**ԲՈՎԱՆԴԱԿՈՒԹՅՈՒՆ**

Ներածություն-------------------------------------------4

1. Ընդհանուր մաս-----------------------------------------6

1.1. Ծրագրավորման լեզվի նկարագրություն-------------------6

1.2. Ուսումնառության նյութեր------------------------------------9

1.3. Համեմատական վերլուծություն------------------------------11

2. Հատուկ մաս---------------------------------------------13

2.1. Թեմա-------------------------------------------------------------13

2.2. Խնդրի դրվածքը-------------------------------------------------26

2.3. Ալգորիթմի նկարագրությունը---------------------------------27

2.4. Ծրագիր-----------------------------------------------------------37

2.5. Ծրագրի բացատրությունը--------------------------------------47

2.6. Ծրագրի օգտագործման տեխնոլոգիան-----------------------62

3. Անվտանգության տեխնիկայի կանոնները------------75

Եզրակացություն------------------------------------------76

Գրականություն-------------------------------------------77

**Ներածություն**

Վերջին տասնամյակներում տեղեկատվական տեխնոլոգիաների արագ զարգացման արդյունքում մարդու գործունեության բոլոր ոլորտները զգալի փոփոխությունների են ենթարկվել։ Այս համատեքստում կրթության ոլորտը դարձավ այն կարևորագույն հատվածներից մեկը, որտեղ նորարարական տեխնոլոգիաների կիրառումը հնարավորություն տվեց վերանայել դասավանդման և ուսումնառության ավանդական մեթոդները՝ ստեղծելով ավելի արդյունավետ և ճկուն համակարգեր:

Ժամանակակից կրթական գործընթացը պահանջում է նորագույն գործիքներ, որոնք կապահովեն ուսուցման մատչելիություն, բազմակողմանի հաղորդակցում և արդյունավետ կառավարման մեխանիզմներ: Դասապրոցեսի թվայնացումն արդեն դարձել է ոչ միայն տեխնիկական հարմարության միջոց, այլ նաև անհրաժեշտություն՝ պայմանավորված կրթության որակի բարձրացման, ուսումնական գործընթացի անհատականացման և արդյունավետության ապահովման հրամայականով։

Կրթական հարթակների ստեղծումն այսօր դիտարկվում է որպես արդյունավետ միջոց՝ վերացնելու կրթության կազմակերպման հետ կապված բազմաթիվ խնդիրներ, ինչպիսիք են հեռավորության պատճառով առաջացող խոչընդոտները, անհատական մոտեցման բացակայությունը կամ տեղեկատվության փոխանցման դանդաղ գործընթացները։ Թվային հարթակները հնարավորություն են տալիս միավորել ուսումնական նյութերը, առաջադրանքները և գնահատականները մեկ համակարգում՝ ուսուցման բոլոր մասնակիցների համար ստեղծելով միասնական, կառուցվածքային և ինտերակտիվ միջավայր։

Վերջին տարիներին, հատկապես համաշխարհային մարտահրավերների, ինչպես օրինակ COVID-19 համավարակի պայմաններում, դրսևորվեց նման հարթակների կարևորությունը: Շատ ուսումնական հաստատություններ ստիպված եղան կարճ ժամանակում անցում կատարել հեռավար ուսուցման, ինչի արդյունքում ի հայտ եկավ նոր պահանջ՝ ունենալ առավել հուսալի, ինտուիտիվ և բազմաֆունկցիոնալ կրթական համակարգեր։ Այս միտումը առավել ուժեղացրեց կրթության ոլորտում թվային լուծումների դերը՝ հաստատելով դրանց կենսունակությունն ու անփոխարինելիությունը ժամանակակից աշխարհում։

Ներկայիս դիպլոմային աշխատանքի շրջանակներում ստեղծվել է ուսումնական գործընթացի կազմակերպման հարթակ, որի նպատակն է ուսուցչի, ուսանողի և ուսումնական հաստատության ադմինիստրատորի գործունեությունը կազմակերպել մեկ միասնական միջավայրում։ Հարթակը հնարավոր է կիրառել ինչպես ավանդական ուսումնական հաստատություններում, այնպես էլ մասնագիտական կենտրոններում կամ անհատական ուսուցման համակարգերում, քանի որ կայքի տրամաբանությունը սերվերային մակարդակում կարող է փոփոխվել ու հարմարեցվել ըստ պահանջի, իսկ ներկայումս այն գործում է որոշակի ընդհանրացված կարգավորումներով։

Միջազգային փորձը ցույց է տալիս, որ թվային կրթական հարթակների ներդրումը զգալիորեն բարելավում է ուսանողների արդյունքները, նպաստում է նրանց մոտիվացիայի բարձրացմանը և կրթական գործընթացի հանդեպ հետաքրքրության աճին։ Ավելին, արդյունավետ կառավարվող կրթական հարթակները հնարավորություն են տալիս ուսուցիչներին կենտրոնանալ բովանդակության ստեղծման և ուսուցման որակի վրա՝ նվազեցնելով կազմակերպչական բեռը։

Այս դիպլոմային աշխատանքի ընթացքում կքննարկվեն հարթակի նախագծման և մշակման հիմնական փուլերը, օգտագործված տեխնոլոգիաները, համակարգի ճարտարապետությունը և դրա գործառնական առանձնահատկությունները։ Կարևոր տեղ է տրվելու նաև հարթակի անվտանգությանը, տվյալների պահպանության և արդյունավետության բարձրացման մեթոդներին։

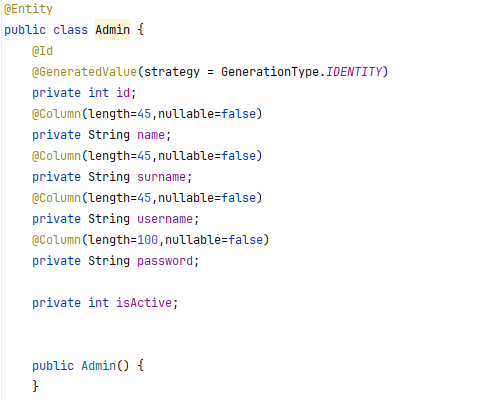
Աշխատանքի վերջնական նպատակը կրթական միջավայրի համար գործնականում կիրառելի, օգտագործման համար հարմար և ժամանակակից պահանջներին համապատասխանող հարթակի ստեղծումն է, որը կնպաստի ուսուցման գործընթացի բարելավմանը և կառավարման մեխանիզմների օպտիմալացմանը։

1. **ԸՆԴՀԱՆՈՒՐ ՄԱՍ**

**1.1** **Ծրագրավորման լեզվի նկարագրություն**

Դիպլոմային աշխատանքի համար որպես հիմնական տեխնոլոգիա ընտրվել է ջավա (**Java**) ծրագրավորման լեզուն, որն օբյեկտ–կողմնորոշված լեզու է և լայնորեն կիրառվում է վեբ ծրագրավորման, տվյալների կառավարման համակարգերում և մի շարք այլ ուղղություններում։ Java-ն ունի բազմաթիվ առավելություններ, որոնք այն դարձնում են շատ հարմար լեզու կրթական հարթակի մշակման համար, ներառյալ այն, որ ջավան ապահովում է բարձր մակարդակի անվտանգություն ու պլատֆորմային անկախություն։

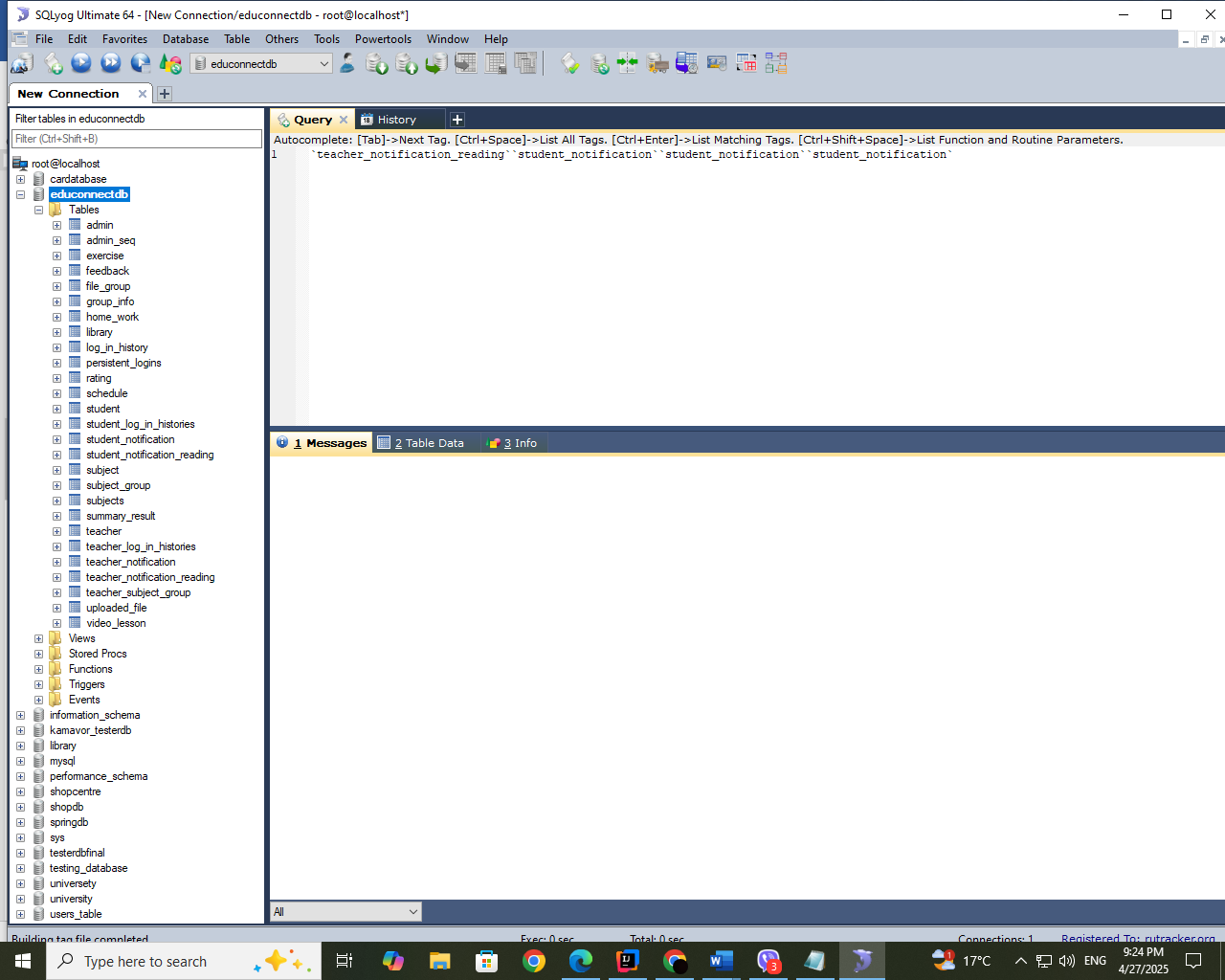
Java-ն բացարձակապես օբյեկտ-կողմնորոշված լեզու է, որտեղ հիմնական տարրերն են դասերը և օբյեկտները։ Դասերը պարունակում են դաշտեր ու հատկություններ և նրանց հետ որոշակիորեն փոխկապակցված մեթոդներ։ Օբյեկտները հանդիսանում են դասի օրինակներ, որոնք ստեղծվում են ծրագրի ընթացքում։

Ջավան ունի մի շարք պատրաստի դասեր, որոնք թույլ են տալիս մաքսիմալ օպտիմալացնել աշխատանքը։ Բացի այդ, **Spring Boot** տեխնոլոգիան (Framework), որն օգտագործվում է այս նախագծում, առաջարկում է ուժեղ միջավայր՝ լիարժեք դինամիկ և անվտանգ համակարգերի մշակման համար։ **Hibernate ORM** (Օբյեկտ-Ռելացիոն Համապատասխանեցում) բաց կոդով գրադարանն էլ ապահովում է տվյալների բազայի հետ հեշտ ինտեգրումը՝ հարմարեցնելով տվյալները օբյեկտ-կողմնորոշված մոդելին։ Այս տեխնոլոգիաների ներդրմամբ ջավա լեզվում իմ ունեցած դասերը վերածվել են հատուկ **Entity**(բառացի՝ օբյեկտ) –ների, այսինքն՝ տվյալների բազայի ամեն աղյուսակին կոդում բաժին է հասնում մեկ դաս և մեկ ինտերֆեյս(**Interface**), իսկ տվյալների հետ աշխատանքը շատ ավելի դյուրին է դառնում (նկար 1. 1)։

Ընտրելով այս տեխնոլոգիաները՝ մշակել եմ կրթական հարթակի բոլոր կարևորագույն ֆունկցիոնալ հնարավորությունները, որոնց թվում են օգտատերերի՝ հարթակ մուտք գործելը, անվտանգության ստուգումները, տվյալների մշակման արագությունը և այլն։

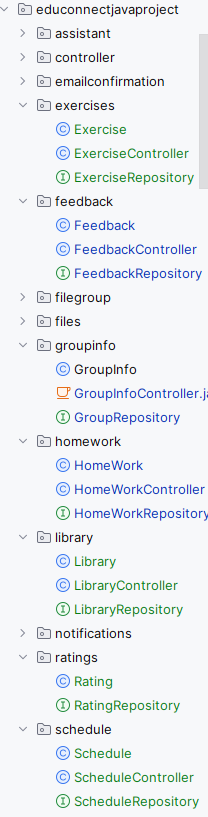
Նախագծում տվյալների պահպանման և կառավարման համար օգտագործվել է **MySQL** տվյալների բազան, որը շատ հզոր և արդյունավետ համակարգ է: **MySQL**-ը բաց կոդով տվյալների բազա է, որն օգտագործվում է բազմաթիվ ծրագրերում: Այն հիմնականում ծառայություններ է մատուցում **SQL (Structured Query Language)** լեզվով, որն էլ թույլ է տալիս շատ հեշտությամբ կատարել հարցումներ, ավելացնել, փոփոխել կամ հեռացնել տվյալներ: Ի շնորհիվ Hibernate տեխնոլոգիայի, տվյալների բազայում աղյուսակների կառուցվածքը (նկար 1․2) բավականին նման է պրոեկտում ունեցածս դասերի տրամաբանությանը։

Նկար 1․1



Նկար 1․2

Ծրագրի ստեղծման ընթացքում կիրառվել է **MVC (Model-View-Controller)** մոդելը։ Spring boot -ի պարագայում Model –ների դերը ստանձնում են ինտերֆեյսերը, ինչպես վերը նշվեց, իսկ Controller -ները պիտի ստեղծվեն Entity -ների հետ միասին՝ կոդի գրագետ բաժանվածություն ունենալու համար։ Նման կառուցվածքային բաժանման արդյունքում պրոեկտի կառուցվածքը կունենա նկար 1․3 –ում ցուցադրված տեսքը։



Նկար 1․3

Դե իսկ բազայից անհրաժեշտ հարցումները կատարելուց հետո օգտագործվել են **Html**, **Css**, **Js**, **Bootstrap**, **Thymeleaf** տեխնոլոգիաները, կայքի գեղեցիկ արտաքին տեսք ստանալու համար։ Thymeleaf -ի կիրառմամբ Html ֆայլը կունենա նման տեսք(նկար1․4)։



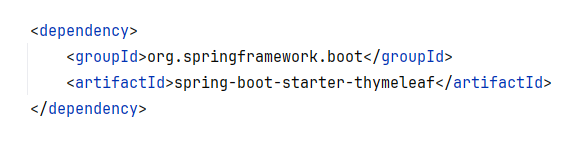
Նկար 1․4

Նկարում երևացող Html ֆայլի թեգերի մեջ գրվող **th** ընդլայնումը Thymleaf –ի կողմից առաջարկվող հիմնական գործիքն է, որի շնորհիվ կարող ենք Controller -ից ստացած տեղեկատվությունը ցուցարդել մեր էջում։ Html ֆայլում Thymeleaf ‑ի ինտեգրման համար անհրաժեշտ է ուղղակի ֆայլի առաջին տողում ունենալ հետևյալ գրելաձևը (նկար 1․5)՝



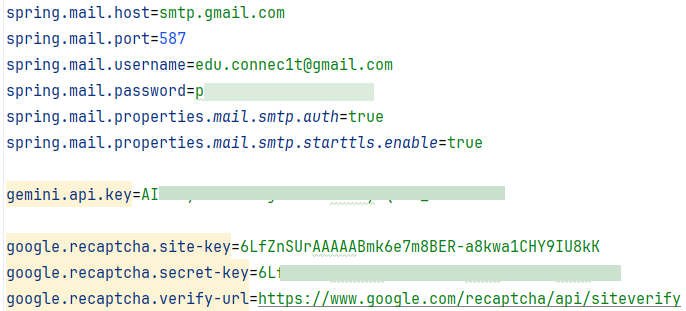
Նկար 1․5

Ինչպես նաև պրոեկտի **pom.xml** ֆայլում ավելացնել անհրաժեշտ կախվածությունները(նկար 1․6)



