Iowa Adventure Cyclist Course Creator

Design document

Team number: SDMAY25_06

Advisor: Julie Rursch

Team Members/ Roles:

Kayley Clark - Technical Lead

Tanner Smith - Algorithm Architect

Nayma Garcia - UI/UX lead

Grant Pierce - Client Relation Manager

Nick Thoms - Testing

Eli Newland - Task Manager

Wan Elisa Wan Sarif - Component Design

Executive Summary

The cycling app project aims to enhance cycling experiences by providing route planning, customization, and tracking features tailors to cyclists' unique needs. With integrations like real-time road conditions, road-blocks warning, and Department of Transportation (DOT) data, the app offers personalized and accessible solutions for recreational and professional cyclists. Through iterative and ethical design principles, the project addresses critical user concerns like data privacy, accessibility, and scalability, ensuring a user-centric and socially responsible product.

Learning Summary

Development Standards & Practices Used

Standard practices

- Agile development methodologies, including iterative sprints and regular retrospectives
- Git-based version control for collaborative coding and change management
- Unit and integration testing for key features to ensure software reliability
- Modular design for scalability and maintainability

Engineering standards considered

- ISO/IEC 25010:2011 (Systems and Software Quality Requirements and Evaluation - SQUARE)
- ISO 9241-210:2010 (Ergonomics of Human-System Interaction – Human Centered Design for Interactive Systems)
- ISO 19116:2004 (Geographic Information - Positioning Services)
- ISO 26000:2010 (Guidance on Social Responsibility)
- ISO/IEC 29119:2013 (Software Testing Standards)

Summary of Requirements

- Efficient route generation
- Filter roads based on user preferences
- Save and share routes
- Export Routes to Garmin devices
- Easy to use UI

Applicable Courses

- SE 309 – Project Management
- SE 319 – Working with React
- ENGL 314 – Technical Report Writing
- SE 339 – Planning out Project Architecture
- SE 317 – Software Testing

New Skills/Knowledge

- Implementing GPS-based route tracking and live updates
- Integrating and working with external APIs like the Department of Transportation (DOT) API.
- Managing ethical challenges in data collection and user privacy
- Designing for scalability and high availability using modular frameworks
- Gaining proficiency in user interface and experience (UI/UX) design principles tailored for mobile applications
- Gaining hands-on experience with Git for managing collaborative coding projects effectively
- Understanding the societal and ethical implications of the app, including fairness in data access and equitable design
- Conducting user interviews and surveys to gather insights for design development

Table of Contents

1 Introduction

1.1 Problem Statement

Cyclists today face a range of challenges that make navigating and planning their rides difficult, whether they are daily commuters or recreational riders. Navigating through heavy traffic, unexpected road conditions, or routes that do not meet a cyclist's specific fitness goal can lead to frustration, wasted time, and even safety concerns. Current navigation and fitness apps often fall short, as they are either designed for drivers or runners, leaving cyclists with limited tools that lack key information like elevation changes, real-time traffic updates, or terrain conditions. To address these issues securely, our app will collect only essential data, such as real-time location and route preferences, while ensuring user consent and end-to-end encryption of personal and location data. This ensures users can safely and confidently plan rides without risking data breaches or misuse.

Cycling is a unique activity that requires tailored solutions. Cyclists need routes that balance safety, efficiency, and performance, with access to data such as road conditions, traffic patterns, and terrain difficulty. These features are essential for cyclists to not only enjoy their rides, but also improve their fitness, reduce their environmental footprint, and engage more deeply with the cycling community. The app will include robust location privacy measures, such as anonymizing sensitive route data and stripping precise start and end points for publicly shared rides. APIs providing real-time data (like traffic and terrain) will be secured with authentication tokens and rate-limiting to prevent unauthorized access or tampering.

Hence, our app seeks to reimagine how cyclists navigate their rides by creating an intuitive cycling app that helps users plan safe, efficient, and personalized routes. The app will display up-to-date Iowa DOT data on traffic, terrain, and road conditions along with elevation data, provide route optimization based on preferences, and allow users to track and save their rides. To further enhance security, users will have the ability to delete their ride history, and the app will implement Multi-Factor Authentication (MFA) and third-party secure login options like Okta or Google OAuth to protect user accounts. This solution will improve the cycling experience for all user types by addressing their core needs and ensuring they can ride safely, efficiently, and enjoyably while maintaining trust in the app's commitment to data protection.

1.2 Intended Users

1.1.1. Casual Cyclists

Casual cyclists are people who ride occasionally, either for leisure or exercise. They are often unfamiliar with the best cycling routes and may feel anxious about encountering busy streets or difficult terrain. To ensure their privacy and security, the app will require explicit user consent before accessing location data and provide users with the option to disable live tracking if they prefer. All collected route and traffic data will be encrypted both during transmission and storage, ensuring it cannot be accessed by unauthorized parties.

Needs:

- An easy-to-use app that provides safe, beginner-friendly routes with clear directions.
- Real-time updates on traffic and the ability to avoid busy areas.
- Ability to filter routes based on terrain, difficulty, road type, etc.

- The app will use secure APIs to deliver traffic data safely, and routes will remain private unless users choose to share them.

Benefits: The app will help casual cyclists feel more confident about navigating new areas by providing them with routes that match their experience level and offering real-time traffic and safety information, all while ensuring their data remains protected.

1.1.2. COMMUTING CYCLISTS

Commuters are people who use cycling as their primary mode of transportation to travel to work, school, or other daily activities. They value efficiency and need to navigate through busy city environments. The app will provide secure route optimization with real-time updates on road closures and traffic conditions using authenticated and rate-limited APIs to ensure accurate, tamper-proof data. To protect commuters' privacy, route data will be anonymized to prevent tracking of individual travel patterns, and users can delete or manage saved routes.

Needs:
- Ability to create optimized routes from point A to point B.
- Ability to create routes that avoid heavy traffic.
- Real-time updates on road closures or detours to minimize delays.
- Estimated time to travel from point A to point B.

Benefits: The app will allow commuting cyclists to save time by suggesting the most efficient paths, helping them avoid traffic jams and arrive at their destination quickly and safely while ensuring their travel data is securely handled and anonymized.

1.1.3. ADVENTURE CYCLISTS

Adventure cyclists are advanced cyclists such as athletes and those who use cycling as a sport or for long-distance travel. They seek challenging terrains and often ride on unfamiliar roads. To address their needs securely, the app will provide detailed terrain maps and elevation data through secure APIs. When sharing or saving routes, users will have access to granular privacy controls to decide who can view their rides. For publicly shared routes, the app will automatically strip personal identifiers like precise start and end locations to ensure privacy.

Needs:
- Detailed route information, including elevation maps and terrain difficulty, to plan their rides.
- Ability to save and share their favorite routes for future training or exploration.
- A wide variety of route options ranging in difficulty.
- Features to motivate and reward exploration.

Benefits: The app will cater to adventure and competitive cyclists by allowing them to explore new routes with confidence, knowing they have all the information they need to take on new challenges. At the same time, robust privacy controls will ensure that their ride data remains secure, and only intended audiences can access shared routes. Additionally, the app achievements encourage exploration and challenges.

2 Requirements, Constraints and Standards

2.1 Requirements & Constraints

The requirements for this project are essential to guide the development process and ensure the app meets the needs of its users. They are divided into functional and non-functional categories, with the latter further split into qualitative and quantitative requirements. These distinctions help ensure both the app's functionality and overall user experience are thoroughly defined, providing a clear framework for development and testing.

2.1.1 Functional

2.1.1.1 Route Generation

The app should allow users to generate routes between chosen locations on the map interface. These routes must adhere to filters provided by the application and user preferences. Users should also be able to add additional stops on routes that are already made.

2.1.1.1.1 Filters

2.1.1.1.1.1 Road Surface Type (paved, gravel, dirt, etc.)

Classifies roads as paved, gravel, dirty, or other types to ensure users can choose a route based on their bike type, riding preference and skill level.

2.1.1.1.1.2 Road Classification (county road, highway, interstate, class-B, etc.)

Roads are given classifications. This is important because the type of road can be a good indicator of how much users are interested in cycling on it. Interstates are not at all bike friendly and some highways have such high-speed limits that they are not a safe choice either. Conversely class-B roads don't see much traffic or maintenance. These roads would be a good choice for users who want a more "off-roading" experience such as adventure cyclists.

