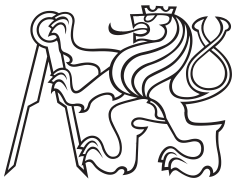
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Softwarové inženýrství a technologie

Aplikace pro vytváření a provádění online testování

Arina Momot

Školitel: Ing. Božena Mannová, Ph.D.
Obor: Katedra počítačů
Květen 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Momot** Jméno: **Arina** Osobní číslo: **492217**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Aplikace pro vytváření a provádění online testování

Název bakalářské práce anglicky:

Application for creating and conducting online testing

Pokyny pro vypracování:

Hlavním cílem práce je analýza, návrh a implementace webové aplikace pro vytváření a provádění online testování. Tato aplikace by měla sloužit jako nástroj pro kontrolu znalostí získaných studenty a sledování průběhu učení.
Proveďte analýzu, návrh, implementaci a testování webové aplikace
1. Seznamte se s problematikou a analyzujte existující systémy poskytující podobnou funkcionalitu.
2. Na základě vyhodnocení těchto poznatků, specifikujte požadavky na funkcionalitu navrhovaného systému.
3. Seznamte se s technologiemi potřebnými pro vytvoření aplikace.
4. Navrhněte architekturu systému.
5. Zvolte nástroje pro implementaci a jejich volbu zdůvodněte.
6. Implementujte aplikaci.
7. Otestujte aplikaci včetně uživatelských testů a výsledky vyhodnoťte.
Při zpracování využívejte prostředky softwarového inženýrství.

Seznam doporučené literatury:

1] Roger S. Pressmann Bruce Maxim: Software Engineering: A Practitioner's Approach , ISBN-10: 9780078022128
[2] H. Hudák, „VŠE TESTER,“ [Online]. Available: http://www.vsetester.wz.cz/
[3] IBM, "Rational Software Architect," [Online]. Available: https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-use-case

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Božena Mannová, Ph.D. kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **11.02.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

Ing. Božena Mannová, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.	
Datum převzetí zadání	Podpis studentky

Poděkování

Chtěla bych poděkovat své vedoucí práce Ing. Boženě Mannové, Ph.D. za odborné vedení a pomoc při zpracování mé bakalářské práce. Děkuji také své rodině a přátelům za neustálou podporu a motivaci po celou dobu mého studia a psaní této práce.

Prohlášení

Prohlašuji, že jsem předloženou bakalářskou práci vypracoval samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s „Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.“

Praha, 20. května, 2022

Abstrakt

Hlavním cílem práce je analyza, návrh a vývoj stabilní, multifunkční, ale zároveň snadno použitelné full-stack webové aplikace "Test&Study" pro vytváření a provádění online testování. Tato aplikace poslouží jako účinný nástroj pro kontrolu kvality znalostí získaných studenty, a urychlí sledování průběhu učení. Web je skvělým nástrojem pro učitele i studenty a pomůže jim připravit se na zkoušky a identifikovat mezery ve znalostech. Webová aplikace vyvíjena jako otevřený software (open source) a bude zpřístupněna pro budoucí opravy a vylepšení. Aplikace je také umístěna na cloudové platformě Heroku, aby ji mohl zdarma používat každý uživatel na internetu. Tato práce popisuje kompletní postup vytvoření webové aplikace "Test&Study" a obsahuje analýzu trhu konkurence, analýzu funkčních a nefunkčních požadavků na aplikaci, zvážení všech možností implementaci, je také popsáno nejoptimálnější řešení získané v průběhu zobecňování nejvhodnějších technologií. Zároveň práce obsahuje podrobný popis architektury a procesu fungování aplikace. K dispozici je také část popisující testování hotové aplikace. Webová aplikace je vyvíjena pomocí stacku MERN, který se skládá z databáze dokumentů – MongoDB, ze základního JavaScript webového serveru – Node.js, Node.js web frameworku – Express.js a z client-side JavaScript frameworku – React.js. Aktuálnost této práce je dána zrychlenou digitalizací vzdělávacího procesu a potřebou vyvíjet nové uživatelsky přívětivé softwarové programy, které by vyhovovaly potřebám učitelů a studentů, usnadňovaly a zpestřili proces učení. Přínosem této práce je provedení důkladné analýzy a vytvoření pohodlné, multifunkční a zároveň snadno použitelné webové aplikace, kterou budou moci zdarma využívat všichni uživatelé internetu. ...

Klíčová slova: webová aplikace, testování, MERN Stack

Školitel: Ing. Božena Mannová, Ph.D.

Abstract

The main goal of this work is the design and development of a stable, multifunctional, but at the same time easy-to-use full-stack web application for creating and conducting online testing. This application will serve as an effective tool for checking the quality of knowledge acquired by students, and will help to quickly monitor the progress of learning. The app is a great tool for both teachers and students and will help them prepare for exams and identify gaps in knowledge. The website is developed as open source software and will be available for future fixes and improvements. The application is also located on the Heroku cloud platform, so that it can be used free of charge by any user on the Internet. This work describes the complete process of creating a web application "Test&Study" and contains an analysis of the competition market, an analysis of functional and non-functional requirements for the application, consideration of all implementation options and also describes the most optimal solution obtained during the generalization of the most appropriate technologies. In addition, the work contains a detailed description of the application architecture and the process of its functioning. There is also a section describing the testing of the finished application. The web application is developed using the MERN stack, which consists of a document database – MongoDB, a premier JavaScript web server – Node.js, Node.js web framework – Express.js and client-side JavaScript framework – React.js. The relevance of this work is due to the accelerated digitalization of the educational process and the need to develop new convenient software products that would satisfy the needs of teachers and students, facilitate and diversify the learning process. The benefit of this work is to conduct a detailed analysis and create a convenient, multifunctional and at the same time easy-to-use web application, which will be free to use by all Internet users. ...

Keywords: web application, testing, MERN Stack

Title translation: Application for creating and conducting online testing

Obsah

Obrázky

Tabulky

Část I

Úvod

Kapitola 1
Úvod

Testovací systém je univerzálním nástrojem pro zjišťování učení studentů na všech úrovních vzdělávacího procesu. Test se skládá ze systému testových položek, standardizovaného postupu provádění, zpracování a analýzy výsledků. Může pomoci studentům připravit se na důležité zkoušky a identifikovat své slabé oblasti, na které je třeba se zaměřit a lépe se naučit. Ale zvládnutí metodiky testování a vytvoření základů testových úloh pro akademické obory vyžaduje v moderních podmínkách mnoho práce učitelů i studentů. Na pomoc jim proto přicházejí informační technologie, které mohou tento proces výrazně usnadnit a urychlit.

Moderní doba rozvoje vysokých škol je charakterizována výrazným nárůstem počtu studentů studujících na kombinované a distanční formy studia. To také způsobuje relevantnost a potřebu programů, které umožňují distanční testování bez opuštění domova.

Zároveň četné studie ukázaly, že stres ze zkoušek je jedním z nejběžnějších typů stresu. Testovací programy mohou studentům pomoci psychicky se připravit na zkoušky a trénovat sebeovládání, schopnost rychle se zapojit do procesu a soustředit se. Také student bude mít představu, co ho asi čeká na zkoušce, bude schopen vypracovat vhodný pro sebe plán přípravy a trochu se uklidnit.

Vybrala jsem si toto téma, protože jsem také student a tím i součástí skupiny, na kterou je tato práce zaměřena a často při přípravě na zkoušky mám otázku, jak to nejúčinněji udělat. Moje osobní zkušenost mi může pomoci určit, jaké požadavky musí splňovat program a jaké příležitosti by měla mít. Zároveň mám skvělou možnost provést průzkum mezi učiteli a studenty na mé univerzitě. Toto téma ve mně vzbudilo obrovský zájem i po technické stránce.

1.1 Základní informace

Před zahájením vývoje webové aplikace je nutné určit účel práce, provést analýzu konkurenčního trhu a standardizovat data a úkoly, které jsou nezbytné pro implementaci vývojových fází.

1.2 Zadání práce

1.2.1 Původní zadání

Hlavním cílem práce je analýza, návrh a implementace webové aplikace pro vytváření a provádění online testování. Tato aplikace by měla sloužit jako nástroj pro kontrolu znalostí získaných studenty a sledování průběhu učení. Proveďte analýzu, návrh, implementaci a testování webové aplikace

1. Seznamte se s problematikou a analyzujte existující systémy poskytující podobnou funkcionalitu.
2. Na základě vyhodnocení těchto poznatků, specifikujte požadavky na funkcionalitu navrhovaného systému.
3. Seznamte se s technologiemi potřebnými pro vytvoření aplikace.
4. Navrhněte architekturu systému.
5. Zvolte nástroje pro implementaci a jejich volbu zdůvodněte.
6. Implementujte aplikaci.
7. Otestujte aplikaci včetně uživatelských testů a výsledky vyhodnoťte.

Při zpracování využívejte prostředky softwarového inženýrství.

1.2.2 Upřesnění zadání

Po několika konzultacích a podrobném rozebrání základního problému, který by měl daný projekt vyřešit, jsme společně s vedoucí projektu dospěli k rozhodnutí, že aplikace bude sloužit jako pomůcka pro studenty a učitele během výuky a přípravě na zkoušky.

1.3 Cíl práce

Hlavním cílem práce je vytvoření full-stack webové aplikace umožňující tvorbu a provádění online testů za účelem usnadnit a zpříjemnit jak pro studenty, tak pro učitele proces přípravy na zkoušky a také identifikovat mezery ve znalostech studentů. Je nutné vytvořit funkční systém, který může být později použít v různých vzdělávacích institucích a poskytne větší komfort pro učitele a studenty při přípravě na zkoušky.

1.3.1 Sekundární cíli práce

K vyřešení stanoveného cíle je nutné provést následující úkoly:

- analýza konkurenčního trhu a identifikace slabých stránek konkurentů za účelem získání konkurenční výhody;

- provedení podrobné analýzy požadavků na vytvořenou aplikaci a způsobů jejich splnění;
- důkladné promyšlení architektury aplikace. Analýza všech možných způsobů její implementace a výběr nejlepšího řešení;
- promyšlené a úplné testování vytvořené webové aplikace pro identifikaci nedostatků a potřeby její vylepšení;

1.4 Metodika

V souladu se zamýšlenými cíli práce jsem určila následující metody výzkumu: přezkoumání teoretických zdrojů o problému (např. metodologická literatura, články, knihy a online zdroje), porovnání již existujících řešení, analýza získaných informací. Generalizace a strukturování informací za účelem vytvoření konkrétního řešení a následně jeho modelování pomocí diagramů. Použití speciálních výzkumných metod (např. sběr vědeckých zdrojů, sběr informací, SWOT analýza). Vývoj softwaru je naplánován pomocí agilní metodiky.

1.5 Relevantnost práce

V současné době můžeme sledovat prudký nárůst používání internetu a digitálních technologií v procesu učení. Relevantnost této práce je způsobena neustále probíhajícím procesem digitalizace metod vzdělávání a uvedením online testování do praxi v mnoha vzdělávacích institucích. Také kvůli nestabilní epidemiologické situaci po celém světě a zavedení distanční výuky ve většině vzdělávacích institucí, poptávka po webových aplikací pro testování studentů výrazně vzrostla.

1.6 Struktura práce

V II části této práce se provádí analýza a porovnání podobných již existujících webových aplikací na trhu. Zároveň popisuje požadavky na aplikaci, její strukturu a také v ní byla provedena podrobná business analýza pomocí různých analytických metod. III část se zaměřuje na zkoumání a vyhodnocování možných řešení na základě analýzy provedené v první části a výběru jednoho z nejvhodnějších řešení pro vývoj webové aplikace. Ve IV části práce je popsán proces implementace webové aplikace, její vlastnosti a způsob fungování. V část je věnována testování již vytvořené aplikace.

1.7 Roadmapa

Projekt bude realizován v několika etapách:

1. Provedení business a softwarové analýzy.
2. Vývoj webové aplikace.
3. Testování webové aplikace.
4. Odstranění vad aplikace.
5. Psaní dokumentace k aplikaci.
6. Prezentace projektu.

Část II

Analýza

Kapitola 2
Analýza

V této kapitole je popsána analytická část aplikace, která je jednou z nejdůležitějších částí při vývoji aplikace a slouží jako základ pro její implementaci. Byla provedena detailní analýza konkurenčního trhu, porovnání analogů a následně vypracování požadavků na aplikaci "Test&Study", které jí umožní mít dobrou konkurenceschopnost na trhu. Tato část také obsahuje technický popis aplikace pomocí různých užitečných metod a diagramů.

2.1 Analýza existujících řešení

Tato část popisuje a analyzuje již existující řešení pro vytváření test kvízů. V dnešní době existuje velké množství takových programů, takže je velmi důležité provést analýzu trhu a porovnat všechny možnosti pro zjištění jejich schopností, silných a slabých stránek. Pomocí analýzy konkurentů bude možné analyzovat úspěšné řešení a chyby konkurentů, identifikovat, co v tomto segmentu trhu je populární a co nového je možné nabídnout, posoudit potenciál internetového projektu jako nástroje k dosažení zisku. To pomáhá nastavit správný vektor vývoje aplikace, minimalizovat rizika a najít body růstu. Nejprve v této části práce je podrobně popsáno každé dostupné podobné webové řešení a poté bylo provedeno jejich srovnání s webovou aplikací "Test&Study" popsanou v této práci.

2.1.1 ProProfs Quiz Maker

ProProfs Quiz Maker je cloudový kvízový software pro vytváření a doručování online zkoušek a testů. Proprofs umožňuje vytvářet otázky s jednou nebo více správnými odpověďmi, otázky s chybějícími slovy nebo také otázky, které vyžaduje podrobnou odpověď. ProProfs obsahuje bezplatné šablony s automatizovaným hodnocením. Aplikace umožňuje vkládat do otazek textové dokumenty a prezentace, soubory PDF, stejně jako obrázky, audio a video soubory. Test může být veřejně dostupný na webu Proprofs nebo vložen na osobní stránku. K dispozici je jak bezplatný plán, tak několik placených, které rozšiřují možnosti platformy. Pro nové uživatele existuje 15denní bezplatná zkušební verze všech tarifů. [1]

Typy otázek: výběr jedné z možností, výběr více z možností, otevřená (krátká a dlouhá), zaškrtávací, prázdná políčka, pravda/lež, přiřazovací, na pochopení, hotspot, dropdown, video (umožňuje odpovědět pomocí videa). Celkem 10 typů otázek.

Uživatelsky přívětivá rozhraní: nepříliš moderní ale celkově příjemné rozhraní. Velmi mnoho různých detailů, potřebuje čas na pochopení, jak používat aplikaci.

Jazyková podpora: arabština, němčina, angličtina, francouzština, hindština, portugalština. Celkem 6 jazyků.

Uživatelské recenze:

- Výhody: existence témat a šablon, snadnost použití.
- Nevýhody: editační rozhraní je těžkopádné, drahé, problémy s exportem výsledků.

Možnosti:

- vytváření a kombinování různých typů otázek v testu: výběr z více možností, shoda, vyplnění prázdného místa, esej;
- automatizované hodnocení;
- vložení dokumentů, prezentací, souborů PDF, obrázků, audia a videa do otázek;
- možnost vložit test na vlastní stránku pomocí kódu;
- přizpůsobitelné šablony kvízů;
- kvíz branding;
- certifikát o ukončení testu;
- kompatibilní se všemi zařízeními – smartphony, tablety nebo PC;

2.1.2 KAHOOT!

Kahoot! umožňuje prováděné testů distančně nebo ve třídě v režimu „tady a teď“, kdy studenti vidí otázku na obrazovce a ihned na ni odpovídají pomocí telefonu nebo počítače. Učitel může zaslat každému žákovi individuální kód, pomocí kterého vstoupí do virtuální učebny. Pohodlná funkce, protože studenti, které se nechtějí rozloučit s telefonem, jsou zapojeny do procesu učení se zájmem a spojují učení s potěšením. V bezplatné verzi jsou pro tvorbu testů k dispozici pouze dva typy otázek: s jednou správnou odpovědí ze čtyř a ve formátu „pravda / nepravda“. Když si zakoupíte pokročilou verzi, budete mít přístup ke všem typům otázek a knihovně s miliony obrázků, které můžete

přidat do svých testů, aby byly zábavné a vizuální. [2]

Typy otázek: quiz, pravda/lež, otevřená (krátká a dlouhá), puzzle, quiz + audio, průzkum, otázka názoru, brainstorm, polouzavřené, slide. Celkem 10 typů otázek.

Uživatelsky přívětivá rozhraní: přehledné a vizuálně příjemné rozhraní. Mnoho jasných a zajímavých detailů. Velké množství sekcí a informací. Pochopení navigace vyžaduje čas.

Jazyková podpora: angličtina, španělština, francouzština, portugalština, němčina, norština, italština, japonština, holandština, turečtina a polština .Celkem 11 jazyků.

Uživatelské recenze:

- Výhody: snadné použití, interaktivní a neobvyklá funkcionalita.
- Nevýhody: drahé tarify.

Možnosti:

- používání připravených kvízů na jakékoliv téma (předmět) pro různé věkové kategorie;
- kombinování různých typů otázek (puzzle, test s více správnými odpověďmi, otevřené otázky, řazení atd.);
- prováděné testů distančně nebo ve třídě v režimu „tady a teď;
- na webu je režim bonusů pro rychlé odpovědi. Soutěžní režim zvyšuje zájem studentů;
- získávání výsledků ve formě zprávy s grafy. Je mozne si prohlédnout individuální pokroky každého studenta a celkový výkon třídy. Studenti sami mohou také sledovat své pokroky podle speciální tabulky. A učitel si uvolní čas díky automatickému sčítání;

2.1.3 Google Forms

Formuláře Google vám umožní sbírat a třídit velké i malé množství informací. A to zcela zdarma. Google Forms je software pro správu průzkumu, který je součástí bezplatné webové sady editorů Dokumentů Google nabízené společností Google. Služba zahrnuje také Dokumenty Google, Tabulky Google, Snímky Google, kresby Google, Weby Google a Google Keep. Formuláře Google jsou k dispozici pouze jako webová aplikace. Aplikace umožňuje uživatelům vytvářet a upravovat průzkumy on-line při spolupráci s ostatními uživateli v reálném čase. Shromážděné informace lze automaticky zadat do Excel tabulky. Výsledný formulář lze studentům zaslat e-mailem nebo pomocí

speciálního kódu vložit na váš web. [3]

Typy otázek: otevřená (krátká a dlouhá), výběr jedné z možností, výběr více z možností, zaškrtávací, dropdown, hodnocení, matice(x2), dotaz na datum, dotaz na čas. Celkem 10 typů otázek.

Uživatelsky přívětivá rozhraní: celkově srozumitelné a uživatelsky přívětivé rozhraní. Obsahuje mnoho funkcí a vše je na jedné obrazovce. Potřebuje čas na pochopení.

Jazyková podpora: angličtina, španělština, francouzština, portugalština, němčina, norština, italština, japonština, arabština,.. Celkem 100+ jazyků.

Uživatelské recenze:

- Výhody: žádné omezení průzkumů, otázek nebo odpovědí.
- Nevýhody: obtížné provádět změny v vytvořených formách, z důvodu vyšší návštěvnosti uživatelů se formulář okamžitě rozbije.

Možnosti:

- zcela zdarma;
- přizpůsoben pro mnoho jazyků;
- umožňuje vytvářet testy s různými testovými otázkami;
- integrován s dalšími službami Google, včetně úspěšně možné použít s Google Classroom;
- možnost sledování statistiky a výsledků testů;
- možnost exportu výsledků testu do Excelu;

2.1.4 Microsoft Forms

Pomocí Microsoft Forms, můžete vytvářet průzkumy, kvízy, testy a ankety a snadno vidět výsledky a data – a to už v okamžiku, kdy jsou pořízena. Microsoft Forms umožňují učitelům vytvářet rychlá formativní nebo sumativní hodnocení. Učitel může vygenerovat odkaz, který vloží do Poznámkového bloku předmětu OneNote nebo do příspěvku v Microsoft Teams. Když žák použije odkaz, spustí se test v zabezpečeném režimu, který brání, aby žák podváděl a přistupoval k externím informacím, jako jsou webové stránky nebo další aplikace. [4]

Typy otázek: výběr jedné z možností, výběr více z možností, otevřená (krátká a dlouhá), hodnocení, dotaz na datum. Celkem 5 typů otázek.

Uživatelsky přívětivá rozhraní: minimalistické a intuitivní srozumitelně rozhraní.

Jazyková podpora: angličtina, španělština, francouzština, portugalština, němčina, norština, italština, japonština, holandština, turečtina a polština,.. Celkem 80+ jazyků.

Uživatelské recenze:

- Výhody: super jednoduché použití, existence témat a šablon, možnost změny vzhled testu.
- Nevýhody: malá funkčnost, nedostatek návodů k použití.

Možnosti:

- vytváření testů pomocí jednoduchých nástrojů a pokynů pro vizuální návrh;
- vytváření a kombinování různých typů otázek v testu;
- sdílení formulářů – duplikace formuláře a sdílení jako šablony, spolupracujte na formuláři, sdílení výsledků formuláře;
- možnost vložení obrázků, videa a matematických funkcí do otázky;
- integrace s jinými produkty Office;
- možnost exportu výsledků testu do Excelu;

2.1.5 ClassMarker

ClassMarker umožňuje provádět průzkumy s různými formáty odpovědí – kromě obvyklých možností existují i eseje. Pro začátek musí učitel vytvořit virtuální učebnu a rozeslat pozvánky studentům. Pro vytvoření a poskytnutí přístupu dalším uživatelům musí být každý v programu registrován. ClassMarker uchovává výsledky všech provedených testů a vede statistiky výkonnosti. Pokud má učitel vlastní webovou stránku, může do ní vložit vytvořený formulář. [5]

Typy otázek: otevřená (krátká a dlouhá), výběr jedné z možností, výběr více z možností, pravda/lež, přiřazovací, gramatická otázka. Celkem 6 typů otázek.

Uživatelsky přívětivá rozhraní: nepříliš intuitivní srozumitelně rozhraní, pro navigaci v této službě musíte být trochu technicky zkušení.

Jazyková podpora: arabština, bulharština, čínština, dánština, holandština, finština, francouzština, němčina, řečtina, maďarština, italština, japonština,.. Celkem 23 jazyků.

Uživatelské recenze:

- Výhody: skvělá hodnota za spravedlivou cenu.
- Nevýhody: existují nevýhody při vytváření matematických otázek.

Možnosti:

- vytváření a kombinování různých typů otázek v testu;
- možnost nastavit počet pokusů pro každého uživatele v testu;
- plně přizpůsobitelné certifikáty, které jsou generovány automaticky;
- možnost exportu výsledků testu do Excelu;
- e-mailové výsledky tvůrcům, asistentům a testujícím;

2.1.6 Porovnání funkcionalit existujících řešení

Pro porovnání všech výše uvedených možností a aby to bylo přehlednější jsem se rozhodla sestavit porovnávací tabulku.

Tabulka 2.1: Porovnání funkcionalit současných řešení

Vlastnost	ProProfs Quiz Maker	Microsoft Forms	KAHOOT	Google Forms	ClassMarker
Počet typů otázek	10	5	10	10	6
Vložení souborů do otázky	+ Obrázky, videa, PDF soubory - Chybí matematické a fyzikální vzorce	+ Obrázky, videa, matematické funkcí	+ Obrázky, videa - Chybí matematické a fyzikální vzorce	+ Obrázky, videa - Chybí matematické a fyzikální vzorce	- Chybí obrázky, soubory, matematické a fyzikální vzorce
Nahrávání materiálů, generování testu	Ne	Ne	Ne	Ne	Ne
Cena za použití	Základní plán zdarma(omezené funkce). Placené plány: Essentials - $0,50 uživatel/měsíc a Premium - $2	Zcela zdarma – je třeba mít Microsoft účet	Bezplatná verze(jednoduché kvízové otázky). Placené plány: Kahoot! Pro - 3€ na učitel/ měsíc a Kahoot! Premium - €6	Zcela zdarma – je třeba mít Google účet	Bezplatná verze(limit pro vytvoření maximálně 100 testů/měsíc. 400 testů za měsíc - 16,50$ 1000 testů/měsíc - 3$.
Hlavní funkce jsou zdarma	Ano	Ano	Ano	Ano	Ano
Export výsledků	CSV, Excel	Excel	Excel	Excel	CSV, Excel
Počet podporovaných jazyků	6	80+	11	100+	23
Kompatibilita (smartphony, tablety, PC)	Všechna zařízení	Všechna zařízení	Všechna zařízení + mobilní aplikace	Všechna zařízení	Všechna zařízení
Podpora OS	Windows, Mac, Linux, Android, iOS	Windows, Mac, Linux, Android, iOS	Windows, Mac, Linux, Android, iOS	Windows, Mac, Linux, Android, iOS	Windows, Mac, Linux, Android, iOS
Podporované prohlížeče	Firefox, Safari, Google Chrome, MS Edge	MS Edge, Google Chrome, Firefox, Safari	Google Chrome, Firefox, Safari, MS Edge	Google Chrome, Firefox, MS Edge, Safari	MS Edge, Firefox, Google Chrome, Safari, Opera

Na základě tabulky můžeme dojít k závěru, že nejvíce naplněné aplikace a v důsledku toho jsou hlavními konkurenty aplikace "Test&Study"jsou ProProfs Quiz Maker,Google Forms a Microsoft Forms. Hlavními výhodami těchto aplikací je poskytování různých typů otázek, podpora velkého množství jazyků a technologií, přítomnost intuitivního rozhraní a co je nejdůležitější, možnost vkládání různých souborů a vzorců.

2.1.7 Závěr

Z analýzy trhu konkurentů vidíme, že v současné době na trhu již existuje velké množství aplikací, které umožňují vytvářet testy. Každý program má své výhody a nevýhody. Webová aplikace "Test&Study"by měla odstranit nedostatky zvažovaných konkurentů bez ztráty jejich výhod. Největším konkurentem, který může splnit i účel aplikace "Test&Study", je ProProfs Quiz Maker, tato aplikace je však komerční a většina jejích funkcí může být použita pouze po zakoupení placené verze. Webová aplikace "Test&Study"bude zase zcela zdarma pro všechny uživatele na internetu, což pro ni poslouží jako významná výhoda na konkurenčním trhu. Aplikace Google Forms a Microsoft Forms mají také vynikající pověst na trhu a jsou zcela zdarma. Umožňuje ale pouze tvorbu formulářů a ve funkční části budou výrazně zaostávat za mnou vyvíjenou aplikací.

Na základě získaných dat můžeme dojít k závěru jaké funkce by měla mít vyvíjená aplikace "Test&Study":

- implementovat do aplikace velké množství užitečných a zajímavých funkcí a poskytnout rozmanitost otázek;
- vytvořit podrobnou dokumentaci, tutoriály a nápovědy pro uživatele, aby nedocházelo k nedorozuměním a komplikacím.
- zajistit stabilitu a bezpečnost aplikace;
- je třeba se zaměřit na vytvoření uživatelsky příjemného, vizuálně atraktivního a zároveň intuitivního rozhraní;
- zamyslet se nad možností změny stylu a dekorace testů;
- aplikace bude v angličtině, protože je nejpopulárnějším jazykem na světě, to usnadní používání aplikace. V budoucnu se plánuje rozšíření i do dalších jazyků;
- aplikace bude zcela zdarma, což pozitivně ovlivní její konkurenceschopnost;
- vyvinout v aplikaci proces vytváření statistik odpovědi a jejich export v různých formátech;
- umožnit proces stahování celého testu ve formátu PDF;

2.2 Business analýza

2.2.1 Využití testování k ověření znalostí studentů

Testy jsou účinným prostředkem pro kontrolu kvality znalostí získaných studenty a operativního sledování pokroku ve výuce.

Testování umožňuje definovat oddíly, které představují největší složitost studia, a případně upravit proces učení v závislosti na výsledcích testů.

Použití této metody umožňuje učiteli získat informace o asimilaci konkrétního materiálu, aniž by trávil čas rozhovory se studenty nebo kontrolou písemné práce. Schopnost zkontrolovat a vyhodnotit znalosti celé třídy za 10-20 minut zlepšuje zpětnou vazbu, dělá ji pravidelnou. Systematická kontrola znalostí nejen přispívá k trvalé asimilaci vyučovacího předmětu, ale také vychovává vědomé postoj k učení, formuje přesnost, pracovitost, cílevědomost, aktivuje pozornost a rozvíjí schopnost analýzy u studentů. Při testovacích kontrolách jsou zajištěny rovné podmínky pro všechny studenty, tj. zvyšuje se objektivita prověřování znalostí. Konečně, tato metoda přináší rozmanitost do vzdělávací práce, zvyšuje zájem o předmět.

2.2.2 Výhody digitální testové formy kontroly znalostí

- automatizace zpracování výsledků;
- objektivita kontroly a výsledků testů;
- uspora studijního času při kontrole znalostí a hodnocení výsledků školení;
- rychlá kontrola kvality přípravy velkého počtu testovaných studentů na širokou škálu otázek;
- kontrola kvality asimilace teoretického učebního materiálu;
- učinnost a spolehlivost: možnost vytvoření mnoho verzí stejného testu pro zabránění podvádění;
- existence velkého množství programů, které usnadňují tvorbu testů;
- možnost provádění testů distanční bez nutnosti opustit svůj dům;
- provádění operativní diagnostiky úrovně asimilace učebního materiálu každým studentem;

2.2.3 Business cíle projektu

Business cíl je prokazatelný výsledek a zadané podmínky realizace celkového úkolu projektu.