Նկար 1․6

MVC -ից զատ, նաև մի շարք արտաքին ծառայություններից օգտվելու և դրանք պրոեկտում ինտեգրելու համար՝ կիրառել եմ **API** (Application Programming Interface) հարցումների հասկացությունը, որով օրինակ՝ ինտեգրվել են էլեկտրոնային հաղորդագրություններ ուղարկելու և մի շարք այլ արտաքին ծառայություններ (նկար 1․7)։



Նկար 1․7

**1.2** **Ուսումնառության նյութեր**

Ի սկզբանե կարևորելով կայքի անվտանգության ապահովումը, առանձնահատուկ ուշադրություն է դարձվել **Spring Boot** տեխնոլոգիայի շրջանակներում տրամադրվող անվտանգության գործիքներին։ Մասնավորապես, խորությամբ ուսումնասիրվել է **Spring Security** մեխանիզմը, որն ապահովում է հուսալի նույնականացում (authentication) և թույլտվությունների կառավարում (authorization)։

Spring Security-ի միջոցով հնարավոր է դարձել իրականացնել օգտատերերի գրանցման, մուտքի, իրավասությունների սահմանման և պաշտպանված ռեսուրսների կառավարման գործընթացներ։ Անվտանգության հետ մեկտեղ, կարևորվել է օգտատերերի համար առավել հարմարավետ պայմաններ ստեղծելու հեռանկարը՝ մասնավորապես այն, որ ամեն անգամ համակարգից օգտվելիս մարդ ստիպված չլինի մուտք գործել, և դրա համար ուսումնասիրել ու Spring Security -ի **Remember me** ֆունկցիոնալության հետ համադրել եմ **Cookie** –ների ընձեռած հնարավորությունները՝ տալով օգտատերերին մուտք գործելուց հետո 30 օր համակարգում մնալու հնարավորություն։ Էջերի թույլտվությունների ստուգման համար կարևոր էր նաև **Session** –ների ուսումնասիրումն ու կիրառումը, ինչը ևս իրականացվել է։ Ուսումնասիրման հիմնական աղբյուրները հանդիսացել են Spring Boot ‑ի պաշտոնական կայքի և git համակարգում մասնավոր մարդկանց կողմից ցուցադրված մի շարք կոդային օրինակները։

Անվտանգության բարձրացմանն ուղղված հաջորդ քայլը եղել է **ReCaptcha** ծառայության աշխատանքի ուսումնասիրումն ու ներդրումը, որը որոշ չափով կարող է կանխարգելել և թույլ չտալ ռոբոտների մուտքը համակարգ։

Անվտանգությունից զատ, ստեղծված կայքում շատ մեծ կարևորություն ունի նաև արհեստական բանականությամբ աշխատող օգնականը, որը տիրապետում է հարթակի վերաբերյալ մեծածավալ ինֆորմացիայի և կարողանում է արդյունավետորեն կատարել աջակցման բաժնի աշխատողների գործառույթները։ Այս օգնականի ներդրումը հնարավորություն է տալիս նվազեցնել մարդկային ռեսուրսների բեռը և արագացնել օգտատերերի հարցումների սպասարկումը։ Դրա իրագործման համար խորությամբ ուսումնասիրվել է **API** հարցումների կառուցվածքը, տվյալների փոխանակման ձևաչափերը (հատկապես **JSON** և **RESTful API**-ներ), ինչպես նաև՝ արհեստական բանականության մոդելների ստեղծման, նախնական տվյալներով մարզման և դրանք արդեն գոյություն ունեցող համակարգերի հետ ինտեգրելու գործընթացը։

Մարզման համար օգտագործվել է իմ կողմից կազմված տեղեկատվական բազան, որը ներառում է կրթական հարթակի օգտագործման կանոնները, հնարավորությունների նկարագրությունները, ինչպես նաև հաճախ տրվող հարցերի պատասխանները։ Ինտեգրման և աշխատանքի արդյունավետության բարձրացման նպատակով ուսումնասիրվել են նաև **Gemini**-ի վերջին թարմացումները և **fine-tuning** (գոյություն ունեցող մոդելի ադապտացում)-ի առանձնահատկությունները, ինչի արդյունքում հնարավոր է դարձել ավելի ճկուն և ադապտիվ համակարգ ստեղծել։ Gemini-ի հնարավորությունները թույլ տվեցին լրացուցիչ հարստացնել օգնականի պատասխանները՝ դարձնելով դրանք առավել համապարփակ և օգտատիրոջ հարցմանը համապատասխան։ Իհարկե դրան զուգահեռ անհրաժեշտ էր ուսումնասիրել նաև **JavaScript** -ի որոշակի նրբություններ, որպեսզի օգտվողը ուղարկեր ու ստանար նամակներ ակնթարթորեն, առանց էջի լրացուցիչ թարմացումների։ Դրա նպատակով ուսումնասիրվել ու կիրառվել է **AJAX** (Asynchronous JavaScript and XML) տեխնոլոգիան (նկար 1․8):



Նկար 1․8

Ջավա լեզվում մեծ թվով դասերի հետ աշխատելիս բավականին ժամանակատար պրոցես կարող էր լինել ամեն դասի համար **հատկությունների** (Getter,Setter) և **կառուցիչների** (Constructor) սահմանումը, ինչի նպատակով ուսումնասիրվեց **Lombok** գրադարանը, որը ինչպես պարզվեց տրամադրում է մի շարք անոտացիաներ, վերը նշված խնդիրների արդյունավետ լուծման համար։ Օրինակ՝ **@AllArgsConstructor** անոտացիան փոխարինում է բոլոր արգումենտներով կոնստրուկտորին և այսպես շարունակ։ Սրա մասին տեղեկատվությունը ևս առկա է git -ում, ու համացանցի մի շարք այլ բաց աղբյուրներում։

Ուսումասիրության ևս մեկ շատ կարևոր առիթ էր էլեկտրոնային հասցեին համակարգից ուղղվող նամակների մեխանիզմը։ քանի որ հարթակում չկա 0–ից գրանցում, այլ կա օգտահաշվի ակտիվացում նախապես տրամադրված էլ․ փոստի միջոցով, ուստի կարևոր էր էլեկտրոնային փոստին ուղղվող հաղորդագրությունների մեխանիզմը գործարկել, ինչը API հարցումների կառուցվածքի ուսումնասիրությունից հետո ընդամենը տեխնիկայի հարց էր, ու **Gmail** –ի տրամադրած API բանալու շնորհիվ հարցը կարգավորվեց։ Այդ ինտերգման մասին մանրամասն ինֆորմացիան ու անհրաժեշտ քայլերի հաջորդականությունը կարելի է գտնել Google -ի պաշտոնական էջերում և փաստաթղթերում։

**1.3** **Համեմատական վերլուծություն**

Համեմատական վերլուծության նպատակն է ուսումնասիրել շուկայում արդեն առկա նմանատիպ գործիքները ու հասկանալ իմ կողմից մշակված համակարգի առավելություններն ու թերությունները։ Կրթական ոլորտում լայն ծավալով կիրառվող ծրագրերն են՝ **Microsoft Teams For Education**, **Google Classroom**, **Emis.am** և այլն: Դասապրոցեսի կազմակերպման տեսանկյունից եթե հայացք գցենք մշակված հարթակին ու այս համակարգերին, ապա կստանանք մոտավորապես այս պատկերը (աղյուսակ 1․1)։

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Հարթակ | **Google Classroom** | **Microsoft Teams** | **Emis.am** | **EduConnect(իմ կողմից մշակված)** |
| **Էլ․ մատյան** | ✅ | ✅ | ✅ | ✅ |
| **Էլ․ օրագիր** | ✅ | ✅ | ✅ | ✅ |
| **Ամփոփիչ արդյունքների ձևավորում** | ❌ | ❌ | ✅ | ✅ |
| **Ճկուն դասացուցակ** | ❌ | ❌ | ✅ | ✅ |
| **Կայքում օգնական՝ ԱԲ -ի գործիքներով** | ❌ | ❌ | ❌ | ✅ |
| **Առաջադրանքների առցանց մեխանիզմ** | ✅ | ✅ | ❌ | ✅  Աղյուսակ 1․1 |

Վերլուծության արդյունքում իրավիճակը հետևյալն է՝ Google Classroom -ում ավանդական մատյան ու օրագիր որպես այդպիսին չկա, բայց կա առաջադրանքների գնահատման հնարավորություն, ինչը մասամբ փոխարինում է վերը նշածին ու բավարարում պահանջը, սակայն ամփոփիչ արդյունքների ձևավորում առաջադրանքների գնահատականներով չի կարող իրականացվել, հետևաբար ամփոփիչ արդյունքների կետը ևս չի բավարարվում։ Դասացուցակի հետ կապված որոշակի սահմանափակված գործառույթներ օգտատերերին Classroom -ը տալիս է, սակայն այնտեղ դասացուցակը լրացնողը ուսուցիչն է իր դասավանդած խմբի համար, բացի այդ էլ չկա կիսամյակի և մոդուլի ընտրության հնարավորություններ՝ հետևաբար այս կետն էլ չի բավարարվում։ Թեպետ google ընկերությունն ունի արհեստական բանականության գործիքներ, սակայն անհատապես այս ծրագրի համար մարզած ու ադապտացված աբ(**արհեստական բանականություն**) օգնական չկա, իսկ առաջադրանքների կատարման մեխանիզմ Classroom -ն ունի։ Microsoft Teams -ում իրավիճակը գրեթե նույնն է, ինչ նախորդ օրինակում, բայց այստեղ դասացուցակին վերաբերող ընդհանրապես ոչ մի ֆունկցիոնալություն չկա, այլ կա օրացույցի բաժին(Calendar), որը երբեմն մարդիկ ստիպված են լինում հարմարեցնել ու կիրառել որպես դասացուցակ։ Emis.am -ի դեպքում իրավիճակը մի փոքր այլ է․ այն սպասարկելով ամբողջ ՀՀ -ի կրթական ցանցը, ունի շատ լավ կառուցված էլեկտրոնային մատյան, օրագիր, ամփոփիչ արդյունքների ձևավորում, ճկուն դասացուցակ և ուսումնական հաստատությունների կառավարման մի շարք այլ գործիքներ, բայց արհեստական բանականությամբ աշխատող օգնականին այնտեղ փոխարինում է **support.emis.am** առանձին կայքը, և կան մի շարք աշխատողներ, որպեսզի պատասխանեն մարդկանց և ուղղություն ցույց տան կայքում, ինչը էապես մեծացնում է կիրառվող ռեսուրսների ծավալը, իսկ մեր պարագայում մարդկային գործոնը փոխարինվում է արհեստական ինտելեկտով։ Առաջադրանքների կատարման մեխանիզմ emis.am -ում ևս չկա։

Ամփոփելով վերը նշվածը հասկանում ենք, որ տեղական ցանցում(local համակարգ) ուսումնական հաստատությանը սպասարկելու համար մեր հարթակն ունի առաջին անհրաժեշտության **բոլոր կարևոր գործիքները**, ունի մի շարք հնարավորություններ, որոնք դարձնում են համակարգը եզակի և պահանջված ու հետագայում էլ կարող է ընլայնվել ու բարելավվել։

**2․ ՀԱՏՈՒԿ ՄԱՍ**

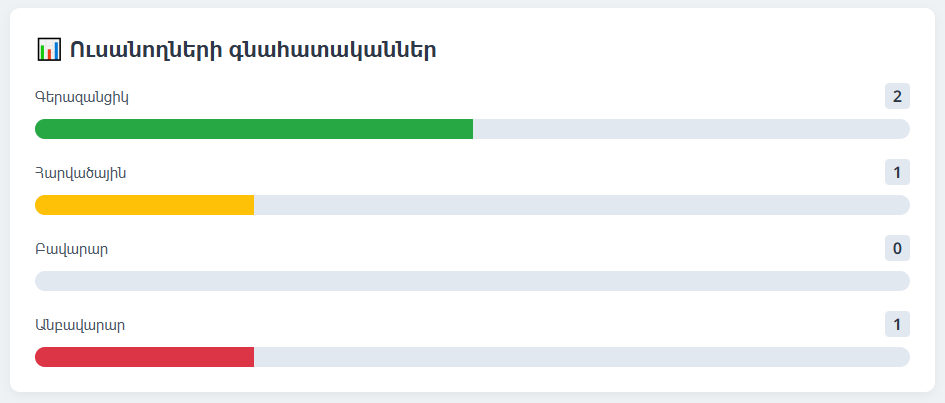
**2․1 Թեմա**

Դիպլոմային նախագծի շրջանակներում ստեղծվել է ուսումնական գործընթացի կազմակերպման հարթակ, որը կոնկրետ հաստատության ուսուցչական ու ուսանողական անձնակազմերին տալու է ուսումնական պրոցեսի թվայնացման մի շարք գործիքներ՝ սկսած էլեկտրոնային մատյանից ու օրագրից, մինչև առցանց գրադարանի ու վիդեոդասերի ֆունկցիոնալություն, լավ մշակված անվտանգություն, արհեստական բանականությամբ աշխատող օգնական և այլն։ Հարթակի ստեղծման աշխատանքներում որպես հիմնական լեզու կիրառվել է **Java** ծրագրավորման լեզուն, որի **Spring Boot** և **Hibernate ORM** տեխնոլոգիաները տվել են բավականին լայն գործառնություններով համակարգ ստեղծելու հնարավորություններ։ Համակարգի տվյալների պահպանման համար օգտագործվել է **MySql** տվյալների բազան՝ լայն ֆունկցիոնալությամբ ու Hibernate ORM -ի հետ լավ ինտեգրմամբ։ Ծրագրի արտաքին հատվածը կամ այսպես կոչված ֆրոնտ-ենդ(Front-End) մասը մշակվել է **HTML**, **CSS**, **Bootstrap 5** օժանդակ տեխնոլոգիաների և **JavaScript** ծրագրավորման լեզվի կիրառմամբ։

Կրթական համակարգում մաթեմատիկական հաշվարկները մեծ դեր են խաղում և առանց դրա անհնար կլիներ պատկերացնել ճիշտ աշխատող համակարգ։ Այս պարագայում մաթեմատիկական հաշվարկների դերը հատկապես մեծ էր ուսանողների միջին գնահատականների ձևավորման ու դրա հիման վրա ուսումնական հաստատությունում վիճակագրական տվյալների ստացման դեպքերում: Նկար 2․1 -ում ցուցադրված է անհրաժեշտ տվյալների ստացման ու կարգավիճակի որոշման մեթոդները, իսկ նկար 2․2 -ում՝ տվյալների ցուցադրումը հարթակում։



Նկար 2․1



Նկար․ 2․2

Ինչպես վերը նշվեց, կիրառվել է Hibernate ORM տեխնոլոգիան։ Ահա կոդի կարգավորումների այն հատվածը, որով ապահովվում է կապը բազայի հետ (նկար 2․3)՝ գրելով անհրաժեշտ պարամետրերը ըստ նախապես սահմանվածի։

Նկար 2․4

Ուսանողների աղյուսակ

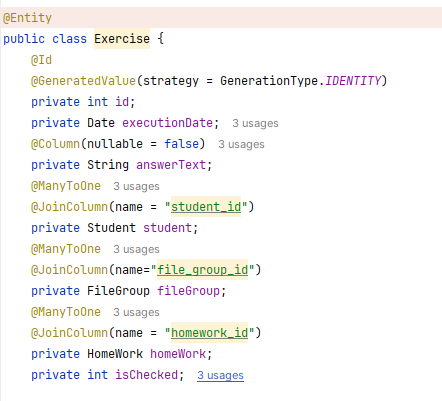


Նկար 2․3

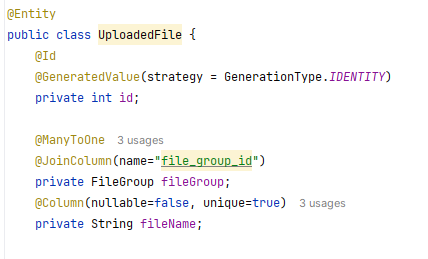
Նշված տեխնոլոգիաների կիրառման շնորհիվ միայն բազան է, որ անհրաժեշտ է ստեղծել այլ ծրագրային միջավայրում, իսկ տվյալների բազայի կառուցվածքը ներկայացնելիս շատ հանգիստ կարելի է ծրագրային կոդի դասերը ներկայացնել, որոնք ստանալով հատուկ անոտացիաներ, դառնում են աղյուսակներ բազայում (նկար 2․4–2․), և բոլոր լրացուցիչ կարգավորումները ևս հնարավոր է դառնում դասերում սահմանել՝ բանալիներ, կապերի տեսակներ, դաշտերի տարատեսակ հատկություններ և այլն։



