Rhodes Adventures Design Report

Supervisor: Kuan-Hsun Chen, Department of Technical Computer Science

Michail Flevaris, Nikolaos Antoniou, Dimitrios Kalopisis, Wahab Ahmed, and Luka Leer

Technical Computer Science, University of Twente

November 2025

Abstract

Rhodes Adventures is a tourist company that organizes outdoor tours of the Greek island of Rhodes for tourists to discover the hidden parts of the island that are not usually found by tourists. The company currently arranges the bookings manually, meaning it might take a lot of time to organize each event. The goal of this project is to change that with a booking system that can automatically organize the events for the company, which includes, for example, the allocation of people into different cars for the events. There will be various algorithms that will help make the company employees' lives easier.

The website is separated into two sections: the client booking page and the employee dashboard, where the employees can manage future events. Since the database on the website contains various personal information about both clients and employees, it is ensured that the website is safe from possible security breaches that could steal this data. The website's User Interface, after being tested by potential users, has been proven to be easy to navigate so that unexpected behavior is prevented and incorrect input is rejected by the system. The design choices and the software implementation of this system are clearly explained throughout the report.

Table of Contents

Contents

Chapter 1

Introduction

Rhodes Adventures is a tourist company located on the Greek island of Rhodes that specializes in off-road excursions and tailor-made activities, which help tourists discover the biggest gems of the island's countryside.

The aim of the project is to develop a booking system for this company, which already has a website, but the organization for the events, except for the booking process, which can be done through the website, is currently done manually. The organization of events could include, for example, the allocation of people into different cars for the safari activity or picking up the customers from their hotels and driving them to the event's starting point. These are all important processes for a tourist-focused company that need to be done automatically, and this is what is done in this project.

In the new website, there are two different sections: the client booking page and the employee dashboard, where the company employees can log in and manage future events. For the organization process to be done automatically, there are algorithms in place that can arrange the client pickups and the allocation of people in each booking to different cars if they are needed in that specific activity, since there could be activities that don't involve cars.

The report will have the following structure. In the chapter ”Requirements Specification,” all the important characteristics of the website are specified. In the ”Planning and Approach” chapter, there will be an outline of the project management throughout the course of the project. In the ”Design” chapter, the decisions made on technologies, the UI design, and the backend will be explained, with the next three chapters, namely ”Diagrams,” ”Algorithms,” and ”API & Frameworks,” explaining in detail the decisions behind their design and their application to this project. In the chapter ”Testing”, there will be a detailed test strategy along with the results that came as a result of it. And finally, in the ”Evaluation” chapter, there is a conclusion and a reflection on the entire project procedure, along with ideas about the future.

Chapter 2

Requirements Specification

To achieve this goal, a bunch of requirements are made. These are based on what was asked to be created by the client.

2.1 Non-Functional Requirements

- As an employee, I would like to specify my availability.
- As an employee, I would like to view, edit, and add available vehicles.
- As an employee, I would like to see which vehicles are available for that activity.
- As an employee, I would like to report unexpected changes to my availability.
- As an employee, I would like to submit the hours I have worked.
- As a user, I would like to pay for my booking easily.
- As a user, I would like to book empty seats on vehicles.
- As a user, I would like to make a booking.
- As a user, I would like to book for multiple people.

2.2 Functional Requirements

2.2.1 Must have

- The app must have a publicly accessible page for potential customers.
- The app must allow potential customers to select from a list of offered activities.

- The app must allow potential customers to be informed of when the selected activities are offered.
- The app must allow potential customers to book any available activity.
- The app must allow potential customers to pay for their booking through Stripe.
- The app must keep in touch with the customer after booking.
- The app must send a confirmation email after payment to the customer.
- The app must allow employees to view a schedule of all activities.
- The app must allow employees to view information about bookings for a specific activity.
- The app must allow employees to create new available time slots for any available activity type.
- The app must allow employees to specify details upon creation, such as a limit to the number of customers that can partake in an activity.
- The app must track damages to vehicles and mechanic's status.
- The app must have a GPS algorithm to assign different drivers to customers.
- The GPS algorithm must give an optimal route from the driver to the customers to the destination.
- The app must track the hours worked for employees.
- The app must inform the employees before the start of an activity.

2.2.2 Should have

- The app should allow customers to select additional options depending on the activity type, such as booking a private vehicle on a tour.
- The app should allow customers to specify details about their booking, such as the number and age of their group.
- The app should remind a customer of an upcoming booking.

2.2.3 Could have

- The app could allow employees to send a message to all customers of an activity by Email.
- The app could send a SMS confirmation of a booking.

Chapter 3

Planning and Approach

3.1 Project Management Approach

The development of the Project was divided into multiple stages to focus on building each part of it individually before integrating it into a final product and then refining it. Rather than use the agile methodology of weekly reviews, a biweekly meetings schedule was used with the supervisor to monitor progress, to redistribute work if needed and to move to the next stage of the project. As well as the supervisor meeting, the peer review feedback meetings were used as a checkpoint to have significant progress done each meeting.
A Git-based workflow was adopted to manage collaboration between team members, and to keep track of work done which could easily be viewed by other members of the team.
The main branch for the git was used to hold the current version of the product, with the integrated aspects while each individual branch would hold the team member's individual units, which need to be integrated into the product.

3.2 Milestones

In order to build the core of the project, a set of milestones were adopted to ensure that each week a significant amount of progress occurred, and the following timeline summarizes the development phase that each week was expected to reach:

- Week 1: Complete the UML diagrams, set up email functionality.
- Week 2: Implement user booking functionality and admin booking management.
- Week 3: Add employee hour tracking, management and vehicle management page.
- Week 4: Implement Stripe payment system for bookings.

- Week 5-6: Finalize a MVP of the website.
- Week 6: Start creating the report and manual.
- Week 7: Integrate the algorithms for the website.
- Week 7-8: Conduct functional and penetration testing: patch security issues, as well as conduct tests to evaluate the project.
- Week 8: Complete user manual
- Week 9: Finalize Design report and project.
- Week 9-10: Prepare final presentation and poster. The project has been on schedule with these milestones, without any deviation.

3.3 Risk Management

In order to deal with possible challenges and to mitigate the risk of impacting the project development, the following risk analysis was used for the project:

Risk	**Impact**	**Likelihood**	**Mitigation Stratetgy**
A team member is unavailable	Medium	Low	Redistribution of tasks to cover the gap created.
Software bug delays development	High	Medium	Cooperate to solve these bugs and allocate extra testing time.
A team member falls behind in their task	Medium	Medium	Cooperate to solve the problem and redistribute tasks to cover lost time.
Supervisor is unavailable	Medium	Medium	Check the supervisor's schedule and arrange extra meetings to make up for missed time.

Table 3.1: Risk analysis for Rhodes Adventure Project

3.4 Planned Deliverables

The planned deliverables of the project include:

- Design Report: A report including
  - The requirement specification, class/activity diagrams showcasing the functionality.

    - of the platform and design of the database, and the justification of the design choices.
    - A test plan and test results, with pointers to the source code.
    - A manual demonstrating how to use the product.
    - The individual contributions of all students.

- Slides of the final presentation.
- A poster showing the most important aspects of the final product.
- The source code of the project.

3.5 Individual Contribution

The following is the workload distribution and responsibilities divided amongst the team members:

- Michalis and Dimitris: Web app development.
- Wahab and Nikos: Design Report and algorithms.
- Luka: Design Report and testing.

3.6 Evaluation and Iteration

In order to ensure our project is up to standard, continuous feedback from the client and supervisor as well as from peer feedback will be taken into account for development. Furthermore when the design of the project is finalized, a test plan will be used to evaluate the project.

Chapter 4

Design

4.1 System architecture

4.1.1 Architectural overview

Given the scope and requirements of the system, a JavaScript-based full-stack framework was the logical choice to streamline development and ensure consistency across the stack. Next.js was selected due to its maturity, comprehensiveness and familiarity within the group. Next.js integrates the presentation, application logic, and data access layers in a unified framework while maintaining separation of concerns.

Presentation layer (client-side)

- shadcn/ui was chosen as the primary component library for building the user interface, providing a set of accessible, customizable components built on top of Tailwind CSS and Radix UI. Rather than building UI components from scratch or adopting a heavyweight component framework, shadcn/ui provides a middle ground. Components are installed into the project (not imported from node_modules), meaning they can be fully customized while starting from accessible, well-designed primitives. The underlying Radix UI primitives ensure WCAG compliance out of the box, handling complex accessibility concerns like keyboard navigation, focus management, and ARIA attributes. Tailwind CSS integration means styling remains co-located with components, improving maintainability. The design system provides visual consistency without rigidity.

- Tailwind CSS, already present by virtue of being a dependency of shadcn/ui, was used for customisation and additional styling needs, leveraging its utility-first approach for rapid UI development, and keeping styles and layout closely tied to each other.

- React components were going to be a given, seeing our choice for Next.js. This allowed us to create reusable, composable UI elements, improving maintainability and consistency across the application.

Application layer (server-side)

- Next.js has amazing support for API routes, allowing us to implement server-side logic without needing a separate backend server. This simplified deployment and reduced operational overhead. An additional benefit is the ability to share code between client and server, such as data models and validation logic, and that the project is deployable as a single unit.
- Business logic for handling bookings, activities, vehicle management, and user roles was implemented within these API routes with TypeScript, ensuring type safety and reducing runtime errors. Type safety catches entire categories of errors at compile time rather than runtime. For example, database query results are typed, preventing field name typos or type mismatches. IDE autocomplete and inline documentation significantly improve developer productivity. Types serve as machine-verifiable documentation that never becomes outdated. The cost of writing type annotations is recovered through reduced debugging time and increased refactoring confidence. For a project developed by a team with varying experience levels, TypeScript's guardrails prevent subtle bugs from reaching production.
- Authentication and authorization were handled using Supabase Auth. An easy choice, as we were already using Supabase for our data layer (see further on). Authentication is a security-critical component where custom implementation risks serious vulnerabilities (password storage, session management, token generation, CSRF protection). Supabase Auth provides production-ready authentication with secure password hashing (bcrypt with per-password salts), session management via HTTP-only cookies, and integration with the database's row-level security. The system handles edge cases like password resets and email verification. The integration with Supabase's database means authentication state is automatically available in database policies, eliminating the need to pass authentication context through application layers.
- Payment processing was integrated using Stripe, leveraging its secure checkout and PCI compliance to handle payments without storing sensitive card data. While this choice was an explicit requirement from the client, it also aligned well with our goals of minimizing security risks and compliance burdens. Payment processing is security-critical and requires PCI DSS compliance. Building a custom payment system would require significant security expertise, ongoing compliance audits, and secure storage of sensitive financial data. Stripe's hosted Checkout solution eliminates these concerns entirely—sensitive card data never touches our

server, automatically achieving PCI compliance. Stripe's mature API, extensive documentation, and widespread adoption meant implementation issues could be quickly resolved through community resources. The multi-payment method support (credit/debit cards, digital wallets, buy-now-pay-later services) provides customer flexibility without additional integration work. While Stripe charges transaction fees, this cost is offset by eliminated compliance costs and reduced development time.

