**SMART TRAFFIC LIGHTS**

BACHELOR OF COMPUTER ENGINEERING

by

Tejas Sheth, 19102038

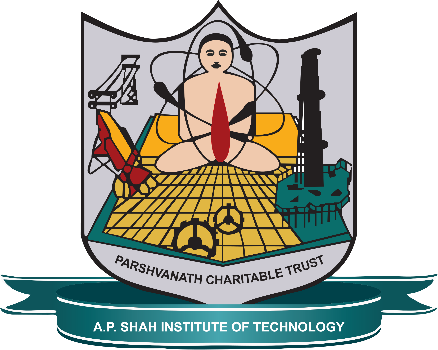
Nidhi Singh, 19102042

Nikita Joshi, 19102033

Jainam Zaveri 20202007

Guide

Prof. Rushikesh Nikam



Department of Computer Engineering

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

2022-2023



A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

CERTIFICATE

This is to certify that the project entitled “Smart Traffic Lights” is a bonafide work of **“Karan Shah(19102038), Nidhi Heniya(19102041), Nikita Joshi(19102033), Jainam Zaveri(20202007)”** submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Engineering**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Guide Project Coordinator

Prof. Rushikesh Nikam Prof. Rushikesh Nikam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Head of Department Principal

Prof. Sachin Malave Dr. Uttam Kolekar

Date:

 A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

**Project Report Approval for B.E.**

This project report for Sem-VII entitled ***Smart Traffic Lights*** by ***Karan Shah(19102038), Nidhi Heniya(19102041), Nikita Joshi(19102033), Jainam Zaveri(20202007)*** is approved for the degree of ***Bachelor of Engineering*** in ***Computer Engineering***, ***2022-23***.

Examiner Name Signature

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date:

Place:

**Declaration**

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Karan Shah 19102038)

(Nidhi Heniya 19102041)

(Nikita Joshi 19102033)

(Jainam Zaveri 20202007)

Date:

**Abstract**

The current traffic management system requires immediate improvement because it is inefficient due to the unpredictable nature of traffic density trends across the day. The system's timers have a fixed time duration for shifting traffic flow between various directions. This causes vehicles to wait for a longer period of time, even though traffic density is minimal. As a fix, the signal timer can be coded to differ depending on the variance of traffic density. This can significantly reduce traffic congestion, in addition to offering numerous other benefits.

The proposed system focuses on four-way intersections with square traffic signals and aims to enhance the existing traffic management system. To process and estimate actual traffic on the roads, the model employs image processing and machine learning. The system will then calculate the duration for every road before enabling the signal.

The system is an intelligent resemblance of a traditional traffic cop who has successfully made better decisions to ensure the smooth flow of traffic at the intersection.

**CONTENTS**

1. Introduction 1
2. Literature Survey 5
3. Limitation of Existing System 9
4. Problem Statement, Objectives and Scope 11

4.1 Problem Statement 11

4.2 Objectives 11

4.3 Scope 11

1. Proposed System 13

5.1 Proposed System Overview 13

5.2 Design details 19

5.3 Methodology 20

1. Experimental Setup 21
2. Project Plan 0
3. Expected Outcome 0

References 0

**LIST OF FIGURES**

5.1.1 Architecture Diagram 0

5.1.2 Data Flow Diagram (level 0) 0

5.1.3 Data Flow Diagram (level 1) 0

5.1.4 Sequence Diagram 0

5.1.5 Activity Diagram 0

5.2.1 Implementation Output 0

5.2.2 Implementation Output 0

**LIST OF TABLES**

2.1 Literature review Table 0

**Abbreviation**

|  |  |
| --- | --- |
| *YOLO* | You Only Look Once |
| *AI* | Artificial Intelligence |
| *TST* | Traffic Signal Timer |
| *ATLTC* | Adaptive Traffic Light Timer Control |
| *UML* | Unified Modeling Language |

**CHAPTER 1**

**Introduction**

SQL is a Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database. SQL is the standard language for Relational Database Systems. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language. SQL is widely popular because it offers the following advantages −

* Allows users to access data in relational database management systems.
* Allows users to describe the data.
* Allows users to define the data in a database and manipulate that data.
* Allows embedding within other languages using SQL modules, libraries & pre-compilers.
* Allows users to create and drop databases and tables.
* Allows users to create views, stored procedures, and functions in a database.
* Allows users to set permissions on tables, procedures and views.

Due to its wide popularity, it makes SQL more susceptible to attacks. SQL is designed to allow people access to information and is therefore inherently vulnerable. SQL is agnostic, meaning it works across database platforms. The upside to this is that it allows code to be database-server agnostic. But it is also the source of the problem. To prevent most vulnerabilities, developers should use parameterized SQL or stored procedures specific to the database server.

One of the big reasons why SQL injection maintains traction is due to improper development planning and the use of insecure development architecture. Making use of unsupported or legacy software or features introduces security holes that may not be patched or caught as quickly as they would with modern software. Running patched and modern versions of software are critical to avoiding security exploits, including SQL injection.

SQL injection is a web security vulnerability that allows an attacker to interfere with queries that an application makes to its database. This attack occurs at the application layer. Through successful SQL injection an attacker can view ,modify ,delete the data ,and also get access to sensitive data. This leads to breach of the three security principles ,CIA, i.e confidentiality ,integrity and authenticity of data. The main consequences are:

* Confidentiality: Since SQL databases generally hold sensitive data, loss of confidentiality is a frequent problem with SQL Injection vulnerabilities.
* Authentication: If poor SQL commands are used to check user names and passwords, it may be possible to connect to a system as another user with no previous knowledge of the password.
* Authorization: If authorization information is held in a SQL database, it may be possible to change this information through the successful exploitation of a SQL Injection vulnerability.
* Integrity: Just as it may be possible to read sensitive information, it is also possible to make changes or even delete this information with a SQL Injection attack.

This attack occurs when you ask for user input, like username, and the user incorrectly fills it with some SQL statement that gets unknowingly executed on one’s Database. SQL Injection is very common with PHP and ASP applications due to the prevalence of older functional interfaces. Because they are relatively easy to implement, and because the potential reward is great, SQL injection attacks are not uncommon.

