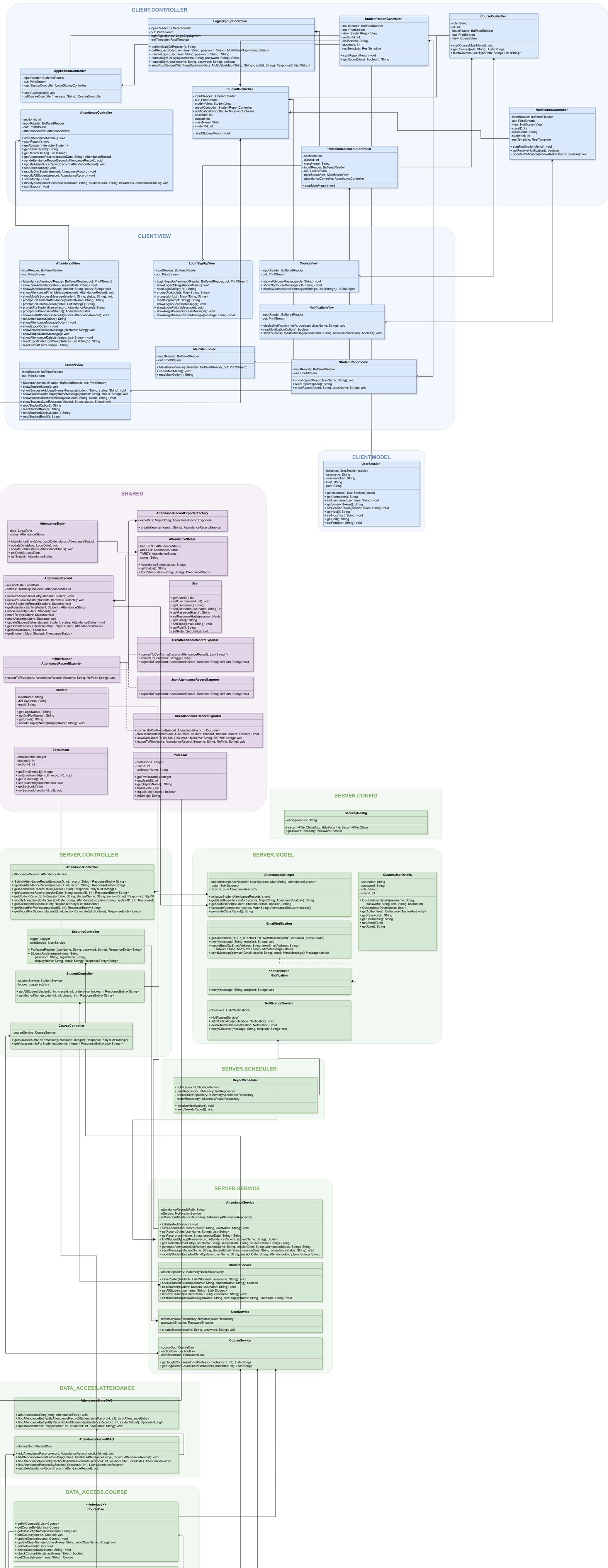
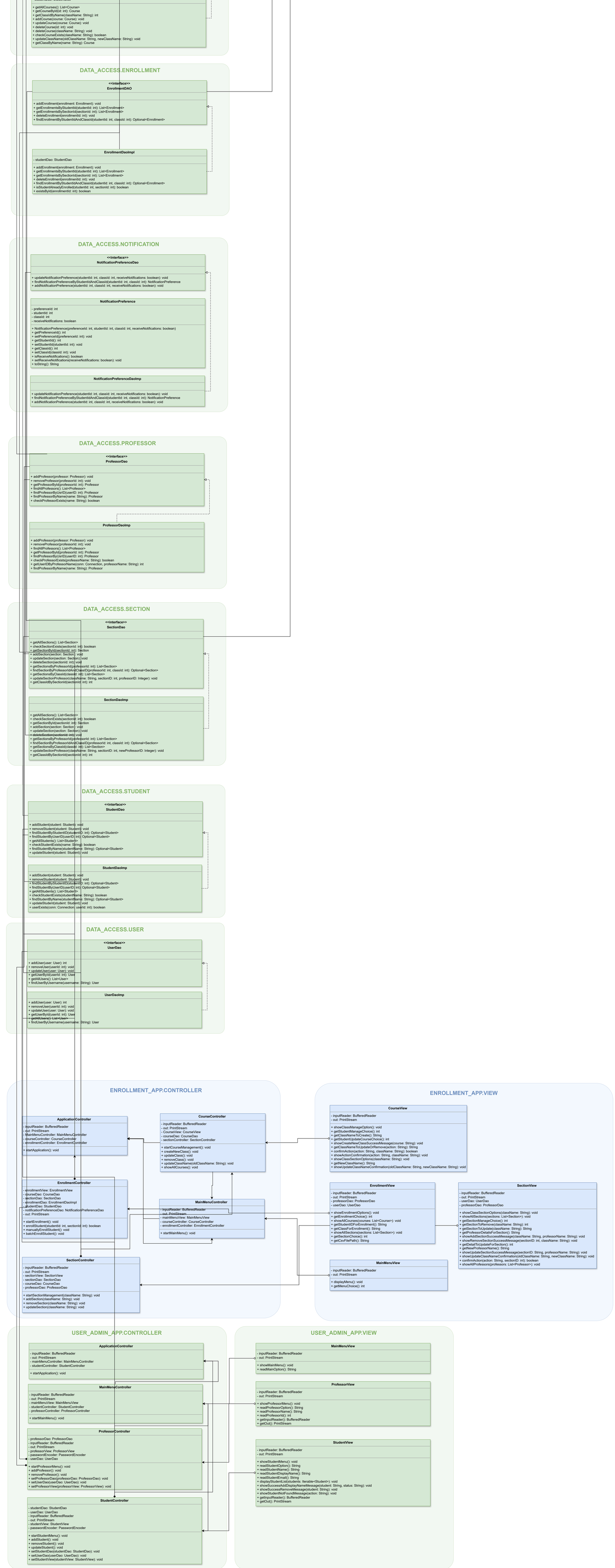

CLIENT.CONTROLLER
LoginSignupController
- inputReader: BufferedReader
- out: PrintStream
- loginSignUpView: LoginSignUpView
- restTemplate: RestTemplate
+ authenticateOrRegister(): String
+ getRequestBody(username: String, password: String): MultiValueMap<String, String>
+ handleLogin(username: String, password: String): String
+ handleSignUpLogin(username: String, password: String): String
+ handleSignUp(username: String, password: String): boolean
+ sendPostRequestWithFormData(formData: MultiValueMap<String, String>, apiUrl: String): ResponseEntity<String>
StudentReportController
- inputReader: BufferedReader
- out: PrintStream
- view: StudentReportView
- sectionId: int
- className: String
- studentId: int
- restTemplate: RestTemplate
+ startReportMenu(): void
+ getReport(detail: boolean): String
CourseController
- role: String
- id: int
- inputReader: BufferedReader
- out: PrintStream
- view: CourseView
+ startCourseMainMenu(): void
+ getCourses(role: String): List<String>
+ fetchCourses(userTypePath: String): List<String>
ApplicationController
- inputReader: BufferedReader
- out: PrintStream
- loginSignupController: LoginSignupController
+ startApplication(): void
- getCourseController(message: String): CourseController
StudentController
- inputReader: BufferedReader
- out: PrintStream
- studentView: StudentView
- reportController: StudentReportController
- notificationController: NotificationController
- sectionId: int
- classId: int
- className: String
- studentId: int
+ startStudentMenu(): void
AttendanceController
- sectionId: int
- inputReader: BufferedReader
- out: PrintStream
- attendanceView: AttendanceView
+ startAttendanceMenue(): void
+ startReport(): void
+ getRoaster(): Iterable<Student>
+ getClassReport(): String
+ getRecordDates(): List<String>
+ getAttendanceRecord(sessionDate: String): AttendanceRecord
+ sendAttendanceRecord(record: AttendanceRecord): void
+ updateAttendanceRecord(record: AttendanceRecord): void
+ startAttendance(): void
+ modifyOneStudent(record: AttendanceRecord): void
+ modifyAllStudents(record: AttendanceRecord): void
+ startModify(): void
+ modifyAttendanceRecord(sessionDate: String, studentName: String, newStatus: AttendanceStatus): void
+ startExport(): void
NotificationController
- inputReader: BufferedReader
- out: PrintStream
- view: NotificationView
- classID: int
- className: String
- studentId: int
- restTemplate: RestTemplate
+ startNotificationMenu(): void
+ getReceiveNotification(): boolean
+ updateNotification(receiveNotifications: boolean): void
ProfessorMainMenuController
- sectionId: int
- classId: int
- className: String
- inputReader: BufferedReader
- out: PrintStream
- mainMenuView: MainMenuView
- attendanceController: AttendanceController
+ startMainMenu(): void
CLIENT.VIEW
AttendanceView
- inputReader: BufferedReader
- out: PrintStream
+ AttendanceView(inputReader: BufferedReader, out: PrintStream)
+ showTakeAttendanceMenu(sessionDate: String): void
+ showMarkSuccessMessage(student: String, status: String): void
+ showAttendanceFinishMessage(records: AttendanceRecord): void
+ showModifySuccessMessage(student: String, status: String): void
+ promptForStudentAttendance(studentName: String): String
