Software Design Document

Australia NSW Traffic Penalty Management Software

James Odike – s5331003  
Thi Ngoc Nhu Nguyen – s5325919

Table of Contents

[1.0 System Vision 3](#_Toc46748622)

[1.1 Problem Background 3](#_Toc46748623)

[1.2 System Overview 3](#_Toc46748624)

[1.3 Potential Benefits 3](#_Toc46748625)

[2.0 Requirements 4](#_Toc46748626)

[2.1 User Requirements 4](#_Toc46748627)

[2.2 Software Requirements **Error! Bookmark not defined.**](#_Toc46748628)

[2.3 Use Cases 5](#_Toc46748629)

[3.0 System Components and Software Design 7](#_Toc46748630)

[3.1 System Components **Error! Bookmark not defined.**](#_Toc46748631)

[3.2 Software Design **Error! Bookmark not defined.**](#_Toc46748632)

[4.0 User Interface Design 11](#_Toc46748633)

# System Vision

## Problem Background

The Australia NSW Traffic penalty data 2011 – 2017 includes information on traffic violations monthly, face value, and related statistics for a specific region. The data covers a range of offences, such as speeding, red light violations, and more. This data was collected and managed by government agencies responsible for traffic enforcement and safety.

## System Overview

To analyse and manage traffic penalty data effectively, we will develop software tailored to the specific needs of authorities and researchers. The software will contain the following key components:

**Data Collection:** The software will collect and aggregate data from various sources, such as law enforcement agencies, traffic cameras, and court records. The data also includes details of the violation, location, date, and fine amount.

**Data Storage and Management:** The software will have a secure database to store and manage the collected data. The database is going to be scalable and capable of handling large volumes of historical and real-time data.

**Data Analysis:** We will Implement algorithms and analytics tools to process and analyse the data. This involves identifying trends, hotspots for traffic violations, repeat offenders, and more.

**Reporting and Visualization:** We will design a user-friendly interface for generating reports and visualizing data. This can help authorities and researchers gain insights from the data quickly and easily through charts, graphs, and interactive dashboards.

**Automation:** We will implement automation for tasks like issuing penalty notices, sending reminders to violators, and generating statistical reports. Automation can streamline administrative processes and reduce human error.

**Compliance and Security:** We will ensure that the software complies with data privacy laws and security standards to protect sensitive information.

## Potential Benefits

Developing a software to analyse the Australia NSW traffic penalty data from 2011-2017 can have several potential benefits such as:

**Improved Traffic Safety:** By analysing historical data, authorities can be able to identify areas with a high frequency of violations and take proactive measures to improve road safety in those locations.

**Efficient Resource Allocation:** The software can help authorities allocate resources more effectively by focusing on areas with the highest need for enforcement.

**Revenue Generation:** Efficiently managing traffic penalties can lead to increased revenue for the government, which can be reinvested in road infrastructure and safety programs.

**Data-Driven Policy Decisions:** Policymakers can use the insights gained from the software to make data-driven decisions about traffic regulations and enforcement strategies.

**Reduced Administrative Burden:** Automating administrative tasks can reduce the administrative burden on law enforcement agencies, making the penalty process more efficient.

**Transparency:** Providing access to data and reports can enhance transparency and public trust in traffic enforcement.

**Research and Education:** Researchers can use the resulting data to conduct studies on traffic behaviour and safety, aimed at improving education and awareness campaigns.

# Requirements

## User Requirements

The user requirements are the needs and expectations of different stakeholders who will interact with the software. In the case of the Australia NSW traffic penalty data management and analysis software, the following stakeholders will be included:

Government Authorities:

* Need access to comprehensive reports and dashboards for policymaking and resource allocation.
* Require tools for monitoring and managing traffic violations and penalties.
* May need automated penalty issuance and payment processing.

Law Enforcement Agencies:

* Require tools for recording and documenting traffic violations.
* Need access to real-time data for patrolling and enforcement purposes.
* May need mobile applications for officers in the field to input data.

