ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
Ш. ЕСЕНОВ атындағы КАСПИЙ ТЕХНОЛОГИЯЛАР ЖӘНЕ ИНЖИНИРИНГ УНИВЕРСИТЕТІ

Қазақ-Неміс тұрақты инженерия институты

ОҚУ ТӘЖІРИБЕСІ ТУРАЛЫ ЕСЕП

Жоба атауы: Student Information System

Білім беру бағдарламасы: 6B07119 Mechatronics
(ББББ ның цифрлық атауы)

Жетекшісі	_____ (қолы)	Кенжебаева Ж.Е. (Жетекшінің Т.А.Ж)
Орындаған	_____ (қолы)	Оспанов Шахмұрат (Студенттің Т.А.Ж)
Орындаған	_____ (қолы)	Берікұлы Сержан (Студенттің Т.А.Ж)
Орындаған	_____ (қолы)	Ұзақберді Жансерік (Студенттің Т.А.Ж)
Орындаған	_____ (қолы)	Ерғожа Әлишер (Студенттің Т.А.Ж)
Орындаған	_____ (қолы)	Мұхтарұлы Диас (Студенттің І.А.Ж)

АҚТАУ 2025

Мазмұны

КІРІСПЕ **3**
Жобаның мақсаты: 3
Негізгі міндеттері: 3
Тақырыптың өзектілігі: 4
БІРІНШІ БӨЛІМ **5**
1. Сайт аналогтары: 5
2. Жоба архитектурасы 11
2.1 Бэкенд архитетурасы 11
2.2 Фронтенд (Клиенттік) Архитектурасы 12
2.3 Дерекқор архитектурасы 13
Деректердің Тұтастығын Қамтамасыз ету Рөлі 15
ЕКІНШІ БӨЛІМ **16**
2.1 Бэкенд инструменттері (Node.js) 16
2.2. JavaScript-тің Негізгі Міндеттері 17
2.3. HTML және CSS 19
2.4. MySQL 19
2.5 Қосымша қолданылған кітапханалар 21
ҮШІНШІ БӨЛІМ **23**
3.1 Сабақ кестесін басқару модулі: 23
3.2 Академиялық календарь модулы 26
3.3 Бағалау Және Нәтижелер Модулі 28
ҚОРЫТЫНДЫ **41**
1. Жобаның Нәтижелері және Мақсаттардың Орындалуы 41
2. Архитектуралық Шешімдердің Тиімділігі 41
3. Келешектегі Даму Жоспарлары 42
ПАЙДАЛЫНЫЛҒАН ӘДЕБИЕТТЕР **43**

КІРІСПЕ

Жобаның мақсаты:

Жобаның негізгі мақсаты – заманауи веб-технологияларды (Node.js, MySQL) қолдана отырып, университет немесе колледж деңгейіндегі шағын студенттік порталдың жұмыс істейтін үлгісін әзірлеу және іске қосу. Бұл портал оқу процесінің негізгі ақпараттық қажеттіліктерін қанағаттандыруға бағытталған.

Негізгі міндеттері:

1. Пайдаланушыларды Аутентификациялау және Авторизациялау: Қауіпсіз тіркелу, кіру және сессияны басқару жүйесін құру. Бұл жүйе әрбір пайдаланушының өзінің рөліне (студент, оқытушы, әкімші) сәйкес келетін ақпаратқа ғана қол жеткізуін қамтамасыз етеді.
2. Деректердің Тұтастығы мен Қауіпсіздігін Қамтамасыз ету: Парольдерді хештеу және SQL транзакцияларын қолдану арқылы деректерді сақтау мен жаңартудың жоғары сенімділігіне қол жеткізу.
3. Оқу Кестесін Басқарудың Икемді Жүйесін Ұсыну: Студенттер үшін өз тобының кестесін қарауды, ал оқытушылар үшін кестені тиімді түрде жаңартуды (сақтауды) қамтамасыз ету.
4. Курстар Мен Ақпараттық Қолжетімділікті Жеңілдету: Студенттер үшін олардың тобына бекітілген курстар туралы, ал оқытушылар үшін олардың жүргізетін пәндері туралы мәліметтерді ұсыну.
5. Пайдаланушы Интерфейсін Әзірлеу: Әр рөлге арналған интуитивті және функционалды Dashboard (басқару панелі) құру, соның ішінде профильді басқару, парольді ауыстыру және аватарды жүктеу мүмкіндіктерін қосу
6. Мұғалімдер үшін ыңғайлы, оңай бағалау жүйесін қамтамасыз ету. Автономды семместірлік бағаны қорытындылау. Студент үшін ыңғайлы баға кестесін көрсетуге мүмкіндік беру.

Тақырыптың өзектілігі:

Қазіргі білім беру жүйесінде цифрландыру және оқу процесінің ақпараттық ашықтығы басты талап болып табылады. Пандемиядан кейінгі кезеңде қашықтықтан және гибридті оқыту форматының өсуіне байланысты, студенттер мен оқытушылардың барлық қажетті ақпаратқа бір терезеден, кез келген уақытта және кез келген құрылғыдан қол жеткізу мүмкіндігі аса маңызды.

Бұл шағын студенттік портал:

1. Уақытты үнемдейді: Ақпаратты (кесте, курстар) қағаз тасымалдағыштардан немесе көптеген арналардан іздеу қажеттілігін жояды.
2. Басқаруды жеңілдетеді: Оқытушылар мен әкімшілерге кестелер мен курстарды басқаруда тиімді, орталықтандырылған құрал береді
3. Қауіпсіздік арттырады: Қазіргі заманғы қауіпсіздік стандарттарын (хештеу, сессияны басқару) қолдану арқылы пайдаланушылардың жеке деректерінің қорғалуын қамтамасыз етеді.

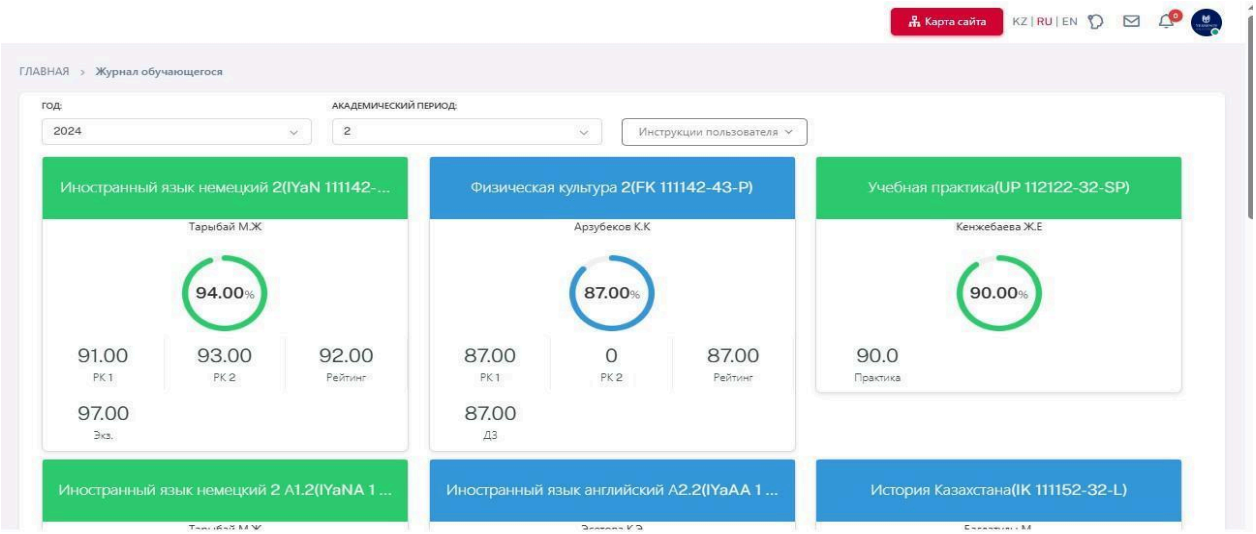
Осылайша, жоба білім беру мекемесінің операциялық тиімділігін арттыруға және оқу процесінің бардық қатысушылары үшін ақпарат алмасудың сапасын жақсартуға тікелей ықпал етеді.

БІРІНШІ БӨЛІМ

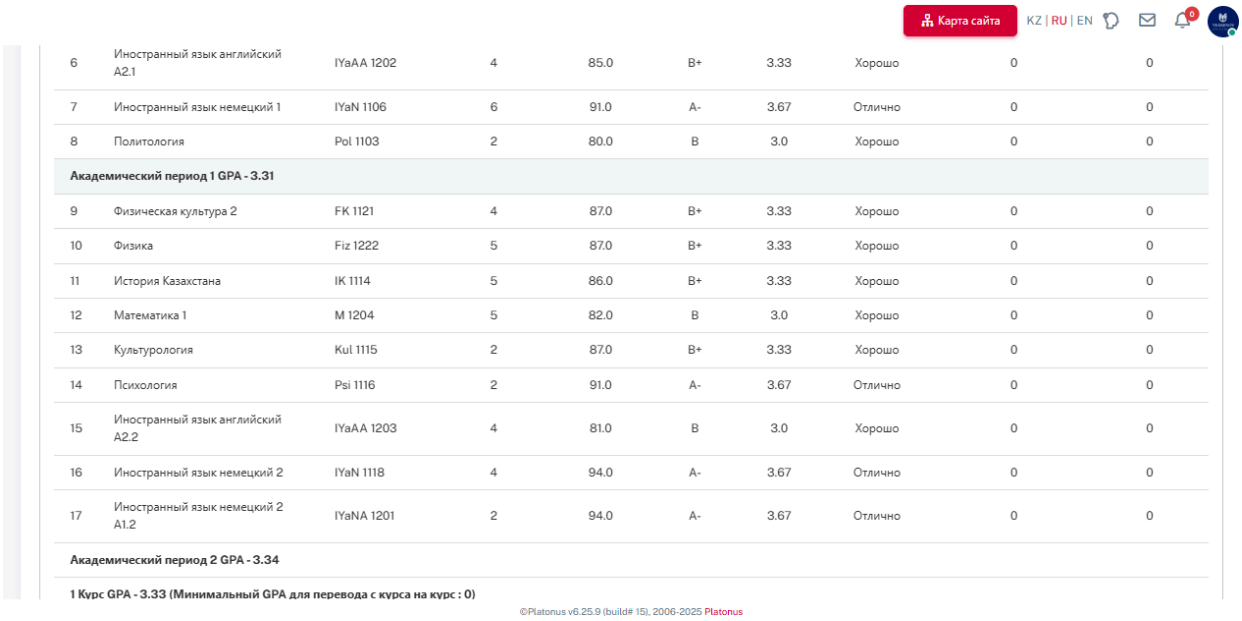
1. Сайт аналогтары:

Қазіргі уақытта студенттік порталдар өте көп. Олар ішіндегі кеңінен таралғандары: Moodle, Platonus, Canvas, Univer және т.б. Бұл порталдар жалпы әлемде университтерде ауқымды қолданылады және оқу процесін қайда жеңілдетеді.

Platonus көбінесе Қазақстандағы колледждер мен университтерде па йдаланылады. Платонус қазақ және орыс тілдеріне толық бейімделген. Білім беру ұйымдарының ішкі жүйесіне сай, яғни деканат, бағалар, сабақ кестесі, студент деректері, журналдар т.б бәрі бір жерде. Басқаларындай автоматтандырылған бағалау(1.1 Сурет) және есеп беру жүйесі(1.2 Сурет) жақсы жұмыс жасайды. Және интеграциясы жеңіл – ҚР БҒМ талаптарына сай есеп беру мүмкіндігі бар (мысалы, академиялық үлгерім т.б).



1.1 - Сурет. Бағалар жүйесі.



Карта сайта KZ | RU | EN

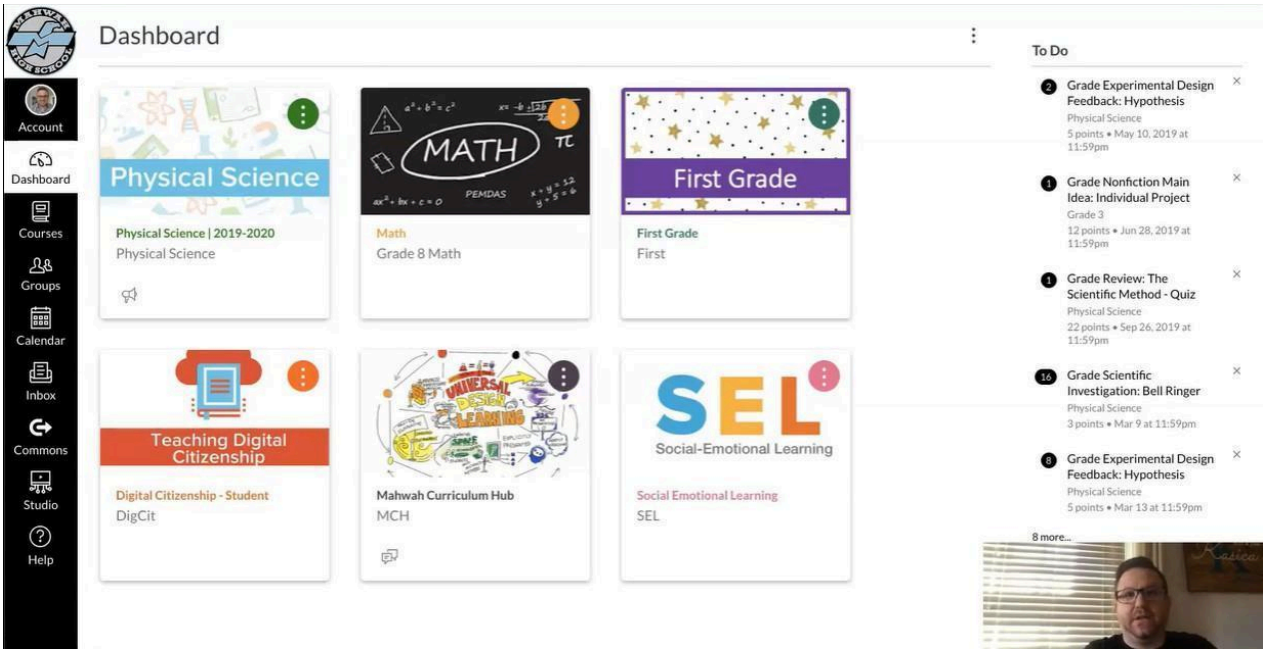
6	Иностранный язык английский A2.1	IYaAA 1202	4	85.0	B+	3.33	Хорошо	0	0
7	Иностранный язык немецкий 1	IYaN 1106	6	91.0	A-	3.67	Отлично	0	0
8	Политология	Pol 1103	2	80.0	B	3.0	Хорошо	0	0
Академический период 1 GPA - 3.31									
9	Физическая культура 2	FK 1121	4	87.0	B+	3.33	Хорошо	0	0
10	Физика	Fiz 1222	5	87.0	B+	3.33	Хорошо	0	0
11	История Казахстана	IK 1114	5	86.0	B+	3.33	Хорошо	0	0
12	Математика 1	M 1204	5	82.0	B	3.0	Хорошо	0	0
13	Культурология	Kul 1115	2	87.0	B+	3.33	Хорошо	0	0
14	Психология	Psi 1116	2	91.0	A-	3.67	Отлично	0	0
15	Иностранный язык английский A2.2	IYaAA 1203	4	81.0	B	3.0	Хорошо	0	0
16	Иностранный язык немецкий 2	IYaN 1118	4	94.0	A-	3.67	Отлично	0	0
17	Иностранный язык немецкий 2 A1.2	IYaNA 1201	2	94.0	A-	3.67	Отлично	0	0
Академический период 2 GPA - 3.34									

