CS 319 - Object-Oriented Software Engineering

Deliverable 1

Spring 2025

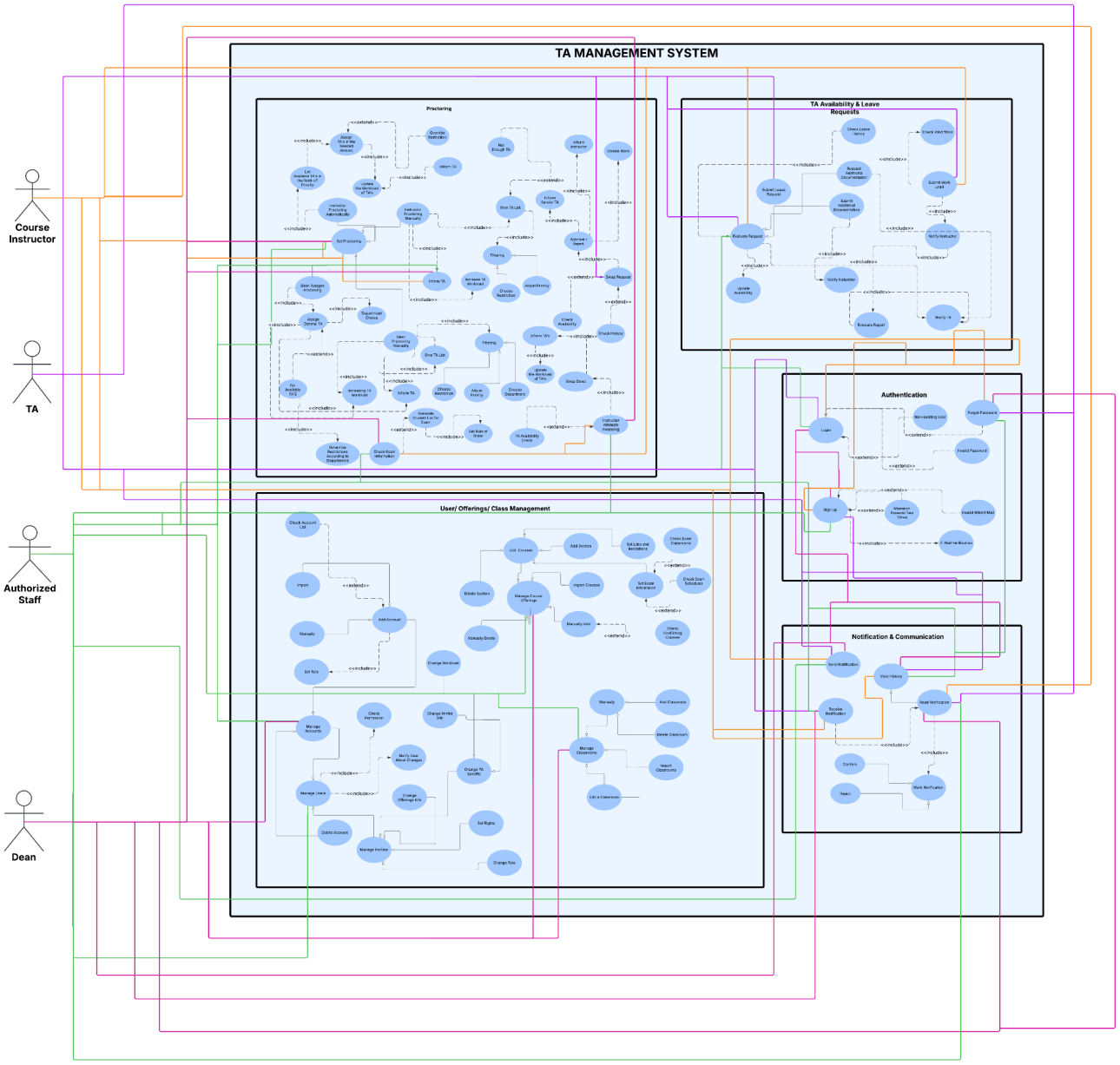
Team 12

Ahmet Kenan Ataman <22203434>
Berfin Örtülü <21802704>
Erdem Uğurlu <22203391>
Mehmet Emre Şahin <22201765>
Gülferiz Bayar <21901442>

TABLE OF CONTENT

1.Use Case Diagram

For high quality interactive version (i.e. association lines can be tracked easily) refer to https://lucid.app/lucidchart/a5d4ceef-6b2f-4613-9c48-524f927de8a3/edit?viewport_loc=-2806%2C-4371%2C10110%2C5857%2C0_0&invitationId=inv_6855d0ab-f808-4403-9c52-7a906aa160ec or the attached UML.pdf fil



2. Use Case Textual Diagram

2.1 User/Offerings/Classrooms Management

Use Case Name: `Manage Users`

Flow of Events: The authorized user encounters the possible changeable entities, which have “users” main title. This “users” consists of the accounts and profiles. Clicking one of the buttons, “Manage Accounts” or “Manage Profiles” the authorized user reaches those pages. If the authorized user changes their mind, they click the “Cancel” button and are directed to the previous page.

Entry Conditions: The user has logged in, clicked on the button “Manage Users”, and they have the permission to manage users.

Exit Conditions: The authorized user is either directed to the proper page according to their choice or they go back by button “Back”.

Use Case Name: `Check Permission` (included by Manage Users)

Flow of Events: The system simply checks if the current user has the permission to reach this functionality. The authorized users are the mentioned user in Manage Users use case.

Entry Conditions: The user has logged in to the system and clicked to the “Manage Users” option.

Exit Conditions: The system triggers a warning with an explanatory message by the Notify User use case if the user does not have the required permission. Otherwise, they are directed to the page where they can manage users.

Use Case Name: `Notify User About Changes` (included by Manage Users)

Flow of Events: If any change is made to an account, notify the account’s owner via the system and/or mail about the changes applied.

Entry Conditions: Any change to an account has been made.

Exit Conditions: The account owner has been sent an informing message.

Use Case Name: `Manage Accounts`

Flow of Events: The user can choose whether to Add Account or Delete Account options. If the Add Account option is clicked, Add Account use case is working, also the same for Delete Account.

Entry Conditions: The user chose Manage Accounts in the Manage Users page.

Exit Conditions: The user reaches to either Add Account or Delete Account page.

