A Template Course for Teaching the Development of Interactive Systems to Students of Human-Computer Interaction[⋆]

Toni Schumacher[1][0009−0001−1459−1206], Maged Mortaga[1][0009−0002−3683−3621], and André Calero Valdez[1][0000−0002−6214−1461]

Institute of Multimedia and Interactive Systeme, University of Lübeck, Ratzeburger Allee 160, 23562 Lübeck, Germany
{t.schumacher,maged.mortaga,andre.calerovaldez}@uni-luebeck.de
https://www.imis.uni-luebeck.de

Abstract. Human-Computer Interaction (HCI) is an essential skill for the future. However, previous observations have revealed a discrepancy between the programming training provided to HCI students and the skillset required in the HCI field. This article describes the development and implementation of a course tailored for 3rd-semester bachelor students of HCI to provide them with practical skills in programming interactive systems. The course, named *Interactive Systems*, spans two semesters and includes a combination of lectures and programming exercises, designed to meet the specific needs of HCI students. This module aims to bridge the gap between general computer science programming courses and the specialized requirements of HCI students. We present the concept, realization, and evaluation of this module.

Keywords: Human-computer interaction · Education · Educational Resources · Computer Science Didactics

1 Introduction

Human-computer interaction (HCI) is an interdisciplinary field that blends principles from computer science, design, psychology, and cognitive science to understand and improve the interaction between humans and computers. As technology continues to advance, the role of HCI professionals has become increasingly critical in creating user-centered designs that enhance user experience and accessibility. Despite the growing importance of HCI, many university programs do not provide specialized training that addresses the unique requirements of HCI students. Instead, these students are often directed to traditional computer science courses, which may not fully cater to their educational needs.

[⋆] This preprint has not undergone peer review or any post-submission improvements or corrections. The Version of Record of this contribution is published in Lecture Notes in Computer Science (LNCS,volume 15374), and is available online at https://doi.org/10.1007/978-3-031-76803-3_12

A significant challenge for HCI students is the difficulty in focusing on the core competencies of HCI, especially when it comes to implementation-related tasks. Traditional computer science courses often emphasize theoretical concepts and programming skills without sufficiently covering HCI's practical, design-oriented aspects. This gap highlights the need to promote knowledge of implementation techniques through interactive, individualized approaches that include practical exercises and reflective activities.

To address these challenges, HCI education should incorporate more hands-on projects that simulate real-world scenarios, allowing students to apply their knowledge in practical settings. Collaborative projects can also be beneficial, as they mimic the multidisciplinary nature of professional HCI work, requiring students to integrate insights from psychology, design, and computer science.

Contribution This article describes our efforts in developing a course for students of human-computer interaction to help them gain practical skills in programming interactive systems. The course, named *Interactive Systems* and held at the University of Lübeck, spans two semesters and includes lectures and exercises, requiring students to participate over a full academic year. Although the content of each semester is relatively independent, they are evaluated as a single joint module.

2 Motivation

The primary motivation for developing a specialized course for HCI students stems from the recognition that the skills and knowledge required in HCI significantly differ from those emphasized in conventional computer science curricula. Several key factors presented underscore the necessity of a tailored educational approach for HCI students.

2.1 Interdisciplinary Nature of HCI

HCI is inherently interdisciplinary, requiring knowledge from multiple domains, including cognitive psychology, ergonomics, design principles, and social sciences, in addition to technical skills in programming and software development. Traditional computer science courses predominantly focus on algorithmic thinking, data structures, and systems programming, which, while essential, do not cover the breadth of topics needed for a comprehensive HCI education. HCI students need to understand how to design and evaluate user interfaces, conduct usability testing, and apply human-centered design principles—all of which are not typically covered in depth in standard computer science programs.

2.2 Focus on User-Centered Design

Unlike students of traditional computer science who often focus on system performance and efficiency, HCI students prioritize user experience and usability.

This user-centered approach requires different skills and methods, such as user research, prototyping, and iterative design processes. Standard programming courses may not address these aspects, leaving HCI students without the necessary tools to design effective and intuitive interactive systems. A specialized course can bridge this gap by integrating user-centered design practices with technical instruction.

2.3 Emerging Technologies and Frameworks

The landscape of interactive systems is rapidly evolving, with new technologies and frameworks continuously emerging. HCI students must stay ahead of the latest developments in web technologies, mobile applications, virtual and augmented reality, and ubiquitous computing. Traditional computer science curricula may not be agile enough to incorporate these rapidly changing technologies into their coursework. A dedicated HCI course can provide timely and relevant instruction on the latest tools and frameworks, ensuring that students are well-prepared for the current job market.

2.4 Focus on Web Development

Web development is integral to HCI education, providing a versatile foundation for creating user interfaces and experiences. Mastering web development equips students with essential skills for designing interactive and responsive interfaces and extends their capabilities beyond web applications. Modern frameworks like React Native and Electron allow HCI professionals to use web development principles to build native mobile apps and desktop applications, broadening their technical expertise and application scope.

2.5 Practical and Applied Learning

HCI education benefits greatly from a hands-on, applied learning approach. Students must engage in practical projects that allow them to apply theoretical knowledge to real-world problems. Standard computer science courses often emphasize theoretical concepts and abstract problem-solving, which, while valuable, do not always translate to the practical skills needed for HCI. A specialized course can focus on project-based learning, where students develop interactive systems, conduct usability studies, and iterate on their designs based on user feedback.

2.6 Collaboration and Teamwork

HCI projects often involve interdisciplinary teams, requiring strong collaboration and communication skills. Students must learn to work effectively with designers, psychologists, and other stakeholders. Traditional computer science courses may not emphasize these soft skills to the same extent. By incorporating collaborative projects and team-based assignments, a dedicated HCI course can better prepare students for the collaborative nature of the field.