+ promptForDateSelection(dates: List<String>): String
+ promptForStudentName(record: AttendanceRecord): String
+ promptForAttendanceStatus(): AttendanceStatus
+ showFinalAttendanceRecord(record: AttendanceRecord): void
+ readAttendanceOption(): String
+ showAttendanceManageOption(): void
+ showExportOption(): void
+ showExportSuccessMessage(fileName: String): void
+ showEmptyDatesMessage(): void
+ showAttendanceDateList(dates: List<String>): void
+ readExportDateFromPrompt(dates: List<String>): String
+ readFormatFromPrompt(): String
LoginSignUpView
- inputReader: BufferedReader
- out: PrintStream
+ LoginSignUpView(inputReader: BufferedReader, out: PrintStream)
+ showLoginOrRegistrationMenu(): void
+ readLoginOrSignUp(): String
+ promptForLogin(): Map<String, String>
+ promptsignUp(): Map<String, String>
+ readInfo(prompt: String): String
+ showLoginSuccessMessage(): void
+ showLoginFailureMessage(): void
+ showRegistrationSuccessMessage(): void
+ showRegistrationFailureMessage(message: String): void
CourseView
- inputReader: BufferedReader
- out: PrintStream
+ showWelcomeMessage(role: String): void
+ showNoCourseMessage(role: String): void
+ displayCoursesAndPrompt(jsonStrings: List<String>): JSONObject
NotificationView
- inputReader: BufferedReader
- out: PrintStream
+ displayNotification(notify: boolean, className: String): void
+ readNotificationOption(): boolean
+ showSuccessUpdateMessage(className: String, receiveNotifications: boolean): void
MainMenuView
- inputReader: BufferedReader
- out: PrintStream
+ MainMenuView(inputReader: BufferedReader, out: PrintStream)
+ showMainMenu(): void
+ readMainOption(): String
StudentView
- inputReader: BufferedReader
- out: PrintStream
+ StudentView(inputReader: BufferedReader, out: PrintStream)
+ showStudentMenu(): void
+ showSuccessAddLegalNameMessage(student: String, status: String): void
+ showSuccessAddDisplayNameMessage(student: String, status: String): void
+ showSuccessRemoveMessage(student: String, status: String): void
+ showSuccessLoadMessage(student: String, status: String): void
+ readStudentOption(): String
+ readStudentName(): String
+ readStudentDisplayName(): String
+ readStudentEmail(): String
StudentReportView
- inputReader: BufferedReader
- out: PrintStream
+ showReportMenu(className: String): void
+ readReportOption(): String
+ showReport(report: String, class Name: String): void
CLIENT.MODEL
UserSession
- instance: UserSession (static)
- username: String
- sessionToken: String
- host: String
- port: String
+ getInstance(): UserSession (static)
+ getUsername(): String
+ setUsername(username: String): void
+ getSessionToken(): String
+ setSessionToken(sessionToken: String): void
+ getHost(): String
+ setHost(host: String): void
+ getPort(): String
+ setPort(port: String): void
SHARED
AttendanceEntry
- date: LocalDate
- status: AttendanceStatus
+ AttendanceEntry(date: LocalDate, status: AttendanceStatus)
+ updateDate(date: LocalDate): void
+ updateStatus(status: AttendanceStatus): void
+ getDate(): LocalDate
+ getStatus(): AttendanceStatus
AttendanceRecordExporterFactory
- exporters: Map<String, AttendanceRecordExporter>
+ createExporter(format: String): AttendanceRecordExporter
AttendanceStatus
- PRESENT: AttendanceStatus
- ABSENT: AttendanceStatus
- TARDY: AttendanceStatus
