ESSEC Visionary

Melek elmoula

Introducing a Java application that redefines university administration and exam management . With its intuitive JavaFX interface and AI-driven exam grading, it simplifies the management of students, professors, groups, paths, subjects, and exams, enhancing both efficiency and accuracy.

Technologies Used:

•JavaFX: Used for building modern, feature-rich desktop applications in Java .

•BootstrapFX: Implements Bootstrap-like styling for modern and responsive UI components.

•MaterialFX: Utilizes material design principles for a contemporary and intuitive user experience.

•MySQL Connector/J: Facilitates interaction with MySQL databases for data storage and retrieval.

•JavaMail (javax.mail): Handles email functionalities for notifications and communication.

•Gson: Manages JSON serialization and deserialization for data interchange.

•HikariCP: Optimizes database connection pooling for improved performance.

•PDFBox: Provides functionality for creating and manipulating PDF documents.

Project Management and Methodology:

•**JIRA**: Used for tracking issues, managing tasks, and facilitating project planning and tracking.

•**Scrum**: Employed as the agile methodology for iterative development, including regular sprints, stand-up meetings, and sprint reviews to ensure continuous progress and adaptation.

Implementation Notes:

•MVC Architecture: Structures the application into Model, View, and Controller layers, promoting separation of concerns and maintainability.

•Efficient Data Handling: Employs HikariCP for superior database connection management, ensuring fast and reliable access to data.

•Database Management: Integrates MySQL Connector/J for database interactions and MySQL Procedural Language for complex operations and events.

•AI Integration: Utilizes Ollama AI for automated grading and feedback, enhancing efficiency and accuracy in evaluations.

•Email Integration: Leverages JavaMail for seamless email communications within the application.

•UI/UX Best Practices: Focuses on delivering an exceptional user experience with well-designed, responsive interfaces that adapt to user needs.

•Dynamic Theme Switching: Supports dynamic switching between dark and light modes, allowing users to customize their visual experience for comfort and preference.

Exception Handling:

Incorrect Login Exception: Manages failed login attempts with a custom exception to handle authentication errors.

Deadline Exceeded Exception: ensures that the system provides timely feedback to the professor about missed deadlines and may enforce restrictions based on the business logic.

EmailNotFoundException: A custom exception that handles cases where an email address provided by the user does not exist in the system database or directory. It can be used to ensure that the user is informed when their email cannot be found during processes such as password recovery

OccupiedDateException: This exception is thrown when a professor attempts to schedule an exam on a date or time that is already occupied or reserved. It ensures that scheduling conflicts are avoided and provides an opportunity for the professor to select an alternative date.

ExamsNumberExceededException: This custom exception addresses situations where the number of allowed exams or test attempts has been exceeded for a given subject. It is used to restrict professor from surpassing the predefined exam limits set by the system.

Login Credential Recovery:

•**Email-Based Recovery**: Allows users to recover their login credentials through email, ensuring secure and user-friendly password management.

Data Warehouse for Student Success Rate:

•**Success Rate Tracking**: The data warehouse tracks annual student success rates by group, path, and gender.

•**Data Analysis**: Computes the percentage of successful students each year, offering insights to help the institution identify trends and make informed decisions to enhance educational outcomes.

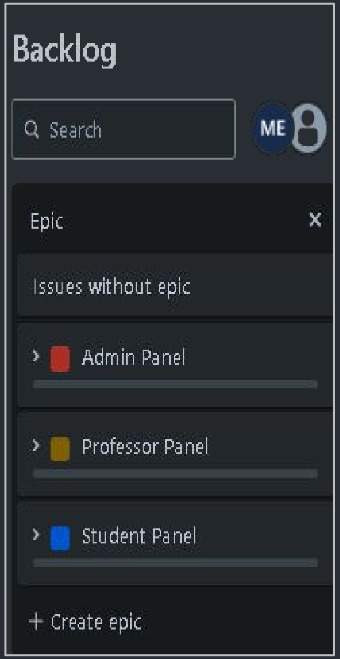
Applying New Concepts:

Generics , HashMap and Observable Lists in Java, along with MySQL Procedural Language for Advanced Database Scripting and Event Management"

Visualize Data Warehouse in PowerBI

Integrating data stored in a data warehouse with PowerBI's visualization capabilities to create insightful and interactive charts and graphs

- JIRA Software on the Atlassian platform to streamline project management.



