**O’zbekiston Respublikasi Oliy va O’rta maxsus ta’limVazirligi**

**Mirzo Ulug’bek nomidagi O’zbekiston Milliy Universiteti**

Mexanika-matematika fakulteti

Programmalash va tarmoq texnologiyalari kafedrasi

# BITIRUV MALAKAVIY ISHI

**Mavzu : Fanlar bo`yicha online test o`tkazish saytini yaratish (ASP.net MVC texnologiya)**

Bajardi: “Amaliy matematika va informatika ” ta’lim yo’nalishi bitiruvchisi

Allaberganova N.M_____

Ilmiy rahbar:f.-m.f.n., dots.

Ishmuhammedov A.X_____

Bitiruv malakaviy ishi kafedradan dastlabki himoyadan o’tdi.

_____sonli bayonnomasi “__”_____2014-yil.

Toshkent-2014

# Mundarija

# Kirish

Jamiyatni axborotlashtirish hozirgi zamonning davr talabiga aylanib bormoqda. Inson faoliyat olib borayotgan barcha sohalar axborotni qabul qilish, qayta ishlash va o'zlashtirish jarayonlari bilan uzviy bog'liq. Hozirgi kunda kompyuter va axborot texnologiyalari jadal sur'atda, yangilanib, o'zgarib, rivojlanishi bilan birga kundalik turmushimizning asosiga aylanib bormoqda [1].

Hozirgi davrda bizni o'rab turgan dunyodagi axborotlar oqimi juda ulkandir. Ular ulkan bo'lishiga qaramasdan yanada ko'payib bormoqda. Shuning uchun kattami - kichikmi tashkilotlar qanday bo'lishidan qat'iy nazar tashkilotdagi mumkin bo'lgan barcha ishlarni tashkil etishga imkon beradigan ma'lumotlarni samarali, qulay tarzda olib borish muammosi yuzaga kelmoqda. Oldingi davrda biror tashkilotning ma'lumotlari qog'ozlarda, turli papkalarda saqlangan bo'lsa, hozirgi davrda esa deyarli barcha tashkilotlar kompyuterlashtirilgan usullar - katta hajmdagi ma'lumotlarni samarali saqlashga yordam beruvchi berilganlar bazasidan foydalanadilar. Bugungi kunda tashkilot qanday bo'lishidan qat'iy nazar ular moliyaviy bo'ladimi, savdo tashkilotlari yoki sanoat bo'ladimi barchasining faoliyatini berilganlar bazasisiz tasavvur qilib bo'lmaydi. Agar berilganlar bazasi bo'lmaganida ular axborot oqimi ichida qolib ketar edilar.

O'zbekistonda axborotlashtirishni rivojlantirish, jamiyat taraqqiyotining barcha sohalarida zamonaviy axborot texnologiyalari, texnika va telekommunikatsiya vositalarini ommaviy ravishda joriy etish hamda ulardan foydalanish, fuqarolarning axborotga bo'lgan talab-ehtiyojlarini yanada to'liqroq qondirish , jahon axborot resurslaridan bahramand bo'lishni kengaytirish uchun qulay shart sharoitlarni yaratish aloqa va axborot kommunikatsiyasi sohasini rivojlantirish asosiy vazifalari hisoblanadi. Bulardan tashqari fuqarolarning erkin tarzda o'zlariga qulay vaqtda internet tarmog'i orqali axborot olishlari va ma'lumotlarni erkin tarzda almashishlarini ta'minlash ham rivojlanishning asosiy vazifalari deyish mumkin.

Oliy Majlisning IX sessiyasida "Kadrlash tayyorlash Milliy dasturi" da zamonaviy axborot texnologialarini o'z vaqtida ishlab chiqish va joriy etishni ta'minlash hamda ta'lim muassasalarining moddiy-texnik bazasini yangi o'quv axborotlari, avtomatlashtirilgan dasturlar, zamonaviy jihozlar, jumladan kompyuter va boshqa texnik vositalar bilan ta'minlash ko'zda tutilgan.

O`quvchi, abiturent, talaba yoshlarni bilim saviyasini oshirishda global internet tarmog`ida ishlovchi testlarning o`rni sezirarli darajada ommalashmoqda.

Masalan, og`zaki , yozma, fanning tabiiy yoki gumanitarligiga qarab semenar, amaliyot, laboratoriya mashg`ulotlari (reyting tizimiga asoslanib) joriy, oraliq va yakuniy nazoratlar bo`lishi mumkin. Ana shu jarayonni testlash jarayoni deb kiritdik.

Bundan tashqari nafaqat oliy va o`rta maxsus o`quv dargohlari vakillari, balki butun bir aholini tafakkurining rivojlanish koeffitsentini aniqlovchi test tizimlari ham talaygina. Ulardan biri IQ Test idir. IQ Test- “Intelligence Quotient”, ya`ni “Inson tafakkurining rivojlanish koeffitsentini aniqlash” degan ma`noni anglatadi.

Bitiruv malakaviy ishi mavzusi “Fanlar bo`yicha online test saytini yaratish (ASP.net va MVC texnologiya)”. Bu fanlar bo’yicha online test saytidan barcha o`smir, o`spirin, o`rta va katta yoshdagi insonlar test topshirishi mumkin. Har bir savolga yetarlicha vaqt ajratilgan. Sayt test topshiruvchining og`irligi yuqori bo`lgan savollarga ajratgan vaqtiga qarab ham xulosa chiqaradi. Bu dasturda test topshirish global, ya`ni internet to`ri orqali amalga oshiriladi.

Shunga o`xshash tashkilotlararo lokal hisoblash to`rlarida testlash tizimlari ham mavjud. Ular asosan tashkilot hodimlarining bilim darajasini tekshirib, uni nazorat qilishga qaratilgan. Shu asosida tashkilotdagi kadrlar malakasi nazorat ostida bo`ladi.

Oliy va o`rta maxsus o`quv yurtlarida ham shu kabi tizimlar mavjud. Ammo to`liq avtomatlashtirish ba`zi joylarda resurs yetishmasligi asosiy muammo bo`lsa,

ba`zi joylarda tizim administratorlari tomonidan yo`l qo`yiladigan qo`pol xatoliklar tufayli orqaga surilgan. Ushbu avtomatlashtirish ishlari to`liq yo`lga qo`yilsa, baholash haqqoniy amalga oshiriladi.

Ushbu malakaviy bitiruv ishim 3 ta bob, xulosa, foydalanilgan adabiyotlar va ilova qismlaridan iborat. Birinchi bobda asosan masalaning bitiruv malakaviy ishining maqsad va vazifalari yoritilgan bo`lib u bo’limdan iborat: bitiruv malakaviy ishining qo`yilishi, masalaning dolzarbligi, II bobda saytni yaratilish va yaratish davomida qo`llaniladigan texnologiyalar, III bobda dasturiy ta`minot haqida aytib o`tilgan.

# I BOB. Bitiruv malakaviy ishining maqsad va vazifalari

## 1.1. Bitiruv malakaviy ishining qo`yilishi

Ushbu bitiruv malakaviy ishining bajariliahidan ko‘zlangan maqsad ASP.net MVC texnologiyasi yordamida global tarmoqda ishlash imkoniga ega bo'lgan online-test saytini yaratishdan iboratdir.

Bunda bir qancha  texnologiyalar o'rganib chiqish talab qilinadi. Mazkur dastur uchun qo'yiladigan asosiy vazifalar quyidagilardan iborat: Online test tizimini yaratishda quyidagi bosqichlarni ko`rib chiqishimiz kerak:

- Berilganlar bazasini tashkil qilish;
- Tizimni boshqarishni tashkil etish;
- Tizim dizaynini ishlab chiqish;
- Foydalanuvchi va boshqaruvchi modullarini ishlab chiqish;

Web-texnologiya asosida yaratilgan berilganlar bazasini xavfsizligi ya'ni unihimoya qilish, foydalanuvchilarning ko'p sondaliklarini hisobga olib, ishlash jarayonini mukammallashtirish zarurligi ham asosiy vazifalardan biri hisoblanadi.

Bitiruv malakaviy ishining asosiy vazifasi foydalanuvchilarning bilimlarini sinash hamda  o'z bilimlarini mustahkamlashdan iborat.

Qo'yilgan masalani yechish uchun quyidagi dasturiy ta'minotlarlami o'rganib chiqish talab qilinadi:

- Visual Studio 2013
- SQL SERVER 2012 lokal

## 1.2. Masalaning dolzarbligi

O‘zbekiston Respublikasi Prezidentining 1992 yil 11 martdagi “Respublika oliy o‘quv yurtlarining kunduzgi bo‘limlariga qabul qilishni takomillashtirish to‘g‘risida”gi PF-361-sonli farmoni mamalakatimiz oliy ta`lim muassasalariga qobiliyatli yoshlarni tanlab olish sohasida ta`limdagi tub islohotlarni boshlab berdi. O‘zbekiston Respublikasi Prezidentining qarorlari va farmonlari, O‘zbekiston Respublikasi Vazirlar Mahkamasining qarorlari, Prezident Devoni topshiriqlariga muvofiq har yili quyidagi tadbirlarni amalga oshirishda test sinovlari o‘tkaziladi[2]:

1. Vazirlar Mahkamasining 2010 yil 18 iyundagi “Oliy ta`lim muassasalariga qabul qilish, talabalar o‘qishini ko‘chirish, qayta tiklash va o‘qishdan chetlashtirish tartibi to‘g‘risidagi nizomlarni tasdiqlash haqida”gi 118-sonli qaroriga asosan oliy ta`lim muassasalarining bakalavriatiga talabalarni qabul qilish uchun;

2. Vazirlar Mahkamasining 2009 yil 1 oktyabrdagi “Fuqarolarni muddatli harbiy va muqobil xizmatlarga chaqirish, safarbarlik chaqiruvi rezervidagi xizmatga qabul qilish bo‘yicha chaqiruv komissiyalari to‘g‘risidagi nizomni tasdiqlash haqida”gi 265-sonli qaroriga asosan chaqiriluvchilarning bilim saviyasini aniqlash uchun;

3. O‘zbekiston Respublikasi Prezidentining 2005 yil 31 oktyabrdagi “O‘zbekiston respublikasi qurolli kuchlari safida muddatli harbiy xizmatni o‘tayotgan harbiy xizmatchilarga beriladigan imtiyozlar tizimini yanada takomillashtirish chora-tadbirlari to‘g‘risida”gi 213-sonli qaroriga asosan muddatli harbiy xizmatni o‘tayotgan harbiy xizmatchilarga oliy ta`lim muassasalariga o‘qishga kirishi uchun tavsiyanomalar berish uchun;

4. Vazirlar Mahkamasining 2010 yil 7 iyundagi “Umumta`lim maktablarining 9-sinf bitiruvchilarini akademik litseylar va kasb-hunar kollejlarida o‘qitish bilan qamrab olishni ta`minlash bo‘yicha normativ-huquqiy bazani yanada takomillashtirish to‘g‘risida”gi 109-sonli qaroriga asosan umumta`lim

maktablarining 9-sinf bitiruvchilarini o‘rta maxsus, kasb-hunar ta`limi muassasalariga qabul qilish uchun;

5. Vazirlar Mahkamasining 2008 yil 7 avgustdagi “Ayrim fanlar chuqur o‘rganiladigan davlat ixtisoslashtirilgan umumta`lim muassasalari faoliyatini takomillashtirish to‘g‘risida”gi 173-sonli qaroriga asosan ayrim fanlar chuqur o‘rganiladigan maktab-internatlarining 5-sinfiga qabul qilish uchun;

6. Vazirlar Mahkamasining 2008 yil 13 oktyabrdagi “O‘zbekiston iqtidorli yoshlarini taqdirlash va moddiy rag‘batlantirish to‘g‘risida”gi 226-sonli qaroriga asosan bakalavriyat talabalaridan O‘zbekiston Respublikasi Prezidenti stipendiatlarini aniqlash uchun;

7. Vazirlar Mahkamasining 2008 yil 13 oktyabrdagi “O‘zbekiston iqtidorli yoshlarini taqdirlash va moddiy rag‘batlantirish to‘g‘risida”gi 226-sonli qaroriga asosan magistratura talabalaridan O‘zbekiston Respublikasi Prezidenti stipendiatlarini aniqlash uchun

8. Vazirlar Mahkamasining 2008 yil 13 oktyabrdagi “O‘zbekiston iqtidorli yoshlarini taqdirlash va moddiy rag‘batlantirish to‘g‘risida”gi 226-sonli qaroriga asosan aspirantlar orasidan O‘zbekiston Respublikasi Prezidenti stipendiatlarini aniqlash uchun;

9. Vazirlar Mahkamasining 2008 yil 13 oktyabrdagi “O‘zbekiston iqtidorli yoshlarini taqdirlash va moddiy rag‘batlantirish to‘g‘risida”gi 226-sonli qaroriga asosan respublika fan olimpiadalarining 4-bosqichi g‘oliblarini aniqlash uchun;

10. O‘zbekiston Respublikasi Prezidentining 2012 yil 10 apreldagi “Davlat boshqaruvi sohasida kadrlar tayyorlashni yanada takomillashtirish chora-tadbirlari to‘g‘risida”gi PF-4435-son Farmoniga asosan O‘zbekiston Respublikasi Prezidenti huzuridagi Davlat boshqaruv akademiyasi magistraturasiga qabul qilish uchun;

11. O‘zbekiston Respublikasi Vazirlar Mahkamasining 2012 yil 10 iyuldagi “Toshkent tibbiyot akademiyasi huzurida Harbiy-tibbiyot fakultetini tashkil etish

to‘g‘risida”gi 203-son qaroriga asosan Toshkent tibbiyot akademiyasining Harbiy-tibbiyot fakultetiga tanlov uchun.

Har yili test sinovidan o‘tayotgan respublika fuqarolari soni **750** mingdan oshadi. Bundan ko`rinib turibdiki fanlar bo`yicha online test saytini yaratish o`quvchi, abiturent, talaba yoshlarimiz va umuman barcha yoshdagi aholi uchun bilimlarini yanada mustahkamlash uchun asos bo`ladi desak mubolag`a bo`lmaydi.

### 1.3. Fanlar bo`yicha online-test saytining qulayligi va afzalligi

Fanlar bo`yicha online-test saytining qulayligi va afzalligi deganimizda biz uning yangi texnologiyalardan foydalangan holda tuzilganligini va bundan tashqari test tizimini ishlatishda quyidagi afzalliklarga ega ekanligini nazarda tutamiz.

*Bular*:

- Ixtiyoriy sistema foydalanuvchisi internet tarmog'i orqali sistemada ixtiyoriy joyda ishlay olishi;
- Sistemadan foydalanayotgan foydalanuvchidan hech qanday dasturiy ta'minot talab qilmasligi;
- Foydalanuvchining ushbu sistemada ishlash jarayonida informatsion sohaga ta’luqli bilimlarni talab qilmasligi;
- Sahifalarning sodda strukturaga ega ekanligi;

# II BOB. Tizimni yaratilish va yaratish davomida qo`llaniladigan dasturlar

## 2.1 Visual Studio 2013 dasturlash muhiti va C# dasturlash tili

Dasturlash tillari har xil maqsadlarga – murakkab matematik masalalarni yechish, iqtisodiy-matematik hisob-kitoblarni amalga oshirishdan tortib musiqa partituralari va kompyuter grafikasini yaratishgacha bo‘lgan ishlarni bajarish uchun ishlatiladi. Zamonaviy dasturlash tillarini yaratish masalasidagi tadqiqotlar dasturlashning o‘zi mavjud bo‘lgan vaqtdan boshlab olib boriladi. Microsoft kompaniyasi tomonidan bu izlanishlar natijasida dasturlashning zamonaviy standartlariga mos keladigan va .NET Framework texnologiyasini rivojlantirishni qo‘llab-quvvatlash uchun mo‘ljallangan C# tili ishlab chiqildi[8].

Oxirgi yillarda C# tili va u bilan bog‘liq bo‘lgan muhit, ya’ni .NET platformasi dasturiy ta’minot yaratuvchilar uchun asosiy yangi texnologiya bo‘lib kelmoqda. .NET platformasi orqali Windows muhitida hosil qilish mumkin bo‘lgan ixtiyoriy dasturlarni yaratish mumkin. C# tili esa yangi til hisoblanib, ushbu muhit uchun maxsus ishlab chiqilgan. C# tilidan foydalanib, dinamik WEB – sahifalarni, tarqatma texnologiya asosida yaratilgan dasturlarni, berilganlar bazasi bilan ishlovchi komponentlarni yoki oynali Windows dasturlarini yaratish mumkin. C# tili yordamida tarmoq yoki Internet dasturlaridan tashqari, Windows platformasida ishlovchi ixtiyoriy dasturni yaratish mumkin. C# va .NET platformasi orqali Windows muhitida dasturlar yaratish metodikasini mukammallashtirish ko‘zda tutilgan[8].

