DOKUMENTACIJA SISTEMA PREPORUKA (RECOMMENDER SYSTEM)

Projekat: MošPosudit - Aplikacija za iznajmljivanje alata

Datum: 05.03.2025

Fakultet informacijskih tehnologija

Student: Sinancevic Zahir IB220116

---

1. OPIS IMPLEMENTACIJE RECOMMENDER SISTEMA

1.1 Opšti Pregled

Sistem preporuka u projektu MošPosudit implementiran je kao **napredni sistem sa dva algoritma** koji omogućava personalizirane preporuke alata korisnicima. Sistem je dizajniran da radi u **tri različita režima** kako bi osigurao maksimalnu fleksibilnost i pouzdanost.

1.2 Dva Implementirana Algoritma

Algoritam 1: Machine Learning - Matrix Factorization

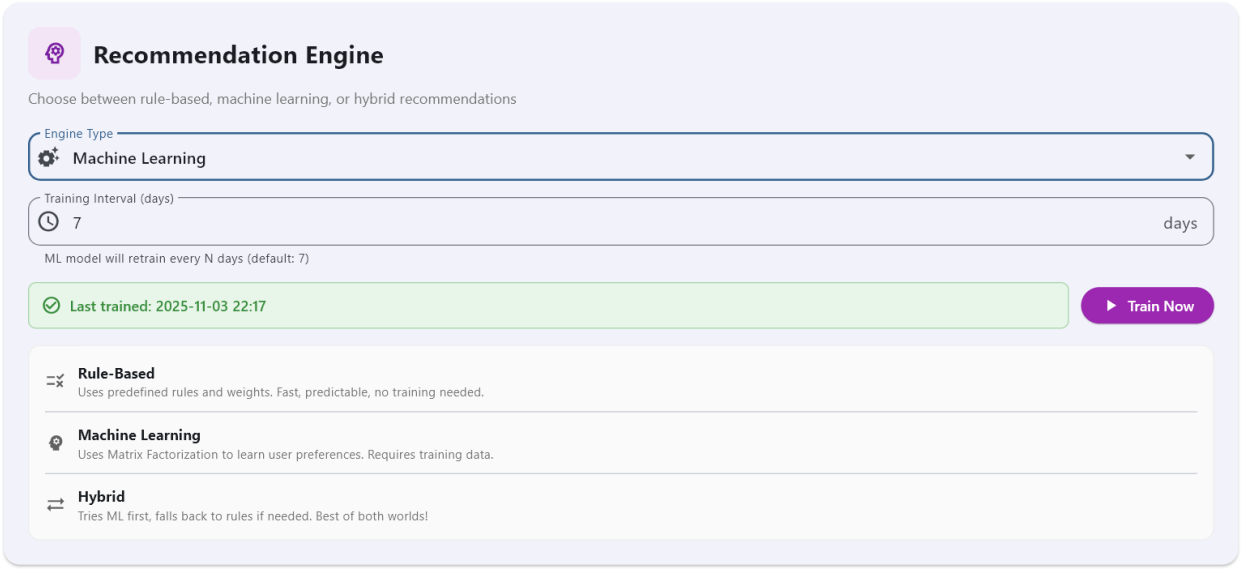
Koristi Microsoft ML.NET biblioteku za predviđanje preferencija korisnika na osnovu historijskih podataka. Algoritam Matrix Factorization dekomponuje matricu interakcija korisnik-alat u latentne faktore i koristi ih za predviđanje score-a za neviđene parove (korisnik, alat).

Algoritam 2: Rule-Based sa Konfigurablinim Parametrima

Koristi unaprijed definirane algoritme sa dinamičkim težinskim faktorima koji se mogu podešavati kroz admin panel. Ovaj algoritam služi i kao fallback kada ML model nije dostupan.

1.3 Tri Režima Rada

1. **MachineLearning Mode** - Koristi isključivo ML algoritam
2. **RuleBased Mode** - Koristi isključivo Rule-Based algoritam (parametri se mogu podešavati)
3. **Hybrid Mode** - Pokušava ML prvo, automatski prelazi na Rule-Based ako ML ne vrati dovoljno rezultata (PREPORUČENO)



1.4 Vrste Preporuka

Sistem implementira dvije glavne vrste preporuka:

