جامعة الملك سعود
King Saud University

كلية علوم الحاسب والمعلومات
قسم هندسة البرمجيات

SMARTSCHEDULE

SWE 481 Advanced Web Engineering Project

Section:79617

Group number:3

Student name	ID
Renad Alotaibi	443200801
Rahaf samkari	443202844
Raghad Alqahtani	443200549
Ruba Alrabiah	443200453

Table of Contents

Table of Figure

Architecture choice

Model-View-Controller (MVC) is one of the most popular architectures for web applications because it naturally separates the user interface (View) from the business logic (Model) and input handling (Controller) [1].

This separation offers several advantages:

- It makes your code more organized and maintainable.
- Multiple developers can work on different components simultaneously without conflicts.
- **Testability**: With its modular structure, MVC simplifies unit testing. Developers can test each component independently, ensuring higher code quality.

Frameworks choice and justification

For our project, we have chosen the following frameworks:

- **Frontend: React.js**
- **Backend: Express.js** (running on Node.js)
- **Project Workspace: React with Vite**

Justification:

1. **Support for MVC Architecture**

Both React.js and Express.js naturally support the **Model-View-Controller (MVC)** design pattern as shown in **Figure 1**.

React.js (View): Efficiently handles the user interface and dynamic updates.

Express.js (Controller/Model): Manages server-side routing, business logic, and interactions with the database.
This separation aligns perfectly with MVC principles, promoting maintainable and modular code.

2. **Familiarity and Efficiency**

Our team has prior experience with React/Express applications from our graduation project currently. This familiarity makes these frameworks ideal for our current project, ensuring faster development, fewer errors, and better collaboration.

3. **Modern and Lightweight Development**

Vite provides a fast and modern workspace for React, improving build times and developer experience. Node.js with Express allows a unified JavaScript environment for both frontend and backend, reducing context switching and streamlining development.

4. **Community and Ecosystem Support**
   Both React.js and Express.js have large communities, extensive documentation, and numerous libraries/plugins, which facilitates problem-solving, integration, and future scalability.

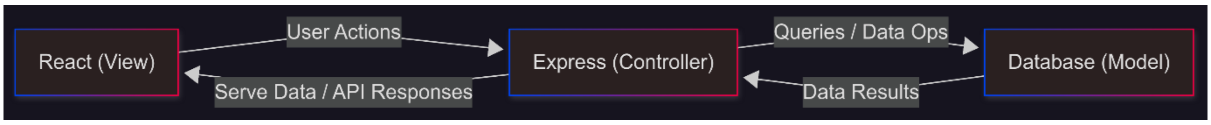

Figure 1:MVC represented using the frameworks chosen

Database

For the database, we chose PostgreSQL hosted via Supabase cloud services, as the project requires a relational SQL database.

ER diagram

For a clear view click the bored link:https://miro.com/app/board/uXjVJL1nLtI=/?share_link_id=209675618715