2.7 Requirements

Given these considerations, it is clear that HCI students require a distinct educational path that addresses their unique needs and prepares them for the challenges of designing user-centered interactive systems. The *Interactive Systems* module is designed to fill this gap, providing HCI students with a comprehensive and practical learning experience that combines technical skills with a strong emphasis on user experience and usability. This tailored approach not only enhances the educational outcomes for HCI students but also ensures that they are well-equipped to contribute to the advancement of human-computer interaction in various professional contexts.

3 Didactic Approach

To help students attain advanced theoretical and practical competencies in the development of interactive systems, we chose didactic approaches that facilitate learning of both theoretical groundwork and practical skills in software development.

As a core guiding principle, we apply constructivist paradigms [9] that align with self-determination theory [4]. We try to pick the best fitting approach to the individual learning pieces and a wide range of teaching tools, ranging from classic lectures (passive learning) to methods such as problem-based learning [1,11] that have been shown to be effective for teaching applicable knowledge [5].

HCI education benefits greatly from a hands-on, applied learning approach. Students must engage in practical projects that allow them to apply theoretical knowledge to real-world problems. Kolb's experiential learning theory [7] states that knowledge is created through experience and reflection. By incorporating hands-on projects and real-world applications, our course allows students to engage deeply with select frameworks and ensures that they will be capable of creating an interactive system themselves after the course.

3.1 Individualized Learning and Motivation

One of the common challenges we encounter is the varying levels of prior education in software engineering and programming among students. Therefore, it is crucial to address these differences by providing individualized materials and helping students identify which skills and theories they have mastered and which ones they still need to work on.

To facilitate learning, we use a combination of didactic approaches. Our foundation is based on self-determination theory [4], which posits that task satisfaction is rooted in three innate psychological needs: autonomy, competence, and relatedness. We design our teaching methods around these principles. We encourage *autonomy* by allowing students to choose lessons and tasks they are comfortable with. We gradually increase the difficulty of problems to build *competence* over time. Finally, we let students select their challenges to foster a sense of *relatedness* in our approach.

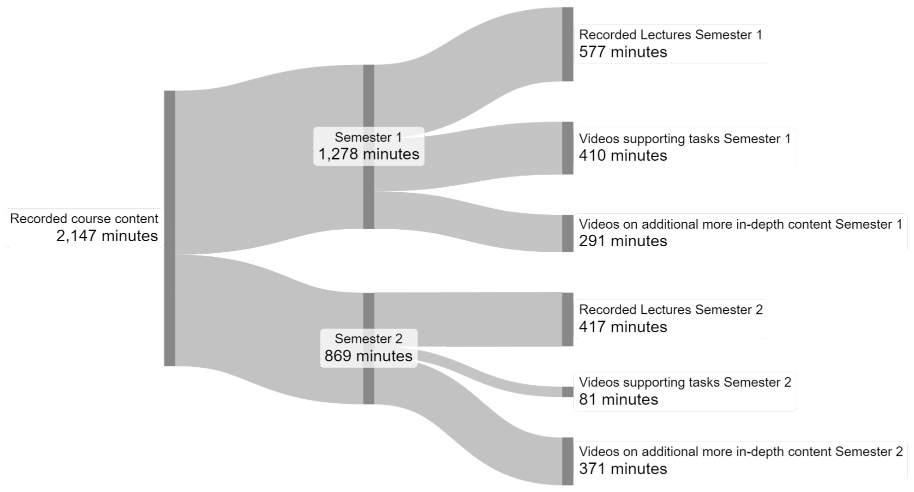


Fig. 1. Division of the recorded course content for each semester into recorded lectures, videos supporting the practical exercises and videos on additional more in-depth content. Information in total minutes of videos

3.2 Lectures and Flipped Classroom

We achieve this by employing a variety of methods that are uniquely combined for our system. Specifically, lectures focused on theory are conducted in person, facilitating peer discussions and reflections on different perspectives during the lecture. This approach is particularly effective when the content requires reflection to deepen understanding, even if it is not inherently difficult.

For practical learning, such as programming skills, we utilize flipped classrooms [2]. A flipped classroom means that the static instructional material usually received by the students in a passive learning environment is pre-recorded as learning videos that are available on a learning platform. Students have the flexibility to access these videos at their convenience, whether at home, during their commute, or any other preferred location, and engage in exercises alongside the video content. An overview of the recorded course content from the winter semester of 2023 and the summer semester of 2024 can be found in Figure 1. In this model, students can learn at their own pace, allowing them to adjust the videos' speed to match their skill levels. This flexibility also enables students to revisit material they might have missed or found challenging. To deepen the skills introduced in learning videos, exercises are given to the students as learning sessions that require them to apply the skills learned. Moreover, for these sessions, we provide exercise office hours, where the instructors are present in the lecture room and assist students when problems occur while conducting the exercises.

3.3 Cooperative and Problem-based Learning

Exercises must be handed in as groups, allowing more complex tasks and facilitating peer-to-peer cooperative learning. Cooperative learning [10] promotes direct communication within groups and fosters teamwork, thus improving communication skills, conflict resolution, and leadership skills [6].

These types of approaches address the required knowledge of the course. We consider them the baseline knowledge and skills. Beyond this baseline lays additional expert knowledge and skills necessary to give students a sense of competence. However, these skills quickly branch into separate sub-skills, which are hard to delineate from one another and rapidly evolve with technological progress. The possible "tree of knowledge" is obviously larger than a single student can learn in one year. Moreover, students may be more or less interested in different parts of this tree. Here, we provide autonomy to the students to pick a "mastery branch". Students pick a specific technology that they apply to a problem they have selected themselves.

