Software Requirements Specification (SRS)
VS-Saturn

Team: Team 2
Authors: Logan Foster, David Holton, Owen Hunter, Evan Smith, Taylor Springob
Customer: Visual Studio Code users
Instructor: Dr. James Daly

1. Introduction

This document will outline the requirements for the proposed VS-Saturn extension to the Visual Studio Code (abbr. to VS-Code) editor. The purpose of the project, as well as its scope and target audience, will be discussed in this introductory section. Additionally, a list of definitions, acronyms, and abbreviations used in this document will be provided in this section to assist the reader's understanding of those terms. Other topics to be discussed in this document include VS-Saturn's hardware and software requirements, an explanation of any functions relevant to its implementation, development constraints, and an outline of the VS-Saturn prototype.

1.1. Purpose

The purpose of this document is to provide explicit requirements for the VS-Saturn project. Specification of these requirements aids the development team's understanding of the project and allows for effective strategizing of the development process. This document also serves as a manual for the development team. It breaks VS-Saturn down into its individual components and specifies the purpose of each of them. This helps to guide the development team regarding the scope and technical specifications of the project. Clients and stakeholders can also benefit from reading this document as it conveys the development team's understanding of the initial requirements put forth.

The target audience of this document is primarily the development team and aims to provide a formal definition of the requirements of VS-Saturn which will assist the development team in reaching a common understanding of development needs. Another goal of this document is to properly inform clients and stakeholders of the design specifications of VS-Saturn. To achieve this end, non-technical descriptions of the requirements and example scenarios are provided in addition to the formal definitions. This document will additionally offer clients and stakeholders some insight into the development process. Lastly, readers will receive an explanation of the strategies employed by the development team in their implementation of VS-Saturn.

1.2. Scope

VS-Saturn is designed as an extension of VS-Code and is thus contained entirely within the VS-Code application. The goal of VS-Saturn is to aid users with time and task management. VS-Saturn project incorporates the Pomodoro technique of time management; traditionally, this technique involves structured cycles of four 25-minute work periods. Each of the first 3 work periods are punctuated with 5-minute breaks and the last work period culminates in a 25-minute break. The Pomodoro technique is meant to ease mental fatigue during long workdays. By using VS-Saturn, users can work on tasks in short bursts of productivity via the Pomodoro method.

If a user prefers not working in VS-Code, it would be less practical to use this product for time management. VS-Saturn can still serve as an effective Pomodoro tracker if used while doing other tasks but it's most useful for programmers who already use VS-Code. Other methods exist for using the Pomodoro technique, but VS-Saturn offers unique features that give it an advantage over other similar types of software. VS-Saturn will keep a timer that is visible to users; when each work or break period has ended, VS-Saturn will alert users so that they can stay on schedule. This will be a non-intrusive form of notification to avoid causing unnecessary disruption to users. Users can additionally create a customizable tasklist in VS-Saturn so that they can set goals and monitor their progress. Lastly, VS-Saturn offers a snooze functionality for its timer so that users have the option of delaying the end of a work or break period.

1.3. Definitions, acronyms, and abbreviations

- **API:** Application programming interface; defines the interactions between extensions and VS-Code.
- **Extension:** An application hosted on VS-Code that offers added functionality [5].
- **Pomodoro:** Invented in the late 1980s by Francesco Cirillo [1], the Pomodoro technique helps solve time management issues by cycling through two states. The first state is a 25-minute work period, while the second is a 5-minute break [2].
- **Users:** Individuals using Pomodoro through VS-Saturn on VS-Code.
- **VS-Code:** Abbreviation for Visual Studio Code, a text editor commonly used for software programming.

1.4. Organization

The remainder of this document is laid out as follows:

Section 2 – Overall Description: p. 4

This section provides context for the VS-Saturn project and discusses design goals. VS-Saturn's core functionality is outlined in this section, and its constraints defined. The target demographic for VS-Saturn is also characterized. Lastly, this section discusses some requirements that fall outside of the project's scope.

Section 3 – Specific Requirements: p. 7

This section provides an enumerated list of the technical requirement specifications for VS-Saturn.

Section 4 – Modeling Requirements: p. 9

Possible use cases for VS-Saturn are compiled in this section. Diagrams have been provided as visual representations of these use cases. By using these accompanying visual aids, this section attempts to further showcase the goals of the VS-Saturn project.

Section 5 – Prototype: p. 26

This section describes the functionality of the VS-Saturn prototype. Sample scenarios of its use are discussed alongside samples of VS-Saturn's UI. Instructions on how to run the VS-Saturn prototype are provided in this section. Any prerequisites for running the prototype are also mentioned. Lastly, a URL to install the prototype has been provided.

Section 6 – Resources: p. 31

This section contains a list of any references made in this document as well as a link to the project's website.

Section 7 – Point of Contact: p. 32

This section contains contact information for any questions about the product described in this document.

2. Overall Description

This section provides context for the VS-Saturn project and discusses design goals. VS-Saturn's core functionality is outlined in this section, and its constraints defined. The target demographic for VS-Saturn is also characterized. Lastly, this section discusses some requirements that fall outside of the project's scope.

2.1. Product Perspective

For many software engineers, it can be difficult to maintain focused effort on work over long periods of time. VS-Saturn aims to solve this by implementing the Pomodoro technique in VS-Code. VS-Saturn sets out to implement this method by inserting a customizable timer alongside a to-do list into the VS-Code interface. Using the extension, users will set a work interval and a subsequent break interval. This allows the user to set intervals that fit best with their workflow. When the timer completes, the user is notified and can choose to take a break or to snooze the timer for a few additional minutes. Because VS-Saturn is an extension running within VS-Code, it is targeted towards existing software development environments. This means VS-Saturn development was constrained to the pre-existing VS-Code interface. VS-Code is a required installation to run VS-Saturn; accordingly, the hardware should be capable of running VS-Code and the user should already be familiar with the basics of the editor environment. Additionally, VS-Saturn uses the VS-Code extension API which utilizes the Electron.js framework, a collection of Chromium, Node.js, and JavaScript [6]. Because TypeScript compiles to common JavaScript and is well-favored for VS-Code extension development, it was chosen as the development language. As seen in figure 2.1-1 below, the VS-Saturn extension runs on a separate Node.js node, communicating to the main VS-Code render process through the extension host [4].

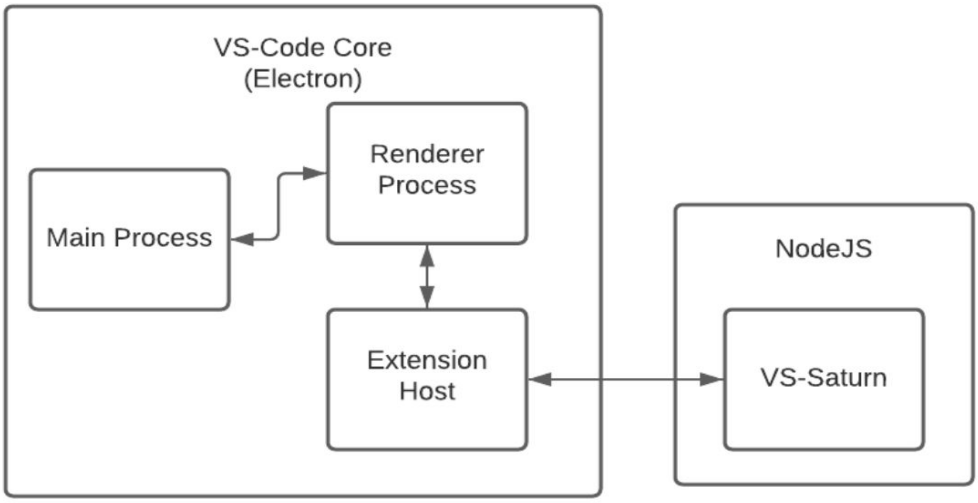


Figure 2.1-1: VS-Saturn Block Diagram

2.2. Product Functions

The major functions of VS-Saturn are as follows:

- Provide a built-in timer to keep track of work and break time.
- Provide task checklists (tasklist) so the user can track work progress, set goals, and add break items to a list to minimize distraction during work periods.
- Allow the user to customize the time intervals for both the work, break, and snooze time.
- Provide a non-intrusive notification when the timer is up by changing the color of VS-Code's non-editor UI elements.
- Allow the user to snooze the work alarm if additional time is needed.

