California University of Pennsylvania

CSC 490: Senior Project

Virtualization of a Turing Machine
Requirements Document

Ji Sue Lee
James Merenda
Daniel Urban

Instructor Comments/Evaluation

Table of Contents

Abstract

The Virtualization of a Turing Machine project is an alternative teaching tool using visual elements and a programming interface. The reason we decided to create this web-based Turing Machine program is to further aid the process of understanding the concept of a Turing Machine in a different and accessible way. Most hard copy texts describing the concept are extremely verbose and complicated. The Virtualization of a Turing Machine is meant to alleviate the difficulties of the concept by providing a visual demonstration as well as an interface to try a different program or write custom programs. The project will be available on an online domain hosted by Cal U or available to download directly. This document entails the necessary steps needed to create a prototype of the project.

Introduction

Background/Overview of Project

The Virtualization of a Turing Machine was thought of by professors at local colleges that needed another method of teaching the concept of a Turing Machine. The idea of a Turing Machine to most is a daunting task to comprehend. With our project, we hope to create an environment where the student or enthusiast has a hands-on learning experience with an interactive Turing Machine. Users will be able to access the project from any internet-enabled device and run pre-made programs built into the website. They will also have the option of writing their own code into an interface that interacts with the Turing Machine. The Turing Machine will also be animated to further enforce the idea of how the concept functions. In the long run, this project will aid teachers and enthusiasts in learning how a Turing Machine operates.

Objective of Project

The main objective of our project is to create an accessible environment where users with an internet-enabled device will run programs loaded on the website or write their own and watch an animation of what the Turing Machine is doing. Other similar projects have preloaded programs, animated Turing Machines, an interface for developing programs, or some combination of the features. Combining all these features into one webpage provides a cohesive experience for the end user that is easily accessible online or offline after downloading the webpage.

Team Details/Dynamics

Every member's work toward the project will be necessary to maintain a structurally sound program. We have worked in groups to collaborate on projects in one way or another. Using this experience to work with each other will be essential for learning from each other and progressively applying our skills to the project at hand.

As a group, we hold weekly check-ups for expressing questions and concerns regarding each step in the project. We discuss these topics through the following applications:

- Discord (Online Chat Application)
- Google Suite (Online Office Suite Applications)

With the tools above and our combined efforts, we will achieve success in developing quality software for teachers or enthusiasts to use.

Application Domain

Project Context

This project is intended for students, teachers, and general enthusiasts to help visualize a given Turing Machine. Users can view preset programs or create their own programs through an interface on the publicly accessible website. The preset programs are common Turing Machine examples that users can select without having any knowledge about how to create a valid Turning Machine. For users who already have knowledge of Turing Machines and wish to improve on that foundation, the project has an interface to allow users to create their own Turing Machine programs. Users may also edit their programs in the webpage and receive feedback on any possible errors.

Initial Business Model

Operating Environment

The website will use HTML and CSS for formatting while the actual Turing Machine visualizer will be written in JavaScript. An Internet connection will be required to first access the website, but the user can then download the source files to use it offline. The site will support most modern internet browsers. Additionally, it will use the browsers' local storage to save the user's custom programs and load them without having to rewrite previous work.

Description of Data Sources

Sets of data are provided by both the program and the user. For the Turing Machine to function, both user-provided data and program instructions are necessary for producing a desired output. For user-provided Turing Machine designs, the user inputs their design in the provided design space, which is then translated to actions that the Turing Machine performs to create an output. Similarly, custom programs written by the user may also be written into the programming interface, which is translated into actions animated by the Turing Machine and creates an output.

Use Case UML Diagram

Figure 1

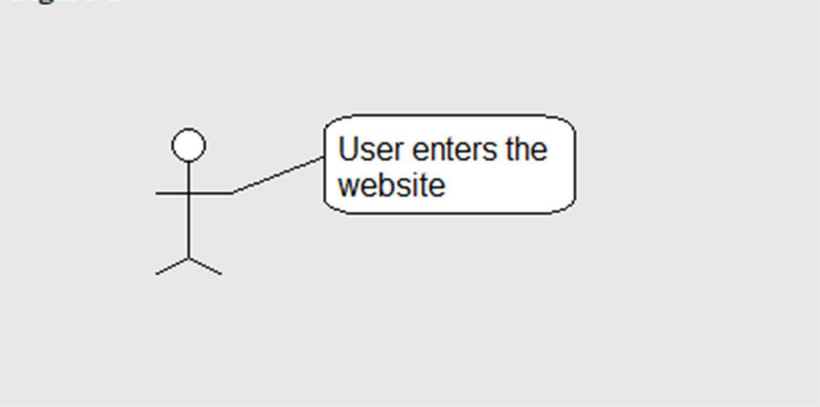
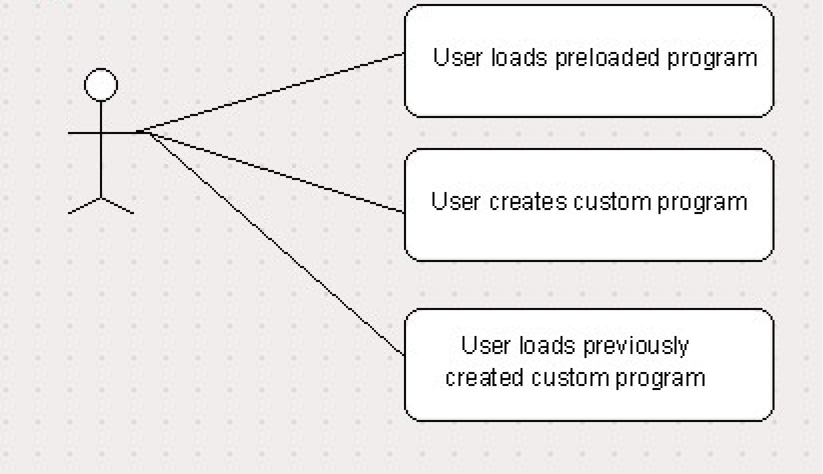


