**Spaced Cards**

Projekt do VIS

**ARTEFAKT 1 – VIZE**

**Proč?**

Je třeba vytvořit moderní a uživatelsky přívětivý systém SpacedCards, který umožní efektivní učení pomocí flashkaret. Tento systém nejenže motivuje uživatele blokováním přístupu k rozptylujícím aplikacím, dokud neprojdou zadaný počet karet, ale také nabízí možnost generovat flashkarty pomocí AI na základě specifických témat. Naším cílem je zlepšit proces učení, podpořit soustředěnost uživatelů a nabídnout nástroj, který je přizpůsobitelný jejich potřebám.

**Co?**

Aplikace umožňuje učení pomocí flashkaret. Uživatel si může vytvářet vlastní balíčky karet nebo využívat AI k automatickému generování obsahu. Kromě toho má aplikace integrované propojení se Screen Time, které omezuje přístup k jiným aplikacím, dokud uživatel neprojde určený počet karet.

**Jak?**

Realizace probíhá prostřednictvím mobilní aplikace vytvořené ve Flutteru s použitím FSRS algoritmů. Data jsou ukládána a spravována v cloudové databázi. UI bude intuitivní a zaměřené na jednoduchost ovládání.

**Kde?**

Aplikace bude dostupná pro iOS a později případně i pro Android. Přístup bude možný odkudkoliv díky cloudové architektuře a uživatelé si ji mohou stáhnout přímo z App Store.

**Kdo?**

Primárními uživateli jsou studenti, kteří chtějí zlepšit své studijní výsledky. Sekundární cílovou skupinou jsou profesionálové hledající způsob, jak si efektivně osvojit nové dovednosti nebo informace případňě si je průběžně opakovat.

**Kdy?**

Aplikace musí být neustále funkční, s minimálními výpadky pro údržbu. Aktualizace budou probíhat pravidelně, aby bylo možné reagovat na zpětnou vazbu uživatelů a přidávat nové funkce. Systém musí být schopen zvládnout nápor uživatelů, zejména během zkouškových období, kdy bude pravděpodobně intenzivně využíván.

**ARTEFAKT 2 – Funkční specifikace**

Use case diagram

A diagram of a software development process

Description automatically generated

**Případ užití UC2: Vytvoření decku**

* **Název:** Vytvoření balíčku flashkaret
* **Kontext:** Uživatel vytvoří nový balíček flashkaret zadáním názvu balíčku a přidáním jednotlivých karet s obsahem (otázka/odpověd). Systém uloží balíček do databáze a synchronizuje jej s ostatními zařízeními.
* **Účastníci:** Uživatel
* **Vstupní podmínky:** Uživatel je přihlášený do systému.
* **Výstup:** Nový balíček je uložen v databázi a v případě prémiového uživatele synchronizován do cloudu.

**Hlavní průběh:**

1. Uživatel klikne na možnost „New Deck“.
2. Systém zobrazí formulář pro vytvoření balíčku.
3. Uživatel zadá název balíčku.
4. Uživatel vybere barvu a emoji balíčku.
5. Uživatel zvolí počet nových karet, které bude denně studovat.
6. Systém uloží nový balíček do databáze.
7. Systém synchronizuje balíček s cloudem (v případě prémiového uživatele).
8. Uživatel obdrží potvrzení o úspěšném vytvoření balíčku.

**Alternativní průběh:**

1. Uživatel nezadá název balíčku.
2. Systém upozorní na nutnost vyplnit pole a čeká na zadání dat
3. Uživatel zadá název a pokračuje v tvorbě.

A diagram of a deck

Description automatically generated

**Případ užití UC9: Generování karet pomocí AI**

* **Název:** Generování karet pomocí AI
* **Kontext:** Uživatel zádá téma a požadovaný rozsah flashkaret. Systém vytvoří obsah flashkaret pomocí AI modelu a uloží je do balíčku.
* **Účastníci:** Prémiový úživatel
* **Vstupní podmínky:** Uživatel je přihlášený a má zakoupené premium
* **Výstup:** Vygenerované karty se po potvrzení uživatelem přidají do decku

**Hlavní průběh:**

1. Uživatel klikne na možnost „AI“ během tvorby karet.
2. Systém zobrazí formulář pro zadání tématu a rozsahu.
3. Uživatel zadá téma a potvrdí akci.
4. Systém odešle data společně s promptem AI.
5. AI modul vrátí seznam navržených karet.
6. Uživatel odstraní či upraví jednotlivé vygenerované kartičky.
7. Uživatel potvrdí uložení balíčku do systému.

**Alternativní průběh:**

1. AI modul vrátí chybu.
2. Systém upozorní uživatele, že generace neproběhla vpořádku.
3. Uživatel změní parametry a požádá o nové generování.

