PiClock

Software Requirements Specification

Jacob Alspaw
Arbazuddin Ahmed
Kareem Taleb
Benjamin Young

An open source, smart alarm clock platform that aims to improve upon the functionality of traditional alarm clocks by integrating new and useful technologies to motivate users. PiClock increases productivity by helping users overcome their early morning exhaustion and providing general information about their upcoming day.

Version 1.0

September 23, 2018

1. Introduction

1.1 Purpose

This Document details the vision and requirements for the PiClock smart alarm clock. It should be used as guidance for the team implementing the PiClock to ensure that the product is built in accordance with the team's vision.

1.2 Overview of Project Features

The PiClock is a smart alarm clock implemented on a Raspberry Pi single-board computer. It requires users to complete a small mental task to shut off the alarm and provides additional helpful features, including a weather forecast, news, motivational quotes, and a countdown until the next major holiday.

2. Vision and Scope

2.1 Business Requirements

2.1.1 Market Opportunity and Consumer Needs

Nearly everyone whose work or school schedule requires that they wake up earlier than they are naturally inclined uses a traditional alarm clock. These devices wail in the user's ear until they manage to hit a button with their flailing arm, at which point the clock goes silent. Unfortunately, the simple act of pressing a button is often insufficient to fully wake a sleeping person; they often drift back to sleep once the alarm stops. This all-too-familiar script has resulted in countless rushed mornings and late arrivals, especially among sleep-deprived college students. Our group has proposed a simple solution to this problem. What if the process of shutting off the alarm was more difficult than the thoughtless press of a button?

If the user, instead of blindly swinging their fist, was forced to solve even a simple mental task, their desire to shut the alarm up would force them to activate their brain. The effort required to complete a non-trivial task under the stress of a blaring alarm will

shock the brain into an alert state and wake people more effectively than a traditional alarm. In addition to this feature, a smart alarm clock can further stimulate the user's brain and help prepare them for the day ahead by providing personalized features beyond those of a traditional alarm clock.

2.1.2 Product Objectives

- Eliminate excess morning grogginess resulting from unnecessary time spent in bed.
- Decrease user's time spent in morning routine and thereby reduce instances of user arriving late to important responsibilities
- Inform users about their upcoming day

2.1.3 Success Criteria

- 5 minutes average time from initial alarm to user getting out of bed.
- 90% usage rate on mornings when user has responsibilities which require them to get out of bed rapidly.
- 50% usage rate on mornings when user does not have such responsibilities (such as weekends).

2.1.4 Risks

- Users may not enjoy the experience of rushing to solve a puzzle while the alarm is going off, and thus may not use the PiClock.
- User may sleep in close proximity to others who could be irritated by the noise of an extended alarm.
- The clock must be extremely reliable, as the alarm failing to go off could cause the user to miss an important morning responsibility, which could have negative consequences for the user.
- It is likely that many PiClock features are also provided by the user's smartphone, which the user may prefer instead.

2.2 Project Vision

2.2.1 Vision Statement

The PiClock is a smart alarm clock that helps early risers get out of bed faster while being better prepared for the day ahead. Instead of the alarm shutting off at the simple press of a button, the PiClock activates the user's brain by requiring that they complete a mental challenge to shut the alarm off. The PiClock also provides weather

metadata, news metadata, and inspirational quotes to further prevent the user from falling back asleep and to give them information about their upcoming day.

2.2.2 Major Features

F01: Basic clock with time and date, present on every screen
F02: User can set alarm to any time and must complete a game to shut it off
F03: User must complete a simple game to shut off the alarm
  F03.1: Tic-Tac-Toe
  F03.2: Concentration memory game
  F03.3: Simon memory game
  F03.4: Trivia game
  F03.5: Math game
F04: User can view the current weather conditions on home screen
F05: User can view the week's weather forecast on home screen
F06: User can view the day's top news stories on home screen
F07: User can view a random motivational quote on home screen
F08: User can view times in different cities around the world
F09: User can observe how many days until next major holiday
F10: Home screen rotates continually rotates between F04,F05,F06
F11: User can view Raspberry Pi's system statistics
F12: User can adjust application settings to change which game is to be completed when alarm rings, game difficulty, which applications appear on home screen, and API configuration

2.2.3 Use Cases

Anyone who sets an alarm to wake up in the morning is a potential user of the PiClock. The PiClock is especially intended for younger individuals - high school and college students - who often go to bed late and need to wake up early and whose natural circadian rhythm favors waking up later in the morning.

2.4 Scope and Limitations

2.4.1 Scope of Releases

Feature	Demo	Release 1	Release 2
F01	Fully Implemented		
F02	Fully Implemented		
F03	Partially Implemented	Fully Implemented	
F04	Implemented Time permitting	Fully Implemented	
F05	Implemented Time permitting	Fully Implemented	
F06	Implemented Time permitting	Fully Implemented	
F07	Implemented Time permitting	Fully Implemented	
F08	Not Implemented	Not Implemented	Fully Implemented
F09	Not Implemented	Not Implemented	Fully Implemented
F10	Implemented Time permitting and if 2 of F05-F08 are implemented	Fully Implemented	
F11	Fully Implemented		
F12	Implemented Time permitting - high priority	Fully Implemented	

2.4.2 Limitations

- User must have all the required hardware outlined in section 5.1
- Some features may require registration with third-party services
- Users are required to have exposure to Linux operating systems

3. Software Requirements

3.1 Clock

The core clock component will be responsible for displaying the current time of day to the user.