1.4.1 Home Recommendations (Preporuke na početnoj stranici)

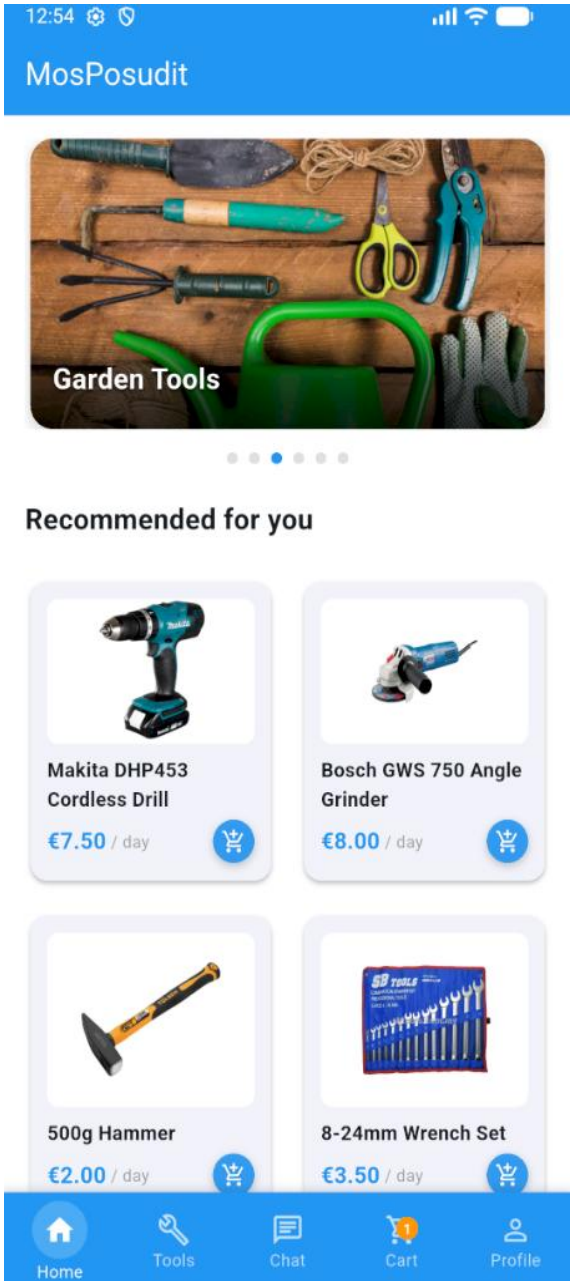
Personalizirane preporuke koje se prikazuju na home screen-u aplikacije:

- **Broj preporuka:** 6 alata
- **Prikazivanje:** Grid layout sa 2 kolone
- **Lokacija:** Ispod kategorija, sekcija "Recommended for you"

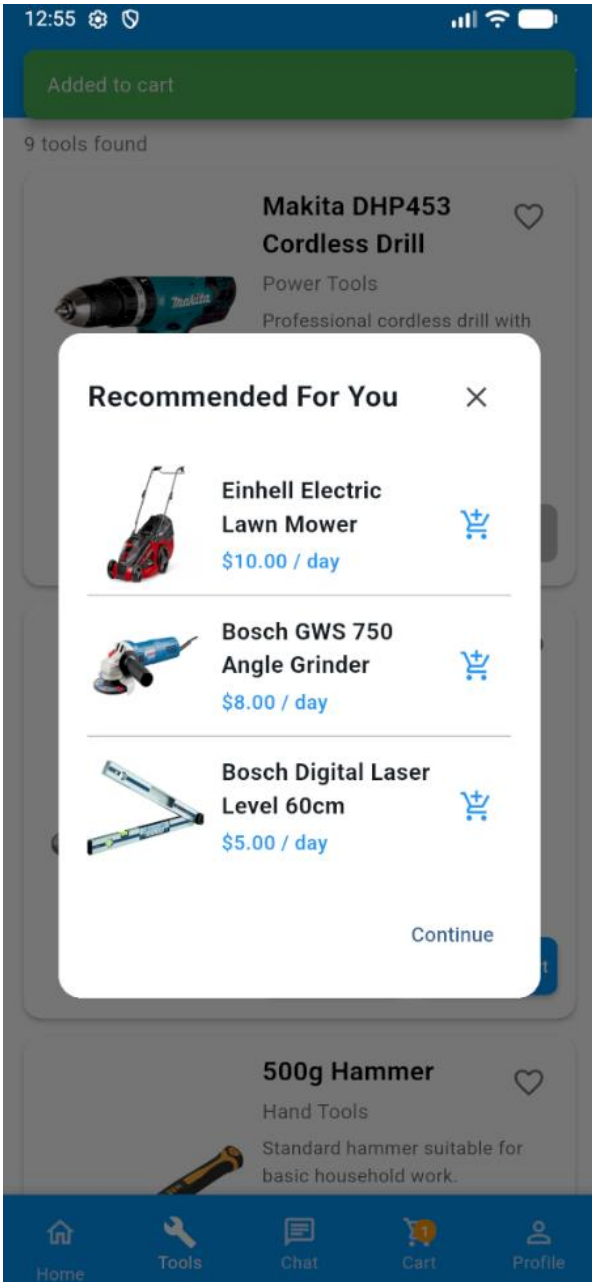
1.4.2 Cart Recommendations (Preporuke u košarici)

