Mobilna aplikacija za samostalno učenje o matricama

Ledenčan, Tia

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:872906

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-10**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

MOBILNA APLIKACIJA ZA SAMOSTALNO UČENJE O MATRICAMA

Završni rad

Tia Ledenčan

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 24.08.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Tia Ledenčan
Studij, smjer:	Programsko inženjerstvo
Mat. br. Pristupnika, godina upisa:	R4528, 27.07.2020.
OIB Pristupnika:	42598949161
Mentor:	doc. dr. sc. Anita Katić
Sumentor:	doc. dr. sc. Bruno Zorić
Sumentor iz tvrtke:	
Naslov završnog rada:	Mobilna aplikacija za samostalno učenje o matricama
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak završnog rad:	U radu je potrebno izraditi mobilnu aplikaciju koja će korisniku omogućiti da samostalno usvoji osnovne pojmove i izvršavanje operacija s matricama, te da stečeno znanje provjeri. Treba opisati postupak izrade aplikacije, testirati ju na primjerima i prikazati rezultate. Tema rezervirana za: Tia Ledenčan Sumentor s FERIT-a: Bruno Zorić
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	24.08.2023.
Datum potvrde ocjene od strane Odbora:	08.09.2023.

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 08.09.2023.

Ime i prezime studenta:	Tia Ledenčan
Studij:	Programsko inženjerstvo
Mat. br. studenta, godina upisa:	R4528, 27.07.2020.
Turnitin podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Mobilna aplikacija za samostalno učenje o matricama**

izrađen pod vodstvom mentora doc. dr. sc. Anita Katić

i sumentora doc. dr. sc. Bruno Zorić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 1

2. MATRICE 2

2.1. Definicija 2

2.2. Vrste matrica 2

2.3. Operacije s matricama 3

2.3.1. Zbrajanje matrica i množenje matrice skalarom 4

2.3.2. Množenje matrica 4

2.4. Determinanta matrice 5

2.4.1. Laplaceov razvoj 6

2.4.2. Svojstva determinanti 6

2.5. Inverz matrice 6

2.5.1. Elementarne transformacije matrice 7

2.6. Postojeća rješenja za učenje o matricama 8

2.6.1. Linear Algebra Solver 8

2.6.2. Symbolab 9

2.6.3. Matriz Gauss Jordan 10

2.6.4. Matrix operations 11

2.6.5. Matrix Calculator Pro 11

2.6.6. Matematika – Matematičke igre 12

3. PROGRAMSKO RJEŠENJE APLIKACIJE ZA SAMOSTALNO UČENJE O MATRICAMA 14

3.1. Specifikacija zahtjeva 14

3.2. Korištene tehnologije 17

3.3. Arhitektura MVVM 19

3.4. Opis korištenja aplikacije 21

3.5. Rješenja specifičnih problema pri izgradnji aplikacije 23

3.6. Testiranje aplikacije 26

4. ZAKLJUČAK 30

LITERATURA 32

SAŽETAK **34**

ABSTRACT **35**

1. UVOD

Učenje je proces povezivanja pojmova, stjecanje trajno pohranjenih podataka u memoriji, znanja. U današnjem svijetu postoje različiti izvori informacija i mogućnosti učenja. Pametni telefoni omogućili su koncept mobilnog učenja. Mobilno je učenje oblik stjecanja znanja u kojemu su materijali za učenje dostupni preko mobilnih uređaja. Kako bi se postiglo što bolje okruženje za učenje, stvaraju se različite aplikacije. Mobilne bi aplikacije trebale kvalitetno pridonijeti procesu učenja uvodeći elemente igre. Ti elementi privlače učenika na daljnje korištenje i napredak u učenju, bez doživljaja pritiska i obveze, s osjećajem priznanja uspjeha i motivacije za daljnje učenje, potičući pozitivne emocije koje efikasno djeluju pri samostalnom učenju. Zato se u današnjim aplikacijama implementira igrifikacija koristeći različite elemente igara, najčešće nekih oblika nagrađivanja za poticanje daljnjeg učenja. Neki od elemenata igrifikacije mogu se primijeniti u aplikaciji pri učenju pojmova o matricama. Matrice su pravokutne tablice koje se sastoje od redaka i stupaca ispunjenih elementima [1]. Postoje različite vrste, pojmovi i operacije vezane uz matrice s kojima će se korisnik upoznati koristeći aplikaciju te dodatno utvrditi znanje rješavajući kviz u pojedinoj cjelini. Kako bi se pospješilo učenje omogućeno je isprobavanje pročitanoga kroz unos matrica i isprobavanje operacija nad njima. Važno je usvojiti osnovne pojmove i operacije nad matricama jer matrice nisu samo dio linearne algebre već su značajne u raznim područjima računalne znanosti. Neke od primjena matrica mogu se pronaći u računalnoj grafici, pri digitalnoj obradi fotografija, u strojnom učenju, kod obrade signala. Navedeno je ostvareno kroz mobilnu aplikaciju za samostalno učenje o matricama što je zadatak ovog završnog rada.

U drugom su poglavlju opisane vrste matrica, osnovni pojmovi i operacije vezane uz njih te slična programska rješenja koja služe pri učenju nekog matematičkog područja. U trećem je poglavlju detaljno opisana implementacija i korištenje *Android* aplikacije za samostalno učenje o matricama te su predstavljeni rezultati ankete testiranja aplikacije. Zadnje poglavlje sažeto govori o rezultatima i mogućim proširenjima aplikacije.

1.1. Zadatak završnog rada

U radu je potrebno izraditi mobilnu aplikaciju koja će korisniku omogućiti da samostalno usvoji osnovne pojmove i izvršavanje operacija s matricama, te da stečeno znanje provjeri. Treba opisati postupak izrade aplikacije, testirati ju na primjerima i prikazati rezultate.

2. MATRICE

2.1. Definicija

Matrica je pravokutna tablica koja se sastoji od nekoliko redaka i stupaca ispunjenih njezinim elementima [1]. Elementi matrice najčešće su brojevi, a mogu biti i drugi objekti, poput funkcija, vektora, diferencijalnih operatora i matrica. Matrica tipa (m, n) sastoji se od *m* redaka i *n* stupaca.

Za označavanje elementa matrice koriste se indeksi, u kojima prvi broj određuje redak, a drugi stupac u kojemu se on nalazi. Opći element, u *i*-tom retku i *j*-tom stupcu matrice A, označava se a_{ij}. Tako se za matricu na slici 2.1. koja je tipa (m, n), opći element u *m*-tom retku i *n*-tom stupcu označava a_{mn}.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Sl. 2.1. Opći oblik matrice

2.2. Vrste matrica

Definirani su različiti tipovi matrica, najčešće prema vrijednostima elemenata matrica i njihovom obliku. Tako su dvije matrice A i B jednake ako su istog tipa i imaju jednake odgovarajuće elemente, kao što je navedeno u izrazu (2-1). Kvadratna je matrica *n*-tog reda matrica kojoj je broj redaka *m* jednak broju stupaca *n*.

$$A, B \in M_{mn}, \qquad A = B \Leftrightarrow a_{ij} = b_{ij}, \qquad \forall i = 1, \ldots, m; j = 1, \ldots, n \tag{2-1}$$

Glavnu dijagonalu kvadratne matrice čine elementi $a_{11}, a_{22}, \ldots, a_{nn}$. Dijagonalna je matrica kvadratna matrica kojoj su elementi izvan glavne dijagonale jednaki nuli. Slika 2.2. prikazuje različite vrste matrica od kojih je dijagonalna označena s D.

Skalarna je matrica dijagonalna matrica kojoj su svi elementi na glavnoj dijagonali jednaki. Jedinična je matrica na slici 2.2. označena s I. Ona je i skalarna matrica jer su joj elementi na glavnoj dijagonali jednaki jedan.

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix} \quad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} 1 & 4 & 0 \\ 0 & 2 & 4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} 1 & -4 \\ -4 & 2 \end{bmatrix} \quad A = \begin{bmatrix} 0 & 6 & -1 \\ -6 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad 0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Sl. 2.2. Različite vrste matrica

Trokutaste matrice su kvadratne matrice kod kojih se razlikuje gornje i donje trokutaste. Na slici 2.2. prikazana je gornje trokutasta matrica trećeg reda označena kao matrica T. Postupak mijenjanja poretka redaka s poretkom stupaca, to jest u prvi stupac matrice stavlja se prvi redak početne matrice sve dok se ne premjeste svi retci, naziva se transponiranje.

Veliku važnost u teoriji matrica ima simetrična matrica, na slici 2.2. označena S. Za simetričnu matricu vrijedi da je kvadratna te da se pri zrcaljenju s obzirom na glavnu dijagonalu ne mijenja. Simetrična matrica jednaka je njoj transponiranoj matrici, to jest vrijedi izraz (2-2).

$$S^T = S, tj.\ s_{ij} = s_{ji}, \forall i, j \tag{2-2}$$

Suprotna matrica od A označava se $-A$ i dobije se tako da se svakom elementu početne matrice promijeni predznak. Antisimetrična je matrica ako vrijedi $A^T = -A, tj. a_{ij} = -a_{ji}, \forall i, j$. Ona je kvadratna i ima nule na glavnoj dijagonali, na slici 2.2. označena je s A. Nul-matrica je matrica kojoj su svi elementi nule, na slici 2.2. označena 0.

Vektori su jednoredne i jednostupčane matrice odnosno matrice koje se sastoje od jednog retka (vektor-redak) ili jednog stupca (vektor-stupac). Transponiranjem vektor-retka dobije se vektor-stupac. Primjer vektora prikazan je na slici 2.3. vektor-redak označen je s A, a vektor-stupac s B.

$$A = \begin{bmatrix} 1 & 4 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Sl. 2.3. Vektor redak i vektor stupac

2.3. Operacije s matricama

Osim unarne operacije transponiranja, neke od važnijih operacija s matricama su zbrajanje matrica, množenje matrice skalarom te množenje matrica.

2.3.1. Zbrajanje matrica i množenje matrice skalarom

Matrice A i B zbrajaju se tako da im se zbroje elementi na istim mjestima. Slika 2.4. prikazuje zbrajanje matrica drugog reda, rezultat zbrajanja je matrica istog tipa.

$$\begin{bmatrix} 2 & -1 \\ 4 & 2 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 6 & 3 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 10 & 5 \end{bmatrix}$$

Sl. 2.4. Zbrajanje matrica

Za skalar $\lambda \in \mathbb{R}$ i matricu $A \in \mathcal{M}_{mn}$ moguće je definirati množenje predstavljeno formulom (2-3), gdje $a_{i,j}$ predstavlja element matrice A.

$$(\lambda a_{ij}) = \lambda(a_{ij}), \forall i, j \tag{2-3}$$

Tako definiran umnožak matrice skalarom prikazan je na slici 2.5. gdje je matrica pomnožena skalarom devet. Iz prethodno navedenih operacija moguće je uočiti kako operaciju oduzimanja između matrica predstavlja zbroj matrice A i suprotne matrice od B, $A - B = A + (-1) \cdot B$.

$$9\begin{bmatrix} -1 & 0 \\ 6 & 3 \end{bmatrix} = \begin{bmatrix} 9 \cdot 1 & 9 \cdot (-1) \\ 9 \cdot 10 & 9 \cdot 5 \end{bmatrix} = \begin{bmatrix} 9 & -9 \\ 90 & 45 \end{bmatrix}$$

Sl. 2.5. Množenje matrice skalarom

U skupu matrica istog tipa, za zbrajanje i množenje skalarom, vrijede sljedeća svojstva:

1. $A + B = B + A$
2. $(A + B) + C = A + (B + C)$
3. $\exists 0 \in \mathcal{M}_{mn},\ \mathrm{A} + 0 = 0 + \mathrm{A}$
4. $\exists! (-A) \in \mathcal{M}_{mn},\ \mathrm{A} + (-\mathrm{A}) = (-\mathrm{A}) + \mathrm{A} = 0.$
5. $\alpha(\beta A) = (\alpha\beta)A$
6. $\alpha(A + B) = \alpha A + \alpha B$
7. $(\alpha + \beta)A = \alpha A + \beta A$
8. $1 \cdot A = A.$

Zbog toga što vrijede prethodno definirana svojstva, skup matrica uz te dvije operacije naziva se vektorski prostor i označava se kao uređena trojka $(\mathcal{M}_{mn}, +, \cdot)$.