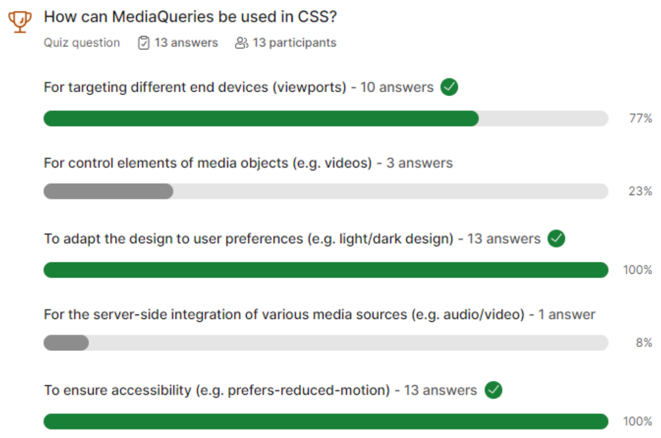
For this project part of the course, we apply problem-based learning [1] as the core method for teaching. In self-selected groups, students take a required and maybe an additional optional project to apply their knowledge. These projects are small real-world scenarios that students choose themselves. Here, they can deepen their preferred knowledge branch and learn to apply it to a problem that is important to them. This fosters a sense of autonomy, competence, and relatedness. The required project is necessary to successfully complete the course, while the optional project can attain additional credit for the final exam.

3.4 Exams and Assessment

The module aims to teach students both theory and practice in developing interactive systems. Our course employs a combination of formative and summative assessments to provide a continuous feedback loop, enhancing student learning and performance. According to Black and Wiliam [3], formative assessment practices, including low-stakes quizzes and regular feedback, play a crucial role in raising educational standards.

Critical theoretical concepts introduced in lectures are evaluated interactively using an audience response system (e.g., in our case, Slido), while practical skills are assessed asynchronously through exercises tailored for flipped-classroom sessions. An example question with accompanying answers presented to the students is illustrated in Figure 2. Students then select the correct answers, followed by the presentation of the correct responses. In cases where there is a significant discrepancy in students' selection of incorrect answers, the topic is revisited and discussed with the students.

Still, developing interactive systems is a holistic skill set that requires knowledge and competencies in a wide range of frameworks (e.g., backend technologies, frontend technologies, DevOps, version control, etc.); therefore, we assess this holistic skill set using projects, as stated above. The project results can be used as bonus credit in the summative final assessment in the form of a written


How can MediaQueries be used in CSS?
Quiz question 13 answers 13 participants

For targeting different end devices (viewports) - 10 answers ✓ 77%

For control elements of media objects (e.g. videos) - 3 answers 23%

To adapt the design to user preferences (e.g. light/dark design) - 13 answers ✓ 100%

For the server-side integration of various media sources (e.g. audio/video) - 1 answer 8%

To ensure accessibility (e.g. prefers-reduced-motion) - 13 answers ✓ 100%

Fig. 2. Assessment of a sample question from a quiz. Following the students' selection of possible answers, the assessment is displayed. Correct responses are indicated in green, whereas incorrect ones remain gray. Both the question and answers were translated into English.

exam and provide flexibility and internal differentiation. Students can—to a certain extent—pick their preferred way of attaining credit, providing an additional layer of autonomy.

3.5 Continuous Feedback and Improvement

To ensure that our approach aligns with the curriculum and students' interests and prerequisites, we elicit continuous feedback using different methods (e.g., short surveys at the end of lectures and formal university evaluation). The module has undergone three large overhauls and continuously improved from both the learner's and teachers' perspectives. In this article, we provide detailed feedback on the quality of those improvements.

4 Course Structure

The *Interactive Systems* module is structured over two semesters, with the first semester focusing on foundational skills and the second semester advancing to more complex technologies and applications.

4.1 Semester 1: Foundations of Web Development

The first semester aims to equip students with basic web development skills using HTML, CSS, and TypeScript. The course content is presented in the following sections.

Introduction to Guided Development Students begin with an introduction to development tools, specifically open source tools such as Visual Studio Code[1] and git[2]. They learn git by solving gamified tasks from learngitbranching.js.org[3]. Students acquire skills in using git and GitLab[4] for project collaboration, resolving conflicts, merging branches, and writing README documents in Markdown. Additionally, students learn the benefits of using a code formatter, such as Prettier[5]. Furthermore, they gain an understanding of the basics of web page rendering by browsers and deepen their knowledge of web development tools, such as the browser's built-in developer tools.

HTML Basics The second lecture addresses the fundamentals of HTML, including tags, nesting, and typical HTML data structures. Students also learn to use multimedia elements, such as `picture`, `audio`, and `video`. Additionally, they gain an understanding of the semantic meaning and appropriate usage of HTML elements. Students are then tasked with designing their own websites and sharing them in the GitLab repository.

Cascading Style Sheets (CSS) Basics Students then learn the fundamentals of CSS, including key concepts such as cascading, inheritance, and specificity. Additionally, they gain an understanding of selectors, nesting, and advanced techniques such as animations, transitions, and transformations. Students apply their knowledge to solve gamified tasks, such as those found on flukeout.github.io[6]. Subsequently, they apply their skills to enhance the websites they previously developed.

CSS Layout The fourth lecture concentrates on CSS layout mechanisms, specifically grid and flex layouts, media queries, float and position layout, to create responsive and accessible designs. The lecture emphasizes HCI aspects, such as developing accessible websites with media queries using principles of universal design and inclusive design, for instance, `prefers-reduced-motion` or `prefers-contrast`. Students apply their knowledge to solve gamified tasks, such

[1] Visual Studio Code. (Microsoft). Integrated Development Environment for Code Editing. Retrieved from https://code.visualstudio.com/.

[2] git. (Software Freedom Conservancy). Distributed Version Control System. Retrieved from https://git-scm.com/.

[3] learngitbranching.js.org. (Peter Cottle). Repository visualizer and sandbox with educational tutorials and challenges. Retrieved from https://learngitbranching.js.org/.

[4] GitLab. (GitLab, Inc.). DevOps Platform. Retrieved from https://gitlab.com/.

[5] Prettier. (James Long and Prettier contributors). Code formatter. Retrieved from https://prettier.io/.

[6] flukeout.github.io. (Luke Pacholski). Learn CSS Layout. Retrieved from https://flukeout.github.io/.

as Flexbox Froggy[7] to learn flexbox layout or Grid Garden[8] to learn grid layout, before applying their skills to enhance the websites they are developing.