Requirement Code	Description
Clock.01	The clock will have a dedicated portion of screen reality, so the time will always be displayed
Clock.02	The clock time will automatically adjust to the system time set in the Raspberry Pi. Adjusting the time on the Raspberry Pi will adjust the time shown on the PiClock.
Clock.03	Times will be displayed in increments of hours and minutes; seconds will be omitted. Therefore, the clock will need to update at least once every minute.
Clock.04	The time can be formatted using either a 12 or 24 hour clock and can be changed by the user in the application's settings page. If the clock time is configured to use the 12 hour format, then the period of day must also be included ("am" or "pm")
Clock.05	Clock time is shown in red when alarm is ringing and is black otherwise

3.2 Alarm

The alarm is the most important feature of the application which sets apart the PiClock from other smart alarm clocks. The core alarm functionality should be persistent in waking up the user at a configured time. The user will need to complete a mental task before the alarm will stop sounding. Once the task is completed, users will have the option to start their day or to continue sleeping by hitting a snooze button which will reset, but also delay the alarm. If the

alarm is not “snoozed”, then it will be removed from the list of alarms.

Requirement Code	**Description**
Alarm.01	The noise emitted by the alarm will be a simple sine wave that alternates on and off.
Alarm.02	Users can adjust the sine wave’s frequency and thus control the pitch of the alarm.
Alarm.03	Users can adjust the sine wave’s amplitude and thus control the volume of the alarm.
Alarm.04	Users can specify the date and time for when an alarm should triggered.
Alarm.05	When the date and time specified for an alarm are reached, the alarm begins making the configured noise.
Alarm.05	Users can set up to 25 alarms.
Alarm.06	Once alarm begins ringing, it will not stop until user successfully completes the game or until failsafe is triggered.
Alarm.07	To safeguard against system failure, there must be two override switches to alarms that will work without failure themselves. The first fail-safe will trigger after a user excessively and continuously fails an alarm’s mental task. The second switch should be implemented as a button in the layout menu.
Alarm.08	Mental task options when settings an alarm will include challenges that are designed to immediately increase brain activity when users wake. Tasks will ensure that users are less likely to fall back asleep (see section 3.3).

3.3 Games

A game is a mental task that the user must complete to shut off the alarm.

Requirement Code	**Description**
Game.01	Games are only initialized upon the triggering of an alarm.
Game.02	If a game fails to initialize within 5 seconds, the user will need to be shown an error message detailing the problem and be given the option to trigger the alarm failsafe.
Game.03	If the user has not completed the game within a reasonable time frame (5 minutes) or excessively fails (5 times), the alarm will shut off as if the user had successfully completed the game.
Game.04	In system settings, User can choose between three difficulty settings (beginner, intermediate and advanced) that modify win conditions.
Game.05	Users are notified of their game progress with updates to user interface labels.
Game.06	User can choose which game will be played when alarm is triggered. Game options are detailed in sections 3.3.1 - 3.3.5.
Game.07	When a user successfully completes a game, they are informed that they have won.
Game.08	If Alarm.07 is triggered, users are informed that they have lost.

3.3.1 Tic-Tac-Toe

This game and its rules will conform to the popular children's game Tic-Tac-Toe. The game is played by two players, *X* and *O*, human and computer. Each player will take turns marking the spaces in a 3x3 grid. The player who succeeds in placing 3 contiguous marks wins.

Requirement Code	**Description**
TicTacToe.01	Game is played by user against and automated program that makes decisions based on random selection.
TicTacToe.02	User and computer take turns placing a symbol - user uses X and computer uses O - in squares on an 3x3 grid, with user playing first (see Fig. 9.3)
TicTacToe.03	Neither player can place a symbol in a square that is already occupied.
TicTacToe.04	If the user at any point places 3 X-marks in a row (along a row, column, or diagonal of the grid), the user wins.
TicTacToe.05	If the program at any point places 3 O-marks in a row, the program wins.
TicTacToe.06	If all of the squares contain a symbol and neither the user nor program has won, the game is a tie.
TicTacToe.07	The user is awarded 2 point for a win, 1 point for a loss, and 0 points for a tie.
TicTacToe.08	Beginner difficulty (see Game.04) requires that the user accumulate 6 points to complete the game. Intermediate requires 4 points and advanced requires 2 points.

3.3.2 Concentration Memory Game

This game and its rules will conform to the popular children's game Concentration. Concentration is a card game in which a set of cards are laid face down on a surface and two cards are flipped over each

turn. The objective of the game is to turn over pairs of matching cards.

Requirement Code	Description
Concentration.01	The game begins with a number of cards with identical backs randomly arrayed face-down in a grid (see Fig. 9.4).
Concentration.02	Each card has a number on the front, and exactly one other card has the same number.
Concentration.03	Each turn, the user flips over two cards by tapping them.
Concentration.04	If the two cards flipped over have the same number on the front, they stay face-up. Otherwise they flip back to face-down.
Concentration.05	The game is completed when all of the cards are face-up (the user has found all the matches)
Concentration.06	The grid of cards is 4x2, 4x3, and 4x4 on beginner, intermediate, and advanced difficulties, respectively.

3.3.3 Simon Memory Game

This game and its rules will conform to the popular electronic game called Simon. Simon is a memory game in which the computer creates a series of button presses that the user is required to repeat. The buttons are illuminated in a sequence that the user must remember and then reiterate. Each round the sequence grows by one button press.

Requirement Code	Description
Simon.01	The screen will appear with four colored buttons (see Fig. 9.5).
Simon.03	The game begins with one button press in the sequence. With each successful reiteration by the user, the sequence will grow by inserting an additional button press at the end of the sequence.

Simon.04	When the sequence is finished, the user attempts to press the buttons in the same order in which they were illuminated.
Simon.05	If the user presses a button out of order, the game informs the user that they were unsuccessful, displays the same sequence again, and gives the user another try.
Simon.06	If the user correctly replicates the entire sequence, they complete the game
Simon.07	The sequence is 4, 6, and 8 buttons long on beginner, intermediate, and advanced difficulties, respectively.

3.3.4 Trivia Game

In the trivia game, the user answers a sequence of randomly generated multiple choice questions. The game will ask the user several multiple-choice questions that the user is expected to answer correctly.