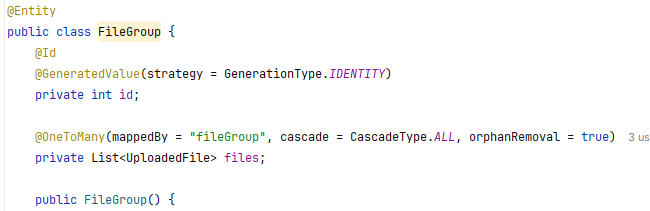
Նկար 2․4



Նկար 2․5



Նկար 2․6

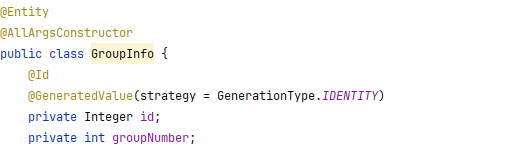


Նկար 2․7

Ֆայլերի ու նրա խմբերի պարագայում նման կառուցվածք է տրվել, քանի որ կարող են լինել դեպքեր, երբ մի առաջադրանքը կամ առաջադրանքի պատասխանը պարունակեն մի քանի ֆայլեր, և այդտեղ անհրաժեշտ է խմբավորում, ինչպես արված է նկարներ 2․6 և 2․7 -ում։



Նկար 2․9



Նկար 2․8



Նկար 2․10

Ծանուցման համակարգում ավելի հավաք տվյալներ ունենալու համար ուսուցչի և ուսանողի ծանուցումները պահվում են տարբեր աղյուսակներում, իսկ նկար 2․11–ում ցուցարդված է ուսուցչի ծանուցումների աղյուսակը։

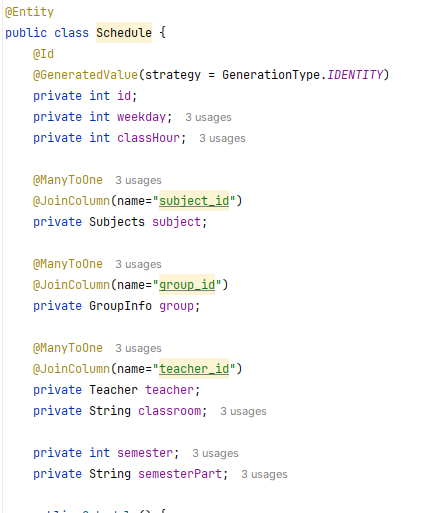


Նկար 2․11

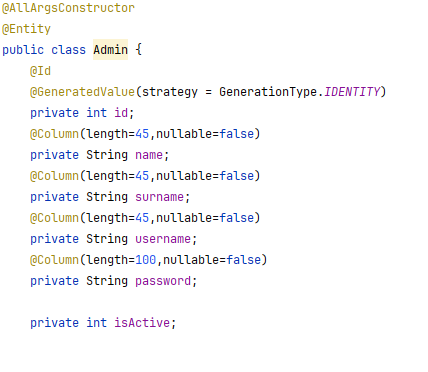


Նկար 2․12

Գնահատականների և դասացուցակի աղյուսակները շատ կարևոր են էլեկտրոնային մատյանի ու օրագրի ձևավորման հարցում։

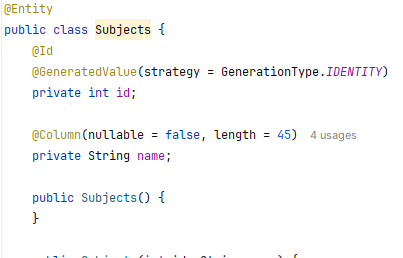


Նկար 2․13

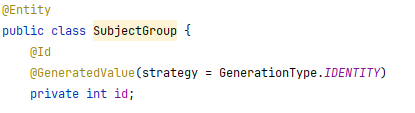


Նկար 2․14

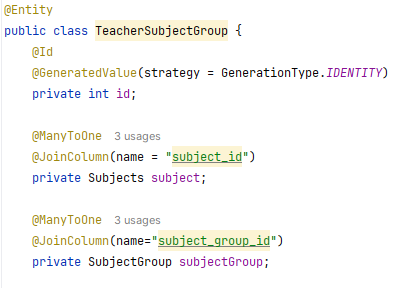
Ադմինիստրատորն ունի հարթակի անխափան աշխատանքն ապահովելու բոլոր լծակները։



Նկար 2․15



Նկար 2․16

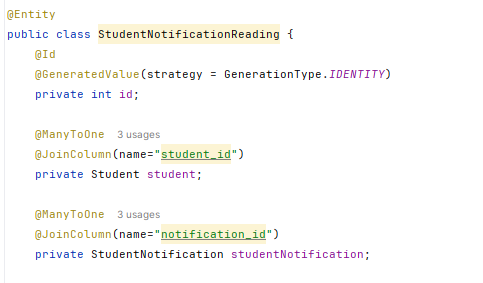


Նկար 2․17

Քանի որ դասախոսը կարող է միաժամանակ մի քանի առարկա դասավանդել, հետևաբար ստեղծվել է 3 աղյուսակ(նկար 2․15–2․17), և նրանց միջև տրվել են կապեր առարկաների ճիշտ խմբեր ստանալու համար։



Նկար 2․18



Նկար 2․19

Նկար 2․19 –ում պատկերված աղյուսակում պարունակվում են տվյալներ յուրաքանչյուր ուսանողի յուրաքանչյուր ծանուցում կարդալու վերաբերյալ։ Բառացիորեն նույն տրամաբանությամբ կա նաև նմանատիպ աղյուսակ ուսուցիչների համար։

**2․2 Խնդրի դրվածքը**

Հարթակի ծրագրային կառուցվածքը նախագծելուց առաջ կատարվել են ուսումնասիրություններ՝ հասկանալու կրթական ոլորտում առաջնային նշանակություն ունեցող հասկացություններն ու կարիքները, որոնք թվայնացվելու դեպքում կարող են զգալիորեն հեշտացնել կրթության կազմակերպումը։ Այս նախագծի նպատակն է ստեղծել կրթական հարթակ, որը կլուծի դպրոցներում կամ ուսումնական հաստատություններում ուսուցման գործընթացի կազմակերպման, գնահատման և կառավարման խնդիրները։ Միևնույն ժամանակ, պիտի ուշադրություն դարձվի կայքի հարմարավետությունը (**User-Friendly**), ինչի նպատակով պիտի լինի արհեստական բանականությամբ աշխատող չաթ–բոտ։ Շատ կարևոր է ուս․ հաստատությունում ունենալ կայքի ադմինիստրատոր(ներ), որը կհետևի կայքի աշխատանքին, ինչպես նաև մշտապես կծանուցի օգատերերին ուսումնական հաստատությունում տեղի ունեցողի մասին։

Ստորև գրված են ի սկզբանե առաջադրված պահանջներ՝

1. **Մատյան և օրագիր**
   * թվայնացված մատյան՝ գնահատականների և բացակայությունների առցանց արձանագրման համար, որը թույլ կտա գնահատում միայն իրական ժամանակի համար,
   * էլեկտրոնային օրագիր, որտեղ ուսանողը կտեսնի իր գնահատականը՝ բոլոր մանրամասնություններով և դասախոսի մեկնաբանությամբ
2. **Առաջադրանքների կառավարման համակարգ**
   * ուսուցիչների կողմից առաջադրանքների ստեղծում, ստուգում, խմբի վիճակագրական տվյալների դիտման թույլտվություն, հանձնարարությանը կից ֆայլերի վերբեռնում
   * Աշակերտների կողմից առաջադրանքների կատարման հնարավորություն՝ սահմանված ժամկետներում, կից ֆայլերի վերբեռնման հնարավորությամբ,
3. **Արհեստական բանականության (ԱԲ) օգնական**
   * ԱԲ գործիք դասախոսներին ու օգտատերերին կայքից օգտվելիս աջակցելու համար, տալով անհրաժեշտ տեղեկատվություն կայքի աշխատանքի վերաբերյալ, այլընտրանք ավանդական աջակցման կենտրոններին
4. **Դասացուցակի կառավարման համակարգ**
   * Դասացուցակի կազմում և մշտապես թարմացում ուսումնական տարվա ընթացքում՝ ներառյալ լսարանների համարները, և այլ մանրամասներ,
   * Աշակերտների և ուսուցիչների համար անհատականացվող դասացուցակների դիտում սեփական օգտահաշիվներում,
5. **Ամփոփիչ արդյունքների ձևավորում**
   * Կիսամյակային և տարեկան գնահատականների ավտոմատ հաշվարկ՝ հիմք ընդունելով ընթացիկ առաջադիմությունը՝ կրթական պրոցեսն ավելի թափանցիկ դարձնելու համար,
   * Առաջադիմության հիման վրա տարբեր ցուցանիշների ձևավորում,
6. **Օգտագործողի անվտանգ ու հարմար ինտերֆեյս**
   * Հարթակի դյուրին օգտագործում բոլոր շահառուների՝ ուսուցիչների, աշակերտների և ադմինիստրատորների համար,
   * Էլ․ հասցեի, գաղտնաբառի փոփոխման հնարավորություն,
   * Գրանցման պրոցեսի փոխարինում ադմինիստրացիայի կողմից նախապես ստեղծված օգտահաշիվների անվտանգ ակտիվացմամբ,
   * Օգտահաշվի հիմնական նկարի սահմանման հնարավորություն, անձին հեշտ նույնականացնելու համար,
   * Հետադարձ կապի հնարավորություն, տեխնիկական խնդիրներին արագ լուծում տալու համար,
   * Ճկուն ծանուցման համակարգ, ծրագրի կողմից ավտոմատ ստեղծվող ու ադմինիստրացիայից ուղակվող ծանուցումների համադրմամբ,
7. **Վիդեոդասերի ու առցանց գրադարանի հնարավորություն**
   * Հնարավորություն ուսանողների համար առցանց կրթվելու, իսկ դասախոսների համար նաև նոր նյութերի ավելացման ֆունկցիոնալություն
   * Ճկուն որոնման և ֆիլտրման համակարգեր անհրաժեշտ նյութը արագ գտնելու համար
8. **Լայն ֆունկցիոնալությամբ ադմինիստրացիա**
   * Հասանելիություն բոլոր անհրաժեշտ տվյալներին,
   * տվյալների ներմուծման ու փոփոխման հնարավորություն, ներառյալ դասախոսների, ուսանողների, առարկաների, խմբերի, դասացուցակի, վիդեոդասերի, գրադարանի, այլ ադմինիստրատորների և ծանուցումների տվյալները,
   * հաստատության կարևորագույն վիճակագրական տվյալների դիտման հնարավորություն,
   * օգտահաշիվների արգելափակման կամ ապաակտիվացման հնարավորություն՝ կասկածելի գործողություն կամ մուտք նկատելու դեպքում։

Նախապես սահմանված պահանջներին համապատասխան իրականացվել է ծրագրի ստեղծման մնացած աշխատանքը՝ սերվերային հատվածից մինչև ինտերֆեյս։

**2․3 Ալգորիթմի նկարագրությունը**

Ծրագրային հատվածն իրականացնելիս մշակվել են մի շարք ալգորիթմներ, որոնք սերվերային մակարդակում մեծ նշանակություն են ունեցել տվյալների ստացման համար, սակայն մինչ դա կարևոր է հասկանալ կայքի ընդհանուր ալգորիթմը։ Հարթակում գործողությունները կկրեն տրամաբանական հերթականություն։

Բլոկ սխեմա 2․1 –ում պատկերված է կայք մուտք գործելիս բացվող էջի ու նրա տրամաբանական շարունակության ալգորիթմը։ Ամեն ինչ սկսվում է մուտք գործելու պրոցեսից, որտեղ սխալ տվյալներ մուտքագրելու դեպքում կատարվում է վերադարձ այդ նույն էջ։ Այդ էջում նաև առկա է շարունակության այլ ճանապարհներ՝ գաղտնաբառի վերականգնում և օգտահաշվի ակտիվացում։

Ուսուցչի էջ

Ուսանողի էջ

Օգտահաշվի ակտիվացում

Գաղտնաբառի վերականգնում

Բլոկ սխեմա 2․1

Այժմ դիտարկենք այն դեպը երբ մուտք գործած օգտատերը ուսուցիչ է։ Այդ պարագայում ևս գործողությունը սկսված կլինի նախորդ օրինակի նույն հաջորդականությամբ և կշարունակվի արդեն ուսուչի էջում(բլոկ սխեմա 2․2): Դա ուսուցչի էջում տեղի ունեցող տրամաբանական շղթայի մի դրվագ է, որը բավարար է հասկանալու համար կայքի աշխատանքի ալգորիթմն ու տրամաբանությունը։ Եթե փորձենք պատկերացնել ուսանողի գործողությունների շղթան, ապա նույնաոճ ալգորիթմ կստանանք՝ միայն թե ալգորիթմի տարրերը կլինեն ուսանողի օգտահաշվի տրամաբանության սահմաններում։

Ուսուցչի էջ

Փոփոխություն մատյանում

Առաջադրանքների բաժին

Վիդեոդասեր

Դասացուցակ

Գործընկերներ

Ամփոփիչ արդյունքների բաժին

Գրադարան

Նոր առաջադրանք

Առաջադրանքի վիճակագրության էջ

Առաջադրանքի ստուգում

Չստուգված առաջադրանքներ

Վիդեոդասի ավելացում

Գրքի ավելացում

Բլոկ սխեմա 2․2

Նախորդ օրնակներում դիտարկվել է ընդհանուր կայքի ալգորիթմը, իսկ հիմա կուսումնասիրենք ծրագրային ալգորիթմներ, որոնք կարևոր դեր ունեն կայքի աշխատանքում։

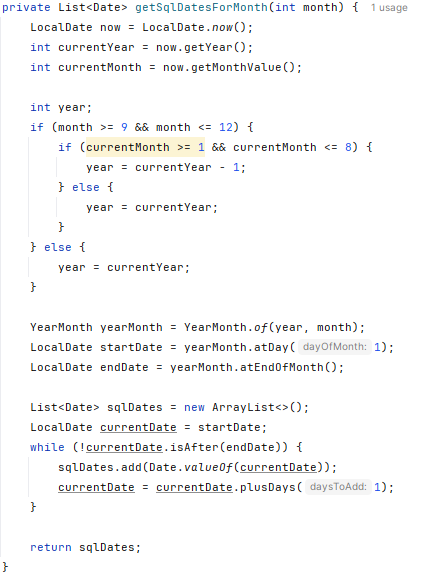
Էլեկտրոնային մատյանը մշակելիս շատ կարևոր է ունենալ կապվածություն դասացուցակի հետ, որպեսզի ավտոմատ կերպով երևան միայն այն ամսաթվերը, երբ պլանավորված դասեր կան, իսկ դա հնարավոր է միայն ամսաթվերի, կիսամյակի, մոդուլի ու շաբաթվա օրերի համակցության շնորհիվ։ Ստորև (նկար 2․20) ցուցարդված է ըստ ամսվա՝ կիսամյակի ու կիսամյակի մոդուլի ստացման ալգորիթմները՝

1. **Ամսաթվերի հստակեցում**



Նկար 2․20

առաջին մեթոդում ստացվում է կիսամյակը, իսկ երկրորդում արդեն կիսամյակի մոդուլը։ Այս տվյալների ստացումից հետո անհրաժեշտ է ըստ ամսվա ստանալ տվյալ ամսվա օրերը (նկար 2․21), որպեսզի հետագայում դրանք համապատասխանեցվեն գնահատականների ամսաթվերի հետ։

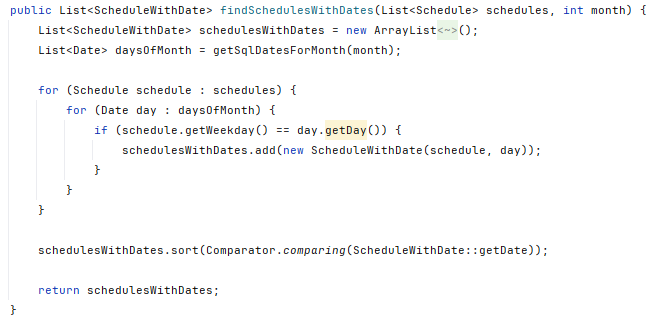


Նկար 2․21

Այս մեթոդը (նկար 2․21) որպես մուտքային պարամետր ընդունում է ամսվա համարը, այնուհետև հասկանում ներկա ժամանակի ամիսը, որպեսզի կարողանա ճշգրիտ ամսաթվեր վերադարձնել, քանի որ շաբաթվա օրերը ամեն տարվա համար ճկուն են և շատ կարևոր է ընտրված ամսից հասկանալ տրամաբանությունը։ Օրինակ՝ եթե հիմա ապրիլ ամիսն է, իսկ որոնելիս ընտրվել է սեպտեմբեր ամիսը, ապա համակարգն ավտոմատ պիտի վերադարձնի նախորդ տարվա սեպտեմբերի ամսաթվերի տվյալները։

1. **Դասացուցակի հստակեցում**

Ամսաթվերի հետ կապված մեթոդների ստացումից հետո կարելի է ստանալ դասացուցակը, բայց դրան ավելացնելով նաև օրվա ամսաթիվը(նկար 2․22), որը կօգտագործվի էլեկտրոնային մատյանի ճշգրիտ տեսքը ստանալիս արտաքին ինտերֆեյսում։

