Student Tracker Database – Developer Guide

Version: POC (No Location table) | Database: Oracle XE 18c/21c or Oracle 12c+

# 1) Executive Summary

This guide explains how to stand up the local Oracle database for the Student Tracker proof-of-concept application, how the schema is designed, how to run the SQL scripts, and how the ASP.NET app connects. It is written so teammates can clone the repo and get running quickly on their own laptops.

# 2) Requirements & What You’ll Install

• Oracle Database XE 18c/21c (preferred) or Oracle 12c+ with a pluggable database (PDB).

• Oracle SQL Developer (GUI) or SQLcl/SQL\*Plus (CLI).

• .NET 8 SDK (for the ASP.NET web app), plus NuGet packages: Oracle.ManagedDataAccess.Core, Oracle.EntityFrameworkCore.

• Git (to clone the repository).

# 3) Repository Layout (db/ folder)

The database assets live in the /db folder. Keep SQL files under version control so anyone can rebuild the database.

student-tracker/  
└─ db/  
 ├─ 01\_tables.sql  
 ├─ 02\_constraints.sql  
 ├─ 03\_indexes.sql  
 ├─ 04\_seed.sql  
 ├─ 05\_views.sql  
 ├─ 06\_procedures.sql  
 ├─ 07\_triggers.sql  
 ├─ 07\_security.sql -- roles, grants, app context, VPD (optional but provided)  
 ├─ 98\_drop\_app\_objects.sql -- safe reset (drops ONLY app objects)  
 └─ 99\_run\_all.sql -- orchestrates everything in order

# 4) Local Oracle Setup (one-time)

Open SQL Developer, create a connection as SYSTEM to your local XE instance:

• Hostname: localhost

• Port: 1521

• Service name: XEPDB1 (XE 18c/21c) –or– SID: XE (older XE 11g)

## 4.1 Create a dedicated schema (user)

-- Run as SYSTEM (or a DBA-capable user)  
CREATE USER STUDENT\_TRACKER IDENTIFIED BY "Strong#Password1"  
 DEFAULT TABLESPACE USERS  
 TEMPORARY TABLESPACE TEMP  
 QUOTA UNLIMITED ON USERS;  
  
GRANT CREATE SESSION TO STUDENT\_TRACKER;  
GRANT CREATE TABLE, CREATE VIEW TO STUDENT\_TRACKER;  
GRANT CREATE SEQUENCE, CREATE PROCEDURE, CREATE TRIGGER TO STUDENT\_TRACKER;  
-- Optional if using DBMS\_SESSION in security context:  
GRANT EXECUTE ON DBMS\_SESSION TO STUDENT\_TRACKER;

## 4.2 Create a developer connection for daily use

In SQL Developer:

• Connection Name: StudentTracker (local)

• Username: STUDENT\_TRACKER

• Password: Strong#Password1

• Hostname: localhost | Port: 1521

• Service name: XEPDB1 (if Test fails, try SID: XE)

# 5) Build the Database

Connect as STUDENT\_TRACKER in SQL Developer, open /db/99\_run\_all.sql, press Run Script (F5). This will:

1) Create tables (01\_tables.sql)

2) Add foreign keys and checks (02\_constraints.sql)

3) Create indexes (03\_indexes.sql)

4) Seed base data (04\_seed.sql)

5) Create views (05\_views.sql)

6) Create stored procs (06\_procedures.sql)

7) Create triggers (07\_triggers.sql)

8) (Optional) Create security roles, app context, and VPD (07\_security.sql)

## 5.1 Optional: Safe Reset

If you need to wipe only app objects and rebuild:

@98\_drop\_app\_objects.sql  
@99\_run\_all.sql

## 5.2 Quick Smoke Test

You can optionally run 10\_test.sql to seed a sample instructor/student, then perform one clock-in/out to validate the procedures and views.

-- as STUDENT\_TRACKER  
@10\_test.sql  
  
-- Verify latest sessions  
SELECT session\_id, session\_type, location\_text, clock\_in\_at, clock\_out\_at, duration\_minutes, status  
FROM time\_session  
ORDER BY session\_id DESC FETCH FIRST 5 ROWS ONLY;

# 6) Schema Design (why it’s set up this way)

Goals from requirements: track student hours (Class, Study, Clinical); instructors can see only their students; admins can see/change everything; students can view their own history.

Key tables:

• ROLE, USER\_ACCOUNT: identity & role-based access at the app level.

• STUDENT\_PROFILE, INSTRUCTOR\_PROFILE: role-specific fields kept separate to keep USER\_ACCOUNT lean.

• COURSE, SECTION, ENROLLMENT: academic scoping so an instructor’s visibility is limited to students enrolled in their sections.

• TIME\_SESSION: the core time tracking record (clock in/out). We removed the Location FK and use a free-text location\_text for simplicity in POC.

• AUDIT\_LOG: captures before/after JSON snapshots for administrative edits to sessions.

Views: v\_timesession\_full (joined reporting view), v\_student\_hours (student-friendly view), v\_instructor\_student\_map (helper mapping to scope instructor visibility).

Why no Location table? For POC speed and flexibility; users can enter a descriptive label. If you later need reporting by physical sites, add a LOCATION table and switch time\_session.location\_text → location\_id.

# 7) Business Rules (enforced in DB)

• Exactly one open session per user (enforced in sp\_clock\_in).