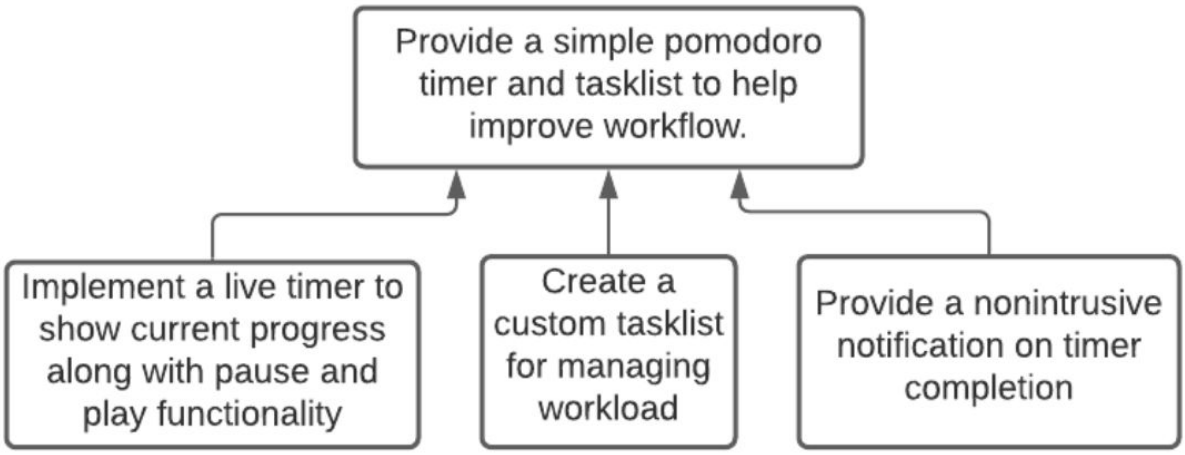


Figure 2.2-1: VS-Saturn high-level goals

2.3. User Characteristics

It is assumed that users have some development experience using VS-Code; the expectation is that users are generally familiar with VS-Code's workflow. Users must be able to install VS-Saturn from the VS-Code Marketplace found online on the Visual Studio website. In order to fully utilize VS-Saturn, users are expected to complete most of their work within the VS-Code application window so they may see notifications.

2.4. Constraints

VS-Saturn is constrained primarily by the limitations of VS-Code. Every platform that VS-Code targets supports VS-Saturn. The VS-Saturn user interface is implemented with the VS-Code API. This gives the user a better experience and fluidity when operating the extension, but limits interface elements to what VS-Code provides to the developers.

Data stored by VS-Saturn through the tasklist feature is stored in a SQLite database provided by the VS-Code API inside the users VS-Code application data folder.

2.5. Assumptions and Dependencies

VS-Saturn software and hardware dependencies need to fulfill VS-Code's requirements. The recommended hardware for VS-Code is a 1.6 GHz or faster processor, at least 1 GB of RAM, and at least 200 MB of free disk space.

The oldest platforms VS-Code targets are OS X Yosemite, Windows 7, Ubuntu Desktop 14.04, Debian 7, Red Hat Enterprise Linux 7, CentOS 7, Fedora 23. Versions after each of those platforms are supported. On Windows platforms, users will also need Microsoft .NET Framework 4.5.2. On Linux platforms, GLIBCXX 3.4.15 or later and GLIBC 2.15 or later are required.

An internet connection is initially required to download the VS-Saturn extension from the VS-Code Marketplace. Once the download is complete, no further internet connection is needed.

2.6. Apportioning of Requirements

For its initial release, VS-Saturn will support basic time and task management functionality as described in this document. Features that may be considered for inclusion in future iterations include:

- The addition of an API to facilitate integration with other VS-Code extensions.
- The ability to trigger custom actions upon the expiry of work or break timers.
- Summaries of task progress at the end of each cycle (e.g. percentage of tasks completed, time spent on each task, etc.).
- The ability to prioritize and reorganize tasklists.
- The option to automatically show the tasklist after a Git commit is made.
- Support for optional push-notifications to the user's phone to alert them to the end of a break period (when they may be away from their computer).

3. Specific Requirements

1. The extension must operate in cycles consisting of a series of alternating, timed “work” and short “break” periods ending with one longer “break” period.
    1.1. The extension must allow the user to configure the duration of the work and short break periods.
        1.1.1. By default, work periods should last 25 minutes and short break periods should last 5 minutes.
    1.2. After a user-configurable number of work periods have elapsed, the extension should begin a long break period instead of a short one.
        1.2.1. Long break periods should also be user-configurable.
            1.2.1.1. By default, long break periods should be 25 minutes.
        1.2.2. There should be, at all times, a visible indication of the user’s progress towards a long break period in terms of work periods completed vs work periods remaining in the current cycle.
    1.3. The extension must, at all times, display the time remaining in the current period (work or break).
    1.4. At the end of a work period, the extension must notify the user in a nonintrusive way.
        1.4.1. The extension must not use sound to notify the user that a work period is ending nor should it cause the editor window to lose focus (e.g. with a popup window).
    1.5. At the end of a break period, the extension must notify the user.
        1.5.1. Break period-end notifications may include sound.

2. The extension must support the creation and subsequent modification of two distinct tasklists, one for work periods and one for break periods.
    2.1. The tasklists should be presented to the user in the form of editable checklists.
    2.2. During work periods, the user should be able to add and remove (check-off) tasks from the work tasklist and add tasks to the break tasklist.
        2.2.1. The break tasklist should never be visible during work periods.
    2.3. During break periods, the user should be able to add and remove (check-off) tasks from the break tasklist.
        2.3.1. The work tasklist should never be visible nor alterable during break periods.
    2.4. Neither of the tasklists need to be saved when VS-Code is quit.

3. The extension must provide the user with ways to interrupt the current period.
    3.1. At the end of a work period, the user should be notified as such and given the option to delay ("snooze") the following break period.
        3.1.1. The snooze duration should be configurable in the extension's settings.
    3.2. At any time, in either work or break periods, the user must have the ability to pause (and subsequently resume) the period's timer.
        3.2.1. The extension should define a keyboard shortcut for the pause/resume operation.
    3.3. At any time, the user should have the option to "start over" by resetting their cycle progress back to the beginning.
        3.3.1. When this option is used, the extension should prompt the user for confirmation before proceeding (i.e. it should take more than 1 click to reset the cycle).
        3.3.2. When resetting the cycle, the extension should also ask the user whether they'd like to keep their current tasklists.

4. The extension must provide an interface for users to change its settings.
    4.1. Users should be able to choose whether or not timers continue counting down when VS-Code is minimized.
    4.2. At the end of a work or break period, users should be able to choose whether confirmation is required before moving on to the next period or whether that will happen automatically.
    4.3. All user-configurable options mentioned in requirements 1-3 should also be found in this interface.
    4.4. The extension must save the user's settings such that they are automatically restored after VS-Code is quit & reopened.

4. Modeling Requirements

Use Cases:

VS-Saturn is an extension that aims to improve the user's productivity through use of the Pomodoro technique. As such, the use case diagram consists of the major aspects that the user will be interacting with directly, as well as their subtasks. The use case diagram for VS-Saturn and its definitions follow:

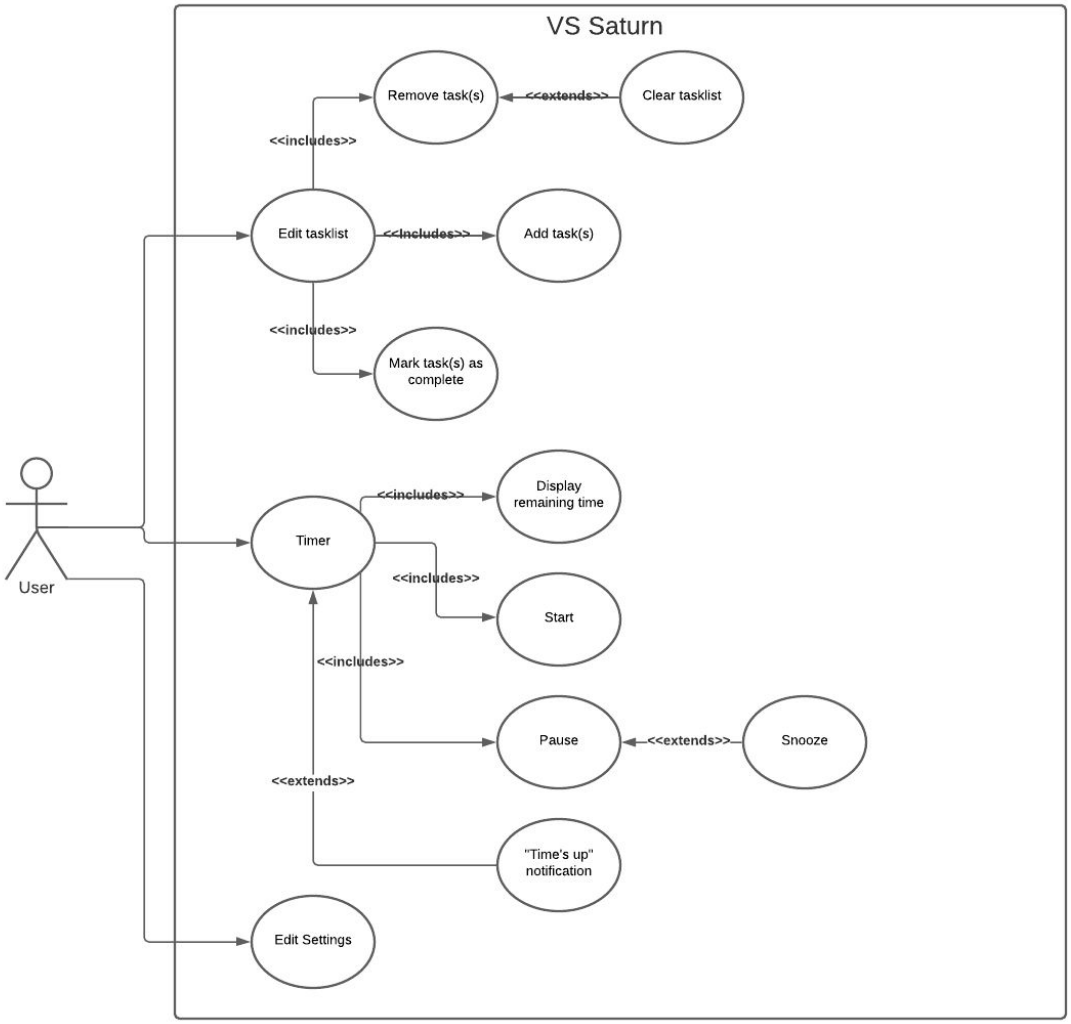


Figure 4-1: Use Case Diagram

Use Case Name:	**Edit Tasklist**
Actors:	User(initiator)
Description:	The user is given a tasklist as part of VS Saturn to which they can edit.
Type:	Primary, Essential
Includes:	Remove task(s), Add task(s), Mark task(s) as complete
Extends:	N/A
Cross-refs:	Requirement 2
Uses cases:	User clicks the “tasklist” icon from the taskbar.

