GYTE Bilgisayar Mühendisliği Bölümü

2013-2014 CSE 102 Dersi

Öğrenci Otomasyon Sistemi Projesi

Teslim Raporu

Adı/Soyadı : Emrah YILDIRIM

No : 111044056

**Programın Input-Output ve Dosyaları**

**1.Başlangıç Dosyaları**

Eğer ilk kez çalıştırılıyorsa Students.csv, Instructors.csv, Users.csv, Courses.csv.   
İlk çalıştırılmayla birlikte tüm .csv dosyalarının .bin dosyaları oluşturulmuştur.

**Dosyaların İçeriği**

**Students.csv:** TCNo, ad, soyad, öğrenciNo, baslangıçYılı

**Instructors.csv:** TCNo, ad, soyad   
**Users.csv:** ID, kullanıcıTipi, kullanıcıAdı, şifre  
**Courses.csv:** dersID, ad, eğitimciID, önşartDersID

Her öğrenci, eğitimci ve ders için ayrı binary dosyaları olacaktır(userID.bin).  
Bazı bilgiler 2 dosyada birden olabilir(öğrenci notu >>öğrenci dosyası ve ders dosyası<<). Bu listeleme işlemleri sırasında kolaylık sağlayacaktır. Örneğin dersi alan öğrencileri sıralamak için oluşturulmuş tüm öğrenci dosyalaırnın içerisine girilip tek tek arama yapması ciddi kayıplar yaşatacaktır ancak ayrı bir ders dosyası içinde bu bilgilerin tutulması ulaşması ve düzenlemesi daha kolay olacaktır.  
  
**öğrenciTcNo.bin:** (TCNo, ad, soyad, öğrenciNo, baslangıçYılı,kayıtlıOlduğuDersSayısı)>>**STRUCT** (aldigiDersID, dersAdı, dersinNotu(not girişi yoksa -1)) >> **STRUCT..  
  
egitimciTcNo.bin:** (TCNo, ad, soyad) )>>**STRUCT** (verdiğiDersID, dersAdı)  
**dersID.bin:** (dersID, dersAdı, egitimciAdi, eğitimciSoyadi, eğitimciID, önşartDersleriArrayi açıkKapalı(1,0), ekle/çıkar(1,0), önşartSayısı, alanÖğrenciSayisi) >>**STRUCT**  
 (dersiAlanÖğrenciAdı, soyadi, ID, notu (giriş yoksa -1)) >>**STRUCT**

**Sistemin İşleyişi**

**Başlangıç**

1. Program çalışmaya kullanıcı adı ve şifre(terminal) alarak başlar.
2. Giriş bilgileri kontrolü yapılmadan önce .csv dosyalarının ve binary veritabanı dosyalarının olup olmadığı kontrol edilir .
3. Binary dosyalar varsa onlar üzerinden 6.adımdan devam edilir.
4. Yoksa .csv veri dosyalarının varlığı kontrol edilir, eğer .csv de yoksa dosya eksikliği hata kodu **(212)** verilerek program sonlandırılır.
5. .csv varsa, bu .csv dosyalarından bilgiler alınarak, bunlar doğrultusunda veritabanı dosyaları oluşturulur. (s**tudents.bin, instructors.bin, users.bin, courses.bin** )
6. Giriş kontrolü binary dosyalar üzerinden yapılır.Önce kullanıcı adı-şifre giriş formatının uygunluğunu, doğruysa daha sonra da sistemde eşleşme olup olmadığını kontrol eder. Her iki durumda da hata var ise giriş hata hata kodu **(0-FALSE)** verilerek tekrar giriş yapılması istenir.
7. Başarılı giriş işleminden sonra giriş yapan kullanıcının bilgileri kendi dosyasından okunarak, **curr\_user\_t** tipinde bir structure değişkenine kaydedilir, ileride gerekli olan bilgilerin temini buradan olacaktır. Adı, soyad, ID, kullanıcı tipi gibi bilgilerin hepsi bu değişkende tutulur.
8. Her kullanıcıya, kullanıcı tipine göre, yapabileceği işlemlerin listesi verilir.