2.3.2. Množenje matrica

Kako bi se mogle pomnožiti dvije matrice potrebno je zadovoljiti određena pravila. Ne može se pomnožiti matrice istog tipa (m, n), za $m \neq n$. Važan je poredak matrica, one moraju biti ulančane. Ulančane su matrice one kojima je broj stupaca prve matrice jednak broju redaka druge

matrice. Za matricu A tipa (m, n), matrica B mora biti tipa (n, p) kako bi mogao postojati umnožak $A \cdot B$. Rezultat množenja bit će matrica C tipa (m, p), čiji su elementi dani formulom (2-4) [2].

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \quad (i = 1, \dots, m; j = 1, \dots, p). \tag{2-4}$$

Matrice se množe tako da se retci prve matrice množe sa stupcima druge. Slika 2.6. prikazuje umnožak ulančanih matrica tipa (2, 2) i (2, 3).

Svojstva matričnog množenja:

- asocijativnost: $(A \cdot B) \cdot C = A \cdot (\mathrm{B} \cdot \mathrm{C})$,
- distributivnost: $(A + B) \cdot C = A \cdot C + \mathrm{B} \cdot \mathrm{C}$,
- umnožak s jediničnom matricom: $A \cdot I_n = A,\ I_m \cdot A = A,\ A \in \mathcal{M}_{mn}$
- transponiranje: $(A \cdot B)^T = B^T \cdot A^T$.

$$\begin{bmatrix} -1 & 0 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 3 & -1 \\ 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} (-1) \cdot 1 + 0 \cdot 0 & -1 \cdot 3 + 0 \cdot 2 & -1 \cdot (-1) + 0 \cdot 2 \\ 2 \cdot 1 + 3 \cdot 0 & 2 \cdot 3 + 3 \cdot 2 & 2 \cdot (-1) + 3 \cdot 2 \end{bmatrix} = \begin{bmatrix} -1 & -3 & 1 \\ 2 & 12 & 4 \end{bmatrix}$$

Sl. 2.6. Množenje matrica

2.4. Determinanta matrice

Kvadratnim je matricama moguće pridružiti skalar zvan determinanta. Determinanta matrice A označava se $detA$ ili $|A|$. Determinanta matrice prvog reda je vrijednost tog elementa. Za matrice drugog reda determinanta je definirana formulom (2-4).

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} a_{22} - a_{21} a_{12} \tag{2-4}$$

Za matrice trećeg reda determinanta se računa nad nekim retkom pri tome pazeći na predznak i zbrajajući umnoške elemenata s determinantama preostalih redaka i stupaca, koje su u slučaju matrice trećeg reda determinante drugog reda. Determinanta matrice koja se dobije brisanjem i-tog retka i j-tog stupca matrice naziva se minora, preciznije, minora elementa a_{ij} i označava $\mathcal{M}_{ij}$. Primjer determinante trećeg reda prikazan je na slici 2.7. Kofaktor (algebarski komplement) elementa a_{ij} matrice A je prikazan formulom (2-5).

$$A_{ij} = (-1)^{i+j} \mathcal{M}_{ij} \tag{2-5}$$

$$\begin{vmatrix} 2 & 1 & 4 \\ 5 & 6 & 7 \\ -1 & 0 & -5 \end{vmatrix} = 2 \cdot \begin{vmatrix} 6 & 7 \\ 0 & -5 \end{vmatrix} - 1 \cdot \begin{vmatrix} 5 & 7 \\ -1 & -5 \end{vmatrix} + 4 \cdot \begin{vmatrix} 5 & 6 \\ -1 & 0 \end{vmatrix} = -18$$

Sl. 2.7. Determinanta matrice 3. reda

2.4.1. Laplaceov razvoj

Laplaceov razvoj determinante predstavlja determinantu matrice n-tog reda, definira se razvojem po nekom retku ili stupcu. Determinanta n-tog reda, za minoru $\mathcal{M}_{ij}$ i kofaktor A_{ij} elementa a_{ij}, razvojem po i-tom retku (2-6) glasi:

$$det\, A = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \mathcal{M}_{ij} \tag{2-6}$$

Laplaceov razvoj po *j*-tom stupcu dan je formulom (2-7).

$$det\, A = \sum_{i=1}^{n} (-1)^{i+j} a_{ij} \mathcal{M}_{ij} \tag{2-7}$$

2.4.2. Svojstva determinanti

Navedena su najvažnija svojstva determinanti.

1. $detA^T = detA$
2. Ako se u determinanti zamijeni dva retka (stupca), determinanta mijenja predznak.
3. Ako se jednom retku (stupcu) determinante doda neki od preostalih redaka (stupaca) pomnožen skalarom, onda se determinanta ne mijenja.
4. Ako se jednom retku (stupcu) determinante doda linearna kombinacija preostalih redaka (stupaca), onda se determinanta ne mijenja.
5. Ako su svi elementi nekog retka (stupca) jednaki nuli, onda je determinanta jednaka nuli.
6. Determinanta se množi skalarom tako što se jedan redak (stupac) pomnoži tim skalarom.
7. Determinanta trokutaste matrice jednaka je produktu elemenata na glavnoj dijagonali.
8. $\det(A \cdot B) = detA \cdot detB$

2.5. Inverz matrice

Kvadratna je matrica *n*-tog reda regularna ako postoji matrica B n-tog reda takva da vrijedi da je $A \cdot B = B \cdot A = I$. Jedinična matrica I n-tog je reda, a B je jedinstvena i naziva se inverz matrice

A te se označava A^{-1}. Svojstvo je regularnih matrica $detA \neq 0$. Inverz matrice moguće je odrediti pomoću matrice kofaktora A^*, formulom (2-8).

$$A^{-1} = \frac{(A^*)^T}{det\,A} \tag{2-8}$$

Slika 2.8. prikazuje primjenu formule (2-8) pri određivanju inverza matrice A. U postupku je vidljivo da se provjerava mogućnost postojanja inverza računanjem determinante. Determinanta iznosi jedan, $1 \neq 0$ stoga zadana matrica ima inverz. Zatim se odrede kofaktori svakog elementa i zapiše ih se na odgovarajuće pozicije u matrici. Transponira se dobivena matrica i podijeli s determinantom.

$$A = \begin{bmatrix} 4 & 7 \\ 1 & 2 \end{bmatrix},\ \det A = 1$$

$$a_{11} = 4,\ A_{11} = (-1)^{1+1} \cdot 2 = 2 \qquad a_{12} = 7,\ A_{12} = (-1)^{1+2} \cdot 1 = -1$$

$$a_{21} = 1,\ A_{21} = (-1)^{2+1} \cdot 7 = -7 \qquad a_{22} = 2,\ A_{11} = (-1)^4 \cdot 4 = 4$$

$$A^* = \begin{bmatrix} 2 & -1 \\ -7 & 4 \end{bmatrix},\ (A^*)^T = \begin{bmatrix} 2 & -7 \\ -1 & 4 \end{bmatrix}$$

$$A^{-1} = \frac{1}{detA} \cdot (A^*)^T = \begin{bmatrix} 2 & -7 \\ -1 & 4 \end{bmatrix}$$

Sl. 2.8. Inverz matrice pomoću kofaktora

Drugi način određivanja inverza matrice je Gauss-Jordanovom metodom. Pri tom postupku matricu se A proširi dopisivanjem jedinične matrice istog reda te je cilj doći do inverzne matrice primjenom elementarnih transformacija.

2.5.1. Elementarne transformacije matrice

Osnovne elementarne transformacije matrica su zamjena dvaju redaka (stupaca), množenje retka (stupca) brojem različitim od nule i dodavanje jednom retku (stupcu) linearnu kombinaciju preostalih redaka (stupaca). Navedene transformacije koriste se pri rješavanju linearnih sustava, računanju determinanti, određivanju ranga matrice te pri pronalaženju inverza matrice [1]. Dvije matrice A i B su ekvivalentne ako je jednu iz druge moguće dobiti nakon konačno mnogo elementarnih transformacija.

Rang matrice A jednak je maksimalnom broju linearno nezavisnih redaka matrice A odnosno maksimalnom broju linearno nezavisnih stupaca matrice A. Rang matrice A određuje se pronalaskom ekvivalentne matrice B u reduciranom obliku prikazanom na slici 2.9. Broj k koji je jednak redu jedinične matrice u reduciranom obliku matrice A jednak je rangu matrice A, $r_A = k$. Ako se radi o matrici n-tog reda, matrica će biti regularna ako je rang jednak n.

$$B = \begin{bmatrix} I_k & 0 \\ 0 & 0 \end{bmatrix}$$

Sl. 2.9. Ekvivalentna matrica matrice A u reduciranom obliku

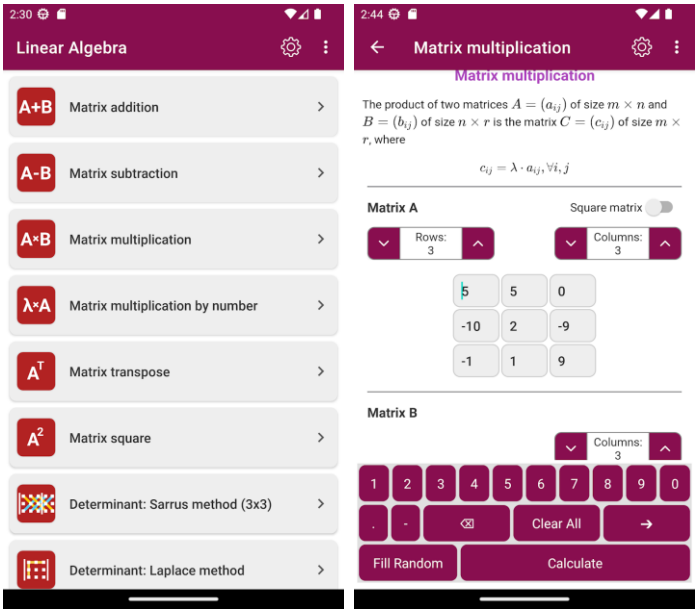
2.6. Postojeća rješenja za učenje o matricama

Postoje različite mobilne aplikacije za učenje o matricama koje u većini implementacija služe kao matrični kalkulatori. Time predstavljaju pomoćne alate pri samostalnom učenju jer korisnik može tijekom vježbanja zadataka provjeriti vlastito rješenje ili dobiti uvid u postupke provođenja različitih operacija nad matricama. Pojedine aplikacije imaju objašnjenja teorijskoga dijela operacija i prikaz rješenja. Sve navedene mobilne aplikacije dostupne su u *Google Play* trgovini.

