THE GOLF BLUEPRINT

Software Development Project
UFCFFF-30-3
University of the West of England
19011230

PROBLEM STATEMENT

What is The Golf Blueprint aiming to solve?

- Course Management is extremely important, but often overlooked by golfers
- 80% of surveyed amateur golfers report having 'Average' or worse course management skills
- Amateur golfers lack detailed resources to help them improve their course management
- The Golf Blueprint aims to solve this problem through visually represented data-driven analytics – specific to The Kendleshire

PROJECT AIMS AND OBJECTIVES

- Create detailed top-down recreations of each hole at The Kendleshire
- Design and Implement a simple system to collect and analyse golf shot data
- Create a secure database to store collected golf shot data and user account information
- Design an intuitive and good-looking website, to be the main point of interaction with the users
- Ensure that the system satisfies the needs of the users, by conducting comprehensive system testing

RESEARCH METHODOLOGY

- Primary Research conducted through a survey – with 86 respondents of Kendleshire members
- Try to gain an understanding of current course management knowledge, and interest levels in The Golf Blueprint
- Gain valuable insight into desired features and design
- Secondary Research exploring existing golf analytic technology – understanding the difference between availability to amateurs compared to professionals

RESEARCH FINDINGS

Primary Research

- **80%** of respondents have 'Average' or worse understanding of course management
- **85%** either 'Very Interested' or 'Interested' in the concept
- **88%** would be willing to contribute data to the resource
- Respondents want the resource to be easy to use

Secondary Research

- Data Analysis is an extremely important feature of the modern game of golf
- Professional golfers utilise data analysis to improve their game, amateur golfers lack such analysis
- Course Management is essential for success, with amateurs neglecting its importance

USER STORIES AND USE CASES

User Stories

- I developed 8 detailed user stories capturing diverse perspectives

- Important to understand the wishes of the users

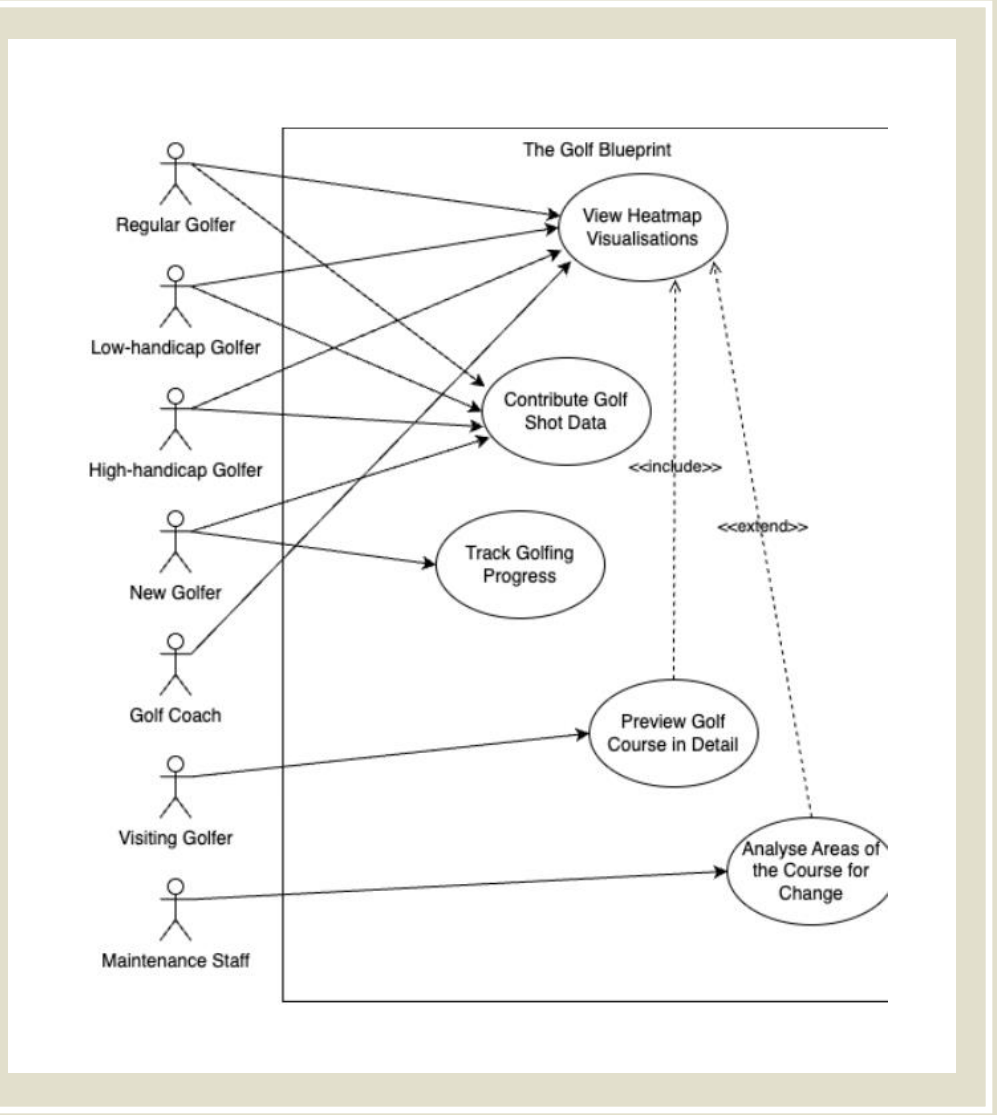
- Utilised when designing the core functionality of the system

