**Event Organizer Design Document**

Authors: Alec Zaitz, Khoi Vu, Bode Reed

March 24, 2025

LC3 Project, Utah Valley University.

Event Organizer

**Introduction**

**User Story**

As an event organizer, I want to plan my all-day event agenda and so that activities are scheduled around the projected sunset time, so that there are no conflicts between daytime and evening events.

We are an unpaid event organizing group of students attending UVU creating an event organizing application for our above user. It will ask the user to input the Event Date, Event Name and Location, and Estimated Sunset Time.

Each team member has their own role in this project. All members will be testing and analyzing the program as it is being created and when the project is completed. Alec Zaitz, Khoi Vu, and Bode Reed are all the programmers of this project. Alec is in charge of the Create Event Schedule Routine, including the Calculate Launch time routine. Khoi is in charge of the Hours, Minutes, TranslateToDecimal, Validate, and Sunset time Routines. Bode is in charge of creating the military time format, as well of its subcategories, and calculator. All members will be responsible for giving the presentation.

**We will approach this with an agile approach by breaking this into smaller “sprints”. We will meet every Tuesday and Thursday to allow for communication between stakeholders. We will deliver one feature at a time according to the list below.**

* Flesh out core functionality of the user’s inputted sunset time below. **(MAIN)**
  + V1: Store a set agenda of events in memory, allow user’s input of sunset time to adjust agenda time.
* Possible functionality if time allows:
  + User inputted events name, date and time of event

**Procedural Design**

* **Main**
  + LOAD address pointers for storage
  + Initialize Hours routine
  + Initialize ValidateHours routine
  + Initialize Minutes routine
  + Initialize ValidateMinutes routine
  + Initialize Launch Start Time routine
  + **User adds in Sunset Time**
  + Grab existing events from memory
    - Possible feature: Allowing event input from user
* **Hours Routine - Alec (Apr. 10th)**
  + PRINT “Enter hour (0 - 23): ”
  + Check for each user input (In-class PP - 3/25)
  + Call TranslateToDecimal Routine
    - Translates user’s input to decimal to allow ValidateHours to function
  + Call ValidateHours Routine
  + Store valid hour in memory
* **Minutes Routine - Bode (Apr. 10th)**
  + PRINT “Enter minutes (0 - 59): ”
  + Check for each user input (In-class PP - 3/25)
  + Call TranslateToDecimal Routine
    - Translates user’s input to decimal to allow ValidateMinutes to function
  + Call ValidateMinutes Routine
  + Store valid minute in memory
* **TranslateToDecimal Routine - Khoi (4/10 due)**

Allows checking for double digit number values to validate whether an entry is correct.

* + Initialize R0 to 0
  + Take in a single digit as ASCII
  + Convert ASCII to decimal
* **ValidateHours Routine - Alec (Apr. 10th)**
  + Error handling: check if imputed number is 0 =< number =< 23
  + If invalid, PRINT “Incorrect input, please enter a number between 0 - 23”
* **ValidateMinutes Routine - Bode (Apr. 10th)**
  + Error handling: check if 0 =< number =< 59
  + If invalid, PRINT “Incorrect input, please enter a number between 0 - 59”
* **SunsetTime Routine - Khoi (4/10 due)**
  + Call Hours Routine, user adds in hour of sunset (0 - 23)
  + Call Minutes Routine, user adds in minute of sunset (0 - 59)
  + Grab Hour from memory
  + Grab Minutes from memory
  + **MilitaryTimeFormat**
    - (HHMM) = Double digit(ASCII) hours followed by double digit(ASCII) minutes (e.g. 2359 = 23(HH)59(MM))
  + Example: PRINT “Sunset time: {Hour} : {Minute}”
* **Calculate Launch Start Time**
  + Increment +1 to minutes input until minutes == 30 || minutes == 00
    - If minutes == 00, hour += 1 (16:47 => 17:00)
  + Return new time
* **Calculate MilitaryTime**
  + Time = hour \* 100 + minute
    - (e.g. hour = 1265 / 100 = 12.65(doesn’t save the .65), minute = 1265 % 100 = 65
    - Adjust: hour = 12 + 1 = 13(HH)
    - 65 - 60 = 05(MM)
    - newTime = 13 \* 100 + 5 = 1305(HHMM)
* **Calculate HourTime(MilitaryTime Subcategory)**
  + Hour = (time / 100)
    - 2300 / 100 = 23(HH)
    - Hour(HH) \* 100 + minute(MM) == 2400 (e.g. 2359 + 0001 = 0000, or 0159 + 0001 = 0200)
* **Calculate MinuteTime(MilitaryTime Subcategory)**
  + Minute = (time % 100)
    - 0059 % 100 = 59(MM)
    - Minute >= 60

**Diagram**