- Resend was selected for transactional email communication due to its modern API, ease of use, and focus on developer experience. Traditional email services (SendGrid, Mailgun) often have complex APIs designed for marketing email. Resend focuses specifically on transactional email with a simpler, more developer-friendly API. The generous free tier (100 emails/day) covers the project's development and initial production needs. Delivery tracking and bounce handling are built in, ensuring reliable email delivery.

- While a custom algorithm was developed for route optimisation, it wouldn't have been possible without leveraging the Google Maps API for calculating distance and travel time matrices. Accurate route optimisation requires real-world distance and travel time data, accounting for road networks, speed limits, and traffic patterns. Building this data infrastructure would be impractical. Google Maps provides the most comprehensive and accurate mapping data available, particularly for tourist destinations. The Distance Matrix API efficiently calculates all pairwise distances needed for optimisation.

Data layer

- **PostgreSQL database**: Managed by Supabase, providing ACID-compliant relational data storage essential for booking transactions and maintaining data integrity across complex relationships between persons, activities, vehicles, and bookings. Booking systems require strong consistency guarantees—an activity should never be over-booked, and payment records must be accurately reflected in booking status. PostgreSQL's ACID compliance ensures these guarantees. Managed hosting via Supabase eliminates operational burden (backups, updates, scaling, monitoring) while providing production-grade reliability. The integrated authentication system and row-level security features reduce the amount of custom security code needed. While real-time features and edge functions were available, the project's requirements didn't necessitate them, validating that Supabase works well as a straightforward managed PostgreSQL solution without requiring adoption of its entire ecosystem.

- **Supabase JavaScript client**: All database interactions were performed exclusively through Supabase's JavaScript client library, providing type-safe database access and automatic query optimisation. This choice eliminated the need for writing raw SQL queries while maintaining full control over data operations.

- **Row-level security (RLS) policies**: Database-level access control ensures that data protection is enforced at the lowest level, preventing unauthorized access even if application-layer security is bypassed. RLS policies were configured to restrict customers to viewing only their own bookings, while employees can access bookings relevant to their assigned activities.
- **Transaction management**: Rather than implementing complex database transaction logic with rollbacks and error recovery, the system design ensures each individual operation leaves the database in a valid state. This "fail-safe" approach simplifies error handling and makes the system more resilient to partial failures. For example, creating a booking without payment initially marks it as "pending", a valid state. When payment completes, the booking transitions to "confirmed", another valid state. There's no intermediate state where data is inconsistent. This approach trades some theoretical flexibility for significant practical simplicity.
- **Timestamp handling**: All timestamps are stored in UTC to ensure consistency across time zones, critical for a tourism business operating in a specific geographic location but potentially serving international customers.
- **Automated backups**: Supabase's managed infrastructure provides automatic database backups, ensuring data durability without requiring custom backup scripts or infrastructure management.

Architectural benefits

The Next.js unified architecture eliminates the need for separate front-end and back-end repositories through a so-called monorepo approach, while maintaining clear separation between presentation, business logic, and data access. This significantly reduces complexity compared to maintaining separate repositories and deployment pipelines. Code sharing between client and server (data types, validation logic, utilities) eliminates duplication and ensures consistency. The single codebase simplifies developer onboarding and reduces context switching. Deployment becomes a single operation rather than coordinating multiple services. Next.js's hybrid rendering capabilities (Server-Side Rendering for dynamic content, Static Generation for performance, Client-Side Rendering for interactivity) optimise both initial load time and interactive performance without requiring separate infrastructure for each rendering strategy.

4.2 Design patterns and principles

4.2.1 Architectural patterns

- **Component-based architecture**: React components serve as the fundamental building blocks of the user interface, promoting reusability and composability. Each UI element, from simple buttons to complex calendar views, is encapsulated as a component with well-defined props and behavior.

- **File-based routing**: Next.js App Router provides automatic routing based on the file system structure, eliminating the need for explicit route configuration. This convention-over-configuration approach reduces boilerplate and makes the application structure immediately apparent from the directory layout.
- **Server-client component separation**: Next.js App Router's distinction between Server Components (rendered on the server) and Client Components (interactive in the browser) was leveraged to optimise performance. Data fetching and non-interactive rendering happen server-side, while interactive elements use client-side hydration only where needed.
- **Library extraction pattern**: Commonly used logic and core algorithms were extracted into dedicated library files. Complex architectural patterns like repository layers, service layers, and dependency injection containers were explicitly avoided. These patterns add indirection and cognitive overhead, making the codebase harder to understand for developers unfamiliar with these patterns. Instead, commonly used logic was extracted into simple utility functions. This approach provides code reuse benefits while keeping the call hierarchy shallow and obvious. For a project of this scale, the simpler architecture is easier to maintain and extend.
- **API route pattern**: Business logic is encapsulated in Next.js API routes, providing clean separation between client-side presentation and server-side data manipulation. Each API endpoint handles a specific operation (e.g., creating bookings, fetching availability) with clear request/response contracts.

4.2.2 Design principles

The system adheres to several key design principles that guided implementation decisions:

- **Simplicity over complexity**: Rather than implementing complex architectural patterns like dependency injection or multi-layered service architectures, the system favors straightforward, easy-to-understand code. This reduces cognitive load for developers and makes the codebase more maintainable.
- **Don't Repeat Yourself (DRY)**: Common logic is extracted into reusable functions and components. For example, date formatting, currency conversion, and validation rules are centralized in library modules rather than duplicated across the codebase.

- **Separation of concerns**: Despite the unified Next.js architecture, clear boundaries exist between presentation logic (React components), business logic (API routes and library functions), and data access (Supabase client calls). This separation makes it easy to locate and modify specific functionality.
- **Convention over configuration**: By leveraging Next.js conventions (file-based routing, API route structure) and Supabase's client library patterns, the system minimizes explicit configuration. This reduces setup complexity and makes the project structure intuitive.
- **Fail-safe design**: Our approach to database updates ensures that the database never enters an invalid state. Each operation is designed to be independently valid, eliminating complex rollback logic and making error recovery straightforward.
- **Type safety first**: TypeScript is used throughout the entire stack, catching errors at compile time rather than runtime. Database schemas are typed through Supabase's type generation, ensuring end-to-end type safety from database to UI.

4.3 Data model

4.3.1 Entity-relationship design

The database schema consists of the following primary entities:

Core entities

- **Person**: Represents individuals using the system, storing essential information such as name and contact details. This entity serves as the base for both customers and employees, following a generalization pattern. The decision to use a single Person table rather than separate Customer and Employee tables reduces duplication and simplifies queries that need to reference individuals regardless of role.
- **Employee**: Extends the Person entity with employment-specific attributes such as role (guide, driver, administrator), hire date, and work schedule. The one-to-one relationship with Person maintains referential integrity while allowing employees to have the same authentication and contact management as customers.
- **Activity**: Represents each instance of an activity offered by the company, including activity type, time and date, and capacity limits. Activities serve as templates for bookings, allowing multiple customers to reserve spots in the same activity instance.

- **Finalised Activity**: Represents an activity which is fixed, such that employees can be assigned and vehicles allocated. This separation between Activity and Finalised Activity allows to minimise data duplication for recurring activities while still supporting scheduling and resource allocation.
- **Booking**: Represents a customer's reservation for a specific activity, linking to both the Person (customer) and Activity entities. Bookings store participant count, special requirements, payment status, and booking timestamps.
- **Activity-specific Booking tables**: Specialized booking tables for different activity types (climbing, safari, buggy tour) extend the base Booking entity with activity-specific attributes. For example, climbing bookings include the shoe size, while safari and buggy bookings include the number of seats reserved. This design balances normalization with practical data access patterns.
- **Vehicle**: Represents the company's fleet, including vehicle type, insurance number, capacity, and current status (available, in use, maintenance). Vehicles are central to activity logistics and maintenance tracking.
- **Service**: Represents maintenance or repair work performed on vehicles, recording date, description, cost, and service provider. This entity enables fleet management and helps predict future maintenance needs.
- **Attendance**: Represents employee work schedules and availability, storing date ranges and shift information. This entity supports both planning (future schedules) and historical tracking (time worked).

Key relationships

- Each Employee is a Person (one-to-one): Implemented via foreign key with CASCADE delete to maintain referential integrity.
- Each Activity can have multiple Bookings (one-to-many): Allows the same activity template to be booked repeatedly.
- Each Booking is associated with one Activity (many-to-one) and one Person (many-to-one): Establishes who booked what.
- Each Activity can become a Finalised Activity (one-to-one): Enhances a scheduled instance with resource allocation.
- Each Finalised Activity can involve multiple Vehicles (many-to-many): Implemented via junction table to support activities requiring multiple vehicles (e.g., safaris).
- Each Finalised Activity can be staffed by multiple Employees (many-to-many): Junction table supports multi-guide activities.

- Each Vehicle can have multiple Services (one-to-many): Maintains complete service history for each vehicle.
- Each Employee can have multiple Attendances (one-to-many): Tracks work schedule over time.

Design rationale

The entity model balances normalization with practical access patterns. Separating the Person and Employee entities avoids duplication of common attributes while maintaining clear role distinctions. The Activity and Finalised Activity separation allows efficient reuse of activity templates while supporting scheduling and resource allocation and preventing unused columns. Similarly, the separation of activity-specific booking tables prevents sparse columns in a single unified Booking table while maintaining clear relationships.

4.3.2 Database schema

The schema implementation leverages PostgreSQL's features for data integrity and performance:

Constraints and validation

- **Primary keys**: Auto-generated UUIDs for all entities, ensuring unique identification.
- **Foreign key constraints**: Enforced at the database level with appropriate CASCADE or RESTRICT policies to maintain referential integrity.
- **NOT NULL constraints**: Critical fields (e.g., person email, activity name, booking date) are required at the database level.
- **CHECK constraints**: Validate business rules such as positive prices, valid date ranges (start date before end date), and capacity limits.
- **UNIQUE constraints**: Prevent duplicate emails in the Person table and ensure unique vehicle registrations.