BG01 - Digitalizace procesu testování.
Priorita: vysoká
Náročnost: vysoká

BG02 - Zrychlení a zjednodušení procesu vytváření testů pro učitele a studenty.
Priorita: vysoká
Náročnost: vysoká

BG03 - Usnadnění identifikaci mezer ve znalostech studentů.
Priorita: vysoká
Náročnost: střední

BG04 - Pomoc studentům a učitelům efektivněji připravit se na zkoušky.
Priorita: vysoká
Náročnost: střední

2.2.4 Business požadavky

Business požadavky jsou běžné úkoly, které musí vzniklý produkt vyřešit. Také určují význam projektu a zdůvodňují jeho potřebu.

BRQ01 - Vytváření webové aplikace pro testování.
Jako student/učitel chci mít možnost distanční vytvářet a účastnit se testů.

BRQ02 - Vytvoření uživatelsky přívětivého a intuitivní pochopitelného rozhraní pro aplikaci.
Jako student/učitel nechci dlouho řešit, jak program používat. Chci vidět vizuálně příjemné rozhraní.

BRQ03 - Poskytnutí možnosti vytvářet zajímavé a rozmanité testy.
Jako student/učitel chci mít možnost vytvářet zajímavé testy s různými typy otázek.

BRQ04 - Umožnění sestavování statistik a exportu výsledku testu.
Jako učitel chci mít možnost sestavit, zobrazit a exportovat statistiky testů absolvovaných studenty. Jako student chci mít možnost prohlédnout a exportovat výsledky absolvovaných testů.

2.2.5 SWOT analýza

SWOT analýza je zhodnocení silných a slabých stránek, příležitostí a rizik daného tématu nebo problematiky. [6] Dále je to univerzální analytická technika zaměřená na zhodnocení vnitřních a vnějších faktorů ovlivňujících úspěšnost organizace nebo nějakého konkrétního záměru (například nového produktu či služby).

SWOT je akronym z počátečních písmen anglických názvů jednotlivých faktorů: [7]

- Strengths - silné stránky, tedy v čem je aplikace dobrá.
- Weaknesses - slabé stránky, tedy v čem je špatná.
- Opportunities - příležitosti, tedy co lze využít.
- Threats - hrozby, tedy na co je nutné dávat pozor.

	SILNÉ STRÁNKY (S)	SLABÉ STRÁNKY (W)
Interní	• Aplikace může být použitá pro jakýkoli školní předmět • Studenti si mohou vytvářet vlastní testy pro samostudium • Možnost rychlého rozvoje systému • Pomoc při výuce pro učitele i žáky • Nízké náklady a snadná implementace • Snadné použití a pochopení • Škálovatelná aplikace	• Úzké zaměření aplikace • Omezené cílové publikum • Vyžaduje připojení k Internetu, které není vždy k dispozici nebo může mít špatnou kvalitu • Vytvoření testu pomocí aplikace trvá nějakou dobu • Složitá údržba stability aplikace
	PŘÍLEŽITOSTI (O)	**HROZBY (T)**
Externí	• Vysoká poptávka po aplikaci mezi uživateli kvůli neustále probíhajícím procesem digitalizace vzdělávání • Existence velkého množství vzdělávacích institucí, které by mohly používat aplikaci • Možnost rozvoje na globální úroveň • Rozvoj aplikace – přidání dalších funkcí • Vytvoření mobilní aplikace • Neustálé zlepšování kvality aplikace	• Velká konkurence, po podrobném průzkumu trhu se ukázalo, že podobných aplikací je hodně • Vznik nové konkurence • Nekvalitně vytvořená aplikace a problémy při použití • Nízká poptávka po aplikaci mezi uživateli • Pomalé připojení k internetu

Obrázek 2.1: SWOT analýza.

Závěr SWOT analýzy

V rámci návrhu řešení se zaměříme na minimalizaci slabých stránek a maximalizaci příležitostí, čímž podporujeme záměr spočívající ve vytvoření co nejpohodlnější a srozumitelné aplikace, v posílení příležitostí souvisejících s přilákáním uživatelů a rozvojem funkčnosti aplikace. Aplikujeme tedy strategii MIN-MAX. I přes úzké zaměření aplikace může tato slabost přispět ke zlepšení a neustálé modernizaci hlavních funkcí aplikace. Hrozbám a slabinám aplikace lze zabránit vytvořením konkurenční aplikace s velkým počtem různých a užitečných funkcí. To pomůže posílit pozici aplikace mezi konkurenty a přilákat uživatele. I přes úzké zaměření aplikace může tato slabost přispět ke zlepšení a neustálé modernizaci hlavních funkcí aplikace. Také se zamyslím a pokusím se aplikovat technologie, které umožní používat aplikaci bez připojení k internetu.

2.3 Analýza řešení

2.3.1 Požadavky na webovou aplikaci

Požadavky jsou specifikací toho, co musí být v systému implementováno. Popisují chování systému, jeho vlastnosti a atributy. Mohou také sloužit jako omezení v procesu vývoje systému. Požadavek musí být proveditelný, měřitelný, testovatelný, musí se vztahovat ke konkrétnímu byznys požadavku, nebo příležitosti a také musí být definovaný dostatečné detailně pro účely návrhu systému. [8]

Funkční požadavky

Funkční požadavky definují základní chování systému. Funkční požadavky mohou zahrnovat výpočty, technické detaily, manipulaci a zpracování dat a další specifické funkce, které definují, co má systém plnit. Funkční požadavky řídí aplikační architekturu systému.

Tabulka 2.2 popisuje funkční požadavky na aplikaci "Test&Study".

Tabulka 2.2: Funkční požadavky

ID	Název	Popis
		Správa uživatelů
FR-1	Uživatelské role	Systém bude podporovat role běžného uživatele (tvůrce a účastník testu) a adminu.
FR-2	Registrace uživatele	Aplikace bude podporovat registrace nového uživatele.
FR-3	Přihlášení uživatele	Aplikace umožní uživateli přihlásit se do vlastního účtu na základě emailu a hesla.
FR-4	Odhlášení uživatele	Aplikace umožní uživateli odhlásit se ze vlastního účtu.
FR-5	Úprava účtu	Aplikace umožní uživateli upravit své údaje ve vlastním účtu.
FR-6	Smazání účtu	Aplikace umožní uživateli smazat svůj účet.
FR-7	Automatické odhlášení	Aplikace automatické odhlásí uživatele z aplikace, pokud nebude po určitou dobu aktivní.
		Správa aplikace
FR-8	Zobrazení seznamu testů	Aplikace umožní uživateli zobrazit seznam svých testů, které vytvořil nebo má přístup.
FR-9	Zobrazení seznamu odpovědí	Aplikace umožní uživateli zobrazit seznam svých odpovědí, které vytvořil.
FR-10	Zobrazení seznamu nedávno vytvořených testů	Aplikace umožní uživateli zobrazit seznam nedávno vytvořených testů, které vytvořil sám nebo má přístup.
FR-11	Zobrazení dokumentaci k aplikaci	Aplikace umožní uživateli zobrazit dokumentaci k aplikaci.
FR-12	Hledání testu v seznamu testů podle tématu	Aplikace umožní uživateli hledat test v seznamu jeho testů podle tématu.

FR-13	Hledání testu v seznamu testů podle názvu	Aplikace umožní uživateli hledat test v seznamu jeho testů podle názvu.
FR-14	Hledání testu v seznamu odpovědí	Aplikace umožní uživateli hledat test v seznamu jeho odpovědí.
FR-15	Filtrování seznamu testů	Aplikace umožní uživateli filtrovat seznam jeho testů.
FR-16	Smazání testu ze seznamu	Aplikace umožní uživateli smazat určitý test ze seznam testů, které vytvořil nebo má přístup.
FR-17	Smazání odpovědi ze seznamu	Aplikace umožní uživateli smazat určitý odpověď ze seznamu jeho odpovědí.
FR-18	Změna směru testů v seznamu testů	Aplikace umožní uživateli změnit směr seznamu testů.
	Správa testu	
FR-19	Zúčastnění v testu	Aplikace umožní uživateli zúčastnit se testu.
FR-20	Přidání otázky do testu	Aplikace umožní tvůrce skupiny přidávat nové otázky do testu.
FR-21	Přidání obrázku do testu	Aplikace umožní tvůrce skupiny přidávat obrázek do testu.
FR-22	Přidání tématu testu	Aplikace umožní tvůrce skupiny přidávat tématu testu.
FR-23	Úprava testu	Aplikace umožní tvůrce testu ho upravit.
FR-24	Změna barvy testu	Aplikace umožní tvůrce testu zaměnit barvu testu.
FR-25	Změna informace o testu	Aplikace umožní tvůrce testu zaměnit informace o testu.
FR-26	Vytvoření testu	Aplikace umožní uživateli vytvořit test.
FR-27	Smazání účastníku z účastníků testu	Aplikace umožní tvůrce testu odebrat účastníka z účastníků testu.
FR-28	Smazání všech členů testu	Aplikace umožní tvůrce testu smazat všechny členy testu.
FR-29	Smazání obrázku testu	Aplikace umožní tvůrce testu smazat obrázek testu.
FR-30	Smazání testu ze svých testů	Aplikace umožní uživateli smazat test ze svých testů.
FR-31	Ukončení testu	Aplikace umožní uživateli ukončit test.
FR-32	Exportování všech testovacích odpovědí do CSV	Aplikace umožní tvůrce testu všechny testovací odpovědi do CSV.
FR-33	Exportování všech testovacích odpovědí do Excel	Aplikace umožní tvůrce testu všechny testovací odpovědi do Excel.
FR-34	Exportování výsledků testu ve formátu PDF	Aplikace umožní uživateli uložit test ve formátu PDF.
FR-35	Opuštění testu	Aplikace umožní uživateli opustit test bez dokončení.
FR-36	Smazání testu	Aplikace umožní tvůrce testu smazat test.
FR-37	Spuštění testu ze špatných odpovědi	Aplikace umožní uživateli spustit testu ze špatných odpovědi.

FR-38	Exportování testu ve formátu PDF	Aplikace umožní tvůrce testu uložit test ve formátu PDF.
FR-39	Ukládání testu	Aplikace umožní tvůrce testu přidávat ukládat test.
FR-40	Sdílení testu	Aplikace umožní tvůrce testu sdílet ho s dalšími uživateli aplikace.
FR-41	Ukončení testu kvůli vyčerpání času	Aplikace automaticky ukončit test, pokud tvůrce testu nastavit časovač na určitou dobu.
FR-42	Zobrazení seznamu odpovědí	Aplikace umožní uživateli zobrazit seznam odpovědí.
FR-43	Zobrazení seznamu účastníků testu	Aplikace umožní tvůrce testu zobrazit seznam účastníků testu.
FR-44	Zobrazení seznamu odpovědí účastníků testu	Aplikace umožní tvůrce testu zobrazit seznam odpovědí účastníků testu.
FR-45	Zobrazení odpovědi účastníku testu	Aplikace umožní tvůrce testu zobrazit odpověď účastníku testu.
FR-46	Zobrazení informace o testu	Aplikace umožní uživateli zobrazit informace o testu.
FR-47	Zobrazení náhledu testu	Aplikace umožní tvůrce testu zobrazit náhled testu.
FR-48	Zobrazení svého odpovědi	Aplikace umožní uživateli zobrazit svůj odpověď.
	Správa otázky	
FR-49	Vytvoření otázky	Aplikace umožní tvůrce testu vytvořit otázky v testu.
FR-50	Smazání otázky	Aplikace umožní tvůrce testu smazat otázku.
FR-51	Úprava otázky	Aplikace umožní tvůrce testu upravit otázku.
FR-52	Kopírování otázky	Aplikace umožní tvůrce testu kopírovat otázku.
FR-53	Odpovídání na otázku	Aplikace umožní uživateli odpovědět na otázku.
FR-54	Přidání možností odpovědí na otázku	Aplikace umožní tvůrce testu přidávat možností odpovědi.
FR-55	Přidání správného odpovědi na otázku	Aplikace umožní tvůrce testu přidávat správné odpovědí k otázce.
FR-56	Přidání poznámky k otázce	Aplikace umožní tvůrce testu přidávat poznámku k otázce.
	Správa odpovědí	
FR-57	Zobrazení svých odpovědí	Aplikace umožní uživateli zobrazit své odpovědí.
FR-58	Zobrazení celého odpovědi (response)	Aplikace umožní uživateli zobrazit svůj celý odpověď.
FR-59	Zobrazení odpovědi (response) účastníku testu	Aplikace umožní tvůrce testu zobrazit odpověď účastníku testu.
FR-60	Zobrazení správných odpovědí	Aplikace umožní uživateli zobrazit správných odpovědí.
FR-61	Zobrazení všech odpovědí	Aplikace umožní uživateli zobrazit všech svých odpovědí.

FR-62	Exportování odpovědi ve formátu PDF	Aplikace umožní uživateli uložit svůj odpověď ve formátu PDF.
FR-63	Ukládání odpovědi	Aplikace umožní uživateli aktualizovat a uložit odpověď.
FR-64	Označení odpovědi jako správné/nesprávné	Aplikace umožní uživateli označit odpověď jako správný/nesprávný
FR-65	Smazání odpovědi	Aplikace umožní uživateli smazat svůj odpověď.
	Správa Adminu	
FR-66	Zobrazení seznamu uživatelů	Aplikace umožní adminu zobrazit seznam uživatelů.
FR-67	Zobrazení seznamu testů	Aplikace umožní adminu zobrazit seznam testů.
FR-68	Zobrazení seznamu odpovědí	Aplikace umožní adminu zobrazit seznam odpovědí.
FR-69	Zobrazení seznamu otázek	Aplikace umožní adminu zobrazit seznam otázek.
FR-70	Smazání uživatele	Aplikace umožní adminu smazat uživatele.
FR-71	Úprava uživatele	Aplikace umožní adminu upravit uživatele.
FR-72	Vytvoření uživatele	Aplikace umožní adminu vytvořit uživatele.
FR-73	Správa odpovědí	Aplikace umožní adminu spravovat odpovědí.
FR-74	Správa otázky	Aplikace umožní adminu spravovat otázky.
FR-75	Správa uživatelů	Aplikace umožní adminu spravovat uživatelé.
FR-76	Správa testu	Aplikace umožní adminu spravovat testy.
FR-77	Správa aplikace	Aplikace umožní adminu spravovat aplikace.

Nefunkční požadavky

Funkční požadavky jsou podporovány nefunkčními požadavky, které omezují návrh nebo implementaci (jako jsou požadavky na výkon, bezpečnost nebo spolehlivost). Zatímco funkční požadavky definují, co systém dělá nebo nesmí dělat, nefunkční požadavky specifikují, jak to má systém dělat. Nefunkční požadavky nemají vliv na základní funkčnost systému a řídí technickou architekturu systému.

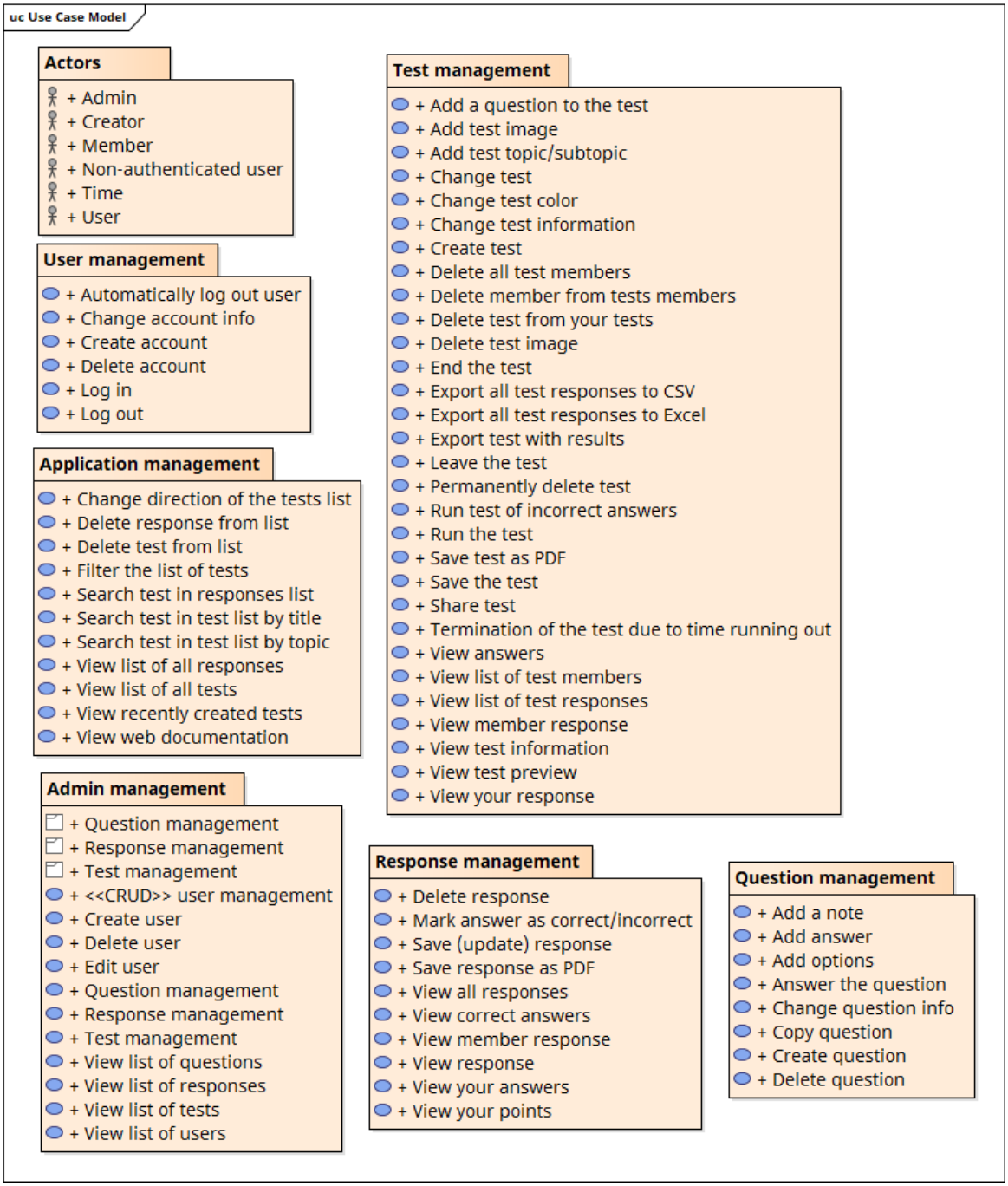
Tabulka 2.3 popisuje nefunkční požadavky na aplikaci "Test&Study.

Tabulka 2.3: Nefunkční požadavky

ID	Název	Popis
NFR-1	Intuitivní a přívětivé uživatelské webové rozhraní	Systém bude poskytovat přívětivé a intuitivní webové rozhraní splňující zásady UI/UX designu pomocí heuristických doporučení.
NFR-2	Podpora prohlížečů	Webová aplikace bude fungovat a správně se zobrazovat na aktuálních verzích Microsoft Edge, Firefox, Google Chrome, Safari, Opera.
NFR-3	SPA	Webové uživatelské rozhraní bude implementováno na principu Single Page Application pro zajištění plynulých přechodů mezi jednotlivými stránkami.
NFR-4	Persistence dat	Veškerá aplikační data budou ukládána do databáze MongoDB a denně zálohována.
NFR-5	Dostupnost	Systém bude dostupný v 99,9 % času (mimo plánovanou údržbu).
NFR-6	Obrana proti CSRF a XSS útokům	Systém bude zabezpečený proti CSRF a XSS útokům.

2.3.2 Diagram případů užití a aktérů

Diagram 2.2 ukazuje všechny balíčky, obsahující případy užití a aktéry aplikace. Aplikace obsahuje 6 aktérů a 6 skupin případů užití: správa uživatelů, správa aplikace, správa testu, správa otázky, správa odpovědí, správa Adminu.

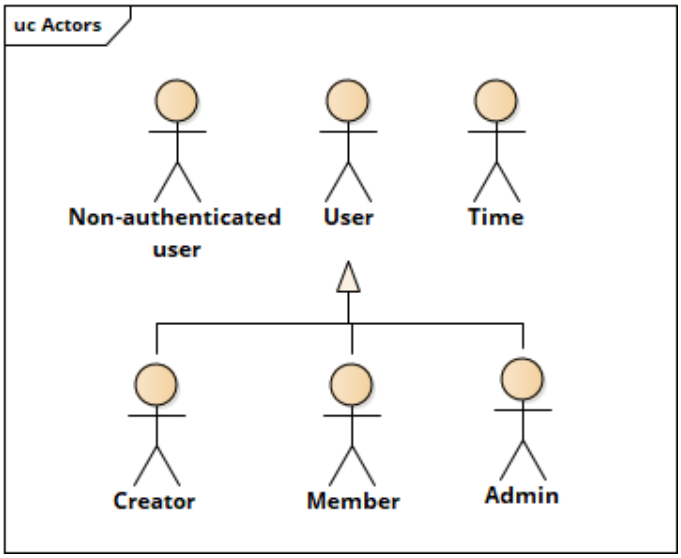


Obrázek 2.2: Diagram případů užití a aktérů.

2.3.3 Seznam uživatelů

Aktér je role, která komunikuje s jednotlivými případy užití. V této roli může být obsazen uživatel nebo externí systém. [9]

- Nepřihlášený uživatel (Non-authenticated user) – anonymní uživatel, který se dosud nezaregistroval nebo se nepřihlásil do svého účtu na webu.
- Uživatel (User) – běžný uživatel aplikace.
- Tvůrce (Creator) – běžný uživatel aplikace, který vytvořil test nebo skupinu.
- Účastník (Member) – běžný uživatel aplikace, který se účastní testu nebo jsou členem skupiny.
- Admin – správce celého systému, má přístup k seznamu uživatelů, testů, skupin, otázek a témat.
- Čas (Time) – časovač pro dokončení nějaké akce.



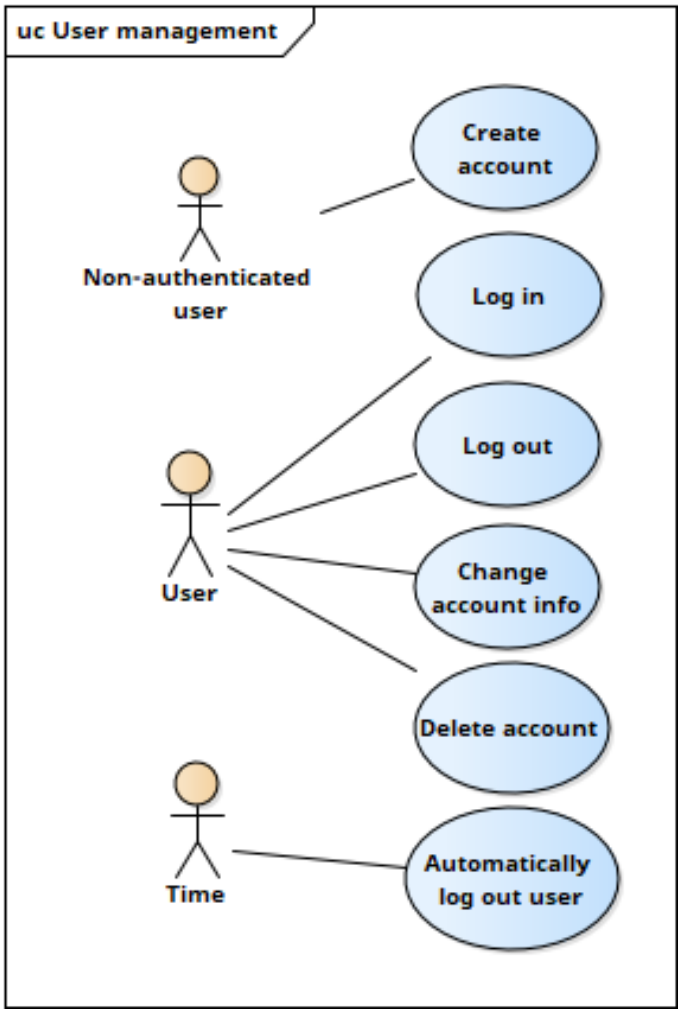
Obrázek 2.3: Seznam uživatelů

2.3.4 Případy užití

Případ užití (anglicky use case) - seznam kroků, který popisuje jeden nebo více způsobů interakce systému s koncovým uživatelem nebo jiným systémem za účelem dosažení konkrétního cíle.
Systém dokáže reagovat na vnější požadavky Aktérů, a taky může i sám iniciovat interakci. Případy užití jsou nejdůležitějším nástrojem pro modelování požadavků, aby bylo možné specifikovat funkčnost vyvíjeného softwaru nebo systému jako celku. Případ použití popisuje „kdo“ a „co“ může dělat s daným systémem nebo co systém může dělat s "někým"nebo "něčím". Technika případu užití se používá k identifikaci funkčních požadavků na chování systému. Každý případ použití ilustruje scénáře chování prostřednictvím jednoho nebo více funkčních požadavků. Mohou obsahovat scénáře chování systému, které popisuje všechny způsoby, kterými mohou uživatelé ovládat software nebo systém. Text scénáře doplňuje grafickou prezentaci případů užití v podobě popisu sledu kroků či akcí, po kterých může uživatel při interakci se systémem dosáhnout požadovaného cíle. [10] [11]

Správa uživatelů

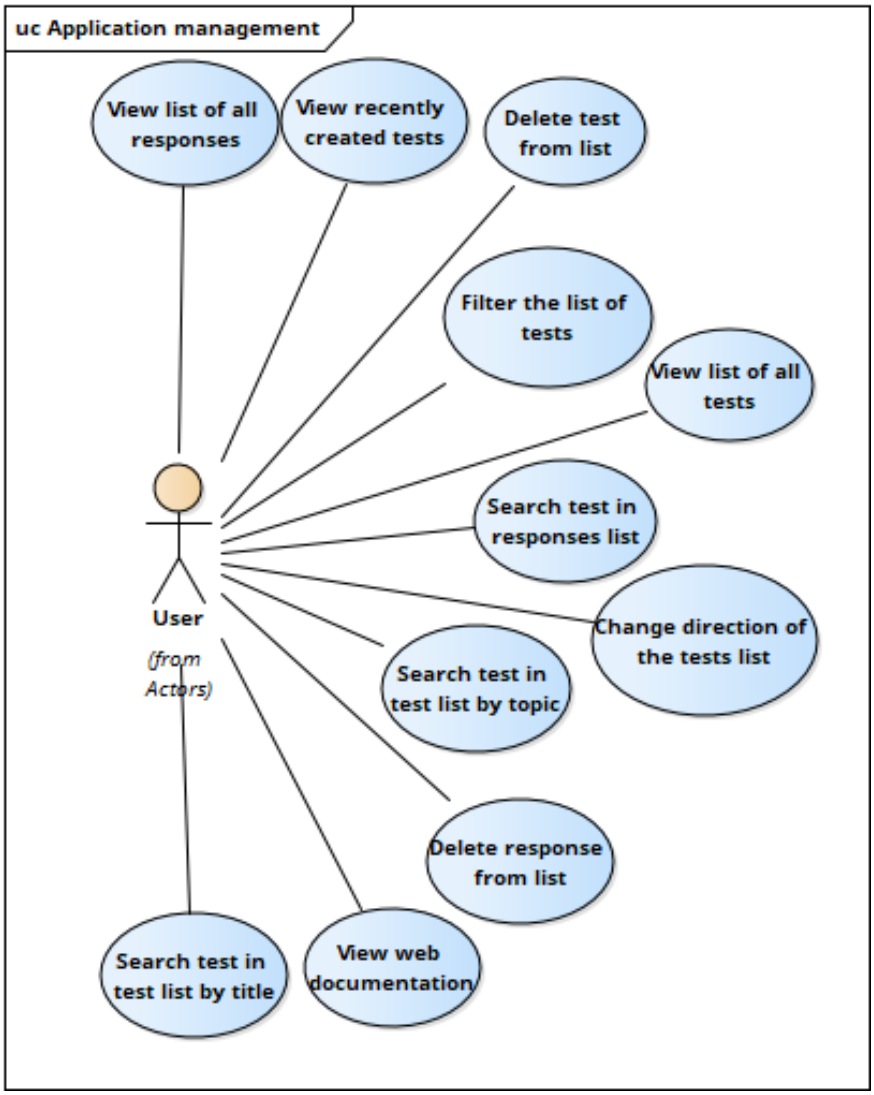
Diagram 2.4 znázorňuje akce, které může uživatel provádět se svým účtem.



Obrázek 2.4: Správa uživatelů.