Requirement Code	**Description**
Trivia.01	To complete the game, the user must correctly answer a certain number of trivia questions.
Trivia.02	For each question, the user will be presented with a list of 4 potential answers (see Fig. 9.7). The correct answer's index within the set of possible answers should be randomized.
Trivia.03	If the user taps an incorrect answer, they are informed that their answer was incorrect and the game moves on to the next question.
Trivia.04	If the user taps the correct answer, they are informed that they are correct, they are given a point, and the game moves on to the next question.
Trivia.05	The game is completed once the user accumulates 3, 4, or 5 points on beginner, intermediate, and advances difficulties, respectively.

Trivia.06	The user is not shown the same question more than once in the same game.
Trivia.07	Trivia questions should be pulled from an extensive 3rd-party database using an API

3.3.5 Math Game

The Math game and its rules will conform to a simple multiple choice questionnaire. The game objective is to correctly answer a set of simple arithmetic multiple-choice questions.

Requirement Code	**Description**
Math.01	To complete the game, the user must correctly answer a certain number of trivia questions.
Math.02	For each question, the user will be presented with a list of 4 potential answers (see Fig. 9.6). The correct answer's index within the set of possible answers should be randomized.
Math.03	If the user taps an incorrect answer, they are informed that their answer was incorrect and the game moves on to the next question.
Math.04	If the user taps the correct answer, they are informed that they are correct, they are given a point, and the game moves on to the next question.
Math.05	The game is completed once the user accumulates 3, 4, or 5 points on beginner, intermediate, and advances difficulties, respectively.
Math.06	The user is not shown the same question more than once in the same game.
Math.07	Questions should be randomly generated in the format "<integer> <operator> <integer>?" where integers are within the set [-25, 25] and answers are within the set [-500, 500]. Allowed operations are addition, subtraction, multiplication, and division.

Math.08	Incorrect answers should be believable, meaning that if the question asks “what is 2 + 2” the user should not see an answer of 95.

3.4 Widget Slideshow

The PiClock software will come with a set of optional widgets that are responsible for displaying helpful information.

Requirement Code	**Description**
Widget.01	Each widget will have its own forked process responsible for accumulating and managing data and then updating the user interface.
Widget.02	Only enabled widgets will run. If the widget is disabled, then the widget’s process will not be forked.
Widget.03	Each widget will be shown on a slideshow. The slideshow will always run unless the alarm is ringing. Each slide will show for 10 seconds before showing the next slide. Once all slides have been shown, the slideshow will repeat like a circle queue.
Widget.04	Any widget that requires the use of a source API to retrieve information should prioritize two API characteristics. If possible, API usage should be free and should not require request authorization using an an API key.
Widget.05	Widgets will require configuration before they can be enabled. Users must enter required API keys when necessary. If a feature has not been configured, it should remain disabled.
Widget.06	Each widget should update immediately after it has been displayed in the slideshow.

3.4.1 Weather and Forecast

The Weather widget will display the configured area's current weather metadata while the Forecast widget will display the forecast for the next 4 days.

Requirement Code	Description
Weather.01	The process will need to communicate with a source API to collect and display information on the current day's high and low temperatures, the humidity, and wind speeds.
Weather.02	The collected information from the source API will need to be matched with a closely related weather infographic. The widget will display the graphic.
Forecast.01	The process will need to communicate with a source API to collect and display the forecast for the next 4 days. The current weather data will not be shown in the forecast as it already has a dedicated slide.
Forecast.02	Each day in the forecast will display the forecasted date, high and low temperatures, and matching infographic.

3.4.2 News

The news widget will be responsible for querying for today's current events from a variety of sources.

Requirement Code	Description
News.01	A selection of news article headlines will be displayed.
News.02	If possible, news article headlines should be included with a short synopsis of each article.
News.03	The articles should be related to currently trending topics.

News.04	Each article should be chosen at random from varying news agencies.

3.4.3 Inspirational Quotes

The inspirational quote widget will display randomized quotes from a variety of people to inspire early morning motivation.

Requirement Code	**Description**
Quotes.01	Quotes should seldom be reused. Every cycle that this widget is shown, the quote should update, preferably to an unused quote.
Quotes.02	Both the quote and its author should be displayed.

3.4.4 System Usage Statistics

The system usage statistics widget will display information about the user's Raspberry Pi system under load.

Requirement Code	**Description**
Usage.01	The statistics that need to be displayed are CPU usage, memory usage, and disk usage.
Usage.02	Each statistic will need to update multiple times per second.
Usage.03	The information will be displayed as a percent of the maximum possible values for the statistic. For example, the CPU can be at 21% usage.
Usage.04	The graphic in which the information is conveyed should change at varying percentage levels.

3.4.5 World Clocks

The clock's functionality will be extended as a widget that periodically displays times in six major cities around the world.

Requirement Code	Description
World.01	The cities that will be used are Tokyo, Los Angeles, New York City, Beijing, London, and Mumbai.
World.02	The time in these countries should be calculated using the Greenwich Mean Time (GMT) Offset of the cities' time zones.
World.03	The times will be displayed in increments of hours and minutes; seconds will be omitted.

3.4.6 Holiday Countdown

The Holiday Countdown widget will help users account for and track upcoming holidays by displaying the number of days until the next configured holiday.

Requirement Code	Description
Holiday.01	The holidays will be configurable by the PiClock user through the application's setting page.
Holiday.02	The application should be prepackaged with major United States holidays throughout 2030.
Holiday.03	Users should be able to opt-out of specific holidays as well as add holidays that are not prepackaged in the release.
Holiday.04	Up to 3 holidays can be configured on the same day.
Holiday.05	The countdown will display the number of days until the next date with major upcoming holiday(s).