- status: String
+ AttendanceStatus(status: String)
+ getStatus(): String
+ fromString(statusString: String): AttendanceStatus
AttendanceRecord
- sessionDate: LocalDate
- entries: HashMap<Student, AttendanceStatus>
+ initializeAttendanceEntry(student: Student): void
+ initializeFromRoaster(students: Iterable<Student>): void
+ checkStudentInRecord(student: Student): void
+ getAttendanceEntry(student: Student): AttendanceStatus
+ markPresent(student: Student): void
+ markTardy(student: Student): void
+ markAbsent(student: Student): void
+ updateStudentStatus(student: Student, status: AttendanceStatus): void
+ getSortedEntries(): Iterable<Map.Entry<Student, AttendanceStatus>>
+ getSessionDate(): LocalDate
+ getEntries(): Map<Student, AttendanceStatus>
User
+ getUserId(): int
+ setUserId(userId: int): void
+ getUsername(): String
+ setUsername(username: String): v
+ getPasswordHash(): String
+ setPasswordHash(passwordHash:
+ getEmail(): String
+ setEmail(email: String): void
+ getRole(): String
+ setRole(role: String): void
CsvAttendanceRecordExporter
+ convertToCsvFormat(record: AttendanceRecord): List<String[]>
+ convertToCSV(data: String[]): String
+ exportToFile(record: AttendanceRecord, filename: String, filePath: String): void
<<interface>>
AttendanceRecordExporter
+ exportToFile(record: AttendanceRecord, filename: String, filePath: String): void
JsonAttendanceRecordExporter
+ exportToFile(record: AttendanceRecord, filename: String, filePath: String): void
Student
- legalName: String
- disPlayName: String
- email: String
+ getLegalName(): String
+ getDisPlayName(): String
+ getEmail(): String
+ updateDisplayName(dispalyName: String): void
XmlAttendanceRecordExporter
+ convertToXmlFormat(record: AttendanceRecord): Document
- createStudentElement(doc: Document, student: Student, studentElement: Element): void
+ saveDocumentToFile(doc: Document, filename: String, filePath: String): void
+ exportToFile(record: AttendanceRecord, filename: String, filePath: String): void
Enrollment
- enrollmentId: Integer
- studentId: int
- sectionId: int
+ getEnrollmentId(): Integer
+ setEnrollmentId(enrollmentId: int): void
+ getStudentId(): int
+ setStudentId(studentId: int): void
+ getSectionId(): int
+ setSectionId(sectionId: int): void
Professor
- professorId: Integer
- userId: int
- professorName: String
+ getProfessorId(): Integer
+ getUserId(): int
+ getDisplayName(): String
+ hashCode(): int
+ equals(obj: Object): boolean
+ toString(): String
SERVER.CONFIG
SecurityConfig
- encryptionKey: String
+ securityFilterChain(http: HttpSecurity): SecurityFilterChain
+ passwordEncoder(): PasswordEncoder
SERVER.CONTROLLER
AttendanceController
- attendanceService: AttendanceService
+ SubmitAttendanceRecord(sectionID: int, record: String): ResponseEntity<String>
+ updateAttendanceRecord(sectionID: int, record: String): ResponseEntity<String>
+ getAttendanceRecordDates(sectionID: int): ResponseEntity<List<String>>
+ getAttendanceRecord(sessionDate: String, sectionID: int): ResponseEntity<String>
+ getStudentRecordEntry(sessionDate: String, studentName: String, sectionID: int): ResponseEntity<St
+ modifyAttendanceEntry(sessionDate: String, attendanceEntry,json: String, sectionID: int): ResponseE
+ getAllStudent(sectionID: int): ResponseEntity<List<Student>>