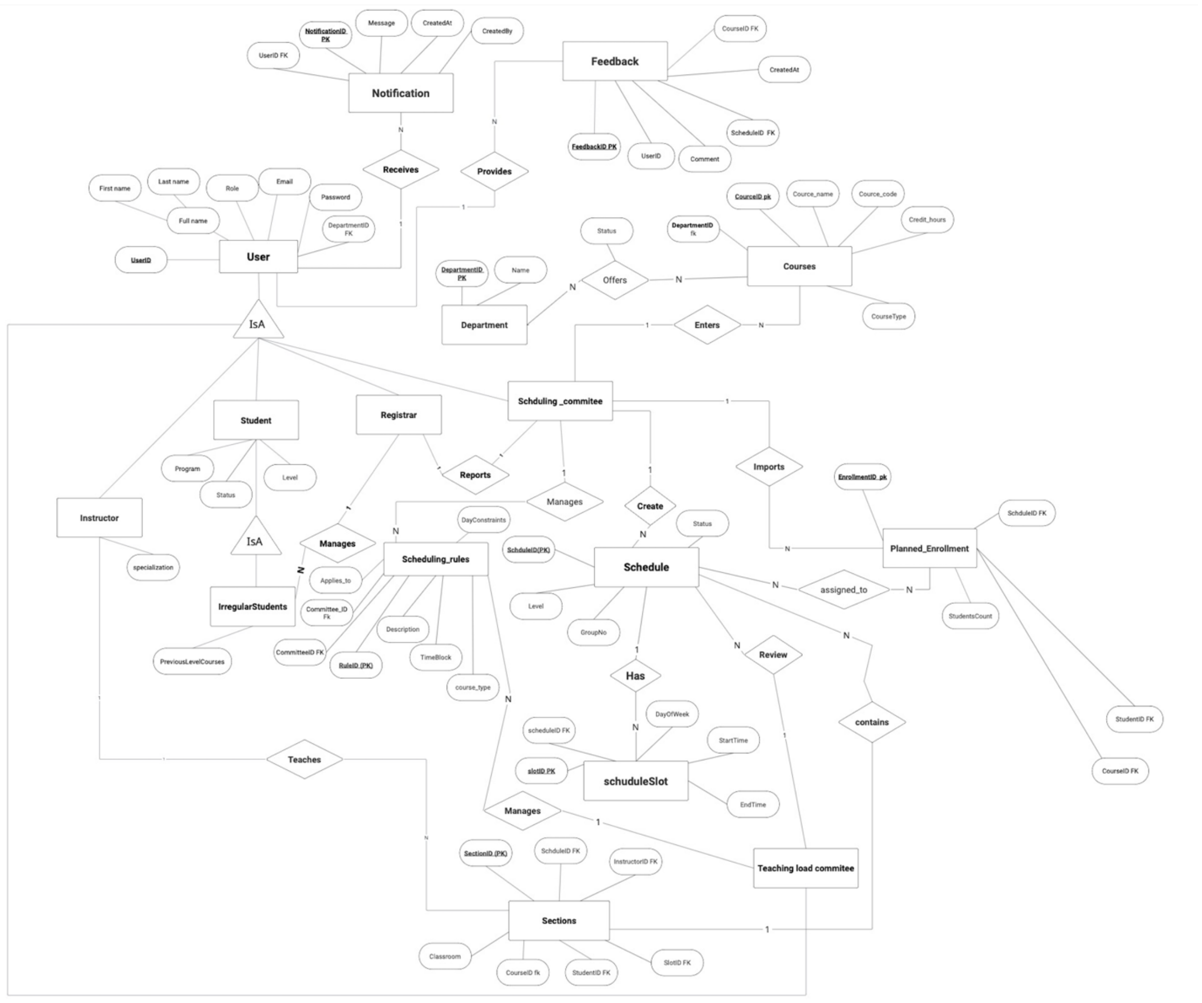

Figure 2: Entity–Relationship (ER) Diagram

Important Notes about entities/attributes for reader understanding:

- **Role** in User means: the type of user (student, faculty member etc) so we can grant access based on that
- applies_to in **Scheduling Rules** (optional, e.g., “all schedules” or a specific level/irregular students) as shown in Table 1.
- In **Courses** course type weather its core, elective as electives come from certain departments registers and core from the scheduling committee.
- **Final note** the id for user is inherited but we mentioned id specific user when we need to use two different types of user id by name such as instructor_Id ,student_Id in section entity just to clarify the design.

Description [for users understanding]	**day_constraints**	**time_block**	**applies_to_level**
Reserve lunch	["Mon","Wed"]	12:00-13:00	all
Electives for multiple levels	-	-	all
2-hour labs continuous	-	14:00-16:00	all

Table 1:rules example

Unclear points in the DB of the system:

points to consider: the scheduling committee can follow timelines of the university scheduling deadlines when they receive it via email, so we won’t consider it as a feature or in our data base.

Also, the enrolment data we assume scheduler follow SWE suggested plan for each level and schedules courses etc, student can view after signing in, so we don’t think we need enrolment data however we still have a “planned _ enrolment” in our data base. Although, we want to go with our suggested approach.

