Архитектурен проект на web-базираната система **UniTrack**

изготвен от: Спас Миленков, Мирела Методиева, Тедислав Жейнов, Димитър Вузев и Николай Парашкевов

28.10.2023 г.

Въведение

Документът има за цел да представи web-базираната система UniTrack от архитектурна гледна точка. Ще се разгледат по-обстойно нейното предназначение, архитектурен обзор и изисквания - функционални и нефункционални.

Изграждането и разработването на проекта се осъществява от екип 3, разпределен по следните роли:

- **Team Lead:**
  - Спас Миленков
- **Backend Team:**
  - Спас Миленков
  - Тедислав Жейнов
- **Front-end Team:**
  - Димитър Вузев

- Мирела Методиева
- Николай Парашкевов ○

DevOps Team:

- Спас Миленков
- Мирела Методиева

Предназначение

Обхват

В тази фаза от жизнения цикъл на проекта изграждането на архитектурата, дефинирането на диаграми, като use-case, activity и class диаграми, играе изключително важна роля при разработването на проекта UniTrack. Всяка от тези диаграми предоставя различен изглед върху архитектурата на системата и помага за разбирането и документирането на функционалностите и структурата на системата. Изготвянето на тези диаграми е съществена част от анализа и проектирането на системата UniTrack, тъй като те предоставят визуална репрезентация на ключовите аспекти на системата. Това позволява на разработчиците и заинтересованите страни да разберат как ще работи системата, какви функционалности ще бъдат налични и какви взаимодействия ще се случват между потребителите и компонентите на системата. Тези диаграми са средство за комуникация и документиране, което улеснява успешната разработка и внедряване на UniTrack.

Актьори

- **Студенти:** Профилна страница, оценки, отсъствия, седмичен график, статистика, препоръчителни материали.
- **Учители:** Профилна страница, лични данни, седмичен график, списък на студентите по групи, добавяне/премахване на оценки и отсъствия.
- **Училищни и Университетски Администратори:** Регистрация, редакция на данни.

- **Супер Администратори:** Одобрение на профили, деактивация на профили, създаване на седмични графици.

Използвани термини и символи

UML (Unified Modeling Language) диаграма е визуален инструмент, използван за моделиране и представяне на структурата, функционалността и взаимодействията на система или процес. UML е стандартен език за моделиране, широко използван в софтуерната индустрия и системните инженери. Този език предоставя набор от символи, обозначения и конвенции, които позволяват на инженерите и разработчиците да представят визуално различни аспекти на дадена система.

Use-Case Диаграми: Използват се за идентифициране и моделиране на функционалните изисквания на дадена система като сценарии или „случаи на употреба“. Тези диаграми показват как различните участници (потребители) комуникират със системата и каква функционалност трябва да поддържа.

Class Диаграми: Представят структурата на системата чрез дефиниране на класове (обекти) и техните взаимоотношения. Class диаграмите показват класове, техните атрибути, методи и как те се свързват помежду си.

Activity Диаграми: Те представят последователността на дейности и контролните структури в рамките на определен процес или функционалност. Activity диаграмите се използват за моделиране на процесите и потока на работа.

API: Application Programming Interface е набор от правила и протоколи, които позволяват на различни софтуерни приложения да комуникират помежду си. Това е начинът, по който различни компоненти на софтуерната система могат да взаимодействат и споделят информация. API действа като посредник между различни приложения, позволявайки им да използват функционалността и данните на други приложения.

CRUD: Акроним, който се използва, за да обозначи основни операции, които могат да бъдат извършвани върху данни. Тези операции са основа за управлението и манипулирането на данни във всяка система или приложение. Ключовото значение на CRUD операциите е, че те позволяват на потребителите да създават, четат, обновяват и изтриват данни в системата.

Актьори: Представлява роля на потребител, който взаимодейства със системата, която моделирате. Потребителят може да бъде човек, организация, машина или друга външна система.

Източници

1. https://www.uml-diagrams.org/
2. https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-usecases
3. https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-actors
4. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/umlclass-diagram-tutorial/
5. https://www.mindmanager.com/en/features/activity-diagram/