### C# tilining yaratilish tarixi

Kompyuter tillari o‘z-o‘zidan emas, balki o‘zaro bir-biriga bog‘liqlikda mavjud bo‘ladi. Har qanday yangi til u yoki bu shaklda oldingi yaratilgan tillarning xossalarini o‘ziga meros qilib oladi, ya’ni ketma-ketlik prinsipi amalga oshiriladi. Natijada bitta tilning imkoniyatlari boshqalari tomonidan foydalaniladi (masalan, yangi

xususiyatlar mavjud kontekstga birlashtiriladi, tilning eski tuzilishlari esa o‘chirib yuboriladi). Kompyuter tillarining evolyusiyasi shunday tarzda ro‘y beradi va dasturlash mahorati takomillashtiriladi[8].

C# tili yuqoridagilardan istisno emas, u boshqa dasturlash tillarining ko‘plab foydali imkoniyatlarini meros qilib oldi va dunyoda eng ko‘p qo‘llaniladigan ikkita kompyuter tillari – C, C++, shuningdek Java tili bilan uzviy bog‘liqdir. C# ni tushunish uchun mazkur bog‘liqlik tabiatini aniqlab olish kerak, shuning uchun oldin biz ushbu uch tilning rivojlanish tarixi to‘g‘risida to‘xtalib o‘tamiz[8].

C tili 1972 yilda Nyu-Djersi shtatining Myurrey-xill shahrida Bell Laboratories kompaniyasining tizimli dastur tuzuvchisi Dennis Richi tomonidan yaratilgan. Bu til o‘zini shunchalik yaxshi ko‘rsatdiki, oxir oqibatda unda Unix operatsion tizimlarining 90% yadro kodlari yozildi (oldin quyi darajadagi til assemblerda yozilgan). C ning vujudga kelishidan oldinroq yaratilgan tillardan, Pascal ulardan eng mashhuri hisoblanadi, yetarli darajada muvaffaqiyatli foydalanilgan, lekin aynan C tili dasturlashning zamonaviy davri boshlanishini belgilab berdi[8].

1960 yillarda dasturlash texnologiyalaridagi strukturaviy dasturlashlarning paydo bo‘lishiga olib kelgan inqilobiy o‘zgarishlar C tilini yaratish uchun asosiy imkoniyatlarni belgilab berdi. Strukturaviy dasturlashlarning paydo bo‘lishiga qadar katta dasturlarni yozish qiyin bo‘lgan, satr kodlari miqdorining oshishi sababli dasturlarning o‘tish joylari chalkash masalalariga aylanib ketishiga olib keladi. Strukturaviy tillar dastur tuzuvchi instrumentariysiga shartli operatorlarni, lokal o‘zgaradigan tartiblarni va boshqa mukammallashtirishlarni qo‘shib bu muammoni hal qildi. Shunday tarzda nisbatan katta dasturlarni yozish imkoniyati vujudga keldi[8].

Aynan C tili kuch, elegantlik va ma’nodorlikni o‘zida muvaffaqiyatli birlashtirgan birinchi strukturaviy til bo‘ldi. Uning bo‘lishi mumkin bo‘lgan xatolar mas’uliyatini tilga emas dastur tuzuvchi zimmasiga yuklaydigan prinsiplar

bilan inobatga olgan holda sintaksisdan foydalanishdagi qisqalik va osonlik kabi xususiyatlari tezda ko‘plab tarafdorlarini topdi.

Bugungi kunda biz mazkur sifatlarni o‘z o‘zidan anglashiladigan deb hisoblaymiz, lekin C da birinchi marotaba dastur tuzuvchiga zarur bo‘lgan ajoyib yangi imkoniyatlar mujassamlashtirilgan. Natijada 1980 yillardan boshlab Cstrukturaviy dasturlash tillari orasida eng ko‘p foydalaniladiganlaridan biri bo‘lib qoldi.

Biroq, dasturlashning rivojlantirish choralariga ko‘ra bundanda kattaroq dasturlarni qayta ishlash muammosi kelib chiqmoqda. Loyiha kodi ma’lum bir hajmga yetgan zahoti (uning raqamli ahamiyati dastur, dastur tuzuvchi, foydalanilgan instrumentlarga bog‘liq bo‘ladi, lekin taxminan 5000 satr kodlari nazarda tutilyapti) C dasturlarini tushunish va kuzatib borishda qiyinchiliklar yuzaga keladi.

Garchi Java tili dasturlarni bir platformadan boshqasiga o‘tkazishning ko‘plab muammosini hal qilgan bo‘lsa ham, zamonaviy Internet-muhitida samarali ishlashi uchun unga bir qator xossalar yetmayapti, ulardan biri  qancha kompyuter tillarini (ko‘p tilli dasturlash) o‘zaro aloqa imkoniyatlarini qo‘llab-quvvatlash hisoblanadi. *Ko‘p tilli dasturlash* deganda turli tillarda yozilgan kodning birgalikda ishlash qobiliyati tushuniladi. Bu imkoniyat katta dasturlarni yaratishda, shuningdek ko‘plab kompyuter tillarida va turli xil operatsion muhitlarda foydalanish mumkin bo‘lgan alohida komponentlarni dasturlashda juda muhimdir.

Windows platformalarini to‘g‘ridan-to‘g‘ri qo‘llab-quvvatlashning yo‘qligi Java ning jiddiy kamchiligi hisoblanadi (Garchi Java-dasturlari Windows muhitida installirlashgan JVM mavjudligida bajarishi mumkin bo‘lsa ham).

Ushbu muammoni hal etish uchun Microsoft kompaniyasi 1990 yillar oxirida bu kompaniyaning umumiy strategiyasi .NET ning tarkibiy qismi hisoblangan C# tilini ishlab chiqdi (tilning bosh me’mori Anders Xeylsberg). Alfa-versiya tili 2000 yil o‘rtalaridan muomala chiqarila boshlandi.

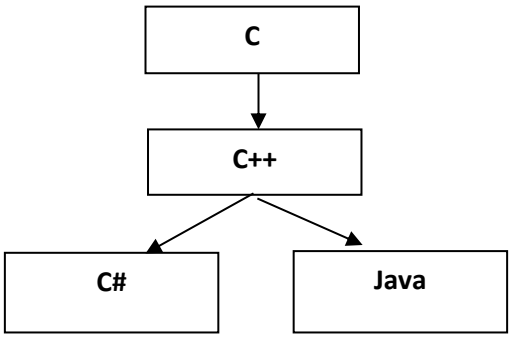
C# tili butun dunyoda keng qo‘llanilayotgan va eng ommabop bo‘lgan C, C++ va Java dasturlash tillari bilan to‘g‘ridan-to‘g‘ri bog‘liqdir. Hozirda amalda barcha professional dastur tuzuvchilar mazkur tilni biladi, shuning uchun ularga asoslangan C# ga o‘tish ortiqcha qiyinchiliklarsiz ro‘y beradi. Xeylsberg, C++ va Java tillari muallifi sifatida, “g‘ildirak kashf” qilmadi, balki kashf etilgan yo‘ldan ketdi – asos sifatida oldin yaratilgan tillardan foydalangan holda e’tiborni yaxshilash va innovatsiyalarga qaratdi.

C# ning genealogik tasnifi 1. rasmda ko‘rsatilgan. C# tili C++ da aniqlagan ob’ektli modelga qurilgan, sintaksisi, ko‘plab kalit so‘zlar va operatorlarni u C tilidan meros qilib olgan. Agarda siz dasturlashning ushbu tillarini bilsangiz, unda sizga C# ni o‘rganishda muammo kelib chiqmaydi.

C# va Java o‘rtasidagi aloqa nisbatan murakkab. Ikkala til ham o‘tkazuvchi kodni yaratish uchun ishlab chiqilgan, C va C++ larga asoslanadi, ularning sintaksisi va ob’ektli modelidan foydalanadi. Biroq, mazkur tillar o‘rtasida to‘g‘ridan-to‘g‘ri aloqa yo‘q, ular ko‘proq umumiy ajdodlarga ega, lekin ko‘plab belgilari bilan farq qiluvchi amakivachchalarga o‘xshaydi. Agarda siz Java da ishlashni bilsangiz, bu C# ni o‘zlashtirishingizni yengillashtiradi, va buning teskarisi, Java ni o‘rganishda C# ning ko‘plab konsepsiyalari bilimlari sizga foydasi tegadi.

C# tili mazkur kitobda ko‘rib chiqiladigan ko‘plab innovatsion xossalarga ega. Birdaniga anglash mumkinki, uning bir nechta eng muhim yangiliklari dasturiy ta’minlash komponentlarini o‘rnatilgan qo‘llab-quvvatlashga taalluqlidir. Ya’ni aslida C# komponentlarga yo‘naltirilgan o‘zida, masalan, dasturiy ta’minlash komponentlarining tarkibiy qismlarini bilvosita qo‘llab-quvvatlovchi elementlarni (xossa, usul va hodisalar kabilar) qamrab oluvchi til sifatida yaratilgan.

Lekin, ehtimol, C# ning eng muhim yangi xususiyati – bu uning ko‘p tilli muhitda ishlash qobiliyatining mavjudligidir.




1. – rasm. C# ning genealogik tasnifi

**C# ning .NET Framework bilan aloqasi.** Garchi C# dasturlash tili sifatida alohida o‘rganilishi mumkin bo‘lsada, lekin uni o‘zi ishlaydigan .NET Framework-muhiti bilan o‘zaro aloqasini ko‘rib chiqish afzalroqdir. Chunki, birinchidan, C# dastlab Microsoft kompaniyasi tomonidan .NET Framework kodini yaratish uchun ishlab chiqilgan, ikkinchidan, .NET Framework C# tili tomonidan foydalaniladigan kutubxonani aniqlab beradi. Modomiki ular bunday chambarchas bog‘liq ekan, .NET Frameworkning umumiy konsepsiyasini va uning C# uchun ahamiyatini tushunish muhimdir[8].

**.NET Framework haqida.** .NET Framework deb platforma mustaqil izohlarni rivojlantirish va bajarishni qo‘llab-quvvatlovchi muhitni talqin qilsa bo‘ladi. U dasturlashning turli tillari izohlarida birgalikda ishlash imkonini beradi, shuningdek Windows uchun umumiy dasturlash modellari va dasturlar o‘tkazuvchanligini ta’minlaydi. Ta’kidlash kerakki, .NET Framework Windows platformasi tomonidan chegaralanmagan va ushbu platforma uchun yozilgan dasturlar kelajakda boshqa platformaga o‘tkazilishi mumkin.

C# tili .NET Framework ning ikki muhim tashkil etuvchilaridan foydalanadi. Birinchi – bu *tilning ijro etish muhitiga bog‘liq bo‘lmagan*(Common Language Runtime, CLR), sizning dasturlaringiz ijrosini

boshqaruvchi va .NET Framework texnologiyasining bir qismi hisoblanuvchi tizim, qaysiki dasturlarga o‘tuvchan bo‘lish imkonini beradi, bir qancha tillardan foydalanish bilan dasturlashni qo‘llab-quvvatlaydi va ma’lumotlarni uzatish xavfsizligini ta’minlaydi.

Ikkinchi tashkil etuvchi - dasturlarga ijro muhitiga kirish imkonini beruvchi *.NET sinflar kutubxonasidir,* masalan, ma’lumotlarni kiritish/chiqarish uchun foydalaniladi. Agar siz endigina boshlagan dastur tuzuvchi bo‘lsangiz, unda sizga *sinf* tushunchasi notanish bo‘lishi mumkin.Sal quyida biz sinflar to‘g‘risida batafsil to‘xtalib o‘tamiz, hozir shunchaki uqtirib o‘tamizki, sinflar dasturlarni tashkillashtirishga yordam beruvchi ob’ektga yo‘naltirilgan konstruksiya hisoblanadi. Hozircha sizning dasturingiz xususiyatlar, .NET sinflarining muayyan kutubxonasi bilan chegaralangan bo‘lsa, u .NET ijro muhiti qo‘llab-quvvatlanadigan barcha joyda ishlatilishi mumkin.

.NET kodi bajarilishi bilan tilga bog‘liq bo‘lmagan bajarilish muhiti (CLR) boshqaradi. U qanday ishlashini bayon qilamiz. C# dasturini kompilyasiya qilganda, biz bajariladigan kodni emas, Microsoftning oraliq tili deb ataluvchi (Microsoft Intermediate Language, MSIL) maxsus psevdokoddan tashkil topgan faylni olamiz. MSIL tili konkret protsessorga bog‘liq bo‘lmagan tashiladigan ko‘rsatmalar to‘plamini aniqlaydi, ya’ni, mohiyati bo‘yicha, MSIL tashiladigan assembler tilini aniqlaydi. Oraliq Microsoft tili mazmunan Java bayt-kodiga o‘xshashligiga sizning diqqatingizni qaratmoqchimiz, ammo ularning orasida farq bor.

CLR tizimi oraliq kodni bajariladigan kodga dastur ishga tushirilgan vaqtda translyasiya qiladi. MSILda kompilyasiya qilingan ixtiyoriy dastur CLR muhiti realizatsiya qilingan ixtiyoriy operatsion tizimda ishga tushirilishi mumkin. Bu .NET Frameworkda dasturlarning tashiluvchanligiga erishilishiga yordam beradigan mexanizmning bir qismidir.

Oraliq Microsoft tili JIT kompilyatoridan foydalanilganda bajariladigan kodga aylanadi (ingl. just in time –kerakli vaziyatda) . Jarayon quyidagicha ishlaydi: .NET

dasturi bajarilganda CLR tizimi JIT kompilyatorini faollashtiradi, qaysiki u MSIL ni mazkur protsessorning ichki kodiga aylantiradi, u ham bo‘lsa dastur qismlarining kodlari ehtiyoj bo‘lgandagina imkon darajasida aylantiriladi. Shunday qilib, sizning C# dasturingiz aslida ichki kod sifatida ijro etiladi, garchi u azaldan MSILda kompilyasiya qilingan bo‘lsada. Bu degani, sizning dasturingizni ishga tushish vaqti amalda xuddi uning birdaniga ichki kodga kompilyasiya qilinganiday, ammo bunda sizda MSILning ustunligi - dasturning tashiluvchanligi paydo bo‘ladi. Bundan tashqari, C# dasturini kompilyasiya qilganda MSILga qo‘shimcha siz yana bitta berilganlarni tasvirlaydigan, dasturingiz tomonidan ishlatiladigan va sizning kodingizga boshqa kod bilan o‘zaro ta’sir qiladigan-metama’lumot komponentani olasiz. Metama’lumotlar MSIL joylashgan fayllarda saqlanadi. Aslida bu CLR muhiti, MSIL va metama’lumotlar to‘g‘risidagi bilimlar C# tilida dasturlash masalasining asosiy qismini realizatsiyasi uchun yetarli, qaysiki bu til ishni kerakli tarzda mustaqil ravishda tashkil qiladi.

## 2.2. Web ilovalarining ishlash tamoyillari hamda ASP.NET MVC texnologiyalari haqida

Web-ilova - bu shunday ilovaki, server bajarayotgan barcha ishlarini Internet orqali klient qurilmasiga yuboradi. Ularni qo'llash uchun Web-browser zarurdir.

Web-texnologiyalar bilan bog'liq texnologiya bilan ishlashda turli xil dasturlar mavjud bo'lib, ular vaqt o'tgan sayin o'zgarib, murakkablashib bormoqda. Ya'ni texnika-texnologiyalarining keskin suratda rivojlanishi yangidan yangi dasturlash tillarini o'rganishni va bilishni talab etmoqa. Hozirgi kunda bu dasturlardan eng zamonaviysi Visual Studio 2013.ASP.NET va berilganlar bazasi SQL Server 2012shular jumlasidandir. Visual Studio 2013- bu dastur yaratuvchi uchun Microsoft platformasida dastur (ilova) yaratish imkonini beruvchi instrumentlar ya'ni uskunalar to'plamidir. ASP.NET bazasida dasturlash uchun VisualStudio.NET redaktoridan foydalaniladi. ASP.NET

ilovasi turli xil dasturlash tillarida yaratilishi mumkin. Odatda bu dasturlash tillari Visual Basic va C# tillari hisoblanadi. Ushbu tizimni yaratish jarayonida nega endi aynan Visual Studio 2013: ASP.NET dan foydalaniladi? degan savolni qo'yilishi tabiiy. Chunki web-texnologiyalar bilan ishlovchi, Web-formalarni yaratish imkoniyati mavjud bo'lgan bir nechta dasturlash tillari mavjud. Aynan Visual Studio 2013: ASP.NET dan foydalaninshning sababi Visual Studio 2013 tarkibiga kiruvchi ASP.NET Web sahifalarni bevosita dasturlash jarayonida yaratish imkonini beradi. Unda web sahifaning HTML kod (Sourse) va C# da dasturlash orqali amalga oshirsa bo'ladi.

ASP.NET boshqa web-ilovalarni yaratuvchi plotformalardan farqli bir qancha afzalliklarga ega. Eng asosiy afzalliklaridan biri bu - dasturlash uskunakari va Windows serveri bilan integratsiyalashuvidir.