Research and Analysis Teams:

* Require access to raw and aggregated data for research and analysis purposes.
* Need data export and integration capabilities with statistical analysis tools.

Citizens and Violators:

* May need an online portal or mobile app to view and pay penalties.
* Require a user-friendly interface for understanding violations and associated fines.
* May need options for contesting penalties or requesting leniency.

Data Privacy and Security Teams:

* Need to ensure compliance with data privacy laws and security standards.
* Require features for user authentication and access control to protect sensitive data.

## Software Requirements

Software requirements describe the technical specifications and functionalities of the software. For the NSW traffic penalty data management and analysis software, the following key software requirements shall be included:

Database Management:

* Database for storing historical and real-time traffic penalty data.
* Data normalization and indexing for efficient querying.

User Interface:

* Intuitive user interfaces for different user types, including administrators, law enforcement, and citizens.
* Data visualization tools for generating reports and dashboards.

Data Integration:

* Integration with external data sources, such as traffic cameras, court records, and government databases.
* Data import and export capabilities for research and analysis.

Automation and Workflow:

* Automation of penalty issuance, reminders, and payment processing.
* Workflow management for handling contested penalties and appeals.

Security and Compliance:

* Robust security measures to protect sensitive data.
* Compliance with data privacy laws (e.g., GDPR, HIPAA) and local regulations.

Scalability and Performance:

* Scalable architecture to handle large volumes of data.
* Efficient data processing and reporting.

## Use Cases & Use Case Diagrams

Use cases describe specific interactions or scenarios in which users will engage with the software. Here are some use cases for the Australia NSW traffic penalty data management and analysis software:

Traffic Violation Recording: Law enforcement officers use the software to record and document traffic violations, including date, time, location, and offender details.

Penalty Issuance and Payment: Automated penalty issuance based on recorded violations, followed by penalty payment by violators through an online portal.

Data Analysis and Reporting: Government authorities and researchers use the software to analyze historical data, identify trends, and generate reports for policymaking.

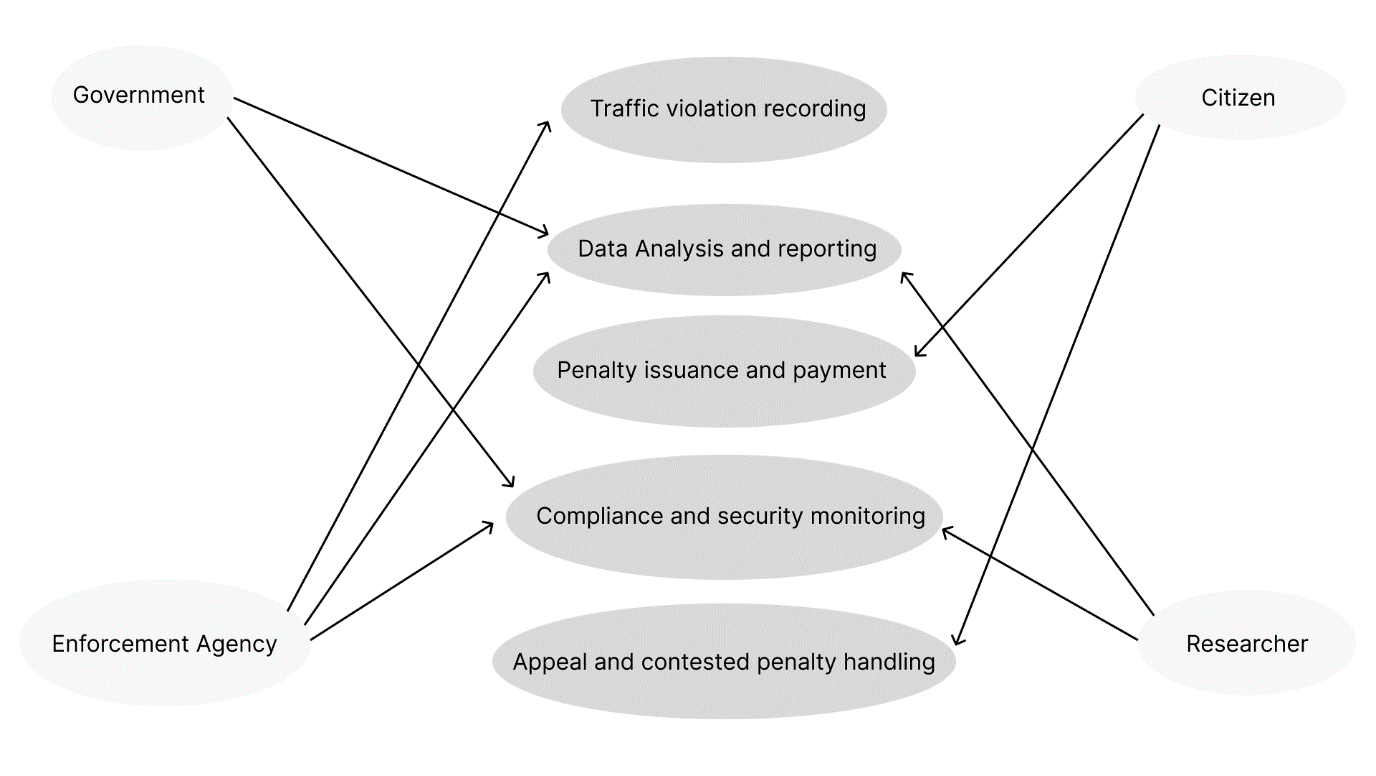
Citizen Access: Citizens and violators access the software to view penalty details, fines, and payment options.