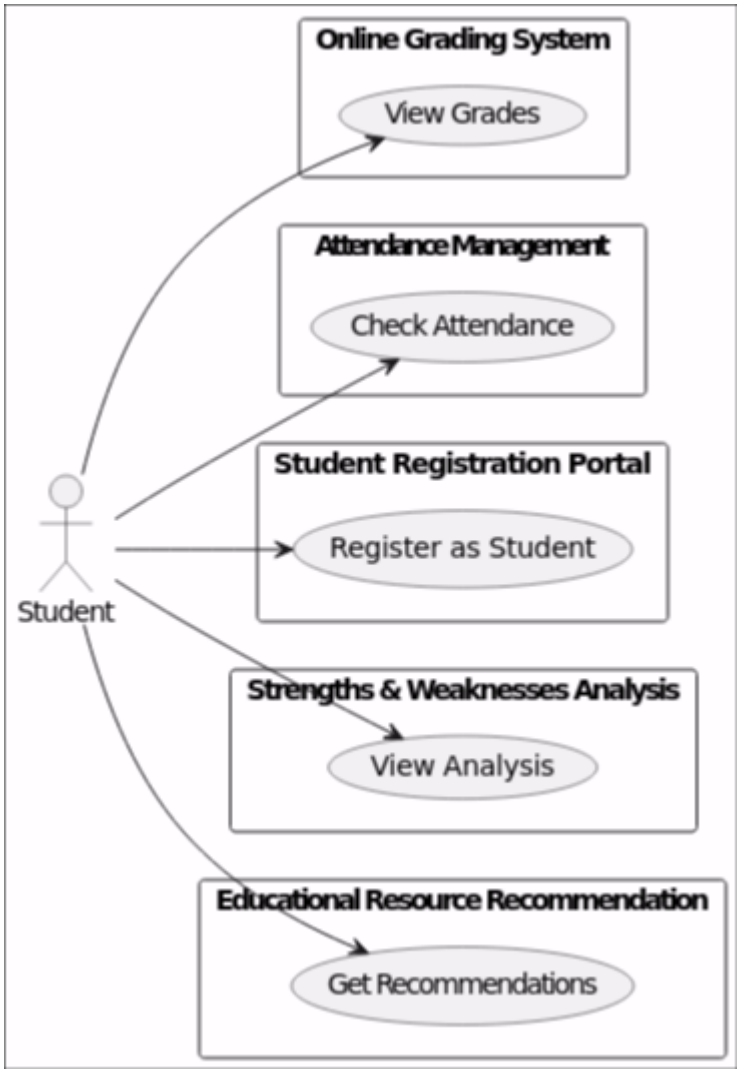
Архитектурен обзор

Use-Case Диаграми: Тези диаграми се използват за идентифициране и документиране на основните функционални сценарии или "примери на употреба", които системата трябва да поддържа. В контекста на UniTrack, те биха могли да включват сценарии като "Ученикът преглежда своята академична информация", "Учителят записва оценка за ученик" и други. Use-case диаграмите помагат да се определят изискванията на системата и да се визуализират основните интеракции между потребителите и системата.

Activity Диаграми: Тези диаграми се използват за моделиране на последователността и потока на дейности в рамките на конкретен процес или функционалност. В UniTrack, activity диаграмите биха могли да показват стъпките, които ученикът или учителят следва, когато използва определена функционалност, като например "Процес на записване на оценка" или "Процес на преглед на академична информация". Тези диаграми предоставят ясен обзор на дейностите в системата и как те се изпълняват.

Class Диаграми: Class диаграмите се използват за моделиране на структурата на системата чрез дефиниране на класове (обекти) и техните взаимоотношения. В UniTrack, class диаграмите биха могли да представят основните ентитети като "Ученик", "Учител", и други, както и техните атрибути и методи. Те помагат да се уточни структурата на данните и взаимодействията между тях.

1. Use-case изглед



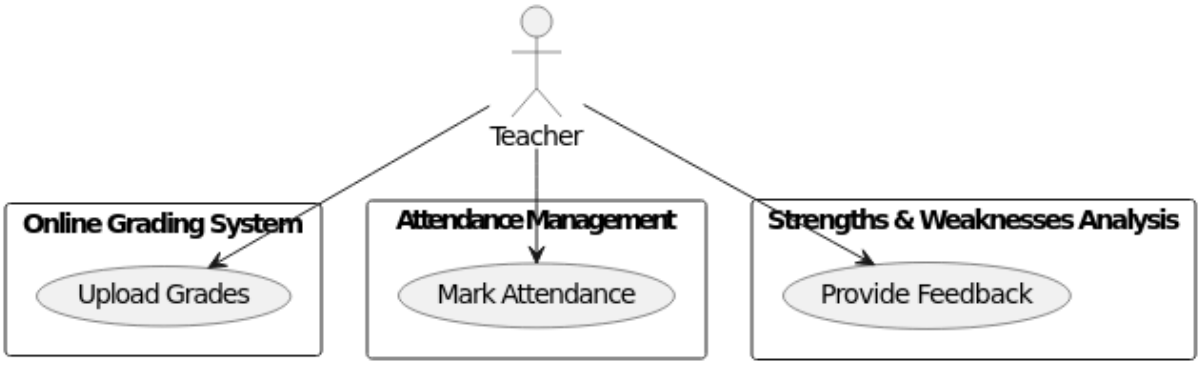
Актьор:

- **Ученик**: Основният потребител, който взаимодейства със системата.

