Software Design

Assignment 1 - Tennis Tournament

Name: Onaci Andreea-Maria
Group: 30431

Teaching Assistant: Teodora Furcovici

Contents

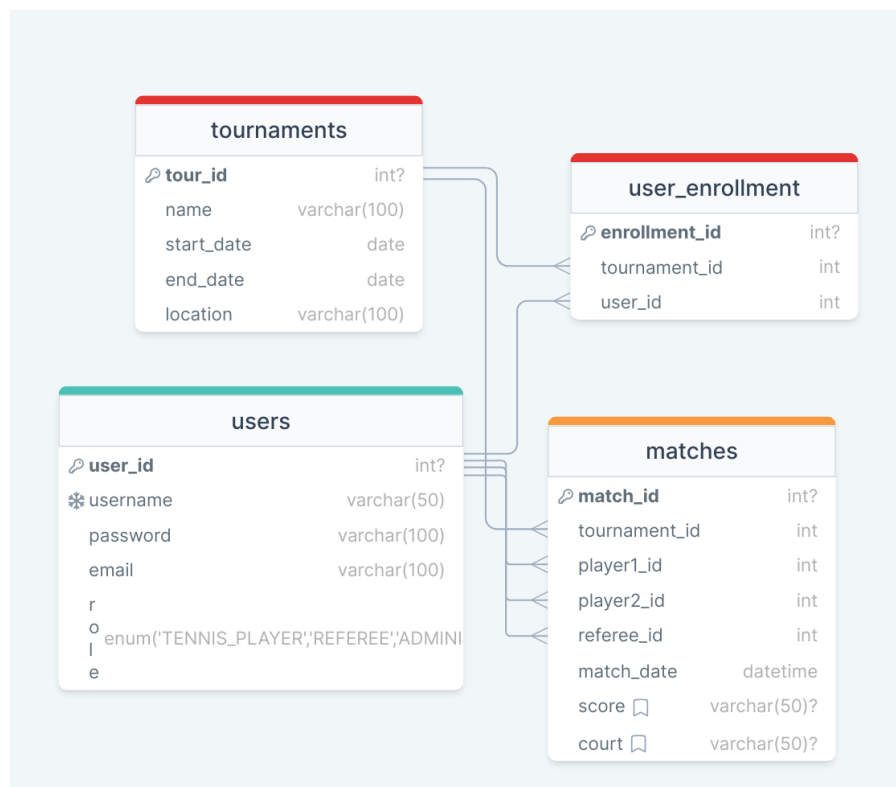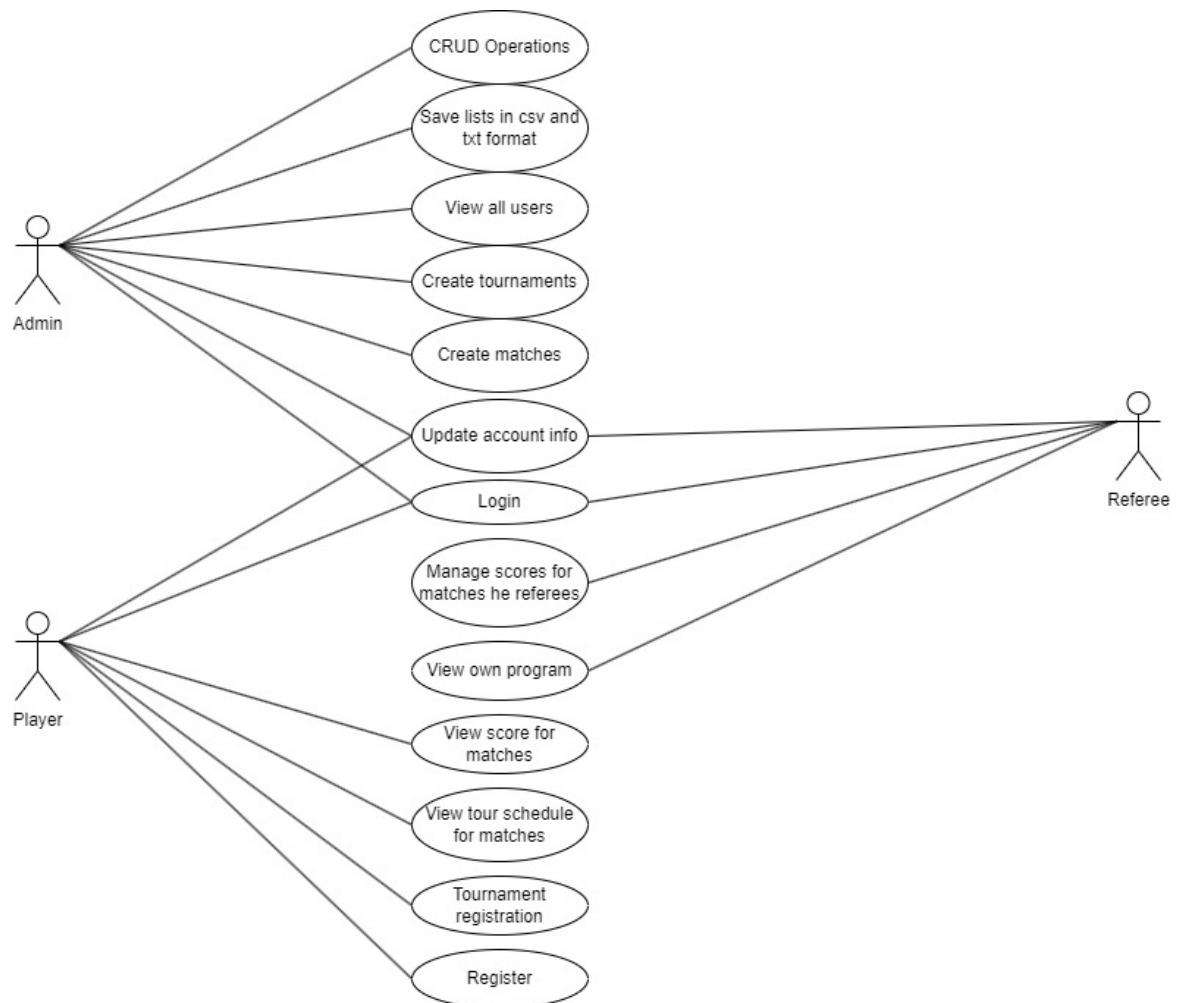# Chapter 1