2.6.1. Linear Algebra Solver

Aplikacija *Linear Algebra Solver* , prikazana na slici 2.8., omogućuje unos matrica do petog reda, omogućene su osnovne operacije s matricama, transpozicija, računanje determinte Sarrusovom i Laplaceovom metodom. Sarrusovo pravilo za računanje determinanti definirano je za kvadratne matrice trećeg reda. Obzirom da je *Linear Algebra Solver* aplikacija namijenjena za studente i inženjere koji koriste matrične izračune [3], cijelo područje linearne algebre uključeno je u sadržaj aplikacije tako su dostupne razne vektorske operacije, određivanje kuta između vektora,

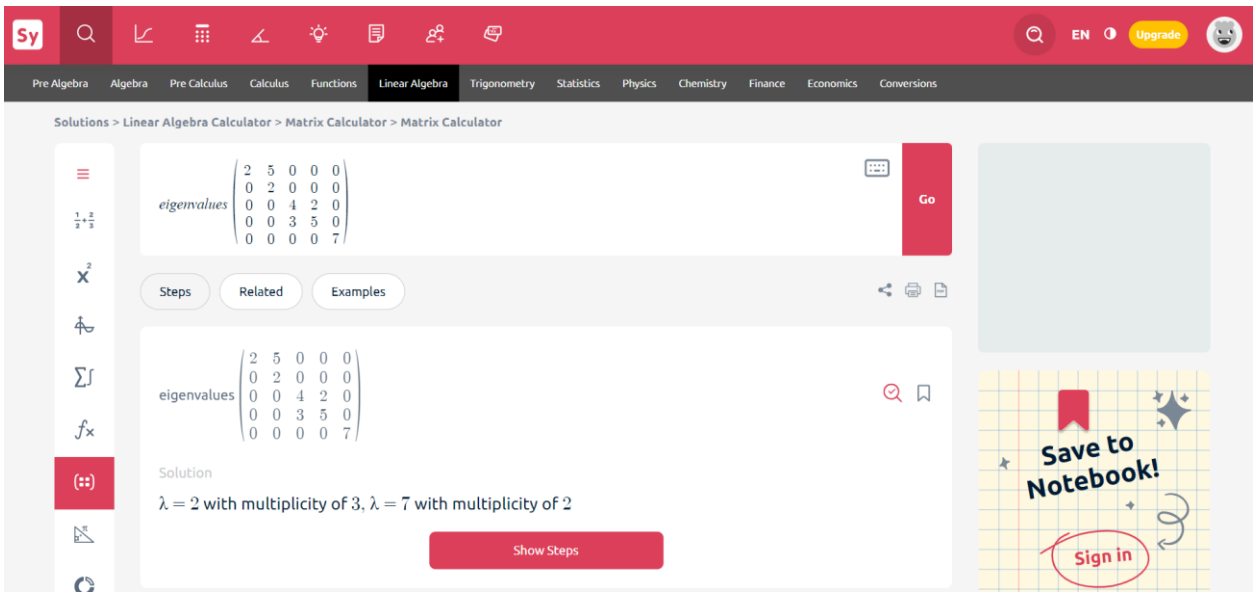
projekcije, volumeni i drugo. Aplikacija za razliku od ostalih aplikacija sličnih funkcionalnosti, podržava osam jezika.



Sl. 2.8. Zaslon sadržaja i zaslon za množenje matrica aplikacije *Linear Algebra Solver*, preuzeto iz [3]

2.6.2. Symbolab

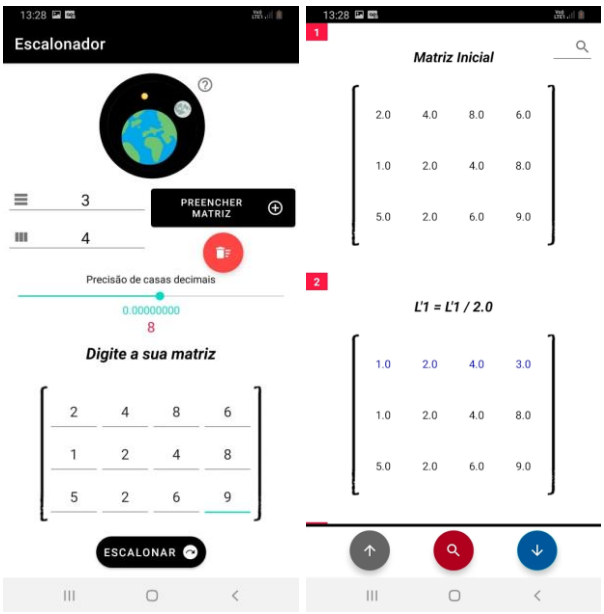
Symbolab nudi aplikacije za učenje i vježbanje različitih područja matematike, s najvećim opsegom gradiva od svih promatranih aplikacija i tristotinjak kalkulatora. Implementirani su matematički kalkulatori, kalkulatori za geometriju, financije, fiziku, kemiju, računanje vremena, te su omogućene razne pretvorbe. Uz to, stvoreni su grafovi za crtanje i prikaz funkcija kao i sučelje za učenje geometrije i crtanje geometrijskih likova. Stvoren je kviz i omogućeno je uvježbavanje algebre, matrica i vektora, funkcija, trigonometrije, osnovnih operacija i zadataka riječima. Kako bi se omogućilo korištenje svih značajki, od praćenja napretka, korištenja pomoći umjetne inteligencije do pregleda cijelog postupka rješavanja zadataka, potreban je *pro plan* koji se naplaćuje. Besplatna verzija omogućuje spremanje bilješki, vježbanje zadataka i upravljanje profilom. Sučelje je jednostavno za korištenje i vizualno ugodnog rasporeda elemenata. Na slici 2.9. prikazan je zaslon web sučelja *Symbolab* matričnog kalkulatora.



Sl. 2.9. Zaslon matričnog kalkulatora *Symbolab*

2.6.3. Matriz Gauss Jordan

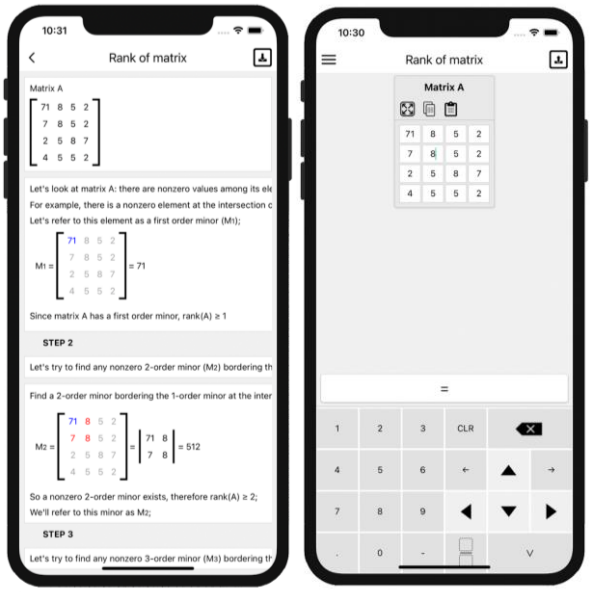
Matriz Gauss Jordan je mobilna aplikacija, prikazana na slici 2.10., neobičnog sučelja, rješava sustave linearnih jednadžbi, pri tom određuje je li sustav moguć, te računa inverzne matrice Gauss-Jordanovom metodom. Prema [4], omogućena je visoka preciznost operacija s 15 decimalnih mjesta i rad s nizovima do 2550 brojeva. Aplikacija je dostupna na pet jezika.



Sl. 2.10. Dva zaslona aplikacije *Matriz Gauss Jordan*, preuzeto iz [4]

2.6.4. Matrix operations

Aplikacija *Matrix operations*, prikazana na slici 2.11., omogućuje rješavanje sustava linearnih jednadžbi, računanje determinante matrice, pronalazak inverzne matrice, određivanje ranga matrice, svojstvenih vrijednosti i vektora, kofaktora, adjungirane matrice te rad s osnovnim operacijama kao što su zbrajanje, množenje, oduzimanje i transponiranje. Za razliku od drugih aplikacija, ova aplikacija ima najviše implementiranih metoda za rješavanje pojedine operacije. Većina operacija ima različite načine rješavanja koje je moguće upotrijebiti, na primjer za determinantu matrice moguće je odabrati rješavanje po prvom retku, Sarrusovim pravilom, svođenjem na trokutasti oblik. Omogućen je unos različitih veličina matrica te se detaljno prikazuje opis rješenja za odabranu operaciju. Sučelje je, za razliku od prethodne aplikacije, jednostavnije i intuitivnije.

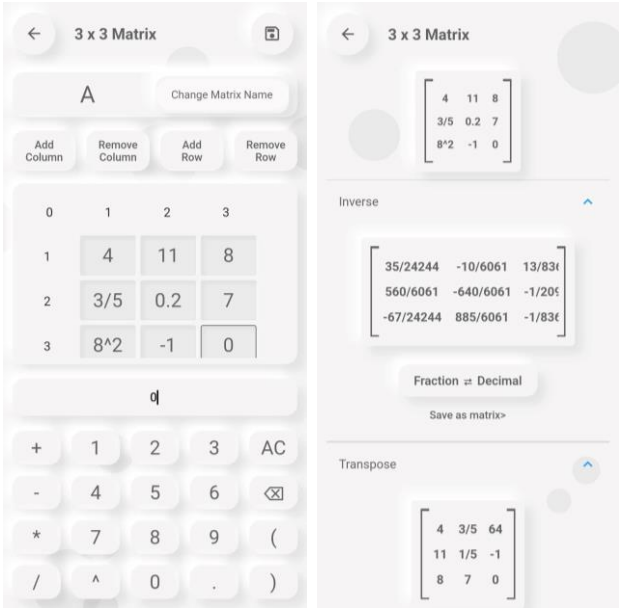


Sl. 2.11. Zasloni za računanje ranga u aplikaciji *Matrix operations*, preuzeto iz [15]

2.6.5. Matrix Calculator Pro

Zasloni aplikacije *Matrix Calculator Pro*, prikazani na slici 2.12., prikazuju mogućnost unosa matrica različitih dimenzija i zaslon nakon odabira spremljene matrice A. Zaslon za unos matrice ne omogućuje odabir retka ili stupca matrice koji bi se obrisao već briše zadnji redak ili stupac, uz to retci su zamijenjeni stupcima. Matricu se može spremiti za upotrebu u matričnim jednadžbama za koje se navodi mogućnost upotrebe determinante matrice. U tom se slučaju determinanta matrice koristi kao skalar za množenje matrice. Nije omogućeno određivanje determinante matrice preko unosa za matričnu jednadžbu. Za prikaz determinante, inverza,

transponirane i kofaktor matrice potrebno je odabrati željenu matricu te su tada vidljivi pojedini podaci za tu matricu kao što je prikazano na slici 2.12. Aplikacija ima manje implementiranih operacija, animiranu pozadinu koja s vremenom odvraća pozornost od ostatka aplikacije.



