Implementacija Sistema Preporuka

1. Opis Implementacije

Sistem preporuka je implementiran koristeći **User-Based Collaborative Filtering** pristup, koji pretpostavlja da će korisnici koji su imali slična mišljenja u prošlosti, vjerovatno dijeliti mišljenje i u budućnosti. Ovaj sistem funkcioniše tako da poredi ponašanje korisnika prilikom ocjenjivanja i preporučuje proizvode koje su ocijenili slični korisnici.

Koraci:

- Model se trenira koristeći podatke o ocjenama koji su pohranjeni u bazi podataka.
- Za svakog korisnika predviđa se koji proizvodi će mu se svidjeti, na osnovu ocjena drugih sličnih korisnika.
- Ako je predviđena ocjena proizvoda iznad određenog praga (3.5 u ovom slučaju), proizvod se preporučuje korisniku.

2. Izvorni Kod: Glavna Logika Sistema Preporuka

Putanja:

C:\Users\User\Desktop\rs 2\eZamjena\eZamjena.Services\ProizvodService.cs

Kod:

```
1 reference
private void TrainModel()
{
    // Učitaj sve ocjene iz baze
    var data = Context.Ocjenas.Select(o => new RatingEntry
    {
        UserId = (uint)o.KorisnikId,
        ProductId = (uint)o.ProizvodId,
        Label = (float)o.Ocjena1
    }).ToList();

    var trainData = mlContext.Data.LoadFromEnumerable(data);

    // Konfiguracija Matrix Factorization trenera
    var options = new MatrixFactorizationTrainer.Options
    {
        MatrixColumnIndexColumnName = nameof(RatingEntry.ProductId),
        MatrixRowIndexColumnName = nameof(RatingEntry.UserId),
        LabelColumnName = "Label",
        NumberOfIterations = 20,
        ApproximationRank = 100
    };

    var trainer = mlContext.Recommendation().Trainers.MatrixFactorization(options);
    model = trainer.Fit(trainData);
}
```

```
3 references
public IEnumerable<Model.Proizvod> RecommendProducts(int userId)
{
    // Učitaj sve proizvode
    var allProducts = Context.Proizvods.Where(p=> p.StatusProizvodaId==1).ToList();//u prodaji

    // Kreiraj PredictionEngine
    var engine = mlContext.Model.CreatePredictionEngine<RatingEntry, RatingPrediction>(model);

    var results = new List<Model.Proizvod>();

    foreach (var product in allProducts)
    {
        var prediction = engine.Predict(new RatingEntry
        {
            UserId = (uint)userId,
            ProductId = (uint)product.Id
        });

        if (prediction.Score > 3.5)  // Primjer: filtriranje proizvoda s visokom predviđenom ocjenom
        {
            results.Add(Mapper.Map<Model.Proizvod>(product));
        }
    }

    return results;
}
```

```
private void TrainModel()
{
    // Učitaj sve ocjene iz baze
    var data = Context.Ocjenas.Select(o => new RatingEntry
    {
        UserId = (uint)o.KorisnikId,
        ProductId = (uint)o.ProizvodId,
        Label = (float)o.Ocjena1
    }).ToList();

    var trainData = mlContext.Data.LoadFromEnumerable(data);

    // Konfiguracija Matrix Factorization trenera
    var options = new MatrixFactorizationTrainer.Options
    {
        MatrixColumnIndexColumnName = nameof(RatingEntry.ProductId),
        MatrixRowIndexColumnName = nameof(RatingEntry.UserId),
        LabelColumnName = "Label",
        NumberOfIterations = 20,
        ApproximationRank = 100
    };

    var trainer = mlContext.Recommendation().Trainers.MatrixFactorization(options);
    model = trainer.Fit(trainData);
}
```