Use Case Name: `Add Account` (Inherited from Manage Accounts)

Flow of Events: The user sees a pop-up screen that has two options: Add Manually and Import Accounts. Add Manually choice invokes the manual addition window. This window is explained in the use case Add Account Manually. They can go back by clicking on the “Cancel” button. The other option is Import, which allows the user to import accounts from a properly created Excel file. Any format mismatching or overlapping accounts will be reported to the user as a warning.

Entry Conditions: The user has clicked on the “Add Account” button.

Exit Conditions: The user either added at least one new account or none or clicked on the “Cancel” button and directed to the previous page.

Use Case Name: `Add Account Manually` (Inherited from Add Account)

Flow of Events: If the user clicks on the “Cancel” button, they are directed to the previous page. Otherwise, a manual addition window is opened. This window shows a form that wants the user to enter the needed information to create the account. This key information includes these: Name, Bilkent mail address, password (is assumed to be used only for the first time that a new account’s owner enters the system), role, and the academic status. Roles include Admin, Dean’s office, department secretary, instructor, and TA. Academic status might differ for TA’s depending on their current academic state like Master’s, PhD and etc. If the user provided the required information and the account is not already existing, a success message is shown to the user and the changes are permanently applied to the system.

Entry Conditions: The user clicks on the “Add Manually” button in the Add Account page’s pop-up.

Exit Conditions: The user either added at least one new account manually or non, clicked on the “Cancel” button and directed to the Add Accounts page again.

Use Case Name: `Import Accounts` (Inherited from Add Accounts)

Flow of Events: The user can go back by the “Cancel” button. Otherwise, in the window that pops, they are asked to upload an Excel file. If the file does not have a proper format or it has any overlapping (already existing user) a proper warning message is shown and no change in the accounts are made. If the importing is successful, a window lists the read accounts to the user. They have two options (buttons): “Save” and “Discard”. Again, discard does not change any data in the system, if the user clicks on the “Save” button, the changes are saved and applied to the system and a success message is shown to the user.

Entry Conditions: The user clicks on the “Import Accounts” button.

Exit Conditions: If the user successfully imports new accounts and clicks “Save”, the new accounts are permanently saved in the system and a confirmation message is displayed. If the file format is incorrect or contains overlapping data, the system shows an appropriate warning message and no changes are made. If the user clicks “Cancel”, the operation is aborted with no changes to the system, and the user is redirected to the Add Accounts page.

Use Case Name: `Check Accounts List` (extends Add Account)

Flow of Events: The system checks whether the account that is tried to be added is already existing. If the mail address is used for an existing account, a related warning pop-up is triggered and the adding process is terminated. If the account has the same name with an existing account, a related warning is shown to the user which does not terminate the process, and the user can add that account in this case. If the account is brand new, nothing is seen by the user, and the system does not try to abolish the process.

Entry Conditions: The user has tried to add a new account.

Exit Conditions: The system either invokes an error, displays a message, and abolish the process, or the user can add the account.

Use Case Name: `Delete Account` (inherited from Manage Accounts)

Flow of Events: The Delete Account page is opened. User can go back (to the Manage Accounts page) by clicking on the “Cancel” button. Otherwise, the user chooses account(s) that they want to delete. At the top of the window, a search bar, filtering options, and sorting options can be used to find the user quickly. When the user chooses all the accounts to be deleted, they can click on the “Delete” button. A pop-up window that shows the chosen accounts with all their information is opened and if the user approves it by clicking on the “Delete” button, the system deletes those accounts permanently and an explanatory success message is shown. Then the user is directed to the Manage Accounts page. If the user chooses “Cancel”, no change is made in the system and the window is closed, the user is still in the Delete Account page.

Entry Conditions: The user has clicked on the “Delete Account” button in the Manage Accounts page.

Exit Conditions: Either chosen accounts are deleted or no change was made to the system. The user is directed to the Manage Accounts page.

Use Case Name: `Manage Profiles` (inherited from Manage Users)

Flow of Events: The user is directed to the Manage Profiles page. This page shows the current accounts. If the user chooses an account, its profile page, which has the options below, is shown. They can go back by clicking on the “Cancel” button.

Entry Conditions: The user has clicked on the “Manage Profiles” button.

Exit Conditions: The user clicks the shown buttons.

Use Case Name: `Change Role` (inherited from Manage Users)

Flow of Events: The user sees the role of the chosen account. They can change their role in the pop-up window. When they click on “Change” a confirming message is shown and if they click on the “Change” again the role of the chosen account is changed in the system and an explanatory message is shown. The user can cancel the process by the “Cancel” button at each step.

Entry Conditions: The user clicked on “Change Role” in an account’s profile.

Exit Conditions: Either the chosen account's role is changed or no change is made. The user clicks on "Cancel" button and is directed to the account's profile page.

Use Case Name: `Set Rights` (inherited from Manage Profiles)

Flow of Events: The user edits the rights of the chosen account. They see a pop-up window that includes the list of rights and they can put a check mark on them, or delete check marks. When they click on "Done" button, a message is shown that asks if the user is sure. If they click on "Apply" the changes are made to the accounts in the system. They can click on "Cancel" at each step to go one page back.

Entry Conditions: The user has clicked on the "Set Rights" button.

Exit Conditions: The user changes some accounts' rights or no change is made. The user is directed to the Manage Profiles page.

Use Case Name: `Change Offerings Info` (inherited from Manage Profiles)

Flow of Events: The user can change an account's offerings information if there is. They see a pop-up window with the offerings information of the account. After they make the changes, they can click on "Apply" button. A warning message asks if they are sure, if they again click on the "Apply" button the changes are made to the account. They can go back by "Cancel" button at each step.

Entry Conditions: The user has clicked on the "Change Offerings Information" button.

Exit Conditions: The account's offerings information is changed or not. The user is directed to the account's profile page.

Use Case Name: `Change TA Specific Information` (inherited from Manage Profiles)

Flow of Events: This option is available for accounts with TA roles. When the user clicks on the “Change TA Specific Information” they reach a pop-up window that have “Change Profile Information” and “Change Workload” buttons. Also the “Cancel” button to go back.

Entry Conditions: The user has clicked on the “Change TA Specific Information” button.

Exit Conditions: When the user either clicks on “Cancel” or other buttons to reach other pages.

Use Case Name: `Change Profile Information` (inherited from Change TA Specific Information)

Flow of Events: The user changes the TA specific profile information like Bilkent ID. The user sees the pop-up window that has this information. They approve twice their changes by clicking the button “Apply”. They see a success message and are directed to the account’s profile page. They can go back by clicking the “Cancel” button at each step.

Entry Conditions: The user has clicked on the “Change Profile Information” button.