Use-Cases:

1. **Online Grading System**: Система за онлайн оценяване.
   - **ViewGrades**: Студентите могат да преглеждат своите оценки.
2. **Attendance Management**: Система за управление на посещаемостта.

    - **CheckAttendance**: Студентите могат да проверяват своята присъственост.
3. **Student Registration Portal**: Портал за регистрация на студенти.
    - **RegisterStudent**: Възможност за регистрация като студент.
4. **Strengths & Weaknesses Analysis**: Анализ на силни и слаби страни.
    - **ViewAnalysis**: Студентите могат да преглеждат анализа за тяхните силни и слаби страни.
5. **Educational Resource Recommendation**: Препоръчване на образователни ресурси.
    - **GetRecommendations**: Студентите могат да получават препоръки за ресурси.
    - **AccessResourceLibrary**: Студентите имат достъп до библиотека с ресурси.

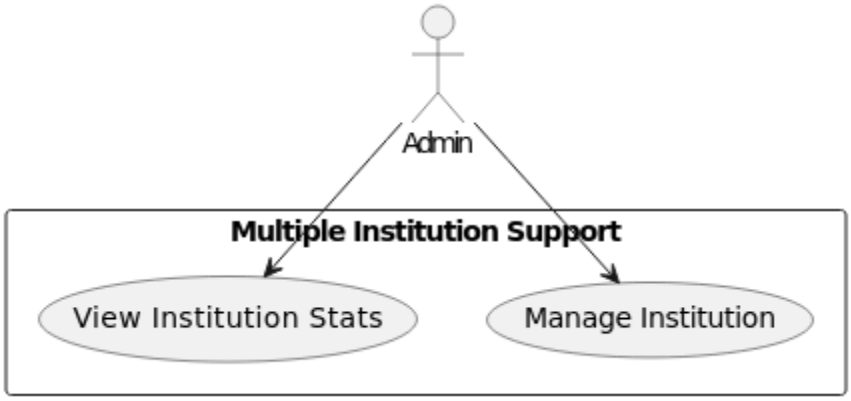


Актьор:

- **Teacher**: Учителят е основният потребител, който взаимодейства със системата в контекста на дадените дейности.

Use-Cases:

1. **Online Grading System**: Система за онлайн оценяване.
    - **UploadGrades**: Учителят има възможност да качва оценки на студентите.
2. **Attendance Management**: Система за управление на присъствието.
    - **MarkAttendance**: Учителят може да отбелязва присъствие или отсъствие на студентите.
3. **Strengths & Weaknesses Analysis**: Анализ на силни и слаби страни на студентите.
    - **ProvideFeedback**: Учителят има възможност да предоставя обратна връзка на студентите относно тяхното представяне.



