BILKENT UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
CS 319 OBJECT-ORIENTED SOFTWARE ENGINEERING
DELIVERABLE 2 ITERATION 1
SPRING 2025
GROUP 2 SECTION 1
16.03.2025

Full Name	ID
Perhat Amanlyyev	22201007
Emiralp İlgen	22203114
Anıl Keskin	22201915
İlmay Taş	22201715
Simay Uygur	22203328

Contents

1.Non-Functional Requirements

1.1 Performance Requirements

- The system shall ensure that critical operations (e.g., login, schedule viewing, data retrieval) complete within **≤2 seconds** under standard load conditions (≤500 concurrent users).
- Chronological sorting, filtering, and dashboard rendering operations shall complete within **≤2 seconds** for optimal usability.
- The system must scale to handle **up to 1000 concurrent users** during peak periods using **Tomcat connection pooling** and **Spring Boot microservices**.
- Real-time updates must be implemented via **WebSockets** or **Server-Sent Events (SSE)** to ensure timely information delivery.
- The system shall employ **lazy loading** and **pagination** for large datasets (e.g., notifications, logs).
- React frontend shall utilize **code splitting** (e.g., **React.lazy**, **Suspense**) to improve initial load times.
- Database queries shall be optimized through proper **indexing** and **query caching** where appropriate.

1.2 Usability and Accessibility Requirements

- The system shall feature a user-friendly **React-based UI** compatible with desktop and mobile devices.
- The user interface shall comply with **WCAG 2.1 AA accessibility standards**, including keyboard navigation support and sufficient contrast ratios.
- Session termination must securely clear all session data (e.g., cookies, local storage) to prevent unauthorized access.
- All dashboards and views shall be role-specific (e.g., TAs, instructors, admins) with consistent design principles.
- React state management (e.g., **Redux** or **Context API**) shall minimize full-page reloads, ensuring seamless user experience.
- Critical user workflows (e.g., approving overrides, submitting swap requests) shall be reachable within **≤3 clicks** from the dashboard.

1.3 Security Requirements

- The system shall implement **Role-Based Access Control (RBAC)** using **Spring Security** with **JWT tokens** stored in **HTTP-only cookies** to mitigate XSS vulnerabilities.
- **Two-Factor Authentication (2FA)** shall be mandatory for administrative users.

- Passwords shall comply with the organization's policy: minimum **12 characters**, with at least **1 uppercase letter**, **1 number**, and **1 special character**.
- Passwords shall be hashed using **bcrypt** with a minimum cost factor of **12**.
- All sensitive data (e.g., IBANs, schedules, exam details) shall be encrypted using **AES-256** both at rest and in transit (via **TLS 1.3**).
- CSRF protection shall be enforced on all relevant endpoints.
- Users shall be logged out after **15 minutes** of inactivity.
- API communications between frontend and backend shall be restricted to **HTTPS** and secure headers (e.g., **CORS**, **Strict-Transport-Security**).

1.4 Reliability and Availability Requirements

- The system shall maintain **≥99.9% uptime**, corresponding to a maximum downtime of **8.76 hours per year**.
- Backend services shall utilize **automatic failover mechanisms** (e.g., Kubernetes readiness/liveness probes).
- Daily **automated backups** shall be performed to enable rapid recovery.
- Bulk operations (e.g., TA imports, mass notifications) shall include **rollback mechanisms** to ensure data consistency and integrity.
- The frontend shall support **offline access** (where applicable) via service workers or **IndexedDB** for caching critical data.

1.5 Scalability and Maintainability Requirements

- The system shall be designed to support an increase from **500 to 2000 concurrent users** with minimal architectural changes.
- The backend shall implement **HikariCP connection pooling** to optimize database connections.
- Spring Boot services shall follow a **layered architecture (Controller-Service-Repository)** to facilitate maintainability and modularity.
- Frontend components (built with **TypeScript**) shall achieve **≥90% unit test coverage**.
- Backend services shall achieve **≥80% test coverage** for unit and integration tests.
- All APIs shall comply with **OpenAPI 3.0** for ease of integration and future extensibility.

1.6 Interoperability and Integration Requirements

- The system shall integrate with the university's **Single Sign-On (SSO)** via **OAuth2**.

- The system shall interoperate with the university’s **SMTP servers** (using **STARTTLS**) for sending transactional emails (e.g., password resets, notifications).
- The backend shall expose **RESTful APIs** adhering to **OpenAPI 3.0** standards.
- External systems and future modules shall be able to communicate via these standardized APIs.

1.7 Data Integrity and Validation Requirements

- All user input forms shall be subject to completeness checks and validation before submission.
- IBAN fields shall be validated using standard checksum algorithms.
- TA assignment workflows shall automatically exclude TAs with time conflicts or other course enrollment restrictions.
- Optimistic locking techniques shall be used to prevent race conditions in TA assignment workflows.
- Data imports (e.g., bulk TA lists) shall undergo **schema validation** and reject invalid entries.

1.8 Logging, Auditability, and Compliance Requirements

- All critical actions (e.g., swap requests, approvals, leave submissions) shall be logged with metadata such as **timestamp**, **user ID**, and **IP address**.
- Logs shall be retained for a minimum of **7 years** to comply with audit and regulatory requirements.
- Log data shall be centralized using **Logback** or integrated with an external system (e.g., **ELK stack**).
- The system shall comply with **GDPR** and **KVKK** regulations, offering users the ability to request data deletion or export.

1.9 Notification System Requirements

- Notifications shall be delivered in real-time or near real-time using **WebSockets** or **SSE**.
- The Dean’s Office shall be notified immediately of critical events (e.g., override approvals).
- Notification lists shall support pagination and filtering.
- Swap and override request responses must generate automatic notifications to relevant TAs and instructors.

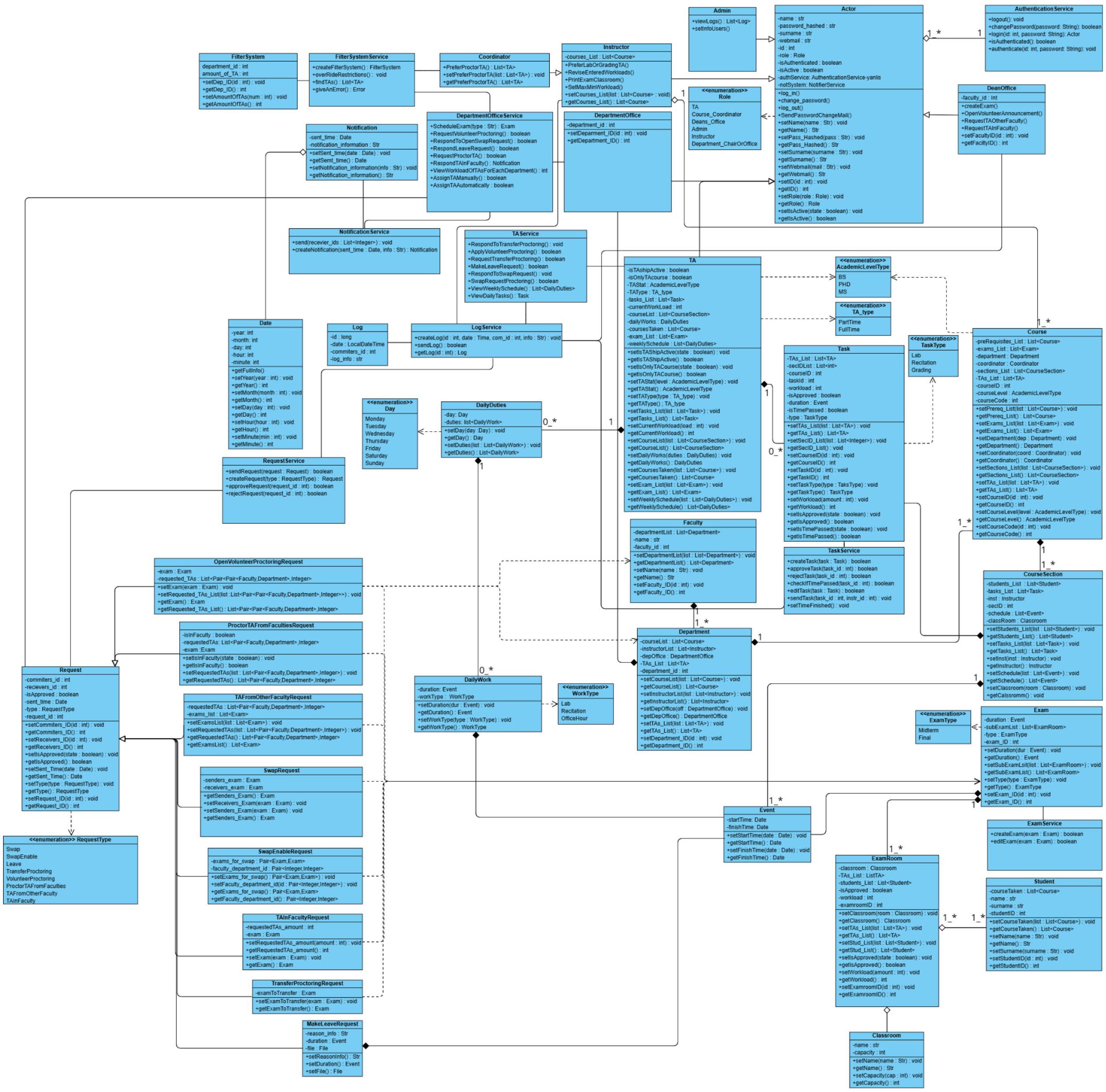
1.10 Proctoring and Scheduling Requirements

- Proctoring, swap, and override requests must be submitted no later than **6 hours** prior to the exam.
- The system shall enforce non-overlapping exam times and locations at both application and database levels.
- Leave approvals shall automatically restrict TA assignments during approved leave periods.
- Rejected leave requests may include justifications optionally provided by approvers.
- The TA assignment algorithm shall ensure equitable workload distribution among available TAs.
- The Dean’s Office shall have real-time access to monitor and manage proctoring requests and exceptions.

