University of Toronto

ECE 532H1 Digital Systems Design

Project Proposal:

Virtual Vaccine Passport Scanner for Remote Entry Approval

Group #3:

Mustafa Kanchwala

Guoxian Wu

Eduardo Stecca Ortenblad

Xuening Dong

January 28, 2022

1.0 Introduction

It is 2022 and COVID is already 2 years old! These days, your vaccine passport is your key to entering a restaurant, school, office building, airports and almost everywhere else you would like to go. Our project implements an exciting solution where a user can scan their vaccine passport at a remote client terminal near an entry point. Once scanned their information is verified by a server in the cloud using a database, and they are approved to enter if they are vaccinated.

The key advantage to our approach is that no more human interaction is going to be needed throughout the entire process of vaccine checks. Although speedup in the vaccine passport scanning process is not guaranteed for our current implementation on Nexys FPGA, this project can be further improved with more powerful CPU cores to almost certainly guarantee a speedup in the entire process. This is because, unlike the current vaccine passport, our project will be implementing custom cores/IP in hardware both on the QR code decoding and on the database side of things.

Finally, it is worth mentioning that QR codes are widely used for a variety of different purposes. Our project could be easily modified to be used in different application such as warehouse managing, ticket scanning, and product manufacturing. We chose a Vaccine passport solution as it is very relevant to our current scenario.

1.1 Existing Similar Work

(1) Web Server and QR Decoder Applications for Xilinx FPGA Boards [1]

This paper describes a system integrating QR code decoder and web server. It can capture and decode QR code information from a video. Also, the web server process certain HTTP requests.
Pros:

It's a portable hardware system that costs less than conventional PC hardware. The web server is very stable due to its lightweight design it is also capable of serving multiple clients.

Cons:

Database is not implemented on FPGA system, so it requires HTTPs connection to access data. Details and statistics about the performance is not provided in this paper.

(2) FPGA Hardware Implementation for Accelerating QR Decoding [2]
This work focused on devising a hardware solution to accelerate the decoding function. The system is built on CYCLONE II FPGA from Altera.

Pros:

Acceleration on the QR decoding process and it has a local database to fetch information from.

Cons:

No meaningful demonstration of the information behind the QR code.

1.2 Goal

The aim of our project is to implement a digital, cloud-based database system, combined with a QR code decoder to mimic and try to improve upon the current vaccination passport system, being widely adopted throughout Canada and the world.

2.0 Project Team

2.1 Mustafa Kanchwala

Strengths:

- Familiarity with Verilog RTL synthesis (for ASIC design) and C/C++ for embedded microcontrollers
- Familiar with computer networking concepts and interfacing with peripherals (eg. SPI)

Weaknesses:

- Minimal hands-on experience with Vivado and Xilinx FPGAs
- No experience with use of Interface Buses such as AXI for interacting with other IP cores

2.2 Guoxian Wu

Strengths:

- Familiar with Verilog/ System Verilog and FPGA RTL design/verification
- Familiar with image process and protocols (I2C/UART) in hardware design

Weaknesses:

- Unfamiliar with TCP/IP network and databases
- Limited experience to Python and software design

2.3 Eduardo Stecca Ortenblad

Strengths:

- Familiar with writing C/C++ code for embedded applications, including TCP servers and clients
- Familiar with web development applications and databases

Weaknesses:

- HDL experience is limited to ECE241 and ECE342
    - Unfamiliar with Vivado and Xilinx FPGAs
    - Unfamiliar with AXI interface

2.4 Xuening Dong

Strengths:

- Familiar with coding in Verilog/System Verilog/C for various purposes and good at debugging issues with simulation
- Familiar with knowledge in computer network (TCP/IP, MAC)

Weaknesses:

- Limited experience in Xilinx FPGAs and Vivado
- No prior experience in digital communication protocols

2.5 Summary

All our team members have an adequate amount of experience with Verilog and hardware design but many of us are lacking experience in Xilinx and Vivado. Therefore, we have assigned the responsibility of designing each custom IP to everyone based on their preferences and skills. Due to the difference in complexity of each IP core, group members will also need to cooperate in certain tasks to balance the workload.