Use Cases

- Derived from User Stories

- Showcase how users interact with the system

- Ensure requirements meet the real needs of the users



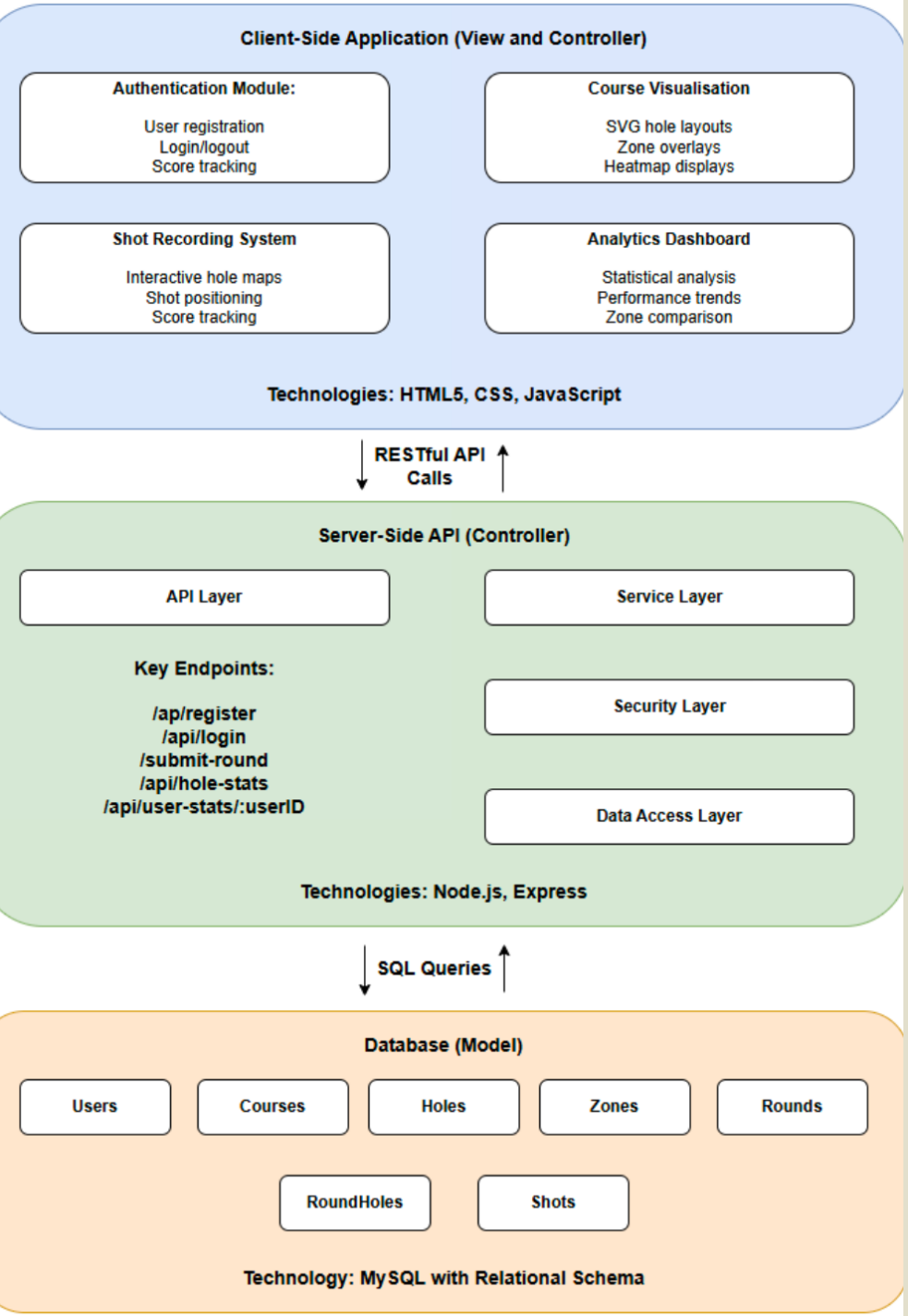
REQUIREMENTS

Data Collection and Input		
ID	**PRIORITY**	**Requirement**
FR7	**Must have**	The system must provide interactive hole visualisations for all 18 holes at The Kendleshire Golf Club
FR8	**Must have**	The system must allow users to mark precise shot locations on each hole visualisation
FR9	**Must have**	The system must record the date for each round.
FR10	**Must have**	The system must allow for the user to enter number of putts and penalty shots.
FR11	**Must have**	The system must store the shot data in a secure database.
