**REZERVASYON SINIFLANDIRMA SİSTEMİ PROJESİ**

**1.Giriş**

Bu proje sayesinde projeyi veri yapıları dersinde öğrendiğimiz ağaç (tree) ve bağlı liste (linked list) yapılarını kullanarak daha önce edinmiş olduğumuz teorik bilgileri pratiğe dönüştürdük ve istenenlere çözüm bulabilmeyi amaçladık. Projemizde farklı kullanıcıların farklı zamanda ve yerlerde yapmış oldukları rezervasyonların kayıtlarını içeren bir rezervasyon verisi üzerinde ağaç (tree) ve bağlı liste (linked list) yapıları kullanılarak temel işlevleri yerine getiren bir uygulama hazırladık.

**2.Temel Bilgiler**

Program C programlama dilinde geliştirilmiş olup, tümleşik geliştirme ortamı olarak “CodeBlocks IDE” kullanılmıştır. Gerekli olan #include <time.h> kütüphanesi eklenmiştir.

**3.Tasarım**

Rezervasyon sınıflandırma sistemi projesinin gerçekleştirilme aşamaları aşağıda sıralanmıştır.

**3.1 Problemin Anlaşılması**

Bir firma farklı alanlarda hizmet veren işletmeler için müşterilerin (hizmet alan) rezervasyon işlemlerini, internet üzerinden yapabilmelerine imkan sağlayan bir sistem tasarlamaya çalıştık. Bu firma rezervasyon bilgilerini içeren sıralı verilerini, işletmelerin hizmet alanlarına göre sınıflandırmak ve bu sayede verilerle ilgili istenen sorgu ve analizlerin daha etkin bir şekilde yapılabileceği bir model geliştirdik. Firmanın bu istekleri doğrultusunda bir program yaptık. Rezervasyon kayıtları “rezervasyon.txt” dosyası içinde tutulmaktadır.

**3.2 Modeli Çıkarma**

Tasarlanan sistemde “rezervasyon.txt” dosyasından çekilen kayıt bilgileriyle meydana getirilecek ağaç yapısı 3 aşamada oluşturulmalıdır:

1-İlk olarak ağacın üst kısmında, “rezervasyon” kök düğümünden başlayarak aşağı doğru kategoriler ve alt kategoriler için kayıtlardaki hiyerarşik sıraya göre alt düğümler oluşturulmalıdır. Her kategori düğüm yapısı aşağıdaki bilgileri tutmalıdır:

Kategori ismi: Kategorinin “rezervasyon.txt” dosyasındaki ismi - Alt kategori sayısı: Kendinden hemen sonra gelen çocuk düğümlerin sayısını gösterir - Kullanıcı sayısı: Kategorinin altında bulunan toplam kullanıcı sayısını gösterir - Rezervasyon sayısı: Kategorinin altında bulunan toplam rezervasyon sayısını gösterir - Kategori yolu: Kategori düğümünün rezervasyon.txt dosyasındaki gösterimi - Derinlik: Kategorinin ağacın kök düğümünden itibaren kaçıncı dereceden çocuk düğümü olduğunu gösterir.

2) İkinci aşamada ağacın altında bulunan kategori düğümlerinden sonra rezervasyon işlemini yapan müşteri bilgilerinin yer aldığı kullanıcı düğümleri Max-Heap Algoritmasına göre oluşturulacak ayrı bir ağaca eklenmelidir. Bu ağacın kök düğümü ise ilgili kategorinin bir çocuk düğümü olacak şekilde bağlanmalıdır. Her bir kullanıcı ağaçlarını oluştururken kullanıcıların o kategorideki toplam rezervasyon sayıları dikkate alınmalıdır. Kullanıcı düğümleri aşağıdaki bilgileri içermelidir: - Kullanıcı id : “rezervasyon.txt” dosyasından çekilen “kullanıcı id” bilgisi - Kategori ismi: “rezervasyon.txt” dosyasından çekilen en alt kategori bilgisi - Rezervasyon sayısı: Bir kulanıcının o kategoriye ait toplam rezervasyon sayısıdır.

3) Modelin oluşturulmasındaki bu son aşamada her bir kullanıcı düğümüne kullanıcıların o kategorideki rezervasyonları, bağlı liste yapısı kullanılarak eklenmelidir. Bağlı listelerdeki her bir düğüm aşağıdaki bilgileri içermelidir: - Yer id: “rezervasyon.txt” dosyasından çekilen “yer id” bilgisi - Rezervasyon zamanı: “rezervasyon.txt” dosyasından çekilen “zaman” bilgisi - Enlem: “rezervasyon.txt” dosyasından çekilen “enlem” bilgisi - Boylam: “rezervasyon.txt” dosyasından çekilen “boylam” bilgisi - Şehir: “rezervasyon.txt” dosyasından çekilen “şehir” bilgisidir.

**4.Kullandığımız Fonksiyonlar:**

* **categories\* categoryDelete(categories \*c,char category\_name[])**

/\*- Kategori silme: Kategori silme işlemi sırasında silinecek kategori düğümüne doğrudan bağlı kullanıcı düğümleri varsa onlar da kategoriyle beraber silinecektir. Ayrıca silinecek kategorinin altında bulunan kategoriler silinmemeli bunun yerine silinecek kategorinin bir üst düğümüne bağlanmalıdır. \*/

* **categories\* categoryAdd(map \*m,categories \*c, char category\_name[],char \*ust\_category\_name)**
* **users\* userDelete(users \*u,char user\_id[])**

// 1. Durumda belli bir kategorinin altında yer alan tüm kullanıcılar silinmelidir.