Exit Conditions: When the user either clicks on “Cancel” or they have made successful changes in the system.

Use Case Name: `Change Workload` (inherited from Change TA Specific Information)

Flow of Events: The user updates the account’s workload. They see a box with the current workload amount. If they changed the workload (and the workload is reasonable, namely non-negative) they can click on the “Apply” button to make the change. They see a success message and are directed to the account’s profile page. They can go back by button “Cancel” at each step.

Entry Conditions: The user has clicked on the “Change Profile Information” button.

Exit Conditions: When the user either clicks on “Cancel” or they have made successful changes in the system.

Use Case Name: `Manage Course Offerings`

Flow of Events: The user is directed to the Manage Course Offerings page. They can go back by the button “Back” or can choose options.

Entry Conditions: The user has logged in, clicked on the button “Manage Course Offerings”, and they have the permission to reach that page.

Exit Conditions: The user has clicked on the button “Cancel”.

Use Case Name: `Manually Delete` (inherited from Manage Course Offerings)

Flow of Events: The user can select a course from the appearing list. They can use the search bar, filtering and sorting choices. After they select the courses to be deleted, they click on the button “Delete”. A confirm message is shown and they can either go with “Delete” or “Cancel” buttons. After they confirm, the changes are made to the system. The button “Cancel” deters this. They can go back by the button “Cancel”.

Entry Conditions: The user has clicked on the button “Manually Delete”.

Exit Conditions: The user has clicked on the button “Cancel” ” in the Manage Course Offerings page.

Use Case Name: `Manually Add` (inherited from Manage Course Offerings)

Flow of Events: The user can go back by the button “Cancel”. A form window appears, the user enters the required information and clicks on the “Add” button to add the course. This information includes: The department, course ID, the course name, and optional ones are additional sections, instructors, a description, schedule for lectures, and extras like labs or recitations. If a course with the same department and course ID, the system shows a warning pop-up and does not add the course. After the user adds the course, the changes are saved and the system directs the user to the Manage Course Offerings page.

Entry Conditions: The user has clicked on the button “Manually Add” in the Manage Course Offerings page.

Exit Conditions: The user has clicked on the button “Cancel” or made some additions.

Use Case Name: `Check Conflicting Courses` (extends Manually Add)

Flow of Events: The system checks that the course that is tried to be added by the user conflicts with an existing course (with both the same department and course ID). If there is a collision, it triggers the warning message mentioned in the use case Manually Add. Otherwise, it does not trigger anything.

Entry Conditions: The user has tried to add a course.

Exit Conditions: The check process for any possible collisions is done.

Use Case Name: `Import Courses` (inherited from Manage Course Offerings)

Flow of Events: The user can go back by the “Cancel” button. Otherwise, in the window that pops, they are asked to upload a proper Excel file. If the file does not have a proper format or it has any overlapping (already existing course) a proper warning message is shown and no change in the course list is made. If the importing is successful, a window lists the read accounts to the user. They have two options (buttons): “Save” and “Discard”. Again, discard does not change any data in the system, if the user clicks on the “Save” button, the changes are saved and applied to the system and a success message is shown to the user.

Entry Conditions: The user has clicked on the button “Import Courses”.

Exit Conditions: Reading the Excel file (successfully or with an error) is done and the user either saves the imported data or goes back to the Manage Course Offerings page.

Use Case Name: `Edit a Course` (inherited from Manage Course Offerings)

Flow of Events: The user sees the list of the current courses. They can choose one of the courses to edit. When they click on a course, that course’s window appears.

Entry Conditions: The user has clicked on the button “Edit a Course” in the Manage Course Offerings page.

Exit Conditions: The user either clicks on the “Cancel” button or makes some changes and is directed to the Manage Course Offerings page.

Use Case Name: `Add Section` (inherited from Edit a Course)

Flow of Events: The user sees a window that shows the current sections and their information. They can add sections by the button “Add”. The system automatically gives the number of the section (if there were three sections, new section is automatically section four), the user is required to add an instructor information (if it is not determined yet, there is “Staff” option), schedule information, and they can add classroom, labs, recitations. By the button “Add” the changes are saved. The user can go back with the button “Cancel”.

Entry Conditions: The user has clicked on the button “Add Section” in a course’s information page.

Exit Conditions: The user either clicks on the “Cancel” button or makes some changes and is directed to the course’s information page.

Use Case Name: `Delete Section` (inherited from Edit a Course)

Flow of Events: The user sees the list of current sections. By selecting one or more sections and clicking on the button “Delete” they can delete those sections. A pop-up message asks the user if they are sure, they need to click on “Delete” again to make the changes. They can go back by the button “Cancel” at each step.

Entry Conditions: The user has clicked on the button “Delete Section” in a course’s information page.

Exit Conditions: The user either clicks on the “Cancel” button or makes some changes and is directed to the course’s information page.

Use Case Name: `Set Labs and Recitations` (inherited from Edit a Course)

Flow of Events: The user sees the list of the labs and recitations of the selected course and the weekly schedule. They can use “Edit Labs” and “Edit Recitations” buttons to reach related windows. In the windows they can add, delete or change the time slots of those activities. They click on the “Apply” button to apply the changes and get directed to the

course’s information page. They can use the “Cancel” button to get the course's information page without any change.

Entry Conditions: The user has clicked on the button “Set Labs and Recitations” in a course’s information page.

Exit Conditions: The user either clicks on the “Cancel” button or makes some changes and is directed to the course’s information page.

Use Case Name: `Set Exam Information` (inherited from Edit a Course)

Flow of Events: The user sees a window that has exam related boxes such as “Add an Exam”, “Edit an Exam” and so on. They can add, delete, edit exams. They can add the classroom and student distribution information to the exam. The system checks for any conflicting information. They click on the “Apply” button to apply the changes and get directed to the course’s information page. They can use the “Cancel” button to get the course's information page without any change.

Entry Conditions: The user has clicked on the button “Set Exam Information” in a course’s information page.

Exit Conditions: The user either clicks on the “Cancel” button or makes some changes and is directed to the course’s information page.

Use Case Name: `Check Exam Schedules` (extends Set Exam Information)

Flow of Events: The system checks for a possible conflict in the exam schedules. If it finds any, it triggers a proper warning message and abolishes the process. Otherwise, nothing is made.

Entry Conditions: The user has tried to add or change an exam's time slot.

Exit Conditions: The system check for collisions is done (with or without any error).