Správa aplikace

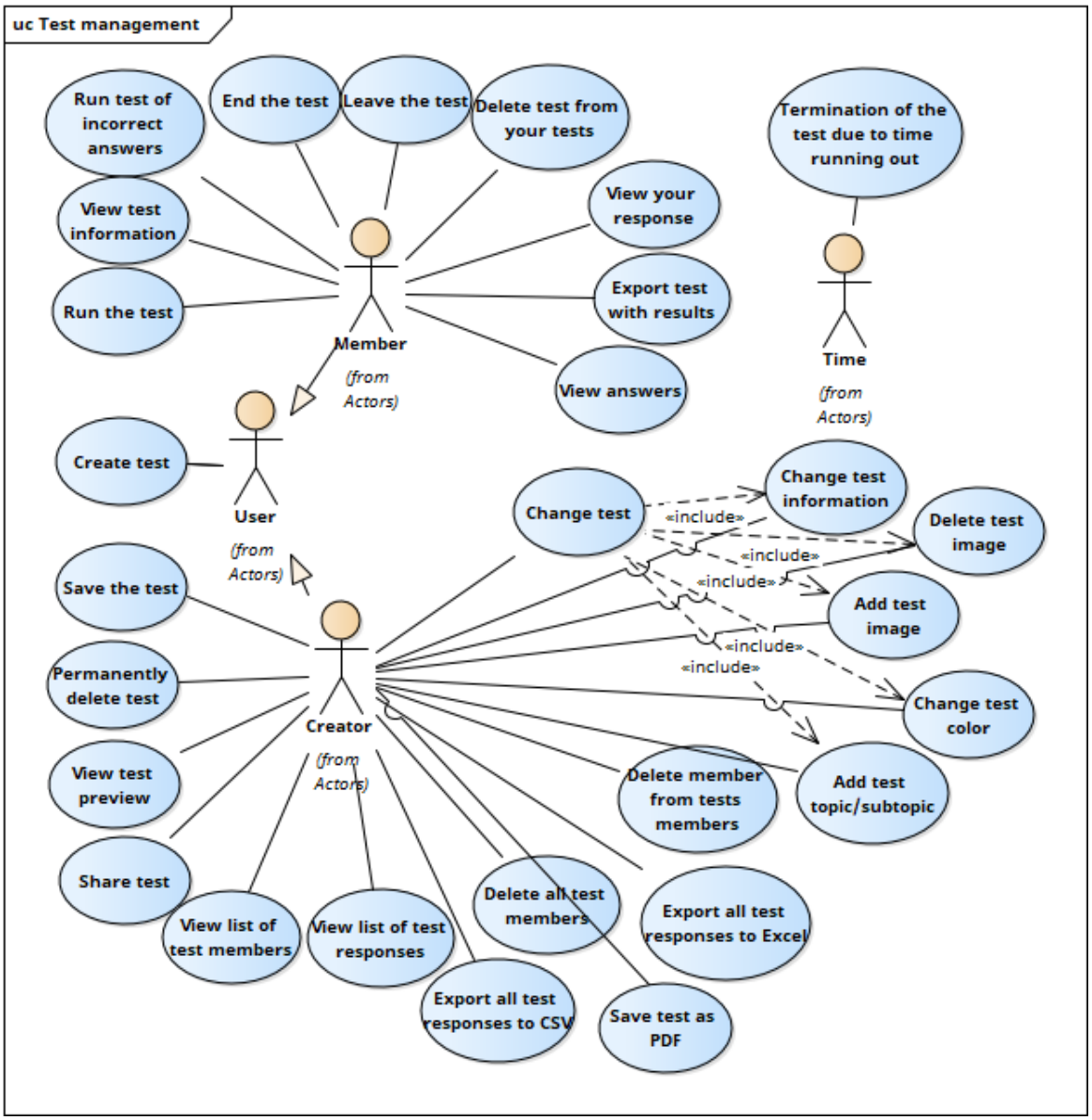
Diagram 2.5 popisuje základní aktivity uživatelů v aplikaci.



Obrázek 2.5: Správa aplikace.

Správa testu

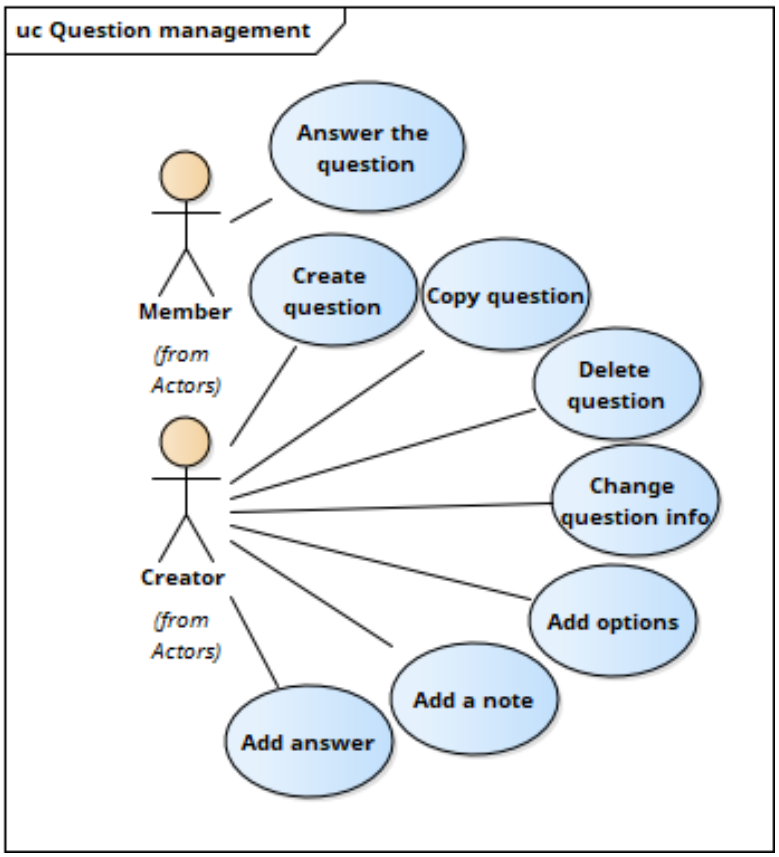
Diagram 2.6 znázorňuje aktivity související s testy, které může uživatel s konkrétní rolí provádět v aplikaci "Test&Study".



Obrázek 2.6: Správa testu.

Správa otázky

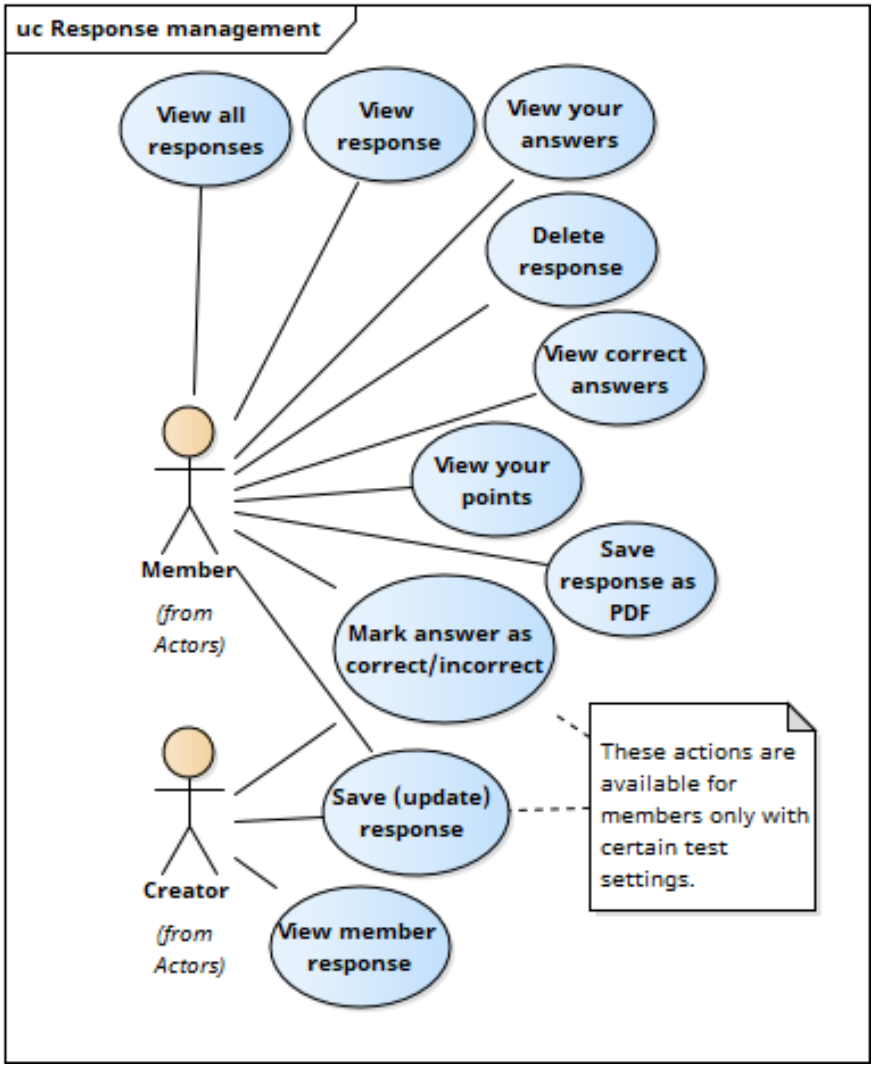
Diagram 2.7 znázorňuje aktivity související s testovacími otázkami, které může uživatel s konkrétní rolí provádět v aplikaci "Test&Study".



Obrázek 2.7: Správa otázky.

Správa odpovědí

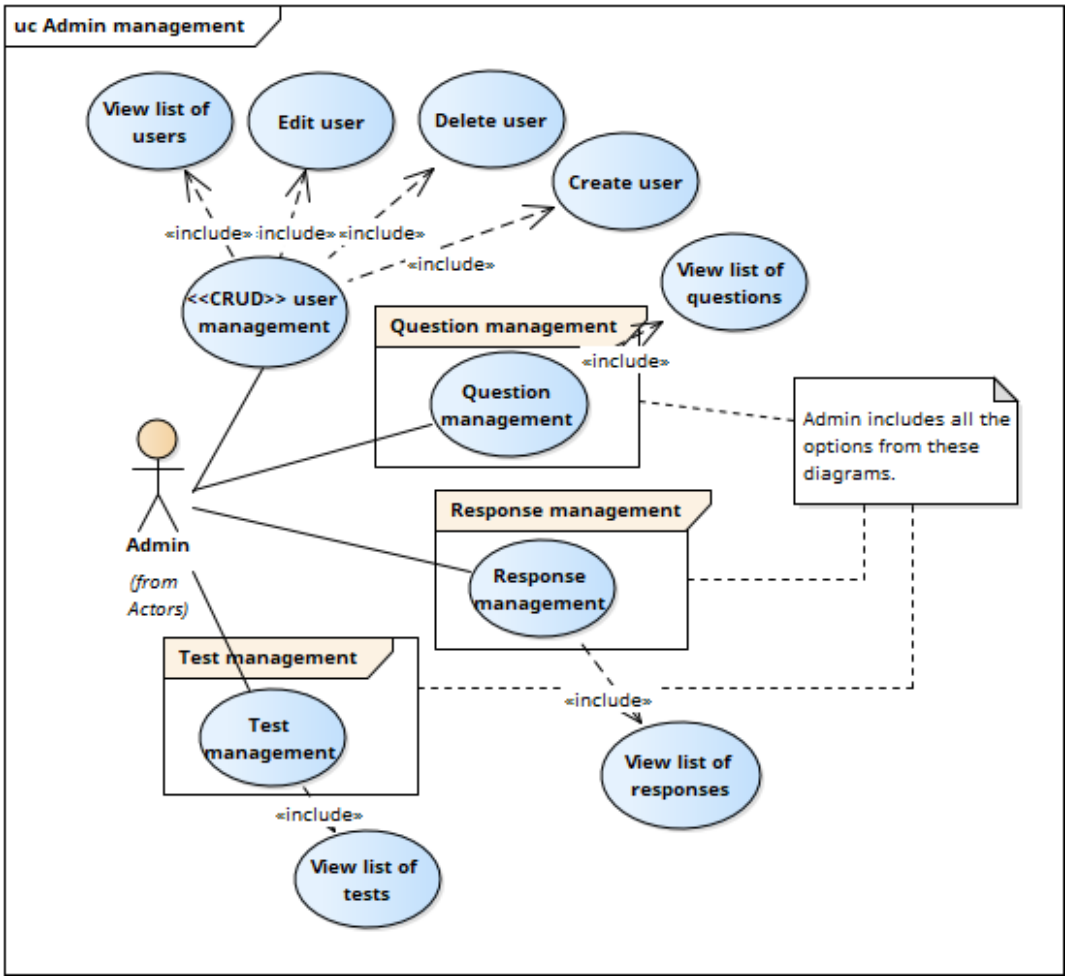
Diagram 2.8 znázorňuje aktivity související s odpověďmi na konkrétní test, které může uživatel se specifickou rolí provádět v aplikaci "Test&Study".



Obrázek 2.8: Správa odpovědí.

Správa Adminu

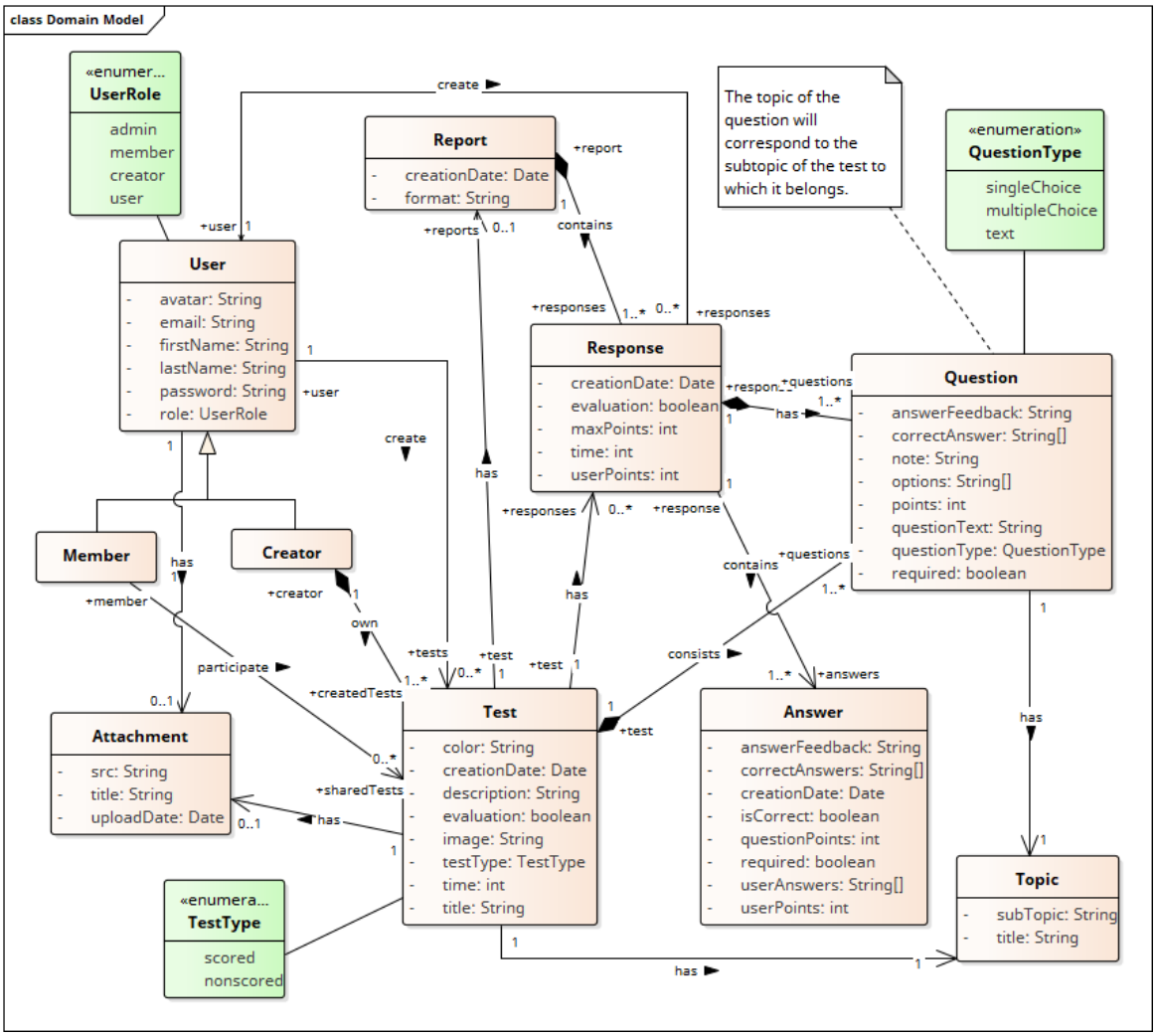
Diagram 2.8 znázorňuje funkce správce v aplikaci.



Obrázek 2.9: Správa Adminu.

2.3.5 Diagram tříd

Diagram tříd je strukturální diagram, který demonstruje obecnou strukturu hierarchie tříd systému, jejich interakce, atributy (pole), metody, rozhraní a vztahy mezi nimi. [10]



Obrázek 2.10: Diagram tříd

Seznam entit aplikace:

- User - uživatel aplikace. Může mít hned několik různých rolí.
- UserRole - role uživatele v aplikaci:
  - Admin - správce aplikace.
  - User - uživatel dostává tuto roli při vytváření účtu v aplikaci.
  - Creator - uživatel dostává tuto roli při vytváření testu.
  - Member - uživatel dostává tuto roli při účasti v testu.
- Attachment - mediální soubory, například profilová fotografie uživatele nebo obrázek testu.

- Test - test vytvořený v aplikaci. Test může mít jeden ze dvou typů testu. Test může obsahovat několik otázek s různými typy. Test může mít také téma a podtéma.
- TestType - typ testů:
  - Scored - bodovaný test. V takovém testu každá otázka může být hodnocena od 0 do 5 bodů. Na konci testu účastník testu obdrží výsledek a uvidí, kolik bodů z maximálního možného počtu získal.
  - Nonscored - nebodovaný test. V takovém testu otázky nemají bodové hodnocení.
- Question - otázka testu. V aplikaci jsou k dispozici tři druhy otázek.
- QuestionType - typ otázky:
  - singleChoice - výběrová otázka s jednou správnou odpovědí.
  - multipleChoice - výběrová otázka s více správnými odpověďmi.
  - text - otevřená otázka. Respondent má k dispozici textové pole, do kterého může napsat cokoliv.
- Response - reprezentuje odpověď účastníka testu. Zahrnuje jeho odpovědi na každou otázku a potřebné informace o testu.
- Answer - odpověď uživatele na konkrétní testovací otázku.
- Topic - téma a podtéma testu nebo otázky.
- Report - reprezentuje exportované soubory, například Excel nebo CSV dokument s odpověďmi účastníků testu na konkrétní test, PDF dokument s obsahem testu nebo odpovědi uživatele.

2.3.6 Základní wireframy

Vzhledem k tomu, že wireframe je rozložení webové stránky, které ukazuje, jaké prvky rozhraní budou na klíčových stránkách, byly proto vytvořeny v angličtině, aby jasně odrážely rozhraní aplikace, která bude vytvořena v tomto jazyce.

Podrobné wireframy jsou uložené v Priloha A - Wireframes.

2.3.7 Sekvenční diagram

Pro lepší pochopení a vizuální znázornění procesu vytváření testu jsem vytvořila Sekvenční diagram, uložený v Priloha B - Sekvenční diagram.

Proces vytváření testu:
Uživatel klikne na tlačítko vytvořit test. Systém zobrazí modální okno pro jeho vytváření. Uživatel vyplní název, popis a typ testu, pak systém zobrazí stránku pro úpravu testu, kde uživatel přidá otázky a klikne tlačítko uložit test. Systém uloží vytvořený test.

Spuštění:

1. Po vytvoření účtu/po přihlášení z hlavní stránky aplikace stisknutím tlačítka “Create test“.
2. Po vytvoření účtu/po přihlášení v panelu testy stisknutím tlačítka “Create test“.

Scénář:

1. Uživatel klikne na tlačítko “Create test”.
2. Systém načte typy testů z databáze a zobrazí dialog pro vytvoření testu.
3. Uživatel vyplní všechna potřebná políčka. IF Uživatel klikne tlačítko “Create test” THEN pokračuj ELSE GOTO hlavní stránka aplikace.
4. Systém vytvoří nový test s unikátním id a zobrazí stránku pro úpravu testu.
5. IF Uživatel klikne na tlačítko “Add question” THEN
   a. Systém načte z databázi všechny typy otázek a přidá komponentu pro novou otázku.
   b. Uživatel výplni potřebné údaje pro otázku.
6. IF Uživatel klikne tlačítko “Preview” THEN Systém zobrazí náhled testu ELSE pokračuj.
7. IF Uživatel klikne tlačítko “Save as PDF” THEN Systém nabídne uložení testu jako PDF souboru ELSE pokračuj.
8. Uživatel klikne na tlačítko “Save test”.
9. Systém uloží vytvořený test.

Část III

Návrh a analýza architektury aplikace

Kapitola 3

Návrh a analýza architektury aplikace

V první části této sekci jsou analyzovány a porovnávány mezi sebou technologie, které mohou být použity k vývoji ompletní webové aplikace "Test&Study". Na základě těchto údajů byly v druhé části sekce vybrány ty nejvhodnější, které budou schopny splnit všechny požadavky na aplikaci a pomohou ji rychle a snadno vyvíjet. Obsahuje také grafy, které vizualizují architekturu aplikace.

3.1 Analýza relevantních technologií

V této sekci jsou zváženy a porovnány všechny možné technologie, které lze použít k vytvoření plnohodnotné webové aplikace, která splňuje požadavky popsané v analýze aplikace. Aplikace, kterou vytvářím, bude obsahovat 3 části: klientská část, serverová část a databáze.

3.1.1 Databáze

V IT prostředí se tradičně rozlišují dva hlavní typy databází: relační (SQL) a nerelační (NoSQL) databáze. Rozdíly mezi nimi jsou, jak jsou navrženy, jaké typy dat podporují, jak ukládají informace.

Příkladem relační databázy mohou být MySQL nebo PostgreSQL.
Příkladem NoSQL databázy může být MongoDB.

V této části porovnám dvě možná řešení a k tomu použiji informace z oficiálního zdroje [12] Protože nejpopulárnější NoSQL databáze je MongoDB, rozhodla jsem se tuto možnost porovnat s SQL databáze.

Hlavní rozdíly MongoDB od SQL databází:

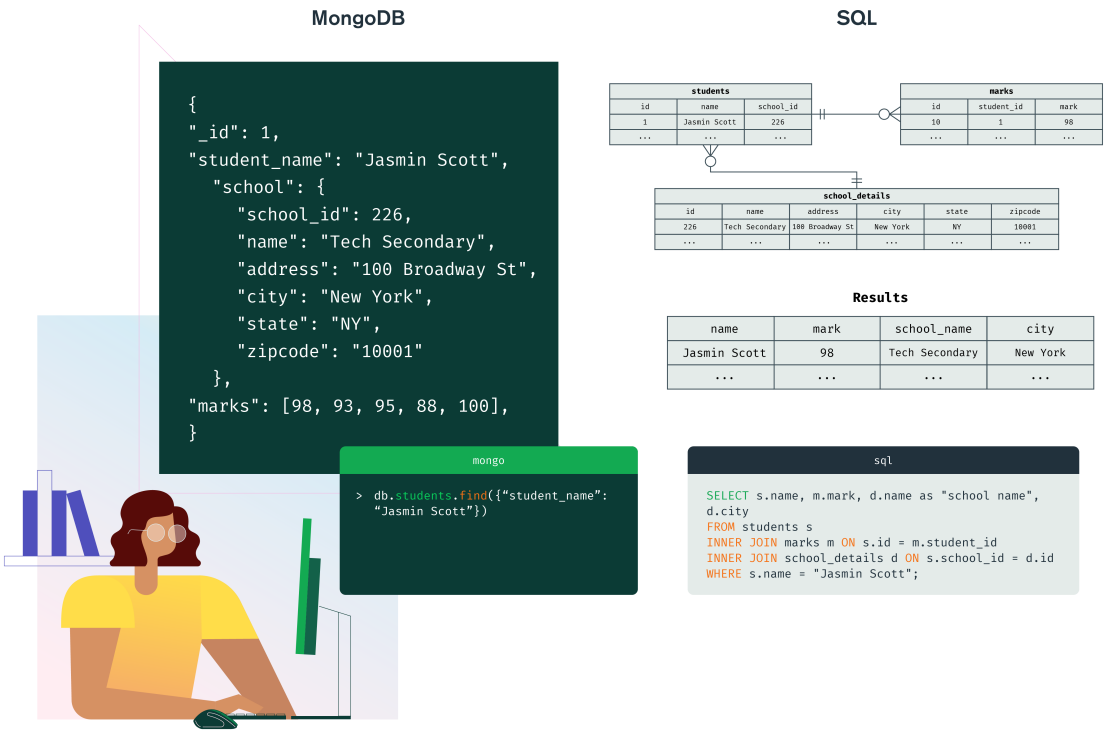
- Reprezentace dát: V MongoDB data jsou prezentovány jako sbírky JSON dokumentů, a v SQL databázích jako řádky a tabulky. MongoDB umožňuje spravovat data jakékoli struktury, nejen definované předem tabulkové struktury, díky čemuž je použití pohodlnější.
- Vkládání dat: SQL databází nenabízí žádné možnosti pro vkládání dat. Můžete použít příkazy JOIN, ale mohou způsobit zvětšení velikosti tabulek a přidání zbytečných polí. Spojení také může být náročné na čas a výkon.
- Provádění změn v databázi: v databázích SQL je třeba vytvořit databázové schéma a datové vztahy před naplněním databáze daty. Většina změn ve schématu vyžaduje proceduru migrace dat, která může převést databázi do offline

režimu nebo snížit výkon aplikace, když je spuštěna. Změna struktury po načtení dat je velmi obtížná. v MongoDB není potřeba datovou strukturu plánovat předem a je mnohem jednodušší ji měnit.

- Jazyk požadavku: relační databází používá SQL, zatímco nerelační DB MongoDB používá MQL, dotazovací jazyk MongoDB.
- Rychlost a výkon: MongoDB je rychlejší než SQL databází. SQL databází je poměrně pomalý ve srovnání s NoSQL při zpracování velkého množství dat.
- Propojení s datovým schématem: v MongoDB může každý objekt sbírky obsahovat různá pole, zatímco v SQL databázích jsou všechny tabulky přísně typizované schéma.
- Kompatibilita: MongoDB je mnohem jednodušší používat s knihovnami JavaScriptu, jako je Node.js, React.js a Express.js. MongoDB prezentuje data jako kolekci dokumentů, nikoli jako tabulky propojené cizími klíči. To umožňuje různé typy dat přenášených přes internet bezpečně ukládat a přistupovat k nim ve webových aplikacích pomocí Node.js.

Tento seznam byl sestaven na základě informací ze zdroje [12] uvedeného v části Literatura a zdroje 80.

Na obrázku 3.1 je vizuálně znázorněn pohled porovnávaných databází.



Obrázek 3.1: Pohled MongoDB a SQL databází [12]

Výsledek: pro vytvoření aplikace byla vybrána No-SQL databaza MongoDB, protože umožňuje rychleji vytvářet aplikace, zpracovávat velmi rozmanité datové typy a efektivnější spravovat aplikací.

3.1.2 Back-end – serverová část

Back-end je vnitřní část webu, která se nachází na serveru a je skryta před uživateli. Pro jeho vývoj lze použít různé jazyky, například PHP, JavaScript (Node.js), Java. Tyto jazyky jsem se rozhodla zvážit jako alternativní možnosti implementace serverové části aplikace.

Porovnání PHP a Node.js:

- Node.js podporuje širokou škálu databází, včetně NoSQL, jako je MongoDB. Kromě toho Node.js dobře funguje s běžnými a relačními databázemi, a to i s komplexními, jako jsou grafické databáze. PHP dobře spolupracuje s běžnými a relačními databázemi, jako je MySQL, ale práce s NoSQL je obtížná. Nastavení NoSQL pomocí PHP je zdlouhavé a náročné.
- Node.js je mnohem rychlejší než PHP, a to díky použití rychlého V8 (JavaScript engine), dlouhodobému připojení k serveru a funkce zpětného volání, které zpracovávají mnoho požadavků současně (asynchronně). PHP je více stará technologie, která pracuje se synchronním kódem a blokuje nové požadavky, dokud nedokončí stávající. Tato funkce zpomaluje celý systém a negativně ovlivňuje uživatelské zkušenosti.
- Node.js vyžaduje mnohem více řádků kódu k provedení stejné funkce jako PHP.
- Použití Node.js na rozdíl od PHP umožňuje vytvářet škálovatelné aplikace, které maximalizují využití paměti počítače a procesoru při provádění různých souběžných požadavků.

Tento seznam byl sestaven na základě informací ze zdroje [13] uvedeného v části Literatura a zdroje 80.

Porovnání Java a Node.js:

- Java je z hlediska svého výkonu velmi rychlá. Použití kompilátorů však trochu zpomaluje jeho výkon. Kromě toho je Java známá svou funkcí uvolňování paměti, která může být zároveň výhodou i nevýhodou. Node.js známý pro svou funkci bez ukládání do vyrovnávací paměti, jednoduše zobrazuje data v částech, a proto výrazně urychluje dobu provádění. Node.js je proto produktivnější než Java.
- Node.js je známý pro svou flexibilitu a rychlý vývoj. Přesto je Java přesným opakem Node.js od první technologie trvá mnohem více času na vývoj kvůli tomu, že je rigidní jazyk.
- Java je mnohem stabilnější než Node.js díky objektově orientovanému přístupu a paradigmatům, která existují už desítky let.
- Obě technologie vám mohou pomoci vytvářet vysoce škálovatelné aplikace. Ale Java je zde lepší, protože umožňuje vestavěný multithreading a paralelní výpočty. Aplikace Node.js mají také tuto nabídku, ale nejsou tak dobře navržené.