Indexing strategy

- **Primary indexes**: Automatic B-tree indexes on all primary keys for fast lookups.
- **Foreign key indexes**: Indexes on foreign key columns to optimise join operations, particularly important for frequently queried relationships like Booking→Person and Booking→Activity.
- **Composite indexes**: Multi-column indexes on frequently queried combinations such as (type, date) for finding available time slots.

- **Timestamp indexes**: Indexes on date and time columns to support efficient sorting and filtering by date.

Data types

- **Timestamps**: TIMESTAMPTZ (timestamp with time zone) for all date/time fields, stored in UTC to ensure consistency, or DATE for date-only fields.
- **Text fields**: VARCHAR with appropriate length limits for names and emails, TEXT for descriptions and notes.
- **Numeric fields**: INTEGER for counts, NUMERIC(10,2) for currency values to avoid floating-point precision issues, and UUID for unique identifiers.
- **Binary fields**: JSONB for storing flexible data structures where needed (e.g., activity-specific booking details).

This schema design provides a solid foundation for the application while remaining flexible enough to accommodate future requirements without major restructuring.

4.4 Interface design

4.4.1 Customer interface

The customer-facing interface prioritizes simplicity and conversion optimisation, guided by the principle that *everyone should understand the app*. This philosophy drove every design decision, from information architecture to visual hierarchy.

Design philosophy

The principle that "anyone should understand the app" drove the minimalist approach. Tourism customers span diverse demographics, technical skill levels, and languages. A simple, obvious interface reduces support burden and increases conversion rates. The minimalist approach reduces cognitive load and eliminates distractions that might prevent booking completion. Research shows that every additional form field or navigation step decreases conversion rates; therefore, the interface presents exactly the information needed at each step, nothing more. The modern, clean aesthetic builds trust and professionalism, essential for a tourism business where customers are making significant financial commitments. User testing with non-developers validated that the interface was immediately understandable without instructions.

Key interface elements

- **Activity catalog**: A visually appealing grid layout showcases available activities with high-quality imagery, concise descriptions, and prominent pricing. Each activity card provides just enough information to pique interest (activity type, duration, brief description) with a clear call-to-action button. Additional details, such as available dates, times and prices, are only revealed on demand, preventing information overload.
- **Booking flow**: A streamlined multi-step process guides customers through booking without overwhelming them. The flow follows a logical progression: (1) Select activity from catalog, (2) Choose date and time from available slots, (3) Enter participant details and special requirements, (4) Complete payment via Stripe Checkout. Progress indicators show completion status and allow navigation between steps. Each step validates input immediately, providing helpful error messages rather than waiting until final submission.
- **Date and time selection**: Available dates are highlighted in an intuitive calendar interface, with unavailable dates simply greyed out and uninteractive. Time slots show remaining capacity to create urgency without being manipulative.
- **Payment integration**: Stripe Checkout is embedded seamlessly within the booking flow, maintaining visual consistency while leveraging Stripe's secure, PCI-compliant payment forms. Customers never feel they've left the site.
- **Confirmation**: After successful payment, a clear confirmation page displays the booking reference, activity details, date/time, and next steps. This information is simultaneously sent via email. The confirmation page can be bookmarked or shared, providing a persistent record.
- **Responsive design**: Every interface element adapts seamlessly to mobile devices. Given the tourist context (users often booking on smartphones while traveling), mobile experience receives equal priority to desktop. Touch targets are appropriately sized, forms use mobile-optimised input types (date pickers, numeric keyboards), and layouts reflow intelligently.

Accessibility considerations

- **WCAG compliance**: All interactive elements meet WCAG 2.1 Level AA standards for color contrast, keyboard navigation, and screen reader compatibility
- **Semantic HTML**: Proper use of heading hierarchy, landmarks, and form labels ensures assistive technologies can parse content correctly

- **Focus management**: Keyboard focus is managed logically through multi-step forms, with clear visual focus indicators
- **Error handling**: Error messages are associated with form fields using ARIA attributes and are announced to screen readers

4.4.2 Employee dashboard

The employee interface emphasizes efficiency and information density, designed for daily operational use by staff who need quick access to comprehensive data. Unlike the customer interface's minimalism, the employee dashboard presents more information simultaneously, as employees are power users who benefit from reduced clicking and navigation.

Design philosophy

Employees interact with the system frequently throughout their workday and develop familiarity with the interface. Therefore, the design optimises for speed and efficiency over discoverability. Related information is grouped spatially and data density is higher to minimize scrolling and navigation. By contrast to the customer interface's minimalism, the employee interface can assume higher technical proficiency and familiarity, justifying its increased information density.

Key interface elements

- **Calendar view**: The primary interface uses FullCalendar to display a visual schedule of all activities, bookings, and employee assignments. The calendar supports multiple views (day, week, month) and activity types for rapid visual scanning. Clicking any activity opens its detail view.
- **Activity detail view**: Comprehensive information for each scheduled activity, including complete participant lists with contact information, special requirements, assigned vehicles, assigned employees, route information (for multi-stop activities like buggy tours), and current booking status. This single-page view provides everything needed to execute the activity.
- **Vehicle management**: Fleet overview showing all vehicles with current status (available, assigned, maintenance), upcoming maintenance schedules, and service history. Filtering and sorting capabilities help quickly identify available vehicles for assignment. Maintenance alerts warn when service is due.
- **Availability management**: Employees can view and update their own work schedule. Administrators can view all employee schedules and make assignments accordingly.

- **Booking management**: Search and filter all bookings by date, activity type, customer name, or status. Quick actions allow modifying bookings, sending confirmation emails, or processing refunds.
- **Route optimisation interface**: For multi-stop activities (buggy tours with hotel pickups), the interface displays the optimised route on a map with pickup sequence, estimated times, and driving directions. The route can be manually adjusted if needed before being finalized.

Responsive considerations

While primarily designed for desktop use (employees typically work from office computers), the employee dashboard remains functional on tablets and phones for field use. Guides can check activity details and participant lists on their phone during activities. However, complex administrative functions are optimised for desktop screens.

4.5 Security design

Security was treated as a first-class concern throughout system design and implementation. The architecture employs defense-in-depth, with multiple layers of security controls to protect against common web application vulnerabilities and tourism-specific threats (unauthorized booking access, payment fraud, data breaches). Complex security features were offloaded to trusted third-party services (Supabase Auth, Stripe) to leverage their expertise and reduce custom implementation risks.

4.5.1 Authentication and authorization

User authentication

The system uses Supabase Auth for all authentication operations, providing enterprise-grade security without custom implementation risks:

- **Registration**: User accounts are created by administrators. This approach prevents spam registrations and ensures that only verified users gain access to the system.
- **Login**: Users authenticate with email/password credentials. Supabase Auth validates credentials and establishes a session upon successful authentication. Failed login attempts are logged and rate-limited to prevent brute-force attacks.
- **Password requirements**: Passwords must meet minimum strength criteria (minimum 8 characters, mix of character types) enforced both client-side (immediate feedback) and server-side (security guarantee). These requirements balance security with usability, avoiding overly complex rules that encourage password reuse.

- **Password security**: Supabase Auth uses bcrypt hashing with automatically generated per-password salts. Bcrypt is computationally expensive by design, making brute-force attacks on leaked password hashes impractical. Even if the database were compromised, password hashes would be difficult to reverse. The use of unique salts per password prevents rainbow table attacks.
- **Password reset**: Users can request password resets via email. A secure, time-limited token is sent to the registered email address. The reset link expires after 1 hour, limiting the window for token interception. After reset, all existing sessions for that user are invalidated.

Session management

- **Secure cookies**: Session tokens are stored in HTTP-only cookies, making them inaccessible to client-side JavaScript. This prevents XSS attacks from stealing session tokens. Furthermore, they are marked as Secure, ensuring transmission only over HTTPS. Sensible configuration of the SameSite attribute and expiration times further enhances security.
- **Token refresh**: Supabase Auth automatically handles token refresh, maintaining user sessions without requiring re-authentication while ensuring tokens have limited lifetimes.

Role-based access control (RBAC)

The system implements three distinct access levels (unauthenticated + 2 user roles) with different permission levels:

- **Unauthenticated (Customer)**: Can view activity catalog and create bookings. Customers cannot access administrative functions.
- **Employee**: Can view all bookings and activities, access the calendar view, see participant details for assigned activities, update vehicle information, and manage their own availability. Employees cannot modify system-wide settings or access financial reporting.
- **Administrator**: Full system access including user management, activity creation and modification, vehicle fleet management, employee assignment, financial reporting, and system configuration.

Role enforcement occurs at multiple levels:

- **Application layer**: API routes check the authenticated user's role before processing requests. Unauthorized requests return 403 Forbidden errors. This prevents malicious users from accessing functionality by crafting API requests.

- **Database layer**: Row-level security (RLS) policies enforce role-based access directly in PostgreSQL, providing a security backstop even if application-layer checks are bypassed. RLS policies are defined for each table, specifying which rows each role can SELECT, INSERT, UPDATE, or DELETE.
- **UI layer**: Interface elements are conditionally rendered based on user role, hiding administrative functions from customers and employees. While this is primarily a UX improvement (not a security control), it guides users toward appropriate actions.

Row-level security (RLS) policies

Supabase's PostgreSQL database enforces access control at the row level, ensuring data isolation:

- **Customer data isolation**: Employees can only view and modify their own data. The RLS policy checks that the authenticated user's ID matches the person_id in the request.
- **Booking access control**: Employees can view all bookings for activities they are assigned to. Administrators can view all bookings. These policies prevent users from enumerating booking IDs to access other customers' information.
- **Activity visibility**: All users can view the activity catalog (public data), but only employees and administrators can view internal activity details like cost breakdown or capacity utilization.
- **Vehicle and service records**: Only employees and administrators can access vehicle information and service history, as this is operationally sensitive.
- **Employee records**: Only administrators can view employee personal information and work schedules (beyond employees viewing their own records).

RLS policies are tested as part of the development process to ensure they cannot be bypassed through unexpected query patterns.

4.5.2 Data protection

Encryption

- **Transport encryption**: All communications use HTTPS with TLS 1.2 or higher, encrypting data in transit between clients and servers. HTTP requests are automatically redirected to HTTPS. This prevents eavesdropping and man-in-the-middle attacks on public networks.

- **Database encryption**: Supabase provides encryption at rest for the PostgreSQL database, protecting data if physical storage media is compromised.
- **Backup encryption**: Database backups are encrypted before storage, ensuring sensitive data remains protected throughout its lifecycle.