Use Case Name:	**Add Task(s)**
Actors:	User(initiator)
Description:	When editing the tasklist the user can add additional tasks to the list.
Type:	Secondary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 2
Uses cases:	User clicks the “add a task” icon from the taskbar

Use Case Name:	**Remove Task(s)**
Actors:	User(initiator)
Description:	When the user wants to delete a task completely from their list.
Type:	Secondary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirements 2.2, 2.3
Uses cases:	When editing the tasklist the user can choose to delete tasks they no longer need.

Use Case Name:	**Clear Tasklist**
Actors:	User(initiator)
Description:	If the user wants to completely clear the tasklist they have the option to.
Type:	Secondary
Includes:	N/A
Extends:	Remove Task(s)
Cross-refs:	Requirement 2
Uses cases:	From the command palette (control+shift+p) the user can clear the tasklist.

Use Case Name:	**Mark Task(s) as complete**
Actors:	User(initiator)
Description:	If a task has been completed, it can be marked as such so the user can keep track of what they have finished so far.
Type:	Secondary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirements 2.2, 2.3
Uses cases:	Option when editing the tasklist

Use Case Name:	**Timer**
Actors:	User(Initiator)
Description:	The timer is the main feature of VS Saturn to keep track of the time in the current cycle.
Type:	Primary, Essential
Includes:	Display remaining time, Start, Stop, Pause
Extends:	N/A
Cross-refs:	Requirements 3.2, 3.2.1
Uses cases:	Represented in the VS Saturn taskbar as a countdown timer.

Use Case Name:	**Start**
Actors:	User(initiator)
Description:	The user starts the timer countdown
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 3.2, 3.2.1
Uses cases:	By clicking the “play” button from the Saturn taskbar the timer will start counting down.

Use Case Name:	**Pause**
Actors:	User(Initiator)
Description:	If the user needs to step away for just a moment, they can pause the timer.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 3.2, 3.2.1
Uses cases:	User clicks the “pause” button from the taskbar to stop the timer.

Use Case Name:	**Snooze**
Actors:	User(initiator)
Description:	When a work cycle ends and the user is notified, they have the option to snooze their break, extending their work cycle by a predefined amount of time (default: 5 minutes)
Type:	Secondary
Includes:	N/A
Extends:	Pause
Cross-refs:	Requirement 3.1
Uses cases:	The user clicks the “snooze” prompt on the “Time’s Up” notification.

Use Case Name:	**Display remaining time**
Actors:	N/A
Description:	Display the current time remaining in a cycle to the user.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 1.3
Uses cases:	Will be displayed on the taskbar.

Use Case Name:	**“Time’s Up” notification**
Actors:	N/A
Description:	When the timer runs out a small dialog box will display, notifying the user that the current cycle has ended.
Type:	Primary
Includes:	N/A
Extends:	Timer
Cross-refs:	Requirements 1.4, 1.4.1, 1.5, 1.5.1
Uses cases:	Will always occur once the timer reaches “0:00” for any given period.

Use Case Name:	**Edit Settings**
Actors:	User(initiator)
Description:	The user can change various settings such as the work cycle length, short and long break cycle lengths, and the amount of cycles in a pomodoro.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirements 4, 4.1, 4.2, 4.3, 4.4
Uses cases:	The user can edit these in the user settings under “extensions” for saturn in VS Code.

Class Diagrams:

The following diagram represents the main classes that are essential to the operation of VS-Saturn, with a data dictionary to follow.

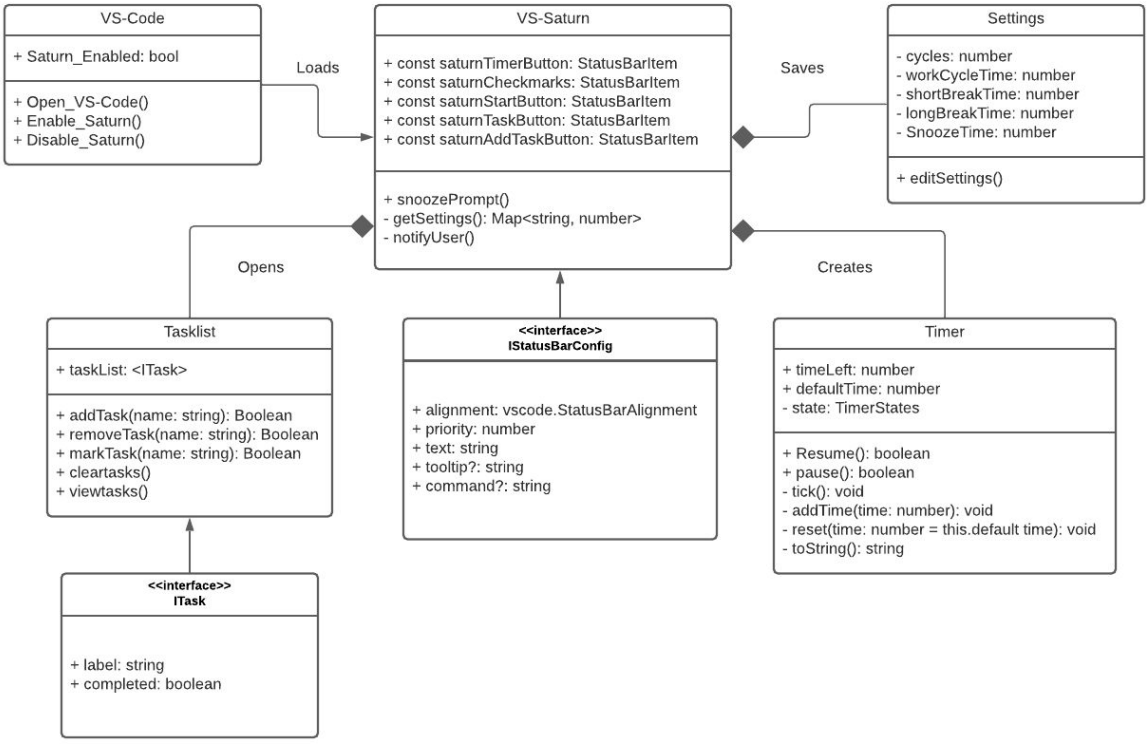


Figure 4-2: Class Diagram

Element Name		**Description**
VS-Code		VS-Code is the editor of choice that the extension will be available for, and where it will load from.
Attributes		
public	Saturn_Enabled: Bool	Simple Boolean to check whether the VS-Saturn extension is enabled or not.
Operations		
public	Open_VS-Code()	Opening the VS-Code application.
public	Enable_Saturn()	Enable the VS-Saturn extension for use.
public	Disable_Saturn()	Turn off the VS-Saturn extension when it does not need to be used.
Relationships	VS-Code is not a class, but it is the actual application that the extension will work on. It will load the VS-Saturn extension and have all the visual elements of it visible to the user such as the pause/play button, tally marks for the cycle, and the timer itself.	

Element Name		**Description**
VS-Saturn		VS-Saturn is the application itself. It is where the user will actually interact with the timer as well as displaying the cycle and notifying the user and editing their tasklist
Attributes		
public	const saturnTimerButton: StatusBarItem	An item on the lower taskbar which displays the time to the user.
public	const saturnCheckmarks: StatusBarItem	An item on the lower taskbar which displays the checkmarks for completed cycles to the user.
public	const saturnStartButton: StatusBarItem	An item on the lower taskbar which displays the Start button for the timer to

		the user (switches to pause button when the timer is running).
public	const saturnTaskButton: StatusBarItem	An item on the lower taskbar which will bring up the tasklist when clicked.
private	const saturnAddTaskButton: StatusBarItem	An item on the lower taskbar which allows the user to add an item to the tasklist when clicked.
Operations		
public	snoozePrompt()	Prompts the user if they would like to snooze the timer for additional work time at the end of a cycle.
private	getSettings(): Map<string, number>	Saturn gets the settings the user has saved and maps them accordingly.
private	notifyUser()	Notify the user when a cycle is over.
Relationships	VS-Saturn is the extension itself. It functions as a controller with the necessary icons for the user to interact with such as a pause/play button for the timer, displaying the checkmarks for completed cycles, and displaying/adding to the tasklist.	

