assignments for that course.
  5. Here the student can click on the submit button of any assignment and also land on the submission page.
  6. On the submission page slots for all files to be submitted are listed.
  7. The student can either click on a file or drrag nd drop a file in the slot to upload it.
  8. During uploading, a progress bar appears which shows the current progress.
  9. Once the progress bar reaches 100%, the slot turns green which confirms that the file is uploaded.

3.2 Excuse Request[1]

- **Agents:**

  1. Students
  2. Professor
  3. Registrar (not directly)

- **Purpose:** Request being officially excused for a task

- **Procedure:**

  1. Student visits authenticated landing page, which shows outstanding assignments for courses.

  2. Clicking on an assignment navigates the student to the assignment page which (among other things) has a button to request an excuse for the specific assignment.

  3. The student clicks a button to request an excuse for this task from the professor.

  4. The task is marked as "EXCUSE PENDING" on the course page. [2]

  5. The professor is notified of the request via an email that is sent to them. This email includes a link to a page on Grader where he/she can confirm the extension request.

  6. (not directly in the system) the student sends a medical excuse to the registrar and waits for them to confirm it (to the professor).

  7. The professor clicks on the link in the mail that was sent to him/her earlier

  8. The professor lands on an (authenticated) page on jGrader where he/she can either confirm or deny the requests.

     - If the excuse was accepted by the registrar, the professor clicks the "Grant Excuse" button
     - If not, he/she clicks the "Deny Excuse" button

---

[1]This use case is only partially implemented in the prototype.
[2]This and the following steps are not available in the prototype.

9. The student is notified via email if the excuse has been granted or denied.

10. At the same time the excuse is marked appropriately on the landing page.

3.3 Extension Request[3]

- **Agents:**

  1. Students
  2. Professor
  3. Registrar (optional, indirectly)

- **Purpose:** Student has an emergency and is not able to complete the assignment at time.

- **Procedure:**

  1. He logs in to jGrader and sees the current timeline of tasks.

  2. He clicks on the assignment link which has a button to request an extension for the current assignment.

  3. He selects from a calendar view the available date when he can submit the solution and complete a message to motivate the extension.

  4. The website shows him a confirmation that an email has been sent to the professor regarding the extension.

  5. The professor receives an email where he has 2 options either to approve/deny the extension request.

  6. The professor decides on approving/not approving and the student receives an email regarding the new time.
     Confirmation of an illness from the registrar might be required

  7. The timeline of the student is changed accordingly.

---

[3]This use case is not implemented in the prototype.

3.4   Grading

- **Agents:**

  1. Student(s)
  2. Teaching Assistant/Instructor

- **Purpose:** A student has submitted assignments, and the grades for the work must be entered into the gradebook.

- **Procedure:**

  1. Teaching Assistant logs into jGrader and selects the course they are grading for from a list
  2. The TA selects the assignment/task submission to be graded
  3. After loading the page, the TA either downloads all submissions, or they download an individual's submission by clicking the down arrow next to the person's name
  4. The TA opens each submission and grades it accordingly on their local system outside of jGrader
  5. The TA enters their feedback into the feedback field, and the grade into the grade field, and clicks update.

     Alternately, they can skip the updating and just update all at once by clicking "Update All"
  6. The grading dashboard now shows the current grading status for the assignment, which is hopefully 100% graded.
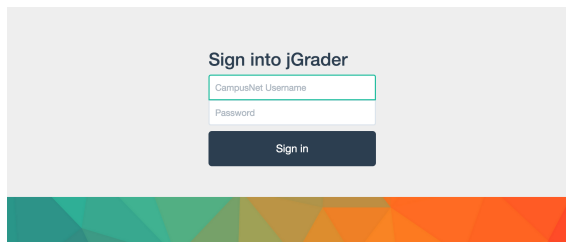
4 Design Documentation

In this section you can find design documentation about individual pages of the system. The title of each section reflects only who wrote the respective section, not who designed and implemented the page in the prototype.
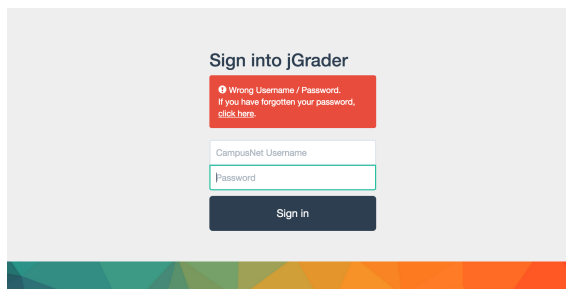
4.1 Authentication Page

