Specifikacija Projekta: Knjiga Recepata

1. Uvod

Ovaj dokument predstavlja specifikaciju projekta u okviru predmeta Verifikacija i validacija softvera, čiji je cilj razvoj srednje kompleksnog softverskog rješenja u obliku knjige recepata na kojem će se moći vršiti određene analize i verifikacije. Knjiga recepata će omogućiti korisnicima pregled, pretragu, dodavanje i upravljanje različitim receptima putem konzolnog interfejsa.

2. Ciljevi projekta

- Razvoj funkcionalne knjige recepata koja omogućava korisnicima interakciju putem konzole.
- Implementacija kompleksnih algoritama za optimizaciju pretrage, sortiranje i druge operacije unutar aplikacije.
- Primjena principa objektno orijentisanog programiranja za osiguranje kvalitetnog dizajna softvera.
- Efikasno upravljanje izuzecima kako bi se osigurala stabilnost aplikacije.
- Korištenje modernih alata za upravljanje verzijama i saradnja unutar tima.

3. Tim i raspodjela zadataka

Projekt će realizovati tim od pet članova, pri čemu svaki član preuzima odgovornost za određenu funkcionalnost:

Član Tima	**Funkcionalnost**	**Opis**
Mahir	Upravljanje receptima	Dodavanje, brisanje i ažuriranje recepata
Umihana	Pretraga i filtriranje	Implementacija napredne pretrage i filtriranja
Nedim	Sortiranje recepata	Sortiranje recepata po različitim kriterijumima
Abdullah	Sigurnost i autentifikacija	Implementacija sigurnosnih mehanizama
Rijad	Formatiranje podataka	Omogućavanje upravljanja receptima u različitim formatima

4. Zahtjevi projekta

4.1 Funkcionalni zahtjevi

1. **Upravljanje receptima**
    - Dodavanje novih recepata.
    - Brisanje postojećih recepata.
    - Ažuriranje informacija o receptima.
2. **Pretraga i filtriranje**
    - Pretraga recepata po nazivu, sastojcima ili kategoriji.
    - Filtriranje recepata prema različitim kriterijima (npr. vrijeme pripreme, težina recepta).
3. **Sortiranje recepata**
    - Sortiranje recepata po abecednom redu, datumu dodavanja, popularnosti ili korisnički definisanim zahtjevima.
4. **Sigurnost i autentifikacija**
    - Implementacija korisničkih računa.
    - Sigurno upravljanje lozinkama (npr. hash lozinke)
5. **Formatiranje podataka**
    - Pretvaranje recepata u formate kao što su JSON ili XML i prosljeđivanje istih na izlaz..
    - Korištenje podataka iz prethodno sačuvanih fajlova.

4.2 Nefunkcionalni zahtjevi

- **Korisnički interfejs**: Bazični konzolni interfejs.
- **Performanse**: Efikasno rukovanje velikim brojem recepata bez značajnog usporavanja aplikacije.
- **Upotrebljivost**: Jednostavan i intuitivan konzolni korisnički interfejs koji omogućava lako navigiranje kroz samu aplikaciju
- **Skalabilnost**: Aplikacija treba da bude dizajnirana tako da omogući lako proširivanje funkcionalnosti

5. Tehnički zahtjevi

5.1 Arhitektura

- **Objektno orijentisani dizajn**: Korištenje interfejsa, apstraktnih klasa i nasljeđivanja za strukturiranje koda.

5.2 Klase i Atributi

Projekat će koristiti najmanje pet ključnih klasa, svaka sa svojim atributima i metodama, kako bi zadovoljio zahtjeve prethodno izloženih funkcionalnosti. Predložene klase su:

1. **Recipe**
    - **Atributi:**
        - `int id` – Jedinstveni identifikator recepta.
        - `String naziv` – Naziv recepta.
        - `List<String> sastojci` – Lista sastojaka.
        - `String kategorija` – Kategorija recepta (npr. doručak, ručak).
        - `int vrijemePripreme` – Vrijeme pripreme u minutama.
        - `String upute` – Detaljne upute za pripremu.
        - `int popularnost` – Broj puta koliko je recept pregledan ili ocijenjen.
    - **Metode:**
        - Getteri i setteri za sve atribute.
        - `toString()` – Prikaz informacija o receptu.
2. **RecipeManager**
    - **Atributi:**
        - `List<Recipe> recepti` – Lista svih recepata.
    - **Metode:**
        - `void dodajRecept(Recipe recept)` – Dodavanje novog recepta.
        - `void obrisiRecept(int id)` – Brisanje recepta po ID-u.
        - `void azurirajRecept(int id, Recipe noviPodaci)` – Ažuriranje recepta.
        - `Recipe getRecept(int id)` – Dohvatanje recepta po ID-u.
        - `List<Recipe> getRecepti()` - Dohvatanje svih recepata

