Department of Computer Science
Programming, Education, and
Computer-Human Interaction

Live Peer Testing for ML Learners focusing on Visual Data

Oliver Lehmann

Bachelor Thesis

2024

Supervisor: April Wang

ABSTRACT

Live Peer testing is the process of creating and sharing tests with others in real time. Previous research has shown this to be an effective tool in programming education. As part of this thesis, we developed a system that extends this paradigm from previously only text-based data to visual data. This system involves a JupyterLab extension, a backend server and database which allows instructors and students to all join a virtual collaborative learning environment. To provide more information about our project, we set up a dedicated webpage. This webpage not only offers comprehensive information about the project but also includes detailed instructions on how users can self-host the system. Additionally, to ensure the effectiveness and user-friendliness of our design, we conducted a user study. This study allowed us to gather valuable feedback, which will inform future improvements to the system.

ACKNOWLEDGMENT

I want to thank my Professor April Wang, for her help and advice throughout this bachelor thesis project. Her input and feedback were invaluable.

I'm grateful to my friends and family for their emotional support during my research and writing. Their encouragement helped me keep going when things got though.

I also want to acknowledge that I used AI language models [1] to help find better words and improve some sentences in this thesis. These tools were helpful in making my writing clearer.

Lastly the thesis uses a template which was provided to me by the Programming, Education, and Computer-Human Interaction (PEACH) Lab which itself uses contents from the paper "Catchy titles are good: But avoid being cute" [20] from professor Jacob O. Wobbrock, the University of Washington.

1 Introduction

Machine learning is becoming an ever increasingly important part of computer science education. The introductory machine learning course or the Visual Computing course taught at ETH Zurich typically employ traditional teaching methods, such as lectures, assignments, and pre-defined Jupyter Notebooks, to help students grasp the mathematical foundations and basic implementations of ML algorithms. At the same time new, promising, innovative teaching paradigms have come forward. In particular collaborative learning which has been shown to increase students' motivation and engagement with what is being learned [6] which increases the efficacy of learning. Collaborative learning is a method that engages multiple students in shared educational activities. In peer assessment, a form of collaborative learning, this activity is evaluating each other's solutions. The PuzzleMe paper [18] introduced a practical tool for teachers to implement peer assessment in real classrooms. It employed two ways of peer assessment. The first is Live Peer Testing which allows students to easily create and share test cases with their fellow peers which then in turn can test their solutions against these tests. The second is Live Peer Code Review which pairs students with different approaches to the solution together.

However, despite the increasing importance of machine learning in computer science education, traditional ML courses often struggle to fully engage students, particularly when dealing with visual data. While the PuzzleMe approach has shown promise with collaborative learning for text-based programming tasks, it falls short when applied to image-centric ML problems. The visual nature of many ML applications, such as computer vision and image processing, presents unique challenges that are not adequately addressed yet. Expanding the PuzzleMe framework to encompass visual data could potentially enhance the learning experience for students in ML courses, particularly those focused on image processing and computer vision. By adapting these collaborative learning strategies to visual ML tasks, we may observe similar gains in student engagement and understanding as seen with text-based problems.

That's why this thesis introduces a new system based on the PuzzleMe framework, specifically designed to address the unique challenges of collaborative learning in visual Machine Learning education. Since the Peer Code Review part of PuzzleMe isn't meaningfully different for visual data we focused our efforts on Live Peer Testing. To this end, we have developed a Jupyter Notebook extension that incorporates all necessary components for collaborative peer testing in visual ML contexts. This extension provides an intuitive interface that allows students to easily create and share tests for their peers. Key challenges we had to overcome is how to validate the correctness of test cases as it's much less black-and-white in image-based tasks compared to traditional programming problems.

Lastly, we deployed the extension using Docker compose on a server to do a user study. This was done so users don't have to install the extension first on their personal machine and just access a Jupyter instance using their browsers. We collected data from the participants using questionnaires and obserserved them while they use the extension. The study affirmed our design choices but also revealed certain pitfalls which we then improved upon.

The contribution of this thesis include:

(1) Insights into the adaptation of collaborative learning strategies from text-based programming to visual data domains, potentially informing future developments in ML education tools and methodologies.
(2) An intuitive Jupyter Notebook extension that facilitates collaborative peer testing for visual ML problems, enabling students to easily create and share test cases for image-based tasks.

2 Related Work

The PuzzleMe paper [18] serves as the primary influence for our research, as we extend its collaborative learning system to the domain of visual data in machine learning education. To comprehensively understand the efficacy of collaborative learning in this context, we need to consider research focusing on three main aspects, as outlined in the PuzzleMe paper: (1) peer assessment as collaborative learning, (2) real-time code sharing in educational settings, and (3) scaling feedback. In addition it's crucial to look at other research focusing on testing involving visual data.

2.1 Peer assessment as collaborative learning

Peer assessment is a process in which students evaluate and provide feedback on each other's solutions. Research has shown that peers can grade each other fairly [13], which is an important prerequisite if we use it to supplement or partially replace instructor evaluation. Even so, students may struggle to provide high-quality feedback due to their beginner-level understanding of the subject[13, 21, 5]. This can be mitigated by giving them a structured framework of evaluation [21, 5] or grouping them in a sensible manner [13].

There are many different ways one can do peer assessment. Like dividing the class in groups of two to do pair programming which was found to help build programming knowledge and computational thinking [9].

Another popular form of peer assessment involves the use of *peer testing*. In this approach, feedback is provided through the exchange of test cases that meet the specified problem statement. Peer tests were shown to improve students' confidence writing software tests and despite the extra workload students reported that they enjoyed the peer tests [16]. Still, peer testing was not well-suited for in-class exercises due to the time constraints associated with submitting and reviewing test cases. However, PuzzleMe addressed these limitations by introducing Live Peer Testing, which enables real-time sharing of lightweight test cases among students. Dave et al.[6] found that peer assessment given and received while the assignment is in progress yields greater motivation for both the reviewer and reviewee.

2.2 Real-time code sharing in educational settings

Real-time code sharing is the practice of instantly exchanging code between instructors and students or between students collaborating with each other. The former includes live coding where the instructor shows their screen while going through the process of solving a programming exercise. Real-time code sharing between students fosters collaboration and peer learning in programming courses. This practice enhances social awareness - knowing what your peers are up to and it supports engagement with others [4]. While various approaches exist, such as collaborative code editors [11, 2], one method we mainly focus on is the sharing of test cases. This approach allows students to exchange and compare their test cases, helping them identify potential issues in their code and improve their problem-solving skills.

2.3 Scaling Feedback

Giving students feedback that is both constructive and timely is crucial to learning [14]. It allows students to quickly apply suggestions and gain valuable insights, enhancing their understanding and performance. Without any special tools it renders difficult for an instructor to meet these demands especially since class sizes are often very big. Thus we need *scaling feedback*: either automatically generate feedback without instructor input or augment the instructors ability to give feedback to many students effectively [15]. Our extension as well as PuzzleMe make heavy use of *learnersourcing* [12]. This approach leverages the collective efforts of students to create new educational content, serving as both a collaborative project and an effective learning opportunity for the participants themselves. In the case of PuzzleMe as well as this paper the created content are test cases. While creating the test cases users engage deeper with the problem and may think of edge cases they wouldn't have thought of otherwise if just tasked to create a working solution.

2.4 Test cases for visual data

Prior research involving the use of visual data in an educational setting found various benefits [17]. Particularly they found benefits in collaboratively creating the test sets. Working together allowed learners to identify and discuss key issues in data set design, such as data representation, diversity, and class balance. Collaborative creation of visual test cases allows for multiple perspectives to be represented in the data, potentially leading to more diverse and balanced test sets. Additionally, the visual nature of the data makes abstract ML concepts more tangible and easier to grasp for beginners.

