

## **Time Scheduler**

### **Object Oriented Programming with Java Project Report**

Team 9

Java Object Oriented Programming

Third Semester

Frankfurt University of Applied Sciences

12<sup>th</sup> February 2021

Trung Nguyen Quoc (1370166)

Khang Nguyen Phu (1370289)

Khoa Nguyen (1370247)

Jatender Singh Jossan (1346747)

Matheus Mapa de Oliveira (1307964)

Instructor: Prof. Luigi La Blunda

Due date: 11<sup>th</sup> February 2021

*11 February 2022*

Nguyen Quoc Trung

Team 9

Class: Java OOP

Frankfurt University of Applied Sciences

Prof. Luigi La Blunda  
Frankfurt University of Applied Sciences  
Nibelungenplatz 1, 60318 Frankfurt am Main

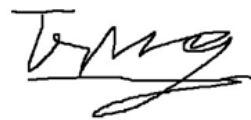
Dear Prof. La Blunda,

This is our Java Object Oriented Programming Report for our class project, Time Scheduler [1], submitted to meet the requirements for Java Object Oriented Programming Subject.

This report describes the context of the project, provides a full description of the project from a software design development, and validates the feasibility of the project.

Thank you very much for taking the time to review this report. Hope you have a great time. If you have any further questions please contact me on 0049 172 568 9107 or email me at [trung.nguyenquoc@stud.fra-uas.de](mailto:trung.nguyenquoc@stud.fra-uas.de).

Sincerely yours,

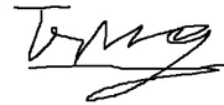
A handwritten signature in black ink, appearing to read 'Trung', with a stylized flourish at the end.

Nguyen Quoc Trung

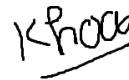
## Disclaimer

We declare that this report is from our own work. We also confirm that all opinions, results, conclusions and recommendations are our own and may not violate the policies of Frankfurt University of Applied Sciences.

Trung Nguyen Quoc  
1370166  
Team Leader



Khoa Nguyen  
1370247



Matheus Mapa de Oliveira  
1307964



Khang Nguyen Phu  
1370289



Jatender Singh Jossan  
1346747



# **Abstract (Khang Nguyen Phu - 1370289)**

## **Time Scheduler**

Team 9

The Time Scheduler System is a computer application built with Java for back-end and JavaFX for the purpose of providing a graphical user interface with account registration and log-in for users to set up for the appointment with each other.

The system also includes an integrated database for data management, enabling an authorized user to shift or delete user accounts and appointments, as well as a reminder function, informing users via e-mail about pending appointments.

This project is part of a computer science curriculum, evaluated and directed by Luigi La Blunda, to gain first-hand experience in Java programming and to implement database and email dispatcher capabilities.

The main problem we have encountered during this project is mapping between database and backend. We also had some trouble when we used JavaFX [2] and its library, CalendarFX [3].

Key words: Java, Computer Application, Integrated Database

# Table of Contents

<b>Disclaimer</b>	<b>3</b>
<b>Abstract (Khang Nguyen Phu - 1370289)</b>	<b>4</b>
<b>Acknowledgements</b>	<b>6</b>
<b>1. Introduction (Trung Nguyen Quoc - 1370166)</b>	<b>1</b>
1.1. Purpose	1
1.2. Audience	1
1.3. Project Scope	1
1.4. Goal	1
<b>2. Context of the Project (Khoa Nguyen - 1370247)</b>	<b>2</b>
2.1. Requirements	2
2.2. Constraints	2
<b>3. Motivation (Matheus Mapa - 1307964)</b>	<b>2</b>
<b>4. Overview of the Event Scheduler System</b>	<b>3</b>
4.1 Architecture Description (Trung Nguyen Quoc - 1370166)	3
4.2 Class Diagram (Trung Nguyen Quoc - 1370166)	4
4.3 Activity Diagram (Khoa Nguyen - 1370247)	5
4.4 Sequence Diagrams (Matheus Mapa, 1307964)	6
4.5 Database Diagram (Jatender Singh Jossan - 1346747)	8
<b>5. Software</b>	<b>8</b>
5.1. User Interface (Khang Nguyen Phu - 1370289)	8
5.2. Database (Jatender Singh Jossan - 1346747)	8
5.3. Login and Register (Khang Nguyen Phu - 1370289)	10
5.4. Add,delete or edit an event (Matheus Mapa, 1307964)	10
5.5. Sending email (Khoa Nguyen - 1370247)	10
5.6. Export schedule (Trung Nguyen Quoc - 1370166)	10
5.7. Reminder function (Khoa Nguyen - 1370247)	10
5.8. Admin function(Khang Nguyen Phu, 1370289)	10
5.9. Security and encryption method (Khoa Nguyen - 1370247)	10
<b>6. Conclusion</b>	<b>11</b>
<b>7. References</b>	<b>11</b>
<b>Appendix A: Glossary of Terms</b>	<b>12</b>
<b>Appendix B, Gantt Chart</b>	<b>12</b>

## Acknowledgements

I am deeply grateful to the faculty members of the Frankfurt University of Applied Sciences, especially Professor Luigi La Blunda, the trainer of this project, for their explanation and guidance on how to write a technical report.