DevOps Fundamentals Students are introduced to basic client-server infrastructure, setting up a web development environment using pnpm[9] as package manager and Vite[10] as dev server and bundler, also applying pre- and post-processors for CSS and using tsc to transpile typescript.

CSS Frameworks Students learn about CSS frameworks such as Tailwind CSS[11], daisyUI[12], OpenProps[13], UnoCSS[14], and Bootstrap[15]. They also explore Font Awesome[16] and Tabler Icons[17] for standardized iconography.

TypeScript Basics The seventh lecture introduces TypeScript, emphasizing its benefits as a statically typed language. Students learn about the advantages of static code analysis using ESLint[18]. Additionally, they are taught about the DOM and Node API, events, and global functions. More advanced TypeScript topics, such as syntactic sugar and array functions, are also covered. Furthermore, students apply these concepts to their website projects.

Asynchronicity In the eighth and ninth lectures, advanced asynchronicity techniques using TypeScript are covered. Students gain an understanding of asyn-

[7] Flexbox Froggy. (Codepip). An interactive game for learning CSS flexbox layout. Retrieved from https://flexboxfroggy.com/.
[8] Grid Garden. (Codepip). An interactive game for learning CSS grid layout. Retrieved from https://cssgridgarden.com/.
[9] pnpm. (contributors of pnpm). Fast, disk space efficient package manager. Retrieved from https://pnpm.io/.
[10] Vite. (Evan You & Vite Contributors). Fast and lean build tool for modern web projects. Retrieved from https://vitejs.dev/.
[11] Tailwind CSS (Tailwind Labs). Utility-First CSS Framework for Rapid UI Development. Retrieved from https://tailwindcss.com/.
[12] daisyUI. (Pouya Saadeghi). Component library for Tailwind CSS. Retrieved from https://daisyui.com/.
[13] OpenProps. (Adam Argyle). CSS library with custom properties to help accelerate adaptive and consistent design. Retrieved from https://open-props.style/.
[14] UnoCSS. (Anthony Fu). Instant on-demand atomic CSS engine.. Retrieved from https://unocss.dev/.
[15] Bootstrap. (The Bootstrap Authors). HTML, CSS, and JS library for developing responsive, mobile first projects on the web. Retrieved from https://getbootstrap.com/.
[16] Font Awesome. (Fonticons, Inc). Icon and font library. Retrieved from https://fontawesome.com/.
[17] Tabler Icons. (Paweł Kuna). Icon library. Retrieved from https://tabler.io/icons.
[18] ESLint. (OpenJS Foundation and ESLint contributors). Static code analysis tool for identifying problematic patterns found in web code. Retrieved from https://eslint.org/.

chronous function calls and promises. They also learn about syntactic sugar for promises using `async` and `await`. Additionally, students are introduced to the architectural software paradigm of Representational State Transfer (REST) and how to use REST APIs with the Fetch API. Students then apply these concepts to their website projects.

Client-Server Architecture The final lecture of the first semester instructs students on setting up a Node.js server using the Express[19] framework, including a database with json-server[20], and developing an API in their client-server architecture for their website. Additionally, students are introduced to server-side implementation architectures, such as Server-Side Rendering (SSR), Client-Side Rendering (CSR), and Static Site Generation (SSG), as well as client-side implementations, including Single Page Applications (SPA) and Multi-Page Applications (MPA).

Optional Project At the end of the first semester, students undertake a comprehensive software project that incorporates all the tools and concepts learned throughout the course. This optional project, which offers the opportunity to earn bonus points for the exam, involves pitching their own project ideas, receiving feedback, and presenting their final work. The project is designed to deepen the students' understanding of the concepts covered and includes implementing a website in form of a client-server architecture. Additionally, it emphasizes Human-Computer Interaction aspects by requiring the creation of a responsive website, adaptable for use on both smartphones up to desktop screens. Students are also instructed to implement accessible websites using the concepts they have learned. Examples of student work can be found in section 5.

4.2 Semester 2: Advanced Web, Mobile and Desktop Development

In the second semester, students build on their foundational web development knowledge and transition to more advanced web, mobile and desktop technologies.

React Framework and Mobile Applications The semester begins with an introduction to the React[21] framework for developing websites and mobile applications. Students learn React basics using TSX, virtual DOM, and reconciliation. They also explore React DevTools and integrate React into their deployment procedures using GitLab for continuous integration and continuous

[19] Express. (OpenJS Foundation). Fast, unopinionated, minimalist web framework for Node.js. Retrieved from https://expressjs.com/.
[20] json-server. (typicode). JSON-based REST-API mocking server. Retrieved from https://github.com/typicode/json-server.
[21] React. (Meta Platforms, Inc. and affiliates). JavaScript library for building component-based user interfaces. Retrieved from https://reactjs.org/.

deployment (CI/CD). Students are then tasked with designing their own websites using React and sharing them in the GitLab repository.

React Advanced Concepts Students delve deeper into React by learning about props, event handlers, conditional rendering, states, the component lifecycle with hooks like (`useState` and `useEffect`). They also study routing using React Router and state management using Context-API. Furthermore, students apply these concepts to their website projects.

React Styling and Frameworks The third week covers React styling and the use of CSS frameworks such as daisyUI, Mantine[22], and shadcn/ui[23] to create advanced and modern user interfaces. Additionally, students learn the benefits of using Component Workshops when implementing websites with component-based frameworks. The students are then tasked to apply styling to their React projects.

Web Applications and Progressive Web Apps (PWAs) Students learn about web applications and PWAs, including progressive enhancement, service worker and the Push Render Pre-cache Lazy-Load (PRPL) pattern. They discuss the benefits and downsides of PWAs compared to traditional web applications.

Native and Hybrid Mobile Applications In the fifth week, students are introduced to native mobile app development for iOS and Android. They deepen their understanding of the software stacks for both iOS and Android, learning about the primary differences between the two operating systems and the apps developed for them. Additionally, students explore hybrid app development using frameworks such as React Native[24]. The course also covers the benefits and drawbacks of native app development and hybrid app development in comparison to web and progressive web development.