Payment security

- **PCI compliance**: By using Stripe Checkout, the system achieves PCI DSS compliance without needing to undergo audits or implement complex controls. Sensitive card data (card numbers, CVV codes) never touches our server or database.
- **Tokenization**: Stripe provides tokens representing payment methods, which can be safely stored in our database for refund processing without exposing card details.
- **Webhook signature verification**: Stripe webhooks are verified using cryptographic signatures to ensure they originate from Stripe and have not been tampered with. Unverified webhooks are rejected.
- **Amount validation**: Payment amounts are calculated server-side based on activity pricing stored in the database. Client-submitted amounts are never trusted. This prevents users from manipulating payment amounts by modifying client-side code.

Input validation

All user input is validated at multiple layers to prevent injection attacks and data corruption:

- **Client-side validation**: HTML5 input attributes (type, required, pattern, min, max) provide immediate feedback and prevent obviously invalid input. Custom JavaScript validation handles complex rules (date range validation, capacity checks). Client-side validation improves UX but is not trusted for security.
- **Server-side validation**: All API routes re-validate input received from clients. This is the primary security control, as client-side validation can be bypassed. Server-side validation uses custom logic to ensure data types, formats, and business rules are enforced.
- **Database constraints**: Database-level constraints (NOT NULL, CHECK, FOREIGN KEY) provide a final validation layer. Even if application code has bugs, invalid data cannot be inserted into the database.
- **Type validation**: TypeScript's compile-time type checking catches type errors during development. Runtime type validation ensures data from external sources (API requests, database queries) matches expected types.

Injection attack prevention

- **SQL injection**: The Supabase JavaScript client uses parameterized queries for all database operations. User input is never concatenated into SQL strings, eliminating SQL injection vulnerabilities. The client library automatically escapes values and handles type conversion safely.
- **XSS (Cross-Site Scripting)**: React automatically escapes output by default, converting special HTML characters to entities. User-provided content (booking notes, names) cannot inject malicious scripts.
- **Content Security Policy (CSP)**: HTTP headers specify which sources can load scripts, styles, and other resources. This defense-in-depth control limits the impact of any XSS vulnerabilities by preventing execution of scripts from unauthorized sources.

CSRF protection

- **Next.js built-in protection**: Next.js API routes automatically include CSRF protection for state-changing requests (POST, PUT, DELETE). Requests must include a valid CSRF token obtained from the application.
- **SameSite cookies**: Session cookies use SameSite=Lax, preventing cookies from being sent with cross-site requests initiated by malicious websites.
- **Origin validation**: API routes verify the Origin and Referer headers to ensure requests originate from the legitimate application domain.

Secrets management

- **Environment variables**: All sensitive credentials (database URLs, API keys, signing secrets) are stored in environment variables, never in source code. This prevents accidental exposure through version control.
- **.env files in .gitignore**: Environment variable files are explicitly excluded from version control. [A template file (.env.example) documents required variables without exposing actual values.]
- **Key rotation**: API keys and secrets can be rotated periodically (quarterly for low-risk keys, immediately if compromise is suspected), minimizing exposure time if a secret is leaked.

4.5.3 Operational security

Logging and monitoring

- **Authentication logs**: Failed login attempts, password resets, and role changes are logged and can be inspected using Supabase Auth's audit logs for security monitoring and incident investigation.

- **Access logs**: API requests are logged with timestamps, user IDs, IP addresses, and requested resources, enabling security audits and anomaly detection.
- **Error logging**: Application errors are logged with context for debugging, but sensitive information (passwords, payment details) is redacted from logs.

4.5.4 Security limitations and trade-offs

While the system implements robust security controls, certain limitations were accepted based on risk assessment:

- **No multi-factor authentication (MFA)**: MFA would significantly improve account security but was not implemented due to time constraints. The risk is mitigated by strong password policies and monitoring for suspicious login activity.
- **Limited rate limiting**: Basic rate limiting is provided by Supabase Auth, but more sophisticated rate limiting (per-endpoint, per-IP) was not implemented. For the expected traffic volumes, this is acceptable.
- **No intrusion detection**: Advanced intrusion detection systems were not deployed, as the managed services (Supabase and Stripe) provide their own security monitoring.
- **Manual security testing**: Automated security testing (SAST, DAST) was not integrated into the CI/CD pipeline. Security testing relied on manual code review during development.

These trade-offs reflect the project scope and risk profile. Due to the limited scope and controlled user base (employees and customers of a small tourism company), the risk of sophisticated attacks is low. We believe the implemented security measures provide adequate protection while remaining implementable within time and resource constraints.

4.6 Integration design

4.6.1 Stripe payment integration

Stripe was chosen as the payment processor based on client requirements, and its integration follows Stripe's recommended patterns for security and reliability:

- **Stripe Checkout**: Rather than building custom payment forms, the system uses Stripe's hosted Checkout page. This approach offloads PCI compliance requirements to Stripe, as sensitive card data never touches our servers. Stripe Checkout provides a professional, mobile-optimised payment interface that supports multiple payment methods (cards, digital wallets) and international currencies.

- **Server-side session creation**: Payment sessions are created server-side in Next.js API routes, preventing client-side tampering with amounts or metadata. The API route validates the booking details, calculates the total amount, and creates a Stripe Checkout Session with appropriate line items and metadata.
- **Client-side redirection**: After session creation, the client is redirected to Stripe's hosted Checkout page. Upon completion (success or cancellation), Stripe redirects back to our application at predefined URLs.
- **Webhook handling**: Stripe webhooks notify our server of payment events (successful payment, failed payment, refunds). Webhook endpoints verify Stripe's signature to prevent spoofing, then update booking status in the database accordingly. This asynchronous pattern ensures accurate payment status even if users close their browser before the redirect completes.
- **Idempotency keys**: All Stripe API calls include idempotency keys to prevent duplicate charges if requests are retried due to network issues. The booking reference serves as the basis for idempotency keys, ensuring exactly-once payment processing.
- **Metadata tracking**: Each Stripe payment includes metadata linking it to the booking, enabling reconciliation and customer service queries.

4.6.2 Email communication

Resend provides transactional email capabilities with a developer-friendly API and generous free tier suitable for the project's scale:

- **Resend API integration**: All emails are sent via Resend's HTTP API from Next.js API routes. Server-side sending ensures email credentials remain secure and allows tracking email delivery status.
- **Email templates**: HTML emails are built from templates that are parameterized with booking details (activity name, date, time, participants, price) for personalization.
- **Transactional workflows**:
  - *Booking confirmation*: Sent immediately after successful payment with booking reference and activity details
  - *Pickup schedule*: Sent to employees assigned to multi-stop activities with optimised route information.
- **Delivery tracking**: Resend provides webhook notifications for email delivery events (delivered, bounced, complained), allowing monitoring of email deliverability and identification of invalid email addresses.
- **Sender authentication**: Proper SPF and DKIM records are configured to improve email deliverability and prevent emails being flagged as spam.

4.6.3 Google Maps integration

Route optimisation for multi-stop activities (particularly buggy tours with hotel pickups) relies on Google Maps APIs:

- **Distance Matrix API**: Calculates driving distances and estimated travel times between all pairs of locations (pickup points, activity destination). This matrix serves as input to the route optimisation algorithm.
- **Geocoding API**: Converts hotel names and addresses into latitude/longitude coordinates required by other APIs.
- **Route optimisation algorithm**: A nearest neighbor algorithm forms the basis of route optimisation, starting from the employee's home location and iteratively selecting the next closest unvisited pickup point. After initial route construction, a two-opt improvement heuristic iteratively swaps route segments to reduce total travel time. The algorithm prioritizes even distribution of stops (preventing one vehicle from handling all pickups) while minimizing total travel time. Vehicle assignment happens after route calculation, as customers can drive themselves in rented buggies, meaning vehicle capacity is not a constraining factor. The algorithm focuses on creating logical, efficient pickup sequences rather than solving a complex vehicle routing problem.
- **Free APIs**: Only free Google Maps APIs are used, removing the need for API keys. This keeps costs low while still providing sufficient functionality for route optimisation.

Chapter 5

Diagrams

5.1 UML Diagrams

There were different kinds of diagrams designed by the group to explain the process of how the website works in various different ways. The types of diagrams that were designed are: Use Case, Sequence, Activity, and State Machine diagrams. Of course, there is also the Class Diagram, which can explain the Database Structure, but that is covered in the next section of the chapter. Two of each type of diagrams were done, one for the customer side, in other words the booking process, and one for the employee dashboard, so they will be explained at their own dedicated subsections with references to the diagrams, which can be found in Appendix A. These diagrams, even though they might seem different, they depict the exact same process, so this is what will be explained in the following subsections.

5.1.1 Booking Diagrams

The diagrams that depict the booking process are the figures: A.1, A.3, A.5, A.8. The first thing that the customer needs to do is to choose the type of activity they want to participate in, as well as the date and time of that activity. After that, the customer needs to select how many people they want to include in their booking, as well as empty seats, if applicable to the activity. The system notifies the client whether this number of people exceeds the maximum number of people allowed for that acitivty by preventing them from adding more people to the booking. If the activity is "Jeep Safari" or "Buggy Tour," then the customer needs to state how many drivers there are in the booking. In other words, people over 21 years old with a driver's license of over 2 years, so they are not considered inexperienced drivers.

The next step is to fill in their contact details, like name, email, and phone number. If the activity is the safari, the client is also asked to fill in how many child seats they need in the safari cars, as this is the only activity where children younger than 12 years old are allowed to participate. For the two mentioned activties, the user also has to fill in the hotel where they are staying so that the company can pick them up from there before the activity starts. For the other two activities, the pickups either do not happen at all or they work differently. This is explained more in the manual C. These details are then stored in the database, and the client is redirected to the payment page, where the payment is handled through Stripe. If the payment is successful, the client is redirected to a confirmation page and is also sent a confirmation email.

5.1.2 Employee Dashboard Diagrams

The diagrams that depict the Employee Dashboard are the figures: A.2, A.4, A.6, A.7, and A.9. When the employee logs into the dashboard, they have many different things to do. They do not need to be in some specific order, as there are multiple different pages that can be used by the employees. They can go to the Employee Hours page, where they can state their availability for future activities as well as clock in and clock out before and after their shifts to keep track of their working hours, which are used to calculate their wages. They can also visit the Vehicle page, where they can keep track of the possible maintenace issues of cars or whether the car has suffered enough damage to be retired. The employee can update the maintenance information for all cars, if needed, or add new cars to the website, if the company has bought one.