Backlog

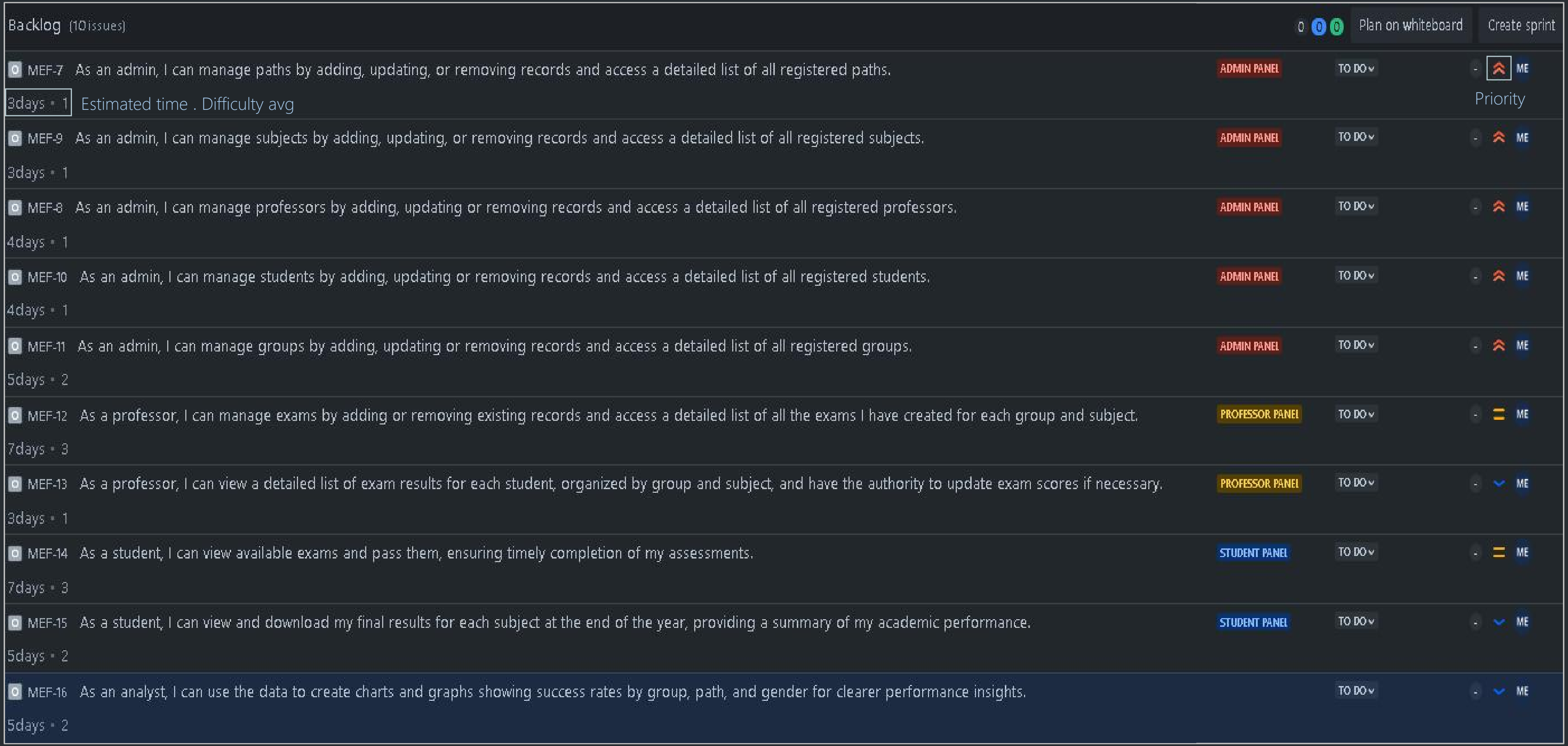
Search ME

Epic ×

Issues without epic

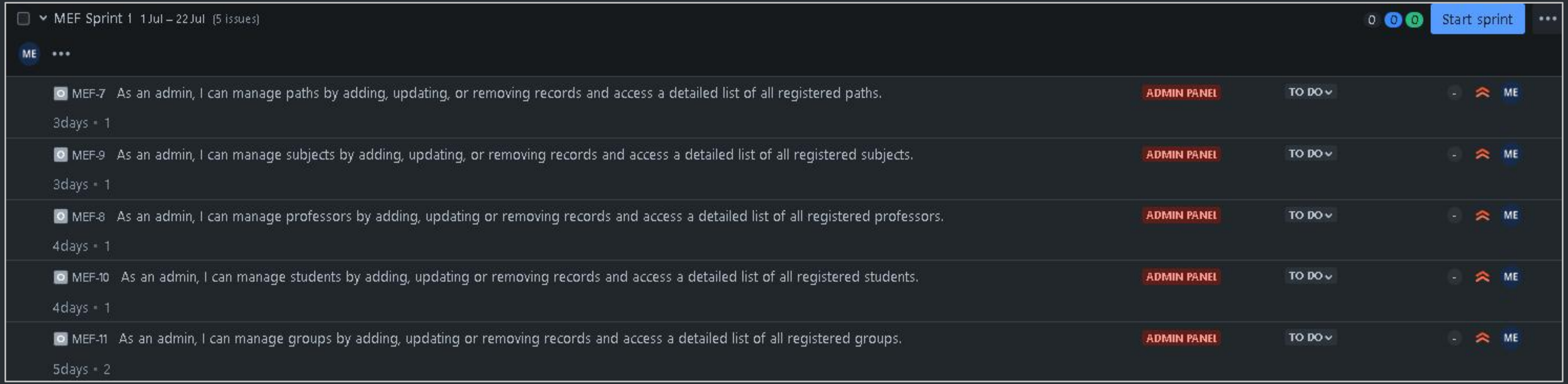
- 🟥 Admin Panel
- 🟨 Professor Panel
- 🟦 Student Panel

+ Create epic



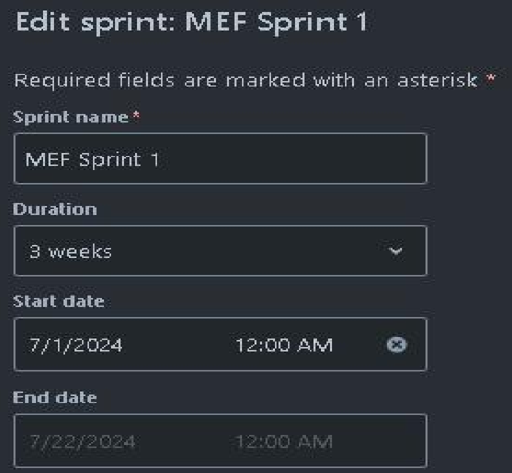
Backlog (10 issues) 0 0 0 Plan on whiteboard Create sprint

MEF-7	As an admin, I can manage paths by adding, updating, or removing records and access a detailed list of all registered paths.	ADMIN PANEL	TO DO	⌃⌃	ME
3days • 1	Estimated time . Difficulty avg			Priority	
MEF-9	As an admin, I can manage subjects by adding, updating, or removing records and access a detailed list of all registered subjects.	ADMIN PANEL	TO DO	⌃⌃	ME
3days • 1					
MEF-8	As an admin, I can manage professors by adding, updating or removing records and access a detailed list of all registered professors.	ADMIN PANEL	TO DO	⌃⌃	ME
4days • 1					
MEF-10	As an admin, I can manage students by adding, updating or removing records and access a detailed list of all registered students.	ADMIN PANEL	TO DO	⌃⌃	ME
4days • 1					
MEF-11	As an admin, I can manage groups by adding, updating or removing records and access a detailed list of all registered groups.	ADMIN PANEL	TO DO	⌃⌃	ME
5days • 2					
MEF-12	As a professor, I can manage exams by adding or removing existing records and access a detailed list of all the exams I have created for each group and subject.	PROFESSOR PANEL	TO DO	=	ME
7days • 3					
MEF-13	As a professor, I can view a detailed list of exam results for each student, organized by group and subject, and have the authority to update exam scores if necessary.	PROFESSOR PANEL	TO DO	⌄	ME
3days • 1					
MEF-14	As a student, I can view available exams and pass them, ensuring timely completion of my assessments.	STUDENT PANEL	TO DO	=	ME
7days • 3					
MEF-15	As a student, I can view and download my final results for each subject at the end of the year, providing a summary of my academic performance.	STUDENT PANEL	TO DO	⌄	ME
5days • 2					
MEF-16	As an analyst, I can use the data to create charts and graphs showing success rates by group, path, and gender for clearer performance insights.		TO DO	⌄	ME
5days • 2					