Use Case Name: `Check Exam Classrooms` (extends Set Exam Information)

Flow of Events: The system checks for a possible conflict in the exam classroom assignments. If it finds any, triggers a proper warning message and abolishes the process. Otherwise, nothing is made.

Entry Conditions: The user has tried to add or change an exam classroom.

Exit Conditions: The system check for collisions is done (with or without any error).

Use Case Name: `Manage Classrooms`

Flow of Events: The user clicks on the button “Manage Classrooms” and reaches the Manage Classrooms page. They can choose any option to manipulate the classroom data or they can go back with the button “Back”.

Entry Conditions: The user has logged in and clicked on the button “Manage Classrooms”.

Exit Conditions: The authorized user is either directed to the proper page according to their choice or they go back by button “Back”.

Use Case Name: `Import Classrooms` (inherited from Manage Classrooms)

Flow of Events: The user clicks on the button “Import Classrooms” and get the importing window. They are asked to upload a properly formatted Excel file. If the Excel file is not formatted properly or has an existing classroom, the system invokes a warning message and the process is terminated with no change. If the file reading is successful, the list of the classrooms is shown to the user. If they click on “Add Classrooms” the changes are made in the system and the user is directed to the Manage Classrooms page. They can go back one page at each step by clicking on the button “Cancel”.

Entry Conditions: The user has clicked on the button “Import Classrooms” in the Manage Classrooms page.

Exit Conditions: The user either clicks on the “Cancel” button or makes some changes and is directed to the Manage Classrooms page.

Use Case Name: `Manually Manage Classrooms` (inherited from Manage Classrooms)

Flow of Events: The user is shown a window that has “Add Classroom” and “Delete Classroom”. They can choose one of them or go back by the button “Cancel”.

Entry Conditions: The user has clicked on the button “Manually Manage Classrooms” in the Manage Classrooms page.

Exit Conditions: The user either clicks on the “Cancel” button or makes some changes and is directed to the Manage Classrooms page.

Use Case Name: `Add Classroom` (inherited from Manually Manage Classrooms)

Flow of Events: The user gets a window that shows a form for the classroom to be added. The required information includes the building, and the code of the classroom. The optional information includes the lecture capacity and the exam capacity. If there is a classroom with both the same building and the same code the system triggers a warning message and no change is made. The user clicks on the “Add Classroom” to attempt to add the classroom.

Entry Conditions: The user has clicked on the button “Add Classroom” in the Manually Manage Classrooms page.

Exit Conditions: The user either clicks on the “Cancel” button or makes some changes and is directed to the Manually Manage Classrooms page.

Use Case Name: `Delete Classroom` (inherited from Manually Manage Classrooms)

Flow of Events: The user sees a window that has the list of the classrooms. They select one or more courses to delete and click on the button “Delete”. If they try to delete a classroom with an assigned exam, the system shows a warning message that includes the classroom and its attached exam, and asks if the user is sure. If they click on the button “Delete” again the classroom(s) is deleted, the changes are applied to the system and the user is directed to the Manually Manage Classrooms page. They can click on the button “Cancel” to go back at each step.

Entry Conditions: The user has clicked on the button “Delete Classroom” in the Manually Manage Classrooms page.

Exit Conditions: The user either clicks on the “Cancel” button or makes some changes and is directed to the Manually Manage Classrooms page.

Use Case Name: `Edit A Classroom` (inherited from Manage Classrooms)

Flow of Events: The user sees the list of the current classrooms. They can click on a classroom to reach their information page. On that page, they can edit the classroom's building, code, lecture capacity and exam capacity. When they make a change and click on the button “Save” the system checks for conflicting information. If there is no conflict, the changes are made and the user is directed to the classroom list. If there is a collision, a pop-up error is shown to the user and no change is saved to the system. The user can go back by the button “Cancel” at each step.

Entry Conditions: The user has clicked on the button “Edit A Classroom” in the Manually Manage Classrooms page.

Exit Conditions: The user either clicks on the “Cancel” button or makes some changes and is directed to the classroom list.

2.2 Proctoring Management

Use Case Name: `Set Proctoring`

It inherits “Instructor Proctoring Automatically”, “Instructor Proctoring Manually”, “Dean Assigns Proctoring” and “Dean Proctoring Manually” cases.

Flow of Events: according to the type of user (separated as dean’s proctoring assignments and instructor assignments) all the users mentioned above will be directed to the operations. There are four main operations, these are “Instructor Proctoring Automatically”, “Instructor Proctoring Manually”, “Dean Assigns Proctoring” and “Dean Proctoring Manually”.

Entry Conditions: The authorized user chooses the set proctoring menu and they enter the properties of the exam such as duration, date, etc.

Exit Conditions: the user clicks the back button.

Use Case Name: `Instructor Proctoring Automatically` (inherited by Set Proctoring)

It includes “List Available TA’s in the Rank off Priority” case.

Flow of Events: After entering the conditions of the exam such as duration of exam, number of proctors, etc., the automatic system will give the TAs whose workload is least for the proctor assignment. There are some restrictions and priorities for this assignment; they will be explained in further cases in order to prevent repetition.

Entry Conditions: The authorized user(as an instructor) chooses the automatic proctoring option.

Exit Conditions: the user clicks the back button.

Use Case Name: `List Available TA's in the Rank off Priority` (included by Instructor Proctoring Automatically)

It includes “Assign TA's in the Needed Amount”.

Flow of Events: Lists Available TAs in the order of priority which are being the same course’s Ta and being in the same department.

Use Case Name: `Assign TA's in the Needed Amount` (included by List Available TA's in the Rank off Priority)

It includes “Update the Workload of TA's” and extends “Override Restriction”.

Flow of Events: Assigns required Tas from the priority list of available TAs.

Use Case Name: `Update the Workload of TA's` (included by Assign TA's in the Needed Amount)

It includes “Inform TA”

Flow of Events: After assigning the TAs to the proctoring duties, it updates the workloads of the TAs.

Use Case Name: `Inform TA` (included by Update the Workload of TA's)

Flow of Events: After assigning TAs to the proctoring duties, the system will inform them.

Use Case Name: `Override Restriction` (excluded by Assign TA's in the Needed Amount)

Flow of Events: In such cases, there is not enough of TA with default restriction, the system will ask to update the restrictions from the user. Restrictions are given by the customer.

Entry Conditions: The menu pops to the user and they make some changes according to their requirements.

Exit Conditions: the user clicks either the back button or save button.