+ getReportForProfessor(sectionID: int): ResponseEntity<String>
+ getReportForStudent(studentID: int, sectionID: int, detail: Boolean): ResponseEntity<String>
SecurityController
- logger: Logger
- userService: UserService
+ ProfessorRegister(userName: String, password: String): ResponseEntity<String>
+ StudentRegister(userName: String,
password: String, legalName: String,
displayName: String, email: String): ResponseEntity<String>
StudentController
- studentService: StudentService
- logger: Logger (static)
+ getAllStudent(studentId: int, classId: int, preference: boolean): ResponseEntity<String>
+ getMethodName(studentId: int, classId: int): ResponseEntity<String>
CourseController
- courseService: CourseService
+ getAllclassesInfoForProfessor(professorId: Integer): ResponseEntity<List<String>>
+ getAllclassesInfoForStudent(studentId: Integer): ResponseEntity<List<String>>
SERVER.MODEL
AttendanceManager
- studentAttendanceRecords: Map<Student, Map<String, AttendanceStatus>>
- roster: Set<Student>
- records: List<AttendanceRecord>
+ initializeStudentAttendanceRecords(): void
+ getDetailAttendanceInfor(records: Map<String, AttendanceStatus>): String
+ generateReport(student: Student, details: boolean): String
+ calculateAttendance(records: Map<String, AttendanceStatus>): double[]
+ generateClassReport(): String
CustomUserDetails
- username: String
- password: String
- role: String
- userId: int
+ CustomUserDetails(username: String,
password: String, role: String, userId: int)
+ CustomUserDetails(user: User)
+ getAuthorities(): Collection<GrantedAuthority>
+ getPassword(): String
+ getUsername(): String
+ getUserId(): int
+ getRole(): String
EmailNotification
+ getCredentials(HTTP_TRANSPORT: NetHttpTransport): Credential (private static)
+ notify(message: String, recipient: String): void
+ createEmail(toEmailAddress: String, fromEmailAddress: String,
subject: String, bodyText: String): MimeMessage (static)
+ sendMessage(service: Gmail, userId: String, email: MimeMessage): Message (static)
<<interface>>
Notification
+ notify(message: String, recipient: String): void
NotificationService
- observers: List<Notification>
+ NotificationService()
+ addNotification(notification: Notification): void
+ deleteNotification(notification: Notification): void
+ notifyObserver(message: String, recipient: String): void
SERVER.SCHEDULER
ReportScheduler
- notification: NotificationService
- userRepository: InMemoryUserRepository
- attendanceRepository: InMemoryAttendanceRepository
- rosterRepository: InMemoryRosterRepository
+ initializeNotification(): void
+ sendWeeklyReport(): void
SERVER.SERVICE
AttendanceService
- attendanceRecordsPath: String
- nService: NotificationService
- inMemoryAttendanceRepository: InMemoryAttendanceRepository
+ initializeNotification(): void
+ saveAttendanceRecord(record: String, userName: String): void
+ getRecordDates(userName: String): List<String>
+ getRecord(userName: String, sessionDate: String): String
+ findStudentByLegalName(record: AttendanceRecord, studentName: String): Student
+ getStudentRecordEntry(userName: String, sessionDate: String, studentName: String): String
+ generateAtterdanceNotification(studentName: String, sessionDate: String, attendanceStatus: String): String