Preporuke koje se prikazuju nakon što korisnik doda alat u košaricu:

- **Broj preporuka:** 3 alata
- **Prikazivanje:** Dialog sa listom
- **Auto-dismiss:** Dialog se automatski zatvara nakon 3 sekunde
- **Funkcionalnost:** Korisnik može dodati preporučene alate direktno iz dialoga



Home Recommendations



Cart Recommendations

2. DETALJAN OPIS ALGORITAMA

2.1 Machine Learning Algoritam - Matrix Factorization

Tehnologija: Microsoft ML.NET

Algoritam: Matrix Factorization (Collaborative Filtering)

Princip rada:

1. Treniranje modela:
   - Učitavanje podataka iz baze (Order → OrderItem tabele)
   - Kreiranje training dataset-a (UserId, ToolId, Label=1 za svaku narudžbu)
   - Treniranje Matrix Factorization modela
   - Čuvanje modela na disk (.zip fajl)
2. Predviđanje (Home Recommendations):
   - Učitavanje treniranog modela u memoriju (cache)
   - Za datog korisnika, predviđanje score-a za sve dostupne alate
   - Isključivanje alata koje je korisnik već iznajmio
   - Sortiranje po score-u (od najvišeg ka najnižem)
   - Vraćanje top N alata
3. Predviđanje (Cart Recommendations):
   - Pronalaženje korisnika koji su iznajmili trenutni alat
   - Za svakog takvog korisnika, predviđanje score-a za ostale alate
   - Računanje prosječnog score-a za svaki alat
   - Vraćanje top N alata sa najvišim prosječnim score-om

Prednosti:

- Visoka personalizacija
- Automatsko otkrivanje skrivenih pattern-a u podacima
- Poboljšava se sa više podataka

Nedostaci:

- Zahtijeva trenirani model
- Cold start problem (novi korisnici/alati)

Putanja: MosPosudit.Worker\Services\MLTrainingService.cs
Screenshot od linije 32-94 (Glavna logika treniranja):

```
// Trains a new ML recommendation model using all available order data
public async Task<bool> TrainModelAsync()
{
    try
    {
        _logger.LogInformation("Starting ML recommendation model training...");

        var trainingData = await PrepareTrainingDataAsync();

        if (trainingData.Count < 10)
        {
            _logger.LogWarning("Not enough training data (minimum 10 interactions required). Current count: {Count}", trainingData.Count);
            return false;
        }

        _logger.LogInformation("Prepared {Count} training samples", trainingData.Count);

        var mlContext = new MLContext(seed: 0);
        var dataView = mlContext.Data.LoadFromEnumerable(trainingData);
        var dataSplit = mlContext.Data.TrainTestSplit(dataView, testFraction: 0.2);

        _logger.LogInformation("Building Matrix Factorization pipeline...");
        var options = new MatrixFactorizationTrainer.Options
        {
            MatrixColumnIndexColumnName = "UserId",
            MatrixRowIndexColumnName = "ToolId",
            LabelColumnName = "Label",
            NumberOfIterations = 20,
            ApproximationRank = 100,
            Quiet = false
        };
```

```
        var pipeline = mlContext.Transforms.Conversion.MapValueToKey("UserId", "UserId")
            .Append(mlContext.Transforms.Conversion.MapValueToKey("ToolId", "ToolId"))
            .Append(mlContext.Recommendation().Trainers.MatrixFactorization(options));

        _logger.LogInformation("Training model... This may take a few minutes.");
        var model = pipeline.Fit(dataSplit.TrainSet);

        var predictions = model.Transform(dataSplit.TestSet);
        var metrics = mlContext.Regression.Evaluate(predictions, labelColumnName: "Label");

        _logger.LogInformation("Model training completed!");
        _logger.LogInformation("Model Metrics - RMSE: {RMSE:F4}, RSquared: {RSquared:F4}",
            metrics.RootMeanSquaredError, metrics.RSquared);
        var modelFileName = $"recommendation_model_{DateTime.UtcNow:yyyyMMdd_HHmmss}.zip";
        var modelPath = Path.Combine(_modelStoragePath, modelFileName);

        mlContext.Model.Save(model, dataView.Schema, modelPath);
        var fileInfo = new FileInfo(modelPath);

        _logger.LogInformation("Model saved to: {Path} (Size: {Size} bytes)", modelPath, fileInfo.Length);

        await SaveModelMetadataAsync(modelFileName, modelPath, trainingData.Count, metrics, fileInfo.Length);

        _logger.LogInformation("ML model training completed successfully!");
        return true;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error training ML recommendation model");
        return false;
    }
```