Statistics vary, but it’s estimated that SQL injection attacks comprise the majority of attacks on software applications. According to the Open Web Application Security Project, injection attacks, which include SQL injections, were the third most serious web application security risk in 2021.

**1.1 SQL Injection**

There has been an exponential rise in the number of dynamic websites. We have to handle large amounts of data stored for varied purposes, as well as their modification has to be fast to provide a rich user experience. For this, we have to rely on relational databases like MySQL, MSSQL, etc. which are based on standard query language SQL.

SQL communication between the website and the SQL server consists of SQL queries. The most common and frequently used operation is data retrieval using the SELECT command with the WHERE clause, an advancement can be the use of multiple SQL queries concatenated with a UNION statement returning just one table.

*SELECT column1, column2 FROM table1 WHERE column1=column2;*

*SELECT column1 FROM table1 WHERE column1=10 UNION SELECT column2 FROM table2 WHERE column2=column1;*

A hacker manages SQL Injection whenever the server-side scripting has an inappropriate input validation, leading to the attacker gaining partial or full access to the query and thus the database.

*SELECT column1 FROM table WHERE column2=user-input;*

*SELECT column1 FROM table WHERE column1=1 OR 1 = 1;*

 This was the most simple SQL Injection, more refined queries are those when the attacker adds a UNION statement and combines multiple queries from the original query. There are many types of conventional and modern SQL Injection attacks. They are primarily divided into

A.) Classical SQLi (Basic SQLi) attacks are the simplest and most frequently used form of SQLi. These may occur when users are permitted to submit a SQL statement to a SQL database through user input. There are 4 different types: Piggy-Backed, Stored Procedures, Union Query, and Alternative Encoding according to Halfond, W. G. et. al. [18] and Wei, K. et. al. [19].

1) Piggy-Backed Queries: Instead of modifying the original query, the attacker intends to develop new queries that piggyback on it. As a result, the DBMS gets inundated with SQL queries. The first query is a standard query that is run normally, while the succeeding ones are run to complete the attack.

*eg. normal SQL statement + ";" + INSERT (or UPDATE, DELETE, DROP) <rest of injected query>*

2) Stored Procedure: When a conventional SQL query (such as SELECT) is generated as a stored procedure, an attacker can inject another stored procedure as a substitute resulting in a denial of service (DOS), or execute remote instructions.

*eg. normal SQL statement + "; SHUTDOWN; " <rest of injected query>*

3) Union Query: An attacker injects a UNION SELECT query to mislead the programme into providing data from a table other than the one intended as stated by Hwang, D. (2022) [20].

*eg. normal SQL statement + "semicolon" + UNION SELECT <rest of injected query>*

4) Alternative Encoding: The attacker modifies the SQLi pattern such that standard detection and prevention systems miss it. In this method, the attacker employs hexadecimal, Unicode, octal, and ASCII code encoding in the SQL Statement.

B.) Advanced SQLi

1) Blind SQLi: Attackers devised strategies to circumvent the lack of error notifications while still knowing whether the input is being treated as a SQL statement. This technique is often used in two variations: content-based blind SQLi and time-based blind SQLi.

2) Fast Flux SQLi: Fast Flux is a DNS method to conceal phishing and malware distribution sites behind a constantly changing network of compromised servers. The Asprox botnet was used to launch the large SQLi assault employing rapid flux. In Fast Flux mode, the DNS (Domain Name Server) hosts many malware-infected IPs at the same time, and the IPs rapidly change.

3) Compounded SQLi: A compound SQLi attack is a pair of two or more attacks that target the webpage and have far-reaching implications than the previous SQLi mentioned. The fast development of detection and mitigation measures for multiple SQLis has resulted in the emergence of compound SQLi. SQLi combined with DDoS attacks, DNS hijacking, XSS, and insufficient authentication are just a few examples.

**CHAPTER 2**

**Literature Survey**

SQL Injection (SQLi) is a type of attack in which an attacker inserts a malicious SQL query into the web application by appending it to the input parameters. Sadeghian, A. et. al. (2013) illustrate the classification of injection attacks like tautologies, illegal / logically incorrect queries, union queries, piggy-backed queries, stored procedures, inference, and alternate encodings [1]. Security researchers have categorized the solutions for SQLi into three main groups: Best code practices, SQLi detection and SQLi runtime prevention. The optimum solution would be writing secure code and among best code practices- parameterized querying is the most secure and efficient technique.

Rai, A. et. al. illustrates the classification and prevention of different SQLi attacks. SQLi is generally classified as In-band SQLi, Inferential SQLi and Out of Bound SQLi [2]. In-Bound SQL injection is further classified as Error-based and Union-based SQLi. Inferential SQLi can be broken down into Boolean-based Blind SQL and Time-based SQL. Defensive techniques that could be used to prevent an SQLi attack include Whitelisting/Blacklisting, prepared statement/ parameterized query, stored procedure, defensive coding practice, taint-based approach, proxy filters, instruction set randomization, low privileges and output Escaping. Different countermeasures work for different SQL Attacks.

Medhane and M. H. A. S.) based their approach on SQLi grammar to identify the SQLi vulnerabilities during software development and SQLi attack based on web-based applications [3]. The attacker’s area unit used SQL queries for assaultive and hence these attacks reshare the SQL queries, thus neutering the behaviour of the program.

John, A. proposed methods consisting of the best features of parse tree validation and code conversion techniques [4]. The algorithm parses the user input and checks whether it's vulnerable if any chance of vulnerability is found it applies code conversion over that input. Results show few drawbacks of code conversion as applying it to every user input is more time consuming and as well as the database also increases. The parse tree validation technique could raise a false alarm if a legitimate user is having blank space in his/her input. The proposed method proved to provide higher security levels than the individual techniques of code conversion and parse tree validation.

Hanmanthu, B. et. al. illustrates the use of the famous decision tree classification techniques to prevent SQLi attacks [5]. The proposed model works by sending different specially planned attack requests to the proposed SQLi decision tree model, and the final SQLi database is created for using classification data. It uses the satisfied analysis technique for finding the SQLi attack and uses the SQL decision tree. Software engineers usually rely on dynamic query building with string concatenation to construct SQL statements. The proposed method makes it possible to engineer different queries based on varying conditions set by users, without the need for manual interactions or error-prone code. The model showed consistency in attack detection and elimination at an average of 82% for all types of attacks. In order to perform a comparative evaluation of the proposed model, authors in [5] compared the proposed model to the other SQL scanning model which includes Acuneits, Netsparker, and Web cruiser and the results of the proposed model show good accuracy in comparison to other models.