Tento seznam byl sestaven na základě informací ze zdroje [14] uvedeného v části Literatura a zdroje 80.

Porovnání Java a PHP:

- Java je považován za bezpečnější jazyk ve srovnání s PHP. Má více vestavěných bezpečnostních prvků, zatímco vývojáři PHP musí navíc používat nějaké frameworky.

- PHP trvá určitou dobu, než spustí každý požadovaný bajtkód.Nejnovější verze PHP se zaměřují na snížení spotřeby paměti a optimalizaci psaní kódu, ale Java však všechny zastíní svou optimalizační taktikou. Jedná se o předběžný programovací jazyk, který nezanechává žádný prostor pro zpožděné zpracování. Java má proto vysoký výkon ve srovnání s PHP.
- Zatímco nejnovější verze PHP umožňují vývojářům urychlit proces psaní aplikací, ale však psaní aplikace v Javě je poměrně rychlejší.
- Dobře napsaný a rozšířený kód v Javě slibuje vysokou úroveň stability ve srovnání s PHP aplikacemi s krátkými kódy.
- Programovací jazyk Java je schopen komunikovat s velkým počtem dalších systémů, včetně peer-web-services, databáze a mainframe back-endů vše ve stejné aplikaci. Navíc díky lepší podpoře nástrojů jsou frameworky Java jednodušší k instalaci, jsou dobře zdokumentovány a testovány.
- Pokud je třeba kódovat stejný software v PHP a Javě, kódování v PHP bude trvat méně času než Java.

Tento seznam byl sestaven na základě informací ze zdrojů [15], [16] uvedených v části Literatura a zdroje 80.

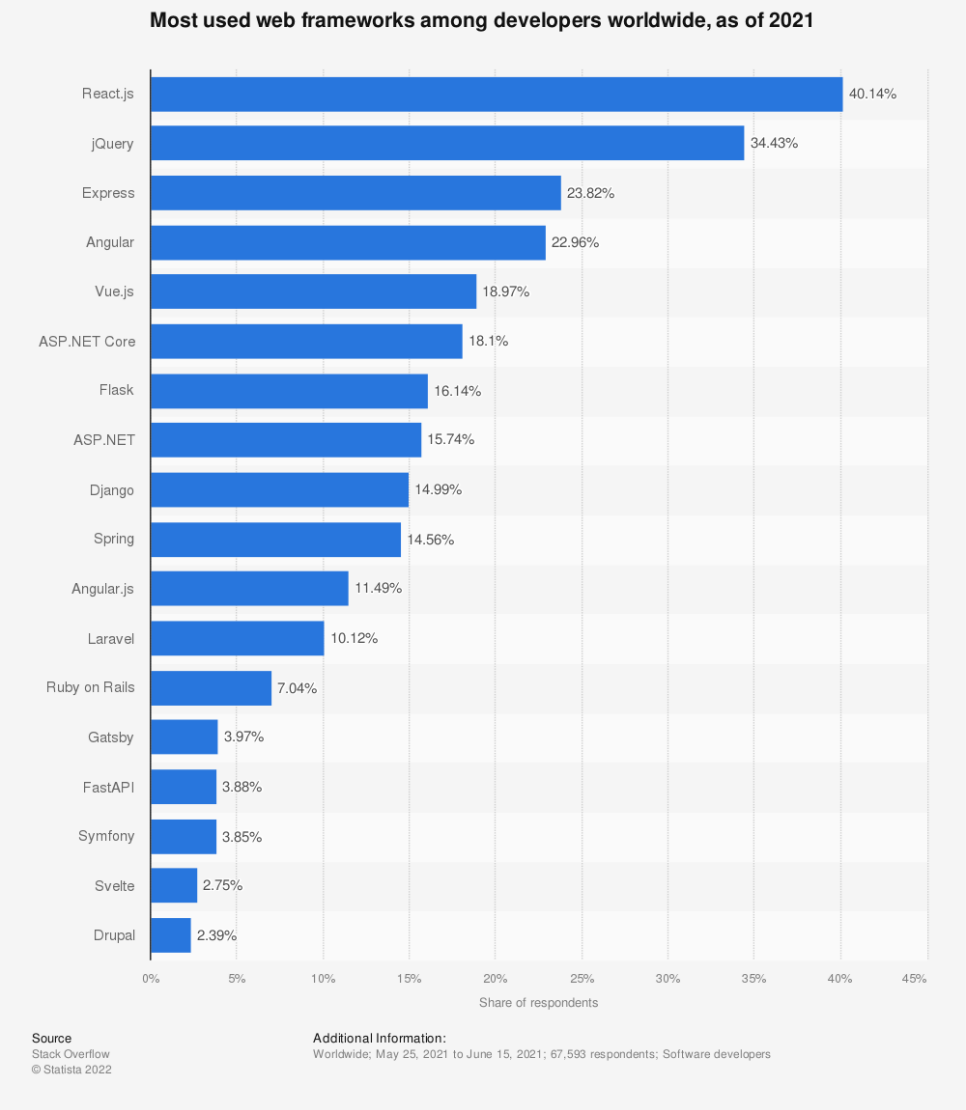
Výsledek: na základě toho, že mám více zkušeností s používáním Node.js a je to nejlepší varianta podle analýzy, pro vytvoření serverové části aplikace bude použita softwarová platforma Node.js.

3.1.3 Front-end – klientská část

Frontend je klientská část produktu, rozhraní, se kterým komunikuje uživatel. V případě webu, je to vše, co prohlížeč dokáže číst, zobrazovat a/nebo spouštět pomocí HTML, CSS a JavaScriptu..

Na základě svých zkušeností a průzkumu vývojářské platformy Stack Overflow jsem se rozhodla zvážit takové populární frameworky jako React.js, Angular a Vue.js.

Žebříček je znázorněn na obrázku 3.2



Obrázek 3.2: Žebříček nejpoužívanějších frameworků 2021 [17]

Každý framework je založen na komponentách a umožňuje rychle vytváření funkčního uživatelského rozhraní.

React VS Vue:

- React má výhodu oproti Vue díky snadné škálovatelnosti. Možnost používání komponent vícekrát zvyšuje škálovatelnost aplikací React. Kvůli dynamické architektuře Vue potřebuji využití knihoven a mixinů pro překonání omezení škálování.
- Pro rendering stanky Vue používá HTML šablony a JSX, zatímco React používá pouze JSX, což je v podstatě rozšíření, které umožňuje vkládat HTML přímo do JS kódu. JSX může zjednodušit mnoho složitých úkolů.
- Když se stav komponenty React změní, všechny komponenty v jejím podstromu budou také znovu vykresleny (re-render). Ve Vue jsou však závislosti monitorovány, aby se zabránilo zbytečnému opětovnému vykreslování (re-renderingu).

- React překonává Vue v rozmanitosti ekosystému a množství dostupných vlastních nástrojů pro vykreslování.

React VS Angular:

- Ve srovnání s Angular frameworky React vykazuje větší flexibilitu. Hlavní rozdíl a výhoda React od Angularu je v tom, že React používá Virtual DOM, který umožňuje aktualizovat pouze požadovaný fragment HTML. Angular používá reální DOM, který jej výrazně zpomaluje kvůli obousměrné vazbě dat.
- React architektura má jednodušší škálovatelnost.
- Angular zaostává ve výkonu ve srovnání s React kvůli většímu počtu prvků DOM.
- React je kompatibilnější s jinými knihovnami, protože používá pouze JavaScript, zatímco Angular používá Typescript, který vyžaduje definici každé knihovny.

Vue VS Angular:

- Vue se snadno učí, používá čistý JavaScript a šablony ve Vue.js jsou psány v HTML. Angular na druhé straně, je těžší se naučit. Pro používání Angularu, musíme nejprve pochopit další koncepty, jako je TypeScript a MVC (Model View Controller). Pro začátečníky je tedy velmi obtížné se učit.
- Angular má velkou komunitu programátorů, která se neustále rozrůstá. Vue je rostoucí komunita. Přestože se rychle stal populárním frameworkem, stále má menší trh ve srovnání s React nebo Angular.
- Vue je menší a lehčí na rozdíl od Angular. Využívá prvky a aplikace třetích stran k integraci mnoha potřebných funkcí. Ale ve většině případů vyžaduje přidání dalších knihoven, protože ve Vue je velmi málo vestavěných základních knihoven. S Angular je rozdíl v tom, že existuje mnohem více vestavěných funkcí a knihoven. Angular se tak stává monolitickým a těžkým frameworkem, který vyžaduje hodně úložného prostoru. Zatímco mnohé z těchto prvků jsou velmi užitečné a nezbytné, jiné by pravděpodobně nikdy nebyly použity.

Tento seznam byl sestaven na základě informací ze zdroje [18] uvedeného v části Literatura a zdroje 80.

Výsledek: po podrobném srovnání a identifikaci všech hlavních vlastností frameworků, také na základě mých osobních zkušeností, jsem dospěla k závěru, že pro vytvoření klientské části aplikace bude použita JavaScript knihovna React.js. Je to nejpopulárnější možnost pro vývoj web aplikací s velkou komunitou programátorů, také má velké množství tutoriálů a různých užitečných doplňků.

3.1.4 Architektura aplikace

Vzhledem k tomu, že aplikace, kterou vytvářím, bude obsahovat 3 části: klientská část, serverová část a databáze, rozhodla jsem se zvážit alternativní možností architektury aplikace.

V současné době ve světě programování existuje několik stacků pro vývoj webových aplikací:

- MEAN stack (MongoDB, Express.js, Angular.js, Node.js)
- MERN stack (MongoDB, Express.js, Angular.js, Node.js)
- MEVN stack (MongoDB, Express.js, Vue.js, Node.js)

- LAMP stack (Linux, Apache, MySQL, PHP (Python nebo Perl))

Každý z nich může sloužit jako alternativní možnost architektury aplikace.

Tato část obsahuje informace ze zdroje [19]
Vzhledem k tomu, že MERN, MEAN a MEVN stacky se liší pouze technologií použitou na straně klienta (React, Angular nebo Vue) a jejich porovnání bylo provedeno výše, rozhodla jsem se v této části porovnat dvě dosti odlišné architektury: MERN vs LAMP:

- Klíčovou výhodou MERN je používání programovacího jazyku JavaScript jak na serverové tak i na klientské straně. LAMP stack používá dostatečně starý programovací jazyk PHP, což může být problém při kompatibilitě s novějšími frameworky.
- Oba stacky běží na operačním systému Linux. Ale MERN stack používá novější verze každé technologie ve svém stacku, díky čemuž je lepší, stabilnější, rychlejší a mnohem jednodušší na správu, zejména během nasazení.
- LAMP Stack byl původně vyvinut pro Unix-like operační systémy, takže může běžet na jakékoli platformě, kde funguje PHP bez výraznějších problémů. MERN stack je ale mnohem lepší, protože používá nejnovější verzi Node.js, díky čemuž je multiplatformní a může bez problémů běžet na jakékoli platformě s podporou JavaScriptu.
- MERN stack má oproti LAMP obrovskou výhodu díky novějším verzím, rychlosti, stabilitě a snadnému použití.
- LAMP existuje déle než MERN stack, používá ho velké množství různých společností a vývojářů, kteří mají velké zkušenosti s vývojem aplikací pomocí LAMP stacku. Mnoho programátorů s radostí sdílí své zkušenosti, takže je možné najít poměrně velké množství informací. Ale MERN má výhodu oproti LAMP díky rychlejším aktualizacím a neustálému zlepšování.
- Stack LAMP je nejlepší volbou pro zvládání velkého provozu a vysokého zatížení díky kompatibilitě s MySQL.
- V MERN stacku je několik skvělých frameworků, jako je Express.js umožňujících snadno vytvářet vysoce výkonné webové aplikace. Na druhou stranu stack LAMP nemá žádné specifické frameworky pro vývoj serverů, a proto zaostává za zásobníkem MERN, pokud jde o výkon a rychlost.
- LAMP používá dynamický typ aplikace, zatímco MERN používá tři typy aplikace: Statické, WCF/RESTful a SOAP. Protože jsou všechny tři typy aplikací dynamické, MERN oproti LAMP má obrovskou výhodu a je lepší volbou pro práci s velkými databázemi, dynamickými daty a webovými aplikacemi na straně klienta.

Výsledek: nakonec jsem si vybrala MERN Stack, který obsahuje moderní vysoce výkonné a neustále se vyvíjející technologie, umožňující rychlý a snadný vývoj webových aplikací.

3.1.5 Hostingy

V této sekci jsem se rozhodla porovnat různé populární hostingy, které dovolují nasazení aplikace zadarmo.

Heroku je cloudová platforma (PaaS), která umožňuje vytváření, nasazování a provozování aplikace v cloudu. Heroku podporuje několik programovacích jazyků: Ruby, Java, Node.js, Scala, Clojure, Python, PHP, and Go.
DigitalOcean je dodavatel cloud computingu, který nabízí infrastrukturu jako službu (IaaS). Podporuje několik programovacích jazyků: Node.js, Python, Ruby, Django, Go, PHP.
*"**Amazon Web Services (AWS)** je světově nejrozšířenější platforma cloud computingu pro poskytování online služeb nabízené americkou společností Amazon.com Inc, ke které lze přistupovat přes internet po celém světě a nevyžaduje správu uživatelů nebo tyto zdroje sledovat."* [20] Podporuje několik programovacích jazyků: Java, Go, PowerShell, Node.js, C#, Python, Ruby.

DigitalOcean vs Heroku:

- DigitalOcean je cloudová služba nebo IaaS, která nabízí možnosti sítí, úložiště a virtualizace. Na druhou stranu, Heroku je PaaS, což označuje softwarové a hardwarové nástroje dostupné na internetu.
- Heroku nabízí vývojářům více svobody soustředit se na optimalizaci svých aplikací. Spravuje infrastrukturní stránku, jako je údržba serverů a databází. DigitalOcean není spravován stejným způsobem. Při používání této platformy byste se museli soustředit jak na vývoj aplikací, tak na správu backendu.
- Vzhledem k tomu, že Heroku nabízí lepší správu aplikací a serverů, je relativně nákladné ve srovnání s DigitalOcean.
- DigitalOcean umožňuje vytvořit virtuální soukromý server. I když je poskytovatel služeb nespravuje to sám, dostupné užitečné příručky, které mohou pomoci takové servery nastavit sami. Heroku takovou možnost nenabízí.
- Heroku nabízí pouze dvě umístění datových center: USA a Evropa. Na druhou stranu je DigitalOcean flexibilnější a poskytuje více možností datových center, včetně USA, Evropy, Asie a Indie.

Tento seznam byl sestaven na základě informací ze zdroe [21] uvedeného v části Literatura a zdroje 80.

AWS vs Heroku:

- Heroku je cloudová platforma založená na kontejnerech a nabízející PaaS, zatímco AWS je zabezpečená platforma cloudových služeb poskytující IaaS, PaaS a SaaS.
- Heroku nabízí prostředí připravené k použití, které umožňuje rychlé nasazení kódu, zatímco proces nasazení služby AWS je poměrně komplikovaný.
- Heroku je nejvhodnější pro startupy [1], střední podniky, zatímco AWS se zaměřuje hlavně na střední podniky a velké podniky.
- Heroku dokáže splnit nízké výpočetní nároky, zatímco AWS dokáže splnit vysoké/velmi vysoké výpočetní nároky.
- Heroku nepotřebuje údržbu infrastruktury, zatímco AWS potřebuje DevOps specialistu.

[1] startup - podnikatelský subjekt, typicky popsaný jako nově založená či začínající společnost, která se na základě inovativní podnikatelské koncepce za použití vyspělých technologií rychle vyvíjí a má velký potenciál hospodářského růstu.

- Heroku podporuje méně geografických oblastí než AWS.

Tento seznam byl sestaven na základě informací ze zdroe [22] uvedeného v části Literatura a zdroje 80.

DigitalOcean vs AWS:

- DigitalOcean poskytuje infrastrukturu jako službu (IaaS), zatímco AWS poskytuje infrastrukturu jako službu (IaaS), platformu jako službu (PaaS) a software jako službu (SaaS).
- DigitalOcean je jednoduchý poskytovatel cloudových služeb, zatímco Amazon Web Service (AWS) je platforma, která nabízí flexibilní, spolehlivá, škálovatelná, snadno použitelná a nákladově efektivní řešení cloud computingu.
- DigitalOcean nabízí automatické škálování, spolehlivou a snadnou správu, na druhou stranu AWS nabízí jednoduchý řídicí panel, skvělou komunitu a snadnou konfiguraci.
- DigitalOcean je vhodnější pro nezávislé vývojáře a malé aplikace, zatímco AWS je vhodný pro velké škálovatelné aplikace.
- DigitalOcean má ve srovnání s AWS méně podporovaných regionů.

Tento seznam byl sestaven na základě informací ze zdroe [23] uvedeného v části Literatura a zdroje 80.

Výsledek: vybrala jsem cloudovou platformu Heroku, protože poskytuje plně spravovanou infrastrukturu, škálovatelný hosting a další softwarové výhody nad rámec čisté architektury IaaS. Také Heroku se snadněji spouští a používá.

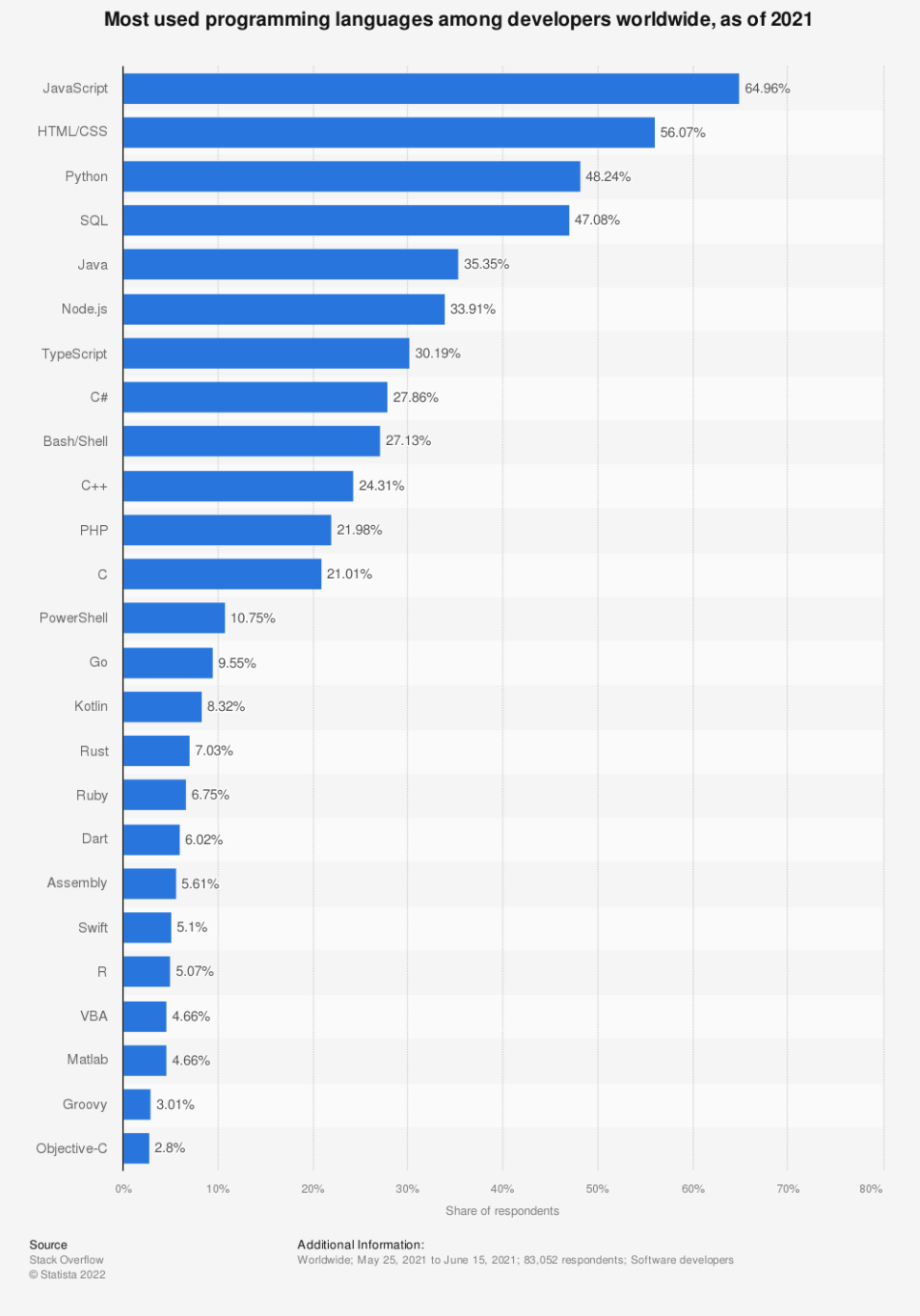
3.2 Vybrané technologie

Tato sekce popisuje návrh architektury aplikace a technologie potřebné pro její implementaci. Řešení bylo vybráno na základě podrobného zkoumání a porovnání všech možných alternativ, které byly popsány v sekci výše.

3.2.1 Programovací jazyk

K dnešnímu dni existuje mnoho řešení pro vytváření rozsáhlých webových aplikací - od programovacích jazyků a knihoven až po řadu pohodlných frameworků.

Na základě mé analýzy a vážení všech výhod a nevýhod různých programovacích jazyků jsem se rozhodla vybrat programovací jazyk **JavaScript**, který již devátý rok zůstává nejčastěji používaným programovacím jazykem na základě průzkumu platformy Stack Overflow. [17] Žebříček je znázorněn na obrázku 3.3



Obrázek 3.3: Žebříček nejpoužívanějších programovacích jazyků 2021 [17]

JavaScript (JS) je vysokoúrovňový programovací jazyk, který podporuje imperativní, funkční, řízené událostmi a další užitečné přístupy. Patří mezi jazyky s dynamickým psaním a je zařazen do skupiny interpretovaných jazyků. Hlavní rysy JS zahrnují:

- Dynamické typizace - typ dat je stanoven v době přiřazení hodnoty konstanty nebo proměnné.
- Interpretovaný jazyk - kód aplikace je interpretován při použití, není nutná

předběžná kompilace.

- Funkce jako objekty první třídy, tj. funkce v JavaScriptu, lze vrátit z funkcí, sdělit jako parametry do jiných funkcí, lze přiřadit k proměnným.
- Podpora prototypového a objektově orientovaného přístupu.
- Univerzálnost - všechny populární prohlížeče podporují JavaScript.

Tento seznam byl sestaven na základě informací ze zdroe [24] uvedeného v části Literatura a zdroje 80.

Také pro rychlejší a snazší psaní aplikace a možnost používat pohodlná rozhraní a třídy na straně klienta jsem se rozhodla použít programovací jazyk TypeScript. **TypeScript** je programovací jazyk pro vývoj webových stránek založený na JavaScriptu. Dělá kód jasnější a spolehlivější, přidává statické typizace (proměnné jsou vázány na konkrétní typy dat) a lze jej také zkompilovat do JavaScriptu. Tento jazyk také pomáhá okamžitě identifikovat chyby provedené v procesu modifikace kódu a umožnuje snadnější škálování a testování aplikace. [25]

3.2.2 Architektura aplikace

Vývoj webové aplikaci bude založen podle tradičního 3-tier architektonického vzoru. Architektura aplikace bude realizována pomocí softwarového stacku MERN. Stack– je to soubor nástrojů a technologií, které se používají ke vývoje programů a zahrnuje programovací jazyky, frameworky, databázové systémy, informační systémy atd. MERN je JavaScript stack navržený pro zjednodušení vývojového procesu. MERN obsahuje čtyři open source komponenty: MongoDB, Express.js, Node.js a React.js. [26] Obrázek 3.4 znázorňuje proces komunikace mezi jednotlivými vrstvami aplikace.

Obrázek 3.4: MERN Stack komunikace.

Mezi výhody stacku MERN patří, že pro psaní klientské a serverové části aplikace se používá jeden programovací jazyk – JavaScript. To usnadňuje psaní kódu a eliminuje potřebu transformace kontextu mezi úrovněmi. Použití No-SQL databázi MongoDB usnadňuje a zrychluje ukládání, manipulace, a reprezentace JSON dat na každé úrovni aplikace. Stack MERN dává možnost vytvářet vysoce efektivní a snadno škálovatelné webové aplikace. [27]

3.2.3 Databáze

Databáze v aplikaci „Tess&Study“ bude implementována pomocí No-SQL databáze MongoDB. MongoDB je nerelační dokumentově orientovaná No-SQL databáze používaná k ukládání různých forem dat aplikace. Místo tabulek a řádků data jsou uložena v flexibilních dokumentech spojených do kolekcí. Každý dokument má strukturu

podobnou JSON (Binary JSON). Do databáze je možné přidávat pole bez potřeby znovu načítat celou tabulku, takže je poměrně snadné škálovatelna. [12] Vzhledem k tomu, že aplikace i databáze používají JavaScript, není třeba převádět žádné objekty, které jsou předávány z aplikace do databáze a naopak. Aplikace může odesílat a načítat objekty bez zpoždění interakcí mezi back-endem a databází.

3.2.4 Back-end – serverová část

Tato sekcí je kompilací informací ze zdrojů [28], [29], [30], [31]
Serverová část aplikace „Test&Study“ bude implementována pomocí technologií Node.js a Express.js. Node.js je běhové prostředí JavaScriptu na straně serveru, které spouští JavaScriptový kód. Express.js je back-endový webový aplikační framework, napsaný v JavaScriptu a fungující uvnitř prostředí Node.js. Express.js poskytuje pluginy, kódové šablony, middleware balíčky a směrovací funkce pro rychlejší a efektivnější vývoj webových aplikací. Express.js zpracovává všechny interakce mezi klientskou částí a databází a zajišťuje bezproblémový přenos dat koncovému uživateli. V aplikaci „Test&Study“ Node.js a Express.js budou plnit roli webového serveru.

Vytvořením XHR požadavků (XMLHttpRequest) nebo použitím metody GET či POST z React.js front-endu je možné připojit k funkcím Express.js, který zajišťuje fungování aplikace. Jakákoli interakce ze klientské strany, která způsobí žádost o změnu údajů, bude odeslána na Express server, který v případě potřeby načte data z MongoDB databáze a vrátí jejich do front-endu aplikace, který se pak zobrazí uživateli. Komunikace mezi klientskou a serverovou částí bude probíhat pomocí vytvořených na serveru API. Express.js využívají Node.js MongoDB drivers (či Promises [2]) k přístupu a aktualizaci dat ve databázi MongoDB.

3.2.5 Front-end – klientská část

Pro vytvoření klientské části aplikace v stacku MERN se používá JavaScriptová knihovna – React.js, která slouží k vytváření uživatelských rozhraní a vývoji dynamických jednostránkových aplikací (SPA) pomocí znovupoužitelných UI komponent. Jednostránková aplikace (SPA) – je aplikace, která běží v prohlížeči a během práce stránku znovu nenačítá.[32] React.js umožní implementovat aplikaci „Test&Study“ jako SPA aplikaci. Tato knihovna dává možnost vytvářet komplexní rozhraní pomocí jednoduchých komponent, připojit je k datům na interním serveru a zobrazovat je ve formátu HTML na straně klienta. React používá JSX, rozšíření syntaxe JavaScriptu, které umožňuje kombinovat HTML s JavaScriptem. [33] Uživatel interaguje s React.js UI komponentami na front-endu aplikace umístěném v prohlížeči. Front-end bude obsluhován serverem aplikace přes Express.js běžící na Node.js.

[2] *Promise - je objekt, představující výsledek asynchronní operace. (Štěpán Zavadil, 2022)*

Část IV

Implementace

Kapitola 4

Implementace

Tato část popisuje implementaci aplikace. První část této sekce obsahuje modely, které vizualizují architekturu aplikace. Druhá část popisuje strukturu aplikace, principy interakce její jednotlivých komponent a podrobně rozebrány technologie, které byly použity k vytvoření aplikace "Test&Study".

4.1 Architektura aplikace

Před zahájením implementace aplikace je důležité detailně promyslet její architekturu. Architektura aplikace umožňuje systému pracovat předvídatelněji, zjednodušuje řízení vývoje a rozvoj systému a umožňuje flexibilně měnit jednotlivé bloky aplikace.