+ sendMessage(studentName: String, studentEmail: String, sessionDate: String, attendanceStatus: String): void
+ modifyStudentEntryAndSendUpdates(userName: String, sessionDate: String, attendanceEntryJson: String): String
StudentService
- rosterRepository: InMemoryRosterRepository
+ saveRoster(students: List<Student>, username: String): void
+ checkStudentExists(username: String, studentName: String): boolean
+ addStudent(student: Student, username: String): void
+ getAllStudent(username: String): List<Student>
+ removeStudent(studentName: String, username: String):void
+ editStudentDisplayName(legalName: String, newDisplayName: String, username: String): void
UserService
- inMemoryUserRepository: InMemoryUserRepository
- passwordEncoder: PasswordEncoder
+ createUser(username: String, password: String): void
CourseService
- courseDao: CourseDao
- sectionDao: SectionDao
- enrollmentDao: EnrollmentDao
+ getTaughtCoursesInfoForProfessor(professorId: int): List<String>
+ getRegisteredCoursesInfoForStudent(studentID: int): List<String>
DATA_ACCESS.ATTENDANCE
AttendanceEntryDAO
+ addAttendanceEntry(entry: AttendanceEntry): void
+ findAttendanceEntriesByAttendanceRecordId(attendanceRecordId: int): List<AttendanceEntry>
+ findAttendanceEntryIdByRecordIdAndStudentId(attendanceRecordId: int, studentId: int): Optional<Long>
+ updateAttendanceEntry(recordId: int, studentId: int, newStatus: String): void
AttendanceRecordDAO
- studentDao: StudentDao
+ addAttendanceRecord(record: AttendanceRecord, sectionId: int): void
+ fillAttendanceRecordEntriesMap(entries: Iterable<AttendanceEntry>, record: AttendanceRecord): void
+ findAttendanceRecordBySectionIDAndSessionDate(sectionId: int, sessionDate: LocalDate): AttendanceRecord
+ findAttendanceRecordsBySectionID(sectionId: int): List<AttendanceRecord>
+ updateAttendanceRecord(record: AttendanceRecord): void
DATA_ACCESS.COURSE
<<interface>>
CourseDao
+ getAllCourses(): List<Course>
+ getCourseById(id: int): Course
+ getClassIdByName(className: String): int
+ addCourse(course: Course): void
+ updateCourse(course: Course): void
+ updateClassName(oldClassName: String, newClassName: String): void
+ deleteCourse(id: int): void
+ deleteCourse(className: String): void
+ checkCourseExists(className: String): boolean
+ getClassByName(name: String): Course
CourseDaoImp
- studentDao: StudentDao


DATA_ACCESS.ENROLLMENT
<<interface>>
EnrollmentDAO
+ addEnrollment(enrollment: Enrollment): void
+ getEnrollmentsByStudentId(studentId: int): List<Enrollment>
+ getEnrollmentsBySectionId(sectionId: int): List<Enrollment>
+ deleteEnrollment(enrollmentId: int): void
+ findEnrollmentByStudentIdAndClassId(studentId: int, classId: int): Optional<Enrollment>
EnrollmentDaoImpl
- studentDao: StudentDao
+ addEnrollment(enrollment: Enrollment): void
+ getEnrollmentsByStudentId(studentId: int): List<Enrollment>
+ getEnrollmentsBySectionId(sectionId: int): List<Enrollment>
+ deleteEnrollment(enrollmentId: int): void
+ findEnrollmentByStudentIdAndClassId(studentId: int, classId: int): Optional<Enrollment>
+ isStudentAlreadyEnrolled(studentId: int, sectionId: int): boolean
+ existsById(enrollmentId: int): boolean
DATA_ACCESS.NOTIFICATION
<<interface>>
NotificationPreferenceDao
+ updateNotificationPreference(studentId: int, classId: int, receiveNotifications: boolean): void