Akinsola Jide et.al. gives us an idea about different types of SQLi attacks as already mentioned by Rai, A. et. al. [6][2]. The three main types are Classic In-band SQLI, Inferential Blind SQLI, and SQLi Based On Out-of-Band. They present the comparative analysis of different supervised ML algorithms to mitigate SQLi attacks. Besides precise accuracy and minimum errors, ML models also require putting several factors into consideration. The following metrics were taken into consideration to decide the effectiveness of the algorithm: Kappa Statistic, True Positive (TP) Rate, Accuracy, True Negative (TN) and (time to build the model (TTB)), for each of the machine learning algorithms.

Tang, P. et.al.(2020) only extract and classifies the URL features [7]. The factors like payload length, keywords and their weights are considered for feature extraction. The URL is classified as malicious or non-malicious using ANN (Artificial Neural Network) models. The method and algorithm used here are multi-layer perceptron (MLP) and LSTM, both of which were implemented using Pytorch. The trained model is deployed in the ISP system so that abnormal behaviours can be found in the network in real-time. One of the drawbacks of using such an approach is that using the LSTM, model recognition is poor with high processing time & has lower accuracy.

Four machine learning models were considered in Kamtuo, K., & Soomlek, C. and they were compared, SupportVector Machine (SVM), Boosted Decision Tree, Artificial Neural Network, and Decision Tree [8]. They have proposed a framework using a compiler platform and ML to detect SQLi in queries which are illegal and logically incorrect on server-side scripting. The dataset consists of 1100 samples of vulnerable SQL commands. After training the model with the dataset it was evaluated in terms of probability of detection, probability of false alarm, precision, accuracy, and processing time. Decision Jungle was the best in performance showing results as the best machine learning model which related to the processing time of 2.4725 seconds and accuracy of 0.9968.

Ross, K. collected traffic from two points: a web application host and a Datiphy appliance node [9]. It is demonstrated that the accuracy obtained with correlated datasets using algorithms such as rule-based and decision-tree are nearly the same as those with a neural network algorithm, albeit with significantly improved performance.

Reinforcement Learning (RL) is known for obtaining knowledge by trial and error and continuously interacting with a dynamic environment. It is characterized by self-improving and online learning, making it one of the intelligent agents (IA) core technologies. The reinforcement signal provided by the environment in RL is to make a kind of appraisal of the action quality of the IA, but not tell the IA how to generate the correct action. The basic model of RL as stated in Qiang, W., & Zhongli, Z. includes a state, action and reward system [10]. Where the IA perceives the environment and chooses an action to obtain the biggest reward value by continuously interacting with the environment. The ultimate goal of RL is to learn an action strategy. The basic theory of reinforcement learning technology is: If a certain system's action causes a positive reward for the environment, the system generating this action lately will strengthen the trend, this is a positive feedback process; otherwise, the system generating this action will diminish this trend. Typical RL method based on the Markov decision-making process (MDP) model includes two kinds: Model-based methods such as the SARSA algorithm and Model-irrelevant methods, such as the TD algorithm and the Q-learning algorithm.

Tian, W. et. al. illustrates methods to generate more effective penetration test case inputs to detect SQLi vulnerability [11]. The model-based penetration test method is found to generate test cases covering more types and patterns of SQLi attack input to thoroughly test the ‘blacklist filter mechanism’ of web applications. Here, the authors proposed two-step penetration test case generation, building and instantiating, where step 1 reveals what test case should be used while step 2 expounds on how many test cases should be used. This study focuses on the adequacy of penetration test case inputs for the SQL injection vulnerability. It builds an experimental platform to verify the proposed test case generation methods.

Ghanem M. C., & Chen T. M. proposes and evaluates an AI-based pentesting system which makes use of RL to learn and reproduce average and complex pentesting activities [12]. The scope is limited to network infrastructures PT planning and not the entire practice. Moreover, the authors tackle the complex problem of expertise capturing by allowing the learning module to store and reuse PT policies in a more efficient way.

Niculae, S. et al. measured the performance of multiple fixed-strategy and learning-based agents [13]. They concluded that Q-learning, with some extra techniques applied and greedy agent initialisation, performed best, surpassing human performance in the given environment.

Hu, Z., Beuran, R., & Tan, Y. suggests an automated penetration testing framework, based on deep learning techniques, particularly deep Q-learning networks (DQN) [14]. The authors conducted an experiment in which a given network host was populated with real host and vulnerable data, to determine the optimal attack path, and to provide viable solutions.

Erdődi, L. et. al. simplified the dynamics of SQLi vulnerabilities by casting the problem as a security capture-the-flag and implementing it as an RL problem [15]. Assuming that the vulnerability has been identified, they rely on RL algorithms to automate the process of exploiting SQLi. They implemented the model using two simulations. The first simulation showed that a simple RL agent based on a Q-learning algorithm can successfully develop an effective strategy to solve the SQLi problem. A tabular Q-learning algorithm can discover a meaningful strategy by pure trial and error and can reach a performance close to the theoretical optimum. Using a table to store the Q-value function allowed them to carry out a close analysis of the learning dynamics of the agent, but this approach had poor scalability. Thus, in the second simulation, they sacrificed interpretability in order to work around the issue of scalability. They deployed a deep Q-learning agent to tackle the same problem as in the first simulation. The deep Q-learning agents were able to learn a good strategy for the SQLi problem as well as provide a solution to the space constraints imposed by the instantiation of an explicit Q-table.

