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 the 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. 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 down the VS-Saturn to its individual components and specifies the purpose of each of them. This helps to guide the development team on the scope and technical specifications of the VS-Saturn project. Clients and stakeholders can also benefit from reading this document, as it conveys the development team's understanding of the initial requirements they put forth.

The target audience of this document is primarily the development team. This document provides a formal definition of the requirements of VS-Saturn which will assist the development team in reaching a common understanding of development needs. Properly informing clients and stakeholders of the design specifications of VS-Saturn is also a goal of this document. To achieve this end, non-technical descriptions of the requirements and example scenarios are provided in addition to the formal definitions. This document will furthermore offer clients and stakeholders some insight into the development process. Lastly, they will receive an explanation for 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. It is thus contained entirely in the VS-Code application. The goal of VS-Saturn is to aid users with time and task management. The VS-Saturn project incorporates the Pomodoro technique of time management. Using VS-Saturn, users can work on tasks in short bursts of productivity via the Pomodoro method. Work periods are split up by short breaks meant to ease mental fatigue during long workdays. With VS-Saturn, users will be able to keep track of these work and break periods.

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 task list 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 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 content of the rest of this document is as follows:

Section 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.

Section 3 – Specific Requirements:

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

Section 4 – Modeling Requirements:

All 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:

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:

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:

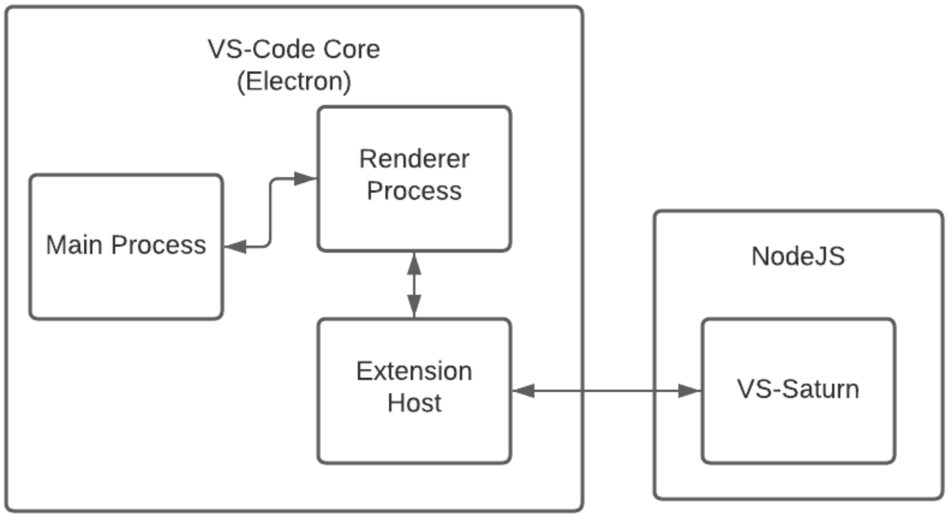
This section contains contact information for any questions about the product described in this document.

2 Overall Description

This section will cover the context that VS-Saturn will be used in, as well as its interfaces and functions. It will additionally go over the constraints under which the system will operate. This section also has the goal of identifying characteristics of potential users. Lastly, this section contains several assumptions on hardware and software requirements, as well as the environment and its interaction with the user.

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 software, users will set a work interval and subsequent a break interval. This allows the user to set intervals that fits best with their workflow. On the completion of the timer the user is notified, and the user can choose to take a break or to let the timer sleep for an additional amount of time. Because VS-Saturn is an extension running with VS-Code, it is targeted towards software development environments. VS-Code is a required installment to run VS-Saturn; coincidingly, 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 JavaScript, it was chosen as the development language. As seen in the following diagram, the VS-Saturn extension runs on a separate Node.js node, communicating to the main VS-Code render process through the extension host [4].



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 a task checklist so the user can track progress and set goals.
- Allow the user to customize the time intervals for both the work and break time.
- Provide a non-intrusive notification when the timer is up by changing the border color.
- Allow the user to snooze the work alarm if additional time is needed.

2.3 User Characteristics

It is assumed that users have some experience using VS-Code for writing programs regardless of programming skill. The expectation is that users have general knowledge with VS-Code's workflow to at least a basic degree. Users must be able to install VS-Saturn from the VS-Code Marketplace found online at 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. VS-Saturn’s user interface is implemented with 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 task list feature is stored in an unprotected SQLite database inside the users VS-Code application data folder.