We look forward to building solid understandings of digital systems design by the end of the project.

3.0 Project Description

Within our project, the system consists of a main TCP server running a database on an FPGA (FPGA2). This database will store our users' personal identification data and vaccination status; refer to Appendix A for a detailed description of the data that will be stored within our database. For simplicity purposes, we will initially implement a static database with a limited number of entries to be accessed. Once our project progresses, we plan to make our database dynamic, where clients can not only access, but also insert, remove, and modify data.

Ideally, our database would be able to have a multitude of simultaneous clients accessing and modifying its data. However, for the purpose of this project we have decided to focus on two clients. A PC client, and an FPGA client (FPGA 1).

FPGA Client (FPGA1): This FPGA will be receiving a QR Code encoded with a user's UID and Name. The QR Code will be sent to FPGA 1 through its station PC (PC1) over UART. It will then proceed to decode the QR code and send a request to the database for that specific user. If the UID matches the user's name, the server will return that user's data from the database to the client. Otherwise, it will return an error. FPGA1 will then proceed to parse the server's response and display it on a GUI using VGA.

PC Client: This can be any PC in the DESL Private Network and is supposed to simulate an Admin user to our database. This client will run through a python script and will be able to send requests to the FGPA server based on the user's needs. Initially this client will only send read requests, but eventually will be able to insert, delete, and modify data from the database.

In the sections to follow, we highlight the technical details of how we plan to implement our project.

3.1 Functional Requirements and Features

We divided the whole project into major requirements and listed them in with different priorities in the table below:

Sl no	Requirement/ Features	Priority	Acceptance Criteria
1	QR Code data should be correctly decoded by the FPGA	High	Verified that information on QR code is correctly decoded by checking with source information.
2	Database Parsing can be performed on the server using information received from the client	High	Server can successfully read the database using Unique Identifier (UID) sent from the client
3	Information can be fetched from the Database when a match is found	High	Once a match of UID is found, server can successfully perform a fetch action and retrieve the correct result.
4	Server can transfer the fetched data to the client via TCP/IP	High	Database Parsing IP retrieves the entire data entry from the database and sends it to the client for TLV decoding

5	Client can receive the data sent from the server using TCP/IP	High	Correct data can be successfully transferred from the Server to Client
6	Data received from the Server can be displayed on the Client	High	Data displayed matches the requested data from the server
7	UART for data transmission between PC1 and FPGA1	High	Display correct information in the terminal
8	VGA Display can show results received from the server, and display visually if access is granted or denied	Medium	Verified that information displayed matches the pre-defined user information and is in the correct position on the screen.
9	TLV encoding is performed on the decoded data before sending via TCP/IP	Medium	TLV encoding is successfully performed on the data received from the QR code
10	Client can perform TLV decoding on the received data received from server	Medium	Decoded data matches encoded data by the server
11	TLV: encoding and decoding of the information to be sent between FPGAs is consistent	If (9) is implemented: High, otherwise, medium	Verified that the decoded message from the encoded message matches the original one.
12	AXI GPIO connected to LEDs showing the access status	Low	Verify that the LED shows the correct vaccine status

3.2 Project Details

Brief Description of Custom IPs:

Sl no	Module	Function Performed	Source
1	QR Code Decoding	Receives a QR code input and performs decoding to reveal information within the QR code	Custom IP
2	Video Interface IP	Interfaces with the Xilinx VGA IP to convert data into VGA display format	Custom IP
3	Database Parsing IP	Interfaces with the MIG IP to read and write data onto the DDR2 memory block. Implements database parsing and fetching action	Custom IP

The general workflow of our project is stated as (Fig 2 in section 3.3):

1. Client (FPGA1) scans a QR code from a computer generated QR code

2. Client (FPGA1) decodes the information stored behind the QR code, encodes it with TLV and then sends the information through network to FPGA 2

3. Server (FPGA2) decodes the TLV information from FPGA 1 and searches for the corresponding entry in the database.

4. Start with a Static Database (few values) on FPGA 2 (could say this is a separate server). It could be eventually developed into a dynamic database to allow user store data in the database.