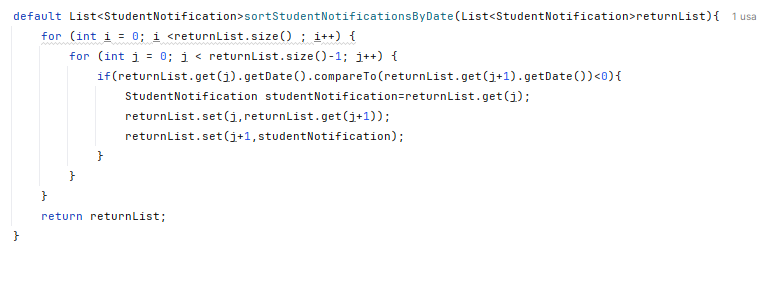


Նկար 2․22

Նկար 2․22 –ում պատկերված կոդի արդյունքում մենք կունենանք տվյալ ամսվա դասացուցակը, սորտավորված ըստ օրերի։ Տվյալ դեպքում դասավորումն իրականացրել ենք ջավա –ի **Comporator** պատրաստի ինտեֆեյսի տրամադրած **comparing** ստատիկ մեթոդի կիրառմամբ։

1. **Պղպջակային տեսակավորում(Bubble Sort)**

Նախորդ օրինակում մենք տեսանք Comporator ինտերֆեյսի պատրաստի մեթոդի կիրառում, տվյալների սորտավորման համար։ Այս օրինակում կդիտարկենք իմ կողմից մշակված սորտավորման մեթոդ, որը կօգտագործենք ծանուցումների՝ ըստ ամսաթվի սորտավորման համար (նկար 2․23)։

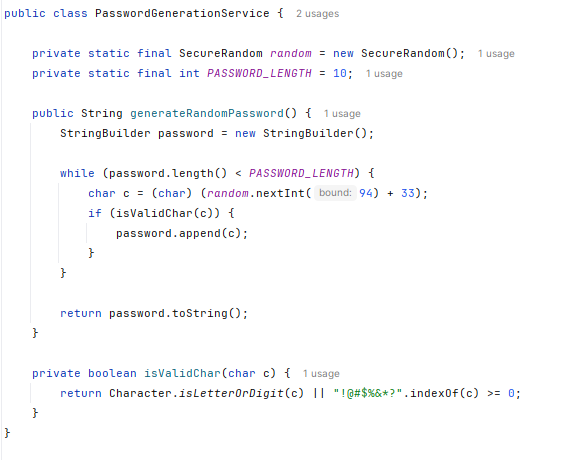


Նկար 2․23

Նկար 2․23 –ում ցուցադրված օրինակում մշակվել է մեթոդ, որով ծանուցումները դասավորվում են ըստ ստեղծման ամսաթվի ու սկզբում ցուցադրվում են ամենաթարմ ամսաթվերը։ Ալգորիթմը, որով մեթոդը մշակվել է, թարգմանաբար անվանում են պղպջակային տեսակավորում(Bubble Sort)։

1. **Գաղտնաբառի ավտոմատ ստեղծում**

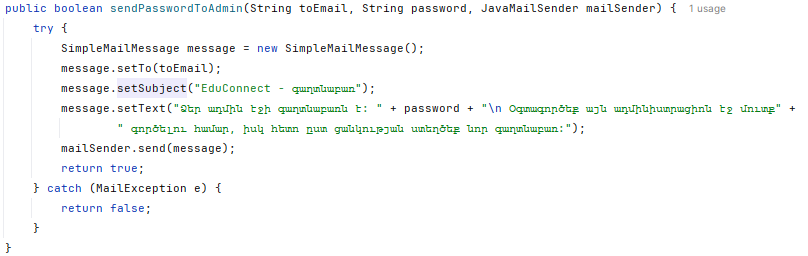
Ադմինիստրատորների ստեղծման դեպքում ևս անհրաժեշտ են որոշակի անվտանգությունը բարձրացնող միջոցառումներ։ Այդ իսկ նպատակով մշակվել է գաղտնաբառի ավտոմատ գեներացման մեխանիզմ (նկար 2․24), որը կստեղծի ապահով գաղտնաբառ ու համակարգը այն կուղարկի նոր ադմինիստրատորի էլ․ հասցեին, ով հետագայում ըստ ցանկության կարող է փոխել գաղտնաբառը։



Նկար 2․24

Այստեղ (նկար 2․24) ստուգվում է, որպեսզի ամեն սիմվոլ լինի թույլատրելի գաղտնաբառի ձևաչափում, իսկ գաղտնաբառի սիմվոլների քանակը հավասար է 10 –ի։

Այս մեթոդն (նկար 2․25) էլ իր հերթին ապահովում է նոր ստեղծած գաղտնաբառի ուղարկումը տվյալ օգտատիրոջը։ Ուղարկումն իրականացվում է **JavaMailSender** և **SimpleMailMessage** պատրաստի դասերի տրամադրած մեթոդների օգնությամբ։



Նկար 2․25

1. **Ֆայլի վերբեռնման ալգորիթմ**

Հարթակի ֆունկցիոնալության քննարկման ընթացքում ընդգծվել է, որ այն ունենալու է առաջադրանքների մեխանիզմ, և որ այնտեղ օգտատերերը պիտի կարողանան մեկ կամ ավելի ֆայլեր վերբեռնել, ինչպես նաև վիդեոդասերի ու գրադարանի դեպքում ևս ֆայլերի վերբեռնման անհրաժեշտություն լինելու է։

Այդ նպատակով մշակվել է հատուկ մեթոդ (նկար 2․26), որը կապահովի օգտատիրոջ ֆայլի պահպանումը նախագծի անհրաժեշտ հատվածում։

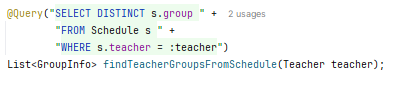


Նկար 2․26

Ֆայլի պահպանման ընթացքում ևս օգտագործվել են մի քանի պատրաստի դասեր, որոնք հեշտացրել են կատարվող աշխատանքը։ Դրանք են **Path** և **Files** դասերը և իրենց պատրաստի մեթոդները։

1. **Ուսուցչի դասավանդվող խմբերը**

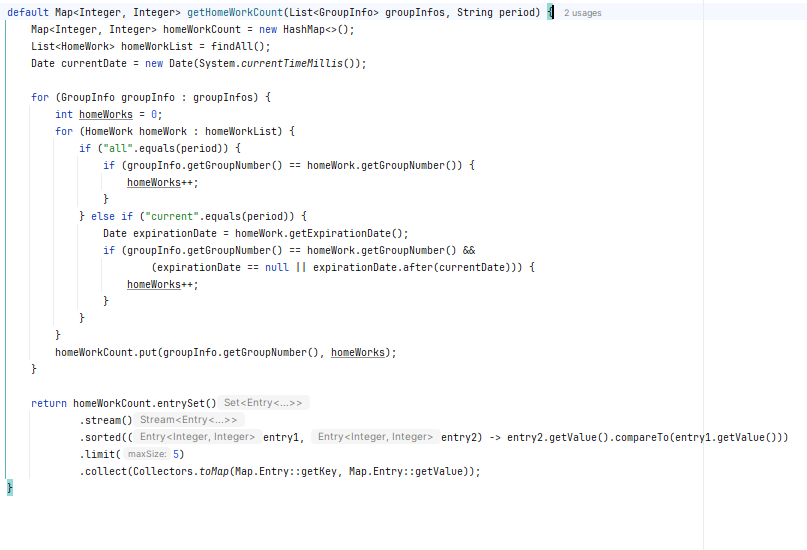
Որքան էլ **Hibernate ORM -ը** տալիս է պատրաստի մեթոդների լայն տեսականի, այնուամենայնիվ լինում են դեպքեր, երբ անհրաժեշտ է կոդում գրել **SQL** կոդ ու կատարել հարցումներ։ Բազայի կառուցվածքից ելնելով, ուսուցչի դասավանդած խմբերը ստանալու միակ ելքը դասացուցակների աղյուսակից հարցում կատարելն էր։ Ահա այն ալգորիթմը, որը ստացվեց նման հարցում կատարելիս (նկար 2․27)՝



Նկար 2․27

1. **Ալգորիմ Data Structure –ի կիրառմամբ**

Պայմանավորված արտաքին ինտերֆեյսի բազմազանությամբ, տարբեր իրավիճակներում պետք է ունենալ ինֆորմացիայի փոխանցման ճկուն եղանակներ։ Այս դեպքում մշակվել է մեթոդ, որը վերադարձնում է ուսանողների խմբերը և իրենց ունեցած տնային աշխատանքների քանակը։ Որպեսզի նոր տվյալների տիպ չմշակվեր սրա համար, այդ տվյալների փոխանցումը կազմակերպվել է **Map** –ի կիրառմամբ, օգտագործելով նրա բանալի–արժեք(**key-value**) յուրահատկությունը (նկար 2․28)։



Նկար 2․28

1. **Ունիվերսալ մեթոդներ**

Շատ հաճախ անհրաժեշտ է լինում մշակել մեթոդներ, որոնք հետագայում կօգտագործվեն մի քանի տիպի օգտատերերի կողմից և կբացառեն կոդի կրկնության ալգորիթմը։ Նման օրինակ է նկար 2․29 -ում պատկերված ալգորիթմը, որը իրականացնում է գրադարանի բազայում պահված ինֆորմացիայի ջնջում, իսկ նման լիազորություն ունեն և՛ ադմինը, և՛ ուսուցիչը։ Ուստի մշակվել է այնպիսի մեթոդ, որը կնախատեսի երկու դեպքերն էլ ու կտա անհրաժեշտ արձագանք։ Բացի այդ, վերոհիշյալ պատկերում առկա է օրինակ, թե ինչպես ունենալ **դինամիկ** **URL**, որը այս դեպքում պարունակում է աղյուսակի այն տվյալի իդենտիֆիկացնող համարը, որն անհրաժեշտ է ջնջել։ Դա կատարվել է Spring Boot -ի տրամադրած անոտացիաների կիրառմամբ։



Նկար 2․29

1. **Անվտանգության վերահսկող մեթոդ**

Առցանց հարթակներում օգտահաշիվների անվտանգությունը մեծ դեր է խաղում, ինչի նպատակով շատ հաճախ տվյալների բազայում պահվում են օգտատիրոջ գործողությունների վերաբերյալ որոշակի մանրամասներ՝ կասկածելի քայլի դեպքում օպերատիվ արձագանքելու նպատակով։ Այդ նպատակով մշակվել է ալգորիթմ, որը օգտվողի՝ հարթակ մուտք գործելիս պահում է որոշակի մանրամասներ։ Այդ մանրամասները ներառում են սարքավորման **IP**(Internet Protocol) հասցեն, օգտագործված սարքավորման մանրամասները՝(**User-Agent**), և իրական ժամանակում գործողության իրականացման ժամանակը(նկար 2․30)։



Նկար 2․30

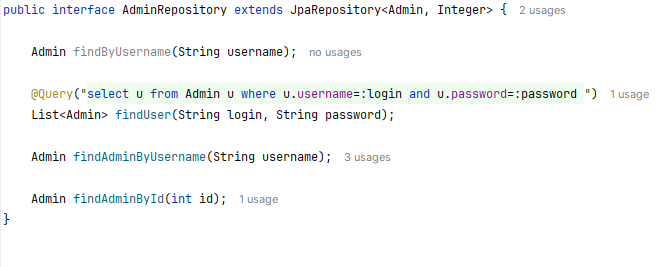
**2․4 Ծրագիր**

Նախորդիվ արդեն իսկ մակերեսորեն ուսումնասիրվել են որոշակի ալգորիթմներ, ինչպես նաև տվյալների բազայի կառուցվածքը, իսկ այժմ արդեն կդիտարկվի բուն կոդը։



Նկար 2․31

Նկար 2․31 -ում ցուցադրված է կայքի անվտանգության ստուգման մեթոդը։



Նկար․2․32



Նկար 2․33



Նկար 2․34

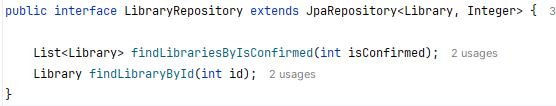
Նկար 2․32-34 -ում պատկերված է կոդային այն կառուցվածքը, որը պատասխանատու է ադմինիստրացիայի՝ համակարգ մուտքի կազմակերպման համար։

Իսկ այժմ հաջորդաբար կներկայացվեն **MVC** կառուվածքում մոդելի դեր ունեցող ինտերֆեյսերից(**Interface**) կարևորները, որոնք կօգտագործվեն ծրագրի ողջ ընթացքում(նկար 2․35-53)։



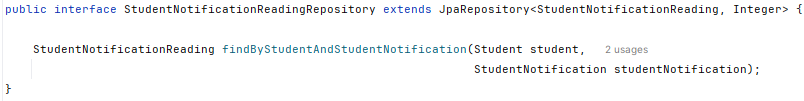
Նկար 2․35



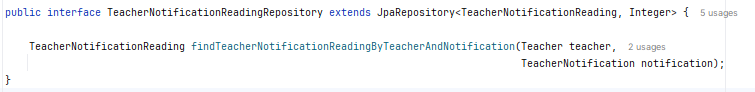


Նկար 2․37

Նկար 2․36



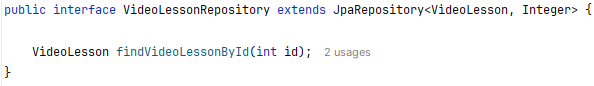
Նկար 2․38



Նկար 2․39



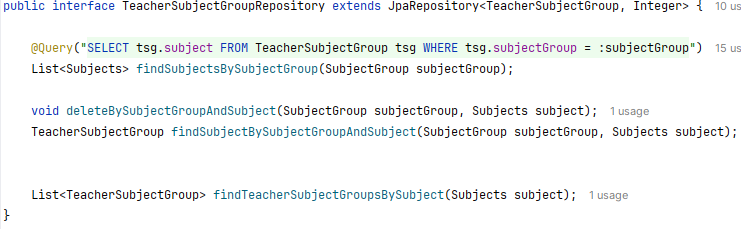
Նկար 2․40



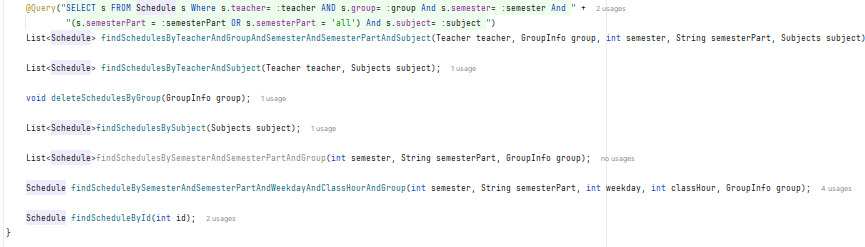
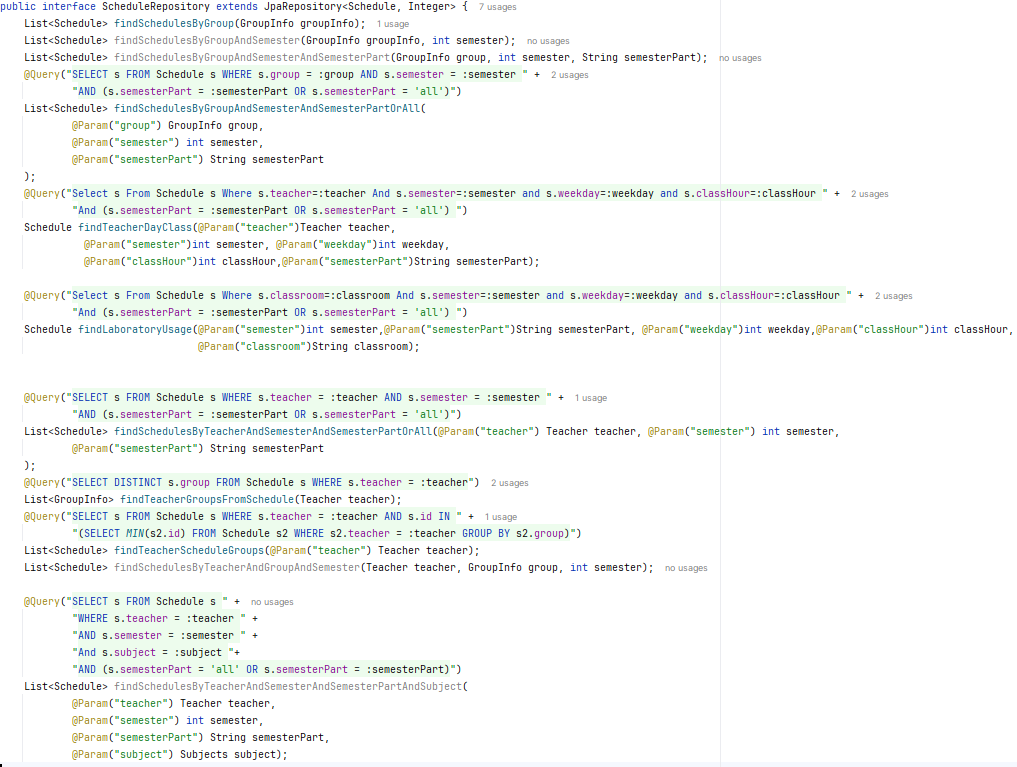
Նկար 2․41



