# IA2 project

## Explore

|  |  |
| --- | --- |
|  | |
| Develop: | |
| |  |  | | --- | --- | | Databases and code | Data and constraints | | To complete the functionality of the website, programming is required using a variety of languages. It would use html for the content, CSS for the styling, and PHP and SQL for manipulating and displaying the data within a database. The data contained within the csv file will be divided into several normalised data tables to improve the maintainability of the database while requiring more complex SQL syntaxis. | When putting the data into the database, there are several data types that define the functionality of the field. Primary keys are used to ensure that there are no duplicate entries, which is often auto incrementing. The most common fields are integers for numbers and VARCHAR for text and BOOL for true or false often used to ‘delete’ rows without removing them to reduce risk of deleting important data. However, due to the unimportance of the data, this is not necessary for the application. VARCHARs would be used for the names of locations and crimes while integers are used for the keys and for the number of reported crimes. |   The above dataflow/entity relationship diagram demonstrates how the data would be taken from the database alongside the inputs taken from users. | |  | | --- | | criteria | | Prescribed The web application complies with   * government web design standards * the *Australian Privacy Act* (1988) * Australian accessibility standards   Users can upload their own reports.  Comply with copyright.  include a link to the legal disclaimer at [www.qld.gov.au/legal/disclaimer](http://www.qld.gov.au/legal/disclaimer) on the landing page.  Upload data using csv file.  Data collected from the Queensland Government Open Data Portal  Display the data for at least one month.  Emailed reports outlining the severity of the current crimes in their area.  Users are eligible to register personal information with the site. Self-Determined Efficient code.  Accurate code.  Using algorithms to insert the data automatically.  Allow people to search for individual data sets outside the emailed reports to improve the functionality of the site. Constraints 7 weeks  Disclaimer linked to the website.  A working application must be made. | | | Useability principles | | | For the website to be an effective learning tool, useability principles must be used to make the website more accessible, which is crucial to the target audience of young children. Below are the ways that the website uses each of the useability principles.  Utility: Minimal options: html, Responsive design: CSS  Safety: data is not inserted through the user, but instead through a person of authority  Learnability and effectiveness  CARP   * Contrast: uses dark cyan colours with white to highlight the content sections. Essential information is displayed with a high contrasting colour to draw attention. * Alignment: the content is aligned in two columns or a single column in the middle of the page * Repetition: The text for headings is the same size across the webpages. The colours are the same, and so is the formatting between the nav bar, the headings, and the content. * Proximity: every element is centred to the page and portrayed within the same block * These are created using CSS styled html. | | |

## Develop: User Interface

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Overall design principles: | | | | | The website’s content uses high colour contrast to highlight the main information. The content uses thick teal borders with white backgrounds contrasted against the dark gray background of the page, making the content easily visible following the useability principle of learnability. Within searched results, the severity is highlighted by its colour: white for mild, yellow for moderate and red for severe. The forms are generally centered in the page using the same contrast, being organised to the center with clear lines. | | | | | Displayed data: | | | | |  |  | |  | | The displayed data was decided to be organised within a simple table so that the information that the user desires is easy to find. Although it is similar to raw data which may be harder for users to understand, the table format displays the information in the clearest way compared to other tested methods such as within a sentence (there were $amount reported cases of $crime in $location on the $date), which may be confusing and unclear. Seen in the above diagrams, the background of the entire table changes to visually signify the severity of the reported crimes making it more readable for users and thus improving the accessibility of the site. | | | | | Form for registering users to the site: | | | | |  | | The form for registering users includes personal information regarding the email, and so a privacy statement is required within the site to warn the user of possible security breaches. When the user submits the form, the data would need to be validated to ensure that the email is valid, there is not an account already for that email and if the password is of sufficient length. The warning level and location utilise dropdown boxes similar to the other forms to reduce the risk of inputing inaccurate results from spelling errors or locations that are not registered with the site. | | | Discaimer and privacy statement: | | | | | The disclaimer that would feature on index.html is taken from the Queensland Government at [Disclaimer | Queensland Government (www.qld.gov.au)](https://www.qld.gov.au/legal/disclaimer)  The privacy statement is also taken from the Queensland Government site at: [Privacy | Queensland Government (www.qld.gov.au)](https://www.qld.gov.au/legal/privacy) | | | | | |  |  | | --- | --- | | User interfaces | | | Index.html | InputData.html | |  |  | | In the index page, the top five results are put into a sidebar sorted by the most recent. To visually signify the severtity of the crime, the table’s background would change from white to yellow to red accordingly. To be easy to read, the data is placed into a table using clean lines and evenly sized rows sorting the information clearly to the user. | When inserting the data, an algorithm would be required to validate if the data is a csv file, and if it follows the same table format as required for the database to reduce the risk of destroying the database with incorrect or inaccurate data entries. Before inserting the data, the file would be opened so that the admin could compare the format of the csv file to confirm the inserting of the data. | | Search.html | Site map | |  |  | | The forms for location, crime, date and warning level for the search and register pages uses the information from the database pasted into a dropdown list to ensure that the inputed data is correct. For other forms in the register an login, an algorithm to validate the data is required. | The site map shows the relationship between the webpages. Since every page is accessible from every other page (outside of validation pages) the site does not require breadcrumbs for accessibility. | | Profile.html |  | |  | The profile page would allow the user to change the information tied to their account and also delete their account if they wish. When deleting the account, to reduce the risk of accidentally pressing the button, a warning would be alerted to the user improving the accessibility of the site. | |