5. After checking with the database, Server (FPGA2) sends the information back to client (FPGA 1) and client (FPGA 1) displays the result.

6. FPGA 1 reviews the information and displays the information on screen for a remote user to confirm vaccine status

7. PC1 or any other PC in the DESL Network can register/ update/ delete the data in Server (FPGA2)

8. Request type will be encoded within the TLV requests

9. FPGA 1 deciphers the command received from FPGA 2 and indicates whether allowing or denying request to enter on LEDs (digital lock GPIO controlled).

10. On VGA (FPGA 1): shows the QR code during decoding and text information of the person/permission after receiving data from FPGA 2

3.3 Block Diagrams

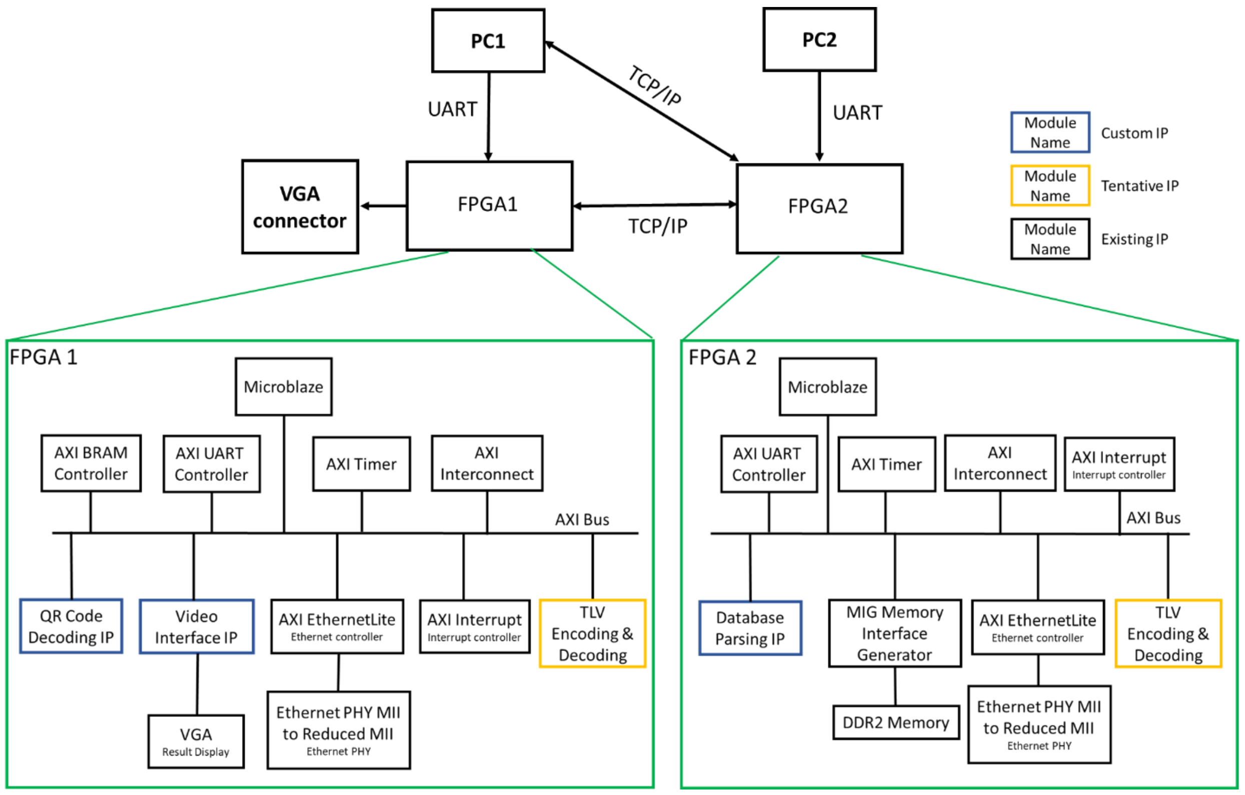


Figure 1: System Level Block Diagram

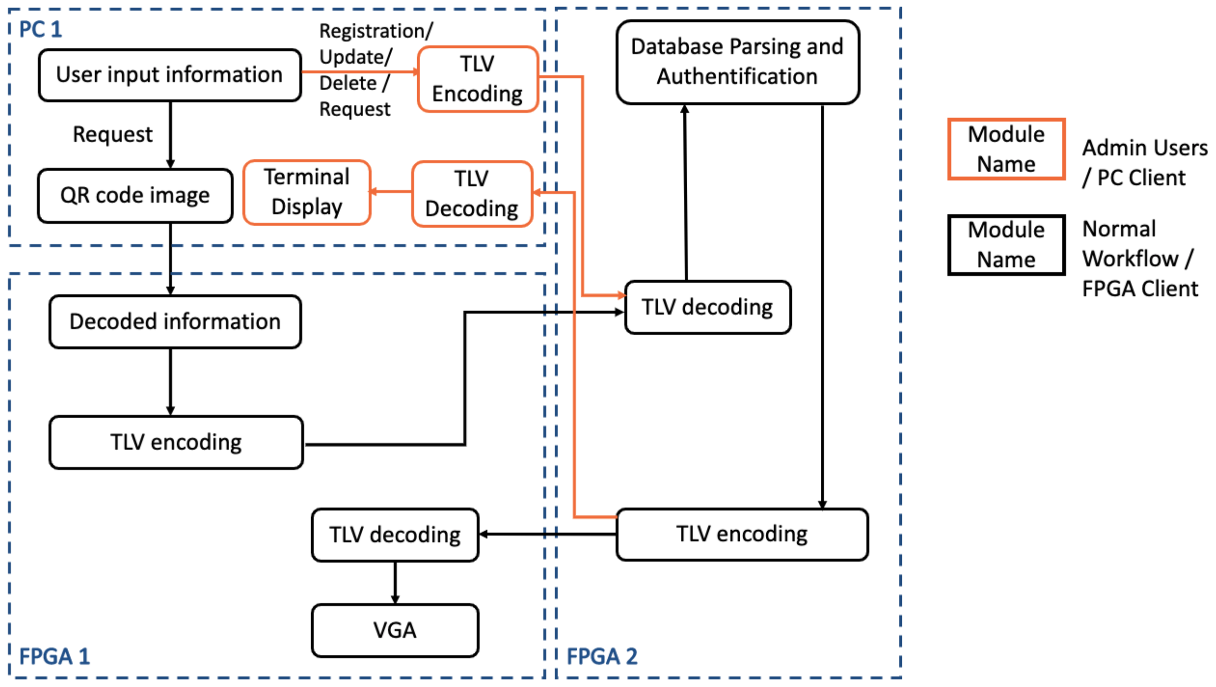


Figure 2: Software/ Hardware Workflow Diagram

4.0 Testing

To test the system, we plan to start with tests on individual modules before integrating the various parts and performing system level testing.

4.1 Unit tests

TCP/IP connection: Test client with a python script server, and test server with a python script client.

Transmission between host PC and FPGA:

1. Use python client script to test the establishment of connection of UART and TCP/IP
2. Map the QR code image into binary matrices and compare the two versions of matrices from FPGA side and the host PC side.

QR decoding correctness:

1. Generate QR code with different information using Python and use it as a reference model for debug.
2. Writing testbenches to build a simulation environment to test the QR decoder.
3. After integrating FPGA1, we can directly connect FPGA1 with PC and send the decoded data to check its correctness.

Database Parsing:

1. Static Database test 1: prepare a static database and upload it to DDR2 memory on the FPGA. Review using Memory Monitor that information is correctly stored
2. Static Database test 2: perform a fetch data action from the DDR2 memory using the Database parsing IP core. Review the data received and compare with the information displayed on the Memory monitor to confirm a good fetch operation was performed
3. Dynamic Database test 1 (if possible): attempt to write to the DDR2 memory using information received from client. Review Memory monitor to confirm if information was successfully added to the correct memory address.
4. Testbench: Use a testbench to simulate a fetch request as received from the MicroBlaze and determine if a known sample data can be correctly fetched.

