Be Aware Everywhere

Introduction

This report details Group 3's final project for the Code First Girls Fullstack stream. It will walk through the objectives of the product that was built, why it's needed, the technical approach to creating the platform and the challenges faced.

Aim

To empower users to make more informed lifestyle, safety and travel decisions based on location-based crime statistics.

Objectives

In order to achieve the aim the team will build a platform that:
- Informs the user of the crimes in their specified location
- Informs the user of ways to change their perceptions on personal safety
- Gives options to plan their journey accordingly

This report is broken down into 5 distinct sections:

- Background: This section will outline the specific details of the web applications conception and how it is navigated
- Specification and Design: This section will outline the web application design through a system architecture diagram and front-end design mock-up
- Implementation and Execution: This section will outline how the team planned and carried out the project, including the libraries and tools that were used

- Testing and Evaluation: This section will outline how the testing and evaluation were planned and carried out
- Conclusion: A short summary including stretch goals and key learnings

Background

As a group of four women, discussing their shared experiences and what sort of platform would be useful to them, it was discovered that all members of the team had experienced a sense of unease in unfamiliar places or when travelling alone. The safety of women walking alone is a well-known issue globally, and as such the team wanted to create an application to improve personal safety.

The aim of this project was to build an open-source web application that can be used by the public, particularly women, to assess the safety of an area, using the police API to render street level crimes in a specified area. The application is called Be Aware Everywhere to reflect its purpose. It runs in the form of a search bar interface with a navigation bar which guides the user through a series of choices to interact with the application.

The Home page allows users to access location-based crimes, by typing in their location either in the form of a valid UK postcode or location. The map will then navigate to the specified location and display a point showing the crimes at location. Another map using downloaded data allows for more interaction with all crimes within a specific area, in this case, Gloucester to render on the map. Clicking on a marker reveals a pop up allowing the user to see the type of crime being shown e.g., sexual assault, theft etc. as well as the date and street name where it occurred.

Other features on Be Aware Everywhere include an About page, this gives the user more information on the application and its mission. The user can also access historical statistical data on the location they have chosen by navigating to the Crime Reports section. Here they can view historical trends in a graphical form and have the option to filter crime by category. The Travel Tips page is aimed at encouraging users to make safer choices, offering reminders on how to keep themselves safe as well as offering travel alternatives externally, such as navigating to Traveline so users can plan end-to-end public transport journeys as well as options for accessing and booking private-hire taxi companies. And finally, a Report crime page has a message form which encourages users to share their own experiences with crime in different locations around the UK or contact the page in general.

Specifications and Design

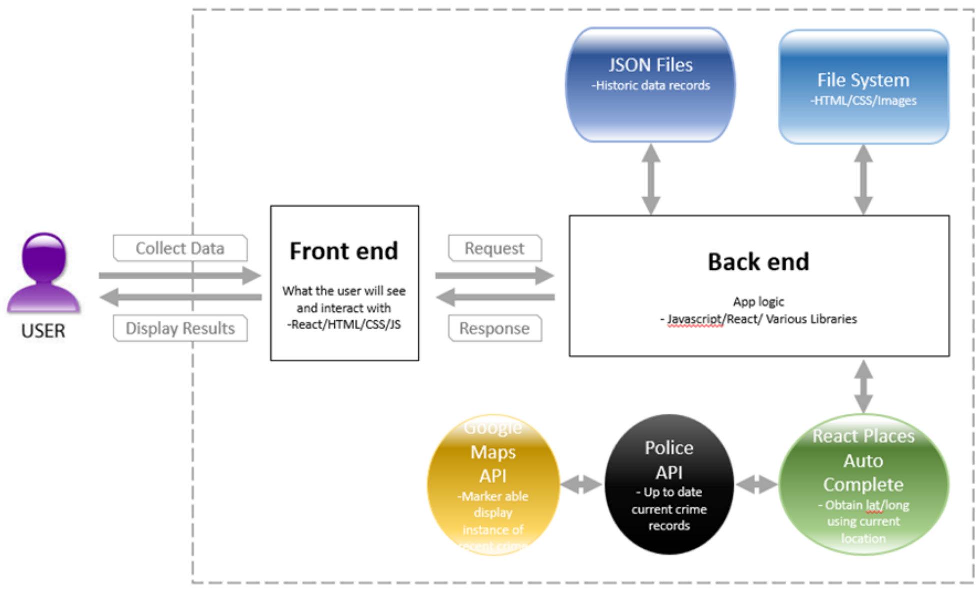
Product specifications

Once it was decided that the project team wanted to build a product to inform users, based on location-based crime statistics, the main system feature requirements for a minimum viable product were then narrowed down:

- **User Interface -** Simple design, which is easy to navigate with consistent formatting and clear branding throughout
- **Responsive to different devices -** Ensure relative sizing and reflowing content when there is a reduction in screen size, ensuring the application can be accessed by mobile phones
- **Information entry point -** Enable location entry specific to user requirements
- **Historical data visualisation -** Options for users to view alternate statistical data to live location to suit various user needs
- **Information page -** Options for users to view advisory information and be redirected to other useful resources
- **Contact page -** Ability for users to send a message or support request by entering their name, email, and message.