Sl. 2.12. Zasloni aplikacije *Matrix Calculator Pro*

2.6.6. Matematika – Matematičke igre

Aplikacija koja nije vezana uz matrice već predstavlja zanimljivu, edukativnu aplikaciju za učenje matematike je *Matematika-Matematičke igre* koja na zabavan način omogućuje usvajanje dijela gradiva osnovne i srednje škole. Sučelje aplikacije prilagođeno je i jednostavno za korisnike. Ostvareno je praćenje napretka kroz rješavanje kvizova i prikupljanje bodova. Omogućeno je učenje teorijskog dijela gradiva i provjeru istog uz pomoć kviza. Slika 2.13. prikazuje glavni zaslon aplikacije *Matematika-Matematičke igre* s cjelinama, napretkom korisnika i prikupljenim bodovima.

Sl. 2.13. Glavni zaslon aplikacije *Matematika-Matematičke igre*

3. PROGRAMSKO RJEŠENJE APLIKACIJE ZA SAMOSTALNO UČENJE O MATRICAMA

Svega 67% roditelja smatra da aplikacije mogu pomoći pri učenju matematike [5]. *Android* aplikacija za samostalno učenje o matricama, *Matrix Learner*, osmišljena je kako bi se korisniku omogućilo učenje dijela gradiva Linearne algebre vezanog uz matrice s mogućnošću isprobavanja i provjere usvojenosti pročitanoga gradiva. U ovom su poglavlju navedeni zahtjevi aplikacije, opisan tijek razvoja aplikacije, opisane su korištene tehnologije te je predstavljena struktura i rezultati testiranja aplikacije.

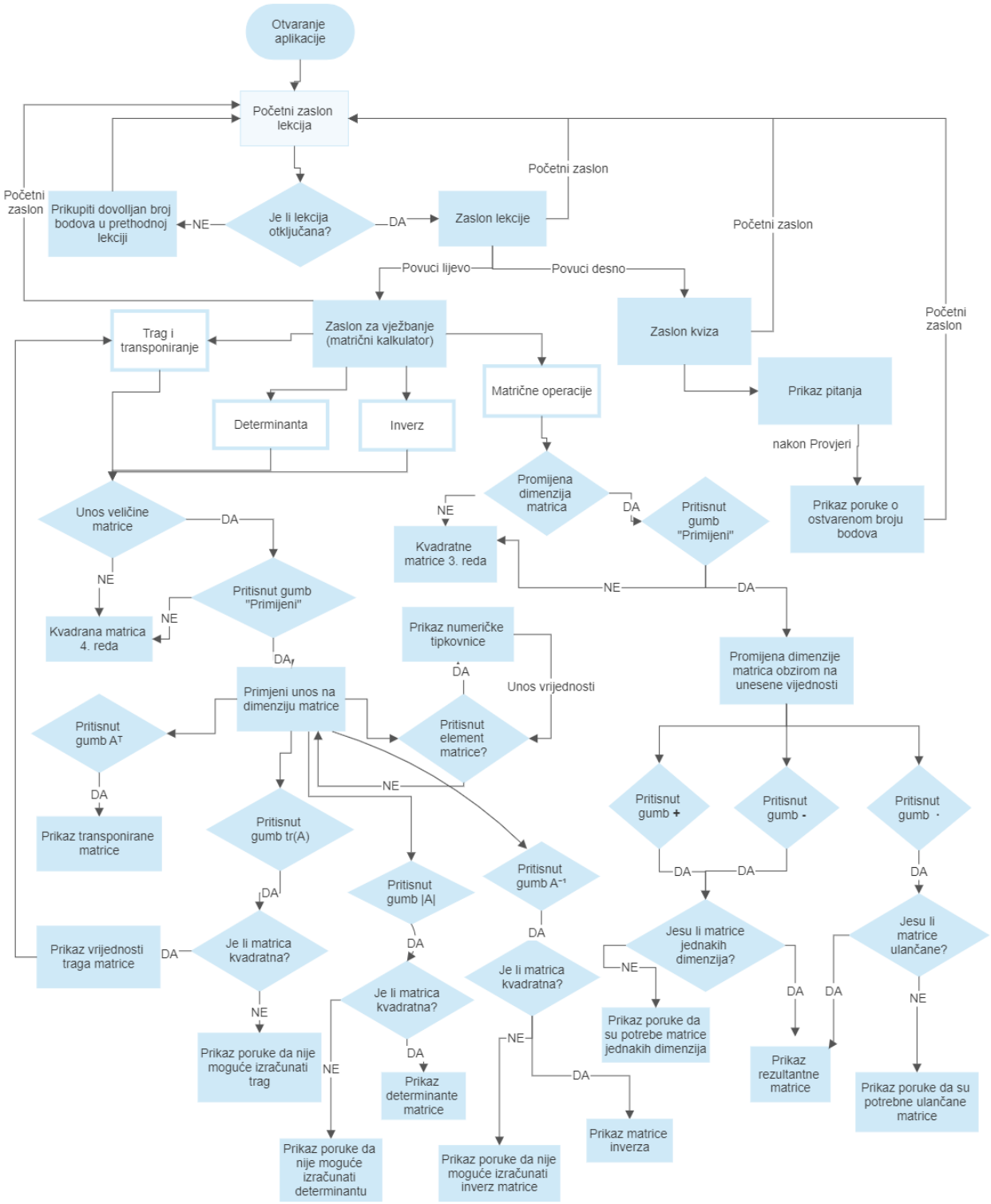
3.1. Specifikacija zahtjeva

Kako bi se omogućilo samostalno učenje o matricama, potrebna je podjela gradiva na lekcije iz kojih učenik može naučiti gradivo. Gradivo je podijeljeno na sedam lekcija kako bi korisnik uspješno usvojio određeni dio gradiva bez opterećivanja s puno novih pojmova. Za provjeru usvojenosti gradiva izrađen je kviz koji nasumično zadaje pitanja, iz baze podataka, vezana uz gradivo. Postoje tri vrste pitanja: točno/netočno, izbor jednog točnog odgovora i nadopunjavanje. Kako bi se motiviralo korisnika, ostvareno je praćenje osvojenih bodova prilikom rješavanja kvizova. Bodovi su prikazani na početnom zaslonu i pomažu pri uvidu u vježbanje određenih lekcija prikazom ostvarenih zvjezdica napretka za pojedinu lekciju. Još jedan implementirani dio koji omogućuje lakše usvajanje gradiva je matrični kalkulator. Izrađen je za isprobavanje i bolje razumijevanje naučenoga gradiva. Na tom je zaslonu korisniku omogućen unos dimenzija matrica i elemenata matrice. U svakoj je lekciji omogućeno isprobavanje gradiva koje je korisnik, unutar te lekcije, pročitao. Sveukupno je ostvareno računanje traga, determinante, inverza matrice, transponiranje matrice, zbrajanje, oduzimanje, množenje matrica te množenje matrice i skalara. Ako nisu zadovoljeni uvjeti izračuna neke operacije, korisniku se prikaže informativna poruka koja ukazuje na nemogućnost izračuna zbog određenog uvjeta pri izračunu, definiranog svojstvima te operacije. Zbog prethodno navedenih mogućnosti, matrični kalkulator može poslužiti pri vježbanju zadataka s matricama. Tablica 3.1. prikazuje zahtjeve aplikacije *Matrix Learner*.

Tab. 3.1. Zahtjevi aplikacije

R. br.	Zahtjev	Opis
1.	Pregled lekcija	Korisniku je pri otvaranju aplikacije prikazan popis lekcija.
2.	Prikaz bodova	Korisnik na početnom zaslonu vidi ukupan broj bodova.
3.	Pregled napretka	Svaka lekcija na početnom zaslonu ima prikaz ostvarenih zvjezdica.
4.	Ograničen pristup lekcijama	Korisnik ne može pristupiti zaključanoj lekciji dok ne prikupi određen broj bodova.
5.	Pregled definicija lekcije	Korisniku je omogućeno učenje dijela gradiva definirano naslovom lekcije.
6.	Promjena dimenzije matrice	Korisnik može promijeniti broj redaka i stupaca matrice na zaslonu za isprobavanje.
7.	Unos elemenata matrice	Na zaslonu za isprobavanje omogućen je unos elemenata matrice.
8.	Izračun traga matrice	Ako su zadovoljeni uvjeti, klikom na gumb tr(A), prikaže se trag matrice.
9.	Transponiranje matrice	Klikom na gumb, prikaže se transponirana matrica unesene matrice.
10.	Operacije s matricama	Ovisno o dimenzijama matrica, klikom na gumb +, − ili ·, prikaže se rezultantna matrica ili informativna poruka.
11.	Izračun determinante matrice	Klikom gumba |A| prikaže se ili determinanta matrice ili informativna poruka ako je matrica nekvadratna.
12.	Izračun inverza matrice	Ako je matrica kvadratna, klikom na gumb A^{-1}, prikaže se matrica inverza, u suprotnom informativna poruka.
13.	Provjera usvojenosti znanja	Za svaku je lekciju izrađen kviz preko kojeg korisnik dobije bodove za točne odgovore.

Na slici 3.1. prikazan je dijagram toka koji prikazuje moguće kretanje korisnika kroz aplikaciju. Otvaranjem aplikacije dolazi se na početni zaslon s lekcijama u kojoj svaka lekcija ima zaslon s dijelom gradiva o matricama, zaslon za isprobavanje predstavljen kao matrični kalkulator i zaslon za kviz. Zaslon s dijelom gradiva sadrži definicije koje su vizualno odvojene karticama u svijetlim bojama koje stvaraju smiren ugođaj za lakše učenje. Zaslon za isprobavanje prilagođen je određenoj lekciji s prikazom matrice i mogućnošću isprobavanja operacija iz te lekcije. Zaslon kviza prikazuje određena pitanja vezana uz lekciju i omogućuje predaju odgovora za prikupljanje bodova koji su namijenjeni za otvaranje sljedeće lekcije.



Sl. 3.1. Dijagram toka aplikacije

3.2. Korištene tehnologije

Matrix Learner je aplikacija izrađena koristeći programski jezik *Kotlin* te *Jetpack Compose*. *Jetpack Compose* je *Androidov* preporučeni alat za izradu korisničkog sučelja. *Jetpack Compose* pojednostavljuje i ubrzava razvoj korisničkog sučelja na *Androidu* s manje koda, snažnim alatima i intuitivnim *Kotlin* aplikacijskim programskim sučeljima (engl. *application programming interface*, API) [6]. Postignut je brži razvoj programske podrške i lakše održavanje jer nije potrebno pisati izgled u XML-u (engl. *Extensible Markup Language*) već se koriste *composable* funkcije. Kako bi funkcija bila *composable* funkcija, potrebno ju je označiti anotacijom *@Composable*. *Material Design* je jezik za dizajn koji je razvio *Google* kako bi pomogao dizajnerima i krajnjim korisnicima da repliciraju *Googleov* rad i objasnio zašto stvari u *Googleu* izgledaju i reagiraju onako kako izgledaju [7]. Najvažniji elementi *Material Designa* su boja, tipografija i oblik. Koristeći *Material Design Jetpack Composeu* je omogućena automatska prilagodba izgleda aplikacije na tamni način što je ostvareno u aplikaciji *Matrix Learner*.