We would also like to thank all my friends and classmates for their assistance during the writing of this report.

During this project, several tasks were divided up as illustrated in the following table:

Task	Involvement <sup>1</sup>				
	TNQ	KNP	KN	MMO	JSJ
Write documentation	x	x	x	x	x
Prepare slides for presentation	x	x	x	x	x
User Interface (Front-end)		x	x	x	
Design overall structure	x	x	x		
Draw diagrams	x	x	x	x	x
Database	x		x	x	x
Login and Register function		x	x		
Create, cancel and shift appointment function				x	
Modeling objects	x		x	x	
Sending email function			x		
Export information	x				
Reminder function			x		
Admin function		x	x	x	
Encryption and security method			x		

---

<sup>1</sup> TNQ = Trung Nguyen Quoc, KNP = Khang Nguyen Phu, KN = Khoa Nguyen, MMO = Matheus Mapa de Oliveira, Jatender Singh Jossan

# **1. Introduction (Trung Nguyen Quoc - 1370166)**

## **1.1. Purpose**

This final report on the Java object-oriented programming project "Time Scheduler" is presented to partially fulfill the requirements of the course "Object-oriented programming in the Java language" at the Frankfurt University of Applied Sciences and contains technical descriptions and architectures. Programs, data flow through programs.

## **1.2. Audience**

The intended audience of this document is our classmates and the course instructor, who will use it as the basis for a determination of a portion of our grade for the class "Java Object Oriented Programming".

## **1.3. Project Scope**

The report contains the details about the structure of the project. It also provides the schedule to complete the project and the human resources management of our group. However the report can not cover the further upgradability of the application.

## **1.4. Goal**

This project mainly focuses on an application to help the user make and manage appointments with each other.

## **2. Context of the Project (Khoa Nguyen - 1370247)**

### **2.1. Requirements**

- User can register and confirm account by a code sent to their email address
- User can login
- User can edit their profile
- User can make a new event
- User can edit an existing event
- User can delete an existing event
- User can choose to be reminded of the upcoming event or not via their email address
- User can export their schedule

### **2.2. Constraints**

- The project must be done by ourselves and cannot be copied from an existing design or major structure on the Internet.
- Proper English must be used in the program.
- Our coding must be in Java and use the Java library so that the software code needs to be limited to some advances.

## **3. Motivation (Matheus Mapa - 1307964)**

The motivation behind this project is to develop an user friendly application, in which the users will be able to create appointments that can be shared with several participants and also be able to edit them, attach documents, delete appointments and have an overview of their appointments through the User Interface.

Users using this application will have some assistance on managing their day-to-day schedules and appointments, as well as who is invited to them, and also share files that other users may download real-time that may bring additional information to the appointment.



## 4. Overview of the Event Scheduler System

### 4.1 Architecture Description (Trung Nguyen Quoc - 1370166)

The Time Scheduler is a PC application, providing a graphical interface with JavaFX. Database is accessed via the External Connections package. The structure of the application of the project follows the Model-View-Controller pattern.