3.5 Application Settings Control

Each feature will have its own set of controls to manage configuration options. All options mentioned below should be implemented and validated.

Requirement Code	Description
Settings.01	All settings will be validated to avoid bad input.
Settings.02	The Raspberry Pi should reboot when leaving the settings page to make sure all new settings have taken effect.
Settings.03	Settings should be grouped by corresponding feature.
Settings.04	Allows for all widgets to be enabled or disabled.
Settings.05	Enables control of the sleep settings for the display. The display can be set to turn off when it is not in use for a certain number of minutes.
Settings.06	Enables clock-time formatting. Time formatting for the clock can be based on a 12-hour or 24-hour cycle.
Settings.07	Allows for control of game difficulty. All of the game difficulty settings can be adjusted with one of three settings: beginner, intermediate, and advanced.
Settings.08	The weather and forecast location should be configurable with an “Where on Earth” Id (WOEID).
Settings.09	Allow users to register their required API key for the News widget.
Settings.10	Allow users to opt-in and opt-out of holiday countdowns. Users should be able to add holidays by entering the holiday name and date. Users should also be able to delete tracked holidays.

4. Software Dependency Requirements

4.1 Operating System

The Raspberry Pi will need to run a flavor of Linux called Raspbian Jessie, the version of which should offer long term support. Raspbian Jessie is the supported OS for the official Raspberry Pi touchscreen. Raspbian Jessie is also one of the most popularly supported flavors of Linux for the Raspberry Pi. To avoid unforeseen operating systems obstacles, the PiClock should run a widely supported and documented operating system.

4.2 Libraries and Frameworks

Libraries and frameworks that support static linking as opposed to dynamic linking should have priority when developing the PiClock software. Such library and framework dependencies are resolved at link time and, therefore, do not require end user installation on their own devices to use. They instead come prepackaged in the executable file after compilation. Our development team should choose these libraries to reduce system setup time required by the user and to reduce the need for end users to have advanced knowledge about linux operating systems.

Standard C++ libraries will be used to complete the project along with the WiringPi library. The WiringPi library is one of very few libraries that enables developers easy access to the GPIO pin set on the Raspberry Pi board. The GPIO pins are used to send a sine wave to the Piezo buzzer that will act as the alarm sound. The pins are also used to power the monitor’s touchscreen controls. The Standard C++ and WiringPi libraries support static linking, meaning that library dependencies are resolved at link time.

In addition, the Qt framework will be used to develop the graphical user interface. Specifically, the Qt framework has amazing support for implementing application animation and creating a graphical user interface through a large collection of supported libraries. The supported Qt IDE called Qt Creator also supports cross-platform compilation between desktop versions of Linux and the Raspberry Pi. Even though there are advantages to cross platform compilation, a

user’s Raspberry Pi will not need to have the Qt library installed to use the application. The Qt framework supports static linking, meaning that library dependencies are resolved at link time.

4.3 Source APIs

Nearly each of the widgets will require the use of a source API for querying purposes; our development team anticipates querying for weather data, news article metadata, times in different cities around the world, trivia questions, motivational quotes, and upcoming holidays. The development team will need to make efforts to minimize reliance on API keys when choosing suitable source APIs. Source APIs that do not use an API key are preferred because it means that users will not need to make an account that is associated with the key for the service. Additionally, avoiding API keys also implies less configuration needed for the PiClock application.

4.4 Data Storage

The PiClock software does not require a complex relational database, nor should it implement an in memory database like a SQLite database for this project. The application does not require persisting large amounts of data. PiClock will need to store data about the application’s configuration settings for each feature. The only other data that PiClock needs to persist is for the alarms. A requirement of the user interface is to limit the user to 25 alarms. 25 alarms is not a large enough number to warrant the implementation and use of a database when other simpler options can perform with negligible differences in read and write times. Because of this, PiClock will use two simple JSON configuration files to persist data for the application’s settings and the list of alarms.

5. External Interface Requirements

5.1 Itemization / Required Parts List

Item	Price	Description
Raspberry Pi 3	$34.99	A single-board computer with wireless LAN and Bluetooth connectivity
Raspberry Pi Touchscreen	$71.00	The official 7” touchscreen display made by Raspberry Pi
Piezo Buzzer	$5.00	A simple and miniature speaker to generate sound for the device
Screen Casing	$18.00	The official screen casing for the 7” touchscreen display and Raspberry Pi

5.2 Purpose

Raspberry Pi 3 Model B: The Raspberry Pi 3 Model B single-board computer allows PiClock a sufficiently powerful platform to operate on. The RPi will be able to handle the strain of running PiClock’s multiple processes in parallel. Its small form factor, low noise production, power consumption efficiency, and customizability make it a perfect choice for our product.

Official Raspberry Pi 7” Touchscreen: The Raspberry Pi 7” touchscreen allows a user to interact with the applications and widgets that will be on the Raspberry Pi 3 computer and will allow for the applications and widgets to be displayed.

Piezo Buzzer: The Piezo Buzzer will generate the sound for this device’s alarm. The buzzer will need to generate a disturbing and loud enough sound to wake the user. Standard C++ libraries will be used to implement the feature along with WiringPi so the application can easily interface with the GPIO pins on the Raspberry Pi board. The GPIO pins will be used to send a sine wave to the Piezo buzzer that will act as the alarm sound.

Screen Casing: The screen and Raspberry Pi chassis should protect the device as well as offer a clean and comforting aesthetic.