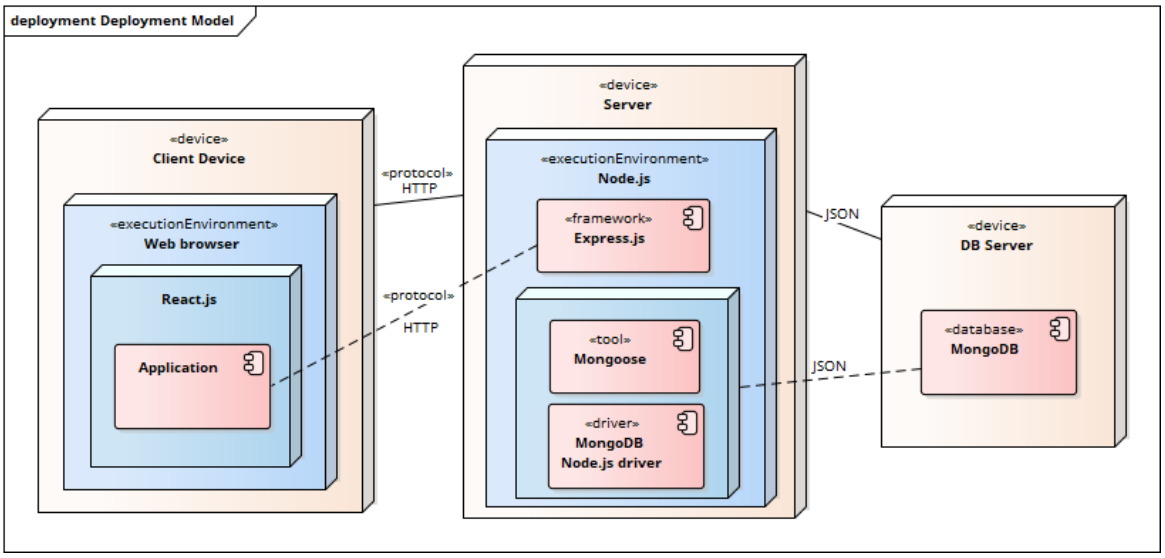
4.1.1 Diagram nasazení

*"**Diagram nasazení (Deployment diagram)** zobrazuje způsob, jakým bude rozmístěna architektura software na architekturu hardware. Diagram ukazuje:*

- *Uzly – hardwarové uzly, na kterých bude system spouštěn*
- *Komponenty – softwarové komponenty (fyzicky samostatné části systému)*
- *Rozhraní – rozhraní pro komunikaci s komponentami (oddělení specifikace od implementace)*
- *Relace – spojení mezi uzly a závislosti mezi komponentami"[34]*

(B. Zimmerová, 2008, s.3)

Na obrázku 4.1 je znázorněn diagram nasazení jednotlivých částí aplikace.



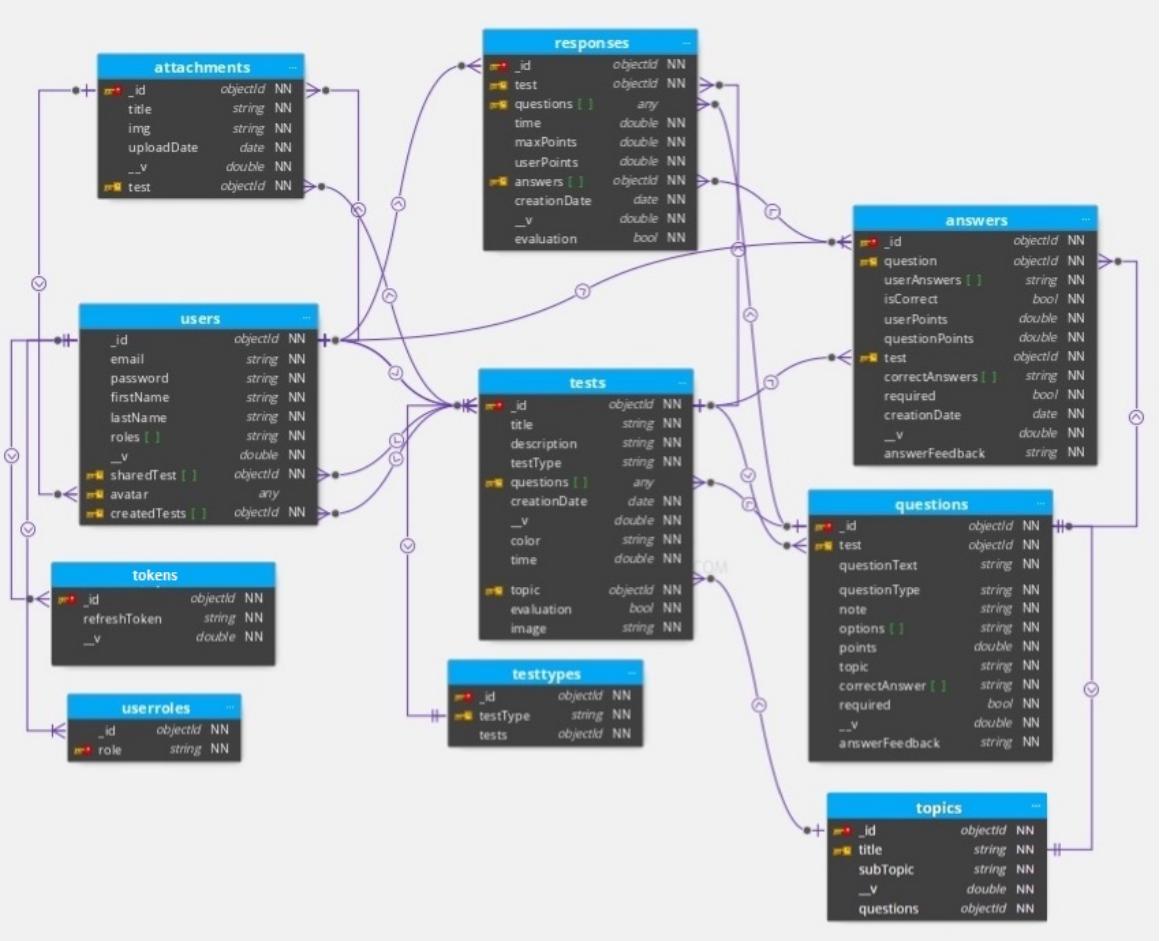
Obrázek 4.1: Diagram nasazení

Webová aplikace "Test&Study"vytvořena podle tradičního 3-tier architektonického vzoru. Třívrstvá architektura organizuje aplikace do tří logických a fyzických vrstev: prezentační vrstva neboli uživatelské rozhraní (Client Device); aplikační vrstva, kde se data zpracovávají (Server); a datovou vrstvu, kde jsou uložena a spravována data spojená s aplikací (DB Server).
Na straně klienta je aplikace vytvořena pomocí knihovny React. Komunikace mezi klientskou částí a serverem probíhá pomocí API vytvořeného na serveru, kde byly vytvořeny speciální terminály pro zpracování požadavků. Framework Express.js na straně serveru zpracovává HTTP požadavky klientské strany a odesílá zpět odpovědi s potřebnými informacemi. Dokumenty JSON vygenerované v rozhraní React.js jsou odeslány na server Express.js, kde jsou zpracovány, a pokud jsou platné, pomocí užitečného nástroje Mongoose jsou uloženy přímo v MongoDB pro pozdější načtení. Mongoose - nástroj pro objektové modelování (ODM) MongoDB pro Node.js, který umožňuje vytvářet modely a závislosti mezi nimi a používat speciální API pro práci s databází. [35]

4.1.2 Databázový model

*"**Databázový model** - nástroj pro reprezentaci struktury a funkcionality databáze. Umožňuje definovat schéma databáze, určující organizaci dat, způsoby jejich ochrany a zajištění správnosti (integritní omezení) a přípustné operace s daty.* (Národní knihovna České republiky , 2014)

Na obrázku 4.2 je znázorněn databázový model NoSQL databáze MongoDB.



Obrázek 4.2: Databázový model MongoDB

MongoDB má dynamickou architekturu schémat, která pracuje s nestrukturovanými daty a úložištěm. Protože jsou data uložena ve flexibilních dokumentech podobných JSON, schéma databáze nemusí být předdefinováno a může být dynamicky upraveno. Dokumenty se skládají z párů klíč-hodnota, které jsou základní jednotkou dat v MongoDB. Dokumenty mohou definovat primární klíč jako jedinečný identifikátor a hodnoty mohou být různé datové typy, včetně jiných dokumentů, polí a polí dokumentů. [36] V mé databázi jsem použil metodu DBRefs pro propojení dokumentů. *"DBRefs jsou odkazy z jednoho dokumentu na druhý pomocí hodnoty pole _id prvního dokumentu, názvu kolekce a volitelně názvu její databáze a také jakýchkoli dalších polí. Soubory DBRefs umožňují snadněji odkazovat na dokumenty uložené ve více kolekcích nebo databázích."* (MongoDB, Inc. 2022) [37]

4.2 Struktura projektu

Graf 4.3 popisuje strukturu projektu a za co jsou zodpovědné určité soubory.

```
Test&Study/
├── client/ ..............složka souborů klienta.
│   ├── public ...........veřejné soubory.
│   ├── src/ .............zdrojový kód klientské strany.
│   │   ├── components ....React komponenty.
│   │   ├── http ..........konfigurace Axios.
│   │   ├── models/ .......typované entity.
│   │   │   └── response ....typované modely odpovědí ze
│   │   │                    serveru.
│   │   ├── pages .........směrování stránek.
│   │   ├── services ......zpracování odpovědí a odesílání
│   │   │                    požadavků na server.
│   │   ├── store .........state handlers.
│   │   ├── styles ........obsahuje scss soubory popisující
│   │   │                    styly.
│   │   └── utils .........sekundární funkce.
│   └── static/ ..........statické soubory.
│       └── images
├── controllers ..........odpovědný za vyřizování příchozích
│                         požadavků a zasílání odpovědí
│                         klientovi.
├── dtos .................jednoduchá rozhraní, které pomáhá
│                         přenášet data mezi procesy.
├── exceptions ...........popis chyb.
├── middlewares ..........mezizpracovací funkce.
├── models ...............schémata databáze MongoDB.
├── router ...............cesty pro zpracování požadavků od
│                         klienta.
├── service ..............dodává data kontrolérům a
│                         přistupují k databázi.
└── app.js ...............konfigurace spouštění serveru.
```

Obrázek 4.3: Struktura projektu

4.3 Použité technologie

Tato část podrobně popisuje technologie, které byly použity k vytvoření aplikace. Aplikace se skládá ze tří samostatných částí, které spolu komunikují pomocí API.

4.3.1 Vývojové prostředí

Pro vývoj aplikace jsem zvolila vývojové prostředí(IDE) WebStorm používané především pro vývoj v JavaScriptu. Obsahuje mnoho užitečných funkcí a pluginů, které pomáhají psát kód rychleji, identifikovat chyby a navrhuji, jak je lze vyřešit.

4.3.2 Back-end – serverová část

Server je spravován hlavně dvěma komponentami MERN stacku: Express.js a Node.js. Tyto dvě komponenty to zpracovávají současně, protože Express.js je serverový framework běžící na serveru Node.js. Express.js je jedním z široce používaných backendových JavaScript framework. Umožňuje snadněji a jednodušeji vytvořet robustní API (Application Programming Interface) a webové servery. Také přidává užitečné funkce do Node.js HTTP (HyperText Transfer Protocol) objektů. Na druhou stranu Node.js hraje samo o sobě velmi důležitou roli. Jedná se o open-source [1] serverové multiplatformní běhové prostředí pro spouštění kódu JavaScript mimo prohlížeč.

Express aplikace používá funkci zpětného volání, jejíž parametry jsou objekty požadavku a odpovědi.

```
app.get('/', function (req, res) {
    // —
})
```

Objekt požadavku představuje požadavek HTTP a má vlastnosti pro řetězec dotazu požadavku, parametry, tělo, hlavičky HTTP atd. Objekt odpovědi představuje odpověď HTTP, kterou aplikace Express odesílá, když obdrží HTTP požadavek.

K napsání moderní multifunkční serverové části aplikace byly použity nástroje:

- **Mongoose** - je správce objektových dat (ODM), který zajišťuje interakce mezi aplikací Express.js a databází MongoDB.
- **Body-parser** je middleware [2], který umožňuje aplikaci číst tělo (tj. obsah) příchozích požadavků (součástí Express.js).
- **Cookie-parser** analyzuje hlavičku cookie a naplní req.cookies objektem, na který ukazují názvy souborů cookie.
- **DotENV** - umožňuje používat soubory s příponou .env pro práci s důvěrnými daty.
- **Validátor** – jednoduchá validace mnoha typů dat.
- **bCrypt** – šifrování citlivých dat, jako jsou hesla.
- **JSON Web Token (JWT)** - představuje způsob pro bezpečnou výměnu informací mezi dvěma stranami.
- **CORS** - technika využívající hlaviček protokolu HTTP, která umožňuje aplikacím běžícím ve webovém prohlížeči přistoupit ke zdroji (datům), který leží na jiné doméně než na které běží aplikace.
- **Nodemon** - okamžitý restart serveru v případě jakýchkoli změn. Nodemon umožňuje po provedení jakýchkoli změn nezastavovat a znovu spuskat server, toto se provádí automaticky.
- **Multer** middleware pro analýzu (parsování) dat ve formátu multipart/form-data (pro ukládání souborů na server).

Tento seznam byl sestaven na základě informací ze zdroje [**NPM**] uvedeného v části Literatura a zdroje 80.

[1] open-source - software s volně přístupným zdrojovým kódem, který tak může každý libovolně upravovat a měnit za účelem vývoje daného nástroje.

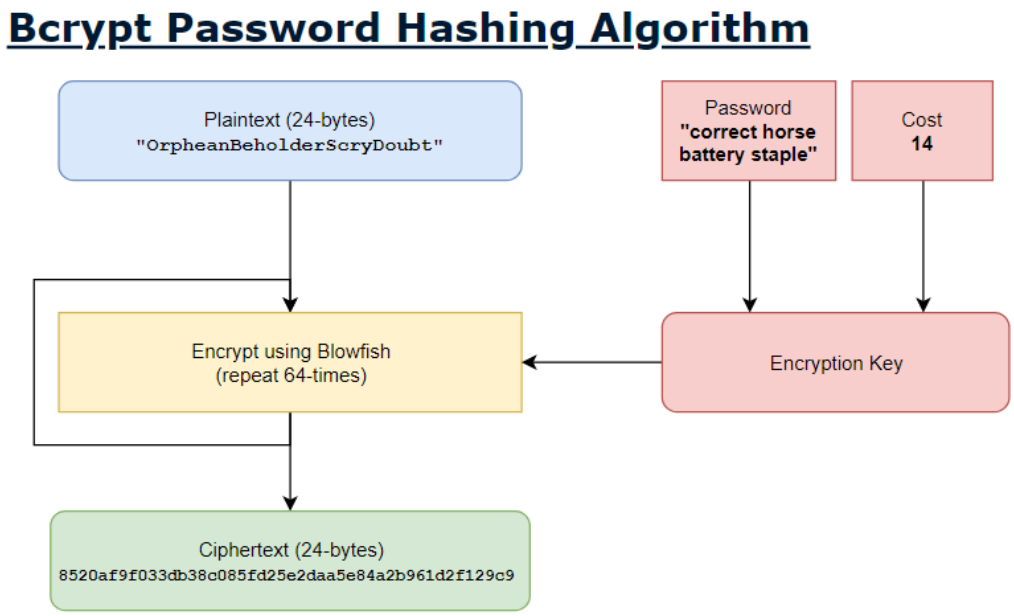
[2] middleware - software, který spojuje různé části aplikace.

4.3.3 Zabezpečení aplikací

Zabezpečení aplikace je zajištěno hašováním hesel pomocí knihovny Bcrypt a autentizace dat pomocí JWT tokenu.

Bcrypt je funkce hašování hesel založená na šifře Blowfish. Hašování hesel je běžný přístup k bezpečnému ukládání hesel v databázi. Hash je jednosměrná funkce, která generuje reprezentaci hesla.

Na obrázku 4.4 je znázorněn algoritmus hašování hesel Bcrypt.

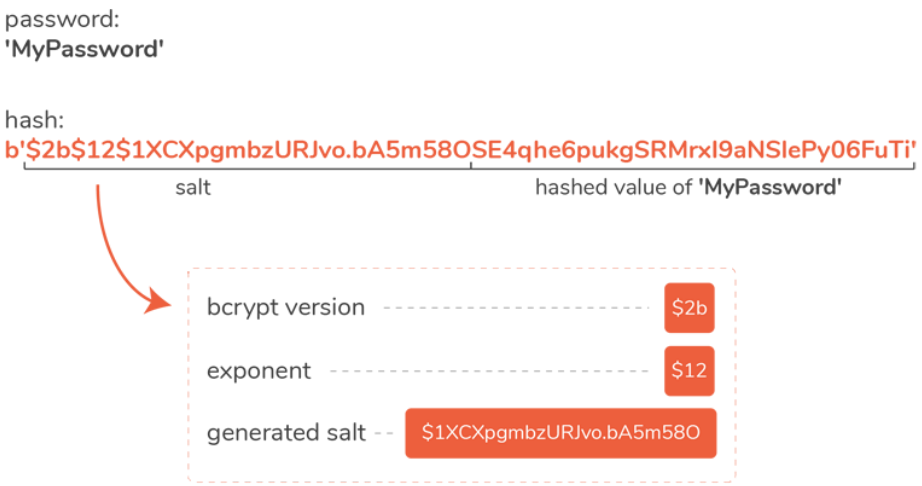


Obrázek 4.4: Algoritmus hašování hesel Bcrypt. [38]

Algoritmus Bcrypt běží ve dvou fázích, znázorněných na obrázku 4.4. V první fázi probíhá šifrování ceny, solí a hesla. Většinu času Bcrypt stráví drahým klíčem. Poté je 192bitová hodnota OrpheanBeholderScryDoubt zašifrována 64krát pomocí eksblowfish v režimu „kódové knihy“ ECB(electronic code book mode) [3] se stavem z předchozí fáze. Výstupem je cena a 128bitová sůl spojená s výsledkem šifrovacího cyklu.

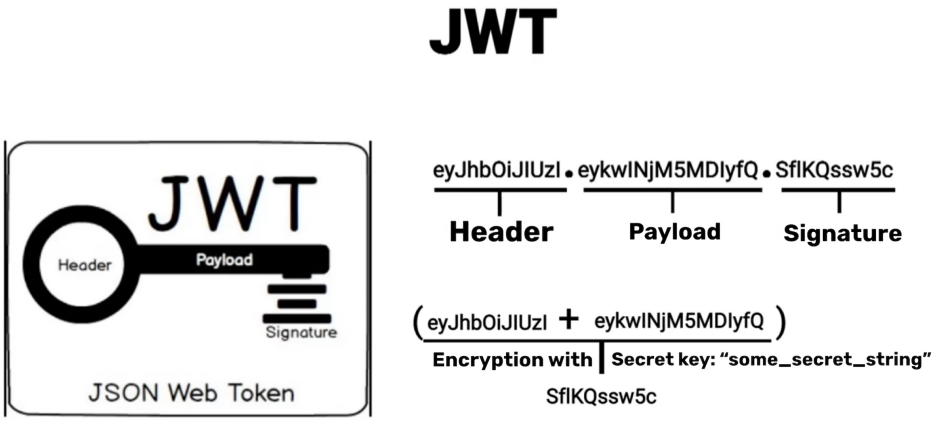
Na obrázku 4.5 je znázorněn výsledek hašování hesla pomocí Bcrypt.

[3] Režim „kódové knihy“ - je režimem nejjednodušším a základním. Bloková šifra se při něm přímo aplikuje nezávisle na jednotlivé bloky, tedy při daném klíči odpovídá stejnému bloku otevřeného textu stejný blok šifrového textu. [39]



Obrázek 4.5: Výsledek hašování hesla pomocí Bcrypt. [40]

JWT je
Na obrázku 4.6 je znázorněn JWT Token.



Obrázek 4.6: JWT Token [41]

JWT Token je řetězec rozdělený na tři části pomocí teček. První část tokenu je hlavička, druhá část jsou některá data, která jsou vložená do tokenu, a třetí část je podpis. Záhlaví musí obsahovat v sobě, který algoritmus se používá k šifrování a volitelné pole, například typ tokenu nebo některé další pomocné prvky. Druhou částí tokenu jsou nějaká data, tato data jsou již vložená přímo do tokenu, nejčastěji jsou to nějaké informace o uživateli, který prochází autorizačním procesem: jeho uživatelské jméno, jeho unikátní identifikátor a například jeho role, aby server mohl určit, k čemu uživatel má přístup a k čemu ne. Hlavička a data jsou v podstatě uložena v čistém textu a dekódovat je může úplně každý, což není nic složitého, takže ukládat tam nějaké důležité informace jako hesla se rozhodně nevyplatí. Zabezpečení je zajištěno podpisem, který vyžaduje získání tajného klíče. Tento tajný klíč bude znát pouze server. To znamená, že se vezme hlavička, vezmou se data a pomocí tajného klíče se to vše zakóduje a výstupem je podpis nutný pro jistotu, že token nebyl zfalšován. To znamená, že pokud neznáte tajný klíč, je téměř nemožné tento

podpis získat. [41]

Autentizace je implementována pomocí přístupových tokenů JWT - access token a obnovovacích tokenů - refresh token. Při úspěšné autentizaci server API vrátí krátkodobý přístupový JWT token, jehož platnost vyprší po 15 minutách, je uložen v úložiště LocalStorage [4], a obnovovací token, jehož platnost vyprší po 7 dnech, je uložen v souboru HTTPOnlycookie. JWT se používá pro přístup k zabezpečeným API cestám (routes) a obnovovací token se používá pro generování nových přístupových JWT tokenů, když (nebo těsně před) vyprší jejich platnost.

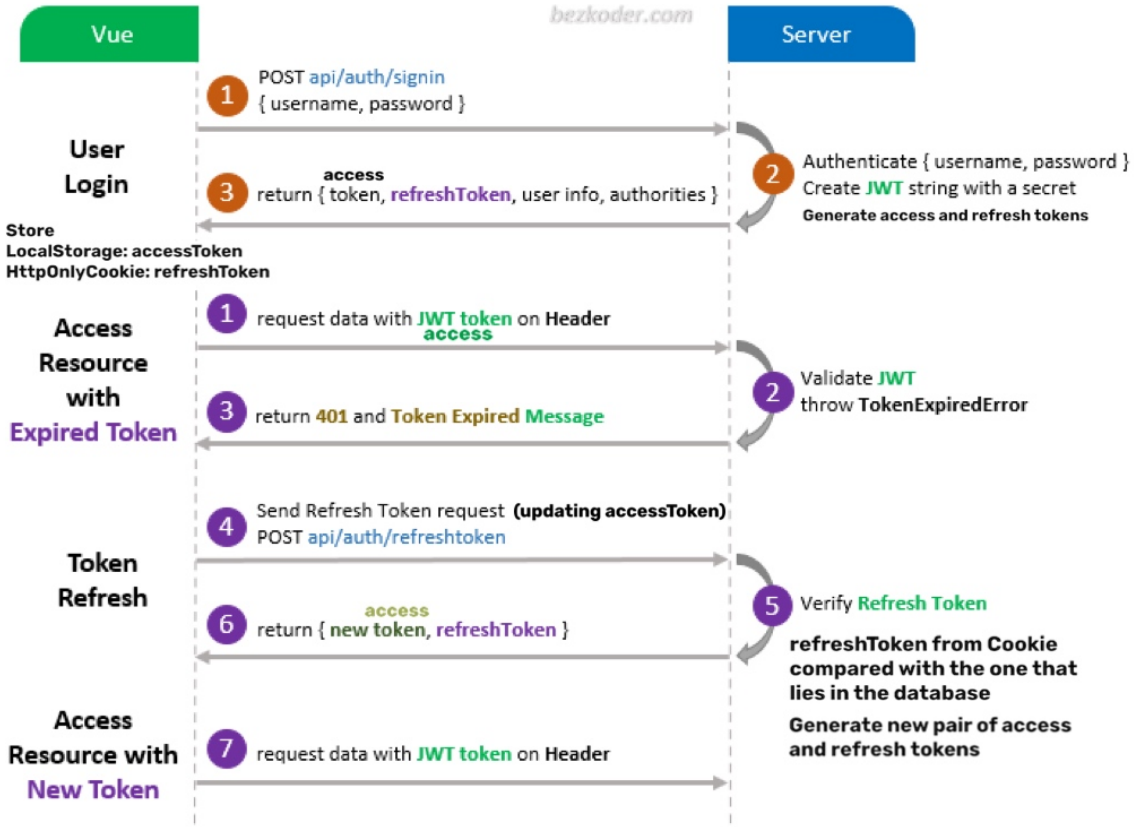
HTTPOnlycookies se používají pro obnovovací tokeny ke zvýšení zabezpečení, protože nejsou přístupné pro JavaScript na straně klienta, to zabraňuje XSS (cross site scripting) útokům [5], a obnovovací tokeny mají přístup pouze ke generování nových JWT tokenů (prostřednictvím /users/refresh - token route), která zabraňuje jejich použití při CSRF (cross site request forgery) útocích [6]. [45]

Na obrázku 4.7 je znázorněn jednoduchý proces obnovení JWT refresh tokenu pomocí Axios Front-end – klientská část.

[4] LocalStorage - úložiště v prohlížeči klienta přístupné JavaScriptem.

[5] XSS (cross site scripting) útok - typ zranitelnosti webové aplikace. XSS útok je založen na vložení kódu (podstrčení) do dynamické webové stránky (JavaScript prováděný na straně klienta). [42]

[6] CSRF (cross site request forgery) útok - je metoda webového útoku, která *"spočívá v tom, že uživatele přimějeme navštívit stránku napadané aplikace, která provádí nějakou akci, aniž by o tom uživatel věděl"* [43]. Cílem útočníka je donutit uživatele provést bez jeho vědomí akci v nějaké webové službě. [44]



Obrázek 4.7: Jednoduchý proces obnovení JWT refresh tokenu pomocí Axios. [46]

Uživatel otevře aplikaci, zadá e-mail a heslo a stiskne tlačítko přihlášení. V těle požadavku je uveden tento e-mail a heslo a a tento požadavek je odeslán na server. Server poté vygeneruje nový pár přístupových tokenů a obnovovacích tokenů. Přístupový token je uložen v úložišti localStorage a obnovovací token je již nastaven serverem v souborech cookie. Uživatel se úspěšně přihlásí do svého účtu v aplikaci a odešle požadavek například na zobrazení všech jeho testů, k tomuto požadavku se přidá autorizační hlavička a na tuto adresu se zapíše access token a požadavek bude odeslán na server. První věc, kterou server udělá, je zkontroluje access token, zda je platný a není zfalšovaný a nevypršela jeho platnost, pak server vrátí stavový kód 200 a vrátí klientovi všechny testy, které požadoval. Pokud platnost access tokenu vypršela a již je neplatný, pak v tomto případě server vrátí chybu se stavovým kódem 401, to znamená, že uživatel není autorizován. Ale na straně klienta je toto chování zajištěno a pro odpověď s kódem 401 existuje interceptor. Kdy klient obdrží odpověď se stavovým kódem 401, okamžitě bude odeslán požadavek na aktualizaci přístupového tokenu. Obnovovací token je již v souborech cookie, takže jej není potřeba nikam přidávat. Z klienta je odeslán požadavek s obnovovacím tokenem, server tento token zpracuje a porovná s tím, který je v databázi, poté vygeneruje a klientovi vrátí nový pár access a refresh tokenů. Po vypršení platnosti access tokenu se tento proces bude opakován. [41]

4.3.4 Databáze

Aplikace využívá No-SQL databázi MongoDB.

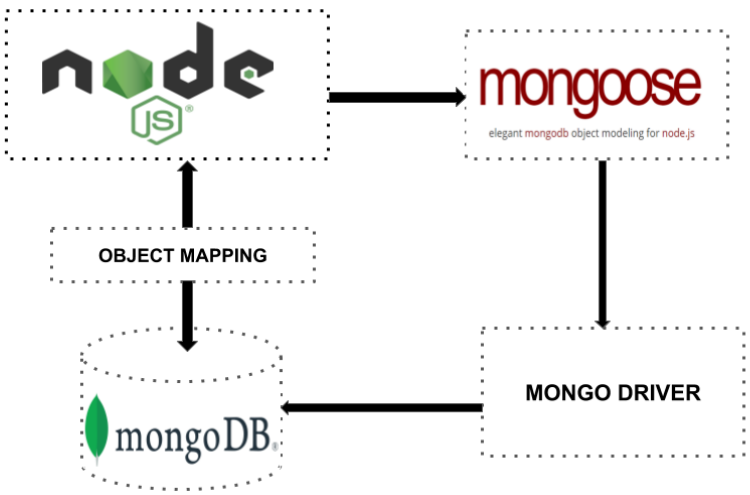
MongoDB - multiplatformní dokumentově orientovaná databáze, ve které jsou data uložena ve flexibilních dokumentech s dotazovacím jazykem založeným na JSON. Obsah, velikost a počet polí v dokumentech se mohou navzájem lišit, což znamená, že struktura dat se může v průběhu času měnit. MongoDB je nejlépe známá pro své flexibilní a škálovatelné funkce. [**MongoDBbook**]

Databázový model aplikace je uveden v sekci Databázový model.

Seznam databázových kolekcí:

- users (uživatelé aplikace)
- userroles (uživatelé aplikace (User, Admin, Member, Creator))
- tokens (refresh token)
- topics (témata testů a otázek)
- testtypes (typ testu (Scored / Nonscored))
- tests (všechny testy aplikací)
- responses (všechny odpovědi na testy aplikací)
- attachments (nahrané soubory)
- answers (odpovědi na otázky v testu)
- questions (testové otázky)

Komunikace mezi databází a serverem je zajištěna pomocí Mongoose, který spravuje vztahy mezi daty, poskytuje ověřování schémat a používá se k překladu mezi objekty v kódu a reprezentaci těchto objektů v MongoDB. Na obrázku 4.8 je znázorněn proces mapování objektů mezi Node.js a MongoDB spravované přes Mongoose.



Obrázek 4.8: Mapování objektů mezi Node.js a MongoDB spravované přes Mongoose. [47]

4.3.5 Front-end – klientská část

Serverová a klientská část spolu komunikují přes REST API [7] ve formátu JSON. "Test&Study"je jednostránková aplikace, vytvořená pomocí JavaScript knihovny **React.js**, která umožňuje vytvářet dynamické client-side aplikace v HTML.