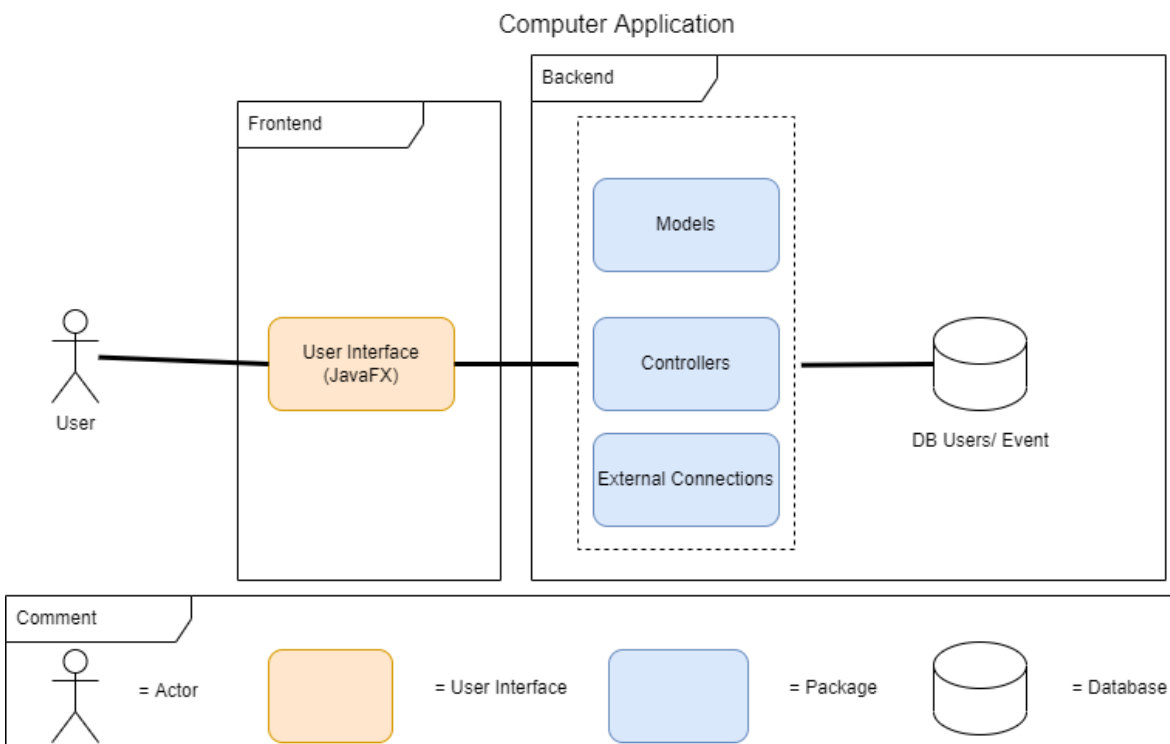


Figure 4.1: Architecture Diagram

## 4.2 Class Diagram (Trung Nguyen Quoc - 1370166)

The class structure follows the Model-View-Controller pattern. The user can interact with the event through the controllers in the System between object User and Event.