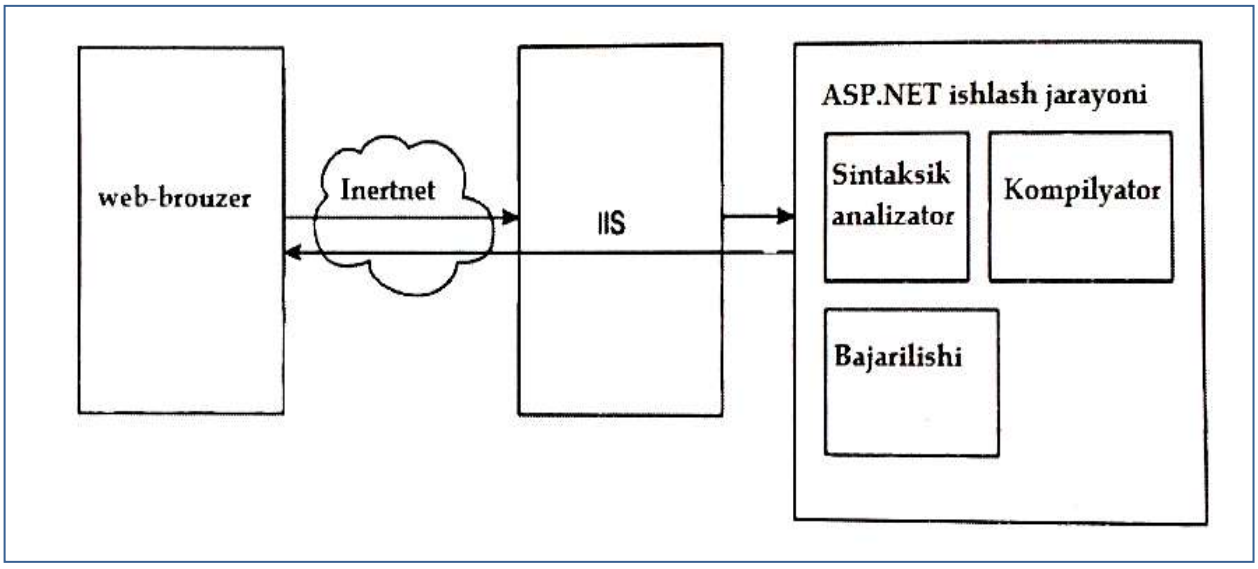
Web-loyihalar bilan ishlash jarayonida ko'pgina hollarda mavjud bo'lgan instrumentlar to'plami kerakli talablarni qanoatlantirmasligi muammosiga duch kelamiz. Web-ilova web-serverni HTML-kodni kliyentga yuborishga majbur qiladi. Bunday kodlar Internet Explorer kabi web-brouzerlarda aks etadi. Brouzerning Adreslar satriga foydalanuvchi URL-adresni kiritganda Web-serverga HTTP- so'rov yuboriladi. HTTP-so'rov tarkibiga so'ralgan fayl nomi va quyidagi qo'shimcha ma'lumotlar kiradi: identifikatsiyalanuvchi kliyent ilovasi, klient tomonidan qo'llab-quvvatlanuvchi tillar, kerakli so'rov bo'yicha qo'shimcha ma'lumotlar. Web-server Web-brouzer asosida foydalanuvchiga matnli darcha, tugmalar va ro'yxatlarni ko'rsatuvchi tarkibiga HTML-kod kiruvchi HTTP-protokolini qaytaradi.

ASP.NET - bu dinamik web-sahifani server tomonda kod yordamida yaratishga mo'ljallangan texnologiya hisoblanadi. Bu web-sahifalar Windowsning kliyent dasturlariga o'xshash turli dasturlar orqali ham yaratilgan bo`lishi mumkin. Kliyent tizimlarda Web-ilovalar uchun ASP.NET ni qo'llash uchun oddiygina Web-brouzer zarur bo'ladi. Bunda Internet Explorer, Opera, Netscepe Navigator,

Firefox yoki ixtiyoriy HTML ni qo'llab-quvvatlovchi web-brouzer va bunda .NET platformasini o'rnatish zaruriyati yo'q.

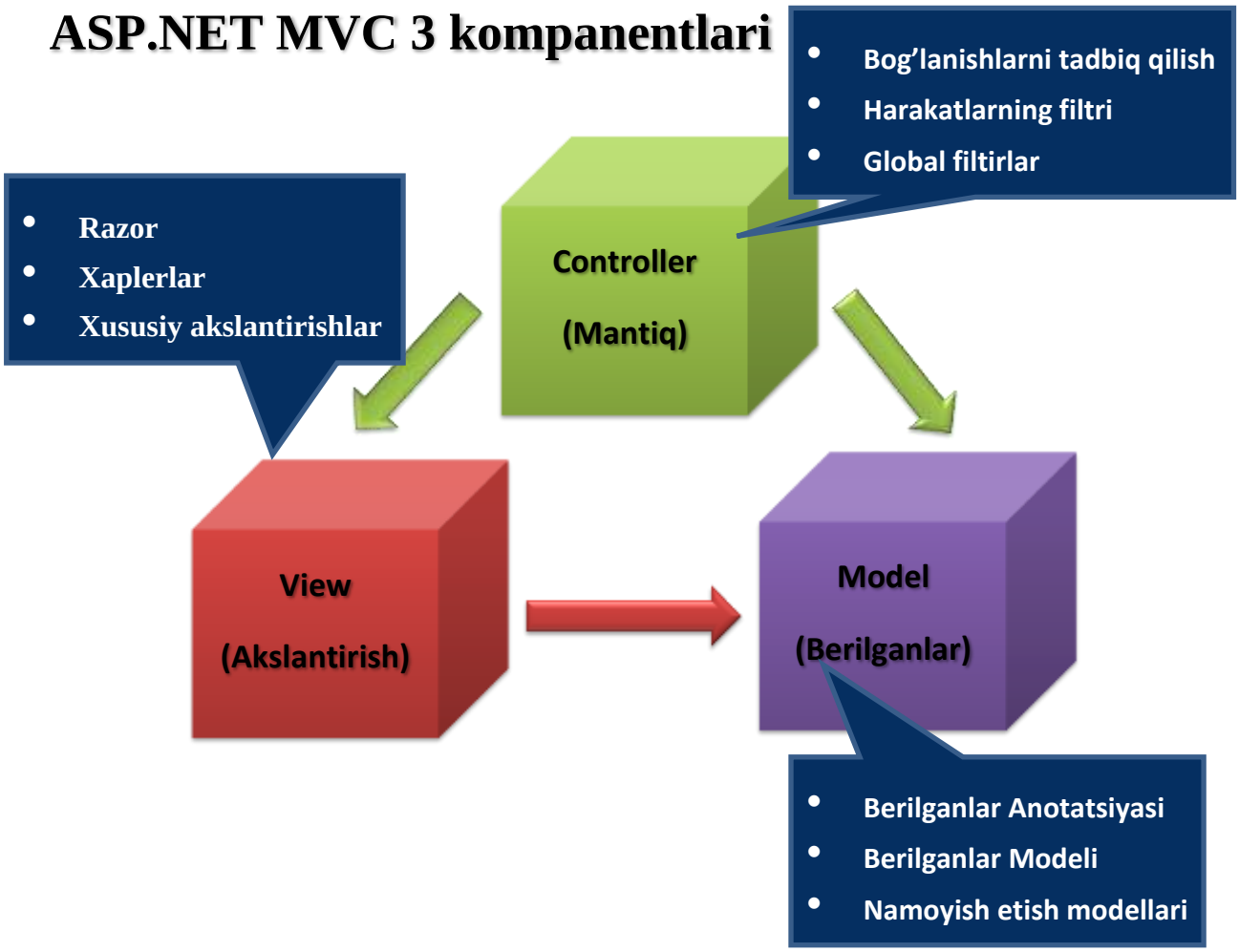
Server tizimlari ASP.NET muhitida bajariluvchi bo'lsa u o'rnatilgan bo'lishi zarur. Agar sistemada Internet Information Services (IIS) xizmati o'rnatilgan bo'lsa, Net Fremwork platformasini installyatsiya qilayotgan vaqtda server bajariluvchi ASP.NET muhitni konfiguratsiya qiladi. Ishlab chiqish jarayonida Visual Studio o'zining shaxsiy ASP.NET Web Development Serveriga ega bo'lagani uchun IIS bilan ishlashga hech qanday hojat qolmaydi.

Quyidagi sxemada Web-brouzerdan kelib tushgan so'rov bo'yicha bajariluvchi ASP.NET muhitini qanday ishlashini 2-rasmda ko'rishimiz mumkin.




2-rasm

Hozirgi kunda Amerika, Yevropa va Isroilda ASP.NET dasturchilar PHP dasturchilariga nisbatan o'n barobar ko'p. Bu esa shuni izohlaydiki, katta va qimmat hisoblanadigan loyihalarni yaratishda PHP dan ko'ra ASP.NET da yozilgan dasturlar tezroq ishlaydi va deyarli ishdan chiqmaydi.




### 2.3. Berilganlar ombori (bazasi) (SQL SERVER 2012) haqida

Ushbu yaratilayotgan loyiha berilganlar bazasi bilan bog'liq holda ishlar ekan, yaratish davomida berilganlar bazasini tashkil qilish uchun SQL Server 2012 dan foydalaniladi. Bu berilganlar bazasini boshqarish tizimi bo'lib, o'z navbatida katta hajmdagi ma`lumotlarni ham tez va sifatli ishlash imkonini beradi.

versiyalarida (DB2, Oracle, Ingres, Informix, Sybase, Progress, Rdb) va hattoki norelyatsion MBBT versiyalarida (masalan, Adabas) "Klient/Server" texnologiyasi va SQL tilidan foydalanadi.

SQL tilida Ma'lumotlarni jadval ko'rinishida tasvirlashga yo'naltirilgan, amallar kontseptsiyasi ko'p bo'lmagan (30 dan kam) ifodalardan iborat kompakt til yaratishga imkon beradi.

SQL tilining ikki xil formasi mavjud: **Interaktiv** va **Joylashtirilgan.** Ko'p xollarda ikkala forma bir xil ishlaydi, lekin ikki xil foydalaniladi:

SQL tilining ***Interaktiv*** formasi berilganlar bazasi o'zida faoliyat ko'rsatadi va buyurtmachi foydalanishi uchun chiqish xosil qilishda ishlatiladi. SQL tilining bu formasida kiritilgan komanda darhol bajariladi va foydalanuvchi shu zaxoti natijaga ega bo'ladi.

SQL tilining ***Joylashtirilgan*** formasi boshqa tilda yaratilgan dasturga joylashtirilgan SQL komandalardan iboratdir.

SQL Interaktiv va joylashtirilgan formalarida ko'p sonli guruxlar yoki qism bo'limlar mavjud. Ular ANSI tomonidan e'tiborga olingan va konseptual darajada foydali, lekin ko'pchilik SQL dasturlar ularni aloxida qayta ishlamaydi, shuning uchun ular aslida SQL komandalarining funktsional kategoriyalaridir.

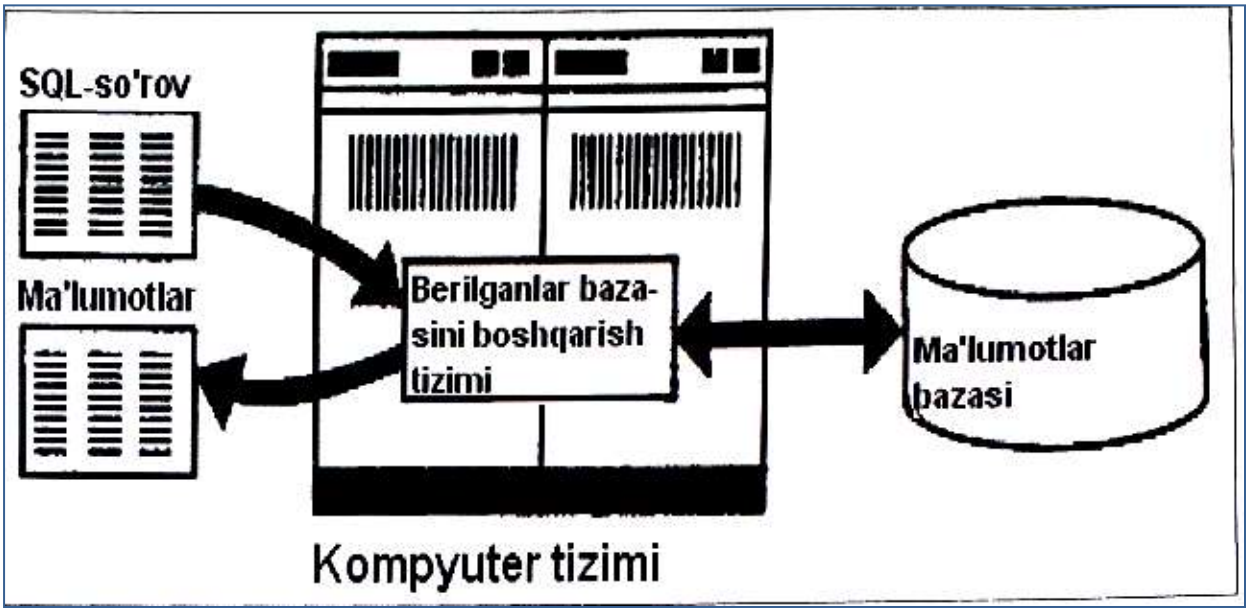
- DDL *(Ma'lumotlarni Ta'riflash Tili)* - ANSI da Sxemani ta'riflash tili, ob'ektlarni (jadvallar, indekslar, tasawurlar va xokazo) yaratuvchi komandalardan iborat.

- DML (*Ma'lumotlarni O'zgartirish Tili*) - bu ixtiyoriy daqiqada jadvallarda qanday qiymatlar saqlanishini aniqlovchi komandalar majmuasidir.

- DCD (*Ma'lumotlarni Boshqarish Tili*) foydalanuvchiga ma'lum ob'ektlar ustida ma'lum ta'sir o'tkazishga ruxsat berish yoki bermaslikni aniqlovchi vositalardan iborat.

SQL Standard ANSI modeli tomonidan aniqlangan va xozirda ISO modeli tomonidan qabul qilingan. Lekin kommertsial berilganlar bazasi dasturlari ANSImodelini ogoxlantirmasdan SQL tilini kengaytiradi, ya'ni foydali bo'lgan turli xil xossalar qo'shiladi.

Har bir tilning o'z ishlash jarayoni mavjud. SQL Serverda ham ishlash jarayoni bo'lib u quyidagi sxema orqali ko'rsatilgan.(3-rasm).




3-rasm

# III BOB. Dasturiy ta`minot

## 3.1. Dasturning modeli

Men ushbu bitiruv malakaviy ishimda turli fanlar bo`yicha online test o`tkazish saytini yaratdim. Bunda ASP.net MVC framework va ma`lumotlar ombori bilan almashish uchun Entity framework imkoniyatlaridan keng foydalandim. Hozirgi kunda yirik texnologiyalar MVC ya`ni model, ko`rinish, boshqaruv arxitekturasidan foydalanib kelinmoqda. Haqiqatdan har qanday web sayt yoki web dasturni bir biriga bog`liq bo`lmagan 3 qismga ajratishlik juda katta qulaylikni olib keladi. Bunda dasturchi bir vaqtni o`zida bir biriga bog`liq bo`lmagan holda alohida elementlarga ya`ni dastur mantig`iga, uning ko`rinishiga va ular o`rtasidagi aloqani o`rnatishga qaratishi mumkin. Saytning asosiy mantig`i model ASP.net MVC da sinflar yordamida quriladi. Entity frameworkning vazifasi sinflar kodidagi maydonlarni ma`lumotlar ombori maydonlariga interpretatsitya qilishdan iborat va aksincha.

Entity framework qurgan modelimizga qarab SQL SERVER 2012 ma`lumotlar omborida jadvallar yaratib ular o`rtasidagi bog`liqlikni o`rnatadi. Agar biz shu jadvallarga ma`lumot kiritish yoki olishni xohlasak, unga yangi so`rovlar bilan ishlaydigan Linq tilida murojat qilishimiz zarur. Bu til SQL tiliga o`xshash bo`lib malumotlar ombori murakkab so`rovlarni soddaroq holda ifodalash mumkin bo`ladi. Linq tilining SQL tilidan afzallik tarafi shundaki, unda murakkab ko`rinishdagi so`rovlarni qisqaroq ifoda etish mumkin bo`ladi.

Test saytida jadval bog`lanishlarida birga ko`p arxitekturasiga ko`ra aloqa o`rnatilgan. Masalan, bitta test bir nechta javob variantlariga ega bo`lishi mumkin ya`ni test jadvalidagi bitta qiymat test javoblari jadvalidagi ko`p qiymatlarga birlamchi kalit bilan bog`langan bo`ladi.