1.11 Data Import and Export Requirements

- The system shall support **bulk data import** of TA lists, schedules, and assignments via **Excel (.xlsx)** files.
- Data imports must validate schema compliance and reject invalid or incomplete entries.
- Exports shall be available in **PDF** and **Excel** formats, optionally including relevant metadata (e.g., export date, filters applied).
- Rollback functionality shall be available for bulk operations affecting critical datasets.

2. Class Diagram

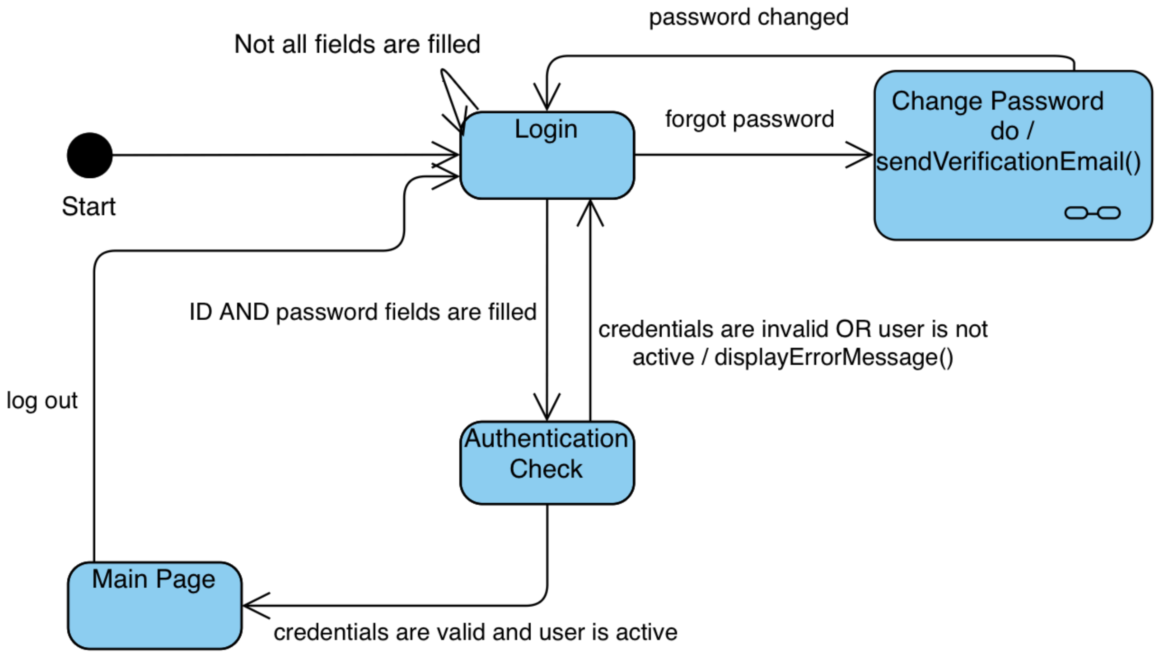

FilterSystem
department_id : int
amount_of_TA : int
+setDep_ID(id : int) : void
+getDep_ID() : int
+setAmountOfTAs(num : int) : void
+getAmountOfTAs() : int
FilterSystemService
+createFilterSystem() : FilterSystem
+overRideRestrictions() : void
+findTAs() : List<TA>
+giveAnError() : Error
Coordinator
+PreferProctorTA() : List<TA>
+setPreferProctorTA(list : List<TA>) : void
+getPreferProctorTA() : List<TA>
Instructor
-courses_List : List<Course>
+PreferLabOrGradingTA()
+ReviseEnteredWorkloads()
+PrintExamClassroom()
+SetMaxMinWorkload()
+setCourses_List(list : List<Course> : void)
+getCourses_List() : List<Course>
Admin
+viewLogs() : List<Log>
+setInfoUsers()
Actor
-name : str
-password_hashed : str
-surname : str
-webmail : str
-id : int
-role : Role
-isAuthenticated : boolean
-isActive : boolean
-authService: AuthenticationService-yanlis
-notSystem: NotifierService
+log_in()
+change_password()
+log_out()
+SendPasswordChangeMail()
+setName(name : Str) : void
+getName() : Str
+setPass_Hashed(pass : Str) : void
+getPass_Hashed() : Str
+setSurname(surname : Str) : void
+getSurname() : Str
+setWebmail(mail : Str) : void
+getWebmail() : Str
+setID(id : int) : void
+getID() : int
+setRole(role : Role) : void
+getRole() : Role
+setIsActive(state : boolean) : void
+getIsActive() : boolean
AuthenticationService
+logout(): void
+changePassword(password: String): boolean
+login(id: int, password: String): Actor
+isAuthenticated(): boolean
+authenticate(id: int, password: String): void
1_*
1
DeanOffice
-faculty_id : Int
+createExam()
+OpenVolunteerAnnouncement()
+RequestTAOtherFaculty()
+RequestTAInFaculty()
+setFacultyID(id : int) : void
+getFaciltyID() : int
<<enumeration>>
Role
TA
Course_Coordinator
Deans_Office
Admin
Instructor
Department_ChairOrOffice
1
DepartmentOffice
-department_id : int
+setDeparment_ID(id : int) : void
+getDepartment_ID() : int
DepartmentOfficeService
+ScheduleExam(type : Str) : Exam
+RequestVolunteerProctoring() : boolean
+RespondToOpenSwapRequest() : boolean
+RespondLeaveRequest() : boolean
+RequestProctorTA() : boolean
+RespondTAInFaculty() : Notification
+ViewWorkloadOfTAsForEachDepartment() : int
+AssignTAManually() : boolean
+AssignTAAutomatically : boolean
Notification
-sent_time : Date
-notification_information : Str
+setSent_time(date : Date) : void
+getSemt_time() : Date
+setNotification_information(info : Str) : void
+getNotification_information() : Str
NotificationService
+send(recevier_ids : List<Integer>) : void
+createNotification(sent_time : Date, info : Str) : Notification
TAService
+RespondToTransferProctoring() : void
+ApplyVolunteerProctoring() : boolean
+RequestTransferProctoring() : boolean
+MakeLeaveRequest() : boolean
+RespondToSwapRequest() : void
+SwapRequestProctoring() : boolean
+ViewWeeklySchedule() : List<DailyDuties>
+ViewDailyTasks() : Task
TA
-isTAshipActive : boolean
-isOnlyTAcourse : boolean
-TAStat : AcademicLevelType
-TAType : TA_type
-tasks_List : List<Task>
-currentWorkLoad : int
-courseList : List<CourseSection>
-dailyWorks : DailyDuties
-coursesTaken : List<Course>
-exam_List : List<Exam>
-weeklySchedule : List<DailyDuties>
+setIsTAShipActive(state : boolean) : void
+getIsTAShipActive() : boolean
+setIsOnlyTACourse(state : boolean) : void
+getIsOnlyTACourse() : boolean
+setTAStat(level : AcademicLevelType) : void
+getTAStat() : AcademicLevelType
+setTAType(type : TA_type) : void
+getTAType() ; TA_type
+setTasks_List(list : List<Task>) : void
+getTasks_List() : List<Task>
+setCurrentWorkload(load : int) : void
+getCurrentWorkload() : int
+setCourseList(list : List<CourseSection>) : void
+getCourseList() : List<CourseSection>
+setDailyWorks(duties : DailyDuties) : void
+getDailyWorks() : DailyDuties
+setCoursesTaken(list : List<Course>) : void
+getCoursesTaken() : List<Course>
+setExam_List(list : List<Exam>) : void
+getExam_List() : List<Exam>
+setWeeklySchedule(list : List<DailyDuties>) : void
+getWeeklySchedule() : List<DailyDuties>
<<enumeration>>
AcademicLevelType
BS
PHD
MS
<<enumeration>>
TA_type
PartTime
FullTime
<<enumeration>>
TaskType
Lab
Recitation
Grading
1_*
Course
-preRequisites_List : List<Course>
-exams_List : List<Exam>
-department : Department
-coordinator : Coordinator
-sections_List : List<CourseSection>
-TAs_List : List<TA>
-courseID : int
-courseLevel : AcademicLevelType
-courseCode : int
+setPrereq_List(list : List<Course>) : void
+getPrereq_List() : List<Course>
+setExams_List(list : List<Exam>) : void
+getExams_List() : List<Exam>
+setDepartment(dep : Department) : void
+getDepartment() : Department
+setCoordinator(coord : Coordinator) : void
+getCoordinator() : Coordinator
+setSections_List(list : List<CourseSection>) : void
+getSections_List() : List<CourseSection>
+setTAs_List(list : List<TA>) : void
+getTAs_List() : List<TA>
+setCourseID(id : int) : void
+getCourseID() : int
+setCourseLevel(level : AcademicLevelType) : void
+getCourseLevel() : AcademicLevelType
+setCourseCode(id : int) : void
+getCourseCode() : int
Task
-TAs_List : List<TA>
-secIDList : List<int>
-courseID : int
-taskId : int
-workload : int
-isApproved : boolean
-duration : Event
-isTimePassed : boolean
-type : TaskType
+setTAs_List(list : List<TA>) : void
+getTAs_List() : List<TA>
+setSecID_List(list : List<Integer>) : void
+getSecID_List() : void
+setCourseID(id : int) : void
+getCourseID() : int
+setTaskID(id : int) : void
+getTaskID() : int
+setTaskType(type : TaksType) : void
+getTaskType() : TaskType
+setWorkload(amount : int) : void
+getWorkload() : int
+setIsApproved(state : boolean) : void
+getIsApproved() : boolean
+setIsTimePassed(state : boolean) : void
+getIsTimePassed() : boolean
1
0_*
TaskService
+createTask(task : Task) : boolean
+approveTask(task_id : int) : boolean
+rejectTask(task_id : int) : boolean
+checkIfTimePassed(task_id : int) : boolean
+editTask(task : Task) : boolean
+sendTask(task_id : int, instr_id : int) : void
+setTimeFinished() : void
Date
-year: int
-month: int
-day: int
-hour: int
-minute: int
+getFullInfo()
+setYear(year : int) : void
+getYear() : int
+setMonth(month : int) : void
+getMonth() : int
+setDay(day : int) : void
+getDay() : int
+setHour(hour : int) : void
+getHour() : int
+setMinute(min : int) : void