+ findNotificationPreferenceByStudentIdAndClassId(studentId: int, classId: int): NotificationPreference
+ addNotificationPreference(studentId: int, classId: int, receiveNotifications: boolean): void
NotificationPreference
- preferenceId: int
- studentId: int
- classId: int
- receiveNotifications: boolean
+ NotificationPreference(preferenceId: int, studentId: int, classId: int, receiveNotifications: boolean)
+ getPreferenceId(): int
+ setPreferenceId(preferenceId: int): void
+ getStudentId(): int
+ setStudentId(studentId: int): void
+ getClassId(): int
+ setClassId(classId: int): void
+ isReceiveNotifications(): boolean
+ setReceiveNotifications(receiveNotifications: boolean): void
+ toString(): String
NotificationPreferenceDaoImp
+ updateNotificationPreference(studentId: int, classId: int, receiveNotifications: boolean): void
+ findNotificationPreferenceByStudentIdAndClassId(studentId: int, classId: int): NotificationPreference
+ addNotificationPreference(studentId: int, classId: int, receiveNotifications: boolean): void
DATA_ACCESS.PROFESSOR
<<interface>>
ProfessorDao
+ addProfessor(professor: Professor): void
+ removeProfessor(professorId: int): void
+ getProfessorById(professorId: int): Professor
+ findAllProfessors(): List<Professor>
+ findProfessorByUserID(userID: int): Professor
+ findProfessorByName(name: String): Professor
+ checkProfessorExists(name: String): boolean
ProfessorDaoImp
+ addProfessor(professor: Professor): void
+ removeProfessor(professorId: int): void
+ findAllProfessors(): List<Professor>
+ getProfessorById(professorId: int): Professor
+ findProfessorByUserID(userID: int): Professor
+ checkProfessorExists(professorName: String): boolean
+ getUserIDByProfessorName(conn: Connection, professorName: String): int
+ findProfessorByName(name: String): Professor
DATA_ACCESS.SECTION
<<interface>>
SectionDao
+ getAllSections(): List<Section>
+ checkSectionExists(sectionId: int): boolean
+ getSectionById(sectionId: int): Section
+ addSection(section: Section): void
+ updateSection(section: Section): void
+ deleteSection(sectionId: int): void
+ getSectionsByProfessorId(professorId: int): List<Section>
+ findSectionByProfessorIdAndClassID(professorId: int, classId: int): Optional<Section>
+ getSectionsByClassId(classId: int): List<Section>
+ updateSectionProfessor(claseName: String, sectionID: int, professorID: Integer): void
+ getClassIdBySectionId(sectionId: int): int
SectionDaoImp
+ getAllSections(): List<Section>
+ checkSectionExists(sectionId: int): boolean
+ getSectionById(sectionId: int): Section
+ addSection(section: Section): void
+ updateSection(section: Section): void
+ deleteSection(sectionId: int): void
+ getSectionsByProfessorId(professorId: int): List<Section>
+ findSectionByProfessorIdAndClassID(professorId: int, classId: int): Optional<Section>
+ getSectionsByClassId(classId: int): List<Section>
+ updateSectionProfessor(claseName: String, sectionID: int, newProfessorID: Integer): void
+ getClassIdBySectionId(sectionId: int): int
DATA_ACCESS.STUDENT
<<interface>>
StudentDao
+ addStudent(student: Student): void
+ removeStudent(student: Student): void
+ findStudentByStudentID(studentID: int): Optional<Student>
+ findStudentByUserID(userID: int): Optional<Student>
+ getAllStudents(): List<Student>