ASP.NET MVC texnologiyasi avtorizatsiya va autentifikatsiya uchun ko`p murakkab turdagi kutubxona yaratgan bo`lib, uning to`liq arxitekturasini bilish

dasturchidan talab etilmaydi. Dasturchi faqat qaysi sahifaga qaysi foydalanuvchi ruhsat etilganligini dastur ko`dida ko`rsatishi yetarli.

Controllerlar yordamida biz web sayt sahifalarini mantiqiy bo`laklarga ajratishimiz mumkin. Bunda har bir Controller faqat maxsus model bilan ish olib boradi. Masalan TestController test modeli bilan UserController foydalanuvchi user modeli bilan aloqa o`rnatadi. Bir Controller bir nechta akslanish(View)lar hosil qilishi mumkin. Barcha Controllerlarda bosh sahifa akslanish(View) indeks hisoblanadi. Agar view ko`rsatilmagan bo`lsa Controller foydalanuvchiga bosh sahifani namoyon etadi. ASP.net MVC texnologiyasida mashrutizatsiya(adres URL) quyidagi tartibda aniqlangan. HostNomi: Port kodi/Controller/View/Id Bunda lokalhost:26126/Test/Index/1.

Agar konfiguratsiya faylida id, View kiritilmagan bo`lsa mos ravishda 1 va index deb qabul qiladi. Index- shart bo`lmagan parameter bo`lib unga ko`pincha sahifa modelining kodi kiritiladi. Masalan agar bu test sahifasi bo`lsa id sifatida test raqamini olishi mumkin.

Controllerda yuqorida aytilganidek har bir View uchun sarlavha qismida ruhsati bor foydalanuvchilar ro`yhatini keltirishimiz mumkin. Bunda faqat shu foydalanuvchilargina sahifani ko`rish imkoniga ega bo`ladi. Boshqa foydalanuvchilar esa ro`yhatdan o`tish sahifasiga o`tkazilib yuboriladi. Bunda tizim “cookie”lar yordamida sahifalar eslab qolinadi. Agar foydalanuvchi ro`yhatdan o`tsa va shu foydalanuvchiga sahifa ko`rishi ruhsat etilgan bo`lsa shu foydalanuvchi sahifaga qaytib kelishi maqsadida cookie ishlatiladi. Controller va akslanish(View) o`rtasidagi aloqa asosan View ning qat`iy tiplashganligi va ko`rsatilgan Model bo`yicha qurilganligi asosida o`rnatilgan. Umuman olganda Controller Viewga Modeldan tashqari ViewBag maydonlari orqali ma`lumot uzatishi mumkin. ViewBag dinamik sinf bo`lib, unga dasturning ishlash davomida ixtiyoriy nom ostida maydon kiritish mumkin bo`ladi. Ko`pincha sahifa nomlari

ViewBag orqali akslantirishga uzatiladi. Bizning holda testdan testga o`tishda testing Id maydoni ViewBag yordamida uzatilgan.

Formalarda kiritish maydonlari uchun validlikka tekshirilgan, ya`ni boshqacha aytganda foydalanuvchi kiritish davomida yo`l qo`ygan qandaydir xatoliklarni oldini olish uchun foydalanuvchi tomonidan va Server tomonidan kiritilgan maydonlar tekshiruvlari olib borilgan. Foydalanuvchi tomonidan tekshirish uchun ASP.net MVC frameworki JavaScript tili va JQuery kutubxonalaridan foydalangan holda juda ko`p tayyor obyektlar amalga oshirilgan. Server tomonidagi tekshirishlar foydalanuvchining brouzerida JavaScript o`chirib qo`yilganligini inobatga olinganligi uchun qayta tekshiriladi.

Xavfsizlik tomoniga keladigan bo`lsak bizda SSL sotib olmaganligimiz uchun xavfsizlikni 100% deb bo`lmaydi. Ko`p web sahifalar xususan nuu.uz SSL ishonchnomasiga ega emas. Bu degani Serverga murojatni qat`iy bir hostdan emas ixtiyoriy hostdan ma`lumot jo`natish mumkinkligi yo`l qo`yib beriladi. Bu esa robat mashinalarning saytga amalga oshiruvchi xakerlik hujumi bo`lishi mumkinligini keltirib chiqaradi. Bizning holda bu kabi muammolarni hal etish uchun ma`lumotlar omboriga faqat chekli sondagi foydalanuvchi ta`sir ko`rsatishi mumkin. Tizimda testni tahrirlash, qo`shish, uni ochirib tashlash faqat admin va o`qituvchi amalga oshirishi mumkin va bu jarayonlarni amalga oshirishdan avval cookielarda foydalanuvchining Id raqami tekshirilib boriladi. Shu yo`l bilan biz qandaydir xavfsizlikning turg`unligini oshirdik.

Tet tizimida 3 turdagi foydalanuvchi ishlashi mumkin: talaba, admin va o`qituvchi. Tizimga faqat ro`yhatdan o`tgandan(avtorizatsiyani amalga oshirgandan) so`ng kiriladi. Ro`yhatdan o`tish login va parolning kiritilish yo`li bilan amalga oshiriladi. Agar foydalanuvchi ilgari test tizimidan foydalanmagan bo`lsa va birinchi marta kirmoqchi bo`lsa u avval registratsiyani amalga oshirishi lozim bo`ladi. Registratsiya oddiy so`rovnoma formasi shaklida bo`lib unga foydalanuvchi o`zining ismi sharifi, eletron pochtasi, foydalanuvchi qaysi nom

ostida kirishi uchun login va parolni ikki marta kiritishi lozim. Shunda foydalanuvchi uchun ma`lumot omborida joy ajratilib, unga oddiy foydalanuvchi degan status beriladi. U bu tizimdan test topshirishi, o`z ko`rsatgichlarini ko`rishi mumkin holos. U ma`lumotlar omboriga ta`sir o`tkazish huquqiga ega emas. Ma`lumot omborlari bilan faqat administrator to`laqonli ishlashi mumkin va qisman o`qituvchi o`z testlari uchun amallarni bajarishi mumkin. O`qituvchi test yaratish, o`z testlarini tahrirlash huquqiga ega. Faqat uning testini topshirgan foydalanuvchilar ko`rsatgichini baholab borishi mumkin. Administrator eng katta huquqqa ega bo`lib, u bu tizimdagi ushbu statusga ega bo`lgan yagona foydalanuvchi hisoblanadi. Yuqoridagi foydalanuvchining huquqlaridan tashqari barcha foydalanuvchilarni nazorat qilish, testlarni tahrirlash yoki umuman o`chirib tashlash, topshirilgan nazoratlarni ko`rish va boshqarish oddiy foydalanuvchini o`qituvchi maqomiga ko`tarish yoki umuman foydalanuvchini o`chirib tashlash kabi huquqlarga ega hisoblanadi.

### 3.2. Berilganlar bazasidagi jadvallar sxemasi

1-jadval bu test savollarining  jadvali bo`lib unda shu test kodi(ID kalit maydoni), test savoli(question), qiyinlik darajasi(range), ko`p javoblilik va test mavzu kodi(Categories jadvaliga tashqi kalit bilan bog`langan)maydonlari mavjud.

| | Name | Data Type | Allow Nulls |
|---|---|---|---|
| 🔑 | ID | int | ☐ |
| | question | nvarchar(1024) | ☐ |
| | range | tinyint | ☐ |
| | multiple | bit | ☐ |
| | categoryId | bigint | ☐ |
| | | | ☐ |

1-jadval.

dbo.Tests berilganlar bazasi jadvali tarkibi quyidagicha:

Bu test javoblarining jadvali bo`lib unda shu test javobining kodi(ID kalit maydoni), test javobi(answer), javobning to`g`ri yoki noto`g`ri ekanligi(correct), qaysi testing javob variant ekanligi(Tests jadvaliga tashqi kalit bilan bog`langan)maydonlari mavjud. dbo.TestAnswers berilganlar bazasi jadvali tarkibi quyidagicha:

| | Name | Data Type | Allow Nulls |
|---|---|---|---|
| 🔑 | ID | int | ☐ |
| | answer | nvarchar(MAX) | ☑ |
| | correct | bit | ☐ |
| | TestId | int | ☐ |
| | | | ☐ |

2-jadval.

Bu test mavzulari jadvali bo`lib unda shu test javobining kodi(ID kalit maydoni), test mavzusi(name), testing fani(subject), tuzgan foydalanuvchi(UserProfiles jadvaliga tashqi kalit bilan bog`langan)maydonlari mavjud. dbo.Categories berilganlar bazasi jadvali tarkibi quyidagicha:

| | Name | Data Type | Allow Nulls |
|---|---|---|---|
| 🔑 | ID | int | ☐ |
| | name | nvarchar(MAX) | ☑ |
| | subject | nvarchar(MAX) | ☑ |
| | testCreator | int | ☐ |
| | | | ☐ |

3-jadval

Bu foydalanuvchi Profile jadvali bo`lib unda shu foydalanuvchining kodi(ID kalit maydoni), foydalanuvchi login(login), ismi(name),familyasi(surname), email, parol (password), testni o`qish (canRead), test hosil qilish(Create), testni tahrirlash maydonlari mavjud. dbo.Users berilganlar bazasi jadvali tarkibi quyidagicha:

| | Name | Data Type | Allow Nulls | Default |
|---|---|---|---|---|
| 🔑 | ID | int | ☐ | |
| | login | nvarchar(MAX) | ☑ | |
| | password | nvarchar(MAX) | ☑ | |
| | canRead | bit | ☐ | |
| | canCreate | bit | ☐ | |
| | canModify | bit | ☐ | |
| | name | nvarchar(MAX) | ☑ | |
| | surename | nvarchar(MAX) | ☑ | |
| | email | nvarchar(MAX) | ☑ | |
| | | | ☐ | |

4-jadval

Bu foydalanuvchining javobini saqlaydigan jadval bo`lib unda shu jadval kod maydoni(ID kalit maydoni), qaysi javobni tanlashi(answered), test savolining kodi(testId), qaysi foydalanuvchi ekanligi(Userid), qaysi vaqtda javob qaytargani(time) va test qachon oshlanganligi(startedTime_ID)maydonlar mavjud. testId, UserId, startedTime_ID (mos ravishda Tests, Users, StartedTimes jadvallariga tashqi kalit bilan bog`langan)maydonlari mavjud.

| | Name | Data Type | Allow Nulls | Default |
|---|---|---|---|---|
| 🔑 | ID | int | ☐ | |
| | answered | int | ☐ | |
| | testId | int | ☐ | |
| | Userid | int | ☐ | |
| | time | bigint | ☐ | |
| | startedTime_ID | int | ☑ | |
| | | | ☐ | |

5-jadval

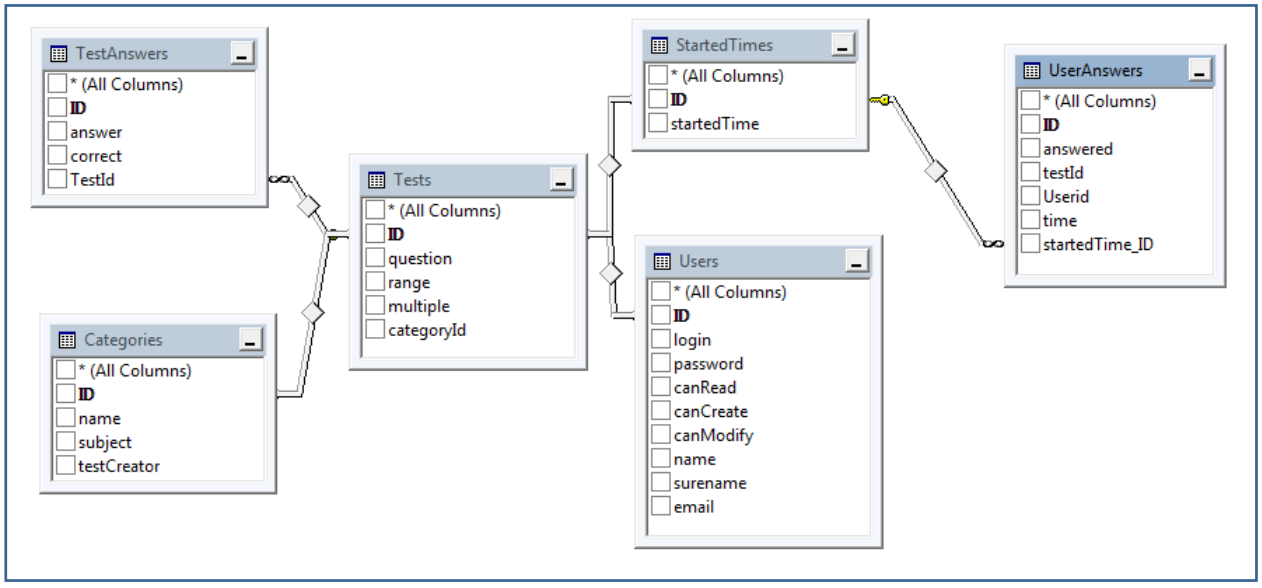
dbo.UserAnswers berilganlar bazasi jadvali tarkibi quyidagicha:

Bu StartedTimes jadvali bo`lib unda test boshlangan vaqtini eslab qoladi. Kodi (ID kalit maydoni), boshlangan vaqti(startedTime) maydonlari mavjud. dbo.StartedTimes berilganlar bazasi jadvali tarkibi quyidagicha:

| | Name | Data Type | Allow Nulls | Default |
|---|---|---|---|---|
| 🔑 | ID | int | ☐ | |
| | startedTime | bigint | ☐ | |
| | | | ☐ | |

6-jadval

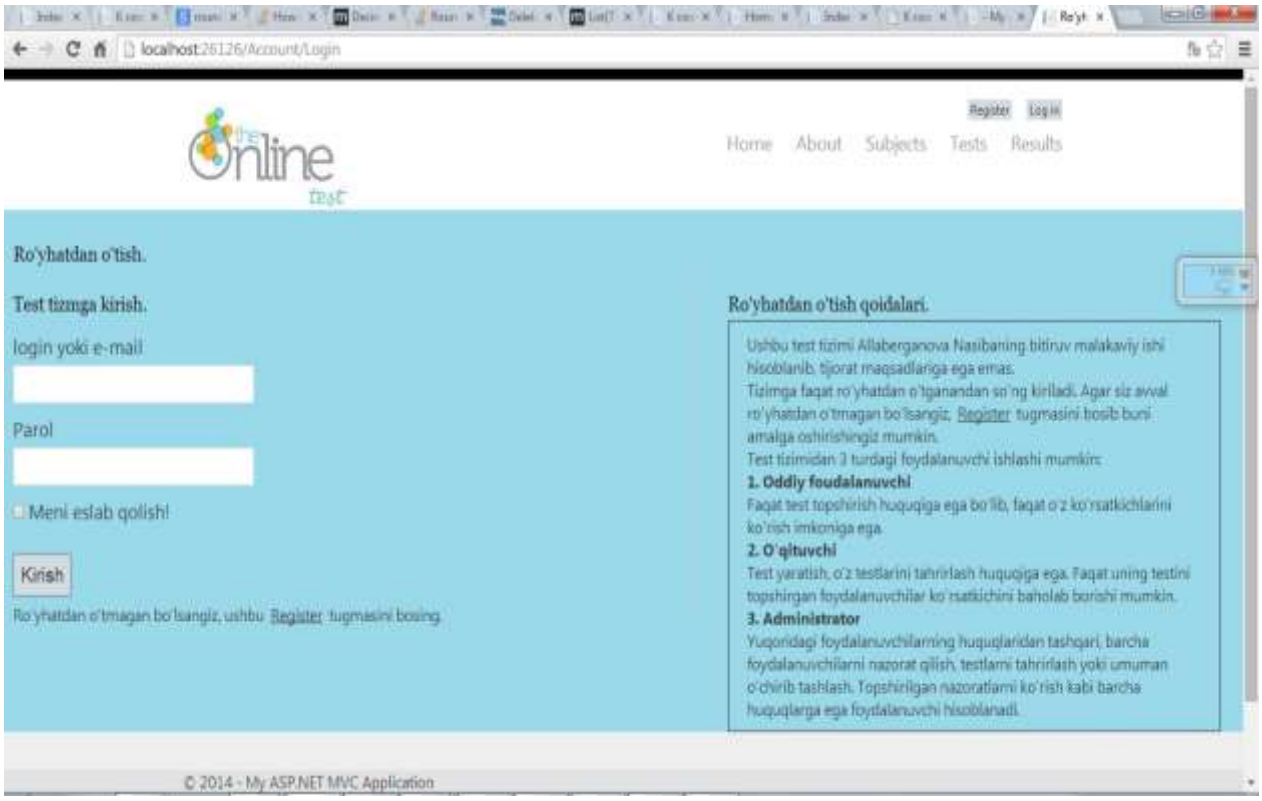
Yuqoridagi jadvallarning o`zaro bog`lanishi quyidagi 1-sxema orqali ifodalangan




1-sxema

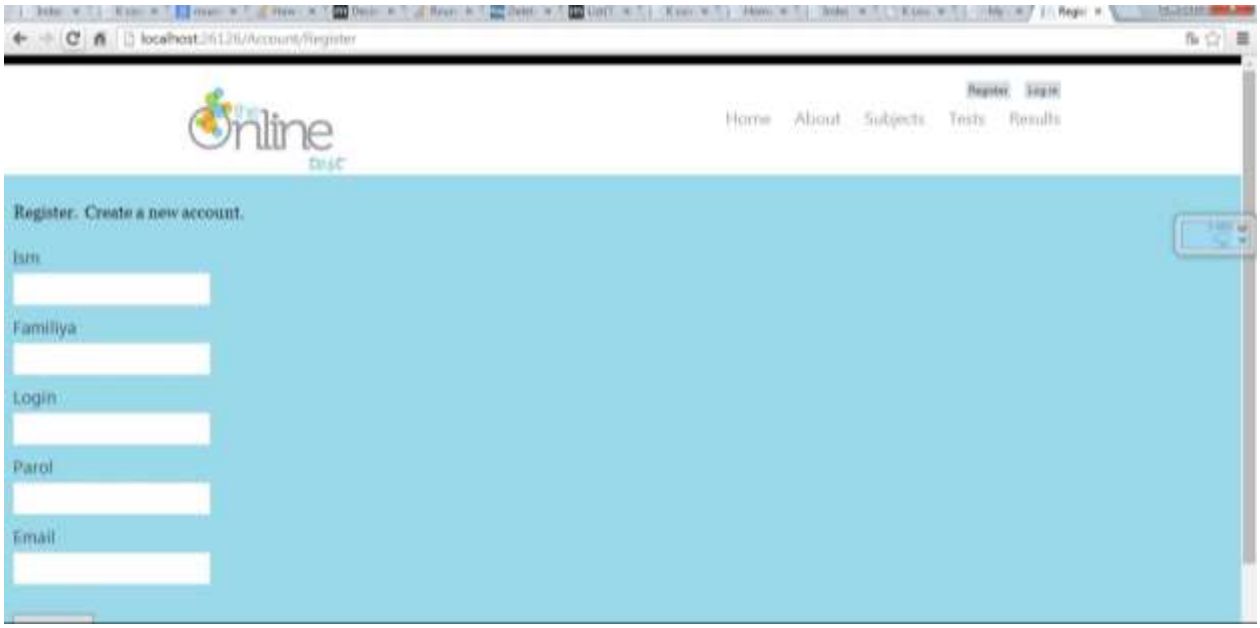
### 3.3. Tizimning Web-brauzerdagi ko`rinishi va strukturasi.