Database schema

```
create table if not exists "Departments" (
  "DepartmentID" int generated by default as identity primary key,
  "Name"         text not null unique
);
create table if not exists "User" (
"UserID" int generated by default as identity primary key,
  "First_name"    text not null,
  "Last_name"     text not null,

  "Full_name"     text generated always as (
                   concat_ws(' ', "First_name", "Last_name")
                 ) stored,
  "Email"         text unique not null,
  "Password"      text not null,
  "DepartmentID"  int references "Departments"("DepartmentID") on delete set null,
  "Role"          text not null check ("Role" in ('student','faculty','registrar'))
);

create table if not exists "Registrars" (
"RegistrarID" int primary key
  references "User"("UserID") on delete cascade
);

create table if not exists "Instructor" (
  "InstructorID" int primary key
  references "User"("UserID") on delete cascade,
  "specialization" text
);

create table if not exists "Students" (
  "StudentID" int primary key
  references "User"("UserID") on delete cascade,
  "program"   text[] not null,
  "level"     int not null
);

create table if not exists "IrregularStudents" (
"StudentID" int primary key
    references "Students"("StudentID”) on delete cascade,
  "PreviousLevelCourses" text[]
);

create index if not exists "idx_User_DepartmentID" on "User"("DepartmentID");
create index if not exists "idx_User_Role"         on "User"("Role");

create table if not exists "Notifications" (
  "NotificationID" int generated by default as identity primary key,
  "Message"        text not null,
  "CreatedAt"      timestamptz not null default now(),

  "CreatedBy"      int not null
    references "User"("UserID") on delete cascade,

  "UserID"         int
    references "User"("UserID") on delete cascade
);
create table if not exists "Feedback" (
  "FeedbackID" int generated by default as identity primary key,
  "Comment"    text,
  "ScheduleID" int references "Schedule"("ScheduleID"),
  "CourseID"   int references "Courses"("CourseID"),

  "UserID"     int references "User"("UserID"),
  "CreatedAt"  timestamp not null default now()
);

create index if not exists "idx_Notifications_UserID"    on "Notifications"("UserID");
create index if not exists "idx_Notifications_CreatedBy" on "Notifications"("CreatedBy");

create index if not exists "idx_Feedback_UserID"     on "Feedback"("UserID");
create index if not exists "idx_Feedback_ScheduleID" on "Feedback"("ScheduleID");
create index if not exists "idx_Feedback_CourseID"   on "Feedback"("CourseID");
create or replace function notify_feedback()
returns trigger
language plpgsql
as $$
declare
  v_msg text;
begin
  v_msg :=
    'New feedback added (FeedbackID ' || NEW."FeedbackID" || ')'
    || case when NEW."CourseID"   is not null then ' on Course '   || NEW."CourseID"   else '' end
    || case when NEW."ScheduleID" is not null then ' on Schedule ' || NEW."ScheduleID" else '' end
    || ' by User ' || NEW."UserID";

  insert into "Notifications"("Message", "CreatedAt", "CreatedBy", "UserID")
  values (v_msg, now(), NEW."UserID", NEW."UserID");

  return NEW;
end;
$$;

/* ========== Trigger Binding ========== */
drop trigger if exists "trg_notify_Feedback_insert" on "Feedback";

create trigger "trg_notify_Feedback_insert"
```

```
after insert on "Feedback"
for each row
execute function notify_feedback();

CREATE TABLE TeachingLoadCommittee (
"CommitteeID" int primary key
  references "User"("UserID") on delete cascade);

create type day_of_week as enum (
  'Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'
);
create table SchuduleSlot (
  "SlotID "    int generated by default as identity primary key,
  "SchuduleID"  int references Schedule(SchuduleID) on delete cascade,
  "DayOfWeek"   day_of_week not null,
  "StartTime"  time not null,
  "EndTime"     time not null,
  constraint CHK_TimeValid check ("StartTime" < "EndTime")
);
create table Sections (
  "SectionID"     int generated by default as identity primary key,
  "SchuduleID"   int references Schedule("SchuduleID") on delete set null,
  "CourseID"      int not null references Courses("CourseID") on delete cascade,
  "InstructorID"  int references Instructor("InstructorID") on delete set null,
  "StudentID "    int references Student("StudentID") on delete set null,
  "SlotID "       int references SchuduleSlot("SlotID") on delete set null,
 " Classroom"        text ,
  constraint UQ_Sec_Schudule_Course unique ("SchuduleID", "CourseID")
);
create table PlannedEnrollment (
 " EnrollmentID"   int generated by default as identity primary key,
 " SchuduleID"     int not null references Schedule("SchuduleID") on delete cascade,
  "StudentID"     int not null references Student("StudentID") on delete cascade,
  "CourseID"      int not null references Courses("CourseID") on delete cascade,
  StudentsCount  int not null default 0,
  constraint UQ_Planned unique ("StudentID", "ScheduleID", "CourseID"),
  constraint CHK_StudentsCount check (StudentsCount >= 0)
);
create table if not exists "courses" (
    "CourseID" int generated by default as identity primary key,
    "course_name" varchar(255) not null,
    "course_code" varchar(50) unique not null,
    "credit_hours" int not null,
    "course_type" varchar(100),
    "DepartmentID" int not null references "Departments"("DepartmentID")
"scheduling_committee_id" int not null references "scheduling_committee"
(" scheduling_committee_id")
);
create table if not exists "schedule" (
    "ScheduleID" int generated by default as identity primary key,
    "level" int,
    "GroupNo" int,
    "status" varchar(50)
" scheduling_committee_id" int references "  Schduling _commitee "("scheduling_committee_id")
  " SectionID " int not null references " Sections "(" SectionID ") on delete cascade
);


create table if not exists "schedule_rules" (
    "rule_id" int generated by default as identity primary key,
    "description" text,
    "applies_to" varchar(100),
    "timeBlock" varchar(100),
    "dayConstraints" text
  "course_type"    text
"committee_id" int references "Schduling _commitee"("scheduling_committee_id")
" CommitteeID " int references "TeachingLoadCommittee "("CommitteeID)"
```

```
);
create table if not exists "SchedulingCommittee" (
    "schedulingcommittee_id" int primary key
        references "users"("user_id") on delete cascade
);
create table if not exists "AssignTo" (
  "ScheduleID"   int not null
    references "Schedule"("ScheduleID") on delete cascade,
  "EnrollmentID" int not null
    references "Planned_Enrollment"("EnrollmentID") on delete cascade,
  primary key ("ScheduleID","EnrollmentID")
);
create table if not exists "Offers" (
  "DepartmentID" int not null
    references "Department"("DepartmentID") on delete cascade,
  "CourseID" int not null
    references "Courses"("CourseID") on delete cascade,
  "Status" text,
  primary key ("DepartmentID","CourseID")
);
```

