



## Assignment of bachelor's thesis

<b>Title:</b>	Exams management and UX in LearnShell
<b>Student:</b>	Thanh Hung Le
<b>Supervisor:</b>	Ing. Jakub Žitný
<b>Study program:</b>	Informatics
<b>Branch / specialization:</b>	Web and Software Engineering, specialization Web Engineering
<b>Department:</b>	Department of Software Engineering
<b>Validity:</b>	until the end of winter semester 2022/2023

### Instructions

LearnShell is a modular system for managing and performing exams with programming assignments in scripting languages, especially Shell. LearnShell currently offers basic functionality for creating assignments and exams.

Design and implement a new front-end solution for creating, managing, and monitoring exams:

1. Analyze the current architecture of LearnShell backend and APIs available, and understand the data representations.
2. Propose a new UX flow for the creation and monitoring of exams for teachers
  - Adding assignments and students to exams conveniently
  - Monitor students during the exam
  - Add an option to prolong the exam for students with special needs
  - Check all the edge-cases and error states that might happen
3. Implement the front-end of your proposed solution.
4. Improve the tooling for rapid front-end development in the LearnShell repository.
5. Document your code, cover it with tests, and set up usability testing around the new features.



Bachelor's thesis

# **EXAMS MANAGEMENT AND UX IN LEARNSHELL**

**Thanh Hung Le**

Faculty of Information Technology  
Department of Software Engineering  
Supervisor: Ing. Jakub Žitný  
January 6, 2022

Czech Technical University in Prague  
Faculty of Information Technology

© 2022 Thanh Hung Le. Citation of this thesis.

*This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).*

Citation of this thesis: Le Thanh Hung. *Exams management and UX in LearnShell*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2022.

# Contents

<b>Acknowledgments</b>	<b>vi</b>
<b>Declaration</b>	<b>vii</b>
<b>Abstrakt</b>	<b>viii</b>
<b>List of abbreviations</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Aim of the work</b>	<b>3</b>
<b>3 LearnShell</b>	<b>5</b>
3.1 Overview . . . . .	5
3.2 History . . . . .	5
3.3 Business process . . . . .	6
<b>4 Analysis</b>	<b>9</b>
4.1 Backend . . . . .	9
4.1.1 Core . . . . .	9
4.1.2 Authentication . . . . .	9
4.1.3 Services . . . . .	11
4.1.4 GraphQL . . . . .	11
4.2 Frontend . . . . .	11
4.2.1 TypeScript . . . . .	11
4.2.2 React.js . . . . .	11
4.2.3 Next.js . . . . .	12
4.2.4 State management . . . . .	12
4.2.5 Data fetching and caching . . . . .	12
4.2.6 Styling . . . . .	13
4.2.7 Testing . . . . .	13
4.3 Similar applications . . . . .	13
4.3.1 MARAST . . . . .	13
4.3.2 ProgTest . . . . .	14
4.4 Requirements . . . . .	15
4.4.1 Functional requirements . . . . .	15
4.4.2 Non-functional requirements . . . . .	15
<b>5 Design</b>	<b>17</b>
5.1 User interface design . . . . .	17
5.2 Exam creation flow . . . . .	18
5.3 Wireframes and graphical design . . . . .	18

<b>6</b>	<b>Project cleanup</b>	<b>21</b>
6.1	Package managing	21
6.1.1	npm	21
6.1.2	Yarn	21
6.1.3	Yarn 2	21
6.1.4	Conclusion	22
6.2	Code formatting	22
6.2.1	ESLint	22
6.2.2	Prettier	22
6.2.3	EditorConfig	22
6.3	Data fetching and types generation	22
6.4	Updating dependencies	24
6.5	Import paths	24
6.6	Notifications	24
<b>7</b>	<b>Implementation</b>	<b>27</b>
7.1	Project structure	27
7.2	Global state	27
7.3	Global components	28
7.3.1	Global navigation	28
7.3.2	Header	28
7.4	Exam template	29
7.4.1	Create Exam template	29
7.4.2	Exam template list	29
7.4.3	Exam template detail	30
7.4.4	Add assignments to Exam template	30
7.5	Exam	30
7.5.1	Exam list	31
7.5.2	Create Exam	31
7.5.3	Start Exam	31
7.5.4	Exam detail	31
7.6	Parallel	31
<b>8</b>	<b>Testing</b>	<b>33</b>
8.1	Usability testing	33
8.1.1	The process	33
8.1.2	Conclusion	33
8.2	Unit testing	34
8.3	End-to-end testing	34
<b>9</b>	<b>Conclusion</b>	<b>37</b>
<b>A</b>	<b>Screenshots of the application</b>	<b>39</b>
	<b>Contents of the enclosed media</b>	<b>49</b>

## List of Figures

3.1	The process of creating and writing Exam . . . . .	7
4.1	Architecture of LearnShell . . . . .	10
4.2	Screenshot of MARAST . . . . .	14
5.1	Graphic design proposal of Parallel detail in Figma . . . . .	19
A.1	Teachers' Profile . . . . .	39
A.2	Exam template list . . . . .	40
A.3	Create Exam template . . . . .	40
A.4	Exam template detail . . . . .	41
A.5	Exam template detail - edit form . . . . .	41
A.6	Create Exam . . . . .	42
A.7	Start Exam . . . . .	42
A.8	Exam detail - finished exam . . . . .	43
A.9	Teacher exam list . . . . .	43

## List of Tables

3.1	Previous works . . . . .	6
-----	--------------------------	---

## List of code listings

1	Example of GraphQL query with fragment . . . . .	23
2	Example of generated React Query hook . . . . .	23
3	Example of usage of the generated hook . . . . .	24
4	Example of old relative paths and new absolute path . . . . .	24
5	Example of old notification call and new one . . . . .	25
6	JSX of Header component . . . . .	28
7	Example of form field with validation . . . . .	29
8	JSX of Accordion component . . . . .	30
9	Example Cypress end-to-end test . . . . .	35

*I would like to thank my supervisor Ing. Jakub Žitný, for the guidance and helpful advice, Ing. Tomáš Kalvoda, Ph.D. and Ing. Ladislav Vagner, Ph.D. for insightful interviews about MARAST and ProgTest. I would also like to thank all teaches of the Unix-like Operating System course, who were part of the usability testing for giving out much-needed comments and remarks, and last but not least, my thanks to my family and friends who supported me throughout the writing process of this thesis.*



## Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the “Work”), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on January 6, 2022

.....

## Abstrakt

Tato práce řeší návrh, implementaci a testování nového řešení pro správu testů v aplikaci LearnShell. Ve stávajícím řešení chybí uživatelské rozhraní pro tvorbu a správu testů. Řešení tohoto problému začalo analýzou back endu a podobných aplikací. Následoval návrh, vylepšení nástrojů používaných při programování frontendu a implementace řešení. Nakonec byla aplikace otestována jak uživateli tak automatickými testy. Výsledkem je funkční systém pro správu testů, která se bude používat při výuce na Fakultě informačních technologií ČVUT v Praze.

**Klíčová slova** LearnShell, TypeScript, React.js, Next.js, GraphQL, UI

## Abstract

This thesis is about designing, implementing, and testing a new solution for exam management in the LearnShell application. The current solution lacks the user interface for creating and managing exams. The solution started with an analysis of the backend and similar applications. The next step was creating a design, improving the tooling used to program the frontend, and implementing the solution. Finally, the application was tested both by usability testing and automatic tests. The result is a functioning system for exam management, which will be used in the Faculty of Information Technology CTU in Prague.

**Keywords** LearnShell, TypeScript, React.js, Next.js, GraphQL, UI

## List of abbreviations

API	Application Programming Interface
BEM	Block Element Modifier
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
CTU	Czech Technical University
DOM	Document Object Model
DX	Developer Experience
ES	ECMAScript
FIT	Faculty of Information Technology
HTML	Hypertext Markup Language
JS	JavaScript
JSX	JavaScript Syntax Extension
KOS	Komponenta Studium
LI-DL	LearnShell Input-Describing Language
ORM	Object-relational mapping
PS1	Programming in Shell 1
REST	Representational State Transfer
SQL	Structured Query Language
SWR	stale-while-revalidate
TS	TypeScript
UI	User Interface
UOS	Unix-like Operating Systems
UX	User Experience





## Chapter 1

# Introduction

Every year, hundreds of new students come to the Faculty of Information Technology to become IT professionals. In their first semester, they face many challenges, including unknown new environments, meeting new people, and university subjects. Among mathematical analysis, programming, and algorithm development, there is also Unix-like Operating Systems course(formerly Programming in Shell). Unix-like Operating Systems course explains the basics behind unix-like operating systems, like architecture, file system principles, and basic scripting in bash command prompt. Unfortunately, some of the new students have never seen Linux operating system, let alone bash. This is the point where project LearnShell comes in.

LearnShell 2.0 is a modular system for managing and performing exams with programming assignments in scripting languages. For teachers, it simplifies the process of creating assignments with custom-made language for describing the input data. Every student receives generated data according to the input description, resulting in a better evaluation of their scripting skills. On top of that, the platform provides a system of gradual evaluation and hints.

Unfortunately, LearnShell lacks a large portion of the user interface for the creation of exams. Currently, it is done by sending raw requests to the backend server, which creates a bottleneck because only a few teachers know how to do it correctly. This final thesis should change that, by providing a better user experience for both teachers and students of the UOS course. More teachers will be able to participate in the exam creation process and students will be able to look at their past exams.

This thesis builds on top of numerous previous contributions to LearnShell, which you can read about in Chapter three, alongside a brief overview of LearnShell and its history. Then you can learn about the current state of the platform, the analysis of backend and frontend services, tools, and technologies. Following that, you will find a quick run-through of two similar applications used in FIT for improving the education and the requirements for the exam management solution. In Chapter five you can find about the design process of the user interface. Before the implementation in Chapter Seven, you can find out, how we improved the frontend repository with better tooling to provide a better developer experience. At the end of this thesis, you can read about both usability and implementation testing followed by the conclusion.





