

Software Design Descriptions

for

CENG Website Management Desktop App

Özgesu Özen
Busenur Aktılav
Ayşenur Erdem
Berk Bildirici
Furkan Emre Yılmaz

Izmir Institute of Technology
14.05.2019

1. Overview

1.1 Revision History

1.2 Glossary

1.3 Scope

1.4 Purpose

1.5 Context

1.6 References

2. Conceptual model for software design description

2.1 Software design in context

3. Design ViewPoints

3.1 Introduction

3.2. Information ViewPoint

3.3. Interface ViewPoint

3.3.1. CENGDesktopWMAApp

3.3.2. Starter

3.3.3.CourseDAO

3.3.4. EmailDAO

3.3.5. EventDAO

3.3.6. UserDAO

3.3.7. CourseModel

3.3.8. EmailModel

3.3.9. EventModel

3.3.10. UserModel

3.3.11. ComponentContainerController

3.3.12. ContainerPosition

3.3.13. CourseEntryController

3.3.14. CourseTableController

3.3.15. ManageCourseController

3.3.16. CoursePopupBuilder

3.3.17. CoursePopupController

3.3.18. ScheduleDay

3.3.19. ScheduleController

3.3.20. ScheduleCourseBuilder

3.3.21. ScheduleCoursesController

3.3.22. AnnouncementController

3.3.23. EventEntryController

3.3.24. EventTableController

3.3.25. SendNotificationBuilder

3.3.26. SendNotificationPopupController

3.3.27. EditEmailListsController

3.3.28. Email List Chooser

3.3.29. ImportEmailListController

3.3.30. Manage Emails Controller

3.3.31. User Management Controller

3.3.32. User Entry Controller

- 3.3.33. User Popup Builder
- 3.3.34. Users Popup Controller
- 3.3.35. Accordion MN Controller
- 3.3.36. Abstract In Window Popup Controller
- 3.3.37. Abstract Popup
- 3.3.38. In Window Popup Manager
- 3.3.39. Status Bar Controller
- 3.3.40. Component Communicator
- 3.3.41 IMenu Communicator

3.4. Interaction ViewPoint

- 3.4.1. Login Sequence Diagram
- 3.4.2. User Management
 - 3.4.2.1. Create User Sequence Diagram
 - 3.4.2.2. Show User Sequence Diagram
 - 3.4.2.3. Delete User Sequence Diagram
- 3.4.3. Notification Management
 - 3.4.3.1. Display Email List Sequence Diagram
 - 3.4.3.2. Import Email List Sequence Diagram
 - 3.4.3.3. Add Email Sequence Diagram (to enter the add email pop up)
 - 3.4.3.4. Add Email Sequence Diagram (to add an email)
 - 3.4.3.5. Delete Email List Sequence Diagram
 - 3.4.3.6. Search and Delete Email Sequence Diagram
- 3.4.4. Course Management
 - 3.4.4.1. Add Course Sequence Diagram
 - 3.4.4.2. Delete Course Sequence Diagram
 - 3.4.4.3. Edit Course Sequence Diagram
 - 3.4.4.4. Update Course Options Sequence Diagram
 - 3.4.4.5. Update Schedule Sequence Diagram (to edit the schedule that filled before)
 - 3.4.4.6. Schedule Sequence Diagram (to create a new schedule)
- 3.4.5. Logout Sequence Diagram

3.5. Logical ViewPoint

- 3.5.1. Reduced Relation
- 3.5.2. Entities
 - 3.5.2.1 Course Data Entity
 - 3.5.2.2 Teaching Staff Entity
 - 3.5.2.3 User Entity
 - 3.5.2.4 IsGivenBy

4. UI Screenshots of Desktop App

- 4.1. Login Page UI
- 4.2. User Management UI
- 4.3. Create User UI
- 4.4. Show User UI
- 4.5. Delete User UI
- 4.6. Course Management UI
- 4.7. Add Course UI

- 4.8. Enable Course UI**
- 4.9. Manage Emails UI**
- 4.10. Import Email List UI**
- 4.11. Search Email UI**
- 4.12. Delete Email UI**
- 4.13. Delete Email List UI**
- 4.14. Announcements UI**
- 4.15. Send Notification UI**
- 4.16. Schedule Courses UI**
- 4.17. Schedule Course Pop-up UI**

1. Overview

1.1 Revision History

Name	Date	Reason For Changes	Version
Group 9	07.04.2019	SDD Template is created.	v0.1.1
Group 9	16.04.2019	Scope, Purpose and E-R Diagram are added.	v0.1.2
Group 9	18.04.2019	UML Diagrams are added.	v0.1.3
Group 9	23.04.2019	Sequence Diagrams are added.	v1.1.1
Group 9	28.04.2019	Diagrams are updated.	v1.1.2
Group 9	03.05.2019	Glossary and Context are added.	v1.1.3

1.2 Glossary

TERM	DEFINITION
SDD	Software Design Descriptions
CENG	Computer Engineering Department
CENG WM	Computer Engineering Department Website Management
Users	Admin/Administrator and Content Manager
Login	To connect the system interface
Logout	To end the connection to the system interface.
CRUD	Create, Read, Update, Delete
WM	Website Management
GUI / UI	Graphical User Interface
Database	Collection of all the information monitored by this system.
ER Diagram	Entity relationship diagram that describes relations between specific domain knowledge.
Java	It is a set of computer system and specifications
JavaFX	JavaFX is a software platform for creating and delivering desktop applications.
IDE	Integrated Development Environment
Eclipse	Eclipse is a free, Java-based development platform.
App	CENG Website Management App
Credential	Evidence of authority
Main Menu	The opening or main page of a website

1.3 Scope

This Software Design Description (SDD) describes the detailed structure of the components of the CENG Desktop App and the precise implementation details required to satisfy the requirements as specified in the Software Requirements Specification (SRS). It is assumed that the reader has read the SRS, since this document also defines the implementation details of the desired behaviour given the requirements within it.

1.4 Purpose

This standard specifies requirements on the information content and organization of SDDs. The standard specifies requirements for the selection of design languages to be used for SDD, and requirements for documenting design viewpoints to be used in organizing an SDD.

1.5 Context

Computer Engineering Website Management App is a desktop application which helps the users of CENG department to get control of the content of the CENG website. This application will be designed as object-oriented. The system will be implemented with Java and JavaFX using Eclipse as IDE. The system will try to give the most accurate result in a most applicable, proper and correct time; so it can respond to users' wants correctly and quickly. It should respond correctly in official information, also quickly in user interface and updating database. The speed of software of the system depends on the computational operations according to the processor(s) of a computer that runs our system. Our Computer Engineering Website Management App is planned to be a desktop application on the system in personal computers which need to have access to the Internet.

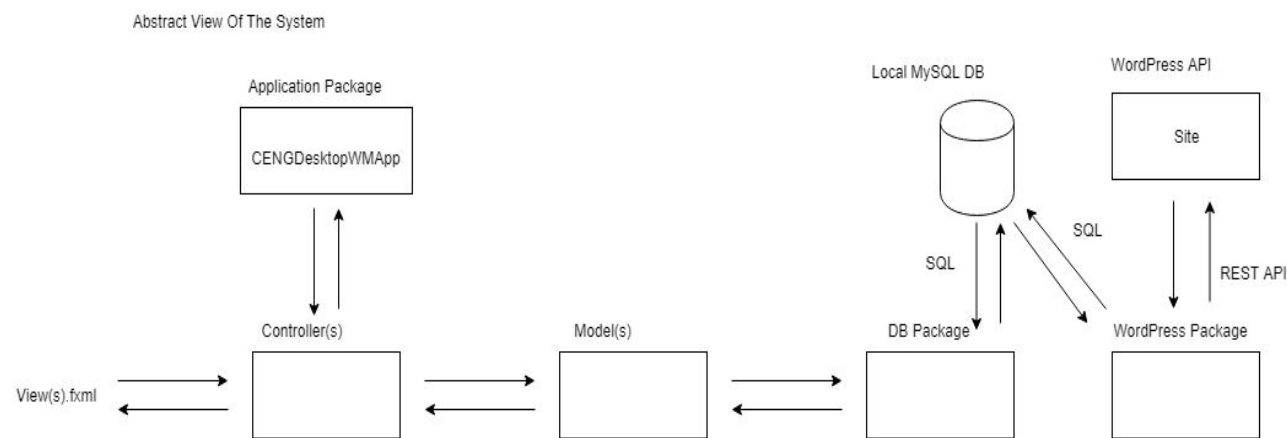
More specifically this system is designed to allow the users to manage all the courses in the department website. Additionally, users can make CRUD processes on the courses. The system allows admin to manage email lists and send events notification to the selected email lists. The system provides communication between users and CENG website. The system also contains a relational database containing the list of courses, staffs of CENG department, and users' information.

1.6 References

IEEE. IEEE Std 1016-2009 IEEE Standard for Information Technology – System Design – Software Design Descriptions. IEEE Computer Society, 2009

2. Conceptual model for software design description

2.1 Software design in context



3. Design ViewPoints

3.1 Introduction

In this part, four main design viewpoints will be explained in detail.

1. Information ViewPoint
2. Interface ViewPoint
3. Interaction ViewPoint
4. Logical ViewPoint

3.2 Information ViewPoint

3.3. Interface ViewPoint

3.3.1. CENGDesktopWMAApp

CENGDesktopWMAApp
- instance: CENGDesktopWMAApp - loginRequired: boolean - loggedUser: UserModel - container: ComponentContainerController - navigation: AccordionMNController - communicator: ComponentCommunicator - statusBar: StatusBarController
+ getInstance(): CENGDesktopWMAApp + initializePopupManager(): CENGDesktopWMAApp + refresh(): CENGDesktopWMAApp + loginRequired(): CENGDesktopWMAApp + loggedInAs(): CENGDesktopWMAApp - login(): CENGDesktopWMAApp + logout(): CENGDesktopWMAApp + build(): ComponentContainerController

Name	Visibility	Return Type / Type	Definition
instance	private	CENGDesktopWMAApp	The object which create statically in the CENGDesktopWMAApp class.
loginRequired	private	boolean	Its default value is false.
loggedUser	private	UserModel	It defines the active user in the system.
container	private	ComponentContainerController	The main border that contains the views.
navigation	private	AccordionMNController	It defines the navigation menu, which located on the left hand side, for the system.
communicator	private	ComponentCommunicator	It defines the each component of the menu navigator.
statusBar	private	StatusBarController	Defines the status bar.
getInstance()	public	CENGDesktopWMAApp	Method which gets the

			object created in the CENGDesktopWMAApp class.
initializePopupManager()	public	CENGDesktopWMAApp	It initializes the popup manager.
reflesh()	public	CENGDesktopWMAApp	It reloads the page.
loginRequired()	public	CENGDesktopWMAApp	It requires the login to access the system.
loggedAs(UserModel user)	public	CENGDesktopWMAApp	It defines whether user is content manager or admin.
login()	public	CENGDesktopWMAApp	It provides the login operation for the user to the system.
logout()	public	CENGDesktopWMAApp	It provides the logout operation for the user from the system.
build()	public	ComponentContainerController	It builds the general view of the system.

3.3.2. Starter

It is the initiative class that starts the app.

Starter
<div>+ start(): void</div> <div>+ main(): void</div>

Name	Visibility	Return Type / Type	Definition
start(Stage primaryStage)	public	void	Method that starts the application.
main(String[] args)	public	void	Main method that defined in the Starter class.

3.3.3.CourseDAO

CourseDAO
- instance: CourseDAO
<div>+ getInstance(): CourseDAO</div> <div>+ push(): boolean</div>

+ getModels(): List<CourseModel>
+ deleteCourse(): boolean
+ editCourse(): boolean
+ getByYear(): List<CourseModel>
+ getByCourseCode(): CourseModel

Name	Visibility	Return Type / Type	Definition
instance	private	CourseDAO	The object which created staticly in the CourseDAO class.
getInstance()	public	CourseDAO	Method which gets the object created in the CourseDAO class.
push(CourseModel courseModel)	public	boolean	Method that provides to access fields related to course model.
getModels()	public	List<CourseModel>	Method that makes needed SQL calls to get all models and wrap all models into a list and return it.
deleteCourse(int courseID)	public	boolean	Method that makes needed SQL call to delete a course from the database and return result of the operation.
editCourse(int courseID)	public	boolean	Method that makes needed SQL call to edit a course from the database and return result of the operation.
getByYear(int year)	public	List<CourseModel>	Method that makes needed SQL call with a given year and return a List<CourseModel>
getByCourseCode(String courseCode)	public	CourseModel	Method that makes needed SQL call with given course code and return a course model.

3.3.4. EmailDAO

EmailDAO
- instance: EmailDAO
+ getInstance(): EmailDAO + push(): boolean + getEmailLists(): List<String> + deleteEmailLists(): boolean + getModels(): List<EmailModel>

Name	Visibility	Return Type / Type	Definition
instance	private	EmailDAO	The object which created staticly in

			the EmailDAO class.
getInstance()	public	EmailDAO	Method which gets the object created in the EmailDAO class.
push(EmailModel emailModel)	public	boolean	Method that provides to access fields related to email model.
getEmailLists()	public	List<String>	Method that makes needed data query to finds all email lists.
deleteEmailLists(String ListName)	public	boolean	Method that makes needed data query to deletes list from persistent storage.
getModels(List<String> groupList)	public	List<EmailModel>	Method that makes needed data query to collect all EmailModels(GroupList) in a List<EmailModel>

3.3.5. EventDAO

EventDAO
- instance: EventDAO
+ getInstance(): EventDAO + push(): boolean + getModels(): List<EventModel> + deleteEvent(): boolean + editEvent(): boolean

Name	Visibility	Return Type / Type	Definition
instance	private	EventDAO	The object which created staticly in the EventDAO class.
getInstance()	public	EventDAO	Method which gets the object created in the EventDAO class.
push(EventModel eventModel)	public	boolean	Method that provides to access fields related to event model.
getModels()	public	List<EventModel>	Method that makes needed data query to finds all events.
deleteEvent(int EventID)	public	boolean	Method that makes needed data query to deletes event from persistent storage.
editEvent(int EventID)	public	boolean	Method that makes needed SQL call to edit an event from the database and return result of the operation.

3.3.6. UserDao

UserDao
- instance: UserDao
+ getInstance(): UserDao + push(): boolean + getModels(): List<UserModel> + deleteUser(): boolean + login(): UserModel

Name	Visibility	Return Type / Type	Definition
instance	private	UserDao	The object which created statically in the UserDao class.
getInstance()	public	UserDao	Method which gets the object created in the UserDao class.
push(UserModel userModel)	public	boolean	Method that provides to access fields related to user model.
getModels()	public	List<UserModel>	Method that makes needed SQL calls to get all models and wrap all models into a list and return it.
deleteUser(int UserID)	public	boolean	Method that makes needed SQL call to delete a user from the database and return result of the operation.
login(String userEmail, String hashedPassword)	public	UserModel	Method that makes needed SQL call to finds matching user model in the database.