2.5 Assumptions and Dependencies

It is assumed that the user's hardware and software already fulfills 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 Window 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 task lists.
- The option to automatically show the task list 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 task lists, one for work periods and one for break periods.
    - 2.1. The task lists 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 task list and add tasks to the break task list.
        - 2.2.1. The break task list 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 task list.
        - 2.3.1. The work task list should never be visible nor alterable during break periods.
    - 2.4. Neither of the task lists 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 task lists.

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.

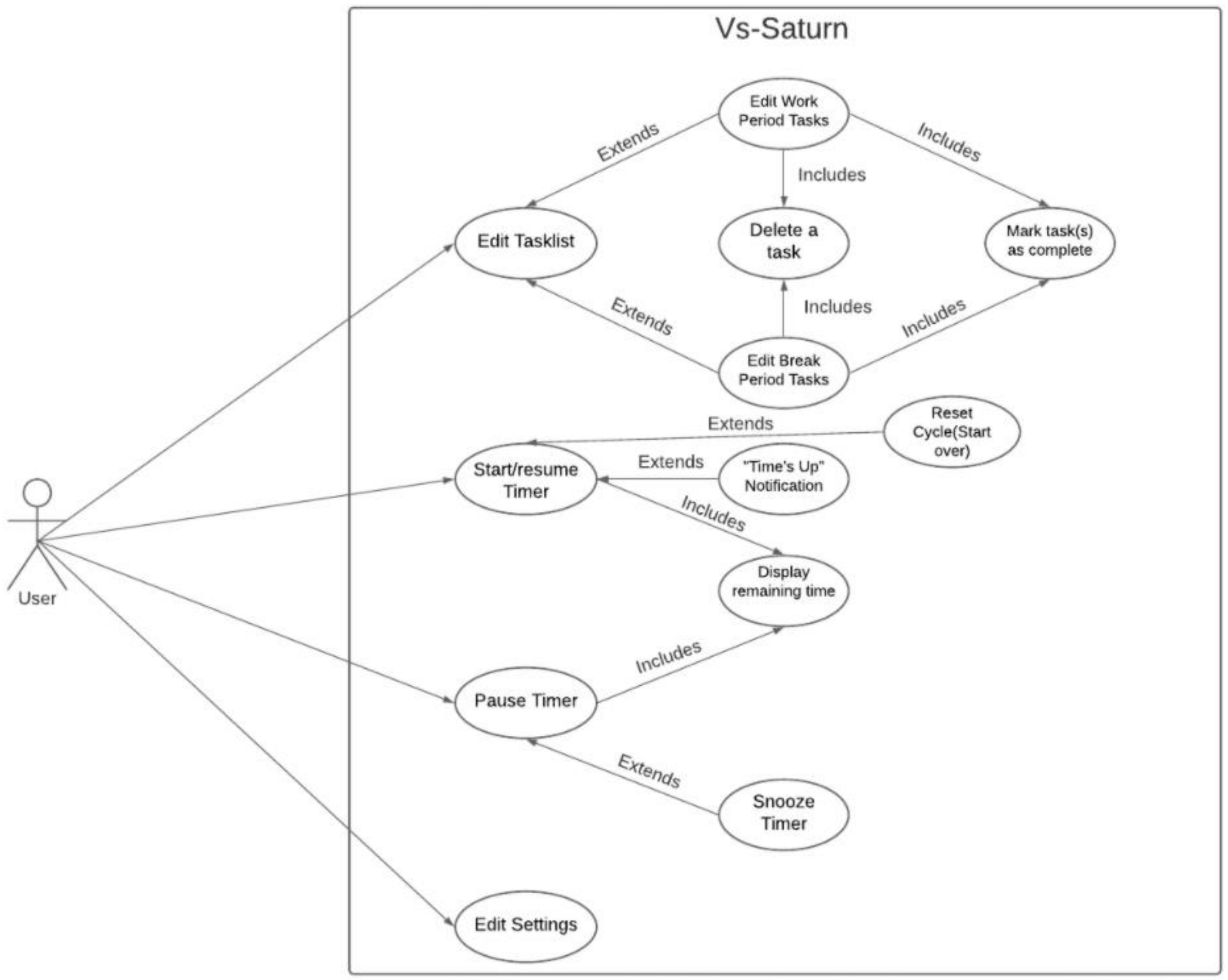
  4.3. All user-configurable options mentioned in requirements 1-3 should also be found in this interface.

  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 is comprised 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:



Use Case Name:	**Edit Tasklist**
Actors:	User(initiator)
Description:	The user wants to edit the tasklist. They can do this by clicking on the icon to open up the tasklist.
Type:	Primary, Essential
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 2
Uses cases:	User clicks the “tasklist” icon from the taskbar

Use Case Name:	**Edit Work Period Tasks**
Actors:	User(initiator)
Description:	The tasks specific to the user’s work period for them to edit.
Type:	Secondary
Includes:	Mark task(s) as complete, Delete a task
Extends:	Edit Tasklist
Cross-refs:	Requirements 2.2, 2.3.1
Uses cases:	When in a work period; opening the tasklist will display these tasks.