Pro implementaci klientské části aplikace byly použity tyto užitečné nástroje:

- **Material UI** - Populární React UI [8] framework, obsahující základní a pokročilé komponenty a umožňující rychlé vyvíjet React aplikace.
- **Axios** - je knihovna JavaScriptu. Axios je Promise [9] based HTTP klient pro prohlížeč a Node.js. Axios usnadňuje odesílání asynchronních požadavků HTTP do koncových bodů REST a provádění operací CRUD [10]. Na straně serveru používá nativní Node.js HTTP modul, zatímco na straně klienta (prohlížeče) používá XMLHttpRequests. Výhody používání Axios: podpora pro zachycování více požadavků a odpovědí, efektivní zpracování chyb, podpora na straně klienta pro ochranu XSRF, má časový limit odpovědi a možnost zrušit požadavky, automatizovaný překlad JSON dat, podpora nahrávání souborů. [48]
- **MobX** - je samostatná knihovna pro správu stavu frontendu aplikace. MobX zajišťuje konzistenci vnitřního stavu front-end aplikace a poskytuje pohodlné nástroje pro jeho změnu. MobX umožňuje implementovat řetězec: "provedení akce"→ "změna stavu"→ "změna zobrazení". Změny probíhají automaticky – ve výsledku je zaručeno, že nenastane okamžik, kdy bude stav nekonzistentní. React v kombinaci s MobX umožňuje neustále přiřazovat vnitřní stav s vizuální reprezentací rozhraní. [49]
- **Next.js** - je framework založený na Reactu, který umožňuje vytvářet webové aplikace s vylepšeným výkonem a vylepšenou uživatelskou zkušeností prostřednictvím pokročilých funkcí předběžného vykreslování, jako je úplné vykreslování na straně serveru (SSR) a generování statických stránek (SSG). SSR umožňuje přístup ke všem potřebným datům pro vytvoření stránky na serveru. Stránka je poté plně odeslána zpět do prohlížeče a okamžitě vykreslena. SSR umožňuje načítání webových stránek za kratší dobu a zlepšuje uživatelskou zkušenost zvýšením rychlosti odezvy. Vykreslování na straně serveru funguje tak, že mění tok požadavků aplikace React tak, že všechny komponenty kromě klienta posílají své informace na server. Se všemi informacemi na serveru může klient získat náhled HTML kódu stránky. Klient může odeslat jeden požadavek na server a získat úplnou HTML stránku namísto požadavku na každou komponentu jednotlivě s vykreslováním na straně klienta. [50]
- **Sass** - je kompilovaný jazyk, který rozšiřuje syntaxi CSS o proměnné, cykly, podmínky, mixiny a funkce. Sass pomáhají psát udržovatelný a neopakující kód. [51]

[7] REST API (Representational state transfer) - je architektura, která umožňuje přistupovat k datům na určitém místě pomocí standardních metod HTTP protokolu.

[8] UI (User Interface) - uživatelské rozhraní, způsoby interakce uživatele s webovou stránkou, aplikací nebo jinou IT službou.

[9] *Promise - je objekt, představující výsledek asynchronní operace. (Štěpán Zavadil, 2022)*

[10] CRUD (Create, Read, Update, Delete) operací - zkratka používaná v programování, která shrnuje čtyři základní operace prováděné na záznamech.

4.3.6 Nasazení aplikace

Aplikace je nasazena na cloudové PaaS-platformě Heroku. Odkaz k dispozici v Priloha D - Seznam odkazů.

Část V

Testování

Kapitola 5

Testovani

Tato část podrobné popisuje testování aplikace Test&Study. Hlavní úkoly testování jsou: identifikovat chyby a defekty, které mohou negativně ovlivnit provoz aplikace a rychle je odstranit; zkontrolovat, že aplikace splňuje požadavky; Data získaná při testování jsou důležitá i při plánování následné strategie vývoje aplikace. Webové aplikací je možné testovat ručně nebo automaticky.

5.1 Testování API

Rozhodla jsem se začít proces testování aplikace kontrolou správnosti API.

REST API vytvořené na serveru aplikace poskytuje „koncové body“, se kterými může komunikovat klientská strana aplikace.

V MERN stacku REST API je URL [1], pomocí kterého je možné provádět specifické funkce, jako je vytváření, čtení, zápis a mazání dat z databáze MongoDB (CRUD operace). Tyto adresy URL jsou funkcemi s HTTP voláními jako iniciátory. HTTP protokol se používá pro komunikaci klienta a serveru. Klientem může být například prohlížeč, ale při mém testování tuto roli bude plnit nástroj Postman. Data budr odeslána prostřednictvím HTTP požadavků a server bude zodpovědný za úpravu databáze a zasílání všeho zpět ve formátu JSON (Javascript Object Notation), což je velmi praktický formát, protože je čitelný pomocí JavaScriptu.

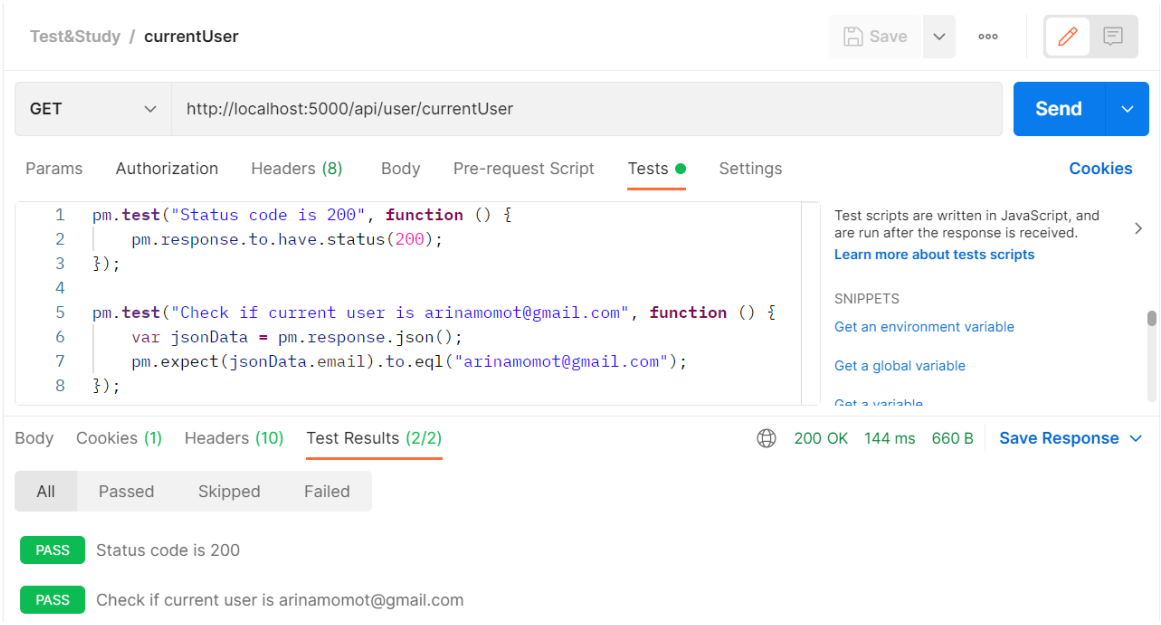
API jsem testovala i během vývoje pomocí užitečného nástroje Postman. Postman je populární API aplikace, která umožňuje vyvíjet, testovat a dokumentovat API. Pomocí Postman je možné odesílat HTTP požadavky službám a přijímat od nich odpovědi. Odpověď serveru obsahuje metadata o stavu a požadovaný obsah. Postman umožňuje otestovat backendové služby a ujistit se, že fungují správně nebo identifikovat chyby.

Nejprve jsem vytvořila kolekci **??**, kde jsou uloženy všechny požadavky(requests), které jsem testovala. Kolekce jsou skupiny požadavků, je možné si to představit jako složku, ve které umístěny požadavky. Dále jsem vytvořila různé HTTP požadavky (GET, POST, PUT, DELETE) na server, které bylo potřeba otestovat.

Pohled Postman kolekce a příklad testování POST požadavku login jsou uvedené v Přílohe B.

Postman také umožňuje vytvářet testy, které zajistí, že API funguje podle očekávání. Vytvořila jsem několik testů a zkontrolovala, zda funkce fungují správně. Příklad jednoduchého testu porovnávajícího očekávaný výsledek a skutečný je znázorněn na obrázku 5.1.

[1] URL (Uniform Resource Locator) - je soubor znaků, který slouží k identifikaci přesného umístění informací na internetu.



Obrázek 5.1: Jednoduchý příklad testu v Postman.

Postman umožňuje zahájit testování jak jednoho požadavku, tak celé kolekce najednou. Kolekci lze spustit dvěma způsoby: pomocí Collection Runner přímo v Postman nebo integrovat testování přímo do aplikace pomocí Newman. Vyzkoušela jsem oba způsoby.

Pro integraci spuštění testování kolekce do mého projektu jsem potřebovala exportovat celou kolekci ve JSON formátu. Pro správné spuštění jsem potřebovala exportovat i prostředí. Exportovaná data jsem umístil do složky test/api na straně serveru v projektu aplikace. Dále bylo nutné přidat speciální skript do package.json pro zahájení testování:

```
"scripts": {
    "test-api": "newman run test/api/Test&Study.postman_collection.json"
}
```

Po spuštění tohoto skriptu se zobrazí výsledky testů.

Výsledek: V procesu vývoje aplikace mi byl nástroj Postman velmi užitečný. Pomohl mi otestovat funkce serveru a databázi bez klientské části, lépe promyslet aplikační logiku a odstranit chyby. Při testování API hotové aplikace nebyly nalezeny žádné chyby.

5.2 Funkční testování

Funkční testování je proces vyhodnocování chování aplikace s cílem určit, zda se všechny vyvinuté funkce chovají podle očekávání. Pro správnou funkci produktu musí všechny procesy fungovat tak, jak je uvedeno v požadavcích: od rozlišení přístupových práv při autorizaci až po správný výstup ze systému.

Funkční testování lze provádět pomocí předem připravených testovacích scénářů nebo průzkumných testovacích metod. Pro testování vytvořené aplikace "Test&Study"jsem

se rozhodla napsat několik klíčových testovacích scénářů.

Testovací scénář je popis počátečních podmínek, vstupních dat, akcí uživatele a očekávaného výsledku. Byly vytvořeny na základě případů užití a jsou uložené v Priloha C - Testovací scénáře.

Při testování podle vytvořených scénářů jsem zaznamenala výsledky každého kroku nebo celého testovacího scénáře, zaznamenávala zjištěné chyby a další podpůrné informace.

Výsledkem testování bylo zjištění 4 závažných chyb a 3 nedostatků, které byly později opraveny.

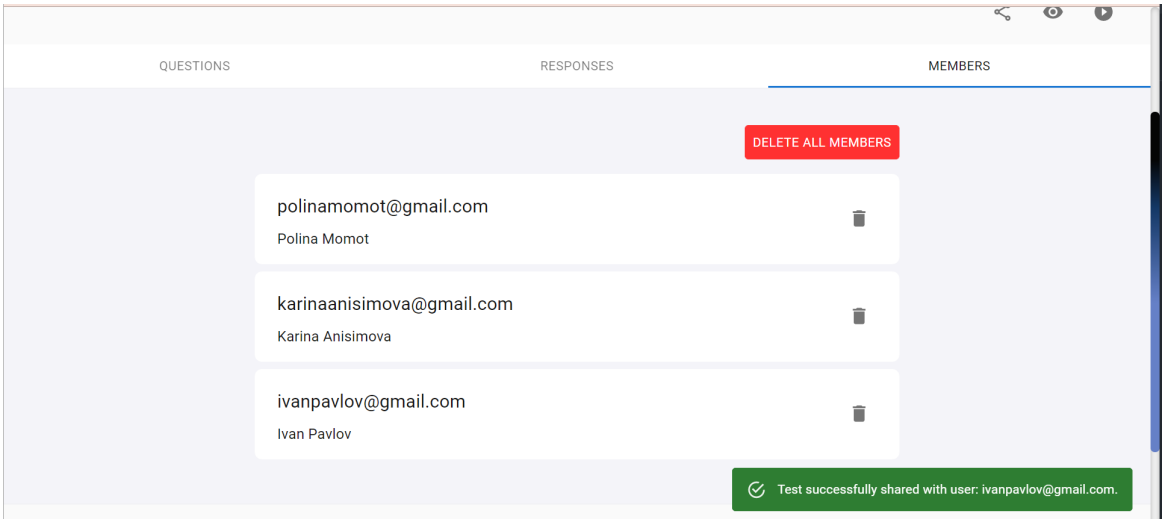
5.3 Uživatelské testování

Hlavním úkolem testování je posoudit zájem uživatelů o vytvářenou aplikaci, prověřit reakci na její funkčnost a design.

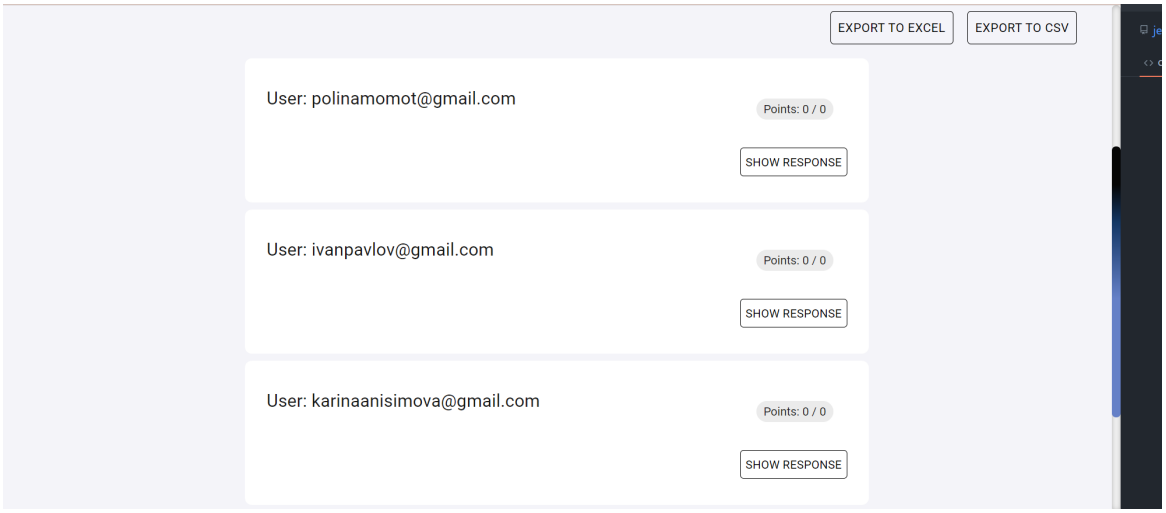
Vzhledem k tomu, že hlavním cílem mé aplikace je pohodlné testování, přišel mi nápad provést průzkum uživatelů přímo pomocí vytvořené aplikace. Tato metoda umožní uživatelům lépe se seznámit a otestovat aplikaci a poskytnout mi podrobné výsledky testování.
Pro realizaci mé myšlenky přímo v aplikaci jsem vytvořila test s otázkami:

1. Ohodnoťte aplikace od 1 do 5, kde 1 je hrozné a 5 je úžasné.
2. Je aplikace srozumitelná?
3. Pokud jste v předchozí otázce odpověděli ne, tak proč? Co lze zlepšit pro lepší pochopení webu?
4. Jak snadné je používání aplikace? Ohodnoťte od 1 do 5, kde 1 je naprosto nepochopitelné a 5 je vše jasné.
5. Využili byste tuto aplikaci pro sebepřípravu na testy?
6. Pokud jste v předchozí otázce odpověděli ne, tak proč? Co lze zlepšit, abyste aplikaci chtěli používat?
7. Našli jste při používání aplikace nějaké chyby? Pokud ano, popište je.
8. Líbí se vám design aplikace?
9. Napište prosím svá doporučení pro další rozvoj stránek.
10. Jaké pocity jste zažili při používání stránky?

Tohoto testování se zúčastnili lidé, kteří jsou cílovou skupinou aplikace – studenti. Také dva z nich mají technické vzdělání, a proto byli schopni zhodnotit i technickou implementaci aplikace. Uživatelské testování mi pomohlo pochopit uživatelskou zkušenost s používáním aplikace a testeři také dali svá doporučení pro další vývoj aplikace. Testeři zjistili chyby a nedostatky, které jsem také později opravila. Pomocí funkce pro uložení testu a odpovědí ve formátu PDF jsem exportovala výsledky uživatelského testování. PDF soubor s testem a odpověďmi účastníků testu je k dispozici v Priloha D - Seznam odkazů.



Obrázek 5.2: Účastníky uživatelského testování.



Obrázek 5.3: Odpovědi účastníků uživatelského testování.

Část VI

Závěr

Kapitola 6

Závěr

Hlavním cílem této bakalářské práce je analýza, návrh a implementace webové aplikace Test&Study pro vytváření a provádění online testování.

Prvním krokem k vytvoření aplikace je detailní analýza, která byla provedena v první části této práce. Velkou a důležitou částí práce je podrobná analýza trhu konkurentů, na základě které byly identifikovány požadavky na vyvíjenou aplikaci. Díky pečlivě provedené analýze se mi podařilo důkladně přemýšlet a navrhnout nejlepší technologické řešení pro vytvoření aplikace. Během práce jsem prostudovala mnoho literatury a dalších informačních zdrojů, abych vyvinula multifunkční a správně fungující aplikaci. Po úspěšné implantaci aplikace jsem se pustila do její testování a vytvoření dokumentace. Testování pomohlo najít a odstranit drobné chyby a přemýšlet o způsobech dalšího rozšíření aplikace.

Podle mého názoru zadání bakalářské práce bylo splněno. Všechny cíle, požadavky a funkce byly realizovány, aplikace je plně funkční a připravená k použití.

6.1 Návrhy na další rozšíření aplikace

Na základě testování a mého osobního přání byly stanoveny body pro další vývoj aplikace:

- Přidání dalších typů otázek do testu.
- Přidání možnosti vkládání různých typů souborů do otázek testu.
- Zlepšení designu aplikace.
- Přidání možnosti vytvářet skupiny pro testování konkrétní skupiny uživatelů.
- Přidání možnosti náhodného generování pořadí otázek v testu.

Seznam použitých zkratek

- OS - Operating system – operační systém
- PC - Personal computer – osobní počítač
- SWOT analýza - Analýza silných (Strengths) a slabých (Weaknesses) stránek spolu s příležitostmi, (Opportunities) a hrozbami (Threats)
- JS - JavaScript
- MERN - MongoDB, Express.js, React.js, Node.js
- MEAN - MongoDB, Express.js, Angular.js, Node.js
- MEVN - MongoDB, Express.js, Vue.js, Node.js
- LAMP - Linux, Apache, MySQL, PHP
- JSON - JavaScript Object Notation
- API - Application Programming Interface
- PDF - Portable Document Format – souborový formát
- SPA - Single Page Application – jednostránková aplikace
- MS - Microsoft
- UI - User Interface – uživatelské rozhraní
- JSX - JavaScript Syntax Extension (JavaScript XML)
- XML - Extensible Markup Language
- HTTP - Hypertext Transfer Protocol
- XHR - XMLHttpRequest
- SQL - Structured Query Language
- USA - United States of America
- PaaS - Platform as a Service
- IaaS - Infrastructure as a Service
- SaaS - Software as a service
- AWS - Amazon Web Services
- DevOps - Development operations
- HTML - HyperText Markup Language

- ORM - Object-relational mapping
- ODM - Object Document Mapper
- IDE - Integrated Drive Electronics
- JWT - JSON Web Token
- XSS útoků Cross-site scripting
- CSRF - Cross-Site Request Forgery
- SSR - Server-Side Rendering

Literatura a zdroje

1. *ProProfs Quiz Maker tour* [online] [cit. 2021-11-20]. Dostupné z: https://www.proprofs.com/quiz-school/tour.php.
2. *KAHOOT!* [Online] [cit. 2021-11-20]. Dostupné z: https://kahoot.com/.
3. *Google Forms* [online] [cit. 2021-11-20]. Dostupné z: https://workspace.google.com/intl/en_ie/products/forms/.
4. *Microsoft Forms* [online] [cit. 2021-11-20]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-365/online-surveys-polls-quizzes.
5. *ClassMarker* [online] [cit. 2021-11-20]. Dostupné z: https://www.classmarker.com/online-testing/how-to-create-online-quiz/.
6. KAŇÁKOVÁ, Eva. *Jak efektivně vést porady.* U Prùhonu 22, 170 00 Praha 7: Grada Publishing, a.s., 2008. ISBN 978-80-247-1625-1.
7. *SWOT analýza.* [Online] [cit. 2021-11-25]. Dostupné z: https://managementmania.com/cs/swot-analyza.
8. ABRAN A. Moore J. W., Bourque P. Guide to the software engineering body of knowledge. In: IEEE Intellectual Property Rights Office, 445 Hoes Lane, Piscataway: Los Alamitos, CA IEEE Computer Society Press, 2004. ISBN 978-07-695-5166-1. Dostupné také z: https://www.math.unipd.it/~tullio/IS-1/2007/Approfondimenti/SWEBOK.pdf.
9. *UML - Online kurz* [online] [cit. 2021-12-10]. Dostupné z: https://www.itnetwork.cz/navrh/uml.
10. FOWLER, Martin. *Destilované UML.* U Prùhonu 22, 170 00 Praha 7: Grada Publishing, a.s., 2009. ISBN 978-80-247-2062-3.
11. *Use-case diagrams* [online] [cit. 2022-01-16]. Dostupné z: https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-use-case.
12. *What is a Document Database?* [Online] [cit. 2022-01-10]. Dostupné z: https://www.mongodb.com/document-databases.

13. OSADCHUK, Serhii. Node.JS vs PHP in 2022: Which One to choose for your project? [Online]. [B.r.] [cit. 2022-01-11]. Dostupné z: https://doit.software/blog/nodejs-vs-php#screen1.

14. HORBEI, Natalia. Node.js Vs. Java: Choosing Perfect Technology in 2022 [online]. [B.r.] [cit. 2022-01-11]. Dostupné z: https://keenethics.com/blog/node-js-vs-java-choosing-perfect-technology.

15. MARTIN, Sophia. Java or PHP: Which is the Best Choice For Web Development in 2021? [Online]. [B.r.] [cit. 2022-01-11]. Dostupné z: https://medium.com/javarevisited/java-or-php-which-is-the-best-choice-for-web-development-in-2020-66da2c63661c.

16. SAMSUKHA, Amit. PHP vs. Java: Differences Similarities for Web Development [online]. [B.r.] [cit. 2022-01-11]. Dostupné z: https://www.emizentech.com/blog/php-vs-java.html.

17. *Stack Overflow Annual Developer Survey* [online] [cit. 2021-12-25]. Dostupné z: https://insights.stackoverflow.com/survey/2021#technology.

18. PATEL, Jeel. Vue Vs Angular – The Ultimate Framework Comparison [online]. [B.r.] [cit. 2022-01-11]. Dostupné z: https://www.monocubed.com/blog/vue-vs-angular/.

19. KAPOOR, Ajay. MERN vs LAMP [online]. [B.r.] [cit. 2022-02-24]. Dostupné z: https://enlear.academy/mern-vs-lamp-f0653b0dc96a.

20. VAILSHERY, Lionel Sujay. Amazon Web Services: Annual revenue 2013-2021 [online]. [B.r.] [cit. 2022-01-14]. Dostupné z: https://www.statista.com/statistics/233725/development-of-amazon-web-services-revenue/.

21. CLARK, Jessica. DigitalOcean vs Heroku | Which is better? [Online]. [B.r.] [cit. 2022-01-14]. Dostupné z: https://blog.back4app.com/digitalocean-vs-heroku/#:~:text=Digital%20Ocean%20provides%20an%20easy,of%20a%20pure%20IaaS%20structure..

22. TAYLOR, David. Heroku vs AWS: What is the Difference? [Online]. [B.r.] [cit. 2022-01-14]. Dostupné z: https://www.guru99.com/heroku-vs-aws.html.

23. TAYLOR, David. Digitalocean vs AWS: 10 Most Important Differences You Must Know! [Online]. [B.r.] [cit. 2022-01-14]. Dostupné z: https://www.guru99.com/digitalocean-vs-aws.html.

24. HAVERBEKE, Marijn. *Eloquent JavaScript, 3rd Edition: A Modern Introduction to Programming*. No Starch Press, 2018. ISBN 978-1593279509.

25. CHERNY, Boris. *Programming TypeScript*. O'Reilly Media, Inc, USA, 2019. ISBN 978-1492037651.

26. SUBRAMANIAN, Vasan. *Pro MERN Stack*. Apress, 13. května 2019. ISBN 978-1484243909.

27. SMITH, Tim. What is the MERN stack and how do I use it? [Online]. [B.r.] [cit. 2022-01-14]. Dostupné z: https://www.iamtimsmith.com/blog/what-is-the-mern-stack-and-how-does-it-work.

28. *NodeJS Docs* [online] [cit. 2022-02-16]. Dostupné z: https://nodejs.org/en/docs/.

29. SYED, Basarat. *Beginning Node.js*. Apress, 25. listopadu 2014. ISBN 978-1484201886.

30. LIM, Greg. *Beginning MERN Stack: Build and Deploy a Full Stack MongoDB, Express, React, Node.js App*. Independently published, 21. června 2019. ISBN 979-8523625503.

31. *ExpressJS Docs* [online] [cit. 2022-02-16]. Dostupné z: https://expressjs.com/.

32. ACHARYA, Durga Prasad. What Are Single Page Applications? Examples, Frameworks, and More [online]. [B.r.] [cit. 2022-02-23]. Dostupné z: https://geekflare.com/single-page-applications/#:~:text=A%20Single%20Page%20Application%20(SPA,some%20of%20it%20needs%20updating..

33. ACCOMAZZO ANTHONY Murray Nathaniel, Lerner Ari. *Fullstack React*. Fullstack.IO, 12. září 2017. ISBN 978-0991344628.

34. *Diagram nasazení a diagram komponent* [online] [cit. 2021-12-29]. Dostupné z: https://www.fi.muni.cz/~buhnova/PV167/12_DiagramNasazeni.pdf.

35. *MERN Stack Explained* [online] [cit. 2022-01-16]. Dostupné z: https://www.mongodb.com/mern-stack.

36. *What Is MongoDB?* [Online] [cit. 2022-01-16]. Dostupné z: https://www.purestorage.com/fr/knowledge/what-is-mongodb.html#:~:text=MongoDB%20stores%20data%20objects%20in,unit%20of%20data%20in%20MongoDB..

37. *Database References* [online] [cit. 2022-01-16]. Dostupné z: https://www.mongodb.com/docs/manual/reference/database-references/.

38. BOYD, Ian. Algoritmus hašování hesel Bcrypt [online]. [B.r.] [cit. 2022-01-14]. Dostupné z: https://crypto.stackexchange.com/questions/6564/why-is-bcrypt-called-a-key-derivation-function.

39. *Provozní režim blokových šifer* [online] [cit. 2022-01-14]. Dostupné z: http://techpedia.fel.cvut.cz/html/frame.php?oid=51&pid=1004&finf=.

40. DIMITRIJE STAMENICAUTHOR David Landup, Jovana Ninkovic. Hashing Passwords in Python with BCrypt [online]. [B.r.] [cit. 2022-01-14]. Dostupné z: https://www.statista.com/statistics/233725/development-of-amazon-web-services-revenue/.

41. *Advanced JWT authorization on React and Node js.* [Online] [cit. 2022-01-16]. Dostupné z: https://youtu.be/fN25fMQZ2v0.

42. *Co je cross-site scripting* [online] [cit. 2022-01-14]. Dostupné z: https://www.sprava-site.eu/cross-site-scripting/.

43. *CSRF* [online] [cit. 2022-01-14]. Dostupné z: https://wiki.knihovna.cz/index.php/CSRF#cite_note-vrana-1.

44. VRÁNA, Jakub. Cross-Site Request Forgery [online]. [B.r.] [cit. 2022-01-14]. Dostupné z: https://php.vrana.cz/cross-site-request-forgery.php.

45. WATMORE'S, Jason. .NET 6.0 - JWT Authentication with Refresh Tokens Tutorial with Example API [online]. [B.r.] [cit. 2022-01-14]. Dostupné z: https://jasonwatmore.com/post/2022/01/24/net-6-jwt-authentication-with-refresh-tokens-tutorial-with-example-api.

46. *Refresh Token with Axios and JWT example* [online] [cit. 2022-01-17]. Dostupné z: https://www.bezkoder.com/vue-refresh-token/.

47. *Introduction to Mongoose for MongoDB* [online] [cit. 2022-01-17]. Dostupné z: https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/.

48. *How to Perform HTTP Requests with Axios – A Complete Guide* [online] [cit. 2022-02-11]. Dostupné z: https://www.atatus.com/blog/how-to-perform-http-requests-with-axios-a-complete-guide/#why-axios.

49. *MobX* [online] [cit. 2021-02-11]. Dostupné z: https://web-creator.ru/technologies/webdev/mobx.

