Theodore R. Popp III

This is a non-exhaustive list but contains probably half of job work and side projects, and all of my class programming projects.

Dropbox

Summer intership at Dropbox.

Tags and Workboard	Wanted to propse a new feature and the utility of file tags.	Implemented file tags and a workboard that could group by tags.	Used ReactJS and python with team of 4.
Accessibility	Some parts of website were not accessible for disabled people or strange characters.	Caused proper handling of uncommon characters and made web elements accessible without use of mouse.	Used html, javascript, css.
Performance	The server was slow.	Did performance analysis to identify slow code and improve the performance.	Python and internal tools.
Version based Previews	Server logic assumed that the most recent version of a file was always used.	Allowed version based actions and created an interface to allow wider use.	Python and secure encodings.
Rewrite	Our backend had some invalid assumptions.	Refactored the majority of our backend to be more modular and work with multiple backends without interfering with the work of others.	Python
Teaching	A new resident was working with parts of the system I had most recently added to.	Provided mentorship and guidance to him in what functions to use and how to test and use tools.	Communication
File History	Code had no natural way to work with file versions.	Added to interface to allow history to be a natural part of a file.	Python
Important Revision Identification	Most file saves include only small changes.	Implemented a revision filter to allow identification of important file revisions with access to only metadata.	Python
Comment Logging	Need knowledge of a person's comments on files.	Added logging of data for later use in our webpage.	Python and protobuffs
View Showing	Want to show what files a person has viewed.	Added the construction of necessary data and depiction on the front end.	Python and ReactJS
Data Change	Our data format wouldn't work anymore.	Created new data format with a fallback to allow migration with no failures. Added versioning so this wouldn't be a concern again.	Python
Interface Creation	Intersystem communication was new and not clearly defined.	Designed and implemented interfaces with frontend and database systems to allow cleaner code.	Python primarily.

Categorization and Filtering	Wanted to group data along time and file type.	Created grouping and filtering systems to allow forms of data location for the user and added filtering of junkfiles.	Python

Bebop

Summer internship involving backend work at a startup.

Project	Problem	Solution	Implementation
CPU profiling	Identify CPU intensive operations in tight loop of server.	Identified inefficient operations, and reimplemented features where reasonable.	Used v8 tools to identify performance bottlenecks.
Memory Profiling	Identify memory leaks in server.	Found and reported memory leaks and probable reasons.	Heap profiling, and diffing after simulating users on the website.
Caching	Remove IO bottlenecks.	Implemented a native LRU cache for single machines in a distributed system.	LRU cache implemented by linked hash-set.
Research	What is available to improve nodejs.	Reported nodejs profiling features and general trends in nodejs modules.	Involved analyzing available tools for v8 profiling, and the general trends of code being implemented in nodejs.
Checksumming	Implement native checksumming to speed up javascript implementation.	2-300x speedup on checksumming.	CRC32 checksumming implemented in a node addon.
Build System	Node addon object files have no place in the build system.	Add makefile definitions to build system for production and test usage.	Made additions to a non-recursive make system.
Load Testing	Test holistic user experience on system while under increasingly heavy loads.	Created code to simulate many users visiting web pages, interacting with the page, and timing response time.	Used selenium and chromedriver along with profiling on the server.
Code Generation	Reflective code is relatively inefficient.	Python script generates code to have the same effect without reflection.	Generate C++ code and macros with a python script to improve upon existing reflective code.
Performance Improvement	Identify performance problems in javascript code.	Restructure code to allow the v8 engine to fully optimize code.	Javascript code was restructured through knowledge of the v8 optimization strategies.

Condensed Matter Research

Project	Problem	Solution	Implementation
FMT Applet	People avoid certain calculations due to their difficulty.	Implement an applet to raise awareness of the accurate calculations and allow ease of use by many.	Implemented in Java with Swing and a lot of math.

Fluid RDF Applet	People don't know fast, analytic methods to determine fluid properties.	I created an applet that allowed teaching of advanced concepts in the field and verification of calculations before experimentation.	Implemented in Java with Swing, along with providing licensing and implementing some Math libraries.