Use Case Name:	**Edit Break Period Tasks**
Actors:	User(initiator)
Description:	The tasks specific to the user’s break period for them to edit
Type:	Secondary
Includes:	Mark task(s) as complete, Delete a task
Extends:	Edit Tasklist
Cross-refs:	Requirements 2.3, 2.2.1
Uses cases:	When in a break period; opening the tasklist will display these tasks.

Use Case Name:	**Delete a Task**
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:	Option when editing 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:	**Start/Resume Timer**
Actors:	User(Initiator)
Description:	By clicking on the play button, the timer will either begin counting down, or resume counting down if it is currently paused.
Type:	Primary, Essential
Includes:	Display remaining time
Extends:	N/A
Cross-refs:	Requirements 3.2, 3.2.1
Uses cases:	User clicks the “play” icon from the taskbar

Use Case Name:	**Reset Cycle (Start over)**
Actors:	User(initiator)
Description:	If the user is interrupted for an extended period of time and wishes to start their pomodoro over from the beginning they can reset completely.
Type:	Secondary
Includes:	N/A
Extends:	Start/Resume Timer
Cross-refs:	Requirement 3.3
Uses cases:	Option the user can have when starting the timer.

Use Case Name:	**“Time’s Up” notification**
Actors:	N/A
Description:	When the timer runs out a notification of switching the border theme will happen to tell the user to either take a break or go back to work.
Type:	Primary
Includes:	N/A
Extends:	Start/Resume 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:	**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 whenever VS-Saturn is in use.

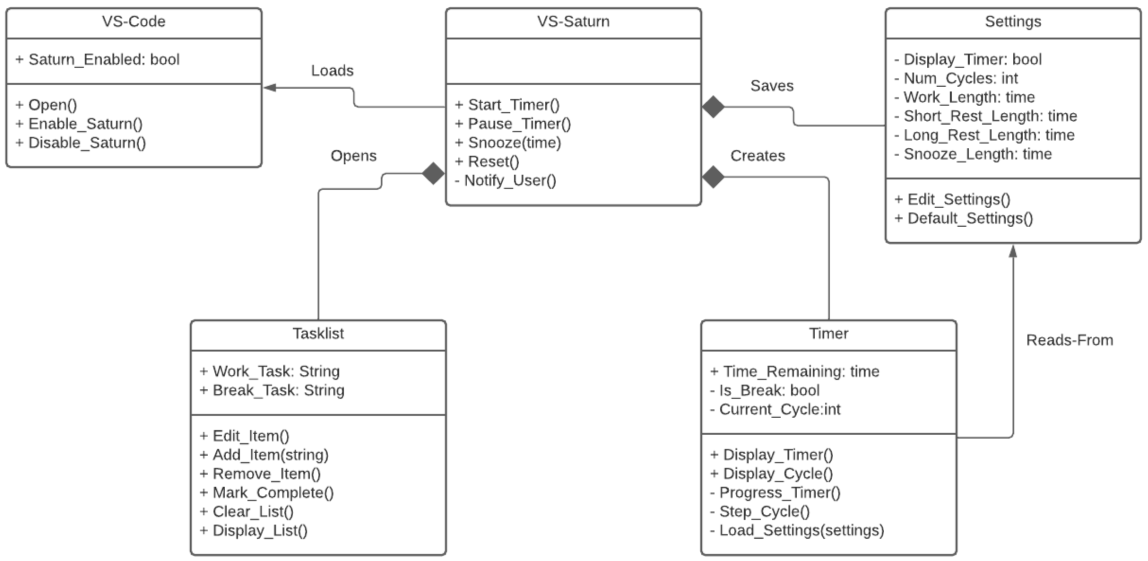
Use Case Name:	**Pause Timer**
Actors:	User(Initiator)
Description:	If the user needs to step away for just a moment, they can pause the timer.
Type:	Primary
Includes:	Display Remaining Time
Extends:	N/A
Cross-refs:	Requirement 3.2, 3.2.1
Uses cases:	User clicks the “pause” button from the taskbar

Use Case Name:	**Snooze Timer**
Actors:	User(initiator)
Description:	When the user wants to pause the timer for a certain amount of time, delaying a break period, they can choose to snooze the timer
Type:	Secondary
Includes:	N/A
Extends:	Pause Timer
Cross-refs:	Requirement 3.1
Uses cases:	User clicks the “snooze” icon next to the pause/play button

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, the amount of cycles in a pomodoro, toggle the timer visibility. They can also restore to default settings here.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirements 4, 4.1, 4.2, 4.3, 4.4
Uses cases:	User clicks the “settings” option for the extension.