Given the success of RL in tackling and solving games, Penetration Testing, when distilled as a capture-the-flag (CTF), can be expressed as a game. However, in the case of penetration testing, an artificial agent may learn only by trial and error while a human hacker may rely on alternative sources of knowledge, deductions, hypothesis testing, and social engineering. Although an RL agent may in principle learn the structure from scratch in a pure model-free way, this may turn out to be a computationally hard challenge. Thus according to Zennaro, F. M., & Erdodi, L.injecting some form of elementary a priori knowledge about the structure of the problem may simplify the learning problem [16]. Some basic forms of apriori knowledge which make the RL agent more efficient are lazy loading, state aggregation and imitation learning. The authors categorized CTFs in groups according to the type of vulnerability they instantiate and the type of exploitation that a player is expected to perform. The prototypical classes of CTF problems considered were port scanning and intrusion, server hacking and website hacking. All the simulations were implemented using the standard RL interface defined in the OpenAI gym library. Simulation 1 solved the Port Scanning CTF problem using the basic tabular Q-learning algorithm. Solving this challenge required learning the problem meaning that the RL agent has to rely strongly on exploration. Simulation 2 solved the Non-stationary Port Scanning CTF problem by extending the previous problem by considering a more challenging scenario in which the target system is not stationary, but it may randomly change in response to the actions of the agent. Introducing non-stationary dynamics made the problem more challenging by preventing the agent from learning the exact structure of the problem with certainty. Despite this, the Q-learning agent was still able to solve the CTF problem in a reasonable yet sub-optimal way. Simulation 3 solved the Server Hacking CTF problem with Lazy Loading which considers a more realistic scenario. The problem presented a serious challenge to the tabular Q-learning agent because of the size of its Q-table. Relying on a priori knowledge in the form of lazy loading controlled the dimensionality of the state and action state pruning the non-relevant states. This method allowed the agent to discriminate between relevant and non-relevant states based on its experience. Simulation 4 solved the Website Hacking CTF problem with State Aggregation preserving most of the complexity of Simulation 3. State aggregation allowed them to inject useful prior information about the structure of the problem, thus simplifying exploration and reducing the number of (state, action) pairs. Simulation 5 solved the Web Hacking CTF problem with Imitation Learning which emulates learning in a teacher-and-student setting, where expert paradigmatic behaviors are offered to a student to speed up its learning. Imitation learning proved to be an effective technique to enable faster learning for the RL agent. The improvement was due to the possibility of introducing the agent's knowledge of the structure of the problem. Instead of encoding knowledge of the structure of the problem in a formal mathematical way, they provided the RL agent with concrete observations about the structure of the problem. The agent could successfully exploit this information in order to learn an optimal policy.

Verme, M. D. et. al. considered the problem of exploiting SQLi vulnerabilities, representing it as a capture-the-flag scenario in which an attacker can submit strings to an input form with the aim of obtaining a flag token representing private information [17]. The attacker was modelled as an RL agent that interacts with the server to learn an optimal policy leading to an exploit. The authors did a comparison between two types of agents, one was a simple structured agent that relied on significant apriori knowledge and used high-level actions and the other was a structureless agent that had limited apriori knowledge and generated SQL statements. The comparison showcased the feasibility of developing agents that relied on less ad-hoc modelling.

**CHAPTER 3**

**Limitation of Existing system**

1 -> latest dataset nhi h so out dated h

2-> in the implementation of the project only tautology can be seen

3-> injection on variety of databases cannot be predicted for now due to limitation of the training daaset

**CHAPTER 4**

**Problem Statement, Objectives and Scope**

**4.1 Problem Statement**

SQL injection is the most common web application vulnerability. The vulnerability can be generated unintentionally by software developers during the development phase.

So far existing methods have considered dealing with SQLi by identifying and classifying the vulnerabilities and relied on supervised machine learning algorithms or unsupervised machine learning algorithms.

The problem of performing SQL injection can be considered a challenge that can be tackled with reinforcement learning methods.

Thus the reinforcement learning agents can help to identify the SQLi attack and prevent it from being executed.

* 1. **Objectives**
* Create a basic Web Application to perform SQLi attacks.
* Detection and Prevention of SQLi Attacks using Reinforcement Machine Learning

**4.3 Scope**

So far existing methods have considered dealing with SQLi by identifying and classifying the vulnerabilities and relied on supervised machine learning algorithms or unsupervised machine learning algorithms. The problem of performing SQL injection can be considered a challenge that can be tackled with reinforcement learning methods. Thus the reinforcement learning agents can help to identify the SQLi attack and prevent it from being executed. Thus the reinforcement learning agents can help to identify the SQLi attack and prevent it from being executed. To improve the security of the website by detecting the injection before executing the query in the backend.

**CHAPTER 5**

**Proposed System**

**5.1 Proposed system overview**

Video monitoring and surveillance systems have been widely used in traffic management in recent years for security, ramp metering, and providing real-time information and updates to travelers. Video monitoring systems can also be used to estimate traffic density and classify vehicles, which can then be used to control traffic signal timers to optimize traffic flow and reduce congestion. Our proposed system aims to create a computer vision-based traffic light controller that can adapt to the current traffic situation. It calculates real-time traffic density using live images from CCTV cameras at traffic intersections by detecting the number of vehicles at the signal and adjusting the green signal time accordingly. To obtain an accurate estimate of the green signal time, the vehicles are classified as a car, bike, bus/truck, or rickshaw. It employs YOLO to detect the number of vehicles and then sets the traffic signal timer based on vehicle density in the corresponding direction. This helps to optimize green signal times, and traffic is cleared at a much faster rate than a static system, reducing unwanted delays, congestion, and waiting time, and thus lowering fuel consumption and pollution.

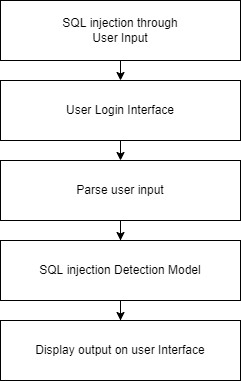


Figure 5.1.1: Architecture Diagram

Architecture Diagram

An architectural diagram is a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element.

Firstly, CCTV/microphone captures data from the traffic signal in the form of images and sound and passes it on to the model. Further the collected data is processed into required format and decisions are calculated there based on the calculated density of traffic of vehicles. Timer is set and is then displayed on to the traffic signal.

Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system (as shown on the DFD flow chart Fig.2 & Fig.3), modeling its process aspects. Often it is a preliminary step used to create an overview of the system that can later be elaborated.