FR12	**Must have**	The system must validate input data to prevent unrealistic or incorrect entries
FR13	**Should have**	The system should limit users to entering a maximum of two rounds per calendar day
FR14	**Should have**	The system should allow users to review and edit their shot data before final submission
FR15	**Could have**	The system could support bulk data entry for multiple rounds in a single session

Created using the MoSCoW Method

- User Account Management
- Data Collection and Input
- Visualisation and Analytics
  - Round History
  - Administration
  - Won't Have

Ensuring user's needs are met while maintaining traceability to research findings



SOFTWARE ARCHITECTURE

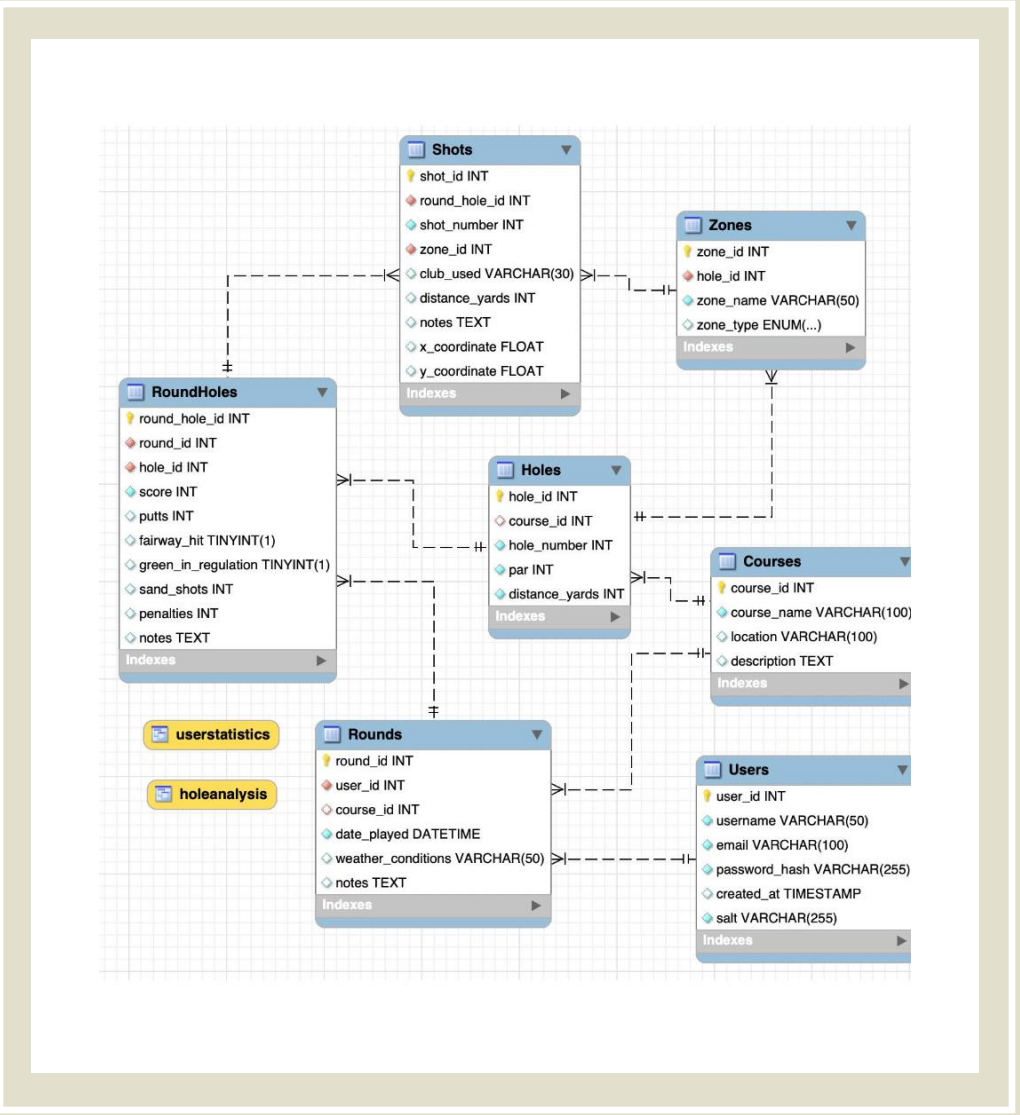
Client-Server Architecture using Model-View-Controller pattern