2.1.1.1.1.3 Average Annual Daily Traffic

This is a measurement of the average amount of traffic on this road. This is helpful to keep users off of cars with lots of roads that may be unsafe.

2.1.1.1.1.4 Terrain

Describes the general characteristics of the route, such as, uneven surfaces, and ground type. This allows users who prefer specific road types to filter out routes they don't enjoy riding.

2.1.1.1.1.5 Difficulty

Rates the overall difficulty of a route based on combined factors of terrain, road type, and traffic level. This helps cyclists select routes based on their individual experience and skill level.

2.1.1.2 Export GPX Data

Users must be able to export route data in GPX format for syncing with their GARMIN devices, enabling seamless integration with cycling computers.

2.1.1.3 Route Sharing

Routes generated from the application should be shareable with other users within the application, or external platforms like social media or email.

2.1.1.4 Route Saving

Generated routes should have "save" functionality such that the application may be closed, and when reopened a route may be loaded without a new generation.

2.1.1.5 User Data Security

Securely handle and store user data, including authentication, saved routes, and preferences, ensuring compliance with moder security standards (e.g., encryption)

2.1.2 Resource

2.1.2.1 No more than 2GB of space on iOS or Android device.

2.1.2.2 Data is up to date from the Iowa DOT database.

2.1.2.3 Application should have ability, if user agrees, to store data locally on a user's device for offline use.

2.1.3 Physical

2.1.3.1 Application must work with Garmin hardware

2.1.3.2 Application must run on Android and iOS operating system compatible devices.

2.1.4 Aesthetic

2.1.4.1 Interface must be easy to navigate by the user.

2.1.4.2 Color formatting should be uniform and professional throughout the application.

2.1.4.3 Map visuals should be rendered clear enough to see location on map, with different device sizing.

2.1.4.4 Application should use distinguishable and understandable icons for navigation

2.1.5 User Experiential

2.1.5.1 Interface is easy to navigate by the user

2.1.5.2 Easily accessible profile page for the user to update account information.

2.1.5.2.1 Account Information:

2.1.5.2.1.1 First and Last Name

2.1.5.2.1.2 Password Reset

2.1.5.2.1.3 Username

2.1.5.2.1.4 Email

2.1.5.2.1.5 User Preferences:

2.1.5.2.1.5.1 Route Type Default

2.1.5.2.1.5.2 Number of Stops Default

2.1.5.2.1.5.3 Traffic Conditions

2.1.6 Economic and Market

2.1.6.1 Accounts will be free to create and use

2.1.6.2 No features will be restricted behind paywalls

2.1.6.3 All resources and data used by the application will be open source and free to use

2.1.7 Environmental

2.1.7.1 Application should be functional in any weather condition where a phone or Garmin would work

2.1.8 User Interface

2.1.8.1 All buttons will be clearly labeled with either icons or text

2.1.8.2 Each page will have a back button to return to the previous page

2.1.8.3 The software will have an interactive map for easy route creation

2.1.8.4 All features and preferences will be easily updated in user settings

2.2 Engineering Standards

Engineering standards play a critical role in making sure that the app is built to industry-recognized best practices. By adhering to these standards, the app can achieve optimal performance, usability, and reliability, while ensuring compatibility with hardware and external systems. These standards provide a foundation for both technical quality and user satisfaction throughout the development process.

2.2.1 ISO/IEC 25010:2011 (Systems and Software Quality Requirements and Evaluation - SQUARE)

2.2.2 This standard helps to define and evaluate the quality of software by establishing quality characteristics such as functionality, reliability, performance efficiency, usability, security, compatibility, maintainability, and portability.

Relevance: It ensures that the cycling app meets high-quality software standards in key areas like performance (e.g., responsiveness during route calculations), security (protecting user data), usability (smooth user interaction), and reliability (providing accurate routes even in offline mode). The app must be easy to maintain and update with minimal disruptions to users.

2.2.3 ISO 9241-210:2010 (Ergonomics of Human-System Interaction – Human Centered Design for Interactive Systems)

This standard focuses on human-centered design principles, which enhance the usability and user satisfaction of interactive systems. It ensures that systems are designed with a user-first approach.

Relevance: For the cycling app, ISO 9241-210 is essential to ensure the app's user interface is intuitive and user-friendly, allowing cyclists to interact with the app efficiently, even while on the go. This standard will guide design decisions around interface layout, icon placement, and overall navigation flow to reduce cognitive load and improve accessibility.

2.2.4 ISO 19116:2004 (Geographic Information - Positioning Services)

This standard provides best practices for handling geographic data and ensuring the effective use of positioning services. It deals with the accuracy, integrity, and reliability of geospatial data.

Relevance: As the cycling app relies heavily on geolocation and map-based services, this standard ensures that the app effectively processes and displays geospatial data, offering accurate and reliable route planning. Following ISO 19116 will ensure the app accurately positions users on a map and provides clear, real- time route guidance, which is essential for safety and navigation.

3 Project Plan

3.1 Project Management / Tracking Procedures

The team is planning to implement both Waterfall and Agile methodologies for this project. Our project adopts a hybrid project management approach, integrating Waterfall for our initial requirement gathering and prototyping stages and Agile for iterative and development and testing. Due to the early milestone planning of the project, the team has found that they work best using KPI's (Key Performance Indicators) to avoid scope creep. Adding KPIs will measure our productivity, identify any bottlenecks, and ensure progress toward milestones. We'll implement KPIs such as Task Completion Rates to gauge overall productivity to track if the team is on schedule. As well as sprint velocity to track how fast the team accomplishes tasks and can make adjustments to sprint planning if needed.

3.2 Task Decomposition

The team is planning to implement both Waterfall and Agile methodologies for this project. Our project adopts a hybrid project management approach, integrating Waterfall for our initial requirement gathering and prototyping stages and Agile for iterative and development and testing. Due to the early milestone planning of the project, the team has found that they work best using KPI's (Key Performance Indicators) to avoid scope creep. Adding KPIs will measure our productivity, identify any bottlenecks, and ensure progress toward milestones. We'll implement KPIs such as Task Completion Rates to gauge overall productivity to track if the team is on schedule. As well as sprint velocity to track how fast the team accomplishes tasks and can make adjustments to sprint planning if needed.

With addition to the waterfall milestones, the team will be utilizing Agile methodologies on a regular day to day basis. The team will hold weekly meetings to discuss progress and assign tasks from a backlog. The project manager will create the tasks for the backlog at the beginning of each milestone start, and help

assign tasks throughout each week. Iterative system testing throughout each creation before milestone deadlines will additionally help to manage scope creep.

3.2.1 Project Creation

The initial phase of the project focuses on setting the groundwork for a successful project. In this stage, the team clarifies goals, timelines, and aligns on primary deliverables. The setup ensures all team members understand the project's objectives and set into a clear direction. Security practices, such as implementing role-based access controls (RBAC), will be outlined during project creation to establish clear boundaries for data access. This foundational work ensures the system is designed with security in mind from the outset.

3.2.1.1 Determine Requirements

Detailed requirements have been documented based on multiple meetings with the client. These sessions have enabled the team to gather feedback and refine the project scope, minimizing the risk of scope creep. By understanding the client's needs and setting measurable goals, we aim to establish a clear vision of the final project, reducing the likelihood of major design changes as we progress. The requirements also include specific security objectives, such as encrypting all sensitive data—like personalized routes—at rest and in transit, using industry-standard protocols such as AES-256 and TLS. Regular consultations will ensure the project adequately addresses data privacy concerns.

3.2.1.2 UI/UX Setup on Figma

The projects UI/UX has been designed in Figma. This gives us an interactive prototype that effectively captures the project's intended functionality and visual aesthetics. Key design elements, such as colors and layout have been presented to the client. This prototype allows for early feedback and also serves as a blueprint for front-end development phase, which reduces the likelihood of significant revisions.

3.2.1.3 System Setup

Setting up the development environment, which includes servers, file structures for both front-end and back-end, version control, project management, is essential in this initial phase. We've already met with our client to obtain a server for our project, enabling a system setup that facilitates collaboration and code management throughout the project. The team has established Git for version control and Discord for daily communication. This has created an integrated environment for efficient collaboration and tracking. Ubuntu server setup will include configuring a firewall, disabling unused ports, and regularly updating patches. SSH keys will be mandatory for server access, and all Git repository commits are signed for authenticity.