Նկար 2․42



Նկար 2․43



Նկար 2․44



Նկար 2․45

Նախորդ նկարներում ցուցադրված էին առավել շատ գործաված ինտերֆեյսերը, բայց պիտի նշեմ, որ դրանք ամբողջը չէին։ Կարելի է ասել, որ ինտերֆեյսերի քանակը ճիշտ այնքան է, որքան բազայում աղյուսակ կամ պրոեկտում **Entity** անոտացիան պարունակող դաս, որոնք արդեն սկզբում դիտարկել ենք։ Հենց սա է **Hibernate ORM** -ի հիմնական նշանակությունը։

**2․5 Ծրագրի բացատրությունը**

Բազայի կառուցվածքի, մոդելի դեր ունեցող միավոր ունեցող միավորների ուսումնասիրություններից հետո կարելի է անցնել արտաքին տեսքի ու սերվերային մասի միջև ինֆորմացիայի փոխանակման շատ կարևոր հատվածին։

Քիչ առաջ մենք նայում էինք, թե ինչպես է ադմինիստրատորը մուտք գործում իր օգտահաշիվ, իսկ այժմ ուսումնասիրենք մի քանի կարևորագույն մեթոդներ, որոնք ապահվում են ֆունկցիոնալություն ադմինիստրատորի համար սերվերային մակարդակում։



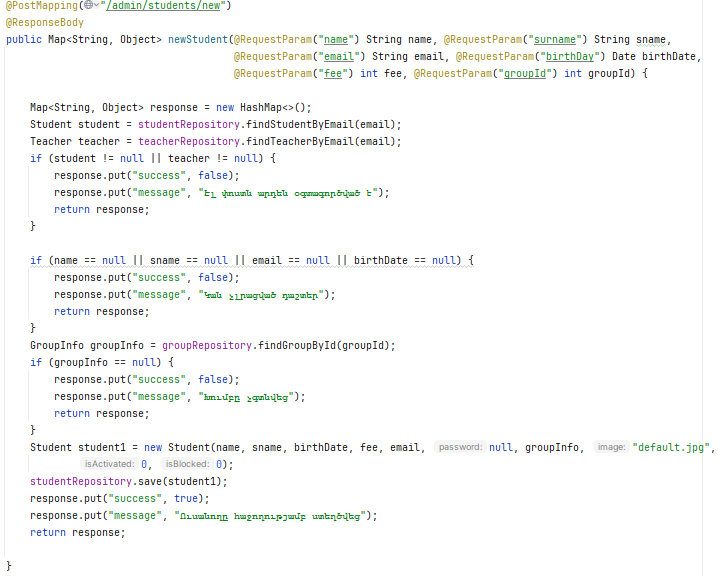
Նկար 2․54

Նկար 2․54 -ում ցուցադրված է ուսանողների տվյալների կառավարման էջ տվյալներ փոխանցելու համար նախատեսված մեթոդը։ Նույն տրամաբանությամբ դիտարկենք ադմինիստատորի համար նախատեսված մի քանի մեթոդներ(նկար 2․55-2․74)։



Նկար 2․55

Նախորդիվ դրված նկարում(նկար 2․55) ապահովվում է մի սեղմմամբ ուսանողին բլոկավորելու և ապաբլոկավորելու ֆունկցիոնալությունը։ Հատկապես պետք է ուշադրություն դարձնել մեթոդի տիպին ու վերադարձվող արժեքի ձևաչափին։ Ամեն բան արված է այնպես, որ ֆրոնտ հատվածից(**JavaScript**) հնարավոր լինի ուղարկել հարցում առանց էջի թարմացման։ Հենց այդտեղ է երևում **Ajax** տեխնոլոգիայի կիրառումը կոդում։



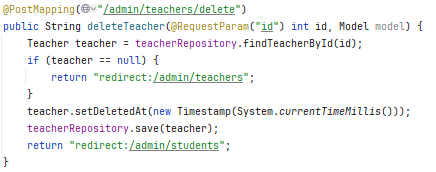
Նկար 2․56

Նկար 2․56 -ում պատկերված է նոր ուսանողի տվյալների մշակման սերվերային հատվածը։ Այս մեթոդը ևս իրականացվում է առանց էջի թարմացման, իսկ մինչև հաստատումը կատարվում են բոլոր անհրաժեշտ ստուգումները։



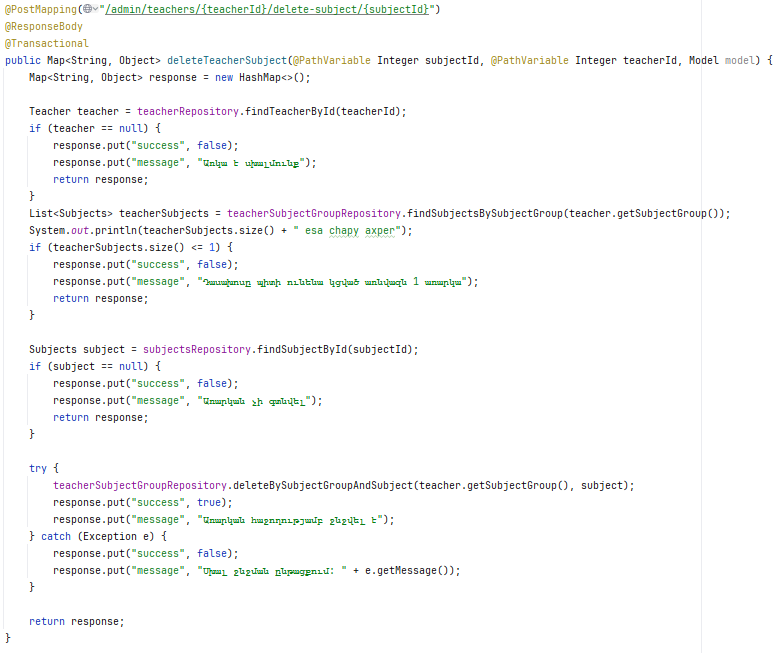
Նկար 2․57

Նկար 2․57 -ում պատկերված է կոնկրետ ուսանողի տվյալների էջի բեռնումը, իսկ դրան հաջորդում է ուսանողի տվյալների հեռացման ալգորիթմը։ Այն ինչ դիտարկեցինք ադմին-ուսանող ֆունկցիոնալության շրջանակներում առկա է նաև ուսուցչի համար, սակայն մեկ տարբերությամբ․ ուսուցչի հեռացման դեպքում օգտագործվում է տվյալների հեռացման **Soft Delete**(փափուկ ջնջում) տեխնոլոգիան, երբ տվյալը ֆիզիկապես չի մաքրվում բազայից, այլ սահամնվում է հատուկ դաշտ, որը ցույց է տալիս նրա ջնջված լինելու հանգամանքը(նկար 2․58)։



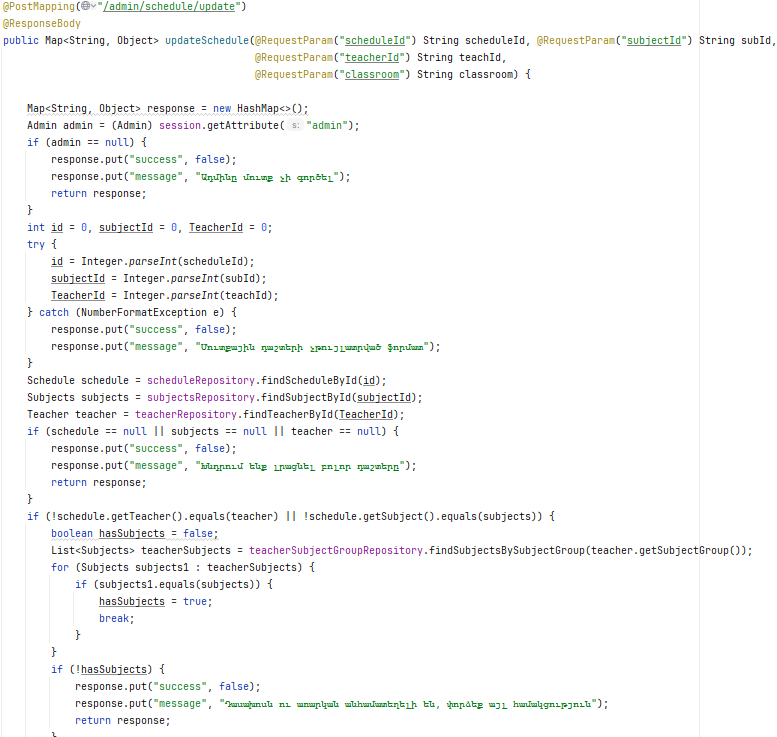
Նկար 2․58

Հաջորդ հետաքրքիր մեթոդը որն արժանի է դիտարկան՝ ուսուցչին կցված առարկաներից մեկի հեռացումն է(նկար 2․59)։

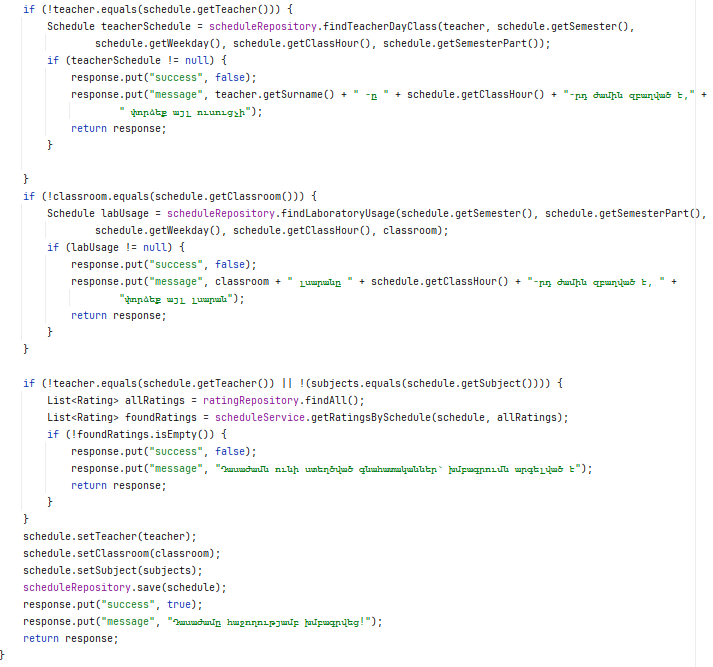


Նկար 2․59

Այժմ դիտարկենք ամենահետաքրքիր մեթոդներից մեկը՝ դասացուցակի թարմացում(նկար 2․60-61)։



Նկար 2․60



Նկար 2․61

Դասացուցակի հետ աշխատանքը իրենից ենթադրում է շատ քանակի տվյալներ, և անճշտություններից խուսափելու համար անհրաժեշտ է մեծ քանակությամբ ստուգումներ իրականացնել։

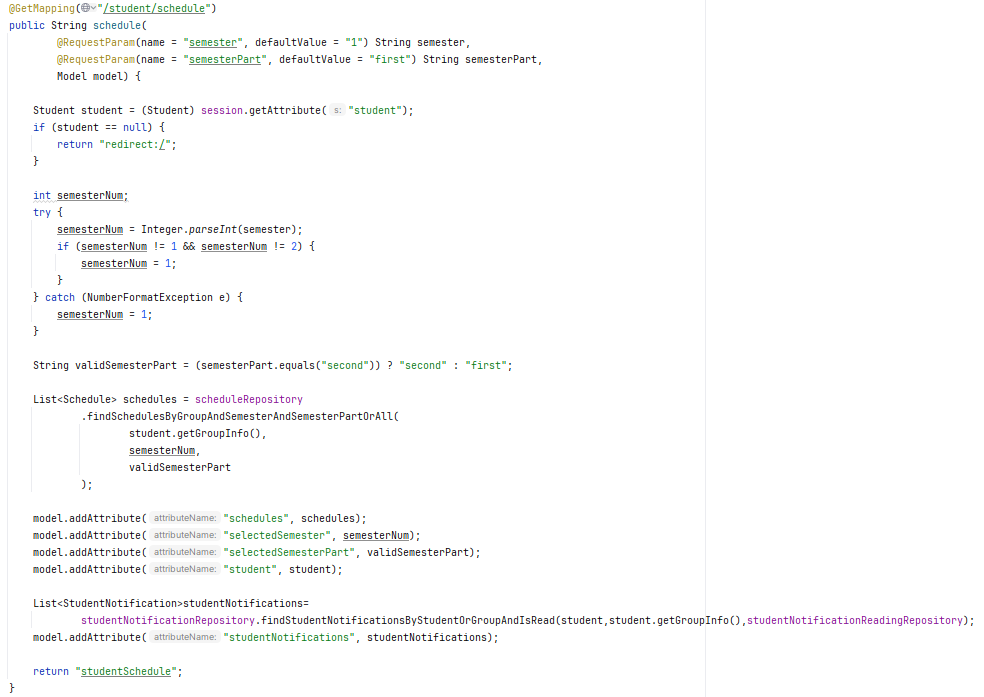
Հետաքրքիր մեթոդներից կարելի է աձանձնանցել նաև ուսուչի էջում էլեկտրոնային մատյանի ձևավորման համար անհրաժեշտ տվյալների փոխանցումը(նկար 2․62)։ Այստեղ շատ կարևոր է ամսաթվերի ու գնահատականների փոխանցում այնպիսի ձևաչափով, որ ֆրոնտ-էնդ մակարդակում հնարավոր լինի գեներացնել էլեկտրոնային մատյան, իսկ այդ ձևաչափերի ստացման մասին նախորդիվ հասցրել ենք խոսել ու դիտարկել այդ մեթոդները։



Նկար 2․62

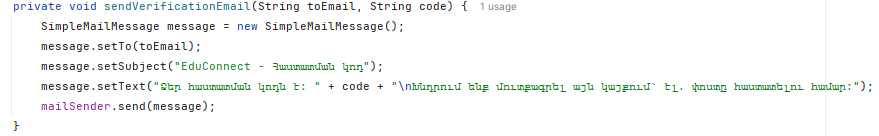
Մեթոդի ստորին մասում կարելի է նաև նկատել կոնսոլային(**Console**) արտածման որոշակի փորձեր, որոնք նախատեսված են պարզելու համար, թե արդյոք տվյալը փոխանցվել է այնպես, ինչպես պետք է։

Ուսանողի էջի համար նախատեսված մեթոդներից առանձնացնեք դասացուցակի ձևավորման մեթոդը(նկար 2․63)։ Այստեղ ևս շատ կարևոր է այն հանգամանքը, որ ամսաթվերն ու շաբաթվա օրերը ճիշտ ձևով համապատասխանեցվեն դասացուցակի տվյալների հետ՝ կախված ամսից, կիսամյակից, կիսամյակի հատվածից։ Նաև նշեմ, որ թե՛ ուսանողի, և թե՛ ուսուցչի համար նախատեսված մեթոդներում մշտապես թարմացվում է ծանուցումների հետ կապված տեղեկատվությունը, որպեսզի նոր ծանուցումները երևան օգտահաշվում։