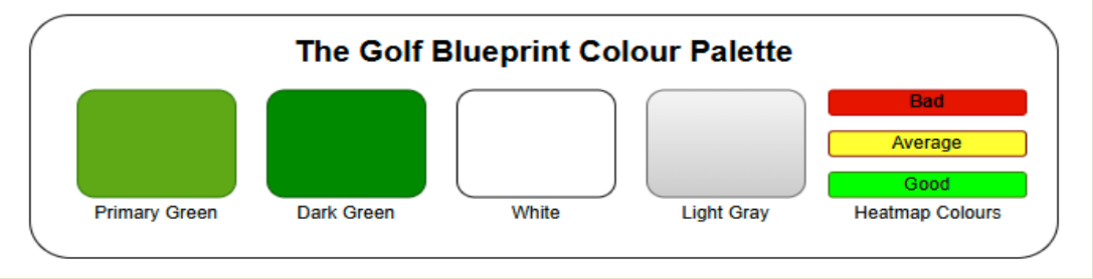
- Client-side application handles user interface and interactions
- Server-side API processes requests
- Persistent data stored through MySQL database

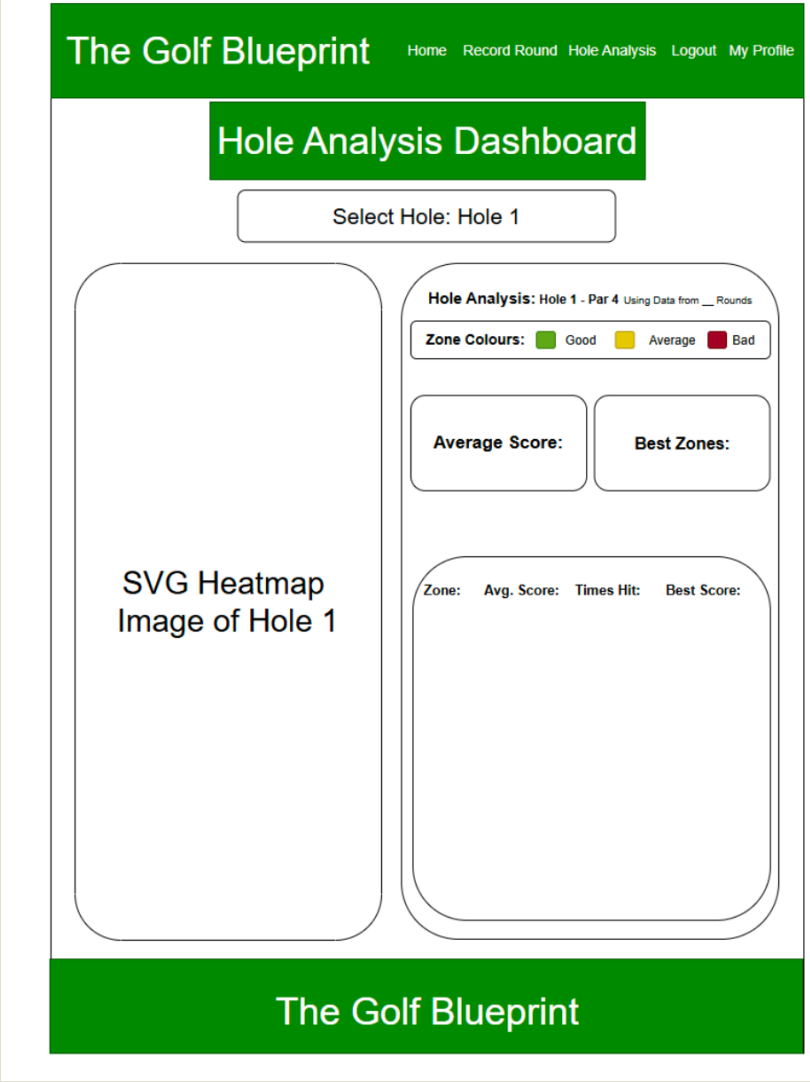
DATABASE DESIGN

Designed to efficiently store and reproduce shot data as useful analytics

- Users Table: Stores account information
- Rounds Table: Each round of golf played by users
- Holes Table: Each of the 18 different holes
- RoundHoles Table: Join table
- Shots Table: Detailed information about individual golf shots
- Zones Table: Defines the areas of each hole
- Courses Table: List of available courses







USER INTERFACE DESIGN

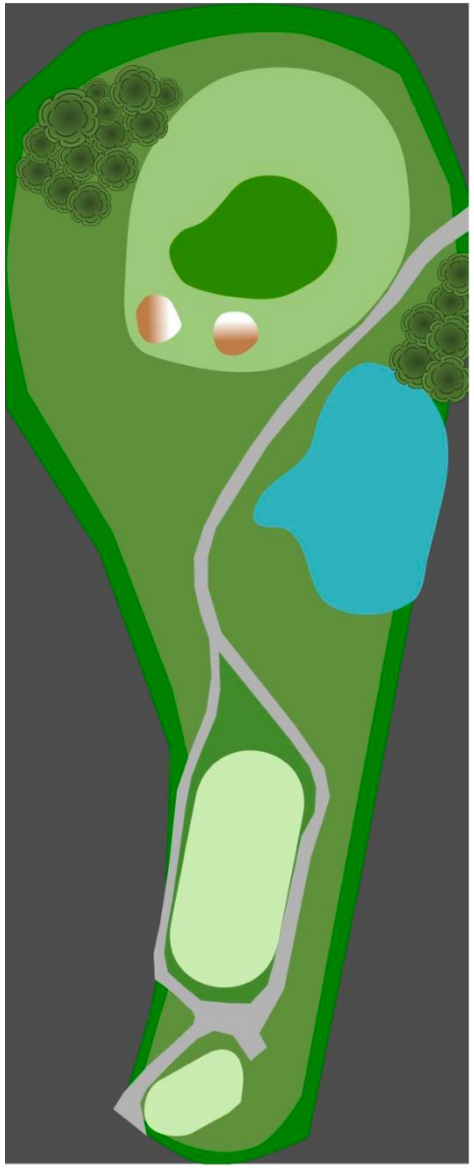
Colour Palette

- Professional aesthetics
- Familiar golf-related colours

Wireframes

- Created for all website pages
- Provided a blueprint for the website before creation
- Allowed focus on usability and looks
- Implementation through CSS followed wireframes closely

Google Earth Satellite Image



SVG Recreation

SVG IMAGE CREATION

- Focus on accurate recreation
- Captured satellite imagery for all holes
- Inkscape
- Traced over each hole using shapes

- Zones drawn on top of each hole
- Holes and Zones hand-drawn for specific detail of each hole

KEY ALGORITHMS

Two core algorithms drive the functionality of The Golf Blueprint

1. Shot Tracking – Captures the user entry of golf shot data – enables for storage into the database

2. Heatmap Visualisation – Fetches golf shot data from the database – transforming it into meaningful course management insights