3.2.1.4 Screen Prototyping

Detailed screen prototypes have been developed, laying out each app screen's function and layout in Figma. These prototypes focused on interactive paths, which helped design a cohesive user experience across the application. This phase enables the front-end team to proceed confidently since design choices have already been collaborated and finalized. This leaves only technical implementation adjustments to occur as they are necessary.

3.2.1.5 Database Integration

Database integration is crucial as it connects the application's front-end with secure data storage on the back-end. This stage involves setting up tables, relationships, and secure access methods to ensure consistency and reliability. The database will store essential information, such as user-generated routes, shared paths, and user-access details. To enhance security, database tables will be designed to minimize data redundancy and enforce privacy principles, including pseudonymization of user identifiers. Additionally, no passwords will be stored in the database, as authentication will be managed through Okta integration, significantly reducing the risk of data breaches.

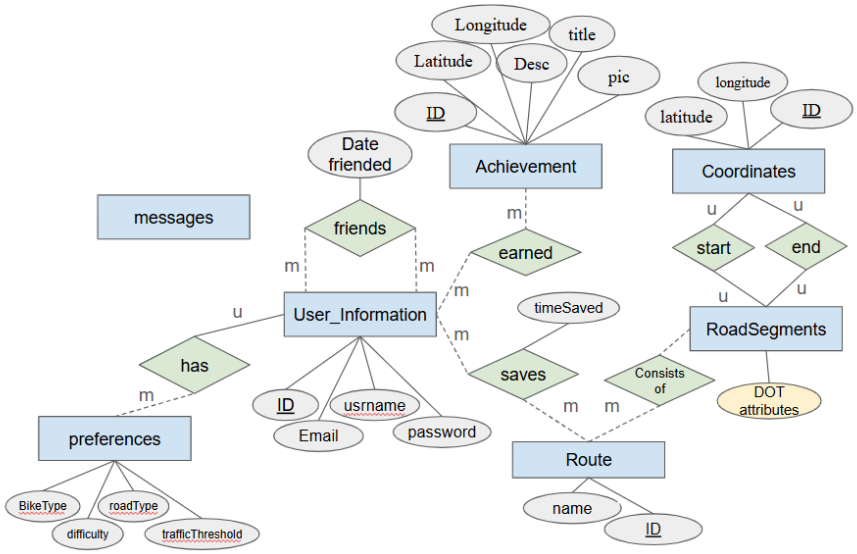


Figure 3.2.1.5: Database ER diagram

3.2.1.6 API Integration

APIs are integrated, this enables interaction between the application and external services such as Iowa DOT. API connections are essential for live route mapping and file exports such as GPX and JSONs to integrate with multiple device types. With API integration complete, the application gains the ability to pull real-time data, enhancing user experience and functionality. APIs integrated into the app will require secure OAuth2 tokens for authentication and rate-limiting to mitigate abuse. API keys are encrypted and stored securely, ensuring that they are not exposed to the codebase or version control. We will use OWASP for vulnerability scanning to test API endpoints regularly to identify any potential exploits.

3.2.2 Feature Implementation

During the feature implementation phase, the team will focus on developing core functionalities that bring the application to life. Each feature was carefully designed and developed to meet the clients' requirements and divided into specific tasks.

3.2.2.1 Okta Login

Integrating Okta to provide secure and efficient user authentication while ensuring enhanced security practices. The process began with setting up an Okta developer account, where the required credentials, such as the Client ID and Domain, were generated. The team then configured an application within the Okta dashboard tailored to the React Native environment. To integrate Okta into the project,

the okta-react-native library was installed and configured with necessary parameters, including the Redirect URI and Scopes, to ensure secure access to user data.

The login flow was designed to redirect users to Okta's hosted login page, where credentials are securely input and validated. By relying on Okta's hosted solution, the application does not store or manage user passwords, further enhancing security and reducing risks associated with credential management. Upon successful authentication, Access Tokens and ID Tokens were returned to manage user sessions. Tokens were securely stored using platform-specific solutions, such as SecureStore for iOS and EncryptedSharedPreferences for Android, with token refresh functionality implemented to maintain uninterrupted user access.

We also implemented error-handling mechanisms to address failed logins, token expirations, or connectivity issues. Rigorous testing was conducted across both iOS and Android devices to ensure the login process was smooth, secure, and reliable.

The successful implementation of Okta login delivers a robust and user-friendly authentication experience while ensuring enhanced security through passwordless storage, while also giving users the option to sign in using other forms of social media. This approach not only protects user credentials but also lays the groundwork for future enhancements, such as Single Sign-On (SSO) and Multi-Factor Authentication (MFA)

3.2.2.2 Route Planning

The route planning feature of the application allows users to generate customizable routes between locations based on user inputs and preferences, allowing for flexibility in planning and navigation. Secure input validation is implemented to ensure that user inputs for route customization cannot compromise the system, for example file uploads for GPX export are restricted to safe file types, with size and content verification to precent malicious payloads. The route planning feature will leverage geographic data to provide users with optimal paths tailored to various parameters, such as distance, terrain, and traffic conditions. To enhance the feature's reliability, data will be sourced from Iowa's Department of Transportation (DOT) database ensuring that the application has access to the most up-to-date information on various road types and conditions. This comprehensive approach will enable users to navigate with confidence, knowing that their routes are informed by the latest available data.

3.2.2.2.1 Algorithm for Pathfinding

The pathfinding algorithm used to create user routes will utilize a customized A* graph search algorithm. Road system data will be used to generate a directed graph that represents the road system. The nodes of the graph represent critical points on the map (road intersections, changes in road type, corners, transitions between road segments in DOT database, etc.) and will store an ID as well as a latitude and longitude value to represent the location of the node on the map. The edges of the graph represent road segments and contain the coordinates of the beginning and end of the road segment a well as the attributes of the road segment (road type, road length, traffic flow, Terrain, etc.). These edges can then be assigned a weight based on the length and the users preferences with the following formula

$$edge.weight = edge.length + \Sigma_{a \in edge.attributes}(attributeCost(a))$$

In the equation, attributeCost(a) is a measure of how much you want to avoid the given road attribute a, as determined by the user's preferences and filter settings, where a higher desire to avoid the attribute results in a larger value. How much an attribute 'a' is dependent on user preferences. If an attribute is to be avoided entirely, then attributeCost(a) = ∞. This equation allows for users to favor one road attribute over another rather than only allowing or disallowing roads with a given attribute, thus achieving more customizable route planning. The pathfinding algorithm will take a starting coordinate and a destination coordinate and perform A* on the weighted graph using the heuristic of straight-line distance to the destination node.

3.2.2.3 Filtering for Map Creation

To enhance user experience, filtering mechanisms will be integrated into the map creation process. These filters will allow users to refine their searches based on criteria's such as terrain types and distance.

3.2.2.4 Export GPX to Garmin Device

Implementation will also include a feature for exporting routes in GPX format, which allows the application to be compatible with Garmin devices. This functionality enables users to transfer their planned routes easily for outdoor and offline navigation.

3.2.2.5 Save Routes

Users will have the capability to save their created and favorite routes within the application. This feature ensures that users can revisit and utilize their preferred routes at any time, this enhances the user experience for usability and convenience. Save routes will also be encrypted in the database, accessible only to authorized users. Access controls prevent one user from viewing another's saved routes without explicit sharing permissions.

3.2.2.6 Share Routes

In addition to saving routes, the application will support route sharing among users. This is done through the friend feature on the application where you can share your routes with your friends. This feature builds collaboration and community engagement to allow users to exchanged their experiences and discoveries. The collective insights gained from shared routes can enhance the overall effectiveness of the application which can benefit all users. Share routes will be secured using one-time access tokens that expire after a set timeframe. Route-sharing will also be logged to track activity to prevent abuse or unauthorized redistribution of user data.

3.2.2.7 Achievements

User will be able to earn achievements in their profile by finding "Hidden Gems". These are locations and routes found across Iowa that are exceptionally unique or interesting. These achievements will be unlocked when a user's position is within a certain distance (varies depending on the size of the Hidden Gem. Larger gems allow you to be further away.) from a Hidden Gem. Users will get a notification when they find a Hidden Gem if notifications are allowed.