MEF Sprint 1 1 Jul – 22 Jul (5 issues) 0 0 0 Start sprint

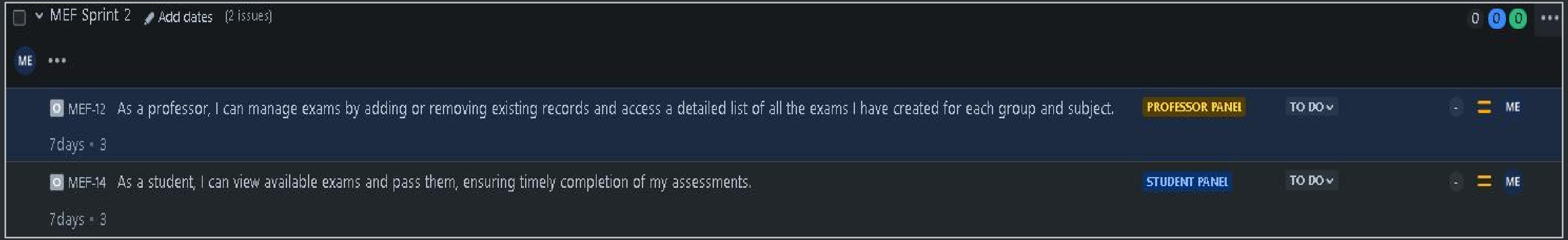
ME •••

MEF-7 As an admin, I can manage paths by adding, updating, or removing records and access a detailed list of all registered paths. 3days • 1	ADMIN PANEL	TO DO	-	ME
MEF-9 As an admin, I can manage subjects by adding, updating, or removing records and access a detailed list of all registered subjects. 3days • 1	ADMIN PANEL	TO DO	-	ME
MEF-8 As an admin, I can manage professors by adding, updating or removing records and access a detailed list of all registered professors. 4days • 1	ADMIN PANEL	TO DO	-	ME
MEF-10 As an admin, I can manage students by adding, updating or removing records and access a detailed list of all registered students. 4days • 1	ADMIN PANEL	TO DO	-	ME
MEF-11 As an admin, I can manage groups by adding, updating or removing records and access a detailed list of all registered groups. 5days • 2	ADMIN PANEL	TO DO	-	ME


Edit sprint: MEF Sprint 1

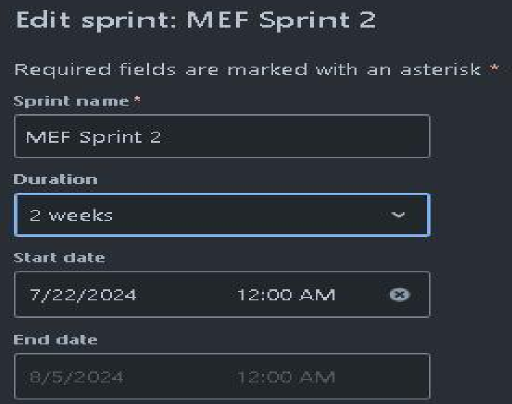
Required fields are marked with an asterisk *

Sprint name *	MEF Sprint 1
Duration	3 weeks
Start date	7/1/2024 12:00 AM
End date	7/22/2024 12:00 AM


MEF Sprint 2 Add dates (2 issues) 0 0 0

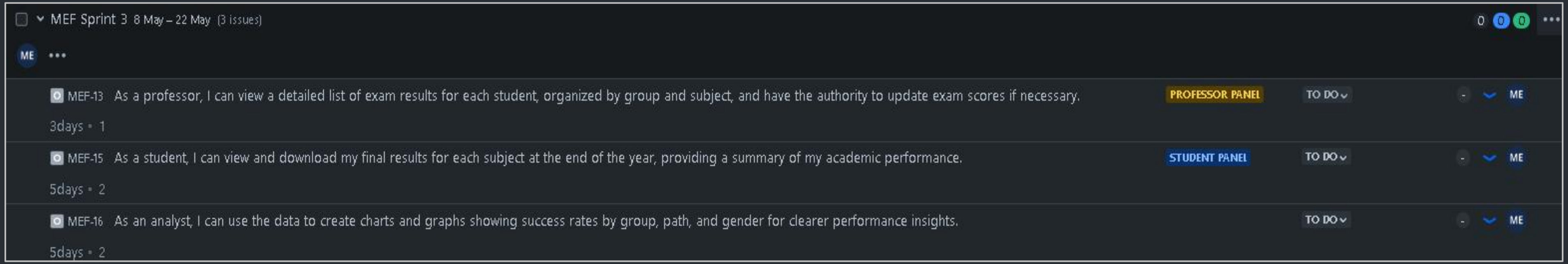
ME •••

MEF-12 As a professor, I can manage exams by adding or removing existing records and access a detailed list of all the exams I have created for each group and subject. 7days • 3	PROFESSOR PANEL	TO DO	-	ME
MEF-14 As a student, I can view available exams and pass them, ensuring timely completion of my assessments. 7days • 3	STUDENT PANEL	TO DO	-	ME


Edit sprint: MEF Sprint 2

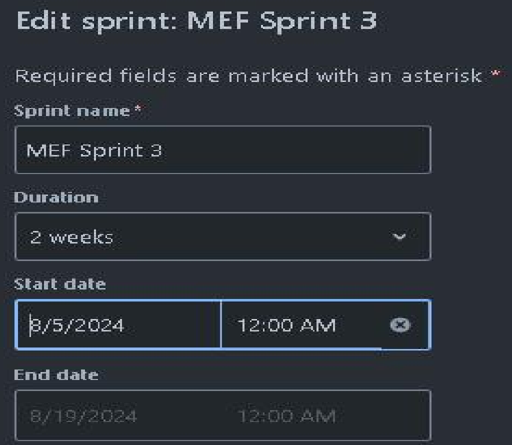
Required fields are marked with an asterisk *

Sprint name *	MEF Sprint 2
Duration	2 weeks
Start date	7/22/2024 12:00 AM
End date	8/5/2024 12:00 AM