Side Projects

Project	Problem	Solution	Implementation
Social Health	Want to help people with rare illnesses find help on the best treatments.	Makinga website to do this.	Using Go and ReactJS.
Slack Correct	Wanted to allow autocorrections in Slack.	Built a listener to slack channels that edits messages upon typos.	Nodejs server with WS connection with Slack.
Chatline	Wanted to make a chat service with chatrooms and history kept through email.	A chat website that makes chatrooms on demand, invites others and emails you the history on completion with no other history kept.	Nodejs and AngularJS.
Product Tribe	Wanted a service that recommended possibly intersting Product Hunt posts.	Suggest posts daily based on comparison of your liked posts to others.	Oauth, email, Nodejs, postgresql.
githook	Need a way to easily copy and run all server programs after modifying and pushing them.	Server that listens for githooks and runs deployment scripts.	Nodejs with sys.exec and githooks.
Algorithms and Data Structures	Want to practice basic tools and need motivation.	Implement a data structure or an algorithm on a daily basis.	Implemented in language of my choice.
Webserver	Needed a way to share projects with friends.	Set up website.	Uses nginx and nodejs.
2048	Friends birthday and I wanted to learn JavaScript	Implemented 2048 using friend's pictures instead of numbers.	Implemented in javascript.
Parallel Sort	Desire to make sorting in Java faster.	Implement parallel sort with optimal algorithms for inputs of size 1 through sizes too big to fit into memory.	Implemented sorting with optimal number of threads for given processor using Fork/Join, with a combination mergesort, heapsort, dual-pivot quicksort, and insertion sort.
Simple machine	Learning lisp and machine architecture.	Implemented simple processor with logic gates and pieces created from them.	Implemented processor simulator with lisp, simulating modular state machines passing signals.
poker card game	Learning Object Oriented Programming.	Implement cards, hands, game, etc.	Used python and created card games like 5 card draw and blackjack.
Textual calendar	Learning to program and my father challenged me.	A calendar made with ascii character is shown when given a month and year.	Used python and various conditionals to create a properly formatted calendar.

Automatic texting	My mother wanted me to text her everyday.	I implemented a shell script to text her everyday at varying times with varying messages.	Used mutt, chron, and awk.
Networking	Want wifi in my dorm room.	Set up wifi after flashing a dd-wrt installation.	WPA-Enterprise with radius server
asm interpreter	Learning lisp.	Implemented symbolic interpretation which evolved into an assembly language interpreter.	Used various lisp features to implement.
Excel Android	I noticed teachers spent a lot of time grading assignments while at sports games.	Made an excel-like programming designed to be a grading calculator.	Eclipse plugin made UI simple and used mysql for data. Ease grading process, provide statistics, and keep all grades.
Math PL	I wanted to interest a friend in programming.	Implemented a programming language that allowed the creation and usage of funcions.	Used python and a frame system to define scope.

Hackathons

I love these events. I see a lot of passion and creativity from the participants.

Project	Problem	Solution	Implementation
Hack Rice	The Rice service for class decisions is bad.	We made a website that would show aggregated ratings of professors and classes, and show class time conflicts.	Python flask was used for the backend while AngularJs, JQuery, and bootstrap were used for the frontend.
HackTx	I wanted to learn Android and make an app kids could play with.	An application was built that allowed timeline images to be built by associating sounds with parts of the image.	The app was implemented on Android and the backend was built on Ruby on Rails.
HackTx	We wanted a better way to organize ideas.	An application that represents tasks as bubbles of diferrent sizes.	The app was implemented with libgdx.

CS 343H – Artificial Intelligence

Project based class that practiced the concepts through pacman variants.

Project	Problem	Solution	Implementation
Search	Finding optimal or near optimal paths through a pacman grid.	Used various "complete" search algorithms, designed heuristics, and used greedy search for large graphs.	DFS, BFS, UCS, A-star, admissible heuristics.
Multiagent	Playing optimally against a competitor (pacman ghosts).	Created agents to consider what the competing agents would do.	MinMax, AlphaBeta, ReflexAgents, Evaluation functions.
Reinforcement Learning	Pacman needs to understand learn what decisions are bad or good.	Find optimal paths through trial and error or annealing.	Value Iteration and [approximate] Q learning.
Tracking	Locate and eat/avoid ghosts through noisy sensors.	Build inference over time to locate ghosts.	Exact and Approximate Inference, Dynamic Bayes nets, Particle filters.
Classification	OCR and train Pacman to study better pacmen to learn their strategies.	Classify important and unimportant traits in training set to provide accurate classification of test set.	Perceptron classifier, a large-margin (MIRA) classifier, and a slightly modified perceptron classifier for behavioral cloning

Final Contest	A competition among the students that built on what we learned. (I made it to the finals)	Allow ghosts to find eat pacman and pacman to avoid ghosts and eat pellets in a competitve pacman variant.	Used techniques learned throughout class.

CS 314H – Data Structures

Test driven class on general data structures and obtaining more experience programming.