3.3.7. CourseModel

CourseModel
- courseID: int - courseCode: String - courseTitle: String - courseInstructor: String - position: String - courseEnable: boolean
+ getModels(): List<CourseModel> + deleteCourse(): boolean + editCourse(): boolean + getByYear(): List<CourseModel> + getByCourseCode(): CourseModel + push(): boolean

Name	Visibility	Return Type / Type	Definition
courseID	private	int	Defines a course ID
courseCode	private	String	Defines a course code
courseTitle	private	String	Defines a name of a course
courseInstructor	private	String	Defines a course instructor
position	private	String	Defines the position of a course in the schedule
courseEnable	private	boolean	Defines whether a course is enable or disable for a term
getModels()	public	List<CourseModel>	Method that accesses CourseDAO object and returns getted List<CourseModel>
deleteCourse(int CourseID)	public	boolean	Method that accesses CourseDAO object and deletes course for CourseDAO.
editCourse(intCourseID)	public	boolean	Method that accesses CourseDAO object and edits course for CourseDAO.
getByYear(int year)	public	List<CourseModel>	Method that accesses CourseDAO object and get a course by given year.
getByCourseCode(String courseCode)	public	CourseModel	Method that accesses CourseDAO object and get a course by given course code.
push()	public	boolean	Method that provides to pass course model as a parameter to push method of CourseDAO.

3.3.8. EmailModel

EmailModel
- emailID: int - nameSurname: String - email: String - grade: String - emailGroup: List<String>
+ getEmailLists(): List<String> + deleteEmailList(): boolean + getModels(): List<EmailModel> + push(): boolean

Name	Visibility	Return Type / Type	Definition
emailID	private	int	Defines an email ID
nameSurname	private	String	Defines the student's name and surname.
email	private	String	Defines an email
grade	private	String	Defines the grade of the student.
emailGroup	private	List<String>	Defines an email group
getEmailLists()	public	List<String>	Method that accesses EmailDAO object and returns gotten List<String>
deleteEmailList(String ListName)	public	boolean	Method that accesses EmailDAO object and delete email list.
getModels(List<String> groupList)	public	List<EmailModel>	Method that accesses EmailDAO object and gets all email models.
push()	public	boolean	Method that provides to pass course model as a parameter to push method of EmailDAO.

3.3.9. EventModel

EventModel
- eventID: String - eventName: String - eventDate: String - eventTime: String - eventLocation: String - eventSelection: boolean
+ getModels(): List<EventModel>

Name	Visibility	Return Type / Type	Definition
eventID	private	String	Defines an event ID.
eventName	private	String	Defines an event name.
eventDate	private	String	Defines an event date.
eventTime	private	String	Defines an event time.
eventLocation	private	String	Defines an event location.

eventSelection	private	boolean	Defines whether an event is selected or not.
getModels()	public	List<UserModel>	Method that accesses EventDAO object and returns List<EventModel>.

3.3.10. UserModel

UserModel
- userID: String - userEmail: String - userName: String - userTitle: String - userType: String
+ getModels(): List<UserModel> + deleteUser(): boolean + login(): UserModel + push(): boolean

Name	Visibility	Return Type / Type	Definition
userID	private	String	Defines a user ID
userEmail	private	String	Defines a user email
userName	private	String	Defines a user name
userTitle	private	String	Defines a user title
userType	private	String	Defines a user type
getModels()	public	List<UserModel>	Method that accesses UserDAO object and returns getted List<UserModel>
deleteUser(int userID)	public	boolean	Method that accesses UserDAO object and delete user.
login(String userEmail, String hashedPassword)	public	UserModel	Method that finds matching user model in the database.
push()	public	boolean	Method that provides to pass course model as a parameter to push method of UserDAO.

3.3.11. ComponentContainerController

ComponentContainerController takes care of the main border that contains the views.

ComponentContainerController
-componentContainer: BorderPane
+ addComponent(): void + reset(): ComponentContainerController - loadFXML(): ComponentContainerController

Name	Visibility	Return Type / Type	Definition
componentContainer	private	BorderPane	The main border that contains the views
addComponent(Node component, ContainerPosition position)	public	void	Adds components to the border
reset()	public	ComponentContainer Controller	Resets the border
loadFXML()	private	ComponentContainer Controller	Loads the fxml script -ComponentContainer.fxml-

3.3.12. ContainerPosition

<<enumeration>> ContainerPosition
CENTER TOP BOTTOM LEFT RIGHT

3.3.13. CourseEntryController

CourseEntryController sets the line for course information.

CourseEntryController
-courseId: Label -courseTitle: Label -courseInstructor: Label -active: boolean -courseCode: int
+deleteCourse(): void +editCourse(): void -loadFXML(): void

Name	Visibility	Return Type / Type	Definition
courseId	private	Label	Label name of course code
courseTitle	private	Label	Label name of course title
courseInstructor	private	Label	Label name of course instructor
active	private	boolean	Activation of course for current semester.
courseCode	private	int	Course code that is unique.
deleteCourse(ActionEvent e)	public	void	Disables the course after delete action.
editCourse(ActionEvent e)	public	void	Shows a pop up window for editing the course after editCourse action.
loadFXML()	private	void	Loads the fxml script -CourseEntry.fxml-

3.3.14. CourseTableController

CourseTableController sets the table for course information.

CourseTableController
-table: VBox
+addItem: CourseTableController -loadFXML(): void

Name	Visibility	Return Type / Type	Definition
table	private	VBox	The table that contains courses
addItem(CourseEntry Controller entry)	public	CourseTableController	Adds the course information to the table line by line
loadFXML()	private	void	Loads the fxml script -CourseTable.fxml-

3.3.15. ManageCourseController

ManageCourseController sets the view for course information.

ManageCourseController
-courseTable: VBox
+addCourse():void

Name	Visibility	Return Type / Type	Definition
courseTable	private	VBox	Main view that contains course table and add course button
addCourse(ActionEvent event)	public	void	Shows a pop up window for adding a new course after addCourse action

3.3.16. CoursePopupBuilder

CoursePopupBuilder
+build(): CoursePopupController

Name	Visibility	Return Type / Type	Definition
build()	public	CoursePopupController	It builds CoursePopupController.

3.3.17. CoursePopupController

CoursePopupController sets the view for editing and adding a course.

CoursePopupController
+applyEdit():void

Name	Visibility	Return Type / Type	Definition
applyEdit(ActionEvent e)	public	void	Applies the changes on the course information

3.3.18. ScheduleDay

<<enumeration>> ScheduleDay
MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY

3.3.19. ScheduleController

ScheduleController sets the view for course schedule table.

ScheduleController
-monday: List<Label> -tuesday: List<Label> -wednesday: List<Label> -thursday: List<Label> -friday: List<Label>
+getScheduleOf(): List<Label> -loadFXML(): void

Name	Visibility	Return Type / Type	Definition
monday	private	List<Label>	Holds the monday lecture hours
tuesday	private	List<Label>	Holds the tuesday lecture hours

wednesday	private	List<Label>	Holds the wednesday lecture hours
thursday	private	List<Label>	Holds the thursday lecture hours
friday	private	List<Label>	Holds the friday lecture hours
getScheduleOf(ScheduleDay day)	public	List<Label>	Gets the lectures of a day
loadFXML()	private	void	Loads the fxml script -Schedule.fxml-

3.3.20. ScheduleCourseBuilder

ScheduleCourseBuilder
+build(): ScheduleCoursesController

Name	Visibility	Return Type / Type	Definition
build()	public	ScheduleCoursesController	It builds ScheduleCoursesController .

3.3.21. ScheduleCoursesController

ScheduleCoursesController sets the main view for scheduling course.

ScheduleCoursesController
-schedule: VBox -yearComboBox: ComboBox<Integer> -courseComboBox: ComboBox<String> -scheduleTable: ScheduleController
+schedule(): void +apply(): void +yearOptionChanged(): void -setYearOption(): ScheduleCoursesController -updateCourseOptions(): ScheduleCoursesController -updateSchedule(): ScheduleCoursesController

Name	Visibility	Return Type / Type	Definition
schedule	private	VBox	Main view that contains year combobox, course combobox, course schedule, schedule button and apply button.
yearComboBox	private	ComboBox<Integer>	Contains years.
courseComboBox	private	ComboBox<String>	Contains course codes.
scheduleTable	private	ScheduleController	Holds the course schedule table.
schedule(ActionEvent event)	public	void	Shows a pop up window for scheduling courses.
apply(ActionEvent event)	public	void	Applies the changes on course schedule.
setYearOption()	private	ScheduleCoursesController	Adds course years to yearComboBox.
yearOptionChanged(ActionEvent event)	public	void	Gets the year from yearComboBox and updates schedule according to year.
updateCourseOptions(int courseYear)	private	ScheduleCoursesController	Updates inside of the courseComboBox according to year.
updateSchedule(int scheduleYear)	private	ScheduleCoursesController	Shows the course schedule according to chosen year.

3.3.22. AnnouncementController

It builds the view for Announcement section and keeps synchronized with the website.

AnnouncementController
- finalEventTable : VBox
+ syncEvents() : void + sendNotification() : void

Name	Visibility	Return Type / Type	Definition
------	------------	--------------------	------------

finalEventTable	private	Vbox	It defines the view of announcement table
syncEvents(ActionEvent)	public	void	Synchronize the data with the website
sendNotification(ActionEvent)	public	void	Builds a popup in order to select the email list groups

3.3.23. EventEntryController

EventEntryController class sets the view for each entry in the event table.

EventEntryController
- eventName : Label - eventDate : Label - eventTime : Label - eventLocation : Label - eventSelection : Label

Name	Visibility	Return Type / Type	Definition
eventName	public	Label	In announcement menu, it defines the “event name” column.
eventDate	public	Label	In announcement menu, it defines the “event date” column.
eventTime	public	Label	In announcement menu, it defines the “event time” column.
eventLocation	public	Label	In announcement menu, it defines the “event location” column.
eventSelection	public	Label	In announcement menu, it defines the “event selection” column.

3.3.24. EventTableController

It creates the view for Event Table in the Announcement section and adds item to the table.

EventTableController
- eventTable : Vbox
+ addItem() : EventTableController - loadFXML() : void

Name	Visibility	Return Type / Type	Definition
eventTable	private	Vbox	It defines the view of announcement table's one entry
addItem(EventEntryController)	public	EventTableController	It adds entry to the Event Table
loadFXML()	private	void	Loads the fxml script -EventTable.fxml-

3.3.25. SendNotificationBuilder

It builds the view for popup.

SendNotificationBuilder
+ build() : SendNotificationPopupController

Name	Visibility	Return Type / Type	Definition
build()	public	SendNotificationPopupController	It makes a call to SendNotificationPopupController

3.3.26. SendNotificationPopupController

It sends the selected events to the selected email lists.

SendNotificationPopupController
+ send() : void

Name	Visibility	Return Type / Type	Definition
send(ActionEvent)	public	void	It sends events to the selected email list groups

3.3.27. EditEmailListsController

It creates the view for Edit Email List in Manage Emails. Then it allows the make search, delete, add operations on selected lists.

EditEmailListsController
- emailList : VBox
+ deleteList() : void + addEmail() : void + searchDelete() : void - findSelectedEmailLists() : List<String> - getSelectedGroups() : List<String> - displayEmailLists() : void

Name	Visibility	Return Type / Type	Definition
emailList	private	Vbox	It contains the views of each email list box
deleteList(ActionEvent)	public	void	It deletes the selected email lists
addEmail(ActionEvent)	public	void	It adds an email to the selected email lists
searchDelete(ActionEvent)	public	void	It either deletes the the selected email lists or searches a name through selected lists
findSelectedEmailLists()	private	List<String>	It finds out which email lists are selected and returns a list of string that contains the selected email lists
getSelectedGroups()	private	List<String>	It accesses to the selected groups.
displayEmailLists()	private	void	It displays the updated email list from the data taken from database

3.3.28. Email List Chooser

It opens the file explorer.

EmailListChooser
+ start() : void

Name	Visibility	Return Type / Type	Definition
------	------------	--------------------	------------

start(Stage)	public	void	It opens file explorer to choose a file, that includes a email list, from computer.
--------------	--------	------	---

3.3.29. ImportEmailListController

It imports the selected files and parses them accordingly.

ImportEmailListController
- listFile : TextField
+ importList() : void - openFileChooser() : File - parse() : void

Name	Visibility	Return Type / Type	Definition
listFile	private	TextField	It is a file to be loaded.
importList(ActionEvent)	public	void	It imports the file from local computer.
openFileChooser()	private	File	It takes the choosen file.
parse(File)	private	void	It parses the file.

3.3.30. Manage Emails Controller

It creates the sections when Manage Emails button is pressed

ManageEmailsController
- importEmailList : VBox - editEmailLists : VBox
+ addSection() : ManageEmailsController

Name	Visibility	Return Type / Type	Definition
importEmailList	private	Vbox	It creates the view for Import Email List section box
editEmailLists	private	Vbox	It creates the view for Edit Email List section box

addSection(Node)	public	ManageEmailsController	It creates the general view for Manage Emails section.
------------------	--------	------------------------	--

3.3.31. User Management Controller

UserManagementController
- userType: ComboBox<String> - email: TextField - nameSurname: TextField - title: TextField - password: PasswordField - passwordAgain: PasswordField - showUsers: Button - createUser: Button
+ UserManagementController(): void + showUser(): void + createUser(): void + setUserTypeOption(): void

Name	Visibility	Return Type / Type	Definition
userType	private	ComboBox<String>	Defines a user type.
email	private	TextField	Defines a user email.
nameSurname	private	TextField	Defines the user's name and surname.
title	private	TextField	Defines a user title.
password	private	PasswordField	Defines a user password.
passwordAgain	private	PasswordField	Defines a user password secondly.
showUser	private	Button	When the user presses this button, he / she will see the list of users.
createUser	private	Button	When the user presses this button, it creates a new user using the createUser() method.
showUser()	public	void	Method lists registered users.

createUser()	public	void	Method that creates a user.
setUserTypeOption()	public	void	Keeps user types in a list and assigns a type to the newly added user.

3.3.32. User Entry Controller

UserEntryController
- nameSurname: Label - email: Label - userType: Label - userID: int
+ UserEntryController() + getUserID(): int + setUserID(): UserEntryController + setNameSurname(): UserEntryController + setUserEmail(): UserEntryController + setUserType(): UserEntryController + delete(): void + loadFXML(): UserEntryController

Name	Visibility	Return Type / Type	Definition
nameSurname	private	Label	Defines the student's name and surname.
email	private	Label	Defines a user email.
userType	private	Label	Defines a user type.
userID	private	int	Defines a user ID.
getUserID()	public	int	Method that accesses userDAO Object and returns getted int.
setUserID(int userID)	public	UserEntryControler	It assign the parameter to the user id.
setNameSurname(String userNameSurname)	public	UserEntryControler	It assign the parameter to the nameSurname.
setUserEmail(String userEmail)	public	UserEntryControler	It assign the parameter to the email.
setUserType(String uType)	public	UserEntryControler	It assign the parameter to the user type.

delete(ActionEvent e)	public	void	Call UserModel static delete method.
LoadFXML(String fxmlFileName)	public	UserEntryController	Loads the fxml script.

3.3.33. User Popup Builder

UsersPopupBuilder
<div>+ UsersPopupBuilder()</div> <div>+ build(): UsersPopupBuilder</div>

Name	Visibility	Return Type / Type	Definition
build()	public	UsersPopupBuilder	It builds the general view of the system.