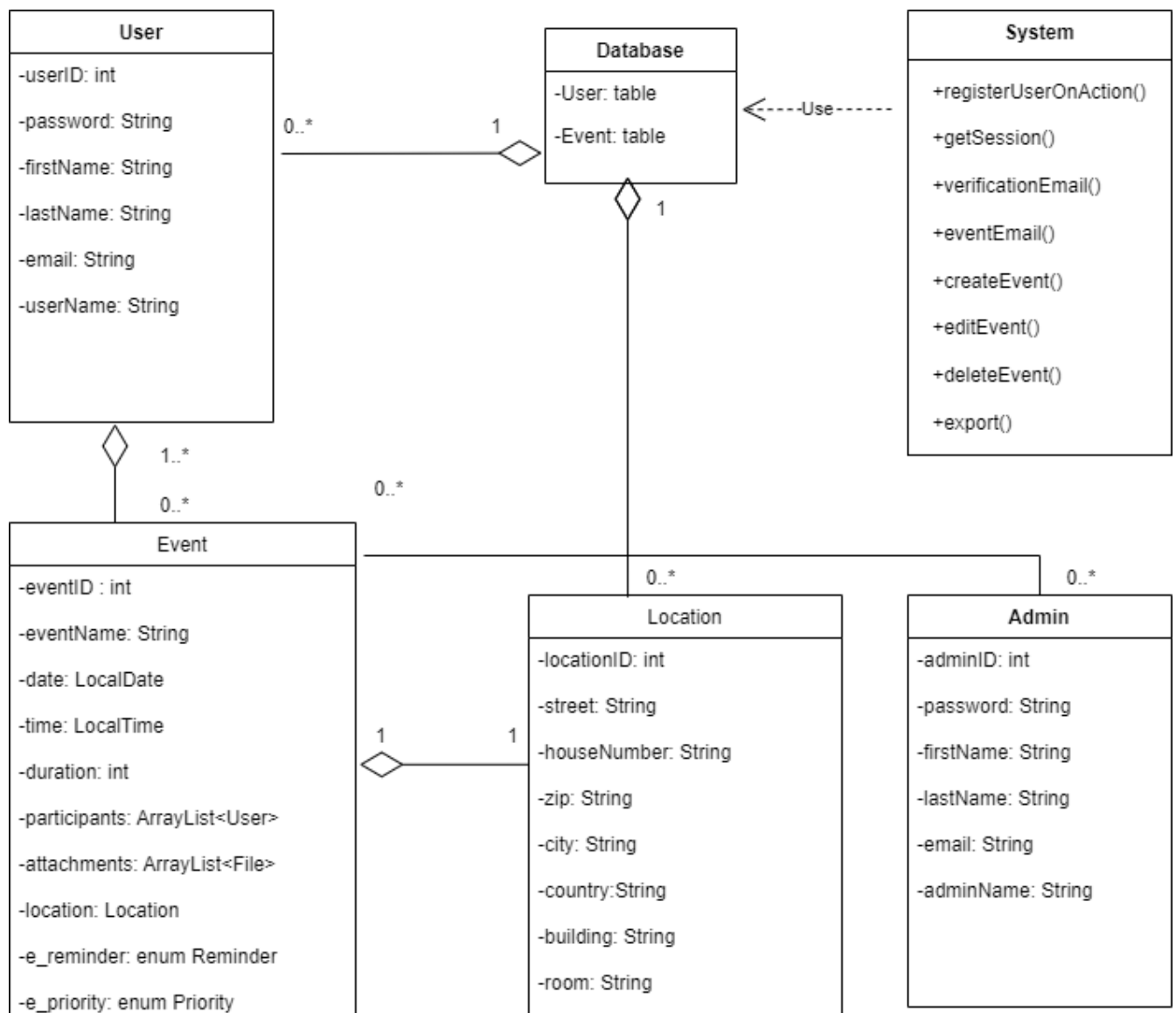


Figure 4.2: Class Diagram

### 4.3 Activity Diagram (Khoa Nguyen - 1370247)

This activity class shows what a user interacts with in the program.