1 Курс GPA - 3.33 (Минимальный GPA для перевода с курса на курс : 0)

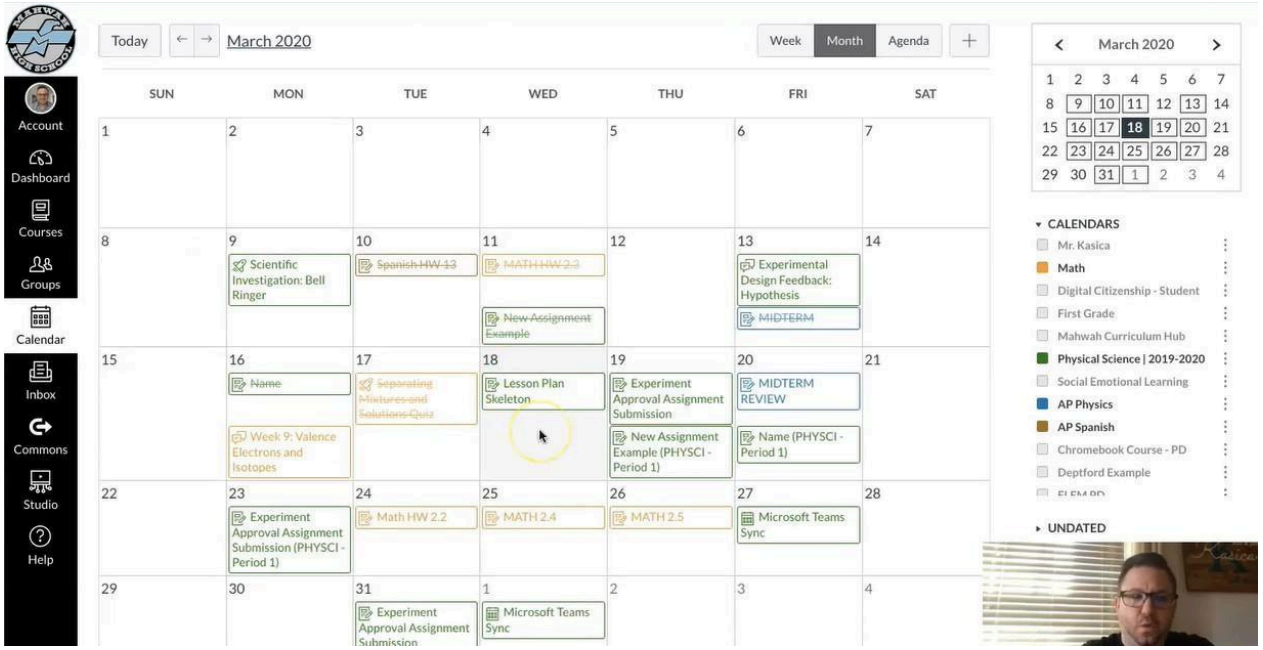
©Platonus v6.25.9 (build# 15), 2006-2025 Platonus

1.2 - Сурет. Транскрипт.

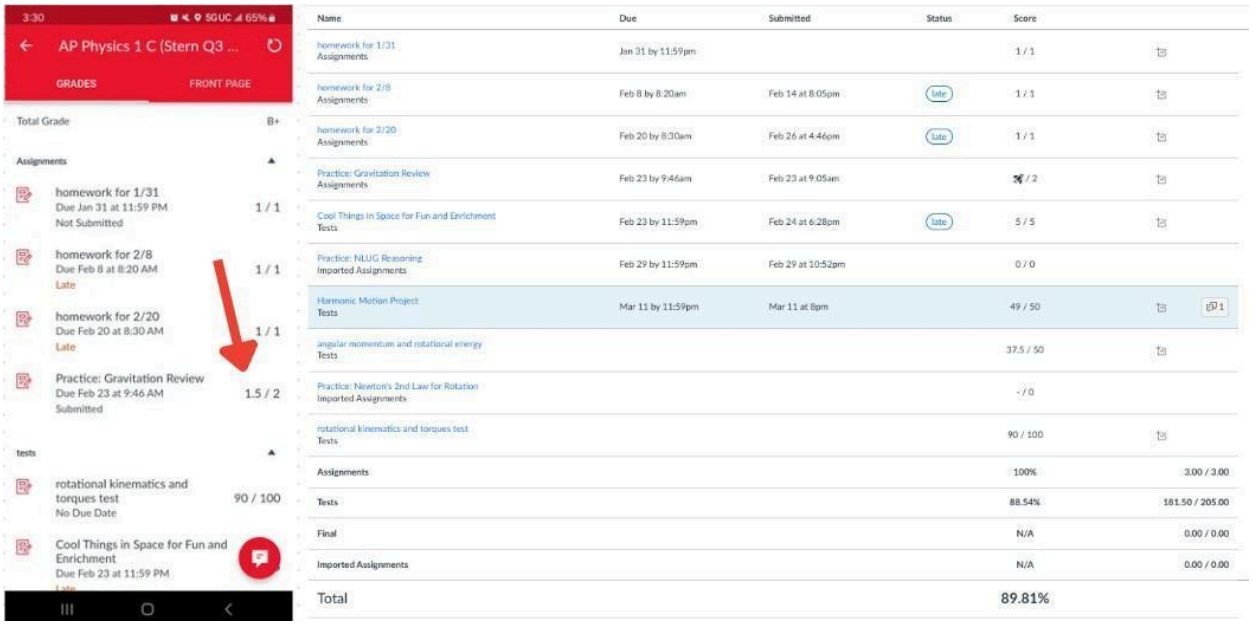
Canvas әлемдік деңгейде кең таралған, мысалға айтсақ АҚШ, Еуропадағы университеттерде. Канвас интерфейсі айтарлықтай заманауй және ыңғайлы навигациямен жабдықталған(1.3Сурет). Интерактивті құралдар көп, мәселен форумдар, тест, бейнежазбалар, тапсырма беру, кері байланыс(1.4 Сурет) және ыңғайлы бағалау системасы (1.5 Сурет). Бұлттық сақтау жүйелерімен (Google Drive, OneDrive) оңай байланысады. Аналитика және статистика арқылы студенттің белсенділігін көруге болады. Алайда Canvas интернетке тәуелді, яғни онлайн режимде жақсы жұмыс істейді. Және бір кемшілігі толық функционалды нұсқасы ақылы, бірақта бұл университетке қиын емес деп санаймыз. Және интерфейс кейде ағылшын тілінде болғандықтар, басқа тілге бейімделу уақытын қажет етеді.



1.3 - Сурет. Canvas басты беті.



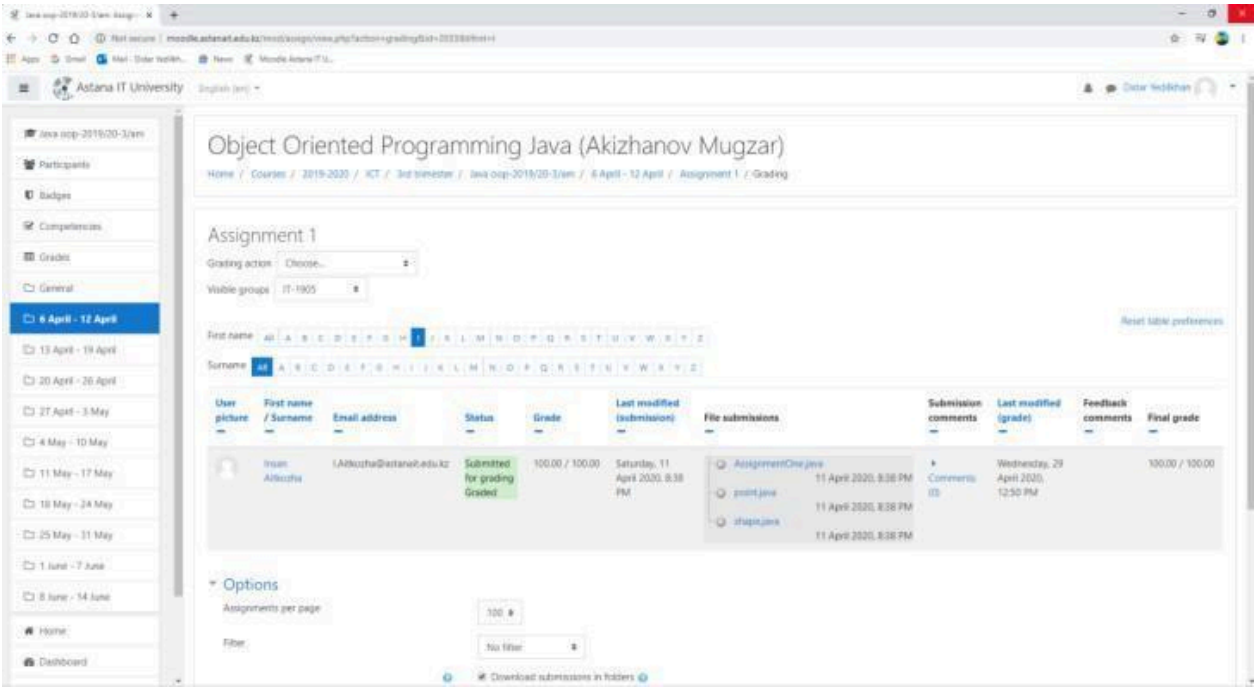
1.4 - Сурет.Календарь жүйесі



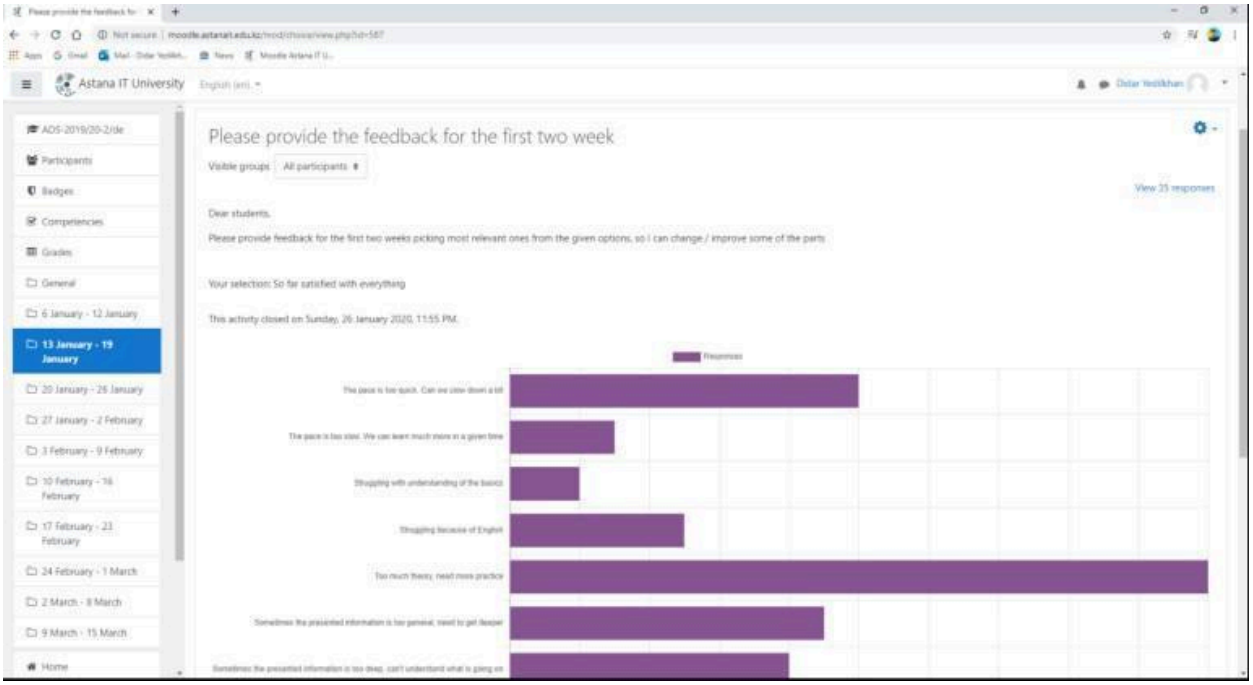
1.5 - Сурет.Бағалар тақтасы

Moodle әлем бойынша ең көп қолдынылатын ашық кодты (open-source) LMS. Бұл портал толығымен тегін және икемді. Кез келген универ өзіне ыңғайлы, қажет етіп жекешелендіре алады (плагиндер, дизайн, модульдер). Сонымен қатар көп функциялы: форум, тест, чат, файл жүктеу, бағалау н\е журнал деген сияқты. Айтақандай мұғалім үшін порталда көптеген функциялар жұмыс жасайды, яғни жеке курсымен жұмыс жасаудан(1.6 Сурет) бастап онлайн сабақтар жүргізіп, кері байланыс алуға дейін(1.8 Сурет). Бұл өз кезегінле Және бұл портал онлайн және оффлайн жұмыс жасау мүмкіндігі бар. Бірақ кейде интерфейсі ескі және күрделі көрінеді. Және басқада порталдар секілді әрине бұл порталда сервер баптауы мен техникалық қызметі арнайы маманды қажет етеді. Тағы бір айтатын нәрсе жоғарыда айтылғандай ккп плагин орнатылса жүйе баяулайды.

1.6 - Сурет.Мұғалім үшін курсы беті



1.7 - Сурет.Оқушыларды бағалау мен бақылау



1.8 - Сурет.Өткен сабақ туралы анализ

Критерий	Platonus	Canvas	Moodle
Тілі	Қазақша\орысша	Ағылшын(кейде көп тілді)	Көптілді
Қолайлылық	Орташа	Өте жоғары	Орташа
Бағасы	Лицензиялы	Ақылы	Тегін
Көрінісі(UI)	Ескі	Заманауй	Классикалық н\е жеке
Техникалық баптау	Оңай	Дайын жүйе	Өз бетімен баптау қажет
Интерактив	Төмен	Жоғары	Орташа\Жоғары

1.1- Кесте. Аналогтар анализі

Осы сайттарды назарға ала отыра (1-Кесте) бізде сайтымызға бардық студенттік порталдар секілді сабақ кестесі, бағалар, және курстарды қосамыз.