Compliance and Security Monitoring: Data privacy and security teams monitor user access and ensure compliance with regulations.

Data Integration and Synchronization: Integration with external data sources (e.g., traffic cameras) to keep the database up to date.

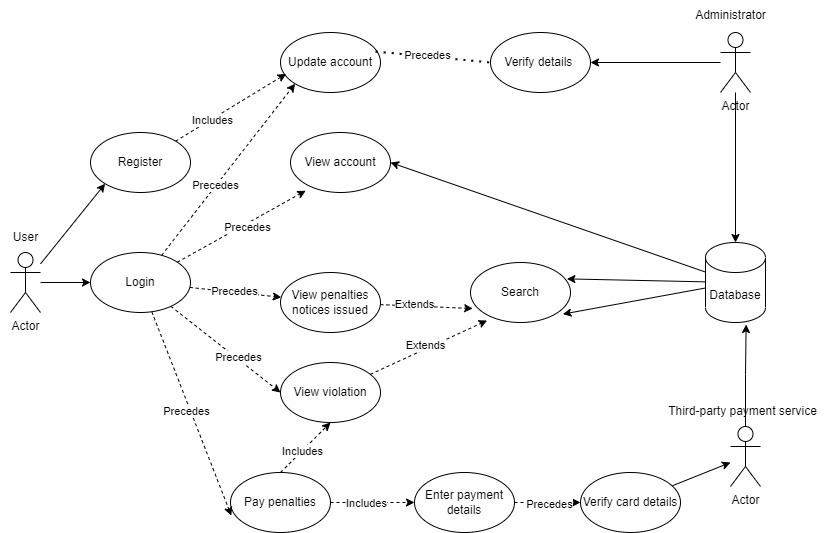
Appeal and Contested Penalty Handling: Handling appeals and contested penalties, including documentation and communication with relevant parties.

AUSTRALIA NSW TRAFFIC PENALTY USE CASE DIAGRAM

Fig 2.3a

# Software Design and System Components

## Software Design



## System Components

### Functions

createAccount(): It is called when users sign up a new account. They are required to give personal information which then will be stored in the database if all details are acceptable.

* Name: str – name of user.
* UserId: str – userIDof user.
* Birthday: date – birthday of user.
* Address: str - address of user.
* Phone\_number: str – phone number of user.
* Email: str – email of user.
* Password: str – password of the account.
* Driver\_license\_number: str – the number of the user’s driver license.