+getMinute() : int
Log
-id : long
-date : LocalDateTime
-commiters_id : int
-log_info : str
LogService
+createLog(id : int, date : Time, com_id : int, info : Str) : void
+sendLog() : boolean
+getLog(id : int) : Log
<<enumeration>>
Day
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
DailyDuties
-day: Day
-duties: list<DailyWork>
+setDay(day :Day) : void
+getDay() : Day
+setDuties(list : List<DailyWork>) : void
+getDuties() : List<DailyWork>
0_*
1
1
RequestService
+sendRequest(request : Request) : boolean
+createRequest(type : RequestType) : Request
+approveRequest(request_id : int) : boolean
+rejectRequest(request_id : int) : boolean
Faculty
-departmentList : List<Department>
-name : str
-faculty_id : int
+setDepartmentList(list : List<Department>) : void
+getDepartmentList() : List<Department>
+setName(name : Str) : void
+getName() : Str
+setFaculty_ID(id : int) : void
+getFaculty_ID() : int
1
1_*
Department
-courseList : List<Course>
-instructorList : List<Instructor>
-depOffice : DepartmentOffice
-TAs_List : List<TA>
-department_id : int
+setCourseList(list : List<Course>) : void
+getCourseList() : List<Course>
+setInstructorList(list : List<Instructor>) : void
+getInstructorList() : List<Instructor>
+setDepOffice(off : DepartmentOffice) : void
+getDepOffice() : DepartmentOffice
+setTAs_List(list : List<TA>) : void
+getTAs_List() : List<TA>
+setDepartment_ID(id : int) : void
+getDepartment_ID() : int
1
1_*
1
1_*
CourseSection
-students_List : List<Student>
-tasks_List : List<Task>
-inst : Instructor
-secID : int
-schedule : List<Event>
-classRoom : Classroom
+setStudents_List(list : List<Student>) : void
+getStudents_List() : List<Student>
+setTasks_List(list : List<Task>) : void
+getTasks_List() : List<Task>
+setInst(inst : Instructor) : void
+getInstructor() : Instructor
+setSchedule(list : List<Event>) : void
+getSchedule() : List<Event>
+setClassroom(room : Classroom) : void
+getCalssromm() : void
1
OpenVolunteerProctoringRequest
-exam : Exam
-requested_TAs : List<Pair<Pair<Faculty,Department>,Integer>
+setExam(exam : Exam) : void
+setRequested_TAs_List(list :List<Pair<Pair<Faculty,Department>,Integer>>) : void
+getExam() : Exam
+getRequested_TAs_List() : List<Pair<Pair<Faculty,Department>,Integer>
ProctorTAFromFacultiesRequest
-isInFaculty : boolean
-requestedTAs: List<Pair<Faculty,Department>,Integer>
-exam :Exam
+setIsInFaculty(state : boolean) : void
+getIsInFaculty() : boolean
+setRequestedTAs(list : List<Pair<Faculty,Department>,Integer>) : void
+getRequestedTAs() : List<Pair<Faculty,Department>,Integer>
Request
-commiters_id : int
-recievers_id : int
-isApproved : boolean
-sent_time : Date
-type : RequestType
-request_id : int
+setCommiters_ID(id : int) : void
+getCommiters_ID() : int
+setReceivers_ID(id : int) : void
+getReceivers_ID() : int
+setIsApproved(state : boolean) : void
+getIsApproved() : boolean
+setSent_Time(date : Date) : void
+getSent_Time() : Date
+setType(type : RequestType) : void
+getType() : RequestType
+setRequest_ID(id : int) : void
+getRequest_ID() : int
<<enumeration>> RequestType
Swap
SwapEnable
Leave
TransferProctoring
VolunteerProctoring
ProctorTAFromFaculties
TAFromOtherFaculty
TAInFaculty
TAFromOtherFacultyRequest
-requestedTAs : List<Pair<Faculty,Department>,Integer>
-exams_list : List<Exam>
+setExamsList(list : List<Exam>) : void
+setRequestedTAs(list : List<Pair<Faculty,Department>,Integer>) : void
+getRequestedTAs() : List<Pair<Faculty,Department>,Integer>
+getExamsList() : List<Exam>
SwapRequest
-senders_exam : Exam
-receivers_exam : Exam
+getSenders_Exam() : Exam
+setReceivers_Exam(exam : Exam) : void
+setSenders_Exam(exam : Exam) : void
+getSenders_Exam() : Exam
SwapEnableRequest
-exams_for_swap : Pair<Exam,Exam>
-faculty_department_id : Pair<Integer,Integer>
+setExams_for_swap() : Pair<Exam,Exam>) : void
+setFaculty_department_id(id : Pair<Integer,Integer>) : void
+getExams_for_swap() : Pair<Exam,Exam>
+getFaculty_department_id() : Pair<Integer,Integer>
TAInFacultyRequest
-requestedTAs_amount : int
-exam : Exam
+setRequestedTAs_amount(amount : int) : void
+getRequestedTAs_amount() : int
+setExam(exam : Exam) : void
+getExam() : Exam
TransferProctoringRequest
-examToTransfer : Exam
+setExamToTransfer(exam : Exam) : void
+getExamToTransfer() : Exam
MakeLeaveRequest
-reason_info : Str
-duration : Event
-file : File
+setReasonInfo() : Str
+setDuration() : Event
+setFile() : File
0_*
DailyWork
-duration: Event
-workType : WorkType
+setDuration(dur : Event) : void
+getDuration() : Event
+setWorkType(type : WorkType) : void
+getWorkType() : WorkType
<<enumeration>>
WorkType
Lab
Recitation
OfficeHour
1_*
Event
-startTime: Date
-finishTime: Date
+setStartTime(date : Date) : void
+getStartTime() : Date
+setFinishTime(date : Date) : void
+getFinishTime() : Date
<<enumeration>>
ExamType
Midterm
Final
Exam
-duration : Event
-subExamList : List<ExamRoom>
-type : ExamType
-exam_ID : int
+setDuration(dur : Event) : void
+getDuration() : Event
+setSubExamLsit(list : List<ExamRoom>) : void
+getSubExamList() : List<ExamRoom>
+setType(type : ExamType) : void
+getType() : ExamType
+setExam_ID(id : int) : void
+getExam_ID() : int
1
ExamService
+createExam(exam : Exam) : boolean
+editExam(exam : Exam) : boolean
1_*
ExamRoom
-classroom : Classroom
-TAs_List : ListTA>
-students_List : List<Student>
-isApproved : boolean
-workload : int
-examroomID : int
+setClassroom(room : Classroom) : void
+getClassroom() : Classroom
+setTAs_List(list : List<TA>) : void
+getTAs_List() : List<TA>
+setStud_List(list : List<Student>) : void
+getStud_List() : List<Student>
+setIsApproved(state : boolean) : void
+getIsApproved() : boolean
+setWorkload(amount : int) : void
+getWorkload() : int
+setExamroomID(id : int) : void
+getExamroomID() : int
1_*
1_*
Student
-courseTaken : List<Course>
-name : str
-surname : str
-studentID : int
+setCourseTaken(list : List<Course>) : void
+getCourseTaken() : List<Course>
+setName(name : Str) : void
+getName() : Str
+setSurname(surname : Str) : void
+setStudentID(id : int) : void
+getStudentID() : int
Classroom
-name : str
-capacity : int
+setName(name : Str) : void
+getName() : Str
+setCapacity(cap : int) : void
+getCapacity() : int

Link for the Class Diagram

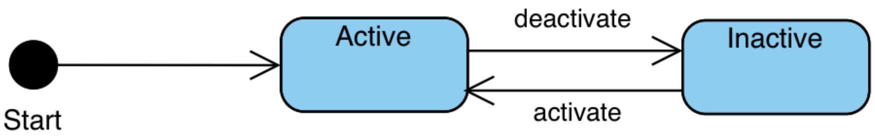
3. State Machine Diagrams

Link for all of the state machine diagrams:
https://online.visual-paradigm.com/share.jsp?id=333439313030372d3133

3.1 State Machine Diagram for the Login System


password changed
Not all fields are filled
Login
forgot password
Change Password
do /
sendVerificationEmail()
Start
ID AND password fields are filled
credentials are invalid OR user is not active / displayErrorMessage()
log out
Authentication Check
Main Page
credentials are valid and user is active