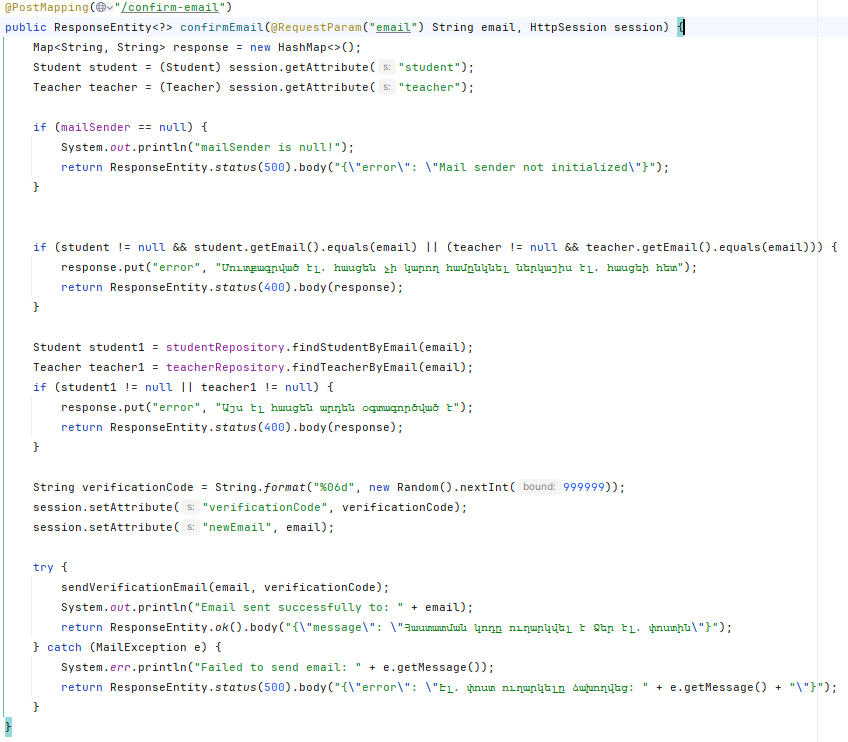


Նկար 2․63

Դիտարկենք տվյալների փոխանցման մի քանի այլ մեթոդներ։



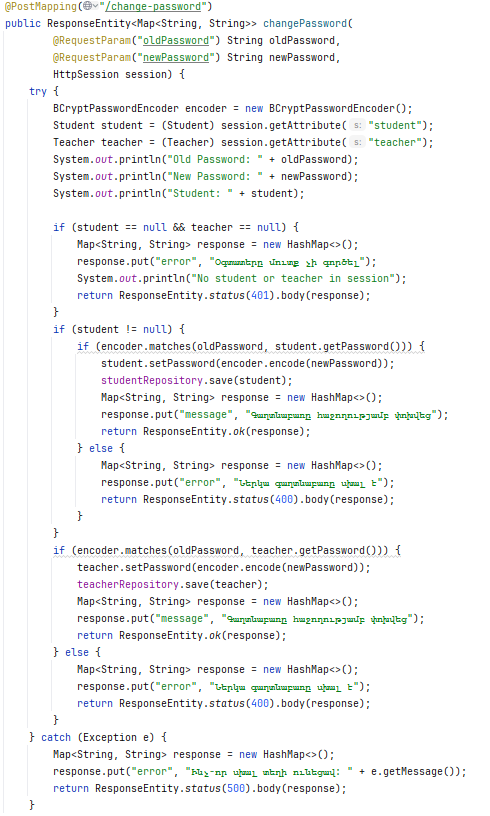
Նկար 2․64



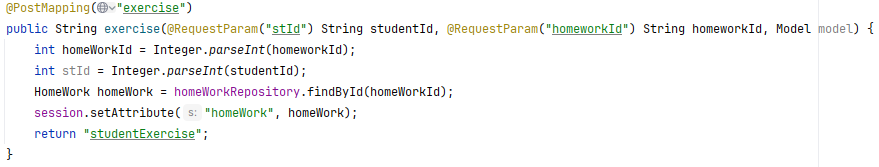
Նկար 2․65



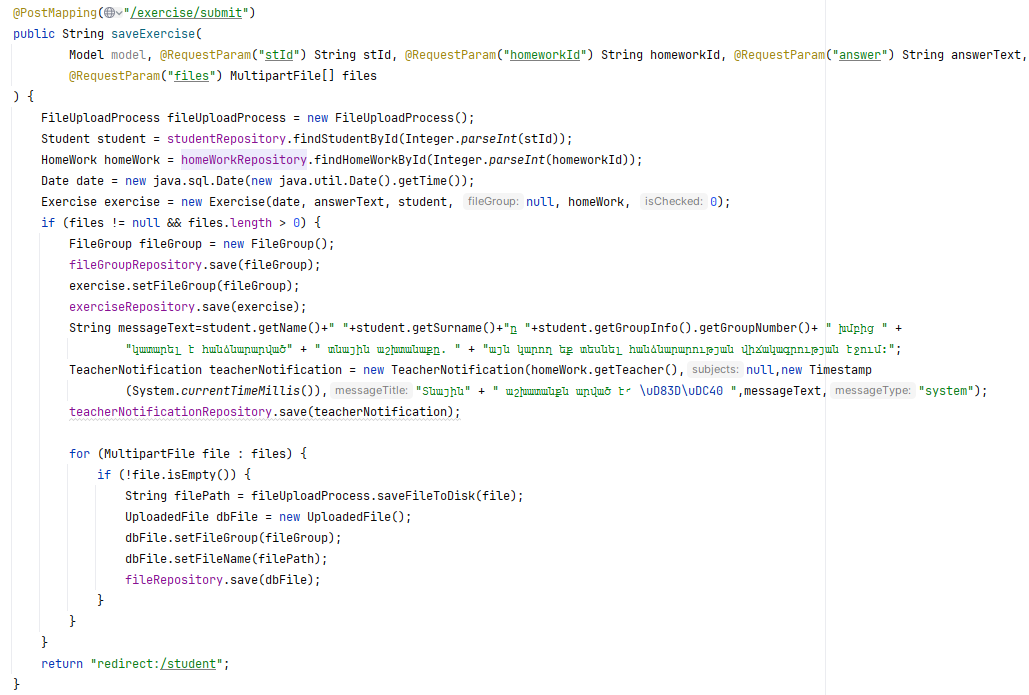
Նկար 2․66



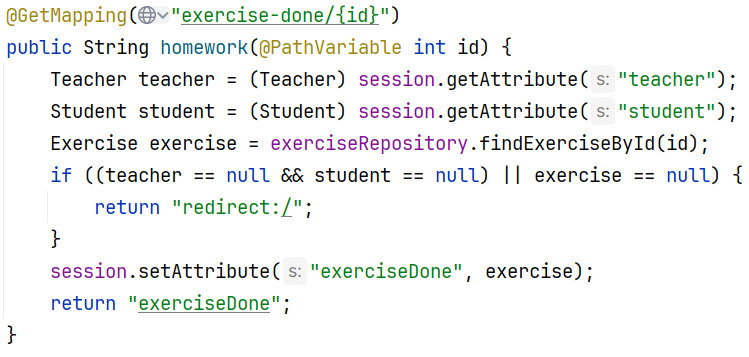
Նկար 2․67



Նկար 2․68



Նկար 2․69



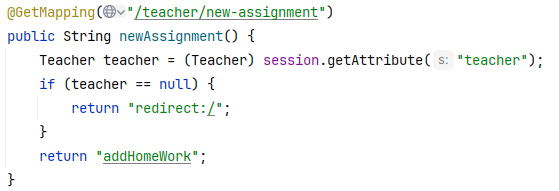
Նկար 2․70



Նկար 2․71



Նկար 2․72



Նկար 2․73

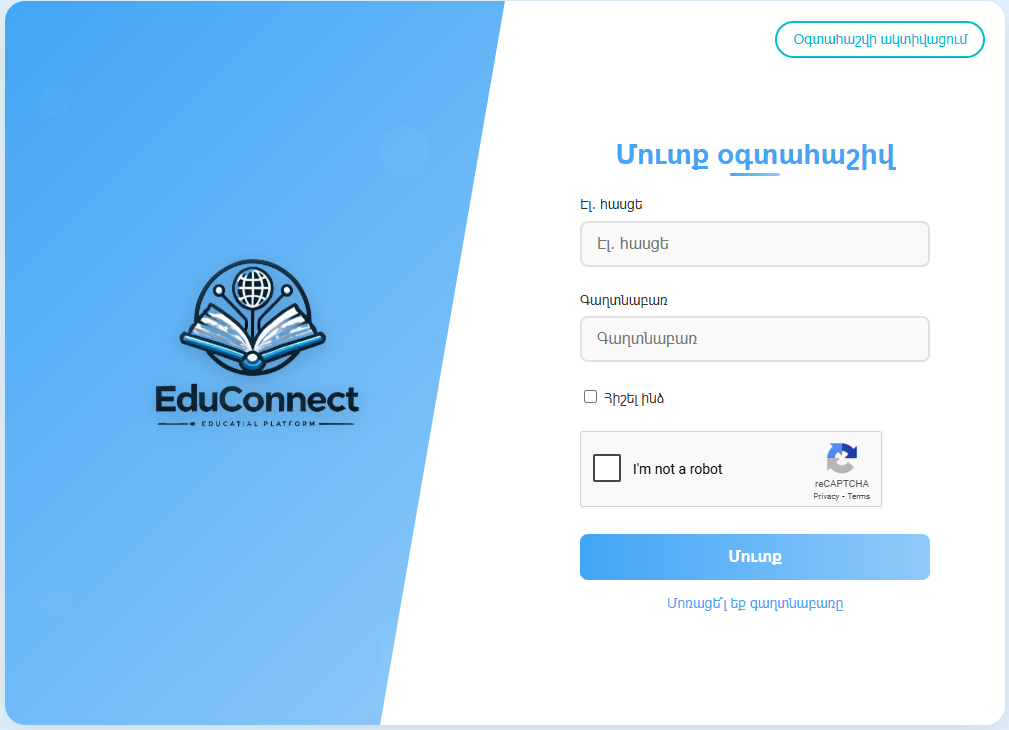


Նկար 2․74

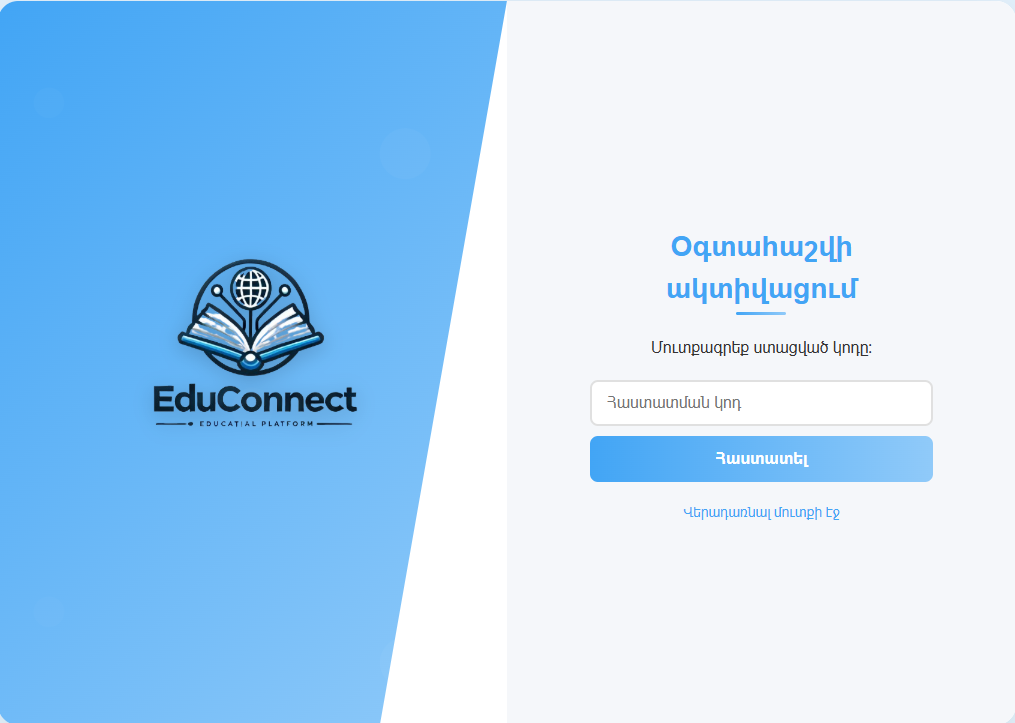
**2․6 Ծրագրի օգտագործման տեխնոլոգիան**

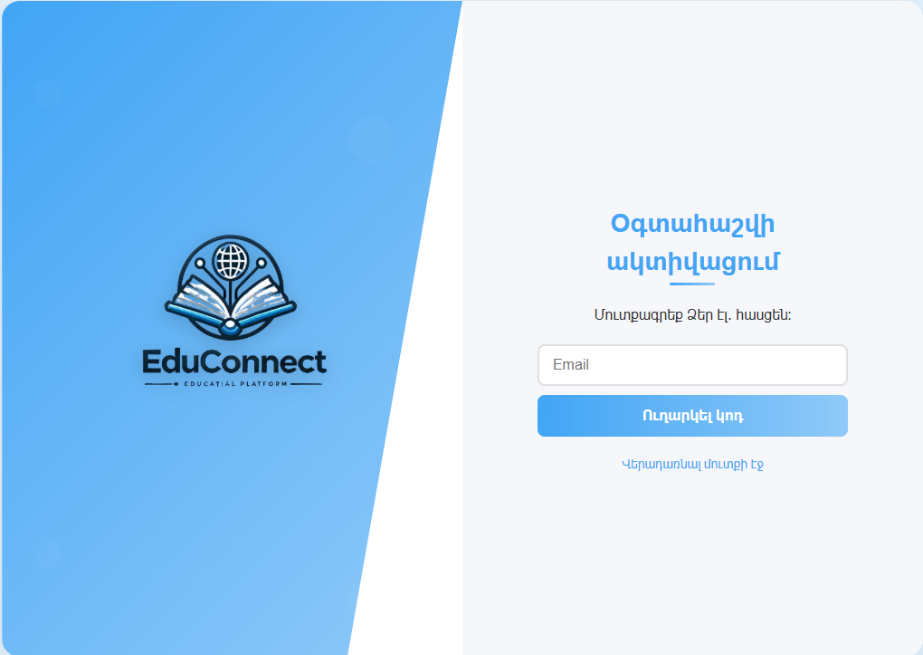
Այս ընթացքում մենք բավականաչափ ուսումնասիրեցինք ծրագրի սերվերային կառուցվածքն, իսկ այժմ կարող ենք տեսնել այն թեստային արդյունքը, որը ստացվել է ընդհանուր աշխատանքի արդյունքում։

Ծրագրի աշխատանքը սկսվում է մուտք գործելու էջից, որտեղից կարելի է շարունակել ուսանողի/դասախոսի հաշիվներ, հաշվի ակտիվացում իրականացնել և վերականգնել գաղտնաբառը(նկար 2․75- 2․98)։



