RevMetrix Project

Detailed Proposal and Design

Spring 2024

Theo Bloomquist, Luke Dodson, Michael Hensel, Liz Mains, Bryce Neptune, Victor Pineda, Enoch Sam, Rob Wood

Table of Contents

Project Teams and Deliverables

Backend

Liz Mains, Victor Pineda, Rob Wood

Databases

Research Database

The backend team will be developing the Research database, similar to the User database, in order to allow for interactions and data transfer to the Research Database UI. The purpose of the database is to allow Professor Hake access to the data needed to conduct his Ph.D. research on RevMetrix. There will be a variety of tables that need to be created in the Research database, such as: Shot, Video, Ball, Light, ddx, ddy, ddz, etc. Each of these tables will be similar to those from the User database, but there will be some unique tables required for the database to be considered complete.

The implementation of the Research database is crucial as it will be the main data provided to the Research Database UI. Because of this, the UI will need to make calls to endpoints tied to the Research database in order to retrieve this data. There will be a plethora of necessary endpoints, most of which will consist of simple Post, Get, and Patch operations, with the majority of the endpoints being Post and Get operations as the main point of the database is to retrieve information from the mobile applications and store it and to provide the data to the UI.

User Database

The User database handles all user information for the RevMetrix Application including login credentials and access to specific information. This information contains many statistics including sessions, games, shots, etc., along with username and password. The User database will need more endpoints, Gets, and Deletes in that order. Get endpoints will be added to efficiently query the database from basic to more complex searches. Basic searches include getting an event from the location of the event, while complex searches include “Histograms of Scores, distribution of 180’s, 190’s, 200’s, 210’s, 220s, etc., for a given Event or date range” [L]. As new data is being added to the User database, Delete endpoints will be added with the purpose of deleting users, events, etc. as new endpoints are created.

Testing

Both databases have testing framework skeletons that will be used throughout the semester, and although the User database has some partially implemented Unit and Integration tests, both the User and Research databases will need a large amount of tests implemented.

Research Database Endpoint Implementation

The Research Database UI will need to implement the ability to request and retrieve data from the Research database through the usage of API endpoints. Thankfully, this has been partially implemented into the mobile application already for the User database, so there is a

skeleton to work with when undergoing the development and testing of this functionality in the UI. Though the endpoints will need to be hit from the application, meaning the requests will be created using Javascript, the endpoints are solely URLs which means that they can be hit from any language that supports curl or HTTPRequest commands.

Research Database UI

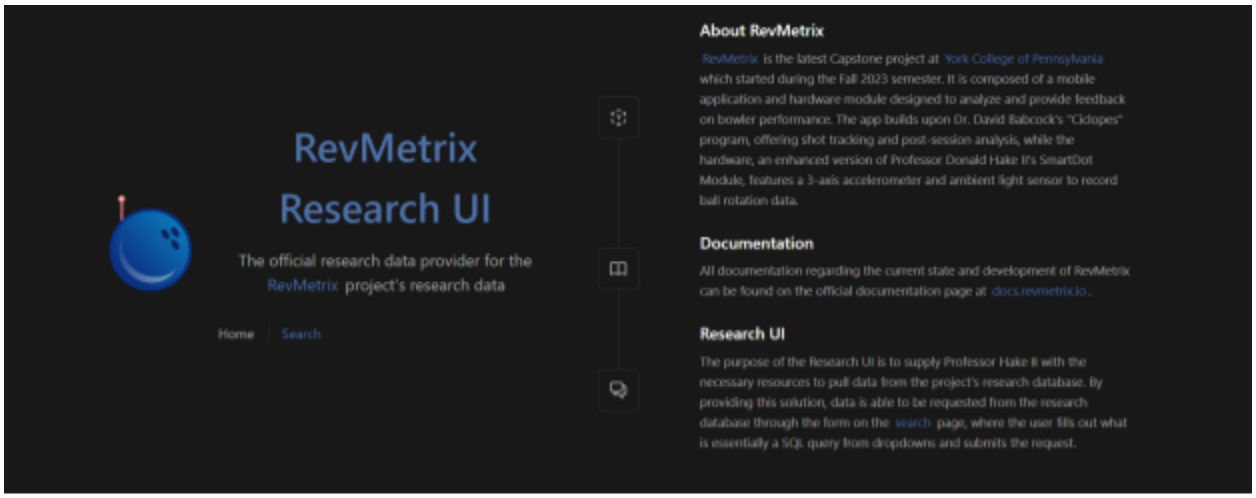

Figure 1.0: Research UI Landing Page

The Research Database UI is what Professor Hake will use to pull specific data from the Research database for his Ph.D. research. The application will consist of a landing page, a search page, and a results page. The landing page, seen above in Figure 1.0, contains information about the project, how to use the web app, and where to find documentation about the project. From here, the user will be able to click the search tab, which will open a form consisting of dropdown boxes, essentially building a user-friendly SQL query based on what the user wants to request from the database. After submitting the form, the application will then send a request to the desired endpoint for the Research database, which will then return the requested information back to the user and display the results on the page.

Ciclopes

Theo Bloomquist, Bryce Neptune, Victor Pineda

Ciclopes

Using a simulated Unity Camera, the simulation team will generate videos, parsed into individual frames. These frame images will then be filtered and tagged for pattern recognition, to identify the ball in each frame. This will be done for each frame and then based on the time between each frame, velocity can be calculated, and then a piecewise curve will be generated between each image to represent the ball path. From these metrics, the program will then calculate the arrow in which the ball crosses, the board in which the curve is tangent to the lane, and the angle of entry into the pins.