## Develop: UI and data

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | The data within the website is obtained from the Queensland data portal. The website would use csv files to automatically input the data using algorithms. | | | Data types from the csv file: LGA Name: since this value is plain text, it would be inserted into the database as a VARCHAR value.  Date: although the values are strings, it would be formatted and inserted into the database as a date value so that it would be more useable with SQL queries.  Crime Types: the crime types, also being in the form of text, would be inserted into the database as a VARCHAR.  The records: since the records are numbers, they would be inserted into the database as INTEGER values. | |  | | The data is in the format of a csv file and so the code would use the fgetcsv function to loop through the results in an array. | | When processing the data, the date would need to be formatted from MD in the csv file to yy/mm/dd for the database. This would require manually identifying the first year in which the data was uploaded for a more useful solution. Within excel, the date has an automatically filled year which is incorrect since it starts at 2001, and so it must be changed. Since the LGA Name column has multiple repeating entries, the algorithm would need to check if the next row is different from the previous row, then the data would be inserted into the database, otherwise the table would be cluttered with repeat entries. Using an algorithm to insert the data would also allow for better maintainability of application making it more reliable for users.  Since there are concerns about misinformation regarding inputting false reports, users would not be eligible to insert reports of their own, despite the requirements mentioned in the criteria. Instead, the users would be directed to their local government agency who would be responsible for inputting the data. This would reduce the overall usage of the site while not fulfilling the requirements from the concept proposal, however the increased reliability of the results is deemed much more important. | | | Use of the Limits Using limits when inserting the data increases the efficiency of the site by not inserting unnecessary data. It would increase the speed of the code while also allowing better admin control over which sets of data are inserted into the database. This is required for the data since each number is its own row in the database, and so there would be millions of rows inserted at once with no limits, overloading the server | |  | Within the tables, foreign keys are used to join the tables. The foreign keys within the tables are:   * crimeID from records, matching with the CrimeID in the crime table * location ID from records, matching with ID from the location table * location from users, matching with ID from location   These foreign keys improve the maintainability of the data through reducing the amount of columns needed to modify if an admin found some results inaccurate. However, this requires additional code to join the tables and get the relevant varchar field for display.  The primary keys within the fields are:   * CrimeID * ID (from location) * ID (from records) * ID (from users) | | The tables were generated using XAMPP. Within the records table, the alert type was deemed to best fit an integer rather than a series of Booleans to reduce the amount of data in the database. When querying, the severity of the crimes are as follows: 0 = mild, 1 = moderate, 2 = severe. The database was designed to be nominalised to suit the above data while also being maintainable and secure. Within the code, instead of using SQL joins to get the VARCHAR values from the tables, it would be queried separately to then be placed in the final query. This reduces the efficiency of the code but ensures that the results from the query are accurate to better suit the users, reducing the risk of displaying incorrect information. It also returns the result in a separate variable which could be used in other algorithms and is not limited to a single array, particularly if the field names in different tables are matching. | |

## Develop: data

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | SQL | | | | | | Since the same SQL commands would be used across multiple scenarios, the statements would utilise PHP variables and placed within a separate included file. This would also increase the maintainability of the code since a change in one file would influence the entire site. However, it opens the site for cyber-attacks and so the queries would be prepared before execution to reduce privacy risks. For more unique statements, the code would be written manually without the use of external code segments.  $INSERT = "INSERT INTO $tablename ($columnName) VALUES ('$value')"; (adds the data into the table based on variables defined within the csv file, but only for the location and crime tables. The records table uses a more specific query)  $CHECKIFEXISTS = "SELECT $columnName FROM $tablename WHERE $columnName = '$value'"; (selects data to check if there are any results, this is used to ensure there are no duplicate entries and is used when inserting data. Additionally, this same code could be used when selecting users to email results utilising the WHERE clause)  $SELECT = "SELECT $columnName FROM $tablename"; (this would be used across the site for several functions, such as the dropdown lists for the forms and searching for results. | | unique SQL statements:  SELECT \* FROM records ORDER BY 'date' LIMIT 5 (used to show the top five most recent records in the home page)  "SELECT \* FROM users WHERE users.email = '".$email."' AND users.password = ' ".$password."'" (Checks whether there is an account using the same email and password based on the users inputs to log them in)  "INSERT INTO `users` VALUES (null, ".$email.",".$password.",".$location.",".$warning.")" (Used to register a new user in the site from register.php by inserting the values posted from a form  "INSERT INTO $tablename VALUES(null,$crimetype,($x-1),$number,$date)" (inserts data into the records table with $crimetype, $number and $date taken from the file and $x is a FOR loop variable) | SQL to create database:  CREATE DATABASE IA2DB  CREATE TABLE crime{  CrimeID int autoincrement,  crime\_types varchar(255)  }  CREATE TABLE location{  ID int autoincrement,  Location varchar(255)  } | CREATE TABLE users{  ID int autoincrement,  email varchar(255),  password varchar(255),  alertType int,  location int  }  CREATE TABLE records{  ID int autoincrement,  locationID int,  crimeID int,  number int,  date date  } | | crime table sample data | location table sample data | users table sample data | records table sample data | | |  |  |  |  | | | The above tables demonstrate how the data for the website is stored within the database. Seen in the records table, there is a data entrie for every value in the csv file and, since the csv file has millions of data entries, the website cannot hold all the necessary data. So, limits would need to be used when inserting the data alongside removing old results to ensure that the website is as optimised as possible. Within the users table, the email and password sections are not encrypted and may provide safety issues to users. However, since the only personal information is their email, it is not necessary but recommended for a future update.  When utilising the data from the database, MySQLI would be used over PDO. MySQLI was chosen since it is simpler to comprihend in comparison to PDO, functioning in a similar way to other programming languages such as C# while PDO is entirely different. Additionally, there are more resources relevant to the sites production using MySQLI over PDO, and so to better understand, debug and implement the code, MySQLI was the better option. | | | | | |