SHOT TRACKING

```javascript
/**
 * Shot Tracking System for The Golf Blueprint
 * Core implementation of the interactive shot data collection
 */

// Track shot data and user interactions
let shotCount = 0;
let puttCount = 0;
let penaltyCount = 0;
let shots = [];
let actionHistory = [];

/**
 * Records a shot when user clicks on the hole visualisation
 */
function recordShot(event) {
    // Calculate exact coordinates where user clicked
    const container = document.getElementById('fullHoleView');
    const rect = container.getBoundingClientRect();
    const x = event.clientX - rect.left;
    const y = event.clientY - rect.top;

    // Identify which course zone was clicked (fairway, rough, etc.)
    const zone = getZoneAtPosition(event.target);

    // Create shot data object with complete information
    const shot = {
        number: ++shotCount,
        x: x,
        y: y,
        zone: zone,
        xPercent: (x / rect.width) * 100,  // For responsive scaling
        yPercent: (y / rect.height) * 100,
        type: 'shot'
    };

    // Store shot data and update history
    shots.push(shot);
    actionHistory.push({ type: 'shot', shotIndex: shots.length - 1 });

    // Create visual marker on the hole visualisation
    addShotMarker(shot);

    // Update UI and save data
    updateDisplay();
    saveHoleData();
}
```

Capture click coordinates relative to hole container

→ Determine the zone that was clicked

↓

Create a shot object with zone information

→ Add the shot marker to the SVG image

↓

Update the shot counter and score totals

→ Store the shot data in the client's local storage

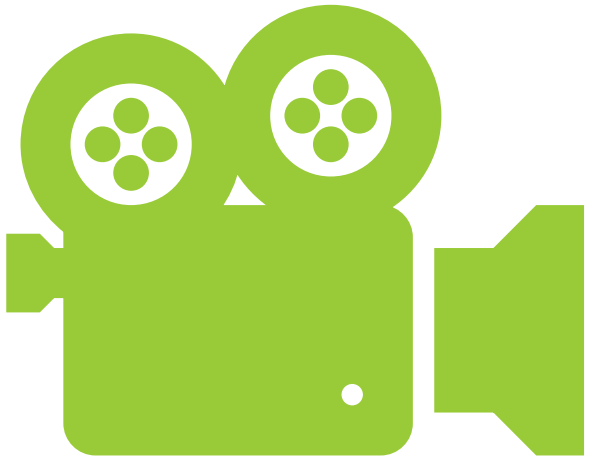
```javascript
/**
 * Creates visual indication of shot location
 */
function addShotMarker(shot) {
    const container = document.getElementById('fullHoleView');

    // Create numbered marker element
    const marker = document.createElement('div');
    marker.className = 'shot-marker';
    marker.textContent = shot.number;

    // Position precisely where user clicked
    marker.style.left = shot.x + 'px';
    marker.style.top = shot.y + 'px';

    container.appendChild(marker);
}

/**
 * Saves shot data to persistent storage
 */
function saveHoleData() {
    const holeNumber = getHoleNumber();

    // Prepare complete hole data
    const holeData = {
        holeNumber: holeNumber,
        shotCount: shotCount,
        puttCount: puttCount,
        penaltyCount: penaltyCount,
        holeScore: shotCount + puttCount + penaltyCount,
        shots: shots,
        actionHistory: actionHistory
    };

    // Save to browser's localStorage
    let roundData = JSON.parse(localStorage.getItem('currentRound') || '{}');
    roundData[`hole${holeNumber}`] = holeData;
    localStorage.setItem('currentRound', JSON.stringify(roundData));
}

// Initialise tracking system when page loads
document.addEventListener('DOMContentLoaded', function() {
    // Load existing data if available
    loadHoleData();

    // Set up click handler for the hole visualisation
    const holeView = document.getElementById('fullHoleView');
    if (holeView) {
        holeView.addEventListener('click', recordShot);
    }

    updateDisplay();
});
```

```javascript
/**
 * Analytics Heatmap System for The Golf Blueprint
 */

// Statistical visualisation algorithm for golf course zones
function updateZonePerformanceVisualization(zoneData, holeNumber) {
    const container = document.getElementById('hole-visualization');

    // Reset previous visualisation state
    container.querySelectorAll('.zone').forEach(zone => {
        zone.style.backgroundColor = '';
        zone.style.opacity = '0';
        zone.style.display = 'none';
    });

    if (!zoneData || zoneData.length === 0) return;

    // Calculate statistical reference points
    const scores = zoneData.map(zone => zone.avgScore);
    const minScore = Math.min(...scores);
    const maxScore = Math.max(...scores);

    // Use median as performance baseline
    scores.sort((a, b) => a - b);
    const medianScore = scores.length % 2 === 0
        ? (scores[scores.length/2 - 1] + scores[scores.length/2]) / 2
        : scores[Math.floor(scores.length/2)];

    // Apply performance-based color coding to each zone
    zoneData.forEach(zone => {
        // Handle different zone naming conventions between front/back nine
        const zoneId = parseInt(holeNumber) < 10
            ? zone.zone.toLowerCase().replace(/ /g, '-')  // Front nine uses hyphens
            : zone.zone.toLowerCase().replace(/ /g, '');  // Back nine has no hyphens

        const zoneElement = document.getElementById(zoneId);

        if (zoneElement) {
            // Color based on performance relative to median (threshold = 15% of range)
            const scoreDiff = zone.avgScore - medianScore;
            const threshold = (maxScore - minScore) * 0.15;

            // Apply color coding based on statistical performance
            if (zone.avgScore < medianScore - threshold) {
                // Better than average - green with opacity based on how much better
                const opacity = 0.5 + Math.min(0.4, Math.abs(scoreDiff / threshold) * 0.4);
                zoneElement.style.backgroundColor = `rgba(59, 243, 46, ${opacity})`;
            } else if (zone.avgScore > medianScore + threshold) {
                // Worse than average - red with opacity based on how much worse
                const opacity = 0.6 + Math.min(0.3, Math.abs(scoreDiff / threshold) * 0.4);
                zoneElement.style.backgroundColor = `rgba(255, 102, 102, ${opacity})`;
            } else {
                // Average performance - orange
                zoneElement.style.backgroundColor = `rgba(255, 152, 0, 0.5)`;
            }

            // Make zone visible and add tooltip with statistics
            zoneElement.style.display = '';
            zoneElement.style.opacity = '1';
            zoneElement.title = `${zone.zone}: Avg. Score ${zone.avgScore.toFixed(1)} (${zone.count} shots)`;

            // Add interactivity for user exploration
            zoneElement.addEventListener('click', () => highlightTableRow(zone.zone));
        }
    });
}
```