Осы аналогтардағы студент пен мұғалім жұмысын жеңілдедетін динамикалармен функциялармен проектті дамытамыз.

2. Жоба архитектурасы.

2.1 Бэкенд архитетурасы.

Жоба Backend-as-a-Service (BaaS) архитектурасын қолданады, мұнда Node.js-тегі сервер API-сервері және статикалық файлдар сервері ретінде жұмыс істейді.

Бэкенд логикалық тұрғыдан екі негізгі модульге бөлінген:

1. Сервер (server.js): HTTP-сұрауларды маршруттауға, статикалық файлдарды беруге және бизнес-логиканы шақырмас бұрын авторизация мен рөлдерді алдын ала тексеруге жауап береді.
2. Авторизация және Деректер Модулі (auth.js): Барлық бизнес-логиканы және дерекқормен тікелей өзара әрекеттесуді қамтиды.

Бекендтің басты міндеті – рөлдік модельді қатаң бақылау (users.role).

- Қолжетімділікті фильтрлеу: Серверге келіп түскен әрбір сұраныс checkSession функциясы арқылы тексеріледі.
- Рұқсаттық Тексеру: server.js ішіндегі маршруттық логика пайдаланушының белгілі бір әрекетті орындауға (мысалы, saveSchedule - тек оқытушыға рұқсат) рұқсаты бар-жоғын тексереді. Бұл фронтендтегі кез келген айла-шарғыдан қорғаныстың соңғы шебі болып табылады.

Бекенд барлық деректер операцияларын тиімді басқарады:

- Дерекқормен Байланыс: Қосылу пулы арқылы дерекқорға сенімді және өнімді қатынауды қамтамасыз етеді (db.js).

- Транзакциялық Басқару: Кесте сияқты маңызды деректерді жаңарту кезінде (saveSchedule), деректердің тұтастығын сақтау үшін SQL транзакцияларын қолданады. Бұл сервердің кез келген қателік кезінде деректерді бастапқы күйіне қайтаруына кепілдік береді.

2.2 Фронтенд (Клиенттік) Архитектурасы.

Архитектуралық Үлгі: Динамикалық Бірбеттік Қосымша (SPA элементтері бар). Жоба таза SPA (Single Page Application) болмаса да, ол SPA элементтерін қолданады. Себебі:

1. Негізгі интерфейс Dashboard (dashboard.html немесе оған ұқсас) бір рет жүктеледі.
2. Содан кейін барлық ақпарат (кесте, курстар, профиль деректері) AJAX-сұраулары (асинхронды API шақырулары) арқылы жүктеледі.
3. Пайдаланушы интерфейсі бетті толық қайта жүктемей, тек JavaScript арқылы динамикалық түрде жаңартылады.

Фронтенд архитектурасының басты ерекшелігі – оның API-ға бағытталуы:

- Асинхронды Деректер Алмасу: Интерфейстің көп бөлігі бетті толық қайта жүктемей, тек API маршруттарына (мысалы, /api/get_schedule) сұрау жіберу арқылы жаңартылады. Бұл пайдаланушы тәжірибесін жылдам әрі үздіксіз етеді.
- Деректерді Визуализациялау: Бекендтен JSON форматында алынған деректер Интерактивті Логика арқылы өңделеді де, оқуға оңай HTML кестелеріне немесе тізімдерге айналып, Құрылымдық Қабаттағы тиісті орындарға орналастырылады.

Бекендтен алынған пайдаланушының рөліне (student, teacher, admin) сәйкес, фронтенд интерфейсті адаптациялайды:

- Қажетті Элементтерді Көрсету: Тек рұқсат етілген элементтерді көрсетеді (мысалы, Админ ғана КУРСТЫ ҚОСУ формасын көрсету).
- Функционалды Шектеу: Белгілі бір рөлдер үшін түймелерді немесе форма өрістерін өшіру (disable) арқылы рұқсат етілмеген әрекеттерді болдырмайды.

Қорытынды: Фронтенд архитектурасы жеңіл, динамикалық және API-ға бағытталған (API-Driven) болып табылады. Ол берілген ролдік модельді сақтай отырып, бекендтен алынған деректерді тиімді көрсетеді, және JavaScript арқылы DOM-ды тікелей басқаруға негізделген.

2.3 Дерекқор архитектурасы.

Реляциялық моделі.

Архитектураның негізі — Реляциялық Модель, мұнда деректер бірнеше логикалық кестелерге бөлінген, олар бір-бірімен айқын байланыстар арқылы біріктірілген.

- Users Кестесі: Барлық пайдаланушылар туралы деректердің орталық тізімі. Бұл кесте басқа барлық кестелер үшін бастапқы кілт (Primary Key) ретінде қызмет етеді.
- Sessions Кестесі: Қауіпсіздік үшін қолданылады. Users кестесімен байланысы бар, авторизацияланған пайдаланушы күйін және оның жарамдылық мерзімін бақылайды.
- Schedule & Courses Кестелері: Негізгі оқу ақпаратын сақтайды. Users кестесімен сыртқы кілттер (Foreign Keys) арқылы байланысады, бұл кестелердегі мұғалімнің кім екенін анықтауға мүмкіндік береді.
- admin_messages кестесі: Студент жағынан админге жазылатын хабарламаларды сақтайды. (is_read ? - логикасы)
- students_grade: Студент үшін Резултаты бөліміндегі OverViewResults тақтасының мәліметтрін сақтайды.

- academic_events: Басты беттегі Академиялық календарь мәліметтерін сақтайды.
- assessment_types: Мұғалім үшін баға қою кестесінің шаблондарын сақтайды.
- student_assessments: Студенттердің негізгі бағаларын сақтайды.

Кесте	Негізгі рөл	Негізгі бағандар
users	Пайдаланушыларды сақтау.(Студент, оқытушы, Админ)	-id, -username, -password(hashed), -role, -group_name, -full_name, (-avatar_path)
session	Пайдаланушының авторизацияланған күйін сақтау.	-id, -user_id, -session_id, -expires_at
schedule	Сабақ кестесі туралы мәліметтер.	-id, -teacher_id, -group_name, -day_of_week, -time_slot, -course, -classroom.
courses	Оқу курстарының тізімі	-id, -name, -description, -credits, -teeacher_id, -group_name, -semester.
student_grades	OverViewResults тақташасының мәліметтері	-id, -user_id, -course_id, -final_score, -letter_grade, -semester_code
academic_events	Академиялық күнтізбе мәліметтерін сақтайды	-id, -title, -start_date, -end_date, -description, -color, -created_at

assessment_types	Мұғалімдер үшін баға қою кестесін шаблондармен қамтамасыз етеді.	-id, -course_id, -assessment_name, -category, -subcategory, -weight, -max_score
student_assessments	Студенттердің әр пән бойынша бағаларын сақтайды	-id, -student_id, -course_id, -assessment_type_id, -score, -date_recorded

1.2- Кесте. Мәліметтер базасының құрылымы

Деректердің Тұтастығын Қамтамасыз ету Рөлі

БД архитектурасы деректердің дұрыстығы мен сенімділігін келесі жолдармен қорғайды:

- Транзакцияларды Қолдау: Кестені жаңарту сияқты күрделі операциялар үшін Транзакциялар механизміне қолдау көрсетеді. Бұл екі немесе одан да көп әрекетті атомарлы (бүтін) бір операция ретінде орындауға мүмкіндік береді – не бәрі сәтті өтеді (COMMIT), не ештеңе өзгермейді (ROLLBACK).
- Кілттер мен Индекстер: Бастапқы және Сыртқы кілттерді пайдалану арқылы деректердің логикалық байланысының бұзылмауын қамтамасыз етеді. Бұл сондай-ақ іздеу операцияларының жылдам орындалуына мүмкіндік береді.

Қорытынды: Жобаның дерекқор архитектурасы қарапайым, реляциялық принциптерге негізделген және функционалдық қажеттіліктерді толық қанағаттандырады. users, sessions, schedule, courses, student_grades, academic_events, assessment_type және student_assessment кестелерінің айқын бөлінуі мен тиісті байланыстары қосымша қолдау көрсетуді және көбейтуді жеңілдетеді. Дерекқор pool - ын және транзакцияларды қолдану жүйенің сенімділігі мен өнімділігін арттырады.

ЕКІНШІ БӨЛІМ

2.1 Бэкенд инструменттері (Node.js).

Неліктен Node.js ?

- Бірыңғай Тілдік Кеңістік (Full-Stack JavaScript). Ең маңызды фактор — бекенд (Node.js) және фронтенд (браузер) үшін тек бір тілді, яғни JavaScript-ті қолдану мүмкіндігі. Бұл бастаушы әзірлеуші үшін немесе шағын команда үшін оқып-үйренуді, код жазуды және қолдауды айтарлықтай жеңілдетеді. Және де әзірлеушіге екі түрлі тілдің (мысалы, Python/PHP және JavaScript) синтаксисін, құрылымын және әртүрлі кітапханаларын меңгеру қажет емес.
- Жаңадан бастаушылар үшін. JavaScript бастапқы бағдарламалау тілі ретінде кең таралған. Node.js-тің өзі конфигурациялауы жеңіл және үлкен, күрделі фреймворктерді қажет етпейді (біздің жобадағыдай, тіпті "Vanilla" Node.js-ті қолдану). Бұл платформада қарапайым HTTP API-ды тез құруға болады, бұл біздің студенттік портал сияқты шағын және орташа жүйелерге өте қолайлы.
- Жоғары өнімділік (Асинхрондылық). Node.js асинхронды және оқиғаға негізделген (event-driven) архитектурада жұмыс істейді. Бұл дегеніміз, ол бір уақытта көптеген сұраныстарды тиімді өңдей алады. Дерекқорға немесе файлдарға сұрау салу сияқты ұзақ операциялар кезінде, Node.js бұл процесті аяқтауды күтпейді, керісінше, басқа сұраныстарды өңдеуді жалғастырады. Біздің жобадағыдай SQL сұраулары сияқты I/O (енгізу/шығару) операциялары көп болатын қосымшалар үшін бұл өте тиімді.
- Қарапайымдылық. Node.js-тің әлемдегі ең үлкен пакет менеджері NPM (Node Package Manager) бар. Бұл дегеніміз, қажетті функцияны іске асыру үшін дайын шешімдер (мысалы, crypto немесе formidable) оңай және тез қол жетімді. Бұл кітапханалардың молдығы жаңа

функционалды (мысалы, аватар жүктеуді) жылдам енгізуге мүмкіндік береді, жобаны дамыту уақытын қысқартады.

Нодада (Node.js) сұрауларды (запростарды) қабылдау және өңдеу әдетте роутинг (маршруттау) арқылы жүзеге асады. Сіздің жобаңыздағы server.js файлы дәл осыны істейді. Ол клиенттен келген HTTP сұрауын (GET, POST) қабылдайды, оның жолын (URL) анықтайды және тиісті функцияны шақырады.

```
if(req.method === "POST" && pathname === "/register") {
        let body = "";
        req.on("data", chunk => body += chunk);
        req.on("end", async () => {
            const data = JSON.parse(body);
            const result = await register(data.username, data.password, data.role);
            res.end(JSON.stringify(result));
            return;
        });
        return;
    }
```

2.1-Сурет. Сервердің POST сұрауы

Бұл код үзіндісі (2.1 - Сурет) Node.js HTTP серверінің клиенттен келген POST сұрауын қалай өңдейтінін көрсетеді: ол алдымен '/register' жолын сүзеді (if(req.method === "POST" && pathname === "/register")), содан кейін req.on("data", ...) арқылы деректерді асинхронды жинап алады, req.on("end", ...) ішінде жиналған деректі JSON.parse() арқылы талдайды, register(...) бизнес-логикасын шақырып, res.end(...) арқылы тіркеу нәтижесін клиентке JSON форматында қайтарады.

2.2. JavaScript-тің Негізгі Міндеттері.

Фронтенд JavaScript коды келесі маңызды міндеттерді

орындайды:

А. Авторизацияны Басқару және Рөлге Қатысты Рендеринг

- Сессияны Тексеру: Бастапқы жүктеу кезінде (немесе onload ішінде) клиенттік JS бекендтегі /api/check_session сияқты маршрутқа сұрау жібереді.
- Интерфейсті Бейімдеу: Бекенд қайтарған рөлге (student, teacher, admin) негізделген.
    1. Егер рөл "student" болса, тек қарауға арналған элементтерді көрсетеді.
    2. Егер рөл "teacher" болса, "Кестені сақтау" түймесін немесе курстарды редакциялау формаларын белсендіреді.
    3. Егер рөл "admin" болса, "Курстарды Басқару" толық формасын көрсетеді.

Б. Деректерді Асинхронды Жүктеу және Көрсету (Data Fetching & Rendering)

- API-мен Өзара Әрекеттесу: fetch() API немесе XMLHttpRequest сияқты құралдар арқылы бекендке сұраулар жібереді. Мысалы:
    1. Кесте: Пайдаланушы "Сейсенбі" күнін таңдағанда, /api/get_schedule маршрутына топ атауымен сұраныс жіберіледі.
    2. Курстар: Студент үшін /api/get_student_courses сұранысы жіберіледі.
- Динамикалық Рендеринг (DOM Manipulation): Бекендтен алынған JSON деректері (мысалы, кесте сабақтарының массиві) қабылданып, JavaScript арқылы HTML-элементтерге (кесте жолдарына, тізімдерге) айналдырылады және тиісті контейнерлерге қойылады.

В. Формаларды Басқару және Деректерді Жіберу

- Кестені Сақтау Формасы: Оқытушының кестені өзгерту формасынан деректерді жинайды.
- API-ға Жіберу: Деректерді жинақтап, оны JSON.stringify() арқылы форматтап, POST сұрауымен /api/save_schedule маршрутына жібереді.
- Пайдаланушыға Хабарлау: Бекендтен қайтарылған нәтижені ({success:

true} немесе {success: false, error: ...}) алып, экранға хабарлама (уведомление) шығарады.

2.3. HTML және CSS.

HTML (dashboard.html және т.б) мазмұндық қаңқаны және өзара әрекеттесу элементтерін (кнопкалар, формалар, кестелер) анықтайды. Порталдың негізгі сүйегі. Ол тек статикалық құрылымды береді; барлық динамикалық деректер (кестелер, курстар тізімі) JavaScript арқылы осы HTML - элементтерге қойылады.

Визуалды Дизайнды және Орналасуды (Layout) басқарады. Интерфейстің сыртқы көрінісін, түстерін, қаріптерін, өлшемдерін және әсіресе мобильді құрылғыларға бейімделуін (адаптивті дизайн) қамтамасыз етеді. CSS деректерді көрсетуді функционалды және эстетикалық тұрғыдан тиімді етеді.