Android Studio je korišteno integrirano razvojno okruženje za izradu *Android* aplikacije. *Koin* je okvir za ubrizgavanje ovisnosti za programski jezik *Kotlin*. Olakšava testiranje i održavanje. Ubrizgavanje ovisnosti služi za pisanje čistog, održivog koda, smanjuje spregnutost klasa i time pomaže u poštivanju SOLID načela održavanja jedne odgovornosti (engl. *Single responsibility principle*, SRP) i inverzije ovisnosti (engl. *Dependency inversion principle*, DIP). *Koin* ubrizgava ovisnosti pomoću modula. Modul je logički blok, koji se tvori koristeći funkciju *module,* u kojem se pišu definicije klasa za ubrizgavanje. Komponente ne moraju nužno biti u istom modulu. Modul pomaže u organizaciji definicija i može ovisiti o definicijama iz drugog modula [8]. Definicije unutar modula mogu biti *singletoni* i tvornice . Ako je riječ o *singletonu*, kada se zahtjeva njegova ovisnost, *Koin* će vratiti jednaku instancu klase, a za tvornice će se stvarati nove instance pri svakom pozivu. Na slici 3.2. prikazano je navođenje svih modula korištenih u aplikaciji, pridijeljen je kontekst u *startKoin* bloku klase aplikacije. Definicije kao što su *viewModeli*, baze, repozitoriji, koje se želi ubrizgati stavljene su u različite module koji su podijeljeni prema značajkama aplikacije.

```
class MatrixApp: Application() {
    override fun onCreate() {
        super.onCreate()
        // Start Koin
        startKoin { this: KoinApplication
            // Feed in Context
            androidContext(this@MatrixApp)
            // Load dependencies module
            modules(appModule, subtopicModule, topicModule, quizModule, pointsModule, subjectModule)
        }
    }
}
```

Sl. 3.2. Pokretanje *Koina*

Room je korištena biblioteka za rad sa *SQLite* bazom podataka. Tri glavne komponente *Rooma* su klasa baze podataka, entiteti podataka koji predstavljaju tablice u bazi podataka i objekti za pristup podacima (DAO) koji pružaju metode koje aplikacija može koristiti za postavljanje upita, ažuriranje, umetanje i brisanje podataka u bazi podataka [9]. Slika 3.3. prikazuje *appModule* u kojemu su definirani *MainViewModel* koji služi za praćenje koji zaslon se treba prikazati klikom na traku kartica. U ovom je modulu definiran i *single* blok za pružanje baze podataka ostalim

```
val converterInstance = Converters()

fun provideSubtopicDatabase(app: MatrixApp):SubtopicDatabase{
    return Room.databaseBuilder(
        app,
        SubtopicDatabase::class.java,
        SubtopicDatabase.DATABASE_NAME
    ) .createFromAsset( databaseFilePath: "database/subtopics_db")
        .addMigrations(
            SubtopicDatabase.migration1To2, SubtopicDatabase.migration2To3,
            SubtopicDatabase.migration3To4, SubtopicDatabase.migration4To5,
            SubtopicDatabase.migration6To7, SubtopicDatabase.migration7To8
        )
        .addTypeConverter(converterInstance)
        .build()
}

val appModule = module { this: Module

    single{ this: Scope  it: ParametersHolder
        provideSubtopicDatabase(app = androidApplication() as MatrixApp)
    }

    viewModel { this: Scope  it: ParametersHolder
        MainViewModel()
    }
}
```

Sl. 3.3. *AppModule*

dijelovima aplikacije. Za instanciranje baze podataka koristi se *databaseBuilder* metoda kojoj se predaju kontekst, klasa definirane baze i ime. Baza podataka se pri prvom pokretanju aplikacije popuni podacima iz predanog dokumenta popunjene baze podataka kroz upotrebu metode *createFromAsset*. Bazi je podataka dodijeljen i pretvarač tipa o kojemu će biti riječi u idućem potpoglavlju.

3.3. Arhitektura MVVM

Model-View-ViewModel (MVVM) je arhitekturni obrazac koji predlaže odvajanje prezentacijskog dijela od poslovne logike aplikacije [10]. Odvajanjem logike od prezentacijskog dijela aplikacije omogućeno je lakše rješavanje razvojnih problema i aplikaciju čini lakšom za testiranje, održavanje i razvoj. MVVM se sastoji od tri dijela: modela, pogleda (engl. *View*) i *ViewModela*.

Pogled predstavlja sve što korisnik vidi na zaslonu. On je povezan s *ViewModelom* koristeći javno stanje *ViewModela*. Kada se promijene podaci, unutar stanja, svi pogledi koji koriste taj *ViewModel* bit će obaviješteni o toj promjeni. Kada korisnik, na početnom zaslonu, klikne jednu lekciju preusmjerit će ga se na zaslon *TopicScreen*, poslat će se zahtjev *ViewModelu* za prikaz definicija te lekcije. Metoda *getSubtopicsByTopicNumber*(), unutar *ViewModela*, sprema dobiveni rezultat u stanje. Stanje je tipa *SubtopicsState* klase u kojoj se nalazi lista objekata *Subtopic*. Korištenjem *SubtopicsState* klase za predstavljanje podataka omogućena su kasnija proširenja na zaslonu s definicijama i prati se obrazac upravljanja stanjem koji je u skladu s načelima *Jetpack Composea*. Zapravo, *ViewModel* je posrednik između modela i pogleda, ažurira model obrađujući korisnički unos. Slika 3.4. prikazuje klasu *TopicViewModela* aplikacije koji služi za dohvaćanje odgovarajućih cjelina za lekcije i predstavlja *ViewModel* MVVM arhitekture. Svaka je lekcija određena *topicNumber* brojem na temelju kojeg se dohvaćaju odgovarajući podaci za prikaz.

Subtopic je entitetska klasa koja predstavlja objekte u lokalnoj bazi podataka kojom upravlja *Room*. *LectureDao* skriva složenost izvršavanja operacija nad bazom podataka nudeći apstrakciju sučeljem. *LectureDao* ima sve potrebne metode za dohvaćanje, promjenu, unos i brisanje *Subtopica*. *Flow* predstavlja generički tok podataka kojim se prati životni ciklus podataka i njihove promjene. *Flow* emitira podatke samo kada su oni potrebni. Podaci se dohvaćaju kao povratni tip *Flow<List<Subtopic>>* i koristi se kako bi se dobila notifikacija kada dođe do promjena podataka u bazi. Olakšan je razvoj aplikacija jer *Room* prati sve promijene u bazi. *ViewModel* pruža podatke *Viewu* dohvaćajući ih iz repozitorija. Repozitorij je *jedini izvor istine* (engl. *single source of truth*) za *ViewModel* jer je repozitorij zadužen za dohvaćanje podataka koji mogu biti u lokalnoj bazi podataka ili dohvaćeni sa servera. Stoga, model predstavlja svu poslovnu logiku, u ovom slučaju

čine ju *TopicUseCase*, *Subtopic*, *LectureDao*, *SubtopicRepository* i njegova implementacija. Iz toga je vidljivo kako je model sloj odgovoran za apstrakciju izvora podataka. Uvođenjem *useCase* klasa, kod postaje upotrebljiviji jer svaki *ViewModel* može koristiti one *useCaseve* koji su mu potrebni za odrađivanje posla, jer *ViewModel* uzima rezultate *useCaseva* i sprema ih u stanje koje koriste pogledi. *UseCase* ne zna odakle mu podaci, je li riječ o podacima iz baze ili su podaci dobiveni preko API-a. *TopicScreen* sve dohvaćene podatke iz stanja *ViewModela* prikazuje korisniku koristeći *composable LazyColumn* s *TopicItemima* što prestavlja pogled u MVVM obrascu. Slika 3.5. prikazuje odnos svih navedenih klasa. Po sličnom načelu ostvarene su funkcionalnosti praćenja bodova i prikaza pitanja u kvizovima.

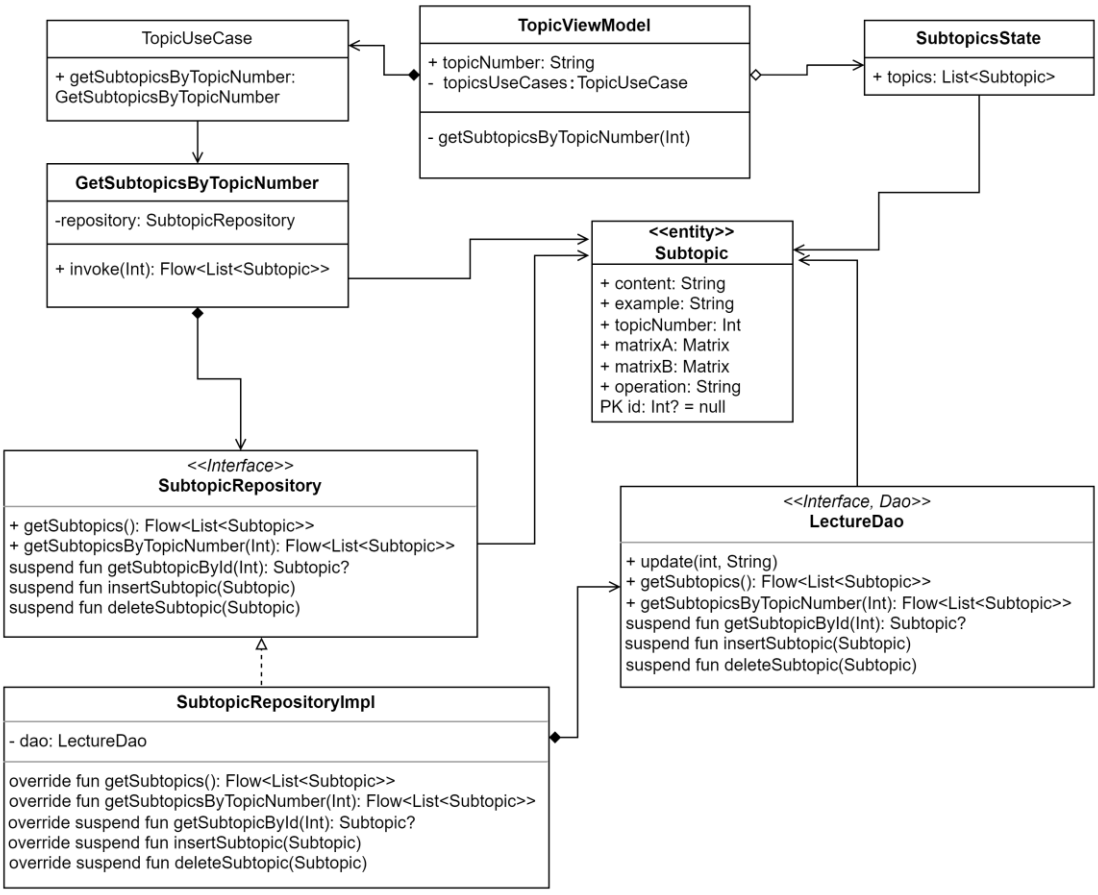
```
class TopicViewModel(
    val topicNumber: String,
    private val topicsUseCases: TopicUseCase
): ViewModel() {

    private val _state = mutableStateOf(SubtopicsState())
    val state: State<SubtopicsState> = _state

    init {
        getSubtopicsByTopicNumber(topicNumber = topicNumber.toInt())
    }


    private fun getSubtopicsByTopicNumber(topicNumber: Int){
        topicsUseCases.getSubtopicsByTopicNumber(topicNumber).onEach { subtopics ->
            _state.value = state.value.copy(
                topics = subtopics.toList()
            )
        }
            .launchIn(viewModelScope)
    }
}
```