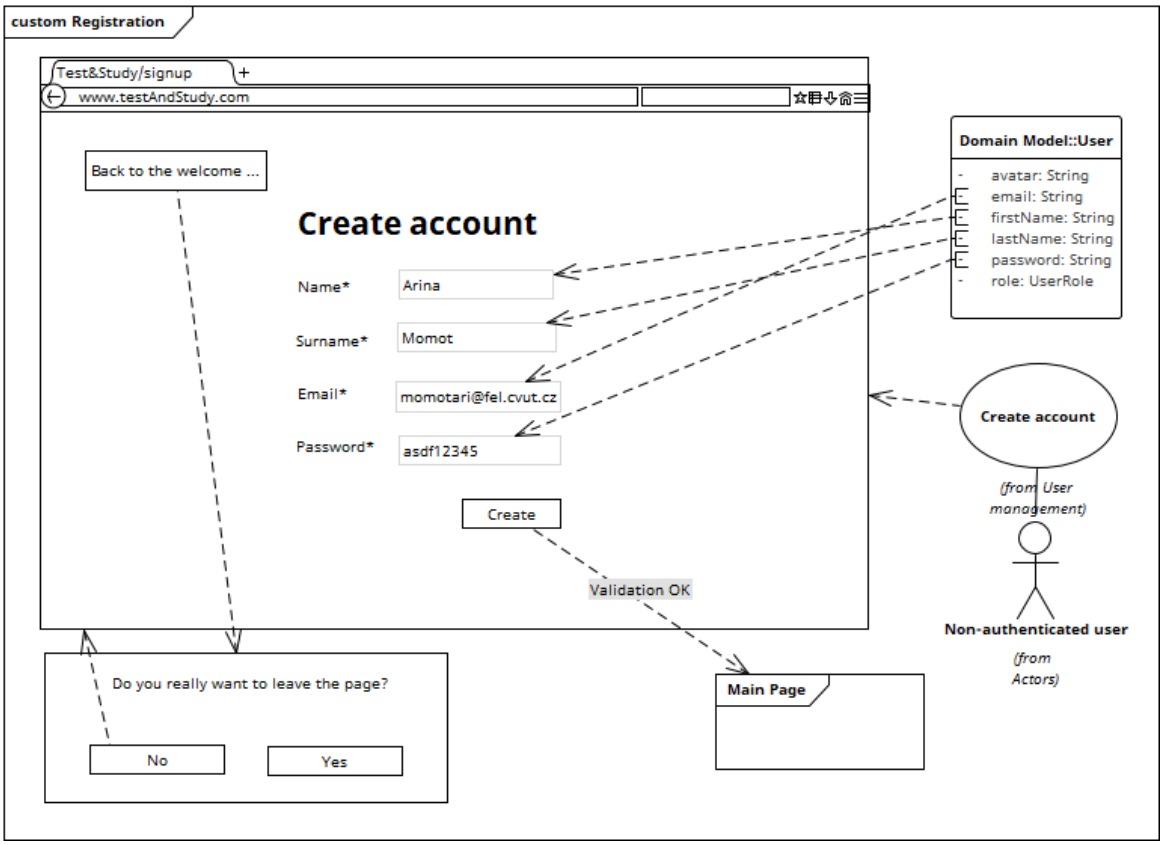
50. *What is Next.js and what is it for?* [Online] [cit. 2021-02-11]. Dostupné z: https://pxstudio.pw/blog/chto-takoe-next-js-i-dlya-chego-on-nuzhen.

51. *Nauč se X za Y minut* [online] [cit. 2021-02-11]. Dostupné z: https://learnxinyminutes.com/docs/cs-cz/sass/.

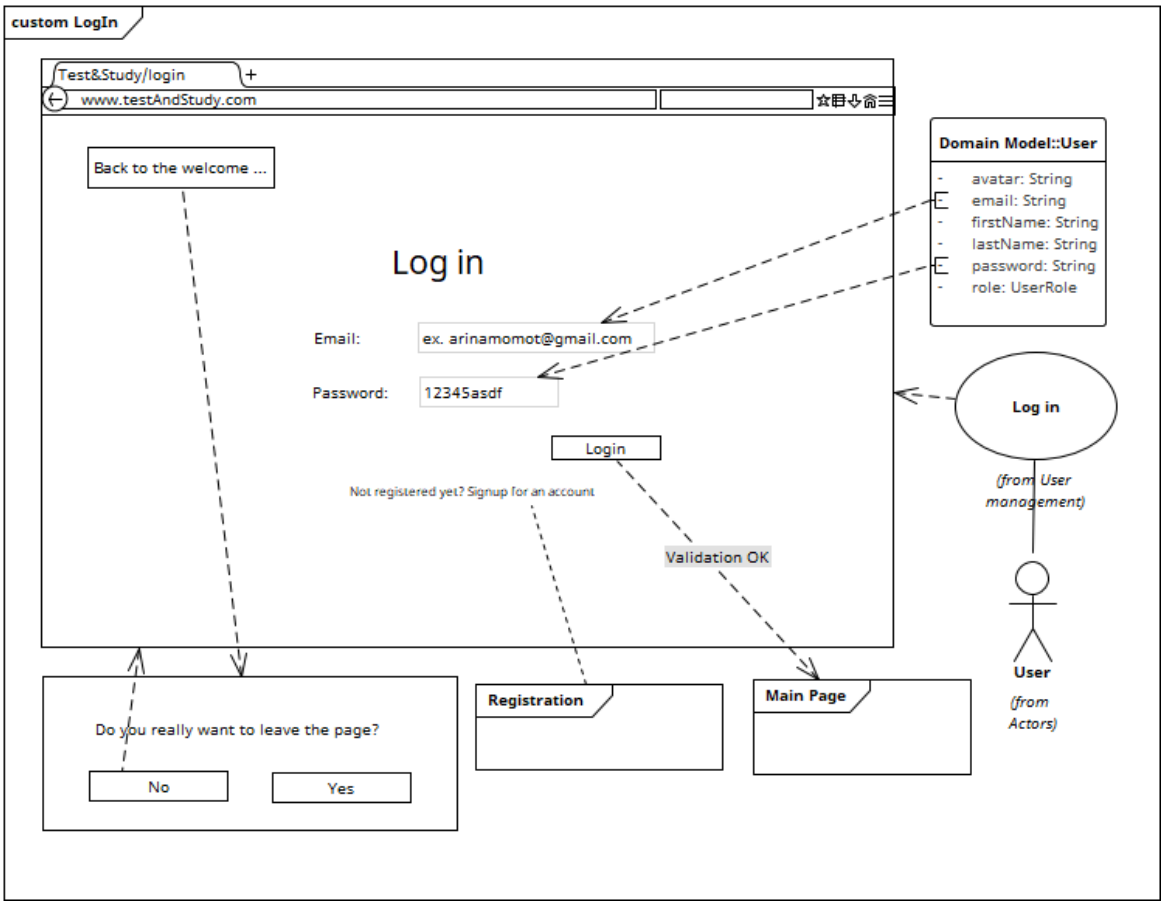
Přílohy

Příloha A

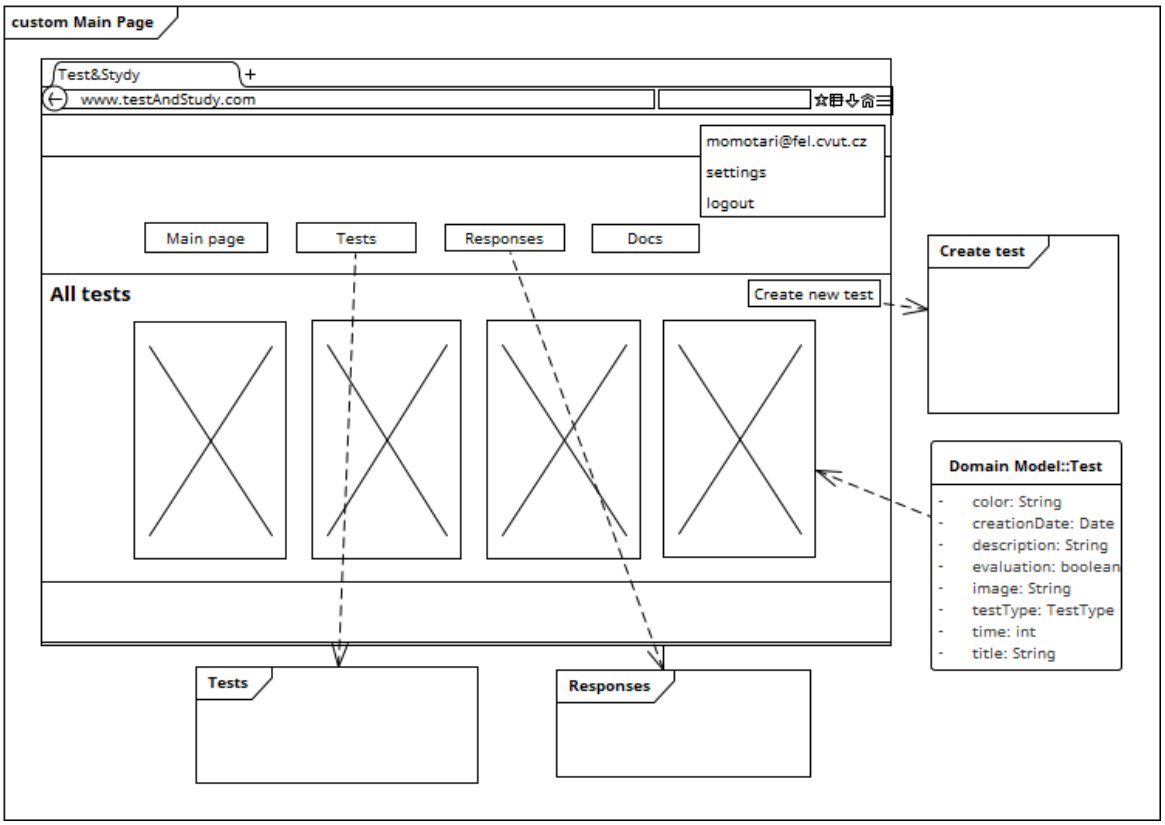
Priloha A - Wireframes



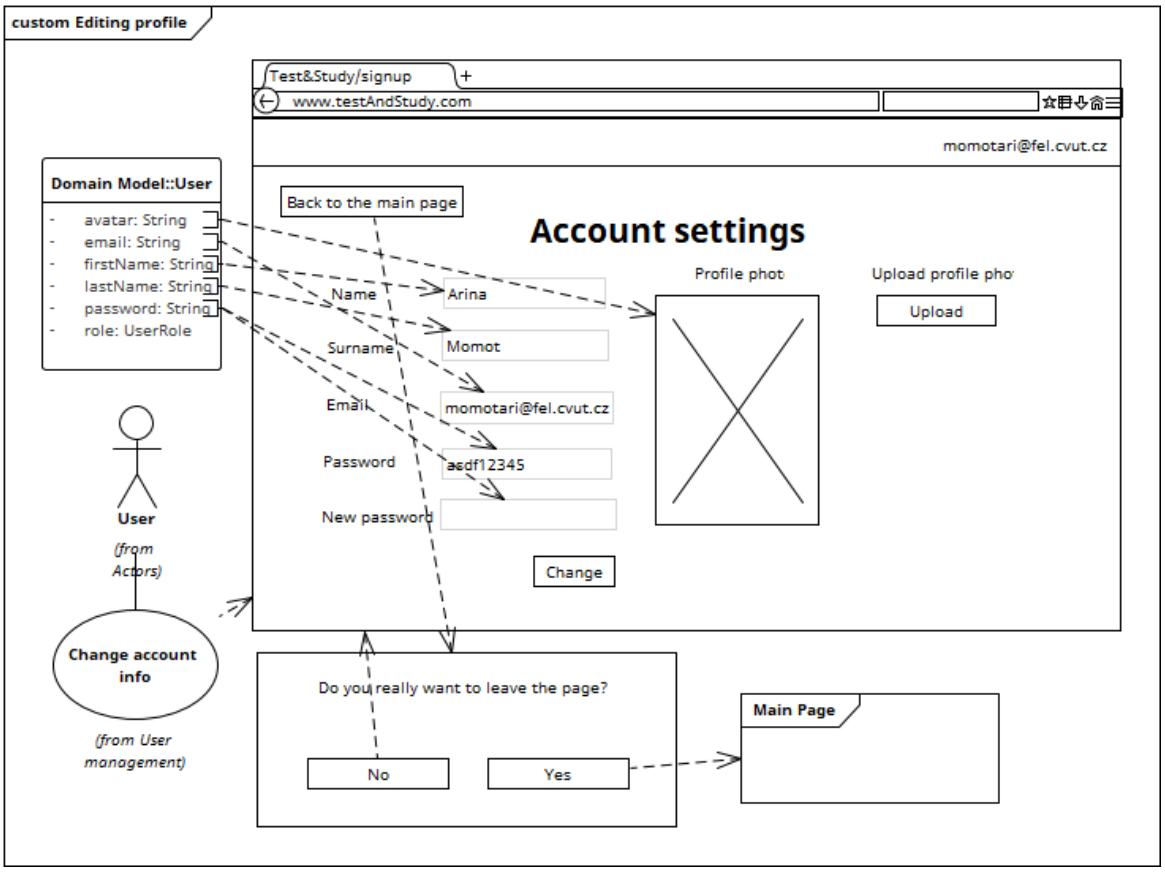
Obrázek A.1: Wireframe registrační stránky aplikace.



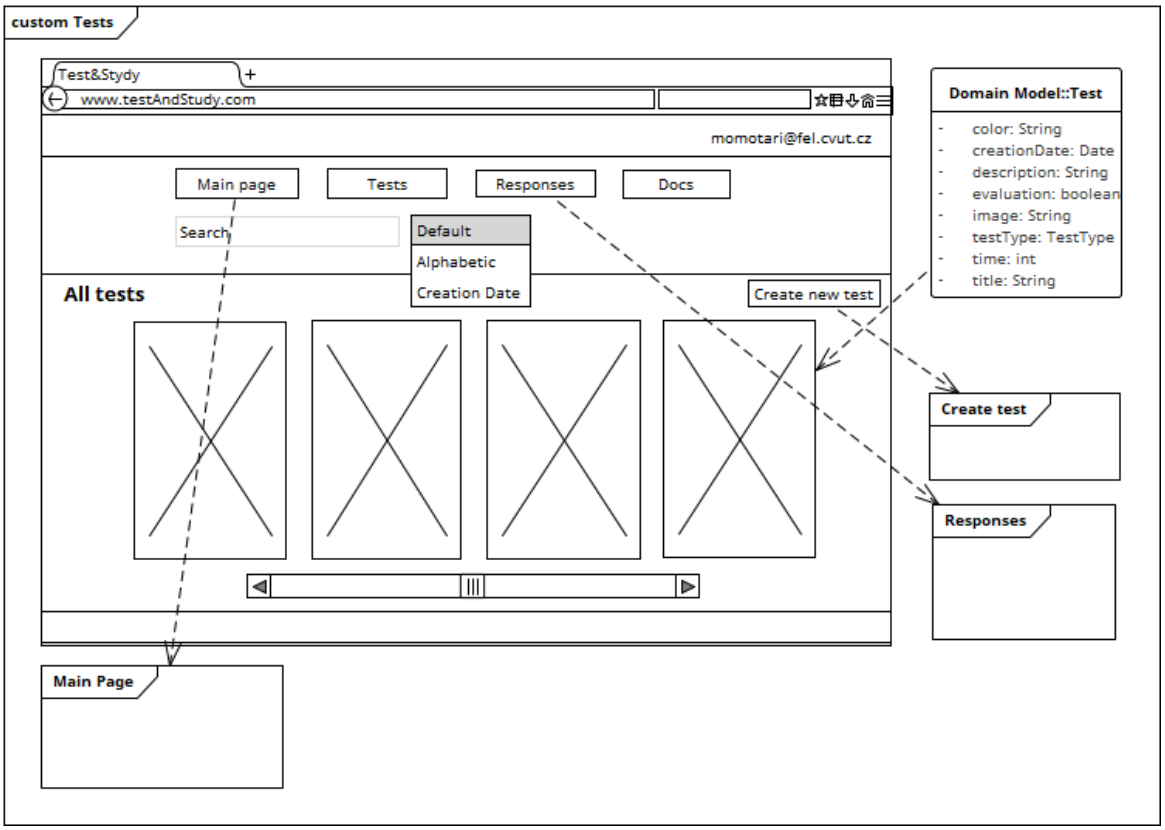
Obrázek A.2: Wireframe přihlašovací stránky aplikace.



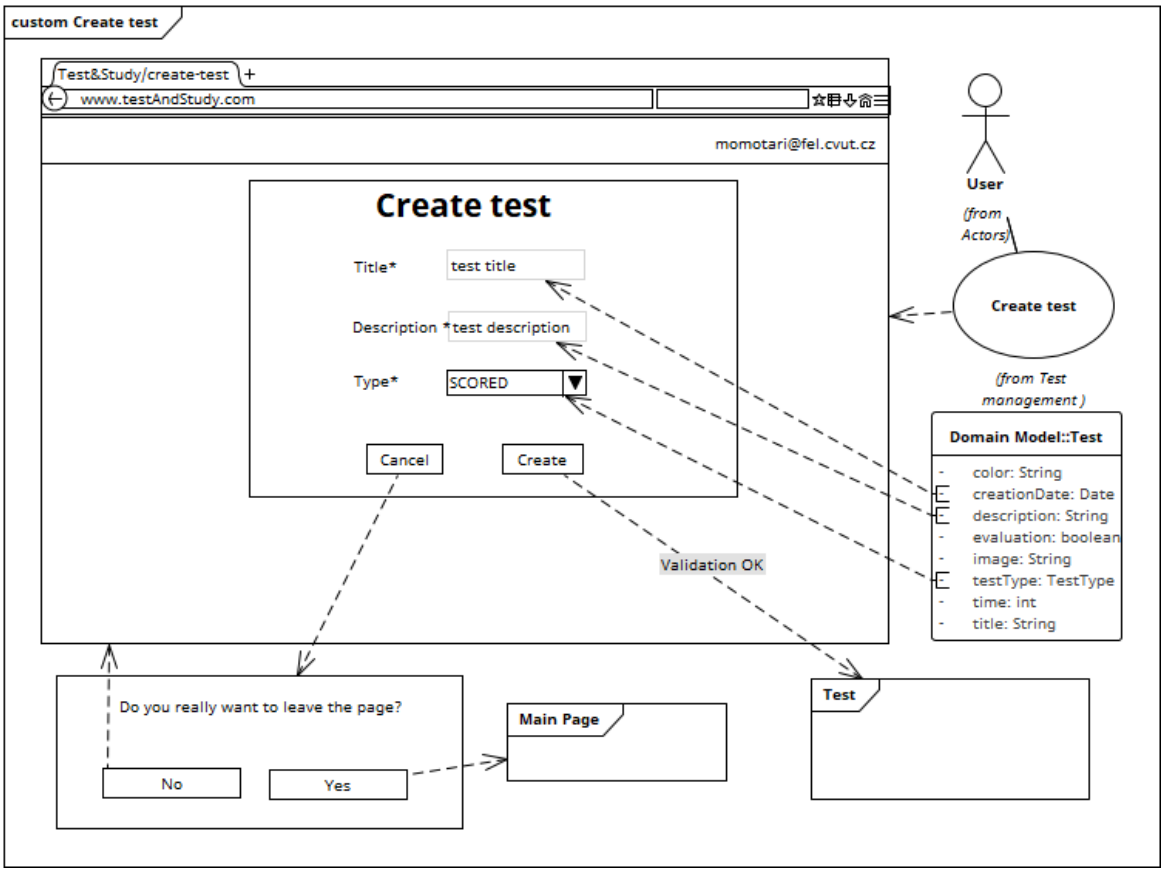
Obrázek A.3: Wireframe hlavní stránky aplikace.



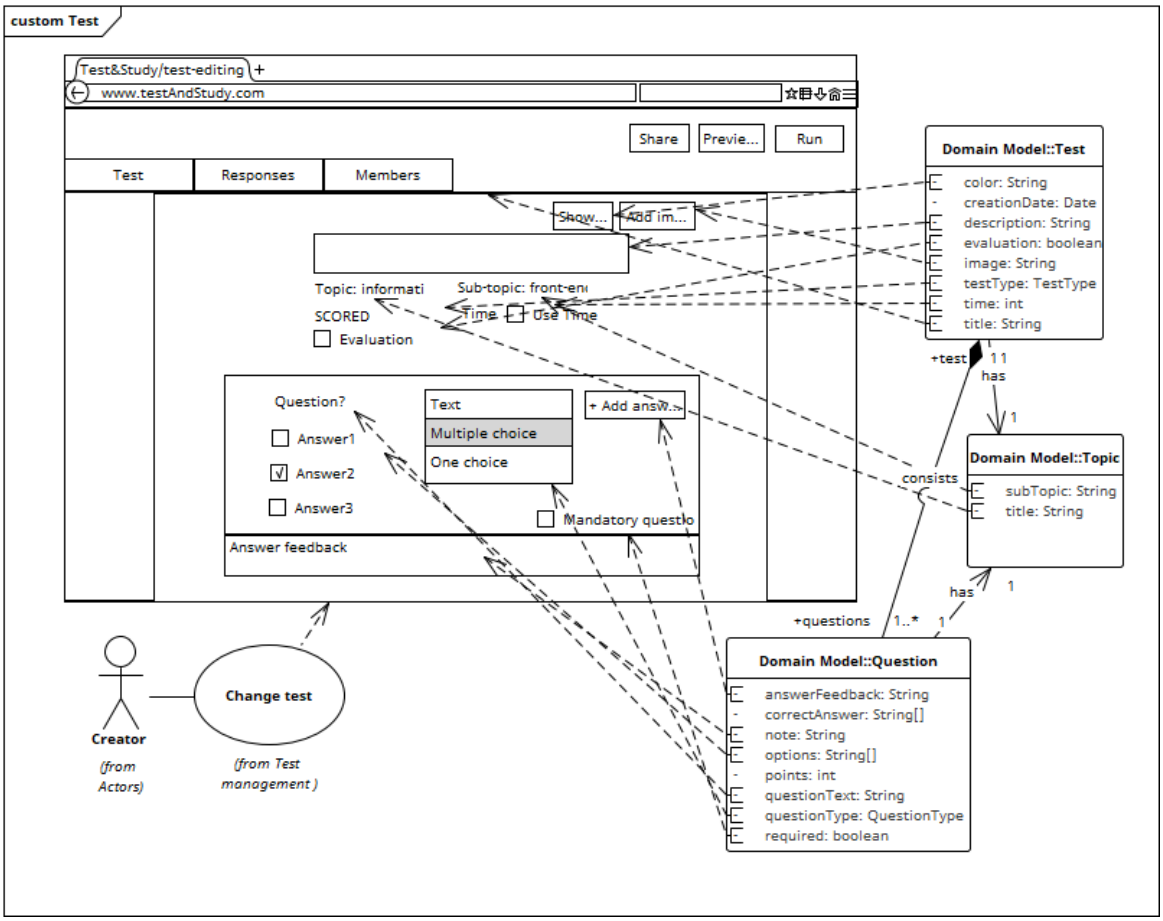
Obrázek A.4: Wireframe stránky profilu uživatele.



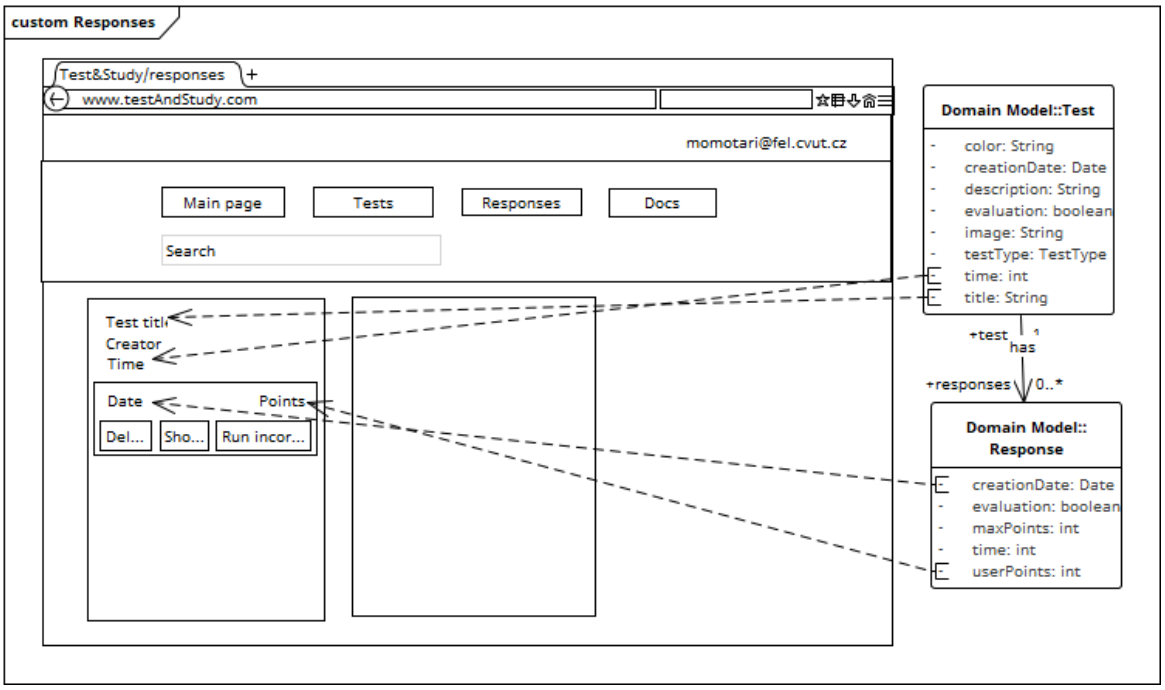
Obrázek A.5: Wireframe stránky se všemi uživatelskými testy.



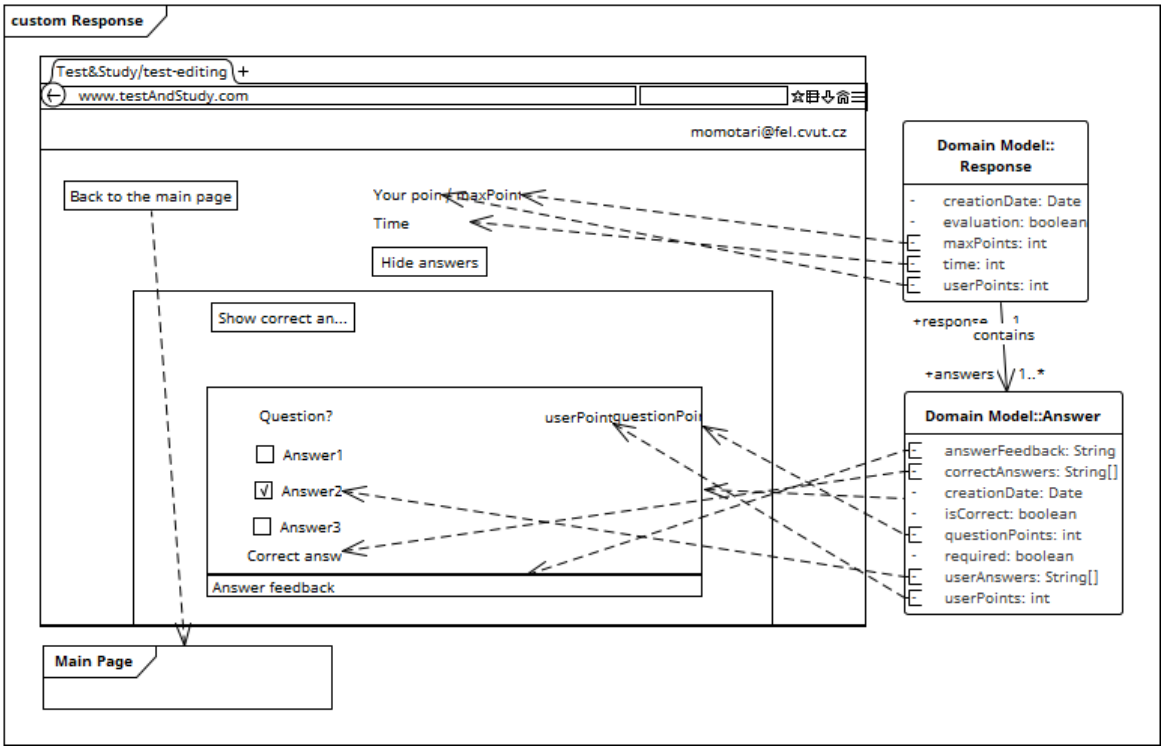
Obrázek A.6: Wireframe stránky s modálním oknem pro vytvoření testu.



Obrázek A.7: Wireframe stránky pro modifikaci testu.