2.4. MySQL

деректер базасының кең таралған басқару жүйесі (СУБД). Сіздің жобаңыз үшін MySQL-ді таңдаудың бірнеше негізгі артықшылықтары (плюстары) бар:

1. Тұтастыққа Кепілдік: MySQL реляциялық дерекқор болғандықтан, ол деректердің тұтастығын (целостность) қамтамасыз етеді. Бұл дегеніміз, кестелер арасындағы байланыстар (Foreign Keys) мен транзакциялар (мысалы, кестені сақтау кезіндегі saveSchedule функциясындағыдай) деректердің логикалық сәйкессіздікке ұшырамауына кепілдік береді.
2. Оңай Меңгеру: MySQL SQL (Structured Query Language) тілін пайдаланады, бұл тіл оқуға және түсінуге оңай. Бастапқы деңгейдегі әзірлеушілер үшін бұл тілмен жұмыс істеу қиындық тудырмайды. Бағдарламалық жасақтаманы орнату және конфигурациялау процесі қарапайым. MySQL-дің көптеген нұсқалары (Community Edition) тегін қолжетімді, бұл жоба бюджеті шектеулі болғанда өте тиімді.

3. Өнімділік (Performance): MySQL кіші және орташа өлшемді жүктемелер үшін (біздің шағын портал сияқты) өте жоғары жылдамдық пен өнімділік көрсетеді. Пулдармен Тиімділік: Node.js-те қосылым пулымен (pool) бірге қолданылған кезде, MySQL сұрауларды жылдам өңдейді, бұл жүйенің жалпы жауап беру уақытын жақсартады.

Дерекқормен өзара әрекеттесу db.js файлы арқылы жүзеге асырылады. Бұл тәсіл әдеттегідей сенімді және тиімді болып табылады:

Технология.

Node.js үшін арналған mysql2/promise драйвері. promise (уәде) функционалын қолдану асинхронды операцияларды (SQL сұрауларын) оңай және таза өңдеуге мүмкіндік береді.

Қосылу Механизмі: Пул (Pool). Әрбір API сұрауы үшін дерекқорға қайта-қайта қосылудың орнына, пул алдын ала белгіленген қосылымдар жиынтығын басқарады. Бұл сервердің өнімділігін және көптеген пайдаланушы бір мезгілде кірген кездегі тұрақтылығын айтарлықтай арттырады. Конфигурация: Қосылу жергілікті серверге (localhost немесе 127.0.1.16) стандартты MySQL порты (3306) арқылы "db_01" базасына жасалған.(2.2-Сурет)

```
const mysql = require("mysql2/promise");

const pool = mysql.createPool({
    host: "127.0.1.16",
    user: "root",
    password: "",
    database: "db_01",
    port: 3306
});

module.exports = pool;
```

2.2-Сурет. Сервер мен мәліметтер базасының байланысы

2.5 Қосымша қолданылған кітапханалар.

1. CRYPTO Модулі (Node.js Ішкі Модулі)

crypto – бұл Node.js-тің криптографиялық функцияларға қол жеткізуді қамтамасыз ететін негізгі модуль. Ол хештеу, шифрлау және кездейсоқ байттарды генерациялау үшін қолданылады.

- Парольді Хештеу (SHA256): hashPassword функциясы (auth.js). crypto.createHash("sha256").update(password).digest("hex") арқылы пайдаланушы парольдерін бір бағытты хештейді.(2.3 – Сурет)

```
function hashPassword(password) {
    return crypto.createHash("sha256").update(password).digest("hex");
}
```

2.3 – Сурет. Құпия сөзді хештеу

2. MYSQL2/PROMISE Модулі (NPM Пакеті)

Бұл – Node.js қосымшаларын MySQL дерекқорымен байланыстыруға арналған тиімді драйвер. Оның "promise" бөлігі асинхронды операцияларды (SQL сұрауларын) оңай басқаруға мүмкіндік береді.

Асинхронды Сұраулар: await pool.query(...) немесе await pool.execute(...) арқылы SQL сұрауларын орындайды. Дерекқор операциялары кезінде сервердің блокталмай жұмыс істеуін және кодты async/await арқылы оқуға жеңіл ету.

3. FORMIDABLE Модулі (NPM Пакеті)

formidable – бұл Node.js-тегі файлдарды жүктеуді (upload) және формаларды өңдеуді, әсіресе multipart/form-data типті сұрауларды жеңілдететін күшті модуль.

Аватар Жүктеуді Өңдеу:

- const form = new formidable.IncomingForm(); form.parse(req, ...) арқылы клиент жіберген файлды қабылдайды. Пайдаланушының аватар файлын серверге тиімді түрде жүктеп, сақтау. Модуль файлдың өлшемі мен жолын автоматты анықтайды.(2.4 – Сурет)

```
const form = new Formidable({
    uploadDir: UPLOAD_DIR,
    keepExtensions: true,
    maxFileSize: 5 * 1024 * 1024, // 5 МБ
});

form.parse(req, async (err, fields, files) => {
    if (err) {
        console.error("Formidable error:", err);
        res.writeHead(400);
        res.end(JSON.stringify({ success: false, error: "Ошибка при разборе файла." }));
        return;
    }
```

2.4 - Сурет.new formidable.IncomingForm() кітапханасы

4. HTTP Cookies (Стандартты Веб Механизмі)
Cookies – бұл сервер клиенттің браузерінде сақтау үшін жіберетін кішкентай деректер бөліктері.

ҮШІНШІ БӨЛІМ

Автоматтандырылған жүйенің негізгі функционалдық модульдері:

3.1 Сабақ кестесін басқару модулі:

Бұл модуль Студенттер мен Оқытушылардың ақпарат алу қажеттіліктерін, сондай-ақ Әкімшілік пен Оқытушылардың кесте құру/өңдеу міндеттерін шешеді.

Рөл	Негізгі әрекет	Техникалық іске асыру
Студент	Өзінің группасына арналған кестені көреді	**dashboard.html** арқылы **student_schedule.html** жүктеледі. **fetch** сұранысы **user.group_name** негізінде **/api/get_schedule** маршрутына жібереді.(3.1 - 3.2 Суреттер)
Оқытушы \ Админ	Топты және күнді таңдау арқылы кез келген кестені көреді, өңдейді және сақтайды	**teacher_schedule.html** интерфейсі қолданылады. Деректер **loadTeacherSchedule()** функциясы арқылы жүктеледі, ал сақтау **saveSchedule()** функциясы арқылы **/api/save_schedule** маршруты арқылы жүзеге асады. (3.3 - 3.4 Сурет)

3.1- Кесте.Сабақ кестесін басқару модулі

```
else if(req.method === "GET" && pathname === "/api/get_schedule") {
    const cookies = cookie.parse(req.headers.cookie || '');
    const sessionId = cookies.session;
    const user = await checkSession(sessionId);


    if (!user) {
        res.writeHead(401);
        res.end(JSON.stringify({success: false, error: "Неавторизованный доступ."}));
        return;
    }

    //Определение группы
    let groupName;
    let dayOfWeek = parsedUrl.query.day;

    if(user.role === 'student') {
        groupName = user.group_name;

    } else if(user.role === 'teacher' || user.role === 'admin') {
        // Учитель может посматривать любую группу, которую он выберет.
        // Для простоты, здесь мы будем запрашивать группу из URL-параметров.
        // /api/get_schedule?group=Логистика
        groupName = parsedUrl.query.group;
    } else {
        res.writeHead(403);
        res.end(JSON.stringify({success: false, error: "Нет прав для просмотра расписания"}));
        return;
    }

    // Получение данных
    const scheduleData = await getSchedule(groupName, dayOfWeek);

    if(scheduleData) {
        res.writeHead(200, {"Content-Type": "application/json"});
        res.end(JSON.stringify({success: true, schedule: scheduleData, group: groupName}));
        return;
    }else{
        res.writeHead(500);
        res.end(JSON.stringify({success: false, error: "Ошибка при получений данных расписания."}));
        return;
    }
}
```

3.1 Сурет. Серверлік сұрау (Запрос)

```
async function getSchedule(groupName, dayOfWeek = null) {
    if (!groupName) return [];
        try {
            let query = `
            SELECT
                s.*,
                u.username AS teacher_name
            FROM
                schedule s
            JOIN
                users u ON s.teacher_id = u.id
            WHERE
                s.group_name = ?
        `;
         let params = [groupName];

        // ДОБАВЛЯЕМ ФИЛЬТР ПО ДНЮ, ЕСЛИ ОН ПЕРЕДАН
        if (dayOfWeek) {
            query += ` AND s.day_of_week = ?`;
            params.push(dayOfWeek);
        }

        query += `
             ORDER BY
                FIELD(s.day_of_week, 'Понедельник', 'Вторник', 'Среда', 'Четверг', 'Пятница'),
                s.time_slot
            `;

        const [rows] = await pool.query(query, params); // <-- ИСПОЛЬЗУЕМ НОВЫЙ МАССИВ ПАРАМЕТРОВ

     return rows;
    } catch (error) {
        console.error("Error fetching schedule:", error);
        return null;
    }
}
```

3.2 Сурет. Деректерді алу\өңдеу коды

```
async function loadTeacherSchedule() {

    const group = document.getElementById('groupSelector').value;
    const day = document.getElementById('daySelector').value;
    const msgBox = document.getElementById('scheduleMessage');


    const tableBody = document.getElementById('scheduleInputTable').querySelector('tbody');
    tableBody.querySelectorAll('input').forEach(input => input.value = '');

    if (!group || !day) {
        msgBox.textContent = 'Выберите группу и день недели для загрузки.';
        msgBox.style.color = 'blue';
        return;
    }

    msgBox.textContent = `⏳ Загрузка расписания для ${group} (${day})...`;
    msgBox.style.color = 'orange';

    try {
        // Отправляем GET-запрос с выбранными параметрами
        // Мы используем /api/get_schedule?group=X, как вы настроили в server.js
        const res = await fetch(`/api/get_schedule?group=${group}&day=${day}`);
        const data = await res.json();

        if (data.success && data.schedule && data.schedule.length > 0) {

            // Заполняем форму данными (schedule - это массив уроков)
            data.schedule.forEach(lesson => {
                const row = tableBody.querySelector(`tr[data-time="${lesson.time_slot}"]`);
                if (row) {
                    // Заполняем предмет и аудиторию
                    row.querySelector('input[data-field="course"]').value = lesson.course;
                    row.querySelector('input[data-field="classroom"]').value = lesson.classroom;
                    // Преподавателя заполнять не нужно, т.к. его ID берется из сессии при сохранении
                }
            });

            msgBox.textContent = `✅ Расписание для ${group} (${day}) загружено. Можно редактировать.`;
            msgBox.style.color = 'green';

        } else {
            msgBox.textContent = `⚠️ Расписание для ${group} (${day}) не найдено. Можно составить новое.`;
            msgBox.style.color = 'orange';
        }
```

3.3 Сурет. Кесте бетін жүктеу

3.4 Сурет. Деректерді алу\өңдеу коды

3.2 Академиялық календарь модулі

Бұл модуль оқу жылының маңызды оқиғаларын (сессия, мерекелер, аралық бақылау кезеңдері) барлық қолданушыларға көрсетеді, бірақ басқару мүмкіндігін тек Әкімшіге береді.

- Көрсету: Календарь **FullCalendar** JavaScript кітапханасы арқылы іске асырылған. Календарь деректері **auth.js** файлындағы **getAcademicEvents()** функциясы арқылы дерекқордан алынып, **/api/get_academic_events** маршруты арқылы жіберіледі.(3.5 - 3.6 Сурет)
- Басқару (Тек Әкімші):
    1. Әкімші жаңа оқиғаларды қосу, барларын өзгерту және жою мүмкіндігіне ие
    2. Жою процесі **deleteCalendarEvent()** функциясы арқылы жүреді, ол **/api/admin/calendar/delete** (сервердегі **deleteAcademicEvent** функциясына сәйкес) маршрутына **POST** сұранысын жібереді. Бұл, жобаның **рөлдік қауіпсіздік** талаптарын сақтай отырып, маңызды деректерді басқару мүмкіндігін қамтамасыз етеді.(3.7- 3.9 Сурет)

```
async function getAcademicEvents() {
    try {
        const query = `
            SELECT
                id,
                title,
                start_date AS start,
                end_date AS end,
                color,
                description
            FROM
                academic_events
            ORDER BY
                start_date;
        `;

        // ВАЖНО: Мы переименовываем поля в 'start' и 'end',
        // чтобы они сразу соответствовали формату, который ждет FullCalendar!

        const [events] = await pool.query(query);

        // Если end_date установлен в базе как NULL, FullCalendar может быть недоволен.
        // Очистим его здесь, если он не установлен.
        const formattedEvents = events.map(event => ({
            id: event.id,
            title: event.title,
            start: event.start,
            // Если end_date существует, мы его возвращаем, иначе null
            end: event.end ? event.end : null,
            color: event.color,
            description: event.description
        }));

        return { success: true, data: formattedEvents };

    } catch (error) {
        console.error("DB Error in getAcademicEvents:", error);
        return { success: false, error: "Ошибка БД при загрузке событий календаря." };
    }
}
```

3.5 Сурет. Деректерді алу\өңдеу

```
else if (req.method === 'GET' && pathname === '/api/calendar/events') {
    const session = req.headers.cookie ? cookie.parse(req.headers.cookie).session : null;
    const user = await checkSession(session);

    // 1. Проверка авторизации: доступен всем авторизованным
    if (!user) {
        res.writeHead(401, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ success: false, error: 'Требуется авторизация.' }));
        return;
    }

    try {
        // 2. Вызываем функцию из auth.js
        const result = await getAcademicEvents();

        res.writeHead(200, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify(result));

    } catch (error) {
        console.error("Server error loading academic events:", error);
        res.writeHead(500, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ success: false, error: 'Ошибка сервера при загрузке календаря.' }));
    }
    return;
}
```

