UML Model **Entities:**

<< abstract>> User

- username : String - password : String

+ User(username : String, password : String)

+ getUsername() : String + getPassword(): String

+ setUsername(username : String) : void + setPassword(password : String) : void

+ authenticate(username : String , password : String) :

Boolean

+ toString() : String



RegularUser

- isBanned : boolean

+ RegularUser(username : String, password :

String)

+ setIsBanned(bool : boolean) : void

+ getIsBanned(): boolean

+ isAdmin(): Boolean

AdminUser

+ AdminUser(username : String, password : String)

+ isAdmin(): boolean

History

action : Action(enum)username : Stringtime : LocalDateTime

+ History(username : String, action : Action(enum))

+ getAction() : Action(enum)
+ getUsername() : String
+ getTime() : LocalDateTime

+ toString() : String

Label

- name : String

- isThematic : boolean

+ Label(name : String, isThematic : boolean):

+ getName() : String

+ setName(name : String) : void

+ isThematic() : booleam

+ toString() : String

Calendar

 $\hbox{-} \ calendar ID: String \\$

- name : String

- description : String

+ Calendar(name : String, description :

String):

+ setName(name : String) : void

+ setDescription(description String): void

+ getName() : String

+ getDescription() : String

+ getCalendarID () : String

+ setCalendarID(CalendarID : String) : void

+ toString(): String

TimeParser

+ parseTime(time : String) : LocalDateTime

- YYYYMMDDHHMM(time : String) : LocalDateTime

- YYDDHHMM(time : String) : LocalDateTime

- DDHHMM(time : String) : LocalDateTime

- YYYYMMDDHH(time : String) : LocalDateTime

 $\hbox{-} \ YYYYMMDD (time: String): Local Date Time$

- MMDD(time : String) : LocalDateTime

- DD(time : String) : LocalDateTime

- intSlice(text : String, i : int, j : int) : int

- tomorrow() : LocalDateTime

- nextDat(time : String) : LocalDateTime

NameDuplicatable

+ assignID(name : String, ids : Iterable<String>, username :

String)



Task

+ Task(name : String, description :

String):

+ toString(): String

+ isTask(): boolean

<<abstract>> Plan

planID : Stringname : Stringdescription : Stringsubplans : List<Plan>endTime : LocalDateTime

+ Plan(name : String, description : String) :

+ getName() : String + getDescription() : String

+ setName(newName : String) : void

+ setDescription(newDescription : String) : void

+ getPlanID(): String

+ setPlanID(planID String) : void + getAllSubPlans() : List<Plan>

+ getSubplan(subPlanID : String) : Plan

+ isChildOf(plan : Plan) : Boolean + addSubPlan(child : PLan) : void + removeSubPlan(child : Plan) : void

+ compareTo(other Plan) : int

+ isTask(): Boolean

+ getEndTime() : LocalDateTime

+ setEndTime(endTime : LocalDateTime) : void



Event

- startTime : LocalDateTime

+ Event(name : String, description : String) :
+ setStartTime(startTime : LocalDateTime) :

+ getStartTime() : LcoalDateTime

+ toString : String + isTask() : boolean

Use Case Class:

UserManager

- users : Map<String, User>
- + getUser(username : String) : User
- + authenticateUser(username : String, password : String) : User
- + createUser(username : String, password : String) : String
- + createAdminUser(username : String, password : String) : String
- + deleteUser(username : String, deleteSelf : boolean) : String
- + banUser(username : String) : String+ unbanUser(username : String) : String- isBanned(username : String) : Boolean

- addUser(newUser : User) : void

HistoryManager

- histories : Map<LocalDateTime, History>
- + HistoryManager()
- + cleanHistories(): void
- + addHistory(username : String, action : History.Action)
- + getHistoriesByUser(username : String) : String
- + clearHistoriesByUser(username): void

LoggingManager

- currUser : User
- + getCurrUsername() : String
- + userLoggedIn(): Boolean
- + adminUserLoggedIn() : Boolean + logIn(user : User) : String
- + logOut() : String

CalendarManager

- calendars : Map<String, Calendar>
- currCalendarID : String
- + getCalendar(calendarID : String) : Calendar
- + createCalendar(name : String, description : String) : String
- + deleteCalendar(calendarID : String) : String
- + setCurrCalendarID(currCalendarID : String) : boolean
- + getCurrCalendarID() : String addCalendar(c : Calendar) : void

PlanManager - allPlans : Map<String, List<Plan>> - timeParser : TimeParser - currPlanID : String + PlanManager(): + getAllPlanIDs(CalendarID : String) : List<String> + handlecreateCalendar(calendarID : String) : void + createTask(name : String, description : String, CalendarID : String) : String[] + createEvent(name : String, description : String, CalendarID : String) : String[] + getAllPlans(CalendarID String): List<Plan> + getPlan(planID : String, CalendarID : String) : Plan + removePlan(PlanID : String, CalendarId : String) : String + changePlanName(name : String, planID : String, CalendarID : String) : String[] + changePlanDescription(description: String, planID: String, CalendarID: String): String + nest(parentPlanID : String, subPlanID : String, CalendarID : String) : String + unnest(parentID : String, subPlanID : String, CalendarID : String) : String + clearTasks(CalendarID : String) : String + clearEvents(CalendarID : String) : String + handleclearCalendar(calendarID : String) : void + setCUrrPlanId(PlanID : String, CalendarID : String) : void + getCurrPlanID(): String + isEvent(planID : String, CalendarID : String) : Boolean + setTime(eventID : String, startTime : String, endTime : String, calendarID : String) : String + setDeadline(taskID : String, deadline : String, calendarID : String) : String + viewSchedule(allPlans : List<Plan>) : String + viewScheduleByLabel(calendarID: String, labelMap: Map<String, Map<String, Label>>, labelName: List<String>): String + viewUpcoming(allPlans : List<Plan>) : String + viewEvents(currCalendarID : String) : String + viewTasks(currCalendarID : String) : String