Database diagram

Chapter 2

Use-case diagram

Chapter 3

Class-diagram

Database Layer

<<enumeration>>
UserRole
+ UserRole();
+ values(): UserRole[]
+ valueOf(String): UserRole

User
+ User();
+ User(String, String, String, UserRole)
userId: Long
password: String
role: UserRole
username: String
id: Long

Tournament
+ Tournament();
location: String

UserUtility
+ UserUtility();
+ getUserIdByUsername(String): Long

Encoder
INSTANCE Encoder

UserEnrollment
+ UserEnrollment(Long, Long);
+ UserEnrollment();
userId: Long
tournamentLocation: String
tournamentId: Long
id: Long
playerUsername: String

ExportRequest
csvExport boolean
txtExport boolean
columnsToExport

Match
+ Match();
player1: User
score: String
player2: User
matchDate: Date
courtName: String
matchId: Long
formattedMatchDate: String
tournament: Tournament

Business Layer

MatchService
+ MatchService();
+ getRefereeMatches(Long): List<Match>
+ createMatch(Match): Match
+ getMatchesByTournamentId(Long): List<Match>
+ updateMatchScore(Long, String): Match
+ getFilteredMatches(ExportRequest): List<Match>
+ getUserMatches(Long): List<Match>

TournamentService
+ TournamentService();
+ getTournamentById(Long): Tournament
+ registerTournament(Tournament): Tournament