Element Name		**Description**
Timer		The Timer class will keep track of the time counting down and display it.
Attributes		
public	timeLeft: number	stores a number value to keep track of time.
private	defaultTime: number	Stores the default time value to be used when resetting the timer.
private	state: TimerStates	keeps track of the current state the timer is in (running, paused).
Operations		

private	Resume(): boolean	Resumes the timer, returns false if already running, true otherwise.
public	pause(): boolean	Pauses the timer, returns false if already paused, true otherwise.
private	tick(): void	Called on every interval of the timer in order to subtract time from it to progress.
private	addTime(time: number): void	Adds time to the timer. Adds it in terms of seconds as an integer value.
private	reset(time: number = this.default time): void	Resets the timer to default and then pauses it.
private	toString(): string	converts the timer into a string in the format of MM:SS, or HH::MM::SS.
Relationships	The timer class is where VS Saturn keeps track of the time for pomodoro. With it the time itself is progressed and ticked down, as well as pausing/starting it and formatting it to be presented to the user.	

Element Name		**Description**
Tasklist		The tasklist is where the user can write down and keep track of different tasks and check them off when done.
Attributes		
public	taskList<ITask>	The tasklist the user views which holds all their tasks.
Operations		
public	addTask(name: string): boolean	Adds a new task to the tasklist. Takes a string for the new task name and returns true if the task was added.
public	removeTask(name: string): boolean	Removes a task from the list. Takes a string for the existing task to be deleted and returns true if the task was found and removed.

public	markTask(name: string): boolean	Marks a task as completed in the tasklist. If it is already completed then the check is removed. Takes the name of the task to be marked as complete and returns true if the task was found and marked.
public	cleartasks()	Clears the entire tasklist.
public	viewtasks()	Displays the tasklist to the user.
Relationships	The tasklist is a part of VS-Saturn and will function as a dropdown list that the user can interact with in order to keep track of things they need to do, and mark them as complete once done, or remove them entirely.	

Element Name		**Description**
Settings		The settings will be where the user can go to change various things about Saturn to customize it to their likings.
Attributes		
private	cycles: number	The number of cycles the user wants for their pomodoro (Default: 4).
private	workCycleTime: number	The amount of time for a work cycle (Default: 25 minutes).
private	shortBreakTime: number	The amount of time for a short break (Default: 5 minutes).
private	longBreakTime: Number	The amount of time for a long rest (Default: 10 minutes).
private	snoozeTime: number	The amount of time to snooze the timer for (Default: 5 minutes).
Operations		
public	Edit_Settings()	Will allow the user to change the various settings of the application.
Relationships	Settings is where the user can customize various parts of VS-Saturn to fit their needs. They can do things such as change the amount of cycles in a	

	Pomodoro, and the length of different timers. These values will all be stored in private variables that the user will be able to change via Edit_Settings().

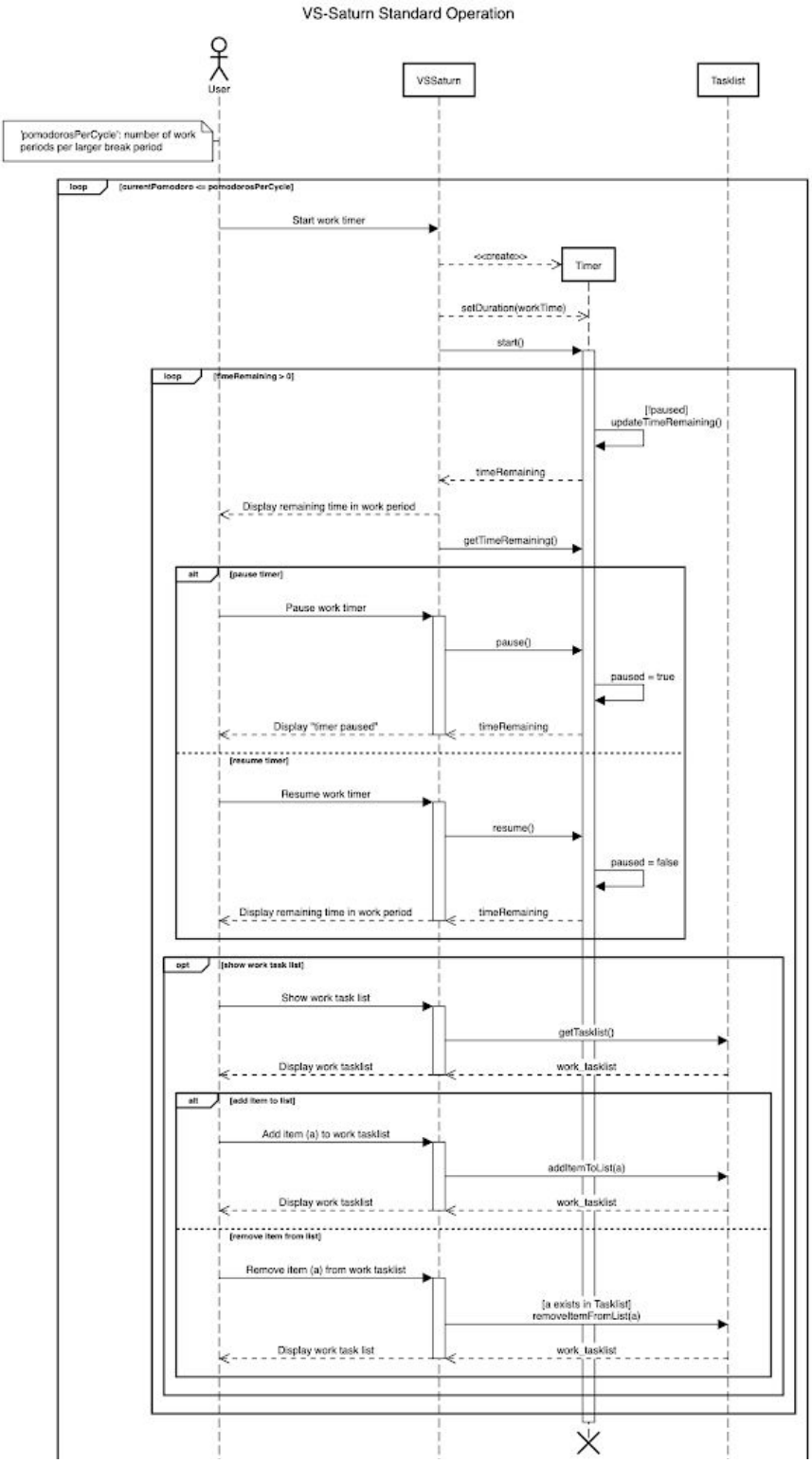
Element Name		**Description**
IStatusBarConfig		An interface that keeps track of various elements for status bar icons.
Attributes		
public	alignment: vscode.StatusBarAlignment	Where/how the icon will be aligned on the status bar.
public	priority: number	the priority of it to help determine placement.
public	text: string	The text to be displayed .
public	tooltip?: string	Tooltip for the item (might not have one).
public	command?: string	The command for the icon when interacted with (might not have one).
Relationships	IStatusBarConfig is a simple interface that VS Saturn uses in order to track and control the status bar items the user will interact with such as the timer, and the pause/play button.	

Element Name		**Description**
ITask		An interface to keep track of tasks.
Attributes		
public	label: string	What the actual task is. This label is what the user sees/enters into the tasklist such as “Finish my math homework”.
public	completed: boolean	Whether the task has been marked as completed or not.

Relationships	ITask is an interface which the tasklist uses. The tasklist is comprised of many Itasks which each have the name of the task, as well as whether it has been completed or not.

Sequence Diagrams:

The following diagrams represent a few different scenarios that a user might encounter while using VS-Saturn with a short description following each to describe them.