3.2.2.7.1 Example Hidden Gems

3.2.2.7.1.1 Iowa State Campanile

3.2.2.7.1.2 Iowa State Capitol Building

3.2.2.7.1.3 801 Grand

3.2.2.7.1.4 High Trestle Trail Bridge

3.2.2.7.1.5 World's Largest Concrete Gnome

3.2.2.7.1.6 Julien Dubuque Monument

3.2.2.7.1.7 American Gothic House

3.2.2.7.1.8 World's Largest Frying Pan

3.2.2.7.1.9 World's Largest Watermelon

3.2.2.7.1.10 The Danish Windmill

3.2.2.7.1.11 The Vermeer Windmill

3.2.2.7.1.12 World's Largest Popcorn Ball

3.2.2.7.1.13 Field of Dreams

3.2.2.7.1.14 World's Largest Wooden Nickel

3.2.2.7.1.15 Hawkeye Point (highest point in Iowa)

3.2.3 System Testing

Thorough system testing will be conducted throughout the implementation phase to identify and resolve any issues promptly. This testing will ensure the features function as intended and meet the quality standards set for the project. Testing will include unit tests, integration tests, and user acceptance testing. Systematically evaluating each component will ensure the application performs reliably under various conditions and meets the user expectation. Feedback gathered throughout the process will allow iterative improvements.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

3.3.1 Determine Requirements

Requirements have been documented in a detailed an organized manner. Well over 3-4 client meetings have been accomplished. Solution has been brought to client's attention with constructive feedback. Requirements should be 95% accomplished, without having any major design changes in application of project.

3.3.2 Project Prototype

The UI/UX is finished on Figma, with a clickable prototype. Colors, design, and screens have been determined as well as presented to client for approval. All major front-end functionality should be determined here, and the only changes in implementation should be if errors occur with map algorithms.

3.3.3 Front End Implementation

Figma designs have 90% been translated onto front end of client application. Screens should be clickable and design choices are fully implemented to 100%.

3.3.4 Back End Implementation

Database and API have been connected to the frontend with 95% accuracy. Algorithms should be developed at this point with 95% accuracy. Saving routes and sharing routes should also be stored and functionality 80% completed.

3.3.5 Additional Features

After finished the main functionality of the app, features like sharing your route with friends, achievements, and exporting routes to different hardware outside of scope will be finished here. Additional features will later be fully mapped out by project manager, based on time management. At least three additional features will be functional on the app here.

3.3.6 User Testing

Assigning different users to test the application. Both android and iOS users should be assigned, all functionality should be at 95%. Following user testing, test cases should be redetermined and evaluated, with adding additional for edge cases.

3.3.7 Final Product

All feature requirements have been met; client has approved the application. Test users have experimented with app, and final iterations and documentation has been completed.

3.4 Project Timeline / Schedule

Requirement	Start Date	Deadline
Determine Requirements		
Schedule regular client meetings	9/1/24	9/1/24
Determine requirements	9/05/24	9/20/24
Solution plan brought to client	11/05/24	11/05/24
Project Prototype		
Design UI/UX models on Figma	10/14/24	10/20/24
Major frontend functionality determined	10/14/24	10/25/24
Color, design, and screens implemented	10/28/24	11/10/24
Prototype progress shown to client	11/15/24	11/15/24
Prototype updated based on client feedback	11/15/24	11/22/24
Prototype testing and validation	11/25/24	12/01/24
Present prototype to client	12/03/24	12/03/24
Frontend Implementation		
Frontend file structure set up on repository	9/10/24	9/20/24
Login page implementation	1/20/25	1/26/25

Home map view implementation	1/26/25	2/9/25
Profile and settings page implementation	2/3/25	2/16/25
Achievements page implementation	2/10/25	2/17/25
Inter page communication implementation	2/17/25	3/10/25
Backend Implementation		
Backend file structure set up on repository	9/10/24	9/20/24
Local database tables set up	11/01/24	11/8/24
Mapping algorithms determined	11/01/24	11/22/24
DOT database and API connection achieved	1/20/25	1/24/25
Crate test data for saving and sharing routes	1/24/25	1/31/25
Routing calculation & saving functionality	2/01/25	2/14/25
Profile preference logic implementation	2/10/25	2/16/25
Backend to frontend communication	2/17/25	2/21/25
Implementing database security checks	2/24/25	3/03/25
Sharing route functionality	3/03/25	3/15/25
Additional Features (optional)	3/01/25	4/01/24
User Testing		
Manual test cases	3/10/25	3/15/25
Automated test cases	3/16/25	3/20/25
Client testing	3/21/25	3/26/25
End-user-testing	3/27/25	3/31/25
Final Product		
Client approval	4/1/25	4/6/25
Finalized documentation	4/7/25	4/11/25

3.5 Risks and Risk Management Mitigation

3.5.1 API Incompatibility or Limitations

Risk: External APIs used may be incompatible or have limitations such as outdated data, insufficient data coverage, or missing data users might need or want (e.g. real-time traffic or detailed terrain data).

Probability: 0.6

Mitigation Plan: Research and prepare alternative APIs to ensure all requirements and user needs are met. Design the application in a way that supports modular API integration, which would allow us to seamlessly switch out APIs if needed.

3.5.2 Unexpected Tool or Technology Limitations

Risk: Certain tools or technologies chosen for implementation (e.g., algorithms for route creation and optimization) may not perform as expected, or require more resources than anticipated.

Probability: 0.5

Mitigation Plan: Conduct tests of all functions of the application in the initial development phases to test performance. Also making sure to conduct these tests on a variety of devices. Evaluate security benchmarks for tools and technologies, ensuring they meet criteria such as secure data handling.

3.5.3 Limited Route Variety

Risk: Due to static data, users may feel restricted if the app lacks variety in the route suggestions, potentially affected app engagement.

Probability: 0.6

Mitigation Plan: Incorporate algorithms that vary route suggestions based on user history or patters, keeping the routes engaging for users. Use anonymized user history to enhance algorithm diversity without compromising individual privacy.

3.6 Personnel Effort Requirements

3.6.1 **Research and Requirements Gathering:** 30 hours

3.6.2 **Design:** 20 hours for wireframes, UI/UX planning, and architecture

3.6.3 **Frontend Development:** 100 hours for UI implementation and functionality integration

3.6.4 **Backend Development:** 80 hours for core features, data setup, and API integration

3.6.5 **Testing:** 50 hours split between unit tests, integration, and user feedback analysis

3.6.6 **Deployment and Evaluation:** 30 hours for setup, final testing, and final presentation

3.7 Other Resource Requirements

3.7.1 Hardware Resources

Testing Devices: Mobile phones (both Android and iOS), tablets, and Garmin or other cycling-related devices to test functionality and compatibility.

3.7.2 Software Resources

For the development of our app, we plan to use React Native Framework in Visual Studio Code to ensure cross-platform compatibility, along with GPX output capabilities for integration with Garmin Edge devices.

The backend will be developed using Node.js in IntelliJ, providing a robust environment for handling API calls, managing requests, and ensuring data consistency.

MySQL will be employed as our primary database, offering efficient data management and storage, especially with our user preferences and route history.

3.7.3 Data Sources and Support

Our project will incorporate third-party data integration from the Iowa Department of Transportation (DOT) to enable real-time routing and road condition updates, enhancing the app's usability and relevance for users.

To ensure the app meets user expectations and performs well in realistic cycling scenarios, we will also gather user feedback from a group of testing participants representing our target demographic. This approach will provide valuable insights into our user needs, functionality requirements, and any potential improvements, ensuring the app is practical to real-world conditions.

4 Design Exploration

4.1 Design Decisions

4.1.1 Export Data onto a GPX Device

Supporting GPX exporting lets users easily export their generated routes to other GPS devices such as the Garmin Edge, commonly used for cycling navigation. This decision is important for our project's success because it makes our app easier for people who rely on external devices to navigate their routes.

4.1.2 Adding Stops to Routes

Allowing users to add multiple stops along a route is crucial for planning trips that include destinations such as gas station stops. Including this is important for the project's success because it makes the app more customizable and suitable for every user type's needs.

4.1.3 Customizable Route Preferences

Giving users the option to customize routes based on their preferences, like road type, traffic type, route length, gives users greater control over their experience and ride. This feature is key to our application because different users have different needs like casual cyclists looking for smooth roads, and adventure cyclists looking for rough terrain. This feature is important to our applications success because it helps make the user experience feel tailored and valuable for every individual user.

4.1.4 User Profiles

Allows users to create and personalize their own profile to save data and preferences. Utilizing AES protocols for data encryption and password protected profiles will ensure user data is private and secure.