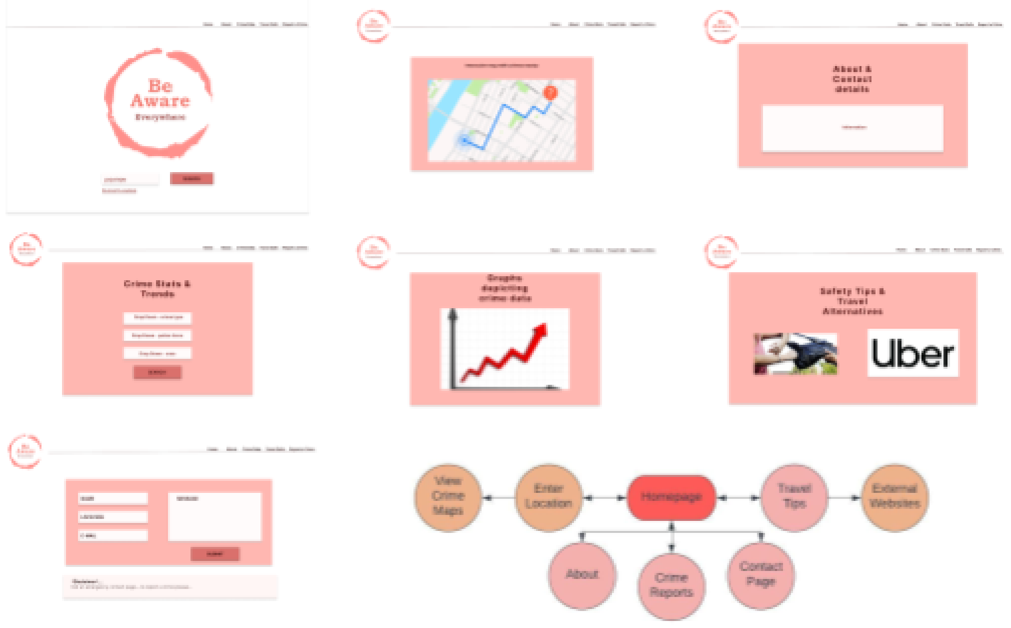
System Architecture Diagram

Below shows the system architecture diagram, showing the data path of the application for the UI interaction


USER
Collect Data
Display Results
Front end
What the user will see and interact with
-React/HTML/CSS/JS
Request
Response
Back end
App logic
- Javascript/React/ Various Libraries
JSON Files
-Historic data records
File System
-HTML/CSS/Images
Google Maps API
-Marker able display instance of recent crime
Police API
- Up to date current crime records
React Places Auto Complete
- Obtain lat/long using current location

Design

Below shows our initial wireframe design of our website front-end, as well as our anticipated user flow



Implementation and execution

Development approach

Firstly, a SWOT analysis was conducted, to determine the team's collective strengths, weaknesses, opportunities, and threats regarding the project. The main constraint identified was the group's asynchronous schedules which led to the utilisation of remote working tools for regular and ad hoc communication, collaborative documents, and sharing code. Based on the SWOT analysis and associated strengths/time constraints it was determined that the team members would assume different roles within the team, these were subject to change and dependent on need. Zainab led the backend construction of application pages, navigation, and primary product use case including mapping and crime data API's; Salha implemented the visualisation of historical data components in both a map and graphical render; Emma focused on content and styling across the website and Tanjin initially focused on maps and rendering the markers and heatmap for the API data points, unfortunately this didn't work out as desired due to time constraints and technical issues, and so focus shifted to testing components.

Due to time constraints, it was difficult to implement agile methodologies, although initially this was the direction the team wished to follow. The project followed a more waterfall approach with deadlines of completion for certain product increments that would not be revisited, although this is regarded as an out-dated method it allowed the project to be completed within the required timeframe. However, where possible agile principles were applied, utilising Scrum frameworks that were tailored to the team's asynchronous/remote working schedules, such as daily stand-ups via Slack messenger and code reviews between pull requests and merges via GitHub. In addition, a recurring video call was allocated which had the function of both a sprint retrospective and sprint planning at each product increment.