3 Design of the Extension

The extension's purpose is to aid programming teachers to teach their students. Students can collaborate by providing test cases for each other. We will go through all important UI Elements.

3.1 Sidebar

The first major design decision was how we should integrate the extension within the Jupyter Notebook user interface. Two approaches were initially considered: *Fullpage view* and *Resizeable Sidebar view*. Below, we outline the advantages and drawbacks of each approach.

3.1.1 Full-page View

The full-page view offers several advantages. It utilizes the entire screen, allowing for larger components and allowing for more information to be displayed simultaneously. This approach provides a more immersive experience for users. However, it also comes with some drawbacks. Users are required to switch between the extension view and code view, which can create mental overhead. If code cells are re-implemented within the full screen view, it could potentially lead to a better user experience, but may feel less like a tradition Jupyter Notebook, which is important for students to gain familiarity with as they are widespread and an important tool to know.

3.1.2 Resizeable Sidebar View

The resizeable sidebar view presents its own set of advantages. It allows for simultaneous viewing of both the extension and code cells, preserving the authentic Jupyter Notebook interface. This is particularly beneficial for students as it allows them to become familiar with this widely used tool. Additionally, it offers easy customization and resizing to accommodate different user preferences. However, this approach is not without its drawbacks. The sidebar view is limited in horizontal space, which means it can't display as much information at once compared to the full-page view. It may also slightly reduce the available space for code cells.

After careful consideration of both approaches, the resizeable sidebar view was ultimately favored for our extension. This decision was primarily driven by our goal to accompany the coding process without overshadowing it, maintaining the authentic Jupyter environment for students to get accustomed to. We believe that despite the limited space, the sidebar still offers sufficient room to display all relevant details for the collaborative aspect of our extension. A user can show and hide the sidebar by clicking on a button.

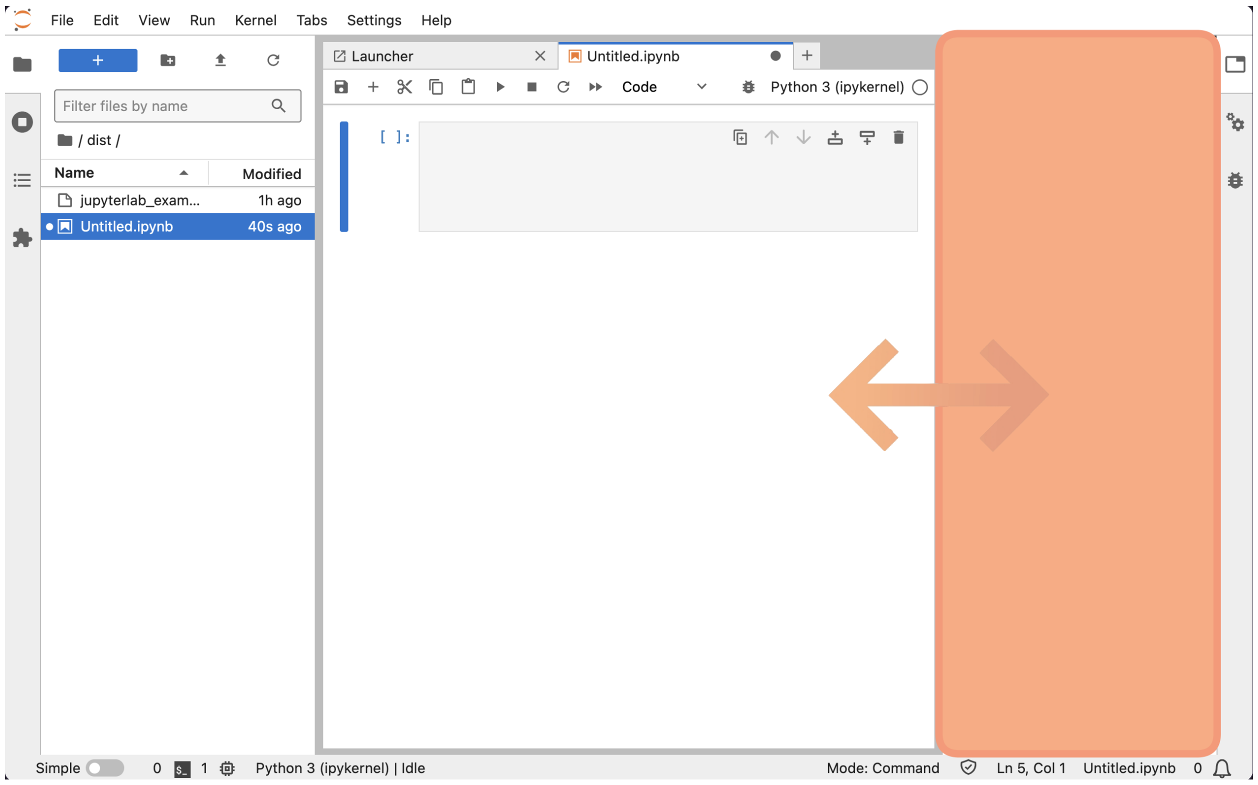


Figure 1: Placement of the resizable sidebar within the Jupyter Notebook user interface

3.2 Login/Register

We settled for a pretty standard Login and Register view. These are the first views presented to users when they open the sidebar. These were required to differentiate between users and to keep track which submissions, test cases or rooms belong to whom. It's also necessary if we want to keep state between sessions (see Figure 2 first and second images).

3.3 Choose Room

This view is shown once a user is logged in (see Figure 2 third image). It allows users to enter a room. They can enter the unique 6-digit Room PIN at the top and click the "Join" button to join a new room. Below they have a scrollable list of all the rooms they have joined in the past already. This allows users to quickly join rooms without having to tediously enter the PIN every time they want to enter a room. At the same time users can create new rooms by clicking the grey "Create New Room" button.

3.4 Task/Create Room

Instructors can create new rooms on the "New Room View" (see Figure 2 fourth image). They can specify a room name and description. Additionally they have the option to upload a Jupyter Notebook file which serves as a starting point for the students to solve the task.

When now a student chooses to enter a room, they are greeted by the Task screen (see Figure 3 first image). It displays the room name and a task description. If the instructor chose to upload a Jupyter Notebook file a download button is shown. This screen also allows students to submit their solution which then is compared against all test cases which other student created. The instructors on the other hand can upload the reference solution which is used to validate test cases, which ensures that they can't be arbitrarily bad or even wrong.