## Chapter 2

# Aim of the work

The goal of this work is to create a new frontend solution for creating, managing, and mentoring exams in LearnShell 2.0 platform. This will improve the overall quality of education in Unix-like Operating System Course, by improving the user experience of both teachers and students. The individual sub-goals follow the software engineering methods. First is an analysis of the current architecture of LearnShell backend and available APIs, next is to propose, design, and implement a new UX flow for the creation and monitoring of exams. Along the way, improve the frontend repository and tooling for rapid development. And lastly, test the solution with both implementation test and usability tests.

An important part of this final thesis is the continuation of LearnShell development. The final sub-goal of this thesis is to improve the tooling and refactoring of the frontend git repository for a better developer experience.





# LearnShell

In this chapter, we will cover the current state and functionality of LearnShell, describe the motivation behind the origin of LearnShell and give a brief history of the project.

## 3.1 Overview

Currently, LearnShell is used for three purposes: online homework, two big tests in the middle and the end of the semester, and small tests before each UOS tutorial. The small tests are an essential part of the course because they promote continuous learning and at the same time give students confidence and reduce the stress of failing a single test [1].

The main advantage is a custom language explicitly created for LearnShell. *LearnShell Input-Describing Language (LI-DL)* [2] is used for describing input data for testing the correctness of students' submissions. This makes creating exams easier, however, the system currently lacks a user interface for creating said exams. This creates a bottleneck because only a few teachers know how to create new exams by sending raw requests to the backend.

## 3.2 History

At the start, written paper scripts were evaluated manually by teachers. This was time-consuming, which meant a limited number of tests and also human error-prone. On the other hand, teachers were able to give partial points if the script was not entirely correct. Nevertheless, the testing was moved to the automated examination tool ProgTest a [3], system for evaluating C/C++ programs already used at FIT. Unfortunately, preparing assignments in this system is complicated and complex, which also means a limited number of exams in the semester.

Finally, a new platform for automated evaluation was created. LearnShell allowed quick and easy creation of bash scripting assignments, with semi-randomly generated input data and gradual evaluation. But the first iteration had several problems. It was coded very quickly in an un-maintainable way resulting in *spaghetti code*. Because of that, at the start of winter semester 2019/2020 a new version of LearnShell has started programming. Over the following semesters, a number of students helped with the development of LearnShell, which you can find in Table 3.1. You can read more about the current version in Chapter 4.

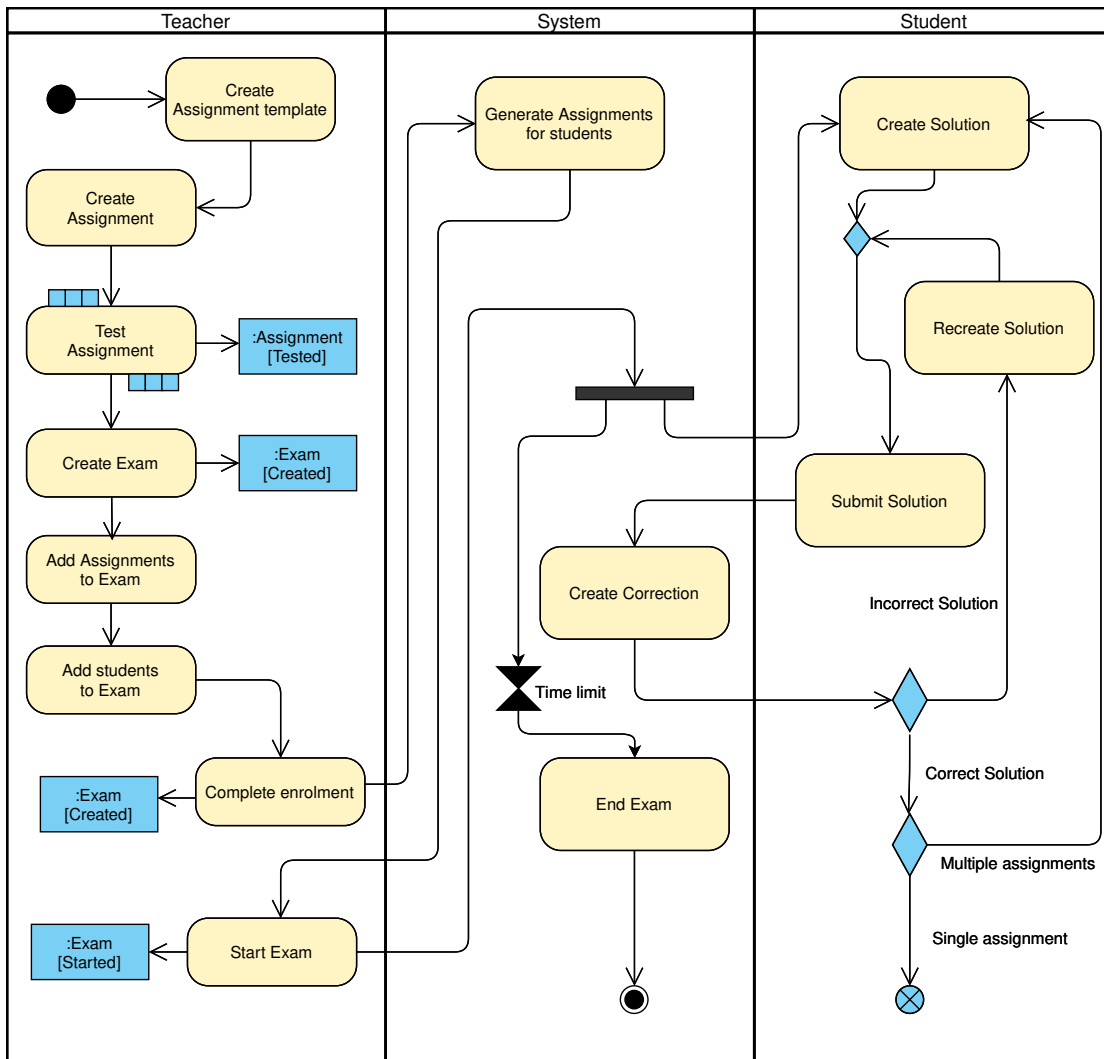
■ **Table 3.1** Previous works on LearnShell

Karel Jílek	Command and script testing system for bash language [4]
Jiří Borský	Input data generator for Bash scripts validation [5]
Matěj Karpíšek	Smart search module for LearnShell [6]
Tomáš Kalabis	Student and teacher analytics module for LearnShell [7]
Samuel Majoroš	Cluster infrastructure for LearnShell [8]
Ilya Ryabukhin	Cluster infrastructure for LearnShell: monitoring and logging [9]
Ondřej Cihlář	Improving LearnShell backend for exams and assignments [10]
Dan Pejchar	Improving LearnShell backend for analytics [11]
Pavel Khunt	LearnShell Security Audit [12]
Zbyněk Juřica	A module for detecting plagiarism in LearnShell [13]
Jaroslav Hampejs	A module for grading in LearnShell [14]

### 3.3 Business process

This section will walk through the process from having a new application without any data in the database to starting an exam for students. At the start of the new semester, all User, Parallel, and Course data are imported from KOS (CTU informational system). According to need, administrators of LearnShell can create multiple Generators and Evaluators services. In the following list and in 3.1 you can find the process of creating and writing an exam.

- 1. Create Assignment Template** Assignment Template connects Assignments and two services described before.
- 2. Create Assignment** Teacher creates Assignment from Assignment Template and adds test cases.
- 3. Create Exam template** Teacher creates an Exam template from which can be created multiple exams.
- 4. Add Assignments to Exam template** Teacher can add several previously created Assignments.
- 5. Create Exam** Teacher creates exam from Exam template.
- 6. Start Exam** Generator service creates data for every student's assignment, and then the exam starts.
- 7. Create solution** Students create a solution, and the Evaluator service creates correction.



■ Figure 3.1 The process of creating and writing Exam



# Analysis

This chapter is about the analysis of both backend and frontend parts of LearnShell and two similar web applications used in the Faculty of Information Technology. From this analysis, we will create requirements for the new exam management for LearnShell, which you'll find at the end of this chapter.

## 4.1 Backend

The central part of the LearnShell backend is a Django app called Core. The Core connects two Flask services Generator and Evaluator, PostgreSQL database and Next.js frontend. Each part is packaged into a container using Docker and connected with Docker Compose. In Figure 4.1 you can see the architecture of LearnShell.

In the following subsections, you will find a brief overview of LearnShell backend components mentioned in the paragraph above. If the kind reader wants to know more about the LearnShell backend and other features created by other students, they can look into their final theses. Ondřej Cihlář described in detail LearnShell's database model [10] or Matěj Karpíšek's in-depth look into Docker [6].

