1. **Opis implementacije recommender sistema**

Za recommender sistem korišten je content-based pristup. Cilj je korisniku preporučiti knjige na osnovu ponašanja sličnih korisnika i njihovih interakcija sa knjigama. U slučaju da je u pitanju novi korsnik po defaultu će mu biti prikazane najpopularnije knjige kao fallback jer recommender neće imati podatke da bi napravio neku predikciju.

Podaci se sastoje od dvije glavne grupe:

Korisničke interakcije – lista rezervacija koje je korisnik napravio

Ocjene interakcija – vrijednosti koje odražavaju kvalitet interakcije: 1.0 za odobrene rezervacije, 0.5 za neodobrene

ML.NET koristi Matrix Factorization tehniku kako bi predstavio interakcije kao matricu korisnik-knjiga u višedimenzionalnom prostoru. Algoritam faktorizuje ovu matricu pronalazeći latentne faktore koji objašnjavaju obrasce ponašanja. Ako korisnik često rezerviše knjige određenog tipa koje rezervišu i drugi slični korisnici, model će identificirati ove skrivene veze i preporučiti knjige koje su ti slični korisnici također rezervisali, čak i ako one nisu očigledno povezane.

2. **Putanja i printscreen source code-a glavne logike recommender sistema**

eBiblioteka/eBiblioteka.Servisi/ Recommender /RecommenderServis.cs

```
    public List<KnjigaDTO> PreporuciKnjige(int korisnikId)
{

    OsigurajTreniranjeModela();

    var korisnikRezervacije = _context.Rezervacijas
    .Where(r => r.KorisnikId == korisnikId && r.Odobrena == true)
    .Select(r => r.KnjigaId)
    .Distinct()
    .ToList();

    if(!korisnikRezervacije.Any())
    {
        return GetPopularneKnjige();
    }

    var preporuke = new List<(Knjiga knjiga, float score)>();
    var dostupneKnjige = _context.Knjigas.
        Where(x => x.Dostupna == true &&
            x.IsDeleted == false &&
            korisnikRezervacije.Contains(x.KnjigaId)).ToList();

    var predictionEngine =
_mlContext.Model.CreatePredictionEngine<KnjigaInteraction,
ScorePrediction>(_model);

    foreach (var knjiga in dostupneKnjige)
    {
        var prediction = predictionEngine.Predict(new KnjigaInteraction
        {
            KorisnikId = (uint)korisnikId,
```

```
            KnjigaId = (uint)knjiga.KnjigaId,
        });

        preporuke.Add((knjiga, prediction.Score));
    }

    var topPreporuke = preporuke.OrderByDescending(x => x.score)
        .Take(5).Select(x => x.knjiga).ToList();

    return _mapper.Map<List<KnjigaDTO>>(topPreporuke);
}

private void OsigurajTreniranjeModela()
{
    if (_model == null)
    {
        lock (_isLocked)
        {
            if (_model == null)
            {
                TrenirajModel();
            }
        }
    }
}

private void TrenirajModel()
{
    if (_mlContext == null)
    {
        _mlContext = new MLContext();
    }

    var trainingData = PrepareTrainingData();
    var dataView =
_mlContext.Data.LoadFromEnumerable<KnjigaInteraction>(trainingData);
    var options = new MatrixFactorizationTrainer.Options
    {
        MatrixColumnIndexColumnName = nameof(KnjigaInteraction.KorisnikId),
        MatrixRowIndexColumnName = nameof(KnjigaInteraction.KnjigaId),
        LabelColumnName = nameof(KnjigaInteraction.Label),
        LossFunction =
MatrixFactorizationTrainer.LossFunctionType.SquareLossRegression,
        Alpha = 0.01,
        Lambda = 0.025,
        NumberOfIterations = 100,
        C = 0.00001,
        ApproximationRank = 64
    };

    var trainer =
_mlContext.Recommendation().Trainers.MatrixFactorization(options);
    _model = trainer.Fit(dataView);
```

```
    using (var fs = new FileStream(Path, FileMode.Create, FileAccess.Write,
FileShare.Write))
    {
        _mlContext.Model.Save(_model, dataView.Schema, fs);
        Console.WriteLine("Data saved");
    }
}

private List<KnjigaInteraction> PrepareTrainingData()
{
    var interakcije= new List<KnjigaInteraction>();

    var rezervacije = _context.Rezervacijas
        .Where(x => x.KorisnikId.HasValue && x.KnjigaId.HasValue)
        .GroupBy(x => new { x.KorisnikId, x.KnjigaId })
        .Select(x => new
        {
            KorisnikId = x.Key.KorisnikId.Value,
            KnjigaId = x.Key.KnjigaId.Value,
            Count = x.Count(),
            Odobrena = x.Any(y => y.Odobrena == true)
        }).ToList();

    foreach (var rezervacija in rezervacije)
    {
        float rating = rezervacija.Odobrena ? 1.0f : 0.5f;

        interakcije.Add(new KnjigaInteraction
        {
            KorisnikId = (uint)rezervacija.KorisnikId,
            KnjigaId = (uint)rezervacija.KnjigaId,
            Label = rating
        });
    }

    return interakcije;
}

private List<KnjigaDTO> GetPopularneKnjige()
{
    var popularneKnjige = _context.Knjigas
        .Where(k => k.Dostupna == true && k.IsDeleted != true)
        .OrderByDescending(k => k.Rezervacijas.Count(r => r.Odobrena == true))
        .ThenByDescending(k => k.Preporuceno == true)
        .ThenByDescending(k => k.KnjigaDana == true)
        .Take(5)
        .ToList();

    return _mapper.Map<List<KnjigaDTO>>(popularneKnjige);
}
```

3. **Putanja i printscreen iz pokrenute aplikacije gdje se prikazuju preporuke**

Prijaviti se kao korisnik na mobilnu aplikaciju (korisnicko ime: korisnik, sifra: korisnik). Na početnoj stranici treba scrollati do kraja stranice i tu će biti sekcija „Za vas“ gdje ce biti prikazani rezultati recommendera.