@startuml

skinparam classAttributeIconSize 0

skinparam shadowing false

skinparam linetype ortho

title Student Management System – Unified Class Diagram (Domain + Repository + Services + Search/Sort + Exceptions)

' =======================

' PACKAGES

' =======================

package "Domain" {

abstract class User {

- userId : int

- name : String

- email : String

- username : String

- password : String

+ login() : void throws AuthenticationException

+ logout() : void

+ upload(file: FileType) : void throws UploadException, ValidationException

}

class Admin {

+ manageCourse() : void

+ manageStudent() : void

+ registerStudent(name: String, email: String) : Student throws ValidationException

+ assignBatch(student: Student, batch: Batch) : void throws NotFoundException

+ assignCourse(student: Student, course: Course) : void throws NotFoundException

+ assignCourse(teacher: Teacher, course: Course) : void throws NotFoundException

+ setDefaultPassword(user: User) : void throws ValidationException

+ verifyAdmission(student: Student) : void throws ValidationException

+ createCredentials(user: User, email: String, password: String) : void throws ValidationException

+ addCourseToStudent(student: Student, course: Course) : void throws NotFoundException

+ addCourseToFaculty(teacher: Teacher, course: Course) : void throws NotFoundException

}

class Teacher {

- id : String

- name : String

- email : String

- username : String

- password : String

- courses : List<Course>

- batches : List<Batch>

+ changePassword(newPassword: String) : void throws ValidationException

+ createAssignment() : void throws ValidationException

+ markAttendance() : void throws ValidationException

+ viewStudents() : void

+ viewSubmittedAssignments() : void

+ viewAllAttendance() : void

}

class Student {

- id : String

- name : String

- username : String

- password : String

- dept : Department

- batch : Batch

- courses : List<Course>

+ changePassword(newPassword: String) : void throws ValidationException

+ viewAssignment() : void

+ markOwnAttendance() : void throws ValidationException

+ viewAttendance() : void

+ viewGrades() : void

}

class Principal {

+ addAdmin(admin: Admin) : void

+ appointTeacher(teacher: Teacher) : void

+ assignCourse(teacher: Teacher, course: Course) : void

+ viewReports() : void

}

class Department {

- name : String

}

class Course {

- courseId : String

- courseName : String

- creditHours : int

- facultyName : String

- classDays : String

- classTimes : String

- classDates : String

}

class Batch {

- yearRange : String "e.g., 2024-2028"

}

class Attendance {

- studentRegNo : String

- courseCode : String

- classDates : String

- status : String

}

class StudentTable {

- studentName : String

- studentRegNo : String

- department : String

- yearOfStudy : String

- courseCode : String

- studentEmail : String

}

' Inheritance

User <|-- Admin

User <|-- Teacher

User <|-- Student

User <|-- Principal

' High-level associations (without 1..\* or labels)

Principal --> Admin

Admin --> Teacher

Admin --> Student

Admin --> Course

Admin --> Batch

Teacher --> Course

Teacher --> Batch

Student --> Course

Student --> Batch

Student --> Department

Attendance --> Student

Attendance --> Course

}

package "Repositories (Persistence)" {

interface "Repository<T>" as IRepository {

+ add(item: T) : void throws RepositoryException, ValidationException

+ update(item: T) : void throws RepositoryException, NotFoundException

+ delete(item: T) : void throws RepositoryException, NotFoundException

+ getAll() : List<T> throws RepositoryException

+ find(criteria: String) : List<T> throws RepositoryException

}

class AdminRepository

class TeacherRepository

class StudentRepository

IRepository <|.. AdminRepository

IRepository <|.. TeacherRepository

IRepository <|.. StudentRepository

AdminRepository ..> Admin

TeacherRepository ..> Teacher

StudentRepository ..> Student

TeacherRepository ..> Attendance

StudentRepository ..> Course

}

package "Services (Cross-cutting)" {

interface "UploadService<T>" as IUploadService {

+ validate(file: T) : boolean throws ValidationException

+ store(file: T) : void throws UploadException

+ saveMetadata(file: T) : void throws UploadException

}

class FileUploadService

IUploadService <|.. FileUploadService

User ..> IUploadService : delegates upload()

}

package "Search" {

interface "Searchable<T>" as ISearchable {

+ search(criteria: String) : List<T>

}

Admin ..|> ISearchable : <Object>

Teacher ..|> ISearchable : <Student>

Student ..|> ISearchable : <Course>

}

package "Sort" {

interface "Sortable<T>" as ISortable {

+ sort(criteria: String) : List<T>

}

Admin ..|> ISortable : <Object>

Teacher ..|> ISortable : <Student>

Student ..|> ISortable : <Course>

}

' =======================

' EXCEPTIONS

' =======================

package "Exceptions" {

abstract class BaseException {

- message : String

- timestamp : DateTime

+ log() : void

}

class AuthenticationException

class AuthorizationException

class ValidationException

class NotFoundException

class UploadException

class RepositoryException

BaseException <|-- AuthenticationException

BaseException <|-- AuthorizationException

BaseException <|-- ValidationException

BaseException <|-- NotFoundException

BaseException <|-- UploadException

BaseException <|-- RepositoryException

}

' Exception associations

User ..> AuthenticationException : may throw

User ..> UploadException : may throw

User ..> ValidationException : may throw

Admin ..> AuthorizationException : may throw

Admin ..> NotFoundException : may throw

IRepository ..> RepositoryException : may throw

IRepository ..> ValidationException : may throw

IRepository ..> NotFoundException : may throw

IUploadService ..> UploadException : may throw

IUploadService ..> ValidationException : may throw

' =======================

' ROLE-BASED NOTES

' =======================

note right of Admin

Can search & sort:

- Students by roll/name

- Teachers by ID/name

- Courses by code/name

Access: FULL

end note

note right of Teacher

Can search & sort:

- Their students

- Attendance by date

Access: RESTRICTED (own classes)

end note

note right of Student

Can search & sort:

- Own records

- Enrolled courses / public list

Access: LIMITED

end note

note bottom of FileUploadService

Role-based upload rules (examples)

- Admin: official materials (PDF, DOCX, JPG)

- Teacher: study materials (PDF, PPT, MP4)

- Student: assignments (DOCX, ZIP, PPT, PDF)

Checks: type, size, due date

Auto-foldering: by Course/StudentID

end note

@enduml