3.3.34. Users Popup Controller

UsersPopupController
<div>- usersTable: VBox</div> <div>+ UsersPopupController()</div> <div>+ addItem(): UsersPopupController</div>

Name	Visibility	Return Type / Type	Definition
userTable	private	VBox	It defines a table for users.
UsersPopupController()	public		It' constructor of Users Popup Controller class.
addItem(Node item)	public	UsersPopupController	Adds item.

3.3.35. Accordion MN Controller

AccordionMNController
- menuCommunicator: IMenuCommunicator
- navigationSelection: String
+ AccordionMNController()
+ selection(): void
+ setNavigationSelection(): void
+ getSelection(): String
+setMenuCommunicator(): void
- loadFXML(): AccordionMNController

Name	Visibility	Return Type / Type	Definition
menuCommunicator	private	IMenuCommunicator	It defines the each component of the menu.
navigationSelection	private	String	holds the navigation selection as a string.
selection(MouseEvent mouseEvent)	public	void	Get the clicked item then read its id to identify it.
setNavigationSelection(String selection)	public	void	It sets the navigation of the selection.
getSelection()	public	String	It gets the navigation of the selection.
setMenuCommunicator(IMenu Communicator menuCommunicator)	public	void	It sets the menu communicator.
loadFXML()	private	AccordionMNController	Loads the fxml script

3.3.36. Abstract In Window Popup Controller

AbstractInWindowPopupController
+ AbstractInWindowPopupController()
+ setPoupDestination(): AbstractInWindowPopupController
+ load(): AbstractInWindowPopupController
+ show(): AbstractInWindowPopupController
+ close(): AbstractInWindowPopupController
+ setPopupTitle(): AbstractInWindowPopupController
+ getPopupTitle(): String
+ loadFXML(): AbstractInWindowPopupController
+isActive(): Boolean
+isLoading(): Boolean

Name	Visibility	Return Type / Type	Definition
setPoupDestination(StackPane destination)	public	AbstractInWindow PopupController	Allows popup destination to be updated.
load(VBox poup)	public	AbstractInWindow PopupController	The load the popup.
show()	public	AbstractInWindow PopupController	This method allows the windows to be displayed.
close()	public	AbstractInWindow PopupController	This method closed the popup.
setPopupTitle(String title)	public	AbstractInWindow PopupController	Allows popup name to be updated.
getPopupTitle()	public	String	Method which gets the popups scene name.
loadFXML()	public	AbstractInWindow PopupController	Loads the fxml script.
isActive()	public	Boolean	Activation of information for the popup scene.
isLoading()	public	Boolean	Checks whether the popup is loaded.

AbstractPopup
+ AbstractPopup() - loadFXML(): AbstractPopup

Name	Visibility	Return Type / Type	Definition
loadFXML()	private	AbstractPopup	Loads the fxml script

3.3.38. In Window Popup Manager

InWindowPopupManager
- instance: AbstractInWindowPopupController - loadedPopup: VBox - popupTitle: Label - popupDestination: StackPane - active: Boolean - loaded: Boolean - showUsers: Button - createUser: Button
+ getInstance(): AbstractInWindowPopupController - InWindowPopupManager() + isActive(): Boolean + isLoading(): Boolean +setPopupTitle(): AbstractInWindowPopupController + getPopupTitle(): String + setPoupDestination():AbstractInWindowPopupController + load(): AbstractInWindowPopupController +show(): AbstractInWindowPopupController + close(): AbstractInWindowPopupController # loadFXML(): AbstractInWindowPopupController

Name	Visibility	Return Type / Type	Definition
Instance	private	AbstarctInWindowPo pUpController	The object which created staticly in theAbstractInWindowPopup Controller
loadedePopup	private	VBox	Keeps the pop-up window loaded.
popupTitle	private	Label	Holds the title of the pop-up window.
popupDestination	private	StackPane	Holds the destination of the pop-up window.
active	private	Boolean	Keeps information whether the pop-up page is active or not.

loaded	private	Boolean	Keeps information whether the pop-up page is loaded or not.
showUsers	private	Button	When this button is pressed, users are listed.
createUser	private	Button	When this button is pressed, the popup opening used to create the new user opens.
getInstance()	public	AbstarctInWindowPo pUpController	Method which gets the object created in the AbstractInWindowPopup Controller.
isActive()	public	Boolean	Activation of information for the popup scene.
isLoading()	Public	Boolean	Checks whether the pop-up page is loaded.
setPopupTitle(String title)	Public	AbstarctInWindowPo pupController	Allows popup name to be updated.
getPopupTitle()	Public	String	Method which gets the popups scene name.
setPopupDestination(StackPane destination)	Public	AbstarctInWindowPo pupController	Allows popup destination to be updated.
load(VBox popup)	Public	AbstarctInWindowPo pupController	The load the popup.
show()	Public	AbstarctInWindowPo pupController	This method allows the windows to be displayed.
close()	Public	AbstarctInWindowPo pupController	This method closed the popup.
loadFXML()	protected	AbstarctInWindowPo pupController	Loads the fxml script

3.3.39. Status Bar Controller

StatusBarController
+ StatusBarController() + logout(): void + publish(): void - loadFXML(): String

Name	Visibility	Return Type / Type	Definition
logout(ActionEvent event)	public	void	It provides the logout operation for the user from the system.
publish(ActionEvent event)	public	void	Pressing the publish button will publish the courses.
loadFXML(String fxmlFileName)	private	String	Loads the fxml script

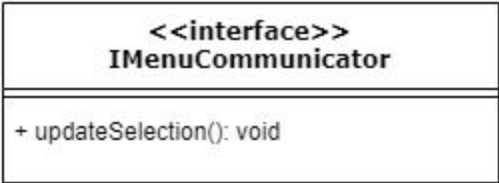
3.3.40. Component Communicator

ComponentCommunicator
- container: ComponentContainerController - menuContentList: List<Pane> - currentSelection: String
+ ComponentCommunicator() + updateSelection(): void + refresh(): void - searchFor(): Pane + setComponentContainerController(): void

Name	Visibility	Return Type / Type	Definition
container	private	ComponentContainer Controller	The main border that contains the views.
menuContentList	private	List<Pane>	It defines the list for holding to content.
currentSelection	private	String	Holds the current selection.
updateSelection(String menuContentID)	public	void	This method provides updates the selection.

reflesh()	public	void	It reloads the page.
searchFor(String menuID)	public	Pane	Method that provides to search for component communicator class and returns the pane.
setComponentContainerController(ComponentContainerController container)	public	void	It assign the parameter to the container.

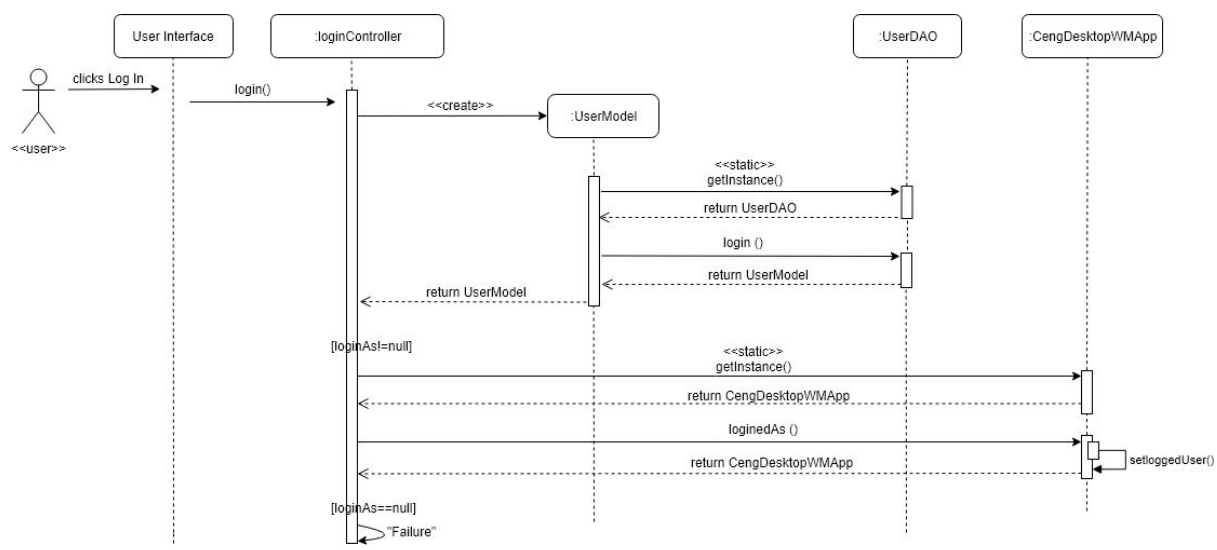
3.3.41. IMenu Communicator



Name	Visibility	Return Type / Type	Definition
updateSelection(String selection)	public	void	This method provides updates the selection.