MEF Sprint 3 8 May – 22 May (3 issues) 0 0 0

ME •••

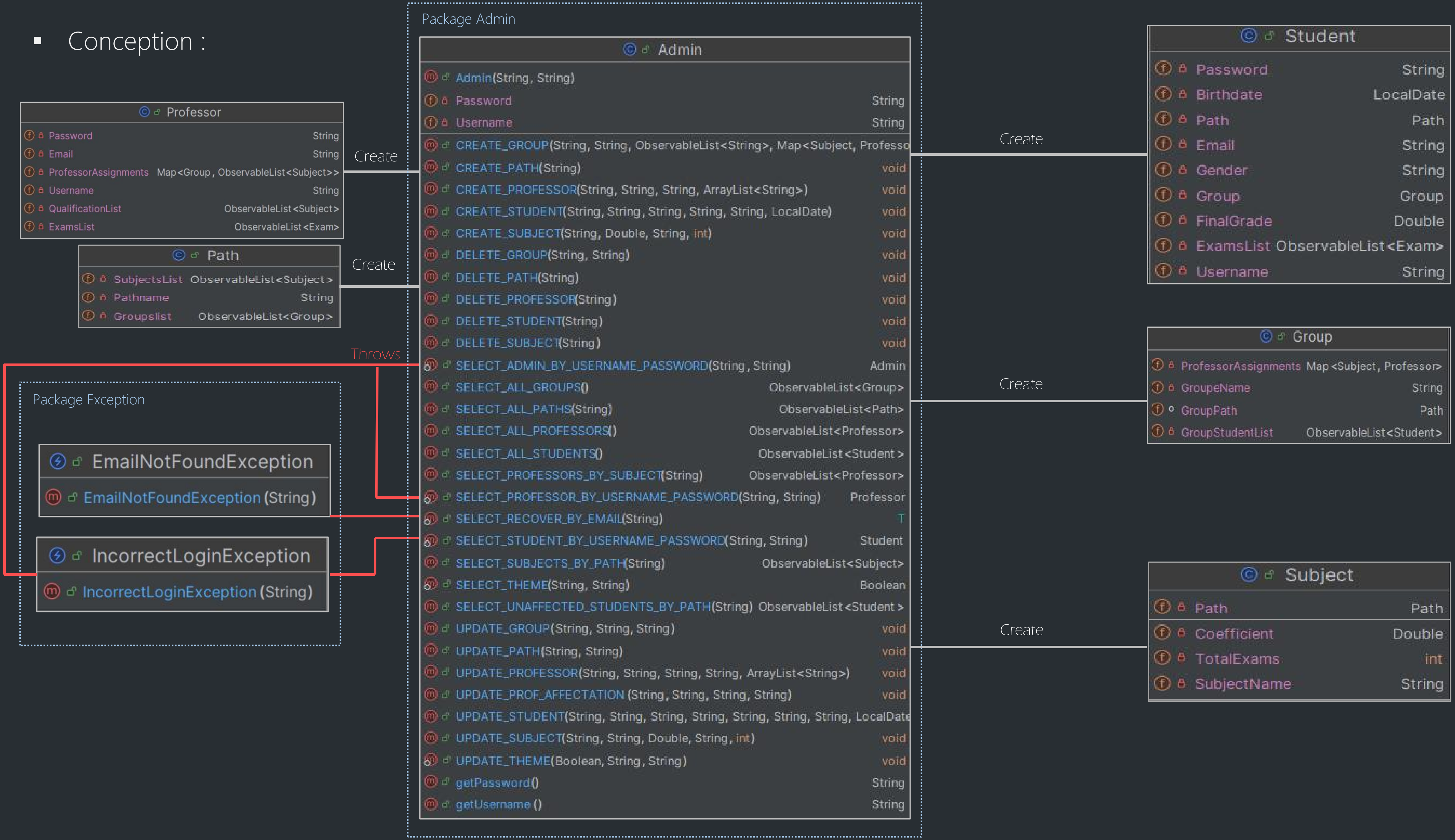
MEF-13 As a professor, I can view a detailed list of exam results for each student, organized by group and subject, and have the authority to update exam scores if necessary. 3days • 1	PROFESSOR PANEL	TO DO	-	ME
MEF-15 As a student, I can view and download my final results for each subject at the end of the year, providing a summary of my academic performance. 5days • 2	STUDENT PANEL	TO DO	-	ME
MEF-16 As an analyst, I can use the data to create charts and graphs showing success rates by group, path, and gender for clearer performance insights. 5days • 2		TO DO	-	ME