Ciclopes Tower

A standalone web page will be hosted on the droplet, graphically displaying a bowling lane and the ballpath derived from Ciclopes. The ballpath displayed will be the most recent thrown ball that is stored in the Backend. Included on this web page there will be fields displaying calculated metrics and user input fields that allow the user to select a previous shot to display.

Mobile Application

Theo Bloomquist, Luke Dodson, Michael Hensel, Liz Mains

The mobile application will require several updates to refine previous work and add other necessary features. Since the app is mostly still in a skeletal form, many pages still need to be added and fleshed out with necessary features and updated UI.

BLE Client

To better replicate the functionality of Professor Hake's original SmartDot, further functionality for the MbientLabs MetaMotion S module must be developed. Currently, the RevMetrix app can communicate with the MMS module to record and read accelerometer data. To fully mimic the SmartDot, the app must also be able to command the MMS to collect gyroscope and ambient light data. Obtaining this functionality will involve constructing the necessary commands to configure and start the MMS module's gyroscope and ambient light sensor. Once this data can be gathered, it must be processed on the app to convert the raw data into meaningful values. Once the accelerometer, gyroscope, and ambient light sensor can collect data individually, commands must be configured to activate all three at the same time. The data collected from this fusion recording must then be processed to separate all of the values. This process will involve reverse engineering byte array commands based off of the deprecated MbientLab MetaMotion C# API.

Additionally, similar work will be done using the Bosch Application board. This board contains the same model of the accelerometer used in the MMS module that will also be used in the new SmartDot. Establishing basic communication with this board from the RevMetrix app will provide the basis for transitioning from communicating with the MMS module to communicating with the Bosch board once the SmartDot is developed.

UI/UX

The mobile app will feature updated screens and functionality that allow users to save games, balls, notes, and more. There will be a screen for gameplay that allows users to record and annotate each frame shot, keeping score as they progress through their games. Their notes will be able to be viewed through the Notes page, or as they flip back through their previous frames. The Notes page will be a collection of users notes about their shots, balls, stats, and any general notes the user would want to save. The Ball Arsenal screen will also be updated to allow users to add more detailed information about their ball as well as connect them to a SmartDot module. There will also be a Stats page created where users will be able to see their

stats averaged out over the course of their games and practices to help highlight patterns of success and areas in need of improvement. With these changes the user will have a more intuitive and seamless experience in saving their games and viewing their stats.

Additionally, UI improvements will need to be made for the data retrieved from the MMS. Currently, the data is only displayed as text. However, to provide a better user experience, the data will be displayed graphically, showing changes in the sensor's acceleration, orientation, and light levels. This graphical display will be a series of line graphs displaying the sensor data over the time the module is active.

Simulation

Theo Bloomquist, Luke Dodson, Michael Hensel, Bryce Neptune, Enoch Sam

BLE Server

There will also be a BLE Gatt server on a separate piece of hardware. The Gatt server will be run on a Raspberry Pi or a desktop application. It will mimic the capabilities of the SmartDot and be able to transmit data to the mobile app. This server will allow testing for the BLE client used by the mobile app. It will also connect to the visualizer and SmartDot simulator to create a smooth end-to-end data transfer.

SmartDot Simulator

The SmartDot simulator is a C# program that is made to simulate the data output of the real SmartDot module that will be built in the future. The program will read the test data from a file that was generated either by the Unity Visualizer or from real data. It will then compress each data entry, and then export it as a byte array. This byte array will then be stored for use in the mobile application [K].

Scoring UI

A scoring UI will be created for the bowler and program user. This UI will display shots, frames, and a game score. Additionally, a ten-pin setup will make up a majority of the interface with the scoring elements smaller. This ten pin display can be toggled to set the pins remaining after each shot. After setting the pins, a button can be clicked to save the shot. This populates the score into the first shot of the current frame. This can be repeated for shots, and then incremented to the next frames. As frames are saved, the total score will be updated. Buttons for fouls, gutterballs, strikes and spares will be present for knocking none or all of the pins.

Visualizer

A Unity simulation to visualize ball throws will be completed with capabilities of straight, angled, and arcing shots. This simulation will represent gutters, pins, a lane, and pins with a pinsetter. This allows for all manner of shots to be simulated and recorded using our simulated camera software. These recordings can later feed into the Ciclopes simulation. The visualizer will represent ball metrics such as linear velocity, angular velocity, linear acceleration, light

levels, ball loft, and ball bounce. These metrics are generated in real-time and can be saved to a file with an associated roll video/frames, and shot parameters. Any shot taken can be replayed with the same shot parameters to allow roll replication.

Visualizer (Virtual Reality)

An associated Virtual Reality executable will be generated from the visualizer. This program will look similar to the visualizer, however, will have a cleaned-up UI and function more like a game. This VR program will allow users to roll a ball down the lane adjusting things such as shot type (straight, angled, arced), shot position, arc apex, speed, and loft. This software will run on Virtual Reality hardware and be a main attraction for the capstone expo to generate RevMetrix hype.