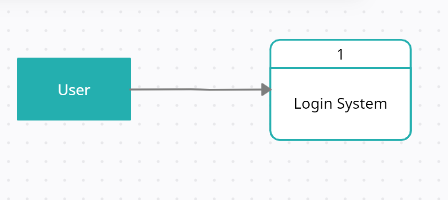


Figure 5.1.2: Data Flow Diagram (level 0)

Level-0

Data collected from components that is input1 and input2 are sent to Model, data calculated from there is sent to the algorithm for making decision and then considered as final output.

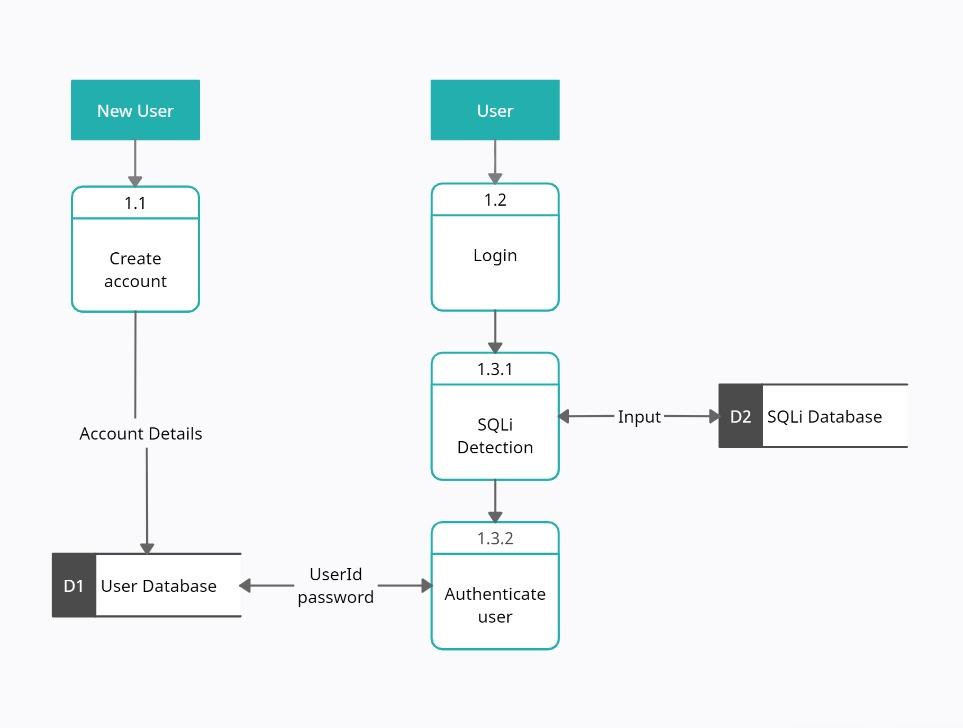
****

Figure 5.1.3: Data Flow Diagram (level 1)

Level-1

Data is collected from camera sensors in the form of images and from sound sensors in the form of audio. This data is then sent to YOLO model and Sound Model respectively. Image Detection and Sound detection is done and the data is then sent as input to Timer algorithm. Further actuators get the information and act according to the decision made and changes the signal light on the traffic signal.

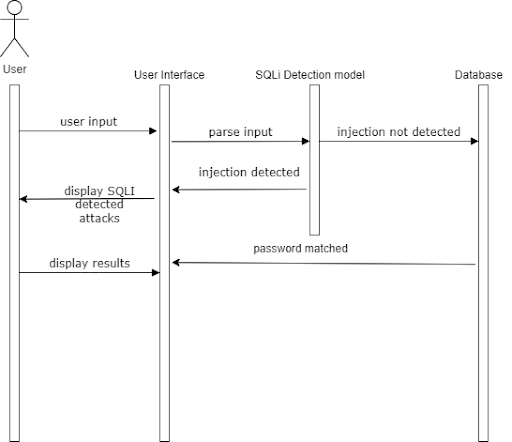


Figure 5.1.4: Sequence Diagram

Sequence Diagram

A sequence diagram is a Unified Modeling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction.

Firstly, the vehicles reach the traffic signal, Sensors present on the signal are then sent requests for capturing the data. Camera and Audio sensor records the data. Data from there is sent to the Model, calculations are done on the data and a decision is taken. This decision data is sent to the actuators. On receiving the data from the model, actuators send the Signal and Cycle timing to the Signal and light timer changes accordingly.

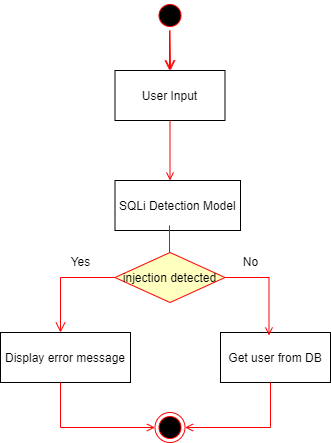
****

Figure 5.1.5: Activity Diagram

Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent.

Firstly, the capturing of data via the sensors takes place then the model comes into action and starts working. Calculation of timer is done. This decision is then sent at the signals where actuators perform timer displaying.

**5.3 Methodology**

The enhanced proposed system uses image processing and object detection to determine real-time traffic density using images from CCTV cameras at road intersections. The image is sent to the vehicle detection algorithm, which employs YOLO. The number of vehicles of each class, such as car, bike, bus, and truck, is counted in order to calculate traffic density. The signal switching algorithm, among other things, uses this density to set the green signal timer for each lane. The red signal times are updated as needed. In order to avoid lane starvation, the green signal time is limited to a maximum and minimum value. A simulation is also created to demonstrate the effectiveness of the system and compare it to the existing static system.

To simulate real-life traffic, a simulation was created from scratch using Pygame. It aids in visualizing and comparing the system to the existing static system. It has a four-way intersection with four traffic lights. On top of each signal is a timer that displays the amount of time until the signal changes from green to yellow, yellow to red, or red to green. Each signal also displays the number of vehicles that have passed through the intersection. Automobiles, bicycles, buses, trucks, and rickshaws arrive from all directions. Some of the vehicles in the rightmost lane turn to cross the intersection to make the simulation more realistic. It is also important to know whether a vehicle will turn or not.