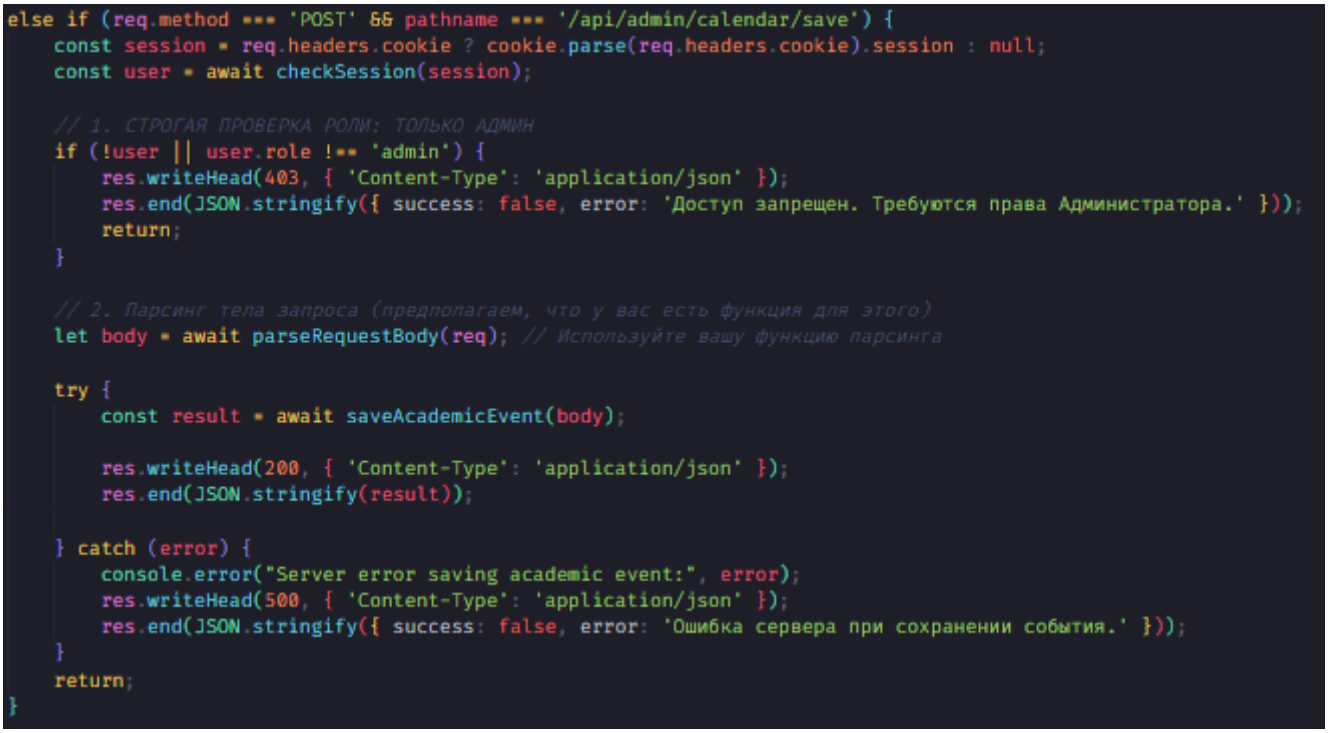
3.6 Сурет. Сервер сұрауы

```
async function saveAcademicEvent(eventData) {
    const { id, title, start, end, color, description } = eventData;

    try {
        if (id) {
            // Режим обновления существующего события
            const query = `
                UPDATE academic_events
                SET title = ?, start_date = ?, end_date = ?, color = ?, description = ?
                WHERE id = ?;
            `;
            await pool.query(query, [title, start, end || null, color, description, id]);
            return { success: true, message: "Событие успешно обновлено." };

        } else {
            // Режим создания нового события
            const query = `
                INSERT INTO academic_events (title, start_date, end_date, color, description)
                VALUES (?, ?, ?, ?, ?);
            `;
            const [result] = await pool.query(query, [title, start, end || null, color, description]);

            // Возвращаем ID нового события, что важно для клиента
            return { success: true, message: "Событие успешно создано.", newId: result.insertId };
        }
    } catch (error) {
        console.error("DB Error in saveAcademicEvent:", error);
        return { success: false, error: "Ошибка БД при сохранении события." };
    }
}
```

3.7 Сурет. Деректерді сақтау коды.

```
async function deleteAcademicEvent(id) {
    try {
        await pool.query("DELETE FROM academic_events WHERE id = ?", [id]);
        return { success: true, message: "Событие удалено." };
    } catch (error) {
        console.error("DB Error in deleteAcademicEvent:", error);
        return { success: false, error: "Ошибка БД при удалении события." };
    }
}
```

3.8 Сурет. Деректерді жою коды



```
else if (req.method === 'POST' && pathname === '/api/admin/calendar/save') {
    const session = req.headers.cookie ? cookie.parse(req.headers.cookie).session : null;
    const user = await checkSession(session);

    // 1. СТРОГАЯ ПРОВЕРКА РОЛИ: ТОЛЬКО АДМИН
    if (!user || user.role !== 'admin') {
        res.writeHead(403, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ success: false, error: 'Доступ запрещен. Требуются права Администратора.' }));
        return;
    }

    // 2. Парсинг тела запроса (предполагаем, что у вас есть функция для этого)
    let body = await parseRequestBody(req); // Используйте вашу функцию парсинга

    try {
        const result = await saveAcademicEvent(body);

        res.writeHead(200, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify(result));

    } catch (error) {
        console.error("Server error saving academic event:", error);
        res.writeHead(500, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ success: false, error: 'Ошибка сервера при сохранении события.' }));
    }
    return;
}
```

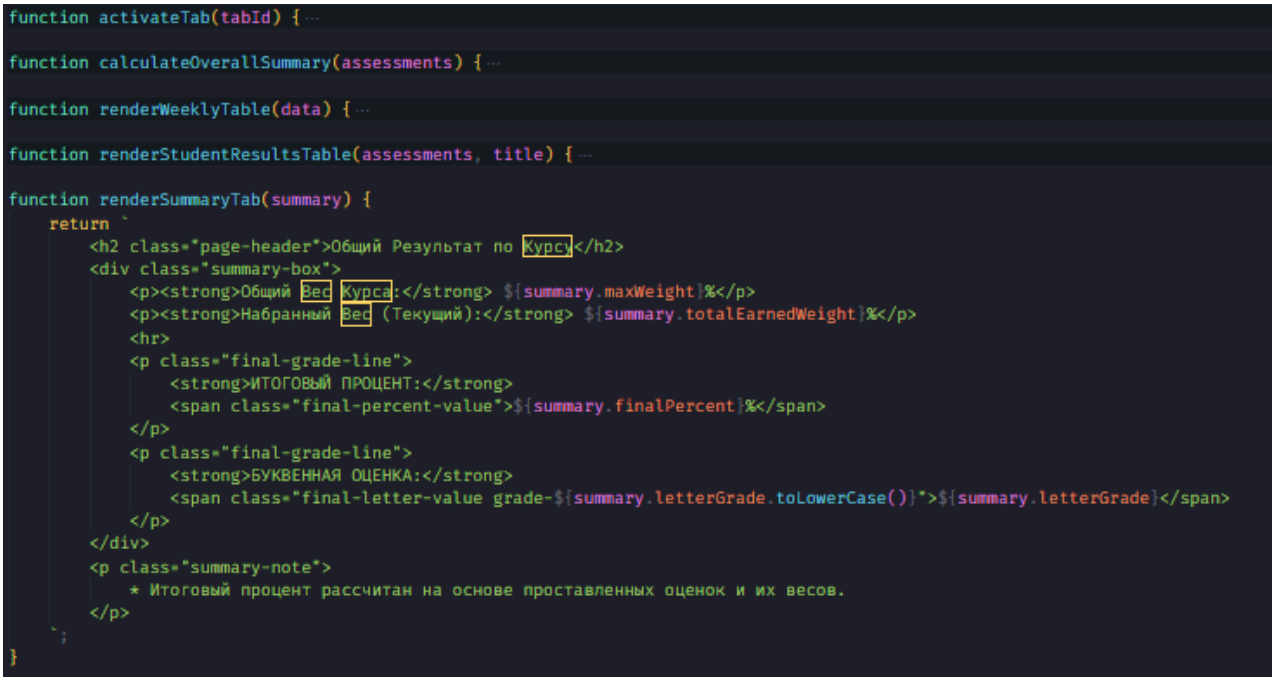
3.9 Сурет. Сервер сұрауы, студент үшін

3.3 Бағалау Және Нәтижелер Модулі

Бұл модуль оқу процесінің нәтижелерін бақылаудың орталық құралы болып табылады

3.3.1 Студенттік Нәтижелерді Көрсету

- Шолу (Summary): Студенттер **renderSummaryTab()** функциясы арқылы курстың жалпы нәтижесін, жинаған баллдарды **(totalEarnedWeight)** және қорытынды буквалық бағаны **(letterGrade)** көре алады.
- Толық Детальдар: **renderDetailsTab()** функциясы жеке бағалау элементтерінің (салмағы, максималды балл, алынған балл) кестесін көрсетеді.
- Іске Асыру Негізі: **getStudentAssessmentDetails** және **getStudentResultsOverview** функциялары дерекқордан алынған деректерді клиенттік жақта өңдеп, студентке арналған пайыздық және әріптік бағаны есептейді.(3.10 - 3.12 Сурет)



```
function activateTab(tabId) {···

function calculateOverallSummary(assessments) {···

function renderWeeklyTable(data) {···

function renderStudentResultsTable(assessments, title) {···

function renderSummaryTab(summary) {
    return `
        <h2 class="page-header">Общий Результат по Курсу</h2>
        <div class="summary-box">
            <p><strong>Общий Вес Курса:</strong> ${summary.maxWeight}%</p>
            <p><strong>Набранный Вес (Текущий):</strong> ${summary.totalEarnedWeight}%</p>
            <hr>
            <p class="final-grade-line">
                <strong>ИТОГОВЫЙ ПРОЦЕНТ:</strong>
                <span class="final-percent-value">${summary.finalPercent}%</span>
            </p>
            <p class="final-grade-line">
                <strong>БУКВЕННАЯ ОЦЕНКА:</strong>
                <span class="final-letter-value grade-${summary.letterGrade.toLowerCase()}">${summary.letterGrade}</span>
            </p>
        </div>
        <p class="summary-note">
            * Итоговый процент рассчитан на основе проставленных оценок и их весов.
        </p>
    `;
}
```

3.10 Сурет. Нәтижелермен жұмыс жасау функциялары

```
else if (req.method === 'GET' && pathname.startsWith('/api/student/results/details/')) {

    const cookies = cookie.parse(req.headers.cookie || '');
    const session = cookies.session;
    const user = await checkSession(session);

    console.log(`[DEBUG] Session ID: ${session}`);
    console.log('[DEBUG] User object from checkSession:', user);

    if (!user || user.role !== 'student') {
        res.writeHead(401, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ error: 'Доступ запрещен.' }));
        return;
    }

    const parts = pathname.split('/');
    const courseId = parseInt(parts[5]);

    if (isNaN(courseId)) {
        res.writeHead(400, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ error: 'Некорректный ID курса.' }));
        return;
    }

    try {
        // user.id берется из активной сессии
        const result = await getStudentAssessmentDetails(user.id, courseId);

        res.writeHead(200, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify(result));
    } catch (error) {
        console.error("Server error fetching student assessment details:", error);
        res.writeHead(500, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ success: false, error: "Ошибка сервера при получении деталей оценок." }));
    }
    return;
}
```

3.11 Сурет. Сервер сұрауы, студент бағалары

3.12 Сурет. Деректерді алу\өңдеу коды

3.3.2. Оқытушының Баға Қою Функционалы

- Деректерді Жүктеу: Оқытушы курсты және топты таңдағаннан кейін, **/api/get_assessments** маршруты арқылы **getAssessmentTableData** функциясы арқылы бағалау кестесінің құрылымы мен студенттер тізімі жүктеледі.
- Деректерді Сақтау: Оқытушы енгізген бағалар **/api/save_assessments** маршруты арқылы **saveAssessments** функциясына жіберіледі. Бұл функция деректерді мұқият тексеріп, оларды дерекқорға жазады(3.13 - 3.15 Сурет)

```
async function getAssessmentTableData(courseId, groupName) {
    try {
        // 1. Получаем список студентов в нужной группе
        const studentsQuery = `
            SELECT
                id AS student_id,
                full_name,
                username
            FROM
                users
            WHERE
                role = 'student' AND group_name = ?
            ORDER BY
                full_name;
        `;
        const [students] = await pool.query(studentsQuery, [groupName]);

        // 2. Получаем все типы оценок для этого курса
        const assessmentsQuery = `
            SELECT
                id,
                assessment_name,
                category,
                subcategory
            FROM
                assessment_types
            WHERE
                course_id = ?
            ORDER BY
                category, subcategory, id;
        `;
        const [assessments] = await pool.query(assessmentsQuery, [courseId]);

        // 3. Получаем все существующие оценки для этих студентов
        const gradesQuery = `
            SELECT
                student_id,
                assessment_type_id,
                score
            FROM
                student_assessments
            WHERE
                course_id = ? AND student_id IN (?)
            ;
        `;
        const studentIds = students.map(s => s.student_id);
        const [gradesRows] = await pool.query(gradesQuery, [courseId, studentIds]);
```

3.13 Сурет. Деректерді алу\өңдеу

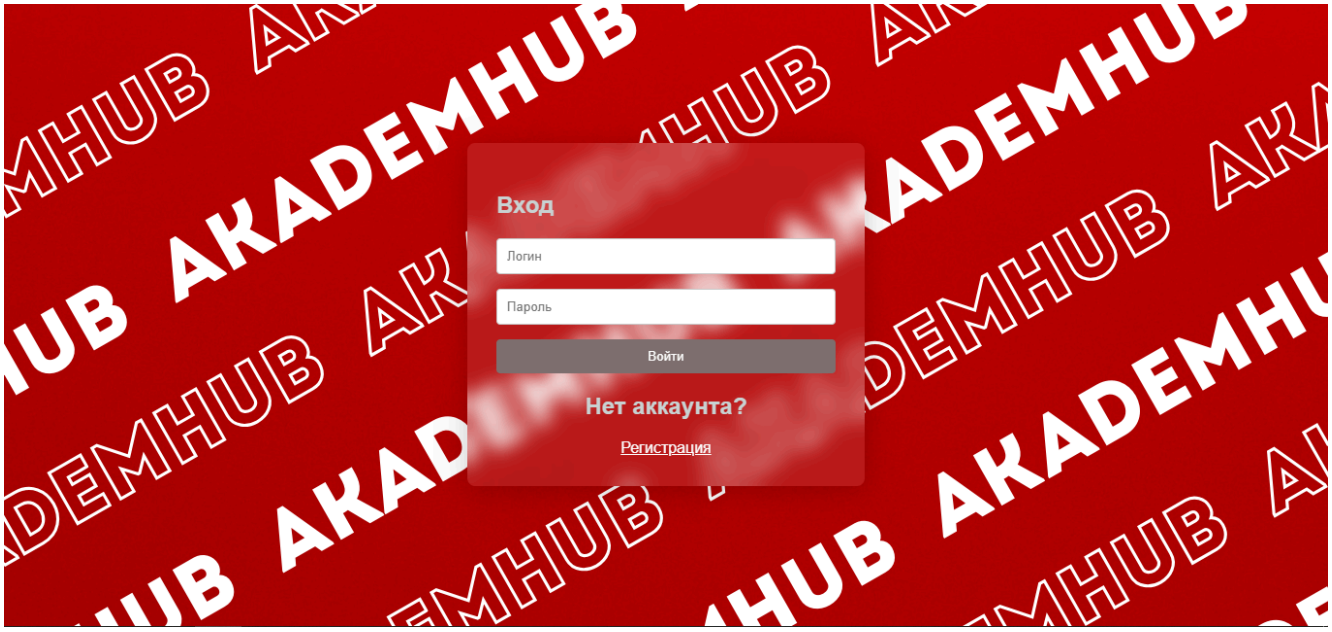
```
function renderTeacherGradingTable(assessments, data, tableId = '') {...


/**
 * Собирает данные из всех полей ввода и отправляет на сервер.
 */
async function saveGrades(e) {
    e.preventDefault();
    const saveBtn = document.getElementById('saveGradesBtn');
    const saveMessage = document.getElementById('saveMessage');

    saveBtn.disabled = true;
    saveMessage.textContent = 'Сохранение...';
    saveMessage.style.color = '#3498db';

    const courseId = document.getElementById('courseSelect').value;
    const gradesToSave = [];

    // Собираем данные из всех полей ввода на форме
    document.querySelectorAll('.grade-input').forEach(input => {
        const score = parseFloat(input.value);

        // Сохраняем только те оценки, которые введены и находятся в допустимом диапазоне
        if (input.value !== '' && !isNaN(score) && score >= 0 && score <= 100) {
            gradesToSave.push({
                student_id: parseInt(input.dataset.studentId),
                assessment_type_id: parseInt(input.dataset.assessmentId),
                course_id: parseInt(courseId),
                score: score
            });
        }
    });

    try {
        const response = await fetch('/api/teacher/save_assessments', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(gradesToSave)
        });
```