Once users finish signing up, the values of variables are stored in the database. A new record is made and the function does not need to return any value (void) but should notify users whether the account is successfully created or not.

checkAccount(userID, password): This function requires 2 parameters which are userID and password. When users log into the software, it will compare userID and password of the user to the details that they had registered. If they are correct, the function will return true, otherwise, it will return false.

updateAccount(userID, address, phone\_number, email, password): This function has 5 parameters. It will update this information in the database and change the value of the variable which is different from its previous value. This function does not need to return any value (void) but should notify users whether the account is successfully updated or not.

showAccount(userID): This function requires a parameter which is userID and returns an Account that has the same userID from the recordings of the database.

showAllViolation(userID): This function returns all the recordings of violations that the user has based on the userID in the database. They include both paid and unpaid offences. This function returns a list of all violations.

showViolation(userID): This function requires a parameter which is userID and returns a Violation that has the same userID from the recordings of the database which has not been paid yet. This function returns a list of unpaid violations.

addViolation(userID, offence\_code, fine\_amount, details): This function is called when the user commits another offence.

* userID: str – a unique identifier of the user.
* Invoice\_ID: str – a unique identifier for the invoicement.
* Offence\_code: int – a unique identifier of offence.
* Fine\_amount: int – The penalty amount for the offence.
* Date: date – the date that the user committed the offence.
* Details: str – the description of the offence with different degrees.

This function will not return a value (void). However, it will update the database after being executed.

searchViolation(userID, start\_date, end\_date): It shows all violations of the user from the start date to the end date.

* userID: str - used to search the user in the database.
* Start\_date: date – the beginning of the searching period.
* End\_date: date – the end of the searching period.

This function will return a list of violations in the range between the start date and the end date in the recordings.

showPenaltyNoticeIssued(): This function shows all the penalty notices issued in NSW. It does not need an input parameter. When this function is called, it will display all the recordings of the offences from the database. In other words, it returns a list of offences.

addPenaltyNoticeIssued(offence\_code, offence\_month, offence\_finyear, offence\_desc, fine\_amount): This function is called when a new offence is approved and issued to add it into the system of penalty notice issued.

* Offence\_code: a unique identifier for the offence.
* Offence\_month: the day that the penalty notice is issued.
* Offence\_finyear: the financial year of the penalty notice.
* Offence\_desc: the description of the offence.
* Fine\_amount: the penalty amount for the offence.

This function will not return a value (void). However, it will update the database after being executed.

searchPenaltyNoticeIssued(start\_date, end\_date): It shows penalty notices issued between the selected time.

* Start\_date: the beginning of the period.
* End\_date: the end of the period.

This function returns a list of offences issued in the range selected.

### Data Structures / Data Sources

Classes:

A diagram of a computer

Description automatically generated

Data structures: Lists

Accounts[]: this list is used to hold the values of all accounts of users whenever it is created. Because the value of the list will be changed when a new user signs up for a new account, the list data type is a good choice rather than using an array. A member of the list is an account, which has the following attributes: id(str), name(str), address(str), phone\_number(str), email(str), password(str), billing\_address(str) and driver\_license(str).

Function using accounts[]: createAccount(), checkAccount(userID, password), updateAccount(userID, address, phone\_number, email, password), showAccount(userID).

Violations[]: a list of all violations. It is appended when users commit a new offence. The member of the list is a violation, which has the following attributes:

* defendant\_ID(str): ID of the user who committed the offence.
* Invoice\_ID: str – a unique identifier for the invoicement.
* Date\_committed: date – the date that the user committed the offence.
* offence\_code(int): the unique code identifying the offence.
* fine\_amount(int): the penalty amount for the offence.
* details(str): the description of commitment.

Functions using violations[]: showAllViolation(userID), showViolation(userID), addViolation(userID, offence\_code, fine\_amount, details) and searchViolation(userID, start\_date, end\_date).