6. Performance Requirements

6.1 Boot and Initialization

- The main layout will initialize within 2 seconds of initial application load, while remaining home screen slides will initialize within 5 seconds of initial application load. Desired load times are independent of Raspberry Pi boot times. Timing requirements will begin after the Raspberry Pi has booted
- Initialization errors should not occur unless tampering to initialization files has occurred. If tampering to initialization has occurred, users will be notified that initialization files have been corrupted

6.2 Feature Slideshow

- All widget slides will need to be performant enough to update between each full slide cycle. Data will need to be acquired efficiently with low latency APIs
- There should not be lag during slide transition animations

6.3 Games

- Games respond within 0.1 seconds to user input from touch screen
- Game services correctly recognize success or failure with 100% accuracy. Users should not experience incorrect results to game outcomes
- When a game has been won, immediately notify the user of success. Likewise, when a game has been lost, immediately notify the user of failure

6.4 Clock and Alarm

- The clock label should update within 0.1 seconds of accuracy to the Raspberry Pi’s internal system clock

- If alarm is set by user, it must trigger within 1 second of the set time with 100% reliability
- If a game is successfully completed, then the alarm shall shut off within 1 second afterward with 100% reliability

7. Security Requirements

Source APIs will likely require API keys associated with third-party accounts. The user will be required to create these accounts themselves and register them with PiClock in the settings page. The user will be required to save their registered API key to the Raspberry Pi if they wish to enable certain PiClock features.

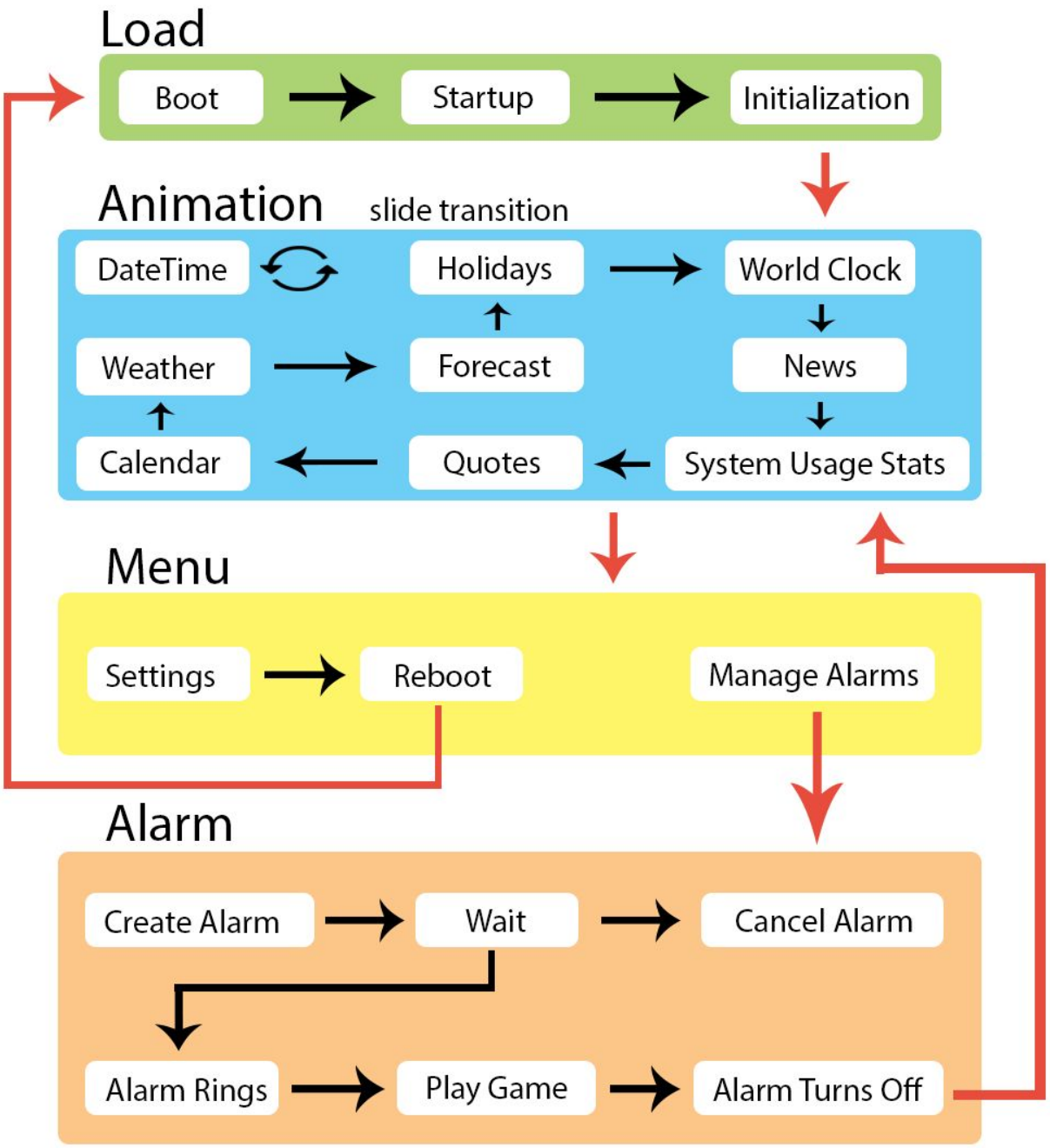
PiClock should attempt to avoid the use of API keys for the sake of feature setup commitments required by end users. Each API key that users are required to retrieve will not be considered extremely sensitive user information. However, PiClock will still take precautions and hide API keys in configuration files only accessible to the application. It is expected of the PiClock software to take precautions with any user data, however this scenario is a classical security problem with no perfect solution.

The general rule of thumb under these circumstances is tricky. The only credentials developers should store on a user's machine are credentials associated with that user, e.g., credentials that enable that user to log into their accounts. You have to assume that anything stored on the user's machine is known by the user, or can easily be learned by the user. The other rule is developers should not hardcode credentials into the program. If a developer must store credentials on the user's machine, store them in some private location. Decent examples would be a configuration file or in a directory, preferably one that is only readable by the application or the user.

PiClock will need to hide user credentials in a configuration file. The file will be hidden somewhere out of prying eyes. Only the application will have read and write access to the file contents. Users will be able to add, change, or remove their credentials through the application’s settings page.