3.2 State Machine Diagram of an Online Status For User Saved in The System



3.3 State Machine Diagram of a Swap Request

4. Activity Diagrams

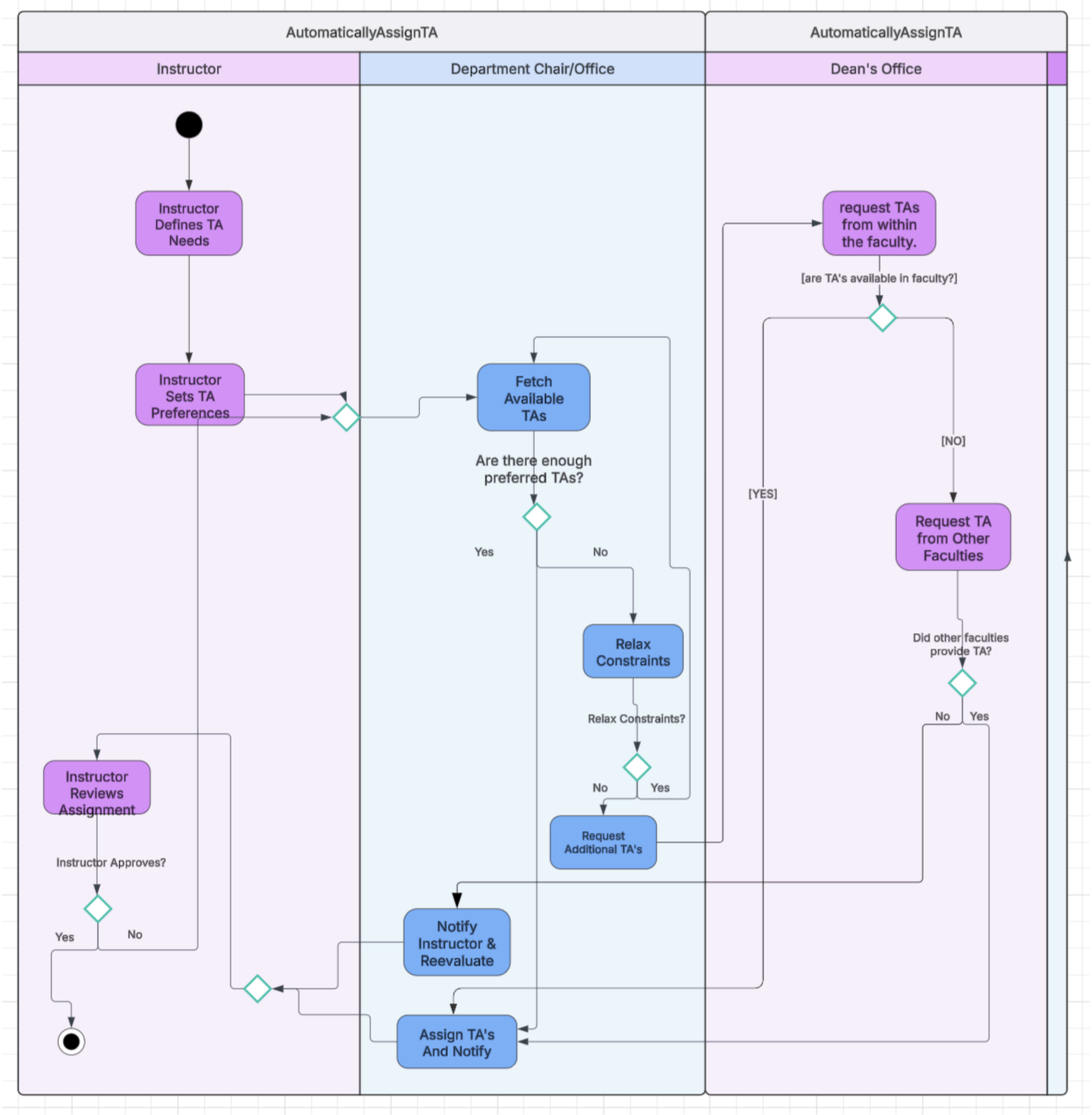

AutomaticallyAssignTA
AutomaticallyAssignTA
Instructor
Department Chair/Office
Dean's Office
Instructor Defines TA Needs
Instructor Sets TA Preferences
Fetch Available TAs
Are there enough preferred TAs?
Yes
No
Relax Constraints
Relax Constraints?
No
Yes
Request Additional TA's
request TAs from within the faculty.
[are TA's available in faculty?]
[YES]
[NO]
Request TA from Other Faculties
Did other faculties provide TA?
No
Yes
Instructor Reviews Assignment
Instructor Approves?
Yes
No
Notify Instructor & Reevaluate
Assign TA's And Notify