HEATMAP VISUALISATION

1. Fetch zone performance data from the API
2. Calculate performance metrics for each zone
3. Apply appropriate colour coding based on performance
4. Adjust colour opacity depending on magnitude of difference
5. Apply styles to the zone elements

```javascript
// API interaction to fetch performance data
function fetchHoleData(holeNumber) {
    // Fetch statistical data for the selected hole
    fetch(`/api/hole-stats?hole=${holeNumber}&shot=all&datePeriod=all-time`)
        .then(response => response.json())
        .then(data => {
            if (data?.zones?.length > 0) {
                // Process data into visualisations
                updateZonePerformanceVisualization(data.zones, holeNumber);
                updateZoneStatsTable(data.zones, holeNumber);
                updateScoreChart(data.zones, holeData[holeNumber].par);
            } else {
                // Handle no data case
                document.getElementById('score-chart').innerHTML =
                    '<div class="loading-placeholder"><p>No data available for this hole</p></div>';
            }
        })
        .catch(error => {
            console.error('Error fetching data:', error);
        });
}

// Initialise dashboard with event listeners
document.addEventListener('DOMContentLoaded', function() {
    document.getElementById('hole-select').addEventListener('change', updateDashboard);
    updateDashboard(); // Load initial hole data
});
```



VIDEO DEMONSTRATION

TESTING AND RESULTS

31 Test Cases developed across 4 components:

1. User Authentication
2. Profile Access
3. Hole/Shot Analysis
4. Round Recording

100% of 'Must Have' requirements have been met

74% of overall requirements have been met

NEXT STEPS

Short-Term Development	Complete ‘Should Have’ Requirements
	Enhance Data Validation
	Conduct Extended User Testing
Long-Term Development Vision	Mobile Application Development
	Integration Capabilities
	Expanded Analytics
	Multi Course Expansion

CONCLUSION

Overall, the project has been a success

- Demonstrates how software can be used to provide insight into improving a golfer's performance
- Successfully followed a software development life cycle
- Utilised an Agile Methodology
- Created Functional and Non-Functional Requirements based on research findings
- Designed and Implemented the Software to meet these core requirements
- Effectively tested the system