Pygame is a cross-platform collection of Python modules for creating video games. It includes libraries for computer graphics and sound that are intended to be used with the Python programming language. Pygame extends the excellent SDL library with functionality. This enables users to create full-featured games and multimedia programs in Python. Pygame is extremely portable, running on almost every platform and operating system. It is free and open source under the LGPL license.

**CHAPTER 6**

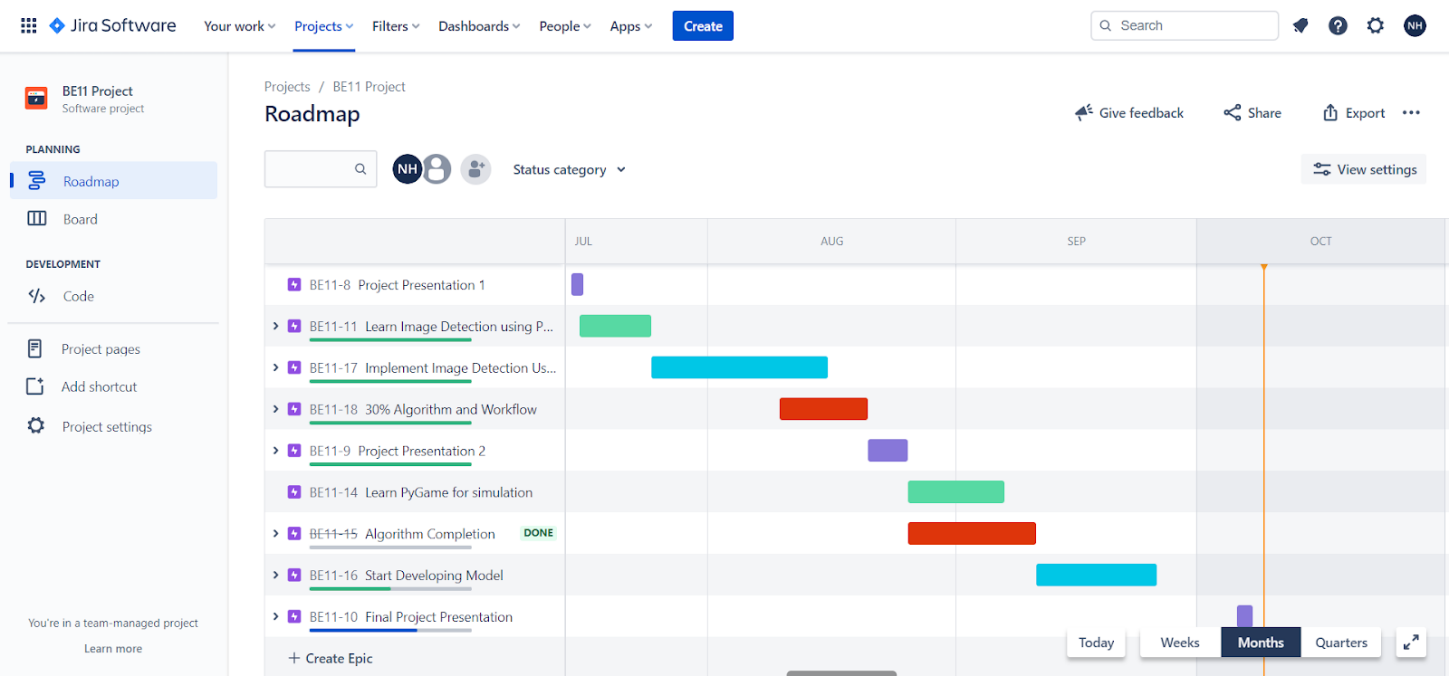
**Experimental Setup**

**6.1 Software Requirements:**

* Python: Python is one of the widely used programming languages for building systems that indulge in Image Processing as well as Machine Learning. Python provides amazingly powerful libraries and tools that help us in achieving the tasks efficiently.
* Windows 10 Operating System
* **Python libraries:** Numpy , Matplotlib, PIL (Python Imaging Library), pandas, streamlit , os.
* **VS Code:** IDE to run, train and test the ML model.Plenty of extensions, open-source, cross-platform support,
* **Git & GitHub:** Version Control System used for collaboration.
* **Streamlit:** Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps.
* **Google Workspace:** Project Management Tool to improve team coordination and file sharing.
* **Teamgantt:** Project Management Tool to improve team coordination and file sharing using Gantt chart.
  1. **Hardware Requirements:**
* **CPU:** 1.8 GHz or faster 64-bit processor; Quad-core or better recommended.
* **RAM:** Minimum of 4GB of ram.
* **Storage:** 4GB of free hard disk space.

**CHAPTER 7**

**Project Plan**



**CHAPTER 8**

**Expected Outcome**

Using modules such as Pygame, random, and math. Displaying a fully functional simulation of a scenario that includes a network of roads, traffic lights, and moving vehicles.

All five types of vehicles must be represented in the simulation, and they must be able to move in all directions regardless of which side they are approaching from. For example, a car approaching from one side of the road must be able to take any of the remaining straight, left, or right roads from the signal. Under no circumstances should the vehicles collide.

The vehicle traffic flow, or the number of vehicles coming from any given direction, must be generated.

Furthermore, the proposed system has some advantages over existing intelligent traffic control systems, such as Pressure Mats and Infrared Sensors. The cost of deploying the system is negligible because footage from CCTV cameras at traffic lights is used, which requires no additional hardware in most cases because intersections with heavy traffic already have such cameras. Only minor alignment may be required. Maintenance costs are also reduced when compared to other traffic monitoring systems, such as pressure mats, which normally suffer wear and tear due to their placement on roads where they are constantly subjected to immense pressure. As a result, the proposed system can be integrated with CCTV cameras in major cities to facilitate better surveillance.

**References**

[1] Sadeghian, A., Zamani, M., & Abdullah, S. M. (2013). A Taxonomy of SQL Injection Attacks. 2013 International Conference on Informatics and Creative Multimedia.

[2] Rai, A., Miraz, M. M. I., Das, D., Kaur, H., & Swati. (2021). SQL Injection: Classification and Prevention. 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM).

[3] Medhane, M. H. A. S. (2013). Efficient solution for SQL injection attack detection and prevention. International Journal of Soft Computing and Engineering (IJSCE), 3, 396-398.