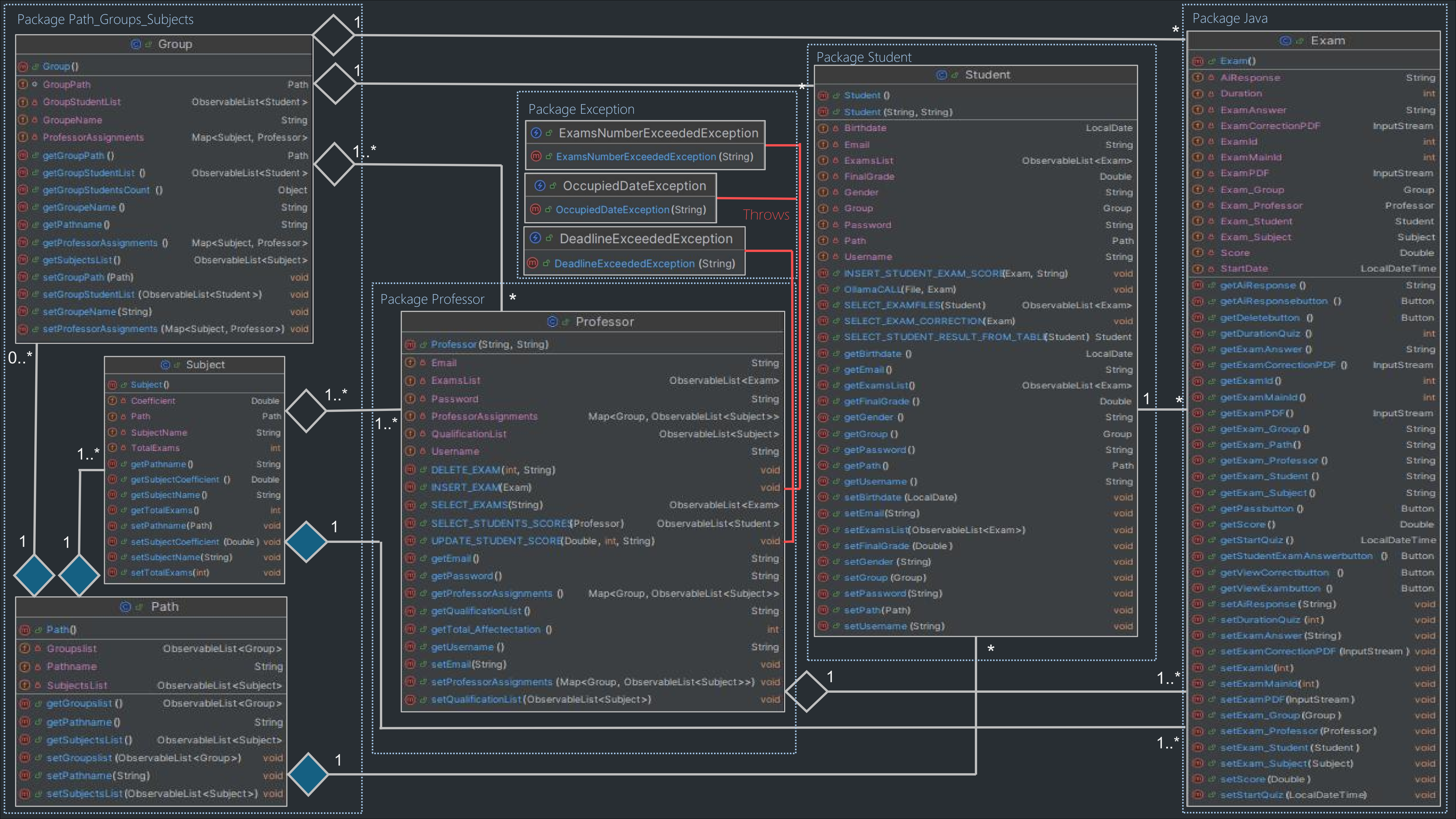

Edit sprint: MEF Sprint 3

Required fields are marked with an asterisk *

Sprint name *	MEF Sprint 3
Duration	2 weeks
Start date	8/5/2024 12:00 AM
End date	8/19/2024 12:00 AM

- Conception :


Package Admin
Admin
Admin(String, String)
Password String
Username String
CREATE_GROUP(String, String, ObservableList<String>, Map<Subject, Professo
CREATE_PATH(String) void
CREATE_PROFESSOR(String, String, String, ArrayList<String>) void
CREATE_STUDENT(String, String, String, String, String, LocalDate) void
CREATE_SUBJECT(String, Double, String, int) void
DELETE_GROUP(String, String) void
DELETE_PATH(String) void
DELETE_PROFESSOR(String) void
DELETE_STUDENT(String) void
DELETE_SUBJECT(String) void
SELECT_ADMIN_BY_USERNAME_PASSWORD(String, String) Admin
SELECT_ALL_GROUPS() ObservableList<Group>
SELECT_ALL_PATHS(String) ObservableList<Path>
SELECT_ALL_PROFESSORS() ObservableList<Professor>
SELECT_ALL_STUDENTS() ObservableList<Student>
SELECT_PROFESSORS_BY_SUBJECT(String) ObservableList<Professor>
SELECT_PROFESSOR_BY_USERNAME_PASSWORD(String, String) Professor
SELECT_RECOVER_BY_EMAIL(String) T
SELECT_STUDENT_BY_USERNAME_PASSWORD(String, String) Student
SELECT_SUBJECTS_BY_PATH(String) ObservableList<Subject>
SELECT_THEME(String, String) Boolean
SELECT_UNAFFECTED_STUDENTS_BY_PATH(String) ObservableList<Student>
UPDATE_GROUP(String, String, String) void
UPDATE_PATH(String, String) void
UPDATE_PROFESSOR(String, String, String, String, ArrayList<String>) void
UPDATE_PROF_AFFECTATION (String, String, String, String) void
UPDATE_STUDENT(String, String, String, String, String, String, String, LocalDate
UPDATE_SUBJECT(String, String, Double, String, int) void
UPDATE_THEME(Boolean, String, String) void
getPassword() String
getUsername () String
Professor
Password String
Email String
ProfessorAssignments Map<Group, ObservableList<Subject>>
Username String
QualificationList ObservableList<Subject>
ExamsList ObservableList<Exam>
Create
Path
SubjectsList ObservableList<Subject>
Pathname String
Groupslist ObservableList<Group>
Create
Throws
Package Exception
EmailNotFoundException
EmailNotFoundException (String)
IncorrectLoginException
IncorrectLoginException (String)
Student
Password String
Birthdate LocalDate
Path Path
Email String
Gender String
Group Group
FinalGrade Double
ExamsList ObservableList<Exam>
Username String
Create
Group
ProfessorAssignments Map<Subject, Professor>
GroupeName String
GroupPath Path
GroupStudentList ObservableList<Student>
Create
Subject
Path Path
Coefficient Double
TotalExams int
SubjectName String
Create


Package Path_Groups_Subjects
Group
Group()
GroupPath Path
GroupStudentList ObservableList<Student>
GroupeName String
ProfessorAssignments Map<Subject, Professor>
getGroupPath() Path
getGroupStudentList() ObservableList<Student>
getGroupStudentsCount() Object
getGroupeName() String
getPathname() String
getProfessorAssignments() Map<Subject, Professor>
getSubjectsList() ObservableList<Subject>
setGroupPath(Path) void
setGroupStudentList(ObservableList<Student>) void
setGroupeName(String) void
setProfessorAssignments(Map<Subject, Professor>) void
Subject
Subject()
Coefficient Double
Path Path
SubjectName String
TotalExams int
getPathname() String
getSubjectCoefficient() Double
getSubjectName() String
getTotalExams() int
setPathname(Path) void
setSubjectCoefficient(Double) void
setSubjectName(String) void
setTotalExams(int) void
Path
Path()
Groupslist ObservableList<Group>
Pathname String
SubjectsList ObservableList<Subject>
getGroupslist() ObservableList<Group>
getPathname() String
getSubjectsList() ObservableList<Subject>
setGroupslist(ObservableList<Group>) void
setPathname(String) void
setSubjectsList(ObservableList<Subject>) void
Package Exception
ExamsNumberExceededException
ExamsNumberExceededException(String)
OccupiedDateException
OccupiedDateException(String)
DeadlineExceededException
DeadlineExceededException(String)
Throws
Package Professor
Professor
Professor(String, String)
Email String
ExamsList ObservableList<Exam>
Password String
ProfessorAssignments Map<Group, ObservableList<Subject>>
QualificationList ObservableList<Subject>
Username String
DELETE_EXAM(int, String) void
INSERT_EXAM(Exam) void
SELECT_EXAMS(String) ObservableList<Exam>
SELECT_STUDENTS_SCORES(Professor) ObservableList<Student>
UPDATE_STUDENT_SCORE(Double, int, String) void
getEmail() String
getPassword() String
getProfessorAssignments() Map<Group, ObservableList<Subject>>
getQualificationList() String
getTotal_Affectection() int
getUsername() String
setEmail(String) void
setProfessorAssignments(Map<Group, ObservableList<Subject>>) void
setQualificationList(ObservableList<Subject>) void
Package Student
Student
Student()
Student(String, String)
Birthdate LocalDate
Email String
ExamsList ObservableList<Exam>
FinalGrade Double
Gender String
Group Group
Password String
Path Path
Username String
INSERT_STUDENT_EXAM_SCORE(Exam, String) void
OllamaCALL(File, Exam) void
SELECT_EXAMFILES(Student) ObservableList<Exam>
SELECT_EXAM_CORRECTION(Exam) void
SELECT_STUDENT_RESULT_FROM_TABLE(Student) Student
getBirthdate() LocalDate
getEmail() String
getExamsList() ObservableList<Exam>
getFinalGrade() Double
getGender() String
getGroup() Group
getPassword() String
getPath() Path
getUsername() String
setBirthdate(LocalDate) void
setEmail(String) void
setExamsList(ObservableList<Exam>) void
setFinalGrade(Double) void
setGender(String) void
setGroup(Group) void
setPassword(String) void
setPath(Path) void
setUsername(String) void
Package Java
Exam
Exam()
AiResponse String
Duration int
ExamAnswer String
ExamCorrectionPDF InputStream
ExamId int
ExamMainId int
ExamPDF InputStream
Exam_Group Group
Exam_Professor Professor
Exam_Student Student
Exam_Subject Subject
Score Double
StartDate LocalDateTime
getAiResponse() String
getAiResponsebutton() Button
getDeletebutton() Button
getDurationQuiz() int
getExamAnswer() String
getExamCorrectionPDF() InputStream
getExamId() int
getExamMainId() int
getExamPDF() InputStream
getExam_Group() String
getExam_Path() String
getExam_Professor() String
getExam_Student() String
getExam_Subject() String
getPassbutton() Button
getScore() Double
getStartQuiz() LocalDateTime
getStudentExamAnswerbutton() Button
getViewCorrectbutton() Button
getViewExambutton() Button
setAiResponse(String) void
setDurationQuiz(int) void
setExamAnswer(String) void
setExamCorrectionPDF(InputStream) void
setExamId(int) void
setExamMainId(int) void
setExamPDF(InputStream) void
setExam_Group(Group) void
setExam_Professor(Professor) void
setExam_Student(Student) void
setExam_Subject(Subject) void
setScore(Double) void
setStartQuiz(LocalDateTime) void
1
*
1
*
1..*
*
0..*
1..*
1..*
1..*
1
1
1
1
*
1
1..*
1..*
1