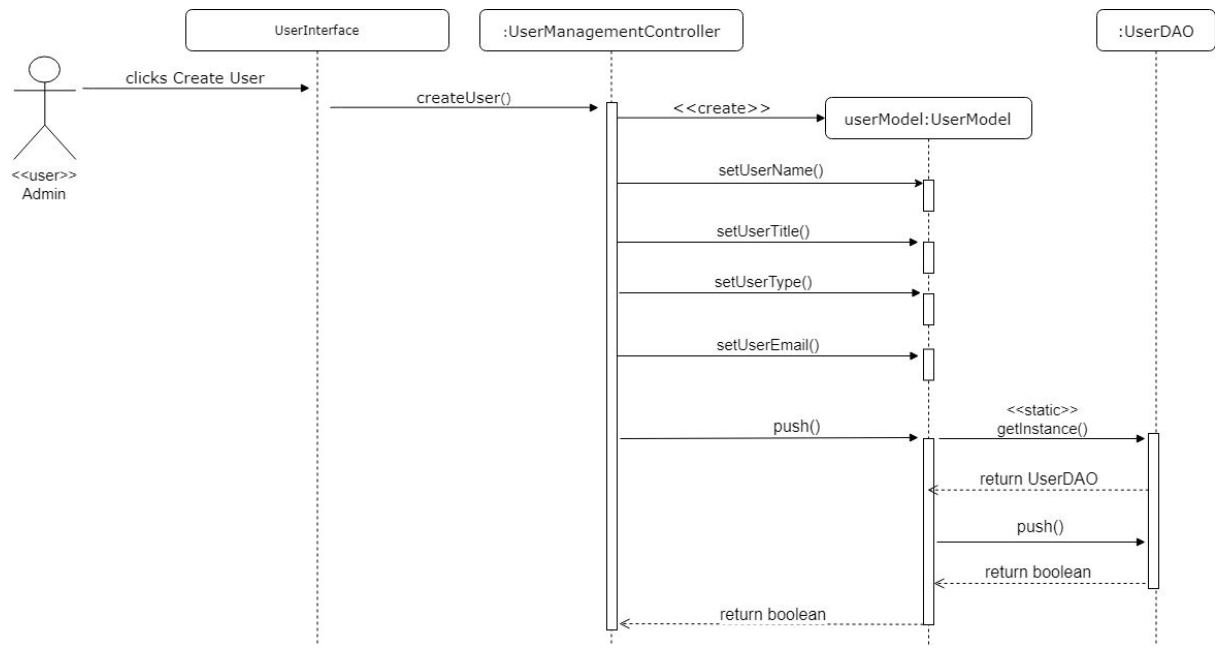
3.4. Interaction ViewPoint

3.4.1. Login Sequence Diagram

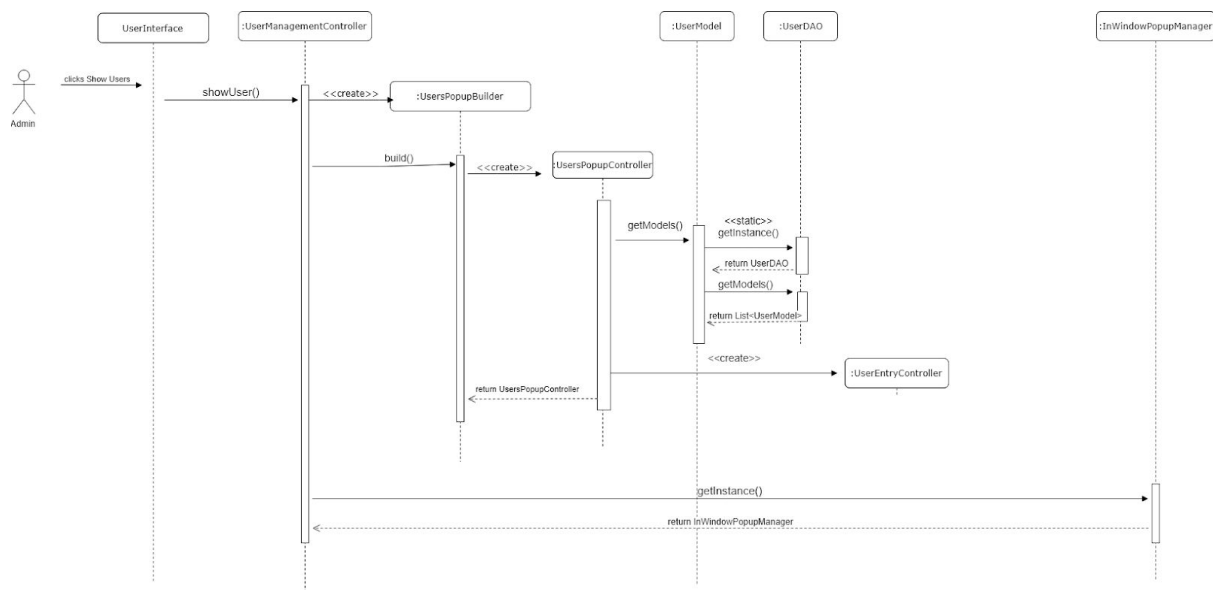


3.4.2. User Management

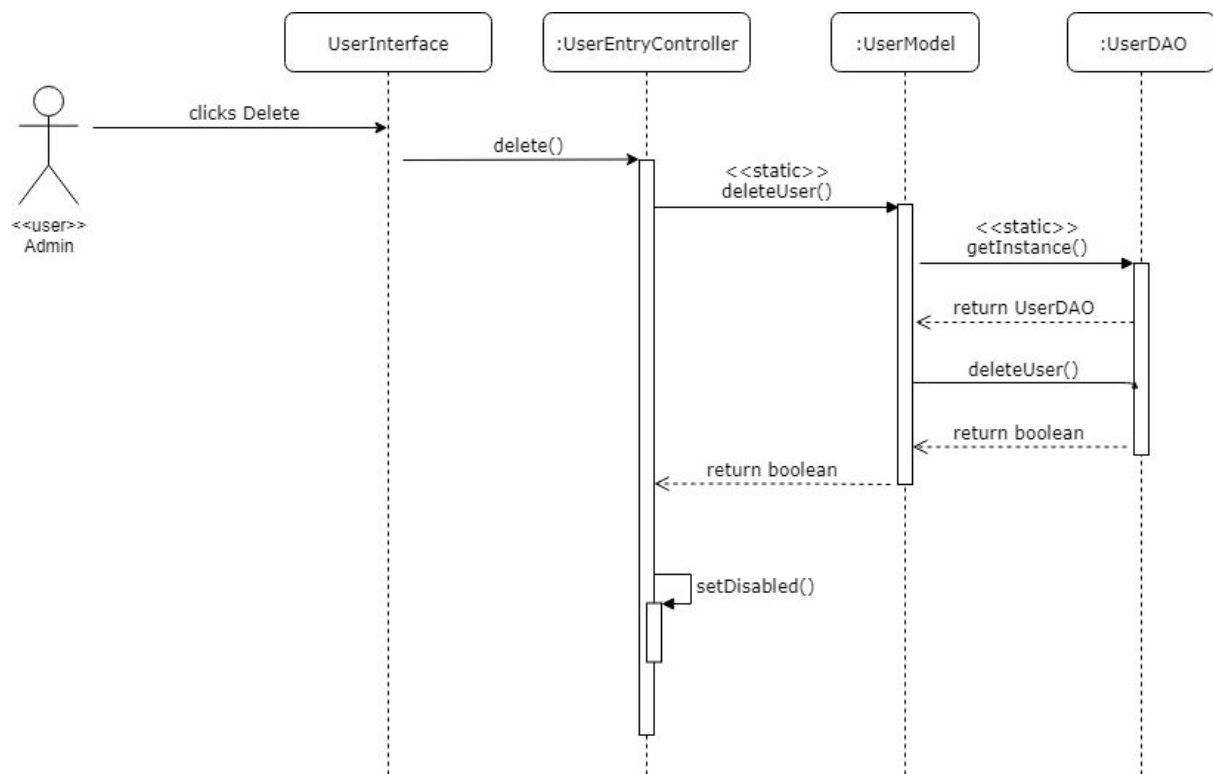
3.4.2.1. Create User Sequence Diagram



3.4.2.2. Show User Sequence Diagram

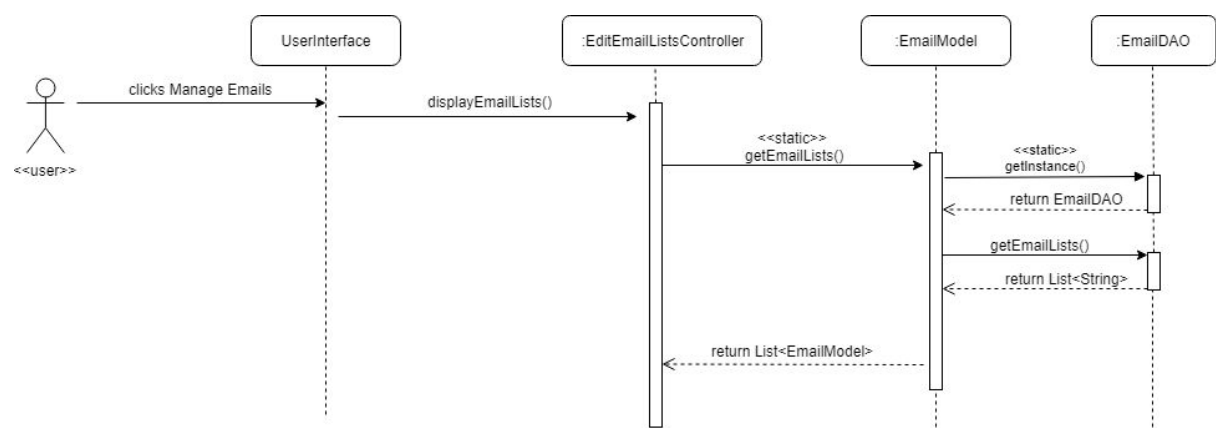


3.4.2.3. Delete User Sequence Diagram

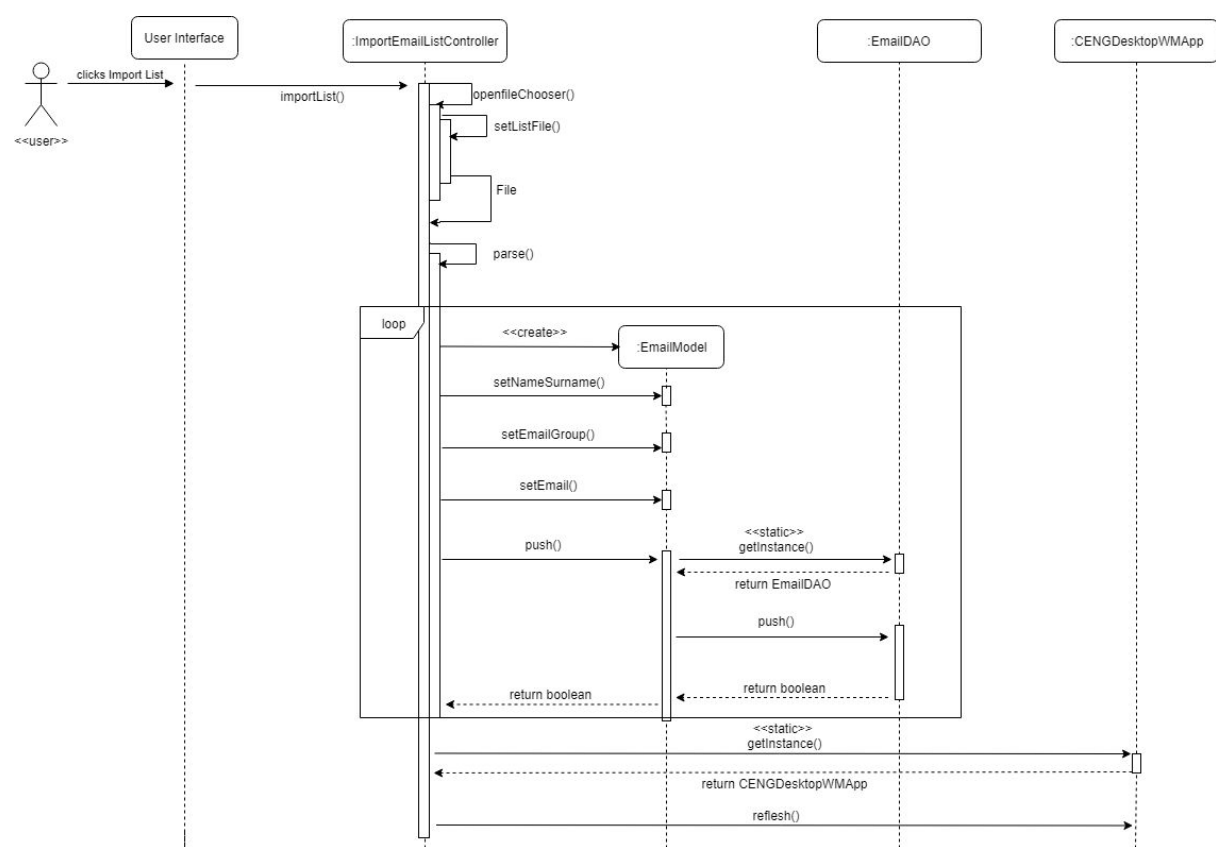


3.4.3. Notification Management

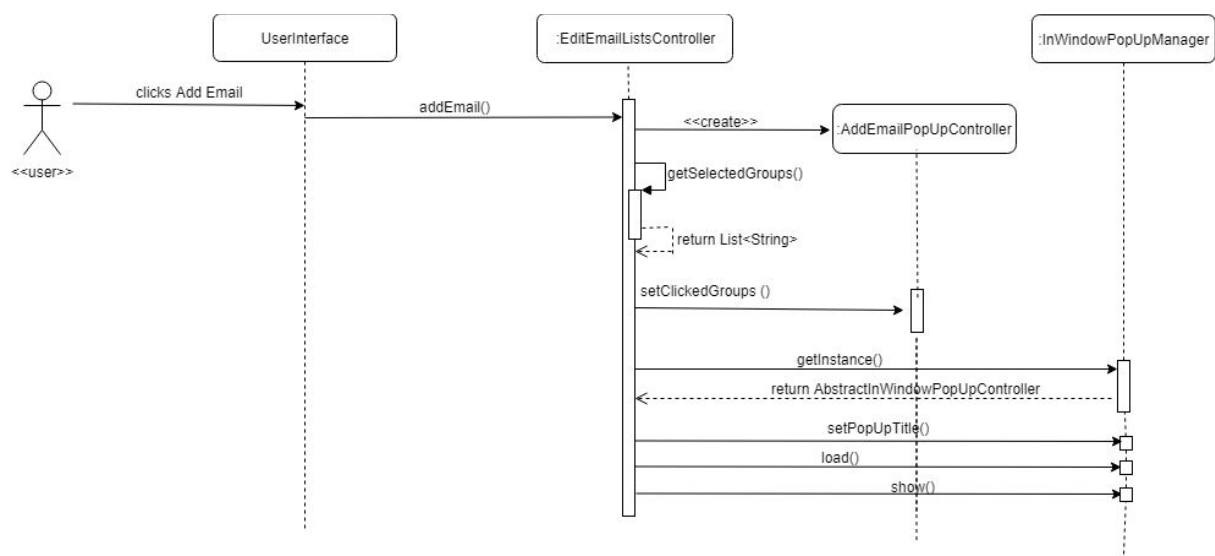
3.4.3.1. Display Email List Sequence Diagram



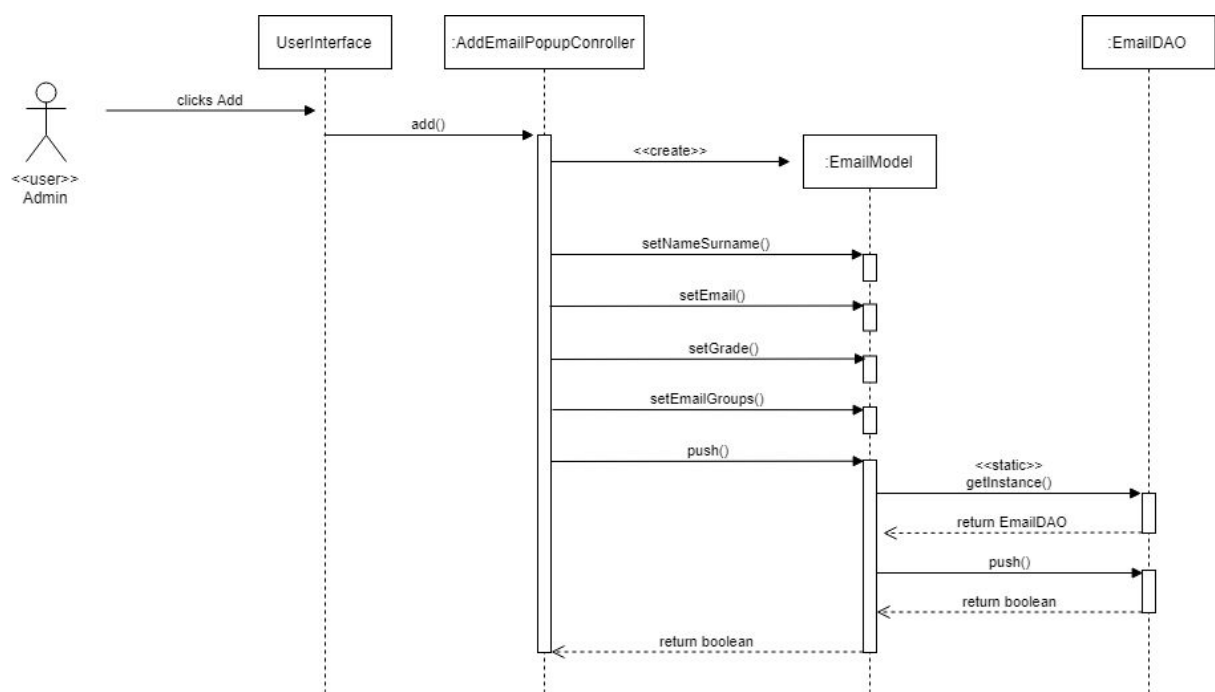
3.4.3.2. Import Email List Sequence Diagram



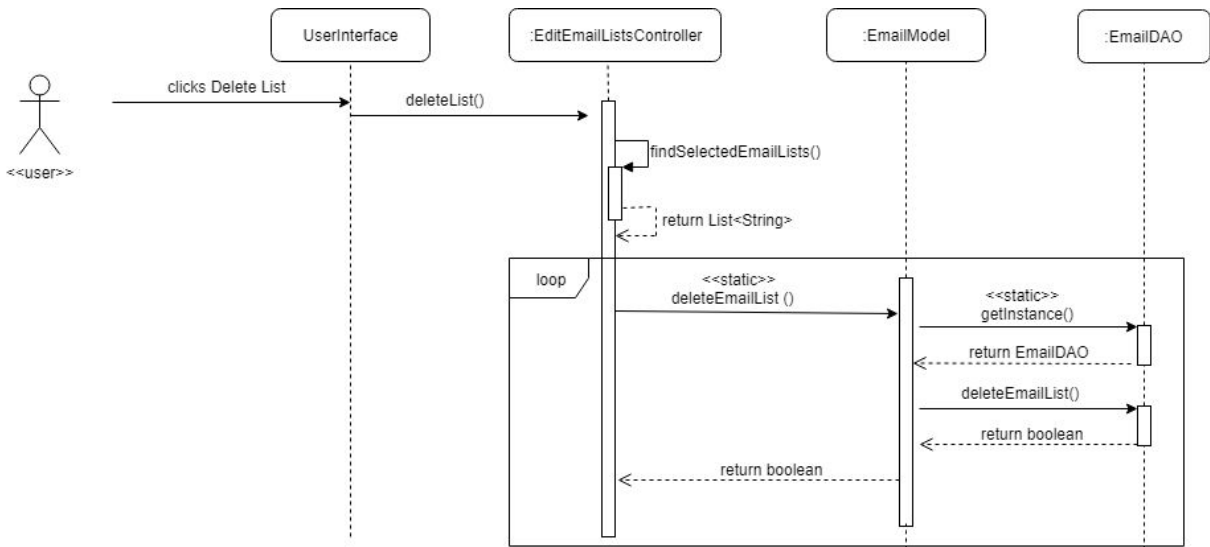
3.4.3.3. Add Email Sequence Diagram (to enter the add email pop up)