Tools and Libraries

- **Trello -** created a Trello board with a backlog of user stories and tickets of tasks to be marked 'doing' or 'completed' so the team could review and monitor workload/progress
- **GitHub –** a shared GitHub repository, so that the team could separately manage the code they were working on from different branches and then merge everything together
- **Slack –** the main communication tool where topics were divided into dedicated channels to organise location of information/updates
- **Google drive –** a shared google drive for project documentation, to allow the whole team to contribute to documentation in real time
- **Zoom -** the team utilised zoom for frequent video calls to discuss goals and review progress where a higher level of communication was required
- **Logo.com -** used to create the branding logo
- **Visual studio code -** used by each team member to individually write and edit code
- **React -** library used to build our user interface
- **React-router-dom -** a React routing library used to route the navigation bar throughout the web application
- **Google Maps API –** external component based on the Google Maps API used to include maps and markers on the website
- **React-places-autocomplete –** a React component to build the user interface for Google Maps and access the locations latitude and longitude
- **UK Police API -** external API used to access street level crime and populate the map
- **Axios** - library used to call from the Police API
- **EmailJS -** library used to enable message to be sent on contact page to default mail client
- **Gmail -** created account to receive messages sent on contact page
- **Leaflet** - an open-source JavaScript library for mobile-friendly interactive maps.
- **React-leaflet** – a library that provides bindings between React and Leaflet
- **SWS** - an instant Static Web Server (for static content)
- **Testing-library/react jest -** a testing library to test React components

Implementation process

User interface (UI) design closely followed the wireframe to establish continuity throughout each page. CSS styling was shared between common elements to ensure consistent design and facilitate user efficiency. When implementing styling, colour, texture, and pseudo-classes were strategically used to create a hierarchy of elements, directing user attention toward or away from features. Components were implemented in keeping with design choices, yet still follow standard practice to avoid causing confusion e.g., hamburger menu symbol for small screens and general component layout. Media queries were implemented to adjust components size and adjust layouts to ensure that the website was responsive at different screen resolutions.

Changes in design to the wireframe were implemented as and when features were determined to have a more efficient layout. For example, initially the map features were intended to be on a separate page, but as the main product feature this was altered to be on the landing page so that it is the first component the user interacts with. The navigation bar was routed for efficiency and to keep code DRY.

Mapping and API implementation proved more difficult than initially expected. A Google API key was set up and a Google map function was implemented with Autocomplete places API, however no data was generated by these API's and the UK Police API was not able to be applied. To apply the UK Police API, a geocodes function, a part of the places-autocomplete, was used to determine the latitude and longitude (latLng). The acquired latLng was then able to be used by the Police API and marker to display the specific location and pinpoint it on the map. An example of the desired map can be found in the document with static data, but the map with live data has one marker and no heat map as of now.

The main goal of the historical data section was to allow the user to access graphical historical data of crime anywhere in the UK by simply selecting their location. The user should be able to view this data in various forms e.g. graphs and bar charts to view statistical data or heatmaps and point maps to show concentration in specific locations in a user-friendly form.

This was to be done using data sourced from the data.police.uk website, where the data is available either through the Police UK API or in csv format. Given the structure of the csv data and the very large number of observations the data needed to be cleaned before it could be properly graphed. Applications such as Excel and R tend to be better suited for handling large datasets, therefore, the decision was taken to download the data in csv format and then clean and ensure that it was in the right format to make graphical representation easier.

The new csv dataset was converted to geoJSON as JavaScript handles this form better and using this data graphs and a complete marker map were coded. The decision to store the data in the application due to time constraints, which led to the dataset needing to be reduced and limited to Gloucestershire, as having a large data file would ultimately impact the loading/running time of the programme and reduce the application's efficiency.

Implementation challenges

The Initial scope of the project was quite ambitious and not fully feasible for the timeframe; however the end product includes the minimum viable product features though, it does not include all the features/functionality initially aimed for. Some specific blockers to the product goals included:

- Crime data API - the map renders a marker for crime data only for the specific latLng point associated with the user input, ideally the API would render markers for an area around the specific latLng but the learning curve to return API data within a polygon would have been too time consuming considering the project deadline
- Geolocation – this was attempted but abandoned as the implementation required a significant amount of research to apply to the existing map code
- SQL database – ideally the historical data would have been stored in a database. However, due to time constraints the decision was made not to connect the application to a database as setting this would have taken significant project time
- Testing - again, there was not a lot of time to be conducting tests at every product increment, instead this had to be done retrospectively
- Heat map – this was unable to render in the webpage despite successfully compiling due to an uncaught error that was too time-consuming to try to rectify

- Asynchronous working - Although identified as a possible constraint with things in place to mediate this, it was inevitable that different schedules/commitments would occasionally cause blockers to project progression
- Learning curve - as the team were all new to JavaScript, a lot of the challenges faced were due to the nature of the ideal components being more advanced than anticipated. As such, a lot of the project time had to be spent researching and overcoming these challenges

Testing and Evaluation

The team approach to testing ensured that both component testing and user acceptance testing (UAT) was undertaken. These were done to check whether the application behaves as expected This was to confirm that individual functions ran as expected and to ensure any errors could easily be debugged and rectified if required. Once the project started UAT began, to ensure the code ran as expected for the end user, from the start to the end of the programme. All team members undertook this and the findings were shared so that errors were identified early and updates undertaken to correct them.