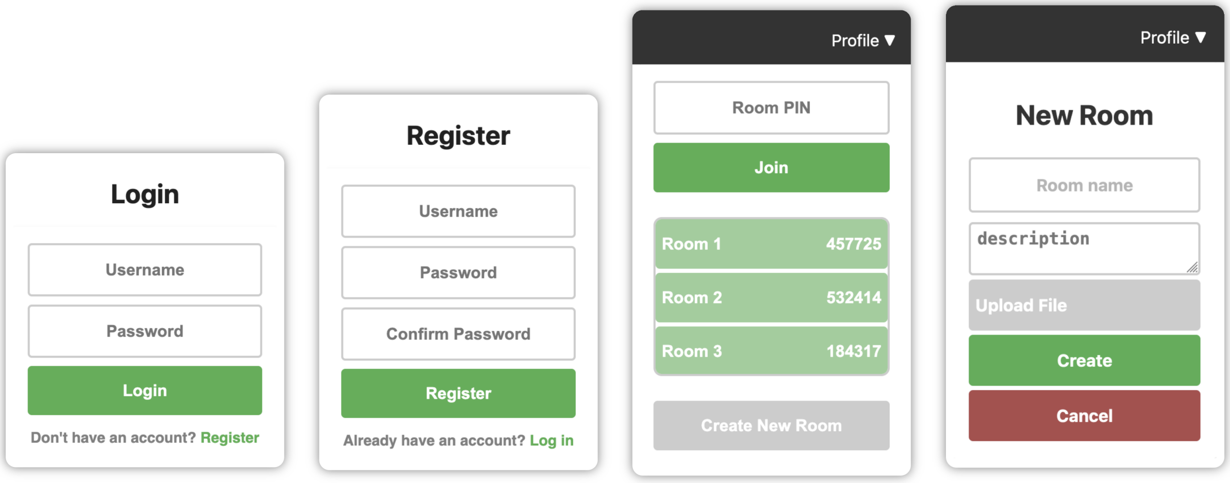


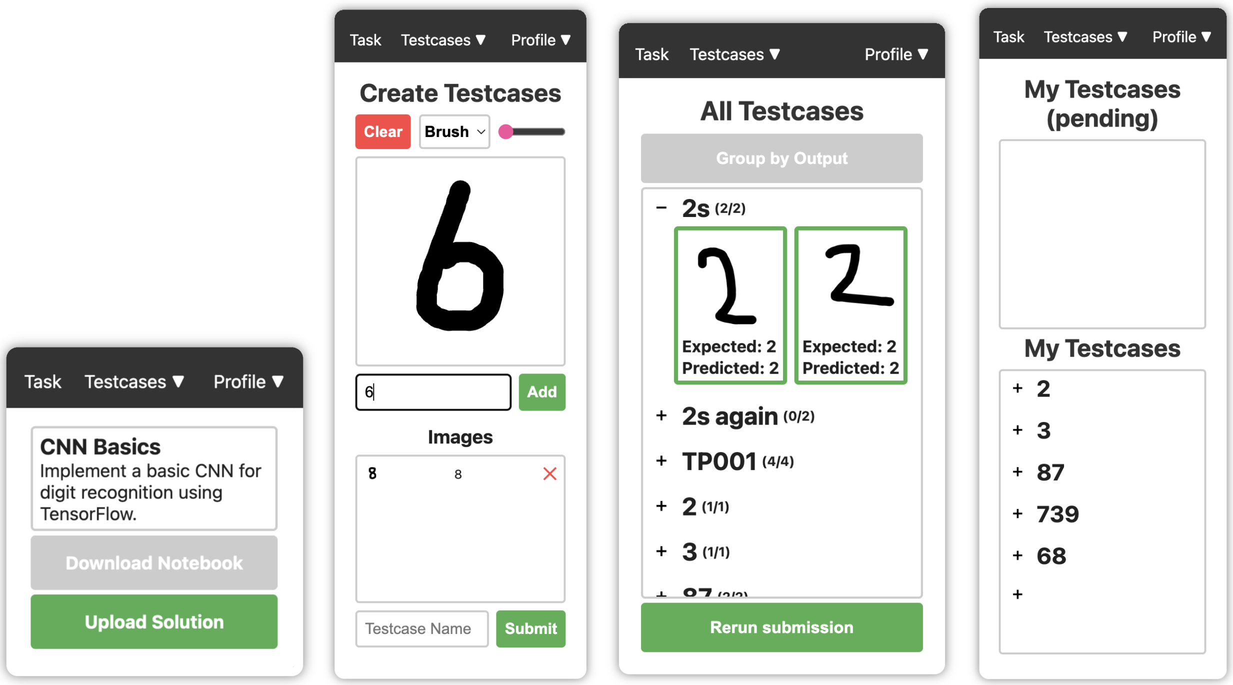
Figure 2: Login, Register and Choose Room Views

3.5 Create Testcases

Students can access this screen by clicking first on the test cases item in the navigation bar on the top of all the views associated with a room and then on the create test case menu item in the newly opened drop-down menu (see Figure 3). This screen contains a canvas that allows users to draw on. They can choose between a brush with variable size and an eraser. In addition, there is a button to clear the whole canvas. Users then give the drawing a label - the expected output of the algorithm to be written in this task if the input were this drawing. It then gets appended to a list at the bottom of the screen (see Figure 3 second image). Once they are happy with the test cases they can give them all a name and submit it. The test set will then be tested against the reference solution submitted by the instructor. If more than 80% of test cases pass the reference solution its automatically admitted. If not it gets turned into a pending test case for review for the instructor. This solves an important problem when going from deterministic code, like in PuzzleMe to probabilistic code present in Machine Learning. There is no black and white correct solution. There are only better and worse solutions.

3.6 View Testcases

This screen is again accessed by first clicking on "Testcases" and then on "All Testcases" in the navigation bar. On this screen all submitted test cases that passed the reference solution or were manually approved by the instructor are shown (see Figure 3 third image). First only the names of the test sets are displayed and one can click on them to show all individual test cases. The number of test cases in one row responsively adapts to the size of the sidebar, so users can effectively look at a large number of test cases at once. The color of the outline around a test case shows if a user passed the test case (green), didn't pass it (red), or the submission hasn't been run against the test case yet (grey). At the bottom is a button to rerun the last submitted solution. This is when new test cases were added since the submission was made. The user also can switch between seeing all the individual test cases as they were made by others or all test cases grouped by the label, by clicking on the grey button at the top of the test cases. In addition to the "All Testcases" screen, there is also a "My Testcases" screen where users can view all the test cases they have personally created, including those that are still pending for approval (see Figure 3 fourth image). Otherwise it functions very similarly to the "All Testcases" screen. For instructors, there is a separate "Pending Testcases" screen that displays all pending test cases submitted by students. This screen allows instructors to review, approve, or modify submitted test cases before they are added to the main test case pool, ensuring that all approved test cases meet the required standards and effectively evaluate student submissions. Again it looks very similar to the two other screens already mentioned that involve showing test cases.



Implement a basic CNN for digit recognition using TensorFlow.

Figure 3: Create Testcases, All Testcases and My Testcases Views

4 Implementation

The whole application consists of three different parts: the JupyterLab frontend, a NodeJS backend and a MongoDB database. The following chapter explores how these parts interact with each other and which architectural decisions were taken.

4.1 JupyterLab Extension

The JupyterLab extension is the interface between the JupyterLab code cells and the backend. It also contains all the frontend UI elements. We are very grateful for the JupyterLab extensions by example repository [8] as it contained starter code concerning the sidebar, styling components using react and typescript as well as manually executing code inside the kernel.

4.1.1 Extracting the solution

In developing our frontend JupyterLab extension, we made a decision to extract the function named *predict()* from the frontend rather than manually extracting individual code cells. This approach was chosen to address the non-linear nature of coding in Jupyter environments, where code is often distributed across multiple cells and may undergo frequent modifications or overwrites during experimentation. By focusing on the *predict()* function, we ensure that we capture the final, intended code along with all its dependencies. By doing this we don't put a restriction of formatting on the user and encourage experimentation and exploration which is often messy. We used the *requestExecute* method to extract the *predict()* function using this piece of python code:

```python
import io
import cloudpickle
import base64

file = io.BytesIO()
cloudpickle.dump(predict,file)
data_bytes = file.getvalue()
function_data = base64.b64encode(data_bytes).decode('utf-8')
function_data
```

We also use the same method to extract the reference solution from the instructor.

4.2 Backend

Our backend uses NodeJS and ExpressJS. NodeJS runs JavaScript on the server, while ExpressJS helps manage routes and connections. The backend serves as an intermediary between the frontend and the database, managing data flow, processing requests, and performing necessary computations. It exposes an API, which provides a structured way for the frontend to interact with the server.

Initially, we explored the use of WebSockets for real-time communication between the client and server. However, we decided to transition to a more traditional HTTP-based API approach. This change was motivated by the desire for a more structured and easily maintainable architecture.