4.2 System level tests

Full System Integration Testing: Build a system level testbench to simulate the connection of hardware blocks. Firstly, we integrate the blocks with high priority to test the main function. As we integrate the whole system, use ILA to monitor the hardware and GDB debug to monitor the software.

Latency monitoring: perform a latency test to determine the total time taken from when a new QR code is read by the client to determining an approval or denial outcome from the server.

Accuracy and Consistency test: Manually compare the information displayed on VGA with the input information from PC.

5.0 Project Complexity

Project Complexity has been determined by using the criteria provided in the Table below. Scores for best-case scenario and worst-case scenario have been assigned based on what the team must achieve at a minimum (worst case) and what the team can aim to achieve if good progress is made on the basic workflow (best case)

Project Complexity area	Points	Best Case (Y/N	Worst Case (Y/N)
Data transfer over USB:			
Transfer data from Desktop to FPGA using UART + MicroBlaze + python script (0.25 points)	0.25		Y
Transfer data from Desktop to FPGA using UART without MicroBlaze involvement (0.75 points)	0.75	Y	
Desktop to FPGA network connection			
Connect using TCP/UDP from python script (0 points)	0	Y	Y
FPGA network connectivity			
Connect to network with TCP/UDP using MicroBlaze + LWIP (0 points)	0	Y	Y
Other Peripherals			
VGA output without MicroBlaze involvement (1.0 point)	1	Y	Y
IP Core implemented in FPGA Hardware			
Difficulty scales with complexity of core (0.25-2 points)	1.5		Y
Difficulty scales with complexity of core (0.25-2 points)	2	Y	
Performance monitoring			
Implement performance monitoring in MicroBlaze (0.10 points)	0.1		Y
Implement performance monitoring in hardware, triggered by hardware (0.50 points)	0.5	Y	
Software algorithm implemented on MicroBlaze			
Difficulty scales with complexity of algorithm (0.10-1.0 points)	0.5	Y	Y
Meaningful visualization of program run/statistics gathered/results obtained (i.e. Marketing)			
Visualize meaningful results with a GUI (0.75 points)	0.75	Y	Y
	Total Score	5.5	4.1
		Best Case	Worst Case

6.0 Risks

As we are building a complex SoC system integrated with many blocks, we are assigning priorities for each functional requirement and making contingency plan in case some blocks do not function well. When we are encountering some difficulties or bugs, we are going to make efforts in the order of assigned priorities. Below are some risks the project team has identified at this stage of the proposal. Contingency plans for each of these have also been provided.

Sl no	Risk	Contingency Plan
1	Implementation challenges encountered for Database parsing IP core. Cannot perform parsing and fetch in hardware.	creating a backup software implementation in MicroBlaze for performing memory parsing and data

Sl no	Risk	Contingency Plan
		fetch from DDR2. Use this approach to debug and resolve issues with hardware implementation
2	Data transfer from desktop to FPGA using UART without MicroBlaze cannot be implemented on time/effectively	Perform data transfer to FPGA using UART via a MicroBlaze and python script
3	Challenges encountered when implementing VGA output without MicroBlaze involvement	Implement VGA output via MicroBlaze to test functionality and use it to identify issues with VGA interface IP core
4	Challenges encountered when implementing QR code decoding IP	Try to perform QR code decoding in Software on MicroBlaze, identify issues with the Hardware implementation and use the Software implementation as guide to help in debugging. Also, we can use bit 0/1 representing the black/white blocks on QR code at first to make the decode handier.
5	Available memory on Nexys DDR board is less than what is needed to implement a larger database	Try to optimize memory allocation to the Block RAM and use space on DDR2, if needed, port the project to Nexys Video Board

7.0 Resource Requirements

For this project, we will be needing the following resources:

Sl no	Resource	Quantity
1	FPGA Board (Nexys 4 DDR)	2
2	Access to MicroBlaze IP Core on Nexys 4 DDR	2
3	VGA Display connected to FPGA Board	1
4	ECF PC's DESL-A Machines	2