2.2 Rule-Based Algoritam sa Konfiguracijom

Home Recommendations - Tri Komponente:

A) Content-Based Filtering (HomeContentBasedWeight - default 30%)

- Analizira omiljene kategorije korisnika iz zadnjih 90 dana
- Preporučuje alate iz tih kategorija koje korisnik još nije iznajmio
- Sortira po cijeni (preferira skuplje alate = bolji kvalitet)

B) Popular/Trending Tools (HomePopularWeight - default 40%)

- Pronalazi najčešće iznajmljene alate u zadnjih 30 dana
- Sortira po broju iznajmljivanja
- Pokazuje "trending" alate

C) Top Rated Tools (HomeTopRatedWeight - default 30%)

- Prikazuje alate sa prosječnom ocjenom ≥ 4.0
- Minimum 2 recenzije obavezno
- Sortira po prosječnoj ocjeni, zatim po broju recenzija

Cart Recommendations - Dvije Komponente:

A) Frequently Bought Together (CartFrequentlyBoughtWeight - default 60%)

- Pronalazi narudžbe koje sadrže trenutni alat
- Analizira koje druge alate su u tim narudžbama
- Sortira po frekvenciji zajedničke pojave
- Princip: "Korisnici koji su iznajmili X, često iznajmljuju i Y"

B) Similar Tools (CartSimilarToolsWeight - default 40%)

- Pronalazi alate iz iste kategorije
- Filtrira po sličnim ocjenama (±1.0 od trenutnog alata)
- Sortira po ocjeni (od najviše ka najnižoj)

Prednosti:

- Ne zahtijeva trenirani model
- Transparentno (znamo zašto je nešto preporučeno)
- Parametri se mogu dinamički podešavati kroz admin panel

Fallback mehanizam:

Ako gornji algoritmi ne vrate dovoljno rezultata:

1. Dopuni sa dostupnim alatima iz iste kategorije
2. Ako još uvijek nema dovoljno, dopuni sa bilo kojim dostupnim alatima
3. Uvijek vraća nešto ako postoje alati u sistemu

Putanja: MosPosudit.Services\Services\MLPredictionService.cs

Screenshot od linije 28-159 (Glavna logika predviđanja):

- Home recommendations (linija 28-89)
- Cart recommendations (linija 91-159)

```
// Gets ML-based recommendations for a user by predicting scores for all available tools
1 reference | Zahir Sirančević, 3 hours ago | 1 author, 1 change
public async Task<List<int>> GetMLRecommendationsAsync(int userId, int count = 6)
{
    try
    {
        // Ensure model is loaded
        await EnsureModelLoadedAsync();

        if (_predictionEngine == null)
        {
            _logger.LogWarning("ML model not available for predictions");
            return new List<int>();
        }

        using var scope = _serviceProvider.CreateScope();
        var context = scope.ServiceProvider.GetRequiredService<ApplicationDbContext>();

        // Get all available tools
        var availableTools = await context.Set<Tool>()
            .Where(t => t.IsAvailable && t.Quantity > 0)
            .Select(t => t.Id)
            .ToListAsync();

        // Get tools user has already ordered (to exclude them)
        var userOrderedToolIds = await context.Set<Order>()
            .Where(o => o.UserId == userId)
            .SelectMany(o => o.OrderItems)
            .Select(oi => oi.ToolId)
            .Distinct()
            .ToListAsync();

        // Predict scores for all available tools the user hasn't ordered
        var predictions = new List<(int ToolId, float Score)>();

        foreach (var toolId in availableTools.Where(id => !userOrderedToolIds.Contains(id)))
        {
            var input = new ToolRating
            {
                UserId = userId,
                ToolId = toolId
            };

            var prediction = _predictionEngine.Predict(input);
            predictions.Add((toolId, prediction.Score));
        }

        // Return top N tools by predicted score
        var recommendations = predictions
            .OrderByDescending(p => p.Score)
            .Take(count)
            .Select(p => p.ToolId)
            .ToList();

        _logger.LogInformation("Generated {Count} ML recommendations for user {UserId}", recommendations.Count, userId);
        return recommendations;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error generating ML recommendations for user {UserId}", userId);
        return new List<int>();
    }
}
```