## Develop: Algorithms

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Required algorithm for severity: | Algorithm for inserting data into the database from a CSV file: | | | | Since the original algorithm used a tiered warning, the variable used would be an int. A string is not used since it is more likely to cause errors from misspells, and so to remove unnecessary validation, an int is more accurate. Since there is no standard to measure the difference in the proposed algorithm, the warning would be allocated depending on the number of cases.  The pseudocode for the algorithm is as follows:  SET warning = 0  GET crimeData  If crimeData >= 5  SET warning = 2  Else if crimeData>=3  SET warning = 1  Else if crimeData>=1  SET warning = 0 | Crimes table | Location table | Records table | | GET csvData  For(x=2;x<count(csvData);x++)  SET value = csvData[x]  QUERY if data already exists  If(Query ==0)  INSERT data  PRINT inserted data  Else  return  CLOSE csvData  This function automatically runs when data is posted from a form that only accepts csv files. This code gets every dataset from the first row, excluding the first two columns (location and date) while also checking for duplicate entries. | GET csvData  WHILE rows in csvData  SET value = csvData[x]  QUERY if data already exists  If(Query ==0)  INSERT data  PRINT inserted data  Else  return  CLOSE csvData  This function automatically runs when data is posted from a form that only accepts csv files. This code loops through the first column of every row in the csv file, which contains the location data, while ensuring no duplicate entries are inserted. | SET lowerlimit, upperlimit  SET num = 0  GET csvData  WHILE data = csvData AND lowerlimit <= num AND upperlimit>=num  SET date = data[1]  CONVERT date to yy/mm/dd  For(x=2;x<count(csvData);x++)  SET number = data[x]  SET severity = (severity algorithm using number)  SET result = SELECT all from crime where crime.id = (x-1)  SET row = fetch(result)  SET crimetype = row[0]  SET Location = data[0]  Send email using severity, date, number and crimetype and location  If(number != 0)  INSERT data using severity, date, number, crimetype and location  PRINT inserted data  Else  Reject data  SET num = num + 1  CLOSE csvData  This function automatically runs when data is posted from a form that only accepts csv files. The upper and lower limits would be manually set so that the necessary data fields are inserted. This also increases the speed of processing the data and reduces the risk of overloading the server, preserving the data. To reduce the amount of unnecessary data in the database, the IF(number!=0) ensures that all entries with no reports are ignored, registering only the relevant data. This also improves the ease of use of the search function by reducing the available option through limiting the datapoints. The function also triggers the algorithm for emailing the user. | | Algorithm for emailing user | Algorithm for registering and logging in users | | | GET severity, date, number, crimetype, location when inserting data  SET result = SELECT all from users WHERE alerttype = severity AND users.location = location  SET string = “”  [same algorithm for severity, setting the string to “severe” instead of 2, “moderate” instead of 1 and “mild” instead of 0  WHILE row = fetch(result)  Email = row[`email`]  SET message using number, date, crimetype  SEND EMAIL to the user with the message  This code sends an email to users that are within the same region of the crime and if it fits within their alert type. A similar algorithm for the severity is used but instead of an int, it is a string that can be sent via the email. This function is triggered when a new record is inserted into the database. | GET form data (email, password, alerttype, location)  VALIDATE the data in the form by:  Checking if the email is valid;  Check if every field has a sufficient value;  For password, check if it is the required length of 8 characters without spaces;  IF any of these are not fulfilled, alert the user.  SELECT all results from users where the email is the same  IF(results ==0)  INSERT into users with email, password, alerttype and location  Change webpage to profile.  ELSE  ALERT user to use a different password  This code validates the data within the form to then decide if the inputs are sufficient. It also validates if an account was already made for the user to avoid having multiple accounts for one email.  Login:  COMPARE username and password with database  Proceed to profile page  Else: display error message | | | Currently, there is no algorithm to allow users to insert data of their own since it would cause risks to the integrity of the data if there were no outside validation to deem if the reported crime happened.  Instead:  DISPLAY location details USING users location data FROM a new data field that contains the locations detail | Severity algorithm for future updates utilises a modifier to quantify the severity of each individual crimes.  SET warning = 0  GET crimeData, modifier  If crimeData >= 5\*modifier  SET warning = 2  Else if crimeData>=3\*modifier  SET warning = 1  Else if crimeData>=1\*modifier  SET warning = 0 | | Due to time constraints, the functionality for the profile page is not developed. However, the functionality is as simple as using a few extra SQL commands with PHP as demonstrated in the other algorithms. The queries would use DELETE to delete the account and ALTER to update the user’s information with the data filled out in the form. | |