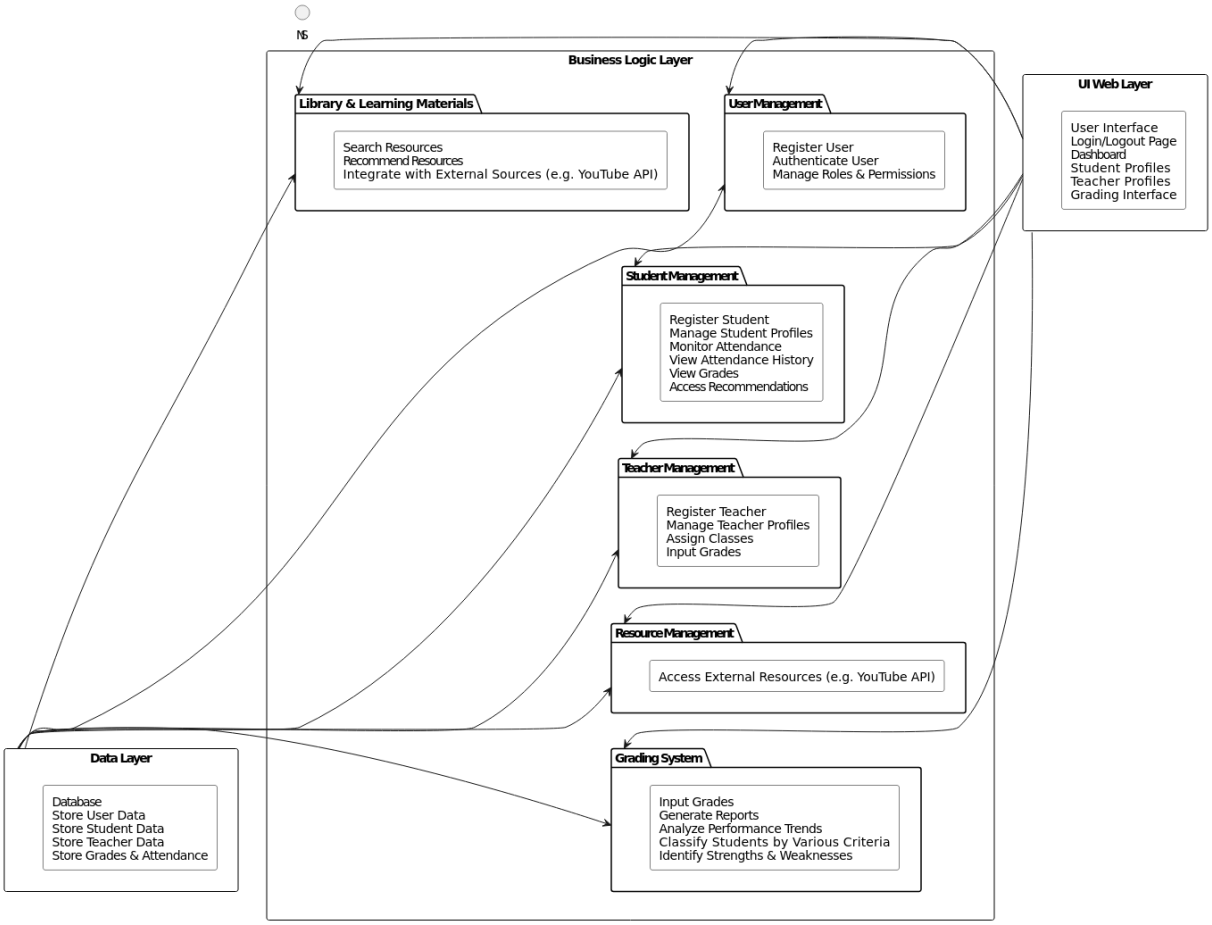
Актьор:

- **Admin**: Администраторът е основният потребител, който управлява и персонализира различни аспекти на системата.

Use-Cases:

1. **Multiple Institution Support**: Поддръжка на множество образователни институции.
    - **ManageInstitution**: Администраторът може да управлява различни образователни институции в рамките на системата.
    - **ViewInstitutionStats**: Администраторът може да преглежда статистическа информация за всяка институция.

2. Логически изглед

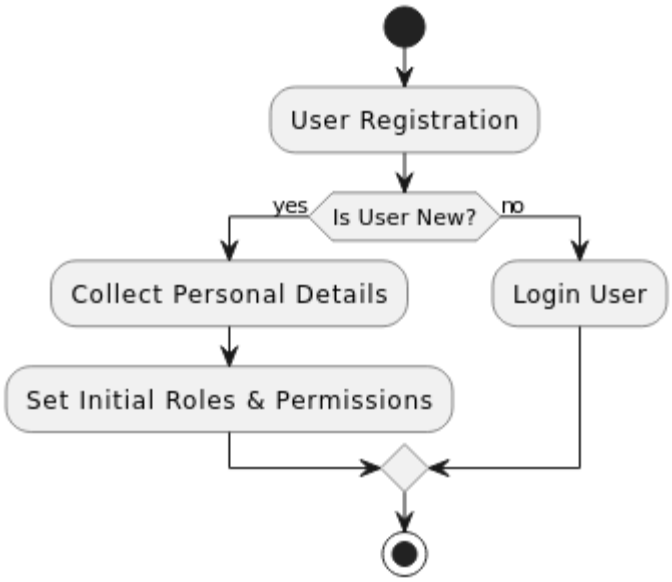


1. **Business Logic Layer:** Съдържа логическите модули на системата, които изпълняват различни операции и обработват информацията.
    - **User Management:** Занимава се с функционалностите за управление на потребители, включително регистрация, управление на роли и разрешения и др.
    - **Student Management:** Предоставя функционалности за управление на студенти, оценки и препоръчителни материали за обучение.
    - **Teacher Management:** Отговаря за регистрация и управление на профилите на учителите, въвеждане на оценки.
    - **Resource Management:** Управлява учебни ресурси и интеграция с външни източници като YouTube.
    - **Grading System:** Обработва оценки и анализира резултати.
2. **Data Layer:** Включва базата данни, която съхранява всичка информация от системата.
3. **User Interface:** Това е предният край на системата, през който потребителите взаимодействат с модулите в Business Logic Layer.