4.1.5 Secret Achievements

Secret achievements put a unique and entertaining spin on the average navigational application. These achievements highlight landmarks and unique easter eggs across Iowa, encouraging cyclists to explore locations that they might otherwise miss. These secret achievements utilize GPS location on the user's device and unlock them when they are nearby.

4.2 Ideation

For the design decision to have customizable route preferences feature, we brainstormed a variety of ways to implement it. We focused on giving users as much control as possible within the scope of this class. Here are three options we considered:

4.2.1 Preset Route Profiles

We considered creating predefined profiles, such as "Casual", "Adventure," and "Commuter," which would automatically adjust preferences like road type and traffic level to preset values. This would allow users to quickly select a profile type they align with and have a route configured to their taste without having to configure specific details manually.

4.2.2 Preference Filters

This idea involved using checkboxes, or sliders to adjust specific route details such as traffic and road type. In addition, we thought of adding broader filters such as “Avoid Highways” or “Prefer Smooth Roads” to give users even more control over their route generation.

4.2.3 Route Suggestions Based on User History

We thought about implementing a dynamic recommendation system that would learn from users’ previous routes, suggesting routes that match past preferences. While this would make the experience more personalized, we decided this was out of scope of our project and would become a stretch goal for the future.

In the end, we chose a combination of **Preset Route Profiles** and **Preference Filters** to implement customizable route generation for our users. This allows for both quick selection of profiles, or more fine-tuning, giving our users a wide range of control based on their taste.

4.3 Decision Making Tradeoff

In our process of deciding what features, we wanted to implement for our application, we weigh the pros and cons.

4.3.1 Preset Route Profiles

Pros: Easy for users to choose common cycling styles like “Casual” or “Adventure” without any manual configuration on their end.

Cons: Limited users' ability to fine tune their route to specific tastes.

4.3.2 Preference Filters

Pros: Gives users more control to adjust preferences such as road type and traffic, ensuring routes fit the individual cyclist's taste.
Cons: Might overwhelm casual users who just want a simple setup.

4.3.3 Route Suggestions Based on User History

Pros: Gives the user a more personal experience, aiming to tailor routes to the individual user with no effort on their end
Cons: Might result in less flexibility for users who like to try a variety of routes, as the suggestions would rely on past routes. There is also a privacy concern as this would require us to store user data.

4.3.4 Final Decision

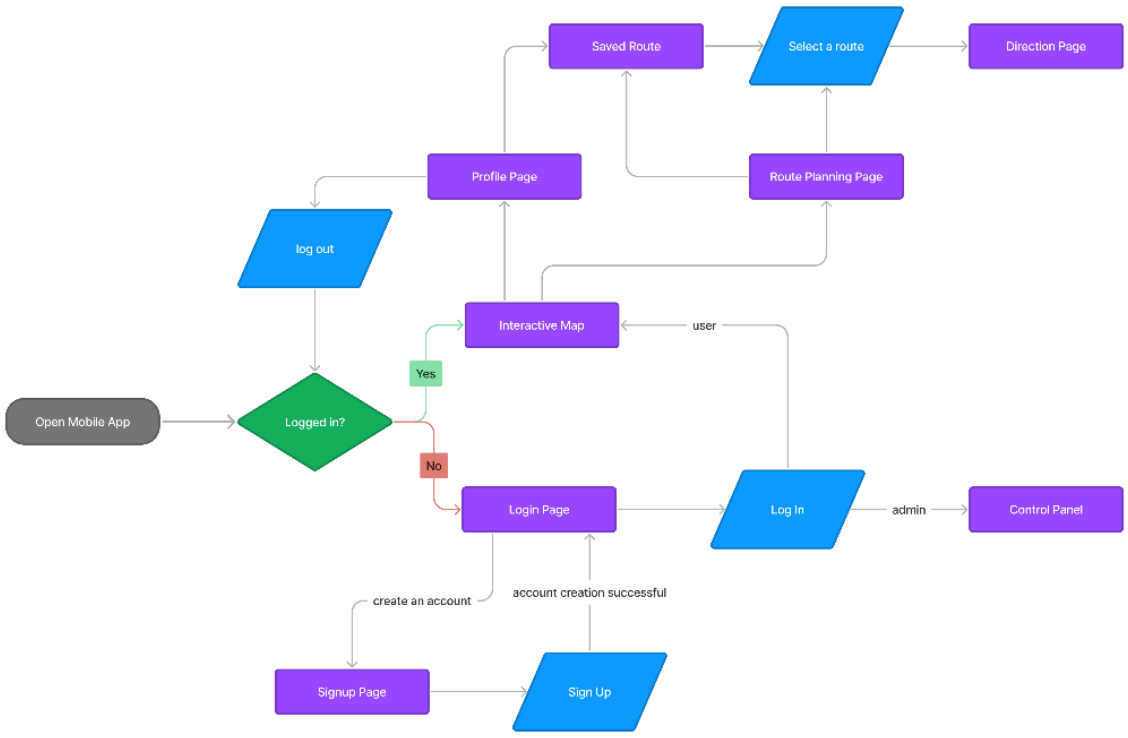
We chose a mix of Preset Route Profiles and Preference Filters. This way, users can either pick a quick profile, or fine-tune details if they want more control. We decided that this combination gives us the right balance of simplicity for users who may feel overwhelmed with too many features, and flexibility for users who do crave highly customizable routes.

4.4 Proposed Design

4.4.1 Overview

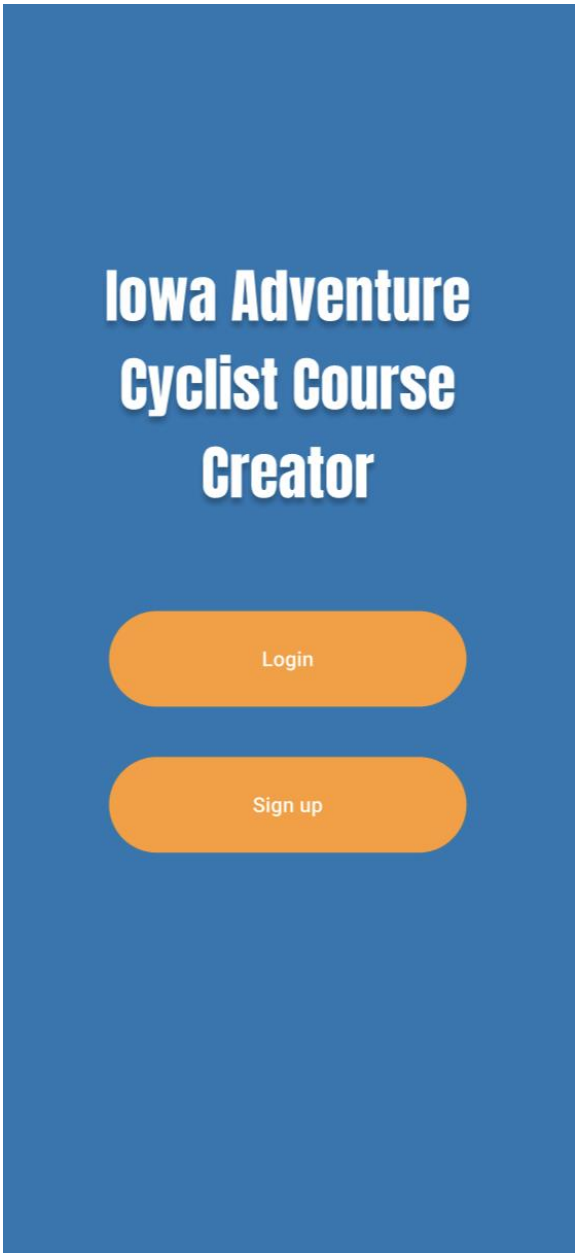
Our current design centers around providing a customizable route-planning app tailored to cyclists' unique needs. The app allows users to specify preferences such as road type, route length, and stops along the way, and then export these routes to GPX- compatible devices. Key components include a robust route algorithm, real-time traffic integration, and data export functionality. Users can seamlessly interact with the app to create, adjust, and follow cycling routes, fostering a user-friendly and versatile experience.