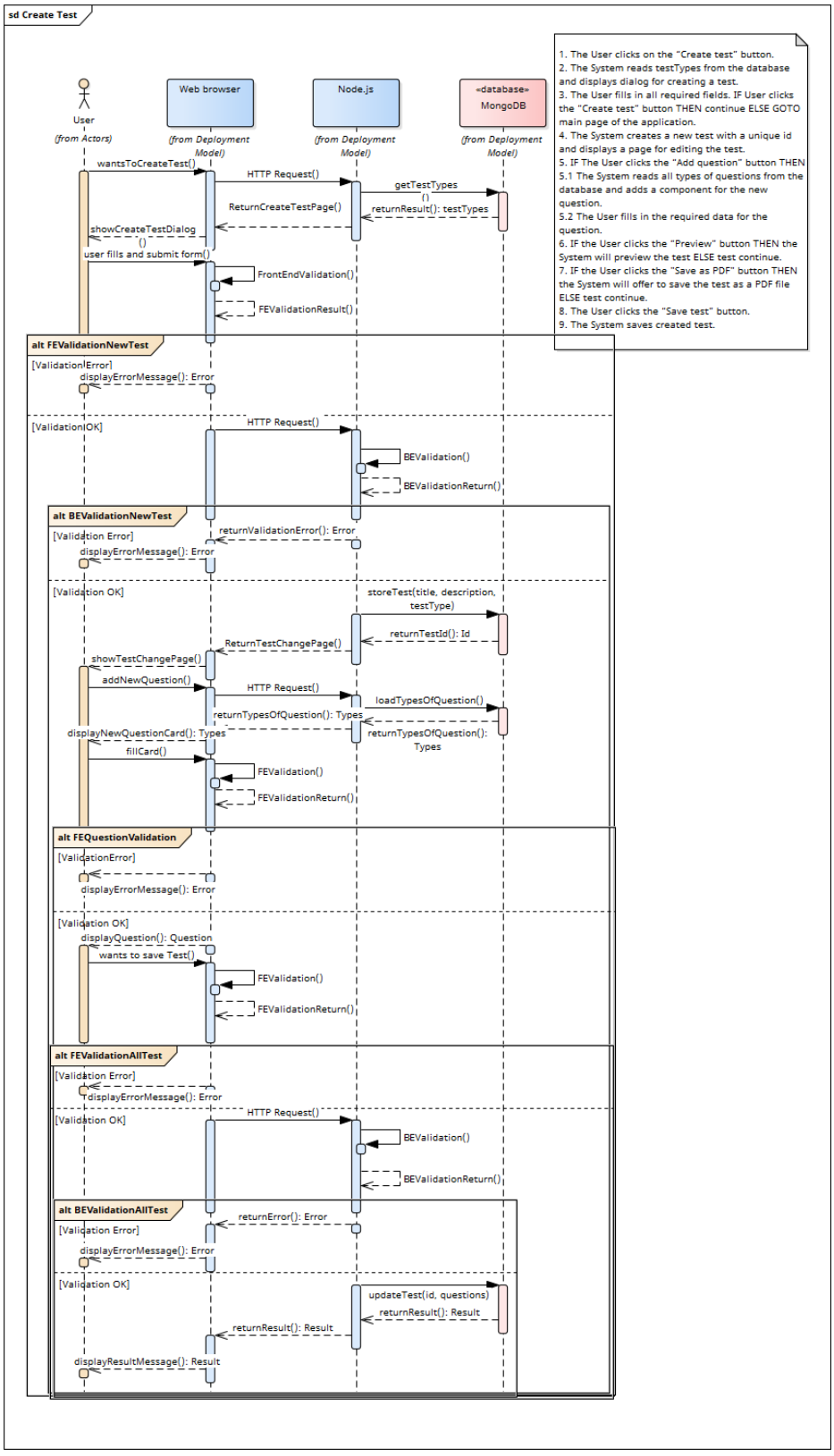
Obrázek A.8: Wireframe stránky se všemi odpověďmi na testy uživatele.



Obrázek A.9: Wireframe stránky s odpovědí na test.

Příloha B

Priloha B - Sekvenční diagram



Obrázek B.1: Sekvenční diagram vytváření testu.

Příloha C

Priloha C - Testovací scénáře

Parametr	Obsah
ID testu	1
Název testu	Úspěšná registrace v aplikaci
Krátký popis	Vytvoření účtu
Hloubka detailu	Střední
Shrnutí testu	Účet založen, pozitivní průchod
Popis testu	1. Uživatel spustí aplikaci. 2. Uživatel klikne na tlačítko “Try it now”. 3. Uživatel vyplní všechna povinná pole pro vytvoření nového účtu. 4. Uživatel klikne na “Terms of Use” a přečte si je. 5. Uživatel klikne na zaškrtávací políčko, čímž potvrdi svůj souhlas s podmínkami. 6. Uživatel klikne na tlačítko “Sign up”. 7. Uživatel se dostane do přihlašovací stránky.
Očekávaný výsledek	Nový účet uživatele bude úspěšně vytvořen a dostane se na přihlašovací stránku.
Vstupní podmínky	Všechny údaje byly vyplněné a byly ve správném formatu.
Testovací data	First Name: Arina Last Name: Momot E-mail Address: arinamomot@gmail.com Password: asd123asd

Obrázek C.1: Testovací scénář úspěšné registrace na webu.

Parametr	Obsah
ID testu	2
Název testu	Neúspěšná registrace v aplikaci
Krátký popis	Vytvoření účtu
Hloubka detailu	Střední
Shrnutí testu	Účet není založen, negativní průchod
Popis testu	1. Uživatel spustí aplikaci. 2. Uživatel klikne na tlačítko “Try it now”. 3. Uživatel vyplní všechna povinná pole pro vytvoření nového účtu. 4. Uživatel klikne na “Terms of Use” a přečte si je. 5. Uživatel klikne na zaškrtávací políčko, čímž potvrdi svůj souhlas s podmínkami. 6. Uživatel klikne na tlačítko “Sign up”. 7. Systém zobrazí chybovou hlášku: a. “User with email address ... already exists.” b. “Validation error.” - uživatel nevyplnil povinná pole nebo zadali údaje ve špatném formátu.
Očekávaný výsledek	Nový účet uživatele nebude vytvořen, zamitnutí transakci, zobrazení chybové hlášky.
Vstupní podmínky	a. Všechny údaje byly vyplněný v správném formatu, ale uživatel s touto e-mailovou adresou již existuje v DB. b. Údaje nebyly vyplněné. c. Všechny údaje byly vyplněný, ale ne v správném formatu.
Testovací data	First Name: Arina Last Name: Momot E-mail Address: arinamomot@gmail.com Password: asd123asd

Obrázek C.2: Testovací scénář neúspěšné registrace na webu.

Parametr	Obsah
ID testu	3
Název testu	Úspěšné přihlášení do aplikace
Krátký popis	Přihlášení do aplikace
Hloubka detailu	Střední
Shrnutí testu	Uživatel přihlášen, pozitivní průchod
Popis testu	1. Uživatel spustí aplikaci. 2. Uživatel klikne na tlačítko “Log in”. 3. Uživatel vyplní všechna povinná pole pro vytvoření přihlášení do aplikace. 4. Uživatel klikne na tlačítko “Log in”. 5. Uživatel se dostane do hlavní stránky aplikace.
Očekávaný výsledek	Uživatel se úspěšně přihlásí do svého účtu a dostane se na hlavní stránku aplikace.
Vstupní podmínky	Všechny údaje byly vyplněné a byly ve správném formatu.
Testovací data	E-mail Address: arinamomot@gmail.com Password: asd123asd

Obrázek C.3: Testovací scénář úspěšného přihlášení do webu.

Parametr	Obsah
ID testu	4
Název testu	Neúspěšně přihlášení do aplikace
Krátký popis	Přihlášení do aplikace -> zobrazení chybové hlášky.
Hloubka detailu	Střední
Shrnutí testu	Uživatel není přihlášen, negativní průchod
Popis testu	1. Uživatel spustí aplikaci. 2. Uživatel klikne na tlačítko “Log in”. 3. Uživatel vyplní všechna povinná pole pro vytvoření přihlášení do aplikace. 4. Uživatel klikne na tlačítko “Log in”. 5. Systém zobrazí chybovou hlášku a. “Invalid password.” - uživatel zadal nesprávné heslo. b. “User with email ... was not found.” - uživatel zadal neplatný e-mail. c. “Validation error.” - uživatel nevyplnil povinná pole nebo zadal údaje ve špatném formátu.
Očekávaný výsledek	Uživatel se nepřihlásí do svého účtu, zamitnutí transakci, zobrazení chybové hlášky.
Vstupní podmínky	a. Nesprávné heslo k účtu. b. Neexistující emailová adresa. c. Všechny údaje byly vyplněný, ale ne v správném formatu. d. Údaje nebyli vyplněny.
Testovací data	E-mail Address: arinaD2momot@gmail.com Password: 123asd123

Obrázek C.4: Testovací scénář neúspěšného přihlášení do webu.

Parametr	Obsah
ID testu	5
Název testu	Úspěšné odhlášení z aplikace.
Krátký popis	Odhlášení z aplikace.
Hloubka detailu	Nízká
Shrnutí testu	Uživatel odhlášen, pozitivní průchod
Popis testu	1. Přihlášený uživatel klikne na ikonu profilu v pravém horním rohu pro otevření dropdown menu. 2. Přihlášený uživatel klikne na tlačítko “Log out” v menu. 3. Uživatel bude přesměrován na uvítací stránku.
Očekávaný výsledek	Uživatel se úspěšně odhlásí ze svého účtu a bude přesměrován na uvítací stránku.
Vstupní podmínky	Uživatel je přihlášen do aplikace.
Testovací data	-

Obrázek C.5: Testovací scénář úspěšného odhlášení z webu.

Parametr	Obsah
ID testu	6
Název testu	Úspěšná změna osobních údajů.
Krátký popis	Změna osobních údajů.
Hloubka detailu	Střední
Shrnutí testu	Osobní údaje byly změněny, pozitivní průchod
Popis testu	1. Přihlášený uživatel klikne na ikonu profilu v pravém horním rohu pro otevření dropdown menu. 2. Uživatel klikne na tlačítko “My account” v menu. 3. Uživatel bude přesměrován na stránku s jeho osobními údaji. 4. Uživatel vyplní všechna pole, která je třeba změnit. 5. Uživatel klikne na tlačítko “Change”.
Očekávaný výsledek	Osobní údaje uživatele bude úspěšně změněny.
Vstupní podmínky	Uživatel je přihlášen do aplikace. Všechny údaje byly vyplněné a byly ve správném formatu.
Testovací data	First Name: Marina Last Name: Morton E-mail Address: marinamorton@gmail.com Current password: asd123asd New password: 123asd123

Obrázek C.6: Testovací scénář úspěšné změny osobních údajů.

Parametr	Obsah
ID testu	7
Název testu	Neúspěšná změna osobních údajů.
Krátký popis	Změna osobních údajů.
Hloubka detailu	Střední
Shrnutí testu	Osobní údaje nebyly změněny, negativní průchod
Popis testu	1. Přihlášený uživatel klikne na ikonu profilu v pravém horním rohu pro otevření dropdown menu. 2. Uživatel klikne na tlačítko “My account” v menu. 3. Uživatel bude přesměrován na stránku s jeho osobními údaji. 4. Uživatel vyplní všechna pole, která je třeba změnit. 5. Uživatel klikne na tlačítko “Change”. 6. Systém zobrazí chybovou hlášku a. “New password cannot be the same as your current password.” - uživatel zadal své aktuální heslo jako nové heslo. b. “Uživatel s tímto e-mailem již existuje.” - uživatel zadal email již existujícího uživatele. c. “Validation error.” - uživatel nevyplnil povinná pole nebo zadal údaje ve špatném formátu.
Očekávaný výsledek	Osobní údaje uživatele nebude změněny, zamitnutí transakci, zobrazení chybové hlášky.
Vstupní podmínky	Uživatel je přihlášen do aplikace. a. Aktuální heslo bylo zadáno jako nové heslo. b. Všechny údaje byly vyplněný v správném formatu, ale uživatel s touto e-mailovou adresou již existuje v DB. c. Údaje nebyly vyplněné. d. Všechny údaje byly vyplněný, ale ne v správném formatu.
Testovací data	First Name: M2arina Last Name: Morton_ E-mail Address: arinamomot@gmail.com Current password: asd123asd New password: asd123asd

Obrázek C.7: Testovací scénář neúspěšné změny osobních údajů.

Parametr	Obsah
ID testu	8
Název testu	Úspěšně nahrávání profilové fotografie.
Krátký popis	Nahrávání profilové fotky.
Hloubka detailu	Střední
Shrnutí testu	Profilová fotka byla nahrána, pozitivní průchod
Popis testu	1. Přihlášený uživatel klikne na ikonu profilu v pravém horním rohu pro otevření dropdown menu. 2. Uživatel klikne na tlačítko “My account” v menu. 3. Uživatel bude přesměrován na stránku s jeho osobními údaji. 4. Uživatel klikne na ikonu fotoaparátu. 5. Uživatel vybere požadovanou fotografii ze svého počítače. 6. Uživatel uvidí náhled fotografie. 7. Uživatel klikne na tlačítko “Upload”. 8. Fotografie se objeví v sekci “Current photo”
Očekávaný výsledek	Profilová fotka bude úspěšně nahrána.
Vstupní podmínky	Uživatel je přihlášen do aplikace. Fotografii je ve správném formátu.

Obrázek C.8: Testovací scénář úspěšného nahrávání profilové fotografie.

Parametr	Obsah
ID testu	9
Název testu	Neúspěšně nahrávání profilové fotografie.
Krátký popis	Nahrávání profilové fotky.
Hloubka detailu	Střední
Shrnutí testu	Profilová fotka nebyla nahrána, negativní průchod
Popis testu	1. Přihlášený uživatel klikne na ikonu profilu v pravém horním rohu pro otevření dropdown menu. 2. Uživatel klikne na tlačítko “My account” v menu. 3. Uživatel bude přesměrován na stránku s jeho osobními údaji. 4. Uživatel klikne na ikonu fotoaparátu. 5. Uživatel vybere požadovanou fotografii ze svého počítače. 6. Uživatel neuvidí náhled fotografie. 7. Uživatel klikne na tlačítko “Upload”. 8. Fotografie se neobjeví v sekci “Current photo”
Očekávaný výsledek	Profilová fotka nebude nahrána.
Vstupní podmínky	Uživatel je přihlášen do aplikace. Fotografii není ve správném formátu.
Testovací data	Fotografie není ve formátu PNG/JPEG/JPG

Obrázek C.9: Testovací scénář neúspěšného nahrávání profilové fotografie.

Parametr	Obsah
ID testu	10
Název testu	Úspěšně vytvoření testu
Krátký popis	Vytvoření testu
Hloubka detailu	Vysoká
Shrnutí testu	Test vytvořen, pozitivní průchod
Popis testu	1. Na hlavní stránce aplikace Uživatel klikne na tlačítko “Create new test”. 2. Uživatel vyplní všechna povinná pole pro vytvoření testu v modálním okně. 3. Uživatel klikne na tlačítko “Create”. 4. Uživatel bude přesměrován na stránku pro úpravu testu. 5. Uživatel klikne na tlačítko “Save”. 6. Uživatel klikne na tlačítko “Back to the main page”. 7. Uživatel se dostane zpět na hlavní stránku aplikace a vytvořený test uvidí v seznamu všech testů.
Očekávaný výsledek	Nový test bude úspěšně vytvořen a objeví se v seznamu všech testů.
Vstupní podmínky	Všechny údaje byly vyplněné a byly ve správném formatu.
Testovací data	Title: Test Description: New cool test. Type: SCORED

Obrázek C.10: Testovací scénář úspěšného vytvoření testu.

Parametr	Obsah
ID testu	11
Název testu	Neúspěšně vytvoření testu
Krátký popis	Vytvoření testu
Hloubka detailu	Vysoká
Shrnutí testu	Test nebyl vytvořen, negativní průchod
Popis testu	1. Na hlavní stránce aplikace Uživatel klikne na tlačítko “Create new test”. 2. Uživatel vyplní všechna povinná pole pro vytvoření testu v modálním okně. 3. Uživatel klikne na tlačítko “Create”. 4. Uživatel nebude přesměrován na stránku pro úpravu testu.
Očekávaný výsledek	Nový test nebude vytvořen a neobjeví se v seznamu všech testů. Zamitnutí transakci, zobrázení chybové hlášky.
Vstupní podmínky	Uživatel je přihlášen do aplikace. a. Údaje nebyly vyplněné. b. Všechny údaje byly vyplněný, ale ne v správném formatu.
Testovací data	Title: Testdwíěísufbafkl%90879 Description: *Více než 1000 znaků*. Type: -

Obrázek C.11: Testovací scénář neúspěšného vytvoření testu.

Parametr	Obsah
ID testu	12
Název testu	Úspěšná změna údajů testu.
Krátký popis	Změna údajů testu + změna barvy testu + nahrávání fotografie testu.
Hloubka detailu	Vysoká
Shrnutí testu	Údaje o testu byly změněny, pozitivní průchod
Popis testu	1. Přihlášený uživatel klikne na tlačítko “Change”. 2. Uživatel bude přesměrován na stránku pro úpravu testu. 3. Uživatel vyplní všechna pole, která je třeba změnit. 4. Uživatel klikne na ikonu palety v pravém horním rohu a změnit barvu testu. 5. Uživatel klikne na ikonu fotoaparátu v pravém horním rohu. 6. Uživatel vybere požadovanou fotografii ze svého počítače. 7. Uživatel uvidí náhled fotografie. 8. Uživatel klikne na tlačítko “Upload”. 9. Fotografie objeví v horní části popisu testu. 10. Uživatel klikne na tlačítko “Save”.
Očekávaný výsledek	Údaje o testu, barva a fotografie budou úspěšně změněny.
Vstupní podmínky	Uživatel je přihlášen do aplikace. Všechny údaje byly vyplněné a byly ve správném formatu.
Testovací data	Title: Test Description: New test. Topic: Biologie, Sub-topic: Animals Type: SCORED Time: 30 Color: beige Fotografie ve formátu PNG/JPEG/JPG

Obrázek C.12: Testovací scénář úspěšné změny údajů testu.

Parametr	Obsah
ID testu	13
Název testu	Neúspěšná změna údajů testu.
Krátký popis	Změna údajů testu + změna barvy testu + nahrávání fotografie testu.
Hloubka detailu	Vysoká
Shrnutí testu	Údaje o testu nebyly změněny, fotografie nebyla nahrána, negativní průchod
Popis testu	1. Přihlášený uživatel klikne na tlačítko “Change”. 2. Uživatel bude přesměrován na stránku pro úpravu testu. 3. Uživatel vyplní všechna pole, která je třeba změnit. 4. Uživatel klikne na ikonu palety v pravém horním rohu a změnit barvu testu. 5. Uživatel klikne na ikonu fotoaparátu v pravém horním rohu. 6. Uživatel vybere požadovanou fotografii ze svého počítače. 7. Uživatel neuvidí náhled fotografie. 8. Uživatel klikne na tlačítko “Upload”. 9. Uživatel klikne na tlačítko “Save”.
Očekávaný výsledek	Údaje o testu, barva a fotografie nebudou změněny.
Vstupní podmínky	Uživatel je přihlášen do aplikace. Fotografii není ve správném formátu.
Testovací data	Title: Testdwíěísufbafkl%90879 Description: *Více než 1000 znaků*. Type: - Topic: sak dsf 21cs f0n Sub-topic: dm dosaiod saiooa Time: -5 Color: beige Fotografie není ve formátu PNG/JPEG/JPG

Obrázek C.13: Testovací scénář neúspěšné změny údajů testu.

Parametr	Obsah
ID testu	14
Název testu	Přidání otázky do testu.
Krátký popis	Otevření testu -> přidání otázky
Hloubka detailu	Vysoká
Shrnutí testu	Otázka úspěšně přidána, pozitivní průchod
Popis testu	1. Přihlášený uživatel klikne na tlačítko “Change” u testu. 2. Uživatel bude přesměrován na stránku pro úpravu testu. 3. Uživatel klikne na ikonu plus pro přidání otázky do testu. 4. Uživatel vyplní všechna pole, která je potřeba pro otázku. 5. Uživatel klikne na tlačítko “Answer key”. 6. Uživatel vyplní všechna pole, která je potřeba pro odpověď. 7. Uživatel klikne na tlačítko “Add Answer Feedback”. 8. Uživatel vyplní toto pole. 9. Uživatel klikne na tlačítko “Done”. 10. Uživatel klikne na tlačítko “Save” pro ukládání testu.
Očekávaný výsledek	Otázka úspěšně přidána.
Vstupní podmínky	Uživatel je přihlášen do aplikace. Všechny údaje byly vyplněné a byly ve správném formatu.
Testovací data	Otázka: • Question text: The basic input device of the PC is the keyboard, the basic output device is: • Options: Monitir, Printer, Mouse • Type: Single choice, • Required • Topic: PC • Note: Easy question Odpověď: • Right answer: Monitor • Points: 2 • Answer Feedback: Monitor is the an output device that displays information in pictorial or text form.

Obrázek C.14: Testovací scénář úspěšného přidání otázky do testu.

Parametr	Obsah
ID testu	15
Název testu	Přidání otázky do testu.
Krátký popis	Otevření testu -> přidání otázky
Hloubka detailu	Vysoká
Shrnutí testu	Otázka nebyla přidána, negativní průchod
Popis testu	1. Přihlášený uživatel klikne na tlačítko “Change” u testu. 2. Uživatel bude přesměrován na stránku pro úpravu testu. 3. Uživatel klikne na ikonu plus pro přidání otázky do testu. 4. Uživatel vyplní všechna pole, která je potřeba pro otázku. 5. Uživatel klikne na tlačítko “Answer key”. 6. Uživatel vyplní všechna pole, která je potřeba pro odpověď. 7. Uživatel klikne na tlačítko “Add Answer Feedback”. 8. Uživatel vyplní toto pole. 9. Uživatel klikne na tlačítko “Done”. 10. Uživatel klikne na tlačítko “Save” pro ukládání testu. 11. Systém zobrazí chybovou hlášku.
Očekávaný výsledek	Otázka přidána. Test není aktualizován.
Vstupní podmínky	Uživatel je přihlášen do aplikace. Chybně zadané údaje.
Testovací data	Otázka: • Question text: *Více než 1000 znaků*. • Options: -, Printer, Mouse • Type: Single choice, • Required • Topic: *Více než 1000 znaků*. • Note: *Více než 1000 znaků*. Odpověď: • Right answer: - • Points: 10 (max 5) • Answer Feedback: *Více než 1000 znaků*.

Obrázek C.15: Testovací scénář neúspěšného přidání otázky do testu.

Parametr	Obsah
ID testu	16
Název testu	Kopírování a mazání otázky v testu.
Krátký popis	Otevření testu -> kopírování otázky -> mazání otázky
Hloubka detailu	Vysoká
Shrnutí testu	Otázka úspěšně kopírována a pak smazána, pozitivní průchod
Popis testu	1. Přihlášený uživatel klikne na tlačítko “Change” u testu. 2. Uživatel bude přesměrován na stránku pro úpravu testu. 3. Uživatel klikne na ikonu kopírovat v již existující otázce. 4. Objeví se nová zkopírovaná otázka. 5. Uživatel klikne na tlačítko “Save” pro ukládání testu. 6. Uživatel klikne na ikonu mazání ve zkopírované otázce. 7. Otázka zmizí. 8. Uživatel klikne na tlačítko “Save” pro ukládání testu.
Očekávaný výsledek	Otázka úspěšně kopírována a pak smazána.
Vstupní podmínky	Uživatel je přihlášen do aplikace. Uživatel má již vytvořený test. V testu už je otázka. Všechny údaje byly vyplněné a byly ve správném formatu.
Testovací data	-

Obrázek C.16: Testovací scénář úspěšného kopírování a mazání otázky v testu.

Parametr	Obsah
ID testu	17
Název testu	Sdílení testu s uživatelem.
Krátký popis	Otevření testu -> přidání účastníku
Hloubka detailu	Vysoká
Shrnutí testu	Test úspěšně sdílen s uživatelem, pozitivní průchod
Popis testu	1. Přihlášený tvůrce testů klikne na tlačítko “Change” u testu. 2. Tvůrce bude přesměrován na stránku pro úpravu testu. 3. Uživatel klikne na ikonu share v pravém horním rohu pro sdílení testu. 4. Objeví se modální okno. 5. Uživatel vyplní políčko pro hledání uživatele, s kterým chci sdílet test. 6. Zobrazí se seznam uživatelů. 7. Uživatel vybere požadovanou osobu ze seznamu a klikne na ni. 8. Uživatel klikne na tlačítko “Share” pro sdílení testu s uživatelem. 9. Uživatel se objeví mezi účastníky testu.
Očekávaný výsledek	Test úspěšně sdílen s uživatelem.
Vstupní podmínky	Uživatel je přihlášen do aplikace. Uživatel má již vytvořený test. Všechny údaje byly vyplněné a byly ve správném formatu.
Testovací data	User email for search: arinamomot@gmail.com

Obrázek C.17: Testovací scénář úspěšného sdílení testu s uživatelem.

Parametr	Obsah
ID testu	18
Název testu	Sdílení testu s uživatelem.
Krátký popis	Otevření testu -> přidání účastníku
Hloubka detailu	Vysoká
Shrnutí testu	Test není sdílen s uživatelem, negativní průchod
Popis testu	1. Přihlášený tvůrce testů klikne na tlačítko "Change" u testu. 2. Tvůrce bude přesměrován na stránku pro úpravu testu. 3. Uživatel klikne na ikonu share v pravém horním rohu pro sdílení testu. 4. Objeví se modální okno. 5. Uživatel vyplní políčko pro hledání uživatele, s kterým chci sdílet test. 6. Zobrazí se seznam uživatelů. 7. Uživatel vybere požadovanou osobu ze seznamu a klikne na ni. 8. Uživatel klikne na tlačítko "Share" pro sdílení testu s uživatelem. 9. Systém zobrazí chybovou hlášku a uživatel se neobjeví mezi účastníky testu.
Očekávaný výsledek	Test není sdílen s uživatelem.
Vstupní podmínky	Uživatel je přihlášen do aplikace. Uživatel má již vytvořený test.
Testovací data	User email for search: • Platná data, ale uživatel je již účastníkem testu: arinamomot@gmail.com • Špatný email nebo prázdné pole

Obrázek C.18: Testovací scénář neúspěšného sdílení testu s uživatelem.

Parametr	Obsah
ID testu	19
Název testu	Složení testu.
Krátký popis	Otevření testu -> spuštění testu
Hloubka detailu	Vysoká
Shrnutí testu	Test úspěšně složen, pozitivní průchod
Popis testu	1. Přihlášený uživatel testů klikne na tlačítko "Run" u testu. 2. Uživatel bude přesměrován na stránku pro složení testu. 3. Uživatel odpoví na všechny potřebné otázky. 4. Uživatel klikne na tlačítko "Submit" pro ukončení testu. 5. Uživatel bude přesměrován na stránku s výsledky testu. 6. Uživatel klikne na tlačítko "Show answers" pro zobrazení svých odpovědí. 7. Uživatel klikne na tlačítko "Show correct answers" pro zobrazení správných odpovědí. 8. Uživatel klikne na tlačítko "Save as PDF" pro ukládání vysledku testu v formátu PDF.
Očekávaný výsledek	Test úspěšně složen.
Vstupní podmínky	Uživatel je přihlášen do aplikace. Uživatel má již vytvořený test. Všechny potřebné otázky byly odpovědné.
Testovací data	-

Obrázek C.19: Testovací scénář úspěšného složení testu.

Parametr	Obsah
ID testu	20
Název testu	Složení testu.
Krátký popis	Otevření testu -> spuštění testu
Hloubka detailu	Vysoká
Shrnutí testu	Test nebyl složen, negativní průchod
Popis testu	1. Přihlášený uživatel testů klikne na tlačítko “Run” u testu. 2. Uživatel bude přesměrován na stránku pro složení testu. 3. Uživatel neodpoví na všechny potřebné otázky. 4. Uživatel klikne na tlačítko “Submit” pro ukončení testu. 5. Systém zobrazí chybovou hlášku. 6. Po vypršení času test bude automatické ukončen.
Očekávaný výsledek	Test automatické ukončen.
Vstupní podmínky	Uživatel je přihlášen do aplikace. Uživatel má již vytvořený test. Všechny potřebné otázky nebyly odpovědné.
Testovací data	-

Obrázek C.20: Testovací scénář neúspěšného složení testu.

Parametr	Obsah
ID testu	21
Název testu	Složení testu pouze ze špatných odpovědí.
Krátký popis	Otevření stránky odpovědí na test -> hledání správného testu a odpovědi -> spuštění testu pouze ze špatných odpovědí
Hloubka detailu	Vysoká
Shrnutí testu	Test úspěšně složen, pozitivní průchod
Popis testu	1. Přihlášený uživatel testů klikne na tlačítko “Responses” na hlavní stránce aplikace. 2. Uživatel bude přesměrován na stránku s odpovědí. 3. Pomocí vyhledávání uživatel najde požadovaný test a odpověď. 4. Uživatel klikne na tlačítko “Run incorrect” pro spuštění testu pouze ze špatných odpovědí. 5. Uživatel bude přesměrován na stránku pro složení testu. 6. Uživatel odpoví na všechny potřebné otázky. 7. Uživatel klikne na tlačítko “Submit” pro ukončení testu. 8. Uživatel bude přesměrován na stránku s výsledky testu. 9. Uživatel klikne na tlačítko “Show answers” pro zobrazení svých odpovědí. 10. Uživatel klikne na tlačítko “Show correct answers” pro zobrazení správných odpovědí.
Očekávaný výsledek	Test úspěšně složen.
Vstupní podmínky	Uživatel je přihlášen do aplikace. Uživatel má již vytvořený test a odpověď s chybami. Všechny potřebné otázky byly odpovědné.
Testovací data	Pro hledání testu: Information technologies

Obrázek C.21: Testovací scénář úspěšného složení testu pouze ze špatných odpovědí.

Příloha D

Priloha D - Seznam odkazů

Github repositář:

- Zdrojový kód: `https://github.com/arinamomot/test-and-study/tree/main/test-and-study`
- Výsledky uživatelského testování: `https://github.com/arinamomot/test-and-study/tree/main/userSurvey`
- Uživatelský manuál: `https://github.com/arinamomot/test-and-study/tree/main/manual`

Heroku: `https://test-and-study.herokuapp.com/`