The authentication page is the entry point to the system and as such we designed it very carefully. It exists in 3 versions:
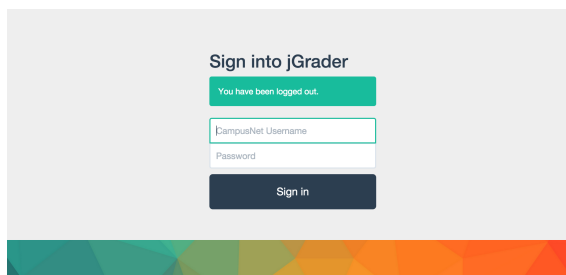
1. A default view:

2. A view that appears if the user enters an incorrect password:

3. A view that appears if the user just logged out:

Each of the versions has the same basic layout. The only real content of the page is centered. It contains a heading, input elements for username and password as well as a submit button. We designed it this way because we want to draw the users attention to its use, being as minimalistic as possible. However because it is the first contact any user has with the system we wanted to leave an impression and added a colorful background image.

When the user accesses the sign in page for the first time, the keyboard automatically focuses the username field. This makes it easy for the user to sign into the system. When the user enters a wrong password they end up the second version of the page, which in addition has an error message and a forgot password link. The keyboard is automatically focused on the password field so the user can quickly re-enter the password since it is more likely that the password, and not the username, is incorrect. To further speed up loggin in, the username field keeps the value entered in the first sign in attempt.

During (paper) prototype testing a tester mentioned that it would be convenient to have reset password functionality, so we added a link for it. On the linked page the user can enter their username and then gets an email with a link to reset the password[4]. This page does not have the username pre-entered or an auto-focus functionality because we want to prevent abuse of the system.

The third version of the authentication is shown after the user signs out of the system. It contains a small message to ensure that the user knows they have been signed out. Furthermore if auto-focuses the username field to enable the user to quickly sign in again. An earlier version did not contain a sign in form but only a message that the user had been logged out, however this was criticized during paper prototype testing.

---

[4]Because this is simply a prototype only the form is implemented, not the actual mail being sent. We also do not have a layout for the reset password page. If this system were to be used at Jacobs, the reset link might be removed because CampusNet accounts are used and we are not capable of resetting CampusNet accounts without Campus IT.

4.2 Student Landing Page
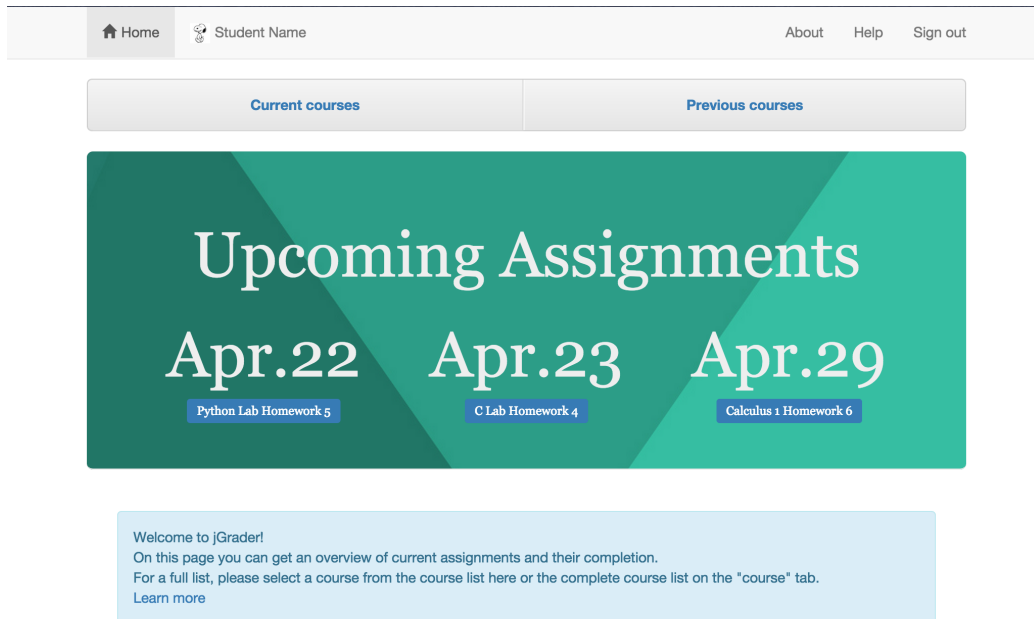
- Students are provided with an overview of upcoming deadlines and how much of the tasks are done (e.g. if the student finished 4 out of 5 tasks it will visually show 80%). This will help students remember all their deadlines.