## Generate: PHP webpages prototype

|  |
| --- |
| The nav bar utilises the bootstrap styling since it is more casual, and so more accessible for the target audience in comparison to the Government template.  The profile page contains forms that the users can select to update their account details as per privacy standards. To improve the page, an algorithm could be used to automatically select the results that are within the account to reduce the risk of accidently changing a value they did not want to.  The webpages are all accessible from the nav bar; therefore, breadcrumbs are pointless.  The arrows do not represent all connections between webpages for simplicity since all webpages connect to each other with the same navbar, so only the index.php page connections are shown. |

## Generate: Evaluation



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Criteria | Evaluation and recommendations of criteria | Extensions | | Useability principles | The useability principles were considered within designing the forms and webpages. However, the website does not contain many accessibility features such as keyboard controls. It also has no safety in ensuring that users data is secure despite not requiring crucial information. Some people use the same password across multiple accounts, and so although it is not personal information it may reveal their information indirectly by giving attackers access to accounts in other websites.  The CSS styling for the form buttons could be updated to make the input fields all the same size and equally lined up to be more visually pleasing. However, the forms are still simple, and this does not have much impact on the useability of the site. | Since the website does not currently fulfill the requirement of allowing users to add their own reports, the site add this functionality in the future. It is better to give direct links to their LGA based on the location in the user database than to allow direct reports for safety considerations. This would, however, require additional data such as the contact details of the agency which the current data does not have.  With the current inserting data functionality, there is no login page for admins to access and so any user could input fake data. For future updates, an admin login form like the user login form must be added to improve the integrity of the data. | | Efficient, accurate code | The code within the website runs quickly and efficiently. When inserting the data with algorithms, the code is inefficient which was remedied with the limits. However, rewriting the code to fill each table in one function rather than a function for each table would greatly increase the speed of the code while also becoming more maintainable. | | Allow users to search for individual data | The functionality for this was properly implemented into the website. The forms were designed to easily search the data using dropdown forms and the results were outputted into easy to read, colour coded tables. However, there is no function to allow users to search for multiple locations or crimes at a time, and so the function is only limited to one result per search which is inefficient. While it can search for data for one month as required, it lacks the detail to be properly useable for the police to find locations to deploy and has limited useability for the users. This could be expanded upon within an update to the site. | Improve the cybersecurity of the site. Outside of preparing SQL statements, there is little security within the site which could be detrimental to users if their personal information gets leaked. Additionally, cyber-attacks could compromise the integrity of the results making the website a means to spread disinformation to users. Since this is only a prototype, cybersecurity is unimportant and so was not implemented in the development. However, for the final build, cybersecurity must be taken into consideration.  Cybersecurity measures could be encrypting passwords and emails, validation emails for creating accounts and creating a “lost your password” function so that users are not locked out of their accounts which could constitute as spam emails from the site since they cannot turn it off. | | Limitations of the data | The data does not display accurate locations outside of the general region. Since the site is primarily for the neighbourhood watch, the locations lack the detail to properly tell users which suburbs the crimes take place. Instead, the site only warns the users from general crime rates in their city. This is a limitation of the data itself, and so the website cannot accommodate this unless more sufficient data is supplied to the Queensland Government Portal. | | Limitations of the warning system | The algorithm for the severity of the results is inaccurate. Since it warns based on the number of crimes, the severity of the crime itself is undistinguished within the site and so higher frequency mild crimes would be reported as severe while lower frequency severe crimes would be recorded as mild. To improve the accuracy of the warning system, the individual crimes could be weighted depending on their sentencing within the crime data table. For example, one case of murder would be measured as severe while one case of a car accident would be mild. | | Social impacts: | The website gives communities the ability to view the safety of their location and places they may plan to move to. Access to crime information that is easily accessible is helpful to civilians to protect themselves while giving the police sufficient information into which areas need more officers. Overall, the website has the capacity to improve the safety of communities and causing a better society.   * Neighbourhood watch members: they would receive the alerts that they could then pass onto their neighbourhood to improve the safety of everyone living in that location. * Police officers: they would have access to the amount of crime in each location, in which they can deploy police officers to places with the highest crime rates making the location safer. * General public: they have access to information that could help them in determining a suitable location to move into, or if they should move away to somewhere safer by using the search function.   Negative impacts:   * personal information may be leaked and may be detrimental to users who use the same passwords across multiple sites, compromising the security of several accounts. * Despite having a disclaimer, inaccurate information could defame some locations in the eyes of the users, being harmful to the communities which are falsely represented.   Legal: all the data is publicly available and there are not images used which would contain copywrite. The website also includes the standard privacy and legal disclaimers that are supplied by the Queensland Government themselves. It also complies with the Australian Privacy Act which allows users to modify their personal information. Although they cannot current change their email and password, the use for the accounts is minimal and can be counteracted by just deleting the account and making another one.  Economic: retail business owners can find places with lower crime activity, and so can position their firm in a place with the lowest risk of thievery and vandalization. This reduces the cost of replacing or fixing the damages caused by crime and so becoming more economically sustainable and competitive. | | |