Desktop Applications In the sixth week, students delve into desktop application development for operating systems such as Windows, macOS, and Linux. They discuss the differences between desktop applications and mobile applications. Furthermore, students are introduced to desktop app development frameworks, such as Electron[25].

[22] Mantine. (Vitaly Rtishchev). React component library. Retrieved from https://mantine.dev/.

[23] shadcn/ui. (shadcn). Component library. Retrieved from https://ui.shadcn.com/.

[24] React Native. (Meta Platforms, Inc.). JavaScript and React library for building native mobile apps. Retrieved from https://reactnative.dev/.

[25] Electron. (OpenJS Foundation and Electron contributors). Library for building cross-platform desktop apps with web standards like Vite and React. Retrieved from https://www.electronjs.org/.

Game Programming, Game Engines and Game Rendering In the seventh, eighth, and ninth weeks, students are introduced to game engines, game programming, and rendering in games. They begin by exploring the fundamentals and concepts of game engines, with an introduction to game engines such as Unity[26]. Additionally, students learn about game programming, covering topics such as game genres and game production efforts. They delve into game concepts and the development of serious games, highlighting their significance in human-computer interaction and interactive systems. Finally, students study rendering in games, focusing on the rendering pipeline, particularly the 3D rendering pipeline as described by Malaka et al. [8], including concepts such as tessellation, culling, lighting, shading, clipping, and viewport mapping.

Specialized Frameworks During the final eight weeks of the semester, students choose to specialize in React Native, Electron, or Unity for their programming project. Working in groups of up to three, they may develop a desktop application using Electron, a hybrid application using React Native, or delve into game programming using Unity. Fundamentals of using these frameworks is provided through prepared video content. Additionally, students can utilize help desk appointments to address their questions. The concept of individualized learning is presented in section 3.

Optional Project At the end of the second semester, students have the opportunity to participate in an additional project before pitching and presenting their final projects, which utilize the specialized frameworks Electron, React Native, or Unity. This optional project, which offers the opportunity to earn bonus points for the exam, involves evaluating the usability of the implemented applications to enhance their understanding of human-computer interaction.

5 Student Results

Following the acquisition of the foundations of web development, students were given the opportunity to pitch their own ideas for a web application, which they would then implement using the technologies learned during the course. This hands-on project allowed students to apply their knowledge in a real-world context, receive feedback, and, for students with prior experience, explore advanced technologies.

The project was optional in the first semester of the course, yet it received a positive response. Among the 61 total students, 28 opted to participate in the optional project. Participation in the project using specialized frameworks was made mandatory for the second semester. However, as of the time of writing, the student projects for the second semester are still ongoing and thus cannot be summarized.

[26] Unity. (Unity Technologies). Cross-platform game engine. Retrieved from https://unity.com/.

This chapter highlights the creativity and technical skills demonstrated by the students through a summary of selected projects from the first semester.

5.1 TankAlarm

The first highlighted project is called *TankAlarm*. It is a responsive website tailored for smartphones that is designed to notify users when a predetermined price threshold for a selected type of gasoline is reached at nearby gas stations. It achieves this by using the Tankerkönig API [27]. Users receive push notifications upon reaching the predetermined price threshold.

5.2 NoteSync

The next project showcased app is *NoteSync*, a learning app that allows students to take notes collaboratively and enhances group-based learning. The idea is to create groups with fellow students and to collect and share a collective knowledge base. The website also allows users to ask questions about certain posts within a group, facilitating peer-to-peer learning and support. The website operates in real-time, using WebSockets, allowing students to see updates and additions as they happen. The messages also support Markdown as well as KaTeX[28] to format the posts or questions.

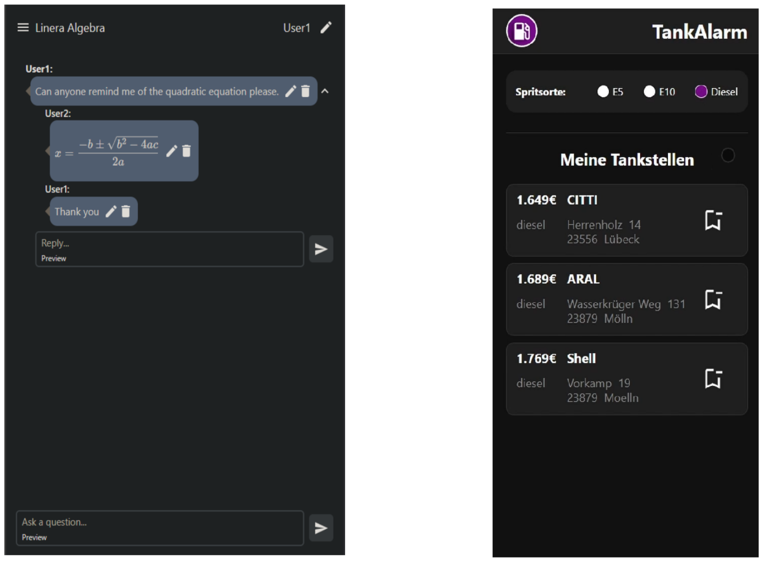
5.3 Flea

The final highlighted project is *Flea*. It is a website designed to help users search and discover flea markets. Additionally, users can create their own flea markets as well. The website offers easily accessible filters to help users find specific types of markets that align with their interests. It also features a map interface that allows users to view the location of different flea markets in their vicinity, providing a visual and intuitive way to explore flea markets.

The students' projects demonstrate remarkable creativity, technical skills, and practical application. From *TankAlarm*, which uses modern web technologies to provide real-time price notifications to *NoteSync*, an innovative learning and knowledge base app facilitating collaborative note-taking, and *Flea*, a community-driven platform to create and discover flea markets, each student project showed their ability to address real-world problems with sophisticated solutions. These projects highlight not only the technical proficiency gained through the course but also the students' ability to understand and meet user needs. The diverse range of applications underscores the broad applicability of the skills learned and the students' readiness to tackle various challenges in the field of HCI. This chapter showcases some of their impressive outcomes, setting a high standard for future students and illustrating the course's effectiveness in preparing them.

