### **Entities**

### Message content: String sender: String time\_sent: LocalDateTime

- Message(content: String, sender: String)
- + getContent(): String
- + getSender(): String
- + getTime\_sent(): LocalDateTime
- + getTitle(): String getSpeaker(): String + toString(): String
  - + getLocation(): String

- datetime: LocalDateTime

usernames: ArrayList<String>

- title: String

- location: String

- speaker: String

- getDatetime(): LocalDateTime
- + getAttendees(): ArrayList<String> setTitle(title: String): void
- + setLocation(location: String): void
- + setDatetime(datetime: LocalDateTime): void
- + setAttendees(usernames: ArrayList<String>): void
- + addAttendees(username: String): void + addAttendees(usernames: ArrayList<String>): void
- + removeAttendees(usernames: ArrayList<String>): void
- + getEventInfo(): Object[]
- +setSpeaker(speaker: String)

### # speaker: String

- + Talk(title: String, location: String, datetime: LocalDateTime)

# Attendee

- # username: String
- # password: String
- # friendList: List<String> # messages: List<Message>
- # loggedIn: boolean
- # role: String
- + Attendee(username: String, password: String)
- + getUsername(): String
- + getPassword(): String
- + getLoginStatus(): boolean + getRole(): String
- + setLoggedIn(value: boolean): void
- + getNumOfFriends(): int

# + getFriendList(): List<String>

# Organizer

## role: String

Event

+ Event(title: String, speaker: String, location: String, datetime: LocalDateTime)

- + getSpeaker(): String
- setSpeaker(String speaker): String

- + Organizer(username: String, password: String)
- + createSpeakerAccount(username: String, password: String): Speaker - assignSpeaker(speakerUsername: String, Talk talk): boolean

- talks: List<String>

-role: String

+Speaker(username: String, password: String) +addTalk(talkId: String)

Speaker

### +aetTalks

EventManager

+ EventManager(title: String, speaker: String, location: String, datetime: LocalDate1

# **Use Cases**

### LoginUserManager

- credentialsMap: HashMap<String, Attendee>
- + LoginUserManager()
- regisetrUser(username: String, password: String, role: String): boolean
- loginUser(username: String, password: String): boolean
- + logoutUser(username: String): void - userRole(username: String): String
- + getAttendee(username: String): Attendee
- getUsername(user: Attendee): String
- + getCredentialsMap(): HashMap<String, Attendee>
- setCredentialsMap(credentialsMap: HashMap<String, Attendee>): void

### Schedule

- startHour: int
- +Schedule(startHour: int, endHour: int)
- +removeEvent(event: Event)
- +editHours(startHour: int, endHour: int)

# usernameToEvents: HashMap<String, List<String>>

- + signUpEvent(username: String, eventTitle: String): boolean
- cancelEventEnrolment(username: String, eventTitle: String): boolean

- BuildingManager
- # building: HashMap<String, Schedule> # buildingName: String
- BuildingManager(buildingName: String)
- +addRoom(name: String, startHour: int, endHour: int): boolean

schedule: Schedule

+ addToSched(): boolean

removeEvent(): boolean

+ addSpeaker(speaker: String)

+ EventManager(event: Event, sched: Schedule)

event: Event

- +addRoom(name: String, sched: Schedule): boolean
- +removeRoom(name: String): boolean +getSchedule(name: String): Schedule
- +getEvent(event: String): Event
- +iterator(): Iterator<Schedule>
- +getScheduleWithEvent(event: String): Schedule
- +toString(): String

keys: List<LocalDateTime

# EventIterato

- scheduleMap: HashMap<LocalDateTime, Event> -endHour: int
- +addEvent(event: Event)
- +getEvent(event: String)
- +eventsAttending(username: String)
- +toString()

### MessageCreator **EventStatus** # receiver: String

-current: int

+hasNext()

+next()

- # message: Message + MessageCreator(toBeMessage: String, receiver: String, senderUserid: String)
  - + getTimeSent(): LocalDateTime
  - + getMessage() : Message

# Controllers

### **AttendeeMenuControlle**

- + username: String
- + role: String
- + menu: UserMenu + building: BuildingManager
- + eventStatusChanger: EventStatusChanger
- + manager: LoginUserManager
- + AttendeeMenuController(username: String, role: String, building: BuildingManager, userManager: LoginUserManager): void + homepage(): void
- + signUpEvent(): void
- + cancelEntrolEvent: void
- + sendMessage(): void
- + addRoom(): boolean + addSpeaker(): void
- + scheduleSpeaker(): void
- + removeEvent(): boolean + messageAttendees(): boolean

+ createEvent(): boolean

- + manageFriendList(): void
- + menuSelection(): void

+manager: EventManager

+ LoginSystem(): void

+ userLoginInfo: UserLoginInfo

- + register(username: String, password: String, role: String)
- + login(username: String, password: String) + logout(username: String): void
- + checkPrivileges(username: String): String

# ReturningUserController

LoginSystem

- + menu: StartingMenu
- + ReturningUserController(): void + usernamePromprt(): String
- + passwordPrompt(): String + logReg(username: String, password: String, role: String): boolean

- +EventController(title: String, speaker: String, location: String, dateTime: LocalDateTime, sched: Schedule)
- +createEvent(): boolean +removeEvent(): boolean