Current State of RevMetrix

High-Level System Architecture

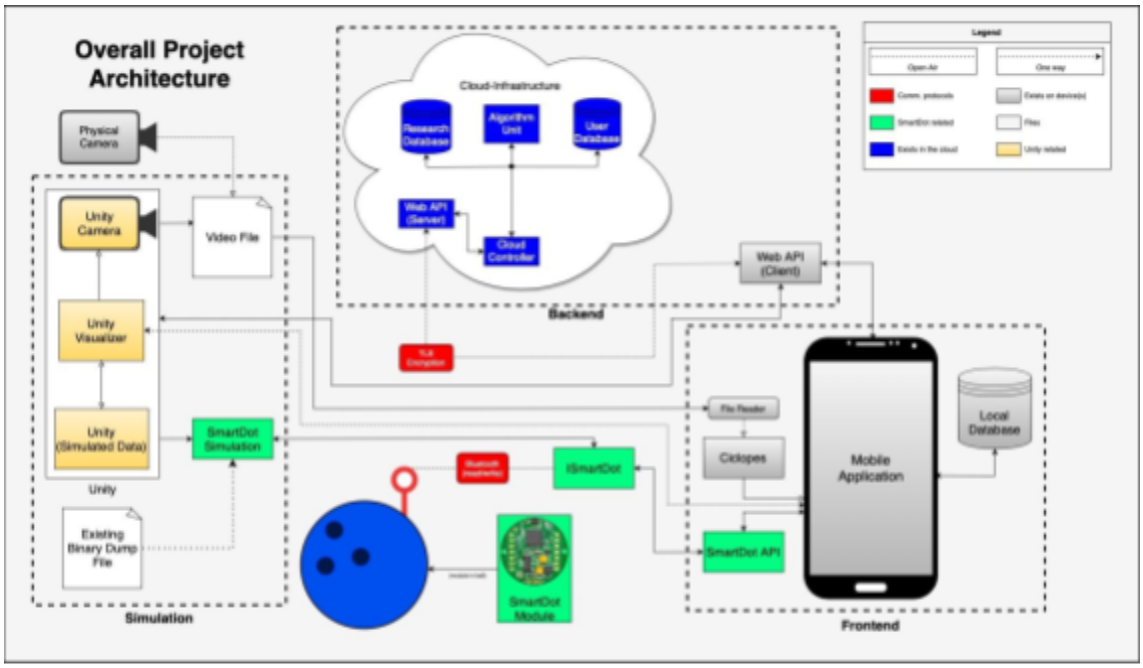

Figure 2.0: Overall System Architecture

As seen above in Figure 2.0, the three main components that makeup RevMetrix's current foundation are the Simulation, Frontend, and Backend components. The Simulation consists of the Unity Camera, Visualizer, and Simulated Data, along with the SmartDot Simulator and the video files created by either a physical camera or the Unity Camera. The Frontend is composed of a FileReader, Ciclopes, SmartDot API, and a local database that are all tied together and implemented into the Mobile Application. The Backend is composed of the DigitalOcean droplet server, which contains the Web API Server interface, Cloud Controller, Research and User databases, and the Algorithm Unit. The Backend also contains the Web API Client interface as it is managed and maintained by the backend team.

Backend

Liz Mains, Victor Pineda, Rob Wood

Databases

The backend consists of two databases, which are the User database and the Research database. Currently, only the User database has partial functionality implemented into the mobile application and the ability to provide data based on API endpoint calls. The Research

database has not yet been implemented, though it will be throughout the Spring 2024 semester in light of the development of the new Research Database UI.

DigitalOcean Droplet

Figure 2.1: RevMetrix DigitalOcean Droplet Server Graphs Page

Along with two databases, the backend also manages and maintains the team's cloud server, referred to as a “droplet,” which is hosted on DigitalOcean as seen above in Figure 2.1. The droplet handles the 24/7 uptime for the NGINX Proxy Manager instance, Research Database UI, SQL Server databases, Web API, and Wiki documentation site. All requests made to any of the services listed above go through the droplet to the respective application, and each domain tied to the project is tied to the IP address of the droplet.

Web API

Finally, arguably the most important part of the backend is the Web API. The Web API consists of two interfaces:

- Web API Client:
    - Other applications in RevMetrix include the API Client interface in order to make requests to send or receive data from the User or Research databases.
- Web API Server:
    - The API Server interface is its own ASP.NET solution and runs on the droplet in order to process requests for data made by other applications in RevMetrix.

Without the Web API, the databases would not be able to be accessed by any of the companion applications of RevMetrix, resulting in a tremendous lack of functionality throughout the project. As of now, the Web API currently has the ability to send and receive data to and from the User database and the project’s companion applications, but lacks the ability to do so with the Research database. Throughout the semester, new API endpoints will be created for the Research database in order to provide this functionality for the development of the Research Database UI.

Ciclopes

Theo Bloomquist, Bryce Neptune, Victor Pineda

Ciclopes was originally created by Dr. Babcock with the purpose of tracking a bowling ball's path as it travels down a lane. This system used a camera to record video of the bowling ball and computer vision algorithms to process the video into useful information. Dr. Babcock's codebase is deprecated, but it will serve as an important reference for implementing the functionality of Ciclopes.

Mobile Application

Theo Bloomquist, Luke Dodson, Michael Hensel, Liz Mains

The Mobile Application was built using a .NET MAUI framework, which provides cross-platform C# functionality, using a single shared code-base. The Mobile Application has a dynamic User Interface, that allows users to select which of the currently built-in functionalities they would like to use. These functionalities are all related to the requirements of the RevMetrix project, such as: uploading video files of the user bowling, creation and management of a user account, processing of user data, and communication with other aspects of the project. This communication is implemented through BLE connection, to communicate with the SmartDot module, and an API, to communicate with the Backend droplet [K].