Saytmizni ishgatushirganimizda bizga quyidagi oyna (4-rasm) hosil bo’ladi. Bu oynadan foydalanuvchi o’zining login paroli orqali saytga kirishi mumkin bo’ladi.




4-rasm

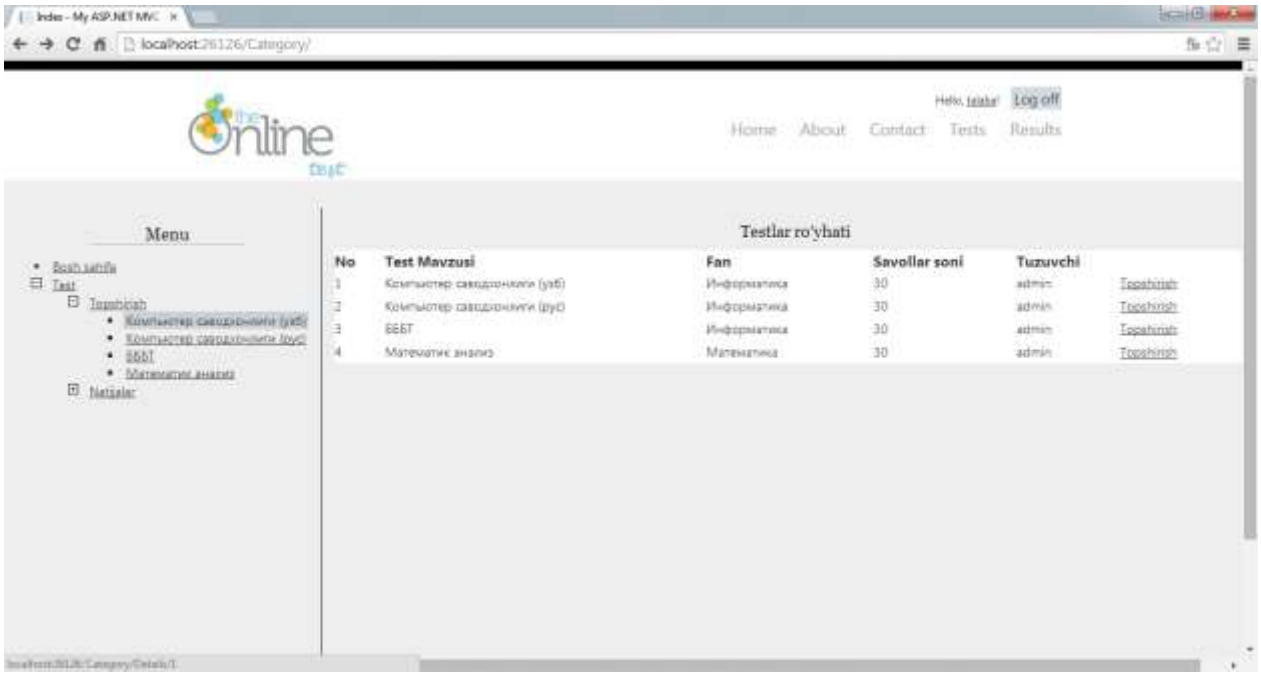
Agarda foydalanuvchida login, parol bo’lmasa u saytning yuqori qismidagi **Register** buyrug’ini tanlab ro’yxatdan o’tishi mumkin bo’ladi. Foydalanuvchi ro’yxatdan o’tish oynasi quyidagicha:




5-rasm

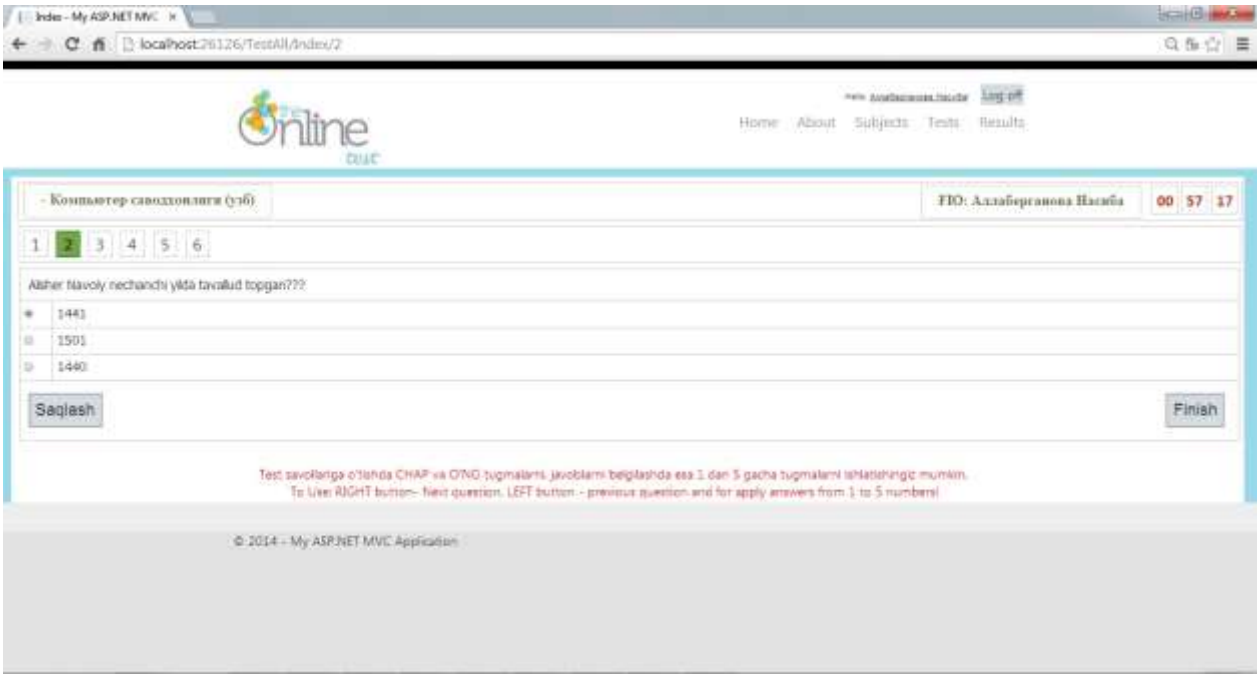
Bu oynada foydalanuvchu ismi, famiyasi, logini va E-mile adresini kiritshi kerak bo'ladi.

Foydalanuvchining bosh sahifasi quyidagicha (6-rasm):




6-rasm

Foydalanuvchining test topshirish sahifasi quyidagicha (7-rasm). Bu sahifada foydalanuvchi test savollariga jovob beradi:

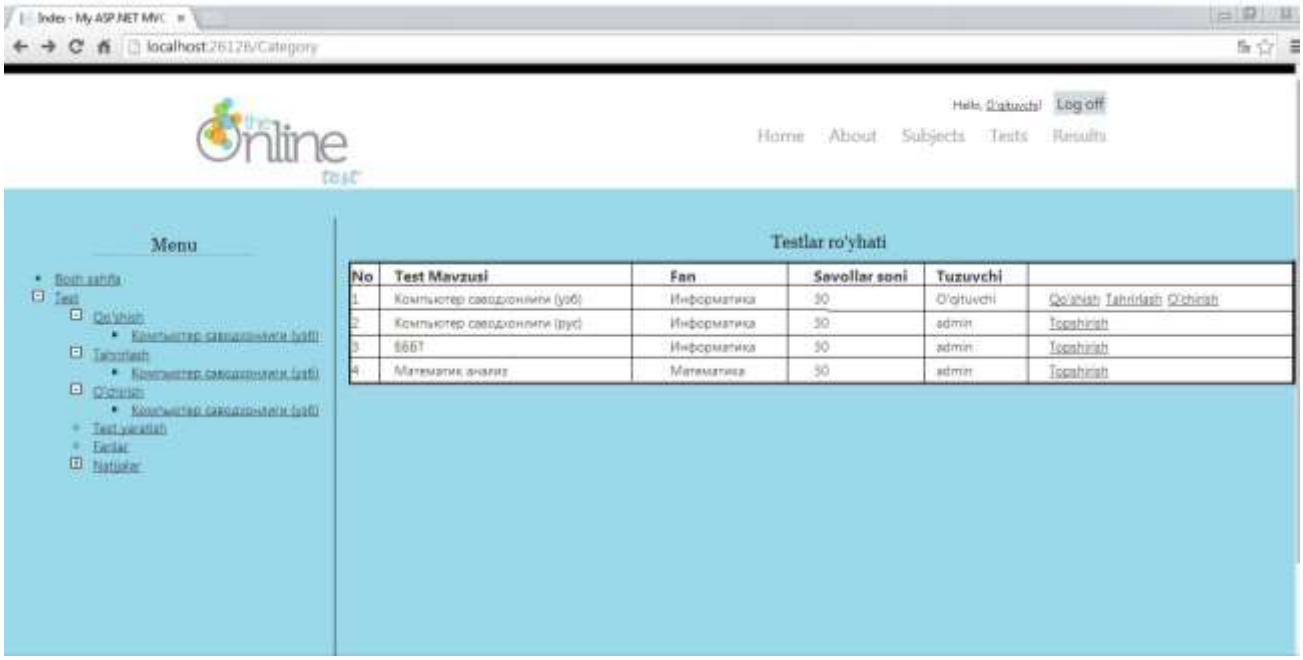



7-rasm

Foydalanuvchining natijalar sahifasi quyidagicha(8-rasm). Bu sahifada foydalanuvchining qaysi fandan test topshirgani, to’gri javoblar soni, noto’g’ri javoblar soni, foydalanuvchining o’zlashtirish ko’rsatkichi va testdan olgan bahosini ko’rishi mumkin bo’ladi :

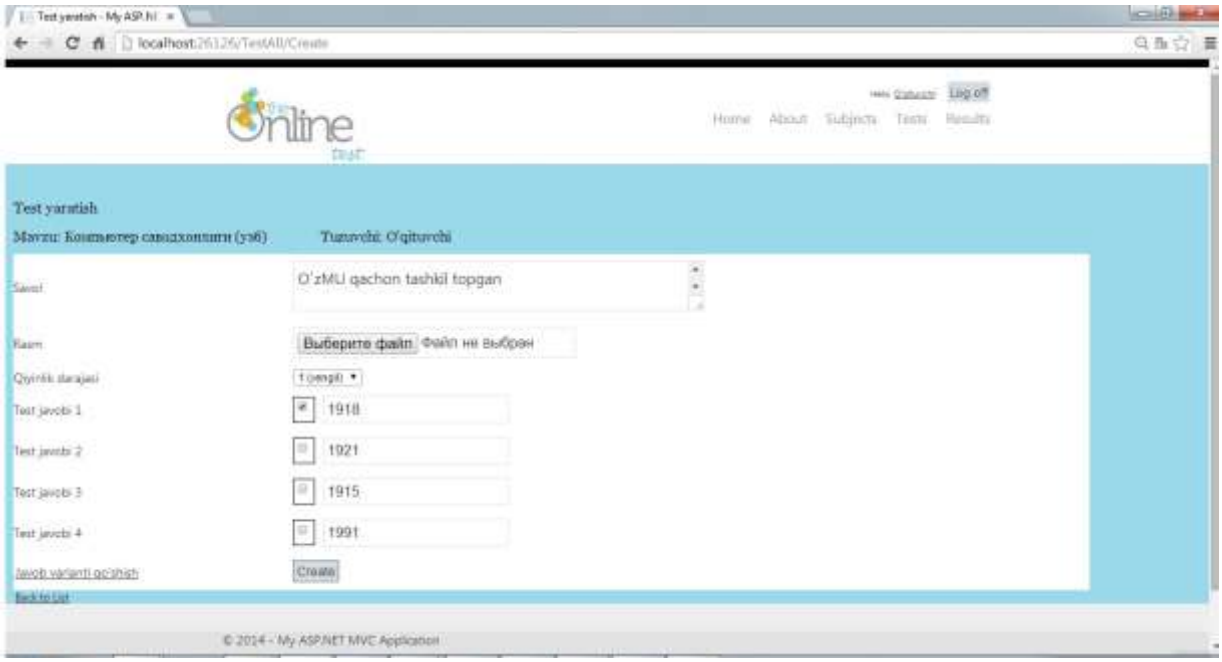

8-rasm

Saytimizdagi o’qituvchi oynasining bosh sahifasi 9-rasmda keltirilgan. Bu sahifa o’rqali o’qituvchi fanlarni taxrirlashi, yangi test savollarini qo’shishi, testlarni taxrirlashi mumkin bo’ladi.



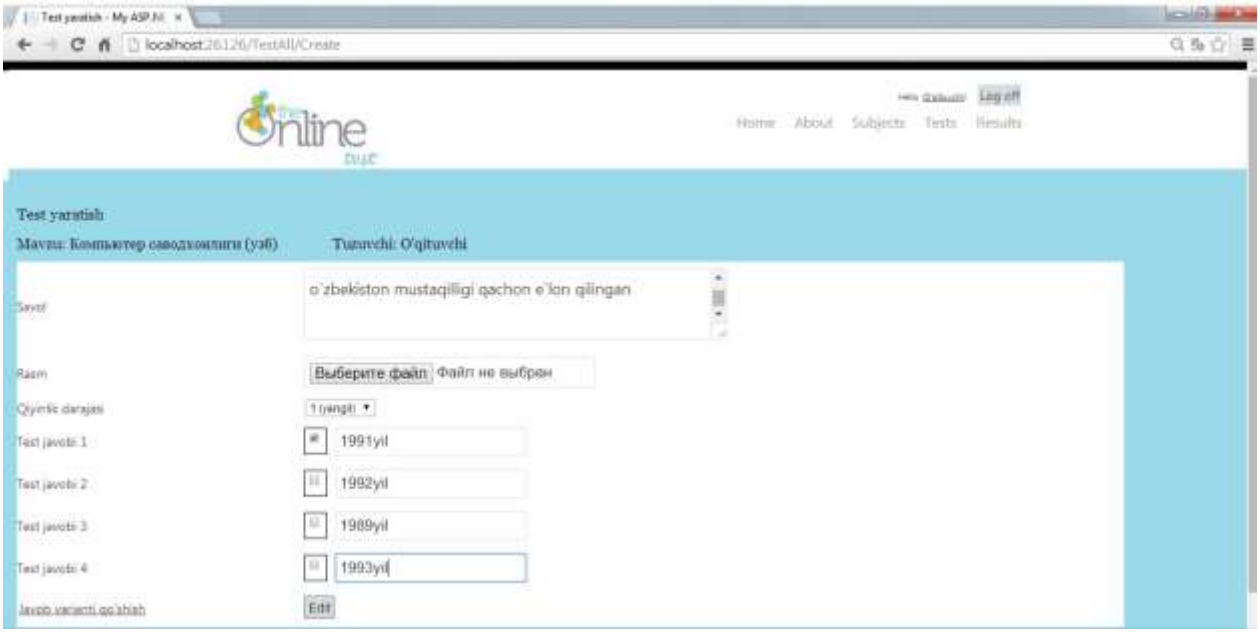

9-rasm

Quyida 10-rasmda o’qituvchining o’qituvchini test savollarini yuklash oynasi keltirilgan.