A crucial part of the backend is the submission handling system. When a user submits their code, the server processes it and runs it against the test cases. This process involves a Python script that decodes the submitted function, loads and executes it with the provided test inputs, and returns the results. The backend then processes these results, comparing them with expected outputs and calculating the number of passed tests.

4.3 Database Design

The database design for this project utilizes MongoDB, a NoSQL document-oriented database, hosted on MongoDB Atlas, a cloud database service. The database schema, as illustrated in Figure 4, consists of five main collections: Room, User, TestCase, PendingTestCase, and Submission.

The Room collection serves as the central entity, containing information about the rooms students can join, including room name, pin code, task details, and references to related entities such as the instructor, participants, and test cases. The User collection stores user information, including username and hashed passwords, with a reference to the rooms they are associated with. TestCase and PendingTestCase collections store information about the test cases including input, expected output, and descriptions. The Submission collection keeps track of user submissions, storing references to the room and user, submission timestamp, and test results.

To optimize performance and handle large files efficiently, we implemented GridFS, MongoDB's specification for storing and retrieving large files. GridFS was utilized to store reference solutions and user submissions, which are typically much larger in size than the rest of the data. This approach significantly improved regular database access speeds by keeping the main collections lean and storing large binary data separately.

Room schema

```
{
  _id: <ObjectId1>,
  room_name: "Code Room",
  pin_code: "332515",
  instructor_id: <ObjectId2>,
  task: {
    description: "This task is...",
    hasnotebook: true,
    notebook: <binary data>
  }, // refence solution stored in GridFS
  reference_solution: <ObjectId6>,
  participants: [<ObjectId2>,<ObjectId3>,...],
  test_cases: [<ObjectId4>,...],
  pending_test_cases: [<ObjectId5>,...]
}
```

User schema

```
{
  _id: <ObjectId2>,
  username: "Alice",
  passwordHash: "2b$10$H...",
  rooms: [<ObjectId1>,...]
}
```

TestCase schema

```
{
  _id: <ObjectId4>,
  room_id: <ObjectId1>,
  user_id: <ObjectId2>,
  input: ["input1", "input2"],
  expected_output: ["output1", "output2"],
  description: "Test case description"
}
```

Submission schema

```
{
  _id: <ObjectId7>,
  room_id: <ObjectId1>,
  user_id: <ObjectId2>,
  code: <ObjectId8>, //stored in GridFS
  timestamp: ISODate("2023-05-24T12:34:56Z"),
  results: [
    {
      test_case_id: <ObjectId4>,
      actual_output: ["result1", "result2"],
      passed: 2
    },...
  ]
}
```

PendingTestCase schema

```
{
  _id: <ObjectId5>,
  room_id: <ObjectId1>,
  user_id: <ObjectId2>,
  input: ["input1", "input2"],
  expected_output: ["output1", "output2"],
  reference_output: ["ref1", "ref2"],
  description: "Pending Name"
}
```

Figure 4: Database Schema

4.4 Hosted JupyterLab instance

To conduct the user study, it was essential to provide each participant with access to a Jupyter instance. Requiring participants to install the extension or even Jupyter itself would be too difficult, which lead us to a web-based solution. This approach would allow users to simply visit a website hosting a Jupyter instance with our extension already pre-installed.

We explored several options to achieve this goal. Initially, we considered JupyterLite, which proved insufficient due to its limited feature set. We then investigated JupyterHub, but its setup process presented too many challenges. In the end, we settled on a solution that involved hosting 10 identical JupyterLab instances using Docker containers. We chose Hetzner Cloud [10] for hosting, as their set pricing structure provides predictable costs without surprises.

4.5 JupyterLite

In our initial approach to hosting the Jupyter frontend for the user study, we explored JupyterLite as a potential solution. JupyterLite offers the advantage of static hosting on GitHub Pages [7], which would have allowed us to scale the frontend indefinitely without incurring additional costs. This approach would have significantly simplified our infrastructure, leaving only the backend and database to be hosted separately.

Unfortunately we discovered a critical limitation with this approach. JupyterLite doesn't support important machine learning libraries like TensorFlow or PyTorch. The lack of support is primarily due to JupyterLite's reliance on Pyodide, a Python distribution that runs entirely in the browser. Pyodide does currently not provide support for these libraries as it would be too much work for the project.

Nevertheless, JupyterLite still proved useful as we utilized it to deploy a static demo version of our extension. This demo, while not connected to any backend or databases, serves as an effective showcase of our work, allowing interested parties to quickly experience the look and feel of our extension. It enables users to interact with all the components within the familiar Jupyter environment, demonstrating the extension's capabilities without the costly backend setup.

5 User study

As part of this bachelor thesis, a user study was conducted to evaluate the effectiveness, usability, and reception of the developed Jupyter Notebook extension in a real-world setting. The primary objectives of this study were to:

- Assess the overall user experience and acceptance of the extension
- Evaluate the usability of the extension across users with varying levels of familiarity with Jupyter Notebooks
- Gather valuable feedback and suggestions for potential improvements

Conducting such a study is very important to discover blind spots of the developer since they are an expert user of their own extension and know exactly how everything works. This familiarity can often lead to overlooking potential usability issues or areas of confusion that novice users might encounter.

Recruiting participants with varying levels of familiarity with Jupyter Notebooks is crucial. This approach allows for a more comprehensive evaluation of the extension's usability across different skill levels. It's particularly relevant given that many students, who are potential users of this extension, often have limited experience with Jupyter Notebooks themselves. By including participants with diverse backgrounds and experience levels, the study aims to gather more varied feedback and ensure that the extension is accessible and beneficial to a wide range of users.

5.1 Study setting

There was no unified hardware used as the study was conducted on the participants personal laptops. They all had access to a dedicated JupterLab instance hosted on the web with the extension preinstalled. The participants were all gathered at the same place and time to be as faithful to a real student teacher setting as possible. To make the study more reproducible and less dependent on person conducting the study, a study protocol was created.

5.2 Procedure

The selection of participants was based on convenience sampling[19]. More information on the participant distribution can be found in section 5.3

First the participants were informed about the purpose of the study and that they would be testing the Jupyter Notebook extension. Then they were asked to fill in a questionnaire (see Appendix A) to anonymously collect *participant data* (age, gender, and how often they use Jupyter Notebooks). Then the users were given a task sheet (see Appendix B) and prompted to complete the task. They were encouraged to think aloud and mention problems they face. The study conductor observed and took notes on the participant's interactions, comments, and any issues encountered. After the task participants were given a System Usability Scale (SUS) questionnaire [3] and asked to fill it out. Additionally, they received another questionnaire with questions to collect more qualitative data (see Appendix C)

5.3 Participant distribution

The five subjects taking part in the study represented a narrow demographic in terms of age but showed diversity in other aspects. All participants were relatively young with minimal age variation. Their collective average age was 23.0 years (median: 22 years, SD: 1.789 years). The gender distribution was balanced: 3/5 participants identified themselves as female, 2/5 as male, and none selected the option to specify a different gender identity. There was also great variation in their use of Jupyter Notebooks: Only one person uses them daily, another person uses them often while the remaining 3 people never used them before.

6 Results

6.1 Qualitative data

Participants appreciated the cohesive color scheme and intuitive button design, featuring green for submissions and affirmative actions, while red was consistently used for cancellations and negative responses. The process of adding new test cases also received positive feedback, with users particularly praising the drawing functionality. One participant also appreciated the solution submission process.