Database Tables Definitions:

Genders_table : Store Gender Options.
Gender_name (Primary Key) : VARCHAR

Themes_table : Store Theme Options.
ID (Part of Composite Key) : Tinyint
Theme_name (Part of Composite Key) : VARCHAR

Essect_paths : Store Information About Paths.
Path_name (Primary Key) : VARCHAR

Essect_groups Store Information About Groups.
Group_id (Part of Composite Key) : INT AUTO_INCREMENT
Group_name (Part of Composite Key) : VARCHAR
Path_name (Part of Composite Key) (Foreign Key Referencing Essect_Paths(path_name)) : VARCHAR

Essect_subjects : Store Information About Subjects.
Subject_name (Primary Key) : VARCHAR
Subject_coefficient : DOUBLE
Path_name (Foreign Key Referencing Essect_Paths(path_name)) : VARCHAR
Totalexams : INT

Essect_admins : Store Information Related To Administrators.
Admin_username (Primary Key) : VARCHAR
Admin_password : VARCHAR
Theme (Foreign Key Referencing Themes_table(id)) : TINYINT
.
Essect_students : Store Student Details.
Student_username (Primary Key) : VARCHAR
Student_password: VARCHAR
Student_email (Unique) : VARCHAR
Student_path(Foreign Key Referencing Essect_Paths(path_name)) : VARCHAR
Student_group (Foreign Key Referencing Essect_Groups(group_name)) : VARCHAR
Group_id (Foreign Key Referencing Essect_Groups(group_id)) : INT
Theme (Foreign Key Referencing Themes_table(id)) : TINYINT
Gender (Foreign Key Referencing Genders_table(gender_name)) : VARCHAR
BirthDate : DATE
FinalGrade : DOUBLE NULL

Essect_professors : Store Professor Details.
Professor_username (Primary Key) : VARCHAR
Professor_password : VARCHAR
Professor_email: (Unique) : VARCHAR
Theme (Foreign Key Referencing Themes_table(ID)) : TINYINT

Essect_professors_subjects : Store The Subjects Mastered By Each Professor.
ID (Primary Key) : INT AUTO_INCREMENT
Professor_name (Foreign Key Referencing Essect_professors(professor_username)) : VARCHAR
Subject_name (Foreign Key Referencing Essect_Subjects(subject_name)) : VARCHAR

Essect_professors_affectation: Store Professor Assignments To Subjects And Groups.
ID (Primary Key) : INT AUTO_INCREMENT
Professor_name (Foreign Key Referencing Essect_Professors(professor_username)) : VARCHAR
Subject_name (Foreign Key Referencing Essect_Subjects(subject_name)) : VARCHAR
Group_name (Foreign Key Referencing Essect_Groups(group_name)) : VARCHAR
Path_name (Foreign Key Referencing Essect_Paths(path_name)) : VARCHAR

Essect_exam : Store Exam Details.
Main_id (Primary Key) : INT AUTO_INCREMENT
ID : INT
Exam_file : LONGBLOB
Exam_correction : LONGBLOB
Path_name (Foreign Key Referencing Groups(path_name)) : VARCHAR
Group_name (Foreign Key Referencing Groups(group_name)) : VARCHAR
Subject_name (Foreign Key Referencing Subjects(subject_name)) : VARCHAR
Start_date : DATETIME NULL
Duration : INT NULL
Professor_username (Foreign Key Referencing essect_professors_affectation(professor_name)) : VARCHAR
Year : YEAR

Essect_exam_results : Store Exam Results For Each Student.
ID (Primary Key) : INT
IDExam (Foreign Key Referencing Essect_exam(Main_id)) : INT
Student_username (Foreign Key Referencing Students(student_username)) : VARCHAR
Student_answer : MEDIUMTEXT
Score : DOUBLE
Ai_response : MEDIUMTEXT

- ## Datawarehouse Fact Table Definition:

Fact Name : Dwh_essect_in_numbers :
Fact Objective : Track and analyze student success metrics based on gender, group, and path.