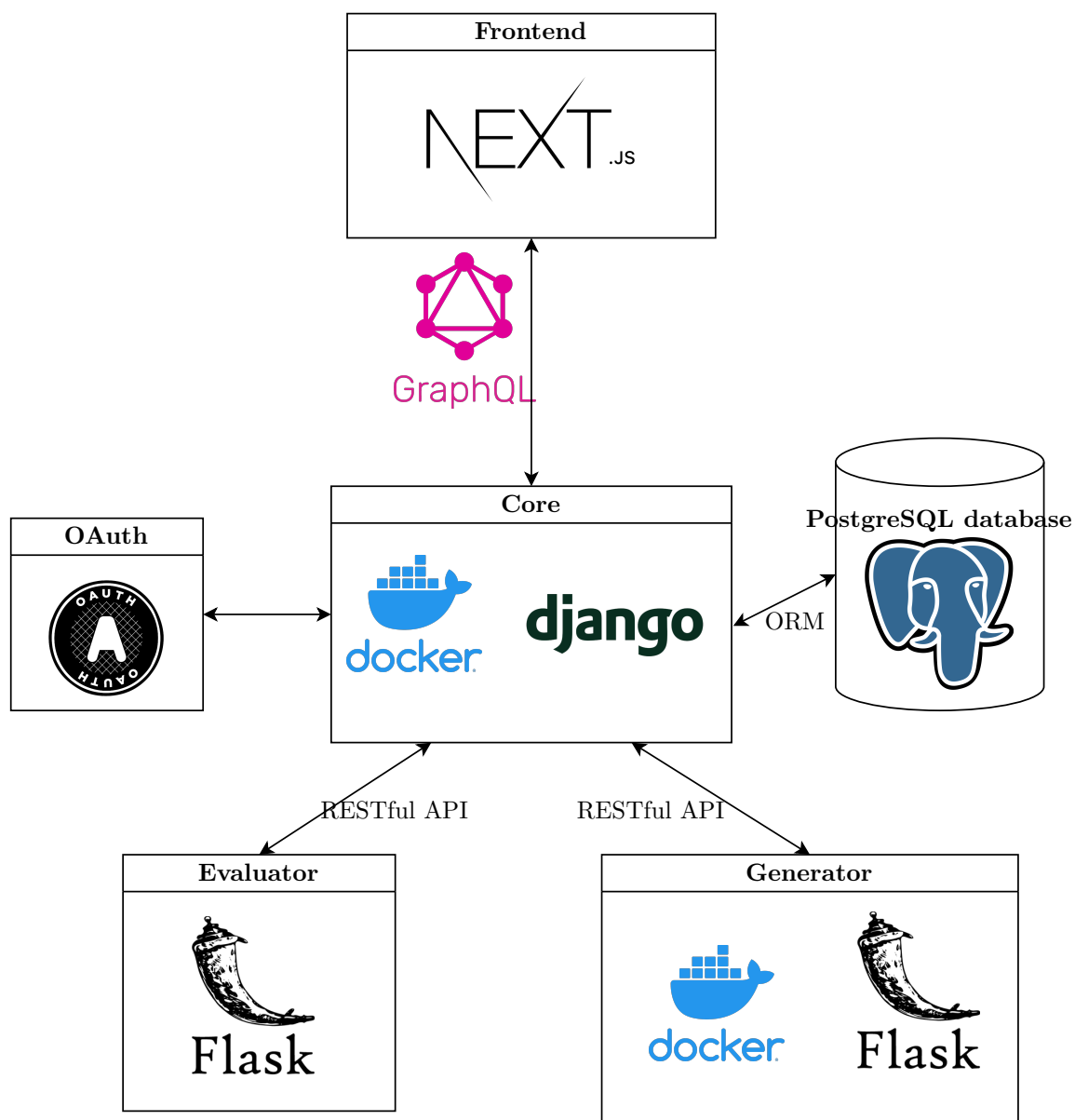
### 4.1.1 Core

The Core is a python app created with the Django framework [15]. Django was made to ease web development by encouraging rapid development and clean design. Django applications are divided into smaller parts called apps, each with its purpose. Django also offers its own database abstraction layer for easier manipulation of data. Each database table is converted to an object (model) using ORM (object-relational mapping).

For storing data, LearnShell uses PostgreSQL [16], an open-source object-relational database system that uses and extends the SQL language.

### 4.1.2 Authentication

To authenticate users, LearnShell extends Django authentication with CTU OAuth 2.0 [17]. OAuth is an open standard for granting third-party applications user information without the need to give them credentials.



■ **Figure 4.1** Architecture of LearnShell

### 4.1.3 Services

Both Generator and Evaluator are python apps created with Flask *microframework* [18]. The "micro" in microframework means that Flask aims to keep the core small and simple but extensible.

The services take data created by a teacher, including data for generating input data for testing the students' written in LI-DL code encoded in base64 (for security). Those are then compiled to a Python representation and when the scripts is run, it generates bash script. After users submits a script, the system generates input data from the input description.

To handle asynchronous requests like generating and correcting exams, LearnShell uses distributed task queue Celery [19] with Redis [20] as a message broker. Celery allows the distribution of work across threads or machines and communicates via messages using fast key value store Redis.

### 4.1.4 GraphQL

GraphQL [21] is a query language for APIs and a server-side runtime for executing queries using a type system defined based on the data. In the backend, programmers can create typed schemas to describe the data. On the other side, clients can explore and request the exact data they need. By creating custom queries or mutations GraphQL prevents under- or over-fetching problems, which traditional endpoint-based APIs like REST have.

LearnShell uses python library *django\_describer* [22] for auto-generating GraphQL API from Django models. It was created by one of the LearnShell creators to automate the creation of GraphQL APIs. We can specify which fields are exposed to the API and which users can access them. In addition to all of the CRUD operations, we can also create additional operations. From this, the library generates the GraphQL API, including pagination, filtering, and ordering.

## 4.2 Frontend

In this section, we will discuss the current frontend solution of LearnShell.

### 4.2.1 TypeScript

TypeScript [23] is a superset of JavaScript. It provides all JavaScript features and adds a type system. The main benefit of typing system is that it can highlight unexpected behavior in the code and lower the chances of bugs. It also improves developer experience by allowing to get better code completion. On top of the typing system, TypeScript adds helpful data structures and constructs like tuples, generics, and decorators.

To run the code in browsers, TypeScript files need to be *transpiled* to JavaScript. Transpilers take a source code and transform them into equivalent source code in the same or different programming language. In the case of TypeScript, it allows us to use modern syntax, which is not part of the official ECMAScript standard yet or is not implemented in all of the browsers and the transpiler converts it to code, which will run all browsers.

TypeScript first appeared in October 2012, after two years of internal development at Microsoft. It was made to tackle the shortcomings of JavaScript while working in large-scale applications.

### 4.2.2 React.js

React [24] is an open-source JavaScript library for building full-fledged web applications. React overarching principle is the separation of concerns: reduce coupling (the degree to each program

module relies on each of the other modules) and increase cohesion (the degree to which elements of a module belong together). To achieve that, React uses components, highly cohesive building blocks, loosely coupled with other components.

React uses *JSX (JavaScript Syntax Extension)* to write declarative views, an optional way to write with HTML-like syntax with the full power of JavaScript. This means users do not have to memorize another syntax and focus on JavaScript.

React is also concerned with performance. It utilizes virtual DOM (Document Object Model), an abstraction of the HTML DOM. This allows faster operations whenever some changes occur. React builds a new virtual DOM subtree on every update, checks the differences between the new one and the old one, computes the minimal set of HTML DOM manipulations, puts them in the queue, and executes the changes in a batch.

In version 16.8, React team introduced hooks, a way how to share stateful logic. Hooks slowly started to replace class-based components due to their ease of usage. React team found that working with classes in JavaScript brings many problems like understating *this* keyword, worse optimization, and low re-usability. Due to these reasons, React embraces functional style of programming, which we followed while working on this thesis.

### 4.2.3 Next.js

Next.js [25] is a framework built on top of React and adds essential features needed for production like bundling, search engine optimization, or code splitting. On top of these features, Next improves developer experience with Fast Refresh (edits are visible within seconds, without losing component state), easy page-based routing, or built-in support for CSS, CSS-in-JS, and TypeScript.

Next.js is maintained (and created) by Vercel, which runs the same-named platform as a service for hosting Next.js and other applications. Vercel provides a smooth developer experience for deploying and scaling applications, and LearnShell is hosted there.

### 4.2.4 State management

An important part of frontend development that React leaves to programmers is state management. As the complexity of frontend applications increased, so has the amount of the State inside of it. Redux [26] handles global state management by setting clear action flow and having three main principles:

**Single source of truth** The global State of an application is stored in an object called *store*. This makes it easier to inspect or debug the application.

**State is read-only** The global State is immutable and can only be changed by emitting *action*.

**Changes are made with pure functions** *Reducers* are pure functions (no side effects and their output only depends on arguments) that take in previous State and action and return next State.

This ensures predictability and makes it easier to work with State in applications.

### 4.2.5 Data fetching and caching

As an actual data fetcher LearnShell uses *graphql-request* [27]. It is a simple and lightweight GraphQL client, featuring TypeScript support (via *graphql* library) and promise-base API.

Caching allows the reusing of previously fetched data efficiently. With APIs based on endpoints (like REST), clients can easily use HTTP caching to avoid re-fetching resources. With GraphQL, where we use (usually) just one endpoint, we can utilize this strategy. That is where



library *SWR* [28] comes into play. The name is derived from an HTTP cache invalidation strategy *stale-while-revalidate*. In this strategy, firstly, the cached data are returned from cache (possibly old), then a fetch request is sent to revalidate the data, and finally, if the cached data are truly stale, replace them with the new. This approach improves user experience by serving immediately some version data, so users do not have to wait and watch loading animations.

## 4.2.6 Styling

LearnShell uses three main ways to add styling: CSS-in-JS library *styled-components* [29], UI library *Atlaskit*, and global CSS style sheet. CSS-in-JS libraries are another approach for writing CSS. Their aim is to solve CSS limitations, such as scoping, or low dynamic functionality. There are a lot of different methodologies to add modularity to CSS, such as BEM, OOCSS, or SMACSS. These methodologies create rules for naming or organizing CSS classes for better clarity and to prevent class name collisions. CSS-in-JS libraries solve these issues by generating unique CSS class names. Another feature of CSS-in-JS libraries is the automatic generation of vendor prefixes. Due to the complex CSS standardization process, new CSS features could take a long time before they are available in most popular browsers. Vendor prefixes are one of the approaches to provide early access to those experimental features.

