AstroCal

Software Design Document

Astrocal

13 October 2022

Second Document Version

1 Introduction

AstroCal is a platform currently functioning on macOS and Windows desktop computers. It is set to display a calendar and show information related to celestial objects such as the sun and events related to these objects such as a solar eclipse.

1.1 Purpose

The intent behind the writing of this System design document is to familiarize the users of the system with the features the astronomical calendar has to offer and the astronomical events the user is shown such as eclipse data.This document is prepared for future developers of the system and users to become familiar with the graphical user interface.

1.2 Scope

· The software product being produced is called “AstroCal”

· Activities the software product will accomplish:

- Show a month calendar with current moon phase showing for a particular day
- Change month, year and day in calendar
- Display sun, moon, and eclipse information on a day page
- Display upcoming moon and sun events such as solar eclipse,lunar eclipse and details related to events such as date, type of event .
- Show a solar and lunar calendar month
- Display information for years 1900 to 2100

The application is intended to show a calendar and moon and sun information ,as well as eclipse information while taking into account a user's location, altitude, latitude, and longitude. An initial goal of the system which could be achieved by the continuation of the product was the translation of the location of solar objects such as the sun into graph format. The benefit of this software is that it shows information related to celestial objects that can be both useful to a regular person, but also a scientist who might use this data in their analysis

A bottom up approach was adopted in relation to the design. Where a paper form of the system was first designed then a more comprehensive prototype image was made with a prototyping tool.In terms of the product architecture the product uses MVC design pattern. The front-end is made using Kivy while the backend is mainly python to handle the data as well as sqlite which handles the geographic location database.The does not interface many external systems. It does rely on the swisseph library to gain all its data.interface. This document will cover more product complexity as the product has more features.

1.3 Definitions, Acronyms, and Abbreviations

MVC - Model View Controller
SDD - Software design document
GUI - Graphical User Interface

1.4 References

There are no references used in the software design document.

2 Design Overview

2.1 Background Information

MacOS and iOS compatibility was a priority when designing the system. AstroCal was designed for desktop with the intention of being adapted for mobile devices. Kivy was used for the GUI as it is compatible with MacOS and iOS. The application was written in python 3.10.7 and uses the Swiss Ephemeris toolset to perform calculations. The Model View Controller design pattern was used to allow multiple developers to work simultaneously on the application.

The key features that application is designed to perform include:

- Display the moon and sun status
- Display the moon-day, sun-day, and the rise and set of the moon and sun
- Display lunar eclipses and solar eclipses
- Display astronomical information in a calendar and a for individual days
- Select a date to display from 1900 to 2100
- Select a geographical location

The stakeholder is Youry Khmelevsky who can be contacted through email at ykhmelevsky@okanagan.bc.ca.

2.2 Alternatives

Tkinter was considered as it is one of the most commonly used GUI programming toolkits for Python. The group decided that Tkinter would not be a good fit for the application because despite Tkinter claiming to be Mac compatible, the application ran very differently when tested on Mac computers.

3 User Characteristics

Potential users of the system may include day to day people wanting to access sunrise or sunset data, or astronomers who may use the data about celestial objects in their practices. A novice level of expertise is needed to interact with the system, though anyone can use it. The system will meet specific design user requirements by providing a good user layout of all the sun and moon information as well as eclipses.

4 Requirements and Constraints

4.1 Performance Requirements

The proposed design ensures that all associated performance requirements are met by providing multiple tabs that handle the different functionality.Three tabs are provided a month tab which provide all calendar related features,a day tab with sun and moon daily information and an events tab which provides all information on the sun and moon events such as eclipses

4.2 Security Requirements

There are no system access restrictions in place for any users. All system users should be able to access all parts of the system. In terms of security of the system, the proposed design is that the user will be able to enter specific data - for example, the date. If that is not provided, default options are provided to the user. This prevents breakage of the system and maintains the security requirements.