VS-Saturn Standard Operation
User
VSSaturn
Tasklist
'pomodorosPerCycle': number of work periods per larger break period
loop
[currentPomodoro <= pomodorosPerCycle]
Start work timer
<<create>>
Timer
setDuration(workTime)
start()
loop
[timeRemaining > 0]
['paused]
updateTimeRemaining()
timeRemaining
Display remaining time in work period
getTimeRemaining()
alt
[pause timer]
Pause work timer
pause()
paused = true
Display "timer paused"
timeRemaining
[resume timer]
Resume work timer
resume()
paused = false
Display remaining time in work period
timeRemaining
opt
[show work task list]
Show work task list
getTasklist()
Display work tasklist
work_tasklist
alt
[add item to list]
Add item (a) to work tasklist
addItemToList(a)
Display work tasklist
work_tasklist
[remove item from list]
Remove item (a) from work tasklist
[a exists in Tasklist]
removeItemFromList(a)
Display work task list
work_tasklist

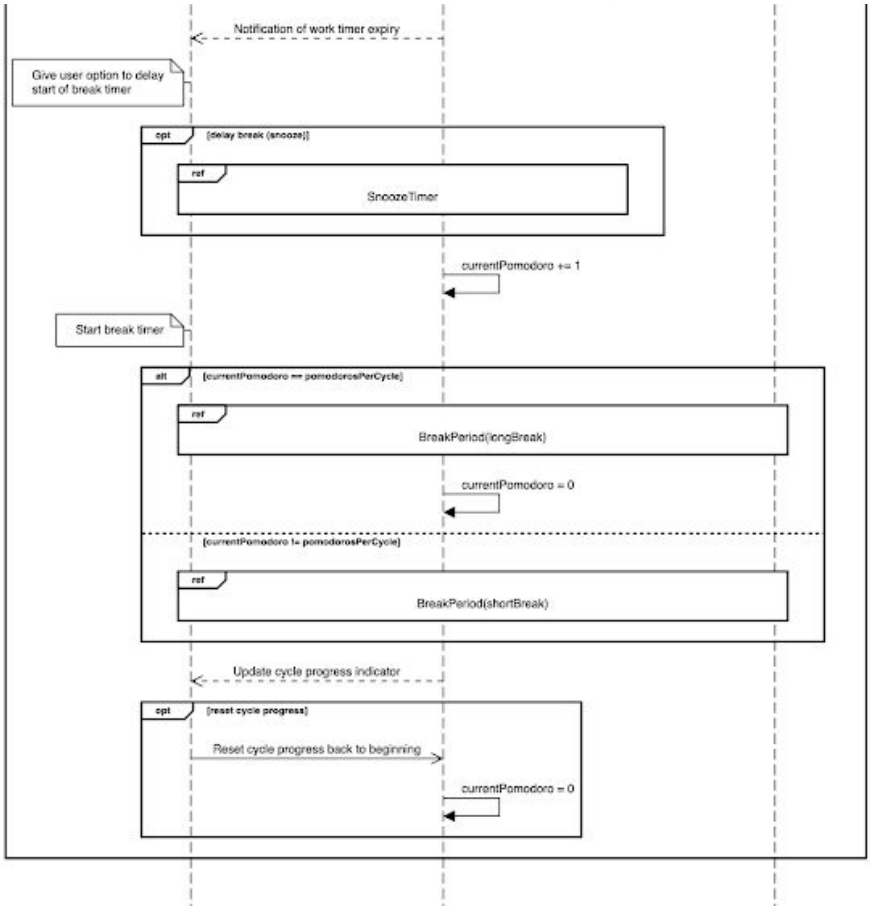


Figure 4-3: Sequence Diagram: Standard Operation

The first diagram outlines the standard sequence that VS-Saturn would go through during normal operation. There are objects for the timer and the tasklist which persist throughout the entirety of the program's duration, as well as a controller (VSSaturn) to keep track of the pomodoro cycles. As execution progresses, the timer will count down and loop through the work period with an option for the user to delay (snooze) the timer when it expires. Once the work timer (and any subsequent snooze sessions) reaches "0:00", the Pomodoro cycle counter is incremented by 1 (updating the cycle indicator) and the user enters a break period. This will be a long break if the final Pomodoro has been reached, or a short break otherwise. The program then starts looping over from the beginning.

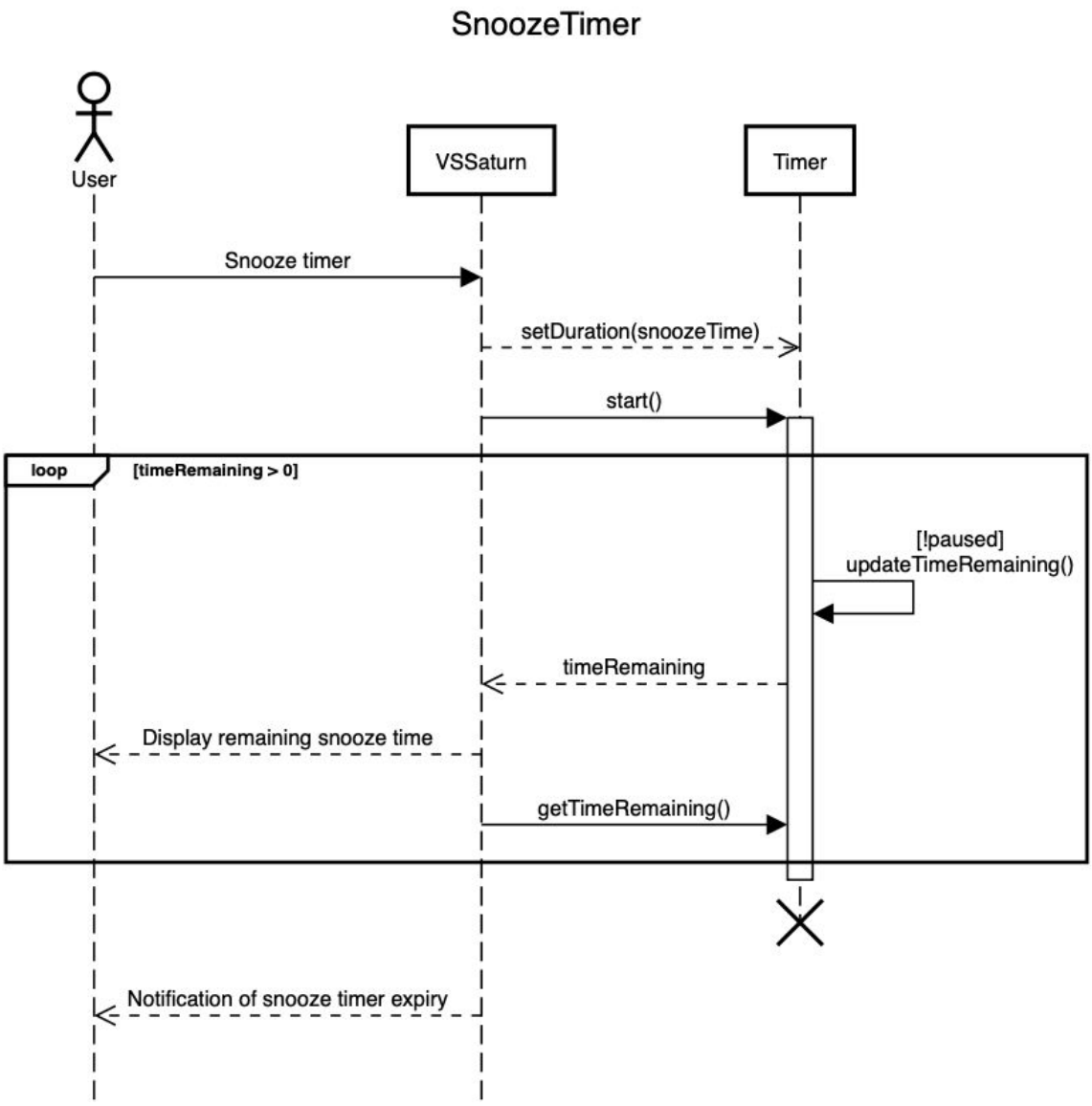


Figure 4-4: Sequence Diagram: snoozeTimer

The second sequence diagram demonstrates the case where the user decides to snooze the timer for a specified amount of time (defined in VS-Saturn's settings). With the duration specified, VS-Saturn enters the loop for the timer's countdown. It will constantly count down in this loop and update the time remaining while displaying this time to the user exactly as it does during both work and break periods. Once the snooze timer expires, the user is notified, and execution continues.