[27] Tankerkönig API. (Tankerkönig). Real-time petrol API. Retrieved from https://creativecommons.tankerkoenig.de/.

[28] KaTeX. (Khan Academy and other contributors). Math typesetting library. Retrieved from https://katex.org/.



User1:

Can anyone remind me of the quadratic equation please.

User2:

$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

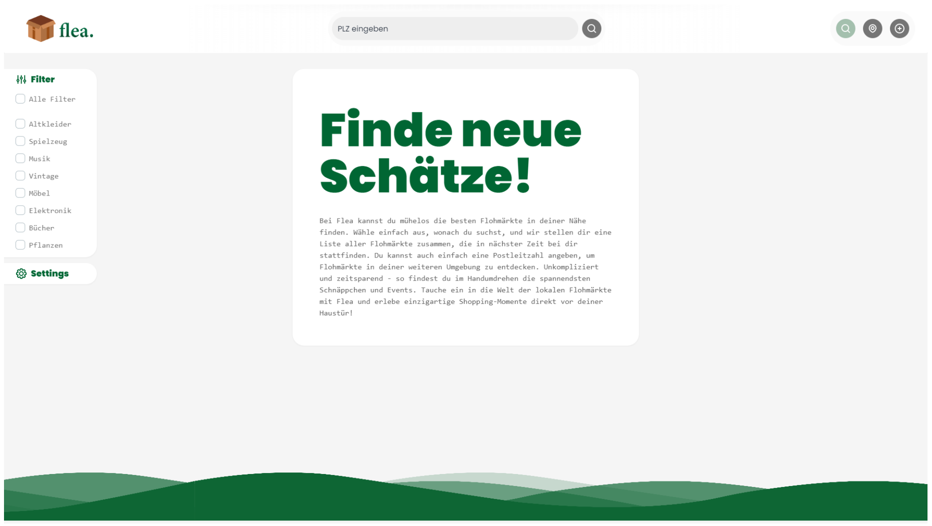
User1:

Thank you

Meine Tankstellen

1.649€	CITTI	
diesel	Herrenholz 14 23556 Lübeck	
1.689€	ARAL	
diesel	Wasserkrüger Weg 131 23879 Mölln	
1.769€	Shell	
diesel	Vorkamp 19 23879 Moelln	

Fig. 3. Screenshot of the *NoteSync* (left) and *TankAlarm* (right) website interface



Finde neue Schätze!

Bei Flea kannst du mühelos die besten Flohmärkte in deiner Nähe finden. Wähle einfach aus, wonach du suchst, und wir stellen dir eine Liste aller Flohmärkte zusammen, die in nächster Zeit bei dir stattfinden. Du kannst auch einfach eine Postleitzahl angeben, um Flohmärkte in deiner weiteren Umgebung zu entdecken. Unkompliziert und zeitsparend - so findest du im Handumdrehen die spannendsten Schnäppchen und Events. Tauche ein in die Welt der lokalen Flohmärkte mit Flea und erlebe einzigartige Shopping-Momente direkt vor deiner Haustür!

Fig. 4. Screenshot of the *Flea* website interface

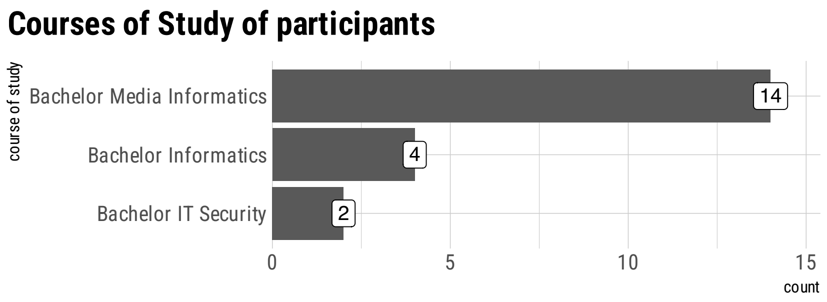


Fig. 5. Courses of study of participants

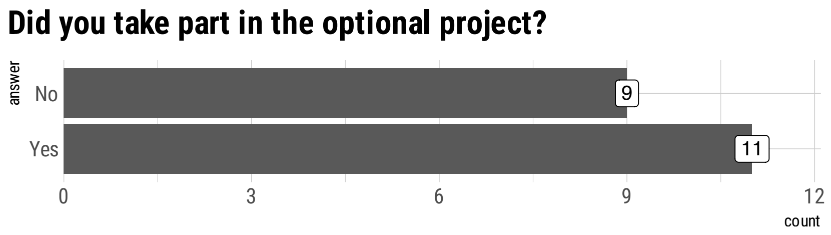


Fig. 6. Participation in optional projects

6 Evaluation

To demonstrate how we ensure the success and continuous improvement of our course, we showcase the results of a mid-semester evaluation. This section, shows how we evaluate the *Interactive Systems* course based on data collected through an online survey during the course.

6.1 Sample

We collected our evaluation data through an online survey administered during the summer semester. Participation in the survey was voluntary. A total of twenty students ($n = 20$) participated, including fourteen from the Media Informatics program, four from the Computer Science program, and two from the IT Security program (see Fig. 5). Among these participants, eleven students engaged in the optional project offered during the course (see Fig. 6).

6.2 Method

We designed the survey to gather both quantitative and qualitative feedback from students about their experiences with the course. The survey included questions about their perceptions of learning success, specific elements of the course they found beneficial, and aspects they felt could be improved. We measured quantitative responses on a six-point anchored Likert scale (translated from the German school grades: 1 = very good, 2 = good, 3 = satisfying, 4 = sufficient, 5 = poor, 6 = unsatisfactory), while open-ended questions provided qualitative insights. Our survey focused on the following areas:

- Participation in the optional project and its perceived learning success.
- Evaluation of course components, including help desk sessions, quizzes, online lectures and recordings, and group work.
- Unique aspects of the course that distinguished it from other courses.