Home recommendations (linija 28-89)

```
public async Task<List<int>> GetMLCartRecommendationsAsync(int toolId, int count = 3)
{
    try
    {
        await EnsureModelLoadedAsync();

        if (_predictionEngine == null)
        {
            _logger.LogWarning("ML model not available for cart predictions");
            return new List<int>();
        }

        using var scope = _serviceProvider.CreateScope();
        var context = scope.ServiceProvider.GetRequiredService<ApplicationDbContext>();

        // Find users who rented this tool
        var userIds = await context.Set<Order>()
            .Where(o => o.OrderItems.Any(oi => oi.ToolId == toolId))
            .Select(o => o.UserId)
            .Distinct()
            .ToListAsync();

        if (!userIds.Any())
            return new List<int>();

        // Get available tools (excluding the current one)
        var availableTools = await context.Set<Tool>()
            .Where(t => t.IsAvailable && t.Quantity > 0 && t.Id != toolId)
            .Select(t => t.Id)
            .ToListAsync();

        // Predict scores for each available tool across all users who rented the target tool
        var toolScores = new Dictionary<int, float>();

        foreach (var otherToolId in availableTools)
        {
            float totalScore = 0;
            foreach (var userId in userIds)
            {
                var input = new ToolRating
                {
                    UserId = userId,
                    ToolId = otherToolId
                };

                var prediction = _predictionEngine.Predict(input);
                totalScore += prediction.Score;
            }

            toolScores[otherToolId] = totalScore / userIds.Count;
        }

        // Return top N tools by average predicted score
        var recommendations = toolScores
            .OrderByDescending(kvp => kvp.Value)
            .Take(count)
            .Select(kvp => kvp.Key)
            .ToList();

        _logger.LogInformation("Generated {Count} ML cart recommendations for tool {ToolId}", recommendations.Count, toolId);
        return recommendations;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error generating ML cart recommendations for tool {ToolId}", toolId);
        return new List<int>();
    }
```

Cart recommendations (linija 91-159)

3. Frontend

Putanja: MosPosudit.UI\shared\lib\services\recommendation_service.dart

Screenshot od linije 6-45

```dart
class RecommendationService {
  final ApiClient _api;

  RecommendationService({ApiClient? apiClient}) : _api = apiClient ?? ApiClient();

  Future<List<ToolModel>> getHomeRecommendations({int count = 6}) async{
    try {
      final res = await _api.get('/Recommendation/home', query: {'count': count});

      if (res.statusCode == 200) {
        final decoded = jsonDecode(res.body);
        if (decoded is List) {
          return decoded.map((e) => ToolModel.fromJson(e)).toList();
        }
      }

      throw Exception('Failed to fetch home recommendations: ${res.statusCode}');
    } catch (e) {
      return [];
    }
  }

  Future<List<ToolModel>> getCartRecommendations(int toolId, {int count = 3}) async {
    try {
      final res = await _api.get('/Recommendation/cart/$toolId', query: {'count': count});

      if (res.statusCode == 200) {
        final decoded = jsonDecode(res.body);
        if (decoded is List) {
          return decoded.map((e) => ToolModel.fromJson(e)).toList();
        }
      }

      throw Exception('Failed to fetch cart recommendations: ${res.statusCode}');
    } catch (e) {
      return [];
    }
  }

}
```

4. ZAKLJUČAK

Implementirani sistem preporuka u MošPosudit aplikaciji predstavlja robustan i moderan sistem koji kombinuje najbolje od dva svijeta: Machine Learning personalizaciju i pouzdane Rule-Based algoritme.

Ključne karakteristike:

1. Dva algoritma:

- Machine Learning (Matrix Factorization)
- Rule-Based (sa konfigurablinim parametrima)

2. Tri režima rada:

- MachineLearning Mode
- RuleBased Mode
- Hybrid Mode (preporučeno)

3. Dvije vrste preporuka:

- Home Recommendations (6 alata)
- Cart Recommendations (3 alata)

4. Konfigurabilan sistem:

- Težinski faktori se mogu podešavati
- Engine selection u realnom vremenu
- Automatsko treniranje ML modela

5. Pouzdanost:

- Multi-level fallback mehanizam
- Uvijek vraća preporuke ako postoje alati