## Generate: Testing



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Process | Test results | Accessibility guidelines checklist Page titles:  • must appear in the browser tab for all pages  • must be appropriate for the page  • must be different for each page.  Alt text:  • must be used for all content images (except decorative images)  • attribute is set to null for decorative images  • appropriately describes the content of the image to which it relates  • gets larger when pages are zoomed.  No images are used in the site, so not necessary.  Headings:  • are on every page (at least one)  • levels on each page have a meaningful hierarchy.  Zooming of pages:  • results in correct display of the page with no horizontal scrolling  • allows all buttons to remain visible.  Non-mouse navigation (keystrokes or tabs):  • of page is in a logical order  • allows access to all page elements.  Fields:  • in forms and other form controls have a visible label to allow interaction with voice input and increase  the clickable area  • that are mandatory are clearly indicated and do not rely on colour alone  • with required formats, such as dates (year, month, day), are clearly indicated. The format of the date is not indicated currently but could be implemented easily in the future.  Error messages (or validation messages):  • are clear and specific CSS styling for the error messages are required, which can easily be implemented in the future  • do not cause the form to be completely reset.  Accessibility options include:  • general instructions for user input at the top of the form or section to which they relate  • text transcripts provided for audio and video elements  • appropriate contrast ratio between text and background (colour contrast)  • a five-second time limit for all moving or flashing content, and the content can be disabled or controlled  by the user. The website has no flashing or moving content, and so it is not necessary. | | Searching for data | The searching data function works as expected. The results display in the proper field with the correct information and the relevant severity colour coding. When searching for results that do not exist, the website does not produce any errors and instead has a blank field. | | Inputting data from file | Inputting data for the file works as expected. The crime types and location tables fill with all the results in the file with the auto incrementing ID. However, the records table causes errors when inserting the data. Upon debugging, it was found that the issue is within the data system, with the PHP not converting the date to the relevant format. With testing on a simple string, the PHP function for converting the date would not work at all and so is not an issue with the code. Using echo, the other values that would be inserted into the table have been defined properly. For demonstration purposes, the data in the records table is fabricated. | | Registering and logging in users | The function for registering users works sufficiently. The validation for checking if there is already an account for the email works consistently and the data from the form is properly inserted into the database. The validation within the login page also works.  When attempting to create an account without filling out the form, the required error message is shown. | | Updating user data within the profile.php page | The functionality for this is yet to be implemented due to time constraints but could be developed in a future build with little difficulty. | | Inserting incorrect values | When attempting to insert incorrect values, the validation algorithms do not allow the form to be inserted and instead displays an error message as expected. With strings using exotic symbols, the mysqli\_escape\_string() function ensured that it would not cause any errors when attempting to insert into the database. | | Displaying the five most recent results in the home page | The code displays five results in the home page as expected. However, the order by date appears to not be functioning properly, ordering only by year but not the month and day and so the date may be better suited within a different format before querying to fix the functionality. | | Emailing users | The functionality for emailing users could not be tested since the site does not have a mailing server setup. However, theoretically, the code should function as expected since it follows similar code to other solutions available on GitHub which are proven to have worked. The message that is emailed could be updated, however, to provide a link to the webpage and reinstate the disclaimers to better fulfill the useability principles and comply with Queensland’s laws. | | Testing links | Every link goes to the required pages. | | Privacy of user data | The login functions use password validation to stop anyone from accessing their accounts. However, the database itself is not yet password protected and is using the default login settings. | | References Australian Government. (2022). *Privacy | attorney-general-department*. Retrieved from Australian Government Attorney-general-department: https://www.ag.gov.au/rights-and-protections/privacy#:~:text=The%20Privacy%20Act%201988%20%28Privacy%20Act%29%20is%20the,federal%20public%20sector%20and%20in%20the%20private%20sector.  GitHub. (2023). Retrieved from github.com  Queensland Government. (2022, March 9). *LGA Reported Offences Number*. Retrieved from Open Data Portal: https://www.data.qld.gov.au/dataset/lga\_reported\_offences\_number  Queensland Government. (2023). *Disclaimer*. Retrieved from Queensland Government: https://www.qld.gov.au/legal/disclaimer  Queensland Government. (2023). *Privacy*. Retrieved from Queensland Government: https://www.qld.gov.au/legal/privacy  w3schools. (2022). Retrieved from https://www.w3schools.com/  WebAIM. (2020, 7 21). *WebAIM: Quick Reference*. Retrieved from WebAIM web accessibility in mind: https://webaim.org/resources/quickref/#:~:text=Accessibility%20guidelines%20and%20techniques%20are%20based%20on%20four,technologies%2C%20mobile%20devices%2C%20old%20devices%2Fbrowsers%2C%20etc.%20Follows%20standards. | | | |