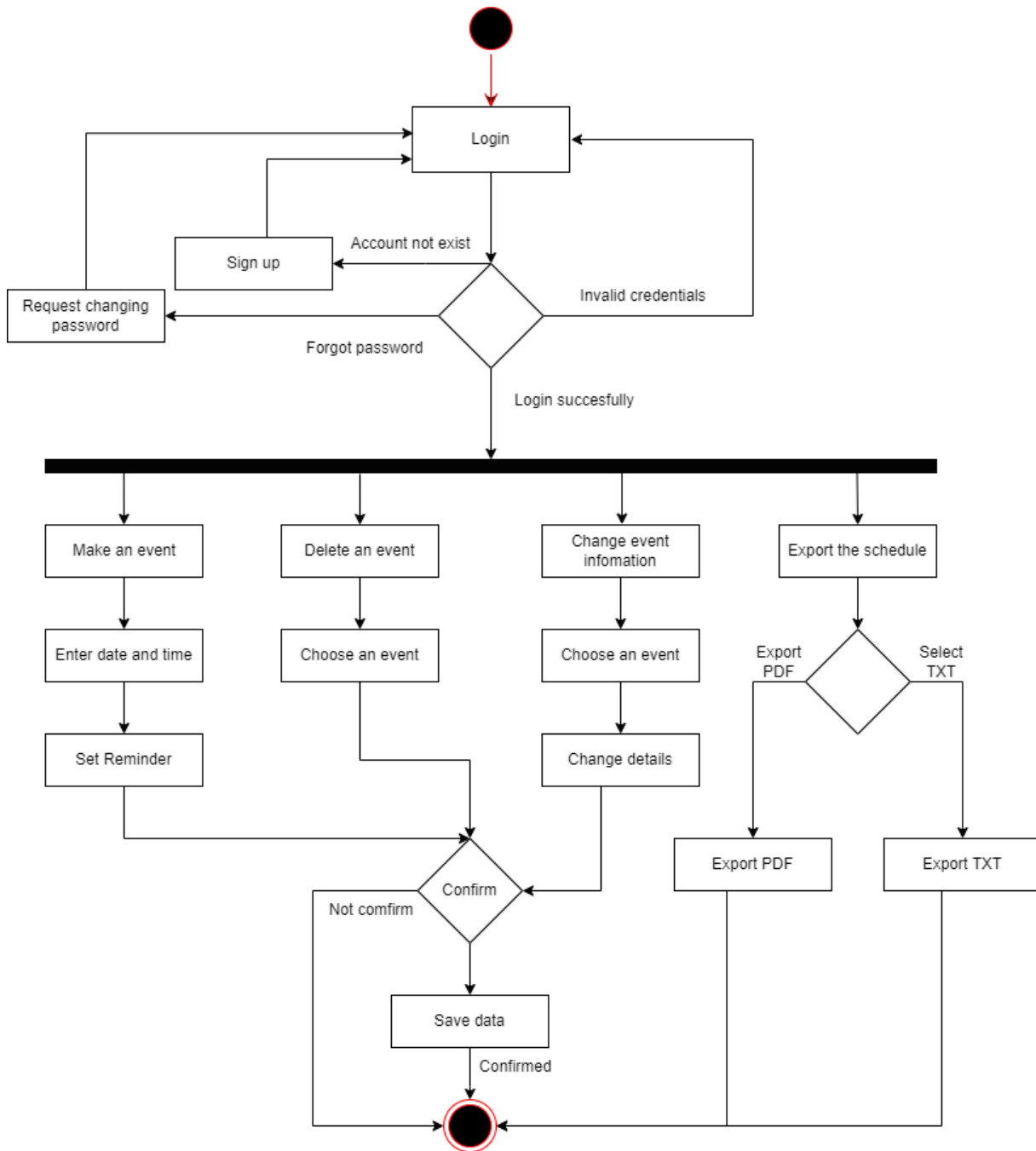


Figure 4.3. Activity Diagram

## 4.4 Sequence Diagrams (Matheus Mapa, 1307964)

### 4.4.1 Create Event Diagram

In the below diagram we see the flow to create an event. First of all the user clicks on the “Create” button after inserting the event data, that triggers the CreateEvent function that passes the information to the EventController Class which in its turn calls the insertNewEvent function to store the information on the persistent database. With this the event is stored in the standalone database and will appear in the UI after the next time the thread refreshes the event view.

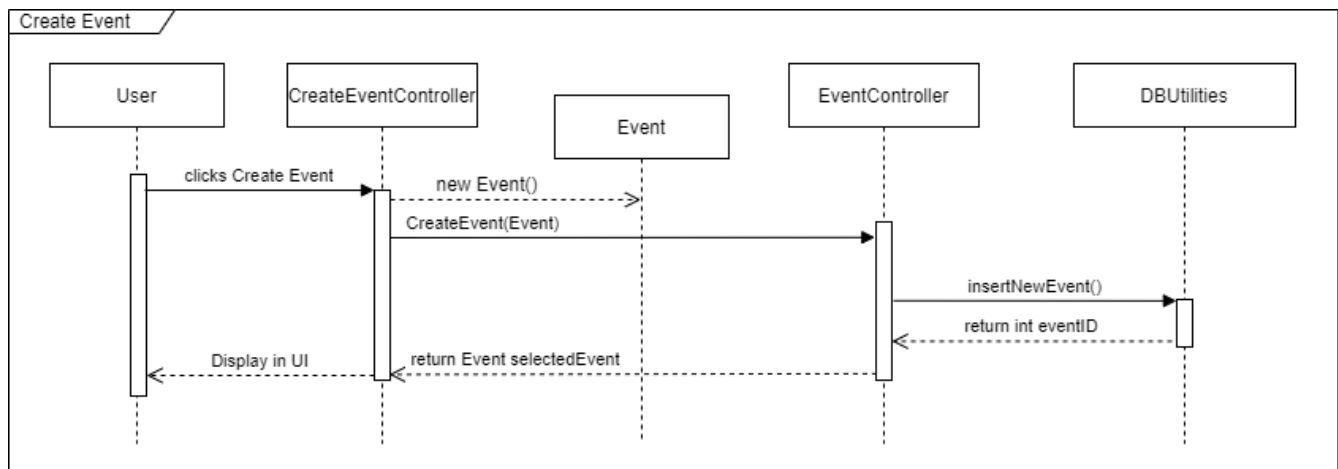


Figure 4.4.1: Create Event Sequence Diagram

### 4.4.2 Edit Event Diagram

In this diagram we see the sequence of editing an already existing event. First of all the User selects the event from a dropdown list of all his events. At this stage the event data will fill the fields in the form, which he can change as he wishes and click on the “Save” button. This will trigger the EditEvent function in the event controller, which in its turn will edit the Event object using the setter methods, and afterwards will call the functions in DBUtilities in order to update the database with the appropriate changes. The selectedEvent with its changes will be returned to the EditDeleteEventController. The changes will be displayed in the user interface as soon as the thread refreshes the event view.