synthetic dataset

Schedule_Rules Table

Insert into schedule_rules (description, applies_to, "timeBlock", "dayConstraints", committee_id, "CommitteeID", course_type) values

('[MANDATORY] Daily lunch break reserved', 'ALL', '12:00-13:00', 'Sun-Thu', null, null, 'ANY'),

('[MANDATORY] Midterm exam window (Mon) 12:00–14:00', 'ALL', '12:00-14:00', 'Mon', null, null, 'ANY'),

('[MANDATORY] Midterm exam window (Wed) 12:00–14:00', 'ALL', '12:00-14:00', 'Wed', null, null, 'ANY'),

('[MANDATORY] Lab 08:00-10:00 (continuous 2h)', 'ALL', '08:00-10:00', 'Sun-Thu', null, null, 'LAB'),

('[MANDATORY] Lab 10:00-12:00 (continuous 2h)', 'ALL', '10:00-12:00', 'Sun-Thu', null, null, 'LAB'),

('[MANDATORY] Lab 13:00-15:00 (continuous 2h)', 'ALL', '13:00-15:00', 'Sun-Thu', null, null, 'LAB');

('[PREF] Elective preferred 08:00–09:00', 'ALL', '08:00-09:00', 'Sun-Thu', null, null, 'ELECTIVE'),