+ checkStudentExists(name: String): boolean
+ findStudentByName(studentName: String): Optional<Student>
+ updateStudent(student: Student): void
StudentDaoImp
+ addStudent(student: Student): void
+ removeStudent(student: Student): void
+ findStudentByStudentID(studentID: int): Optional<Student>
+ findStudentByUserID(userID: int): Optional<Student>
+ getAllStudents(): List<Student>
+ checkStudentExists(studentName: String): boolean
+ findStudentByName(studentName: String): Optional<Student>
+ updateStudent(student: Student): void
+ userExists(conn: Connection, userId: int): boolean
DATA_ACCESS.USER
<<interface>>
UserDao
+ addUser(user: User): int
+ removeUser(userId: int): void
+ updateUser(user: User): void
+ getUserById(userId: int): User
+ getAllUsers(): List<User>
+ findUserByUsername(username: String): User
UserDaoImp
+ addUser(user: User): int
+ removeUser(userId: int): void
+ updateUser(user: User): void
+ getUserById(userId: int): User
+ getAllUsers(): List<User>
+ findUserByUsername(username: String): User
ENROLLMENT_APP.CONTROLLER
ApplicationController
- inputReader: BufferedReader
- out: PrintStream
- MainMenuController: MainMenuController
- courseController: CourseController
- enrollmentController: EnrollmentController
+ startApplication(): void
CourseController
- inputReader: BufferedReader
- out: PrintStream
- CourseView: CourseView
- courseDao: CourseDao
- sectionController: SectionController
+ startCourseManagement(): void
+ createNewClass(): void
+ updateClass(): void
+ removeClass(): void
+ updateClassName(oldClassName: String): void
+ showAllCourses(): void
EnrollmentController
- enrollmentView: EnrollmentView
- courseDao: CourseDao
- sectionDao: SectionDao
- enrollmentDao: EnrollmentDaoImpl
- studentDao: StudentDao
- notificationPreferenceDao: NotificationPreferenceDao
- out: PrintStream
+ startEnrollment(): void
+ enrollStudent(studentId: int, sectionId: int): boolean
+ manuallyEnrollStudent(): void
+ batchEnrollStudent(): void
MainMenuController
- inputReader: BufferedReader
- out: PrintStream
- mainMenuView: MainMenuView
- courseController: CourseController
- enrollmentController: EnrollmentController
+ startMainMenu(): void
SectionController
- inputReader: BufferedReader
- out: PrintStream
- sectionView: SectionView
- sectionDao: SectionDao
- courseDao: CourseDao
- professorDao: ProfessorDao
+ startSectionManagement(claseName: String): void
+ addSection(claseName: String): void
+ removeSection(claseName: String): void
+ updateSection(claseName: String): void
ENROLLMENT_APP.VIEW
CourseView
- inputReader: BufferedReader
- out: PrintStream
+ showClassManageOption(): void
+ getStudentManageChoice(): int
+ getClassNameToCreate(): String
+ getStudentUpdateCourseChoice(): int
+ showCreateNewClassSuccessMessage(course: String): void
+ getClassNameToUpdateOrRemove(action: String): String
+ confirmAction(action: String, claseName: String): boolean
+ showActionConfirmation(action: String, claseName: String): void
+ showClassSectionOptions(claseName: String): void
+ getNewClassName(): String
+ showUpdateClassNameConfirmation(oldClassName: String, newClassName: String): void
EnrollmentView
- inputReader: BufferedReader
- out: PrintStream
- professorDao: ProfessorDao
- userDao: UserDao
+ showEnrollmentOptions(): void
+ getEnrollmentChoice(): int