3.14 Сурет. Деректерді сақтау функциясы

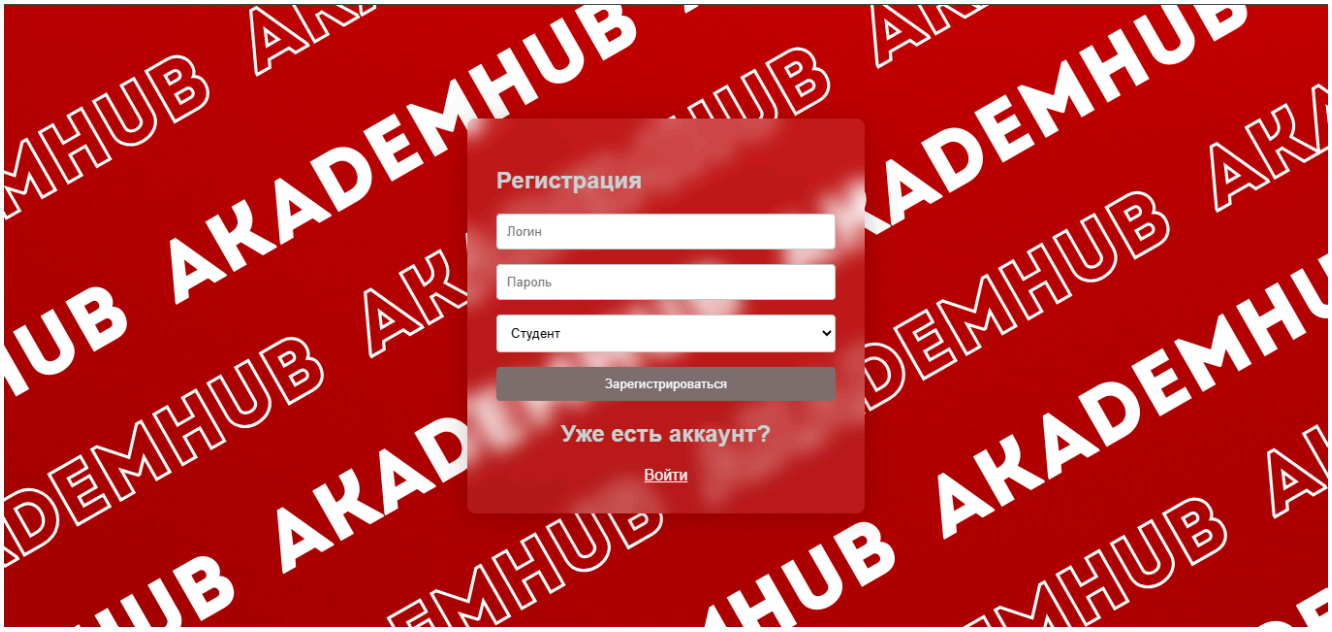
```
else if (req.method === 'POST' && pathname === '/api/teacher/save_assessments') {
    const session = req.headers.cookie ? cookie.parse(req.headers.cookie).session : null;
    const user = await checkSession(session);

    if (!user || (user.role !== 'teacher' && user.role !== 'admin')) {
        res.writeHead(403, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ error: 'Доступ запрещен.' }));
        return;
    }

    let body = '';
    req.on('data', chunk => {
        body += chunk.toString();
    });

    req.on('end', async () => {
        try {
            const assessments = JSON.parse(body);

            if (!Array.isArray(assessments)) {
                res.writeHead(400, { 'Content-Type': 'application/json' });
                res.end(JSON.stringify({ success: false, error: 'Некорректный формат данных.' }));
                return;
            }

            const result = await saveAssessments(assessments);

            res.writeHead(200, { 'Content-Type': 'application/json' });
            res.end(JSON.stringify(result));

        } catch (error) {
            console.error("Server error processing save assessments:", error);
            res.writeHead(500, { 'Content-Type': 'application/json' });
            res.end(JSON.stringify({ success: false, error: "Ошибка сервера при обработке оценок." }));
        }
    });
    return;
}
```

3.15 Сурет. Деректерді сақтау бойынша сұраныс

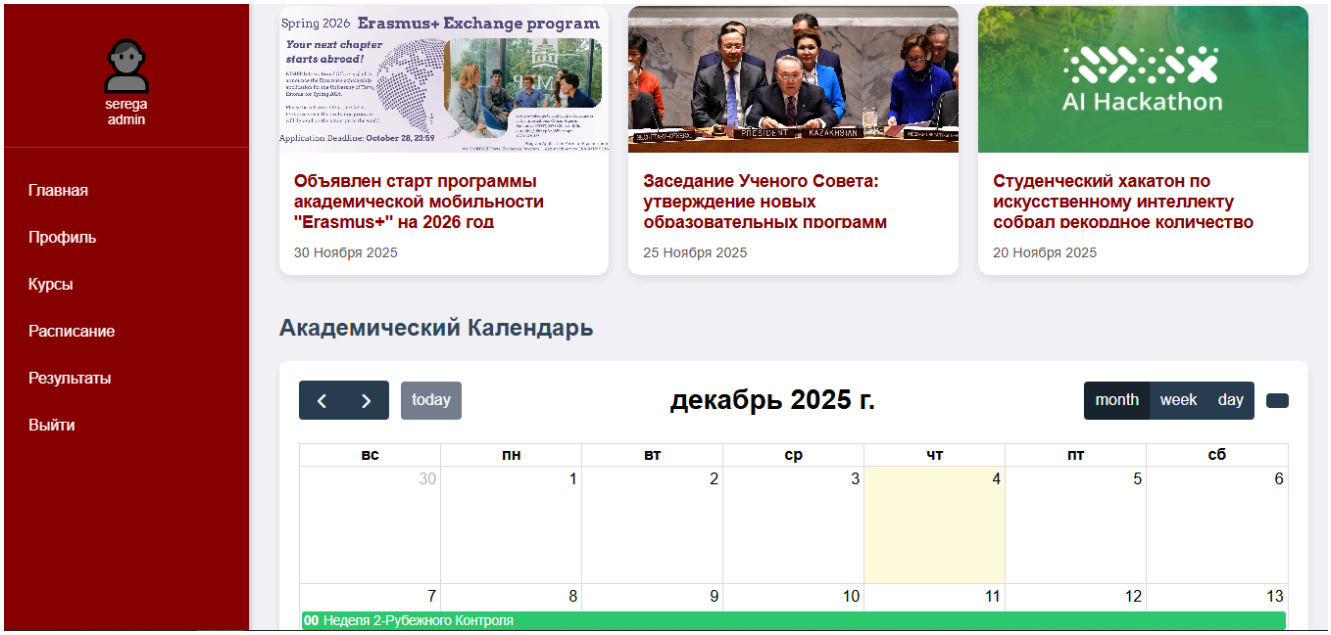
Төменде негізгі сайттың толық суреттері көрсетілген. (3.16 - 3.28 Суреттер).