|  |  |
| --- | --- |
| For demonstration purposes, only the important PHP coded elements are shown since the html and CSS are insignificant to the overall functionality of the site. However, the rest of the code is shown in the video.Code for Index.php: <div class = "sideBox"style = "width: 20%; float:left;overflow: scroll;">    <?php  include 'Chunks/connect.php';  $num = 5;    $colors = Array("#247675","#2FA09E","#33B2B0","#24C1BF","#1AD0CE");  $severityColors = Array("#ffffff","#cccc00","#e60000");  $sql = "SELECT \* FROM records ORDER BY 'date' LIMIT 5";  $result = mysqli\_query($conn,$sql);  $x = 0;  // loop through the queried data  while ($row = mysqli\_fetch\_assoc($result)){  //get the values from the query  $number = $row['number'];  $location = mysqli\_query($conn, 'SELECT location FROM Location WHERE location.ID = '.$row['locationID'].'');  $crime = mysqli\_query($conn, 'SELECT crime\_types FROM crime WHERE crime.CrimeID = '.$row['crimeID'].'');  $row2 = mysqli\_fetch\_array($location);  $location2 = $row2['location'];  $row3 = mysqli\_fetch\_array($crime);  $crime2 = $row3['crime\_types'];  $index = 0;  if($number>=5){  $index = 2;}  else if($number>=3){  $index = 1;}  else{  $index = 0;}  $time = $row['date'];  //create a box to display the information  echo "<div style = 'background-color: ".$colors[$x]."; padding: 10px;border:4px solid;'>";  echo "<table style = 'background-color: ".$severityColors[$index]."'>";  echo "<tr><th>Amount</th><th>".$number."</th></tr>";  echo "<tr><th>Crime</th><th>".$crime2."</th></tr>";  echo "<tr><th>Time</th><th>".$time."</th></tr>";  echo "<tr><th>Location</th><th>".$location2."</th></tr>";  echo "</table></div>";  $x += 1;  }  ?>  </div> | Code for connecting to the database (connect.php): <?php  $servername = "localhost";  $dbname = "IA2Website";  $username = "root";  $password = "";  //make connection  $conn = mysqli\_connect($servername, $username, $password, $dbname);  // Check connection  if ($conn->connect\_error) {  die("Connection failed: " . $conn->connect\_error);  }  ?> SQL statements from external file (SQLstatements.php): <?php  //this file is included after the variables are defined  //ensure that there are no undefined variables for the include to work  if($value==null){  $value = "";  }  if($columnName==null){  $columnName = "";  }  if($tablename==null){  $tablename = "";  }  $SELECT = "SELECT $columnName FROM $tablename";  $SELECTDISTINCT = "SELECT DISTINCT $columnName FROM $tablename";  $INSERT = "INSERT INTO $tablename ($columnName) VALUES ('$value')";  $CHECKIFEXISTS = "SELECT $columnName FROM $tablename WHERE $columnName = '$value'";  ?> Navbar used in all webpages (NavBar.html) <nav class="navbar navbar-expand-lg navbar-light "style="border-left: 12px solid #247675;background-color:#D5D5D5">  <a class="navbar-brand" href="#">Neighbourhood Watch</a>  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">  <span class="navbar-toggler-icon"></span>  </button>  <div class="collapse navbar-collapse" id="navbarSupportedContent">  <ul class="navbar-nav mr-auto">  <li class="nav-item active buttonHover">  <a class="nav-link" href="index.php">Home <span class="sr-only">(current)</span></a>  </li>  <li class="nav-item active buttonHover">  <a class="nav-link" href="InputData.php">Upload data</a>  </li>  <li class="nav-item active buttonHover">  <a class="nav-link" href="SearchData.php">Search for data</a> |
| </li>  <li class="nav-item active buttonHover">  <a class="nav-link" href="register.php">Register</a>  </li>  <li class="nav-item active buttonHover">  <a class="nav-link" href="login.php">Login</a>  </li>  </ul>  <form class="form-inline my-2 my-lg-0">  <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">  <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>  The navbar uses bootstrap CSS styling because of its simpler and more casual design which better suits the website compared to the Queensland Government template.  </form>  </div>  </nav> Code for login.php <?php  include "Chunks/connect.php";  //only run the function if data is posted from a form  $run = isset($\_POST['submit']);  if($run){  //get values from the fomr  $email = $\_POST["email"];  $password = $\_POST["password"];  //run SQL to check if there are any accounts using both the email and password  $stmt = $conn->prepare("SELECT \* FROM users WHERE users.email = '".$email."' AND users.password = ' ".$password."'");  $stmt->execute();  $result = $stmt->get\_result();  $user = $result->fetch\_assoc();  $row = $result -> fetch\_assoc();  // Free result set  $result -> free\_result();  $conn -> close();  //check if there are any accounts with the same ewmail and password  if($row!