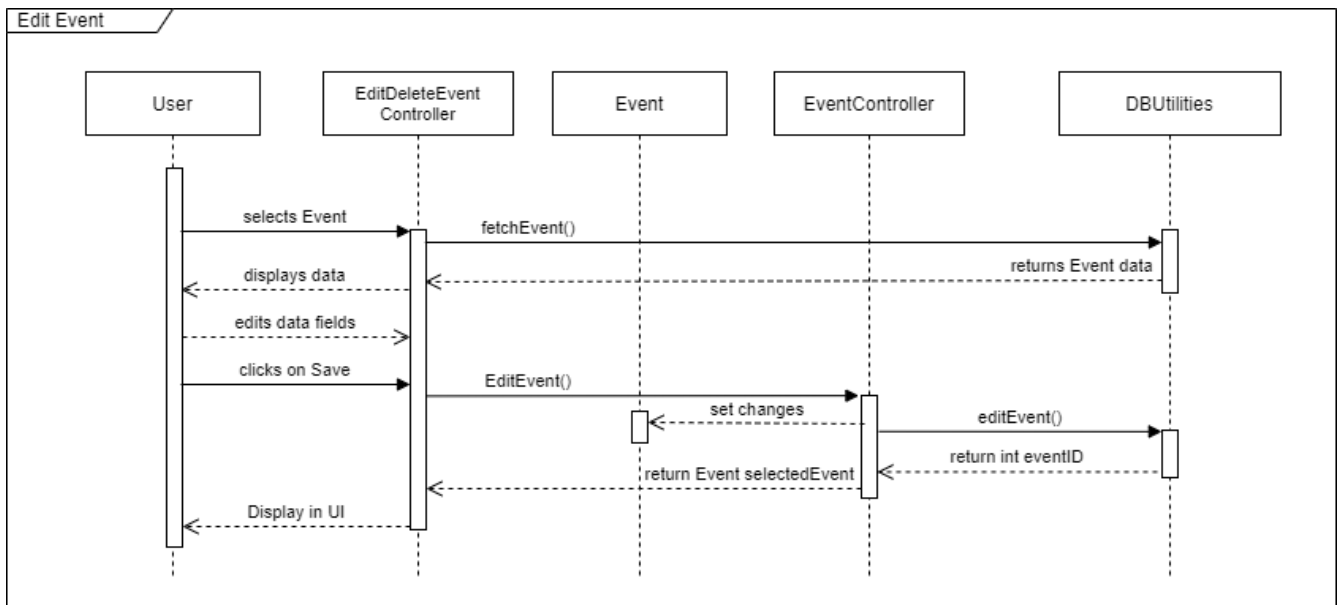


Figure 4.4.2: Edit Event Sequence Diagram

### 4.4.3 Delete Event Diagram

In the diagram below we see the flow to delete an existing event. First of all, similarly to the edit event flow, the user selects the event from a dropdown list of all his events. The event data will fill the fields in the form, so he can verify that this is the event that he wishes to delete. After clicking on the “Delete” button, which will trigger the DeleteEvent function in the EventController, which in its turn calls deleteUser\_EventBridge, deleteAttachment, and deleteEvent functions, deleting the necessary entries from the database. After this is done, the EventController will return an exit code 0 to indicate the deletions were successful. The event will disappear from the user interface as soon as the thread refreshes the event view.

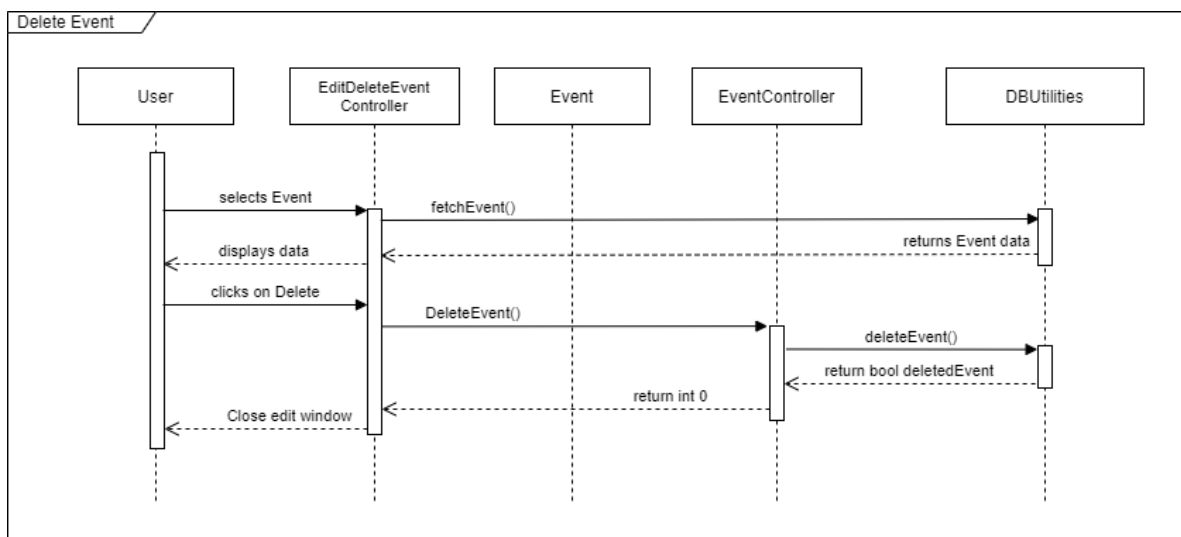


Figure 4.4.3: Delete Event Sequence Diagram.