[4] John, A. (2015). SQL Injection Prevention by adaptive algorithm. IOSR journal of computer engineering, 17, 19-24.

[5] Hanmanthu, B., Ram, B. R., & Niranjan, P. (2015). SQL Injection Attack prevention based on decision tree classification. 2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO).

[6] Akinsola Jide, E. T., Oludele, A., & Idowu Sunday, A. (2020). SQL injection attacks predictive analytics using supervised machine learning techniques. International Journal of Computer Applications, (4), 139-149.

[7] Tang, P., Qiu, W., Huang, Z., Lian, H., & Liu, G. (2020). Detection of SQL injection based on artificial neural network. Knowledge-Based Systems, 105528. doi:10.1016/j.knosys.2020.105528

[8] Kamtuo, K., & Soomlek, C. (2016). Machine Learning for SQL injection prevention on server-side scripting. 2016 International Computer Science and Engineering Conference (ICSEC).

[9] Ross, K. (2018). SQL injection detection using machine learning techniques and multiple data sources.

[10] Qiang, W., & Zhongli, Z. (2011). Reinforcement learning model, algorithms and its application. 2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC).

[11] Tian, W., Yang, J.-F., Xu, J., & Si, G.-N. (2012). Attack Model Based Penetration Test for SQL Injection Vulnerability. 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops. doi:10.1109/compsacw.2012.108

[12] Ghanem, M. C., & Chen, T. M. (2019). Reinforcement learning for efficient network penetration testing. Information, 11(1), 6.

[13] Niculae, S., Dichiu, D., Yang, K., & Bäck, T. (2020). Automating penetration testing using reinforcement learning.

[14] Hu, Z., Beuran, R., & Tan, Y. (2020). Automated Penetration Testing Using Deep Reinforcement Learning. 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW).

[15] Erdődi, L., Sommervoll, Å. Å., & Zennaro, F. M. (2021). Simulating SQL injection vulnerability exploitation using Q-learning reinforcement learning agents. Journal of Information Security and Applications, 61, 102903.

[16] Zennaro, F. M., & Erdodi, L. (2020). Modeling penetration testing with reinforcement learning using capture-the-flag challenges: trade-offs between model-free learning and a priori knowledge. arXiv preprint arXiv:2005.12632.

[17] Verme, M. D., Sommervoll, Å. Å., Erdődi, L., Totaro, S., & Zennaro, F. M. (2021, November). SQL Injections and Reinforcement Learning: An Empirical Evaluation of the Role of Action Structure. In Nordic Conference on Secure IT Systems (pp. 95-113). Springer, Cham.

[18] Halfond, W. G., Viegas, J., and Orso, A. A Classification of SQL-Injection Attacks and Countermeasures. In SSSE (2006).

[19] Wei, K., Muthuprasanna, M., & Suraj Kothari. (2006). Preventing SQL injection attacks in stored procedures. Australian Software Engineering Conference (ASWEC’06).

[20] Dr. Drew Hwang, “SQL Injection”, 2022, https://hwang.cisdept.cpp.edu/swanew/text/SQL-Injection.htm

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, ‘‘BERT: Pre-training of deep bidirectional transformers for language understanding,’’ 2018, arXiv:1810.04805.

[22] Shankar Das, S., Serra, E., Halappanavar, M., Pothen, A., & Al-Shaer, E. (2021). V2W-BERT: A Framework for Effective Hierarchical Multiclass Classification of Software Vulnerabilities. arXiv e-prints, arXiv-2102.

[23] “OWASP Top Ten”, https://owasp.org/www-project-top-ten

**CHAPTER 8**

**Appendix**

**Appendix A**

**Python Download and Installation**

**MySQL Download and Installation**

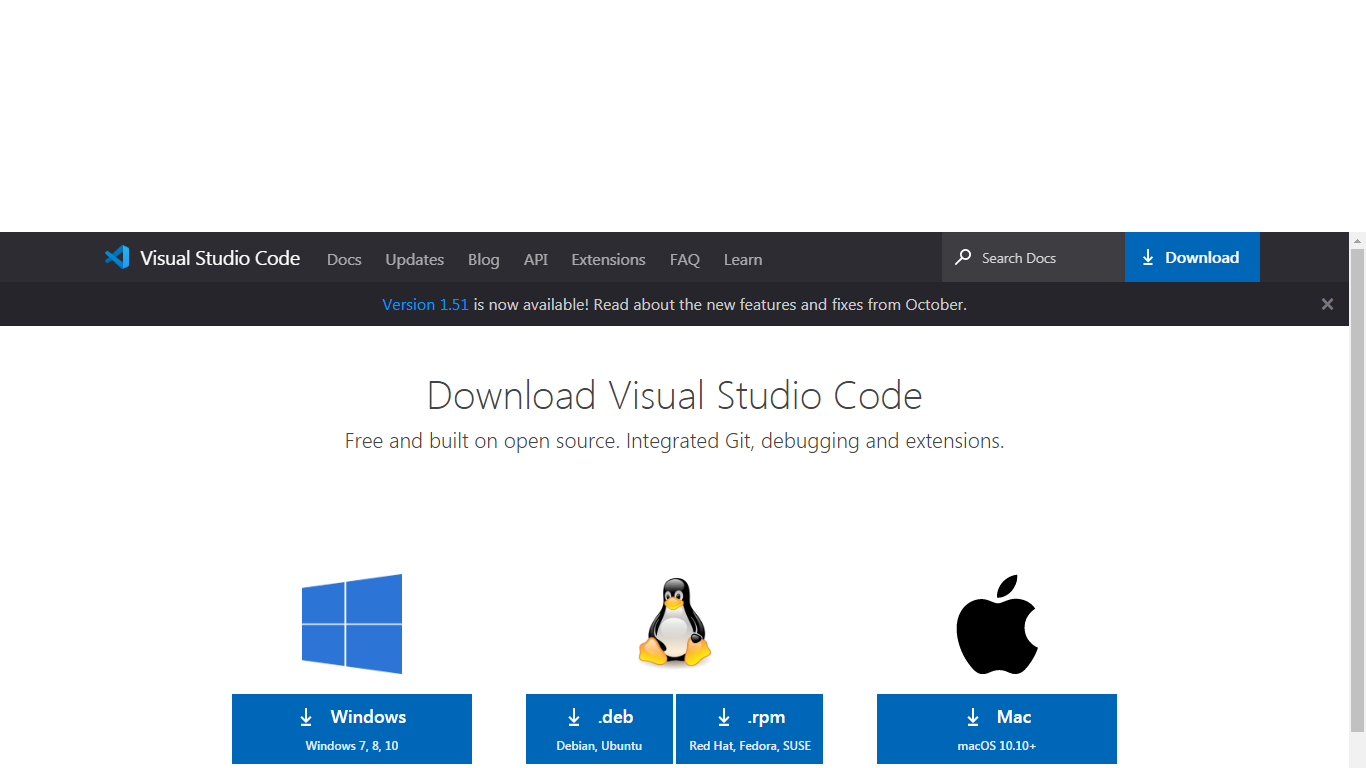
**Visual Studio Download and Installation**