('[PREF] Elective preferred 09:00–10:00', 'ALL', '09:00-10:00', 'Sun-Thu', null, null, 'ELECTIVE'),

('[PREF] Elective preferred 13:00–14:00', 'ALL', '13:00-14:00', 'Sun-Thu', null, null, 'ELECTIVE'),

('[PREF] Elective preferred 14:00–15:00', 'ALL', '14:00-15:00', 'Sun-Thu', null, null, 'ELECTIVE'),

('[LIMIT] Maximum 3 courses per day', 'ALL', null, 'Sun-Thu', null, null, 'ANY'),

('[LIMIT] Minimum 30-minute gap between classes', 'ALL', null, 'Sun-Thu', null, null, 'ANY');

Department table

```
INSERT INTO Departments (Name) VALUES

("Mathematics"),

("Physics & Astronomy"),

("Statistics and Operations Research"),

("software engineering"),

("information systems"),

("Information Technology"),

("Computer Science"),

("Islamic Studies");
```

courses Table

```
INSERT INTO Courses (DepartmentID, course_name, course_code, credit_hours,
course_type, SchedulingCommitteeID) VALUES

(7, 'Computer Programming (1)', 'CSC111', 4, 'Mandatory', NULL),

(2, 'General Physics (1)', 'PHYS103', 4, 'Mandatory', NULL),

(1, 'Discrete Mathematics', 'MATH151', 3, 'Mandatory', NULL),

(1, 'Integral Calculus', 'MATH106', 3, 'Mandatory', NULL),

(6, 'Computer Communications & Networks', 'CENX303', 3, 'Mandatory', NULL),

(1, 'Linear Algebra', 'MATH244', 3, 'Mandatory', NULL),

(7, 'Computer Programming (2)', 'CSC113', 4, 'Mandatory', NULL),

(2, 'General Physics (2)', 'PHYS104', 4, 'Mandatory', NULL),

(4, 'Introduction to Software Engineering', 'SWE211', 3, 'Mandatory', NULL),

(7, 'Data Structures', 'CSC212', 3, 'Mandatory', NULL),

(4, 'Software Security Engineering', 'SWE314', 3, 'Mandatory', NULL),

(7, 'Computer Organization', 'CSC220', 3, 'Mandatory', NULL),

(4, 'Software Requirements Engineering', 'SWE312', 3, 'Mandatory', NULL),

(7, 'Operating Systems', 'CSC227', 3, 'Mandatory', NULL),
```

```sql
(4, 'Software Quality Assurance', 'SWE333', 2, 'Mandatory', NULL),

(4, 'Software Design & Architecture', 'SWE321', 3, 'Mandatory', NULL),

(5, 'Introduction to Database Systems', 'IS230', 3, 'Mandatory', NULL),

(4, 'Web Application Development', 'SWE381', 3, 'Mandatory', NULL),

(8, 'Professional Ethics', 'IC107', 2, 'Mandatory', NULL),

(4, 'Software Construction Laboratory', 'SWE444', 2, 'Mandatory', NULL),

(4, 'Human-Computer Interaction', 'SWE482', 3, 'Mandatory', NULL),

(4, 'Software Testing and Validation', 'SWE434', 3, 'Mandatory', NULL),

(4, 'Graduation Project I', 'SWE496', 3, 'Mandatory', NULL),

(4, 'Practical Training', 'SWE479', 1, 'Mandatory', NULL),

(4, 'Software Engineering Code of Ethics & Professional Practice', 'SWE477', 2,
'Mandatory', NULL),

(8, 'Current Issues', 'IC108', 2, 'Mandatory', NULL),

(4, 'Graduation Project II', 'SWE497', 3, 'Mandatory', NULL),

(4, 'Software Project Management', 'SWE466', 3, 'Mandatory', NULL),

(4, 'Software Maintenance and Evolution', 'SWE455', 2, 'Mandatory', NULL);
```

References

[1] “MVC Architecture | Ramotion Branding Agency,” *Web Design, UI/UX, Branding, and App Development Blog*, May 02, 2023. https://www.ramotion.com/blog/mvc-architecture-in-web-application/