Class Diagrams:

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



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()	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.
Operations		
public	Start_Timer()	This will make a call to the timer class that it should start counting down.
public	Pause_Timer()	Makes a call to the timer class to stop the timer from counting down.
public	Snooze(time)	The user can specify an amount of time to snooze the timer for.
public	Reset()	Resets the cycles and the timer if the user wants to start over from the first pomodoro cycle
private	Notify_User()	Notify the user that the timer is up by changing the color of the VS-Code's border.
Relationships	VS-Saturn is the extension itself. It functions as a controller of sorts with operations to call the timer class to start, stop or reset the timer, and notify the user when the time is up.	

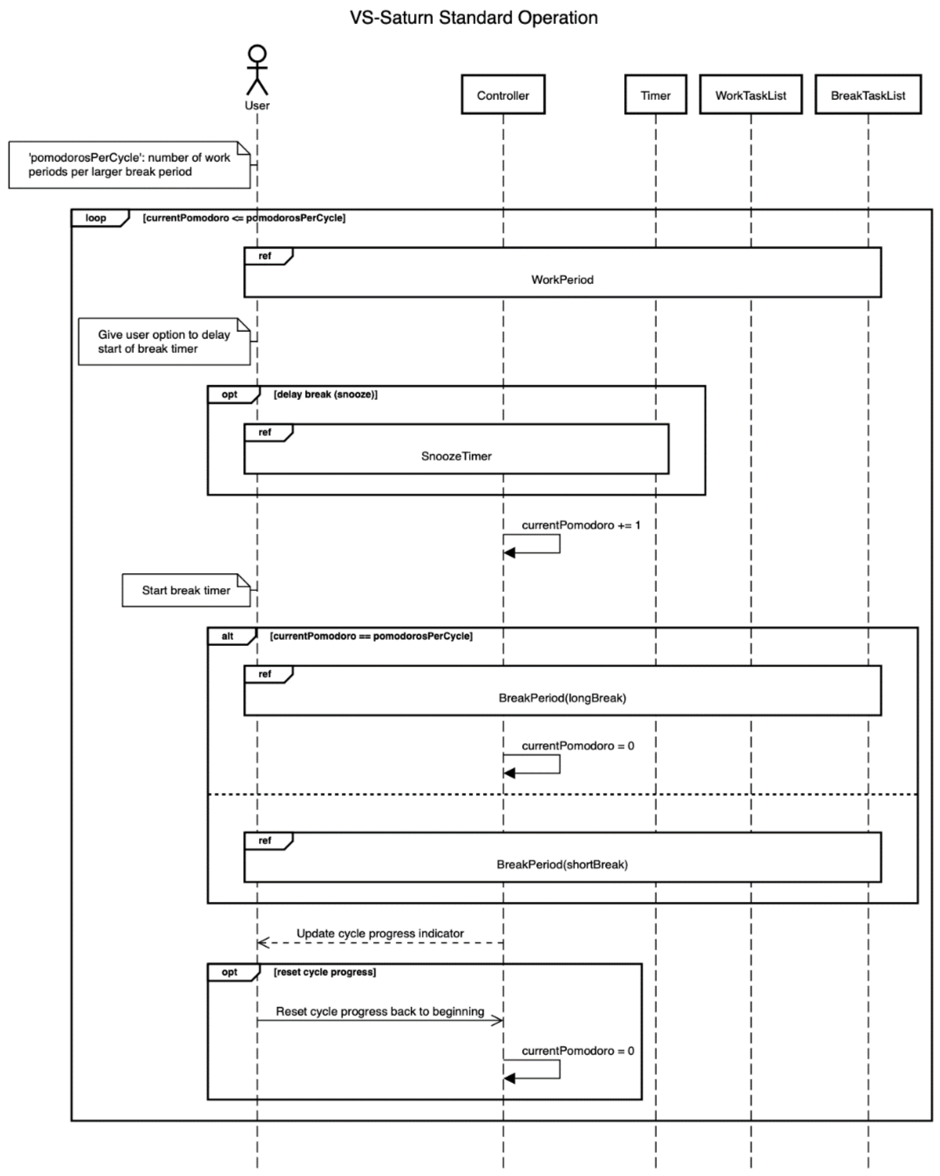
Element Name		**Description**
Timer		The Timer class will keep track of the time counting down and display it.
Attributes		
public	Time_Remaining: time	Where the amount of time for the timer is actually kept track.
private	Is_Break: bool	Keep track if it is currently a break or not to determine the timer.
private	Current_Cycle: int	Integer value to keep track of how many pomodoro cycles have been gone through.
Operations		
private	Progress_Timer()	Counts down the timer until it reaches 0. Once it reaches 0 it will check if Is_Break() is true. If so, it will then check Current_Cycle to determine which break timer to be used.
public	Display_Timer()	Show the timer on the screen for the user to see.
public	Display_Cycle()	Display on the screen a tally system of the amount of cycles the user has gone through.
private	Step_Cycle()	Adds 1 to the cycle, checks to see if the maximum number of cycles was reached; if so, it will reset to 0 to start over.
private	Load_Settings(settings)	Loads the timer settings that the user specifies.
Relationships	Timer is probably the biggest part of the VS-Saturn extension. This keeps track of the time remaining in a cycle, as well as the current cycles, and display this information to the user. In addition, load the custom settings (if any) the user has specified for the timer and cycles.	