Some participants encountered challenges when running code within the Jupyter Notebook environment. This difficulty, which is inherent to Jupyter Notebooks rather than the extension itself, may be attributed to varying levels of familiarity with the platform. These observations suggest that future user studies or classroom implementations should allocate additional time to familiarize students with the Jupyter environment, ensuring a smoother experience for all users.

During the testing phase, users encountered a few bugs and inconsistencies in the extension. For instance, one menu item labeled "update profile" led to a blank page, as the feature was not yet implemented. Some confusion arose during the test case creation process regarding the term "label." Several users initially misinterpreted it as a field for naming their drawing, rather than specifying the correct classification. To address this, one participant suggested replacing "label" with "result" for clarity. Another issue noted was the ability to submit a test case without specifying a label or naming the entire test set, which is not desirable behavior. Additionally, one user experienced a technical glitch where the digital pen remained in drawing mode even after releasing the mouse button.

One participant suggested using more symbols and pictograms instead of just text for buttons. There was also a request for additional labels, such as for the slider that adjusts the drawing pen size. Some confusion arose with the predict function; a user expected it to process one image and output one result, rather than handling multiple images and producing an array of results. This indicates a need for better information for students about extension behavior. Several pieces of feedback were specific to the task used in testing and less relevant to the extension itself.

These included suggestions to make the task more interesting and a request for an indicator showing the center of the drawing pad, which was important for the particular task.

To enhance the collaborative aspects of the extension, several suggestions were made. One proposal was to implement a voluntary code-sharing feature, allowing users to publish their submitted code for others to view. Another suggestion was to show who created each test case. Someone mentioned that a brief introduction or user guide would be helpful. Finally, one participant proposed implementing additional methods for code testing beyond peer testing.

In a further interview, when asked about the decision to place the extension in a sidebar rather than a full-page view, one participant expressed a preference for this approach. They explained that it helps students become familiar with the native Jupyter interface. The participant argued that a full-page layout would essentially be recreating Code Expert on the Jupyter backbone.

6.2 System Usability Scale (SUS)

The detailed answers given in the SUS questionnaire can be viewed in Figure 5. We see that the only overwhelmingly negative rated statement was S4, which can be attributed to some users not having much experience with the Jupyter Notebook interface and them feeling a bit lost in the result submission. S3, S5, S6, S8, and S9 were rated only positively or neutrally, which shows that all the participants, regardless of skill level, enjoyed using the system once they got accustomed to it. The average SUS score across all participants is 61 and a standard deviation of 9.7, which indicates some variation within the data. If we group the people by using Jupyter Notebooks daily or often and never we get the following values. The former have an average SUS score of 67.5 with standard deviation of 5.0 while the latter have a SUS score of 56.67 with a standard deviation of 9.65.

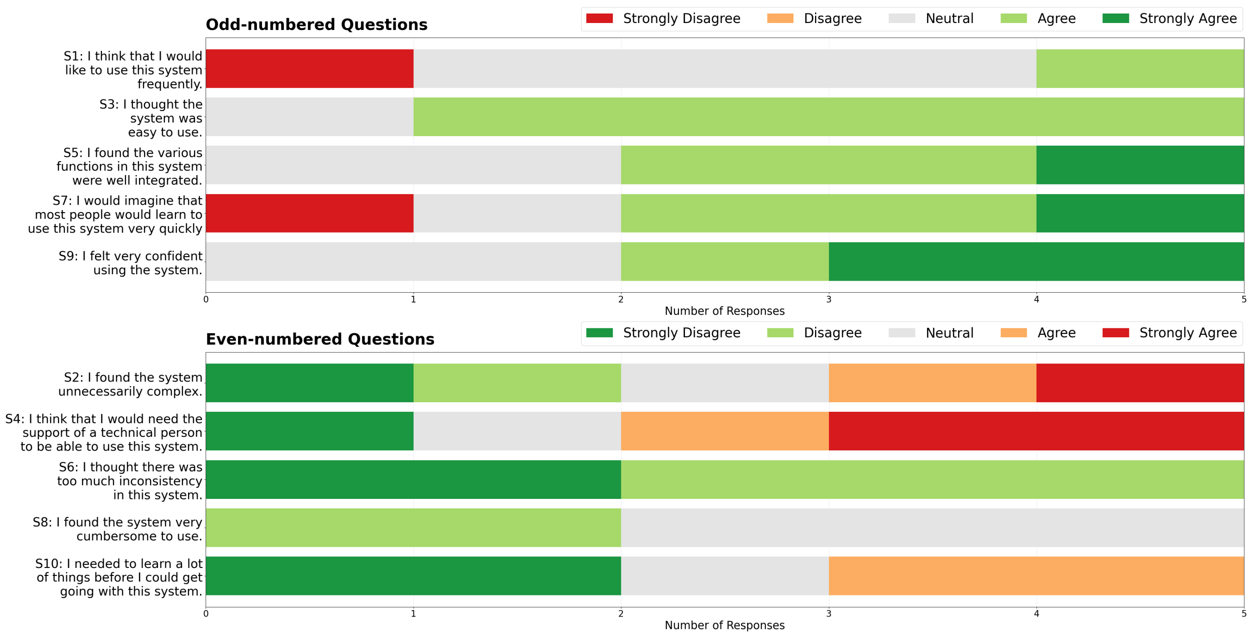


Figure 5: Chart showing the system Usability Scale

7 Discussion

The System Usability Scale (SUS) results provide a quantitative measure of the extension's usability. The average score of 61, while not exceptional, is very promising. The notable difference in scores between frequent and infrequent Jupyter Notebook users (67.5 vs. 56.67) further emphasizes the need for familiarization with the environment. This disparity suggests that with proper introduction and practice, the extension's usability could significantly improve for all users.

Important to note is that users overwhelmingly reported that they liked to create the test cases using the extension. An especially encouraging finding is that many users didn't just create one test per test set but went beyond the minimum requirement and created multiple images per test set. This enthusiasm for test case creation is a significant indicator of the extension's success. It demonstrates that users found the process engaging and enjoyable rather than viewing it as a tedious task or chore. This level of engagement is crucial for effective learning and peer testing, as it encourages students to create more diverse and comprehensive test sets. This enjoyment in creating test cases aligns perfectly with the goal of adapting collaborative learning strategies to visual data.

Also, as a result of the user testing some modifications to the extension has been made. The update profile button has been removed as it is an unimportant feature anyway. Also, we replaced the name "label" with "result" on the create test cases view because it caused confusion for multiple people. Also it shouldn't be possible to create test sets that don't have a name or test cases without a result/label. This has also been addressed.

7.1 Limitations

The user study has several limitations that should be considered. The sample size was relatively small, which limits the generalizability of the findings. Additionally, the study was conducted in a controlled environment, which may not fully reflect real-world classroom dynamics. The participants' varying levels of experience with Jupyter Notebooks may have influenced their perceptions and performance, potentially skewing the results. Future studies should aim for a larger, more diverse sample and consider testing in actual classroom settings to provide better insights into its usability.

8 Future Work

8.1 More Peer Assessment Methods

Incorporate a wider range of peer assessment techniques, drawing inspiration from systems like PuzzleMe. Including Live Peer Code review which pairs students intelligently to maximize the learning effect when they discuss their approaches with each other. Also incorporate exercise types present in the PuzzleMe paper, other than coding, namely multiple choice and free response questions. With the inclusion of more exercise types also the concept of the room must be rethought as currently a room is dedicated to one coding exercise only. But it could contain a multitude of related coding exercises, multiple choice questions and free response questions.