Dimension: genders_table
Attributes:
Gender_name (Primary Key)

Dimension: essect_groups
Attributes:
Group_Name (Part of Composite Key)
Path_Name (Part of Composite Key)

Fact Table Schema:

#Gender_name (Foreign Key referencing genders_table(Gender_name)) : VARCHAR
#Group_Name (Foreign Key referencing essect_groups(Group_name) : VARCHAR
#Path_Name (Foreign Key referencing essect_groups(Group_path)) : VARCHAR
Students_count (Measure: Number of students by group)
Success_rate (Measure: Percentage of students with a passing grade by group)
Year

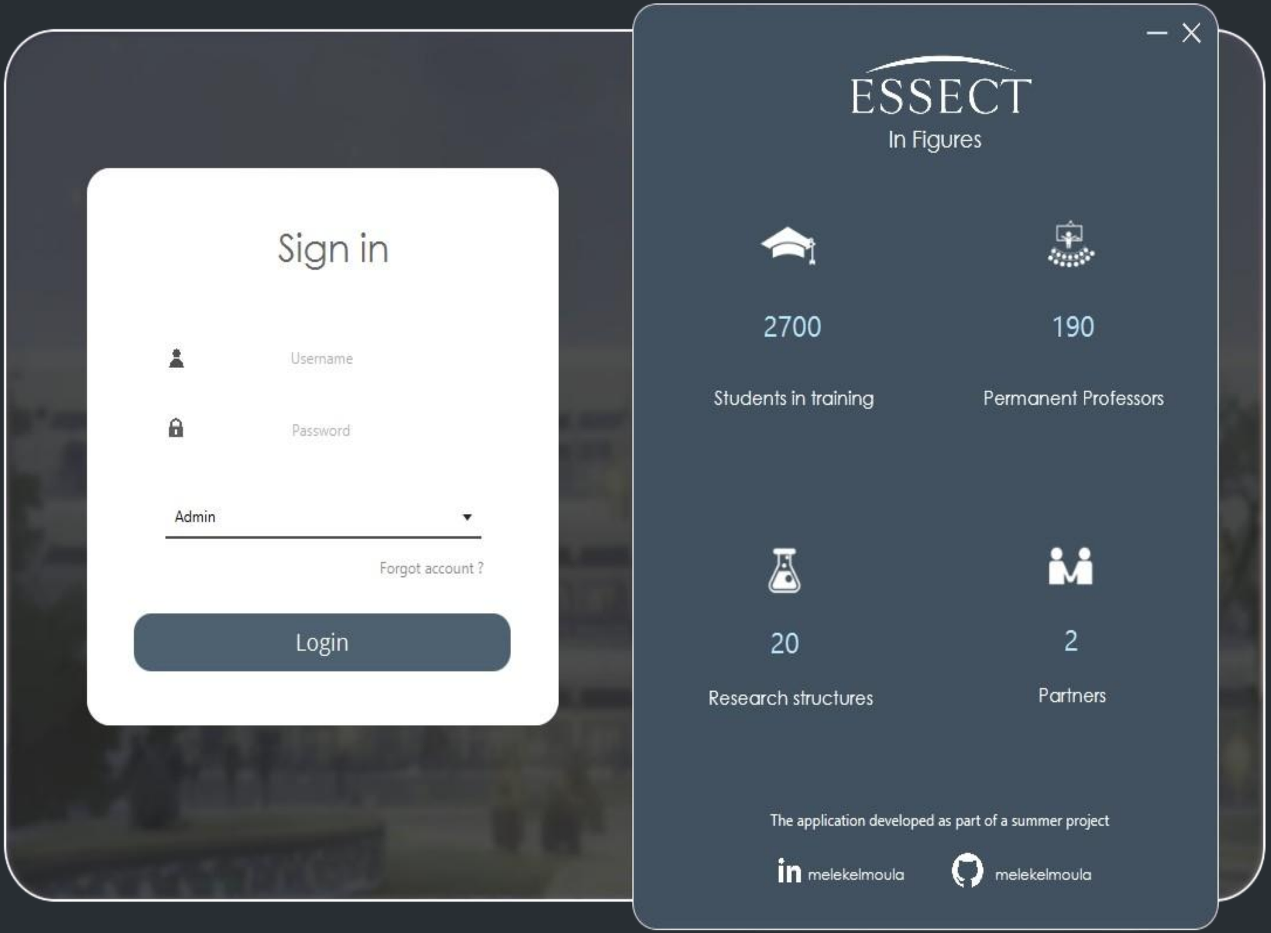
Focus:
Axe: Gender, Group, Path
Measures: Number of students, Success rate, Year

- ## Events Management :

```
DELIMITER $$

CREATE DEFINER=`root`@`localhost`
EVENT `INSERT_MISSING_STUDENTS_EXAMS`
ON SCHEDULE AT '2023-09-13 07:05:48'
ON COMPLETION PRESERVE
DISABLE
DO
BEGIN
DECLARE var_exam_MainID INT;
DECLARE var_startdate DATETIME;
DECLARE var_duration INT;
DECLARE var_groupname VARCHAR(255);
DECLARE var_pathname VARCHAR(255);
DECLARE var_subjectname VARCHAR(255);
DECLARE var_endtime DATETIME;
DECLARE done INT DEFAULT FALSE;

DECLARE exam_cursor CURSOR FOR
SELECT MainID, Startdate, Duration, Group_Name, Path_Name, Subject_Name
FROM essect_exam
WHERE EXTRACT(YEAR FROM Startdate) = EXTRACT(YEAR FROM CURRENT_DATE);
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN exam_cursor;

exam_loop: LOOP
FETCH exam_cursor INTO var_exam_MainID, var_startdate, var_duration, var_groupname, var_pathname, var_subjectname;
IF done THEN
LEAVE exam_loop;
END IF;

SET var_endtime = DATE_ADD(var_startdate, INTERVAL var_duration MINUTE);
IF CURRENT_TIMESTAMP > var_endtime THEN
INSERT INTO essect_exam_results (IDExam, Student_answer, Score, AI_response, Student_Username)
SELECT var_exam_MainID, 'Absent', 0, 'Absent', s.Student_Username
FROM essect_students s
WHERE s.Student_Username NOT IN (
SELECT eer.Student_Username
FROM essect_exam_results eer
WHERE eer.IDExam = var_exam_MainID )
AND s.Student_Path = var_pathname
AND s.Student_Group = var_groupname;
END IF;
END LOOP;

CLOSE exam_cursor;
END $$
DELIMITER ;
```

```sql
DELIMITER //
CREATE DEFINER=`root`@`localhost` EVENT `UPDATE_STUDENTS_GRADE_INSERT_DWH` ON SCHEDULE
EVERY 1 YEAR STARTS '2024-07-01 06:33:32' ON COMPLETION NOT PRESERVE ENABLE DO BEGIN
-- Drop temporary table if it exists
DROP TEMPORARY TABLE IF EXISTS TempStudentScores;

-- Create a temporary table to store computed total scores and total coefficients
CREATE TEMPORARY TABLE TempStudentScores AS
SELECT
er.Student_Username,
s.Subject_Name,
Sum(er.Score) * s.Subject_Coefficient AS Avg_Score_Per_Subject,
s.Subject_Coefficient AS Total_Coefficient
FROM
essect_exam_results er
JOIN
essect_exam e ON er.IDExam = e.MainID
JOIN
essect_subjects s ON e.Subject_Name = s.Subject_Name
GROUP BY
er.Student_Username, s.Subject_Name;

-- Drop another temporary table if it exists
DROP TEMPORARY TABLE IF EXISTS TempStudentAggregatedScores;

-- Create another temporary table to sum the average scores and coefficients across subjects
CREATE TEMPORARY TABLE TempStudentAggregatedScores AS
SELECT
Student_Username,
SUM(Avg_Score_Per_Subject) AS Total_Score,
SUM(Total_Coefficient) AS Total_Coefficient
FROM
TempStudentScores
GROUP BY
Student_Username;

 -- Update FinalGrade using the aggregated scores table
UPDATE essect_students st
JOIN TempStudentAggregatedScores tss ON st.Student_Username = tss.Student_Username
SET st.FinalGrade = ROUND(
IF(tss.Total_Coefficient > 0, tss.Total_Score / tss.Total_Coefficient, 2));

-- Insert data into data warehouse for student statistics
INSERT INTO dwh_essect_in_numbers (Gender_Name, Group_Name, Students_count, Path_Name, Success_rate,
Year)

SELECT
Gender AS Gender_Name,
Student_Group AS Group_Name,
COUNT(*) AS Students_count,
Student_Path,
 (SUM(CASE WHEN finalgrade >= 10 THEN 1 ELSE 0 END) / COUNT(*)) * 100 AS Success_rate,
YEAR(CURDATE()) AS Year

FROM
essect_students
GROUP BY
Gender,
Student_Group,
Student_Path
ON DUPLICATE KEY UPDATE
Students_count = VALUES(Students_count),
Success_rate = VALUES(Success_rate),
Year = VALUES(Year);

-- Clean up temporary tables
DROP TEMPORARY TABLE IF EXISTS TempStudentScores;
DROP TEMPORARY TABLE IF EXISTS TempStudentAggregatedScores;
END

DELIMITER ;
```