Use Case Name: `Instructor Proctoring Manually` (inherited by Set Proctoring)

It includes “Give TA List”, “Inform TA” and “Increase TA Workload”

Flow of Events: The user will select the TAs for the proctoring assignments by manually selecting from the given list. The related operations of this case will be explained in further cases.

Entry Conditions: The user(as an instructor) chooses the manual proctoring menu.

Exit Conditions: the user clicks the back button.

Use Case Name: `Give TA List` (included by Instructor Proctoring Manually)

It includes “Filtering” and extends “Not Enough TA”

Flow of Events: The user will select the TAs for the proctoring assignments by manually selecting from the given list. The related operations of this case will be explained in further cases.

Entry Conditions: The user receives a TA list and they select the TAs from this list.

Exit Conditions: the user clicks either the back button or the done button.

Use Case Name: `Filtering` (included by Give TA List)

It inherits “Choose Restriction” and “Adjust Priority”

Flow of Events: Filters the TAs according to the user’s choices in order to provide a user-friendly interface to the users. These filterings will be reflected in the list.

Entry Conditions: The user enters the filtering menu by clicking the filtering option.

Exit Conditions: the user clicks either the back button or the done button.

Use Case Name: `Choose Restriction` (inherited by Filtering)

Flow of Events: The user filters and chooses the restrictions for TAs.

Entry Conditions: The user enters the restriction menu by clicking the restriction option.

Exit Conditions: the user clicks either the back button or the done button.

Use Case Name: `Adjust Priority` (inherited by Filtering)

Flow of Events: The user filters and chooses the priorities for TAs.

Entry Conditions: The user enters the priority menu by clicking the priority option.

Exit Conditions: the user clicks either the back button or the done button.

Use Case Name: `Not Enough TA` (extended by Give TA List)

Flow of Events: After the user filters the TAs, she/he receives the TA list. However, if there is not enough TA with the given filterings the system will warn the user.

Use Case Name: `Inform TA` (included by Instructor Proctoring Manually)

Flow of Events: After the user assigns TAs to the proctoring duties, the system will inform TAs.

Use Case Name: `Increase TA Workload` (included by Instructor Proctoring Manually)

Flow of Events: After the user assigns TAs to the proctoring duties, the system will increase TAs' workloads.

Use Case Name: `Dean Assigns Proctoring` (inherited by Set Proctoring)

It includes "Assign Optimal TA".

Flow of Events: After entering the conditions of the exam such as duration of exam, number of proctors, etc., the automatic system will give the TAs whose workload is least for the proctor assignment.

Entry Conditions: The user(as a Dean) chooses the automatic proctoring menu.

Exit Conditions: the user clicks the back button.

Use Case Name: `Assign Optimal TA` (included by Dean Assigns Proctoring)

It includes “Increasing TA Workload”, “Inform TA”, “Department Choice” and extends “No Available TAs”.

Flow of Events: Assigns the least workload TA for the proctoring duty from all/some departments.

Use Case Name: `Increasing TA Workload` (included by Assign Optimal TA)

Flow of Events: After assigning TAs to the proctoring duties, the system will increase their workload.

Use Case Name: `Inform TA` (included by Assign Optimal TA)

Flow of Events: After assigning TAs to the proctoring duties, the system will inform them.

Use Case Name: `Department Choice` (included by Assign Optimal TA)

Flow of Events: Since for the Dean’s exams there are no department restrictions the user picks the departments of the TAs.

Entry Conditions: The department choice menu pops up to the user and the user selects the departments.

Exit Conditions: the user clicks either the back or the done button.

Use Case Name: `No Available TAs` (extended by Assign Optimal TA)

It includes “Override Restrictions According to Departments”.

Flow of Events: In such cases, there is not enough amount of TA with default restriction, and the departments that are picked by the user the system will ask to update the restrictions and the department choice from the user. Restrictions are given by the customer.

Use Case Name: `Override Restrictions According to Departments` (included by No Available TAs)

Flow of Events: updates restrictions and the department choices if needed.

Entry Conditions: The restrictions menu pops up to the user and the user selects the departments.

Exit Conditions: the user clicks either the back or the done button.

Use Case Name: `Dean Proctoring Manually` (inherited by Set Proctoring)

It includes “Give TA List”, “Inform TA” and “Increasing TA Workload”.

Flow of Events: After entering the conditions of the exam such as duration of exam, number of proctors, etc., the user will select TAs from the list that the system gives.

Entry Conditions: The user(as a Dean) chooses the manual proctoring menu.

Exit Conditions: the user clicks the back button.

Use Case Name: `Give TA List` (included by Dean Proctoring Manually)

It includes “Filtering”.

Flow of Events: After the filtering the system gives the TA list to the user and the user selects TAs from this list to assign them manually.

Entry Conditions: The user receives a TA list and they select the TAs from this list.

Exit Conditions: the user clicks either the back button or the done button.

Use Case Name: `Filtering` (included by Give TA List)

It inherits “Choose Department”, “Choose Restriction” and “Adjust Priority”

Flow of Events: Filters the TAs according to the user’s choices in order to provide a user-friendly interface to the users. These filterings will be reflected in the list.

Entry Conditions: The user enters the filtering menu by clicking the filtering option.

Exit Conditions: the user clicks either the back button or the done button.

Use Case Name: `Choose Restriction` (inherited by Filtering)

Flow of Events: The user filters and chooses the restrictions for TAs.

Entry Conditions: The user enters the restriction menu by clicking the restriction option.

Exit Conditions: the user clicks either the back button or the done button.

Use Case Name: `Adjust Priority` (inherited by Filtering)

Flow of Events: The user filters and chooses the priorities for TAs.

Entry Conditions: The user enters the priority menu by clicking the priority option.

Exit Conditions: the user clicks either the back button or the done button.

Use Case Name: `Choose Department Priority` (inherited by Filtering)

Flow of Events: The user filters and chooses the departments of TAs.

Entry Conditions: The user enters the department menu by clicking the department option.

Exit Conditions: the user clicks either the back button or the done button.

Use Case Name: `Increasing TA Workload` (included by Dean Proctoring Manually)

Flow of Events: After assigning TAs to the proctoring duties, the system will increase their workload.

Use Case Name: `Inform TA` (included by Dean Proctoring Manually)

Flow of Events: After assigning TAs to the proctoring duties, the system will inform them.

Use Case Name: `Check Exam Information`

It extends “Generate Student List for Exam”.

Flow of Events: Gives the Exam information such as classes, date, duration, TAs etc. to the user.

Entry Conditions: The user selects the exam information option.

Exit Conditions: the user clicks the back button.