=null){  //since the form posts to itself, php sessions are used to transfer the posted results to the profile page  session\_start();  $\_SESSION['post\_data'] = $\_POST;  $url = "profile.php";  $\_SESSION['run'] = true;  //relocate user to the profile page  header('Location: '.$url);  }  else{  echo "email or password is incorrect";  }  }  ?> Code for register.php <?php  include "Chunks/connect.php";  $run = isset($\_POST['submit']);  if($run){  //validate if all form elements are filled  if($\_POST["email"]!=null&&$\_POST['password']!=null){  echo "run";  $email = $\_POST["email"];  $email = mysqli\_real\_escape\_string($conn,$email);  //run SQL to check if there are any accounts with the same email  $stmt = $conn->prepare("SELECT \* FROM `users` WHERE users.email = '".$email."'");  $stmt->execute(); | $result = $stmt->get\_result();  $user = $result->fetch\_assoc();  //check if a user already exists and give an error message  if($user!=null){  echo "email already in use"; }  else{  $email = $\_POST["email"];  $email = mysqli\_real\_escape\_string($conn,$email);  $password = $\_POST["password"];  $password = mysqli\_real\_escape\_string($conn,$password);  $location = $\_POST["Location"];  $location = mysqli\_real\_escape\_string($conn,$location);  $sel = "SELECT ID FROM `location`WHERE location.Location = '".$location."'";  $res = mysqli\_query($conn,$sel);  $rowthing = mysqli\_fetch\_array($res);  $location = $rowthing['ID'];  $location = (int) $location;  $warning = $\_POST["warning"];  $warning = mysqli\_real\_escape\_string($conn,$warning);  $warning = (int) $warning;  echo $email;  echo $password;  echo $warning;  echo $location;  $nullVar = null;  //insert values into the database, registering a new user  $stmt = $conn->prepare("INSERT INTO `users` VALUES (?, ?, ? , ?, ? )");  $stmt->bind\_param('issii', $nullVar, $email, $password , $warning , $location);    The bind param is more cyber secure than just posting, which can be modified in the other code later for better security.  $stmt->execute();    //transmit the posted data to another webpage so that the profile page has the accurate data  session\_start();  $\_SESSION['post\_data'] = $\_POST;  $url = "profile.php";  $\_SESSION['run'] = false;  //relocate user to the profile page  header('Location: '.$url);  }  // Free result set  $result -> free\_result();}  else{  echo 'input an email or password';  }  }  ?> |
| Upload.phpEmail user function function emailUser($severity,$date,$number, $crimetype,$location){  global $email;  global $password;  echo".$email";  include 'Chunks/connect.php';  $sql = "SELECT \* FROM `users` WHERE alertTypre = '$severity'AND `location` = $location";  $result = mysqli\_query($conn, $sql);  $str1="";  Severity algorithm which returns a string so that it can be emailed to the user.  if($severity>=5){  $severitystr = "severe";  }  else if($number>=3){  $severitystr = "moderate";  }  else{  $severitystr = "mild";  }  while($row = mysqli\_fetch\_assoc($result)) {  $email = $row['email'];  $str2 = "we are emailing to warn you of a ".$severitystr." case of ".$crimetype." at your location.  The crime was recorded on the ".$date;  ini\_set("SMTP", "sendmail");  mail($email,"crime warning", $str);  print $str;  }  } Code for inserting data if (isset($\_POST['submit']))  {  //acceptable file types  An algorithm to ensure that the inserted data is of the correct data type which can be processed by the rest of the code to reduce errors.  $fileMimes = array(  'text/x-comma-separated-values',  'text/comma-separated-values',  'application/octet-stream',  'application/vnd.ms-excel',  'application/x-csv',  'text/x-csv',  'text/csv',  'application/csv',  'application/excel',  'application/vnd.msexcel',  'text/plain'  );  // Validate whether selected file is a CSV file  if (!empty($\_FILES['file']['name']) && in\_array($\_FILES['file']['type'], $fileMimes))  {  // Open uploaded CSV file with read-only mode  $csvFile = fopen($\_FILES['file']['tmp\_name'], 'r');  //upload data into database  if($run){  //Put first row into the crimes database  $start = 2; //value for the for loop  $data = fgetcsv($csvFile); //read first row of csv in an array  $tablename = "crime";  $columnName = "crime\_types";  echo "table: ".$tablename . "<br>data: <br>"; | for($x=$start;$x<Count($data);$x++){  $value = mysqli\_real\_escape\_string($conn,$data[$x]);  //include the sql from the other file  include 'Chunks/SQLStatements.php';  //check if the value is already in the table before inserting  $result = mysqli\_query($conn,$CHECKIFEXISTS);  if (mysqli\_num\_rows($result)==0) {  //insert into the database  mysqli\_query($conn, $INSERT);  echo " ".$data[$x];  }  }  $tablename = "records";  echo "table: ".$tablename . "<br>data: <br>";  $num = 1;  $start = 2;  while (($data = fgetcsv($csvFile))!