8.2 Larger-scale Study

Conduct a more extensive study involving a larger and more diverse group of participants. This could include students taking introductory programming courses, more experienced users, and instructors. Such a broad study would provide a richer understanding of how the extension performs across different skill levels and use cases. In contrast to our initial study, which included three novices unfamiliar with Jupyter Notebooks, one highly proficient user with teaching experience, and one regular user, this expanded study would offer a more comprehensive view of the tool's efficacy. To address the challenges faced by novice users in our original study, we propose incorporating a dedicated introduction to Jupyter Notebooks for beginners before they engage with the extension. This preparatory step would ensure that all participants have a baseline understanding of the Jupyter environment, allowing them to focus more effectively on the extension's features during testing.

8.3 Integration of AI

Explore the integration of Large Language Models (LLMs) and other AI technologies to enhance both the testing process and overall learning experience. The integration could significantly benefit smaller classes where peer resources might be limited. Leveraging AI to generate test cases ensures that students encounter a wide range of test cases they might not otherwise encounter in a smaller class setting. The AI could also offer automated feedback on code, providing students with immediate insights into their work. This rapid feedback loop, much quicker than waiting for peer or instructor reviews, allows students to iterate and improve their code more efficiently. When students find themselves stuck, the AI could offer targeted hints or explanations, reducing the dependency on instructor availability which is especially useful for large classes.

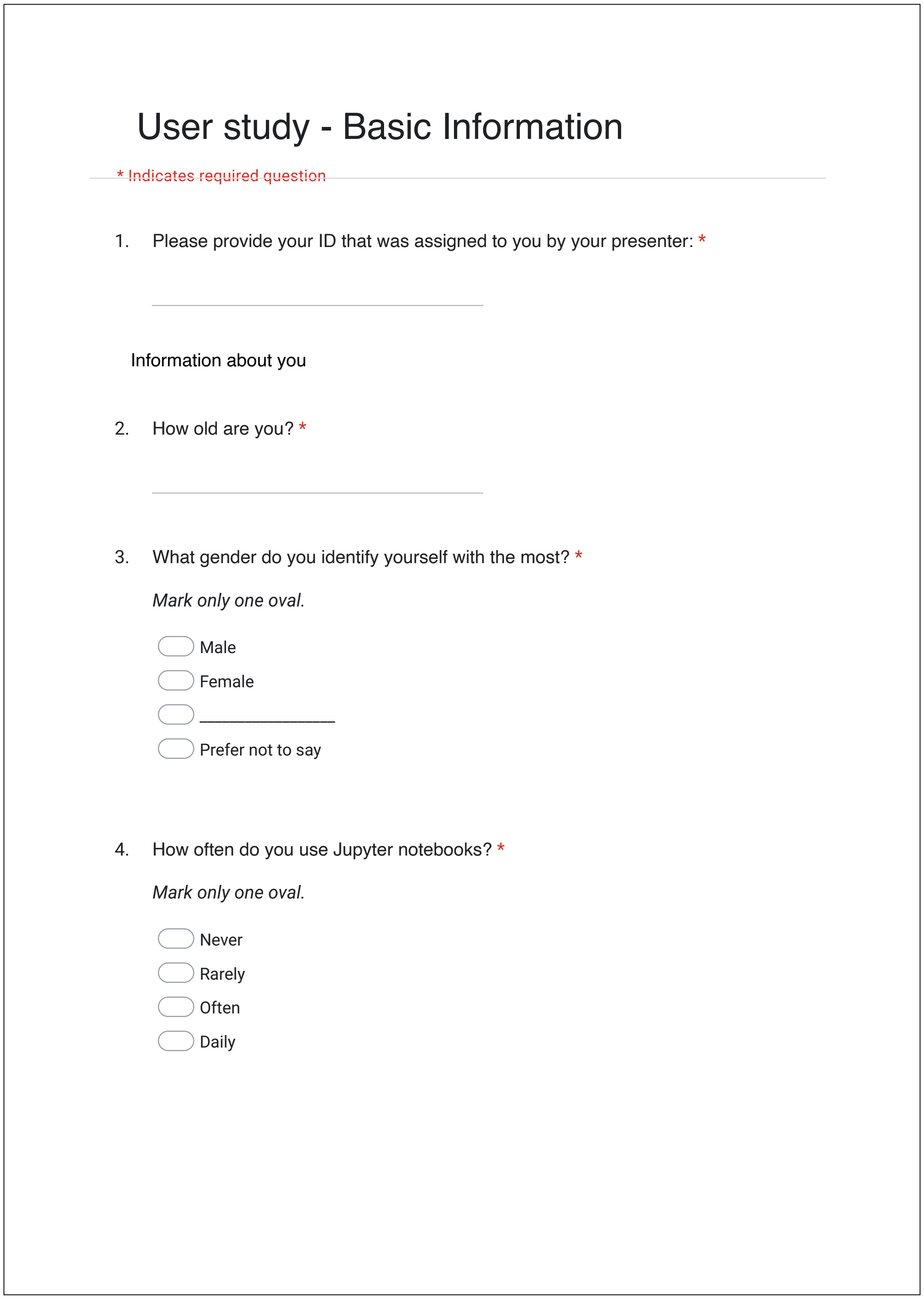
8.4 Allow to create tests with other forms of data

Expand the capabilities of the extension to support a wider range of data types in test creation. Currently, the focus is primarily on visual data. Incorporating support for audio and video data would significantly broaden the scope of testable scenarios. This expansion could be particularly beneficial for courses dealing with multimedia processing, computer vision, or audio analysis. Additionally, enhance the drawing tools within the extension to allow for more sophisticated visual input methods. This could include features like drag and drop local images, shape creation, and annotation tools.

9 Conclusion

This thesis introduces a Jupyter Notebook extension designed to facilitate collaborative peer testing for visual machine learning problems. Our work bridges the gap between text-based programming education and visual data in ML, providing valuable insights into adapting collaborative learning strategies for this domain. The extension's intuitive design and engaging test case creation process were well-received by users, demonstrating its potential to enhance ML education. While our study revealed areas for improvement, such as the need for better Jupyter Notebook familiarization, the overall positive reception suggests a strong foundation for future work. With the source code openly available on GitLab, anyone can use our groundwork to further develop, contribute to, or adapt the extension. Potentially informing future developments of educational tools in ML using visual data.

A Participant Questionnaire



User study - Basic Information

* Indicates required question

1. Please provide your ID that was assigned to you by your presenter: *

Information about you

2. How old are you? *

3. What gender do you identify yourself with the most? *

   Mark only one oval.