Use Case Name: `Generate Student List for Exam` (extended by Check Exam Information)

It includes “Set Rule of Order”.

Flow of Events: takes the exam information and orders it by the order rule.

Entry Conditions: The user selects the rule of the order (alphabetic or random).

Exit Conditions: the user clicks either the back button or the done button.

Use Case Name: `Set Rule of Order` (included by Generate Student List for Exam)

Flow of Events: Sets the rule of order either alphabetically or randomly for the list.

Use Case Name: `Instructor Attempts Swapping`

It includes “Swap Done” and extends “TA Availability Check”

Flow of Events: The user attempts to create a swap request for the proctoring TAs.

Entry Conditions: The user (as an instructor or dean) selects the Swap menu.

Exit Conditions: the user clicks the back button.

Use Case Name: `TA Availability Check` (extended by Instructor Attempts Swapping)

Flow of Events: Checks for the TA availability for the swapping request. If the TA is not available then the request cannot be applied. Also, it informs the user.

Use Case Name: `Swap Done` (included by Instructor Attempts Swapping)

It includes “Inform TA”

Flow of Events: The swap request is accepted and the protectors are updated.

Use Case Name: `Inform TA` (included by Swap Done)

It includes “Update the Workload of TA's”

Flow of Events: Inform the TAs about the swap.

Use Case Name: `Update the Workload of TA's` (included by Inform TA)

Flow of Events: updates the proctors' workload according to the swap.

Use Case Name: `Swap Request`

Includes “Approve/ Reject” and extends “Check Availability”, “Check History”.

Flow of Events: updates the proctors' workload according to the swap.

Entry Conditions: The user (as a TA) selects the Swap menu.

Exit Conditions: the user clicks the back button.

Use Case Name: `Check Availability` (extended by Swap Request)

Flow of Events: Checks the availability of the TA in the swap request if TA is not available it informs the TA that created the request.

Use Case Name: `Check History` (extended by Swap Request)

Flow of Events: Checks the swap history and if the swap has already happened it informs the TA that created the request.

Use Case Name: `Approval/ Reject` (included by Swap Request)

It includes “Inform Sender TA”, “Inform Instructor” and “Update Work”

Flow of Events: case of approval or rejection. It depends on the TA that is the subject of the swap request.

Entry Conditions: The user (as a TA) selects the approve or reject button.ss

Exit Conditions: the user clicks the back button.

Use Case Name: `Inform Sender TA` (included by Approval/ Reject)

Flow of Events: Inform the TA who created the swap request about the swap.

Use Case Name: `Inform Instructor`(included by Approval/ Reject)

Flow of Events: Inform the staff about the swap.

Use Case Name: `Update Work` (included by Approval/ Reject)

Flow of Events: updates the proctors' workload according to the swap.

2.3 TA Availability & Leave Requests

Use Case Name: `Submit Leave Request`

Flow of Events :
In this use case, the TA logs into the system and navigates to the “Submit Leave Request” interface. The TA then provides the required details such as the start date, end date, and reason for leave. After confirming that all information is accurate, the TA submits the request, causing the system to record it in the database. At this point, the **Check Leave History** use case is triggered to reference any past leaves or relevant historical data. Finally, the submitted request is set for further review in the **Evaluate Request** stage.

Entry Conditions: The TA must be logged into the system and has the rights to submit a leave request.

Exit Conditions: A new leave request is stored in the system. The request transitions to the evaluation process.

Use Case Name: `Evaluate Request`

Flow of Events:
When the authorized user chooses to evaluate a leave request, the system presents a list of pending submissions. The reviewer selects one and examines its details, including any previously recorded history from **Check Leave History** if relevant. If the request information is insufficient, the reviewer may invoke **Request Additional Documentation**, prompting the TA to provide more data. The TA can then use **Submit Additional Documentation** to upload the requested files, after which the reviewer can proceed with the evaluation. Depending on the situation, the reviewer either approves or denies the request (handled internally as part of the evaluation). If the leave is approved, the system calls **Update Availability** to reflect the approved dates on the TA’s schedule. The system may also trigger **Notify Instructor** to inform instructors of any potential impact, and it may invoke **Evaluate Report** for a more detailed analysis if needed. Throughout or after the decision, the TA is informed via the **Notify TA** use case as appropriate.

Entry Conditions: The user (Instructor, Admin, etc.) has permission to review leave requests. At least one submitted request is pending evaluation.

Exit Conditions: The request is either approved (and availability updated) or denied (with no schedule change). Relevant notifications (to TA, Instructor) or further documentation requests may have been issued.

Use Case Name: `Request Additional Documentation` (included from Evaluate Request)

Flow of Events:
While examining a leave request in the **Evaluate Request** stage, the reviewer may decide that more details or proof are required before making a final decision. In that case, the "Request Additional Documentation" step is activated, prompting the system to create a notification for the TA outlining what extra documents are needed. Afterward, the TA will respond through **Submit Additional Documentation**, and once those files are uploaded, the reviewer can continue the evaluation process.

Entry Conditions: The reviewer is in the middle of evaluating a leave request and requires additional information.

Exit Conditions: The system has recorded the request for additional documentation. The TA is notified and expected to submit the necessary files.

Use Case Name: `Submit Additional Documentation`

Flow of Events:
In response to a "Request Additional Documentation," the TA accesses the "Submit Additional Documentation" interface. The TA uploads the requested files (e.g., medical reports, official letters) and confirms the submission. The system stores these documents, links them to the original leave request, and notifies the reviewer that the new materials are available, allowing the **Evaluate Request** use case to proceed with the now-updated request data.

Entry Conditions: The TA has received a request to provide additional documentation. The TA is logged in and can access the submission page.

Exit Conditions: The requested documents are attached to the leave request. The evaluation process can resume with the new information.

Use Case Name: `Update Availability` (included from Evaluate Request)

Flow of Events:
Whenever a leave request is approved during **Evaluate Request**, the system updates the TA's availability calendar to block out the approved leave dates. This ensures that any scheduling or workload assignment features (e.g., proctoring tasks, lab sessions) reflect the TA's new unavailability. The system also records the leave in the TA's record so that other modules, such as **Check Valid Work** or reporting features, see the correct availability.

Entry Conditions: A leave request has just been approved in the evaluation step.

Exit Conditions: The TA's availability calendar is updated with the leave period. Other system modules can accurately reference the TA's current schedule.