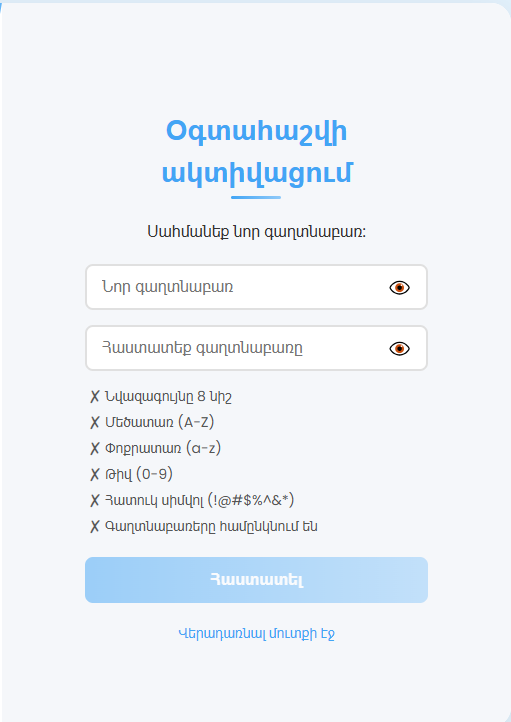
Նկար 2․75



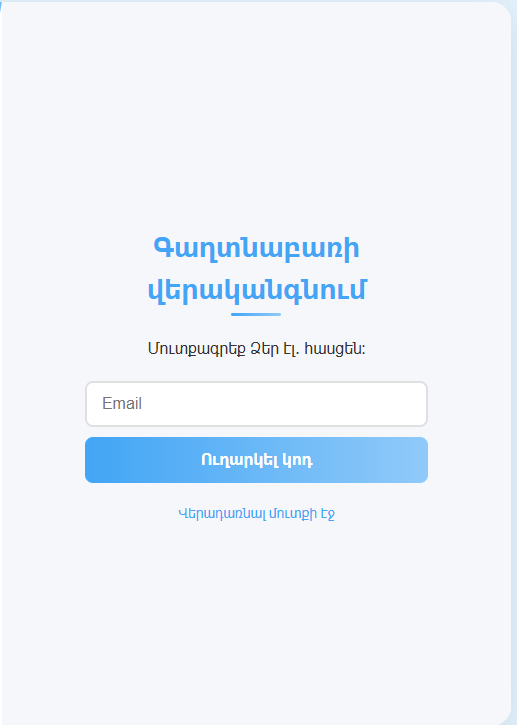


Նկար 2․76

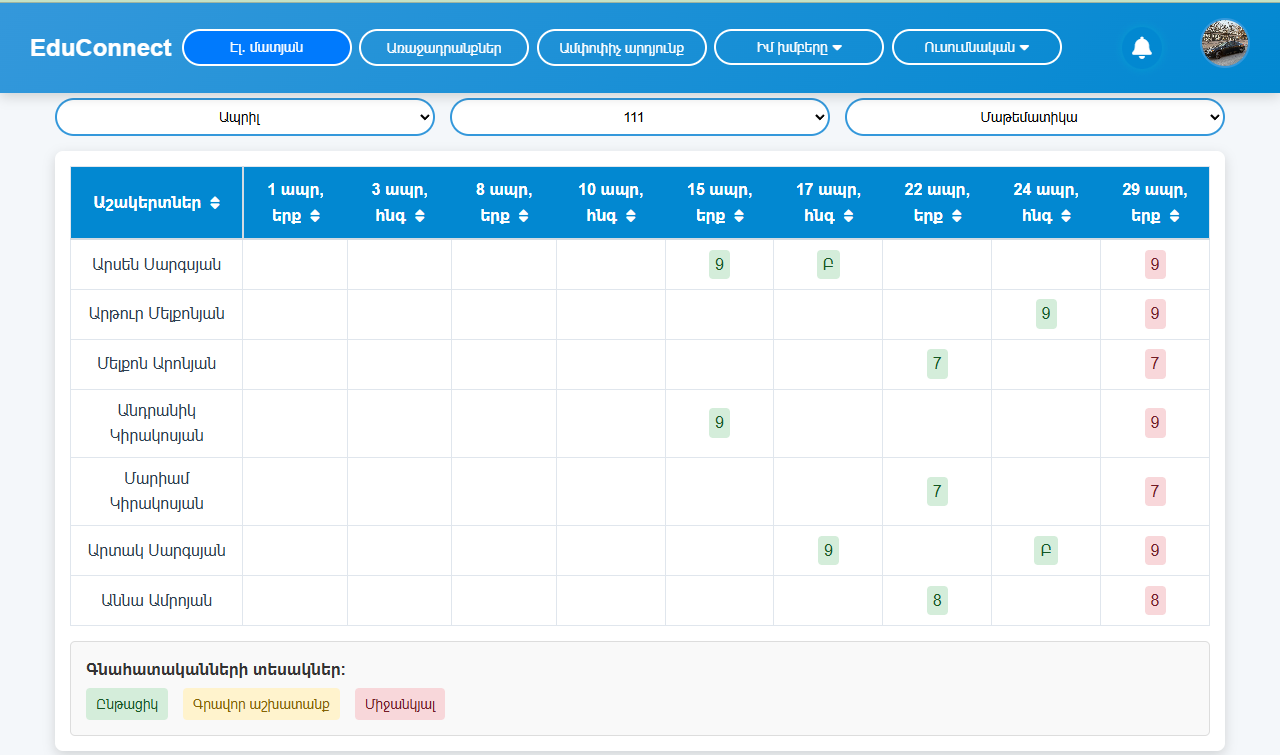
Նկար 2․77



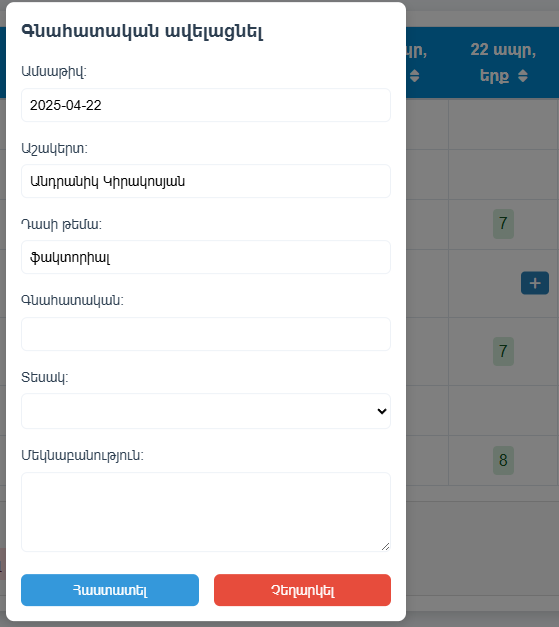
Նկար 2․78



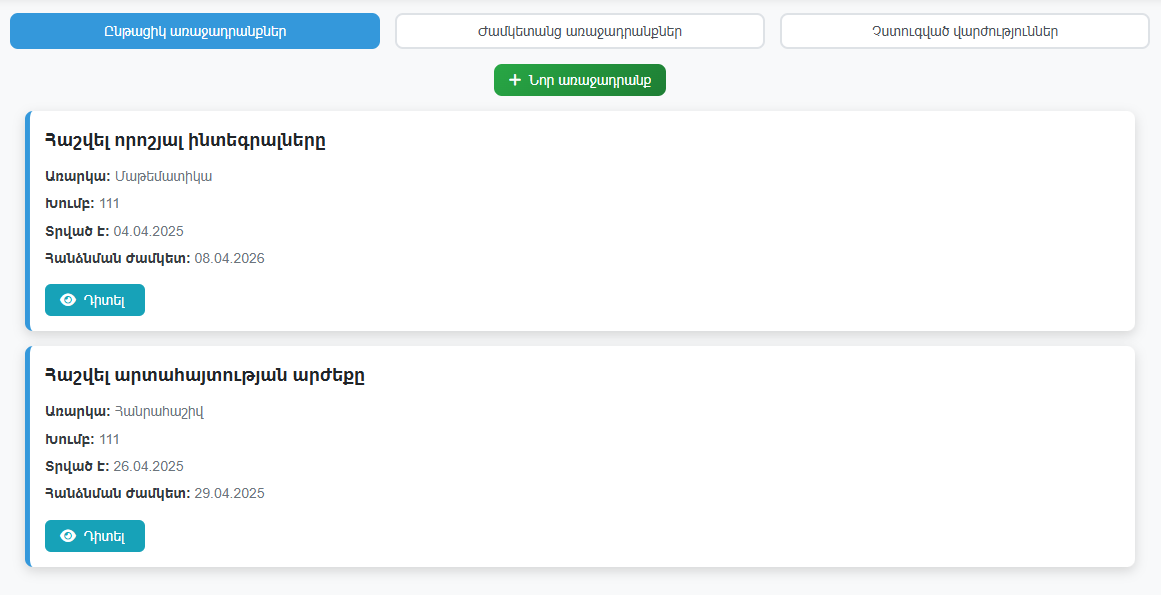
Նկար 2․79



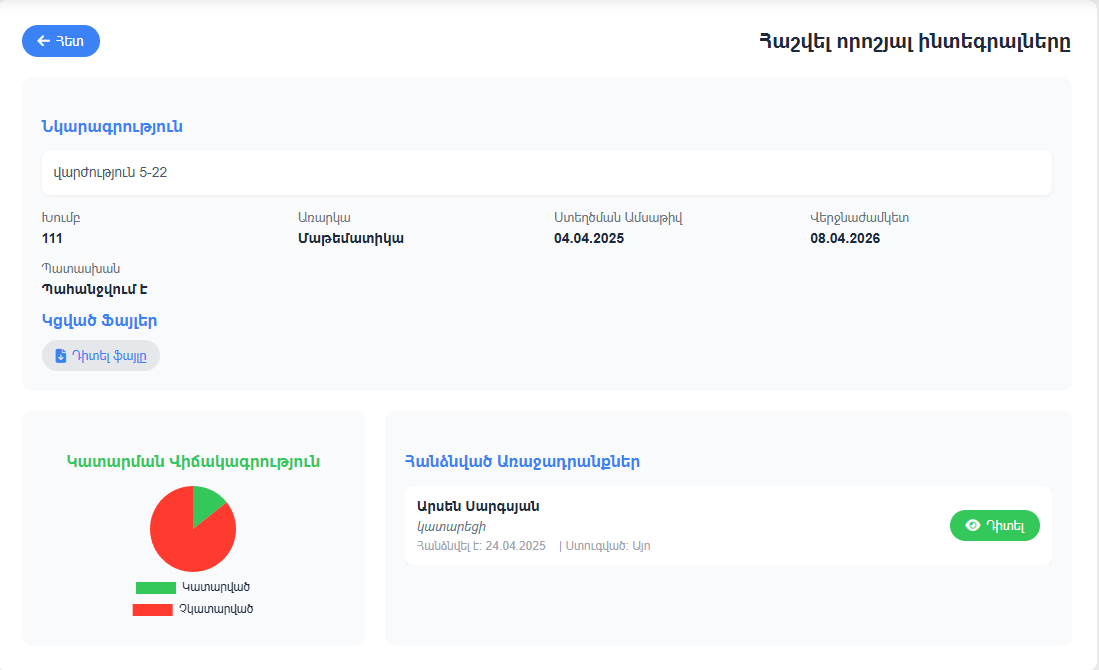
Նկար 2․80



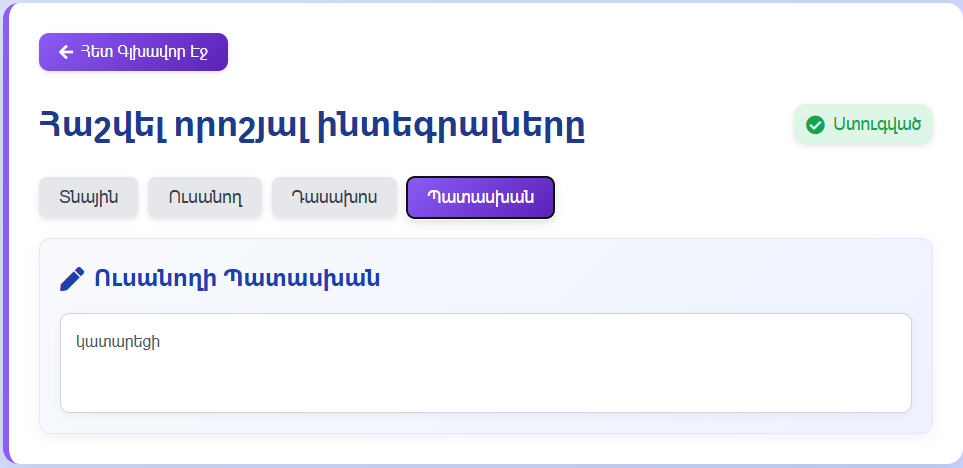
Նկար 2․81



Նկար 2․82



Նկար 2․83



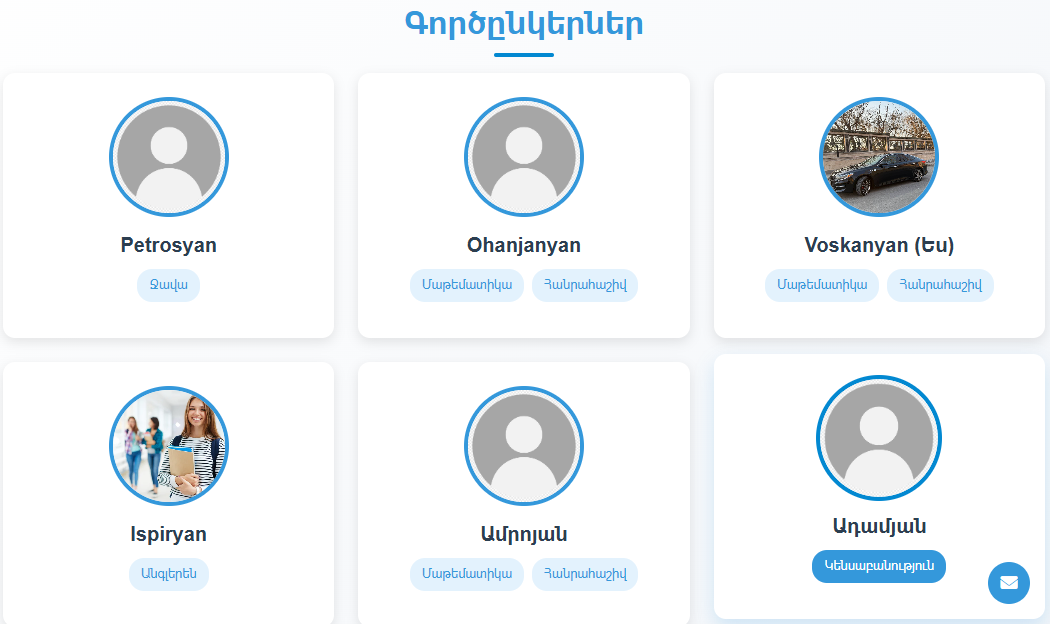
Նկար 2․84



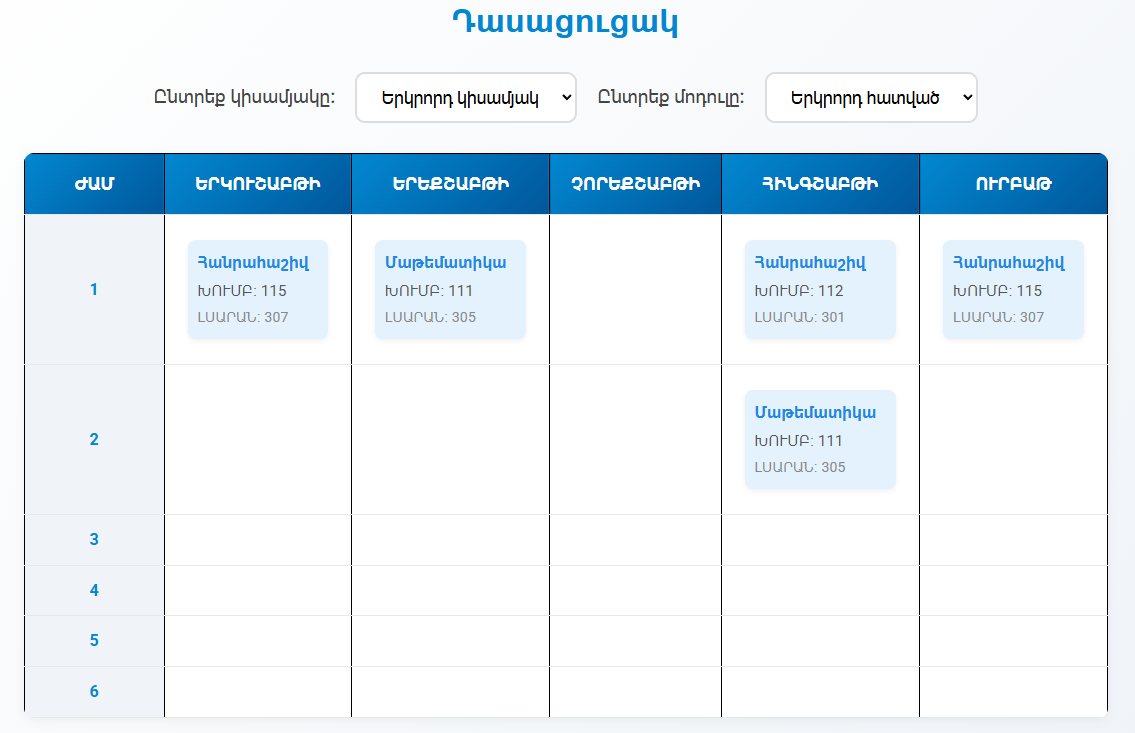
Նկար 2․85



Նկար 2․86



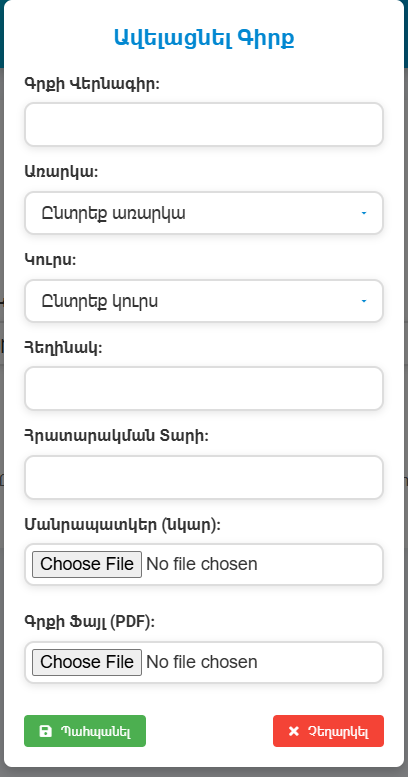
Նկար 2․87



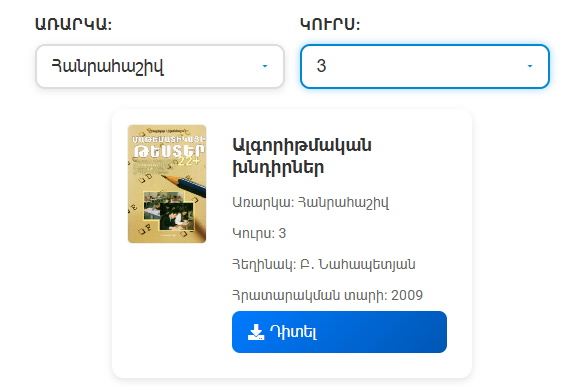
Նկար 2․88



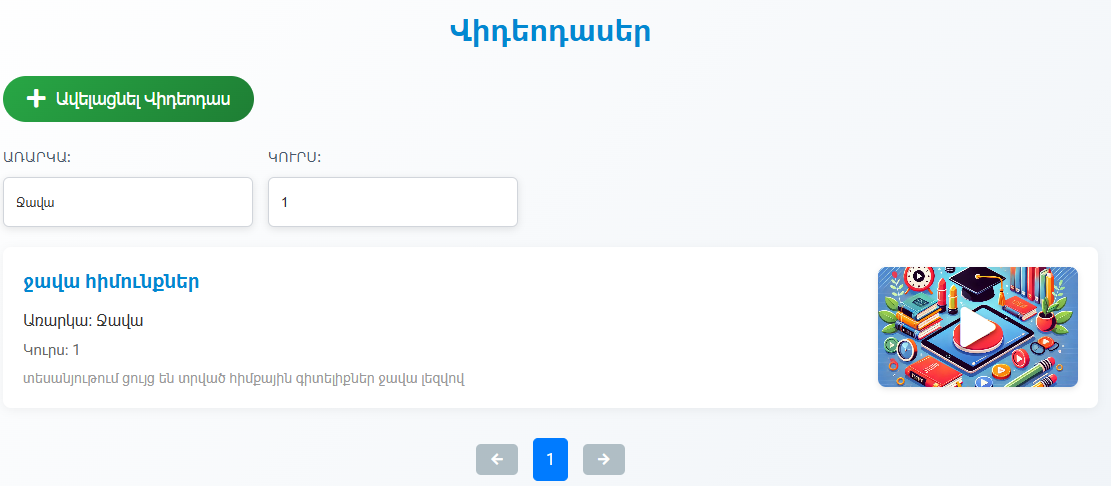
Նկար 2․89



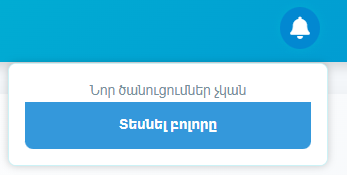
Նկար 2․90



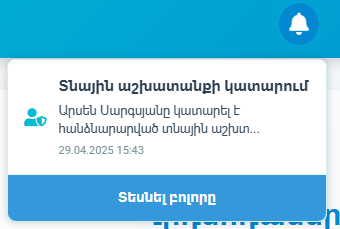
Նկար 2․91



Նկար 2․92



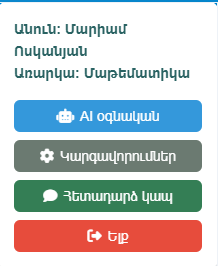
Նկար 2․93



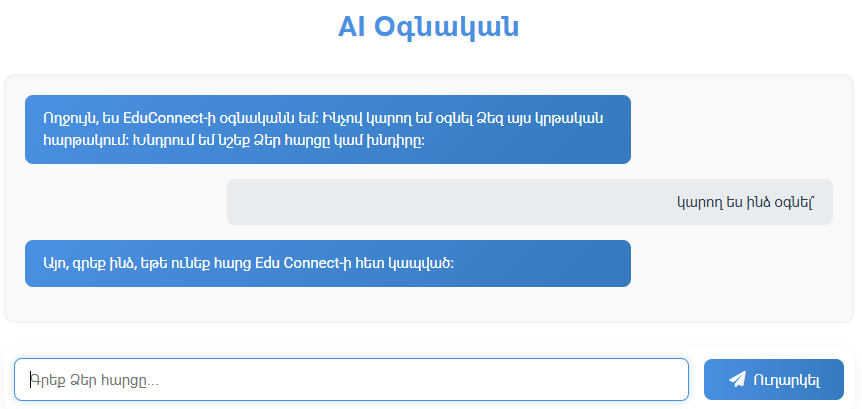
Նկար 2․94



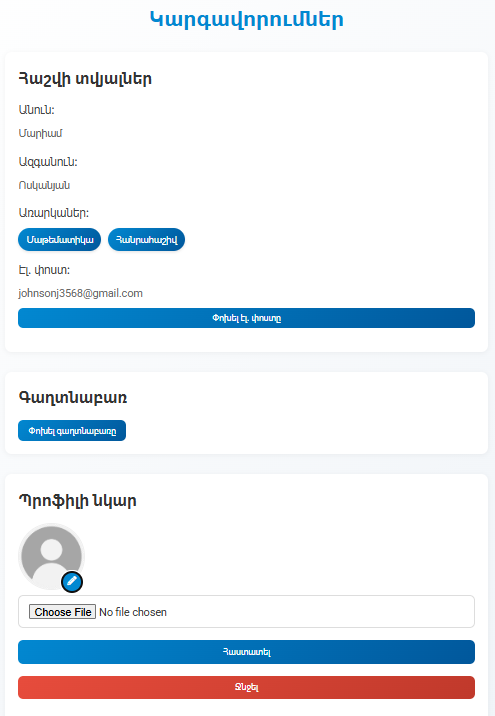
Նկար 2․94



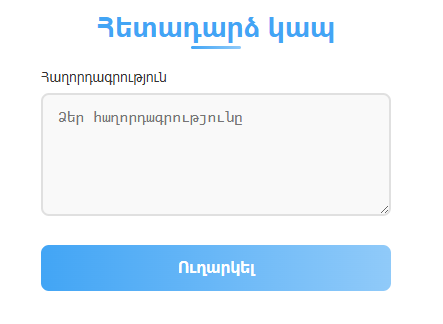
Նկար 2․95



Նկար 2․96



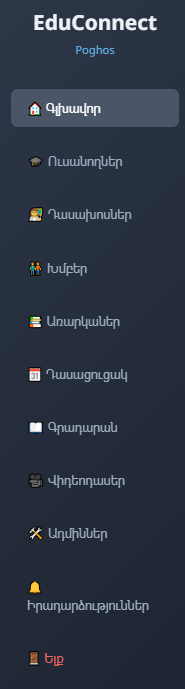
Նկար 2․97



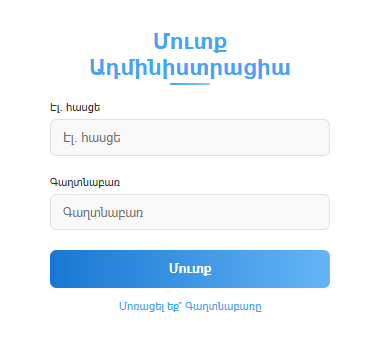
Նկար 2․98

Նախորդիվ մենք որոշ մանրամասնություններով ուսումնասիրեցինք դասախոսի էջի ֆունկցիոնալությունը, որը իհարկե ունի որոշ նմանություններ ուսանողի էջին, իսկ տրամաբանությունն առհասարակ նույնն է ողջ հարթակի աշխատանքում։

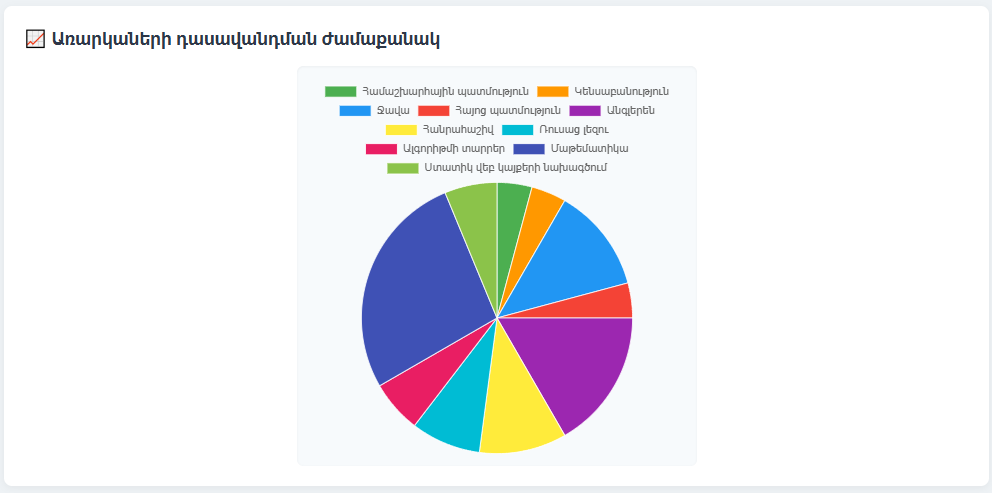
Այժմ շատ հակիրճ անդրադառնանք նաև ադմինիստրացիոն էջի գործառույթներին, նայելով միայն ամենահետաքրքիր հատվածները(նկար 2․99-2․105)։



Նկար 2․100



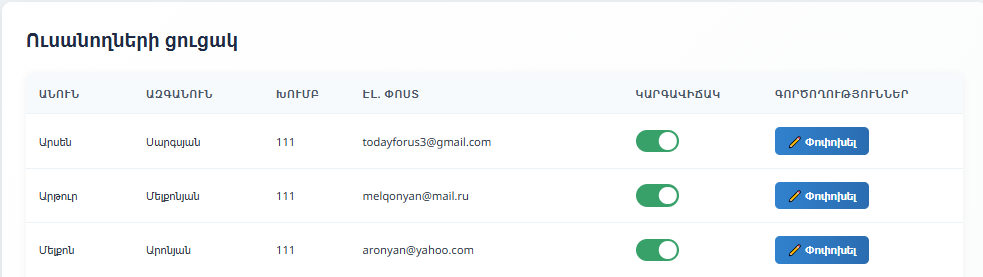
Նկար 2․99



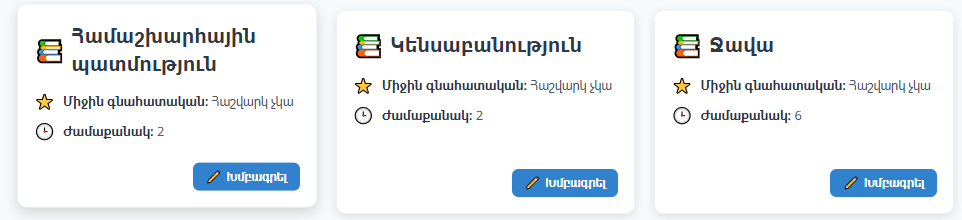
Նկար 2․101



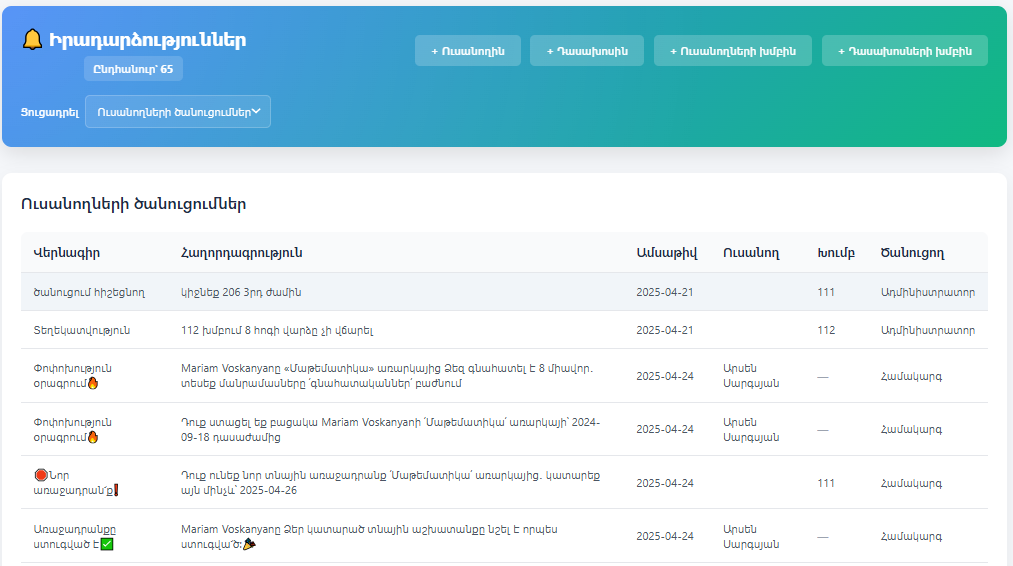
Նկար 2․102

****

Նկար 2․103



Նկար 2․104



Նկար 2․105

Հատկապես հետաքրքիր է ծանուցումների մեխանիզմի կառուցվածքն ու առաջարկվող ֆունկցիոնալությունը ադմինիստրացիայի համար(նկար 2․105)։ Ադմինիստրացիայի մնացյալ ֆունկցիոնալությունը շարունակվում է վերևում ցուցադրված նախկին նկարների տրամաբանության սահմաններում։

Նախորդիվ ներկայացված լուսանկարները, որոնք արված էին հարթակի աշխատանքի ընթացքում, ստեղծում են մոտավոր պատկերացում, թե ինչ կանոններով ու հերթականությամբ է պետք առաջնորդվել համակարգից օգտվելիս։**3․ Անվտանգության տեխնիկայի կանոնները**

Եթե փորձենք շարադրել հավելվածի օգտագործման որոշակի կանոններ, ապա կարող ենք ասել՝

* Հարկավոր է լինել ուշադիր տվյալների հետ աշխատելիս՝ հատկապես օրագիր ու մատյան,
* Հարկավոր է մշտապես հետևել ծանուցումներին, գուցե դրանք ուղղված են կոնկրետ անձի,
* Հարկավոր է ուշադիր լինել հաստատությանը էլ․ հասցե տրամադրելիս, որպեսզի գրանցումն անցնի առանց խնդիրների,
* Հարկավոր է ուշադիր լինել մուտքային տվյալների հետ, չկիսելով դրանք ոչ մեկի հետ,
* Հարկավոր է զգույշ գործել գրադարանի ու վիդեոդասերի ավելացման ժամանակ, քանի որ դրանք հասանելի են բոլորին,
* Ադմինիստրացիայի պարագայում հարկավոր է լինել աչալուրջ և օգտագործել ստեղծված գործիքները ճիշտ նպատակին ծառայեցնելու համար։

**Եզրակացություն**

Տեղեկատվական տեխնոլոգիաների արագ զարգացումը մեծապես ազդում է կրթության ոլորտի վրա՝ ստեղծելով նոր հնարավորություններ դասավանդման, կառավարման և հետադարձ կապի կազմակերպման համար։ Այս դիպլոմային նախագծի շրջանակում մշակված կրթական հարթակը հենց այդ պահանջներից է բխում՝ առաջարկելով ժամանակակից, հասանելի և արդյունավետ լուծում՝ ուսումնական գործընթացի թվայնացման համար։