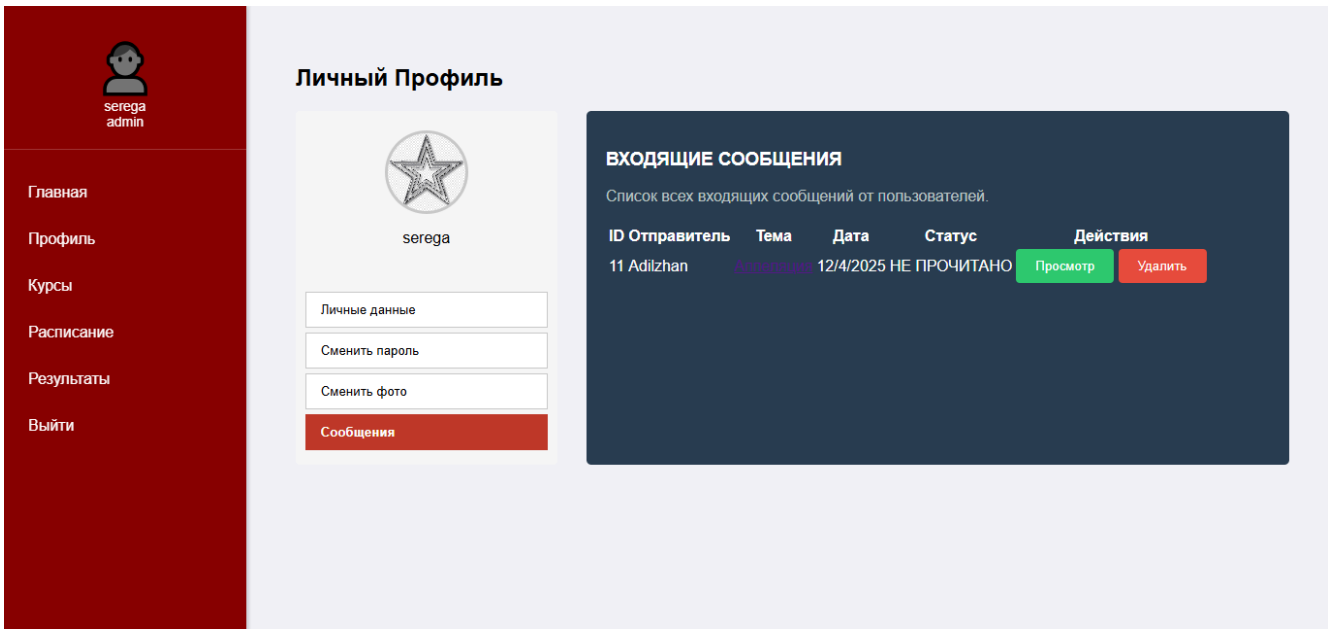
3.16 Кіру бөлімі



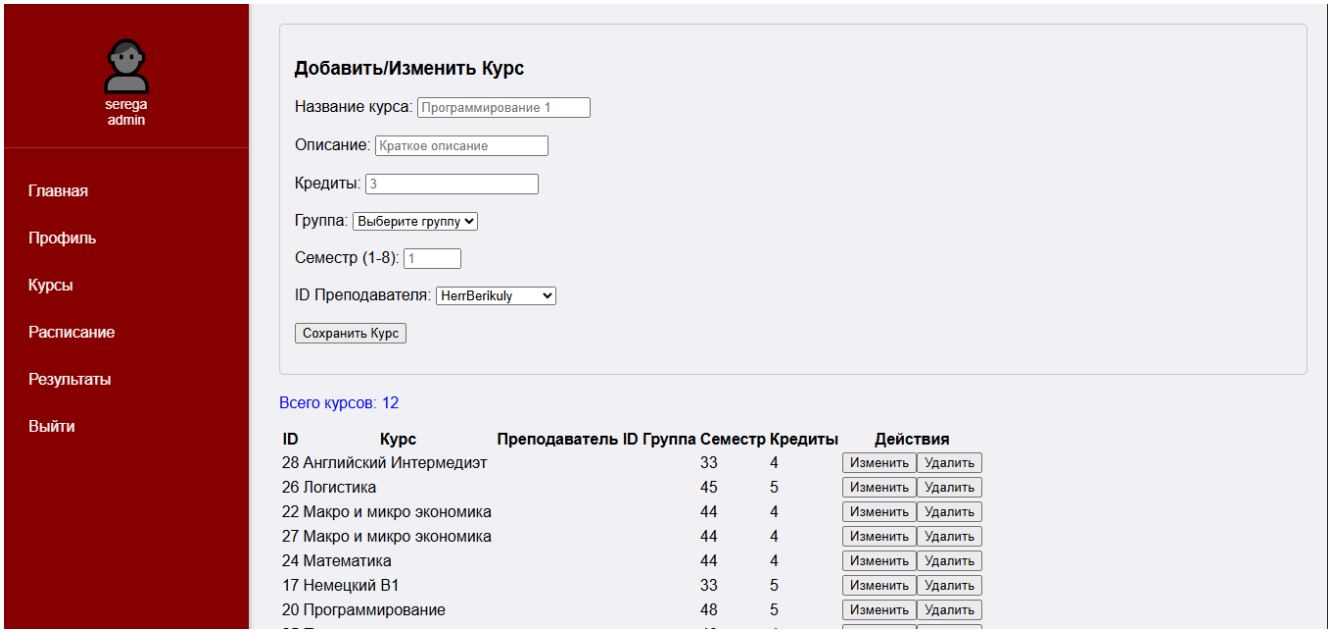
3.17 Тіркелу бөлімі



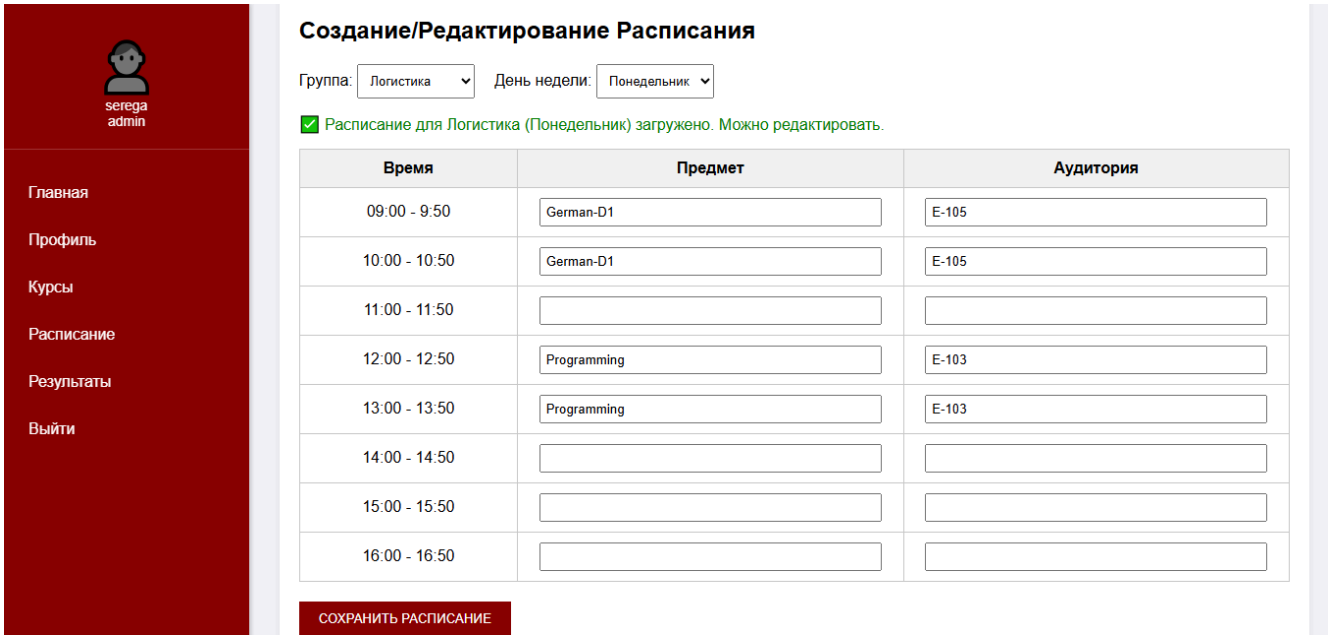
3.18 Басты бет



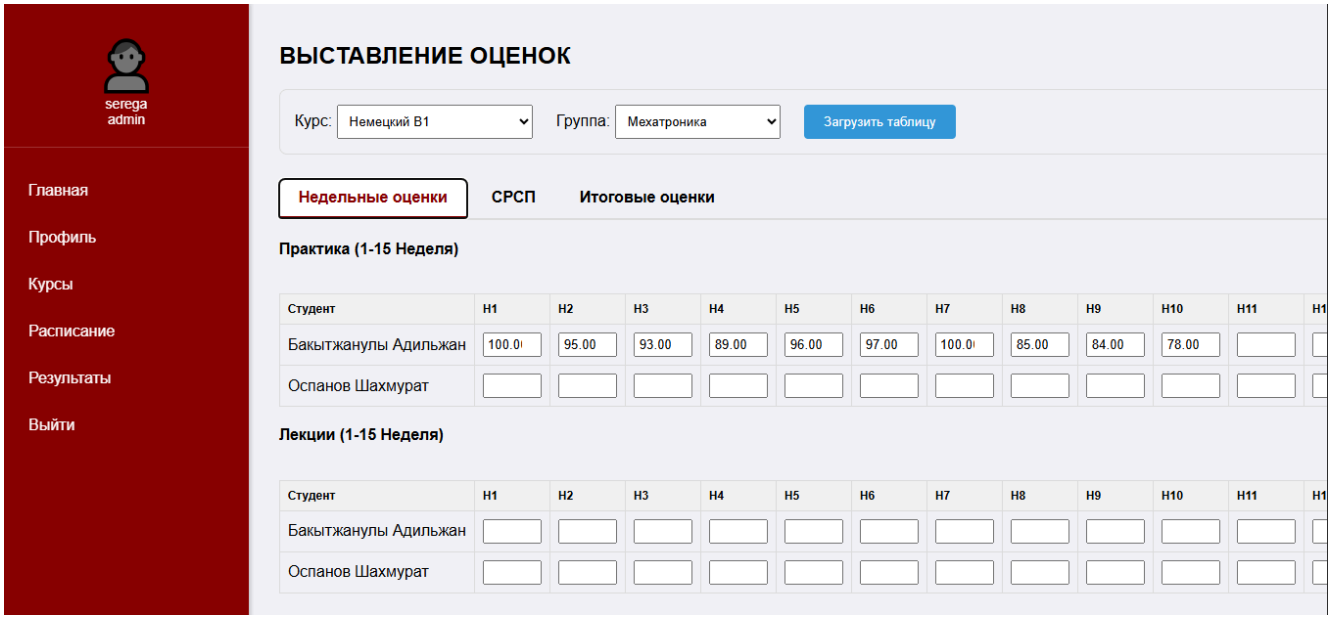
3.19 Жек кабинет және Админ үшін хабарламалар



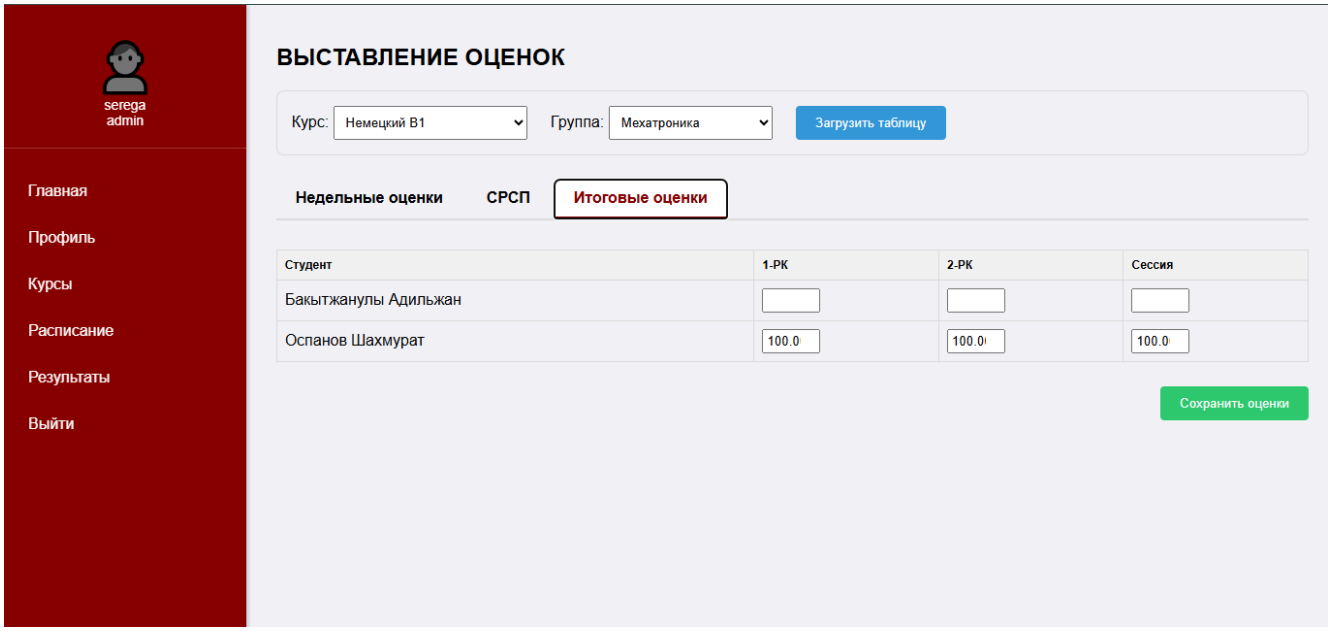
3.20 Курс қосу бөлімі (Админ)



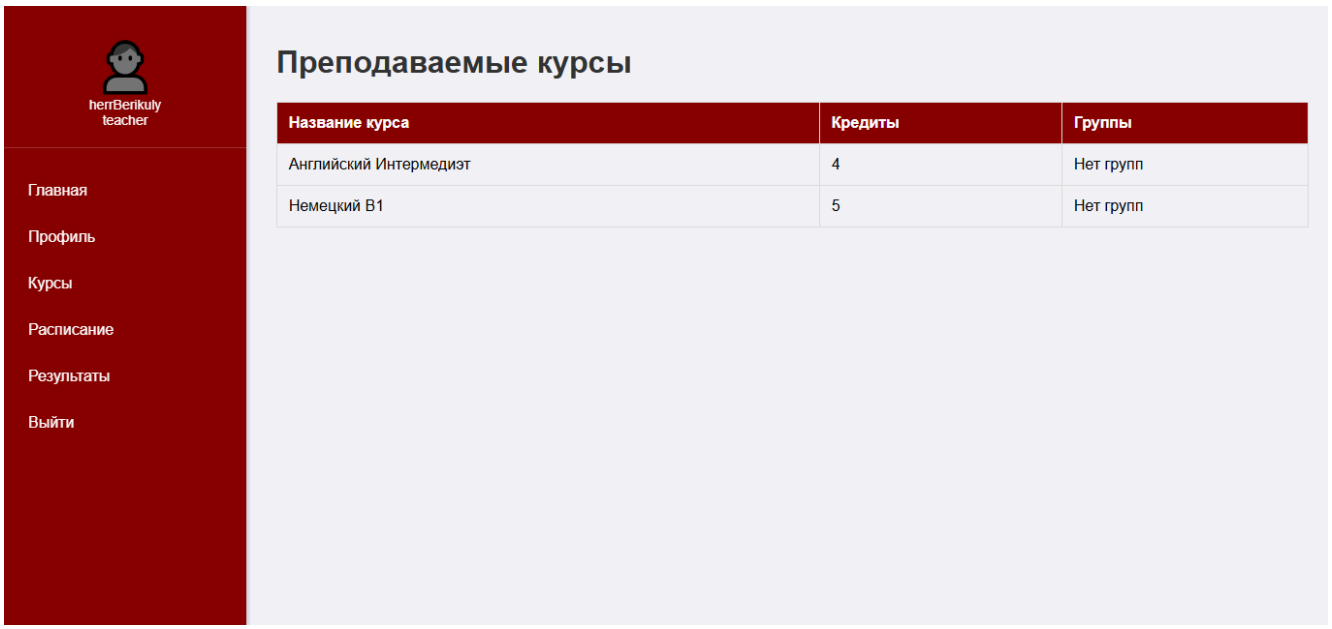
3.21 Сабақ кестесін құру бөлімі(Админ, Оқытушы)



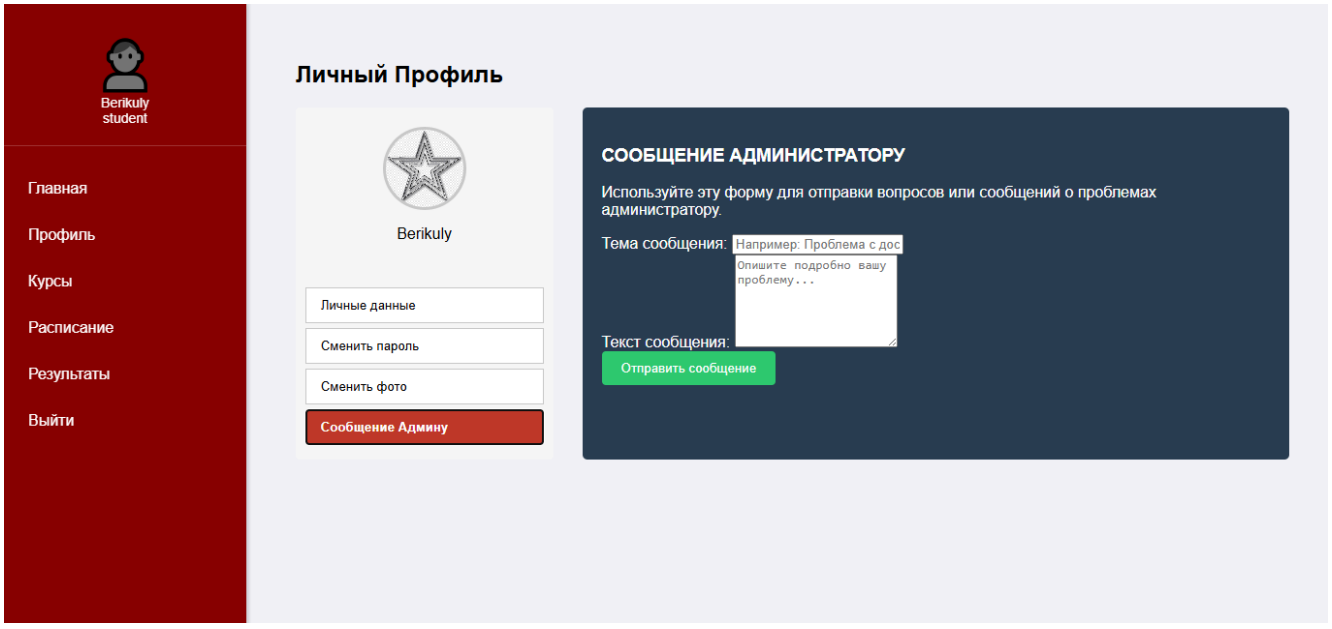
3.22.1 Баға қою бөлімі (Админ, Оқытушы)



3.22.2 Баға қою бөлімі



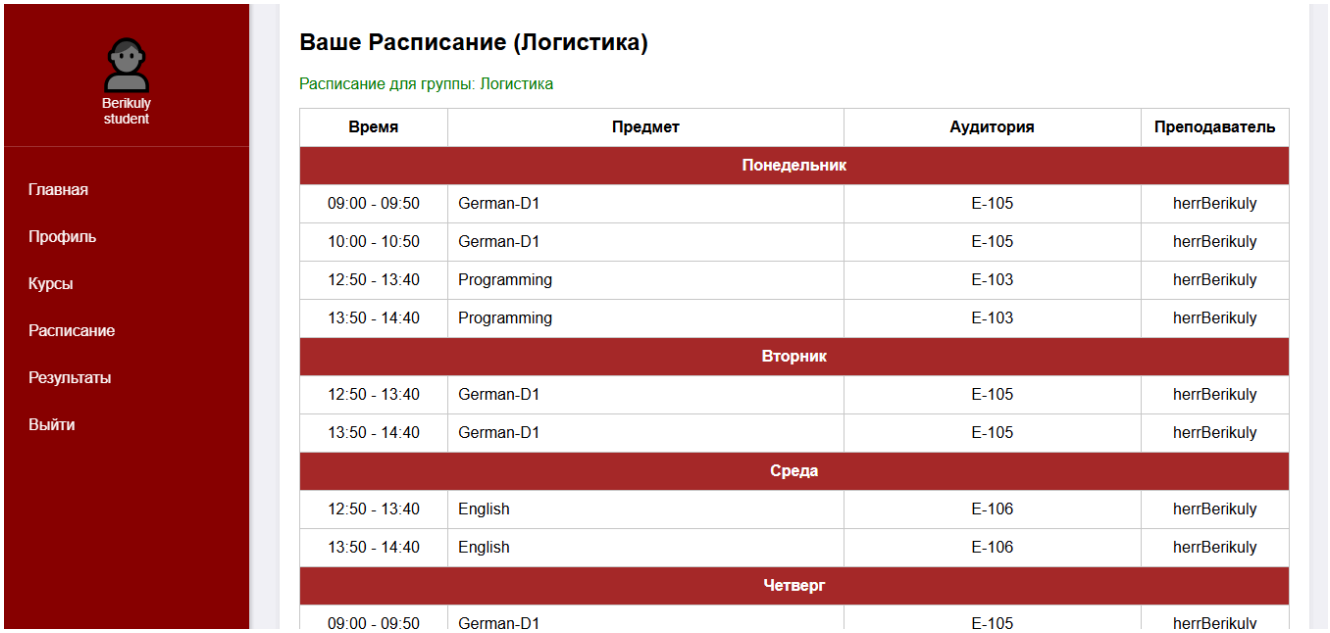
3.23 Курстар бөлімі (Оқытушы)