Diagram 1: The activity diagram for automatically assigning the TA process

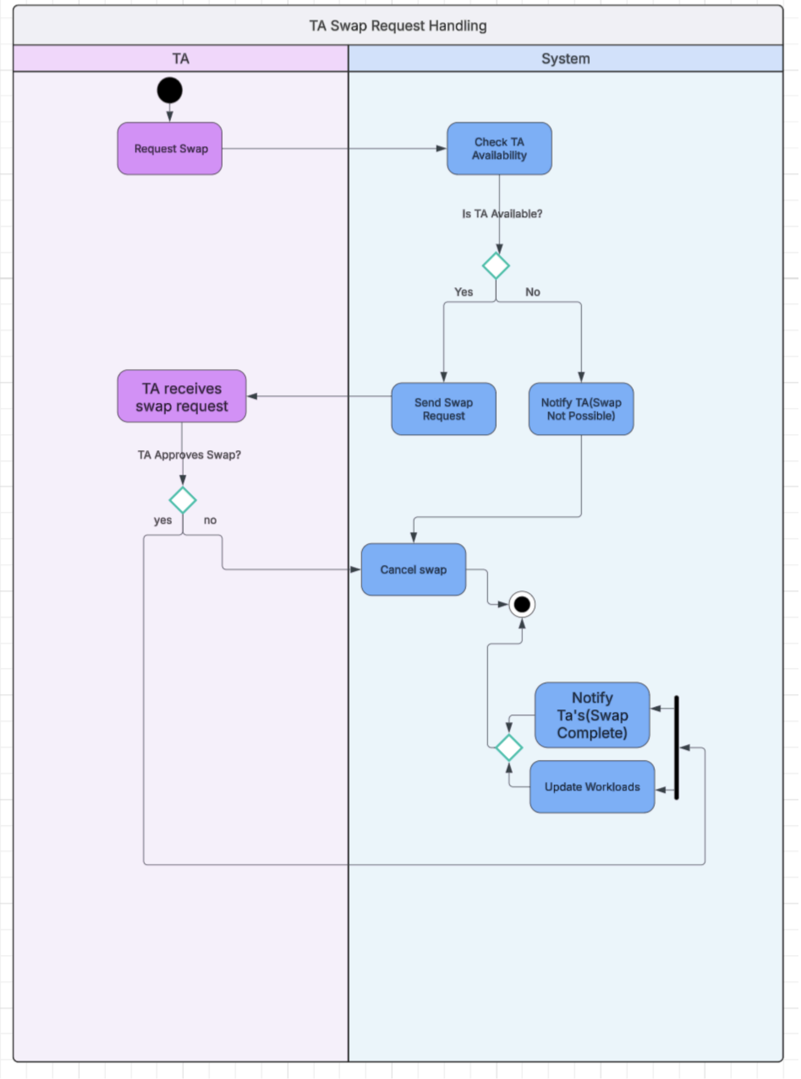


Diagram 2: The activity diagram of TA swap request handling process

5. Sequence Diagrams

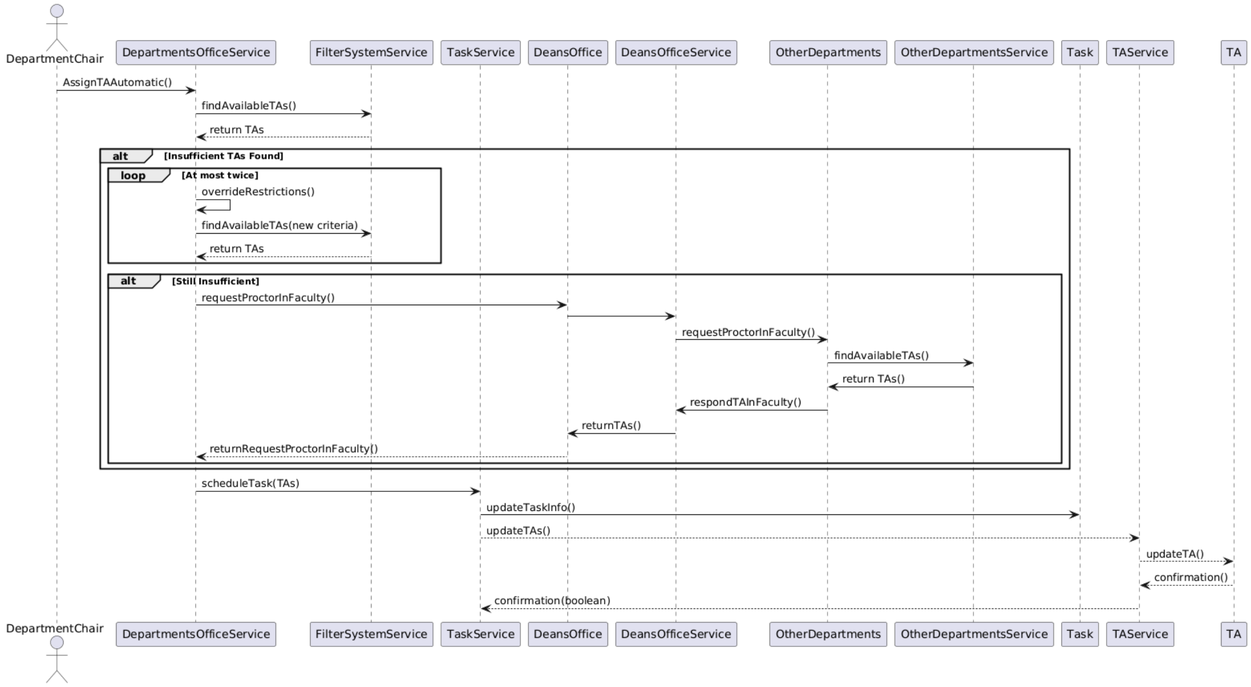
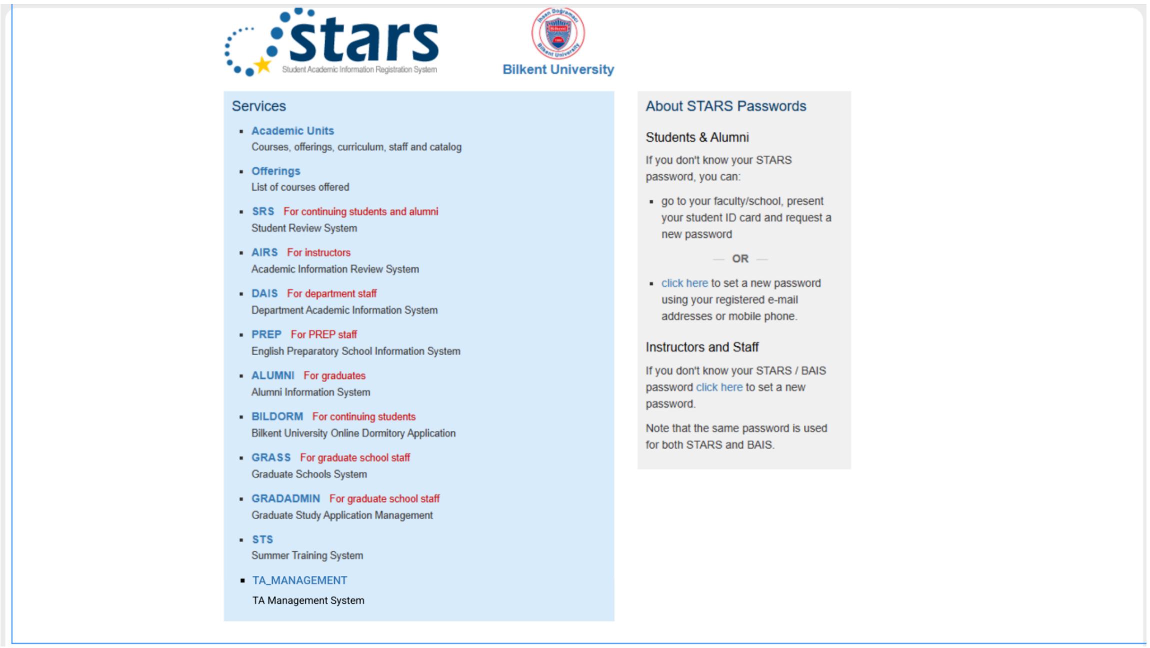


Diagram 1 : Sequence diagram of AssignTAAutomatic

https://www.planttext.com?text=fLNDReCm3Bxp53wsXxu0XwdHfaezRMfvWJI4DLf8MEAqwjilHfG21OgqIFma_lpP_cB2rYDt64gTS87MmOUiuhoK1j_FNBc4PageYXlisFndKIWXZz9TugC7sYcDqXrl7cKvPi-u_xcl3zYvQNYdriVuFl4iNIUpMMCRE1dK2PktoV9IODwhRvCn5j2MFBAlriaKcfX6D4YXK2Pd5wuqFscPCHyvHd2mcSt4IGpEG7IFDN2Di3S-44H9qfPvsDbWyWJYqDPMm11Awn7mIdL3C-QIdmvkBz8vbSk3zEYKG6KDLR64T9aYHbv1E1MHYZzu_wKI4SX6aFf6eXrHQTsJRg4oRITIZDyGTVXobhwRlTbn4JJUUdgqy87tdJE5CN0duh1_buGT-epnp3Nb5C-BjEwAlyZakKvozfKrUSPUg3Yz5Hly7tlcfRSl2_heC2ivDHTTimT2vz1BmOkpp4EKEowkOjRhf6E-erC8LSwn1krDOLS35C5OIzbWQmNQvGRqiDR6eQkmfb2EJZLhdhsxUVUG9sjrr74TZw5jB3_-GFu0

6.Mockups



stars

Student Academic Information Registration System

Bilkent University

Services

- **Academic Units**
  Courses, offerings, curriculum, staff and catalog
- **Offerings**
  List of courses offered
- **SRS** For continuing students and alumni
  Student Review System
- **AIRS** For instructors
  Academic Information Review System
- **DAIS** For department staff
  Department Academic Information System
- **PREP** For PREP staff
  English Preparatory School Information System
- **ALUMNI** For graduates
  Alumni Information System
- **BILDORM** For continuing students
  Bilkent University Online Dormitory Application
- **GRASS** For graduate school staff
  Graduate Schools System
- **GRADADMIN** For graduate school staff
  Graduate Study Application Management
- **STS**
  Summer Training System
- TA_MANAGEMENT
  TA Management System

About STARS Passwords

Students & Alumni

If you don't know your STARS password, you can:

- go to your faculty/school, present your student ID card and request a new password

— OR —

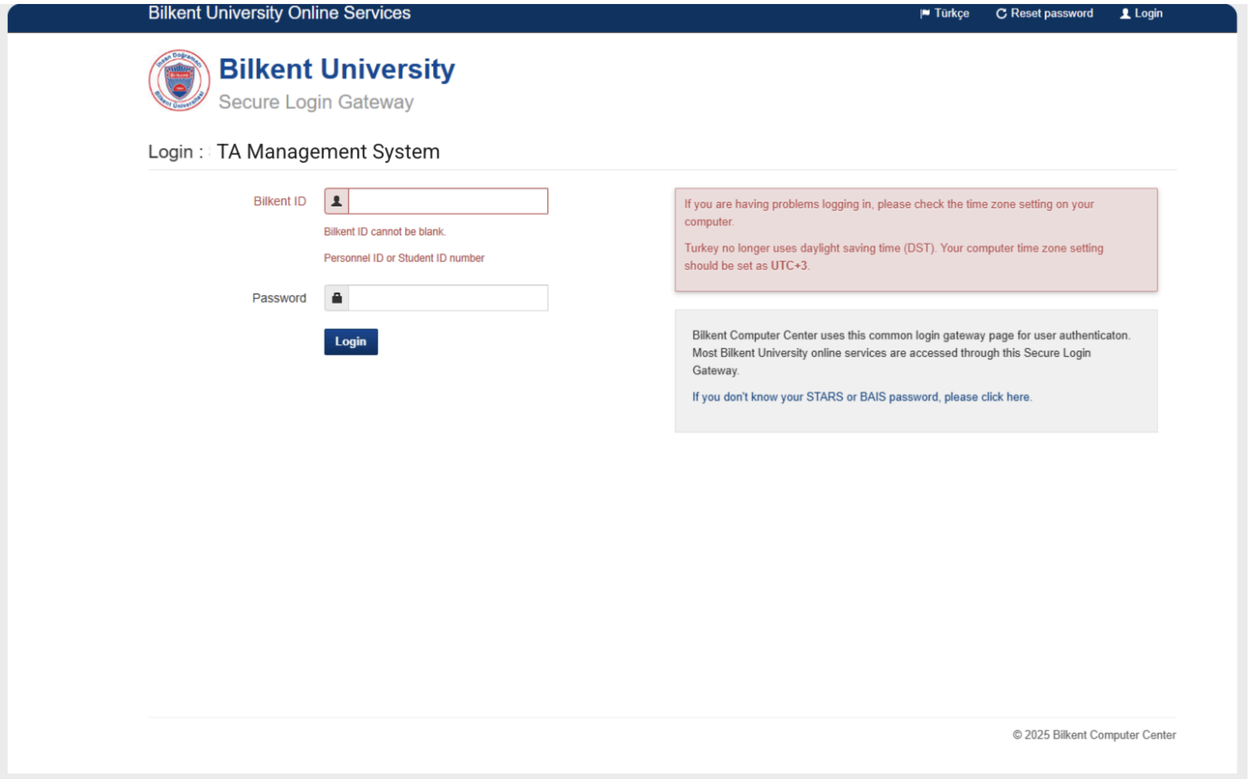
- click here to set a new password using your registered e-mail addresses or mobile phone.

Instructors and Staff

If you don't know your STARS / BAIS password click here to set a new password.

Note that the same password is used for both STARS and BAIS.

Figure 1: First Screen



Bilkent University Online Services

Bilkent University

Secure Login Gateway

Login : TA Management System

Bilkent ID

Bilkent ID cannot be blank.

Personnel ID or Student ID number

Password

If you are having problems logging in, please check the time zone setting on your computer.