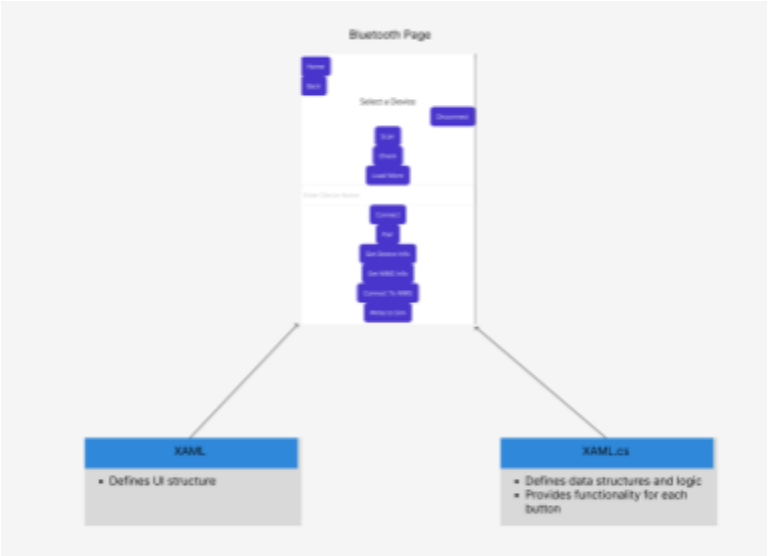

Figure 2.2: Frontend Model-View View-Model Diagram

Figure 2.2 describes the design structure of the frontend application. Each page on the mobile application uses a single-class, Model-View-ViewModel [14], structure that includes an Xaml file and an Xaml.cs file. The Xaml file organizes the visual elements of each page, such as styling and the design of the page. The Xaml.cs page is the logic behind each page and is where the functionality is derived from. For every interactive visual element in the Xaml file,

there is a corresponding function performing the logic in the Xaml.cs file. More work will be performed in the Xaml files to provide a more detailed and engaging user experience [K].

The mobile application and GATT server application are equipped with Bluetooth Low Energy (BLE) functionality to allow communication between devices such as the SmartDot and its substitutes. BLE functions for the mobile application are provided via the dotnet-bluetooth-le Nuget package [K]. The GATT server application will incorporate 2 or more services to provide and receive data from the mobile app. The first service will contain information about the server application itself, as is customary for BLE communication and the subsequent services will hold characteristics for the mobile app to communicate with. Figure 2.3 below demonstrates how the relationship between the mobile application (GATT client) and GATT server application will function.

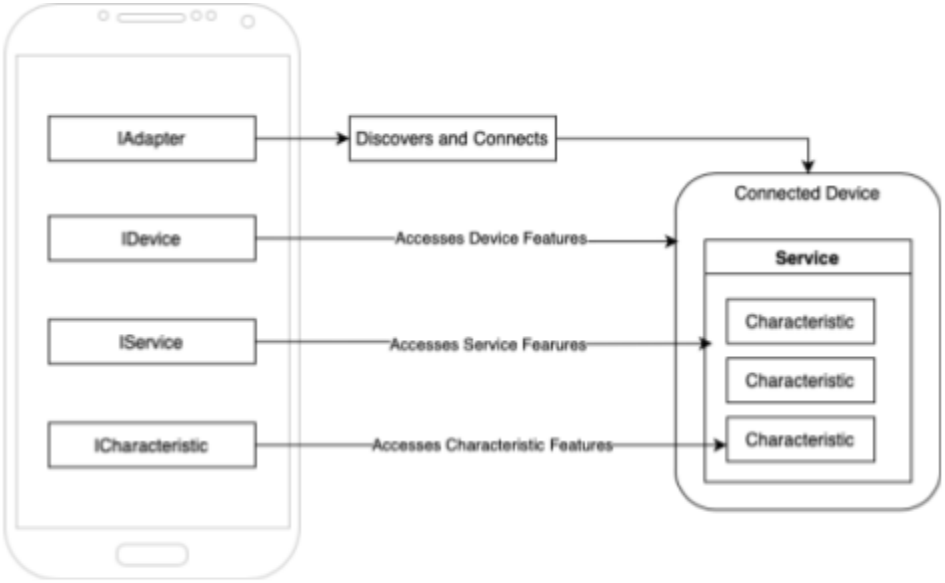

Figure 2.3: BLE Communication using the DotNet Plugin

Visualizer

Theo Bloomquist, Bryce Neptune, Enoch Sam

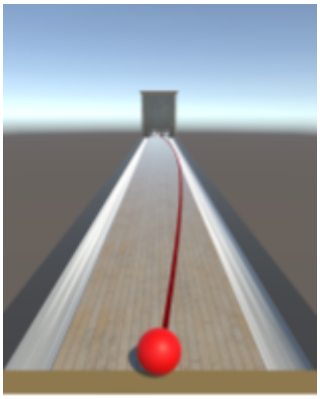

Figure 2.4: Unity Visualizer

A visualizer with lane, gutters, pinsetter, pins, and ball has been created in Unity, as seen in Figure 2.4 to the right. This visualizer allows for shots to be taken either straight, angled, or arced. Initial position can be controlled as well as angle and arc apex location. Most shots move based on an initial velocity and then subsequent friction slowing linear velocity. The one exception to this velocity movement is arcing shots move based on translations and not actual velocity. The visualizer is capable of

recording a shot and outputting an mp4 video of the shot from different angles, this will be useful for Ciclopes testing.