Element Name		**Description**
Tasklist		The tasklist is where the user can write down and keep track of different tasks for their work/break and cross them off when done.
Attributes		
public	Work_Task: String	A task the user specifies for their work cycles.
	Break_Task: String	A task the user specifies for their break cycles.
Operations		
public	Edit_Item()	Make a change to an existing item.
public	Add_Item(string)	Add an item to the task list.
public	Remove_Item()	Remove an item from the task list.
public	Mark_Complete()	Mark off an item as completed.
public	Clear_List()	Clear the specified list the user wants.
public	Display_List()	Display the specified list to the user.
Relationships	The tasklist is a part of VS-Saturn and will function as a dropdown or popup list of sorts 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.	

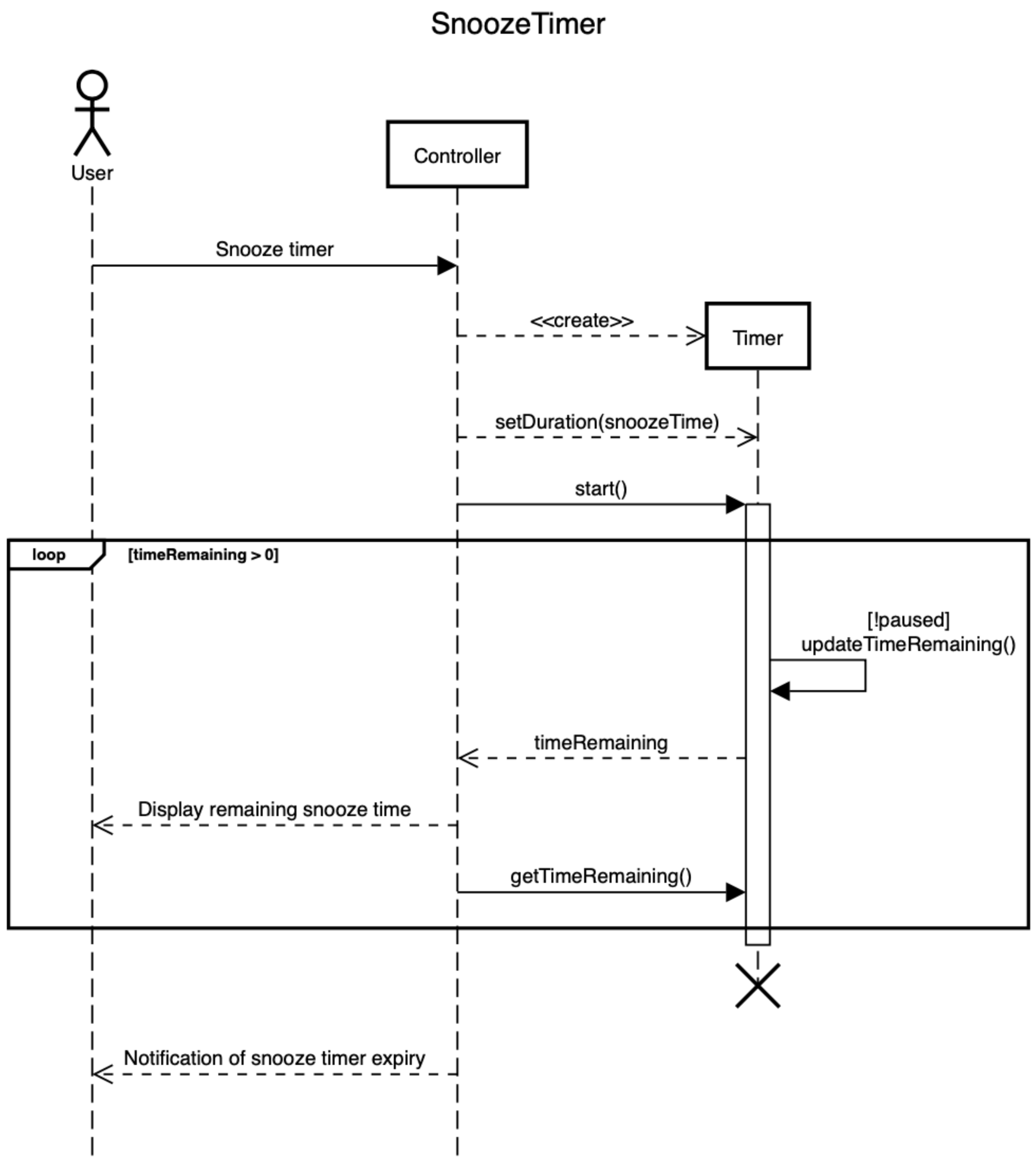
Element Name		Description
Settings		The settings will be where the user can go to change various things about Saturn to customize it to their likings.
Attributes		
private	Display_Timer: bool	Whether the timer will be shown or not (Default: yes).
private	Num_Cycles: int	The amount of cycles the user wants for their pomodoro (Default: 4).
private	Work_Length: time	The amount of time for a work cycle (Default: 25 minutes).
private	Short_Rest_Length: time	The amount of time for a short rest (Default: 5 minutes).
private	Long_Rest_Length: time	The amount of time for a long rest (Default: 20 minutes).
private	Snooze_length: time	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.
public	Default_Settings()	Restores all the settings to their default values.
Relationships	Settings is where the user can customize various parts of VS-Saturn to fit their needs. They can do things such as toggle the visibility of the timer, 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().	