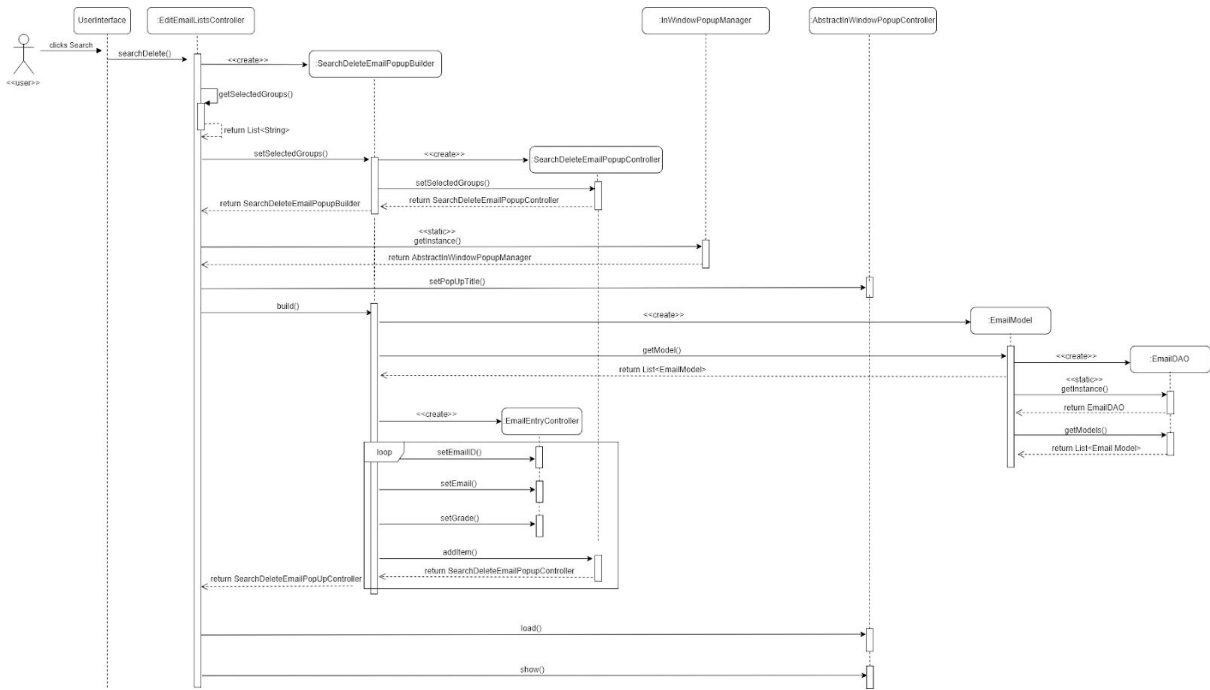
3.4.3.4. Add Email Sequence Diagram (to add an email)



3.4.3.5. Delete Email List Sequence Diagram

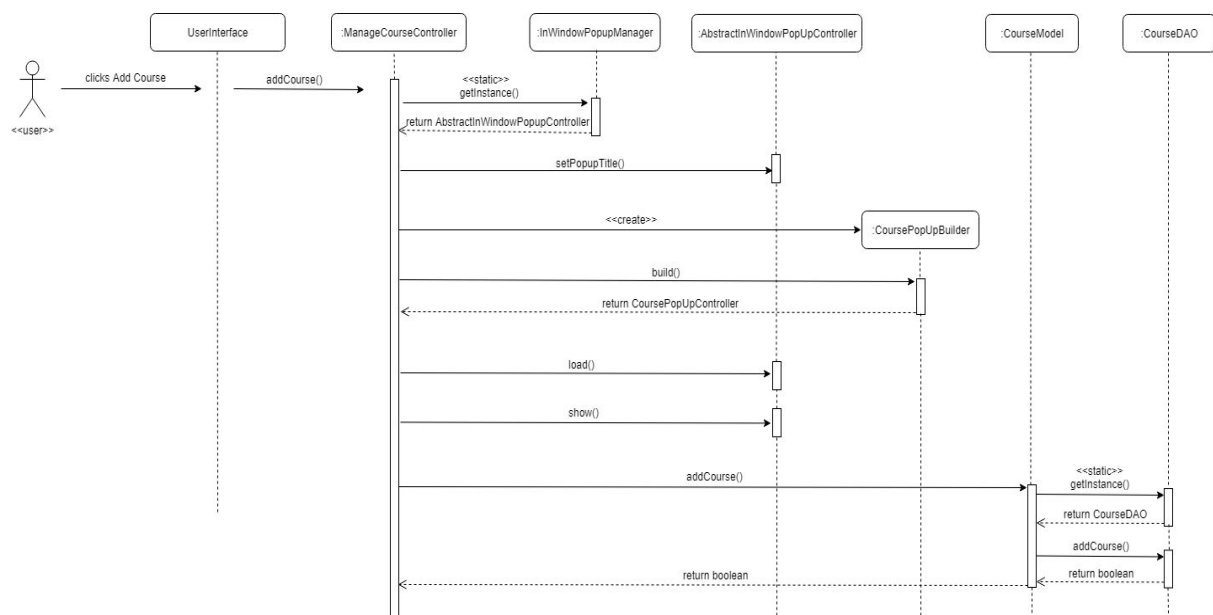


3.4.3.6. Search and Delete Email Sequence Diagram

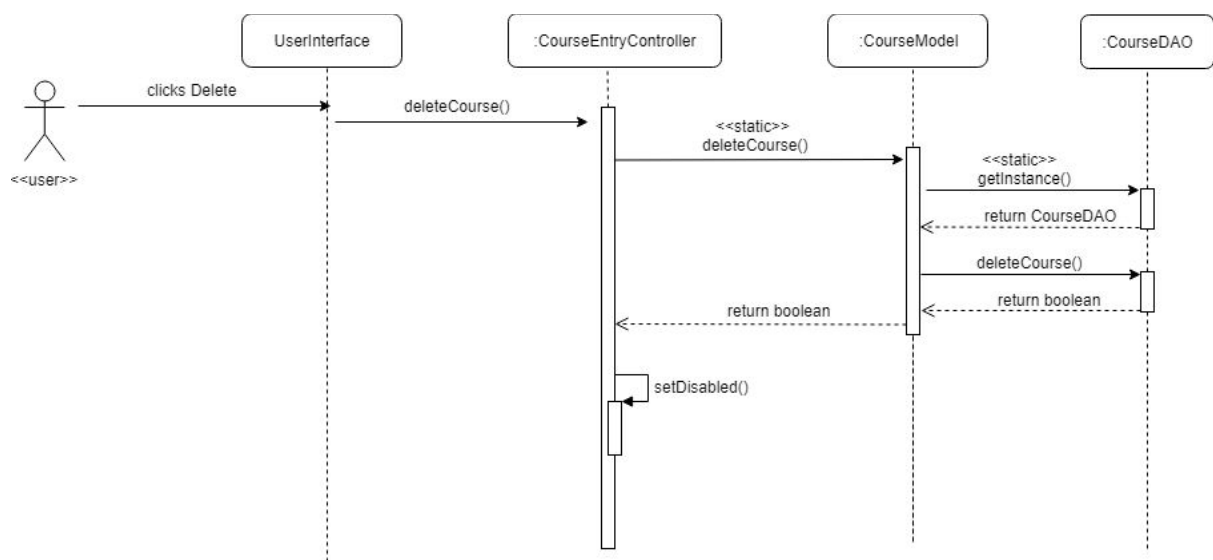


3.4.4. Course Management

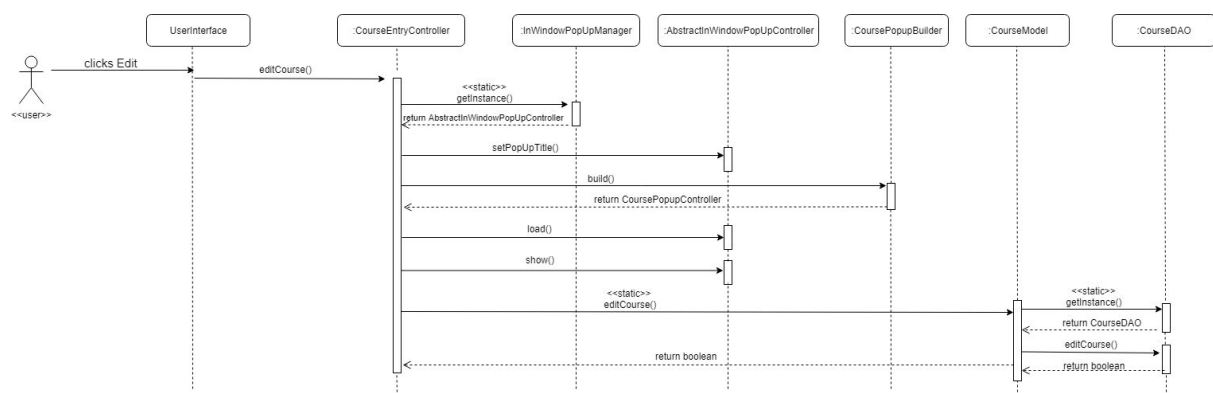
3.4.4.1. Add Course Sequence Diagram



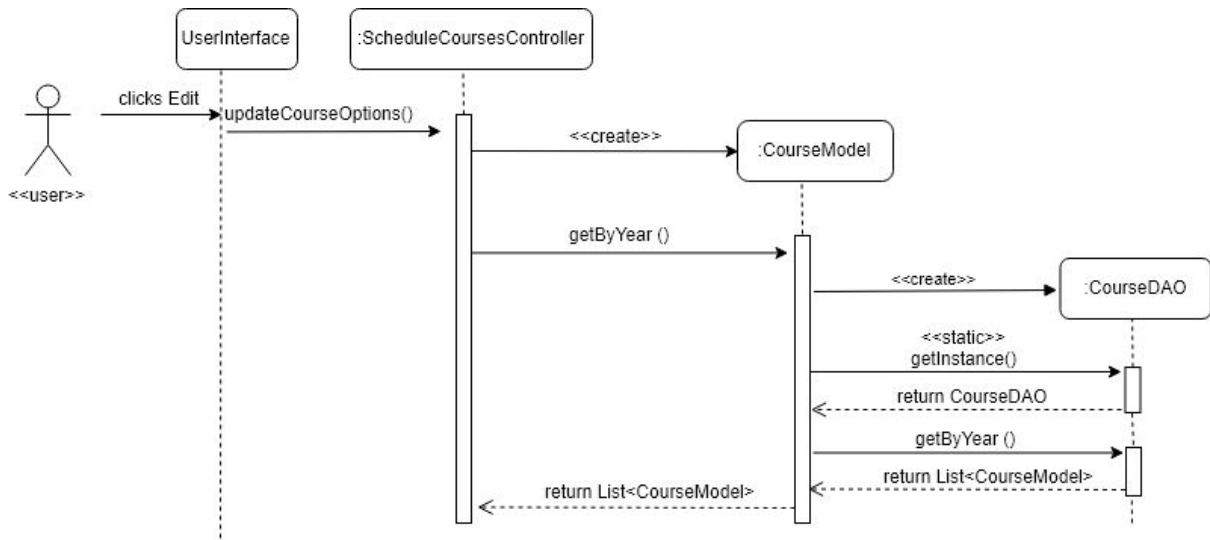
3.4.4.2. Delete Course Sequence Diagram



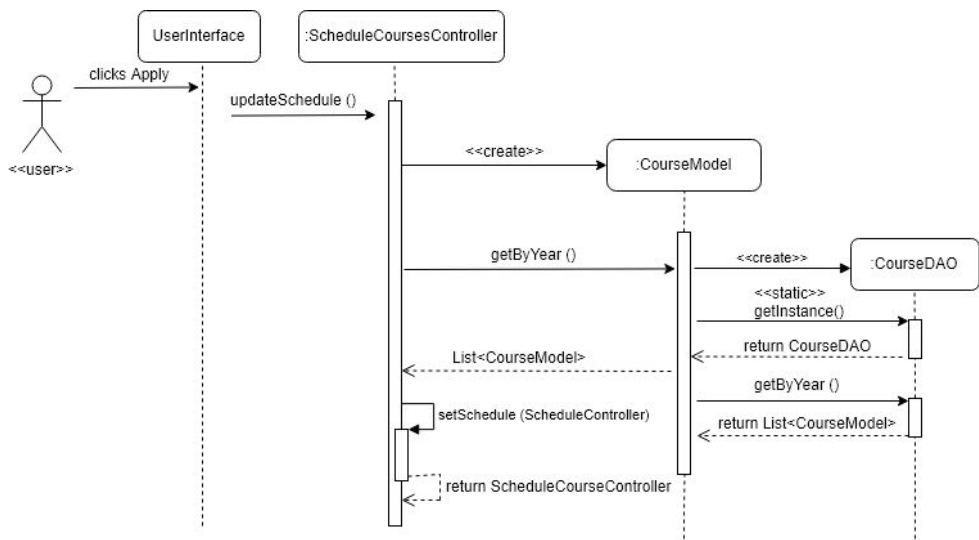
3.4.4.3. Edit Course Sequence Diagram



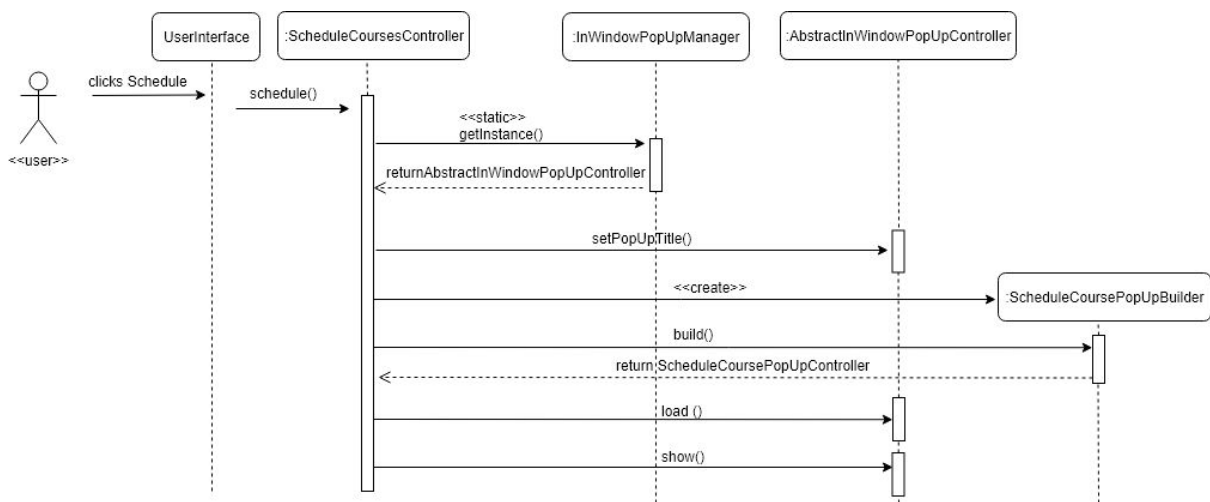
3.4.4.4. Update Course Options Sequence Diagram



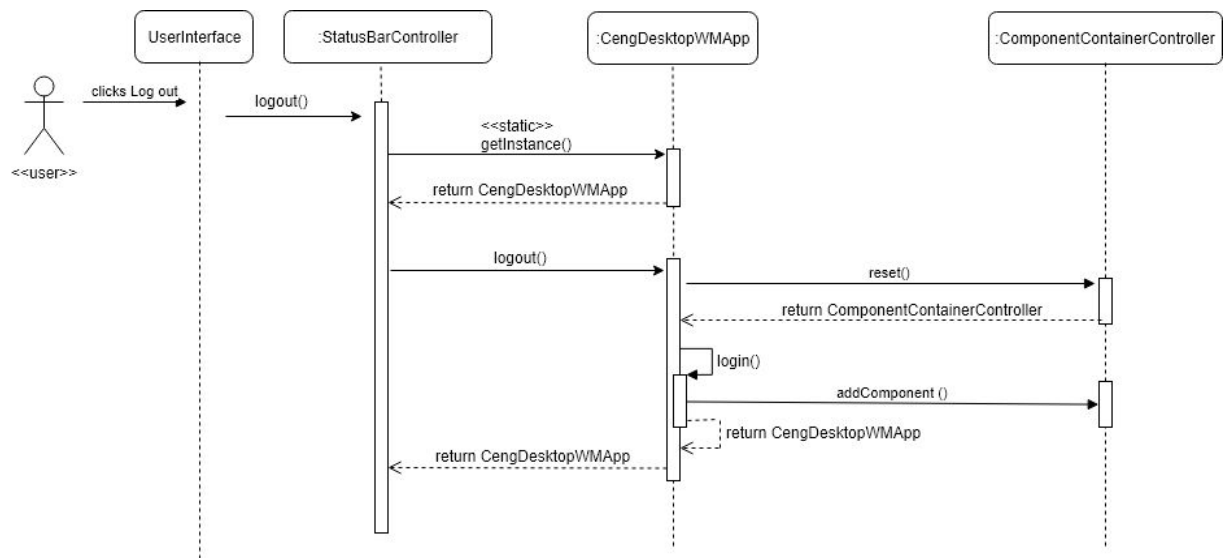
3.4.4.5. Update Schedule Sequence Diagram (to edit the schedule that filled before)