A diagram of a diagram

Description automatically generated with medium confidence

**Případ užití UC5: Tvorba kartiček**

* **Název:** Tvorba kartiček
* **Kontext:** Uživatel se proklikne na požadovaný deck a začne vytvářet jednotlivé kartičky. Do kartiček píše otázky a odpovědi.
* **Účastníci:** Uživatel
* **Vstupní podmínky:** Uživatel je přihlášený.
* **Výstup:** Nové kartičky v decku, které jsou uloženy do databáze.

**Hlavní průběh:**

1. Uživatel otevře požadovaný deck
2. Uživatel zaklikne sekci pro tvorbu flashkaret.
3. Uživatel vyplní obsah karty (otázku a odpověď) a uloží ji do balíčku.
4. Uživatel přidává další karty opakováním kroku 4.
5. Uživatel dokončí tvorbu balíčku kliknutím na tlačítko zpět.
6. Systém uloží kartičky do databáze.

**Alternativní průběh:**

1. Uživatel nezadá otázku či odpověď kartičky.
2. Systém neumožní vytvoření nové kartičky.
3. Uživatel doplní název či odpověď kartičky.
4. Uživatel přidá kartičku do decku.
5. Systém uloží kartičku do databáze.

A diagram of a work flow

Description automatically generated

**ARTEFAKT 3 – Funkční specifikace**

**První model domény:**

A screenshot of a computer

Description automatically generated

**Použité technologie a postupy:**

* + Frontend
    - Jako programovací jazyk jsem si zvolil flutter, jelikož umožňuje vytvářet multiplatformní aplikace (web, iOS, android).
    - Má aplikace je offlien first, tudíž používám SQLite jako “offline” databázi a PowerSync, která synchronizuje data s remote databází
  + Backend
    - Jako hlavní databázi jsem zvolil Supabase, jelikož se celkem jednoduše napojuje na mou offline first databázi. Supabase dále poskytuje robustní backend logiku a různé funkce, které jsou v aplikaci použity.
    - Pro generaci AI kartiček využívám API od Gemini (Google)

**ARTEFAKT 4 – Skica**

A screenshot of a mobile application

Description automatically generated

A screenshot of a computer screen

Description automatically generatedA screenshot of a screenshot of a mobile device

Description automatically generated

**ARTEFAKT 5 – Doménový model**

**Třídní diagram:**

A screenshot of a computer

Description automatically generated

**Interakce:**

* + Anonymní uživatel
    - Uživatel, který není přihlášený. Má značně omezený přístup k naší aplikaci, ovšem core funkcionilata je pro něj přístupná.
    - Vytváří decky, upravuje je, maže je
    - Vytváří kartičky, upravuje je, maže je
    - Studijní řežim s použitím FSRS algoritmů
  + Přihlášený uživatel
    - Má přístup ke všem funkcím aplikace, ale omezeně
    - Jeho data si může synchronizovat s remote úložištěm
    - Může sdílet své decky s ostatními uživateli
    - Má omezený počet využití AI generace
  + Prémiový uživatel
    - Má přístup ke všem funkcím aplikace a to bez omezení
    - AI generace karet bez omezení

**Sekvenční diagram:**

**A screenshot of a computer

Description automatically generated**

**Použité vzory:**

**Provider Pattern:** Umožňuje sdílet a spravovat stav nebo závislosti mezi komponentami v aplikaci.

**Builder Pattern:** Odděluje konstrukci složitých objektů od jejich reprezentace, takže stejný proces může vytvořit různé objekty.

**Factory Pattern:** Vytváří objekty bez nutnosti specifikovat konkrétní třídu, která je instanciuje.

**Riverpod (Observer) Pattern:** Umožňuje objektům reagovat na změny ve stavu jiného objektu formou notifikací.

**Command Pattern:** Zapouzdřuje příkazy do objektů, čímž umožňuje jejich flexibilní volání, ukládání nebo vrácení zpět.

**ARTEFAKT 6 – Architektura projektu**

**A diagram of a software development

Description automatically generated**

**Presentation Layer**

**Widgets:** Odpovídají za zobrazení uživatelského rozhraní a reagují na interakce uživatele. Data získávají od stavových managementů.

**States:** Spravují aktuální stav widgetů, například načtená data nebo chybové stavy.

**Controllers:** Obsahují logiku pro řízení akcí widgetů, jako je odesílání dat nebo volání služeb.

**Application Layer**

**Services:** Zprostředkovávají komunikaci mezi vrstvami a zajišťují obchodní logiku. Například spravují volání API nebo načítání dat z úložiště.

**Domain Layer**

**Models:** Reprezentují data a jejich logiku, například objekty, se kterými aplikace pracuje.

**Data Layer**

**Repositories:** Abstrahují přístup k datovým zdrojům a implementují jednotné rozhraní pro získávání dat.

**DTOs:** Slouží jako přechodné objekty pro přenos dat mezi aplikací a datovými zdroji.

**Data Sources:** Obsahují konkrétní implementaci přístupu k datům, například z API nebo databáze.