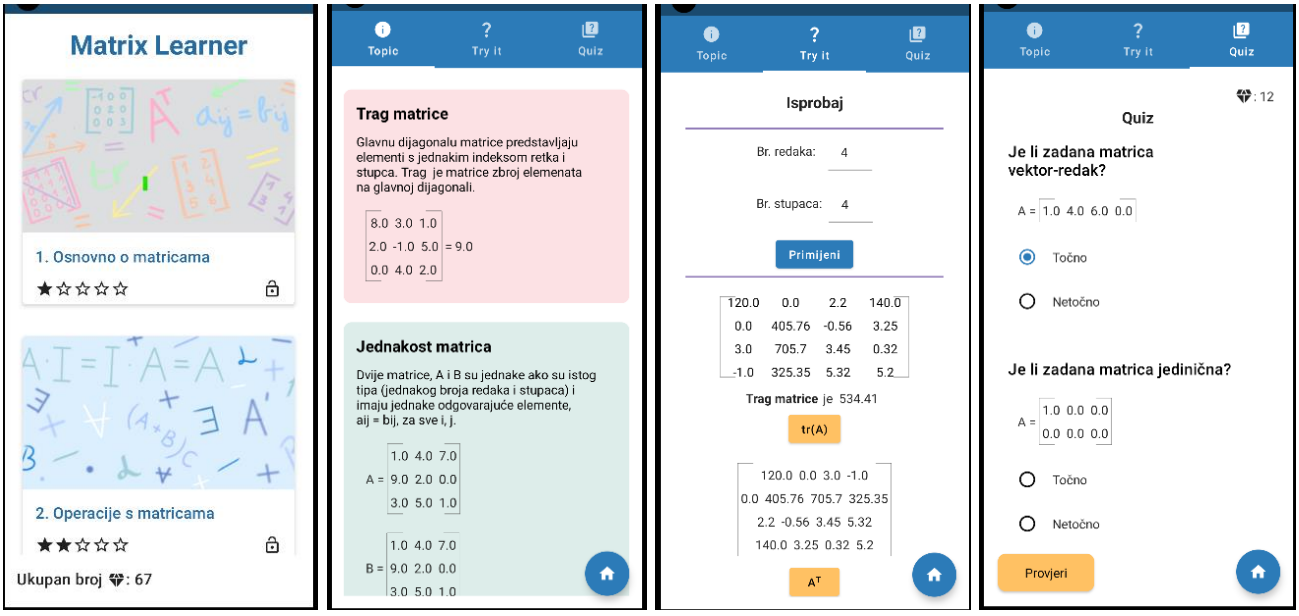
Sl. 3.4. Klasa *TopicViewModel*



Sl. 3.5. Dijagram klasa vezanih uz *TopicViewModel*

3.4. Opis korištenja aplikacije

Ulaskom u aplikaciju prikaže se početni zaslon s izlistanim lekcijama prikazan na slici 3.6. pod a). U svakoj je lekciji dio gradiva iz kojeg korisnik može naučiti osnovne pojmove o matricama. Na početnom su zaslonu lekcije vizualno odvojene kao kartice sa slikom i nazivom lekcije. Ispod naziva prikazane su zvjezdice koje označavaju korisnikov napredak, koliko je korisnik prikupio bodova rješavajući kviz. Korisnik ne može proizvoljno odabrati koju lekciju želi rješavati dok ne prikupi dovoljno bodova iz prethodne lekcije. Navedeno je moguće uočiti na slici 3.7. pod a), gdje je četvrta lekcija zaključana jer u trećoj lekciji nije prikupljeno minimalno sedam bodova za otvaranje četvrte lekcije. Broj bodova, koji je potrebno ostvariti, naznačen je pored ikone zaključanog lokota. Kada otključa sve lekcije, korisnik može proizvoljno ulaziti u sve lekcije.



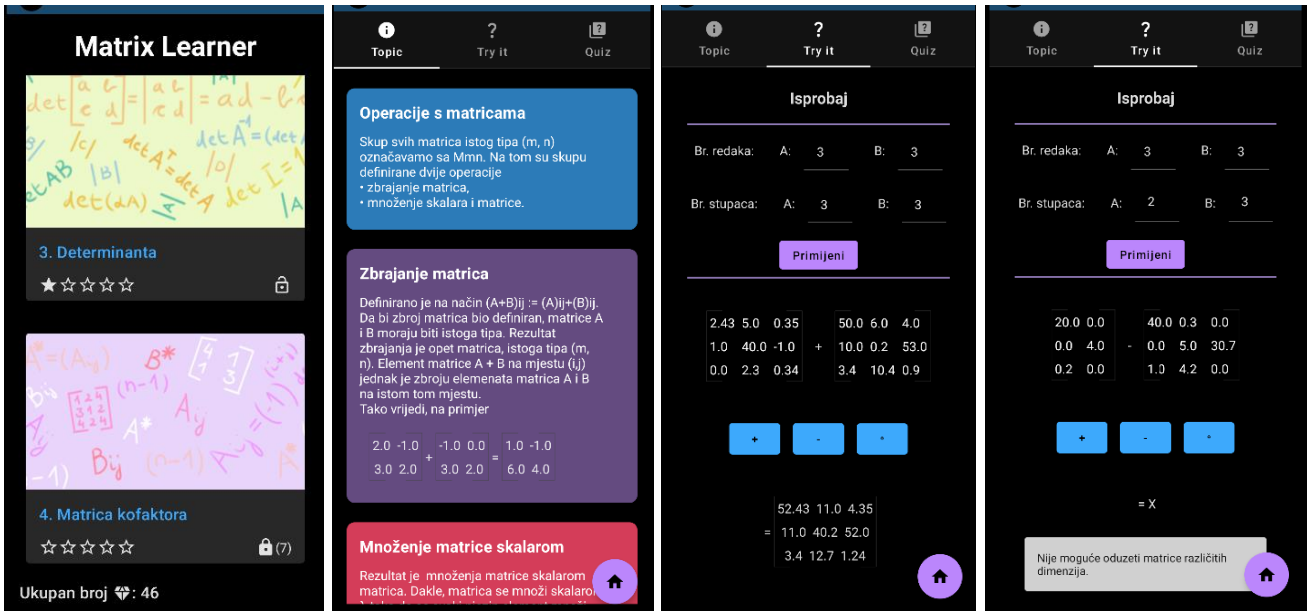
a) b) c) d)

Sl. 3.6. Zasloni aplikacije *Matrix Learner*

Ako je lekcija otvorena, pritiskom na sliku ili naslov lekcije, korisniku se prikažu definicije vizualno odvojene bojom, na slici 3.6. zaslon b) prikazuje neke od definicija prve lekcije koja upoznaje korisnika s osnovnim vrstama matrica. Slika 3.7. b) prikazuje definicije, u tamnom načinu, iz druge lekcije koja govori o operacijama matrica. Pritiskom kartice „*Try it*“ prikaže se zaslon za isprobavanje određenih dijelova gradiva, implementirano kao matrični kalkulator. Na slici 3.6. pod c), prikazan je zaslon za isprobavanje, u prvoj lekciji, gdje se može za matrice do četvrtog reda odrediti trag i transponirana matrica. Kod druge lekcije omogućeno je isprobavanje operacije zbrajanja, oduzimanja i množenja matrica, vidljivo na slici 3.7. pod c). Za ostale je lekcije omogućeno izračunavanje determinante i inverza matrice. Izračunavanje determinante i inverza moguće je samo za kvadratne matrice. Ako korisnik pokuša dobiti rezultat za nekvadratne matrice, prikazat će se poruka o nemogućnosti izračunavanja istog za nekvadratne matrice. Informativna se poruka prikaže, u svim lekcijama, na korisničkom zaslonu ako nije zadovoljen uvjet izračuna određen pravilima linearne algebre. Informativna poruka, koja se prikaže korisniku kada nisu jednake dimenzije matrica, kojima se želi oduzeti elemente, vidljiva je na slici 3.7. pod d).

Svaka lekcija, zbog potrebe provjere stečenog znanja, ima kviz s nasumično odabranim pitanjima iz baze. Zaslon kviza prikazan je na slici 3.6. d). Postoje tri vrste pitanja: točno-netočno, odabir jednog točnog odgovora i nadopunjavanje. Nakon što korisnik odgovori na pitanja, pritiskom na gumb provjeri, prikaže se poruka o prikupljenom broju bodova te je omogućen pregled predanih odgovora. Nema negativnih bodova za netočne odgovore. Bodovi su prikazani kao dijamanti koji

omogućuju otključavanje lekcija i prikaz zvjezdica na početnom zaslonu. Broj osvojenih zvjezdica, za pojedinu lekciju, raste s više prikupljenih bodova u toj lekciji. Na sva je tri zaslona omogućen odlazak na početni zaslon klikom na okrugli gumb s kućicom koji je postavljen za lakšu navigaciju korisnika.



a) b) c) d)

Sl. 3.7. Zasloni aplikacije *Matrix Learner* u tamnom načinu rada

3.5. Rješenja specifičnih problema pri izgradnji aplikacije

U procesu stvaranja aplikacije nailazi se na nepoznanice kako nešto ostvariti. U nastavku su navedeni neki od razriješenih zanimljivih problema. Kako bi se omogućio rad s matricama napravljena je klasa *Matrix* s metodama za postavljanje i dohvaćanje elemenata matrice kao i metode za operacije nad matricama korištene za zaslon isprobavanja. Napravljene su i *composable* funkcije za prikaz i upis elementa matrice. Zbog potrebe spremanja matrica u bazu podataka, radi kasnije upotrebe u lekcijama i kvizu, potrebno je imati na umu da nije moguće pohraniti korisnički definirane tipove podataka bez stvaranja pretvarača. Pretvarači tipa su metode koje govore *Roomu*, biblioteci za rad sa *SQLite* bazom podataka, kako pretvoriti korisnički definirane tipove podataka u i iz ugrađenih tipova podataka koje *Room* može pohraniti [11]. Slika 3.8. prikazuje pretvarače tipa koje *Room* identificira zbog oznake *@TypeConverter*. Prvi je pretvarač iz ugrađenog tipa *String* u korisnički tip *Matrix* koji se koristi pri dohvaćanju podataka iz baze, a drugi je iz tipa *Matrix* u *String* korišten pri spremanju u bazu podataka.

```
@ProvidedTypeConverter
class Converters {
    @TypeConverter
    fun toMatrix(value: String): Matrix? {
        if (value.isEmpty()) {
            return null
        }
        val rows = value.split( ...delimiters: ";" ,":")
        val matrix = Matrix(rows.size, rows[0].split( ...delimiters: ",").size)
        for (i in rows.indices) {
            val cols = rows[i].split( ...delimiters: ",")
            for (j in cols.indices) {
                if (cols[j].isNotEmpty()) {
                    matrix[i, j] = cols[j].toDouble()
                }
            }
        }
        return matrix
    }

    @TypeConverter
    fun fromMatrix(matrix: Matrix): String? {
        val rows = Array(matrix.rows) { i ->
            matrix.data[i].joinToString( separator: ",")

        }
        return rows.joinToString ( separator: ";")
    }
}
```