>>Öğrenci Modu: Ders ekleme, ders çıkarma, dersleri listeleme,transkript oluşturma

>>Eğitimci Modu: Ders açma, dersi bitirme, ders modlarını değiştirme, not verme, not düzenleme, derslerini listeleme ve kaydetme (name\_surname\_openCourses.csv), verdiği derslere dair notları listeleme ve kaydetme(courseID\_grades.csv)

**Kullanıcı İşlemleri**

1. **Öğrenci Modu  
   Ders Ekleme:** Öğrenci, ekleme çıkarma modu açık, eklemeye müsait olan ve önşartı varsa önşartını tamamladığı dersleri alabilir. Bu işlemi seçtiğinde alabileceği dersler listelenecek ve ID sini girerek seçim yapacaktır. Seçim yapılınca, giriş doğruysa ve tamamlanmamış önşart dersi yoksa, derse kayıt yapılacaktır. Dersin bilgisi öğrencinin kendi dosyasına, öğrencinin bilgisi ise dersin dosyasına kaydedilecektir.  
    **Ders Çıkarma:** Öğrenci, ekleme çıkarma modu açık, aldığı derslerden herhangi birini çıkarabilir. Bu işlemi seçtiğinde çıkarabileceği dersler listelenir ve ID sini girerek seçim yapar. Seçilen ders o öğrencininse ve ekleme öıkarma modu, dersin bilgisi öğrencinin kendi dosyasından, öğrencinin bilgiside dersin dosyasından silinecektir. Bu işlemde silinen dersin kaydı tutulmaz.  
     
   **Dersleri Listeleme:** Öğrenci o an kayıtlı olduğu dersleri bu seçeneği seçerek listeleyebilir. İsterse de bunları kaydedebilir, kayıt dosyasının ismi “öğrenciNo\_courses.csv” şeklinde yazılacaktır.  
     
   **Transkript Oluşturma:** Öğrenci bu seçenekte, mezun aldığı dersleri, notlarını ve bunların ortalamasını bir dosyaya kaydedebilir. Bu seçenek seçildiğinde, öğrencinin mezun olduğu ers var mı diye kontrol edilir, varsa bunlar dosyaya yazılır yoksa hata mesajı verilerek işlemin yapılamadığı bilgisi kullanıcıya verilkir. Kayıt şekli “öğrenciNo\_transcript.txt”
2. **Eğitimci Modu  
   Ders Açma:** Bu seçenekte eğitimcinin verdiği dersler açık ve ya kapalı olma durumlarıyla birlikte listelenir. Eğitimci istediği dersi, dersin ID sinin girerek seçecek ve ders kayıtlanmaya açılmış olacaktır. Ancak dersin açılabilmesi için kendi dersi olması gerekir, eğer seçilen ders eğitimcinin değilse hata mesajı yazılarak uyarılır. Kendi dersi ise, dersin açıldığı bilgisi dersin kendi dosyasına ve tüm derslerin bulunduğu dosyaya yazılacaktır.  
     
   **Dersi Bitirme:** Bu seçenekte eğitimcinin verdiği ve açık olan dersler listelenir. İstenilen ders IDsi girilerek seçilecek ve ders bitirlimiş olacaktır ancak yine dersin eğitimciye ait olduğunun kontrolü yapılır ver eğitimciye aitse işlem gerçekleştirilebilir. Ayrıca bundan önce dersi alan her öğrencinin not atamasının yapılmış olması gerekir. Eğer notu girilmeyen öğrenci/öğrenciler varsa notların girilmesi istenir. Daha sonrasında ders kapatılabilir. Kapatma işlemi ile birlikte, dersin kendi dosyasına kapandığı bilgisi yazılır ve dersi alan öğrenci bilgileri bu dosyadan silinir. Aynı şekilde öğrencilerin dosyasından da bu dersin bilgileri silinir. Mezun olunan derslerin tutulduğu her öğrenci için ayrı bir studentID\_old.bin adında bir dosya oluşturulur ve dersin bilgileri buraya yazılır. Eğer bir ders 2. Kere alınmışsa en son alınan not geçerli olacağından ilk bilgiler silinir ve yenileri yazılır.  
     
   **Ders Modu Değiştirme:** Bu seçenekte eğitimcinin açık olan dersleri listelenir.Dersin kayıtlanmaya açık olup olmayacağını değiştirebilir. Dersin eğitimciye ait olup olmaması koşulu aranır.  
     