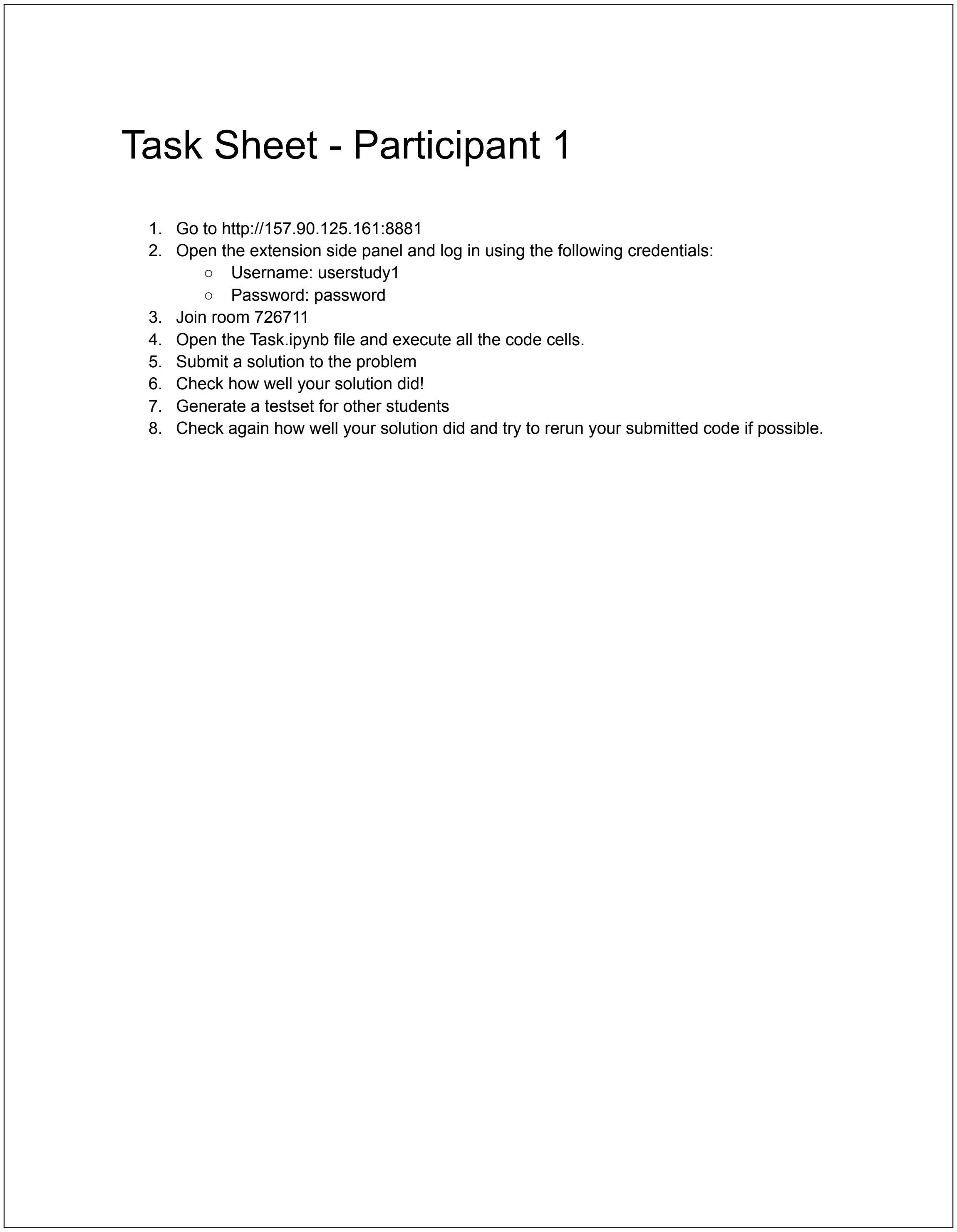
   - Male
   - Female
   - ____________
   - Prefer not to say

4. How often do you use Jupyter notebooks? *

   Mark only one oval.

   - Never
   - Rarely
   - Often
   - Daily

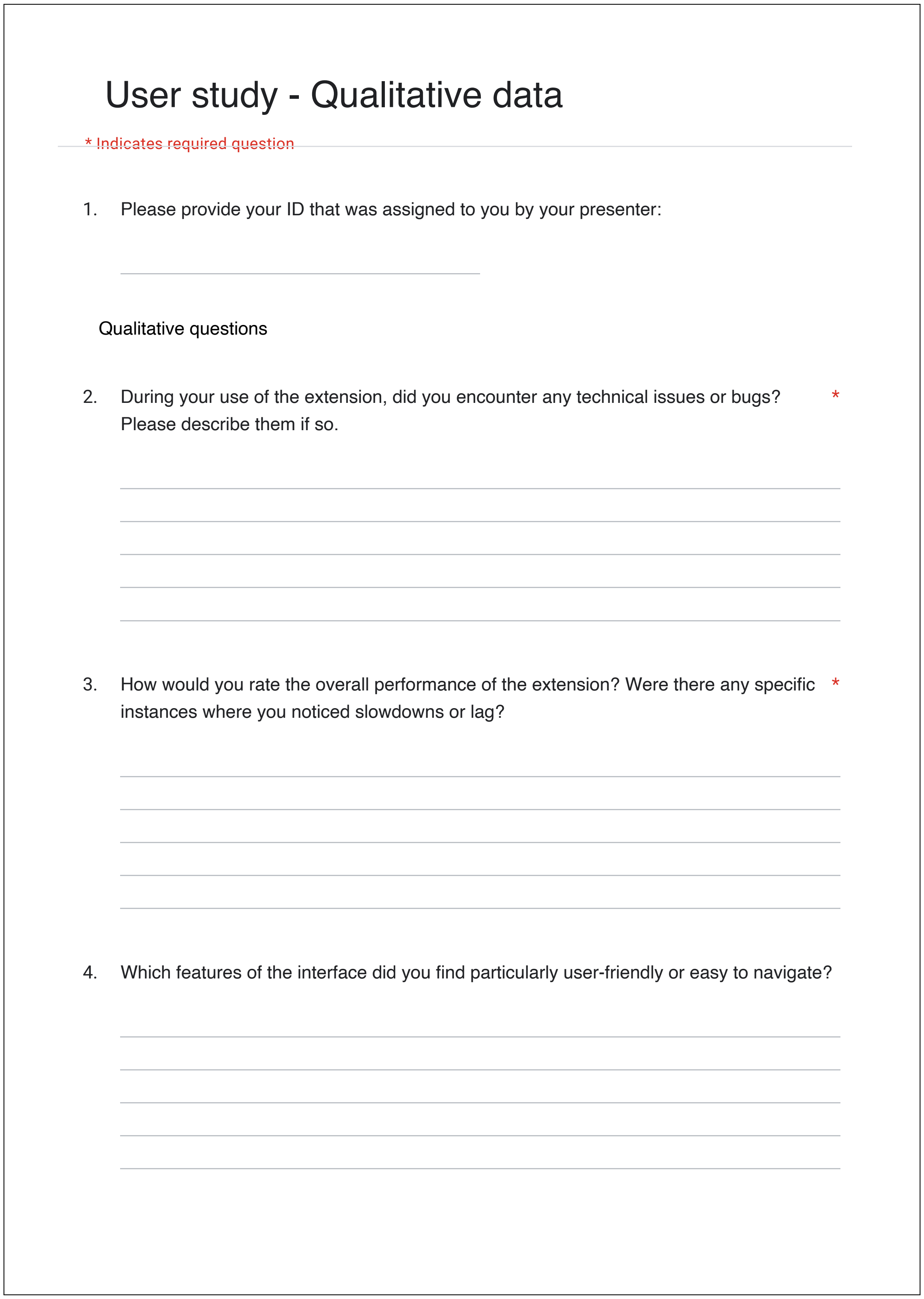
B Task Sheet



Task Sheet - Participant 1

1. Go to http://157.90.125.161:8881
2. Open the extension side panel and log in using the following credentials:
   - Username: userstudy1
   - Password: password
3. Join room 726711
4. Open the Task.ipynb file and execute all the code cells.
5. Submit a solution to the problem
6. Check how well your solution did!
7. Generate a testset for other students
8. Check again how well your solution did and try to rerun your submitted code if possible.