- Links on the visual representation of the percentages will allow students to quickly navigate to the assignments that are due.

- To limit the information that is on the screen previous courses are only shown when the student clicks on the rider for previous courses. This ensures that the page provides an easily processable overview without clutter.

- In the Enrolled Courses section, the student can access all courses he is currently enrolled in.

4.3   Profile Pages

The profile pages are mostly informational, and could be useful for a course instructor to get an overview of who is taking a class and what other courses they are enrolled in.

4.4 Current Courses Page

Current courses	Previous courses

Enrolled Courses

Python Lab

Grade so far:

48%

C Lab

Grade so far:

57%

Calculus I

Grade so far:

39%
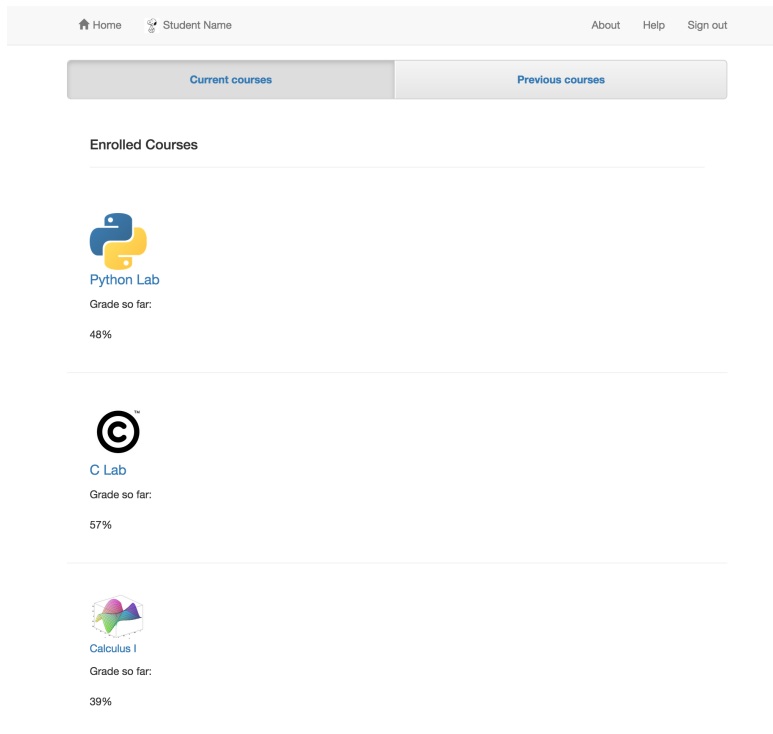
The current courses page features a sparse, left-justified layout in order to simplify the task of getting to the course page the user needs–minimal information beyond a grade overview, large & attention-catching course icons, and the page navigation buttons is included. Following the rule-of-thirds by left-justifying the content helps the user process the information quicker.

Current courses	**Previous courses**

Python Lab

Current Homework   Previous Homework Assignments

#	Deadline was	HW Name	Submission	Feedback	Grade
1	22 February	Classes and templates	✔	Very well designed	95%
2	1 March	Try Catch and Exceptions	✔	Please review thread exceptions	73%
3	8 March	GUI	✔	Crashes on loading	50%
4	12 March	System design	✘	No upload found	0%

Instructors:

Dr. Kinga Lipskoch

Credits:

2.50 ECTS

Partial Grades:

Final Exam (65%, Mandatory)
Course Assignment (35%)

Description:

This lab unit is a continuation of the first semester lab Programming in Python I. It covers advanced topics of Python programming such as object oriented programming, advanced data structures, file handling, debugging techniques and problem solving using frameworks. This lab is a service lab aimed at students who do not need to learn how to program in C in a mandatory lab. Students studying ECE, EECS, CS and ACM need to take NatSciLab CS II instead.

Python Lab

*Deadline for this assignment is : **20 April***

Submit

View assignment	Request Extension	Request Excuse

The course page layout is designed to provide an easy to use and comprehensive dashboard for each course. It contains current assignments, previous assignments, tools to view further information, request extensions & excuses, and an overview of the course grading breakdown, instructors, credits, and other important information.

More frequently accessed elements, such as the assignment page, are located at the top of the screen to minimise scrolling. Small amounts of color are used to highlight actionable items in order to help the user quickly find what they need. In some situations, the student may only have minutes or even seconds to submit a file, and navigational aids help. A large and wide submit button means that the user only has to search vertically through the page.