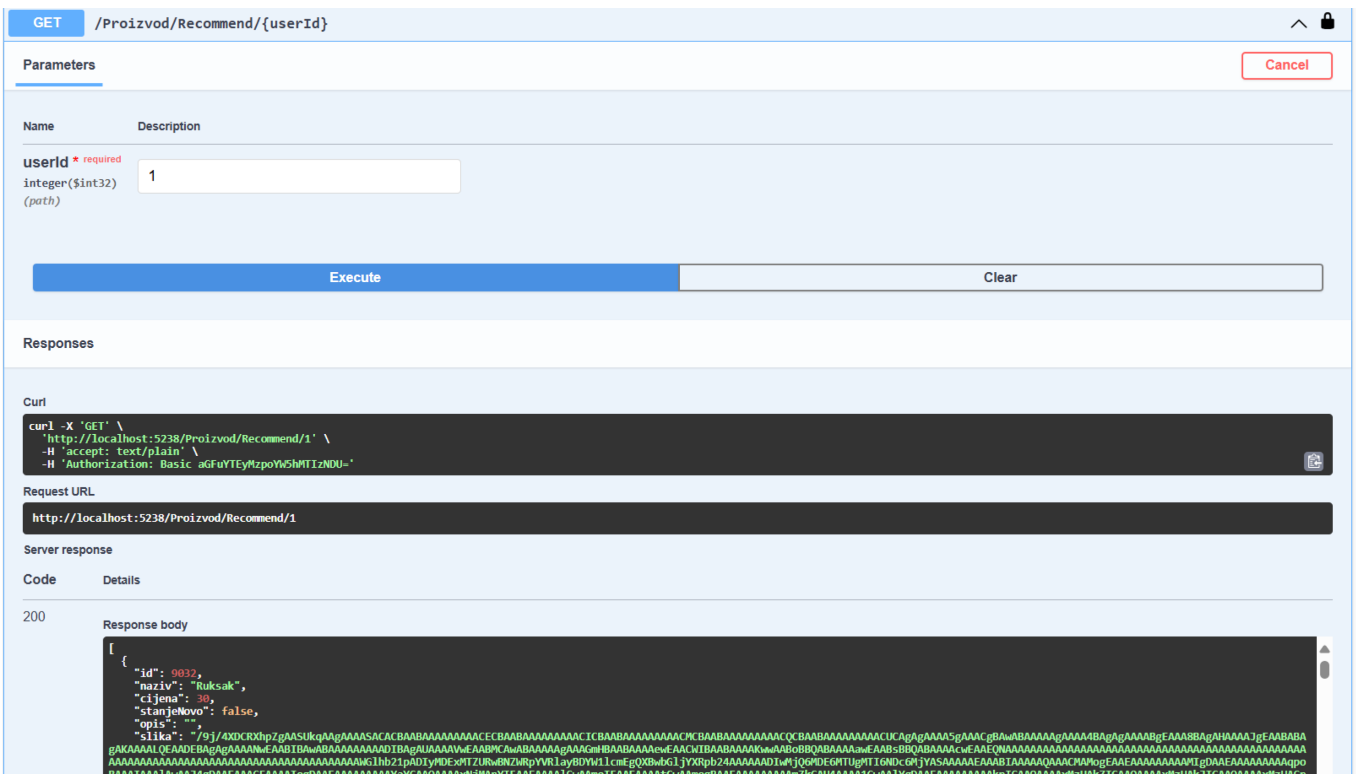
```
public IEnumerable<Model.Proizvod> RecommendProducts(int userId)
{
    // Učitaj sve proizvode koji su na prodaji
    var allProducts = Context.Proizvods.Where(p => p.StatusProizvodaId == 1).ToList(); // Na prodaji

    // Kreiraj PredictionEngine
    var engine = mlContext.Model.CreatePredictionEngine<RatingEntry, RatingPrediction>(model);

    var results = new List<Model.Proizvod>();

    foreach (var product in allProducts)
    {
        var prediction = engine.Predict(new RatingEntry
        {
            UserId = (uint)userId,
            ProductId = (uint)product.Id
        });

        if (prediction.Score > 3.5)  // Filtriranje proizvoda s visokom predviđenom ocjenom
        {
            results.Add(Mapper.Map<Model.Proizvod>(product));
        }
    }

    return results;
```

}

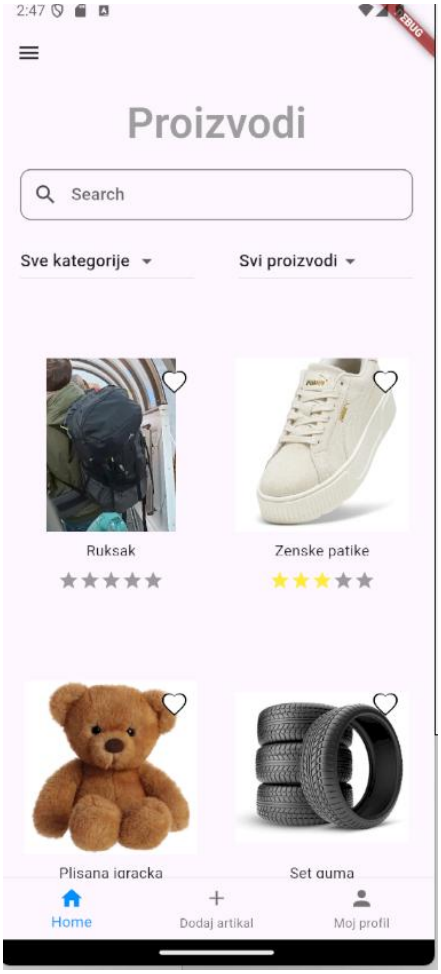
3. Screenshot iz Pokrenute Aplikacije

Priložite screenshot ekrana aplikacije na kojem su prikazani preporučeni proizvodi korisniku. Screenshot treba prikazivati interfejs nakon što se izvrši metoda RecommendProducts, a preporučeni proizvodi na osnovu sličnosti korisnika trebaju biti vidljivi.

- Funkcija GetUserSpecificProducts prvo dobije preporučene proizvode za određenog korisnika pomoću RecommendProducts metode.
- Zatim se preostali proizvodi (koji nisu u vlasništvu korisnika) dodaju u listu, a preporučeni proizvodi su postavljeni na vrh te liste.
- Na taj način korisnik prvo vidi proizvode koji su mu specifično preporučeni, dok ostali proizvodi slijede ispod njih.



Na prethodnoj slici prikazan je api koji vraca preporucene proizvode koji ce se korisniku prikazati prije ostaloh proizvoda u listi svih proizvoda.



Lista proizvoda koja je sortirana na način da se prvo prikažu preporučeni proizvodi,