**Final Project Executive Summary: FDIT**

Ethan Weber

LDS Business College

IT 295 – Capstone Project

**Project Overview**

My capstone project was titled “FDIT,” an abbreviated form of its full name of “Family Dog Information Tracker.” The objective of this assignment was to build a closed system for a family to be able to track their newly adopted pets activities and as a result grow more accustomed to the needs and habits of their pets quickly and with less wasted communication of asking each member of the family when the dog was last fed, watered, walked, etc.

It was for this task FDIT was conceived and designed. FDIT is a system containing two key components. The first of these components is the central dashboard device, a small device which allows you to record the latest happenings of your pet at the press of a button. When these buttons, each containing an action, are pressed it logs those events to a database of events with accompanying information such as date and timestamp to enable any member of the family to quickly glance at the device and see the pets most recent activity.

The second key component to this system was the FDIT companion app. A lightweight application which would allow you to check the pet’s latest activity when you were away from home, say for example, considering whether to buy more treats or food and wishing to see how much your new pet is consuming on an hourly basis.

A simplified diagram of how this system works is pictured following.

A screenshot of a cell phone

Description generated with very high confidence

**Milestone Review**

The final deliverable of the FDIT project was intended to be the complete FDIT system consisting of the previously mentioned two key components, the FDIT software running on a small central machine along with the FDIT companion application (SQLio.) This section contains a list of FDIT projects set milestones along with details. Following each milestone, I will give a brief reflection on the state of the project at the corresponding points in time as well as describing both victories and challenges which came while trying to achieve milestones.

**April 23rd – May 7th**

Central computing unit created – Central computing unit has been assembled and is running a LAMP stack as well as communicating with the local network.

This milestone was achieved at the designated time of May 7th. It was accomplished by first selecting the platforms I would choose to build the project on, determining the overall direction of the project, and finally gathering the necessary resources to assemble the hardware components of what would be needed for the central computing unit. The result of this milestone was a raspberry pi computer with a touch screen hooked up to the devices GPIO pins. The raspberry pi then had Raspbian Linux installed and a wi-fi adapter hooked to it to be able to connect wirelessly.

One of the challenges that was run into during this stage of product development was dealing with the GPIO touch screen and finding relevant drivers. The manufacturer I had purchased the part from at a discounted price had a dead link listed as their drivers. After much research I was able to find an alternative driver being provided to purchasers of the same product from another vendor and was able to get the screen up and running.

One of the best features implemented at this point of time was to install VNC and SSH clients on my development devices and the raspberry pi enabling me to develop software for the raspberry pi using a “headless” configuration with my development machine acting as the monitor.

**May 7th – May 28th**

Family Dog Info Tracker (FDIT) created – FDIT has been written and compiled and is minimally operational with recording data and storing it to a database after key actions.

The accomplishing of this milestone began with research into what language would be easiest to develop for on the raspberry pi while still fulfilling my projects requirements to have the software both provide a touch screen compatible GUI to the user as well as work with SQLite databases.

After research I determined that my best route was to attempt to write the software in Python using the TKinter GUI framework to provide the UI elements. I determined this because this would not only be able to transfer easily between my development machine and raspberry pi, but also ran on multiple devices and could be tested on either.

After quite a bit of work I was able to get FDIT up and running. The software featured a 3 “panel,” or “screen,” layout which was able to be navigated via touch. On the main screen of FDIT buttons are featured which, when pressed, log a timestamped event to an SQL database on the machine which is also displayed in text form on the FDIT main interface. This made it easy for users to touch the screen to record events as well as quickly glance at the screen to see recent events. Full features and functionality will be detailed later in this document, this milestone was also completed by its target date.

**May 28th – June 11th**

Central computing unit running FDIT – Central computing unit is now successfully running FDIT and is recording data to the database, database of information is accessible to other computers within internal network.

The milestone above was one of deployment and testing of the FDIT software. During this milestone several challenges were faced and overcome with deploying FDIT on the raspberry pi itself. One of the first challenges that was encountered was attempting to have the FDIT python application launch itself on device startup on the correct display (meaning the touch screen.) There wasn’t much of an issue getting the software to run on startup, but the bigger issue was coming with the fact that if run on a normal startup the FDIT software was actually displaying to an invisible “virtual” display which couldn’t be interacted with on the touch screen. To resolve this issue a file in /etc/xdg/autostart called “FDIT.desktop” was created to launch a shell script on startup. The shell script would then launch FDIT.py using python3 on the proper display and would display full screen.

The next obstacle during this milestone was to determine how the database software would be served to the companion application which was to be created in the next step. To solve this problem apache was installed on the raspberry pi and FDIT was configured to write its database to the externally accessible /var/www/html/ directory where it could then be downloaded from. This milestone was also completed by its set date.

**June 11th - July 2nd**

FDIT Companion App created – Basic smart device application capable of accessing and displaying recently recorded activity from central computing unit has been created.