8.0 Milestones

A high-level view of task allocation:

PC 1	FPGA 1	FPGA 2	PC2
QR Code Generation (python script): Xuening	**QR Code IP**: Guoxian **VGA IP**: Xuening **UART data transfer**: Guoxian	**Database parsing IP**: Mustafa	**Use UART to preload database from PC2**: Mustafa

PC 1	FPGA 1	FPGA 2	PC2
UART data transfer (python script): Xuening **TCP Client (python script)**: Eduardo	**TCP Client (Microblaze)**: Eduardo **TLV Encoding and Decoding**: Eduardo	**TCP Server (Microblaze)**: Eduardo **TLV Encoding and Decoding**: Eduardo	(preload memory into FPGA directly without Microblaze)

8.1 Milestone #1: Research and initiation

M1Task1: Eduardo: Design and document data encoding and decoding for TCP Server and Clients using TLVs

M1Task2: Xuening: Start on the VGA, finish at least displaying static images on the screen. Write a python script for generating QR codes for future test purposes.

M1Task3: Guoxian: Research on QR code decoding, test C/Python code for QR code decoding and write a document including the functional requirements, interfaces and the algorithm used in RTL design

M1Task4: Mustafa: Start by identifying a framework for the Database retrieval IP. Write a sample dataset to the DDR2 memory and confirm information is being stored appropriately (static database).

8.2 Milestone #2: Basic functionality

M2Task1: Eduardo: Python scripts for TLV encoding and decoding – integrate with TCP Server and client. This will be used to test FPGAs' client and server, and to modify the database.

M2Task2: Xuening: Continue working on the VGA implementation, including showing some texts (e.g. permission, personal information) on the screen once received response.

M2Task3: Guoxian: Design RTL code the QR code decoder implemented on FPGA.

M2Task4: Mustafa: Implement a mechanism to parse the data stored in the DDR2 when a request is received by FPGA2 and perform fetch if matching data is found.

8.3 Milestone #3: Finish most units

M3Task1: Eduardo: Write TLV encoding and decoding in C for the MicroBlaze, this will be used to test the database and as a backup resource in case we cannot get an IP to do it.

M3Task2: Xuening: Finalize the VGA part, including showing the information in an appealing manner. Start on PC - FPGA data transfer script for sending images to FPGA 1 (starting from using Microblaze and python script).

M3Task3: Guoxian: Build test environment of QR code decoding with software decoding as a reference model and start testing the design code. Start on RTL simulation and debugging

M3Task4: Mustafa: Perform testing and improve the parsing and fetching of information from the database. If possible, attempt to write new data that's received from client which is not currently present in the database (dynamic database)

8.4 Milestone #4:

M4Task1: Eduardo: Start IPs for TLV encoding/decoding

M4Task2: Xuening: Test the data transfer script, draft on a version of PC-FPGA communication only with the UART. Work together with Guoxian to complete the workflow in FPGA 1. Catch up with any delayed previous milestone.

M4Task3: Guoxian: Continue to debug and test the custom IP. Connect PC1 and FPGA1 with UART. Work together with Xuening to build up preliminary system on FPGA1.

M4Task4: Mustafa: Interface with MicroBlaze to ensure data fetch requests can be received and processed effectively with several use cases. Implement testbenches to validate IP functionality. Work with Eduardo to ensure requests coming to FPGA 2 can be fulfilled appropriately by database parsing IP core.

8.5 Milestone #5: Initial integration and debugging

M5Task1: Xuening/Guoxian: Perform system level testing on FPGA 1 including checking the quality of communication between PC1 and FPGA 1, correctness of information decoded from the QR code and correctness of information displayed on VGA.

M5Task2: Eduardo/Mustafa: Begin system integration and testing of hardware and software on FPGA 2. Conduct extensive debugging activity during this week

8.6 Milestone #6: Final integration and debugging

M6Task1: Eduardo/Xuening/Guoxian/Mustafa: Finalize full system integration, test overall functionality and connections PC-FPGA or FPGA-FPGA. Adjust functions and fix problems as needed for optimal system performance.