   **Not Verme:** Eğitimci bu seçeneğe gittiğinde, eğitimci tarafından verilen ve açık olan tüm dersler listelenir. Eğitimci ID girerek bir ders seçer, ders eğitimciye ait değilse ana menüye dönülür ve hata mesajı verilir. Aksi halde dersi alan öğrenciler listelenir. Eğitimci bir öğrenci ID si ve notu girer. Öğrenci ID si ve not giriş doğruysa notlandırma işlemi yapılır. Hatalı bir durum varsa, uyarı mesajı yazılır ve duruma göre ana menüye ya da komut alma işlemine dönülür. Notlar 1-100 arasında ve ondalıklı biçimde olacaktır.  
     
   **Not Düzenleme:**Eğitimci bu seçenekte, **not verme**  işleminde yapılan işlemleri yapar. Farklı olarak, notu değiştirebilmesi için önceden bir not girmiş olması koşulu aranır. Eğer girilen bir not yoksa ilk notu girmesi istenir. Not giriş yapılmışsa yeni notu girebilir.  
     
   **Açılmış Dersleri Listeleme:**  Eğitimci bu seçenekte kendi verdiği ve açık olan derslerin bir listesini görür ve isterse bunları kaydedebilir. Kayıt dosyası eğitimciAd\_Soyad\_openCourses.csv şeklinde olacaktır.  
     
   **Notları Listeleme:** Önce eğitimcinin verdiği açık olan dersler listelenir, daha sonrasında eğitimci bunlardan birini seçer, eğer seçilen ders kendininse, dersi alan öğrencilerin notları listelenir. Kaydedip istemediği sorulur kayıt yanıtı alınırsa dersID\_grades.csv dosyasına kayıt yapılır. Hatalı durumlarda hata mesajı verilir ve ana menüye dönülür.

**FONKSİYON PROTOTİPLERİ VE KULLANIM AMAÇLARI**

**extern int** checkUser**(const char \***name**, const char \***password**);  
 >>** Kullanıcı giriş bilgilerini kontrol eden fonksiyon. Bilgiler eşleşiyorsa TRUE, yanlış giriş varsa FALSE return eder.

**extern int** fileControls**(void);  
 >>** Dosyalarla ilgili kontrol fonksiyonlarının wrapper fonksiyonu, input parametre almaz, dosya yoksa FILE\_ERROR, işlem başarılıysa 0 return eder.

**extern int** checkAllFiles**();  
 >>**  checkBinaryFiles() ve checkCsvFiles() ı kullanarak her 2 dosya türü içinde kontrol yapar dosyalar türlerinden en az 1i mevcutsa TRUE hiç biri yoksa FILE\_ERROR hata kodu return eder.

**extern void** getAccessInfo**(curr\_user\_t \***currentUser**);  
 >>** Sistem kullanıcı adı ve şifreyi terminalden alarak başliyor ancak terminal girişi olmazsa ya da yanlış giriş olduysa tekrar kullanıcı adı şifre giriş yapmak için kullanılan fonksiyon curr\_user\_t \* structure ı alır ve bilgileri bu structure ın üyelerine kaydeder

**extern int** getCommand**(int** type**);  
 >>** Program içersinde komut almaya yarayan fonksiyon **int type** parametresi ile kullanıcı tipini alır kullanıcı tipleri makro olarak belirtilmiştir. (STUDENT – INSTRUCTOR) **int command** ı return eder (komut) yanlış komut girişinde doğru girilene kadar istemeye devam eder.

**extern int** checkBinaryFiles**();  
 >>** Bulunulan dizinde databese(.bin) dosyalarının varlığını kontrol eder hepsi varsa TRUE herhangi biri eksikse FALSE return eder.

**extern int** checkCsvFiles**();  
 >>** Bulunulan dizinde başlangıç csv dosyalarının varlığını kontrol eder, hepsi varsa TRUE herhangi biri eksikse FALSE return eder.

**extern char \***readStudents**(student\_t \***student**, FILE \***studentFile**);  
 >>** Verilen dosyadan öğrenci okuması yapan fonksiyon. Her seferinde 1 okuma yapar **student\_t \*student** structure pointer ı ve **\*studentFile** file pointer ı alır, dosya sonu olup olmadığını gösteren bir **char \*status** return eder.

**extern void** writeAllStudent**();  
 >>** readStudents() ı kullanarak Students.csv den tüm öğrencileri okur. Tüm bu bilgileri Students.bin dosyasının içerisine kaydeder ayrıca her öğrenci için öğrenci ID(TC) isminde binary dosyası oluşturur.

**extern char \***readInstructors**(instructors\_t \***instructor**, FILE \***instructorFile**);  
 >> V**erilen dosyadan eğitimci okuması yapan fonksiyon. Her seferinde 1 okuma yapar instructors\_t \*instructor structure pointer ı ve \***instructorFile** file pointer ı alır dosya sonu olup olmadığını gösteren bir **char \*status** return eder.