Regarding the schedule page, the employee can view the schedule for the upcoming activities for the next week or month, whichever they choose as the filter. By selecting an activity, they are directed to its manifest page, where they can view more information about the activity or even edit it in terms of the maximum number of participants or the times that it starts and ends. Then, a request is sent to the database to update the details of that edited activity. The employees can also create a new activity by filling in the information about the new activity, more information on the manual C.

Last but not least, the Finalize Activity Page, where the employees can assign employees, allocate vehicles to the people of each activity, and arrange their pickups. That page is used only for the activities that require pickups, in other words, the safari and the buggy activities. First, they allocate the available employees to that activity, then they allocate the vehicles used for the activity, along with the people from each booking into different cars, and then arrange the pickups from the customers' hotels. This is where the algorithms explained in Chapter 6 are executed.

5.2 Class Diagram

Figure B.1 shows the Class Diagram. In this diagram, activity scheduling, bookings, employees, attendance, vehicles, specialized booking details are handled. The core entities are handled in the following way:

- Person - This represents a customer or participant who makes a booking with key fields being the id which is the identifier, and the person having their personal details as their contact info, with created_at as when the person was added to the system. It is linked to booking via person_id and a person can have multiple bookings.

- Activity - This represents a scheduled activity. Contains the unique identifier of the activity, the details of the activities and whether it is finalized or not. One activity can have many bookings. Each activity can either not be finalized or be finalized, having a 1 to 0..1 relationship.

- Booking - This represents a reservation made by a customer. Contains the unique identifier, the identifiers for the activity and the person as well as the details of the booking. Bookings belong to one activity and one person.

- Booking Subtypes - This is the different activity types a booking can have, namely climbing, safari and buggy. Climbing adds shoe_size for the climbing gear. Safari having drivers_in_group, child_seats and empty_seats for the transportation to the safari. Finally buggy having pickup whether the client wants to be picked up and also has drivers_in_group and empty_seats for the transportation for the client.

- Employee - Represents staff members responsible for organizing, guiding and managing activities. Contains the unique identifier for the record, and the details of the employees as well as where they currently reside. Each employee has an attendance record.

- Activity_Employee - Represents employee that will be present for the activity. Contains the identifier for the record, and the identifiers of the activity and employees. Also contains when the employee was assigned and by who.

- Attendance - Represents employees' hours and participation. Contains the identifier and the employee's identifier, with the work data logged of the employee as well as their availability. Each attendance record belongs to one employee, making it a 1 to 1 relationship.

- Vehicle - Represents vehicles used for the activities. Having the identifier, and the details of the vehicles. One vehicle can have many service records.

- Service - Represents the maintenance tracking or repairs for vehicles. Having fields of the unique identifier, the car's unique identifier and the type of service, the costs and the mechanic for the cars. Many services can belong to one vehicle.

- Finalized_Activities - Represents the finalized activity. Having the unique identifier for the class, the activity's identifier, the details of the activity, the final details and the employees scheduled for this activity. Linked to the activity and employee.

Chapter 6

Algorithms

As said earlier in the report, some processes on the employee side will happen automatically with the help of algorithms running in the background, which will help turn the event organization from manual to automatic. Both of these algorithms will be running on the "Finalize activity" page, explained in other parts of the report and they include the "Car Allocation" and the "Client Pick-ups" algorithms.

6.1 Car Allocation Algorithm

The Car Allocation Algorithm is used only for the Safari and Buggy activities. For each specific activity on a specific date, the algorithm takes as input a list of all the different bookings made for that activity. Each booking contains the name of the person that booked it and the number of adults, children, and empty seats, if any, in that booking. It also includes the number of drivers available in that booking, the nationality of the people in the booking, the pick-up location to help in the next algorithm, and the booking ID to distinguish it from the rest of the bookings.

With that information as input, the algorithm attempts to allocate the people in each booking into the available company cars for that activity. Each car has a total of 4 seats, one used by the driver of each car. There are several constraints that come with this algorithm. First, bookings are separated according to the pickup location. So, if two different bookings are not in the same pickup location, people from these two bookings cannot be allocated in the same car. The goal is to keep people from the same reservations in the same car. It goes without saying that if the total number of people in a booking is more than one car can hold, one more will be used for that booking. Also, in that case, the algorithm tries to always have at least two people from the same booking together. For example, in a booking of five people, the people will be separated into two cars, three in the one and two in the other. The goal is that no person from a booking is in a car with anybody else from their booking group.

However, if the total number of cars or drivers does not suffice to keep everyone in the same cars as their friends from the same booking, this is considered a state of emergency, where the remaining people from some bookings might be scattered across different bookings where the car still has a free seat. If there is such an emergency state, the nationality of the people in the booking is taken into account so that the remaining people that will be mixed in other cars will at least be able to communicate with each other in the same language. Another very important constraint is that each child in a booking must always be accompanied by at least one adult from the same booking, who will be responsible for the safety of that child. Of course, the number of empty seats booked in each booking is also taken into account. The empty seat means that if a person booked, for example, a safari activity for 3 people with 1 empty seat, that would mean that nobody else from the other bookings can be in that car with the people from this booking, not even in a state of emergency, since people pay for this extra seat, too.

So, after the algorithm receives all this input, it can then easily assign the groups into different cars for that activity and proceed to the next step, which is the "Client Pick-ups Algorithm" explained right below.

6.2 Client Pick-ups Algorithm

For the client pick-ups algorithm, using the employee location, client hotel locations and the final destination locations, the distance between each of these points were calculated to be used to determine the optimal path from employee to destination. To calculate the distances, the google distance matrix API was used, which was mentioned before, but essentially this API calculates distance between points (origins and destinations) using the real life traffic, roads and the mode of travel which in this case is by driving with 10000 free uses per month being sufficient for the project as there are a limited number of activities that can happen per month and below the usage cap. To determine the optimal path, the Traveling Salesman Problem (TSP) was used. The reason that TSP is used over Dijkstra or other graphing algorithms is because there is a fixed origin, being the employee's location and a fixed destination, which is where the activity is being hosted. The employee must have an optimal path to the destination with picking up clients in between. Unlike in the TSP where it takes returning into consideration, this was not utilized in the algorithm as the algorithm is only meant to take the employee to the destination. With this taken into account, both nearest neighbor and 2-opt algorithms were used to create this route.
Nearest neighbor algorithm is a greedy algorithm which takes for example 3 points, starting and destination, checks which of point A, point B, or point C is closest to the starting, for clarity purposes let's say point A is closest to starting, and then it checks which of point B and point C is closest to point A. It repeats this procedure until the full route has been created. While this creates a route, it is unoptimal, because there may be a point which is closer to the starting,

but can be in between another set of two points, and in that case, going to that point is creates a bigger time gap. To combat this issue, 2-opt algorithm was used. The 2-opt algorithm is a heuristic that checks if there are any 2-changes that make the route shorter. After computing a length of the routes, the route will be changed by the 2-opt algorithm only if there is an improvement. Therefore, with creating a route in nearest neighbor, the 2-opt algorithm optimizes this route, and returns an optimal route.

Chapter 7

Testing

Our testing approach emphasises user testing to ensure the system meets actual user needs and provides a positive experience. We continually tested the system by using it as intended, simulating real-world scenarios to identify and address issues early in the development process. To ensure a neutral perspective, we also involved individuals not directly involved in the development to perform testing. Through this approach, we gain the following benefits:

- Immediate observation of user behavior and difficulties
- Real-time verbal feedback during task execution
- Identification of usability issues automated testing cannot detect
- Validation that the system meets actual user needs and expectations

This section is organized into:

- **Test plan**: Describes the testing strategy, scope, and specific test cases designed for each system component, with detailed user testing procedures
- **Test results**: Presents the outcomes of test execution, including user test interviews and reports, identified issues, and resolutions implemented.

7.1 Test plan

A comprehensive testing strategy was developed to ensure the system meets all functional and non-functional requirements.

7.1.1 Test objectives and scope

Test objectives

- Verify all functional requirements are implemented correctly

- Validate non-functional requirements (performance, security, usability)
- Identify and fix defects
- Ensure system stability under expected load
- Validate user interface intuitiveness and accessibility

Scope

Testing covers:

- All MUST HAVE requirements
- Implemented SHOULD HAVE requirements
- Implemented COULD HAVE requirements
- API endpoints and integrations
- Database operations and data integrity
- User interface across devices and browsers

7.1.2 User test cases

Predefined test cases were created for critical functionalities users might perform, covering:

- Booking process
- Payment processing
- Employee scheduling
- Vehicle management
- Route optimization
- User authentication and authorization
- Email notifications

Booking service tests

Table 7.1: Booking service user tests

Test ID	Test description	Status
IT-BS-01	Complete booking flow from selection to confirmation	Completed
UT-BS-02	Create booking with valid data should succeed	Completed
UT-BS-03	Create booking with invalid date should fail	Completed
UT-BS-04	Create booking exceeding capacity should fail	Completed
UT-BS-05	Calculate total price with discount should be accurate	Completed
UT-BS-06	Validate customer data should reject invalid email	Completed
UT-BS-07	Check availability should return correct status	Completed
IT-BS-08	Payment processing integration with Stripe	Completed
IT-BS-09	Email notification sent after booking	Completed
IT-BS-10	Database correctly stores booking and customer data	Completed
IT-BS-11	Availability updated after successful booking	Completed

Route optimisation tests

Table 7.2: Route optimisation user tests

Test ID	Test description	Status
UT-RO-01	Calculate shortest route for 3 stops	Completed
UT-RO-02	Handle single stop route correctly	Completed
UT-RO-03	Account for vehicle capacity constraints	Completed
UT-RO-04	Prioritize pickup locations correctly	Completed
UT-RO-05	Generate valid route with all waypoints	Completed

7.1.3 System test cases

In addition to the specific user test cases above, more global tests to validate the working of the entire system were designed.

Table 7.3: System tests

Test ID	Test description	Status
SYS-01	Customer browses activities, books, pays, receives confirmation	Completed
SYS-02	Employee logs in, creates activity, assigns vehicle	Completed
SYS-03	Administrator views reports, exports data	Completed
SYS-04	System handles fully booked activity correctly	Completed
SYS-05	Multi-user concurrent booking scenario	Completed
SYS-06	Route planning with multiple pickup locations	Completed

7.1.4 Security test cases

To validate the security of the system, specific test cases focusing on common vulnerabilities and attack vectors were created.