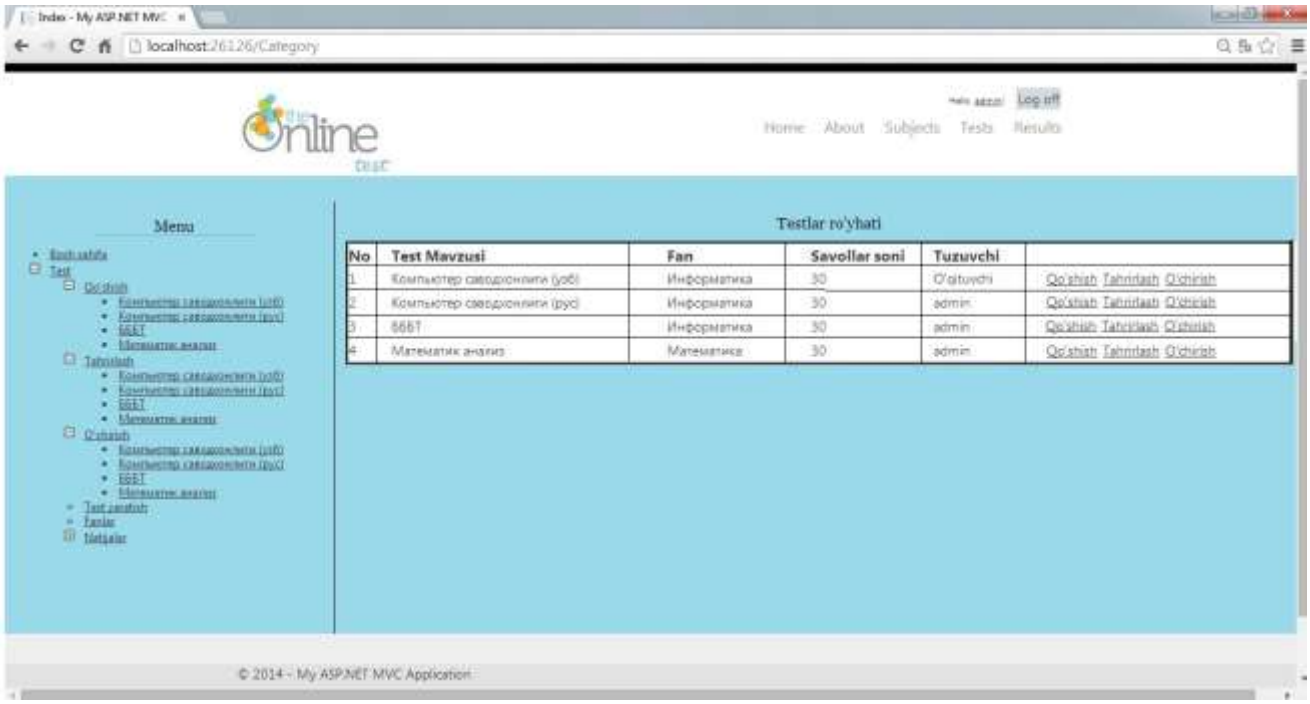

10-rasm

Quyida 11-rasmda o’qituvchi test savollarini taxrirlash oynasi keltirilgan:




11-rasm

Test saytimizning administrator bosh sahifasi 12-rasmda keltirilgan. Bu yerda administrator foydalanuvchilar bilan ishlash, fanlarni taxrirlash, yangi fan qo’shish, testlarni taxrirlash, yangi test qo’shish o’chirishi mumkin bo’ladi:



# Xulosa

O`zbekiston mustaqillikka erishgadan so`ng davlatimizning asosiy qonuni – Konstitutsiyamizga ega bo`ldik. Vaqt o`tishi bilan hukumatimiz rahbarining O`zbekistonda “Kadrlar tayyorlash milliy dasturi” va “Axborot texnologiyalari sohasida kadrlar tayyorlash tizimini takomillashtirish” qarorlari alohida o`rin tutadi. Undagi g`oyalar avvalo XXI asrda yashaydigan, Vatan va yurt mustaqilligini mustahkamlovchi yoshlarning manfaatini ifodalaydi. Dasturda e`tirof etilgan kadrlar tayyorlash tizimini rivojlantirishning asosiy yo`nalishlarida shu narsa alohida belgilandiki, ta`lim tizimini yaxlit axborotlashtirish hozirgi zamon talablariga muvofiqlashuvini ta`minlaydi. Bu dasturni o`tgan davr mobaynida sarhisob qilib, ularning eng muhim xususiyatlaridan biri sifatida hozirgi ta`lim dasturida nazarda tutilayotgan zamonaviy axborot texnologiyalari, ta`limni kompyuterlashtirish va kompyuterlash tarmoqlari tizimida ta`lim jarayonini axborot bilan ta`minlashni yanada rivojlantirishni nazarda tutmoqdamiz. Zero, Milliy dasturni ro`yobga chiqarishga doir tashkiliy ishlarda e`tirof etilganidek, “Ta`limni axborot bilan ta`minlash tizimini shakllantirishva rivojlantirish, uni jahon axborot tizimi bilan bog`lash, ommaviy axborot vositalarining ta`lim sohasidagi vazifalarini belgilash” malakali mutaxassislarni tayyorlashning muhim mezonlaridan biridir[2].

Xozirgi kunda o`quv jarayoni monitoring va boshqaruvning avtomatlashtirilgan tizimni yaratish va takomillash eng dolzarb masalalardan biridir. Ushbu malakaviy bitiruv ishi “Fanlar bo`yicha online test saytini yaratish”. Bu malakaviy bitiruv ishida barcha foydalanuvchilar foydalanishi yordam ko`rsatuvchi dastur tuzish ko`zda tutilgan edi. Bu yo`lda Visual Studio 2013 muhiti va SQL SERVER 2012 dasturidan foydalangan holda online test tizimini yaratdim. Bu test tizimining asosiy qulayliklari – foydalanuvchilar ixtiyoriy vaqtda o`zini bilimlarini tekshirib, kamchiliklari ustida ishlashi mumkin. Barcha topshirilgan test natijalari berilganlar bazasida saqlanishi esa foydalanuvchi o`z bilim darajasi

ortishi yoki kamayishini ko`rish imkonini beradi. Bu esa juda muhim ko`rsatgich deb o`ylayman.

Ushbu test tizimi ixtiyoriy mutaxassislik uchun qo`llanishi mumkinligi ham ahamiyatlidir (albatta, berilganlar bazasida kerakli fan bo`yicha savollar mavjudligini hisobga olish kerak).

Ushbu test tizimi foydalanuvchilar imtixonga tayyorlanishida qulayliklar yaratib beradi deb o`ylayman va yordam ko`rsatadi deb umid qilaman.

# Foydalanilgan adabiyotlar.


1. I.A.Karimov “Barkamol avlod-O`zbekiston taraqqiyotining poydevori” Toshkent 1998.
2. O’zbekiston Respublikasi Prizidenti qarorlari.*www.lex.uz.*
3. Xakimov M.X,Gaynazarov S.M. “Berilganlar bazasini boshqarish tizimlari”.
4. Sh.A.Nazirov, A.Ne’matov, R.V. Qobulov Ma’lumotlar bazasini dasturlash chuqurlashtirilgan kursi.
5. Steven Sanderson, Pro ASP.NET MVC FRAMEWORK.
6. Djeffri Palermo - ASP.NET MVC 4 v Deystvii – 2012.
7. Magdanurov G.I., Yunev V.A. - ASP.NET MVC Framework (Professionalnoe programmirovanie) - 2010.
8. SHildg G.Polniy spravochnik po C#.
9. V.V. Labor C# -Sozdanie prilojeniy dlya Windows.
10. Devid Seppa, Microsoft ADO.NET.
11. http://www.compbook.net

# Ilova

Test Model

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace TestOnMvc.Models
{
    public class Test
    {
        public int ID { get; set; }
        [Required]
        [StringLength(1024)]
        public string question { get; set; }

        [Range(1,3)]
        [Required]
        public byte range { get; set; }
        public bool multiple { get; set; }

        public long categoryId { get; set; }
        public virtual ICollection<TestAnswer> answers {get; set; }
    }
}
```

TestAnswer Modeli

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;

namespace TestOnMvc.Models
{
    public class TestAnswer
    {
        public int ID { get; set; }
        public string answer { get; set; }
        public bool correct { get; set; }
        public int TestId { get; set; }
    }
}
```

Users modeli

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace TestOnMvc.Models
{
    public class User
    {
```

```
        public int ID { get; set; }
        public string name { get; set; }
        public string surename { get; set; }
        public string login { get; set; }
        public string email { get; set; }

        public string password { get; set; }
        public bool canRead { get; set; }
        public bool canCreate { get; set; }
        public bool canModify { get; set; }
    }
}
```

Category modeli

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace TestOnMvc.Models
{
    public class Category
    {
        public int ID { get; set; }
        public string name { get; set; }
        public string subject { get; set; }
        public int testCreator { get; set; }
    }
}
```

UserAnswer modeli

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace TestOnMvc.Models
{
    public class UserAnswer
    {
        public int ID { get; set; }
        public int answered { get; set; }
        public int testId { get; set; }
        public int Userid { get; set; }
        public long time { get; set; }
        public StartedTime startedTime { get; set; }
    }
}
```

SatrtedTime modeli

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace TestOnMvc.Models
{
    public class StartedTime
    {
```

```
        public int ID { get; set; }
        public long startedTime {get; set;}
    }
}
```

Ma’lumotlar omborini boshqarish TestOnMvcDb modeli

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;
```

Foydalanuvchilarni boshqarish AccountModels modeli

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.Entity;
using System.Globalization;
using System.Web.Security;

namespace TestOnMvc.Models
{
    public class UsersContext : DbContext
    {
        public UsersContext()
            : base("DefaultConnection")
        {
        }

        public DbSet<UserProfile> UserProfiles { get; set; }
    }

    [Table("UserProfile")]
    public class UserProfile
    {
        [Key]
        [DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity)]
        public int UserId { get; set; }
        public string UserName { get; set; }
    }

    public class RegisterExternalLoginModel
    {
        [Required]
        [Display(Name = "Login")]
        public string UserName { get; set; }

        public string ExternalLoginData { get; set; }
    }

    public class LocalPasswordModel
    {
        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Amaldagi parol")]
        public string OldPassword { get; set; }

        [Required]
```

```
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.",
MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Yangi parol")]
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Yangi parolni tasdiqlash")]
        [Compare("NewPassword", ErrorMessage = "Parollar mos kelmayapti.")]
        public string ConfirmPassword { get; set; }
    }

    public class LoginModel
    {
        [Required]
        [Display(Name = "login yoki e-mail")]
        public string UserName { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Parol")]
        public string Password { get; set; }

        [Display(Name = "Meni eslab qolish!")]
        public bool RememberMe { get; set; }
    }

    public class RegisterModel
    {
        [Required]
        [Display(Name = "Login")]
        public string UserName { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.",
MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Parol")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Parolni tasdiqlash")]
        [Compare("Password", ErrorMessage = "The password and confirmation password do not
match.")]
        public string ConfirmPassword { get; set; }
    }

    public class ExternalLogin
    {
        public string Provider { get; set; }
        public string ProviderDisplayName { get; set; }
        public string ProviderUserId { get; set; }
    }
}


namespace TestOnMvc.Models
{
    public class TestOnMvcDb : DbContext
    {
```

```
        public DbSet<Test> Tests { get; set; }
        public DbSet<TestAnswer> Answers { get; set; }
        public DbSet<TestAll> AllTests { get; set; }

        public DbSet<Category> Categories { get; set; }
        public DbSet<UserAnswer> UserAnswers { get; set; }

        public DbSet<User> Users { get; set; }

        public DbSet<StartedTime> StartedTimes { get; set; }


    }
}
```

---

HomeController - Bosh oyna Controllerlari

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using TestOnMvc.Models;
using TestOnMvc.Models;

namespace TestOnMvc.Controllers
{
    public class HomeController : Controller
    {
        TestOnMvcDb _db = new TestOnMvcDb();
        public ActionResult Index(string searchTerm = null)
        {
           /* var model =
                from t in _db.Tests
                where searchTerm == null || t.question.StartsWith(searchTerm)
                orderby t.answers.Count() descending
                select new TestViewListModel
                {
                    ID = t.ID,
                    question = t.question,
                    range = t.range,
                    CategoryId = t.categoryId,
                    multiple = t.multiple,
                    NumberOfAnswers = t.answers.Count()
                };
            */
            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Your app description page.";

            return View();
        }

        public ActionResult Contact()
```

```
        {
            ViewBag.Message = "Your contact page.";

            return View();
        }

        protected override void Dispose(bool disposing)
        {
            if(_db != null)
            {
                _db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

TestAllController

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using TestOnMvc.Models;

namespace TestOnMvc.Controllers
{
    public class TestAllController : Controller
    {
        private TestOnMvcDb db = new TestOnMvcDb();
        List<UserAnswer> answerList = new List<UserAnswer>();
        public int getCorrectAnswer(TestAll test)
        {
            if (test.correct1) return 1;
            if (test.correct1) return 2;
            if (test.correct1) return 3;
            if (test.correct1) return 4;
            if (test.correct1) return 5;
            return 0;
        }
        //
        // GET: /TestAll/

        public ActionResult Index(int id=1, int start = 0)
        {
           /* if(testId>0&&(answer.answered != 0))
            {
                db.UserAnswers.Add(answer);
                db.SaveChanges();
            }*/
            ViewBag.id = id;

            List<UserAnswer> asnwerList1 = db.UserAnswers.ToList();
            foreach (UserAnswer answer in asnwerList1)
            {
```

```
        if (answer.Userid == 1) //answer.time > answer.startedTime &&
            answerList.Add(answer);
    }
    ViewBag.answerList = answerList;
    return View(db.AllTests.ToList());
}


[HttpPost]
public ActionResult Index(UserAnswer answer,int id=1, int testNumber = 1)
{
    if (answer.answered != 0)
    {
        db.UserAnswers.Add(answer);
        db.SaveChanges();
    }
    return RedirectToAction("Index", new { id = id });
}
//
// GET: /TestAll/Details/5

public ActionResult Details(int id = 0)
{
    TestAll testall = db.AllTests.Find(id);
    if (testall == null)
    {
        return HttpNotFound();
    }
    return View(testall);
}

//
// GET: /TestAll/Create

public ActionResult Create()
{
    return View();
}

//
// POST: /TestAll/Create

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create(TestAll testall)
{
    if (ModelState.IsValid)
    {
        db.AllTests.Add(testall);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    return View(testall);
}

//
// GET: /TestAll/Edit/5

public ActionResult Edit(int id = 0)
{
```

```
    TestAll testall = db.AllTests.Find(id);
    if (testall == null)
    {
        return HttpNotFound();
    }
    return View(testall);
}

//
// POST: /TestAll/Edit/5

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit(TestAll testall)
{
    if (ModelState.IsValid)
    {
        db.Entry(testall).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(testall);
}

//
// GET: /TestAll/Delete/5

public ActionResult Delete(int id = 0)
{
    TestAll testall = db.AllTests.Find(id);
    if (testall == null)
    {
        return HttpNotFound();
    }
    return View(testall);
}

//
// POST: /TestAll/Delete/5

[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
    TestAll testall = db.AllTests.Find(id);
    db.AllTests.Remove(testall);
    db.SaveChanges();
    return RedirectToAction("Index");
}
[Authorize(Users="admin")]
[HttpGet]
public ActionResult UploadTest()
{
    Category model = new Category();
    return View(model);
}

[HttpPost]
public ActionResult UploadTests(Category cat, string file)
{
    FileControl fc = new FileControl();
```

```
            List<TestAll> tests = fc.uploadFile(file);
            foreach (TestAll item in tests)
                db.AllTests.Add(item);
            db.SaveChanges();
            return View(tests);
        }
        public ActionResult TestResult()
        {

            int score = 0;
            int totalScore =0;
            int mark = 0;
            List<UserAnswer> model = db.UserAnswers.ToList();
            foreach (var answer in model)
            {
                bool isset = false;
                foreach(var item in answerList){
                    if(item.testId == answer.testId)
                        isset = true;
                }
                if (!isset)
                    answerList.Add(answer);
            }
            ViewBag.testAnswered = answerList.Count;