8. Finite State Transition Diagram

This diagram shows the states of the PiClock during its typical operation, flowing from startup to the home screen to settings to the alarm. The diagram depicts the required state transition changes.



9. User Interface Requirements

Below are the design mockups for the alarm clock software. These may vary depending on the user's settings, but show the required designs for the application's main features. All PiClock features should inherit a mobile-minded design pattern to make navigation easy and incentivise use of the touchscreen.

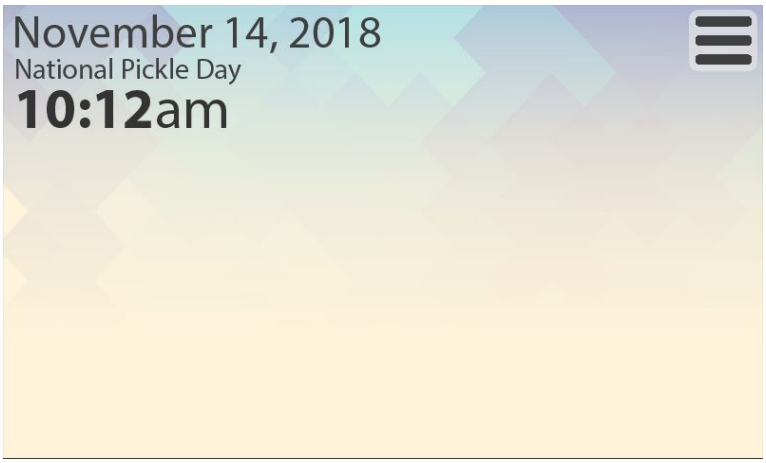


Fig. 9.1: Layout

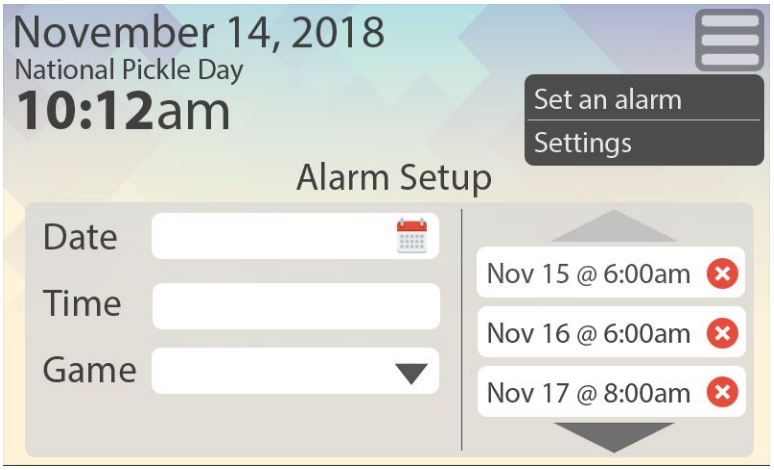


Fig. 9.2: Alarm Setup

Fig. 9.3: Tic-Tac-Toe

Fig. 9.4: Concentration

Fig. 9.5: Simon

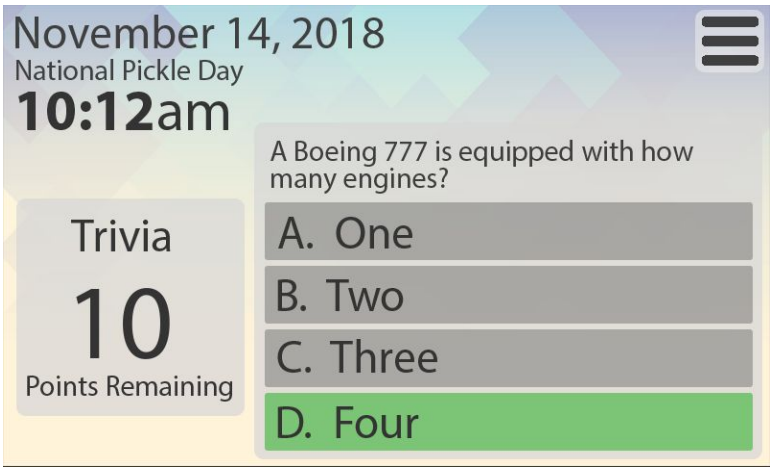


Fig. 9.6: Trivia

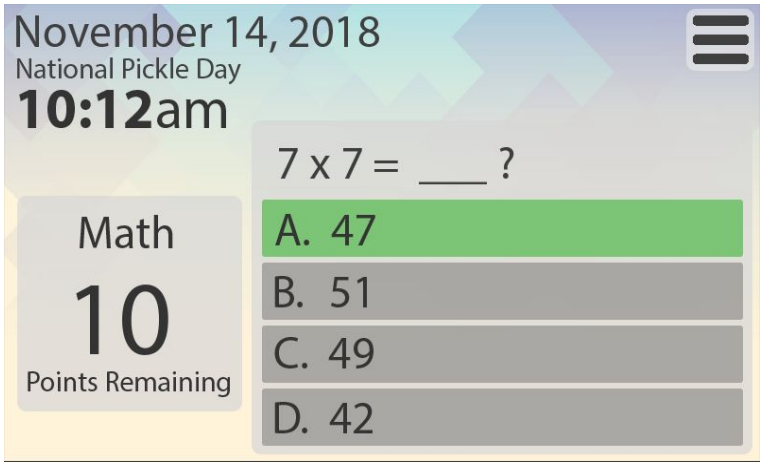


Fig. 9.7: Math Game

Fig. 9.8: Today's Weather

Fig. 9.9: Forecast

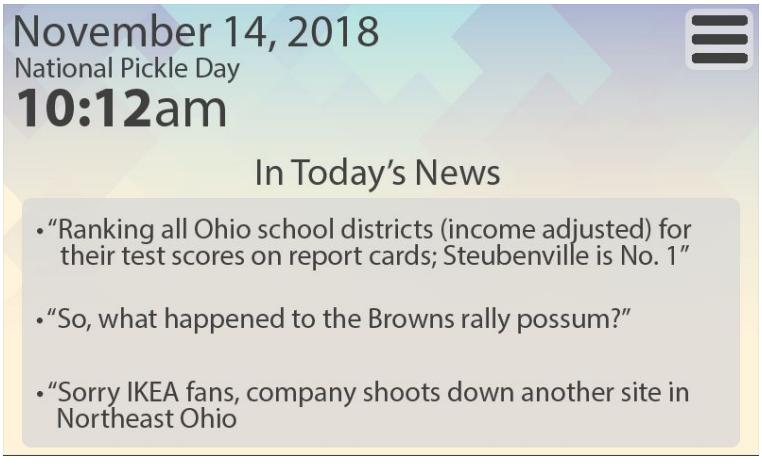


Fig. 9.10: News

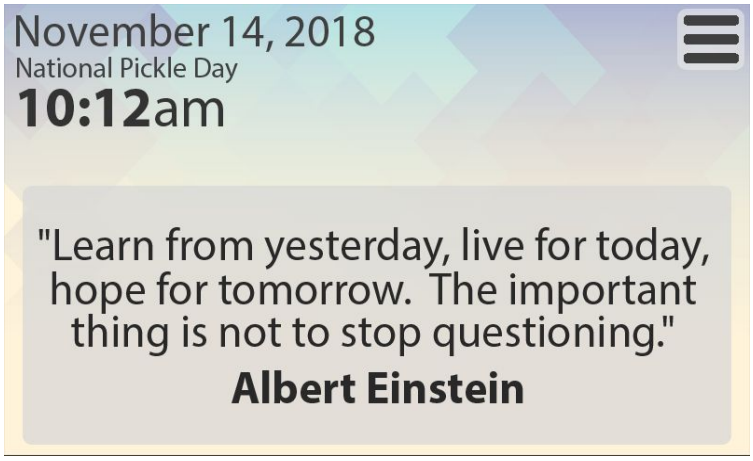


Fig. 9.11: Quotes

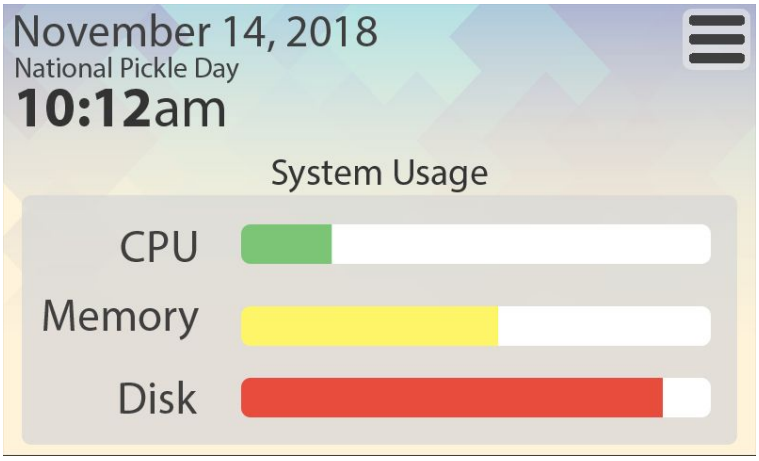


Fig. 9.12: System Usage

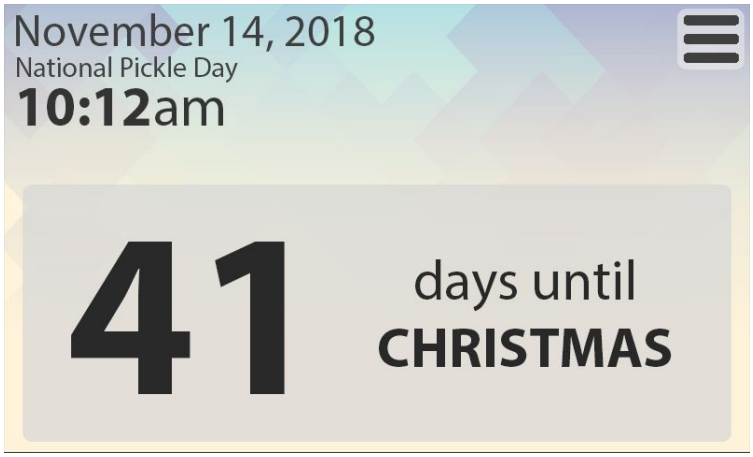


Fig. 9.13: Holidays

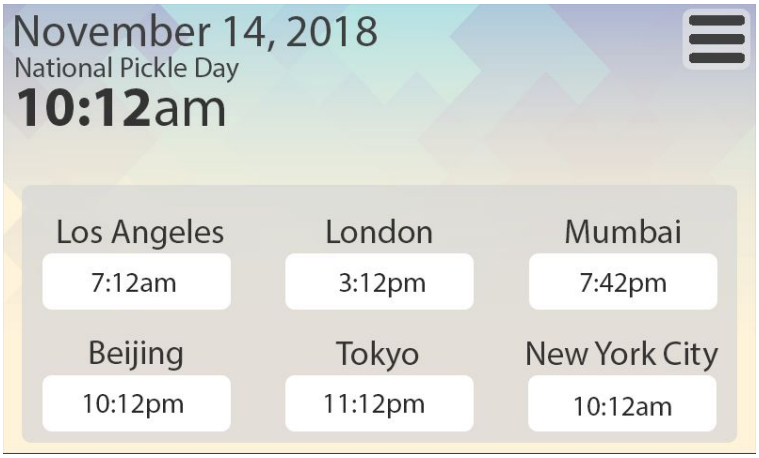


Fig. 9.14: World Clocks

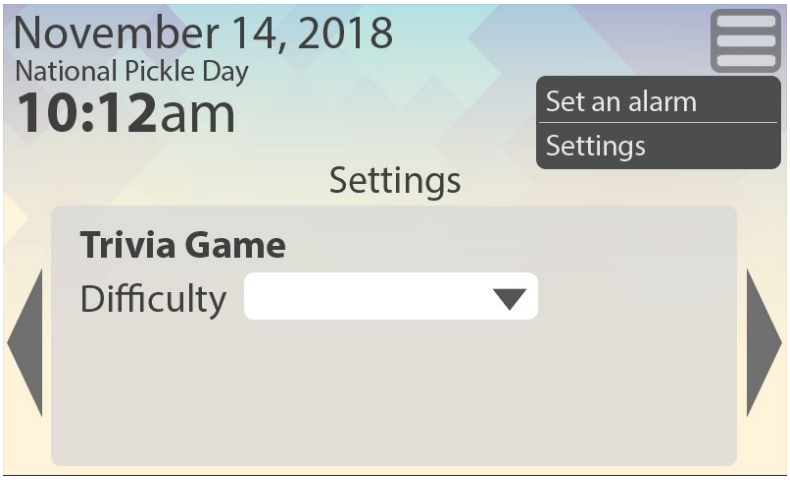


Fig. 9.15: Settings Control

Component	**Requirements**
Layout	• The layout must show on every screen • Colors used in components should always be chosen from a defined color palette • Three labels will appear in the top left corner. The order of labels from top to bottom should be date, holiday, and time. The time label should stand out from other labels • When an alarm is ringing, the clock label's text-color should indicate a waring • The menu button should appear in the top right of the layout, opposite the layout content. When the menu button is pressed, a dialog box should open with options to manage alarms or to access PiClock settings • The widget slideshow will be displayed below the main layout components. Each slide and its contents will slide right to left across the screen (via animation) after a 10 second period
App Settings	• Settings will be grouped by the features they affect • The settings page can be navigated by pressing buttons on either side of the groupings • After leaving the app page, users should be shown a message telling them that their Raspberry Pi will need to reboot