- Design :



Sign in

Username

Password

Admin

Forgot account ?

Login

ESSECT

In Figures

2700

Students in training

190

Permanent Professors

20

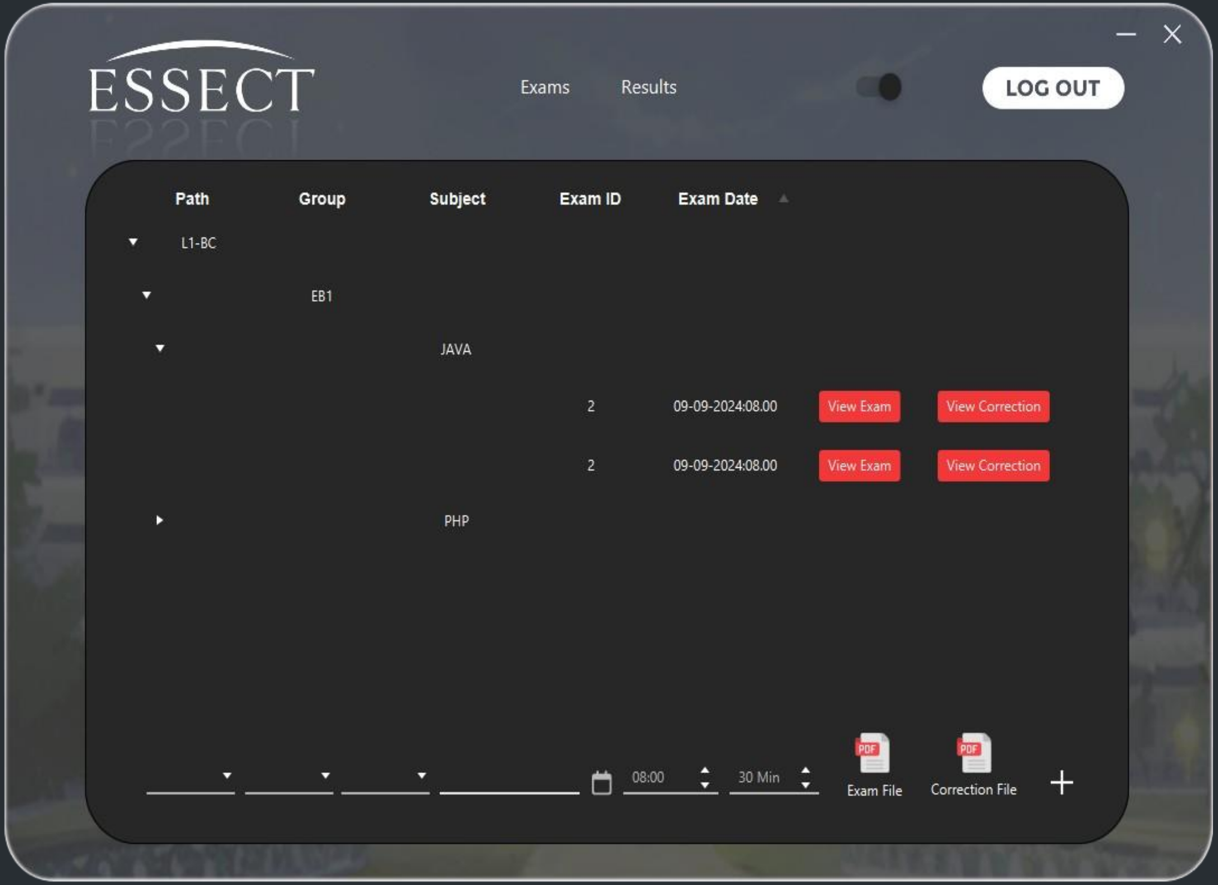
Research structures

2

Partners

The application developed as part of a summer project

melekelmoula melekelmoula



ESSECT

Exams Results LOG OUT

Path	Group	Subject	Exam ID	Exam Date		
▾ L1-BC						
	▾ EB1					
		▾ JAVA				
			2	09-09-2024:08.00	View Exam	View Correction
			2	09-09-2024:08.00	View Exam	View Correction
		▸ PHP				

08:00 30 Min Exam File Correction File +

As i wrap up this JavaFX project, the focus has been on critically assessing and improving the application's functionality and user experience. This project is intended as a foundation for ongoing enhancement, and I welcome any constructive feedback to further refine and elevate its quality. Your insights and suggestions are greatly appreciated as I strive to make meaningful advancements.

Thank you for your attention and support throughout this journey.