Styled components use ES Template Literals to write CSS rules inside of a string. This means we can write with regular kebab-case syntax, but syntax highlighting and code completion must be added with an editor or editor plugins. ES Template Literals also allow to interpolation of JavaScript values and dynamic functionality through props.

The next part of LearnShell styling consists of *Atlaskit* [30]. It is a collection of reusable UI components made by Atlassian. LearnShell mainly uses structural and navigational packages like *page*, *page-layout*, or *global-navigation* but also smaller components like *button*, *spinner*, or *logo*. The components speed up the development process and include additional accessibility, but they offer little customizability.

The last part of LearnShell styling is the global style sheet. It consists of several utility classes (similar to *Tailwind* [31]) to quickly add the margin, padding, or change text size. The style sheet also contains CSS classes created from previous additions to LearnShell, which should be converted to Styled components to keep the consistency.

## 4.2.7 Testing

For unit testing, LearnShell uses JavaScript framework *Jest* [32]. *Jest* was originally designed for testing React applications (*Jest* and *React* were created by the same company) but now it works with other JavaScript libraries. *Jest* focuses on simplicity and does not require much configuration.

## 4.3 Similar applications

This section is about two similar web applications used in FIT to supplement the education in various courses.

### 4.3.1 MARAST

*MARAST* [33] is a platform, used for continuous students' knowledge testing throughout the semester and big final exams. It was created during the winter semester of the academic year 2012/2013 for internal usage as a tool for generating paper exams. Nowadays, *MARAST* offers a wide selection of questions spanning from mathematical analysis, linear algebra, statistics

The screenshot shows the MARAST dashboard with a dark navigation bar at the top containing 'MARAST', 'Exercises', 'Quizzes', 'Lectures', 'Discussion', 'Blog', and 'Thanh Hung'. The main content area is titled 'Dashboard' and features several panels:

- Welcome to MARAST!**: A blue panel with a message from Tomáš Kalvoda dated 17. 5. 2021, stating the site is for teaching mathematics at FIT CTU and mentioning a lack of English examples.
- BI-AG1: Algorithms and Graphs 1** (B211): A yellow panel with a 'Quizzes' button and the message 'There are no compulsory quizzes in this course.' It also has an 'Exercises' button and a 'My exercises' table with 45 displayed, 0 solved, and 0 saved items.
- BI-AAG: Automata and Grammars** (B201): A grey panel with 'Quizzes' and 'Exercises' buttons, and a 'My exercises' table with 45 displayed, 0 solved, and 0 saved items.
- BI-PPA: Programming Paradigms** (B201): A grey panel with 'Quizzes' and 'Exercises' buttons, and a 'My exercises' table.
- BI-PS: Probability and Statistics** (B201): A grey panel with 'Quizzes' and 'Exercises' buttons, and a 'My exercises' table.
- BI-ZDM: Elements of Discrete Mathematics** (B201): A grey panel with a 'Quizzes' button and a 'My exercises' table.

■ **Figure 4.2** Screenshot of MARAST

to graph theory and automata. Among other features belong different color modes, built-in JavaScript calculators, commenting system, blog, and lectures.

Users can log in to MARAST with three different means: CTU OAuth, local login, and Google OAuth. Local login can be used when the CTU OAuth is not working, and Google OAuth is for users not yet enrolled to FIT so that they can view public courses. After login with CTU OAuth, MARAST sends a request to Usermap API [34] to check users' roles and grant them access to different courses. Limited access to an exam for a specific group can be done with a password. Otherwise, all students have access to all quizzes or exams in the course.

While creating exams, teachers have a wide variety of setting to choose from. They can select from types of questions, language, questing tags, and much more. Teachers can also see the complete history of students' answers (number of tries, length from opening the quiz to finishing it) and IP addresses, and a hash of the sessions.

Unfortunately, MARAST has not a good way to help students with special needs. One way to circumvent this is to create another exam/quiz with a password or add all the students to a different "semester," which only contains exams. In case of cheating, the teacher can manually adjust the score of the assignment/exam. [35]

### 4.3.2 ProgTest

As mentioned in Chapter 1, ProgTest is used to evaluate C/C++ programs. It is also used to students' theoretical knowledge with short single-/multi-choice quizzes.

ProgTest allows to test students' programs to the smallest details and such it offers an enormous selection of settings and tuning to set how will the programs will be tested. According to the author [36], over the years, more and more setting options were added resulting in a bit of a

*feature creep*. These settings made creating assignments feel bloated and over-complicated.

ProgTest offers good features for students with special needs. Every student has additional information in their profile and according to this info, the time limit of an exam automatically changes.

## 4.4 Requirements

This section analyzes the requirements for the solution. They originate from the assignment, communication with the supervisor, and analysis of LearnShell and similar applications.

### 4.4.1 Functional requirements

Functional requirements determine the behavior and functionality of an application.

**F1: Exam templates** Teacher will be able to create and manage Exam templates.

**F2: Add assignments to an Exam template** Teacher will be able to add and remove assignments from the Exam template.

**F3: Exams** Teacher will be able to create and manage Exams.

**F4: Add students to exam** Teacher will be able to add students from their parallel and any other parallel.

**F5: Manage students during an exam** Teacher will be able to prolong exam duration for a specific student with special needs. The teacher will also be able to kick a student from the exam if he notices suspicious behavior during the exam.

### 4.4.2 Non-functional requirements

Non-functional requirements determine the limitations and standards for an application.

**NF1: Language** The solution will be written in TypeScript using modern (ES6 and higher) syntax.

**NF2: Platform** The solution will be integrated to the existing LearnShell platform.

**NF2: Framework** The solution will use features provided by React.js and Next.js.

**NF4: Styling** The solution will extend current styling through Atlaskit components and Styled components.

Next.js uses Babel to handle Typescript, which means we automatically get e backward compatible JavaScript code that can run on all browsers. As for CSS styles, we will be mostly using Styled components, which provide vendor prefixes for all browsers. And lastly, because creating and managing exams will be mostly done on desktop computers, the responsibility of the application has a small priority.





## Chapter 5

# Design

In this chapter, we show the process of designing the user experience and user interface for LearnShell. We had a somewhat easier job because the target audience is just one group: teachers of Unix-like Operating System course (formerly Programming in Shell 1). This means the users are more or less knowledgeable about the LearnShell application.

### 5.1 User interface design

While designing the new user interface for exam management, we followed 10 Usability Heuristics for User Interface Design by Jakob Nielsen [37]. These principles were developed over 20 years ago, but they are still relevant. The list consists of:

**Visibility of system status** “*The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time.*”

Every page in LearnShell has a clear heading. In the main navigation, the user can clearly see what part of the website the user locates, and the result of every user action is displayed with notification.

**Match between system and the real world** “*The design should speak the users’ language. Use words, phrases, and concepts familiar to the user, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.*”

**User control and freedom** “*Users often perform actions by mistake. They need a clearly marked “emergency exit” to leave the unwanted action without having to go through an extended process.*”

Users can go back and forth on multi-step forms.

**Consistency and standards** “*Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform and industry conventions.*”

The new pages and components follow the same design system as the old ones as well as the Atlassian Design System, which are the Atlassian UI components made with.

**Error prevention** “*Good error messages are important, but the best designs carefully prevent problems from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.*”

**Recognition rather than recall** “Minimize the user’s memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design (e.g., field labels or menu items) should be visible or easily retrievable when needed.”

In the multi-step forms, users can see already filled information and also on which step they are on.

**Flexibility and efficiency of use** “Shortcuts — hidden from novice users — may speed up the interaction for the expert user such that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.”

After every successful user action, there is a link to the next step, which pre-fills inputs.

**Aesthetic and minimalist design** “Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility.”

**Help users recognize, diagnose, and recover from errors** “Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution.”

Error messages and notifications are clear and simple.

**Help and documentation** “It is best if the system does not need any additional explanation. However, it may be necessary to provide documentation to help users understand how to complete their tasks.”

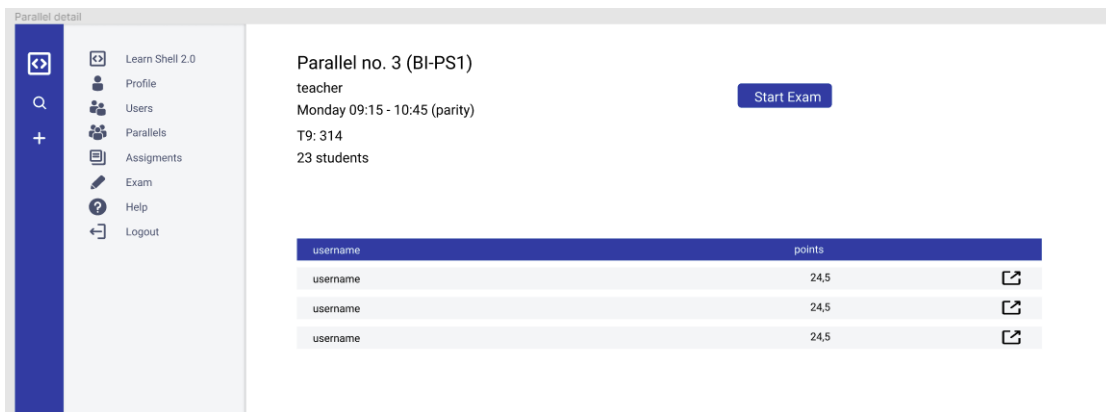
On the help page of LearnShell, users can learn about the new exam management workflow, and on strategic parts of the website are placed icon tooltips which help users to quickly get the information without the need of the help page.

## 5.2 Exam creation flow

At the start of the designing process, the thought was to put multiple steps of the exam creation process on the same page (Create Exam template and Add assignments, create and start Exam). This approach would speed up the creation, but the usability testing (more about that in Section 8.1) made it clear, that the page was cluttered and un-intuitive. Following the advice, [38]: “User is the most happy if the interface contains only one control element.” we decided to put each of the steps on its own page.

