Studentrack

Actors: User, Student, Lecturer, Recruiter System

Use Cases: Manage Premium Features, Manage Course Participation, Read and Analyze Statistics, Login User, Manage Time Order, Manage User, Register User, Manage Module

Use Case	Register User
Pre Conditions	• User is currently not registered
Description	• User enters relevant data (first name, last name, birthday, password, mail address, type: student (major) or lecuturer (faculty, title)) • Check for potential duplicate by mail address • Show confirmation
Post Conditions	• User is registered now • User data is saved persistently

Use Case	Login User
Pre Conditions	• User is registered
Description	• Users enters login data (mail address and password) • Validate login data • Show confirmation and welcome page
Post Conditions	• User is loggged in and has an active session

Use Case	Manage User
Pre Conditions	• User is logged in
Description	• User gets formular/page for managing own data • User changes data of their choice (same relevant data as "Register User"), including deletion of account • Validate data • Enter user password for data changes
Post Conditions	• User data is changed • Changes are stored persistently

Use Case	Manage Premium Features
Pre Conditions	• User is logged in • User is student • User has a bank account
Description	• User gets information about the current state of premium features • User enables/disables premium features • User allows/prohibits payment (one time only or as suscription) • Transaction at bank system is triggered • Show confirmation
Post Conditions	• User premium status is updated • Bank is informed • Change are stored persistently

Use Case	Manage Course Participation
Pre Conditions	• User is logged in • User is student • User has no/failed grade in related module for registration OR • User is currently enrolled for course
Description	• User gets formular for course registration • User chooses courses • User confirms the choices for signing in/out • Add/Remove Time Order for User in course • Show Confirmation
Post Conditions	• Course Participation of User is updated • Time Orders are managed according to updates • Changes are stored persistently

Use Case	Read and Analyze Statistics
Pre Conditions	• User is logged in • User has correct permissions for requested statistic depending on User type and premium features
Description	• User chooses statistic of interest (about themself or global anonymized statistics) • User gets data about the requested topic as confirmation
Post Conditions	

Use Case	Manage Time Order
Pre Conditions	• User is logged in • User has access to own (and own course related) Time Orders • User is student
Description	• User gets overview page about own time orders (with options for create, read, change, delete) • User edits start/end/both of Time Order • User confirms changes (if present) • Show confirmation
Post Conditions	• Time Order data is changed (for create, change or delete) • Changes are stored persistently

Use Case	Manage Module
Pre Conditions	• User is logged in • User is lecturer
Description	• User gets overview page about own modules • User edits Module data • User confirms changes (if present) • Show confirmation
Post Conditions	• Module data is changed (for create, change or delete) • Changes are stored persistently

User
- long userId
- String firstName
- String lastName
- Date birthday
- String password
- String mailAddress
- Date membershipStart

Major
- long majorId
- String academicTitle
- int studentNumber

Student
- int ects
- Grade averageGrade
- boolean isPremiumUser

Lecturer
- String title

Module
- long moduleId
- String name
- int ects
- int creditHours
- String description

Course
- long courseId
- Date startDate
- Date endDate

Faculty
- long facultyId
- String name

Grade
- long gradeId
- double value
- String description
- int tryNumber

TimeInvest
- long timeInvestId
- long duration
- Course course

TimeOrder
- long timeOrderId
- Timestamp start
- Timestamp end

Relationships / labels:
- located at
- studies
- is responsible for
- is dean of
- works for
- takes
- makes
- 0..*
- 1

Presentation

WebApp

<<http/html>>

Services

ModuleService

StatisticService

UserService

TimeService

TimeManagementSystem

RecruiterSystem

IUserService

BankSystem

<<interface>>
IUserService

+ loginUser(User): User

+ getUserData(User): User

+ getGrade(User, Module): Grade

+ getAllGrades(User): Map<Module, Grade>

+ getTimeInvest(User, Module): TimeInvest

+ getAllTimeInvests(User): Map<Module, TimeInvest>

Persistence

Grade Repo

Module Repo

UserRepo

TimeRepo