**extern void** writeAllInstructors**();  
 >>** readInstructors() ı kullanarak Instructors.csv den tüm eğitimcileri okur, tüm bu bilgileri Instructors.bin dosyasının içerisine kaydeder ayrıca her eğitimci için eğitimci TCNo isminde binary dosyası oluşturur.

**extern char \***readCourses**(courses\_t \***course**, FILE \***courseFile**);   
 >>** Verilen dosyadan ders okuması yapan fonksiyon, her seferinde 1 okuma yapar **courses\_t \*course** structure pointer ı ve **\*courseFile** file pointer ı alır dosya sonu olup olmadığını gösteren bir **char \*status** return eder.

**extern void** writeAllCourses**();  
 >>** readCourses() fonskiyonunu kullanarak Courses.csv den tüm dersleri okur tüm bu bilgileri Courses.bin dosyasının içerisine kaydeder. Ayrıca her ders için dersin ID si isminde bir binary dosyası oluşturur.

**extern char \***readUsers**(users\_t \***user**, FILE \***userFile**);  
 >>**  Verilen dosyadan kullanıcı okuması yapan fonksiyon. Her seferinde 1 okuma yapar **users\_t \*user** structure pointer ı ve **userFile** file pointer ı alır dosya sonu olup olmadığını gösteren bir **char \*status** return eder.

**extern void** writeAllUsers**();  
 >>** readUsers() fonskiyonunu kullanarak **Users.csv** den tüm kullanıcıları okur tüm bu bilgileri **Users.bin** dosyasının içerisine kaydeder

**extern int** checkAvailableOpen**(const char \***courseID**);  
 >>**  Girilen dersin var olup olmadığını, varsa da açılmış ve ders eklemeye müsait olup olmadığının kontrolunu yapar. Input parametre olarak dersin ID sini(**courseID**) alır ders yoksa **USER\_COURSE\_ERR** var ama açık ya da ekleme süresi içinde değilse **FALSE** return eder.

**extern int** preqControl**(const char \***studentID**, const char \***courseID**);  
 >>** Alınacak dersin önşart durumunu kontrol eder, **studentID** ve **courseID** input olarak alır, önşart saglanıyorsa **TRUE** yoksa **FALSE** return eder.

**extern int** enrolCourse**(const char \***courseID**, const char \***studentID**);  
 >>** Ders ekleme çıkarma işlemini yapan fonksiyon. **char \*courseID** ve **char \*studentID** paramatre olarak alır hem öğrencinin hem de dersin dosyasına karşılıklı bilgileri kopyalar bilgi girişi yanlışsa dosya açılamaz ve **USER\_COURSE\_ERR** return edilir aynı ders 2 kere alınmak isteniyorsa da hata verilir.

**extern int** listCourses4Add**();  
 >>** ders eklenmeye müsait olan tüm dersleri listelenir eğer açık olan ders yoksa ekrana hata mesajı yazılır.

**extern int** listAndSaveEnrolledCourses**(const char \***studentID**, int** saveMode**);  
 >>** Öğrencinin aldığı dersleri listeleyen ve istenirse kaydeden fonksiyon.  
  **char \*studentID**: Öğrenci ID'si alır(TC kimlik Numarası) **int saveMode** : Kayıt işleminin sorulup sorulmayacagini kontrol eder.

**extern int** disenrolCourse**(const char \***courseID**, const char \***studentID**, int** byWho**);  
 >>** Ders kaydı silme islemini yapar **char \*courseID, char \*studentID, ve byWho** integer paramatresi alır.  
 int byWho : Silme işlemini kimin yapacağıdır, eger oğrenci yapacaksa o ogrenci ile alakalı işlem yapılır, dersin ve öğrencinin dosyasından tüm bilgiler silinir. Eger işlemi egitimci yapiyorsa ogrenci durumunda yapilanlara ek olarak ogrenciIDsi ile eski dersler dosyası oluşturulur ve bu ders kaydedilir.