Wiki

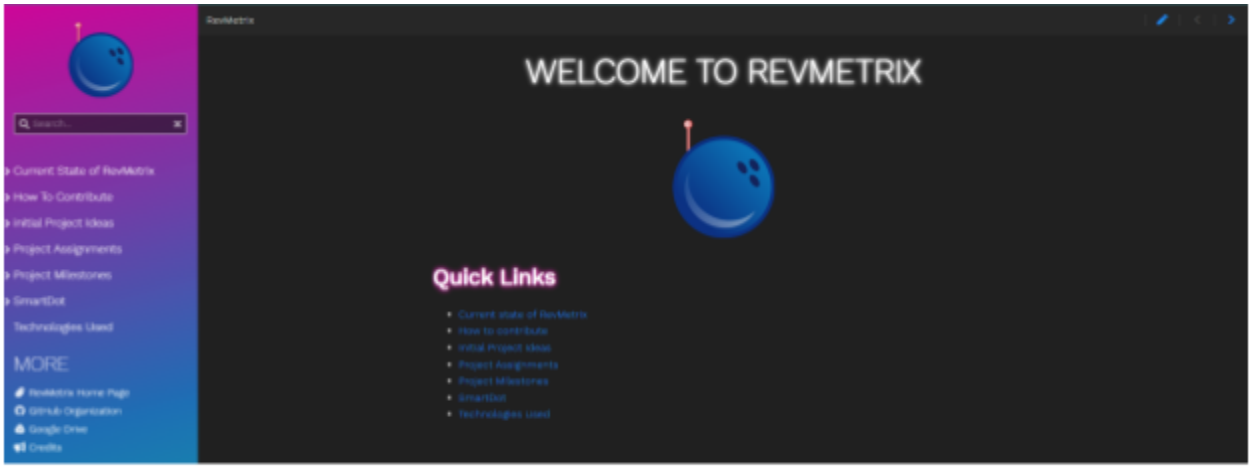

Figure 2.5: RevMetrix Documentation Wiki

Currently, the RevMetrix team has a Wiki documentation site (https://docs.revmetrix.io)[27], seen above in Figure 2.5, that is up to date with any and all progress from the start of the project to the end of the Fall 2023 semester. The wiki holds updated information about the project and is going to be maintained and updated throughout the Spring 2024 semester and (hopefully) the following semesters. On the site, users can find general information regarding RevMetrix, follow the development of the project under “Current State of RevMetrix,” check out the tools that the team used throughout the project, and so forth.

Overall Goal of RevMetrix

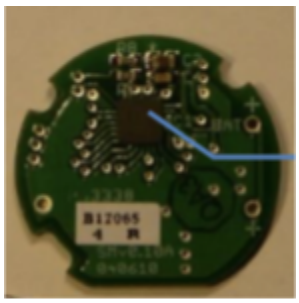

Figure 2.6: SmartDot Module PCB Layout

The RevMetrix project aims to solve the problem of how a bowler is unable to detect environmental changes during a session and therefore is unable to predict how the ball should be shot during the upcoming frames, which results in errors and miscalculations in the bowler’s shots throughout the session. The solution to this problem is to provide the bowler with a complete analysis of their shots through the combinational use of a soon-to-be improved version of both Professor Hake’s SmartDot Module, seen in Figure 2.6, and Dr. Babcock’s

Figure 2.7: Ciclopes Shot Trail

Ciclopes software, seen in Figure 2.7, tied into a unified mobile application. Through the use of these two improved technologies, a bowler can improve the mechanics of their shots by reviewing metrics provided through the application on a per-shot basis or simply reviewing metrics for an overall session to analyze possible changes or inconsistencies in their performance.

Major Technical Challenges

Backend

Liz Mains, Victor Pineda, Rob Wood

- Creating and implementing functional API endpoints for the Research database into both the Server and Client interfaces
- Creation of additional endpoints for the User database for use in the mobile application
- Development of the Research Database UI and implementation of endpoint call functionality into the web application
- Polishing and possible redesign of the NuGet packages provided through GitHub

Ciclopes

Theo Bloomquist, Bryce Neptune, Victor Pineda

- Generating shot videos using visualizer and dividing the video into individual frames.
- Researching Object Detection algorithms and discovering which is the best for our specific set of data
- Creating prototypes from the algorithms researched and finding the algorithm best fit
    - Testing image filtering techniques and finding pattern recognition algorithms that suit a lane with noise, and balls of different colors/patterns
- Test Lane, Ball, & Pin Detection
- Test Edge Detection & Object Segmentation
- Graphically display ball path using Ciclopes Tower
- Transport data gathered by Ciclopes to Ciclopes Tower for graphical display

Mobile Application

Theo Bloomquist, Luke Dodson, Michael Hensel, Liz Mains

- Working around any limitations and/or bugs MAUI may still have
- Reverse engineer commands to communicate with MMS and Bosch sensors
- Graphically display acceleration, gyroscopic, and light data from sensors

Simulation

Theo Bloomquist, Luke Dodson, Michael Hensel, Bryce Neptune, Enoch Sam

- Converting the existing Unity project to a Virtual Reality project, this involves updating the Unity camera used and adjusting overarching project settings. Export this as a separate Virtual Reality executable.
- Updating the visualizer to include light level simulation.
- Updating simulation to transfer linear velocity into angular velocity as the ball gets further

down the lane.
- Update visualization to include ball loft, and a related ball bounce.
- Update visualizer to allow for shot speed parameters being set.
- Create and integrate a user-friendly scoring interface.
- Allow the visualizer to intake shot parameters from file as well as save shot parameters to file and between shots.
- Adding an overhead visualizer camera to better display the scene.
- Create arcing physics to allow arced shots to roll with velocity as opposed to transposition.
- Generate physical lane texturing and create an oil field using coefficient of friction across the lane.
- Create GATT server application to transmit simulator data through
- Create GATT server application for Raspberry Pi
- Implementing GATT server application into the simulator application
    - Transmit simulator data through the GATT servers read characteristic