Use Case Name: `Check Leave History` (included from Submit Leave Request)

Flow of Events:
Once the TA submits a new leave request, or a reviewer opens the **Evaluate Request** page, the system consults the TA's past leave records to identify any patterns, constraints, or cumulative leave totals. This process may influence the reviewer's decision, especially if the TA has already exhausted certain leave entitlements or if there are overlapping leaves. The system simply retrieves historical data from the database and makes it available during evaluation.

Entry Conditions: A TA leave request is submitted or being evaluated. The system needs to reference past records.

Exit Conditions: The reviewer or the system has contextual information about the TA's previous leaves. The data can inform subsequent decisions (approval, denial, or extra documentation needs).

Use Case Name: `Submit Work Load`

Flow of Events:
When a TA or Instructor wishes to record or modify the TA's workload—such as assigned lab hours or other duties—they open the "Submit Work Load" function. The system receives the updated workload details and then automatically triggers **Check Valid Work** to verify there is no conflict with ongoing or approved leave dates. If no issues arise, the system finalizes the

workload changes. If there are conflicts, it warns the user or blocks the update until the conflict is resolved.

Entry Conditions: The user (TA or Instructor) needs to add or update the TA's workload.

Exit Conditions: The TA's workload is updated in the system, or the user is informed of leave conflicts that must be addressed. Any instructors or relevant parties may be notified if changes occur.

Use Case Name: `Check Valid Work` (include from Submit Work Load)

Flow of Events:
Upon receiving a new or modified workload assignment from **Submit Work Load**, the system checks if the specified tasks overlap with any approved or pending leave in the TA's schedule. If a conflict is detected, the system issues a warning or error to the user, indicating that the time period is already marked as leave. Should no conflict exist, the assignment proceeds without interruption, and the workload record is saved.

Entry Conditions: The system is about to finalize a workload update for a TA. The TA has at least one leave request pending or approved in the same timeframe.

Exit Conditions: The system confirms no scheduling conflict, or it blocks the update and alerts the user about the overlap. Workload data is either saved successfully or requires adjustment by the user.

Use Case Name: `Notify Instructor` (included from Evaluate Request and Submit Work Load)

Flow of Events:
Whenever an action in the system impacts a course's staffing—such as an approved TA leave or a change in TA workload—the system may invoke "Notify Instructor." This process involves sending a message (email or in-system notification) to the relevant Instructor(s) so that they are aware of the modification and can plan accordingly. The Instructor will be informed of the TA's updated schedule, potential changes in coverage, or any immediate actions required.

Entry Conditions: A TA's leave request has been approved, or workload changes affect a specific course. The system detects that the Instructor needs to be informed.

Exit Conditions: The Instructor receives notification about the TA's status or assignment changes. The system records that the Instructor has been notified.

Use Case Name: `Evaluate Report` (included from Evaluate Request)

Flow of Events:
In some cases, the reviewer may request a comprehensive analysis before deciding on a TA's leave. The "Evaluate Report" step compiles relevant data, including prior leave requests, supporting documents, current workload, and other historical metrics. Once generated, this consolidated report is presented to the reviewer, who can then make a more informed decision within **Evaluate Request**. If necessary, the reviewer can also proceed to request more documentation or finalize an approval or rejection.

Entry Conditions: The reviewer needs detailed information (e.g., a summary of the TA's performance, prior leave usage, or other metrics).

Exit Conditions: The reviewer has an in-depth report to guide the final verdict. The leave evaluation can proceed with full information.

Use Case Name: `Notify TA` (included from Evaluate Request)

Flow of Events:
Whenever a significant action occurs regarding a TA's leave request—such as approval, denial, or a call for additional documentation—the system initiates "Notify TA." The TA receives an email or an in-system alert summarizing the event and any required next steps. This ensures that TAs remain updated on their request status and know if they must provide more information.

Entry Conditions: The system processes a leave request outcome or needs more details from the TA.

Exit Conditions: The TA is fully informed about the request's status and any follow-up tasks. The system records that the notification has been delivered.

2.4 Notifications & Communication

Use Case Name: `Send Notification`

Flow of Events: The sender or the system composes a message, specifies recipients, and finalizes the notification details. The system then delivers the notification to the designated users through in-app alerts or email.

Entry Conditions: A valid event or need for notification must exist, and the sender (if human) must have permission to send notifications.

Exit Conditions: The system logs the notification, and recipients are set to receive it.

Use Case Name: `Receive Notification`

Flow of Events: The system routes the newly created notification to the intended users, marking it as new and, if applicable, sending email or external alerts. Recipients see the message in their inbox or dashboard but have not yet read or acted on it.

Entry Conditions: A notification must have been sent, and the recipient must be valid in the system.

Exit Conditions: The notification appears in the recipient's list, and the system records its delivery.

Use Case Name: `View History`

Flow of Events: The user opens a history page or archive, and the system retrieves previously stored notifications. The user can search, filter, or read specific entries for reference or auditing.

Entry Conditions: The user is logged in and allowed to see stored notifications.

Exit Conditions: The system displays past notifications, and the user may access or revisit any needed details.

Use Case Name: `Read Notification`

Flow of Events: The user opens a notification to see its content, including any message, links, or attachments. By reading it, the user becomes aware of what is requested or announced.

Entry Conditions: A notification must be in the user's inbox, waiting to be read.

Exit Conditions: The user has viewed the notification's content and may proceed to confirm, reject, or mark it.

Use Case Name: `Mark Notification` (included by Read Notification)

Flow of Events: After reading, the user can change the notification's status to indicate it is read, archived, or otherwise processed. The system records this update, which affects how the notification is displayed or tracked.

Entry Conditions: The user has accessed the notification and needs to mark its status.

Exit Conditions: The system notes the new status, and the notification may no longer appear as unread.

Use Case Name: `Confirm` (extended by Read Notification)

Flow of Events: A notification may prompt the user to approve a request or acknowledge information. By choosing to confirm, the user's acceptance is recorded, and the system may trigger subsequent actions.

Entry Conditions: The notification must contain a request that can be accepted, and the user must have the authority to confirm it.

Exit Conditions: The confirmation is logged, and any linked processes, such as approvals or updates, move forward.

Use Case Name: `Reject` (extended from Read Notification)

Flow of Events: If a notification contains a request the user does not agree with, the user selects reject. The system registers this refusal and informs any related workflows or users.

Entry Conditions: The notification must allow rejection, and the user must have the right to refuse.

Exit Conditions: The rejection is saved, and any dependent processes or records are updated to show that the user did not approve the request.