The use of automated testing is efficient in the long run which is why these were used by combining the react test library and jest, to write automated tests in the App.test.js file. It’s important to have a test priority as it helps resolve the most critical defects at an early stage of a testing cycle and it increases understanding of what features bring the most value to the application and are more important to the end-user.

It is also important to test both happy paths and unhappy paths. Happy path testing makes sure the application behaves as a user would expect and unhappy path testing helps us create helpful error messages or workflow reminders as well as let us test that these error messages and suggestion alerts behave as intended. In conclusion, the prioritisation of the project’s component testing strategy was:

- High-value features (happy path testing)
- Edge cases of the high-value features (unhappy path testing)
- Sensitive components (prone to break easily) if there are any

The following table will explain the testing strategy (happy path and unhappy path) of the component tests.

Component to test	**Testing strategy**	
	Happy path testing	**Unhappy path testing**
Google map component with crime information	Snapshot test ○ What happens when the map centre changes Unit tests ○ Google maps testing (test that map renders - google maps api testing) ○ Police crime api testing - test mock api calls	N/A
Search bar component	Simple unit test - Do a userEvent test where user typing text: userEvent.type (“testSearch”) is in the component itself: expect(screen.getByTestId(searchingTestElement)).toHaveValue(“testSearch”);	No location found
NavBar component	Simple unit test - test that NavBar component renders and then conduct unit tests on the different NavBar components by testing that clicking on the different page links on it directs to the correct / corresponding route.	Testing a bad route (non existent page on the NavBar)
Report crime component testing	Simple unit test for the sendMail function using the userEvent *click* on the submit button	Testing an error is logged in the console for invalid email addresses

Testing was similarly more difficult than initially expected. For example, to test the user input in the search bar would normally be a matter of rendering the component including the input and then screen.getByTestId() to declare an input variable and then mock user typing with the userEvent.type() and then just see that the input variable has the value of the dummy message. However, since the search component was part of the google places autocorrect function, this did not work. More researching and studying of both the react testing library and API testing would be needed in the future, to work out how to manipulate the syntax and test these.

One largely acknowledged limitation of this project was time: as a team it was agreed that component and unit testing were intended to be done on the whole project however there were challenges in some parts of the code with pulling out functions to be tested, due to the limited understanding of the syntax in React. With more time to focus on this, the team would have enhanced the testing, testing as component functions are created, and therefore the functionality of the website and map features would be enhanced.

CONCLUSION

The aim of this project was to build an open-source web application that can be used by the public to assess the safety of an area by using the UK Police API to render street level crimes in a specific area. Despite the challenges presented as well as knowledge limitations, this aim was successfully achieved by the team. The project has stayed true to the initial project brief and wireframe design as much as possible, this was a key focus throughout the styling and building stages. Creating a readable and consistent code layout has meant that a cohesive product has been developed, with clear testing strategies applied at various points.

The project offered the opportunity for the team to consolidate existing knowledge and conduct research. Also due to the team's scheduling conflicts and availability, the team have effectively split tasks fairly and in keeping with team members strengths, whilst also offering the opportunity to knowledge share and support learning. The project was broken down into parts to be handled by the team individually but was seamlessly combined due to the continued asynchronous communication throughout the project. The team created an open and supportive atmosphere early in the project and so when issues and limitations occurred they were shared and discussed and decisions made collectively whether to continue with lengthy issues or to park them and move on.

This project has the scope to be further developed and become a valuable asset to many people. Initially the team had identified several user-centric features that they wished to implement, but due to time and knowledge constraints, several of these had to be moved into stretch goals for potential future development

Stretch goals:

- Add a geolocation to the map
- Markers to render API crime information within a location polygon with information pop-ups
- Colour-coded heat mapping to allow crime hotspots to be easily identifiable
- UK-wide statistical crime data availability, broken down by police forces, locations, and crime types
- Incorporating Traveline's public transport API so users can plan journeys directly through the website, making it a one-stop app
- Dark mode for the overall web app

The key learning from this project was that communication and working together were key for the team to meet the aims and objectives of the project, leading to the development of a product that the team is proud of.