The FDIT companion app milestone was the next task to be done to get FDIT up and running and proved to also serve as biggest challenge to finishing FDIT. Development start with the set up and installation of Android Studio. Development was begun by designing a basic UI which proved more difficult than at first thought due to a new change in how Android layouts are designed referred to as “Data Binding,” which allows you to program your application to be able to interact with the UI. After much research Data binding was learned and interactions within the apps UI were able to be programmed.

A Java class was then written to be able to handle saved information between application sessions such as remote download location and the name of the database file.

The next obstacle which presented itself was working with an externally downloaded SQLite database with Androids built in SQLite accessing classes / packages. After trial, error, and research a method of opening external databases was found and implemented. Another java class was written to store my SQL handling code and SQL queries which were called from the main class.

The final obstacle which presented itself in the development of the FDIT app was syncing the database. This was due to two issues, the first being that there are two types of storage on android, internal and external, which were hard to unify and have the database use only one or the other, and the other problem being that when synced the original database wouldn’t move and would result in a ton of downloads while still only opening your first downloaded file which had the proper name. The first of these issues was fixed by changing save and open functions to use only the internal memory. The second of these issues was fixed by clearing the previous database file from memory with every new synchronization of the database. This milestone was begun at the set date but was not actually completed until during time set aside for the next milestone due to the introduction of so many issues and a lot of time spent debugging.

**July 2nd – July 23rd**

FDIT System Operational – Central computing unit, FDIT software, and FDIT companion app all minimally functioning, and integrated with, one another.

This milestone was the final one and involved mainly bug testing, code cleanup, fixing remaining issues, and polishing anything which could be. During this milestone much of the time was spent attempting to see why various aspects of the two different software applications were conflicting with each other and fixing issues to be able to have a complete system. Much of the code for both application was cleaned up and commented and some of it was refined to give nicer looking output (such as string formatting for the companion app.) By the end of this milestone FDIT was up and functioning. Buttons could be pressed on FDIT on the raspberry pi which would update the database, the database could then be synced and displayed to reflect those updates in the application.

**FDIT’s Current State**

This section has been set aside to review the product as it currently is as well as provide some screenshots of the software while also describing some of its features which were note fully described previously. FDIT is currently written, compiled, and up and running. The FDIT application will now run on the raspberry pi where it will keep track of timestamped events and sync with the companion app smartphone application named “SQLio.”

Pictured below are some screenshots of the FDIT application running on the raspberry pi and displaying to the touchscreen:

FDIT Splash Screen:

A screenshot of a cell phone

Description generated with very high confidence

FDIT Settings Screen:

A screenshot of a cell phone

Description generated with very high confidence

FDIT Main Display:

A picture containing indoor, table, sky, sitting

Description generated with high confidence

One helpful feature I was able to implement into FDIT was that to create the main display, set the number of buttons, and set recordable actions within the application is all done dynamically at run time through an “INI” save file within the applications same directory known as “FDITsave.ini” Through this save file users of the system could rename any of the buttons and display anywhere from 1 to 8 buttons on the screen at a time. Changing this save file also changes the title for the “event” stamp put into the SQLite database. An example of this save file configuration is pictured below.

A screenshot of a cell phone

Description generated with very high confidence

**SQLio Companion App:**

SQLio Splash Screen:

**A screenshot of a cell phone

Description generated with very high confidence**

SQLio Settings:

**A screenshot of a cell phone

Description generated with very high confidence**

SQLio Database Viewer:

**A screenshot of a cell phone

Description generated with very high confidence**

SQLio was designed to work with more than just my custom FDIT.db database which is why the custom file name feature in settings was implemented. Another feature of SQLio is that it allows you to change the pointer for where your database file is stored just in case you wish to migrate the system to another network.

**FDIT Unit:**

**A picture containing electronics

Description generated with high confidence**

**Moving Forward**

I’m not sure what lies in FDIT’s future. I’ve contemplated a couple of directions I may like to go forward with it, the primary one being to design a more robust and polished version of what I have and potentially market it as a rentable feature for first time dog owners for which it would provide an inexpensive solution to a common problem. Another avenue I may explore is continuing the development and polishing of SQLio and launching it to fill niche needs of viewing a constantly refreshing remote SQLite databases.

**Reflection**

When I started this project, I had hoped to receive a greater knowledge of the many systems I would be working with, particularly programming using Java, Python, and SQL. Fortunately, a greater knowledge of all these systems and an affirmation that I can put something like this together myself is something I gained throughout FDIT’s development process.

There were many points where development was frustrating, especially as difficult surprises were encountered. Often retired to sleep frustrated that I was unable to solve the problem that night with the only thought in my head being that I would have to try for a couple hours on the next day. However, given the chance to choose to do this again I would because the learning experience for the different systems I’ve worked with has been invaluable.

Though I am unsure of how I might proceed with the project I thoroughly enjoyed the journey and am proud of what I was able to accomplish in a relatively short amount of time. I am also satisfied with how much I have learned about the ins and outs of the many technologies which made this project possible including Linux, Python, Java, SQL, and Android.