**EventControlle** 

# sender: String

# receiver: String

# messageString: String

+ MessageController()

+ sendMessage()

+ deleteMessage()

- saveMessages()

# messageSystem: Message Creator

# outbox: HashMap<String, ArrayList<Message>>

+ getAllMessages(): ArrayList<String>
+ getMessageForMe(me: String): ArrayList<String>

- getMessagesFromSender(sender: String): ArrayList<String>

LoginMenu

+ usernamePrompt()

+ passwordPrompt()

MessageController(sending: String, receiving String, inputMessage String)

+ allMessageHelper(HashMap<String, ArrayList<Message>> hm): ArrayList<String>

- getMessagesToReceiverFromSender(sender: String, receiver: String): ArrayList<String>

# inbox: HashMap<String, ArrayList<Message>>

# allMessages: MessageGateway

### **EventStatusChanger**

- + evenStatusGateway: EventStatusGateway
- + signUpChanger(username: String, eventTitle: String): boolean + cancelChanger(username; String, eventTitle: String): boolean

# ImportExportEventController

Message Controller

importEvents(): BuildingManager exportEvents(events: BuildingManager): void

# gateway: EventGateway

### InitialPrompt

+getBuilding(): BuildingManager

-password: String -username: String

+organizer: Attended

-menu: StartingMenu

+NewUserController()

+usernamePrompt: String

+passwordPrompt: String

-building: BuildingManager

+getReadout(): String

+toFriends(inputMessage: String)

- -role: String -menu: LoginMenu -presenter: StartingMenu
- -buildingManager: BuildingManager

+BuildingController(buildingName: String) +BuildingController(building: BuildingManager)

OrganizerMessageController

+toAllSpeakers(inputMessage: String, manager: LoginUserManager)

+toAllAttendee(inputMessage: String, manager: LoginUserManager)

NewUserController

BuildingController

+addRoom(name: String, startHour: int, endHour: int): boolean

+toOneSpeaker(oneSpeaker: String, inputMessage: String)

+toOneAttendee(oneAttendee: String, inputMessage: String)

+logReg(username: String, password: String, role, String)

+InitialPrompt(buildingManager: BuildingManager) +startProgram(): void

# UserMenu

- username: String
- UserMenu(username: String)

**Presenters** 

- optionalAttendee(): void
- optionalOrganizer(): void - awaitResponse(): void
- invalidResponse(): void
- rprintBuildingSchedule(building: BuildingManager): void
- + eventPrompt(action: String): void signUpEventStatus(eventTitle: String, status: String): void
- + cancelEnrolStatus(eventTitle: String, status: String): void + manageEvent(): void
- + sendMessageUser(): void + sendMessageContent(): void
- + printSomething(): void + createRoomName(): void
- + createRoomStart(): void + createRoomEnd(start: int): void
- createSpeakerName(): void
- speakerMade(): void enterEvent(): void + printMessages(): void
- createEventName(): void + createEventRoom(): void - createEventYear(): void
- + createEventMonth(): void - createEventDav(): void + createEventHour(): void
- + createEventSpeaker(): void + createEventSpeakerName(): void

# StartingMenu

- + initialPrompt(): void
- + alreadyRegistered(): void
- + rolePrompt(): void + loggedInPrompt(): void
- + uPrompt(): void + pPrompt(): void
- + failedPrompt(): void

- + MessageGateway(): void + getInboxPath(): String
- + getOutboxPath(): String
- + setOutboxPath(outboxPath: String): void
- setInboxInProgram(inbox: HashMap<String, ArrayList<Message>>): void setOutboxInProgram(outbox: HashMap<String, ArrayList<Message>>): void
- + removeMessage(sender: String, receiver: String, message: Message): void
- setInbox(): void + setOutbox(): void

# **EventGateway**

+ setEvents(events: BuildingManager): void

# Gateways

# MessageGateway

- + inboxPath: String outboxPath: String

- setInboxPath(inboxPath: String): void
- getInboxInProgram(): HashMap<String, ArrayList<Message>>
- getOutboxInProgram(): HashMap<String, ArrayList<Message>>
- addNewMessage(sender: String, receiver: String, message Message): void
- + getInbox(): HashMap<String, ArrayList<Message>>
- + getOutbox(): HashMap<String, ArrayList<Message>>
- + getEvents(): BuildingManager

### + setEventStatus(eventStatus: EventStatus): void + loadFromFile(filePath: String): EventStatus + saveToFile(filePath: String): void

EventStatusGateway

- FriendListGateway
- + friendlist: HashMap<String, ArrayList<String> + filePath: String

+ eventStatus: EventStatus

getEventStatus(): EventStatus

- + FriendListGateway(): void + getFriendlist(): HashMap<String, ArrayList<String>
- + setFriendlist(HashMap<String, ArrayList<String>): void + getFilePath(): String

### + setFilePath(filePath: String): void + getFriendlistFile(): HashMap<String, ArrayList<String> + set FriendlistDatabase(): void

UserLoginInfo + loginUserManager: LoginUserManager

- + UserLoginInfo(): LoginUserManager
- + getLoginUserManager(): LoginUserManager + setLoginUserManager(loginUserManager: LoginUserManager)
- + setFileUserLoginInfo(String filePath): void
- + getFileUserLoginInfo(String filePath): LoginUserManager