## 5.3 Wireframes and graphical design

To plan out the page structure and relation between pages, we used wireframes. Wireframes are a rough representation of a website created to arrange the elements. For the creation of wireframes, we used the website Miro [39]. We drew the frames on a tablet and pencil and then reviewed them on PC. For the graphical design, we used Figma [40].



■ **Figure 5.1** Graphic design proposal of Parallel detail in Figma





# Project cleanup

Before implementation, the frontend repository cleanup and package update were due. In this chapter, we will show which parts needed service and why. On top of that, we will revisit some tools described in Chapter 3 and explain why we replaced them.

## 6.1 Package managing

Package managers are key tools for modern (frontend) development. They simplify downloading, updating, and auditing JavaScript packages. This allows faster development through using already created utilities, functions, or whole applications instead of programming everything from scratch.

### 6.1.1 npm

npm (Node package manager) [41] is the default package manager for the JavaScript runtime environment Node.js and the world's largest software registry. Its main features are broad support and community adoption.

### 6.1.2 Yarn

Yarn [42] was created in 2016 as an answer to npm problems (at that time). It offers enhanced security, better stability, and it is faster. Another great feature is interoperability between it and npm. This means that both Yarn and npm can be used in the same project. On top of that Yarn offers online mode, workspaces, and resolving issues around versioning.

### 6.1.3 Yarn 2

Yarn 2 is brought drastic changes (so big the authors gave it another name: Berry) over Yarn and it caused a huge divide between programmers. It introduced Plug'n'Play model which removes *node\_modules* directory. This solves one of the problems of npm (*node\_modules* can get very big in size), but it broke compatibility with many projects relying on the *node\_modules* folder structure (mainly projects using React Native). Backward compatibility was later added but due to that Yarn 2 is likely to have a slow adoption rate for the foreseeable future.

## 6.1.4 Conclusion

In the end Yarn 2 was chosen for the future proof, efficiency, preference, and mainly speed. The Plug'n'Play model does not break any changes in the project (but we choose to keep `node_modules`).

## 6.2 Code formatting

This section will introduce essential tools called code formatters. The objective of these tools is to make more consistent code across developers and to detect problematic code patterns that could lead to potential bugs. All tools mentioned below were installed/configured and used in the LearnShell project.

### 6.2.1 ESLint

ESLint [43] statically analyzes JavaScript/TypeScript code to quickly find problems. ESLint can fix many of those problems automatically (on file save or with a command), so users can avoid errors with find-and-replace algorithms. Rules in ESLint are configurable and can be loaded. ESLint covers both code quality and coding style issues. It is built into most text editors and was immediately added to the frontend project.

### 6.2.2 Prettier

*“By far the biggest reason for adopting Prettier is to stop all the ongoing debates over styles. It is generally accepted that having a common style guide is valuable for a project and team but getting there is a very painful and unrewarding process. People get very emotional around particular ways of writing code, and nobody likes spending time writing and receiving nits.”* [44]

Prettier is an opinionated code formatter, which improves code readability, especially in projects with more than one person. Prettier set a generally accepted code style and helps current and future programmers of LearnShell.

### 6.2.3 EditorConfig

*“EditorConfig helps maintain consistent coding styles for multiple developers working on the same project across various editors and IDEs (Integrated Development Environment). The EditorConfig project consists of a file format for defining coding styles and a collection of text editor plugins that enable editors to read the file format and adhere to defined styles. EditorConfig files are easily readable and they work nicely with version control systems.”* [45]

## 6.3 Data fetching and types generation

While *SWR* provided a lightweight and simple way to cache fetched data, it did not quite work for LearnShell. The decision was to replace it with either Apollo Client or *react-query*. Apollo Client is the most popular caching GraphQL client. It offers a wide variety of features, but those features were not relevant to LearnShell, so *react-query* was selected as a good middle ground between Apollo Client and *SWR*. *react-query* offers stale-while-revalidate strategy as *SWR* but includes more customizability. For example, it allows setting custom time periods where the fetched data are considered *“fresh”* (in LS used for authentication). Other features include garbage collection (automatic disposal of old data), powerful dev tools, and render optimization.

React query has also nice plugin integration with *GraphQL Code Generator* (*graphql-codegen*) library. *graphql-codegen* was added to LearnShell because it can generate TypeScript types from GraphQL schema. On top of generating types, *graphql-codegen* can also generate types for GraphQL operations (queries and mutations) and fragments. From these operations, we can use another plugin to generate ready-to-use react-query hooks. This saves a lot of time and massively improves developer experience.

In Code listing 1, you can see example of a query operation in *.gql* file. Using this file and types generated with GraphQL Code Generator we can call *yarn codegen:hooks* in terminal in generate React hook shown in Code listing 2. We can then easily use this hook inside our React components like in Code listing 3.

```
// ExamTemplateList.gql

#import '@gql/fragments/CourseBasicInfo.gql'

query ExamTemplateList($courseId: ID) {
  ExamTemplateList(courseId: $courseId) {
    results {
      course {
        ...CourseBasicInfo
      }
      id
      name
      timelimit
    }
    totalCount
  }
}
```

■ **Code listing 1** Example of GraphQL query with fragment

```
export const useExamTemplateListQuery = <
  TData = Types.ExamTemplateListQuery,
  TError = unknown
>(
  variables?: Types.ExamTemplateListQueryVariables,
  options?: UseQueryOptions<Types.ExamTemplateListQuery, TError, TData>
) =>
useQuery<Types.ExamTemplateListQuery, TError, TData>(
  variables === undefined ?
    ['ExamTemplateList'] :
    ['ExamTemplateList', variables],
  graphqlRequest<Types.ExamTemplateListQuery,
  Types.ExamTemplateListQueryVariables>(ExamTemplateListDocument, variables),
  options
);
```

■ **Code listing 2** Example of generated React Query hook

```
import { useExamTemplateListQuery } from '@src/react-query-hooks'  
  
const { data, error} = useExamTemplateListQuery()
```

■ **Code listing 3** Example of usage of the generated hook

## 6.4 Updating dependencies

Even though a project might work just fine without the need to update dependencies, there are several advantages of keeping them up to date.

The main reason is to prevent security vulnerabilities. Despite the fact that our code might be secure, any third-party code we use, directly or indirectly as a dependency of another package, can contain security issues. By updating dependencies, we try to stay on top of these problems, as well as get bug fixes or performance improvements.

The next reason is to prevent major breaking changes later when we are forced to upgrade for one reason or another. If we did not upgrade our dependencies for a long time, we could face an enormous refactor of the codebase when needing to upgrade to the latest version.

Due to these reasons, we updated the dependencies we use. There were no breaking changes but updating Next.js package brought several performance improvements and features.

## 6.5 Import paths

Import paths are used to locate files while importing them as JavaScript modules. These paths can be either relative or absolute. While relative import paths are working out-of-the-box absolute paths bring many advantages. To enable an absolute path, configure *jsconfig* or *tsconfig* file for JavaScript or TypeScript project respectively. By using absolute path we will be able to:

- Copy import paths from another file without changing them.
- Easily locate file location by looking at the path.
- Move files without needing to change import paths.
- Have generally shorter import paths.

```
import { fetcher } from '../../../../modules/api'  
  
import { fetcher } from '@modules/api'
```

■ **Code listing 4** Example of old relative paths and new absolute path

## 6.6 Notifications

Notifications are an important part of an application because they help clarify events on the website and show what is happening after a user action. LearnShell uses a custom notification system based on Redux. Notifications *flags* were saved in a global state and called with *dispatch* function from Redux. This brought no advantages over a third-party package, so we replaced it with Node package *react-toastify* [46]. It adds simple, customizable notifications with an easy

way to create them. This change decluttered the codebase and added a clearer way to show users what is happening in the application.

```
dispatch(assignmentCreatedFlag('error', 'Failed to generate assignment  
for students'))  
  
toast.error('Failed to generate assignment for students')
```

■ **Code listing 5** Example of old notification call and new one



# Implementation

In this chapter, we will show the implementation of our solution. The basic workflow for each part of the application was to create GraphQL operations, generate react-query hooks using the *GraphQL Code Generator* library, and create components and pages.

In the next sections, you can find the file structure of the project and the detailed implementation of individual parts of the application.

## 7.1 Project structure

The project follows the structure set by Next.js and previous contributors.

<code>.yarn</code>	.....	folder with Yarn 2 configuration files
<code>components</code>	.....	folder with reusable React components
<code>gql</code>	.....	folder GraphQL operations and fragments
<code>impl</code>	.....	zdrojové kódy implementace
<code>layout</code>	.....	folder for page layouts
<code>modules</code>		
<code>core</code>	.....	folder containing Redux files
<code>api.ts</code>	.....	file with functions for fetching data
<code>pages</code>	.....	folder containing individual pages
<code>pages-styles</code>	.....	folder with styled-components
<code>public</code>	.....	folder with media
<code>src</code>	.....	folder with generated types and operations
<code>styles</code>	.....	shared CSS and styled-components styles
<code>utils</code>	.....	utility functions

## 7.2 Global state

To improve user experience, we use the global state to preserve users' work. Before transitioning to a next/previous step, we save information about the Exam template/Exam/Parallel to pre-fill it, saving users' time. If the user is not coming to a page from one of those links but rather through the global navigation, the application will notice this and reset the global state.

## 7.3 Global components

In the following subsections you can read about components used throughout the whole application.

### 7.3.1 Global navigation

Global navigation consists of a list of links to main pages and a sidebar containing drawers. The drawer is a collapsible element containing additional control elements in our case search drawer and drawer with shortcuts links for the creation of Assignments, Templates, and exams.