Alarm Management	• The alarm management page will be seperated into the creation properties on the left and set alarms on the right • Date, time, and game inputs are required to set an alarm • The set alarm in the list will have a button to cancel the alarm. These alarms should be ordered by next date and time ascending (next to occur) • To limit memory usage by the user interface, only allow up to 25 alarms to be set. If 25 alarms have been set, disable the section of interface that allows setting alarms
Alarm Games -General	• All games will be separated into two sections. The left section will show general game information like the game's title and user progress. The right section will show labels and buttons pertinent to the game's playing field. • Progress labels should update throughout a game

	• Once a user has won a game or the failsafe triggers, the PiClock software should change to a transition page that allows the user options to “snooze” or “wake up”. Hitting the “snooze” button will turn off the display and “snooze” the alarm. Hitting the “wake up” button will start the widget slideshow animations
Alarm Games -Tic-Tac-Toe	• The Tic-Tac-Toe grid buttons will be displayed in a grid measuring 3 wide and 3 tall • Once a button has been pressed, it should be filled with an X or an O to signify which player “owns” the grid position
Alarm Games -Concentration	• The Concentration grid size will depend on the configured difficulty setting. Therefore, the number of buttons required will also depend on the configured difficulty setting • Each card should have a colorful pastel back and a number on the face. Cards with matching numbers will be paired • Two cards should be selected by a user before they are flipped face-up. Unmatched face-up cards will remain flipped for 2 seconds and then returned face-down. Cards that have been matched should be left face-up
Alarm Games -Simon	• The Simon button grid will measure 2 wide and 2 tall. There will be one button of each color: blue, green, red, yellow • When the computer iterates the sequence, then the matching button in the sequence should flash • Buttons should flash when pressed by the user when entering a sequence
Alarm Games -Trivia	• Questions should be placed above the answer buttons • All correct answers should be shown after an answer submission. The correct answer’s corresponding button button background will change to green • If the wrong answer is submitted, the corresponding buttons background should be shown in red