Sequence Diagrams:

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


VS-Saturn Standard Operation
User
Controller
Timer
WorkTaskList
BreakTaskList
'pomodorosPerCycle': number of work periods per larger break period
loop [currentPomodoro <= pomodorosPerCycle]
ref WorkPeriod
Give user option to delay start of break timer
opt [delay break (snooze)]
ref SnoozeTimer
currentPomodoro += 1
Start break timer
alt [currentPomodoro == pomodorosPerCycle]
ref BreakPeriod(longBreak)
currentPomodoro = 0
ref BreakPeriod(shortBreak)
Update cycle progress indicator
opt [reset cycle progress]
Reset cycle progress back to beginning
currentPomodoro = 0

The first diagram follows the normal sequence that VS-Saturn would go through under normal operation. There are objects for the timer and the task list which persist throughout the entirety of the program's duration, as well as a controller to keep track of the pomodoro cycles. As execution progresses, the timer will count down and loop through the work period with giving the user an option to delay (snooze) the timer. Eventually, once the timer reaches "0:00", the Pomodoro cycles are incremented by +1 and enter the break cycle. This will be a long break if the final Pomodoro has been reached, or a short break otherwise. Afterwards, the indicator to the user of how many cycles they have gone through gets updated, setting the cycles back to 0 if they have reached the final amount. The program then starts looping over from the beginning all over again.



In the second diagram, is the sequence for if the user decides to snooze the timer for a specified amount of time. When the user wishes to snooze the timer, they first must set a duration for this snooze timer. 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. Once the snooze time runs out, the user is notified, and VS-Saturn returns to the main sequence.

5 Prototype

The prototype includes the core interface requirements features for the project. Most of the user interface is displayed in the bottom left of the VS-Code status bar at the bottom of the application window. This includes a timer 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 task list button, add task button. The task list button displays an interface that let 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 task list.

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+Shfit+P or Cmd+Shift+P on OS X). 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 and 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 then download and unpack the latest prototype release from the GitHub repository:

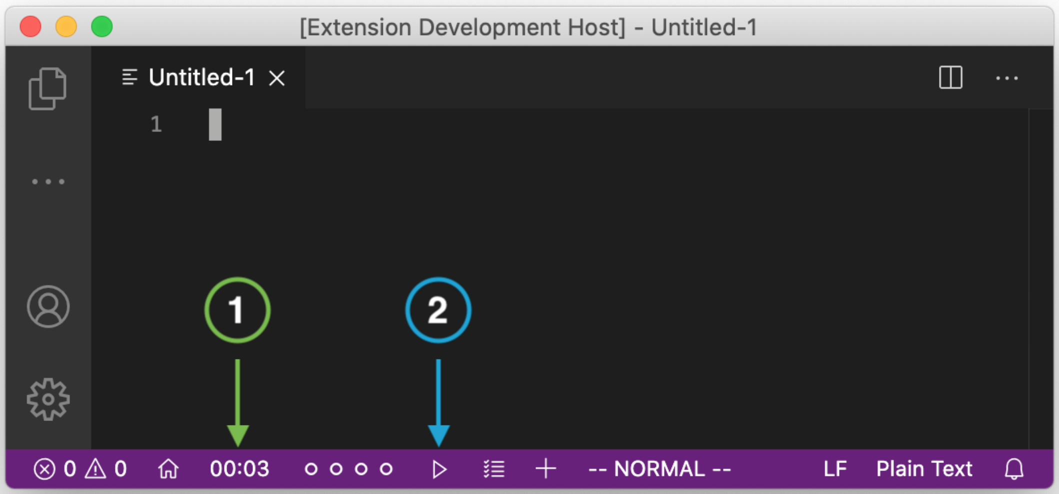
https://github.com/davidholton/vs-saturn/releases/tag/prototype-v1.0

Once unpacked, users can run "npm install" in the terminal inside the unpacked prototype folder. They can then pen 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.

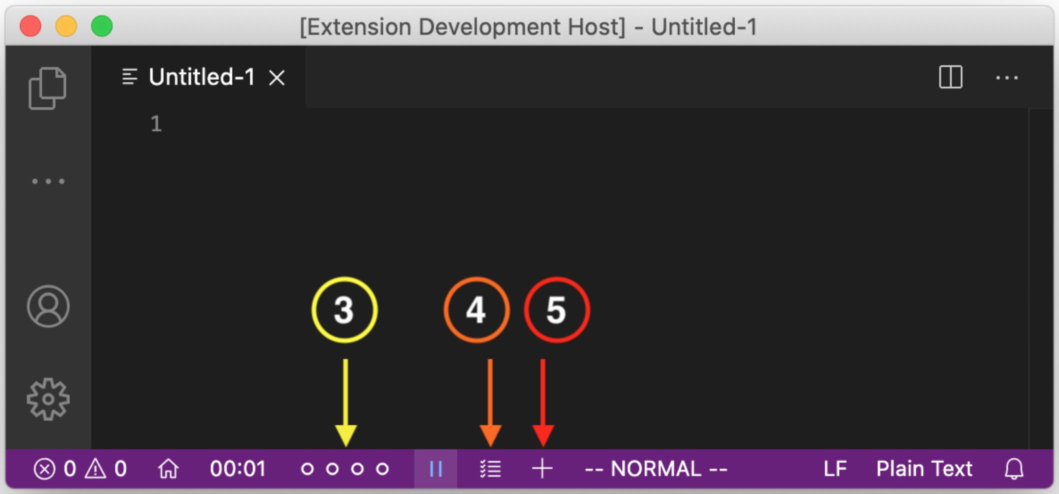
To run the extension hit "F5" while in VS-Code or from the menu bar click Run > Start Debugging. From there, a VS-Code Extension Development Host window will appear. In the bottom left VS-Saturn's interface will appear and commands will be found in the command palette (Ctrl+Shift+P or Cmd+Shift+P on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

5.2 Sample Scenarios

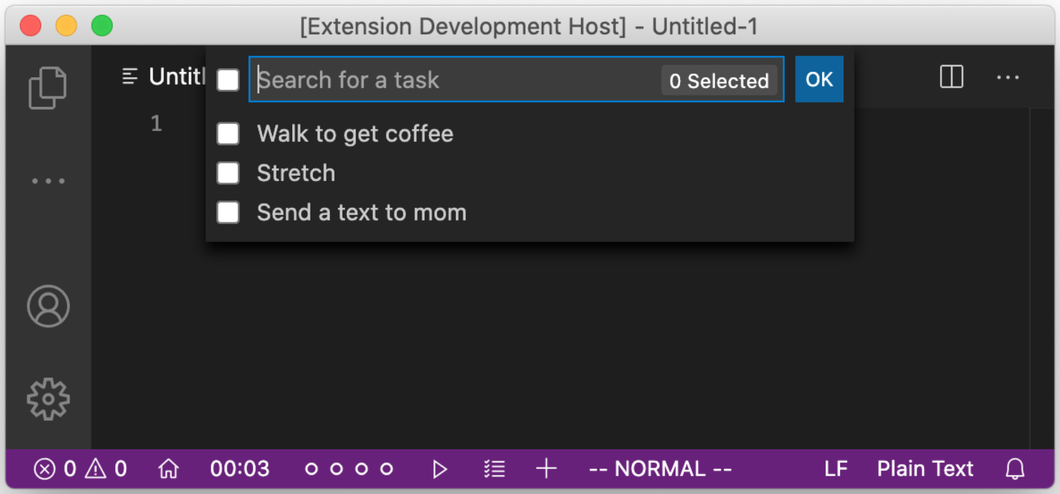
VS-Saturn starts at the beginning of the user's first work period, displaying the time allotted ① until the user presses the start/resume button ②.