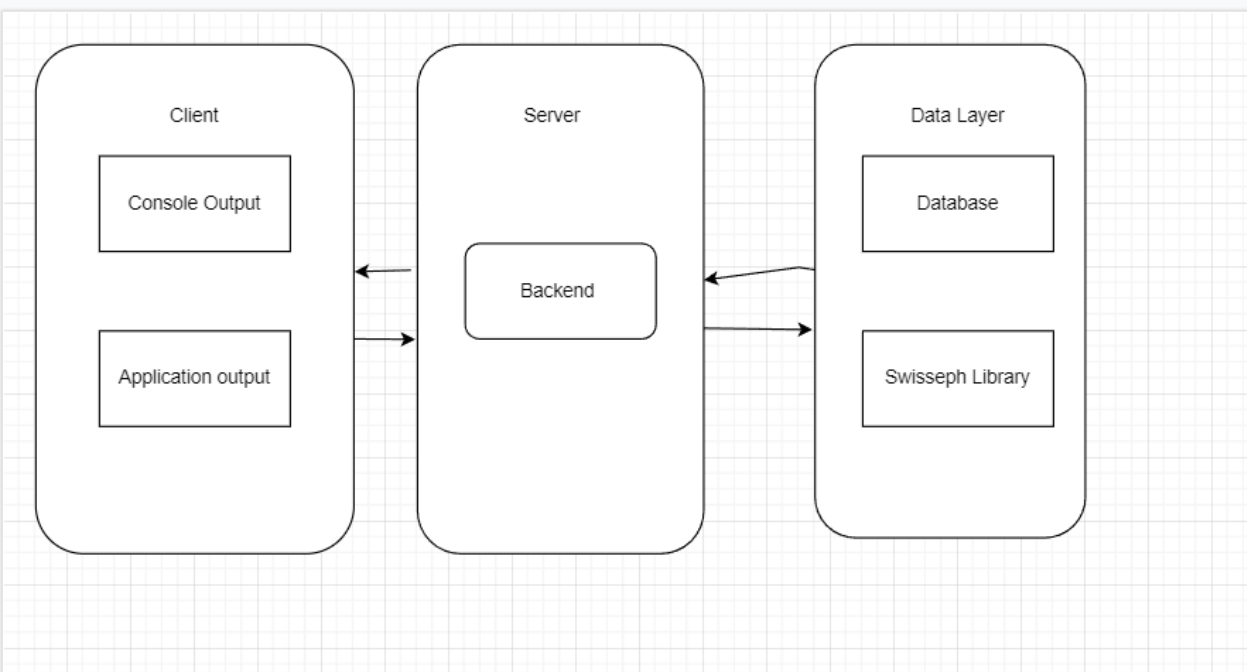
4.3 Design Constraints

Design limitations that affected the product was the time of 5 weeks allocated to the production of the product, which was not long enough. Requirements that placed constraints on the system design have been removed from the current list of tasks - for example, the creation of graphs for the location of celestial objects. Such functionality could have been developed if the project were to continue longer. Scheduling issues were a main limitation in accomplishing product requirements.

5 System Architecture

As of now we have parts of the system like the database communicating with the backend to display to the console for the client and parts of the data layer like swisseph which is

responsible for all calculations and communicating with the backend to display both in the console and the Graphical user interface application itself.



6 Detailed Design

6.1 Description for Component n

Functions:

- celestial_rise_or_set(): returns a formatted text of when the celestial objects i.e. the sun and moon rise and set
- utc_hack(): returns a datetime format from a different format
- format_24hour_time_output(): formats a datetime object into 24 hour text output
- get_RiseSet(): returns datetime format of the rise and set of a celestial object
- getDateOfNextFullMoon_UTC(): gets the date of the next full moon in utc time
- getDaysTillFullMoon(): return days till next full moon based on current local time zone
- getWhenSolEclipseLoc(): returns comprehensive data related to solar eclipses
- getWhenLunEclipseLoc(): returns a wide array of data related to lunar eclipses
- getMoonStatusHelper(): returns the current illumination of the moon
- getMoonStatus(): returns the current phases of the moon
- getVariableDayLength(): returns the amount of daylight time a day has
- getDateOfNextNewMoon(): returns the date of the next new moon
- Main_menu is responsible for displaying all data to a terminal and connects all menus
- Sun_menu is a menu that leads with all solar events in console

- Moon_menu is a menu that leads with all lunar events in console
- getCurrentdate(): retrieves today's date
- getCurrentMonth(): retrieves the present month
- calendarStartDay() and setUpDaysInMonth() are functions that work together in the building of the Calendar GUI
- getLocation() returns the elevation,timezone,latitude and longitude of a particular city,country combination
- show_help() displays instructions of how to run the application
- clear() : clears console
- restrictInputToRangeInclusive() commitor user input to be in certain range
- checkInputType() makes sure user input is of expected type
- getInput() works with input function to request input from the user
- getInputSanitized() cleans input from the user
- getDateFromUser() gets date from the user
- get_date_formatted() gets current date in formatted way
- get_current_year_month_day(0 gets current year,month and day
- getMonth() displays data for each day of the current month
- getMoonRise(),getMoonSet(),getSunRise() and getSunSet() return moon and sun data for the GUI
- eventView() is responsible for displaying event tab
- monthView() displays month view
- dayView() displays day view