== false&&$num <= 5){  $dateString = $data[1]; // The date value from the CSV file  echo $dateString;  $date = DateTime::createFromFormat('MYj', $dateString);  if($date){  $date = $date->format('Y-m-d');  }    for($x=$start;$x<27;$x++){  $number = mysqli\_real\_escape\_string($conn,$data[$x]);  $index = getSeverity($number);    $column = 'crime\_types';  $table = 'crime';  $sql = "SELECT $column FROM $table WHERE crime.CrimeID = ($x-1)";  $result = mysqli\_query($conn,$sql);  $row = mysqli\_fetch\_array($result);  $crimetype = $row[0];  $location = mysqli\_real\_escape\_string($conn,$data[0]);  emailUser($index,$date,$number,$crimetype,$location);  if($number!=0){  $insert = "INSERT INTO $tablename VALUES(null,$location,$crimetype,$number,$date)";  mysqli\_query($conn,$insert);  echo "".$crimetype;}  echo $date;  }  echo '<br><br>';  $num +=1;  }  //Put first column into the location database    $tablename = "location";  $columnName = "Location";  echo "table: ".$tablename . "<br>data: <br>";  $prevnum="";  $data = fgetcsv($csvFile);  while($data = fgetcsv($csvFile,1000,",")){  $value = mysqli\_real\_escape\_string($conn,$data[0]);  include 'Chunks/SQLStatements.php'; |
| if($prevnum!=$data[0]){  $result = mysqli\_query($conn,$CHECKIFEXISTS);  if (mysqli\_num\_rows($result)==0) {  //insert into the database  mysqli\_query($conn, $INSERT);  echo " ".$value;  }}  $prevnum = $data[0];  }  // Close opened CSV file  fclose($csvFile);  //header("Location: index.php");  }  //display csv file's contents  else{  echo "<html><body><center><div style='border:inset;backround-colour:lightgrey;'><table>\n\n";  // Fetching data from csv file row by row  while (($data = fgetcsv($csvFile)) !== false) {  // HTML tag for placing in row format  echo "<tr style = 'border:solid;'>";  foreach ($data as $i) {  echo "<td>" . htmlspecialchars($i) . "</td>";  }  echo "</tr> \n";  }  // Closing the file  fclose($csvFile);  echo "\n</table></center></div></body></html>";  }  }  // display error if the file type is not csv  else  {  echo "Please select valid file";  }  }  ?> Search Data <form action="SearchData.php" method="post" enctype="multipart/form-data">  <div class="form\_dropdown">  <label for="crime">Select crime type:</label>  <select name="crime" id="crime">  <?php  $tablename = "crime";  $columnName = "crime\_types";  include 'Chunks/SQLStatements.php';  $result = mysqli\_query($conn, $SELECTDISTINCT);  if ($result->num\_rows > 0) {  // output data of each row  while($row = $result->fetch\_assoc()) {  echo "<option value='" . $row[$columnName]. "'>" . $row[$columnName]. "</option>";  }  }  ?>  </select>  </div>  //repeated for the rest of the forms, just with separate variable | <?php  $run = isset($\_POST['submit']);  if($run){  //get data from form  $Location = $\_POST["Location"];  $crime = $\_POST["crime"];  $date = $\_POST["Date"];  //get all necessary data from SQL queries  $sql1 = "SELECT \* FROM `crime` WHERE crime.crime\_types = '".$crime."'";  $result = mysqli\_query($conn,$sql1);  $result = mysqli\_fetch\_assoc($result);  $sql2 = "SELECT \* FROM `location` WHERE location.Location = '".$Location."'";  $result2 = mysqli\_query($conn,$sql2);  $result2 = mysqli\_fetch\_assoc($result2);  $sql3 = "SELECT \* FROM `records` WHERE records.LocationID = '".$result2['ID']."' AND records.crimeID = '".$result['CrimeID']."' AND records.date = '".$date."'";  $result3 = mysqli\_query($conn,$sql3);  $severityColors = Array("#ffffff","#cccc00","#e60000");  while ($row = mysqli\_fetch\_assoc($result3)){  $number = $row['number'];  $location = mysqli\_query($conn, 'SELECT location FROM Location WHERE location.ID = '.$row['locationID'].'');  $crime = mysqli\_query($conn, 'SELECT crime\_types FROM crime WHERE crime.CrimeID = '.$row['crimeID'].'');  $row2 = mysqli\_fetch\_array($location);  $location2 = $row2['location'];  $row3 = mysqli\_fetch\_array($crime);  $crime2 = $row3['crime\_types'];  $index = 0;  //algrithm to determine severity  The code changes the background colour of the results using an index to better differentiate the results on the index page and be more visually appealing.  if($number>=5){  $index = 2;}  else if($number>=3){  $index = 1;}  else{  $index = 0;}  $time = $row['date'];  //create the data box  echo "<div class='centered\_box'style = 'float:right;'>";  echo "<div 'padding: 10px;border:4px solid;'>";  echo "<table style = 'background-color: ".$severityColors[$index]."'>";  echo "<tr><th>Amount</th><th>".$number."</th></tr>";  echo "<tr><th>Crime</th><th>".$crime2."</th></tr>";  echo "<tr><th>Time</th><th>".$time."</th></tr>";  echo "<tr><th>Location</th><th>".$location2."</th></tr>";  echo "</table></div>";  echo "</div>";  }  }  ?> |