            // Check results
            var tests = db.AllTests.ToList();
            // Nado ostavit' tolko otveti polz ID po Categoriyu

            foreach(var test in tests)
            {
                foreach(var answer in answerList)
                {
                    if (answer.answered == getCorrectAnswer(test) && answer.testId ==
test.ID)
                    {
                        score += test.range;
                        break;
                    }
                }
                totalScore += test.range;
            }
            ViewBag.totalTestNumber = tests.Count;
            ViewBag.totalScore = totalScore;
            double persantage = (double)score * 100 / totalScore;
            ViewBag.score = score;
            ViewBag.persantage = Math.Round(persantage, 2);
            if (persantage < 101 && persantage > 85)
                mark = 5;
            if (persantage < 86 && persantage > 70)
                mark = 4;
            if (persantage < 71 && persantage > 55)
                mark = 3;
            if (persantage < 56 && persantage > -1)
                mark = 2;
            ViewBag.mark = mark;
            return View(answerList);
        }
        public ActionResult View1()
        {
```

```
            return View();
        }
        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

CategoryController

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using TestOnMvc.Models;

namespace TestOnMvc.Controllers
{
    public class CategoryController : Controller
    {
        private TestOnMvcDb db = new TestOnMvcDb();

        //
        // GET: /Category/

        public ActionResult Index()
        {
            return View(db.Categories.ToList());
        }

        //
        // GET: /Category/Details/5

        public ActionResult Details(int id = 0)
        {
            Category category = db.Categories.Find(id);
            if (category == null)
            {
                return HttpNotFound();
            }
            return View(category);
        }

        //
        // GET: /Category/Create

        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /Category/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
```

```
public ActionResult Create(Category category)
{
    if (ModelState.IsValid)
    {
        db.Categories.Add(category);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    return View(category);
}

//
// GET: /Category/Edit/5

public ActionResult Edit(int id = 0)
{
    Category category = db.Categories.Find(id);
    if (category == null)
    {
        return HttpNotFound();
    }
    return View(category);
}

//
// POST: /Category/Edit/5

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit(Category category)
{
    if (ModelState.IsValid)
    {
        db.Entry(category).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(category);
}

//
// GET: /Category/Delete/5

public ActionResult Delete(int id = 0)
{
    Category category = db.Categories.Find(id);
    if (category == null)
    {
        return HttpNotFound();
    }
    return View(category);
}

//
// POST: /Category/Delete/5

[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
```

```
            Category category = db.Categories.Find(id);
            db.Categories.Remove(category);
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        public ActionResult View1()
        {
            return View(db.Categories.ToList());
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

AccountController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Transactions;
using System.Web;
using System.Web.Mvc;
using System.Web.Security;
using DotNetOpenAuth.AspNet;
using Microsoft.Web.WebPages.OAuth;
using WebMatrix.WebData;
using TestOnMvc.Filters;
using TestOnMvc.Models;

namespace TestOnMvc.Controllers
{

    [Authorize]
    [InitializeSimpleMembership]
    public class AccountController : Controller
    {
        //
        // GET: /Account/Login

        [AllowAnonymous]
        public ActionResult Login(string returnUrl)
        {
            ViewBag.ReturnUrl = returnUrl;
            return View();
        }

        //
        // POST: /Account/Login

        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult Login(LoginModel model, string returnUrl)
        {
            if (ModelState.IsValid && WebSecurity.Login(model.UserName, model.Password,
persistCookie: model.RememberMe))
            {
```

```
                return RedirectToLocal(returnUrl);
            }

            // If we got this far, something failed, redisplay form
            ModelState.AddModelError("", "The user name or password provided is
incorrect.");
            return View(model);
        }

        //
        // POST: /Account/LogOff

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult LogOff()
        {
            WebSecurity.Logout();

            return RedirectToAction("Index", "Home");
        }

        //
        // GET: /Account/Register

        [AllowAnonymous]
        public ActionResult Register()
        {
            return View();
        }

        //
        // POST: /Account/Register

        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult Register(RegisterModel model)
        {
            if (ModelState.IsValid)
            {
                // Attempt to register the user
                try
                {
                    WebSecurity.CreateUserAndAccount(model.UserName, model.Password);
                    WebSecurity.Login(model.UserName, model.Password);
                    return RedirectToAction("Index", "Home");
                }
                catch (MembershipCreateUserException e)
                {
                    ModelState.AddModelError("", ErrorCodeToString(e.StatusCode));
                }
            }

            // If we got this far, something failed, redisplay form
            return View(model);
        }

        //
        // POST: /Account/Disassociate

        [HttpPost]
```

```
        [ValidateAntiForgeryToken]
        public ActionResult Disassociate(string provider, string providerUserId)
        {
            string ownerAccount = OAuthWebSecurity.GetUserName(provider, providerUserId);
            ManageMessageId? message = null;

            // Only disassociate the account if the currently logged in user is the owner
            if (ownerAccount == User.Identity.Name)
            {
                // Use a transaction to prevent the user from deleting their last login
credential
                using (var scope = new TransactionScope(TransactionScopeOption.Required,
new TransactionOptions { IsolationLevel = IsolationLevel.Serializable }))
                {
                    bool hasLocalAccount =
OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));
                    if (hasLocalAccount ||
OAuthWebSecurity.GetAccountsFromUserName(User.Identity.Name).Count > 1)
                    {
                        OAuthWebSecurity.DeleteAccount(provider, providerUserId);
                        scope.Complete();
                        message = ManageMessageId.RemoveLoginSuccess;
                    }
                }
            }

            return RedirectToAction("Manage", new { Message = message });
        }

        //
        // GET: /Account/Manage

        public ActionResult Manage(ManageMessageId? message)
        {
            ViewBag.StatusMessage =
                message == ManageMessageId.ChangePasswordSuccess ? "Your password has been
changed."
                : message == ManageMessageId.SetPasswordSuccess ? "Your password has been
set."
                : message == ManageMessageId.RemoveLoginSuccess ? "The external login was
removed."
                : "";
            ViewBag.HasLocalPassword =
OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));
            ViewBag.ReturnUrl = Url.Action("Manage");
            return View();
        }

        //
        // POST: /Account/Manage

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Manage(LocalPasswordModel model)
        {
            bool hasLocalAccount =
OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));
            ViewBag.HasLocalPassword = hasLocalAccount;
            ViewBag.ReturnUrl = Url.Action("Manage");
            if (hasLocalAccount)
            {
```

```
                if (ModelState.IsValid)
                {
                    // ChangePassword will throw an exception rather than return false in
certain failure scenarios.
                    bool changePasswordSucceeded;
                    try
                    {
                        changePasswordSucceeded =
WebSecurity.ChangePassword(User.Identity.Name, model.OldPassword, model.NewPassword);
                    }
                    catch (Exception)
                    {
                        changePasswordSucceeded = false;
                    }

                    if (changePasswordSucceeded)
                    {
                        return RedirectToAction("Manage", new { Message =
ManageMessageId.ChangePasswordSuccess });
                    }
                    else
                    {
                        ModelState.AddModelError("", "The current password is incorrect or
the new password is invalid.");
                    }
                }
            }
            else
            {
                // User does not have a local password so remove any validation errors
caused by a missing
                // OldPassword field
                ModelState state = ModelState["OldPassword"];
                if (state != null)
                {
                    state.Errors.Clear();
                }

                if (ModelState.IsValid)
                {
                    try
                    {
                        WebSecurity.CreateAccount(User.Identity.Name, model.NewPassword);
                        return RedirectToAction("Manage", new { Message =
ManageMessageId.SetPasswordSuccess });
                    }
                    catch (Exception)
                    {
                        ModelState.AddModelError("", String.Format("Unable to create local
account. An account with the name \"{0}\" may already exist.", User.Identity.Name));
                    }
                }
            }

            // If we got this far, something failed, redisplay form
            return View(model);
        }

        //
        // POST: /Account/ExternalLogin
```

```
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult ExternalLogin(string provider, string returnUrl)
        {
            return new ExternalLoginResult(provider, Url.Action("ExternalLoginCallback",
new { ReturnUrl = returnUrl }));
        }

        //
        // GET: /Account/ExternalLoginCallback

        [AllowAnonymous]
        public ActionResult ExternalLoginCallback(string returnUrl)
        {
            AuthenticationResult result =
OAuthWebSecurity.VerifyAuthentication(Url.Action("ExternalLoginCallback", new { ReturnUrl
= returnUrl }));
            if (!result.IsSuccessful)
            {
                return RedirectToAction("ExternalLoginFailure");
            }

            if (OAuthWebSecurity.Login(result.Provider, result.ProviderUserId,
createPersistentCookie: false))
            {
                return RedirectToLocal(returnUrl);
            }

            if (User.Identity.IsAuthenticated)
            {
                // If the current user is logged in add the new account
                OAuthWebSecurity.CreateOrUpdateAccount(result.Provider,
result.ProviderUserId, User.Identity.Name);
                return RedirectToLocal(returnUrl);
            }
            else
            {
                // User is new, ask for their desired membership name
                string loginData =
OAuthWebSecurity.SerializeProviderUserId(result.Provider, result.ProviderUserId);
                ViewBag.ProviderDisplayName =
OAuthWebSecurity.GetOAuthClientData(result.Provider).DisplayName;
                ViewBag.ReturnUrl = returnUrl;
                return View("ExternalLoginConfirmation", new RegisterExternalLoginModel {
UserName = result.UserName, ExternalLoginData = loginData });
            }
        }

        //
        // POST: /Account/ExternalLoginConfirmation

        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult ExternalLoginConfirmation(RegisterExternalLoginModel model,
string returnUrl)
        {
            string provider = null;
            string providerUserId = null;
```

```
            if (User.Identity.IsAuthenticated ||
!OAuthWebSecurity.TryDeserializeProviderUserId(model.ExternalLoginData, out provider, out
providerUserId))
            {
                return RedirectToAction("Manage");
            }

            if (ModelState.IsValid)
            {
                // Insert a new user into the database
                using (UsersContext db = new UsersContext())
                {
                    UserProfile user = db.UserProfiles.FirstOrDefault(u =>
u.UserName.ToLower() == model.UserName.ToLower());
                    // Check if user already exists
                    if (user == null)
                    {
                        // Insert name into the profile table
                        db.UserProfiles.Add(new UserProfile { UserName = model.UserName
});
                        db.SaveChanges();

                        OAuthWebSecurity.CreateOrUpdateAccount(provider, providerUserId,
model.UserName);
                        OAuthWebSecurity.Login(provider, providerUserId,
createPersistentCookie: false);

                        return RedirectToLocal(returnUrl);
                    }
                    else
                    {
                        ModelState.AddModelError("UserName", "User name already exists.
Please enter a different user name.");
                    }
                }
            }

            ViewBag.ProviderDisplayName =
OAuthWebSecurity.GetOAuthClientData(provider).DisplayName;
            ViewBag.ReturnUrl = returnUrl;
            return View(model);
        }

        //
        // GET: /Account/ExternalLoginFailure

        [AllowAnonymous]
        public ActionResult ExternalLoginFailure()
        {
            return View();
        }

        [AllowAnonymous]
        [ChildActionOnly]
        public ActionResult ExternalLoginsList(string returnUrl)
        {
            ViewBag.ReturnUrl = returnUrl;
            return PartialView("_ExternalLoginsListPartial",
OAuthWebSecurity.RegisteredClientData);
        }
```

```
        [ChildActionOnly]
        public ActionResult RemoveExternalLogins()
        {
            ICollection<OAuthAccount> accounts =
OAuthWebSecurity.GetAccountsFromUserName(User.Identity.Name);
            List<ExternalLogin> externalLogins = new List<ExternalLogin>();
            foreach (OAuthAccount account in accounts)
            {
                AuthenticationClientData clientData =
OAuthWebSecurity.GetOAuthClientData(account.Provider);

                externalLogins.Add(new ExternalLogin
                {
                    Provider = account.Provider,
                    ProviderDisplayName = clientData.DisplayName,
                    ProviderUserId = account.ProviderUserId,
                });
            }

            ViewBag.ShowRemoveButton = externalLogins.Count > 1 ||
OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));
            return PartialView("_RemoveExternalLoginsPartial", externalLogins);
        }

        #region Helpers
        private ActionResult RedirectToLocal(string returnUrl)
        {
            if (Url.IsLocalUrl(returnUrl))
            {
                return Redirect(returnUrl);
            }
            else
            {
                return RedirectToAction("Index", "Home");
            }
        }

        public enum ManageMessageId
        {
            ChangePasswordSuccess,
            SetPasswordSuccess,
            RemoveLoginSuccess,
        }

        internal class ExternalLoginResult : ActionResult
        {
            public ExternalLoginResult(string provider, string returnUrl)
            {
                Provider = provider;
                ReturnUrl = returnUrl;
            }

            public string Provider { get; private set; }
            public string ReturnUrl { get; private set; }

            public override void ExecuteResult(ControllerContext context)
            {
                OAuthWebSecurity.RequestAuthentication(Provider, ReturnUrl);
            }
        }
```

```
        private static string ErrorCodeToString(MembershipCreateStatus createStatus)
        {
            // See http://go.microsoft.com/fwlink/?LinkID=177550 for
            // a full list of status codes.
            switch (createStatus)
            {
                case MembershipCreateStatus.DuplicateUserName:
                    return "User name already exists. Please enter a different user
name.";

                case MembershipCreateStatus.DuplicateEmail:
                    return "A user name for that e-mail address already exists. Please
enter a different e-mail address.";

                case MembershipCreateStatus.InvalidPassword:
                    return "The password provided is invalid. Please enter a valid
password value.";

                case MembershipCreateStatus.InvalidEmail:
                    return "The e-mail address provided is invalid. Please check the value
and try again.";

                case MembershipCreateStatus.InvalidAnswer:
                    return "The password retrieval answer provided is invalid. Please
check the value and try again.";

                case MembershipCreateStatus.InvalidQuestion:
                    return "The password retrieval question provided is invalid. Please
check the value and try again.";

                case MembershipCreateStatus.InvalidUserName:
                    return "The user name provided is invalid. Please check the value and
try again.";

                case MembershipCreateStatus.ProviderError:
                    return "The authentication provider returned an error. Please verify
your entry and try again. If the problem persists, please contact your system
administrator.";

                case MembershipCreateStatus.UserRejected:
                    return "The user creation request has been canceled. Please verify
your entry and try again. If the problem persists, please contact your system
administrator.";

                default:
                    return "An unknown error occurred. Please verify your entry and try
again. If the problem persists, please contact your system administrator.";
            }
        }
        #endregion
    }
}
```

Bosh sahifa View

```
@model IEnumerable<TestOnMvc.Models.Category>

@{
    ViewBag.Title = "Index";
}

<nav style="width: 25%; min-height: 500px; border-right: 1px solid #151515; float: left;"
>
    <center><h1 style="width:50%; border-bottom:1px dotted #808080">Menu</h1></center>
    <ul class="expandable">
        <li>
            @Html.ActionLink("Bosh sahifa", "Index", "Home")

        </li>
        <li>
            <a href="#">Test</a>
            <ul class="expandable">
                <li>
                    <a href="#">Qo'shish</a>
                    <ul class="expandable">


                        @foreach (var item in Model)
                        {
                            <li>@Html.ActionLink(item.name, "Details", new { id =
Model.ToList()[0].ID })</li>
                        }

                    </ul>
                </li>
                <li>
                    <a href="#">Tahrirlash</a>
                    <ul class="expandable">
                        @foreach (var item in Model)
                        {
                            <li>@Html.ActionLink(item.name, "Details", new { id =
Model.ToList()[0].ID })</li>
                        }
                    </ul>
                </li>
                <li>
                    <a href="#">O'chirish</a>
                    <ul class="expandable">
                        @foreach (var item in Model)
                        {
                            <li>@Html.ActionLink(item.name, "Details", new { id =
Model.ToList()[0].ID })</li>
                        }
                    </ul>
                </li>

                <li>
                    <a href="#">Test yaratish</a>

                </li>
                <li>
                    <a href="#">Fanlar</a>

                </li>
```

```
                <li>
                    <a href="#">Natijalar</a>
                    <ul class="expandable">

                    </ul>
                </li>
            </ul>
        </li>
    </ul>
</nav>
<div id="content" style="margin-left: 25%;  width: 75%; padding: 0 15px; ">
    <center>
        <h2>Testlar ro'yhati</h2>

        <table width="100%">
            <tr>
                <th>
                    No
                </th>
                <th>
                    Test Mavzusi
                </th>
                <th>
                    Fan
                </th>
                <th>
                    Savollar soni
                </th>
                <th>
                    Tuzuvchi
                </th>
                <th></th>
            </tr>
            @{
                int i = 0;
            }
            @foreach (var item in Model)
            {
                <tr>
                    <td>
                        @{
                            i++;
                        }
                        @i
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.name)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.subject)
                    </td>
                    <td>
                        30
                    </td>
                    <td>
                        @if (i == 1)
                        {
                            <text>O'qituvchi</text>
                        }
                        else{
                            <text>admin</text>
```