UserService
+ UserService();
+ validateCredentials(String, String): boolean
+ deleteUserById(Long): void
+ loginUser(User): String
+ getUserRole(String): String
+ getUserById(Long): User
+ getUserByUsername(String): User
+ updateUser(Long, String, String, String, UserRole): String
+ registerUser(User): User
+ isUserReferee(Long): boolean
+ getUserIdByUsername(String): Long

UserEnrollmentService
+ UserEnrollmentService(UserEnrollmentRepository)
+ isUserEnrolled(Long, Long): boolean
+ registerUserForTournament(UserEnrollment): void

Persistance Layer

<<interface>>
UserRepository
+ findByUsernameAndPassword(String, String): User
+ updateUser(Long, String, String, String, UserRole)
+ getUserRoleByUsername(String): String
+ save(User): User
+ deleteById(Long): void

<<interface>>
MatchRepository
+ findByTournamentId(Long): List<Match>
+ findRefereeMatches(Long): List<Match>
+ findUserMatches(Long): List<Match>

<<interface>>
TournamentRepository

<<interface>>
UserEnrollmentRepository
+ existsByTournamentIdAndUserId(Long, Long): boolean
+ findByTournamentId(Long): List<UserEnrollment>

View Package

UserController
+ UserController();
+ getMatchesScoresByUserId(String): ResponseEntity<List<Match>>
+ getMatchesByUserId(String): ResponseEntity<List<Match>>
+ index(Model): String
+ showAdminViewUsersDashboard(Model): String
+ updateUser(Long, String, String, String, UserRole): ResponseEntity<U
+ showRefereeOwnProgram(String, Model): String
+ showUpdateAccountDashboard(Model): String
+ registerUser(User): ResponseEntity<String>
+ getUserByUsername(String): ResponseEntity<User>
+ updateMatchScore(Long, String): ResponseEntity<String>
+ showScheduleDashboard(String): String
+ showCreateUserPage(): String
+ showDeleteAccountDashboard(): String
+ showExportInfo(Model): String
+ showLoginPage(): String
+ showCreateMatches(Model): String
+ showUpdateAccountInformation(Model, String): String
+ loginUser(User): ResponseEntity<?>
+ showManageMatchesScore(String, Model): String
+ showPlayerDashboard(String, Model): String
+ showAdminUserManagementDashboard(Model): String
+ getUserById(Long): ResponseEntity<User>
+ showMatchesScore(String, Model): String
+ showRefereeDashboard(String, Model): String
+ showRegisterPage(): String
+ createUser(User): ResponseEntity<String>
+ deleteUserById(Long): ResponseEntity<String>
+ showTournamentRegistrationDashboard(String): String
+ showAdminDashboard(String, Model): String
+ getMatchesScoreByUserId(String): ResponseEntity<List<Match>>
allUsers: ResponseEntity<List<User>>

MatchController
+ MatchController();
+ updateMatchScores(List<Match>): ResponseEntity<String>
+ showScheduleDashboard(Model): String
+ getMatchesByRefereeUsername(String): ResponseEntity<List<M
+ createMatch(Map<String, Object>): ResponseEntity<?>
+ getMatchesByUserUsername(String): ResponseEntity<List<Match>
+ getMatchesByTournamentId(Long): ResponseEntity<List<Match>
+ getMatchesScoresByUsername(String): ResponseEntity<List<Ma
+ exportMatchInformation(ExportRequest): ResponseEntity<String>

TournamentController
+ TournamentController();
+ getMatchesByTournamentId(Long): ResponseEntity<List<Match>
+ registerForTournament(Long, UserEnrollment): ResponseEntity<
+ createTournament(Tournament): ResponseEntity<Tournament>
+ showTournamentRegistrationDashboard(String, Model): String

Assignment1Application
+ Assignment1Application();
+ main(String[]): void

Package Layer