**extern int** createTranscript**(const char \***userID**);  
 >>** Ögrenci transkripti olusturan fonksiyon input parametre olarak **userID**(studentID) alir. Ögrencinin kendi dosyasi ve mezun oldugu derslerin dosyasindan ders ve kisisel bilgileri alir

**extern int** listCourses4Instructor**(const char \***instructorID**, int** forWhat**, int** saveMode**);  
 >>** Eğitimcinin yapacağı işleme göre ekrana ders adlarını yazar.   
 Input Paramatreler: **char \*instructorID, int forWhat**: fonskiyonun yapacağı işi tanımlar, makro olarak yazılmıştır.  
 BEGIN: Ders açılacağını  
 FINAL: Dersin bitirileceğini   
 GRADE: Not verme işlemi yapılacağını  
 int saveMode: Kayıt sorusunun sorulup sorulmayacağını belirtir.  
  
  
**extern int** processCourse**(const char \***courseID**, int** stateFlag**, const char \***userID**);  
 >>** derslerin modlarını ayarlayan fonksiyon **char \*courseID** ve **int stateFlag**(yapılacak işlem) paramatrelerini alır.  
 stateFlag: yapılacak işlem   
 userID:kullanici kullanici dersin sahibi degilse USERCOURSE\_ERR return edilir. İşlemler makro ile belirtilmiştir   
 OPEN : Ders açma CLOSE: dersi bitirme   
 AVAILABLE: Ders ekleme çıkarma modu açık  
 NAVAILABLE: Ders ekleme çıkarma modu kapalı

**extern void** saveOpenCourses**(const char \***instructorID**, courses\_t \*courses, int** size**);  
 >>** Eğitimcinin açık olan derslerini kaydeder. **instructorID** ve **courses\_t** tipinde derslerin arrayini ve buyuklugunu alır. **egitimciAdi\_soyadi\_opencourses.csv** olarak dosya acar ve kayıt yapar.

**extern int** listStudents4Grade**(const char \***courseID**, const char \***currentUserID, **int** saveMode**);  
 >>** Not verme ve düzenleme için dersi alan öğrencileri listeler.   
 **courseID**, instructorID(**currentUserID**) ve int **saveMode** alır.  
 saveMode kayıt sorusunun sorulup sorulmayacagini belirtir (**YES ve NO**).

**extern double** getGrade**(void);  
 >>** Not verme ve duzenleme işlemi için girilecek notu alan fonksiyon 1-100 arası not verilebilir. 0 nota dahil değildir girilen notu double olarak return eder.

**extern void** saveGrades**(courses\_t \***course**, crs\_std\_t \***students**);  
 >>** Girilen notların kaydedilme işlemini yapar dersin structure pointerını ve ogrencilerin bulundugu arrayi alir.

**extern int** assignGrade**(const char \***studentID**, const char \*courseID,   
 const char \***currentUserID**, double** grade**);  
 >>** Not verme işlemini yapan fonksiyon öğrenci, eğitimci ve o anki kullanicinin IDlerini ve notu alir eger ogrenci yoksa, ders yoksa ya da o anki kullanicinin dersi degilse **USER\_COURSE\_ERR** return eder. islem basarili olursa **0** return edilir

**extern int** isGraded**(const char \***courseID**, const char \***studentID**,   
 const char \***currentUserID**, int** forWhat**);  
 >>** Dersin notlandirilip notlandırılmadığını kontrol eder ders, öğrenci, kullanıcı IDleri yanında işlemin ne için yapılacağını belirten bir integer paramatre(forWhat) alır. forWhat işlemin dersi alan tüm öğrenciler için mi, yoksa tek bir öğrenci için mi yapılacağını belirtir. Eger ders yoksa **USER\_COURSE\_ERR** return eder.  
 İşlem basarili olursa 0 return edilir.

**extern void** finalizeCourse**(const char \***courseID**, const char \***currentUserID**);  
 >>** Eğitimcinin dersi sonlandırmasını saglayan fonksiyon. Dersin ve o anki kullanicinin   
 IDsini alir dersi alan tüm ogrenciler icin disenrol islemini yapar ve dersi kapatir.  
 Dersin kapanması için tüm öğrencilerin not girişlerinin yapılmış olması gerekir, eksik varsa uyarı mesajı verilir.

**extern void** findCurrentUserInfo**(curr\_user\_t \***currentUser**);  
 >>** Giriş yapan kullanıcının tüm bilgilerine ulaşır, bunları curr\_user\_t tipindeki structure a kaydeder. Input parametre olarak curr\_user\_t tipinde structure pointer alır.