Взаимодействия:

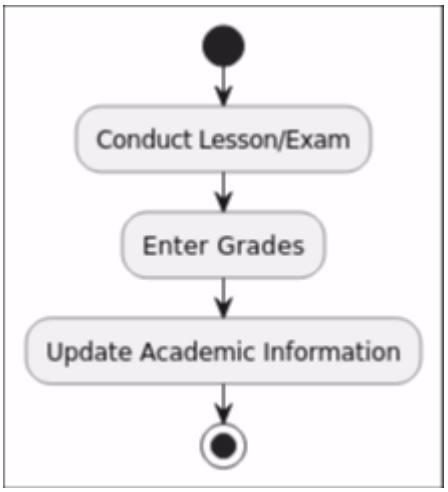
- Потребителският интерфейс (User Interface) комуникира директно с всички модули в Business Logic Layer.
- Всеки от модулите в Business Logic Layer се свързва с базата данни (Database) в Data Layer за съхранение и извличане на информация.

3. Процесен изглед



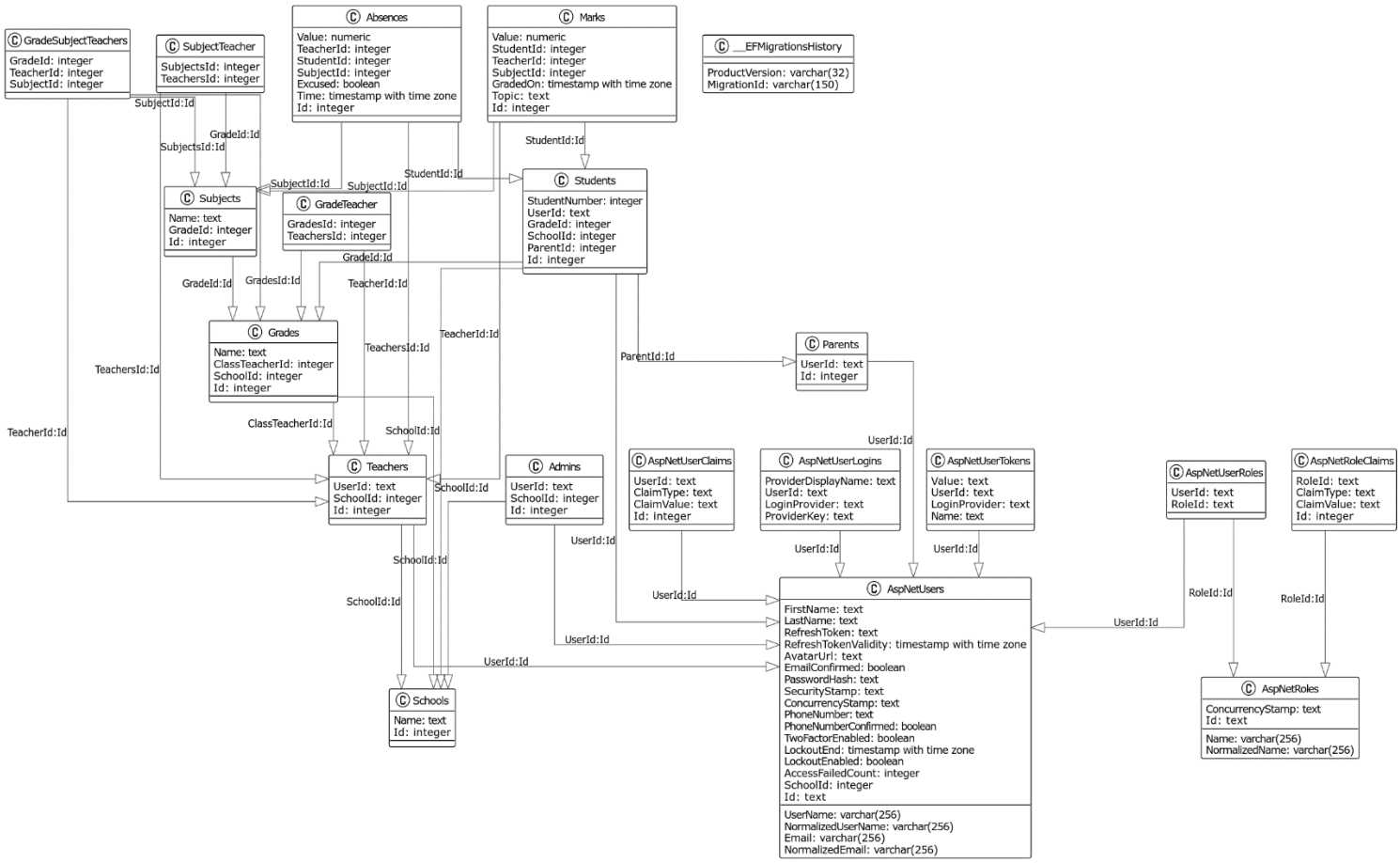
1. **Проверка дали потребителят е нов:** Системата проверява дали потребителят вече има съществуващ профил.
    - **да (yes):** Ако потребителят е нов, процесът продължава към следващите стъпки.
    - **не (no):** Ако потребителят вече има профил, той просто се влиза в системата ("Login User").