Table 7.4: Security tests

Test ID	Test description	Status
SEC-01	SQL injection attempts rejected	Completed
SEC-02	XSS attacks prevented	Completed
SEC-03	Unauthorized API access blocked	Completed
SEC-04	Password complexity enforced	Completed
SEC-05	HTTPS enforced for all pages	Completed
SEC-06	Session timeout after inactivity	Completed
SEC-07	CSRF token validation	Completed
SEC-08	Role-based access control working	Completed

7.1.5 Performance test cases

Performance testing focuses on ensuring the system meets non-functional requirements related to speed, responsiveness, and stability under load.

Load testing objectives

- Verify system meets NFR1 (page load ¡ 3s)
- Validate concurrent user capacity (NFR2: 50 users)
- Identify performance bottlenecks

Performance test scenarios

Table 7.5: Performance Test Scenarios

Scenario	Target	Result
Booking page load	¡ 3s	Completed
Activity list API	¡ 1s	Completed
Payment processing	¡ 5s	Completed
50 concurrent bookings	No errors	Completed
Dashboard load time	¡ 3s	Completed
Route calculation	¡ 10s	Completed

7.1.6 Usability test cases

User testing methodology

Our primary approach to usability testing involves **facilitated user testing sessions** where we observe potential users interacting with the system in real-time. This methodology provides rich qualitative data about user behaviour, difficulties, and expectations.

Session structure

1. **Pre-session briefing** (1 minute):
   - Explain the purpose without revealing specific test objectives
   - Encourage thinking aloud during task execution
   - Emphasize that we're testing the system, not the user
2. **Task execution** (15 minutes):
   - User performs predetermined tasks while facilitator observes
   - Real-time notes on user behavior, hesitations, and errors
   - Minimal intervention unless user becomes completely stuck
   - Recording of verbal feedback and observations
3. **Post-session interview** (5 minutes):
   - Gather overall impressions and satisfaction
   - Discuss specific difficulties encountered
   - Collect suggestions for improvements
   - Rate interface intuitiveness and clarity

User testing participants Testing is conducted with three distinct user groups:

Customer group (5-7 participants)

- Age range: 18-60
- Mix of technical proficiency levels
- Tourists or potential tourists to Rhodes
- No prior knowledge of the system

Employee group (3-5 participants)

- Current or former activity center employees
- Varying computer literacy levels
- Experience with scheduling and customer management
- Familiar with Rhodes Adventures operations

Administrator group (2-3 participants)

- Management-level experience
- Comfortable with administrative software
- Decision-making authority
- Interest in analytics and reporting

User testing tasks

Table 7.6: Customer user tasks

Task ID	Task description
CUT-01	Find and view details for a specific activity (e.g., ”jeep safari”)
CUT-02	Book an activity for 3 people on a specific date
CUT-03	Complete the payment process (test mode)
CUT-04	Find booking confirmation information
CUT-05	Attempt to book a fully booked activity and understand why booking is not possible

Customer tasks

Table 7.7: Employee user tasks

Task ID	Task description
EUT-01	Log into the employee dashboard
EUT-02	View today’s scheduled activities and assigned participants
EUT-03	Create a new activity instance for next week
EUT-04	Update vehicle status to ”in maintenance”
EUT-05	View and understand the route map for an activity
EUT-06	Find contact details for customers on today’s bookings
EUT-07	Set personal availability for next month

Employee tasks

Table 7.8: Administrator user tasks

Task ID	**Task description**
AUT-01	View overall booking statistics for the current month
AUT-02	Create a new activity type with pricing and capacity
AUT-03	Assign an employee to a specific activity
AUT-04	Export booking data for a date range
AUT-05	Configure system settings
AUT-06	Add a new employee account with appropriate permissions

Administrator tasks

Data collection during user testing

Quantitative metrics

- Task completion rate (successful/failed/abandoned)
- Time to complete each task
- Number of errors per task
- Number of times help was requested
- Interface intuitiveness rating (1-10 scale)
- Overall satisfaction score (1-10 scale)

Qualitative data

- Verbal feedback during task execution (”think aloud” protocol)
- Observed pain points and confusion
- Positive reactions and smooth interactions
- Post-session interview responses
- Suggestions for improvements
- Comparison with competing systems they’ve used

Acceptance criteria for user testing

The system passes user testing if:

- Task completion rate $\geq 80\%$ across all user groups
- Average interface intuitiveness rating $\geq 7/10$
- Average satisfaction score $\geq 7/10$

- Average task completion time within expected ranges
- No critical usability issues preventing task completion
- Mobile responsiveness meets user expectations (NFR9)

7.1.7 Compatibility testing

Browser compatibility testing

The system will be tested on major browsers to ensure cross-browser compatibility:

Table 7.9: Browser Compatibility Test Plan

Browser	**Version**
Chrome	Latest
Firefox	Latest
Safari	Latest
Edge	Latest

Device compatibility testing

Testing will be performed on various device types and resolutions:

- Desktop (landscape only: 3440×1440, 1920×1080)
- Tablet (landscape and portrait: 2360×1640, 1920×1200)
- Mobile (portrait only: 1080×2340, 720×1600)

7.2 Test results

This section summarises the outcomes of all testing activities conducted in accordance with the test plan. It includes the results of user, system, security, performance, usability, and compatibility testing. All tests were executed on the staging environment using sample production-like data.

7.2.1 User test results

Booking service tests

Table 7.10: Booking service test results

Test ID	Description	Result
IT-BS-01	Complete booking flow from selection to confirmation	Pass
UT-BS-02	Create booking with valid data should succeed	Pass
UT-BS-03	Create booking with invalid date should fail	Pass
UT-BS-04	Create booking exceeding capacity should fail	Pass
UT-BS-05	Calculate total price with discount should be accurate	Pass
UT-BS-06	Validate customer data should reject invalid email	Pass
UT-BS-07	Check availability should return correct status	Pass
IT-BS-08	Payment processing integration with Stripe	Pass
IT-BS-09	Email notification sent after booking	Pass
IT-BS-10	Database correctly stores booking and customer data	Pass
IT-BS-11	Availability updated after successful booking	Pass

Summary: 11/11 tests passed successfully.

Route optimisation tests

Table 7.11: Route optimisation test results

Test ID	Description	Result
UT-RO-01	Calculate shortest route for 3 stops	Pass
UT-RO-02	Handle single stop route correctly	Pass
UT-RO-03	Account for vehicle capacity constraints	Partial
UT-RO-04	Prioritise pickup locations correctly	Pass
UT-RO-05	Generate valid route with all waypoints	Pass

Summary: 4/5 tests passed. The vehicle allocation algorithm does not correctly identify empty seats all the time.

7.2.2 System test results

Table 7.12: System test results

Test ID	Description	Result
SYS-01	Customer browses activities, books, pays, receives confirmation	Pass
SYS-02	Employee logs in, creates activity, assigns vehicle	Pass
SYS-03	Administrator views reports, exports data	Pass
SYS-04	System handles fully booked activity correctly	Pass
SYS-05	Multi-user concurrent booking scenario	Partial
SYS-06	Route planning with multiple pickup locations	Pass

Summary: 5/6 system tests passed. Minor issues remain with concurrency handling.

7.2.3 Security test results

Table 7.13: Security test results

Test ID	Description	Result
SEC-01	SQL injection attempts rejected	Pass
SEC-02	XSS attacks prevented	Pass
SEC-03	Unauthorized API access blocked	Pass
SEC-04	Password complexity enforced	Pass
SEC-05	HTTPS enforced for all pages	Fail
SEC-06	Session timeout after inactivity	Pass
SEC-07	CSRF token validation	Pass
SEC-08	Role-based access control working	Pass

Summary: 7/8 security tests passed. HTTPS was not enforced for all pages.

7.2.4 Performance test results

Table 7.14: Performance test results

Scenario	Target	Result	Status
Booking page load	¡ 3s	2.6s avg	Pass
Activity list API	¡ 1s	0.85s avg	Pass
Payment processing	¡ 5s	4.2s avg	Pass
50 concurrent bookings	No errors	Failed requests	Fail
Dashboard load time	¡ 3s	2.8s avg	Pass
Route calculation	¡ 10s	9.4s avg	Pass

Summary: Performance targets were generally met. Concurrency however had issues.

7.2.5 Usability test results

Quantitative summary

Table 7.15: User testing quantitative metrics

Metric	**Target**	**Result**
Task completion rate	$\geq$ 80%	92%
Average intuitiveness rating	$\geq$ 7/10	8.4/10
Average satisfaction score	$\geq$ 7/10	8.7/10
Average help requests per session	$\leq$ 2	1.2
Average task completion time	Within expected	Met

Qualitative observations

Table 7.16: Usability qualitative feedback summary

Observation	**Group**	**Impact**	**Action Taken**
"Book now" button not visible on small screens	Customer	Medium	Increased button contrast and padding
Route map loads well on mobile	Employee	Positive	N/A
Admin export button unclear	Administrator	Low	Added tooltip and label
General interface feedback: "clean and professional"	All	Positive	N/A

Summary: Overall user satisfaction and intuitiveness exceeded expectations. Minor usability improvements were implemented based on feedback.

7.2.6 Compatibility test results

Table 7.17: Cross-browser and device compatibility results

Platform	**Result**
Chrome (Latest)	Pass
Firefox (Latest)	Pass
Safari (Latest)	Pass
Edge (Latest)	Pass
Mobile (1080×2340)	Pass
Tablet (2360×1640)	Pass
Desktop Ultrawide (3440×1440)	Pass
Mobile (720×1600)	Pass
Tablet (1920x1200)	Pass

Summary: All major browsers and device resolutions displayed full compatibility. No rendering or functionality issues were detected.

7.2.7 Overall conclusions

- **Functional coverage:** Over 90% of all test cases passed.
- **Usability:** Participants rated the interface 8.4/10 on average.
- **Performance:** Meets non-functional performance thresholds.
- **Security:** Meets most security tests, however HTTPS was not enforced for all pages.
- **Compatibility:** Fully responsive and consistent across devices.

Outstanding issues:

- Minor issues with vehicle allocation algorithm.
- Minor concurrency edge case during simultaneous bookings.

Conclusion: The system meets the defined acceptance criteria and is considered ready for deployment, with minor non-critical issues logged for post-release improvement.

Chapter 8

Evaluation

In this chapter, the completed work during the project period will be summarized, and a reflection of the process and future work suggestions will be provided.