3. **SearchService**
    - **Atributi:**
        - Nema specifičnih atributa.
    - **Metode:**
        - `List<Recipe> pretraziPoNazivu(String naziv)` – Pretraga po nazivu.
        - `List<Recipe> pretraziPoSastojcima(List<String> sastojci)` – Pretraga po sastojcima.
        - `List<Recipe> filtrirajPoKategoriji(String kategorija)` – Filtriranje po kategoriji.
        - `List<Recipe> filtrirajPoVremenuPripreme(int maxVrijeme)` – Filtriranje po vremenu pripreme.
        - BinarnaPretragaPoNazivu(List<Recipe> recepti, String naziv) - Interna metoda koja koristi binarnu pretragu za pronalaženje svih recepata koji se poklapaju sa zadanim nazivom
4. **SortService**
    - **Atributi:**
        - Nema specifičnih atributa.
    - **Metode:**
        - `List<Recipe> sortirajPoNazivu(List<Recipe> recepti)` – Sortiranje po nazivu.
        - `List<Recipe> sortirajPoPopularnosti(List<Recipe> recepti)` – Sortiranje po popularnosti.
        - QuickSort(List<Recipe> recepti, int low, int high) - Privatna metoda koja implementira algoritam brze pretrage (QuickSort) za sortiranje liste recepata.
        - Partition(List<Recipe> recepti, int low, int high) - Pomoćna metoda za algoritam QuickSort.
        - Swap(List<Recipe> recepti, int i, int j) - Pomoćna metoda koja zamjenjuje dva recepta na zadatim indeksima `i` i `j` unutar liste.
5. **AuthenticationService**
    - **Atributi:**
        - `Map<String, String> korisnici` – Rječnik(mapa) korisničkih imena i hashiranih lozinki.
    - **Metode:**
        - `boolean regustrujKorisnika(String korisnickoIme, String lozinka)` – Registracija novog korisnika.
        - `boolean autentifikujKorisnika(String korisnickoIme, String lozinka)` – Autentifikacija korisnika.
        - `void hashLozinka(String lozinka)` – Hashiranje lozinke (npr. SHA-256).
6. **DataExportImportService**

- **Atributi:**
  - Nema specifičnih atributa.
- **Metode:**
  - `void eksportujRecepte(String format, List<Recipe> recepti)` – Izvoz recepata u odabrani format (JSON/XML).
  - `List<Recipe> importujRecepte(String format, String putanja)` – Uvoz recepata iz datoteke.
  - void ExportToJson(List<String> recepti) - Privatna metoda koja izvozi recepte u JSON format
  - void ExportToXml(List<String> recepti) - Privatna metoda koja izvozi recepte u XML format
  - List<Recipe> ImportToJson(String putanja) - Privatna metoda koja uvozi recepte iz JSON datoteke sa zadane putanje
  - List<Recipe> ImportToXml(String putanja) - Privatna metoda koja uvozi recepte iz XML datoteke sa zadate putanje

5.3 Algoritmi

Svaka funkcionalnost će implementirati barem jedan kompleksan algoritam:

- **Pretraga i filtriranje**: Implementacija binarne pretrage ili hash tabele za efikasnu pretragu.
- **Sortiranje recepata**: Implementacija brzog sort algoritma (Quick Sort) ili merge sort.
- **Sigurnost i autentifikacija**: Korištenje kriptografskih algoritama za hashiranje lozinki (npr. SHA-256).

5.4 Upravljanje Izuzecima

- Definisanje prilagođenih izuzetaka za specifične situacije (npr. `RecipeNotFoundException`).
- Korištenje try-catch blokova za dohvatanje i rukovanje izuzecima na odgovarajući način.

6. Radni plan

6.2 Alati

Projekat će koristiti sljedeće alate kako bi se osigurala efikasna implementacija, kontrola verzija i testiranje:

- **Razvojno okruženje**:
  - **Visual Studio**
    - **Opis:** Integrisano razvojno okruženje (IDE) za programski jezik C#. Omogućava napredno uređivanje koda, debuggovanje, i upravljanje projektima.
- **Kontrola Verzija**:
  - **Azure DevOps Platforma**
    - **Opis:** Korištenje Azure DevOps za upravljanje Git repozitorijem. Omogućava timsku saradnju, praćenje promjena u kodu, upravljanje granama (branching), i kreiranje pull zahtjeva (pull requests) za integraciju promjena.

- **Testiranje**:
  - **MSTest**
    - **Opis:** Microsoftov testni framework za C#. Koristiće se za pisanje i izvođenje unit testova kako bi se osigurala ispravnost pojedinačnih komponenti aplikacije.
- **Dokumentacija i Praćenje Zadataka**:
  - **Azure Boards**
    - **Opis:** Alat za praćenje zadataka i bugova, omogućavajući efikasno upravljanje projektom i transparentnost unutar tima.

Korištenjem ovih alata, tim će moći da razvija, testira i održava aplikaciju na struktuiran i organizovan način, osiguravajući visoku kvalitetu koda i efikasnu saradnju među članovima tima.

7. Zaključak

Završeni softver će biti konzolna aplikacija koja pruža korisnicima jednostavan način za upravljanje receptima, uz osiguranu sigurnost i efikasnost. Kako bi se osigurala visoka kvaliteta softvera, aplikacija će biti temeljito testirana i provjerena kroz različite faze testiranja. Korištenjem MSTest frameworka, tim će razviti opsežan skup unit testova za sve ključne komponente sistema, čime će se garantovati ispravnost i pouzdanost svakog modula. Sve navedene mjere će omogućiti da finalni softver ne samo da zadovoljava osnovne zahtjeve projekta, već i da pruža korisnicima pouzdano i efikasno rješenje za upravljanje receptima.