**extern void** printMenu4Students**(const char \***name**, const char \***surname**);**

**extern void** printMenu4Instructors**(const char \***name**, const char \***surname**);**

**extern void** getIDCommand**(char \***commandID**);  
 >>** String olarak alınacak komutları alır input olarak \*commandID karakter array i alır   
  
**extern int** studentChoice1**(curr\_user\_t** currentUser**);  
extern int** studentChoice2**(curr\_user\_t** currentUser**);**

**extern int** instructorChoice1**(curr\_user\_t** currentUser**);  
extern int** instructorChoice2**(curr\_user\_t** currentUser**);  
extern int** instructorChoice3**(curr\_user\_t** currentUser**);  
extern int** instructorChoice4**(curr\_user\_t** currentUser**);  
extern int** instructorChoice5**(curr\_user\_t** currentUser**);  
extern int** instructorChoice6**(curr\_user\_t** currentUser**);  
extern int** instructorChoice8**(curr\_user\_t** currentUser**);**

**STRUCTURES**

**typedef struct  
{  
 char** tc[12], name[NM\_SN\_SIZE], surname[NM\_SN\_SIZE], no[NM\_SN\_SIZE];  
 **int** startYear, courseCount;  
**} student\_t;  
  
 >>**

**typedef struct  
{  
 char** tc[TC\_SIZE], name[NM\_SN\_SIZE], surname[NM\_SN\_SIZE]; **}instructors\_t;**

**typedef struct**

**{  
 char** ID[ID\_SIZE];  
**}preq\_t;**

**typedef struct**

**{  
 char** ID[ID\_SIZE], name[NM\_SN\_SIZE];  
 **char** instructorName[NM\_SN\_SIZE], instructorSurname[NM\_SN\_SIZE], structorID[ID\_SIZE];  
 **preq\_t** preq[PREQ\_SIZE];  
 **int** isOpen, isAvailable, preqNum, stdCount;  
**}courses\_t;**

**typedef struct**

**{  
 char** ID[TC\_SIZE], type[TYPE\_SIZE];  
 **char** userName[NM\_SN\_SIZE], password[PASS\_SIZE];   
**}users\_t;**

**typedef struct  
{  
 char** ID[TC\_SIZE], type[TYPE\_SIZE]; **char** name[NM\_SN\_SIZE], surname[NM\_SN\_SIZE];  
 **char** userName[NM\_SN\_SIZE], password[PASS\_SIZE]; **}curr\_user\_t;  
typedef struct  
{  
 char** name[NM\_SN\_SIZE], surname[NM\_SN\_SIZE];  
 **char** ID[ID\_SIZE];  
 **double** grade; **}crs\_std\_t;**

**typedef struct  
{  
 char** ID[ID\_SIZE], name[NM\_SN\_SIZE];  
 **double** grade; **}std\_crs\_t;**

**BEKLENEN VE ELDE EDİLEN SONUÇLAR**

Analiz raporunda, yazılan yapılması planlanan işlemlerin tamamına yakını gerçekleştirilmiştir. Artı olarak, transkript oluşturma, not düzenleme gibi işlemler eklenmiştir. Structure ve fonksiyonlarda değişikliğe gidilerek sistemin daha efektif ve güvenli şekilde çalışması amaçlanmıştır. Her işlemde kontrol sayıları artırılarak sistemin güven seviye çıtası yükseltilmeye çalışılmıştır.

Kod tekrarından kaçınmak adına benzer işlemleri yapacak fonksiyonları tekrar tekrar yazılmamış, işlemi belirten bir değişken daha parametre olarak fonskiyona geçirilerek fonksiyonun kullanım alanı genişletilmiştir.

Her kullanıcı için ayrı birer dosya oluşturulmuştur. Bu şekilde, bilgiler kategorize edilerek, düzenleme kolaylığı ve gerektiğinde bilginin ulaşılabilirliği artırılmaya çalışılmıştır. Bu şekilde aynı bilgiler farklı yerlerde birden fazla kez tutulduğundan bir miktar veri alanı kaybı yaşanmaktadır, ancak bu işlemin, sistemin çalışma performansı üzerine etkisi düşünüldüğünde bu kaybın ihmal edilebilir düzeyde olduğu görülecektir.

Görece olarak fazla hafıza biriminin kullanılacağı kısımlarda, hafızada yer alma (memory allocating) işlemi yapılmıştır. Yapılabildiğince fazla hafıza kullanımının önüne geçilmeye çalışılmıştır.

Sonuç olarak tek kişillk kısımda yapılması istenen tüm özellikler mevcuttur.