Wiki

Rob Wood

- Implementation of a login on the Wiki to restrict access to the Swagger test endpoints
- Maintenance and timely updates of content on the Wiki to allow team members to document and reference each other's progress
- Reformatting of folder structure to reflect the Spring 2024 semester progress and, for lack of better words, "archive" that of the Fall 2023 semester

Technologies

Bosch Sensor Boards

The Bosch Application and IMU Shuttle boards will be used in creating the actual SmartDot. The board is fitted with a BMI270 accelerometer, the same hardware used in the MMS [1]. Commands will be developed in the mobile app to communicate with the Bosch sensor to build the foundation of actual SmartDot communication.

ChatGPT 3.5

ChatGPT 3.5 is a text-based AI based on a large language model that can assist users in general troubleshooting or provide information on a topic with ease [24]. RevMetrix uses ChatGPT 3.5 mostly for troubleshooting during development sprints, but has also used it to assist in formatting papers and setting up cloud-server configurations.

DigitalOcean

DigitalOcean is a cloud infrastructure provider through which RevMetrix rents the project's server and deploys, manages, hosts, and scales its applications [2].

Discord

Discord is a communication platform designed for various interest groups that the RevMetrix team uses as the main source of communication outside of the classroom [3]. Here, the team converses about development, potential issues, and requirements of assignments, along with the occasional voice call for working on course assignments.

Docker Engine, Docker Desktop, and Docker Hub

Docker Engine is what RevMetrix uses to create environment containers per application but encapsulating and packaging the applications and dependencies into lightweight, isolated containers [4]. Docker Desktop is an application for MacOS and Windows that provides an interface that makes it simple for users to manage containers running from their machines. Docker Desktop also is used to start the Docker Engine without running any commands from a machine's terminal, making it a one stop shop for most Dockerized applications. Finally, Docker Hub is a website where users can create and manage existing Docker repositories, and where all pushes and pulls using Docker go to get or send the updated container information. Each of these tools plays a crucial role in not only the development and constant uptime of the applications of RevMetrix, but also in the CI/CD of every application of the project as it is as simple as 3 shell commands to push a new container to the cloud and pull it down to the server using GitHub Actions, elaborated on below.

Draw.io

Draw.io is the main program that RevMetrix uses to create, manage, and edit any diagrams related to the project. It is useful due to the above reasons but also because of its features such as seeing editing history and seeing who is editing the diagram in real time [5].

GitHub and GitHub Actions

GitHub is a web-based platform that uses Git for its version control and provides a collaborative environment for software development that allows multiple people to work on projects at the same time [6]. RevMetrix uses GitHub as a sort of “home base” for all of the applications of the project, and the team even has its own verified GitHub Organization. Here, the team is able to take advantage of GitHub Actions, which can be configured to run automatically based on different criteria, though RevMetrix usually configures them to run on every commit to a main or master branch [7]. GitHub Actions allows the team to push a commit to the main branch of a repo, build and push a docker-compose file to Docker Hub, log into the DigitalOcean droplet through ssh, spin down the running docker-compose file for that application, pull the newest container/compose file, and spin the application back up with the updated code in just about 1 minute, all from the click of a single button.

HUGO

Hugo is used as the main framework for the development of the Wiki documentation site. It is a static-site generator that is meant to be used with GitHub pages, but can be configured as a standalone application, much like how RevMetrix has configured it to run on the droplet [8].

JetBrains Rider

Rider is an IDE for .NET development that supports various programming languages, though RevMetrix solely uses it for its easily configurable support for C# [9].

JetBrains WebStorm

WebStorm is an Integrated Development Environment (IDE) for JavaScript, TypeScript, & HTML. With multiple functionalities designed to enhance development capabilities, such as a smart editor, developer tools, Github support and more [10], WebStorm is a popular choice for web development projects.

MAUI (.NET Multi-Platform Application UI Development)

MAUI is a framework for building cross-platform applications with .NET [21]. It allows developers to create a single codebase that can be deployed on different platforms, such as Android, iOS, Windows, and macOS. MAUI requires .NET 6 or higher to run because it is only these versions of .NET that provide the needed libraries required for each specific platform implementation. This includes NET for iOS, NET for Android, NET for macOS, and WinUI3. Each of these libraries has access to a Base Class Library (BCL), that abstracts specific details

of the specific implementations. Mono runtime provides an execution environment for iOS, Android, and MacOS. While .NET CoreCLR provides an execution environment for Windows.

MetaMotion S

The MetaMotion S (MMS) module will be used for testing and simulating SmartDot communication from the mobile app via bluetooth LE. The MMS is fitted with various sensors including an accelerometer, gyroscope, and ambient light sensor [22]. These sensors, primarily the accelerometer, are similar models to those which will be on the actual SmartDot. The use of the MMS allows for accurate testing for SmartDot communication while the actual device is still being developed.

Microsoft .NET Core

.NET Core is an open-source, cross-platform framework and runtime for building cloud-based applications compatible with different types of machines [11]

Microsoft Office 365

Microsoft Office 365 is a suite of cloud-based productivity tools and services such as Excel, Powerpoint, Word, and other tools like Teams [12]. RevMetrix currently uses Excel from this application stack to process and manage different types of data regarding the SmartDot Simulator.