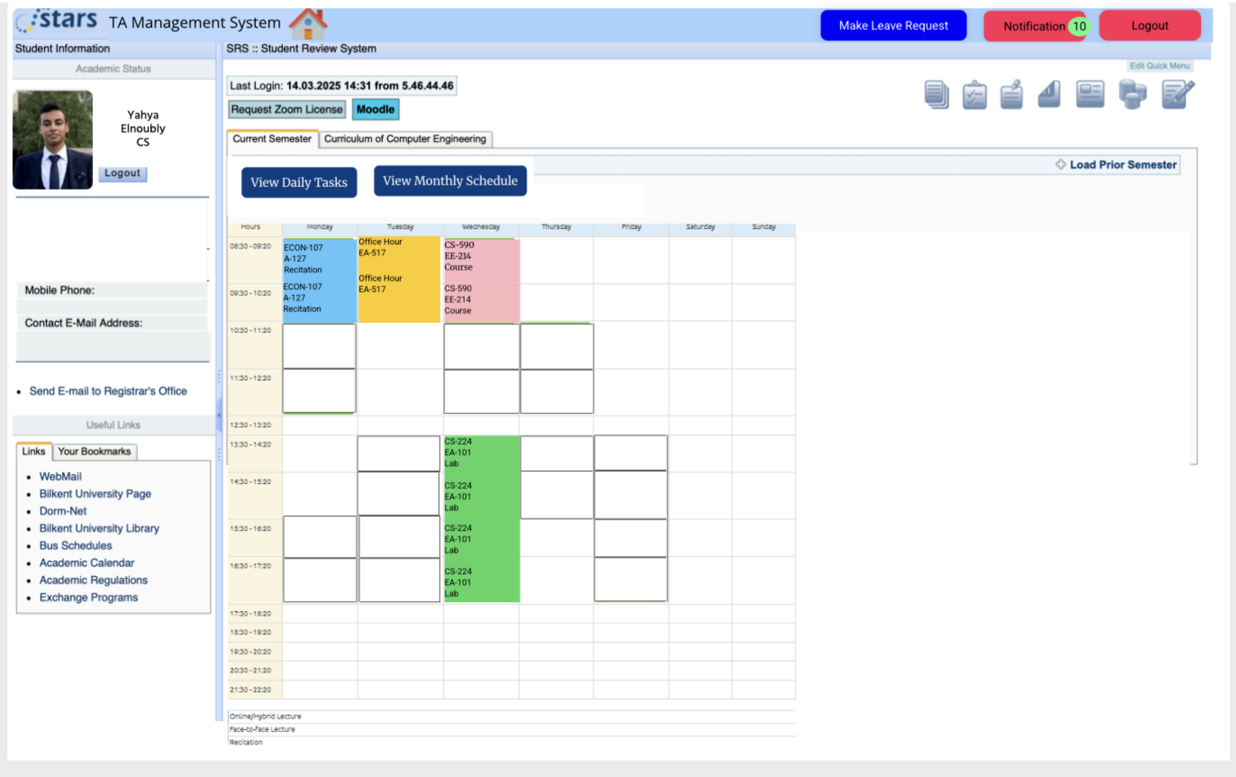
Turkey no longer uses daylight saving time (DST). Your computer time zone setting should be set as UTC+3.

Bilkent Computer Center uses this common login gateway page for user authenticaton. Most Bilkent University online services are accessed through this Secure Login Gateway.

If you don't know your STARS or BAIS password, please click here.

© 2025 Bilkent Computer Center

Figure 2: Login Page of the TA Management System



Student Information

Academic Status

Yahya
Elnoubly
CS

Mobile Phone:

Contact E-Mail Address:

- Send E-mail to Registrar's Office

Useful Links

- WebMail
- Bilkent University Page
- Dorm-Net
- Bilkent University Library
- Bus Schedules
- Academic Calendar
- Academic Regulations
- Exchange Programs

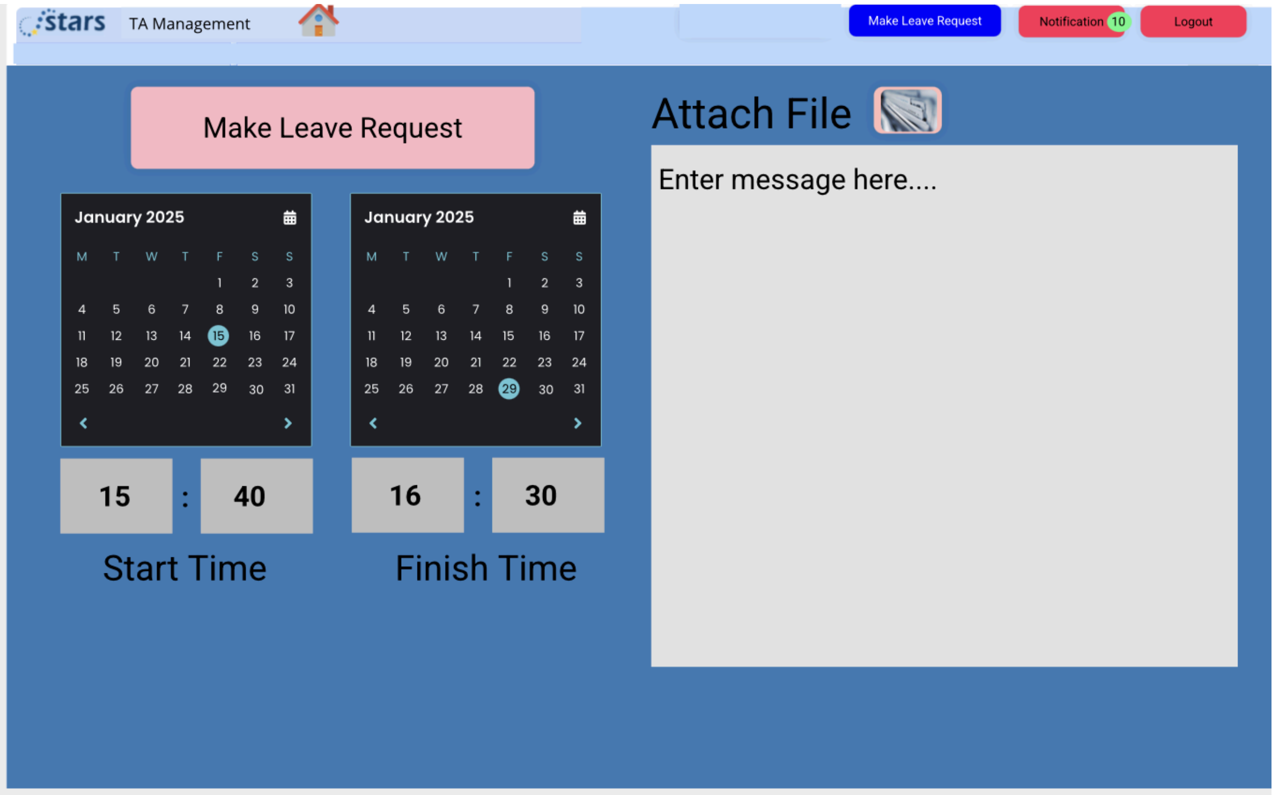
SRS :: Student Review System

Last Login: **14.03.2025 14:31 from 5.46.44.46**

Hours	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
08:30 - 09:20	ECON-107 A-127 Recitation	Office Hour EA-517	CS-590 EE-214 Course				
09:30 - 10:20	ECON-107 A-127 Recitation	Office Hour EA-517	CS-590 EE-214 Course				
10:30 - 11:20							
11:30 - 12:20							
12:30 - 13:20							
13:30 - 14:20			CS-224 EA-101 Lab				
14:30 - 15:20			CS-224 EA-101 Lab				
15:30 - 16:20			CS-224 EA-101 Lab				
16:30 - 17:20			CS-224 EA-101 Lab				
17:30 - 18:20							
18:30 - 19:20							
19:30 - 20:20							
20:30 - 21:20							
21:30 - 22:20							

Online/Hybrid Lecture
Face-to-face Lecture
Recitation

Figure 3: TA Main Page



January 2025

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

15 : 40

Start Time

January 2025

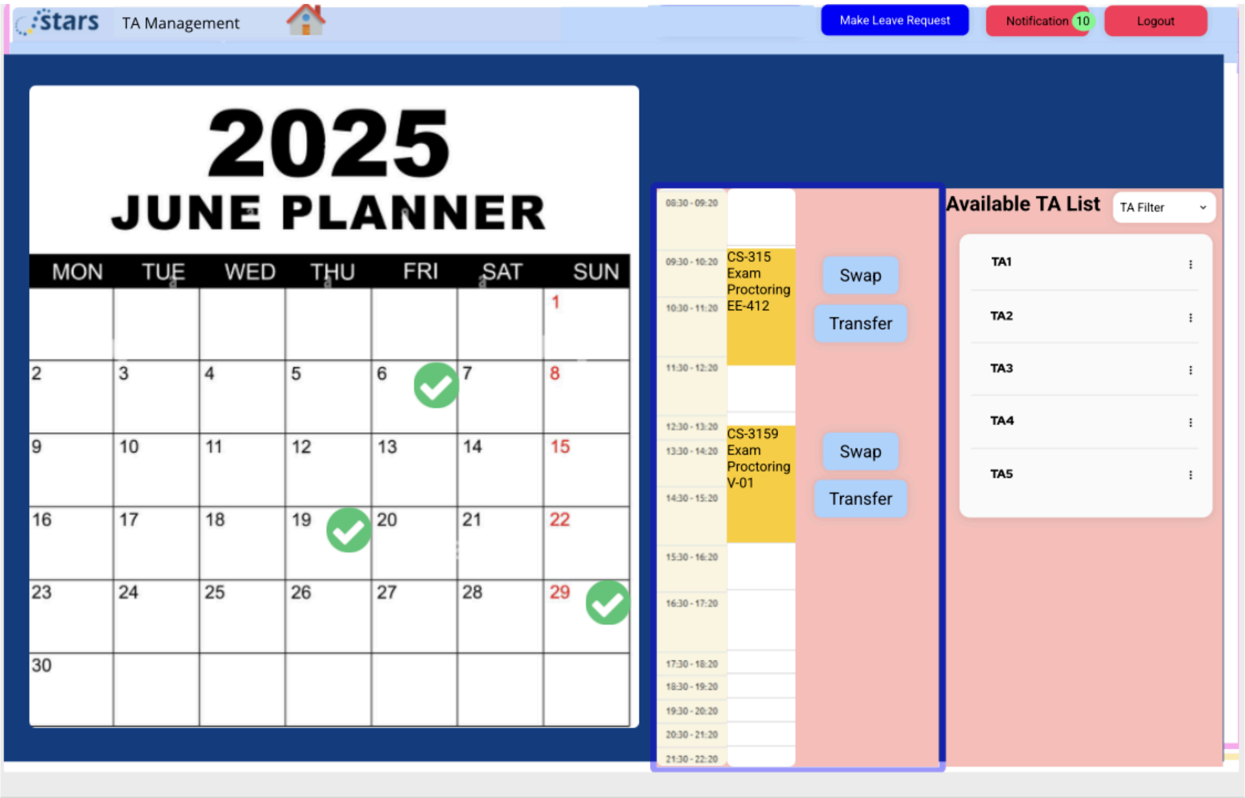
M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

16 : 30

Finish Time

Enter message here....