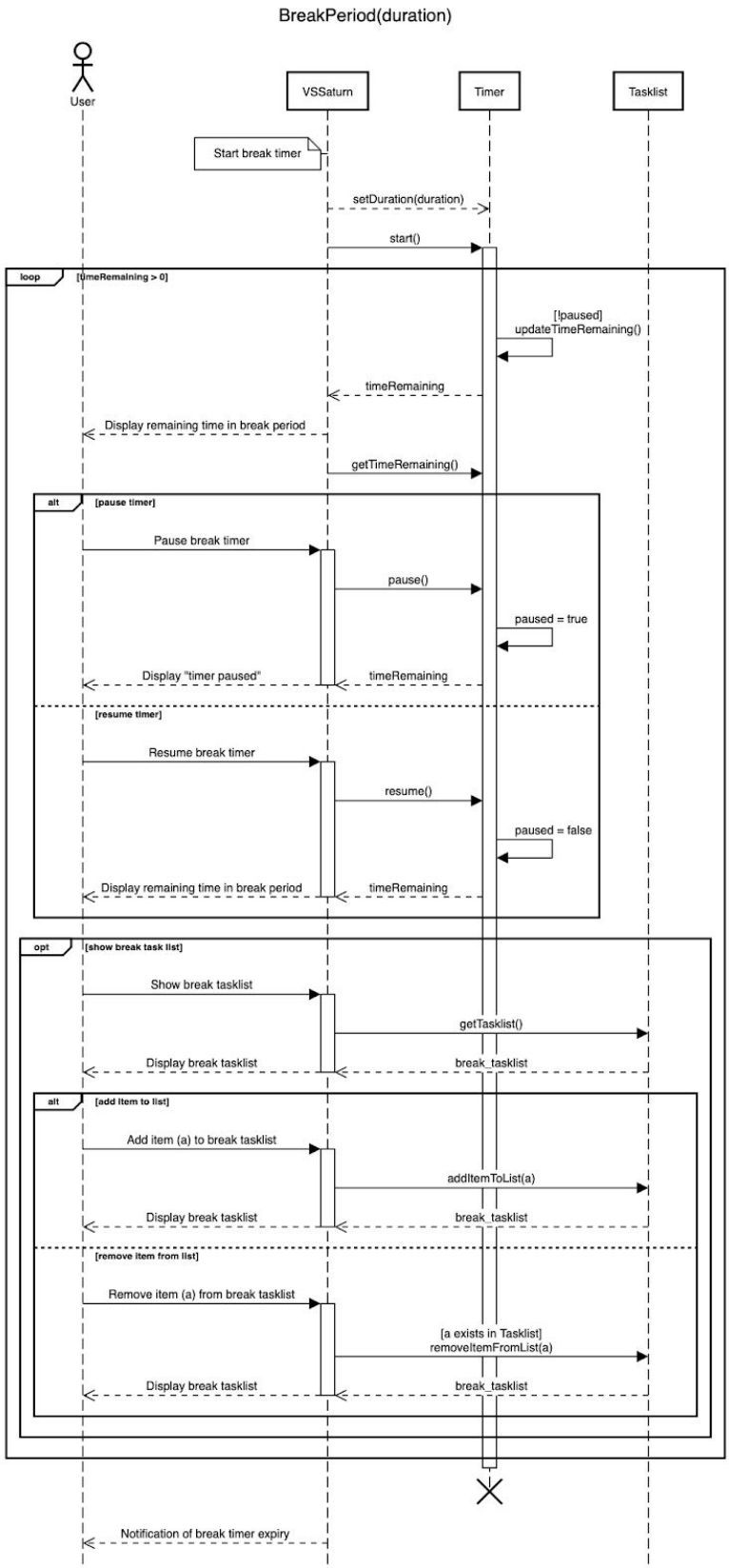

BreakPeriod(duration)
User
VSSaturn
Timer
Tasklist
Start break timer
setDuration(duration)
start()
loop
[timeRemaining > 0]
[!paused]
updateTimeRemaining()
timeRemaining
Display remaining time in break period
getTimeRemaining()
alt
[pause timer]
Pause break timer
pause()
paused = true
Display "timer paused"
timeRemaining
[resume timer]
Resume break timer
resume()
paused = false
Display remaining time in break period
timeRemaining
opt
[show break task list]
Show break tasklist
getTasklist()
Display break tasklist
break_tasklist
alt
[add item to list]
Add item (a) to break tasklist
addItemToList(a)
Display break tasklist
break_tasklist
[remove item from list]
Remove item (a) from break tasklist
[a exists in Tasklist]
removeItemFromList(a)
Display break tasklist
break_tasklist
Notification of break timer expiry

Figure 4-5: Sequence Diagram: breakPeriod

The third and final sequence diagram defines the workings of the BreakPeriod ref from the "Standard Operation" sequence diagram. This sequence follows largely the same logic as the work period loop in the first diagram with the key distinction being the separate break tasklist that is accessible during break periods.

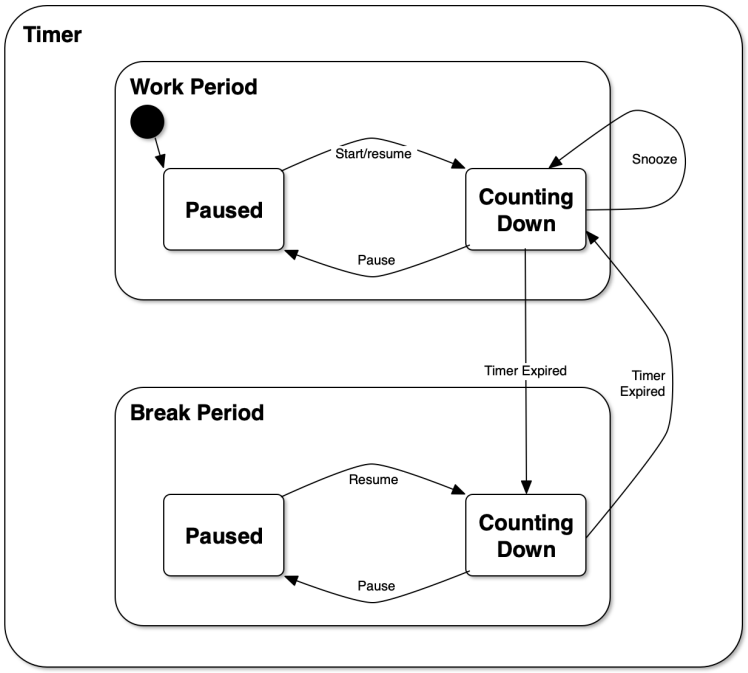


Fig. 4-6: VS-Saturn Timer state diagram

The state diagram of figure 4-6 shows the default operation of Saturn.

5. Prototype

The prototype includes the core interface requirements for the project. Most of the user interface is displayed at the left of the VS-Code status bar found at the bottom of the application window. This includes a display of the remaining time left in the current cycle, checklist of the progress of the Pomodoro technique checkpoints, a “pause/ play” button to control the state of the timer, a button to display the tasklist, and an “add task” button. The tasklist button displays an interface that lets the user view and mark current tasks as completed. The “add task” button prompts the user to input a new task to be added to the tasklist.

Since this is a prototype, the time between each checkpoint is only thirty seconds so the flow of the program can be viewed in a timely manner. When the time is up for the current cycle, a notification in VS-Code alerts the user and asks if they want to snooze.

All the features the buttons add can be reachable from the command palette (Ctrl+Shift+P or Cmd+Shift+P on macOS). Two commands that are found in the command palette but not available as buttons are the Reset Saturn and Clear Tasks commands. This is to remove clutter from the status bar and prevent accidental triggering of these commands.

5.1. How to Run Prototype

To run the prototype, the user’s system must satisfy the software and hardware requirements from Section 2.1; in addition, users will need npm which is installed with Node.js.

Installer executables and Linux binaries for Node.js can be found on their website download page:

https://nodejs.org/en/download/

No special install options are required to run VS-Saturn; users can download and unpack the latest prototype release from the GitHub repository:

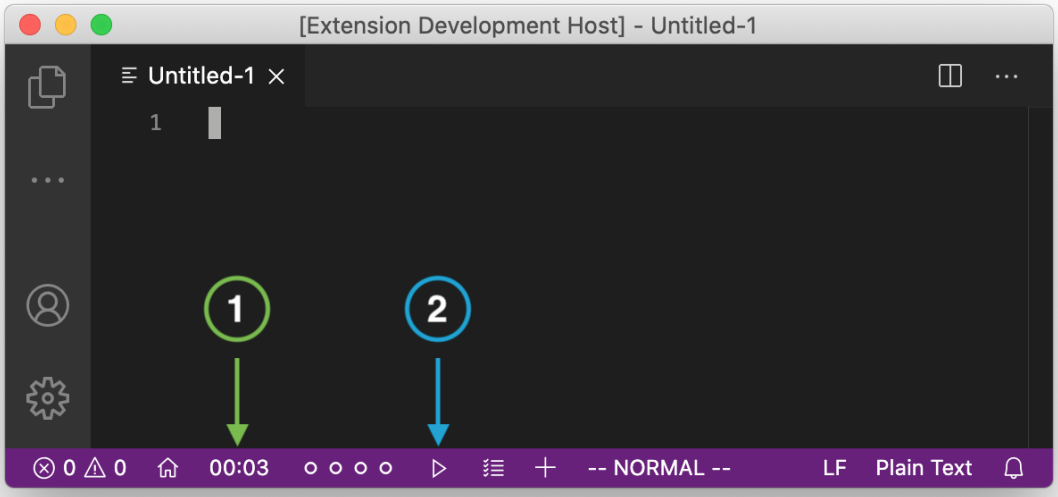
https://github.com/davidholton/vs-saturn/releases/tag/prototype-v1.0

Once unpacked, users can run “npm install” in the terminal inside the unpacked prototype folder. They can then open the prototype project in VS-Code by either dragging the unpacked prototype folder onto an open VS-Code application or running “code .” in a terminal inside the unpacked prototype folder (if VS-Code is recognized by the user’s PATH).