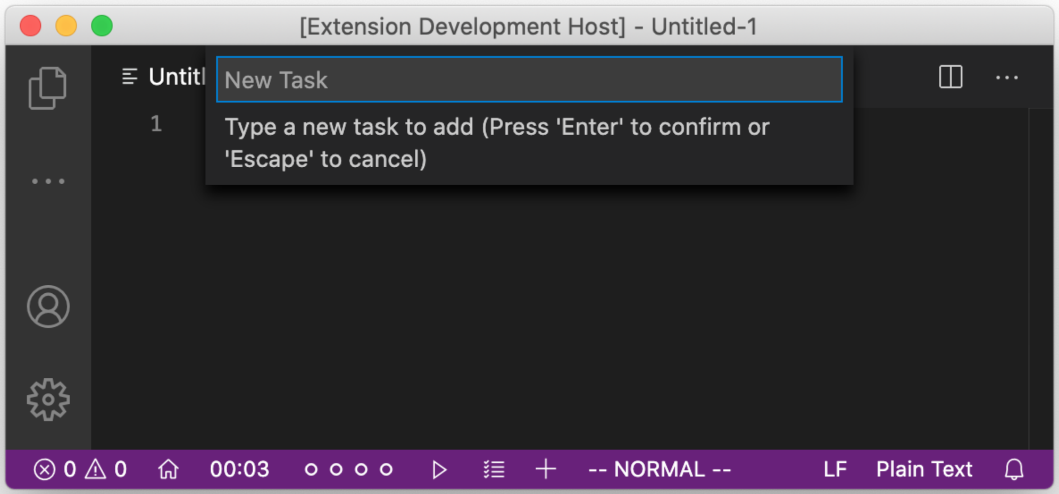
At this point, the start/resume button becomes a pause button and the work timer begins counting down towards zero. Toggling the pause/resume button allows the user to pause and resume the current timer at any point. The cycle progress indicator ③ displays the number of completed work periods as filled-in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



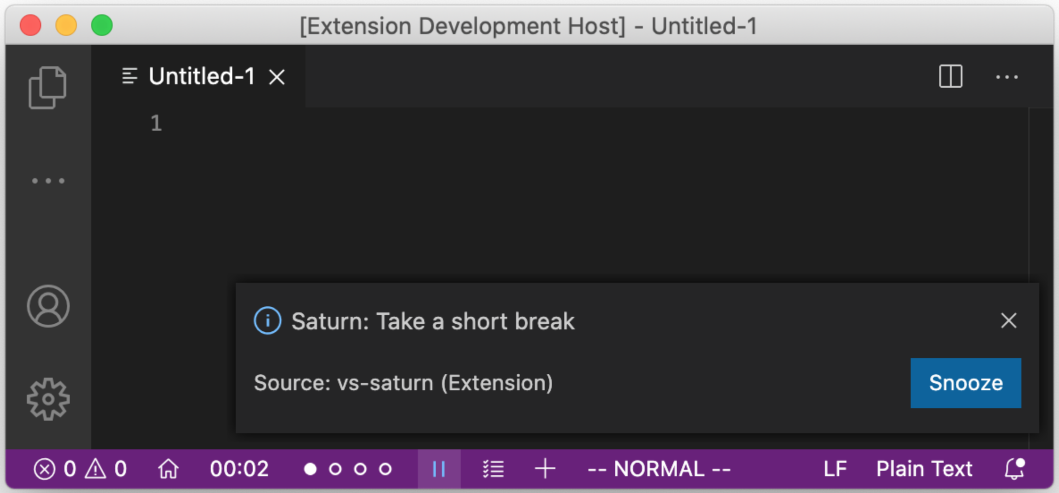
The user can search for tasks by typing them in the box at the top of the task list; items can be marked as completed by clicking their adjacent checkbox. When the user enters a break period, the break task list will be shown in this same interface in place of the work task list (thus maintaining strict 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). In the final product, this will be replaced with a less intrusive notification method and will be accompanied by a break timer and break task list.


[Extension Development Host] - Untitled-1
Untitl
New Task
Type a new task to add (Press 'Enter' to confirm or 'Escape' to cancel)
0 0
00:03
-- NORMAL --
LF
Plain Text

Once all circles in the cycle progress indicator are full, the extension will grant the user a longer break period.



Saturn: Take a short break

Source: vs-saturn (Extension)

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.