Microsoft SQL Server (Developer Edition)

SQL Server is a database engine designed for development and testing purposes and is what the databases of the RevMetrix backend run off of [13].

Microsoft Visual Studio 2022 (Community Edition)

MS Visual Studio 2022 is an IDE that is widely used to create applications across different platforms and is mainly used by RevMetrix for the development of the Web API [17].

NuGet Package Manager

NuGet Package Manager is a package manager for .NET development that allows developers to easily integrate third-party libraries into their applications [23]. RevMetrix uses NuGet to package and distribute the different interfaces of the Web API for other team members to use in the companion applications.

OpenJDK 17

OpenJDK 17 is a Java Development Kit that is used for the development of the mobile application in order to support Android devices [25].

Raspberry Pi

The GATT server will be developed on a raspberry pi 4. The Pi will be using Raspberry Pi OS (previously known as Raspbian). And the Server will be programmed using Python and the Bluez framework. The Bluez framework is a python framework for working with bluetooth.

SQL Server Data Tools

SQL Server Data Tools is a set of tools for database development within Visual Studio that provides a centralized development environment for deploying schemas and testing database interactions/connections [15].

SQL Server Management Studio (SSMS)

SQL Server Management Studio (SSMS) is an integrated environment for managing SQL Server databases and is used to connect to SQL Server instances and databases that are running/hosted in the cloud [16]. This is primarily how it is used by the RevMetrix team.

Unity and Unity Hub

Unity is a popular cross-platform game development engine. It provides support for the development of both 2D and 3D games and simulations, and supports popular platforms including mobile devices, desktops, and virtual reality along with a C# interface [28][K]. Unity Hub is the main management tool for Unity projects, as it allows for developers to manage the projects efficiently and provides access to the assets store in the event that developers need a little more power [29]. RevMetrix uses Unity as the main development platform for the simulation, and will be attempting to use it for the development of the virtual reality aspect of the simulation this spring.

Visual Studio Intellicode

IntelliCode is an extension for Visual Studio that uses machine learning to assist developers in writing code more efficiently. It provides intelligent code completion suggestions based on patterns learned from existing code [18].

Vue.js

Vue is a JavaScript framework for building interactive user interfaces. It's built on top of HTML, CSS, and Javascript and provides unique functionality that increases the Reactivity & Declarative Rendering capabilities of a project [30][K].

Xamarin

Xamarin is a cross-platform application development framework that allows developers to create native mobile applications for iOS and Android using the C# programming language. Xamarin enables code sharing across different platforms, reducing development effort. Until Net

Maui was released, Xamarin.Forms was the framework used by developers to create cross-platform applications. With the advent of MAUI, Xamarin.Forms have been phased out, however, many Xamarin dependencies are still used for cross platform compilation [19][K].

Interface Requirements

Ciclopes UI

The Ciclopes UI will have its own distinct lane depiction, showing a detailed overhead of the lane including: boards, lane arrows, lane dots, and pin formation. This UI will use graphing to draw a ball path as the image recognition software gives ball coordinates from either a real recording or simulated recording taken from the visualizer. Ciclopes will also have additional side panels to display metrics such as on what board the ball crosses the arrows and dots, as well as the arc apex location and angle of entry of the ball into the pins. The amount of statistics and displayed information from the video parsing is subject to change based on feature creep and the state of technology achieved by implementing different forms of object recognition.

ISmartDot Interface

The ISmartDot interface is a huge project requirement as it will be the main method of reading and writing data to/from the SmartDot Module. Currently able to pass in arbitrary commands, the ISmartDot interface allows for custom transmission protocols which opens up a door for a variety of commands such as read page, read ball page, write page, set defaults, and much more. Once commands are developed to communicate with the Bosch sensor that will be used in the SmartDot, this interface can be further integrated into the mobile application.

Mobile Application UI

The Mobile Application UI is one of the most important interface requirements out of the project. Due to the UI being the centralized data point for each companion application of RevMetrix, the design and development of the mobile app's user interface needs to be moving forward almost all semester. As new features are added to different applications, along with those added to the mobile application, the design may have to change to accommodate for the other changes to function properly. Therefore, the Mobile Application UI is a huge requirement for this semester's development stages of RevMetrix.

Research Database UI

The Research Database UI is going to be a huge requirement this semester as it is needed for Professor Hake to conduct his Ph.D. research. A web application will need to be created and hosted on a subdomain of revmetrix.io in order for Professor Hake to access the UI, and API Client functionality will need to be implemented into the web application in order to provide him with the desired data from the Research database.

Visualizer UI

The visualizer UI was left in a rough and unfinished state as of last semester. This will need to be addressed, likely by adding the scoring interface discussed above. This scoring UI will simply show a frame count, and the score: both per frame, and total score. The scoring UI

will be toggleable in the visualizer but will default to displayed. Additionally, our current screen of keystrokes used for controlling the visualizer will be moved to a separate window allowing the controls to be displayed or hidden with a toggle, defaulting to hidden. This will reduce clutter and allow for more thorough input descriptions. Additionally, this will allow for creation of a separate controls section for the Virtual Reality executable version of the visualizer.

Web API - Client Interface

The Web API Client interface already exists and has partial functionality, allowing the application to send requests for data to and from the User database hosted on the server. However, it will need to be expanded upon as the Research Database UI will need to implement functionality using the Client interface to interact with the Research database.