Figure 4: Leave Request Form Page



2025
JUNE PLANNER

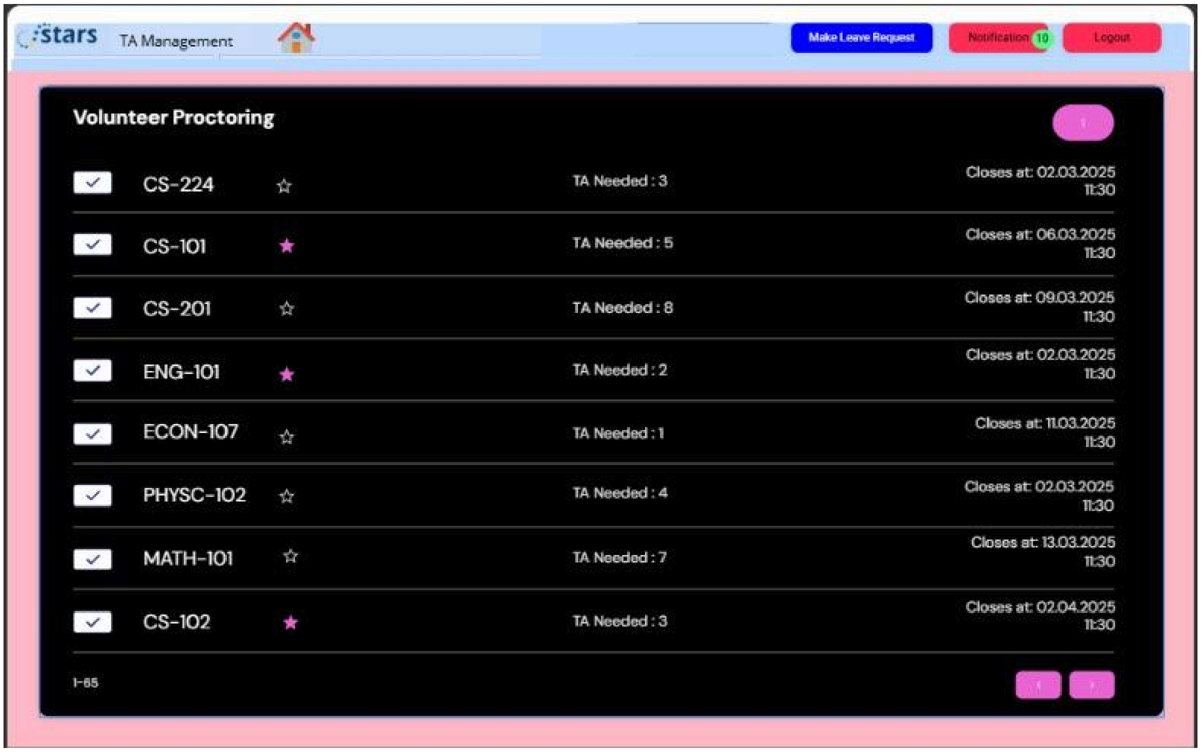
MON	TUE	WED	THU	FRI	SAT	SUN
						1
2	3	4	5	6 ✓	7	8
9	10	11	12	13	14	15
16	17	18	19 ✓	20	21	22
23	24	25	26	27	28	29 ✓
30						

08:30 - 09:20	
09:30 - 10:20	CS-315 Exam Proctoring EE-412
10:30 - 11:20	
11:30 - 12:20	
12:30 - 13:20	CS-3159 Exam Proctoring V-01
13:30 - 14:20	
14:30 - 15:20	
15:30 - 16:20	
16:30 - 17:20	
17:30 - 18:20	
18:30 - 19:20	
19:30 - 20:20	
20:30 - 21:20	
21:30 - 22:20	

Available TA List

- TA1
- TA2
- TA3
- TA4
- TA5

Figure 5: Monthly Schedule of TA

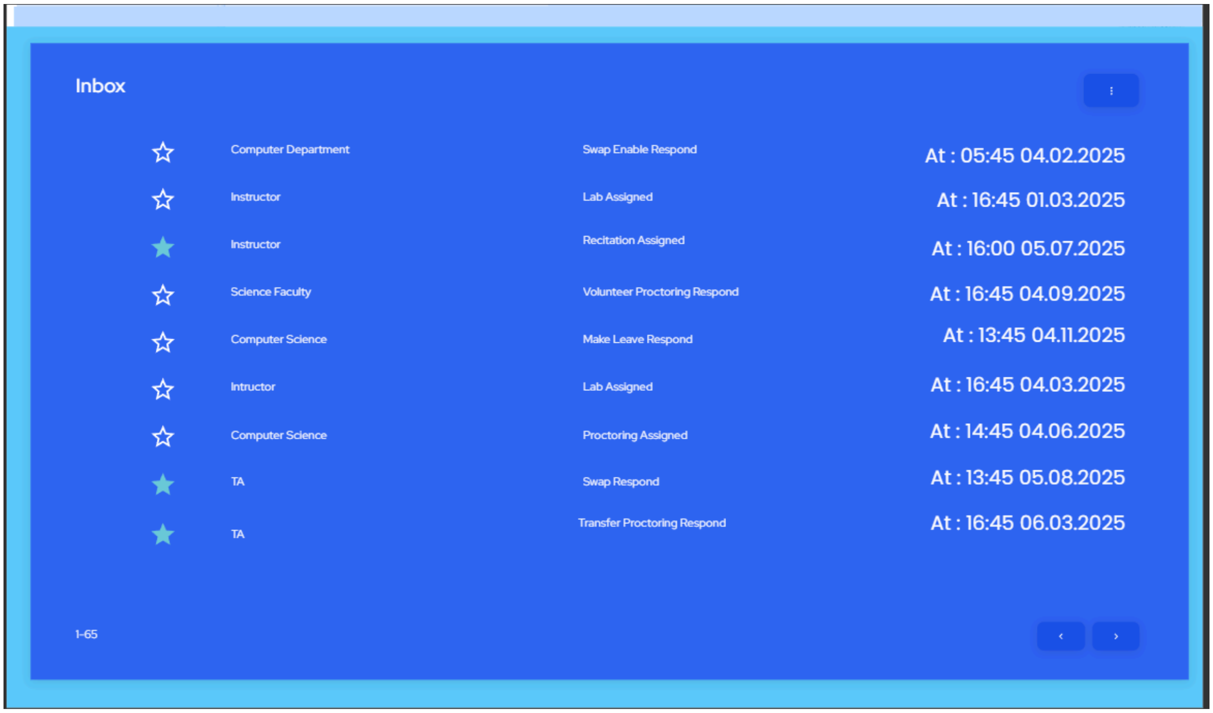


Volunteer Proctoring

☑	CS-224	☆	TA Needed : 3	Closes at: 02.03.2025 11:30
☑	CS-101	★	TA Needed : 5	Closes at: 06.03.2025 11:30
☑	CS-201	☆	TA Needed : 8	Closes at: 09.03.2025 11:30
☑	ENG-101	★	TA Needed : 2	Closes at: 02.03.2025 11:30
☑	ECON-107	☆	TA Needed : 1	Closes at: 11.03.2025 11:30
☑	PHYSC-102	☆	TA Needed : 4	Closes at: 02.03.2025 11:30
☑	MATH-101	☆	TA Needed : 7	Closes at: 13.03.2025 11:30
☑	CS-102	★	TA Needed : 3	Closes at: 02.04.2025 11:30

1-65

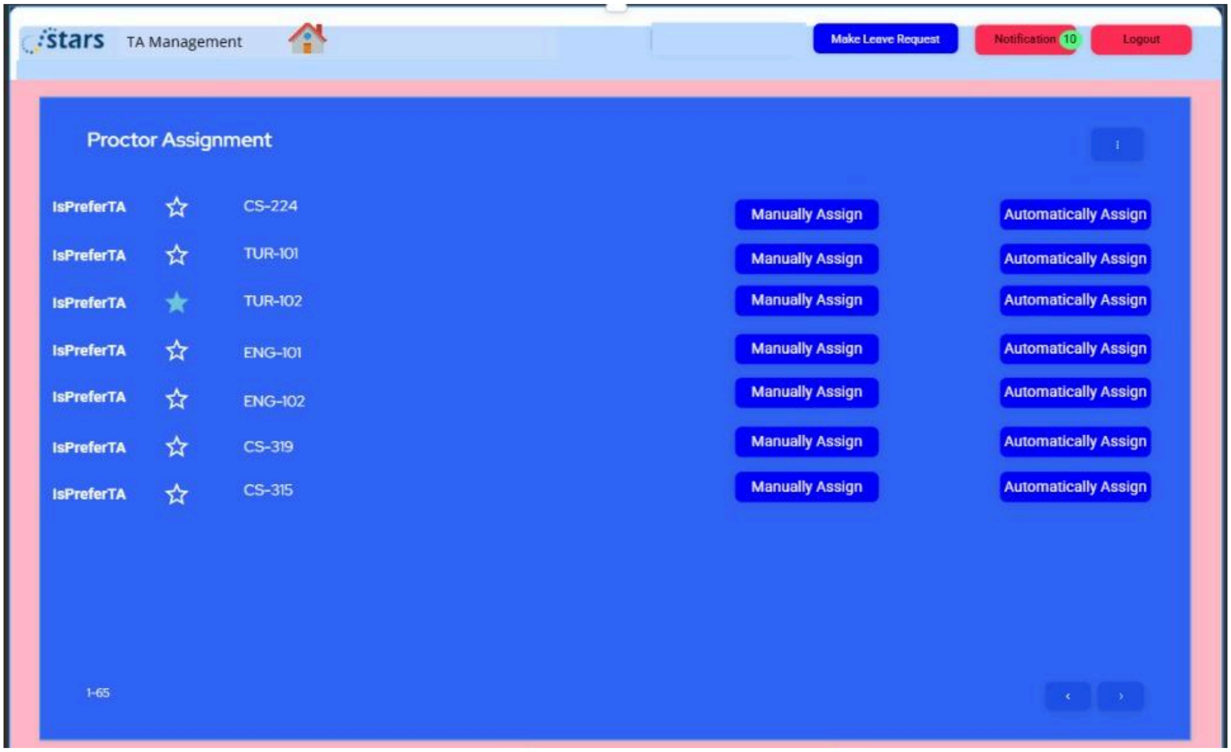
Figure 6: Volunteer Proctoring Announcements List