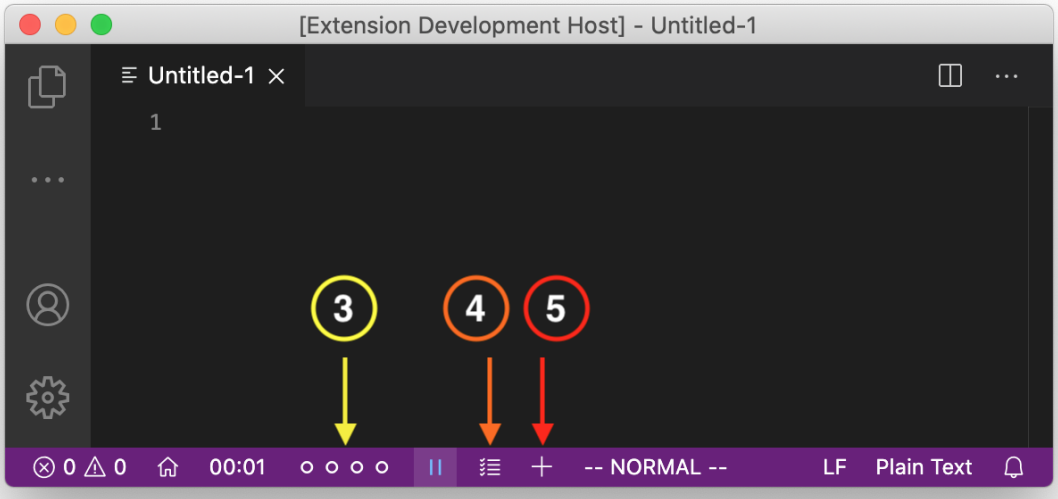
To run the extension, press “F5” while in VS-Code or, from the menu bar, click “Run” > “Start Debugging”. From there, a VS-Code Extension Development Host window will appear. In the bottom left VS-Saturn’s interface will appear; commands can also be found in the command palette (accessed with Ctrl+Shift+P or Cmd+Shift+P on macOS). Then, users can press the play button to start VS-Saturn.

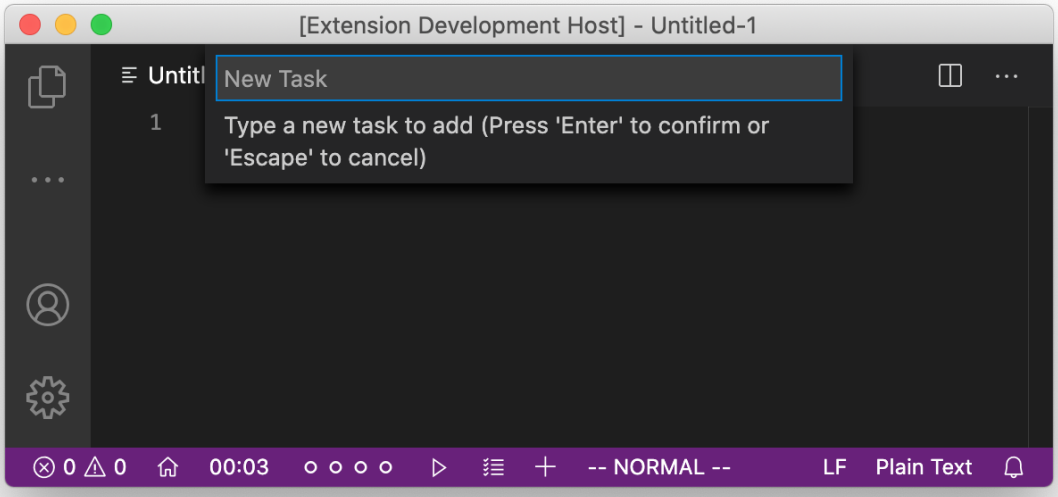
5.2. Sample Scenarios

VS-Saturn starts at the beginning of the user's first work period, displaying the time allotted ① until the user presses the start/resume button ②.

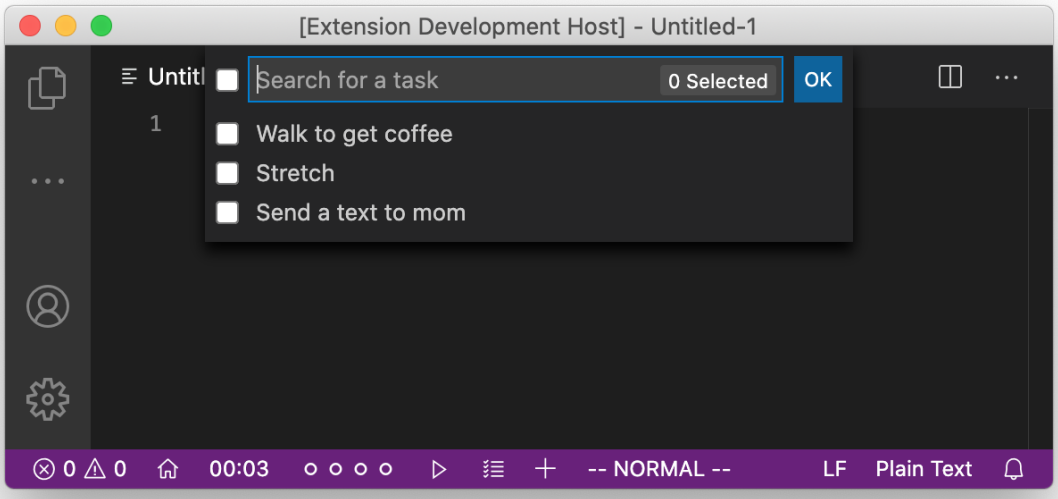


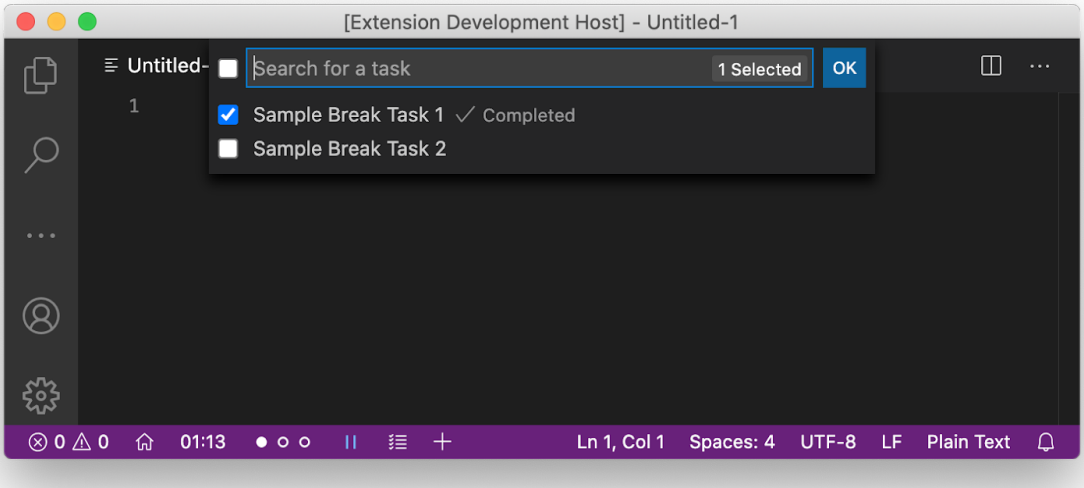
At this point, the start/resume button becomes a pause button and the work timer begins counting down towards zero. Toggling the pause/resume button allows the user to pause and resume the current timer at any point. The cycle progress indicator ③ displays the number of completed work periods as filled-in circles (work periods remaining in the cycle are shown as circle outlines). At any time, the user can view the work tasklist by clicking the tasklist button ④ or add a new item to the list by clicking the add button ⑤.



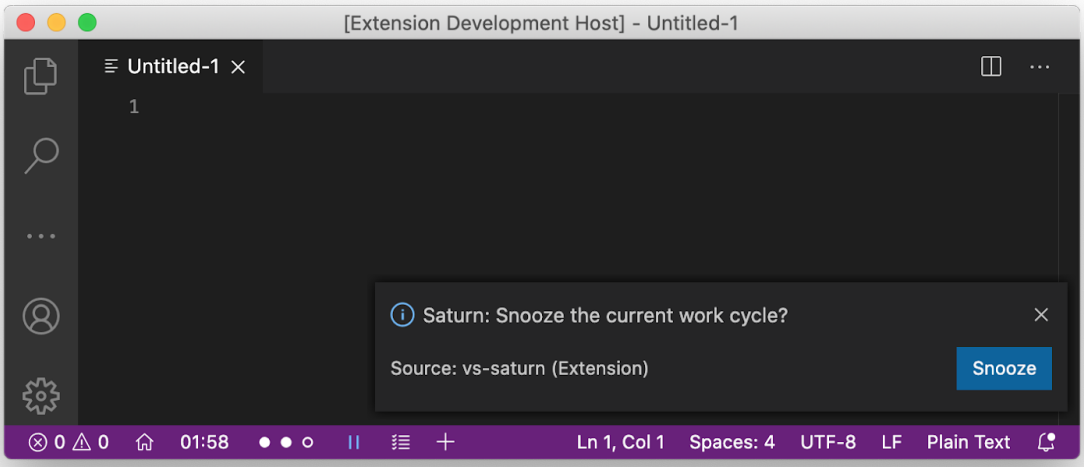


The user can search for tasks by typing them in the box at the top of the tasklist; items can be marked as completed by clicking their adjacent checkbox. When the user enters a break period, the break tasklist will be shown in this same interface in place of the work tasklist (thus maintaining strict separation between work and break time).