8.1 Requirements Satisfied

The following table shows the requirements that we have satisfied:

Non-Functional Requirements / Functional Requirements	**Implemented (Yes/No)**
Non-Functional Requirements	
As an employee, I would like to specify my availability.	Yes
As an employee, I would like to view, edit, and add available vehicles.	Yes
As an employee, I would like to see which vehicles are available for that activity.	Yes
As an employee, I would like to report unexpected changes to my availability.	Yes
As an employee, I would like to submit the hours I have worked.	Yes
As a user, I would like to pay for my booking easily.	Yes
As a user, I would like to book empty seats on vehicles.	Yes
As a user, I would like to make a booking.	Yes
As a user, I would like to book for multiple people.	Yes
Functional Requirements – Must Have	
The app must have a publicly accessible page for potential customers.	Yes
The app must allow potential customers to select from a list of offered activities.	Yes

The app must allow potential customers to be informed of when the selected activities are offered.	Yes
The app must allow potential customers to book any available activity.	Yes
The app must allow potential customers to pay for their booking through Stripe.	Yes
The app must keep in touch with the customer after booking.	Yes
The app must send a confirmation email after payment to the customer.	Yes
The app must allow employees to view a schedule of all activities.	Yes
The app must allow employees to view information about bookings for a specific activity.	Yes
The app must allow employees to create new available time slots for any available activity type.	Yes
The app must allow employees to specify details upon creation, such as a limit to the number of customers that can partake in an activity.	Yes
The app must track damages to vehicles and mechanic's status.	Yes
The app must have a GPS algorithm to assign different drivers to customers.	Yes
The GPS algorithm must give an optimal route from the driver to the customers to the destination.	Yes
The app must track the hours worked for employees.	Yes
The app must inform the employees before the start of an activity.	Yes
Functional Requirements – Should Have	
The app should allow customers to select additional options depending on the activity type, such as booking a private vehicle on a tour.	Yes
The app should allow customers to specify details about their booking, such as the number and age of their group.	Yes
The app should remind a customer of an upcoming booking.	No
Functional Requirements – Could Have	
The app could allow employees to send a message to all customers of an activity by Email.	No
The app could send a SMS confirmation of a booking.	No

8.2 Discussion

8.2.1 What went well

MVP & Client Satisfaction

One of our most important achievements was the client's satisfaction with the quality of the project. According to the client, the product contains all the basic characteristics that it should have and even more. Some future adjustments will be needed for the website to be ready for use by the company, mainly scalability issues discussed in a following section. They have also stated that the website is very easy to navigate and also quite intuitive, as in the tasks that both employees and customers should be doing while using the website.

User Testing

Another critical aspect of the project was to make sure that the future users find the website easy to work with, and, apart from the client themselves, the users who tested the website were really satisfied with the website in terms of being easy to navigate and perform the user tasks. The user testing was performed meticulously, with attention to detail of every single website functionality. During testing, it was ensured that the website would prevent the users from doing things that they were not supposed to, such as booking a fully booked activity. In fact, it was in user testing that some bugs in the system were detected and later fixed by the group.

8.2.2 What didn't go well

Communication

While we did have a team-bonding session before the module, where we went to grab beers at a bar, our communication had remained subpar. Due to planning the project early on and assigning tasks accordingly for the project amongst team members, there was very little check-in on the progress of the tasks as the project continued on. Furthermore, with the long deadlines there were often times when there was no communication with certain team members, and thus some members were left in the dark about the current state of the project. This mainly continued on until week 4, where it did improve with most team members starting to be more actively involved; however, the main issues remained. The tasks were quite large, so team members having a long deadline, understandably so, had disincentivized communication as the focus on the task was too great. So while the communication had somewhat improved with a bit of progress reports after week 4, there were still many times where there was no communication at all, and team members were left in the dark or unaware that a section was completed. This sometimes led to duplicate work, as a section would be completed, but with the lack of communication, another team member thought to complete it, leading in both doing the same work. In the future, this could be improved by holding regular weekly progress reports, or by splitting small tasks amongst team members rather than large tasks, to keep everyone's progress consistent and to communicate in order to integrate these smaller tasks.

Testing

The testing of the project was held far later than ideal, leading to suboptimal testing. This means that some functionalities were not given enough time to be patched, while others were hastily tested, meaning that the accuracy of the results can be a bit questionable. An example would be the concurrency issues, where this was not given enough time to be fully tested to make sure that concurrency loads work well with multiple users. Other aspects that have issues would be edge cases, as some few edge cases would report back with an error.

8.3 Future Work

Even though the client was satisfied with the quality of the product so far, there is always room for improvement. One of the major things that was not implemented eventually because of the time constraint was the confirmation of booking by SMS. At the start of the project, it was one of the features that the group wanted to include in the final product but ended up not being implemented in the end.

More critical aspects of the product that needs improvement is the website's scalability as well as concurrent bookings. First of all, since the website is not hosted in a big enough server, it cannot sustain more than, for example, 20 people on it at the same time, which makes sense because it is hosted locally as of now. Regarding the concurrent bookings, there has been no function implemented for managing two bookings for the same activity that are happening simultaneously. For example, the website does not account for cases where, for example, a safari activity has 26 out of 30 seats booked, and two people want to book the remaining for at the same time. There is no lock existing for that case, and it will definitely need to be implemented in the future for the company to use the product.

And one last thing that the group would like to implement in the future is being able, as a customer, to edit your booking. For example, the customer should be able to change the date of the activity in which they will participate or get a refund if they want to cancel the reservation they made. However, it will not be possible to change the number of people involved in a booking. For that, a cancellation would be needed so that they can make a new booking with a different number of people or type of activity that they want to do.

8.4 Conclusion

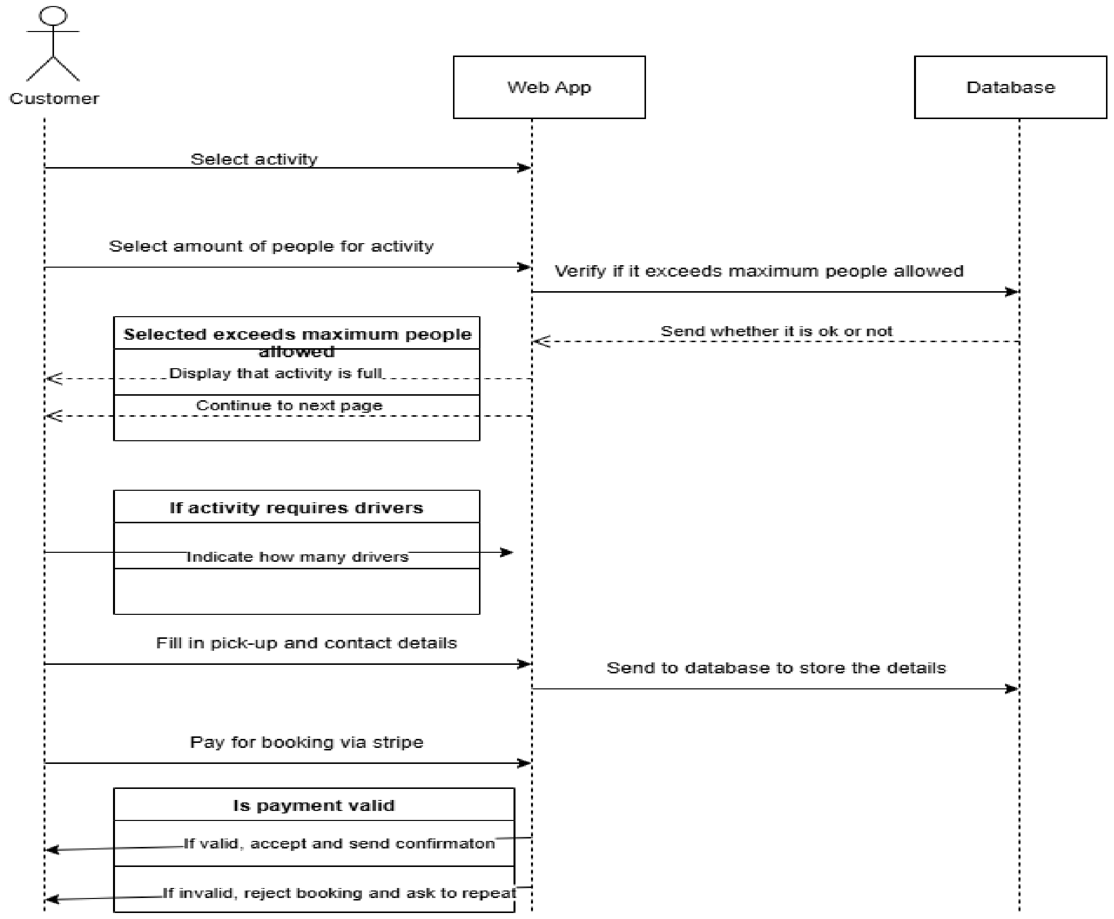
In conclusion, the Rhodes Adventure booking project aimed to create a booking system with the maintainer page, and this goal has been achieved. The app has been successfully built that fulfils the core requirements.
The team has demonstrated strong focus on the design of the project and functionality. While the core requirements were fulfilled, certain optional requirements were left unimplemented, mainly due to the sheer amount of must have

requirements that needed to be implemented and tested in order to function properly. Despite issues in scalability and concurrency, the project achieved the intended client's intended objectives.
The project has also offered valuable lessons in teamwork and communication. As these things were a big hurdle, and did not prevent the completion of the project, this is a learning experience to be improved in the future.
Overall despite hurdles and certain things that could have been improved, the project is successfully completed. With these lessons, in the future would help in developing a more robust project, or developing the project much earlier, which would help with thoroughly complete testing.