## 4.5 Database Diagram (Jatender Singh Jossan - 1346747)

In this entity-relation-diagram you can see the attributes and entities used in the database. We used five entities representing the objects: user, event, location, attachment and participant.

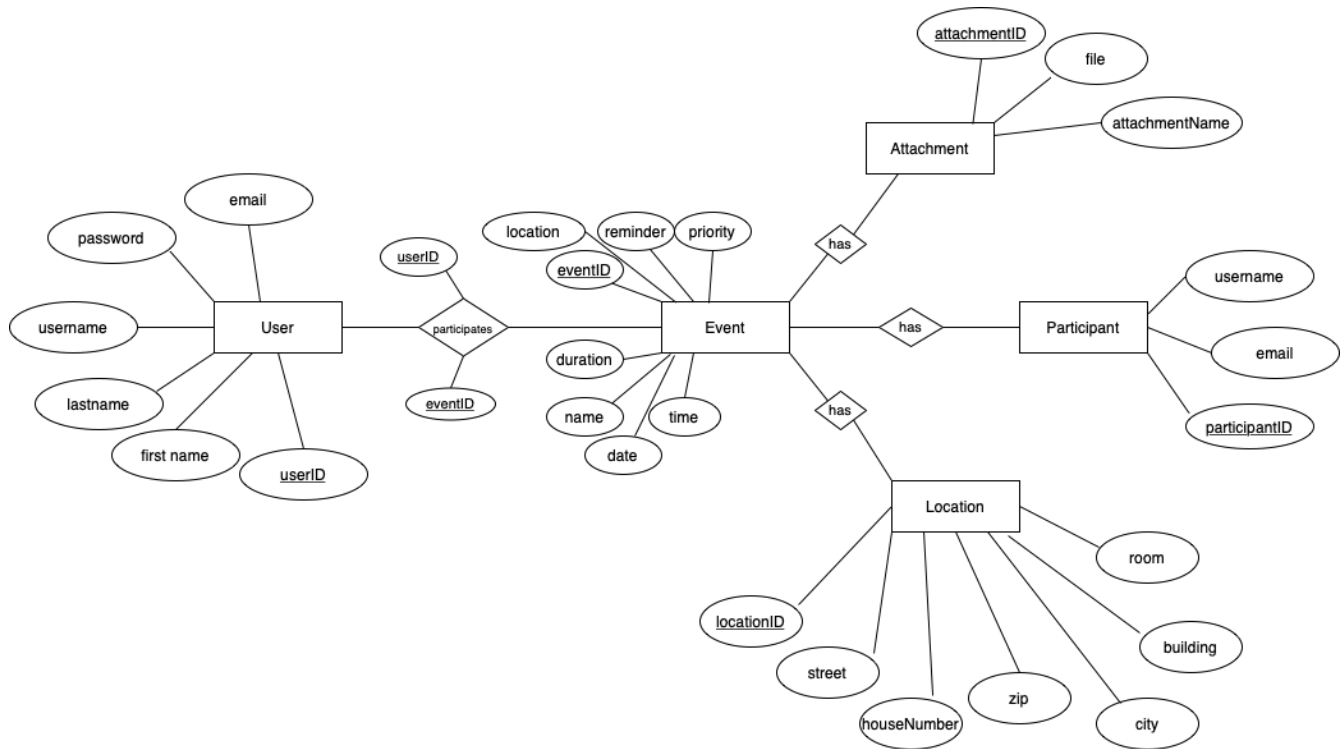


Figure 4.5 Database Diagram

## 5. Software

### 5.1. User Interface (Khang Nguyen Phu - 1370289)

We used JavaFX[2] and the CalendarFX[3] library to construct the UI.

### 5.2. Database (Jatender Singh Jossan - 1346747)

We used MySQL as DBMS and JDBC Connector to establish the connection.

#### Relational model

Please note:

- Every attribute which is underlined in a table, is/are the primary key(s) of this table.
- Every attribute which has an exclamation mark (!) in front of it, is/are the foreign key(s) of this table

### **User - Event: n:m relation**

User (UserID, firstname, last name, username, password, email)

Event (EventID, eventName, eventDate, eventTime, duration, location, participants, attachments, reminder, priority)

participates(!UserID, !EventID)

Since the entities "user" (participant) and "event" have a n:m relation (a participant can participate in multiple (m) events and an event can have multiple (n) participants), the relation by itself gets a entity which safes the primary keys of both tables to have reference which user is going to participate in which event. The primary keys of both entities - user and event - are the foreign keys of the new entity "participates". They are the primary keys of this entity too.