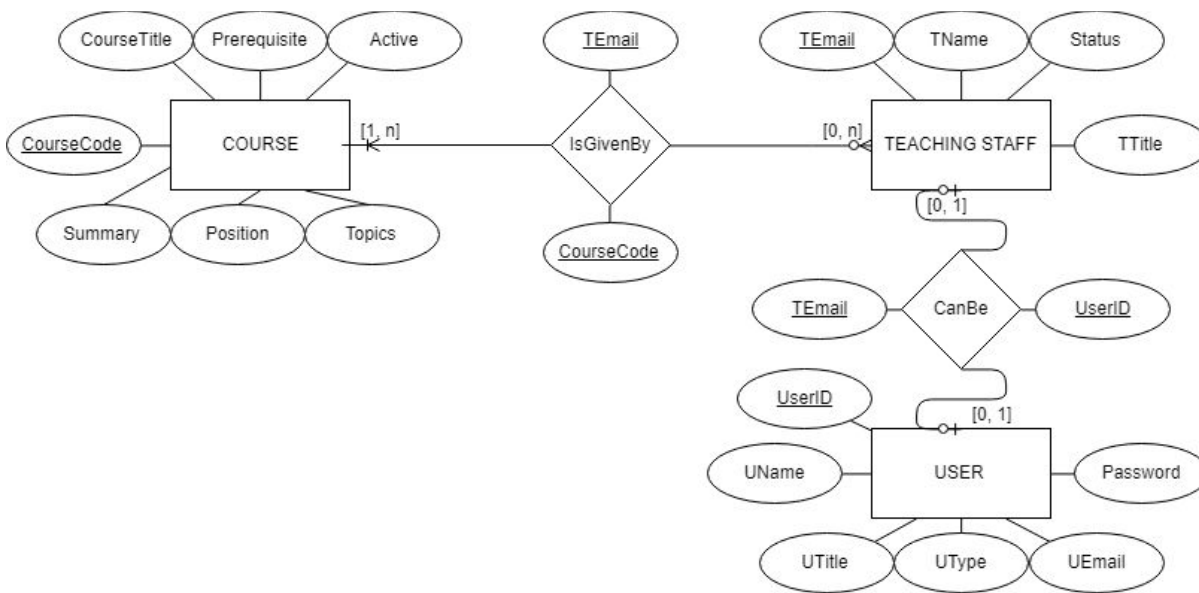
3.4.4.6. Schedule Sequence Diagram (to create a new schedule)



3.4.5. Logout Sequence Diagram



E-R Diagram



3.5.1. Reduced Relation

Course (CourseCode, CourseTitle, Prerequisite, Topics, Active, Summary, Position)
TeachingStaff (TEmail, TTitle, TName, Status, **UserID**)
User (UserID, UEmail, UTitle, UName, UType, Password)
IsGivenBy (CourseCode, TEmail)
CanBe(UserID, TEmail)

3.5.2. Entities

3.5.2.1 Course Data Entity

Data Item	Type	Description	Comment
Course Code	String	The unique code for a course.	Used as a primary key in the database.
Course Title	String	The name of the course.	None.
Prerequisite	String	Course that must be completed with a satisfactory grade before enrolling another course.	May be several prerequisite courses.
Topics	String	It tells the each week's course content in a compact way.	There are 14 topics for a semester.
Active	Bit	It indicates if a course is active or not for that semester	It takes 0 or 1 values.
Summary	String	It gives a preview for the course.	None.
Position	Pointer	By this integer number, we will know where each lecture should be located in the weekly course schedule.	None.

3.5.2.2 Teaching Staff Entity

Data Item	Type	Description	Comment
TEmail	String	Email of teaching staff of CENG department.	Used as a primary key in database
TTitle	String	Levels of professional expertise	None.
TName	String	Name of teaching staff of CENG department.	None.
Status	Integer	It indicates if teaching staff is a faculty member or research assistant.	None.
UserID	Integer	ID number of each user.	Used as a foreign key in database

3.5.2.3 User Entity

Data Item	Type	Description	Comment
UserID	Integer	ID number of each user.	Used as a primary key in database
UEmail	String	Email of users	None.
UTitle	String	It indicates that if user is a faculty member, research assistant or administrative staff.	None.
UName	String	Name of the users	None.
UType	String	It indicates if user is a Content Manager or Admin.	None.
Password	String	Password of users in hashed format.	None.

3.5.2.4 IsGivenBy

Data Item	Type	Description	Comment
Course Code	String	The unique code for a course.	Used as a primary key in database
TEmail	String	Email of teaching staff of CENG department.	Used as a primary key in database

UI Screenshots of Desktop App

The image shows a web browser window titled "CENG Desktop Website Management App (IZTECH)". The page has a light gray background. In the center, there is a large, faint, black outline of a person's head and shoulders, serving as a placeholder for a profile picture. Below this, the word "Email" is displayed in a small, black, sans-serif font. Underneath the email label is a white text input field with a thin gray border, containing the email address "busenuraktivilav@std.iyte.edu.tr". Below the email field, the word "Password" is displayed in the same font. Underneath the password label is a white text input field with a thin blue border, filled with ten black dots to represent a password. At the bottom center of the form area is a gray rectangular button with rounded corners and the text "Log In" in a black, sans-serif font.


Login Page UI

The screenshot displays the 'CENG Desktop Website Management App (IZTECH)' interface. On the left, a sidebar contains three menu items: 'User Management' (highlighted with a blue arrow), 'Notification Management', and 'Course Management'. The main content area is titled 'User Management' and features a form for adding a new user. The form includes the following fields and controls:

- User Type:** A dropdown menu with a downward arrow.
- Email:** A text input field.
- Name & Surname:** A text input field.
- Title:** A text input field.
- Password:** A text input field.
- Password Again:** A text input field.
- Buttons:** Two buttons at the bottom, 'Show Users' and 'Create User'.

The interface is clean and modern, with a light gray background and white input fields.

User Management UI

 CENG Desktop Website Management App (IZTECH)