4.4.2 Detailed Design and Visuals(s)



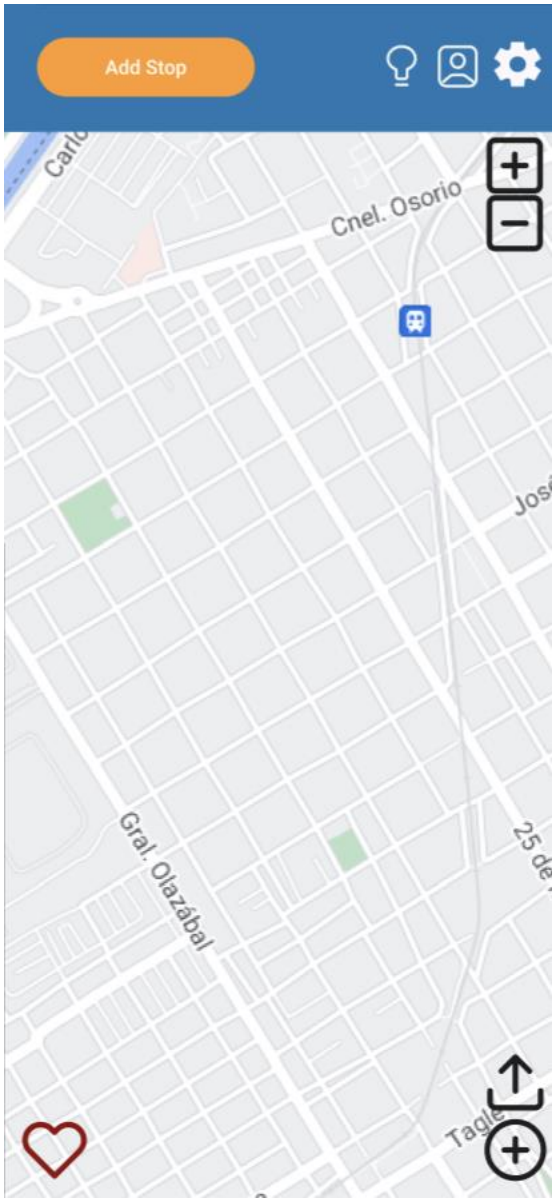
Screen Flow Diagram

This outlines the decision flow for each of our screens



Initial Startup Page Design

This page will greet users when the app is first launched. It prompts them to sign in or create an account



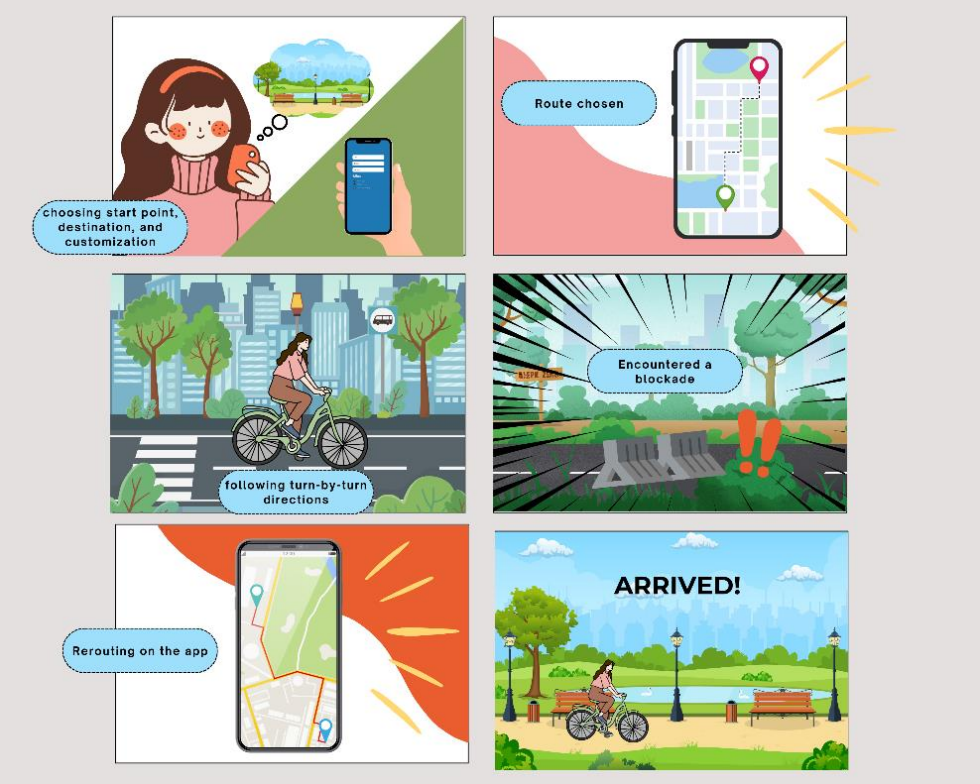
Initial Map Page Design

This is a mockup of what the actual map might look like

Initial Saved Route Page Design

This is what a user might see when looking over their previous routes

4.4.3 Functionality



Imagine a cyclist ready to embark on a new route or adventure. The user opens the app, sets their starting point and destination. Then, they could customize their preferences, like terrain, difficulty, and traffic conditions. Based on these selections, the app processes live GPS and Iowa DOT data to generate optimized routes, factoring in real-time road conditions, traffic, and hazards.

Once the user selects a route, they can stay on the app or export the route data to platforms like Strava or Garmin Connect. The app provides turn-by-turn directions and alerts any sudden changes, such as closures or traffic. The app can also recalculate mid-ride if the user encounters an obstacle or makes an unplanned stop, ensuring they stay on track.

As the ride concludes, users can view and share the route on their profile. They can also view their friends' posted routes and message them about it, contributing to a shared community database. The app combines real-time responsiveness with user customization, creating a reliable tool for cyclists of all experience levels.

4.4.4 Areas of Concern and Development

4.4.4.1 User Privacy and Data Security

Cyclists might be concerned about how their location and route history data are stored and used. The app will ensure transparent communication about data collection practices through a clear, easy-to-understand privacy policy. Users will have the ability to opt-out of tracking, disable location-based services when not needed, and delete their ride history within the app. Additionally, location and route data will be encrypted in transit using SSL/TLS protocols and at rest with strong

encryption algorithms like AES-256. Sensitive information, such as login credentials, will be protected using secure authentication from Okta, ensuring that unauthorized access is prevented. User data will be anonymized wherever possible to reduce exposure risks, and secure role-based permissions will ensure that only authorized users and systems access sensitive endpoints.

4.4.4.2 User Experience

For an inclusive experience, the app must cater to a range of cycling proficiencies, from casual riders to experienced cyclists. Ensuring a user-friendly, intuitive interface is crucial. To maintain a secure user experience, the app will incorporate secure session management, where user sessions automatically time out after periods of inactivity to prevent unauthorized access. Additionally, multi-factor authentication (MFA) will be offered as an option for users who want an extra layer of account protection. User-friendly prompts and alerts will notify users about privacy and security settings during onboarding, ensuring they understand their options and can manage their data confidently. These measures will enhance trust without overwhelming users with advanced features.

4.4.4.3 Cross-Platform Consistency

Maintaining a seamless user experience across Android, iOS, and other platforms requires rigorous testing, especially for real-time data updates and map rendering. To ensure security while achieving consistency, all cross-platform integrations will comply with OWASP Mobile Security Guidelines, which focus on securing APIs, data storage, and user input validation. Secure communication protocols, such as HTTPS, will be enforced to protect data in transit, preventing interception or tampering. Additionally, secure API tokens will be used for all backend interactions, and rate-limiting will be implemented to prevent abuse or denial-of-service attacks. These measures ensure that functionality and security are consistent across all devices.

4.4.4.4 GPS and Data Accuracy

Precise GPS functionality is vital for reliable route tracking and customization. While external factors like signal interference can affect accuracy, the app will mitigate this by verifying and sanitizing GPS data before processing it. Location data transmitted between the app and the server will be encrypted using SSL/TLS protocols to ensure it cannot be intercepted or altered. Real-time DOT data retrieved through APIs will also be validated for integrity, with mechanisms in place to handle stale or incomplete data gracefully. To secure API connections, authentication tokens will be used, and rate-limiting will protect against excessive requests that could disrupt service. By implementing these security measures, the app will ensure that GPS and real-time road data remain accurate, secure, and reliable.

4.5 Technology Considerations

This project will be completed in the form of a mobile device application. The application will be available on both iOS and Android devices to reach a wider range of users. To implement, the team has chosen React Native with TypeScript for the frontend due to its ability to run seamlessly on both platforms and the team's familiarity with JavaScript, ensuring efficient development. Security considerations for the frontend include implementing HTTPS protocols to secure communication with the backend and validating user inputs to prevent vulnerabilities such as cross-site scripting (XSS).