### **Event - location: 1:1 relation**

Event (EventID, eventName, eventDate, eventTime, duration, location, participants, attachments, reminder, priority)

Location (locationID, street, streetNumber, zip, city, country, building, room, !eventID)

This 1:1 relation (one event takes place at one location) can be treated as a 1:n relation (one event takes place at several (n) places) too, which we decided to do here. The entity "event" got the primary key of the "Location" entity as a foreign key. This is the reference point for knowing which event takes place at which location.

### **Event - Participants: 1:n relation**

Event (EventID, eventName, eventDate, eventTime, duration, location, reminder, priority)

Participant (participantID, username, email, !eventID)

Since an event can have multiple participants, we decided for a 1:n relation (one event (1) can have multiple (n) participants. The primary key of the event entity is the foreign key of the participants entity. This is mandatory for knowing which event has which participants.

### **Event - Attachments: 1:n relation**

Event (EventID, eventName, eventDate, eventTime, duration, location, participants, attachments, reminder, priority)

Attachment (attachmentID, file, !eventID)

Furthermore, besides the participants, an event can have multiple attachments which is why we decided for a 1:n relation (one event (1) can have multiple (n) attachments). This is also mandatory for knowing which event has which attachments.

### **5.3. Login and Register (Khang Nguyen Phu - 1370289)**

The user initially comes to our system with the login page. The user can login into our system to make any changes about their event or add a new event if they have an account. If not, they can create a new account. All the information of the user will be saved in the database, and the password will be hashed.

### **5.4. Add,delete or edit an event (Matheus Mapa, 1307964)**

After logging-in into our system and using the add new event function, the user will provide some basic information related to the event (title, time, attachment, reminder). After making a new event, the user is able to delete or change information about the event.

### **5.5. Sending email (Khoa Nguyen - 1370247)**

We use the Javamail API to send email. The system will send a confirmation email to the user whenever the user creates a new account and send a reminder email about the upcoming event (if the user wants the system to remind them about that event).

### **5.6. Export schedule (Trung Nguyen Quoc - 1370166)**

The user has 2 options to export the schedule. First one is exporting all events into a txt file. The second one is exporting the calendar and its event in the specific time interval into a pdf file.

### **5.7. Reminder function (Khoa Nguyen - 1370247)**

The user has four options for reminder: None, 10 minutes, 1 hour, 3 hours. Based on the appointment date and the chosen option, we will calculate the reminder date and then send an email to the user on that day.

### **5.8. Admin function(Khang Nguyen Phu, 1370289)**

The admin can log in through the log in section and is able to view, edit and delete all the users profile.

### **5.9. Security and encryption method (Khoa Nguyen - 1370247)**

We are using SHA512 to hash the password.



## 6. Conclusion

Developing this program, our aim was to have a reliable program which can be used for making appointments. Moreover, we tried to combine multiple tasks in one program, e.g. not only having the opportunity to make an appointment for yourself but also to have the opportunity to make an appointment with your colleagues and sharing this event with all of them only with one click.

All in all, it was a good experience to work in a team. If one could solve a problem we had a team of five students helping each other in every possible situation. Due to our sprint beside the courses, we had the opportunity to summarize what we have done so far and what we need to focus on until the next sprint. Looking back on our work, we have gathered some good knowledge about how to work with the IntelliJ IDE, Java and how to stay up to date with github which indeed will help us in our future working.

## 7. References

[1]Blunda L. L. *Time Scheduler*. 2021.

[2]“JavaFX Documentation Project.” *GitHub Pages*, 7 December 2021, <https://fxdocs.github.io/docs/html5/>.

Accessed 10 January 2022.

[3]Lemmermann, Dirk. “CalendarFX 8 Developer Manual.” *DLSC*, 1 February 2017,

<https://dlsc.com/wp-content/html/calendarfx/manual.html>. Accessed 25 January 2022.

## Appendix A: Glossary of Terms

- DBMS: Database Management System
- JDBC: Java Database Connectivity
- OOP: Object Oriented Programming
- SHA512: Secure Hash Algorithm 512
- API: Application Programming Interface

## Appendix B, Gantt Chart

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Analyze and draw diagrams for project							
Create login, register and sending email function							
Create appointment manipulation function							
Create all database function							
Create UI for login and register							
Create UI as a calendar for appointment manipulation							
Create export txt function							
Write final documentation							
Review the project and prepare for presentation							

Figure 8.1: Gantt chart: Red means the weeks in which the activities were performed.