3.24 Студент үшін Админге хат жазу бөлімі

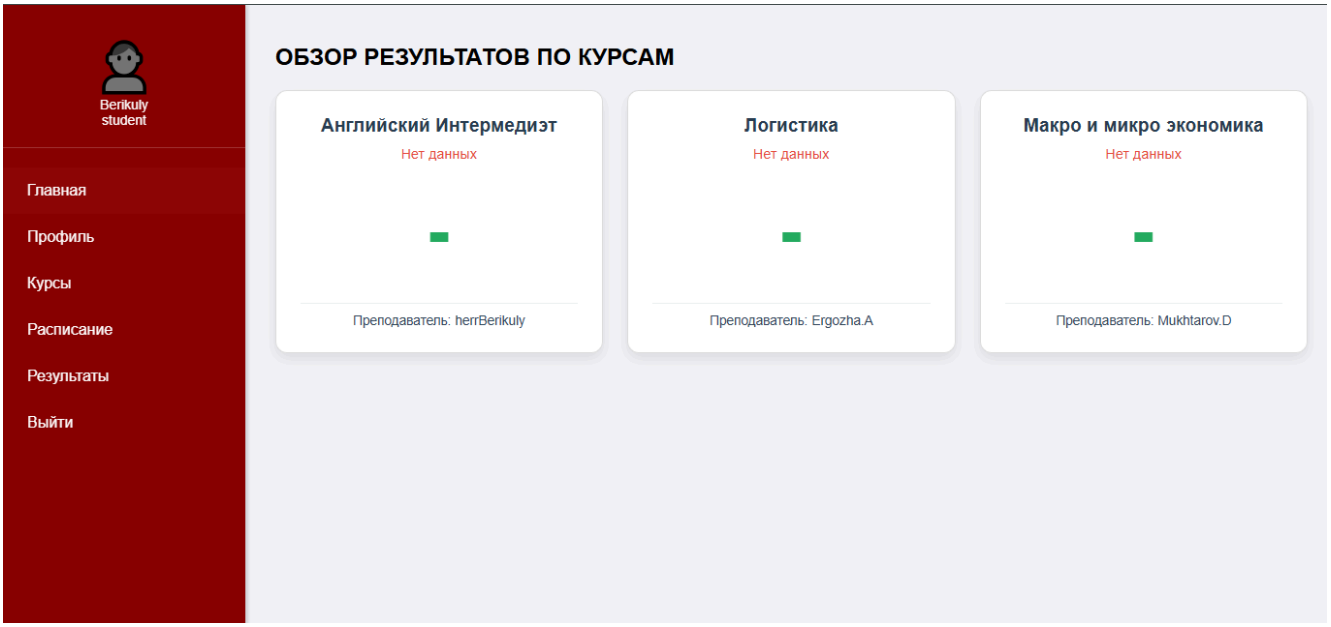
Курс	Преподаватель	Кредиты	Описание
Логистика	Ergozha.A	5	Основы логистика
Макро и микро экономика	Mukhtarov.D	4	Экономика
Английский Интермедиэт	herrBerikuly	4	Енглиш

3.25 Курстар бөлімі (Студент)

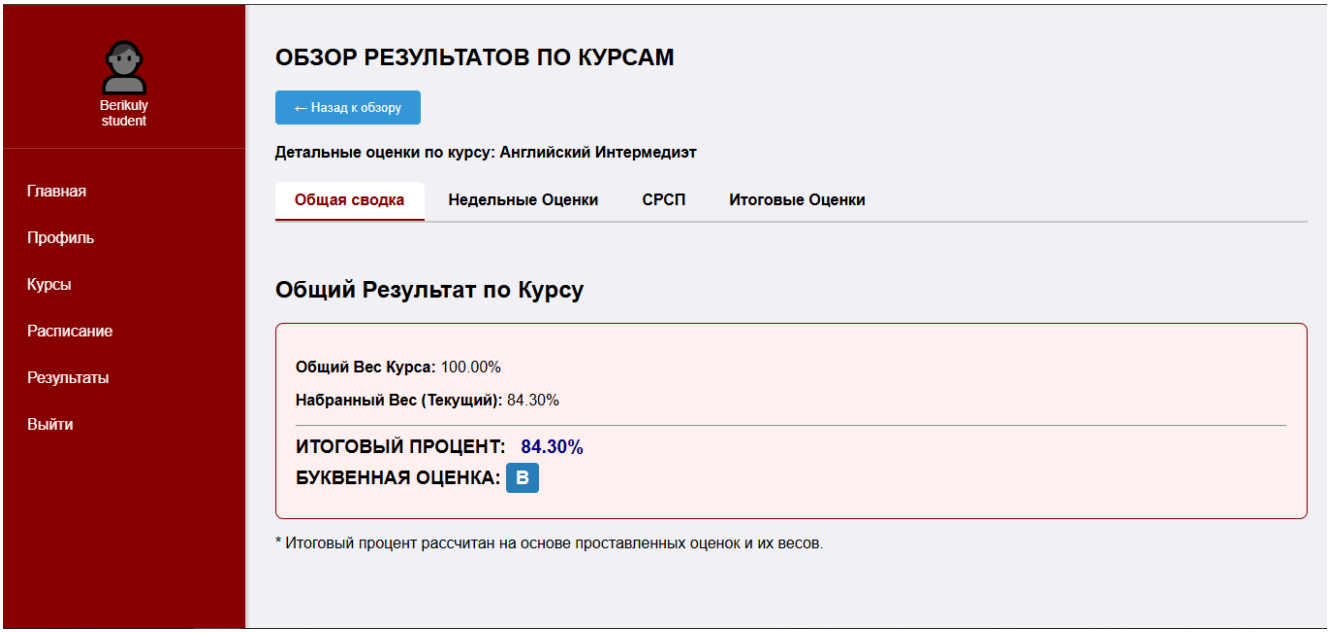


Время	Предмет	Аудитория	Преподаватель
Понедельник			
09:00 - 09:50	German-D1	E-105	herrBerikuly
10:00 - 10:50	German-D1	E-105	herrBerikuly
12:50 - 13:40	Programming	E-103	herrBerikuly
13:50 - 14:40	Programming	E-103	herrBerikuly
Вторник			
12:50 - 13:40	German-D1	E-105	herrBerikuly
13:50 - 14:40	German-D1	E-105	herrBerikuly
Среда			
12:50 - 13:40	English	E-106	herrBerikuly
13:50 - 14:40	English	E-106	herrBerikuly
Четверг			
09:00 - 09:50	German-D1	E-105	herrBerikuly

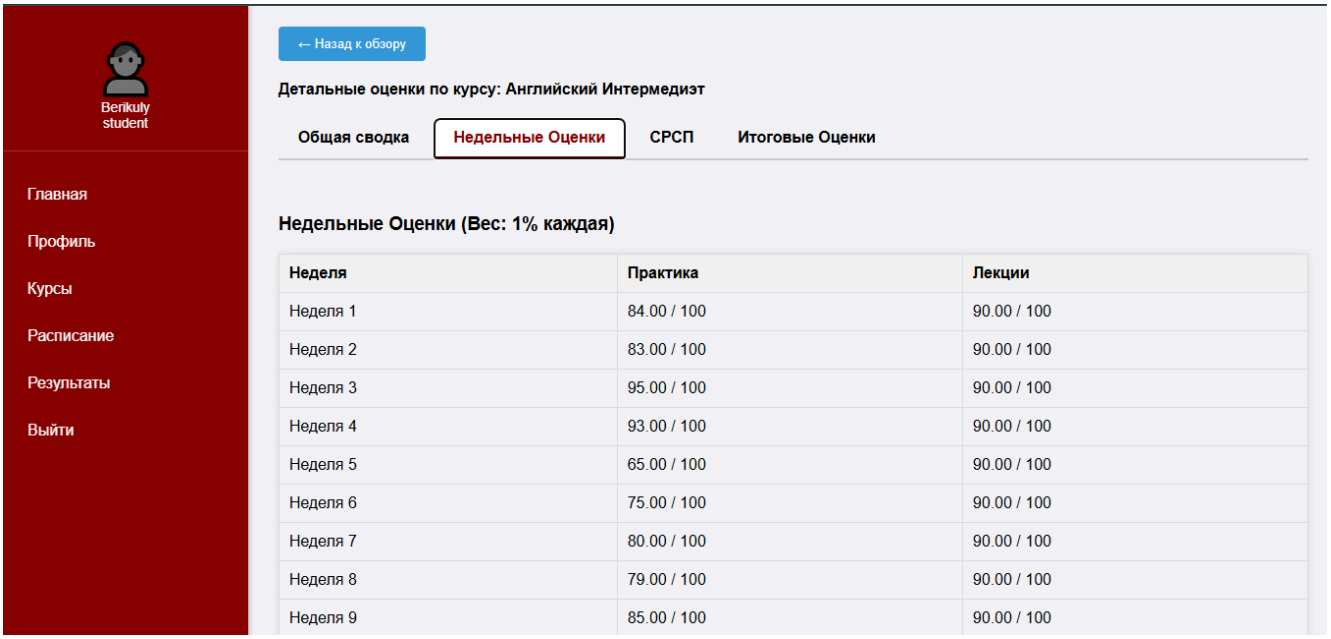
3.26 Сабақ кесте бөлімі (Студент)



3.27 Пән бағалары тақтасы (Студент)



3.28.1 Толық пән бағалары



3.28.2 Толық пән бағалары

ҚОРЫТЫНДЫ

1. Жобаның Нәтижелері және Мақсаттардың Орындалуы

Осы "Шағын Студенттік Портал" жобасының әзірленуі барысында қойылған барлық негізгі мақсаттар мен міндеттер толығымен орындалды. Жоба Node.js серверлік платформасы, MySQL дерекқоры және Full-stack JavaScript қағидаларына негізделген функционалды, қауіпсіз және тұрақты веб-қосымшаны құруға мүмкіндік берді.ы

Негізгі нәтижелер:

- Қауіпсіз Рөлдік Модель: Студент, оқытушы және әкімші рөлдері айқын бөлініп, әрбір API маршрутында (мысалы, server.js ішінде) қатаң рұқсат тексеруі жүзеге асырылды. Бұл деректердің рұқсатсыз қолданылуына жол бермейді.
- Тұрақты Сессия Жүйесі: Парольдерді SHA256 арқылы хештеу, сессия ID-лерін кездейсоқ генерациялау (crypto) және Cookie-лерді HttpOnly флагымен пайдалану арқылы жоғары деңгейдегі қауіпсіздік қамтамасыз етілді.
- Деректердің Тұтастығы: SQL транзакцияларының (saveSchedule функциясында) қолданылуы арқасында, кесте деректерін жаңарту кезінде ақпараттың логикалық қайшылыққа ұшырауы толығымен жойылды.
- Модульдік Архитектура: Бекенд логикасының Басқару/Маршруттау (server.js) және DAO/Бизнес-логика (auth.js) деңгейлеріне бөлінуі кодты оқуды, қолдауды және болашақта кеңейтуді жеңілдетеді.

2. Архитектуралық Шешімдердің Тиімділігі

Жобада таңдалған архитектуралық шешімдер оның тиімділігін арттырды:

- А. Node.js-ті Таңдаудың Тиімділігі

Node.js-ті пайдалану арқылы бір тілдік орта құрылды, бұл әзірлеуді жеңілдетіп қана қоймай, сонымен қатар сервердің дерекқормен және сұраулармен асинхронды жұмыс істеуі есебінен оның өнімділігін арттырды. Студенттік портал сияқты I/O операциялары (дерекқорға сұраулар) басым болатын жүйелер үшін бұл ең оңтайлы шешім болып табылады.

- Б. MySQL және Пулды Қолдану

MySQL-дің реляциялық тұрақтылығы оқу курстары мен кестелері сияқты құрылымды деректерді сақтауға тамаша сәйкес келді. Қосылым пулын қолдану сервердің тұрақтылығын және көптеген пайдаланушы бір мезгілде кірген кездегі реакция жылдамдығын қамтамасыз етті.

3. Келешектегі Даму Жоспарлары

Бұл жоба жұмыс істейтін негізгі үлгі болып табылады. Оны одан әрі дамыту үшін келесі бағыттар ұсынылады:

- Қауіпсіздікті Арттыру: Парольдерді хештеу үшін SHA256 орнына bcrypt немесе scrypt сияқты заманауи және баяу алгоритмдерге көшу.
- Функционалды Кеңейту: Бағалау жүйесін (бағалар, рейтингтер) немесе хабарландыру жүйесін (мысалы, WebSockets арқылы) қосу.
- Frontend-ті Жаңарту: Клиенттік JS-тің орнына React, Vue немесе Angular сияқты заманауи JavaScript фреймворктерін пайдалану арқылы интерфейстің күрделілігін және жылдамдығын арттыру.

Қорытындылай келе, "Шағын Студенттік Портал" жобасы Node.js, MySQL және қауіпсіз кодтау әдістерін тиімді қолданудың сәтті мысалы болып табылады. Жоба қойылған білім беру және техникалық міндеттерді толықтай шешті.

ПАЙДАЛЫНЫЛҒАН ӘДЕБИЕТТЕР

1. Бүркітбаев, М. Ә. Ақпараттық жүйелердің негіздері. — Алматы: Эверо, 2020.

2. Ермекбаев, Б. С. Веб-технологиялар және веб-бағдарламалау негіздері. — Алматы: Қазақ университеті, 2021.

3. W3Schools. “HTML, CSS және JavaScript Tutorials.” — https://www.w3schools.com

4. Mozilla Developer Network (MDN). “JavaScript Documentation.” — https://developer.mozilla.org

5. FreeCodeCamp. “Responsive Web Design and Front-End Development.” — https://www.freecodecamp.org

6. Khan Academy. “Intro to JS: Drawing & Animation.” — https://www.khanacademy.org/computing

7. ISO/IEC 25010:2011 — Systems and Software Engineering – System and Software Quality Models.

8. Stack Overflow Community. “Frontend Development Discussions and Code Examples.” — https://stackoverflow.com