C Qualitative data Questionnaire



User study - Qualitative data

* Indicates required question

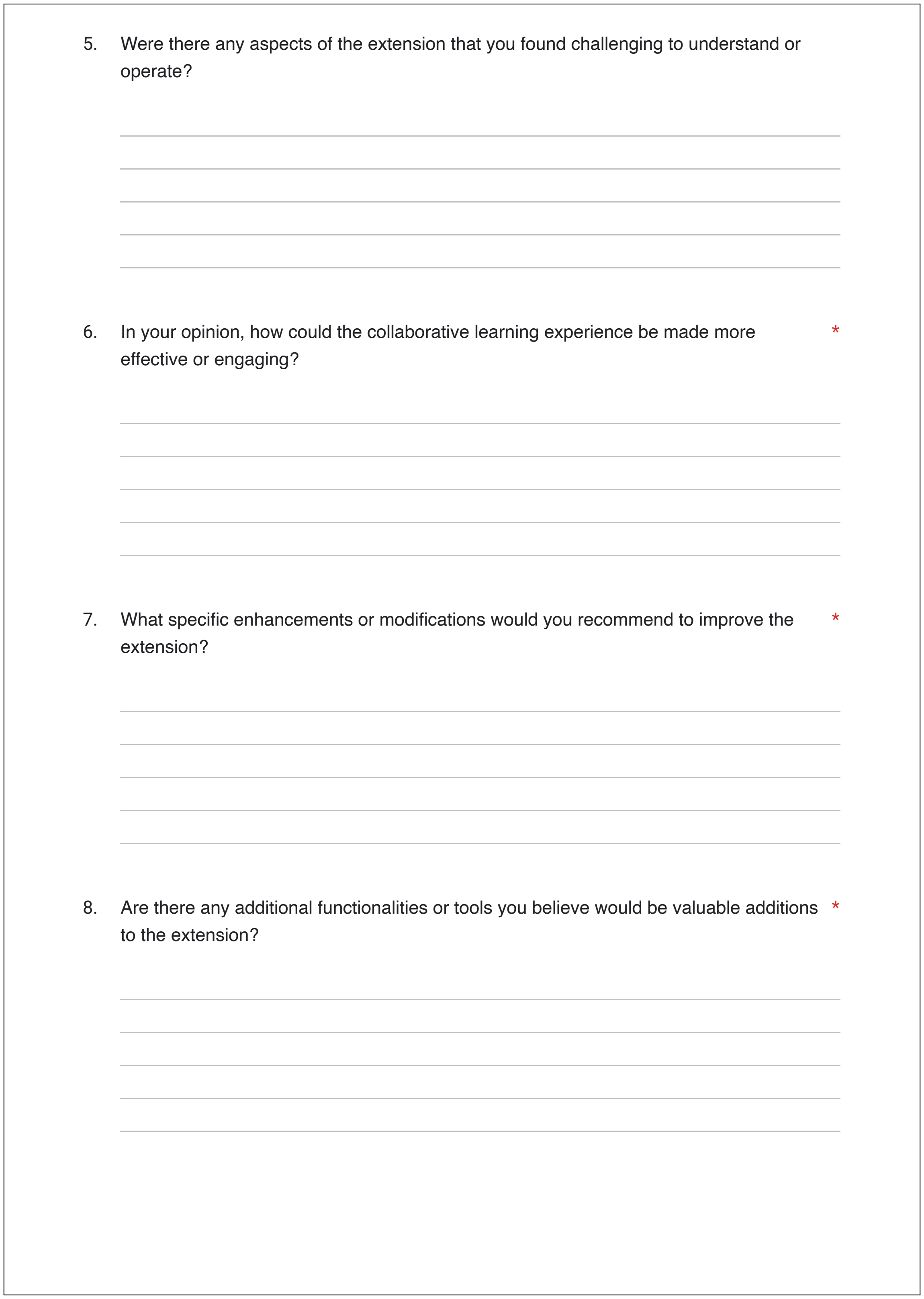
1. Please provide your ID that was assigned to you by your presenter:

Qualitative questions

2. During your use of the extension, did you encounter any technical issues or bugs? Please describe them if so. *

3. How would you rate the overall performance of the extension? Were there any specific instances where you noticed slowdowns or lag? *

4. Which features of the interface did you find particularly user-friendly or easy to navigate?



5. Were there any aspects of the extension that you found challenging to understand or operate?

6. In your opinion, how could the collaborative learning experience be made more effective or engaging? *

7. What specific enhancements or modifications would you recommend to improve the extension? *

8. Are there any additional functionalities or tools you believe would be valuable additions to the extension? *

References

[1] Anthropic. 2024. Claude 3.5 Sonnet. https://www.anthropic.com/.

[2] Marcel Borowski, Johannes Zagermann, Clemens N Klokmose, Harald Reiterer, and Roman Rädle. 2020. Exploring the benefits and barriers of using computational notebooks for collaborative programming assignments. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 468–474.

[3] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[4] Jeongmin Byun, Jungkook Park, and Alice Oh. 2020. Cocode: Co-learner screen sharing for social translucence in online programming courses. In *Extended abstracts of the 2020 CHI conference on human factors in computing systems*. 1–4.

[5] Julia Cambre, Scott Klemmer, and Chinmay Kulkarni. 2018. Juxtapeer: Comparative peer review yields higher quality feedback and promotes deeper reflection. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[6] Dave Clarke, Tony Clear, Kathi Fisler, Matthias Hauswirth, Shriram Krishnamurthi, Joe Gibbs Politz, Ville Tirronen, and Tobias Wrigstad. 2014. In-flow peer review. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*. 59–79.

[7] JupyterLite Contributors. 2024. JupyterLite Deployment. https://jupyterlite.readthedocs.io/en/latest/quickstart/deploy.html. [Online; accessed September-2024].

[8] Project Jupyter Contributors. 2024. JupyterLab Extensions by Examples. https://github.com/jupyterlab/extension-examples.

[9] Jill Denner, Linda Werner, Shannon Campe, and Eloy Ortiz. 2014. Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education* 46, 3 (2014), 277–296.

[10] Hetzner Online GmbH. 2024. Hetzner Cloud. https://www.hetzner.com/cloud/.

[11] Philip J Guo, Jeffery White, and Renan Zanelatto. 2015. Codechella: Multi-user program visualizations for real-time tutoring and collaborative learning. In *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 79–87.

[12] Juho Kim et al. 2015. *Learnersourcing: improving learning with collective learner activity.* Ph. D. Dissertation. Massachusetts Institute of Technology.

[13] Chinmay Kulkarni, Koh Pang Wei, Huy Le, Daniel Chia, Kathryn Papadopoulos, Justin Cheng, Daphne Koller, and Scott R Klemmer. 2013. Peer and self assessment in massive online classes. *ACM Transactions on Computer-Human Interaction (TOCHI)* 20, 6 (2013), 1–31.

[14] Julie G Nyquist and Rima Jubran. 2012. How learning works: Seven research-based principles for smart teaching. *The Journal of Chiropractic Education* 26, 2 (2012), 192–193.

[15] Rishabh Singh, Sumit Gulwani, and Armando Solar-Lezama. 2013. Automated feedback generation for introductory programming assignments. In *Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation*. 15–26.

[16] Joanna Smith, Joe Tessler, Elliot Kramer, and Calvin Lin. 2012. Using peer review to teach software testing. In *Proceedings of the ninth annual international conference on International computing education research*. 93–98.

[17] Tiffany Tseng, Jennifer King Chen, Mona Abdelrahman, Mary Beth Kery, Fred Hohman, Adriana Hilliard, and R Benjamin Shapiro. 2023. Collaborative Machine Learning Model Building with Families Using Co-ML. In *Proceedings of the 22nd Annual ACM Interaction Design and Children Conference*. 40–51.

[18] April Yi Wang, Yan Chen, John Joon Young Chung, Christopher Brooks, and Steve Oney. 2021. Puzzleme: Leveraging peer assessment for in-class programming exercises. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–24.

[19] Wikipedia. 2024. Convenience sampling — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Convenience%20sampling&oldid=1221977149. [Online; accessed August-2024].

[20] Jacob O Wobbrock. 2015. Catchy titles are good: But avoid being cute.

[21] Alvin Yuan, Kurt Luther, Markus Krause, Sophie Isabel Vennix, Steven P Dow, and Bjorn Hartmann. 2016. Almost an expert: The effects of rubrics and expertise on perceived value of crowdsourced design critiques. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. 1005–1017.