Հարթակը նախատեսված է ուսումնական հաստատությունների համար և ներառում է հիմնական կրթական գործառույթներ՝ առաջադրանքների ստեղծում և հանձնում, գնահատականների վարման համակարգ, ուսումնական օրագիր, դասացուցակ, ամփոփիչ արդյունքների ավտոմատ մշակում, ճկուն ծանուցումների համակարգ, օգտահաշիվն անվտանգ դարձնող մի շարք գործառույթներ, ինչպես նաև արհեստական բանականության օգնական՝ կայքում ուղղություն ցույց տալու համար։

Նախագիծը արդիական է հատկապես Հայաստանում կրթության թվայնացման համատեքստում՝ ապահովելով արդյունավետ փոխգործակցություն ուսուցիչների, աշակերտների և ադմինիստրացիայի միջև։ Հարթակն ունի լայն հնարավորություններ ընդլայնման և այլ համակարգերի հետ ինտեգրման համար, ինչի շնորհիվ այն կարող է կիրառվել տարբեր կրթական հաստատություններում՝ սկսած հանրակրթական դպրոցներից մինչև մասնագիտական ուսումնական կենտրոններ։

Ուսումնասիրելով շուկայում առկա պրոդուկտները և ամփոփելով կատարված աշխատանքը կարելի է եզրակացնել, որ ստեղծված հարթակը կարող է բավարարել ուսումնական հաստատության նվազագույն պահանջները, կրթական գործընթացի բուն ուսումնական մասի թվայնացման շրջանակներում։ Իհարկե հասկանալի է, որ ուսումնական համակարգն ունի բազմաթիվ կողմեր՝ սկսած վարչատնտեսական համակարգերից մինչև հաշվապահություն և այլն, բայց կոնկրետ իմ հարթակի դեպքում շեշտը դրված է կրթական մասի վրա։ Միևնույն ժամանակ պիտի նշեմ, որ որքան էլ համակարգն ունի լայն ֆունկցիոնալություն, կրթական օղակն այնքան բազմաշերտ է, որ կարելի է անընդհատ ընդլայնել ու խնդիրները ավելի խորապես դիտարկել, ինչը կարվի հետագայում, եթե պրոդուկտը դառնա կիրառելի իրական միջավայրում։

Ամփոփելով պիտի նշեմ նաև, որ նման նախագծի իրականացումն ունեցավ շատ մեծ նշանակություն մասնագիտական հմտություններիս զարգացման գործում։

**Գրականություն**

1. Կ․ Պետրոսյան, ջավա լեզվի հիմունքներ, 2023 (Հասանելի չէ առցանց տարբերակ)
2. Packt publishing, Learning Spring Boot 3.0, 2022

<https://www.google.am/books/edition/Learning_Spring_Boot_3_0/soakEAAAQBAJ?hl=ru&gbpv=1&dq=2.+Packt+publishing,+Learning+Spring+Boot+3.0&pg=PA23&printsec=frontcover>

1. Packt publishing, Spring Security, 2024

<https://www.google.am/books/edition/Spring_Security/uBUMEQAAQBAJ?hl=ru&gbpv=1&dq=Learning+Spring+Boot+3.0:+Simplify+the+Development+of+Production-grade+Applications+Using+Java+and+Spring&pg=PA567&printsec=frontcover>

1. Sun Microsystems, Java Tutorial Third Edition, 2001

<https://www.google.am/books/edition/The_Java_Tutorial/rl6SFYAdL6YC?hl=ru&gbpv=1&dq=sun%20microsystems%20java%20third%20edition&pg=PT2&printsec=frontcover>

1. Dave Thau, The Book Of JavaScript, 2nd Edition, 2007

<https://www.google.am/books/edition/The_Book_of_JavaScript_2nd_Edition/MrHaC5HAva8C?hl=ru&gbpv=1&dq=books%20for%20ajax%20js&pg=PR3&printsec=frontcover>

1. Packt publishing, Java Hibernate Cookbook, 2015

<https://www.google.am/books/edition/Java_Hibernate_Cookbook/rdlOCwAAQBAJ?hl=ru&gbpv=1&dq=hibernate%20java%20books&pg=PP2&printsec=frontcover>

1. Paul Deck, Spring MVC Second Edition, 2016

<https://www.google.am/books/edition/Spring_MVC_A_Tutorial_Second_Edition/O1LzCwAAQBAJ?hl=ru&gbpv=1&dq=book%20about%20mvc%20spring&pg=PT7&printsec=frontcover>