4.5 Previous Courses Page

This page offers an overview for the courses the student took previously, grouped by semester. The previous courses page is very similar in design to the current courses page, with addition of simple marks to indicate whether the student passed or failed the completed course, and a tab bar to navigate through previous semesters.

All the courses are presented in a table each row being separated by a new line for better understanding of the page. Each course has a default picture so the user can easily find the desired course in the list.

The semester list grows horizontally because it can be compacted better than the list of courses, is less likely to be used, and because it is more difficult to scroll horizontally than vertically on most computers. A less-used feature should use the more difficult action, leaving more natural actions for commonly used features.

Orienting the list of semesters horizontally versus the vertical orientation of the courses helps reinforce that the user is navigating through a different set of information than when they simply scroll through courses.

Each course has as information the final grade which was obtained by the student and a check mark if it was a passing grade. If the grade is below 45% the sign changes to a red cross. This can be seen for GenCS where the grade is 35%.

After the student clicks on a course:

Current courses	Previous courses

Previous Courses / Robotics Fall 2014

Robotics

Instructors:

Prof. Dr. Andreas Birk

Credits:

5.00 ECTS

Grading scheme:

| Final Exam (40%, Mandatory) |
| Midterm Exam (30%) |
| Weekly Homework (30%) |

Grade Details:

88%

Final Exam	Midterm Exam	Weekly Homework	Final Grade	Course Status
80%	60%	89%	88%	Pass ✔

Description:

Robotics is a field that spans the entire range from low-level mechatronics and signal processing to high-level cooperation protocols of intelligent agents, and thus touches large portions of both CS and EE. Correspondingly, the course aims at an integrative, practically oriented education that enables students to practically combine methods he/she has encountered in various more specialized courses before. The course is offered biannually (alternating with ``Autonomous Systems'').

The student can see the instructor, how may credits he got for the course, grading scheme and the description. We show in a table the grade components and the score he got for each of them. We use green as background for passes course and red for a course which has been failed.

4.6   TA Landing Page

The TA landing page is mostly the same as the student menu, and contains no major differentiation in the prototype beyond easy links to the grading pages for current courses. In a production environment, the current and previous courses would show any courses for which they have TA permissions.

When the TA enters the grading overview page, they see a list of assignments for that course, along with the current status of the grading for the assignment. Colour coding helps the user to quickly navigate an information-dense page. Aqua indicates that grading is complete and there is no further action required. Yellow indicates that grading is partly finished, and orange indicates that grading has not started. Grey is used to indicate a non-actionable assignment that has received no submissions yet.

4.7   TA Grading Page

Python Lab - Assignment 3 overview

<div>Download all submissions ⬇</div>

Vlad Ungureanu ⬇

Feedback:

| | Grade | 69% | Update |

Nick Lee ⬇

Feedback:

| | Grade | 100% | Update |

Naomi Pentrel ⬇

Feedback:

| | Grade | 78% | Update |

Roxana Nadrag ⬇

Feedback:

| | Grade | | Update |

Each assignment grading page follows the same layout, with the page content being centred to help the user focus. A simple heading communicates what assignment is being graded, two batch jobs can be run from a well-known location (always the top of the page), and the remainder of the page is simply a list of submissions in need of grading.

No default field is initially focused; this is because the TA must first download the submissions. Tabbing through the page with the keyboard will always follow the order "Feedback, Grade, Submit" in order to minimise the amount of steps involved with grade entry, which is important for grading large numbers of small assignments. We placed the comment box before the grade box in order to encourage TAs to provide comments for grades; when tabbing through the page, the comment field will be selected first, which makes it tougher to skip.

All submission boxes are colour-coded, such that a graded assignment is aqua, an ungraded assignment is orange, and an empty submission is grey. Ungraded submissions are not omitted because it is sometimes necessary for assignments to be submitted via alternate methods such as paper or email. While the purpose of a grey field is slightly different from what it was on the overview page (non-actionable), most blank submissions will never receive a grade other than zero, and so they can be considered non-actionable for the most part.

Ample space is provided between assignment entries in order to reduce the chance of visual confusion. In a complete implementation, JavaScript would also be used to turn the currently selected region's background to a light grey in order to further differentiate what's being edited. Avoiding a pure grid layout also helps in creating a more comfortable view for the user.

Different sizes between the comment and grade fields provide further visual differentiation, and the larger size of the feedback box implies its significance. In many programming labs, the feedback is more important than the grade.