Offences[]: a list of all offences. It is appended when a new penalty notice is issued. The member of the list is an offence, which has the following attributes:

* Offence\_code: a unique identifier for the offence.
* Offence\_month: the day that the penalty notice is issued.
* Offence\_finyear: the financial year of the penalty notice.
* Offence\_desc: the description of the offence.
* Fine\_amount: the penalty amount for the offence.

Functions using offences[]: showPenaltyNoticeIssued(), addPenaltyNoticeIssued(offence\_code, offence\_month, offence\_finyear, offence\_desc, fine\_amount) and searchPenaltyNoticeIssued(start\_date, end\_date).

### Detailed Design

Data Structure SearchResult:

List of Rows //Each row is a dictionary or object with column names as keys  
Function searchTableByColumns(tableName, searchValue1, searchValue2):

result = an empty list of SearchResult

//Execute SQL query to search for rows where either LOCATION\_CODE or SPEED\_BAND matches the search values  
 query = "SELECT \* FROM tableName WHERE LOCATION\_CODE = searchValue1 OR SPEED\_BAND = searchValue2;"

// Execute the query and store the result in the variable 'result'  
 result = executeSQLQuery(query)

return result

# User Interface Design

For the initial user interface design, we used some digital design tools to create a foundation for the user interface (UI) and user experience design (UX). The following tools are what contributed to this design phase:

4.0.1 Figma: Figma was the most essential tool for us to make a collaborative design with team members in real-time and be able to gather feedback from stakeholders. Using Figma, we accomplished the completion of creating a prototype of the software package.

4.0.2 User research: To make design decisions, we conducted user research using surveys, interviews, and analytics platforms which helped us to gain an insight into user preferences, pain points, as well as behaviours.

During the design phase, these were our key findings:

User Needs and Pain Points: Through user research, we succeeded in identifying the primary needs and pain points of the target audience. These insights helped us prioritize features and design elements that addressed these issues effectively.

Competitor Analysis: Analysing competitors' interfaces (e.g. [fleetster](https://www.fleetster.net/fleet-software/fleet-management/traffic-fine-management)) provided valuable insights into industry standards and best practices. It allowed us to identify opportunities for differentiation and innovation.

User Personas: Creating user personas helped us to understand the diverse user base and tailor the design to suit their specific needs, preferences, and behaviours.

Content Hierarchy: Establishing a clear content hierarchy was crucial to ensure that users easily access the most important information.

Accessibility: Ensuring that the interface was accessible to a wide range of users, including those with disabilities, was a key consideration. We also considered accessibility guidelines and tools to make informed design decisions to meet the needs of those with a disability.

Usability Testing: Conducting usability testing with prototype versions of the interface allowed for iterative improvements. User feedback was invaluable in refining the design and addressing usability issues.

Visual Design Trends: Staying up to date with current visual design trends and guidelines helped us to ensure that the user interface looked modern and appealing to users.

## Structural Design

**1. Dashboard:**

Overview: The dashboard serves as an overview of the system's key metrics and alerts.

Design Choice: Using a dashboard ultimately ensures that users can quickly assess the system's status and any urgent issues.

**2. Menu and Navigation:**

Main Menu: The main menu consists of primary navigation options such as "Dashboard," "Citations," "Payments," "Reports," and "Settings."

Design Choice: A clear main menu simplifies navigation, making it easy for users to access the core functionalities of the software.

**3. Citations Management:**

Citation List: This section displays a list of all issued citations, sortable by various criteria.

Citation Details: Users can click on a citation to access detailed information, including the violator's details, violation type, location, and fine amount.

Design Choice: Grouping citation-related features together streamlines the process of managing and reviewing violations.

**4. Payments:**

Payment Processing: This section enables users to process payments for fines, with options for various payment methods.

Payment History: Users can view payment history and transaction details.

Design Choice: Separating payment-related functions ensures that users can quickly access and process payments without distractions.

**5. Reports:**

Standard Reports: Users can generate standard reports on citation statistics, revenue, and trends.

Custom Reports: Advanced users can create custom reports with specific parameters.

Design Choice: Providing both standard and custom reporting options caters to different user needs and levels of expertise.