Alarm Games -Math	• Questions should be placed above the answer buttons • All correct answers should be shown after an answer submission. The correct answer’s corresponding button background will change to green

	• If the wrong answer is submitted, the corresponding buttons background should be shown in red
Widgets -Weather	• All data should stand out (bolded) • The left part of the slide will show a weather graphic that matches current weather conditions • The center of the slide will prominently display the current temperature • All other textual weather data will be displayed on the right part of the slide. The data will be in order of importance: location, predicted temperature high and low, humidity, and wind speed
Widgets -Forecast	• The location data will be displayed at the the top-center of the slide • The forecast will consist of the next 4 days not including the current day. The information that will be shown in each day's forecast box from top to bottom will be shown in order of importance: the date, a weather graphic that matches predicted weather conditions, and the predicted temperature high and low
Widgets -News	• The news widget will display information about 3 popular articles in a bulleted list. Each bullet should contain the article title paired with a brief description
Widgets -Quotes	• The quotes widget will display a single quote and author. The quotes should be placed in quotations above the author • All text should be centered and the author label should be bolded
Widgets -System Usage	• Graphics should be displayed in order of CPU, memory, and disk usage. A percentage bar graph will be used to display usage statistics • Usage graphic color will change at varying percentage levels. In increments of 25% starting at 0% and ending at 100%, graphics should be displayed in green, yellow, orange, and red respectively
Widgets -Holidays	• Countdown will display on the slide's left side in a very large font size • Right side will show what holiday is approaching using a normal font size

Widgets -World clocks	• The 6 clocks will be displayed in a grid 3 wide and 2 tall • The city label for each clock should be displayed above the clock's time label • Clocks should be displayed using the 12 or 24 hour format that is configurable through the application's setting page