2.5 Authentication

Use Case Name: `Login`

Flow of Events: The user opens the login page, provides a username (or email) and a password, then submits the credentials. The system verifies the information. If they match a valid account, the user is granted access; otherwise, the system can trigger “Non-Existing User” or “Invalid Password” as extensions.

Entry Conditions: The user must be on the login interface, and the authentication service must be operational.

Exit Conditions: If credentials are correct, the user is logged in and gains access to authorized features; if not, the system proceeds with the appropriate extended use case for handling errors.

Use Case Name: `None-Existing User` (extended by Login)

Flow of Events: When the system finds no matching account for the username/email during **Login**, it triggers “Non-Existing User.” A message informs the user that the account does not exist. The user may be prompted to check spelling, sign up, or request assistance.

Entry Conditions: The user has just attempted to log in with credentials that do not match any account in the system.

Exit Conditions: The system alerts the user that no such account is registered, and the login flow does not proceed until the user corrects their input or chooses another action.

Use Case Name: `Invalid Password` (extended by Login)

Flow of Events: During **Login**, if the username/email is valid but the password is incorrect, “Invalid Password” is triggered. The system may allow the user to retry, display a warning about incorrect credentials, or offer a “Forgot Password” link.

Entry Conditions: The user provided a recognized username/email, but the password check failed.

Exit Conditions: The user is informed of the invalid password. If repeated attempts fail, further security measures could apply. The user remains outside the system until correct credentials are entered or a password reset is done.

Use Case Name: `Forgot Password`

Flow of Events: The user selects “Forgot Password” at the login interface. The system requests identifying information and sends a reset link or code. The user verifies their identity by following the link or entering the code, then sets a new password.

Entry Conditions: The user must indicate they cannot recall their password, and the system must have a mechanism to reset credentials.

Exit Conditions: If verification is successful, the user's password is updated, and they can log in again using the new credentials.

Use Case Name: `Sign Up` (extends to Invalid Bilkent Mail, Mismatch Password Two Times, includes E-Mail Verification)

Flow of Events: The user chooses to sign up by providing their Bilkent email address, name, and desired password. The system checks whether the email is valid and whether both password fields match. If these conditions are met, **E-Mail Verification** is included to confirm ownership of the email. Otherwise, the process may extend to "Invalid Bilkent Mail" or "Mismatch Password Two Times."

Entry Conditions: The user reaches the signup interface and intends to create an account.

Exit Conditions: A new account is created if all validations pass, and the user is prompted to confirm ownership of their email. If errors arise, the appropriate extended use cases handle them.

Use Case Name: `Mismatch Password 2 times-mail` (included from Sign Up)

Flow of Events: During the **Sign Up** process, the user must confirm the same password twice. If the two entries do not match, the system triggers "Mismatch Password Two Times" by alerting the user and requesting they re-enter the passwords correctly.

Entry Conditions: The user has submitted a signup form where the second password field does not match the first.

Exit Conditions: The user remains in the signup flow, unable to proceed until the passwords match.

Use Case Name: `Invalid Bilkent Mail` (included from Sign Up)

Flow of Events: When a user attempts to register but enters an email address not recognized as a valid Bilkent domain, “Invalid Bilkent Mail” is triggered. The system informs the user that they must use a proper Bilkent email format to continue.

Entry Conditions: The user has submitted the signup form with an email address that fails Bilkent-specific checks.

Exit Conditions: The user is prompted to correct the email address, and the system does not create an account until a valid email is provided.

Use Case Name: `E-mail Verification` (included from Sign Up)

Flow of Events: After a user passes basic checks (valid Bilkent email and matching passwords), the system includes “E-Mail Verification” by sending a confirmation link to the provided address. The user must follow the link or input a verification code to confirm they own the email. Only after successful verification does the system finalize the account.

Entry Conditions: The user has completed the Sign Up form correctly, and the system is ready to confirm email ownership.

Exit Conditions: The email address is verified; the new account becomes active, and the user can log in with their chosen credentials.

3.Tech Stack

`Programming Language:` `The chosen tool is` **`Phyton`**

We have chosen **Python** as the programming language for our project because it is versatile, widely used in web development, and supports powerful frameworks like **Django** which we are planning to use.

Front Development: The chosen tool is **React**

For the frontend of our **TA Management System**, we have chosen **React**, a modern **JavaScript library** for building **fast, dynamic, and reusable** user interfaces.

- **Faster & More Dynamic UI**: React enables a **smooth user experience**.
- **Component-Based Architecture**: The UI is built using **reusable components**, making development **efficient and scalable**.
- **Flexible Integration with Django**: React will **consume Django's REST API**, ensuring a clear separation between frontend and backend. React will fetch data from **Django's REST API** using **Axios**

By using React, we ensure a **modern, user-friendly, and responsive UI** for the TA Management System.

Backend Development: The chosen tool is **Django** (phyton)

We will be using **Django**, a high-level Python web framework, backend development of our TA Management System. Django comes with built-in features for database management, authentication, security, and scalability.

Using Django for **backend development** offers several advantages:

- **Faster Development**: Built-in functionalities reduce coding effort.
- **Seamless Backend**: Handles business logic, authentication, and APIs efficiently.
- **Strong Security**: Includes **built-in authentication and security mechanisms**.
- **Deployment Compatibility**: Works **natively with Apache2** and **mod_wsgi** on Linux.

Additionally, **Django integrates smoothly with MySQL**, which will be our database. We will also use Django's built-in **email system** for user authentication and notifications. Considering the features and the opportunities that Django offers us we are planning to use it.

Database: The chosen tool is **MySQL**

The customer's demand.

Web Server: The chosen tool is **Apache2**

The customer's demand. We will use mod_wsgi to bridge Django and Apahce2.

Authentication & Security: The chosen tool is **Django**

We will use **Django's built-in authentication system** to manage user authentication securely.Django's security features will ensure the protection of user credentials and data.

- **Secure User Logins**: Supports TAs, professors, and admins.
- **Password Hashing**: Uses industry-standard encryption techniques.
- **CSRF & SQL Injection Protection**: Built-in security against common web attacks.
- **Email-Based Authentication**: Supports email verification, password resets, and notifications.

Linux system: The chosen tool is **Ubuntu**

Since the customer's requirement is that the application must run on Linux, we have chosen Ubuntu as the operating system.We will configure Ubuntu to optimize performance and security for Django deployment.

- **User-Friendly & Widely Used**: Easier to configure and maintain.
- **Supports All Chosen Tools**: Works well with Django, Apache2, mod_wsgi, and MySQL.
- **Strong Community Support**: Regular updates and security patches.