► User Management

► Notification Management

► Course Management

User Type

Admin

Email

admin@iyte.edu.tr

Name & Surname

Admin ADMIN

Title

Professor

Password

•••••

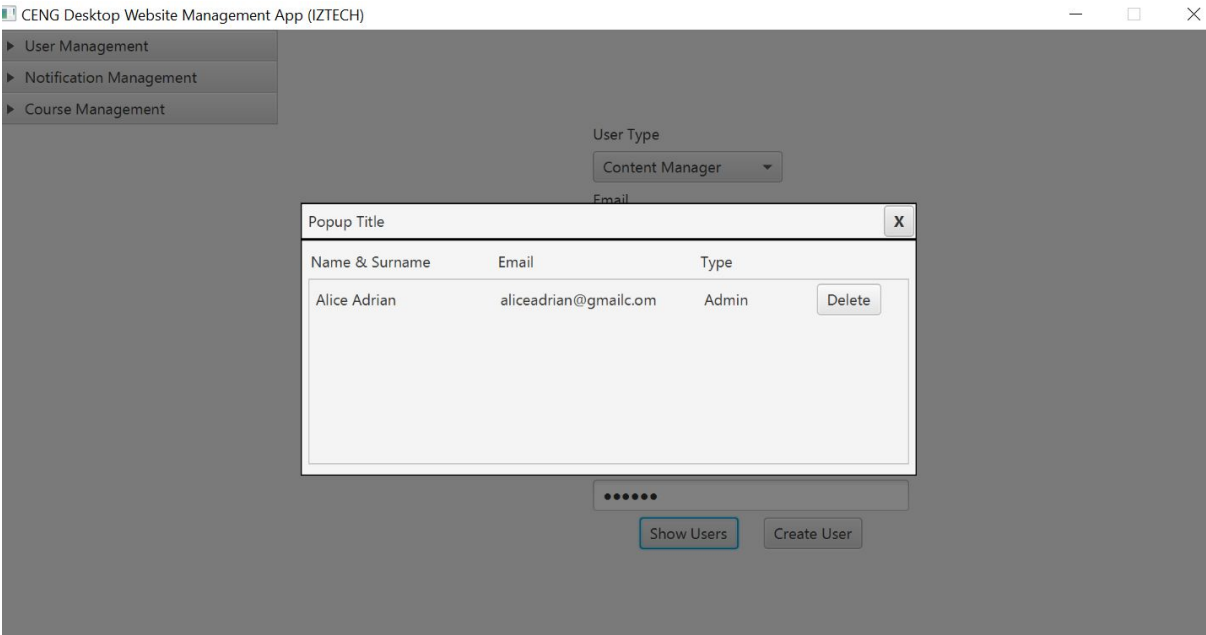
Password Again

•••••

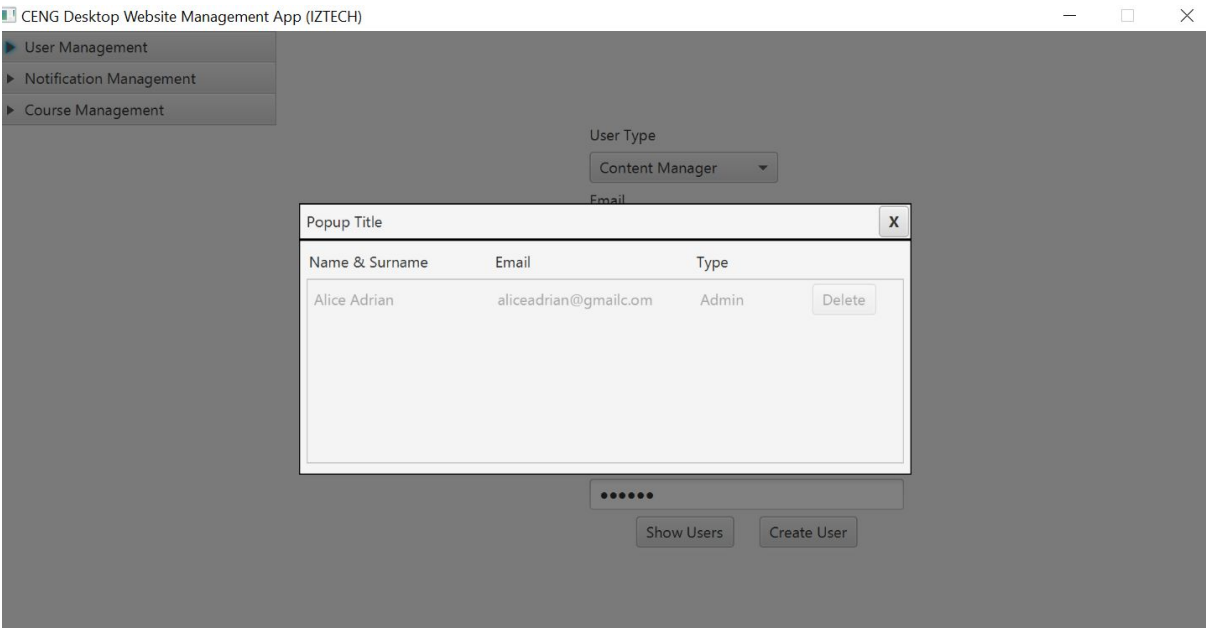
Show Users

Create User

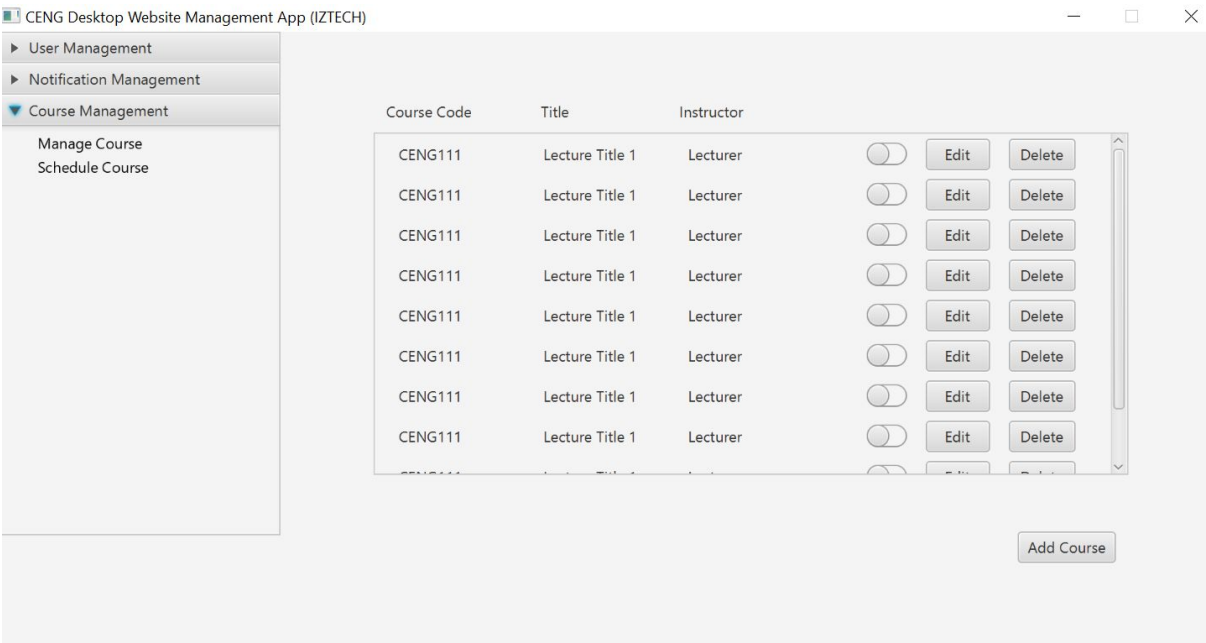
Create User UI



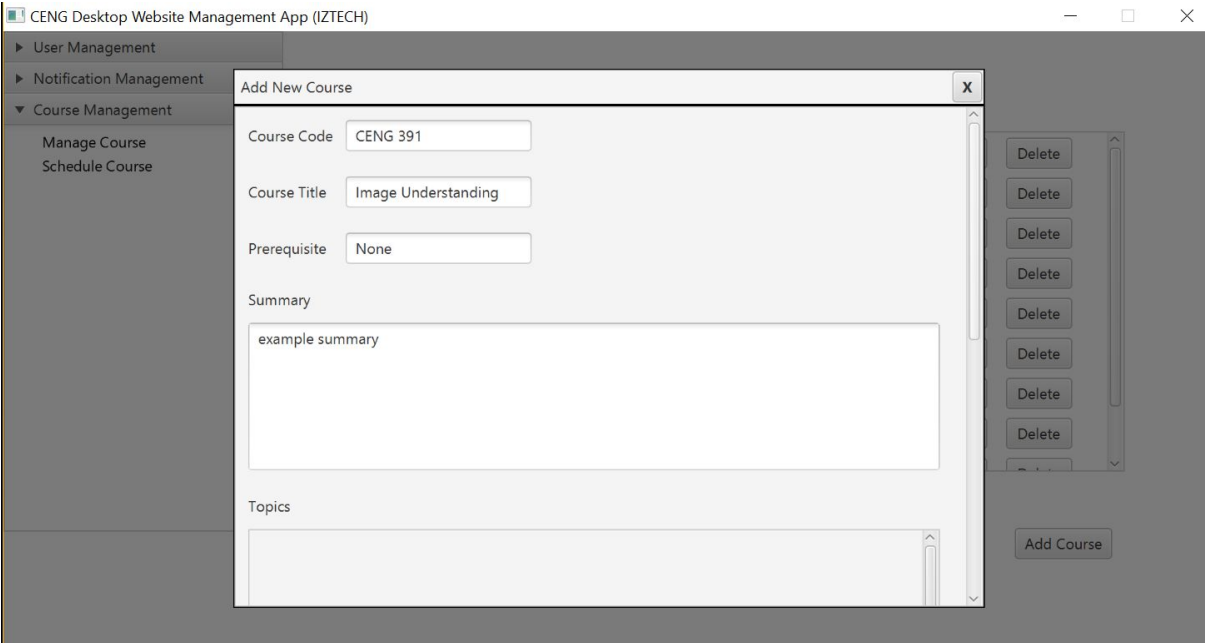
Show User UI



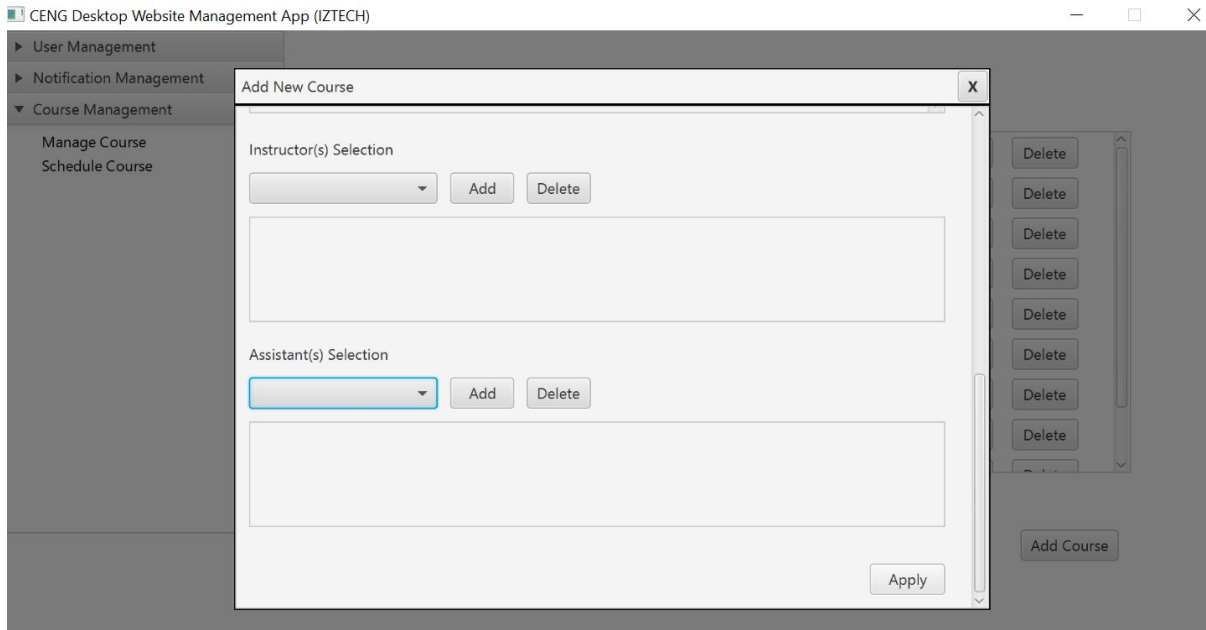
Delete User UI



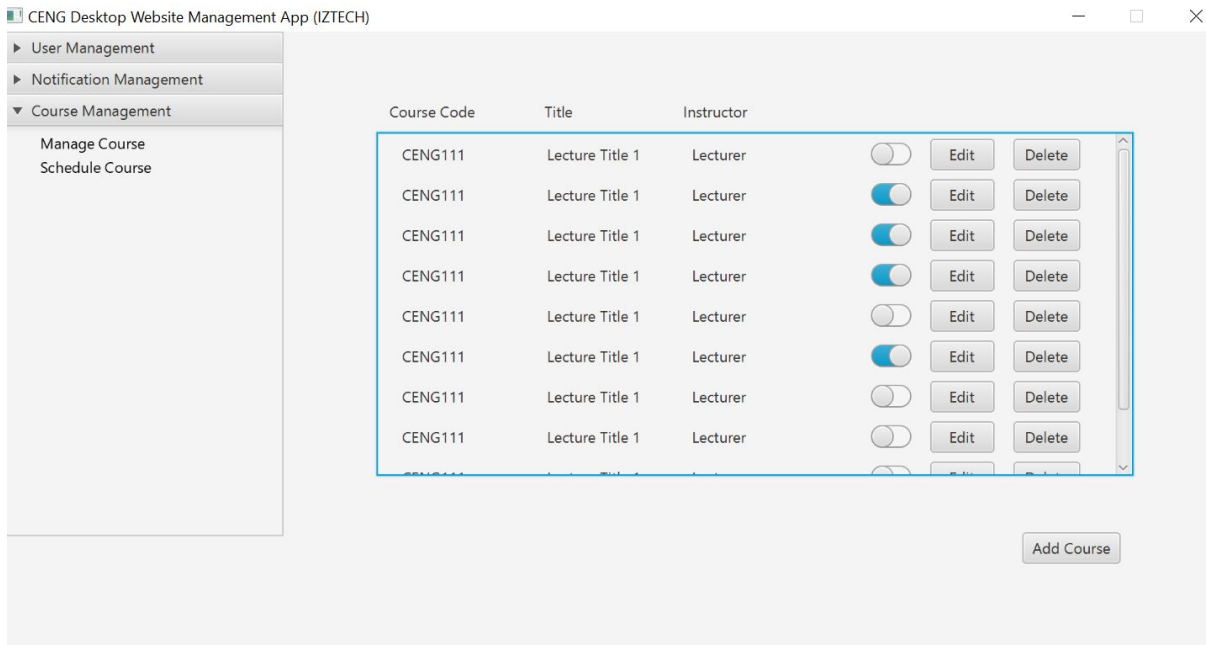
Course Management UI



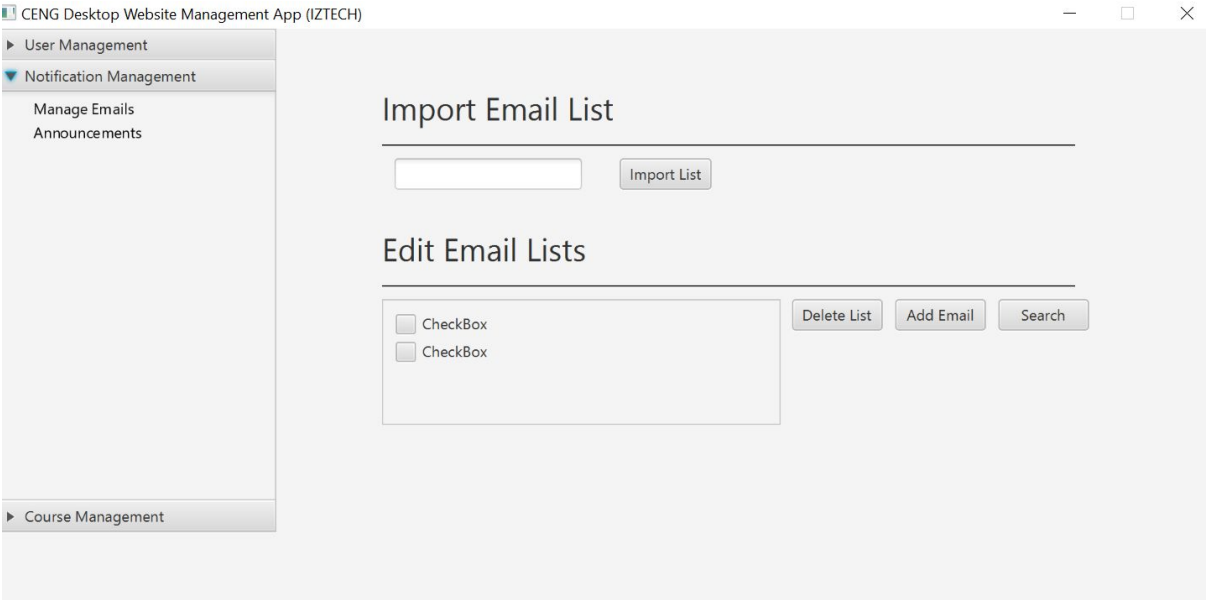