1. Visit the official website and go to https://www.python.org/downloads/. Click the Download button.
2. Once we click the download button, it might ask for a location to save the file. Select an appropriate location and then proceed towards the installation.
3. Double Click the downloaded .exe file and select the Add Python to PATH checkbox below to ensure it is automatically added to the Windows Environment variable. Else we have to do it later on manually. Once the box is checked, click on Install Now.
4. At the time of installation of python, the pop-up will show that the installation is in progress here.
5. Once the setup is complete, we will get a message like this. Click on the Close button to finish the installation of python.
6. Once Python is installed, go to the Windows search bar and type Python, and we will find a desktop app called Python 3.7 (32-bit). Click on that and a command prompt will open.

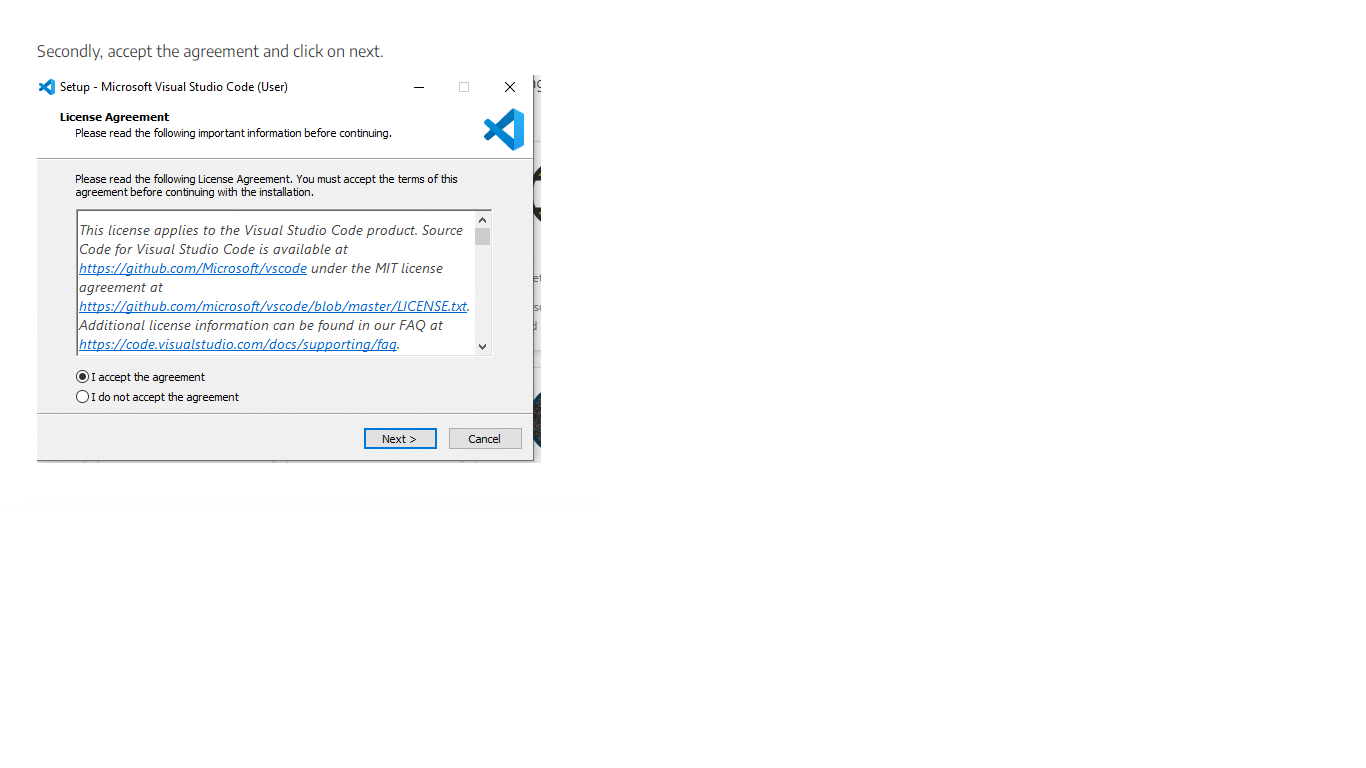
**Appendix B**

**VS Code Download and Installation**

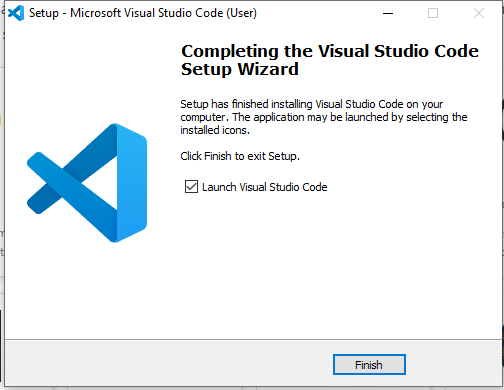
1. Download VS code from <https://code.visualstudio.com/download>
2. Download the Visual Studio Code installer for Windows. Once it is downloaded, run the installer (VSCodeUserSetup-{version}.exe). Then, run the file



1. Accept the agreement and click “next.”



1. After accepting all the requests press the finish button. By default, VS Code installs under: **“C:\users{username}\AppData\Local\Programs\Microsoft VS Code.”**



1. If the installation is successful, you will see the following