### 7.3.2 Header

The header component is used on all pages to display breadcrumb navigation and main heading. Heading can be passed either as a prop or as a custom component as a child. Breadcrumb navigation uses Atlassian same-named component, but we replace the last breadcrumb item with our own element because the last item should not be a link (not empty nor cyclic link).

```
export default function Header({ title, breadcrumbs, children }: Props) {
  const CustomBreadcrumb = React.forwardRef((props, ref) =>
    <span>{title}</span>)
  CustomBreadcrumb.displayName = "CustomBreadcrumb"

  return (
    <PageHeader
      breadcrumbs={
        <Breadcrumbs>
          {breadcrumbs?.map(breadcrumb => (
            <BreadcrumbsItem
              key={breadcrumb.link}
              text={breadcrumb.text}
              href={breadcrumb.link}
            />
          ))}
          {/* last item not link */}
          <BreadcrumbsItem component={CustomBreadcrumb} text={' '} />
        </Breadcrumbs>
      }
      disableTitleStyles
    >
      {children ? children : <h1>{title}</h1>}
    </PageHeader>
  )
}
```

■ Code listing 6 JSX of Header component



## 7.4 Exam template

Exam template consists of four pages: list of all Exam templates, detail of the template, create a template page, and page for adding assignments to a template.

### 7.4.1 Create Exam template

Create Exam template is a simple page containing a form and info about the newly created template. The form is made using the Atlaskit form component, which offers an easy way to add forms with validation. This is just enough functionality, so there is no need for another form library like formik [47].

```
const TimeLimitField = () => (
  <Field
    aria-required={true}
    name='time-limit'
    defaultValue={template.timelimit / 60}
    validate={value => {
      if (!value) return
      if (+value < 1) return 'TOO_SMALL'
      if (!Number.isInteger(+value)) return 'NOT_INTEGER'
    }}
    label='Time Limit (in minutes)'
    isRequired
  >
    ({ fieldProps, error, valid }) => (
      <>
        <TextField
          testId="timelimit-input" type='number' min={1} {...fieldProps}
        />
        {error === 'TOO_SMALL' &&
          <HelperMessage>
            Time limit must be greater than 1 minute
          </HelperMessage>}
        {error === 'NOT_INTEGER' &&
          <HelperMessage>
            Time limit must be whole number
          </HelperMessage>}
        </>
      )}
    </Field>
  )
)
```

■ **Code listing 7** Example of form field with validation

### 7.4.2 Exam template list

Exam template list is a basic page containing a list of all Exam templates created with the Atlaskit Dynamic table package. The dynamic table displays rows of data with built-in pagination and sorting. We show ID, name, time limit, course, and icon with a link to detail.

### 7.4.3 Exam template detail

Exam template detail contains basic info about the template and a list of assignments added to it. Template edit (reusing the create template component form) and delete operations, as well as deleting added assignments, are done using Atlaskit modal. Upon clicking on a button/link modal popup opens, where users can complete the action.

### 7.4.4 Add assignments to Exam template

Add assignments to Exam template page consists of two select inputs to find the Exam template we want to work with and component AssignmentList used in two slightly different ways. One is to display assignments already added to an Exam template, and the second is to display all assignments that we can add to the template. AssignmentList returns a list of Assignments inside of Accordion component, a collapsible element, which can, on click show the description of the Assignment.

```

<AccordionWrapper>
  <AccordionTitleWrapper>
    <Link href={link} passHref>
      <a>
        <Tooltip content='Detail'>
          <ShortcutIcon label='Detail' />
        </Tooltip>
      </a>
    </Link>
    <AccordionTitle>{title}</AccordionTitle>
    {collapsible && (
      <AccordionIcon AccordionIcon onClick={e => handleCollapse(e)}>
        <Tooltip content={collapsed ? 'Show description' : 'Hide description'}>
          {collapsed ? (
            <HipchatChevronDownIcon label='Show description' />
          ) : (
            <HipchatChevronUpIcon label='Hide description' />
          )}
        </Tooltip>
      </AccordionIcon>
    )}
  </AccordionTitleWrapper>
  <{!collapsed && <AccordionContent>{children}</AccordionContent>}>
</AccordionWrapper>

```

■ **Code listing 8** JSX of Accordion component

## 7.5 Exam

The exam part of the application consists of redesigned exam detail for teachers, Exams page now displays either exam created by teacher or students' exams they attended, and new create and start Exam pages.

### 7.5.1 Exam list

There are three exam lists available: teachers can see exams that they created, administrators can access a list of all exams, and students can now see a list of the exams that they participated in. All of that lists are made using the Atlaskit dynamic table component.

### 7.5.2 Create Exam

Create Exam page is a multi-step form where teaches can go back and forth on each step. Progress can be seen in the progress tracker (another Atlaskit component). It is wrapped around a component that controls the logic of the step. The wrapper takes in an array of steps object, current step, and the completed step number as props. Depending on the step and completed step ProgressTracker component allows you to visit the previous or next step or even click on the name and jump back and forth. Bellow the progress tracker, users can see info about what they selected.

The form consists of three simple steps: selecting the course, choosing an Exam template, and confirming step.

### 7.5.3 Start Exam

Star Exam page is also a multi-step form similar to Create exam, containing progress tracker and info about selected Exam template, Parallel and added students. Instead of the traditional selection of students, LearnShell uses text input for usernames. In all of the classrooms used for the UOS course, there is a bash script that periodically checks all computers, and if a student is logged in, it prints their usernames to terminal/file. This ensures that students cannot join the exam from outside the classroom and makes the process of adding students easier.

Adding students is done through the text area element, which is then parsed by trimming and squishing all white space characters. This string is then parsed and cross-checked with students in the selected parallel. If a username has not been found, it is searched in the backend and added to the displayed list of students. The list of inputted students is color-coded (students from selected parallel, different parallel, and not found students), and the summary is visible in the info section below the progress tracker. With this approach, teachers can be more confident that all students in the classroom are added, and it is a good middle ground between fetching all the students (like in the previous version) and fetching one student at a time.


### 7.5.4 Exam detail

Improved Exam detail now provides better information to teachers. The most significant improvement is a list of students taking the exam. Teachers can now see their name, username, additional time, and score. In addition, teachers can manage the student while writing the exam. Teachers can prolong the time limit for students with special needs and kick students from the exam if they notice suspicious behavior.

## 7.6 Parallel

For convenience, we added teachers' Parallel list and detail pages. In the Parallel detail, teachers can now see information about the parallel as well as a detailed list of students in the parallel. The user score is not unfortunately taken from the database, it is calculated in the backend from all students' assignments, so it might take a long time. Because of that, we at first fetch and display data without the score while sending the second request, and after the second response containing the user score arrives, we re-write the data.





## Chapter 8

# Testing

In this chapter, we will explain how the new additions of LearnShell were tested. In the following sections, you can read about usability testing of the LearnShell website and about two ways of testing the implementation: unit and end-to-end testing. To make the implementation testing more manageable, we added *data-testid* attributes to some of the HTML elements to create better queries for locating the elements on the page and the DOM.

### 8.1 Usability testing

Usability testing is a method of testing to gather feedback from real users. In usability testing sessions, participants are being observed while performing certain tasks in the application. The goal is to uncover areas of confusion and to find elements of the application, which improve the overall user experience.

#### 8.1.1 The process

The process of usability testing of LearnShell was inspired by [48]. The target group for this testing was teachers of Unix-like Operating Systems. The testing took place in person with the following setup: one camera focused on the keyboard and mouse, the second facing the test subject, and the last was screen capture. Because of previous teaches' knowledge of the LearnShell application, we chose unmoderated and explorative usability testing. This approach can reveal users' thinking processes and give us their immediate feedback. During the session, we occasionally asked follow-up questions to their comments, and if the users got stuck, we directed them in the right direction.

#### 8.1.2 Conclusion

As mention in Section 5.2, the usability testing revealed that Create Exam template and Add assignments to Exam template too cluttered and needs to be split. Similarly Create Exam and Started exam had same problem and were split. Other than this Other than this, there were some problems with consistency of some elements.

## 8.2 Unit testing

Unit tests check the functionality of the smallest units in applications - in our case, React components. Components are tested independently to isolate issues that might arise.

Alongside Jest, we used React Testing Library a [49], lightweight solution for testing React components. Its primary guideline is: “*The more your tests resemble the way your software is used, the more confidence they can give you.*” Rather than working with instances of rendered React components, React Testing library provides a way to work with actual rendered DOM nodes.

The next tool used in unit testing is Chrome extension Testing-Playground. Testing-Playground helps find the best queries to select elements for the open-source Testing-Library family. It adds a new tab to Chrome DevTools, which allows us to hover/select an element in the browser, and the extension displays the best queries to select the element, as well as some other alternative queries.

To ensure proper functionality of all components, we created a custom render, which contains our theme and Redux store provider, React Query client, and toast notification container. Instead of importing the renderer from the testing library, we use our custom renderer in our test files.

## 8.3 End-to-end testing

End-to-end testing focuses on testing high-value parts of an application. This technique tests the entire application from start to end to make sure that the application flow behaves as expected. It basically automates manual testing in the browser and ensures that all integrated parts work correctly.

For end-to-end testing of LearnShell, we chose Cypress [50], an all-in-one testing framework, assertion library, with mocking and stubbing. Cypress bundles other testing tools like Mocha and Chai to provide their best features. For better developer experience, we also added Cypress Testing Library, which allows us to add DOM Testing Library commands, which we used in React Testing Library. This is especially helpful in end-to-end tests because we want to query elements as a user would - by role and inner text (for example, a button with the text 'Create Exam'). If this type of querying is not possible, we can fall back to custom *data-testid* attributes.

Before each test, we need to log in to LearnShell, which poses quite a challenge because of OAuth. This process of authentication makes several redirects between different servers, and unfortunately, Cypress block all cross-platform request. Because of that, we made in the particular backend endpoint to log us just for this kind of testing, and before each test, we call custom command to log us in.