+ showAllCourses(courses: List<Course>): void
+ getStudentIDForEnrollment(): String
+ getClassForEnrollment(): String
+ showAllSections(sections: List<Section>): void
+ getSectionChoice(): int
+ getCsvFilePath(): String
SectionView
- inputReader: BufferedReader
- out: PrintStream
- userDao: UserDao
- professorDao: ProfessorDao
+ showClassSectionOptions(claseName: String): void
+ showAllSections(sections: List<Section>): void
+ getSectionManageChoice(): int
+ getSectionToRemove(claseName: String): int
+ getSectionToUpdate(claseName: String): String
+ getProfessorDetailsForSection(): String
+ showAddSectionSuccessMessage(claseName: String, professorName: String): void
+ showRemoveSectionSuccessMessage(sectionID: int, claseName: String): void
+ getDetailToUpdateForSection(): int
+ getNewProfessorName(): String
+ showUpdateSectionSuccessMessage(sectionID: String, professorName: String): void
+ showUpdateClassNameConfirmation(oldClassName: String, newClassName: String): void
+ confirmAction(action: String, sectionID: int): boolean
+ showAllProfessors(professors: List<Professor>): void
MainMenuView
- inputReader: BufferedReader
- out: PrintStream
+ displayMenu(): void
+ getMenuChoice(): int
USER_ADMIN_APP.CONTROLLER
ApplicationController
- inputReader: BufferedReader
- out: PrintStream
- mainMenuController: MainMenuController
- studentController: StudentController
+ startApplication(): void
MainMenuController
- inputReader: BufferedReader
- out: PrintStream
- mainMenuView: MainMenuView
- studentController: StudentController
- professorController: ProfessorController
+ startMainMenu(): void
ProfessorController
- professorDao: ProfessorDao
- inputReader: BufferedReader
- out: PrintStream
- professorView: ProfessorView
- passwordEncoder: PasswordEncoder
- userDao: UserDao
+ startProfessorMenu(): void
+ addProfessor(): void
+ removeProfessor(): void
+ setProfessorDao(professorDao: ProfessorDao): void
+ setUserDao(userDao: UserDao): void
+ setProfessorView(professorView: ProfessorView): void
StudentController
- studentDao: StudentDao
- userDao: UserDao
- inputReader: BufferedReader
- out: PrintStream
- studentView: StudentView
- passwordEncoder: PasswordEncoder
+ startStudentMenu(): void
+ addStudent(): void
+ removeStudent(): void
+ updateStudent(): void
+ setStudentDao(studentDao: StudentDao): void
+ setUserDao(userDao: UserDao): void
+ setStudentView(studentView: StudentView): void
USER_ADMIN_APP.VIEW
MainMenuView
- inputReader: BufferedReader
- out: PrintStream
+ showMainMenu(): void
+ readMainOption(): String
ProfessorView
- inputReader: BufferedReader
- out: PrintStream
+ showProfessorMenu(): void
+ readProfessorOption(): String
+ readProfessorName(): String
+ readProfessorId(): int
+ getInputReader(): BufferedReader
+ getOut(): PrintStream
StudentView
- inputReader: BufferedReader
- out: PrintStream
+ showStudentMenu(): void
+ readStudentOption(): String
+ readStudentName(): String
+ readStudentDisplayName(): String
+ readStudentEmail(): String
+ displayStudentList(students: Iterable<Student>): void
+ showSuccessAddDisplayNameMessage(student: String, status: String): void
+ showSuccessRemoveMessage(student: String): void
+ showStudentNotFoundMessage(action: String): void
+ getInputReader(): BufferedReader
+ getOut(): PrintStream