Web API - Server Interface

The Web API Server interface also exists, with the same amount of functionality, allowing the processing of Client requests and storage/retrieval of data in the User database hosted on the server. This too will need to be expanded upon to provide functionality to do the same with the Research database.

Resources

[A] Discord: RevMetrix (Discord)
[B] Documentation: RevMetrix Wiki
[C] Fall 2023 Course GitHub: CS400 - CS Capstone I (RevMetrix Project)
[D] Spring 2024 Course GitHub: CS402 - CS Capstone II (RevMetrix Project)
[E] GitHub: RevMetrix (github.com)
[F] GitHub Actions: Features - GitHub Actions
[G] Google Drive: RevMetrix (Google Drive)
[H] RevMetrix Thesis Paper: Hake-MEngESci-Masters-Thesis.pdf
[I] RevMetrix Thesis Powerpoint: Extracting Bowling Ball Reaction Data
[J] RevMetrix Research User Interface: Research UI
[K] RevMetrix Project Technical Report: Technical Report
[L] RevMetrix Bowler User Interface: Bowler UI

References

[1] Bosch (2024) Datasheet. Available at: [https://www.mouser.com/datasheet/2/783/bst_bmi270_sf000-2486543.pdf].

[2] DigitalOcean. (2023, Dec. 9). Droplets. Available at: [https://www.digitalocean.com/products/droplets].

[3] Discord. (2023, Oct. 19). Mobile/Desktop application. Retrieved from [https://discord.com/].

[4] Docker. (2023, Oct. 27). Docker Engine. Retrieved from [https://docs.docker.com/engine/].

[5] Draw.io. (2023, Dec. 6). Draw.io Diagrams. Retrieved from [https://app.diagrams.net/].

[6] GitHub. (2023, Dec. 9). YCP-Rev-Metrix. Retrieved from [https://github.com/YCP-Rev-Metrix].

[7] GitHub Actions. (2023, Dec. 8). GitHub Actions Documentation. Retrieved from [https://docs.github.com/en/actions].

[8] HUGO [Static-Site Generator & Framework]. Available at: [https://gohugo.io/].

[9] JetBrains. (2023, Dec. 5). Rider - JetBrains. Retrieved from [https://www.jetbrains.com/rider/].

[10] JetBrains (2024). Meet WebStorm. Available at [https://www.jetbrains.com/help/webstorm/meet-webstorm.html].

[11] Microsoft. (2023, Nov. 9). .NET Core. Retrieved from [https://github.com/dotnet/core/releases].

[12] Microsoft. (2023, Nov. 29). Microsoft Office 365. Retrieved from [https://www.microsoft.com/en-us/microsoft-365].

[13] Microsoft. (2023, Nov. 1). Microsoft SQL Server. Retrieved from [https://www.microsoft.com/en-us/sql-server/].

[14] Microsoft. "MVVM in .NET MAUI Architecture." Available at: [https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm]. Accessed on: Feb. 18, 2024.

[15] Microsoft. (2023, May 8). SQL Server Data Tools. Retrieved from [https://learn.microsoft.com/en-us/sql/ssdt].

[16] Microsoft. (2023, Mar. 31). SQL Server Management Studio. Retrieved from [https://learn.microsoft.com/en-us/sql/ssms].

[17] Microsoft. (2023, Dec. 9). Visual Studio. Retrieved from [https://visualstudio.microsoft.com/].

[18] Microsoft. (2023, Nov. 8). Visual Studio IntelliCode. Retrieved from [https://visualstudio.microsoft.com/services/intellicode/].

[19] Microsoft. (2023, Nov. 10). Xamarin. Retrieved from [https://dotnet.microsoft.com/en-us/apps/xamarin].

[20] Microsoft Corporation. (2023). ASP.NET [Web development framework]. Available: Microsoft, [https://dotnet.microsoft.com/apps/aspnet].

[21] Microsoft Corporation. (2023). .NET MAUI [Software development framework]. Available: Microsoft, [https://dotnet.microsoft.com/apps/maui].

[22] MMS (2024). Description. Available at: [https://mbientlab.com/store/metamotions/].

[23] NuGet [Package Manager]. Available at: [https://www.nuget.org/].

[24] OpenAI. (2023). ChatGPT [Large language model]. Retrieved from [https://chat.openai.com].

[25] OpenJDK. (2022, Mar. 19). OpenJDK 12. Retrieved from [https://openjdk.java.net/projects/jdk/12/].

[26] RevMetrix. (2023). RevMetrix Google Drive [Google Drive folder]. Available: Google Drive, [https://drive.google.com/drive/folders/1Z_TtR7IVxIza5YR7sO4sxDVqkagdBix5?usp=drive_link].

[27] RevMetrix. (2023, Dec. 6). RevMetrix Wiki [Documentation]. Available: RevMetrix, [https://docs.revmetrix.io/].

[28] Unity Technologies. (2023, Jul. 18). Unity. Retrieved from [https://unity.com/].

[29] Unity Technologies. (2023, Nov. 6). Unity Hub. Retrieved from [https://unity.com/unity-hub].

[30] Vue.js (2024). Introduction. Available at: [https://vuejs.org/guide/introduction.html].

[31] YCPCS. (2023). RevMetrix Project [Webpage]. Available: GitHub Pages, [https://ycpcs.github.io/cs400-fall2023/projects/RevMetrix-Project/index.html].

[32] YCPCS. (2024). RevMetrix Project [Webpage]. Available: GitHub Pages, [https://ycpcs.github.io/cs402-spring2024/projects/RevMetrix-Project/index.html].