In Code listing 9, you can see an example of short end-to-end test. After login, we visit the Profile page (while developing the server runs on localhost port 3000). Then we navigate through main navigation to the Create Exam template page, where we fill out the info about our new Exam template and create it. Afterwards we visit the detail and delete it to restore the state of application. This short example shows how we automate the manual testing we would need to do, to ensure that our changes did not break our application.

```
it('create and delete Exam template', () => {
  cy.visit('http://localhost:3000/profile')

  // create Exam template
  cy.findByRole('link', { name: /Exam templates/i }).click()
  cy.findByRole('link',
    { name: /create new Exam template/i }).click()
  cy.wait(2000)
  cy.get('.course-select__control').type('{enter}')
  cy.findByRole('textbox',
    { name: /name/i }).type('cypress test Exam template')
  cy.findByRole('button', { name: /create template/i }).click()
  cy.findByText(/Exam template created successfully/i).should('exist')

  // delete Exam template
  cy.findByRole('link',
    { name: /go to Exam template detail/i }).click()
  cy.findByRole('button',
    { name: /delete this template delete this template/i }).click()
  cy.findByRole('button', { name: /confirm delete/i }).click()
  cy.findByText(/~Successfully DELETED Exam template .*/i).should('exist')
}
```

■ Code listing 9 Example Cypress end-to-end test





# Conclusion

The main goal of this thesis was to create a solution for creating and managing exams in the LearnShell application, used in the Unix-like Operating Systems course at the Faculty of Information Technology CTU in Prague. After analyzing the architecture of the LearnShell backend, we proposed a new UI and implemented it. Along the way, we also took a look at the frontend repository and refactored and added tooling for improved developer experience. We also performed usability testing and changed the application accordingly, and last but not least, we added an implementation test to make sure that the application worked as intended.

The solution covers all the functional and non-functional requirements described in Section 4.4. The result of this thesis is working on exam creation and management of exams in the LearnShell application. This work also helps the future development of LearnShell by improving the tooling used in the frontend repository, which will increase the speed and frequency of subsequent updates.

This solution is a good base for future exam management and the writing process. Among the following needed features is better filtering of Assignments and Exam templates (like the system of tags in MARAST) or changeable exam names (not the same name as the Exam template). LearnShell would also benefit from general features like localization, responsibility, or themes.



# Appendix A

## Screenshots of the application

The screenshot displays a user profile for '7k7xre6kDeuBGMn'. The sidebar on the left includes navigation items: Learn Shell 2.0, Users, Parallels, Assignments, Exam Templates, Exams, Help, and Logout. The main content area shows the user's name and surname, a 'Home' section, and status tags: ACTIVE, ADMIN, and STAFF. Below this, it lists '1 assignment', '1 course', and '4 parallels'. The 'Teaching parallels' section lists four items: #4 (BI-PS1) Wed 11:00, TK:PU1; #5 (BI-PS1) Wed 12:45, TK:PU1; #6 (BI-PS1) Wed 14:30, TK:PU1; and #7 (BI-PS1) Wed 18:00, TK:PU1. At the bottom, the 'Semester Score' is 1, and personal details include Username: 7k7xre6kDeuBGMn, Date joined: 13. 10. 2020, and Email: 5v7jpOLK@fit.cvut.cz.

■ Figure A.1 Teachers' Profile

Exam Templates

## Exam Templates

[+ Create new Exam template](#)

ID	Name	Time Limit	Course	Delete	Detail
185	EXAM 12 [11:00] — SOCH CERMAKOVA	10 minutes	BI-PS1		
184	EXAM 12 [9:15] — SOCH CERMAKOVA	10 minutes	BI-PS1		
183	EXAM 12 [9:15] — BARINKA JANCICKA	10 minutes	BI-PS1		
182	EXAM 12 [9:15] — ZITNY	10 minutes	BI-PS1		
181	EXAM 12 [7:30] — MUZIKAR ZITNY	10 minutes	BI-PS1		
180	EXAM 12 [7:30] — SOCH CERMAKOVA	10 minutes	BI-PS1		
179	LS-EXAM SED5	30 minutes	BI-PS1		
178	LS-TEST SED4	30 minutes	BI-PS1		
177	LS-TEST SED2	30 minutes	BI-PS1		
176	LS-TEST SED	30 minutes	BI-PS1		
175	TMP TEST PARAL 5	10 minutes	BI-PS1		
174	TMP TEST PARAL 4	10 minutes	BI-PS1		

< 1 ... 4 **5** 6 ... 13 >

■ **Figure A.2** Exam template list

Exam Templates / Create Exam Template

## Create Exam Template

Name \*

Time Limit (in minutes) \*

Select a course \*

[Create template](#)

**Exam template created**

Name: hello reader  
Time Limit: 420 min  
Course: BIK-PS1

[Go to exam template detail](#)

[+ Add assignments to this template](#)

Exam template created successfully

■ **Figure A.3** Create Exam template

Exam Templates / test redirect

## test redirect

Course: BI-PS1

Time Limit: 10 minutes

[Add assignments to this template](#)
[Create exam from this template](#)

[Delete This Template](#)
[Edit This Template](#)

Assignments (5)

Assignment ID	Name	Points	Description	Owner	Remove
687	[OLD] 006 - řádky ze souboru	2	<p>V proměnné <code>FILE</code> se nachází cesta k souboru. V proměnných <code>BEGIN</code> a <code>COUNT</code> se nacházejí čísla. Vypíšte na standardní výstup z tohoto souboru <code>COUNT</code> řádků od řádku <code>BEGIN</code>. Příklad: <code>BEGIN = 10</code> a <code>COUNT = 3</code>. Ze souboru vypíšete řádky 10, 11 a 12. Je zaručeno, že vstup vždy bude dávat smysl.</p> <p>V proměnné <code>FILE</code> se nachází cesta k souboru, jehož řádek má formát jako soubor <code>/etc/passwd</code>, tedy: <code>username:skrytý hash:hesla:uid:gid:komentář:domo</code> adresář:výchozí shell</p> <p>Sečtěte GID všech uživatelů, jejichž <code>**příhlasovací shell je hash**</code> a na standardní</p>	4j7oYJA5xRA8sqb	

■ Figure A.4 Exam template detail

Exam Templates / test redirect

## test redi

Course: BI-PS1

Time Limit: 10 min

[Add assignn](#)
[Delete This](#)

Assignments

Assignment ID

**Edit exam template test redirect**

Name \*  
test redirect

Time Limit (in minutes) \*  
10

Select a course \*  
BI-PS1

[Edit template](#)

[Close](#)

Assignment ID	Name	Points	Description	Owner	Remove
687	[OLD] 006 - řádky ze souboru	2	<p>10" a <code>COUNT = 3</code>. Ze souboru vypíšete řádky 10, 11 a 12. Je zaručeno, že vstup vždy bude dávat smysl.</p> <p>V proměnné <code>FILE</code> se nachází cesta k souboru, jehož řádek má formát jako soubor <code>/etc/passwd</code>, tedy: <code>username:skrytý hash:hesla:uid:gid:komentář:domo</code> adresář:výchozí shell</p> <p>Sečtěte GID všech uživatelů, jejichž <code>**příhlasovací shell je hash**</code> a na standardní</p>	4j7oYJA5xRA8sqb	

■ Figure A.5 Exam template detail - edit form

Exams / Create exam

### Create exam

■ Figure A.6 Create Exam

Exams / Start exam

### Start exam

■ Figure A.7 Start Exam

Exams / FINAL EXAM [18:00] — CERNY BARINKA

## FINAL EXAM [18:00] — CERNY BARINKA

### Details

Time Limit: 00:45:00  
 Number of students: 52  
 Enrollment completed: true  
 Started: true  
 Start time: 18:10:12  
 End time: 18:55:12









### Actions

+ Add students
















✓ Complete enrollment

▶ Start exam

### Assignments (4)

-  2020-vt-sd (7 points) 
-  2020-vt-re (4 points) 
-  2020-vt-ac (4 points) 
-  2020-vt-pf (var 22) (5 points) 


Search username

Name	Username	Additional time	A1	A2	A3	A4	Points	Manage
Surname Name	1cAu73d2vl	0	7	4	4	5	20	
Surname Name	3ekEriVgd3	0	7	4	4	5	20	
Surname Name	u5DZl9yglx	0	0	0	0	0	0	
Surname Name	BMB6oGY5	0	0	0	0	0	0	
Surname Name	ArtVbDimlv	0	0	0	0	0	0	
Surname Name	uqcMu9UX	0	7	4	4	0	15	
Surname Name	92cOHNBn	0	0	0	0	1	1	
Surname Name	9vmXkZjsq	0	0	4	0	1	5	
Surname Name	ijrVemAeLh	0	0	0	0	0	0	
Surname Name	n28cebBGé	0	0	2	0	0	2	
Surname Name	9j7LWm4pC	0	0	0	0	5	5	
Surname Name	oFQRgrlQic	0	7	4	0	5	16	
Surname Name	jANbqhju4e	0	0	4	0	5	9	
Surname Name	tRl1jkc8XAc	0	7	4	4	5	20	
Surname ..	25CiffuJP2M	0	0	4	4	0	8	

■ Figure A.8 Exam detail - finished exam

Exams





































## Exams

 List of all exams

+ Create an exam

▶ Start an exam

### Exams created by lethanhh

ID	Name	hasStarted	enrollmentCompleted	Delete	Start	Detail
854	test redirect	No	No			
853	test state 5	No	No			
852	test redirect	No	No			
851	test redirect	No	No			
834	test before test	Yes	Yes			
833	Test 1 [14:30] KASPAR	No	No			
832	Test 1 [16:15] KASPAR	No	No			
233	test exam sub	Yes	Yes			
226	wsl test exam22	Yes	Yes			
224	Test 1 [14:30] KASPAR	Yes	Yes			
223	Test 1 [14:30] KASPAR	No	No			
198	wsl test exam2	Yes	Yes			