// 2. durumda ise belli bir kategori altındaki belli bir kullanıcı bulunarak silinmelidir.

// 3. durumda ise belli bir kullanıcının tüm kategorilerdeki kullanıcı düğümleri silinmelidir.

* **reservation\* reservationDelete(reservation \*r,int reservation\_id)**
* **reservation\* reservationAdd(reservation \*r , char location\_id[],char reservation\_time[],char latitude[],char longitude[],char city[])**

**6.ÖRNEK KODLAR:**

reservation\* reservationDelete(reservation \*r,int reservation\_id){

reservation \*iter=r;

while(iter->next != NULL && (iter -> next -> reservation\_id != reservation\_id))

iter = iter -> next;

if(iter -> next -> reservation\_id == reservation\_id){

reservation \*temp = iter->next;

iter -> next = temp -> next;

temp -> next -> prev = iter;

while(temp != NULL)

temp -> reservation\_id -= 1;

free(temp);

printf("Reservation was delete!\n");

return r;

}else{

printf("Reservation not found..!\n");

return r;

}

Örnegin, bu kodda rezervasyon silme işleminin nasıl gerçekleştiği yani ağaç üzerinde silinecek rezervasyona nasıl ulaşıldığı gösterilmiştir.

--/\* silme işini yapan esas fonksiyonumuz \*/--

struct node\* deleteNode(struct node\* root, int key)

{

    // kök düğüm NULL ise root'u return ediyor

    if (root == NULL) return root;

    // Silinmek istenen elemanı öncelikle bulmamız gerekiyor.

    // Bu yüzden eğer silmek istediğimiz düğüm root'dan küçükse sol alt ağaca gidiyor

    if (key < root->key)

        root->left = deleteNode(root->left, key);

    // Eğer silmek istediğimiz düğüm root'dan büyükse sağ alt ağaca gidiyor

    else if (key > root->key)

        root->right = deleteNode(root->right, key);

     else    {

        // Eğer düğümün solda çocuğu yoksa...

        if (root->left == NULL)

        {

            struct node \*temp = root->right;

            free(root);

            return temp;

        }

        // eğer düğümün sağda çocuğu yoksa...

        else if (root->right == NULL)

        {

            struct node \*temp = root->left;

            free(root);            return temp;       }

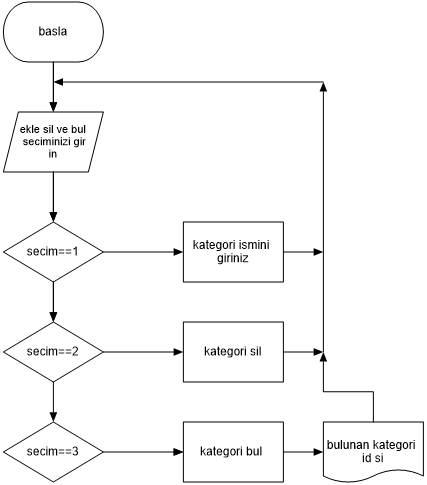
**6.Düğüm Ekleme, Silme ve Kategori Mantığı:**

Projede dosyadan satır satır veri okuma işlemi yapılır , veri okuma işleminde kullanıcı id , yer id , zaman ,enlem , boylam , şehir , kategori : alt kategori kavramları virgüller yardımıyla ayrılır kategorilerden alt kategori tespiti ise “:” yardımı ile yapılır Daha sonra okunan satırın işlemleri yapılır yani kategorisi sınıflandırılır, ağaç yapısına eklenir , okunan satıra ait , kategori düğümü oluşturulur birinci satırın işlemleri yaptırılıp ikinci satıra geçildiğinde yeniden dosyadan veri okunur bu sefer ikinci satır okunur ve karşılaştırmalar yapılır eğer diğer satırdaki kullanıcı id ile aynı id ye sahip ise yeni kullanıcı oluşturulmaz aynı karşılaştırma kategori ve alt kategoriler içinde yapılır ve kategori veya alt kategori düğümleri bu karşılaştırmalara göre oluşturulur. Program çalıştığında konsolda kullanıcıya yapmak istediği işlem sorulur kullanıcının seçtiği işleme göre isterler yerine getirilir örneğin kullanıcı kategori işlemlerini seçti sonrasında da kategori arama işlemini seçti bu durumda Kategorinin ismi ile ağaç üzerinde arama yapılarak kategori bilgileri ekrana basılır sonrasında ise kullanıcıya bu kategorinin altına yeni bir kategori ekleme ya da mevcut kategoriyi silme işlemleri için seçenek sunulur.

**7.Sonuçlar**

Verilen proje amacına ulaşmıştır. C dilinde dosya sistemleri grubumuzca öğrenilmiştir. Çalışmamız boyunca geçen sürede bir probleme sonuç üretme adına deneyim kazanılmıştır.

**8.Kategori Algoritmasi:**

****