For the backend, the team opted for Node.js due to its flexibility and smooth integration with React Native. To ensure the backend remains secure, all API endpoints will be protected with token-based authentication using tools provided from Okta such as OAuth2.0 and OpenID Connect (OIDC). Additionally, role-based access control (RBAC) will be implemented to ensure only authorized users can access certain resources.

The database will be a static MySQL database hosted on an Ubuntu server, chosen for its reliability and ease of integration with Node.js. To maintain data security, the database will be secured with firewall restrictions, encrypted backups, and limited user permissions. Since user authentication will be handled by Okta, passwords will not be stored in our database, reducing the risk of exposure. The Ubuntu server will also have regular updates applied, and vulnerability scans will be conducted to identify and mitigate potential security risks.

While alternative tools like Android Studio for Android development or Xcode for iOS could have been considered, they would require programming two separate codebases, increasing the complexity of maintaining secure, synchronized updates. For the backend, frameworks like Spring were not chosen due to the higher resource demands and project scope. Similarly, while cloud-based solutions such as AWS offer scalability, they were avoided due to cost constraints. However, the team remains focused on implementing strong security protocols to ensure the SQL database and backend infrastructure remain resilient to potential threats.

4.6 Design Analysis

The team has made progress in setting up the core structure of the application, but much of the foundational work remains, particularly screen development. We chose React Native for cross-platform compatibility, and initial tests show that our choice has enabled smooth integration for both iOS and Android without the need for separate frontend codebases. So far, only a few essential screens have been implemented and basic user flows are under development. On the backend, we managed to establish a functional connection between Node.js and our SQL databased hosted on an Ubuntu server. This backend endpoint being established allowed us to confirm the compatibility of Node.js with our frontend setup and SQL server, which allows us to handle data retrieval and updates. We have also started implementing basic security measures, such as encrypted connections between the frontend and backend, and have begun planning additional cybersecurity features to protect user data and application integrity.

The proposed design from section 4.3 has shown promise. The use of React Native minimized the workload required to support both iOS and Android. Our team's familiarity with JavaScript has allowed us to quickly build prototypes to test different functionalities. Node.js has also been efficient for a backend and MySQL hosted on our Ubuntu server has been reliable for initial testing. These tools align well with our needs and the data demands of our application and confirm our decisions to use a MySQL server for cost and compatibility. We have also started developing a cybersecurity framework for the application, including data encryption, secure API calls, and regular server updates to reduce vulnerabilities. Although these tools show promise in initial testing, we recognize that as the application scales, we need to enhance security protocols further to mitigate risks.

Given the sensitivity of user data, cybersecurity of a key priority. We are working to implement secure connections such as HTTPS and SSL/TLS protocols for our backend API, to encrypt data transmission between the client and server. We also plan on using authentication mechanisms such as Okta to secure user sessions and prevent unauthorized access to sensitive endpoints. This also reduces the amount of data we must store in our database. Additionally, database security if a priority, will restrict direct access to the MySQL server and implement regular vulnerability scans and data backups. We also have considerations to incorporate tools to detect and alert suspicious activity to secure our framework.

Moving forward, our focus will be building the remaining screens and completing the applications' primary workflows while enhancing cybersecurity measures that can be scalable. As we add a more complex backend functionality, we will also implement role-based access control to restrict permissions based on user's roles, adding an additional layer of security. We plan to conduct extensive testing not only for functionality but also for security to identify and fix any potential vulnerabilities. While our initial setup and testing indicated that our design is feasible, continued development will allow us to refine our backend and frontend for usability and efficiency.

5 Ethics and Professional Responsibility

5.1 Areas of Professional Responsibility/Codes of Ethics

We selected the **IEEE Code of Ethics** as our reference framework for professional responsibility. Below is a table summarizing key areas of responsibility, their definitions, relevant items from the IEEE Code of Ethics, and how our team has adhered to these areas during the project.

Area of Responsibility	Definition	Relevant Item from IEEE Code of Ethics	Team Actions
Public Safety and Welfare	Ensuring that the design protects and benefits users and the public.	"Hold paramount the safety, health, and welfare of the public."	Implemented features like real-time traffic updates to promote safer route selection.
Workplace Professionalism	Acting responsibly and ethically within the team and project environment.	"Be honest and realistic in stating claims or estimates."	Regular team retrospectives to address challenges transparently and collaboratively.
Fairness and Equity	Ensuring equal access and treatment for all users.	"Treat fairly all persons and do not engage in acts of discrimination."	Focused on providing rural coverage through community-driven contributions.
Sustainability	Minimizing negative impacts on the environment and promoting sustainable practices.	"Seek, accept, and offer honest criticism of technical work."	Minimized server resource usage by optimizing app performance and algorithms.
Privacy and Confidentiality	Protecting user data and respecting their privacy preferences.	"Disclose promptly factors that might endanger the public or the environment."	Designed robust opt-in data collection policies and enabled data deletion options for users.

Our team has excelled in ensuring privacy and confidentiality by designing a robust data-handling framework. We implemented opt-in policies, provided clear explanations of data use, and offered users the ability to delete their data at any time. These measures promote trust and align with ethical standards

While we have taken steps to minimize server resource usage, we could improve by incorporating renewable energy-powered hosting solutions or exploring carbon-offset programs. These measures would reduce the environmental impact of our application.

5.2 Four Principles

The following table connects the broader context areas with the four ethical principles (beneficence, nonmaleficence, respect for autonomy, and justice):

Broader Context Area	Beneficence	Nonmaleficence	Respect for Autonomy	Justice
User Safety	Promotes safer routes for users.	Avoids harmful recommendations.	Allows users to choose alternative paths freely.	Ensures features are accessible to all users.
Environmental Impact	Encourages eco-friendly cycling routes.	Limits server power consumption.	Informs users about sustainable practices.	Expands features to underserved regions.
Data Privacy	Protects user data from misuse.	Avoids collecting unnecessary data.	Provides clear data control options.	Ensures equal treatment for all user data.
Community Engagement	Builds features with user input.	Avoids exclusion of any community voices.	Empowers users to co-create solutions.	Shares resources equitably among communities.

Our app's focus on safer route recommendations contributes to user welfare. We ensure this by continuously testing and updating our route algorithms based on real-world data. On the other hand, our current reliance on non-renewable energy for app hosting negatively impacts the environment. In future iterations, we will explore renewable hosting options and educate users about eco-friendly practices.

5.3 Virtues

Our team has identified three core virtues that guide our design process and collaboration: **Empathy**, **Collaboration**, and **Integrity**.

Empathy

Empathy plays a pivotal role in ensuring that our app addresses the diverse needs of its users. By actively engaging with potential users through surveys, interviews, and usability testing, we strive to understand their challenges and preferences. This virtue is reflected in features like customizable routes and hazard alerts, which are designed to cater to cyclists of varying skill levels and goals. Empathy also extends to accommodating underserved communities by prioritizing rural route data and inclusivity in design. By putting ourselves in the users' shoes, we ensure our app truly enhances their cycling experience.

Collaboration

Collaboration is the backbone of our team's success. Each member brings unique skills and perspectives, and we foster an environment where these differences are leveraged to achieve shared goals. Regular

meetings, clear communication, and the use of collaborative tools like Git and Trello ensure that everyone stays aligned and productive. Collaboration is also evident in our engagement with external stakeholders, such as incorporating feedback from cycling communities and working with data providers like the Department of Transportation. This virtue ensures that the project remains a cohesive effort, combining technical expertise and user-focused innovation.

Integrity

Integrity underpins all our decisions and actions throughout the project. This means being honest about the app's limitations, such as potential GPS inaccuracies or data privacy concerns, and taking proactive steps to address them. For example, we have implemented clear opt-in policies and robust security measures to safeguard user data. Integrity also involves delivering on our promises, ensuring that the app meets its stated objectives and provides reliable, high-quality functionality. By adhering to ethical standards and maintaining transparency, we build trust with both users and stakeholders.

6 Closing Material

6.1 Conclusion

Our project aims to enhance the cycling experience by providing a user-centric application that emphasizes route safety, customizability, and accessibility. To date, we have implemented several key features, including route planning, traffic updates, and data export, while adhering to ethical and professional responsibilities.