2. **Събиране на лични данни:** Системата иска от новия потребител да въведе своите лични данни.
3. **Задаване на начални роли и права:** След като потребителят въведе своите лични данни, системата му задава начални роли и права в зависимост от типа на регистрация или други критерии.
4. **Край:** Това е крайната точка на регистрационния процес, след която



1. **Провеждане на урок/изпит:** Началото на процеса, където се провежда учебен урок или изпит.
2. **Въвеждане на оценки:** След провеждането на изпита или урока, учителят или оценяващият въвежда оценките на студентите.
3. **Въвеждане на оценки (отново):** След като е получено известие за проблем или липсващи оценки, учителят отново въвежда оценките.
4. **Актуализиране на академичната информация:** След успешното въвеждане на оценките, академичната информация на студента се актуализира в системата.

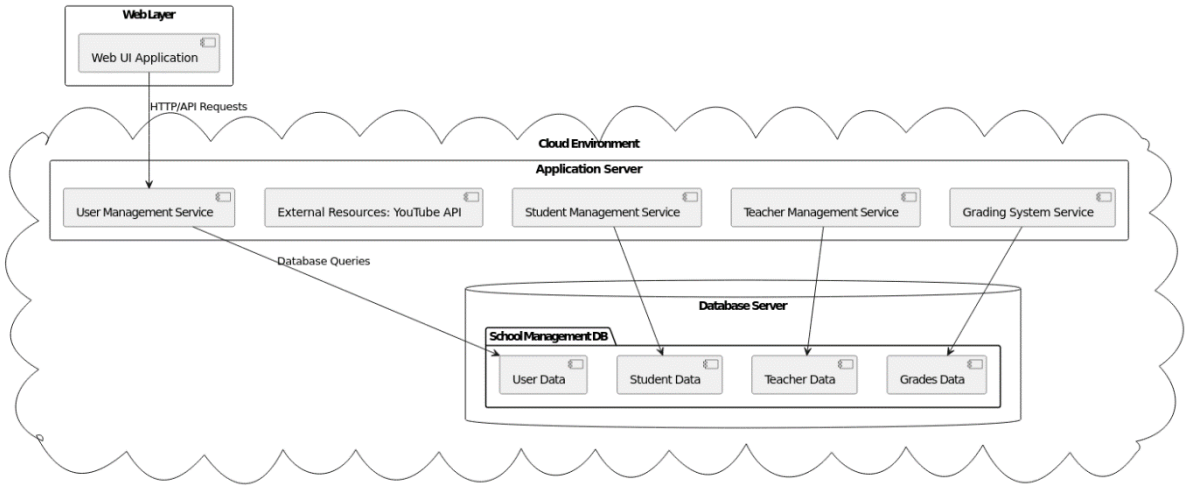
4. Изглед на данните



1. Entities:
    - School (Училище): Съдържа информация за училището като име и местоположение.
    - Admin (Администратор): Съдържа информация за администраторите на училищата.
    - Teacher (Учител): Характеристики като име, предмет, който преподава, и училище, в което работи.
    - Student (Ученик): Информация за учениците като име, клас и училище.
    - Parent (Родител): Детайли за родителите на учениците.
    - Grades (Оценки): Оценки, получени от учениците за различни курсове.
    - Absence (Отсъствие): Записи за отсъствията на учениците.
2. Intermediate Tables (Междинни таблици):

- Тези таблиците служат за свързване на различни ентитети в много-до-много връзки. Примери включват TeacherGrades, StudentGrades, TeacherAbsence, StudentAbsence .