Classes:

- CalGrid: Creates the calendar grid layout for the GUI
- CalendarApp: The class that helps in running of the GUI application
- HBoxLayoutExample: class creates home page view of GUI application

6.1.1 Processing for Component n

Overall all components take data such as the a day,time,year,a location and a celestial object type and use the appropriate function to perform calculations and display the output in a particular format using the menu functions for the console or pass the data to the GUI class to be processed and displayed in the GUI screen.

6.1.2 Component n Interface Description

The main_menu component relies on other menu components to produce the expected output. celestial_rise_set relies on three components, getRiseSet which gets a raw datetime data,

utc_hack changes this time in local time and format_24hour_time_output changes that time into 24 hour format.The components perform calculations and the result of these calculations are displayed to the user.

7 Data Architecture

The data used in this application comes from the Swiss Ephemeris (swisseph) library, developed by AstroDienst. This library is a helpful tool for displaying sunrise, sunset, moonrise, and moonset times as well as moon phases for a particular location on a particular day.

7.1 Data Analysis

Several functions within the swisseph library are used for calculations given different parameters, which is then output as data to the user. One of these in particular, the "rise_trans()" function, which calculates rising and setting times for the sun and moon, takes as parameters a given Julian calendar day (which is then converted to its Gregorian equivalent), and a set of geopositional coordinates in latitude, longitude, and altitude.

7.2 Output Specifications

When a user views information for a particular day, they will be able to see the rise and set information for their location on that day as output within a GUI window. The times shown are adjusted for the user's time zone, as the program initially generates a time in the UTC time zone.

8 User Interface

The AstroCal application can be run through a command line interface or through a desktop GUI application.

Desktop GUI Application

The graphical user interface for the desktop application is created with Kivy. Kivy is an open source Python framework that is compatible with iOS, macOS, Android, Windows and Linux. The application has 3 pages that can be traveled to through a navigation bar at the bottom of the application. The pages are 'month', 'day' and 'events'.

Month Page:

The month page is the default page of the application. It displays astronomical information in a monthly calendar. By default it will display the current month the user is in. Above the calendar there is an ‘events’ button and a ‘moon’ button that can be used to switch the contents of the calendar. If the user clicks the 'events' button, the calendar will display any events happening in the month such as lunar eclipses. If the user clicks the 'moon' button, the calendar will display the phase of the moon for every day in the month. The events and phases of the moon are represented in the calendar with graphical icons and have text below them to identify what they are. Every cell in the calendar is a button. When a user clicks on a day in the calendar it will redirect them to that day in the day page. At the top of the application there is a header that contains the date, location, a right arrow button and a left arrow button. If the user clicks on the date then they will be able to change the month and year of the calendar. If the user clicks on the location then they will be able to change the location. If the user clicks on the left or right arrow buttons they will be able to go one month forward or one month backward respectively.

Day Page:

The day page displays astronomical information for a specific day. By default it will display the current day that the user is in. The information displayed on this page includes the phase of the moon and the times of the sunset, sunrise, moonset, and moonrise. It will also tell the user if there is an event happening that day.

Events Page:

The events page will display information about upcoming or past astronomical events.

Command Line Interface

The command line interface was written in Python 3.10.7 and can be launched by running the console_output.py file in a terminal. When the console program is launched it will print the main menu. The main menu has four menu options for the user to select from. The options are '1. Sun Events', '2. Moon Events', '3. View Solar Month','4. Change Date’, and '5. Exit’. The user can select a menu item by typing in the number associated with it. If the user types in a number that isn’t provided as an option then the console will print ‘error not an option’ and then print the main menu again.

Sun Events:

The sun events menu has four options for the user to select from. The menu options are '1. Solar Eclipse', '2. Day Lengths', and '3. Back’. If ‘solar eclipse’ is selected, the console will print the time the eclipse starts, the totality, the time it ends and the duration. If ‘ day lengths’ is selected, the console will request the user to type the amount of days they want to view,the user can enter up to 500 days and if they do not provide input the user is given days length for 30 days. The console will print the current day and the time the day begins, the time the day ends and the length of the day. If ‘back’ is selected, then the console will return to the main menu.