We have made notable progress toward our goals, especially in data privacy, UI/UX design, and algorithm development. Moving forward, our best plan of action is to:

1. Optimize algorithms for improved efficiency.
2. Expand feature coverage in underserved regions.
3. Incorporate innovative features that enhance user engagement.

In future iterations, we will focus on addressing gaps in sustainability and exploring innovative ways to further empower our users. By continually aligning with ethical principles and professional standards, our project has the potential to significantly benefit the cycling community.

7 Team

7.1 Team Members

1. Kayley Clark
2. Tanner Smith
3. Nayma Garcia
4. Grant Pierce
5. Nick Thoms
6. Eli Newland
7. Wan Elisa Wan Sarif

7.2 Required Skill Sets on project

The following skill sets are required to meet the project's goals and requirements:

- Software Development: Knowledge of React Native for mobile app development.
- Algorithm Design: Understanding pathfinding algorithms such as Dijkstra's for route planning.
- Database Management: Proficiency in MySQL for managing route and user data.
- API Integration: Experience integrating third-party APIs, including the Iowa DOT API for traffic and road conditions.
- UI/UX Design: Skills in creating user-friendly interfaces and enhancing user experience.
- Testing and Quality Assurance: Ability to design and execute robust test plans.
- Project Management: Capability to organize tasks, manage deadlines, and ensure team coordination.
- Cybersecurity: Knowledge of secure login processes and data privacy policies.

7.3 Skill set covered by team

Skill Set	Team Member(s)
Software Development	Kayley Clark, Eli Newland, Wan Elisa Wan Sarif
Algorithm Design	Tanner Smith
Database Management	Kayley Clark, Eli Newland
API Integration	Kayley Clark, Eli Newland, Tanner Smith
UI/UX Design	Nayma Garcia, Wan Elisa Wan Sarif
Testing and Quality Assurance	Nick Thoms, Nayma Garcia
Project Management	Eli Newland
Cybersecurity	Grant Pierce

Team Expertise Highlights

- **Kayley Clark**: Brings extensive experience in AWS, MySQL, database management through enterprise systems, React, and C#. Proficient in RESTful APIs using Java and Spring, with leadership skills honed through Agile/Scrum frameworks.
- **Tanner Smith**: Leverages a strong mathematical foundation and practical algorithm development skills from industry internships and outdoor problem-solving experiences.
- **Nayma Garcia**: Offers diverse coding experience and proficiency in various frameworks, excelling in UI/UX design with a focus on efficient and effective teamwork.
- **Grant Pierce**: Certified in A+, Network+, and Sec+, with specialized expertise in GPS systems, cloud security, and application-peripheral integration.
- **Nick Thoms**: Strong coding foundation from ISU coursework, eager to learn and excel in testing and quality assurance.
- **Eli Newland**: Skilled in cloud security, React Native, and fostering team discussions to align project goals with deadlines.
- **Wan Elisa Wan Sarif**: Experienced in React and Java applications, UI/UX design, and database management, including MySQL and MongoDB.

7.4 Project Management Style Adopted by Team

Our team has adopted an **Agile** project management style. This approach allows us to remain flexible, iterate on deliverables, and incorporate feedback throughout the development process. Agile suits the dynamic nature of our project, enabling us to prioritize tasks and adjust to changing requirements effectively. We conduct regular sprint planning meetings and retrospectives to track progress and ensure continuous improvement.

7.5 Initial Project Management Roles

Role	Team Member	Responsibilities
Technical Lead	Kayley Clark	Oversees technical decisions and resolves software-related issues.
Algorithm Architect	Tanner Smith	Designs and implements pathfinding algorithms for optimal route planning.
UI/UX Lead	Nayma Garcia	Leads interface design and user experience enhancements.
Client Relation Manager	Grant Pierce	Manages client communication and ensures project alignment with client needs.
Testing Lead	Nick Thoms	Develops and executes testing strategies to ensure software quality.
Task Manager	Eli Newland	Organizes tasks, monitors deadlines, and ensures balanced workload distribution.
Component Designer	Wan Elisa Wan Sarif	Leads the design and implementation of specific app components.

7.6 Team Contract

Team Members:

8. Kayley Clark
9. Tanner Smith
10. Nayma Garcia
11. Grant Pierce
12. Nick Thoms
13. Eli Newland
14. Wan Elisa Wan Sarif

Team Procedures

Day, time, and location (face-to-face or virtual) for regular team meetings:

Tuesday evenings 7pm over discord, or in person 1213 Coover.

Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):

Discord, Email, Notion

Decision-making policy (e.g., consensus, majority vote):

Majority Vote

Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

Per meeting, volunteer, record kept in Notion Page.

Participation Expectations

Expected individual attendance, punctuality, and participation at all team meetings:

Arrive to meetings on time and prepared to discuss progress, issues and next steps

Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

Equal responsibility shared among teammates.

Expected level of communication with other team members:

Active in discord and responding to teammates within a reasonable amount of time. ****

Expected level of commitment to team decisions and tasks:

Weekly commitment to tasks enough to put time down for weekly reports.

Leadership

1. Kayley Clark : Technical Lead - Responsible for answering any technical questions related to software languages or IDE's.
2. Tanner Smith : Algorithm Architect - Responsible for creating efficient and accurate algorithms to solve encountered problems.
3. Nayma Garcia : UI/UX lead - Responsible for overseeing the design of the applications interface and user experience.
4. Grant Pierce : Client Relation Manager - Responsible for building and maintaining relationship with clients and ensuring needs are met
5. Nick Thoms : Testing - Responsible to lead all test coverage, as well as determine critical test classes and criteria.
6. Eli Newland : Task Manager - Break down work and ensure load is shared throughout the team
7. Wan Elisa Wan Sarif : Component Design - Responsible to lead ownership from features to the application.

Strategies for supporting and guiding the work of all team members:

Promoting communication and setting deadlines.

Strategies for recognizing the contributions of all team members:

Provide specific, personalized feedback for contributions.

Collaboration and Inclusion

Describe the skills, expertise, and unique perspectives each team member brings to the team.

1. Kayley Clark : AWS, MySql, database management through enterprise systems. React and C# application experience, RESTful API's using Java and Spring, Agile/Scrum framework. Leadership skills through various organizations.
2. Tanner Smith : Math Minor, Tutor, fair amount of experience doing outdoor activities (Eagle scout, geocacher, etc.), Joh Deere SE internship for 2 summers.
3. Nayma Garcia : Experience with various languages and frameworks. Willing to work efficiently and effectively with teammates.
4. Grant Pierce : A+, Network+, Sec+ certified. Experience with GPS systems, specifically where routes have limited data. Experience with cloud security and configuring applications with peripherals.
5. Nick Thoms : Typical coding experience through ISU coursework, willingness to communicate and learn with teammates
6. Eli Newland : Experience with cloud / cloud security, experience with various languages and frameworks i.e. React Native, willingness to have discussions with team to push towards a desired end.
7. Wan Elisa Wan Sarif : React and Java application experience. Usually worked on frontend. Interest in UI/UX design. Basic database knowledge, including MySql and MongoDB.

Strategies for encouraging and supporting contributions and ideas from all team members:

Set clear expectation for participation. Encourage everyone to contribute by setting participation goals, ensuring that everyone understands their role and responsibility to contribute.

Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)

The team can utilize the beginning of each meeting as a standup, where they may present roadblocks on the project or with the team members.

Goal-Setting, Planning, and Execution

Team goals for this semester:

Design and build a minimum viable product, lay groundwork for successful second semester

Strategies for planning and assigning individual and team work:

Divide the work into manageable chunks so that they can be easily delegated

Strategies for keeping on task:

Set goals for meetings to ensure a consistent forward progress

Consequences for Not Adhering to Team Contract

How will you handle infractions of any of the obligations of this team contract?

First infraction, meeting with task manager to discuss

What will your team do if the infractions continue?

Second infraction, discussion with team

Third infraction, escalation to discussion with Julie / 4910 professors

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*

b) *I understand that I am obligated to abide by these terms and conditions.*

c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1. **Kayley Clark**_____*DATE 09/17/2024*
2. **Nayma Garcia**_____**DATE 09/17/2024_**
3. **Eli Newland**_____**DATE 09/17/2024_**
4. **Grant Pierce**_____*DATE 09/17/2024*
5. **Tanner Smith**_____**DATE 09/17/2024_**
6. **Nick Thoms**_____**DATE 09/17/2024_**
7. ***Wan Elisa Wan Sarif***_____***DATE__09/17/2024***