LabelManager

- labels: Map<String, Map<String, Label>>
- + getLabels(): Map<String, Map<String, Label>>
- + handleNewPlan(planID : String) : void
- + handleDeletePlan(planID : String) : void
- + handleClearUserPlans(planIDs : List<String>) : void
- + handleSetPlanName(oldID : String, newID : String) : void
- + containsPlanID(planID : String) : boolean
- + addThematicLabel(labelName : String, planID : String) : String
- + addProgressLabel(labelName : String, planID : String) : String
- + set Label (label Name: String, label Name2: String, planID: String): String
- + deleteLabel(labelName : String, planID : String) : String
- + clearPlans(planIDs : List<String>) : void
- + clearLabels(planID : String) : String
- + toString(planID : String) : String

ShellData

- users : UserManager
- history : HistoryManager
- logging : LoggingManager
- calendar : Calendar Manager
- plans : PlanManager
- labels : LabelManager
- presenter : ShellPresenter
- saver : Saver
- + ShellDate():
- + getUsers(): UserManager
- + getHistory() : HistoryManager
- + getLogging() : LoggingManager
- + getPresenter() : ShellPresenter
- + getCalendar() : CalendarManager
- + getLabels() : LabelManager
- + getPlans(): PlanManager
- + getSaver() : Saver

Controller:

ShellController

- data : ShellData
- menus : Map<MenuState, ShellMen>
- + ShellController():
- + run(): void

MenuState

+ enum

<<abstract>> Presenter

- + getInput() : String
- + displayResult(result : String)
- + getArguments(allowEmpty : String, agrs :

String...)

+ getArguments(allowEmpty : Boolean,

args: String)

- + loggedOutPrompter(): String
- + userPrompter() : String
- + adminUserPrompter(): String
- + editCalendarPrompter() : String
- + editPlanPrompter() : String
- + editLabelPrompter(): String

ShellPresenter

- + getInput(): String
- + displayResult(result : String) : void
- + getArguments(allowEmpty : String, args : String...) : String[]
- + getArguments(allowEmpty : Boolean, args : String) : String[]
- # loggedOutPrompter(): String
- # userPrompter(): String
- # adminUserPrompter(): String
- # editCalendarPrompter(): String
- # editPlanPrompter(): String
- # editLabelPrompter(eventPropter boolean) : String

ShellMenuEditLabel

- + ShellMenuEditLabel(data: ShellData):
- + run(data : ShellData) :
- currPlanID(data: ShellData): String
- back(): void
- editLabel(data : ShellData) : void
- deleteLabel(data : ShellData) : void
- clearLabels(data : ShellData) : void
- addThematicalLabel(data: ShellData): void
- addProgressLabel(data : ShellData) : void
- viewLabels(data : ShellData) : void

ShellMenuLoggedOut

- + run(data : ShellData) : void
- logIn(data : ShellData) : void
- newUser(data : ShellData) : void

ShellMenuEditCalendar

- +ShellMenuEditCalendar(data ShellData)
- + run(data : ShellData) : void - back(data : ShellData) : void
- viewSchedule(data : ShellData) : void
- viewScheduleByLabel(data: ShellData): void
- viewUpcoming(data: ShellData): void
- editPlans(): void

GateWay:

<<abstract>> ShellMenu

- + commands : Map<String, Runnable>
- menu State : MenuState
- # setMenuState(menuState : MenuState) : void
- # getMenuState(): MenuState
- # handelInput(data : ShellData, input : String) : void
- + run(data : ShellData)

*S*hellMenuEditPlan

- + ShellMenuEditPlan(data : ShellData)
- + run(data ShellData) : void
- /- currCalendarId(data : ShellData) : String
- back() : void
- addEvent(data : ShellData) : void
- addTask(data : ShellData) : void
- addSubevent(data : ShellData) : void
- addSubtask(data : ShellData) : void
- editname(data : ShellData) : void
- editDescription(data : ShellData) : void
- editLabels(data : ShellData) : void
- editTime(data : ShellData) : void
- nestPlan(data : ShellData) : void
- unnestPlan(data : ShellData) : void
- deletePlan(data: ShellData): void
- clearTasjs(data: ShellData): void
- clearEvents(data: ShellData): void
- viewEvents(data: ShellData): void
- viewTasks(data : ShellData) : void

ShellMenuUser

- + run(ShellData : data)
- # viewHistory(data : ShellData)
- # logOut(data : ShellData) : void
- ${\it \# deleteOwnAccount(data:ShellData):void}\\$
- # editCalendar(data : ShellData) : void
- # save(data : ShellData) : void

ShellMenuAdminUser

- + ShellMenuAdminUser(data: ShellData)
- + run(data : ShellDate) : void
- newAdminUser(data ShellData) : void
- banUser(data ShellData): void
- unbanUser(data ShellData) : void
- deleteNonAdminAccount(data ShellData) : void

<<abstract>> Saver

+ save(saveable : Serializable)

+ load() : Object

ShellSaver

- filename : String

+ save(saveable : Serializable) : String

+ load() : Object