Moon Events:

The moon events menu has five options for the user to select from. The menu options are '1. Lunar Eclipse', '2. View Moon Status', ‘3. Date of Next New Moon’, ‘4. Date of Next Full Moon’, and '5. Back’. If ‘view today’ is selected, the console will print the current day and the time of the moonrise and moonset. If ‘lunar eclipse’ is selected, the console will print the time the eclipse starts, the totality, the time it ends and the duration. If ‘view moon status’ is selected, the console will print the phase of the moon for the current day. If ‘date of next new moon’ is selected, then the console will print the date of the next new moon.If ‘date of next full moon’ is selected, then the console will print the date of the next full moon. If ‘back’ is selected, then the console will return to the main menu.

View Month:

The console will print the current solar month and year. For every day in the current month the console will print the day of the week,the moon phase, times of the sunset, sunrise, moonset, and moonrise.

Change Date:

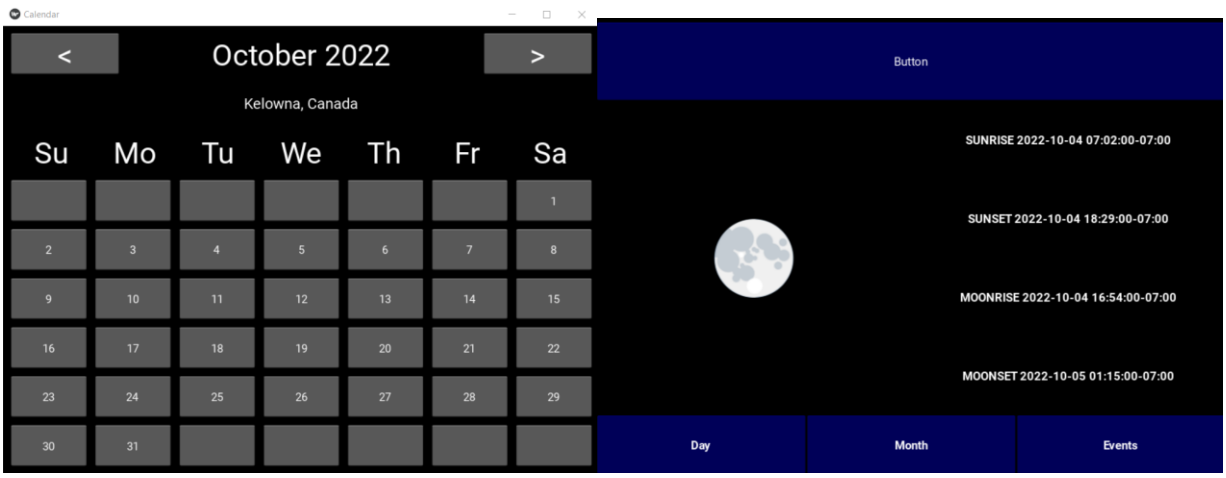
The user is asked to select a year between 1000 and 3000 ,if user does enter anything they input is defaulted to the current year.The user is asked to enter a month between 1 and 12 for the solar month otherwise their input is defaulted to their current month.Then a user can enter a day between 1 and 31 ,otherwise their input defaults to the current day.That way the user changes a date.

Exit:

The console will print ‘Bye’ and then the program will terminate.

8.1 [Module X] Interface Design

Desktop GUI Application:



Desktop GUI Application Prototype:

8.2 Functionality

- Navigation bar
  - Composed of 3 buttons labeled ‘Month’, ‘Day, ‘Events’
  - When clicked switches between month, day and events pages
  - Used in the month, day and events page
- Right arrow button
  - When clicked changes calendar to the next month
  - Used in the month and page page
- Left arrow button
  - When clicked changes calendar to the previous month
  - Used in the month and page page
- Events button
  - When clicked updates the calendar to display the events in that month
  - Used in the month page
- Moon button
  - When clicked updates the calendar to display the phases of the moon in that month

    - Used in the month page
- Calendar
    - Composed of 42 buttons representing each day in the month.
    - When a day in the calendar is clicked the application is redirected to the day page for the day selected
    - Used in the month page

9 Non-Functional Requirements

This application is intended to immediately give users the data they request, without delay. As this is a small application which is not very graphically intensive, it should load in a timely manner. Users should not see performance of their system negatively impacted by running it. Additionally, since this application uses Python version 3.10.7, the application is compatible with PCs running Windows 8 or later, or on a Mac machine running macOS version 10.9 (Mavericks) or later. Any issues with performance users may experience would most likely be attributable to having too many other open applications at once, or attempting to run the application on older hardware.