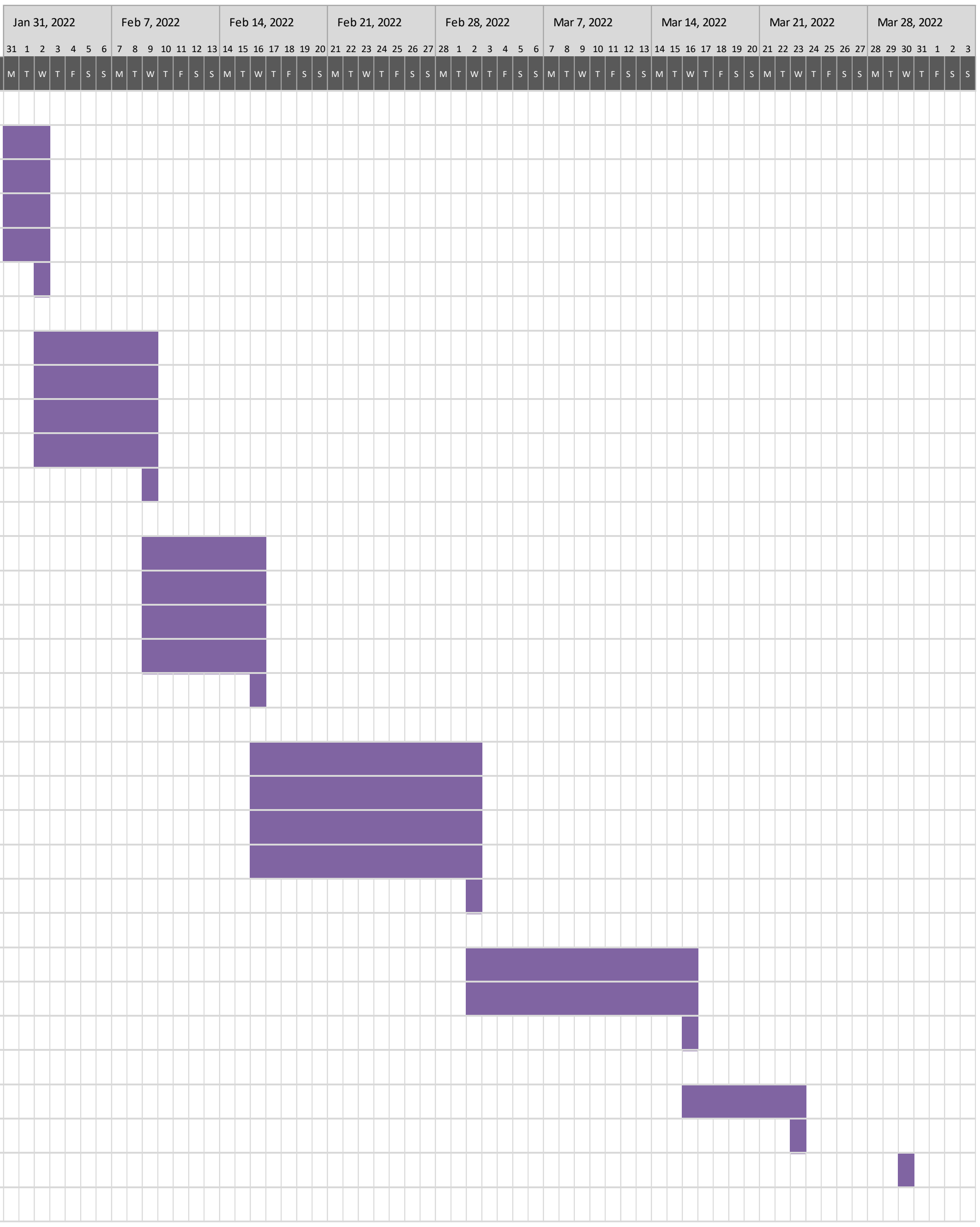
Project Gantt Chart

Virtual Vaccine Passport Scanner for Remote Entry Approval

University of Toronto
ECE 532 Group 3

Project Start:	Mon, 1-31-2022
Display Week:	1

TASK	ASSIGNED TO	PROGRESS	START	END
Week 1				
M1 Task1	Eduardo	0%	1-31-22	2-2-22
M1 Task 2	Xuening	0%	1-31-22	2-2-22
M1 Task 3	Guoxian	0%	1-31-22	2-2-22
M1 Task 4	Mustafa	0%	1-31-22	2-2-22
Milestone 1 Submissior	All	0%	2-2-22	2-2-22
Week 2				
M2 Task1	Eduardo	0%	2-2-22	2-9-22
M2 Task 2	Xuening	0%	2-2-22	2-9-22
M2 Task 3	Guoxian	0%	2-2-22	2-9-22
M2 Task 4	Mustafa	0%	2-2-22	2-9-22
Milestone 2 Submissior	All	0%	2-9-22	2-9-22
Week 3				
M3 Task 1	Eduardo	0%	2-9-22	2-16-22
M3 Task 2	Xuening	0%	2-9-22	2-16-22
M3 Task 3	Guoxian	0%	2-9-22	2-16-22
M3 Task 4	Mustafa	0%	2-9-22	2-16-22
Milestone 3 Submissior	All	0%	2-16-22	2-16-22
Week 4				
M4 Task 1	Eduardo	0%	2-16-22	3-2-22
M4 Task 2	Xuening	0%	2-16-22	3-2-22
M4 Task 3	Guoxian	0%	2-16-22	3-2-22
M4 Task 4	Mustafa	0%	2-16-22	3-2-22
Milestone 4 Submissior	All	0%	3-2-22	3-2-22
Week 5				
M5 Task 1	Xuening/ Guoxian	0%	3-2-22	3-16-22
M5 Task 2	Eduardo/Mustafa	0%	3-2-22	3-16-22
Milestone 5 Submissior	All	0%	3-16-22	3-16-22
Week 6				
M6 Task 1	All	0%	3-16-22	3-23-22
Milestone 6 Submissior	All	0%	3-23-22	3-23-22
Final Demo	All	0%	3-30-22	3-30-22
Final Report	All	0%	4-7-22	4-7-22



9.0 References

[1] Z. Horvat, V. Ilić and M. Nikolić, "Web Server and QR Decoder Applications for Xilinx FPGA Boards," *2018 Zooming Innovation in Consumer Technologies Conference (ZINC)*, 2018, pp. 148-151.