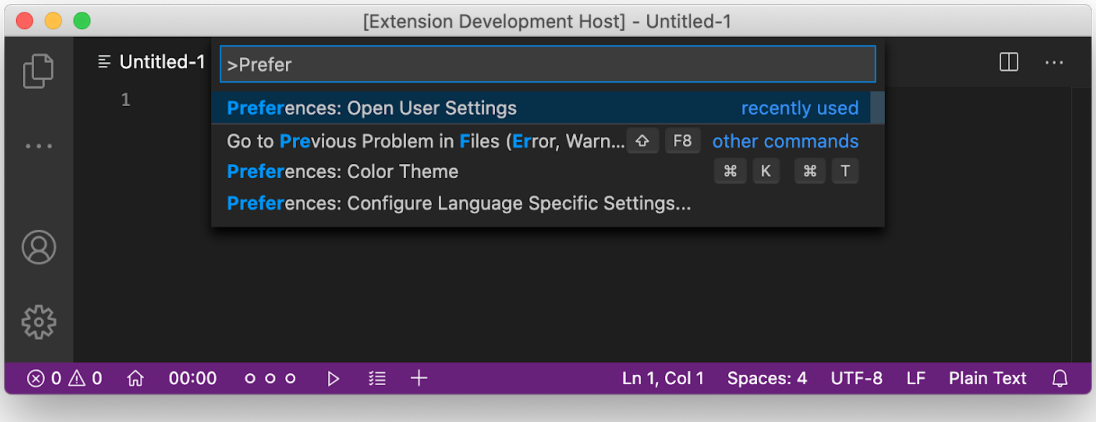


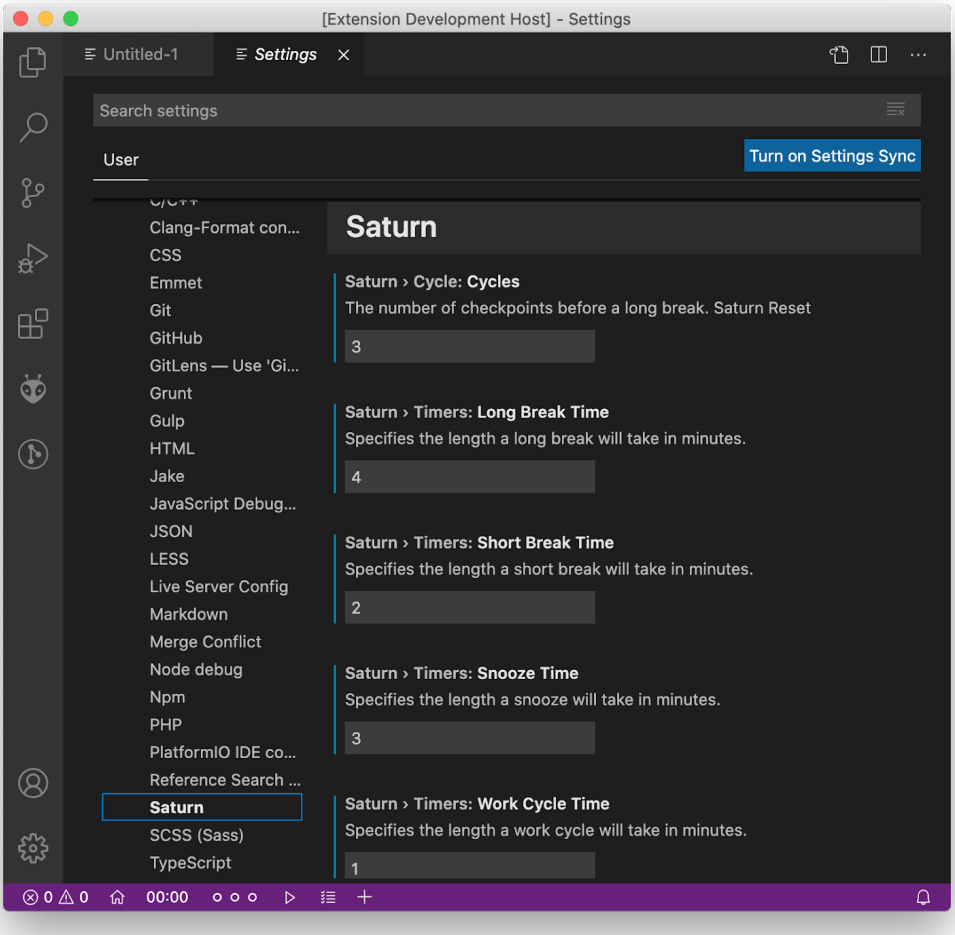


Once the timer reaches zero, the extension displays a notification reminding the user to take a short break (and offers the opportunity to temporarily delay that break). Once all circles in the cycle progress indicator are full, the extension will grant the user a longer break period.

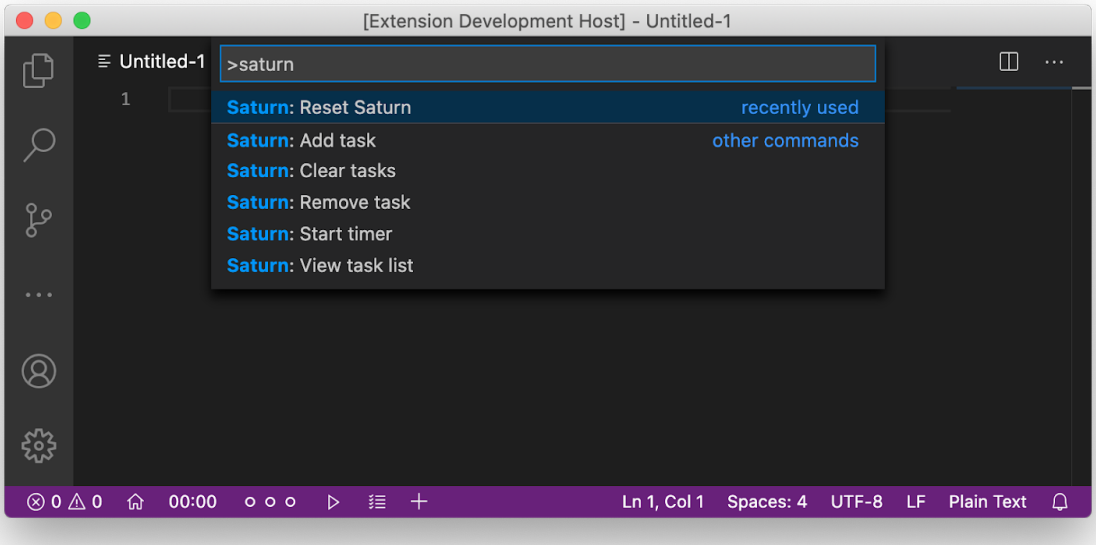


VS-Saturn supports customizing the number of work periods per cycle, as well as the duration of the work, short and long break, and snooze timers. These settings are accessed through VS Code's preferences (Ctrl/Cmd+Shift+P, then search for "Preferences").


[Extension Development Host] - Untitled-1
Untitled-1
>Prefer
1
Preferences: Open User Settings
recently used
Go to Previous Problem in Files (Error, Warn...
F8
other commands
Preferences: Color Theme
Preferences: Configure Language Specific Settings...
0 0
00:00
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Plain Text


[Extension Development Host] - Settings
Untitled-1
Settings
Search settings
User
Turn on Settings Sync
Clang-Format con...
CSS
Emmet
Git
GitHub
GitLens — Use 'Gi...
Grunt
Gulp
HTML
Jake
JavaScript Debug...
JSON
LESS
Live Server Config
Markdown
Merge Conflict
Node debug
Npm
PHP
PlatformIO IDE co...
Reference Search ...
Saturn
SCSS (Sass)
TypeScript
Saturn
Saturn › Cycle: Cycles
The number of checkpoints before a long break. Saturn Reset
3
Saturn › Timers: Long Break Time
Specifies the length a long break will take in minutes.
4
Saturn › Timers: Short Break Time
Specifies the length a short break will take in minutes.
2
Saturn › Timers: Snooze Time
Specifies the length a snooze will take in minutes.
3
Saturn › Timers: Work Cycle Time
Specifies the length a work cycle will take in minutes.
1
0 0
00:00

All of VS-Saturn's functions are accessible via the command palette (Ctrl/Cmd+Shift+P, then search for "saturn") including commands like "reset" which have no other UI-element associated with them.



6. References

[1] Website on Pomodoro: https://francescocirillo.com/pages/pomodoro-technique

[2] Francesco Cirillo's Pomodoro book: https://books.google.com/books?hl=en&lr=&id=rinKDQAAQBAJ&oi=fnd&pg=PT8&ots=fDXSFNwSDj&sig=vWY0L_SIt7TXQ-BYmKx3L4dd928#v=onepage&q&f=false

[3] VS-Code description: https://code.visualstudio.com/docs/editor/whyvscode

[4] Makeup of VS-Code extensions: https://code.visualstudio.com/api/get-started/extension-anatomy

[5] VS-Code Extension Host process: https://code.visualstudio.com/api/advanced-topics/extension-host

[6] VS-Code extension API: https://code.visualstudio.com/api/references/vscode-api

[7] Project website: http://www.cs.uml.edu/~dholton/vs-saturn

7. Point of Contact

For further information regarding this document and project, please contact **Prof. Daly** at University of Massachusetts Lowell (james_daly at uml.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.