• Class sessions require a Section (CHECK + procedure validation).

• duration\_minutes is computed as a virtual column when clock\_out\_at is set.

• status transitions: open → closed (clock out); flagged for admin review if needed (manual).

# 8) Stored Procedures API (clock in/out)

## 8.1 sp\_clock\_in

Params: p\_user\_id, p\_session\_type ('Class'|'Study'|'Clinical'), p\_location\_txt (optional), p\_section\_id (required if Class), p\_source ('web'|'mobile'|'kiosk'|'admin\_edit'), p\_notes (optional).

Behavior: validates active user, enforces one open session, enforces Class→Section rule, inserts a new open session with server timestamp.

BEGIN  
 sp\_clock\_in(  
 p\_user\_id => :userId,  
 p\_session\_type => :type,  
 p\_location\_txt => :locationText, -- e.g., 'Library – Zone A' or NULL  
 p\_section\_id => :sectionId, -- required if type='Class'  
 p\_source => 'web',  
 p\_notes => :notes  
 );  
END;

## 8.2 sp\_clock\_out

Params: p\_user\_id. Behavior: finds the single open session for that user, sets clock\_out\_at and status='closed'.

BEGIN  
 sp\_clock\_out(p\_user\_id => :userId);  
END;

# 9) Security Model (roles, grants, and optional row-level security)

App DB roles: APP\_STUDENT, APP\_INSTRUCTOR, APP\_ADMIN. Grants are least privilege:

• Students: SELECT v\_student\_hours; EXECUTE sp\_clock\_in/out.

• Instructors: SELECT v\_timesession\_full and v\_instructor\_student\_map (optionally filtered by VPD).

• Admins: full DML on app tables + SELECT on views + EXECUTE procedures.

Optional: Application Context (ST\_CTX) + VPD policy on v\_timesession\_full to auto-filter rows:

• STUDENT → ts.user\_id = current user

• INSTRUCTOR → ts.user\_id IN students enrolled in their sections

• ADMIN → no filter

-- App sets context per request:  
BEGIN  
 pkg\_security\_ctx.set\_context(p\_user\_id => :userId, p\_role\_name => :roleName); -- roleName: 'Student'|'Instructor'|'Admin'  
END;

# 10) ASP.NET Integration (local dev)

NuGet: Oracle.ManagedDataAccess.Core, Oracle.EntityFrameworkCore.

## 10.1 Connection String (local)

// appsettings.json  
{  
 "ConnectionStrings": {  
 "AppDb": "User Id=STUDENT\_TRACKER;Password=Strong#Password1;Data Source=localhost:1521/XEPDB1;Pooling=true;"  
 }  
}

## 10.2 EF Core setup (Program.cs)

builder.Services.AddDbContext<AppDbContext>(opt =>  
 opt.UseOracle(builder.Configuration.GetConnectionString("AppDb")));

## 10.3 Setting DB context for VPD (optional)

await db.Database.ExecuteSqlInterpolatedAsync($@"  
BEGIN  
 pkg\_security\_ctx.set\_context(p\_user\_id => {userId}, p\_role\_name => {roleName});  
END;");

# 11) Typical Queries

## 11.1 Is user clocked in?

SELECT session\_id, session\_type, section\_id, clock\_in\_at  
FROM time\_session  
WHERE user\_id = :userId AND status = 'open';

## 11.2 Student history

SELECT session\_type, location\_text, section\_id, clock\_in\_at, clock\_out\_at, duration\_minutes, status  
FROM time\_session  
WHERE user\_id = :userId  
ORDER BY clock\_in\_at DESC  
FETCH FIRST 100 ROWS ONLY;

## 11.3 Instructor view (their students only)

SELECT ts.\*  
FROM time\_session ts  
WHERE ts.user\_id IN (  
 SELECT e.student\_id  
 FROM section s  
 JOIN enrollment e ON e.section\_id = s.section\_id  
 WHERE s.primary\_instructor\_id = :instructorId  
)  
ORDER BY ts.clock\_in\_at DESC;

# 12) Troubleshooting

• ORA-12514/ORA-12154: Swap Service name=XEPDB1 ↔ SID=XE and Test again.

• Listener not running: Start 'OracleOraDB…TNSListener' service or run lsnrctl status.

• Object name already exists: run 98\_drop\_app\_objects.sql to reset, then 99\_run\_all.sql.

• Procedure errors: run SHOW ERRORS after creating procs; fix typos and rerun 06\_procedures.sql.

• Time zone display: values are server timestamps; convert in app layer if needed per user profile.

# 13) Appendix – Objects Inventory

Tables: ROLE, USER\_ACCOUNT, STUDENT\_PROFILE, INSTRUCTOR\_PROFILE, COURSE, SECTION, ENROLLMENT, TIME\_SESSION, AUDIT\_LOG, PERMISSION, ROLE\_PERMISSION.

Views: V\_TIMESESSION\_FULL, V\_STUDENT\_HOURS, V\_INSTRUCTOR\_STUDENT\_MAP.

Procedures: SP\_CLOCK\_IN, SP\_CLOCK\_OUT. Triggers: TRG\_TIMESESSION\_AUDIT.

Security (optional): Roles APP\_STUDENT/APP\_INSTRUCTOR/APP\_ADMIN, PKG\_SECURITY\_CTX, ST\_CTX context, VPD policy on V\_TIMESESSION\_FULL.