6.3 Results

Out of the 20 students, 11 participated in the optional project. Of these, 6 students reported that their learning success from the optional project was very high with absolute certainty, while 5 students indicated their learning success as mostly high (see Fig. 7).

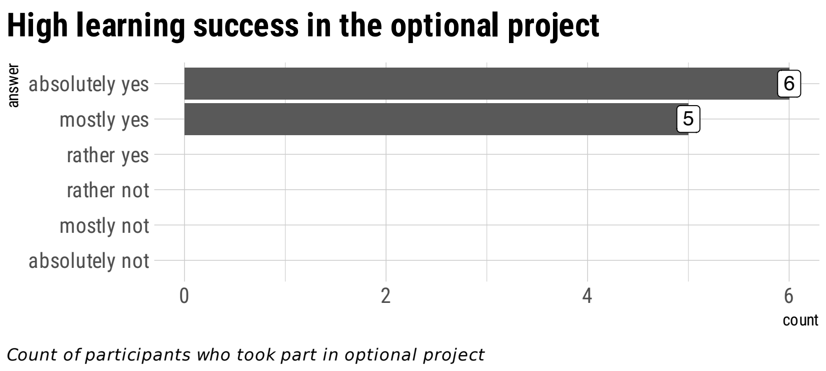


Fig. 7. Students reporting high learning success in the optional project

The qualitative survey highlighted several positive aspects of the course (see Fig. 8:

- **HelpDesk Sessions**: Students appreciated the help desk sessions, noting that they provided valuable support and allowed for detailed questions to be addressed.

– **Quizzes**: The quizzes were well-received as they helped reinforce learning and provided a self-assessment tool for students.
– **Online Lectures and Recordings**: Students highly valued the availability of online lectures and recordings, which offered flexibility and the opportunity to review the material at their own pace.
– **Group Work**: Group projects and collaborative tasks were seen as beneficial, fostering teamwork and allowing students to apply their knowledge in practical settings.

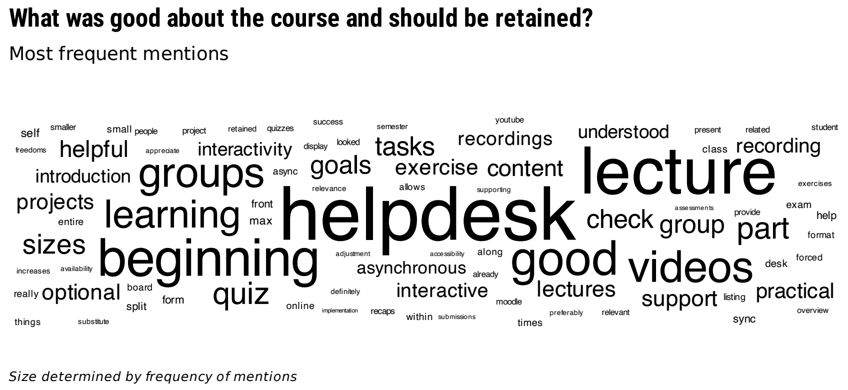


Fig. 8. Students reporting on what they liked most about the course

Students particularly appreciated the following unique aspects of the course that stood out in comparison to other courses in their course of study (see Fig. 9):

– **Relatable Practical Content**: The course content was highly relevant and applicable to real-world scenarios, making it more engaging and useful for students.
– **Good Fit of Lectures**: The alignment of lectures with course objectives and practical exercises was noted as a strength, enhancing the overall learning experience.

The quantitative evaluation shows generally positive feedback for the course improvements (see Fig. 10), with the majority of aspects receiving high ratings (1 or 2). The most highly rated improvements were the enhancement of recordings with video jump marks and the restructuring of exercise sessions as a help desk format, both receiving overwhelmingly positive feedback. The listing of learning objectives in the Moodle course and the redesign of the Moodle course overview had more mixed reviews but were still generally well-received. The definition of learning objectives in the slide sets, labeling of asynchronous learning units,

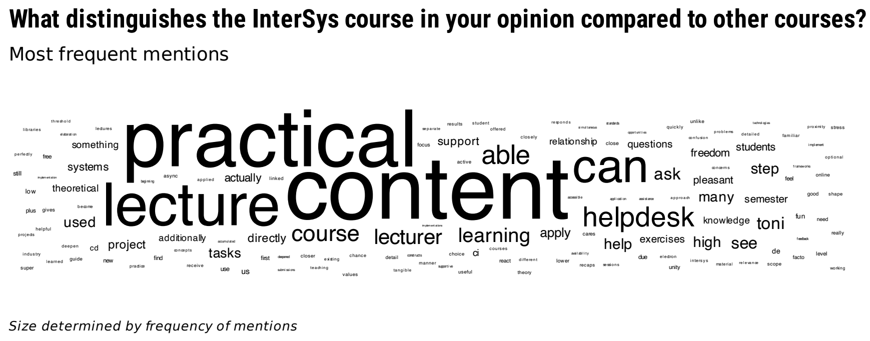


Fig. 9. Students reporting on how the course stood our compared to other courses

and reduction of exercise group sizes were also favorably rated, indicating that students appreciated these changes.

6.4 Conclusion

Our evaluation of the *Interactive Systems* course indicates a high level of student satisfaction, particularly with the practical, hands-on aspects of the course and the support provided through help desk sessions. The optional project was a significant contributor to perceived learning success. The combination of theoretical and practical components, along with flexible learning options such as online lectures and recordings, contributed to a positive learning experience.

7 Conclusion and Future Work

The *Interactive Systems* module is designed to provide HCI students with a comprehensive and practical learning experience in programming interactive systems. By combining foundational skills with advanced technologies, the course ensures that students are well-equipped to meet the specific challenges of their field. The module also emphasizes flexibility, allowing students to learn at their own pace and focus on areas of personal interest.