Inbox

☆	Computer Department	Swap Enable Respond	At : 05:45 04.02.2025
☆	Instructor	Lab Assigned	At : 16:45 01.03.2025
★	Instructor	Recitation Assigned	At : 16:00 05.07.2025
☆	Science Faculty	Volunteer Proctoring Respond	At : 16:45 04.09.2025
☆	Computer Science	Make Leave Respond	At : 13:45 04.11.2025
☆	Intructor	Lab Assigned	At : 16:45 04.03.2025
☆	Computer Science	Proctoring Assigned	At : 14:45 04.06.2025
★	TA	Swap Respond	At : 13:45 05.08.2025
★	TA	Transfer Proctoring Respond	At : 16:45 06.03.2025

1-65

Figure 7:

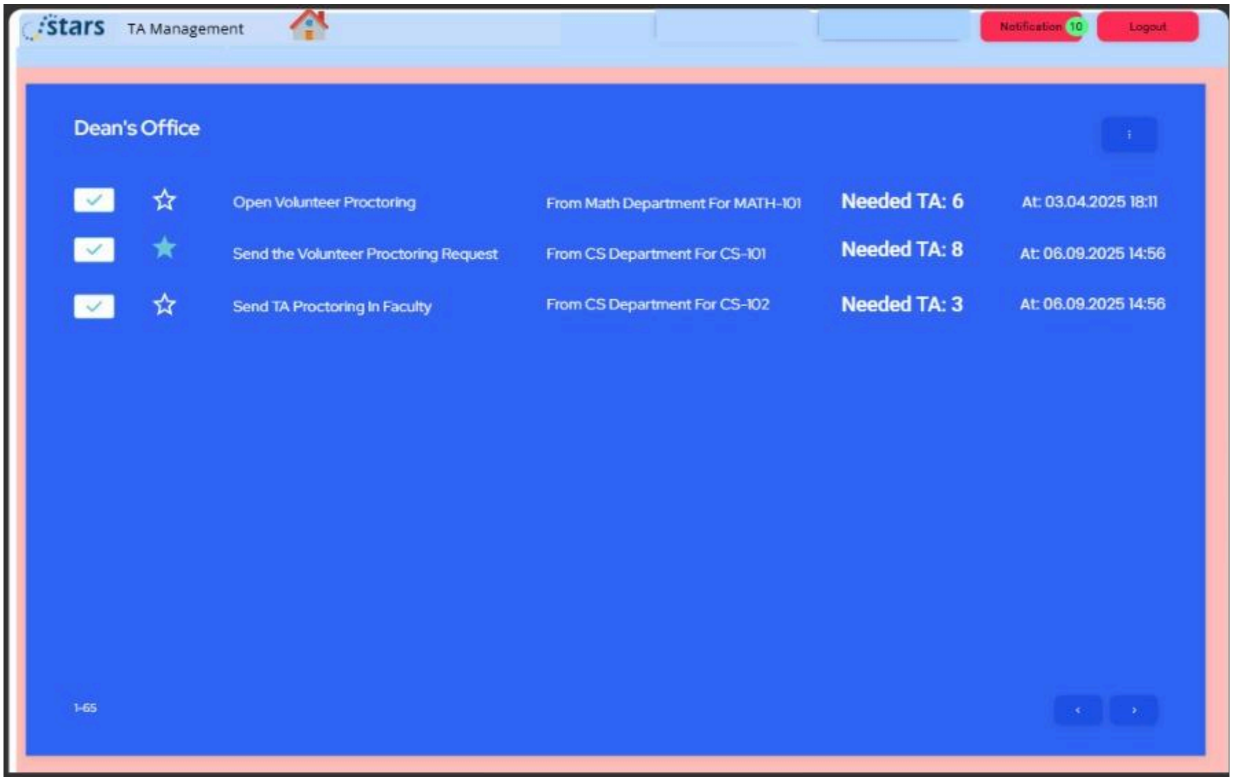


Proctor Assignment

IsPreferTA	☆	CS-224	Manually Assign	Automatically Assign
IsPreferTA	☆	TUR-101	Manually Assign	Automatically Assign
IsPreferTA	★	TUR-102	Manually Assign	Automatically Assign
IsPreferTA	☆	ENG-101	Manually Assign	Automatically Assign
IsPreferTA	☆	ENG-102	Manually Assign	Automatically Assign
IsPreferTA	☆	CS-319	Manually Assign	Automatically Assign
IsPreferTA	☆	CS-315	Manually Assign	Automatically Assign

1-65

Figure 8: Proctoring Assignments List page in the Dean's Office's

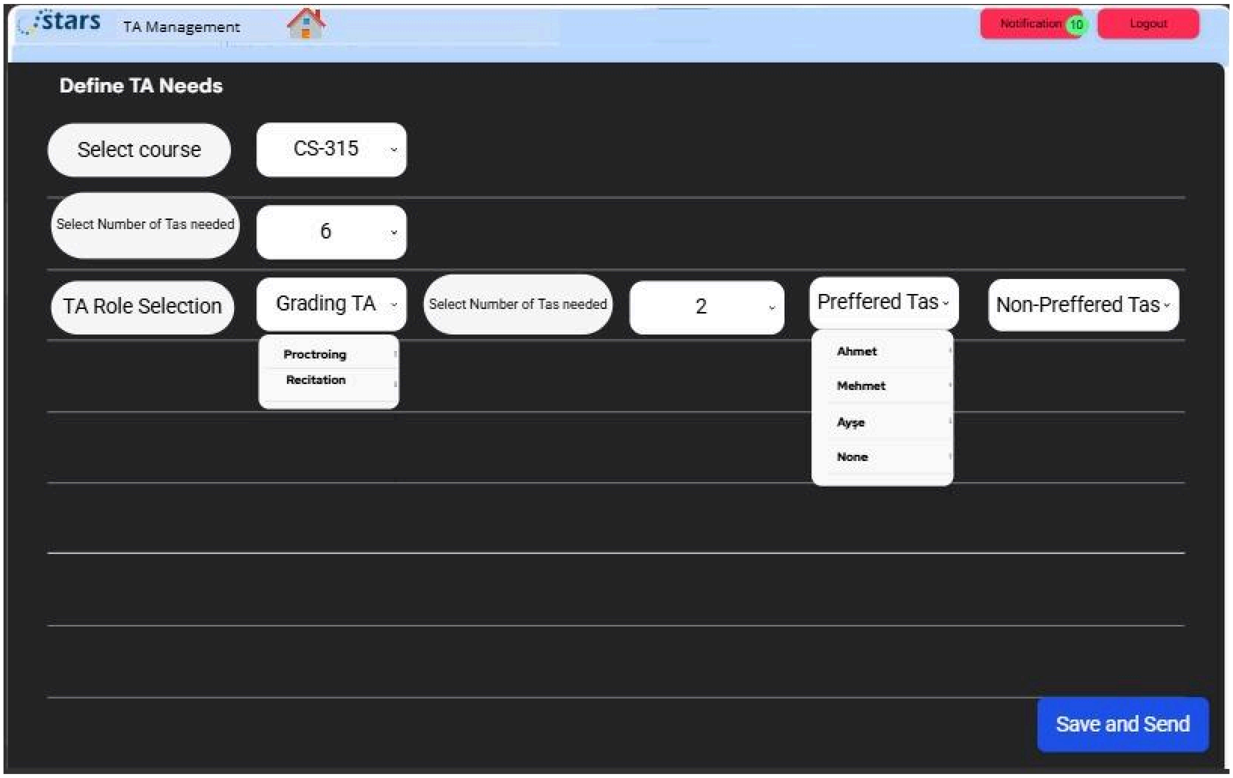


Dean's Office

☑	☆	Open Volunteer Proctoring	From Math Department For MATH-101	Needed TA: 6	At: 03.04.2025 18:11
☑	★	Send the Volunteer Proctoring Request	From CS Department For CS-101	Needed TA: 8	At: 06.09.2025 14:56
☑	☆	Send TA Proctoring In Faculty	From CS Department For CS-102	Needed TA: 3	At: 06.09.2025 14:56

1-65

Figure 9: TA Proctoring Requests List Page



Define TA Needs

Select course CS-315

Select Number of Tas needed 6

TA Role Selection Grading TA

Select Number of Tas needed 2

Figure 10: Preferring Proctor TA