[2] Alhammami, Muhammad, “FPGA Hardware Implementation for Accelerating QR Decoding,” Journal of Engineering and Applied Science, vol. 11, pp. 3273-3278.

Appendix A: Database Entry Information

For each user (entry) in our database, we will store the following information

Field	Data Type	Size (Bytes)	Description
UID	Unsigned Integer (unique)	8	A unique identifier for the database entry. This might not be stored directly into the database and just be calculated based on a hash function.
First Name	String	64	User's First Name
Last Name	String	64	User's Last Name
Date of Birth	Date*	2	User's Date of Birth
Vaccination Status	Enum	1	User's vaccination status encoded as an Enum with the following format: 1. Unvaccinated 2. Partially vaccinated 3. Fully vaccinated
First Dose Vaccine Type (optional)	Enum	2	User's fist dose vaccine type encoded as an Enum with the following format: 1. AstraZeneca 2. Bharat Biotech 3. Janssen/Johnson & Johnson 4. Moderna 5. Pfizer-BioNTech 6. Sinopharm BIBP 7. Sinovac 8. Other
First Dose Date (optional)	Date*	2	Date of first vaccine
Second Dose Vaccine Type (optional)	Enum	2	
Second Dose Date (optional)	Date*	2	
Third Dose Vaccine Type (optional)	Enum	2	

Third Dose Date (optional)	Date*	2	

*: Dates will be encoded into 16 bits with the following format: dddddmmmmyyyyyyy.

Day: 5 bits (0-31)

Month: 4 bits (1-12)

Year: 7 bits (Years since 1900)