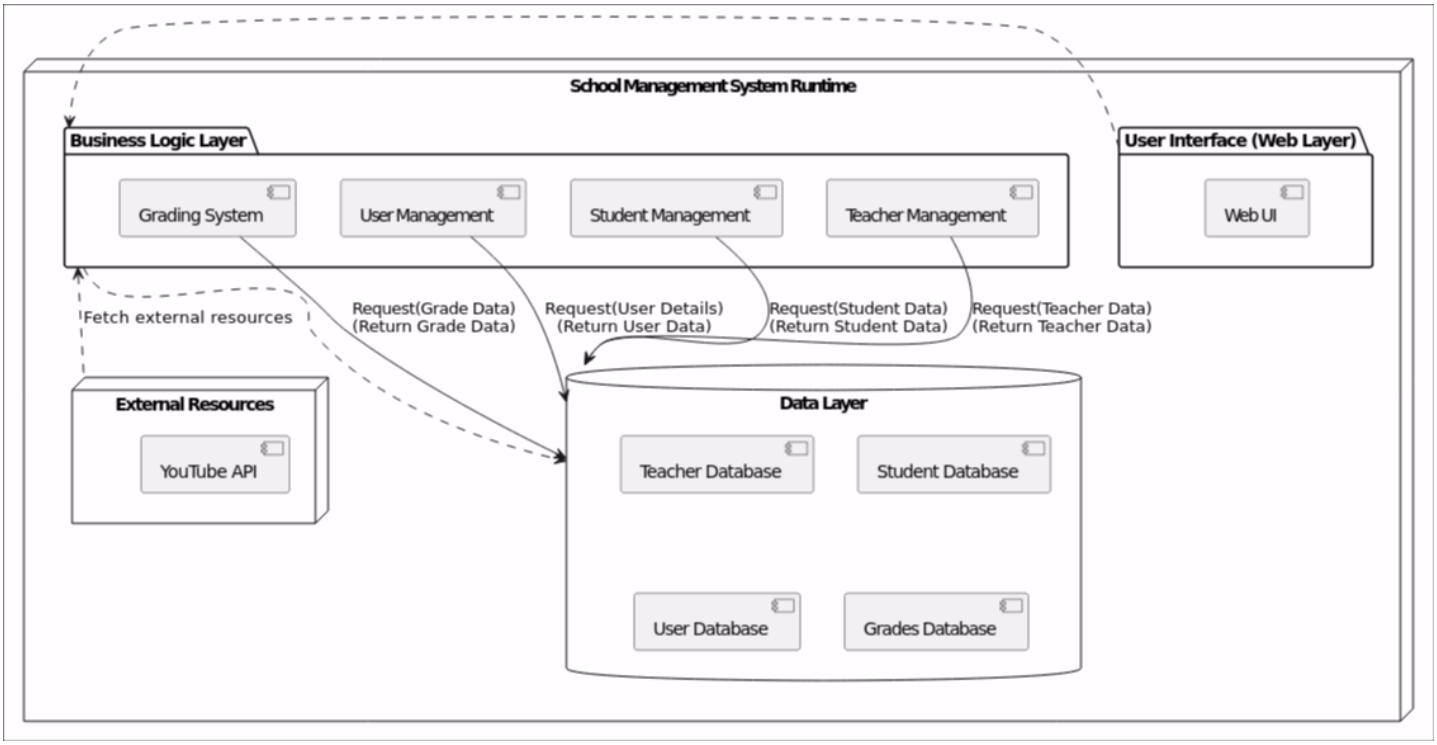
3. Relationships (Връзки):
   - Връзките представляват отношенията между различните ентитети и междинни таблици. Примери включват "assigns" (назначава), "views" (разглежда), "attends" (посещава) и др.



5. Изглед на внедряването

1. **User Interface (Web Layer)** (Потребителски интерфейс):
   - **Web UI**: Уеб интерфейсът, с който крайните потребители взаимодействат.
   - **API Gateway**: Точка на вход в системата, която управлява и маршрутизира API заявките.
2. **Data Layer** (Слой за данни):
   - Тази част на системата съхранява всички данни, свързани с потребители, ученици, учители, ресурси и оценки.
3. **External Resources**:
   - **YouTube API**: Външен източник, от който системата може да изтегли видео ресурси или друга информация.
4. **Business Logic Layer** (Слой с бизнес логика): Тук са разположени различните компоненти, които обработват заявките от интерфейса и работят с данните.

○ **User Management**, **Student Management**, **Teacher Management**, **Grading System**: Тези компоненти представляват различни модули на системата, които обработват заявките, свързани с тяхната функционалност. Те взаимодействат с базата данни (CRUD операции) и/или с външни услуги като YouTube API .



Взаимодействие:

○ Компонентите в "Business Logic Layer" получават заявки от "User Interface" и взаимодействат с "Data Layer" за CRUD операции.

○ Някои от компонентите, като "Teacher Management" , взаимодействат с "External Resources" като YouTube API, за да получат външни ресурси.