At present, the course materials are only available in German. We intend to translate these materials into English to make them accessible to a broader audience. Furthermore, all course materials are developed with the objective of being transformed into open educational resources, thereby benefiting a wider academic community. Our future plans include creating a massive open online course (MOOC) featuring lecture content, individual gamified exercises, reflection tasks, and self-assessments for learners. The MOOC is intended to be a free learning resource available to everyone.

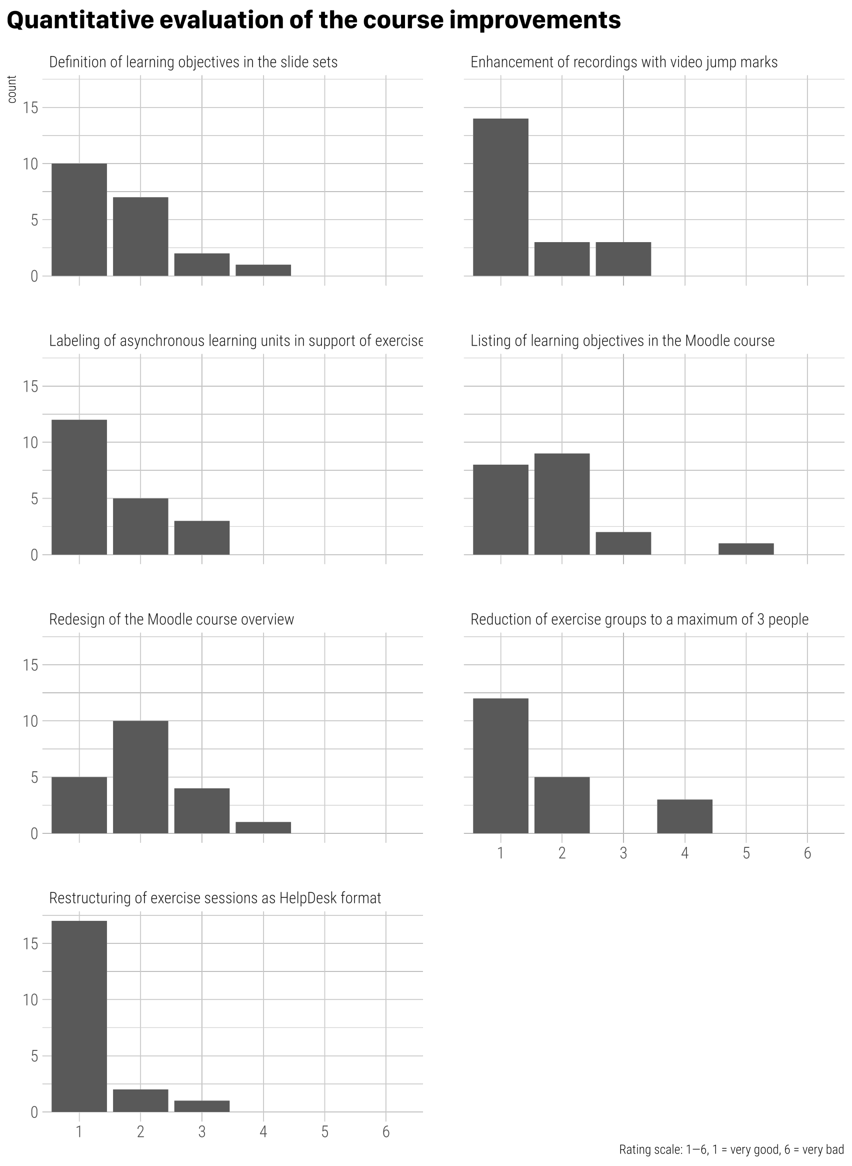


Fig. 10. Students reporting on how the course changes were perceived

Acknowledgements We would like to thank the students for their continuous feedback on how to improve this course. Translation of German qualitative data to English was done semi-automatically using ChatGPT 4o after checking for privacy issues. No privacy sensitive data was transmitted to OpenAI. A translation check was done manually by the authors.

References

1. Barrows, H.S., Tamblyn, R.M., et al.: Problem-based learning: An approach to medical education, vol. 1. Springer Publishing Company (1980)
2. Bergmann, J., Sams, A.: Flip your classroom: Reach every student in every class every day. International society for technology in education (2012)
3. Black, P., Wiliam, D.: Inside the black box: Raising standards through classroom assessment. Granada Learning (1998)
4. Deci, E.L., Ryan, R.M.: Self-determination theory. Handbook of theories of social psychology **1**(20), 416–436 (2012)
5. Hmelo-Silver, C.E.: Problem-based learning: What and how do students learn? Educational psychology review **16**, 235–266 (2004)
6. Johnson, D.W., Johnson, R.T., Smith, K.A.: Cooperative learning returns to college what evidence is there that it works? Change: the magazine of higher learning **30**(4), 26–35 (1998)
7. Kolb, D.: Experiential learning: experience as the source of learning and development. Prentice Hall, Englewood Cliffs, NJ (1984)
8. Malaka, R., Butz, A., Hussmann, H.: Medieninformatik Eine Einführung. Pearson Deutschland (2009), https://elibrary.pearson.de/book/99.150005/9783863266523
9. Piaget, J.: Piaget's theory. In: Inhelder, B., Chipman, H.H., Zwingmann, C. (eds.) Piaget and His School: A Reader in Developmental Psychology. pp. 11–23. Springer Berlin Heidelberg, Berlin, Heidelberg (1976). https://doi.org/10.1007/978-3-642-46323-5_2, https://doi.org/10.1007/978-3-642-46323-5_2
10. Slavin, R.E.: Research on cooperative learning and achievement: What we know, what we need to know. Contemporary educational psychology **21**(1), 43–69 (1996)
11. Wood, D.F.: Problem based learning. BMJ **326**(7384), 328–330 (2003)