Figure 2


User loads preloaded program
User creates custom program
User loads previously created custom program

Description

The user enters the site and either selects a preloaded program to load or creates a new program (Figure 1). If a preloaded program is selected, the user can then run it immediately (Figure 2). If the option to create a custom program is chosen, the editing interface appears, allowing the user to immediately create a new program and run it (Figure 2). Additionally, if the user has previously created a custom program, the option to load that program is also available (Figure 2).

Initial Requirements

Functional

The Virtualization of a Turing Machine will allow the user to pick from a pre-existing list of examples or input their own code. Once the program is loaded, an animation of the Turing Machine will start and produce an output. Users can pause and resume the animation at any time along with stopping the animation or viewing it frame by frame. Additionally, users will be able to input their own data and modify the pre-existing examples to whatever they desire.

The examples are provided in a drop-down menu and include examples such as "Binary Addition," "Decimal to Binary Conversion," "Duplicate Binary String," etc. Each example may require a different number of "tapes" for the Turing Machine animation, represented by rows of boxes to hold data.

Additionally, users may view more information about Turing Machines on a different section of the website. Information regarding the project, the people who designed it, and other contact info will also be included.

The software behind the virtualization will be written primarily in JavaScript, with additional code using CSS and HTML.

Due to this project being in early development, these requirements are mostly outlined for our plans for the project. Once the project enters production, our goal is to have a simple and an aesthetically pleasing program for users.

Nonfunctional

The Turing Machine Virtualization is basic and easy to comprehend while made in a professional manner. Users will have an easy time using the tool while on a website that feels intuitive and visually well-designed. The code for the examples will also be well-commented for users to further understand what the code is doing.

Documentation

Table 1

Documents in the Project

Documents	Description
Proposal	Outlines the initial idea and basic concepts.
Requirements	Lays the groundwork for what the project will entail and require in terms of software and hardware.
Specifications	Provides more details about the project.
Design	Outlines how the project will be made.
Project Log	Keeps track of the team member progress and contributions.
User Manual	Provides detailed instructions on how to operate the website.

Testing / Revisions

Our team split this document into four parts. Each of us worked on one part assigned by the team leader. We all proofread the document prior to reviewing it with the Writing Center and submitting the assignment. We used Google Docs to all view and edit the document at the same time, and we each worked on our own respective parts until it was time to review the final draft. After a few revisions and a mutual agreement between group members, we decided the requirements document was ready to submit.

Appendix A: Technical Glossary

CSS

Cascading Style Sheets: Another markup language used alongside HTML for additional website formatting.

HTML

Hypertext Markup Language: A markup language used to format the appearance and distribute elements in a webpage.

JavaScript

A lightweight programming language generally used for traditional programs embedded within webpages.

Interface

The Turing Machine editor/apparatus itself.

Turing Machine

An abstract machine model that simulates algorithms through manipulating symbols on a strip of tape with a set of rules.

User

A person interacting with the site and program.

Appendix B: Team Details

James Merenda:

James Merenda was the workflow leader and assigned tasks to the rest of the team. He was also the main editor of the document and revised where necessary. James also corresponded with the Writing Center for grammar and context editing.

Daniel Urban:

Daniel Urban oversaw the Initial Requirements section, documented the revision process of this document, and designed the workflow authentication appendix.

Ji Sue Lee:

Ji Sue Lee wrote the Application Domain section and the Initial Business Model section and relevant subsections.

Appendix C: Workflow Authentication

I, James Merenda, hereby attest that I have done the work documented herein.

Signature: ____

Date: ________________

I, Ji Sue Lee, hereby attest that I have done the work documented herein.

Signature: ____

Date: ________________

I, Daniel Urban, hereby attest that I have done the work documented herein.

Signature: ____

Date: ________________

Appendix D: Report from the Writing Center