○ Връзките между различните модули и слоеве показват направлението на комуникацията и типа на информацията, която се обменя.

6. Изглед на имплементацията

Нефункционални изисквания

1. Достъпност

Този атрибут за качество е от съществена важност, тъй като трябва да се осигури непрекъсната достъпност до системата, която също така трябва да бъде удобна за използване от всички заинтересовани страни. За тази цел е необходимо да бъдат налице:

- Непрекъсната наличност: Системата трябва да бъде налична 24/7, без прекъсване за планирана поддръжка или възстановяване от смущения.
- Крос-платформена съвместимост: Системата е достъпна от различни устройства и операционни системи, включително компютри, смартфони и таблети тъй като е реализирана под формата на WEB система.
- Удобен интерфейс: Интерфейсът на системата е интуитивен и лесен за използване, така че дори независимо от техническия опит на потребителите, те да могат да го използват без проблем.

2. Разширяемост

Системата трябва да бъде способна да се разширява и допълва с нови функционалности и модули, без да изисква значителни промени в основната структура. Това е възможно чрез:

- Модулна архитектура: Системата е проектирана с модулна архитектура, която позволява лесното добавяне и премахване на модули.
- Използване на стандартизирани интерфейси: Разширенията и новите функционалности използват стандартизирани интерфейси, които улесняват интеграцията.
- Документация и обучение: Новите разработчици и администратори трябва да могат лесно да се запознаят със системата и процеса на добавяне на нови функционалности.
- Съвместимост с бъдещи технологични промени: Системата е проектирана с оглед на съвместимост с бъдещи технологични тенденции и стандарти. Модуларната архитектура и използването на стандартизирани

интерфейси позволява внедряването на промени съобразени с бъдещите технологични тенденции.

3. Производителност

Анализират се ресурсите, необходими за постигане на желаната производителност, при различни сценарии. Използваните тактики включват ограничаване на натоварването, планиране и др

- ○ Време за зареждане на началната страница
  - Изисква се зареждане на началната страница за по-малко от 3 секунди, за да осигури бърз достъп до информацията за потребителите.
- ○ Поддръжка на множество потребители
  - Системата трябва да поддържа поне 10 000 активни потребители, които да могат да влизат, търсят информация и използват приложенията едновременно, без значително забавяне на отговора.
- ○ Изпълнение на заявки за оценки
  - Системата трябва да обработва заявки за преглед и обновление на ученически оценки в рамките на 1 секунда, за да осигури бърза информация за успехите на учениците.

4. Сигурност

Сигурността трябва да бъде интегрирана в цялостния дизайн и развитие на платформата и да бъде постоянно проследявана и подобрявана, за да се осигури защита на данните и непрекъсната достъпност на системата при различни ситуации.

- o Ограничаване на достъпа: Управляване правата на потребителите според техните роли.
- o Защита на данните: Използване криптиране и сигурни бази данни.
- o Обновления и пачове: Поддържане на системата актуална.

5. Възможност за тестване

Интеграция на технологии и инструменти за тестване на приложения:

○ Използване на инструменти за тестване на C# код:
- Система трябва да бъде проектирана със съвместимост с инструменти за тестване на C# код, като xUnit.

○ Документация и коментари в кода:
- Създаване на подробни коментари и документация в кода, които да обясняват работата на всяка част от приложението, което да помага на тестерите при изготвянето на тестови случаи.

6. Интероперабилност

Системата трябва да бъде способна да се интегрира с други софтуерни системи и технологии, които се използват в учебната среда, като системи за управление на учебни материали, системи за дистанционно обучение, електронни учебници и други. Това ще стане чрез:

○ Поддържане на стандарти: Системата трябва да използва стандартизирани протоколи и формати за обмен на данни.
○ API и уеб услуги: Предоставяне на добре документирани API и уеб услуги, които позволяват на други системи да комуникират с вашата платформа.
○ Съвместимост с различни операционни системи и браузъри: Поддържане на различни операционни системи и уеб браузъри, така че потребителите могат да използват системата от различни устройства.

7. Използваемост

Системата трябва да бъде удобна и лесна за използване от всички потребители, включително ученици, учители и администратори. Това ще бъде постигнато чрез:

○ Интуитивен потребителски интерфейс: Дизайн на потребителския интерфейс, който е интуитивен и позволява на потребителите лесно да навигират и извършват задачи.
○ Достъпност(роли): Потребителите с различни нива на техническа подготовка и евентуални ограничения трябва да могат да използват системата.
○ Бързина и ефективност: Системата трябва да осигури бърз достъп до информация и функционалности, без излишни стъпки или забавяния.