Sl. 3.8. Pretvarači tipa

Za pravilno dohvaćanje slika i tekstova koji se koriste na početnom zaslonu aplikacije nije dovoljno upisati i spremiti podatke u *drawable* i *string* resurse jer se spremljene cjelobrojne vrijednosti koje predstavljaju dani resurs mogu promijeniti kada se dodaju novi resursi i prevede aplikacija. Iz tog su razloga stvorene *enum* klase u kojima su resursima pridružene jedinstvene oznake i broj koji je spremljen u bazu podataka. Slika 3.9. prikazuje *enum* klasu za mapiranje slika sa spremljenom vrijednosti. Metoda *asRes(stored: Int)* služi za dohvaćanje odgovarajuće slike, poziv te metode prikazan je na slici 3.10. Slika 3.10. prikazuje isječak koda kartice početnoga zaslona gdje je na pritisak slike omogućen prikaz zaslona lekcije ako su zadovoljeni uvjeti. Ako je riječ o prvoj lekciji, nema uvjeta koji mora biti zadovoljen, lekcija nije zaključana. Za ostale je lekcije stavljeno ograničenje ako korisnik nije prikupio dovoljan broj bodova u prethodnoj lekciji ne može otvoriti iduću lekciju. Minimalan broj bodova koje korisnik mora prikupiti definirani su u bazi podataka gdje se, u istoj tablici, spremaju ostvareni korisnički bodovi. Ako korisnik pokuša

ući u lekciju za koju nema zadovoljen uvjet, prikaže mu se poruka „Nije ostvaren dovoljan broj bodova u prethodnoj lekciji.“.

```
enum class StoredDrawable(
    @DrawableRes val resId: Int,
    val stored: Int
) {
    BASICS(R.drawable.sketch1,  stored: 0),
    OPERATIONS(R.drawable.sketch2,  stored: 1),
    DETERMINANT(R.drawable.sketch3,  stored: 2),
    COFACTOR(R.drawable.sketch4,  stored: 3),
    LAPLACE(R.drawable.sketch5,  stored: 4),
    INVERSE(R.drawable.sketch6,  stored: 5),
    TRANSFORMATIONS(R.drawable.sketch7,  stored: 6);

    companion object {
        @DrawableRes
        fun asRes(stored: Int):Int = values().first{ it.stored == stored }.resId
    }

}
```

Sl. 3.9. *Enum* klasa *StoredDrawable*

```
Image(
    painter = painterResource(id = StoredDrawable.asRes(subject.id)),
    contentDescription = stringResource(id = StoredString.asRes(subject.id)),
    modifier = Modifier
        .fillMaxWidth()
        .height(194.dp)
        .clickable {
            if(topicNumber == 1) {
                navController.navigate( route: Screen.TabLayout.route+ "?topicNumber=$topicNumber")
            }
            if (previouslyEarnedPoints >= previousMinimumPoints) {
                navController.navigate( route: Screen.TabLayout.route+ "?topicNumber=$topicNumber")
            } else {
                scope.launch{ this: CoroutineScope
                    scaffoldState.snackbarHostState.showSnackbar(snackbarMessage)
                }
            }
        },
    contentScale = ContentScale.Crop
)
```

Sl. 3.10.Implementacija slike na kartici početnoga zaslona

Za navigaciju je korišten *NavController*. *NavController* je središnji API za navigacijsku komponentu. Ima stanje i prati *back stack composablea* koji čine zaslone u aplikaciji i stanje svakog zaslona [12]. *Back stack* je povijest korisnikove navigacije, stog koji bilježi kojim se

zaslonima kretao korisnik. Svaki *NavController* mora biti povezan s jednim *NavHostom*. *NavHost* povezuje *NavController* s putanjama koje su definirane za pojedini zaslon aplikacije. Zbog više lekcija, koje se razlikuju sadržajem, u putanje se dodaje argument *topicNumber*, broj koji definira koji je sadržaj potrebno prikazati na zaslonu.

Kako bi aplikacija radila očekivano, bez rušenja, korisniku je ograničen unos elemenata matrice na brojeve stotica s dva decimalna mjesta. Slika 3.11. prikazuje metodu za formatiranje broja korištenu u metodi za unos elemenata matrice na zaslonu za isprobavanje.

```
private fun formatNumber(number: Double): Double {
    val symbols: DecimalFormatSymbols = DecimalFormatSymbols.getInstance()
    symbols.decimalSeparator = '.'

    val decimalFormat = DecimalFormat( pattern: "##0.00")
    decimalFormat.maximumFractionDigits = 2
    decimalFormat.maximumIntegerDigits = 3
    decimalFormat.decimalFormatSymbols = symbols

    val formattedNumber = decimalFormat.format(number)
    return formattedNumber.toDouble()
}
```

Sl. 3.11. Formatiranje broja

3.6. Testiranje aplikacije

Osim testiranja aplikacije koristeći jedinične testove za provjeru matričnih operacija i testova korisničkog sučelja pri razvoju aplikacije, aplikacija je dana korisnicima kako bi se ispitala njezina jasnoća i korisnost pri samostalnom učenju o matricama.

Jedinično testiranje predstavlja provjeru točnosti pojedine komponente programske podrške neovisno o okolini ili o vanjskim ovisnostima te komponente. Za jedinične testove inicijalizirane su dvije matrice korisničkog tipa podatka *Matrix*, prikazane na slici 3.12. U klasi *Matrix* definirane su metode transponiranja, zbrajanja, oduzimanja, množenja matrica, računanje traga matrice i množenje matrice i skalara za koje su napravljeni jedinični testovi. Jedinični testovi napravljeni su i za računanje determinante i inverza matrice. Na slici 3.13. prikazani su jedinični testovi metoda za računanje traga matrice i transponiranje matrice. Kako bi funkcije bile testne označava ih se anotacijom *@Test*. Metoda *assertEquals* provjerava jednakost predanih argumenata i omogućuje testnoj metodi prikazivanje rezultata kao prolaska ili pada testa. Za provjeru transponirane matrice potrebno je provjeriti svaki element matrice. Ako bi se provjeravali objekti matrica rezultiralo bi padom testa zbog uspoređivanja instanci objekata koje se razlikuju.

```
    private val matrix = Matrix( rows: 3,  cols: 3)
    private val matrix2 = Matrix( rows: 3, cols: 2)

    init {
        //first matrix
        matrix[0,0] = 1.0
        matrix[0, 1] = 2.0
        matrix[0, 2] = 3.0
        matrix[1, 0] = 4.0
        matrix[1, 1] = 5.0
        matrix[1, 2] = 6.0
        matrix[2, 0] = 7.0
        matrix[2, 1] = 8.0
        matrix[2, 2] = 9.0

        //second matrix
        matrix2[0,0] = 1.0
        matrix2[1,0] = 3.0
        matrix2[2,0] = 5.0
        matrix2[0,1] = 7.0
        matrix2[1,1] = 9.0
        matrix2[2,1] = 2.0
    }
```

Sl. 3.12. Inicijalizirane matrice za jedinične testove

```
@Test
fun matrix_MatrixTrace_GetCorrectTrace() {

    val trace = matrix.getTrace(matrix)
    val expected = 15.0

    assertEquals(expected, trace)

}

@Test
fun matrix_MatrixTranspose_GetCorrectTransposedMatrix() {

    val transposed = matrix2.transpose()
    val transposedMatrix = Matrix( rows: 2, cols: 3)
    transposedMatrix[0,0] = 1.0
    transposedMatrix[1,0] = 7.0
    transposedMatrix[0,1] = 3.0
    transposedMatrix[1,1] = 9.0
    transposedMatrix[0,2] = 5.0
    transposedMatrix[1,2] = 2.0

    assertEquals(transposedMatrix[0,0], transposed[0,0])
    assertEquals(transposedMatrix[1,0], transposed[1,0])
    assertEquals(transposedMatrix[0,1], transposed[0,1])
    assertEquals(transposedMatrix[1,1], transposed[1,1])
    assertEquals(transposedMatrix[0,2], transposed[0,2])
    assertEquals(transposedMatrix[1,2], transposed[1,2])
}
```

Sl. 3.13. Testiranje metoda za računanje traga i transponiranje matrice

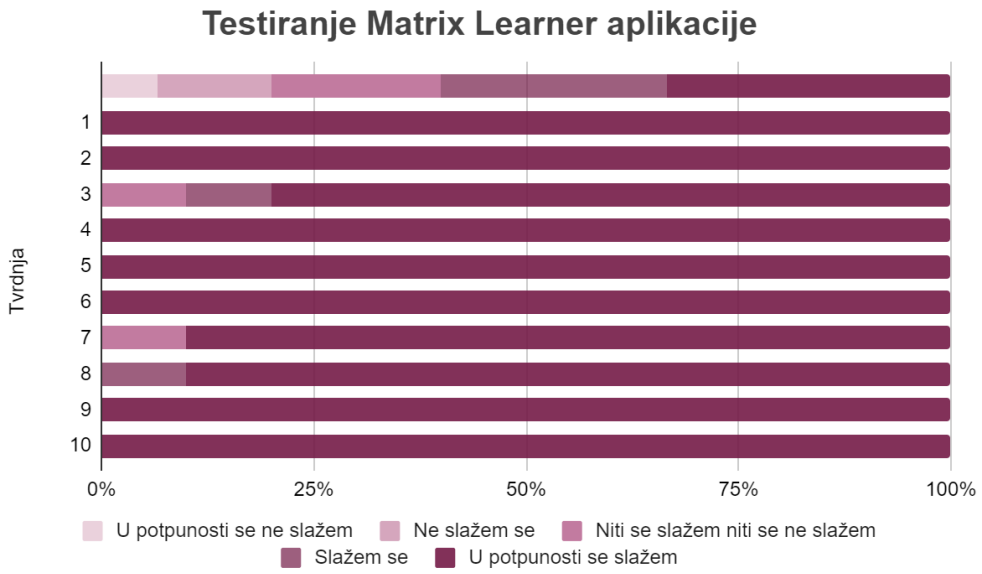
Upitnikom su prikupljene informacije o zadovoljstvu deset korisnika. Prikupljene informacije predstavljaju zadovoljstvo dizajnom aplikacije i procjene korisnika koliko svaka od ugrađenih funkcionalnosti pomaže u samostalnom učenju o matricama. Saznanja iz upitnika su nužna kako bi se uvidjeli nedostaci i nastavilo poboljšavati aplikaciju. Upitnik ima deset tvrdnji. Svaka je tvrdnja označena ocjenom od jedan do pet, prema Likertovoj ljestvici, gdje pet predstavlja potpuno slaganje korisnika s tvrdnjom, a jedan neslaganje. Uz to su postavljena dva neobavezna pitanja. U tablici 3.2. nalazi se deset tvrdnji i dva pitanja postavljena korisnicima u upitniku.

Slika 3.14. prikazuje rezultate testiranja. Odgovori su za većinu tvrdnji jednaki potpunom slaganju jer su ispitanici predstudentska i studentska populacija koja je upoznata s korištenjem mobilnih uređaja, a time i aplikacija sličnih karakteristika. Bez obzira na pozitivno iskustvo korisnika moguća su poboljšanja aplikacije dodavanjem dodatnih indikatora napretka. Zbog malog uzorka ispitanika i nedostatka uputa za korištenje aplikacije postoji mogućnost neslaganja svih korisnika s intuitivnošću aplikacije. Stoga postoji mogućnost dodavanja kratkih uputa na početnom zaslonu i zaslonu za isprobavanje. Iz ankete se može uočiti da je gradivo pravilno podijeljeno kako bi korisnik iz njega najviše naučio. Boje i čitljiv tekst pomažu korisniku da ostane raspoložen naučiti gradivo pojedine lekcije. Druga je važna stvar, oko koje su korisnici aplikacije bili jednoglasni, važnost kviza nakon svake lekcije te njegova korisnost. Uvođenjem kviza u lekciju omogućeno je uključivanje i motiviranje korisnika za provjeru znanja te utvrđivanje istog ponovnim rješavanjem kviza pojedine lekcije. Navedeno predstavlja ključne karakteristike igrifikacije aplikacija za učenje.