Add Course UI



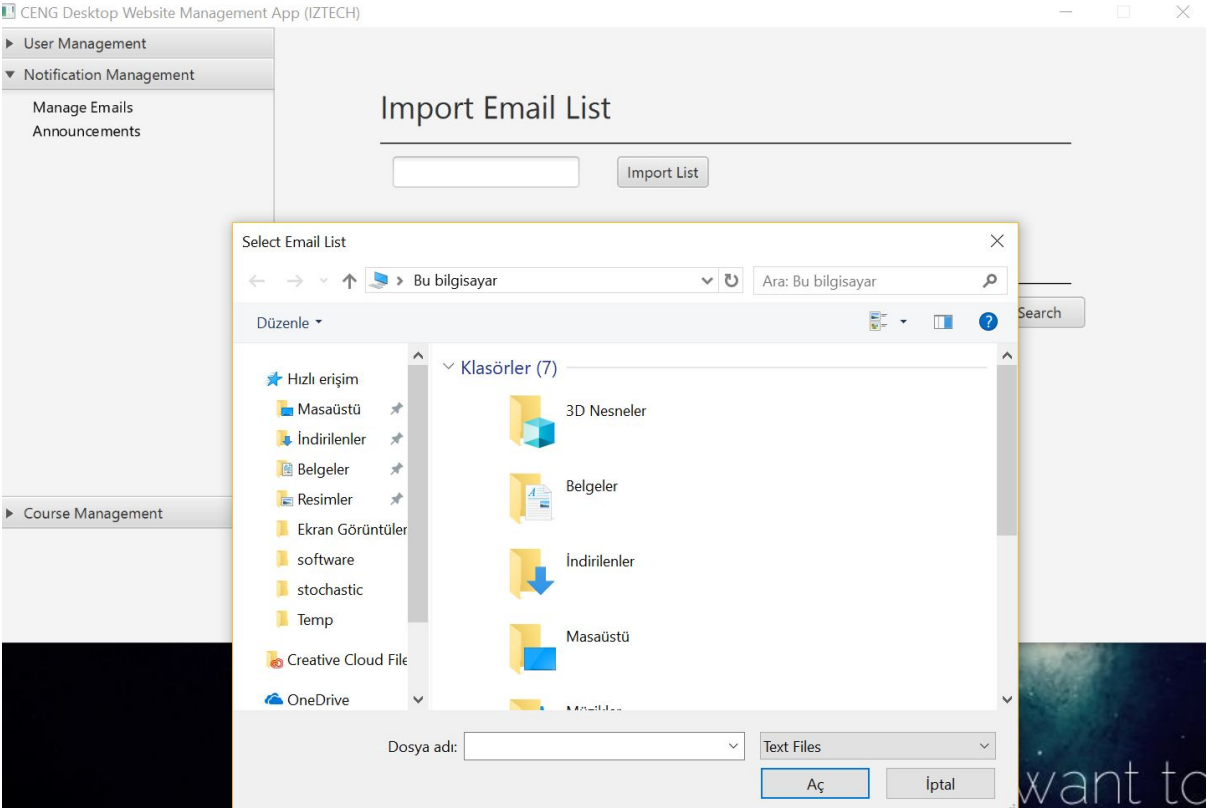
Add Course UI



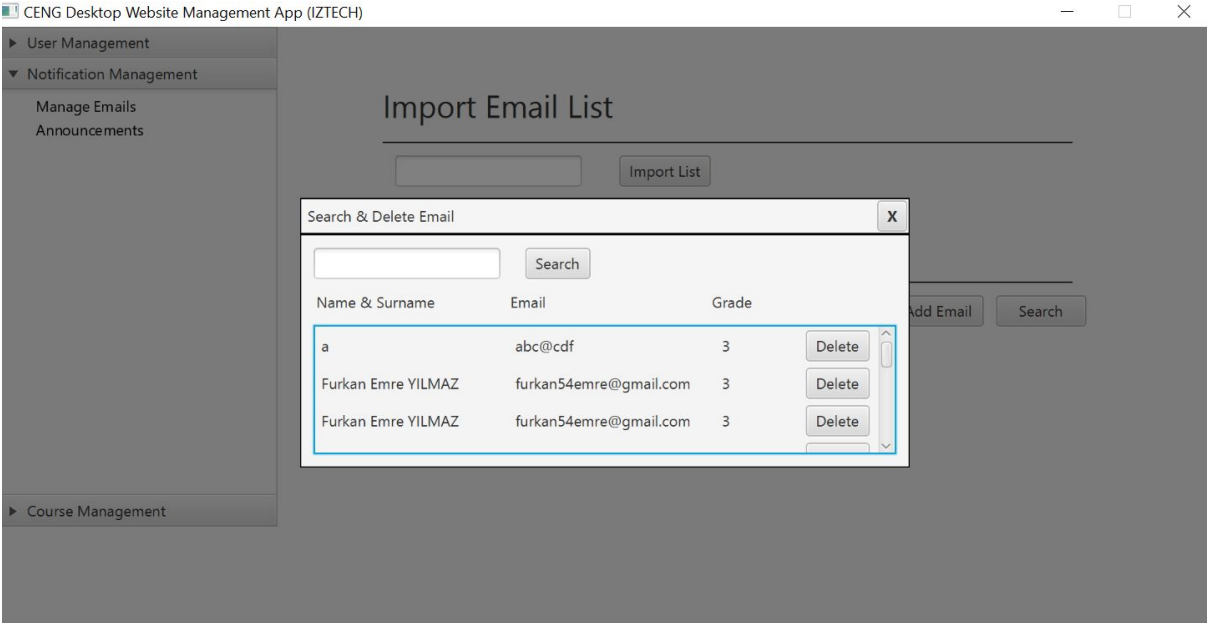
Enable Course UI



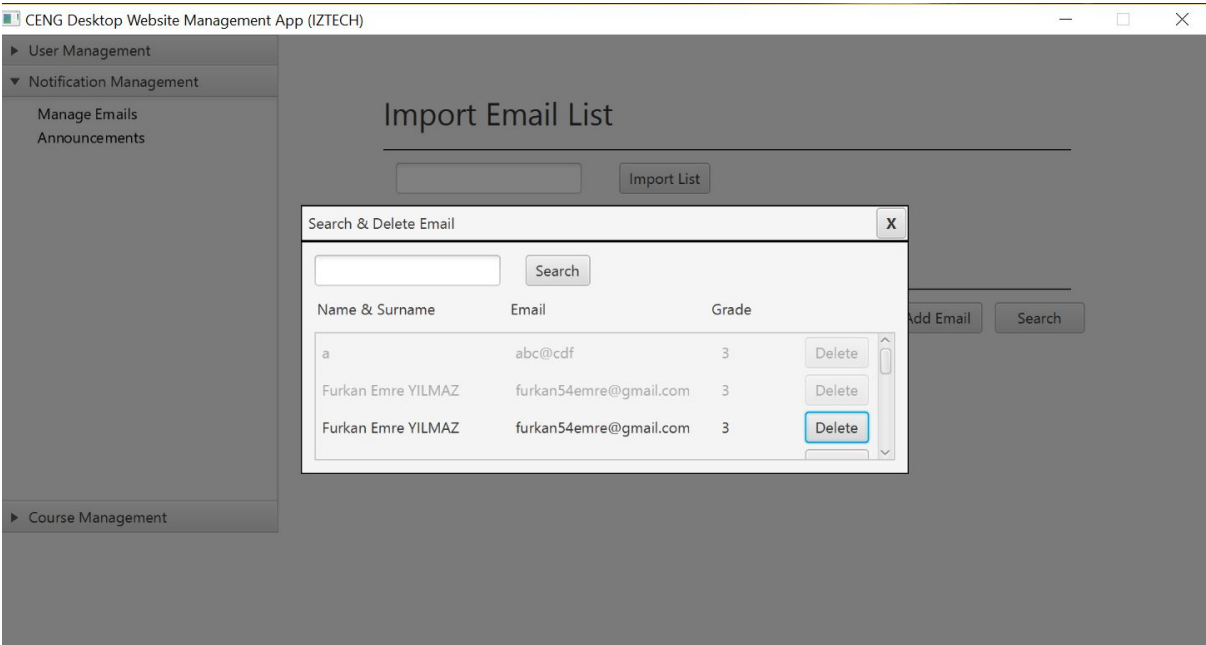
Manage Emails UI



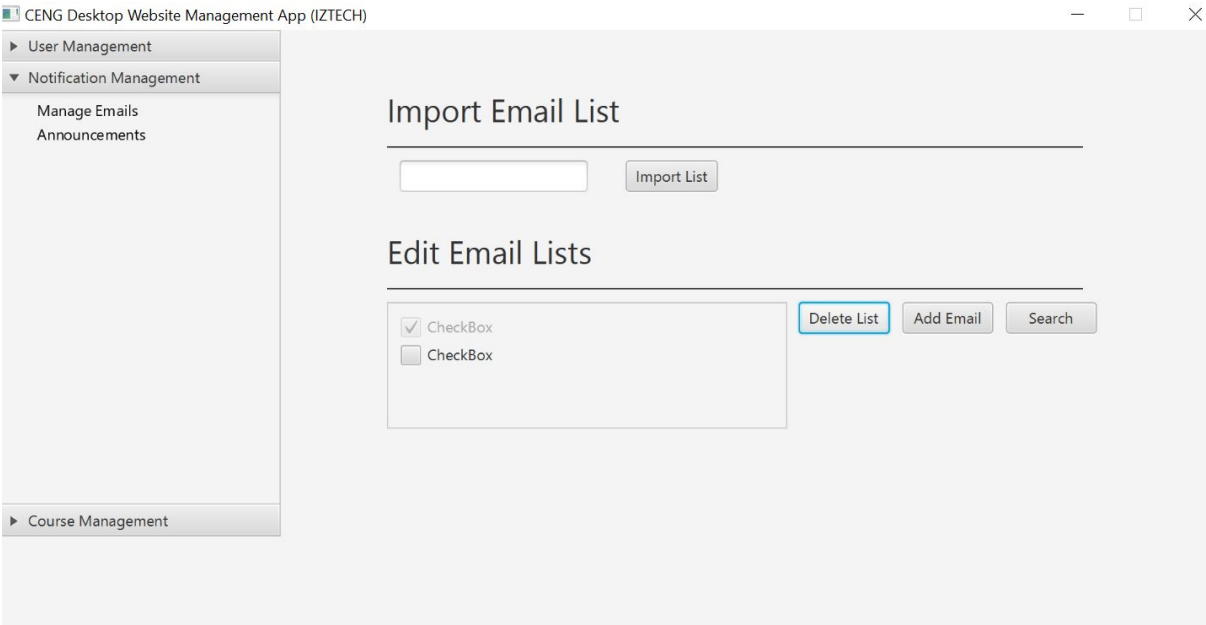
Import Email List UI



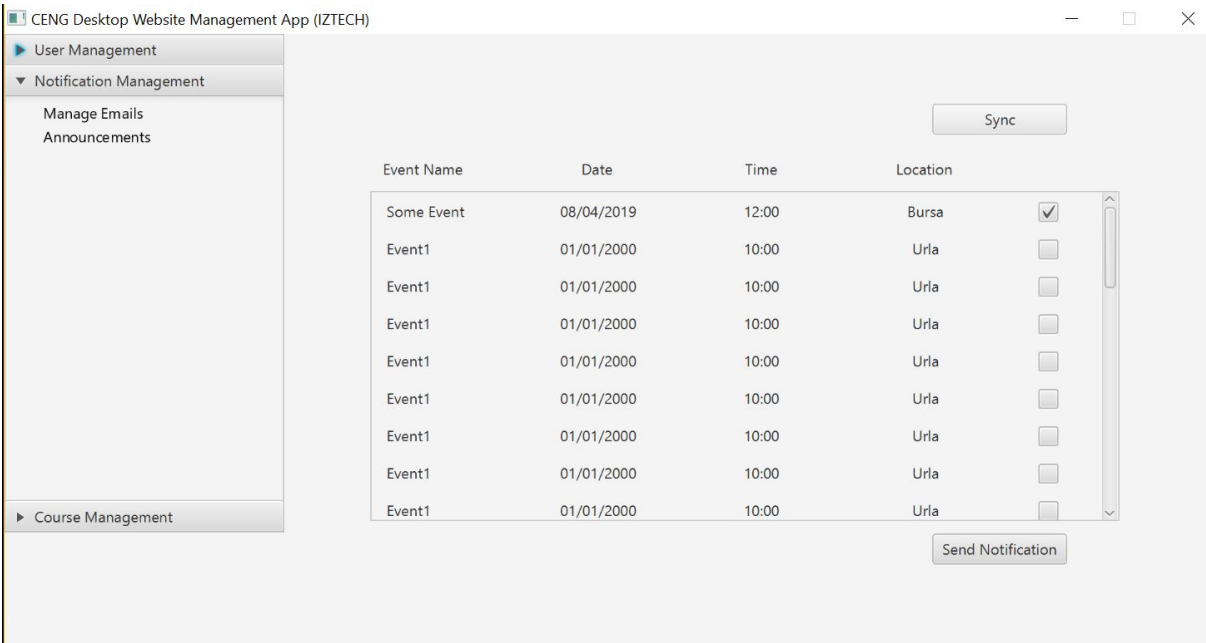
Search Email UI



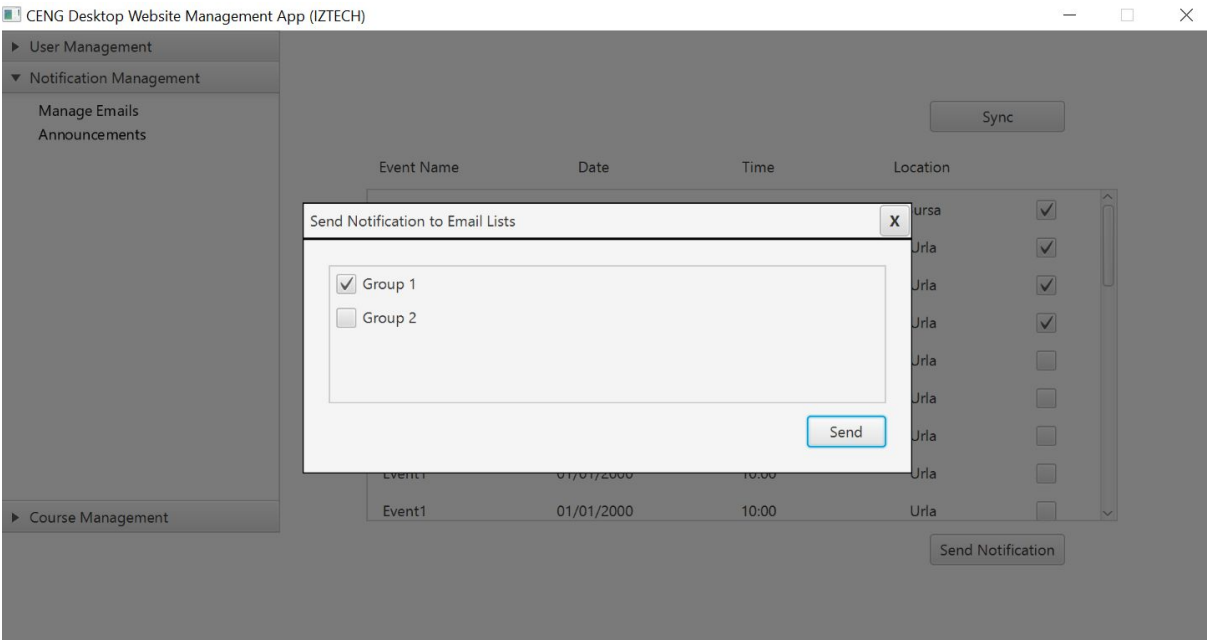
Delete Email UI



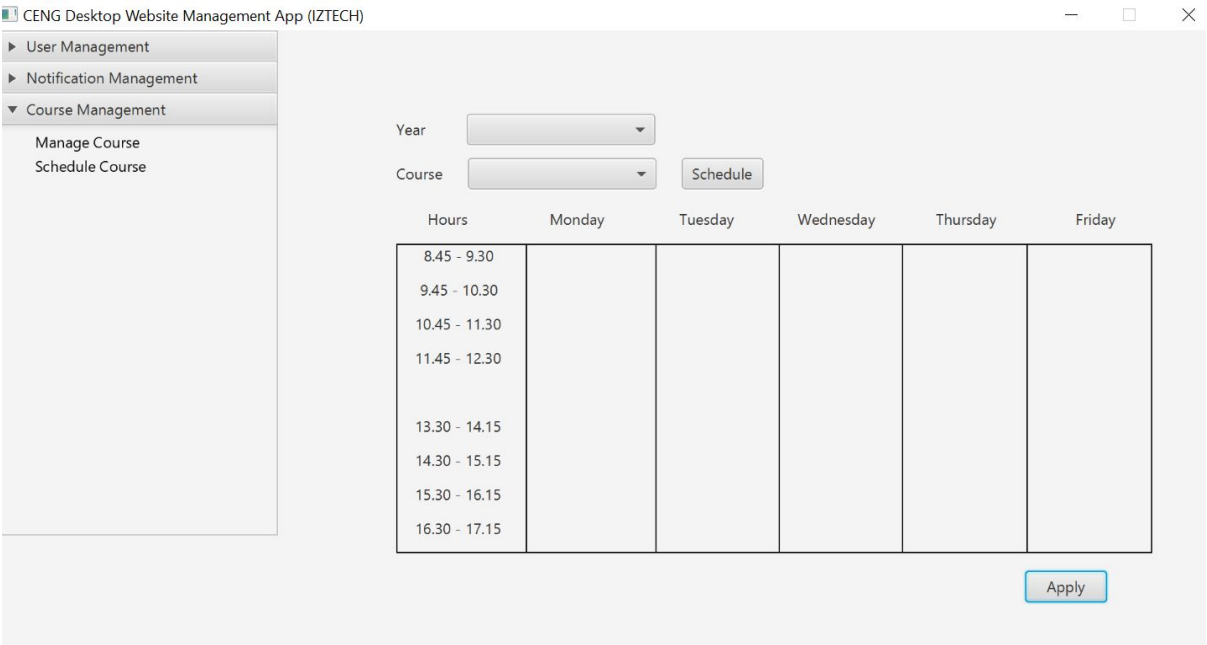
Delete Email List UI



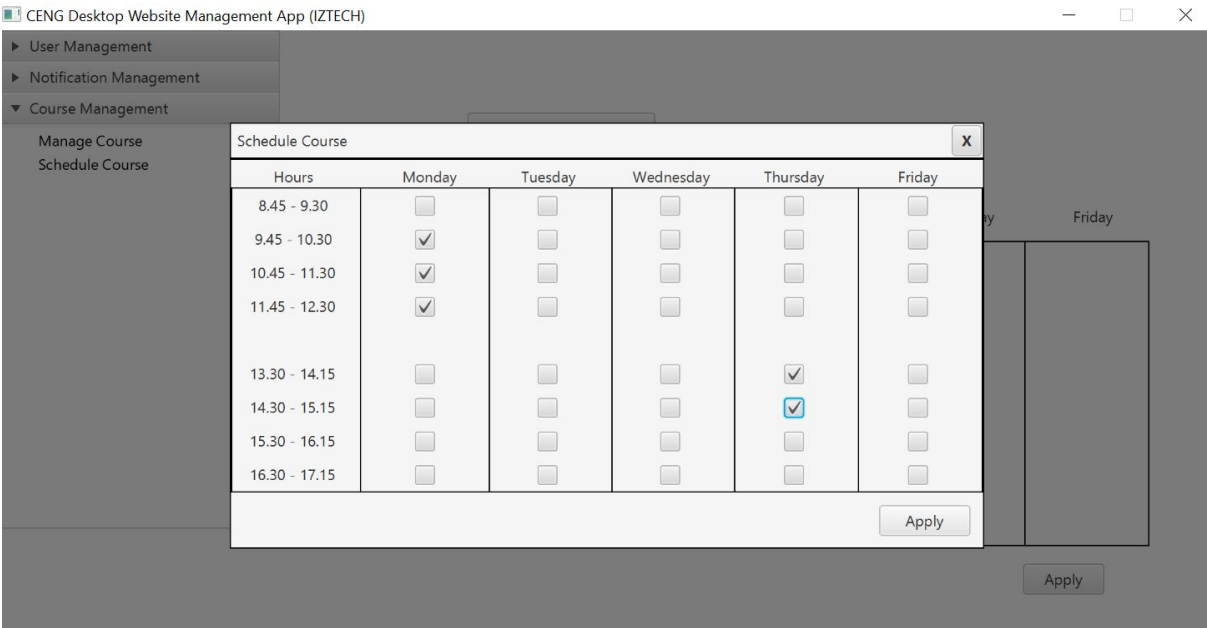
Announcements UI



Send Notification UI



Schedule Courses UI



Schedule Course Pop-up UI