**6. Settings:**

User Management: Administrators can add, modify, or remove user accounts with different access levels.

Configuration: System settings and preferences, including fine categories and notification settings.

Design Choice: Centralizing user management and configuration options in one place simplifies system administration.

**7. Search and Filters:**

Search Bar: A search bar which allows users to find citations or violators quickly.

Filtering Options: Filters based on date, location, and violation type which would aid in narrowing down search results.

Design Choice: Robust search and filtering options enhance usability, especially for users dealing with a large volume of citations.

**8. Notifications:**

Alerts: The system provides notifications for overdue fines, system updates, and other important events.

Design Choice: Notifications ensure that users stay informed and can take necessary actions promptly.

**9. Accessibility and Compliance:**

Accessibility Features: The software follows accessibility standards, making it usable by individuals with disabilities.

Compliance: It complies with data privacy and security regulations, ensuring the protection of sensitive information.

Design Choice: Prioritizing accessibility and compliance is essential for legal and ethical reasons as well as making software available for use by all users including those with a disability.

**10. Help and Support:**

Documentation: Access to user manuals, Frequently Asked Questions (FAQs), and video tutorials.

Contact Support: Users can contact customer support for assistance.

Design Choice: Offering help and support resources enhances user self-sufficiency and satisfaction.

The structural design promotes ease of use, efficiency, and a logical flow for law enforcement officers issuing citations or administrators managing fines and reports. Grouping related features together and offering intuitive navigation options contribute to a positive user experience while ensuring compliance and security.

## Visual Design

**Login page:** Before going to the management software, the user must sign in/ sign up for their account. The value of userId and password then be checked with the database. If they are accurate, users can access the software and vice versa.

A login screen with a car and a logo

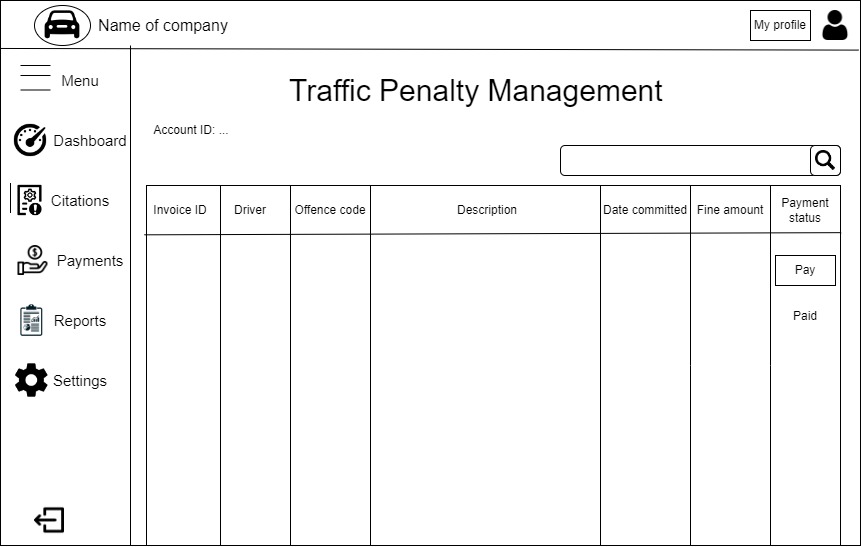
Description automatically generated

**Home page:** This is the initial page (dashboard) when users come to the software. It will show the notable information such as notification, number of offences they are having at the time, the total amount of fine they must pay and the total amount of paid fine.

A screenshot of a traffic penalty management

Description automatically generated

**Citations page:** This page will show the details of all violations that the user has. In terms of payment status, when the fine has not been paid, user can click the pay button which leads the user to the payment page and proceeds with the payment. In contrast, if the payment is already done, it would be a text that tells the user that the fine is paid.



**Payment page:** On this page, the fine details are presented at the top of the page to help the user check if the information is correct before paying the fine. Users must input the card details and billing address for the payment. When users pay for the first time, the software will ask if they want to save the payment details for the next transaction to save the user’s time.