< 1 2 >

■ Figure A.9 Teacher exam list





# Bibliography

1. KAŠPAR, Jiří; MUZIKÁŘ, Zdeněk. Evaluation of Student Skills in Unix Base Scripting Course. In: *Proceedings of the 2019 3rd International Conference on E-Education, E-Business and E-Technology* [online]. New York, NY, USA: Association for Computing Machinery, 2019, pp. 23–27 [visited on 2021-12-25]. ICEBT 2019. ISBN 978-1-4503-7256-5. Available from DOI: 10.1145/3355166.3355169.
2. JILEK, Karel. *Welcome to LI-DL's documentation! — LI-DL documentation* [online]. 2018 [visited on 2021-12-30]. Available from: <https://li-dl.readthedocs.io/en/latest/>.
3. VAGNER, Ladislav. *progtest.fit.cvut.cz - ProgTest* [online]. 2021 [visited on 2021-12-29]. Available from: <https://progtest.fit.cvut.cz/>.
4. JILEK, Karel. *Command and script testing system for bash language*. 2018. Bachelor's Thesis. Czech Technical University in Prague, Faculty of Information Technology.
5. BORSKÝ, Jiří. *Input data generator for Bash scripts validation*. 2019. Bachelor's Thesis. Czech Technical University in Prague, Faculty of Information Technology.
6. KARPÍŠEK, Matěj. *Smart search module for LearnShell*. 2021. Bachelor's Thesis. Czech Technical University in Prague, Faculty of Information Technology.
7. KALABIS, Tomáš. *Student and teacher analytics module for LearnShell*. 2021. Bachelor's Thesis. Czech Technical University in Prague, Faculty of Information Technology.
8. MAJOROŠ, Samuel. *Cluster infrastructure for LearnShell*. 2021. Bachelor's Thesis. Czech Technical University in Prague, Faculty of Information Technology.
9. RYABUKHIN, Ilya. *Cluster infrastructure for LearnShell: monitoring and logging*. 2021. Bachelor's Thesis. Czech Technical University in Prague, Faculty of Information Technology.
10. CIHLÁŘ, Ondřej. *Improving LearnShell backend for exams and assignments*. 2021. Bachelor's Thesis. Czech Technical University in Prague, Faculty of Information Technology.
11. PEJCHAR, Dan. *Improving LearnShell backend for analytics*. 2021. Bachelor's Thesis. Czech Technical University in Prague, Faculty of Information Technology.
12. KHUNT, Pavel. *LearnShell Security Audit*. 2021. Bachelor's Thesis. Czech Technical University in Prague, Faculty of Information Technology.
13. JUŘICA, Zbyněk. *A module for detecting plagiarism in LearnShell* [online]. 2021 [visited on 2021-11-21]. Available from: <http://hdl.handle.net/10467/95113>. Bachelor's Thesis. Czech Technical University in Prague, Faculty of Information Technology.
14. HAMPEJS, Jaroslav. *A module for grading in LearnShell*. 2021. Bachelor's Thesis. Czech Technical University in Prague, Faculty of Information Technology.

15. FOUNDATION, Django Software. *Django* [online]. 2021 [visited on 2021-12-06]. Available from: <https://www.djangoproject.com/>.
16. GROUP, PostgreSQL Global Development. *PostgreSQL* [online]. 2021 [visited on 2021-12-25]. Available from: <https://www.postgresql.org/>.
17. JIRŮTKA, Jakub. *OAuth 2.0 (Main.oauth2)* [online]. 2017 [visited on 2021-12-25]. Available from: <https://rozvoj.fit.cvut.cz/Main/oauth2>.
18. PALLETS. *Flask* [online]. 2010 [visited on 2021-12-29]. Available from: <https://flask.palletsprojects.com/en/2.0.x/>.
19. SOLEM, Ask. *Celery - Distributed Task Queue — Celery 5.2.3 documentation* [online]. 2021 [visited on 2021-12-29]. Available from: <https://docs.celeryproject.org/en/stable/>.
20. REDIS. *Redis* [online]. 2021 [visited on 2021-12-25]. Available from: <https://redis.io/>.
21. FOUNDATION, The GraphQL. *GraphQL | A query language for your API* [online]. 2021 [visited on 2021-12-29]. Available from: <https://graphql.org/>.
22. JILEK, Karel. *django-describer: A tool for automated generation of several APIs from a Django webapp.* [Online]. 2020 [visited on 2021-12-29]. Available from: [https://github.com/karlosss/django\\_describer](https://github.com/karlosss/django_describer).
23. MICROSOFT. *JavaScript With Syntax For Types.* [Online]. 2021 [visited on 2021-12-20]. Available from: <https://www.typescriptlang.org/>.
24. INC., Meta Platforms. *React – A JavaScript library for building user interfaces* [online]. 2021 [visited on 2021-12-17]. Available from: <https://reactjs.org/>.
25. VERCEL. *Next.js by Vercel - The React Framework* [online]. 2021 [visited on 2021-12-17]. Available from: <https://nextjs.org>.
26. ABRAMOV, Dan. *Redux - A predictable state container for JavaScript apps. | Redux* [online]. 2021 [visited on 2021-12-22]. Available from: <https://redux.js.org/>.
27. PRISMA. *graphql-request* [online]. Prisma Labs, 2022 [visited on 2022-01-05]. Available from: <https://github.com/prisma-labs/graphql-request>. original-date: 2017-05-10T20:05:07Z.
28. VERCEL. *React Hooks for Data Fetching – SWR* [online]. 2021 [visited on 2021-12-29]. Available from: <https://swr.vercel.app/>.
29. STYLED-COMPONENTS. *styled-components* [online]. 2021 [visited on 2021-12-20]. Available from: <https://www.styled-components.com>.
30. ATLASSIAN. *Atlassian by Atlassian* [online]. 2021 [visited on 2021-12-20]. Available from: <https://atlassian.com/>.
31. INC., Tailwind Labs. *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.* [Online]. 2021 [visited on 2021-12-29]. Available from: <https://tailwindcss.com/>.
32. META, Platforms. *Jest · Delightful JavaScript Testing* [online]. 2021 [visited on 2021-12-29]. Available from: <https://jestjs.io/>.
33. KALVODA, Tomáš; KLOUDA, Karel. *MARAST* [online]. 2021 [visited on 2021-12-29]. Available from: <https://marast.fit.cvut.cz/>.
34. JIRŮTKA, Jakob. *Usermap API (Main.usermap-api)* [online]. 2015 [visited on 2021-12-29]. Available from: <https://rozvoj.fit.cvut.cz/Main/usermap-api>.
35. KALVODA, Tomáš. *MARAST* [personal interview]. 2021.
36. VAGNER, Ladislav. *ProgTest* [personal interview]. 2021.

37. NIELSEN, Jakob. *10 Usability Heuristics for User Interface Design* [online]. 2020 [visited on 2021-12-25]. Available from: <https://www.nngroup.com/articles/ten-usability-heuristics/>.
38. STANÍČEK, Petr. *Dobrý designér to všechno ví!* First Edition. Kamenné Žehrovice, 2016. ISBN 978-80-260-9427-2.
39. *The Visual Collaboration Platform for Every Team / Miro* [online]. 2022 [visited on 2022-01-02]. Available from: <https://miro.com/>.
40. *Figma: the collaborative interface design tool.* [Online]. 2022 [visited on 2022-01-02]. Available from: <https://www.figma.com/>.
41. NPM. *npm* [online]. 2021 [visited on 2021-12-19]. Available from: <https://www.npmjs.com/>.
42. INC., Meta Platforms. *Yarn Platforms* [online]. 2021 [visited on 2021-12-19]. Available from: <https://yarnpkg.com/>.
43. OPENJS, Foundation. *ESLint - Pluggable JavaScript linter* [online]. 2021 [visited on 2021-12-29]. Available from: <https://eslint.org/>.
44. *Prettier · Opinionated Code Formatter* [online]. 2021 [visited on 2021-12-29]. Available from: <https://prettier.io/index.html>.
45. HUNNER, Trey; XU, Hong. *EditorConfig* [online]. 2021 [visited on 2021-12-19]. Available from: <https://editorconfig.org/>.
46. KHADRA, Fadi. *React-toastify / React-Toastify* [online]. 2021 [visited on 2021-12-19]. Available from: <https://fkhadra.github.io/react-toastify/introduction>.
47. FORMIUM. *Formik* [online]. 2020 [visited on 2021-12-29]. Available from: <https://formik.org/>.
48. HOTJAR, Ltd. *What is Usability Testing? (and What it Isn't) / Hotjar* [online]. 2021 [visited on 2022-01-04]. Available from: <https://www.hotjar.com/usability-testing/>.
49. DODDS, Kent C. *Testing Library | Testing Library* [online]. 2021 [visited on 2021-12-29]. Available from: <https://testing-library.com/>.
50. CYPRESS.IO. *JavaScript End to End Testing Framework* [online]. 2021 [visited on 2021-12-29]. Available from: <https://www.cypress.io/>.



# Contents of the enclosed media

readme.txt	brief description of the content of the medium
src	source code of implementation
├─ ls-web	frontend of LearnShell
├─ ls	backend of LearnShell
├─ ls-ps1-generator	Generator service
├─ ls-ps1-evaluator	Evaluator service
text	thesis text
├─ thesis	thesis text in L <sup>A</sup> T <sub>E</sub> X
├─ thesis.pdf	thesis text in PDF format
usability_testing	folder with videos of usability testing
cypress_test.mp4	screen recording of Cypress end-to-end test