Project	Problem	Solution	Implementation
Web Crawler	Implement web search engine.	Implemented crawler to index pages and efficient search engine and query builder.	Used Java, anonymous classes, serialization, etc.
Treap	Implement a binary tree structure with an average logarithmic height.	Created a combination tree and heap structure	Involved requiring the structure follow binary tree definition on values and heap definition on randomly assigned priorities.
Boggle	Implement a Boggle game and compute all words on the given board.	Created Boggle game and algorithm to find all words from a dictionary on the board.	Used Java Swing and a trie data structure.
Tetris	Implement a Tetris game and artificial intelligence.	Created a graphical game and competitive AI to increase difficulty.	Created game through use of Java AWT and singletons. Improved artificial intelligence through genetic algorithm.
Critter	Create critter game with autonomous competitive creatures.	Created interpreter for assembly-like language to define behavior of critters and created critter definition for competition.	Simple interpreter and transpiler were created..
Random Writer	Implement monte-carlo chain and text parsing.	Created parser and generated various, statistically similar text files.	Monte-carlo done through graph of nodes with weights.
Image Manipulator	Implement image manipulation such as removing red or rotating.	Rotate, change color, change size, and reflect images.	Read in file as pixels and created new image through required operations.

CS 439H – Computer Operating Systems

Class focused on implementing key abstractions and memory management systems similar to use in modern operating systems.

Project	Problem	Solution	Implementation
Transactional Files	I wanted a way to modify files concurrently.	Implement a transactional filesystem with merging or commit failures.	Keep track of last commit time to discover if a file has been modified since the transaction start.
Shell	A shell would be very useful.	Implement a shell to execute programs.	Text parsing and ‘execv‘ to execute an ELF file while keeping all file descriptors.
Security	Our system isn’t secure and is inefficient.	Implement reference counting and argument checking.	Ensured all file descriptors were the expected type and available to the process.
FAT	We need a filesystem.	Implemented a FAT based filesystem.	Implemented logic to correctly interpret the file system structure.
System Calls	We need a way for user code to interface with the kernel.	Implement system calls to access resources and execute privileged code.	Used interrupts to switch to kernel mode and execute required code.

ELF	We need a way to execute programs.	Implement ELF file parsing and 'exec'.	Parse file based on ELF format. Clear all memory and then set up based on ELF specification.
Virtual Memory	All processes are sharing the same memory.	Implement virtual memory, and mapping to physical addresses.	Handled page faults by mapping in a new physical frame.
Idle	The OS isn't prepared for if no process is ready to run	Introduce an idle process.	Run an idle process when nothing else is ready to run, and switch to a ready process as soon as possible.
Concurrency Control	Kernel's data structures can be corrupted if processes switch at the wrong time	Prevent interrupts in small intervals when necessary	Used disable and enabling interrupts to protect critical sections.
Threads	Implement multiple threads and allow for concurrency.	Created threads similar to Java and implemented semaphores along with buffers for uses such as piping processes.	Threads had to be initialized with certain stack data. Buffers were checked for race conditions.
Malloc	Implement malloc and free with several strategies and compare performance.	Implemented malloc and free through linked-list of coalescing free blocks various malloc strategies.	Used C and various sanity tests to ensure no memory leaks or semantic errors were encountered.
Windows	The VGA interface should not be directly used.	Implement windows that allow applications to draw on screen within proper bounds.	Created operating system level abstraction for screens.

CS 429H – Computer Systems and Architecture

Class focused on basic computer architecture knowledge and assembly instructions.

Project	**Problem**	**Solution**	**Implementation**
Perf Lab	Improve performance of image manipulation features.	Improved memory and cpu efficiency.	Used cache aware loops and minimized necessary calculations.
Malloc	Create memory management.	Implemented malloc and free for a variable size heap.	Created in C with features such as coalescing and an expanding size limit.
Pipe Lab	Speed up code through pipelining code.	Implemented pipelining and an example use of the code.	Used Hardware Control Language (HCL) to create pipelining sppecifications.
Data Path Lab	Implemented additional assembly instructions through a CPU hardware emulator.	Implemented specified additional instructions.	Used Hardware Control Language (HCL) to implement features feasible with the given architecture.
Buffer Bomb	We needed to execute pieces of code at incorrect times while defeating buffer overflow security features.	Execution flow of a program was changed through buffer overflow attacks.	Dis-assembly of code, structuring assembly instructions with hexadecimal input and inputting data and correctly positioning the code on a stack.
Binary Bomb	We had to diffuse a bomb. Incorrect inputs led to it exploding and us losing points.	Proper inputs were found for 7 stages.	Proper inputs were found through dis-assembly of the code and correct interpretation of the instructions and data.
Assembly Lab	We needed to learn assembly.	We implemented linked lists, iterative and recursive traversing, and memcpy in assembly.	Only assembly was used for the previously stated operations.
Bit Lab	Bit manipulations can be faster than using other instructions.	We implemented various operations without the use of loops and other features.	Only bitwise operations and possibly if statements were used and with the fewest number of instructions we could think of.