Tab. 3.2. Pitanja u upitniku

R. br.	Pitanje
1.	Aplikacija je intuitivna za korištenje. Lako se snalazim i razumijem uloge pojedinih dijelova (gumba, polja za popunjavanje).
2.	Vizualna odvojenost definicija različitim bojama u pojedinoj lekciji pomaže mi lakše usvojiti gradivo.
3.	Zvjezdice mi daju zadovoljavajući uvid u napredovanje pri učenju svake lekcije. (Nije mi potreban dodatni pokazatelj napredovanja.)
4.	Tekst je u aplikaciji čitljiv i jasan.
5.	Podjela gradiva na više manjih lekcija pomaže mi pri usvajanju gradiva.
6.	Gradivo je podijeljeno u logične cjeline koje nisu velikog obujma.
7.	Smatram da bih korištenjem aplikacije naučio/la dano gradivo.
8.	Matrični kalkulator za isprobavanje operacija nad matricama pridonosi boljem razumijevanju teorijskog dijela lekcije.
9.	Kviz na kraju pojedine lekcije pomaže mi pri vlastitom vrednovanju naučenosti gradiva u toj lekciji.
10.	Smatram da mi kviz na kraju svake lekcije pomaže usvojiti gradivo te lekcije.
11.	Postoji li nešto što Vam je zasmetalo u aplikaciji?
12.	Imate li prijedlog za poboljšane aplikacije?

Za neobavezna pitanja dobiven je jedan prijedlog za poboljšanje aplikacije: „Bilo bi zabavnije kada bi se dobivale medalje, uz već postojeće dijamante i zvjezdice. Da se dodjeljuje zlatna medalja ako je osoba svojim vježbanjem u pojedinoj lekciji dobila svih pet zvjezdica.“ . Na temelju prijedloga, jedna od mogućnosti za proširenje aplikacije je stvaranje zaslona korisničkog profila s osvojenim medaljama za pojedine lekcije kako se ne bi pretrpao početni zaslon. Uz uvođenje korisničkog profila poželjno bi bilo implementirati ljestvicu poretka korisnika prema osvojenim bodovima, medaljama ili nivou za što bi bilo potrebno dijeljenje podataka API-em. Dodatni elementi nagrađivanja stvaraju veći poticaj pri učenju i ujedno predstavljaju jedan od temeljnih elemenata igrifikacije aplikacija.



Sl. 3.14. Prikaz rezultata testiranja aplikacije

4. ZAKLJUČAK

Studenti tehničkih fakulteta susreću se s kolegijem Linearna algebra koji u svojem sadržaju ima matrice i razne matrične operacije, transformacije, dekompozicije i jednadžbe koje se koriste u drugim područjima tehničkih znanosti. Stoga je nastala potreba digitalnog prilagođavanja sadržaja kolegija prirodnih znanosti s jednostavnim i zanimljivim sučeljem za učenje i usvajanje osnovnih pojmova i koncepata. Poznato je da su digitalni alati praktični i korisni pri usvajanju novih znanja zbog jednostavnosti i mobilnosti. Zbog toga je u porastu upotreba edukacijskih mobilnih aplikacija i alata. Cilj je ovog rada bio izraditi aplikaciju za samostalno učenje o matricama koja sadrži osnovne pojmove o matricama i njihovim operacijama. U radu su definirani osnovni pojmovi oko matrica, vrsta matrica, operacija s matricama, metodama za računanje determinante i inverza matrice. Također, opisane su funkcionalnosti različitih aplikacija koje su povezane s matricama, najčešće matrični kalkulatori i aplikacija s igrifikacijskim elementima za učenje matematike. Ostvareno je programsko rješenje za samostalno učenje o matricama. Predstavljeni su zahtjevi aplikacije, korištene tehnologije za izradu aplikacije, MVVM arhitekturni obrazac. Prikazan je dijagram toka aplikacije kako bi se uočile sve mogućnosti korisničkog kretanja kroz aplikaciju. Opisani su i načini rješavanja različitih problema pri izradi aplikacije i rezultati testiranja aplikacije.

Praktična preglednost gradiva, omogućena jednostavna provjera usvojenosti gradiva rješavajući kviz i matrični kalkulator za pomoć ili provjeru pri rješavanju zadataka s matricama predstavljaju implementirane prednosti *Matrix Learner* aplikacije. Zbog ograničenosti dimenzije matrica i elemenata matrice koji mogu biti samo realni brojevi, postoji mogućnost proširenja aplikacije kako bi postala praktična za rad s većim dimenzijama matrica i kompleksnim brojevima. Postojanjem jediničnih testova omogućena je laka provjera ispravnosti koda nakon kasnijih promjena. Korisničko testiranje aplikacije pokazalo je zadovoljstvo korisnika svim implementiranim dijelovima aplikacije.

Postoji mogućnost za brojna proširenja i unaprjeđenja. U aplikaciji Matrix Learner stvorila se mogućnost uvođenja dodatnih elemenata nagrađivanja poput medalje koja bi se dodijelila korisniku za lekciju u kojoj je osvojio svih pet zvjezdica. Takva vrsta virtualne nagrade bila bi dodatan element igrifikacije koja bi potaknula korisnika da riješi sve lekcije više puta, a time i nauči svo gradivo u aplikaciji. Nadalje dodavanje zaslona korisničkog profila s osvojenim medaljama predstavljao bi praktičan prikaz svih medalja koje je korisnik osvojio, ali i onih koje tek može osvojiti. Ljestvica poretka i nivoi korisnika predstavljali bi komponente igrifikacije

aplikacije s notom natjecanja i bolje povratne informacije o vlastitom uspjehu. Daljnje unaprjeđivanje aplikacije moglo bi uključiti rad s kompleksnim brojevima. Sljedeća je mogućnost proširivanje dimenzije matrica jer se vrlo često u praksi radi s vrlo velikim matricama kao što je primjer konvolucija slika. Kako bi se proširio opseg pitanja kviza moguće je uvođenje opcije koja bi omogućila dodavanje korisničkih pitanja za kvizove.

LITERATURA

[1] N. Elezović, Linearna algebra, Element, Zagreb, 2016.

[2] S. Kurepa, Uvod u linearnu algebru, Školska knjiga, Zagreb, 1990.

[3] ALGSoftwareLab, Linear Algebra Solver [online], Google Commerce Ltd, 2022., dostupno na: https://play.google.com/store/apps/details?id=com.algosoftware.matrix [07.06.2023.]

[4] M. Vinicius, Matriz Gauss Jordan [online], Google Commerce Ltd, 2021., dostupno na: https://play.google.com/store/apps/details?id=com.viniciusmello.escalonamento [09.06.2023.]

[5] Grunwald Associates LLC, Living and Learning with Mobile Devices: What Parents Think About Mobile Devices for Early Childhood and K–12 Learning, 2013., dostupno na: https://grunwald.com/pdfs/Grunwald%20Mobile%20Study%20public%20report.pdf [02.07.2023.]

[6] Android Developers, Jetpack Compose Tutorial [online], Google, dostupno na: https://developer.android.com/jetpack/compose/tutorial [18.07.2023.]

[7] Enginess, What is Material Design and How Can It Be Used? [online], 2016., dostupno na: https://www.enginess.io/insights/design-trend-material-design. [18.07.2023.]

[8] Koin, Injecting in Android [online], dostupno na: https://insert-koin.io/docs/reference/koin-android/get-instances [14.07.2023.]

[9] Room, Save data in a local database in Room [online], dostupno na: https://developer.android.com/training/data-storage/room. [15.07.2023.].

[10] R. Mishra, MVVM (Model View ViewModel) Architecture Pattern in Android [online], GeeksforGeeks, dostupno na: https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/#article-meta-div [19.07.2023.]

[11] Android Developers, Referencing complex data using Room [online], Google, dostupno na: https://developer.android.com/training/data-storage/room/referencing-data [20.07.2023.]

[12] Android Developers, Navigating with Compose [online], Google, dostupno na: https://developer.android.com/jetpack/compose/navigation [23.05.2023.]

[13] K. Horvatić, Linearna algebra (IV izdanje) II. dio, Sveučilišna naklada Liber Zagreb, Zagreb, 1986.

[14] HMLA, Matrix operations [online], Google Commerce Ltd, 2017., dostupno na: https://play.google.com/store/apps/details?id=com.highermathematics.linearalgebra.free. [21.06.2023.]

[15] Bazzigate corp, Matrix Calculator (Algebra) [online], Google Commerce Ltd., 2022. dostupno na: https://play.google.com/store/apps/details?id=com.algebra.matrix.calculator. [21. 06.2023.]

[16] AdamSoftware, Matematika - Matematičke igre [online], Google Commerce Ltd, 2020., dostupno na: https://play.google.com/store/apps/details?id=com.companyname.MaturaMatematyka. [21.06.2023.]

[17] S. Criollo-C, A. Guerrero-Arias, Á. Jaramillo-Alcázar, i S. Luján-Mora, "Mobile Learning Technologies for Education: Benefits and Pending Issues," *Applied Sciences*, vol. 11, no. 9, p. 4111, Travanj 2021 [online], dostupno na: http://dx.doi.org/10.3390/app11094111 [01.08.2023.]

[18] C. H. Symbolab, Symbolab, Making Math Simpler, Course Hero Symbolab Ltd., [online], dostupno na: https://www.symbolab.com [16. 08.2023]

SAŽETAK

U teorijskom dijelu rada opisani su osnovni pojmovi i operacije s matricama korištene u aplikaciji za samostalno učenje o matricama. Kako bi se korisniku omogućilo samostalno usvajanje osnovnih pojmova, gradivo je podijeljeno u sedam lekcija za koje je izrađen matrični kalkulator i kviz s pitanjima iz te lekcije. Praćenjem osvojenih bodova korisnika, otključavanjem lekcija, korištenjem kviza i stvaranjem vizualno ugodnog sučelja ostvareni su igrifikacijski elementi u aplikaciji koji potiču korisnika na samostalno učenje. *Android* aplikacija izrađena je u programskom jeziku *Kotlin* s *Jetpack Composom,* bibliotekom *Room* i *Koinom*. U radu su opisani zahtjevi, korištene tehnologije i MVVM arhitekturni obrazac. Dijagram toka prikazuje moguće kretanje korisnika kroz aplikaciju. Uz to je opisano korištenje aplikacije i problemi s rješenjima pri izradi aplikacije. Rezultati testiranja aplikacije ukazuju na pozitivno korisničko iskustvo i zadovoljstvo implementiranim značajkama aplikacije.

Ključne riječi: *Android* aplikacija, *Jetpack Compose*, matrice, MVVM, samostalno učenje

ABSTRACT

Mobile application for independent learning about matrices

In the theoretical part of the thesis, the basic concepts and operations with matrices used in the application for self-learning about matrices are described. In order to enable the user to learn basic concepts independently, the material is divided into seven lessons for which a matrix calculator and a quiz with questions from that lesson have been created. By monitoring the user's earned points, unlocking lessons, using quizzes and creating a visually pleasant interface, gamification elements were realized in the application that encourages the user to study independently. The Android application is built using the Kotlin programming language with Jetpack Compos, the Room library and Koin. The paper describes the requirements, used technologies, and the MVVM architectural pattern. The flow diagram shows the possible movement of the user through the application. In addition, application usage and problems with solutions when building application are described. The application testing results indicate a positive user experience and satisfaction with the implemented application features.

Keywords: Android application, Jetpack Compose, matrices, MVVM, independent learning