```
                    }
                </td>
                <td>

                        @Html.ActionLink("Qo'shish", "Details", new { id = item.ID })
                    @Html.ActionLink("Tahrirlash", "Details", new { id = item.ID })
                    @Html.ActionLink("O'chirish", "Details", new { id = item.ID })


                </td>
            </tr>
        }

    </table>
</center>
</div>

Ro’yhatdan o’tish oynasi Login
@model TestOnMvc.Models.LoginModel

@{
    ViewBag.Title = "Ro'yhatdan o'tish";
}

<hgroup class="title">
    <h1>@ViewBag.Title.</h1>
</hgroup>

<section id="loginForm">
<h2>Test tizmga kirish.</h2>
@using (Html.BeginForm(new { ReturnUrl = ViewBag.ReturnUrl })) {
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Test tizimga kirish.</legend>
        <ol>
            <li>
                @Html.LabelFor(m => m.UserName)
                @Html.TextBoxFor(m => m.UserName)
                @Html.ValidationMessageFor(m => m.UserName)
            </li>
            <li>
                @Html.LabelFor(m => m.Password)
                @Html.PasswordFor(m => m.Password)
                @Html.ValidationMessageFor(m => m.Password)
            </li>
            <li>
                @Html.CheckBoxFor(m => m.RememberMe)
                @Html.LabelFor(m => m.RememberMe, new { @class = "checkbox" })
            </li>
        </ol>
        <input type="submit" value="Kirish" />
    </fieldset>
    <p>
        Ro'yhatdan o'tmagan bo'lsangiz, ushbu @Html.ActionLink("Register", "Register")
tugmasini bosing.
    </p>
}
</section>
```

```
<section class="social" id="socialLoginForm">
    <h2>Ro'yhatdan o'tish qoidalari.</h2>
    @Html.Action("ExternalLoginsList", new { ReturnUrl = ViewBag.ReturnUrl })
</section>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Registrasiya oynasi Register View

```
@model TestOnMvc.Models.RegisterModel
@{
    ViewBag.Title = "Register";
}

<hgroup class="title">
    <h1>@ViewBag.Title.</h1>
    <h2>Create a new account.</h2>
</hgroup>

@using (Html.BeginForm()) {
    @Html.AntiForgeryToken()
    @Html.ValidationSummary()

    <fieldset>
        <legend>Registration Form</legend>
        <ol>
            <li>
                <label for="name">Ism</label>
                <input name="name" />
            </li>
            <li>
                <label for="surename">Familiya</label>
                <input name="surename" />
            </li>
            <li>
                @Html.LabelFor(m => m.UserName)
                @Html.TextBoxFor(m => m.UserName)
            </li>
            <li>
                @Html.LabelFor(m => m.Password)
                @Html.PasswordFor(m => m.Password)
            </li>
            <li>
                <label for="email">Email</label>
                <input name="email" />
            </li>

        </ol>
        <input type="submit" value="Register" />
    </fieldset>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Test oynasi Index

```
@model IEnumerable<TestOnMvc.Models.TestAll>
```

```
@{
    ViewBag.Title = "Index";
    int testNumber = ViewBag.id;
    IEnumerable<TestOnMvc.Models.UserAnswer> answers =
(IEnumerable<TestOnMvc.Models.UserAnswer>)ViewBag.answerList;
}
@using (Html.BeginForm("Index", "TestAll")) {
<table width="99%" style="margin-top:-20px">
    <tbody>
        <tr>
            <th>
                <div class="content_0">
                    <div id="user_profile" style="float:left;margin:1px 6px;">
                        <h3>- Компьютер саводхонлиги (узб)</h3>
                    </div>
                    <div style="float:right">
                        <div class="timer" id="timer_1">00</div>
                        <div class="timer" id="timer_2">57</div><div class="timer"
id="timer_3">17</div>
                    </div>
                    <div id="user_profile">

                        <h3>
                            FIO:
                            Аллаберганова Насиба
                        </h3>
                    </div>
                </div>
            </th>
        </tr>
        <tr>
            <th>
                <div class="content_1">


                    @{
                        int i = 0;
                    }

   @foreach (var item in Model)
   {
                        i++;
                        string className = (i ==
testNumber)?"question_selected_2":"question_st";
                        @Html.ActionLink(i.ToString(), "Index", "TestAllController", null,
new { @class = className, onclick = "return false;" });

    }


                </div>
            </th>
        </tr>
        <tr>
            <th>

                <div id="testForm">
                    @{
                        if (testNumber == 0)
```

```
                        {
                            testNumber++;
                        }
                        string testType = (Model.ToList()[testNumber - 1].multiple) ?
"checkbox" : "radio";
                    }

                            <input id="id" type="hidden" name="id" value="5"/>
                            <input id="testId" type="hidden" name="testId"
value="@Model.ToList()[testNumber - 1].ID">
                            <input id="userId" type="hidden" name="userId" value="1">
                            <input id="time" type="hidden" name="time" value="">
                            <input id="date" type="hidden" name="date" value="27.05.2014">
                            <input id="testNumber" type="hidden" name="testNumber"
value="@testNumber" />
                            <table id="quizes">
                                <tbody>
                                    <tr>
                                        <th colspan="2">
                                            <p>
                                                @Model.ToList()[testNumber - 1].question
                                            </p>

                                            <p>
                                                @if (Model.ToList()[testNumber -
1].imageUrl != null)
                                                {<text><img
src="@Model.ToList()[testNumber - 1].imageUrl"></text>}
                                            </p>
                                        </th>
                                    </tr>
                                    <tr>
                                        <td width="1%">
                                        @{
                                            int ansRadio = 0;
                                            bool saved = false;
                                            foreach(var ans in answers){
                                                if (ans.testId ==
@Model.ToList()[testNumber - 1].ID)
                                                {
                                                    saved = true;
                                                    ansRadio = ans.answered;
                                                }

                                            }
                                        }
                                            <input type="@testType"
onclick="radio_check(this)" id="hovered_0" value="1" name="answered"
                                            @if(saved && ansRadio==1){
<text>checked="checked"</text>}>
                                        </td>
                                        <td valign="top"
onclick="checked_element(&amp;#39;hovered_0&amp;#39;,&amp;#39;radio&amp;#39;);">
                                            <p>
                                                @Model.ToList()[testNumber - 1].answer1
                                            </p>
                                        </td>
                                    </tr>
                                    <tr>
                                        <td width="10"><input type="@testType"
onclick="radio_check(this)" id="hovered_1" value="2" name="answered"
```

```
                                                    @if (saved && (ansRadio == 2)) {
<text> checked="checked" </text> }>
                                            </td>
                                            <td valign="top"
onclick="checked_element(&#39;hovered_1&#39;,&#39;radio&#39;);">
                                                <p>
                                                    @Model.ToList()[testNumber - 1].answer2
                                                </p>
                                            </td>
                                        </tr>
                                        @if (Model.ToList()[testNumber - 1].answer3 != null)
                                        {
                                            <tr>
                                                <td width="10"><input type="@testType"
onclick="radio_check(this)" id="hovered_2" value="3" name="answered"
                                                    @if (saved && (ansRadio == 3)) { <text>
checked="checked" </text> }>
                                                </td>
                                                <td valign="top"
onclick="checked_element(&#39;hovered_2&#39;,&#39;radio&#39;);">
                                                    <p>
                                                        @Model.ToList()[testNumber -
1].answer3

                                                    </p>
                                                </td>
                                            </tr>
                                        }
                                        @if (Model.ToList()[testNumber - 1].answer4 != null)
{
    <tr>
        <td width="10">
            <input type="@testType" onclick="radio_check(this)" id="hovered_3" value="4"
name="answered"
                   @if (saved && ansRadio == 4) { <text> checked="checked" </text>  }>
        </td>
        <td valign="top" onclick="checked_element(&#39;hovered_3&#39;,&#39;radio&#39;);">
            <p>
                @Model.ToList()[testNumber - 1].answer4
            </p>
        </td>
    </tr>
}
                                        @if (Model.ToList()[testNumber - 1].answer5 != null)
{
    <tr>
        <td width="10">
            <input type="@testType" onclick="radio_check(this)" id="hovered_4" value="5"
name="answered"
                   @if (saved && ansRadio == 5) { <text> checked="checked" </text>  }>
        </td>
        <td valign="top" onclick="clicked(&#39;hovered_4&#39;,&#39;radio&#39;);">
            <p>
                @Model.ToList()[testNumber - 1].answer5
            </p>
        </td>
    </tr>
}
                                        <tr>
                                            <th colspan="2">
                                                <input type="hidden" id="qolgan_vaqt"
name="qolgan_vaqt" value="3600">
```

```
                                                <input type="submit" value="Saqlash"
name="saqlash" onclick="clicked(@testNumber)">
                                                <input type="hidden" value="1"
name="question">
                                                <input style="float:right;left:-1em;"
type="button" value="Finish" onclick="if(confirm(&#39;Testni yakunlashni
hohlaysizmi?&#39;))location.href=&#39;?category=check_answers&#39;">

                                            </th>
                                        </tr>
                                    </tbody>
                                </table>

                            </div>
                        </th>
                    </tr>
                    <tr>
                        <td align="center" style="color:red">
                            <br>
                            Test savollariga o'tishda CHAP va O'NG tugmalarni, javoblarni
belgilashda esa 1 dan 5 gacha tugmalarni ishlatishingiz mumkin.
                            <br>
                            To Use: RIGHT button- Next question, LEFT button - previous question
and for apply answers from 1 to 5 numbers!
                        </td>
                    </tr>
                </tbody>
            </table>
}
Test yaratish oynasi Create View

@model TestOnMvc.Models.TestAll

@{
    ViewBag.Title = "Test yaratish";
}

<h2>Test yaratish</h2>

<h1>Mavzu: Компьютер саводхонлиги (узб)
             
Tuzuvchi: O'qituvchi</h1>

@using (Html.BeginForm()) {
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)


    <table width="90%" style="padding-left:15px">
        <tr>
            <td>Savol</td>
            <td><textarea cols="15"></textarea></td>
        </tr>
        <tr>
            <td>Rasm</td>
            <td><input type="file"  /></td>
        </tr>
        <tr>
            <td>Qiyinlik darajasi</td>
            <td><select>
                <option>1 (yengil)</option>
```

```
                    <option>2 (o'rta)</option>
                    <option>3 (qiyin)</option>
                </select></td>
        </tr>
        <tr>
            <td>Test javobi 1</td>
            <td><div style="width:20px; padding-left:5px; height:30px; margin-top:6px;
border:1px solid black; float:left"><input type="checkbox" /></div><input type="text"
style="margin-left:10px" /></td>
        </tr>
        <tr>
            <td>Test javobi 2</td>
            <td><div style="width:20px; padding-left:5px; height:30px; margin-top:6px;
border:1px solid black; float:left"><input type="checkbox" /></div><input type="text"
style="margin-left:10px" /></td>
        </tr>
        <tr>
            <td>Test javobi 3</td>
            <td><div style="width:20px; padding-left:5px; height:30px; margin-top:6px;
border:1px solid black; float:left"><input type="checkbox" /></div><input type="text"
style="margin-left:10px" /></td>
        </tr>
        <tr>
            <td>Test javobi 4</td>
            <td><div style="width:20px; padding-left:5px; height:30px; margin-top:6px;
border:1px solid black; float:left"><input type="checkbox" /></div><input type="text"
style="margin-left:10px" /></td>
        </tr>
        <tr>
            <td rowspan="2"><a href="#">Javob varianti qo'shish</a></td>
</tr>
        <tr>
            <td rowspan="2"><input type="submit" value="Create" /></td>
        </tr>



    </table>


}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Testni tahrirlash oynasi

```
@model TestOnMvc.Models.TestAll

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>

@using (Html.BeginForm()) {
```

```
@Html.AntiForgeryToken()
@Html.ValidationSummary(true)

<fieldset>
    <legend>TestAll</legend>

    @Html.HiddenFor(model => model.ID)

    <div class="editor-label">
        @Html.LabelFor(model => model.question)
    </div>
    <div class="editor-field">
        @Html.EditorFor(model => model.question)
        @Html.ValidationMessageFor(model => model.question)
    </div>

    <div class="editor-label">
        @Html.LabelFor(model => model.range)
    </div>
    <div class="editor-field">
        @Html.EditorFor(model => model.range)
        @Html.ValidationMessageFor(model => model.range)
    </div>

    <div class="editor-label">
        @Html.LabelFor(model => model.multiple)
    </div>
    <div class="editor-field">
        @Html.EditorFor(model => model.multiple)
        @Html.ValidationMessageFor(model => model.multiple)
    </div>

    <div class="editor-label">
        @Html.LabelFor(model => model.correct1)
    </div>
    <div class="editor-field">
        @Html.EditorFor(model => model.correct1)
        @Html.ValidationMessageFor(model => model.correct1)
    </div>

    <div class="editor-label">
        @Html.LabelFor(model => model.answer1)
    </div>
    <div class="editor-field">
        @Html.EditorFor(model => model.answer1)
        @Html.ValidationMessageFor(model => model.answer1)
    </div>

    <div class="editor-label">
        @Html.LabelFor(model => model.correct2)
    </div>
    <div class="editor-field">
        @Html.EditorFor(model => model.correct2)
        @Html.ValidationMessageFor(model => model.correct2)
    </div>

    <div class="editor-label">
        @Html.LabelFor(model => model.answer2)
    </div>
    <div class="editor-field">
        @Html.EditorFor(model => model.answer2)
```

```
    @Html.ValidationMessageFor(model => model.answer2)
</div>

<div class="editor-label">
    @Html.LabelFor(model => model.correct3)
</div>
<div class="editor-field">
    @Html.EditorFor(model => model.correct3)
    @Html.ValidationMessageFor(model => model.correct3)
</div>

<div class="editor-label">
    @Html.LabelFor(model => model.answer3)
</div>
<div class="editor-field">
    @Html.EditorFor(model => model.answer3)
    @Html.ValidationMessageFor(model => model.answer3)
</div>

<div class="editor-label">
    @Html.LabelFor(model => model.correct4)
</div>
<div class="editor-field">
    @Html.EditorFor(model => model.correct4)
    @Html.ValidationMessageFor(model => model.correct4)
</div>

<div class="editor-label">
    @Html.LabelFor(model => model.answer4)
</div>
<div class="editor-field">
    @Html.EditorFor(model => model.answer4)
    @Html.ValidationMessageFor(model => model.answer4)
</div>

<div class="editor-label">
    @Html.LabelFor(model => model.correct5)
</div>
<div class="editor-field">
    @Html.EditorFor(model => model.correct5)
    @Html.ValidationMessageFor(model => model.correct5)
</div>

<div class="editor-label">
    @Html.LabelFor(model => model.answer5)
</div>
<div class="editor-field">
    @Html.EditorFor(model => model.answer5)
    @Html.ValidationMessageFor(model => model.answer5)
</div>

<div class="editor-label">
    @Html.LabelFor(model => model.categoryId)
</div>
<div class="editor-field">
    @Html.EditorFor(model => model.categoryId)
    @Html.ValidationMessageFor(model => model.categoryId)
</div>

<p>
    <input type="submit" value="Save" />
```

```
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}

Testni o’chirib tashlash oynasi Delete
@model TestOnMvc.Models.TestAll

@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>
<h3>Are you sure you want to delete this?</h3>
<fieldset>
    <legend>TestAll</legend>

    <div class="display-label">
         @Html.DisplayNameFor(model => model.question)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.question)
    </div>
    <div class="display-label">
         @Html.DisplayNameFor(model => model.range)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.range)
    </div>
    <div class="display-label">
         @Html.DisplayNameFor(model => model.multiple)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.multiple)
    </div>
    <div class="display-label">
         @Html.DisplayNameFor(model => model.correct1)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.correct1)
    </div>

    <div class="display-label">
         @Html.DisplayNameFor(model => model.answer1)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.answer1)
    </div>

    <div class="display-label">
         @Html.DisplayNameFor(model => model.correct2)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.correct2)
```

```
    </div>

    <div class="display-label">
         @Html.DisplayNameFor(model => model.answer2)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.answer2)
    </div>
    <div class="display-label">
         @Html.DisplayNameFor(model => model.correct3)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.correct3)
    </div>
    <div class="display-label">
         @Html.DisplayNameFor(model => model.answer3)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.answer3)
    </div>
    <div class="display-label">
         @Html.DisplayNameFor(model => model.correct4)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.correct4)
    </div>
    <div class="display-label">
         @Html.DisplayNameFor(model => model.answer4)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.answer4)
    </div>
    <div class="display-label">
         @Html.DisplayNameFor(model => model.correct5)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.correct5)
    </div>
    <div class="display-label">
         @Html.DisplayNameFor(model => model.answer5)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.answer5)
    </div>
    <div class="display-label">
         @Html.DisplayNameFor(model => model.categoryId)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.categoryId)
    </div>
</fieldset>
@using (Html.BeginForm()) {
    @Html.AntiForgeryToken()
    <p>
        <input type="submit" value="Delete" /> |
        @Html.ActionLink("Back to List", "Index")
    </p>
}
```