Appendices

Appendix A

UML Diagrams



Customer
Web App
Database
Select activity
Select amount of people for activity
Verify if it exceeds maximum people allowed
Send whether it is ok or not
Selected exceeds maximum people allowed
Display that activity is full
Continue to next page
If activity requires drivers
Indicate how many drivers
Fill in pick-up and contact details
Send to database to store the details
Pay for booking via stripe
Is payment valid
If valid, accept and send confirmaton
If invalid, reject booking and ask to repeat

Figure A.1: Customer Sequence Diagram

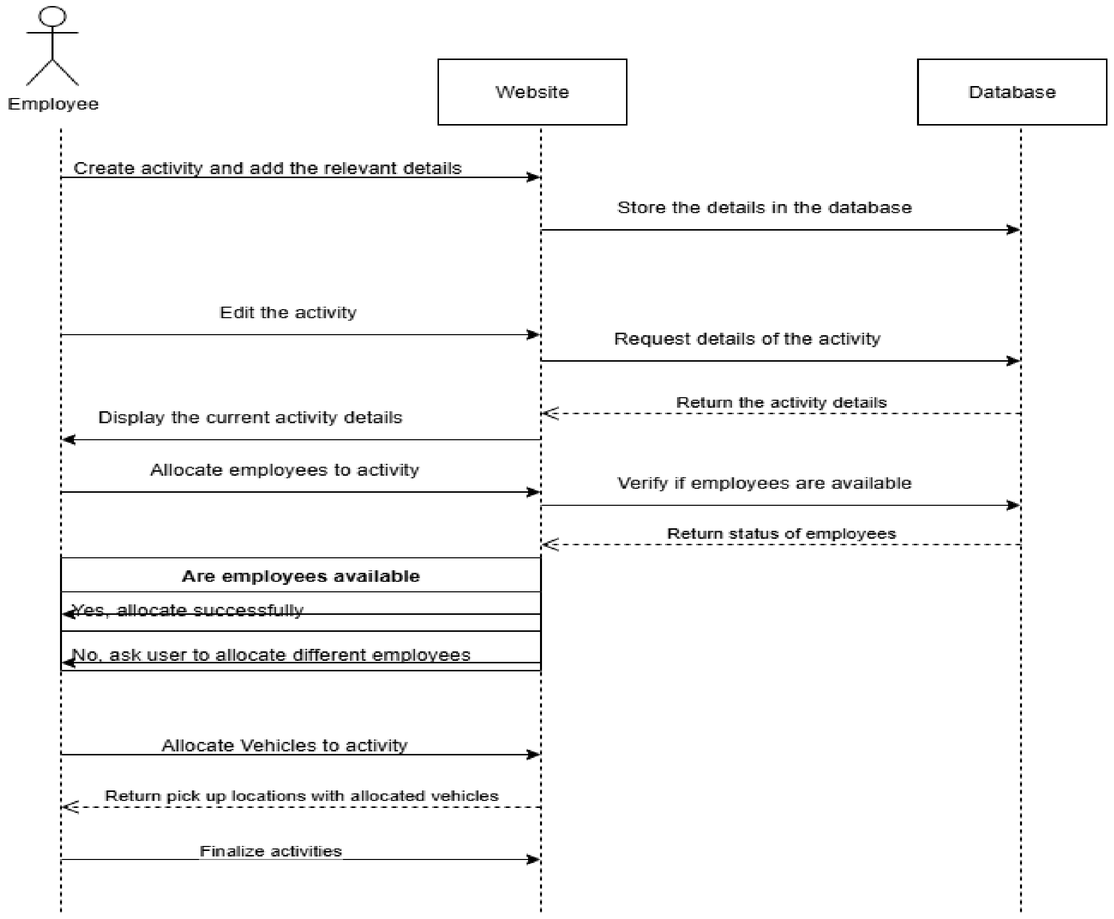

Employee
Website
Database
Create activity and add the relevant details
Store the details in the database
Edit the activity
Request details of the activity
Return the activity details
Display the current activity details
Allocate employees to activity
Verify if employees are available
Return status of employees
Are employees available
Yes, allocate successfully
No, ask user to allocate different employees
Allocate Vehicles to activity
Return pick up locations with allocated vehicles
Finalize activities

Figure A.2: Employee Sequence Diagram

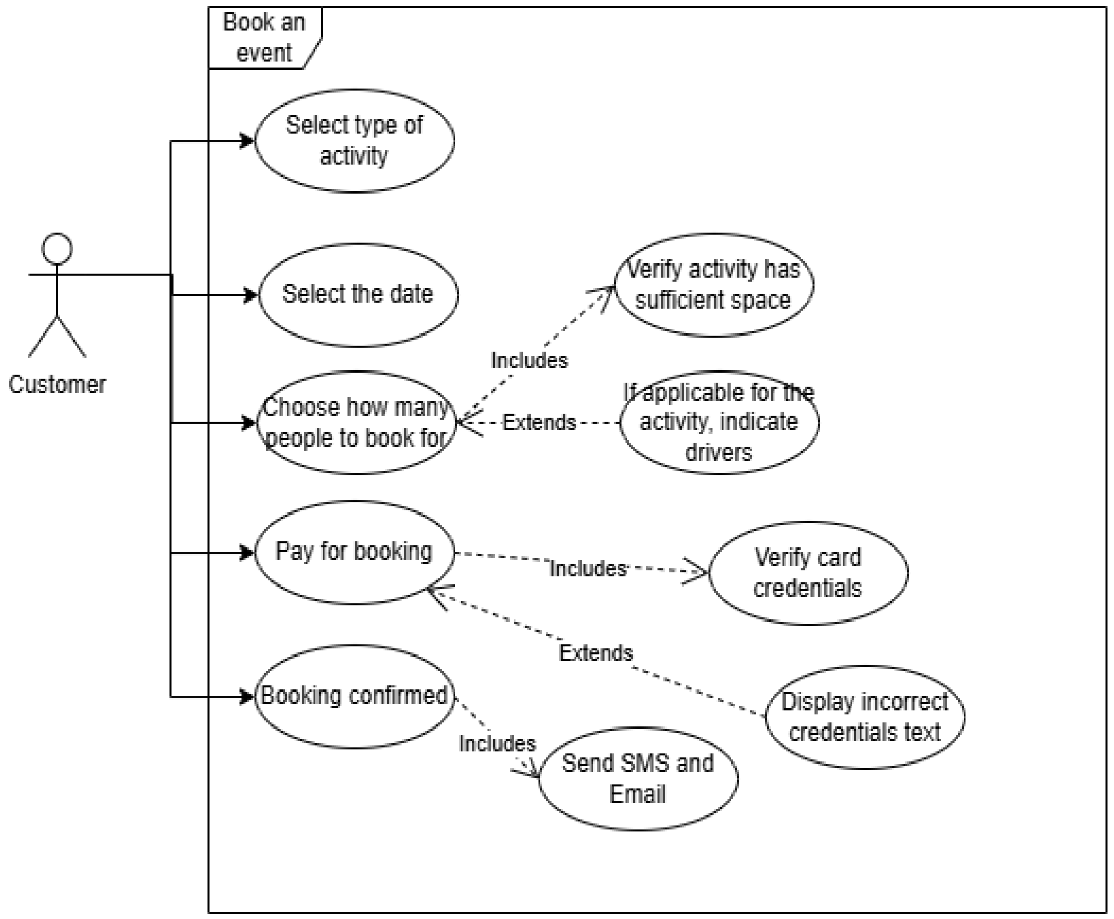


Figure A.3: Customer Use Case Diagram

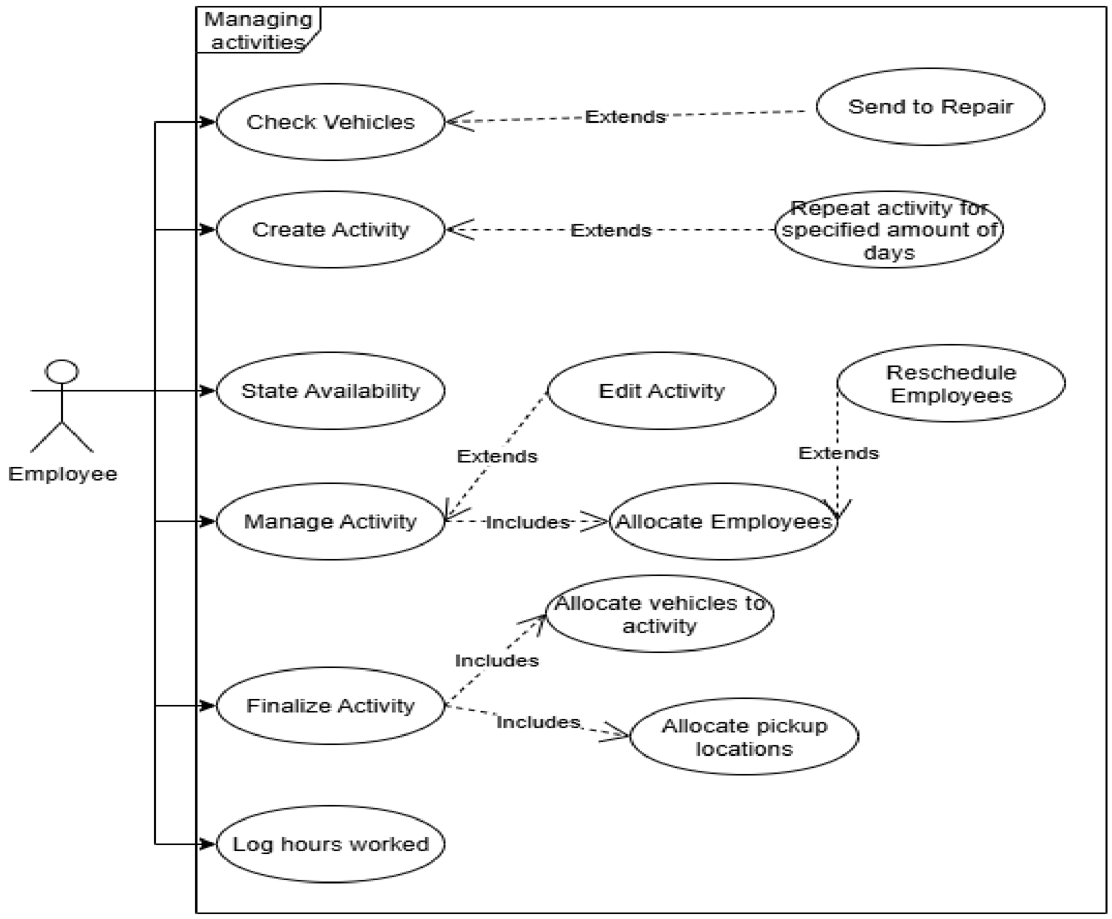
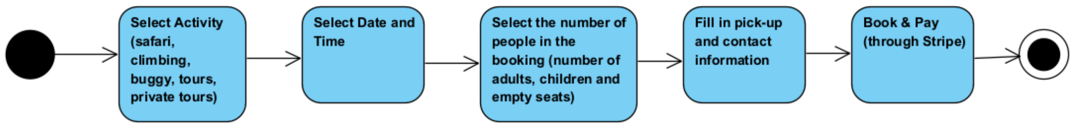


Figure A.4: Employee Use Case Diagram



Select Activity (safari, climbing, buggy, tours, private tours)
Select Date and Time
Select the number of people in the booking (number of adults, children and empty seats)
Fill in pick-up and contact information
Book & Pay (through Stripe)

Figure A.5: Booking Activity Diagram

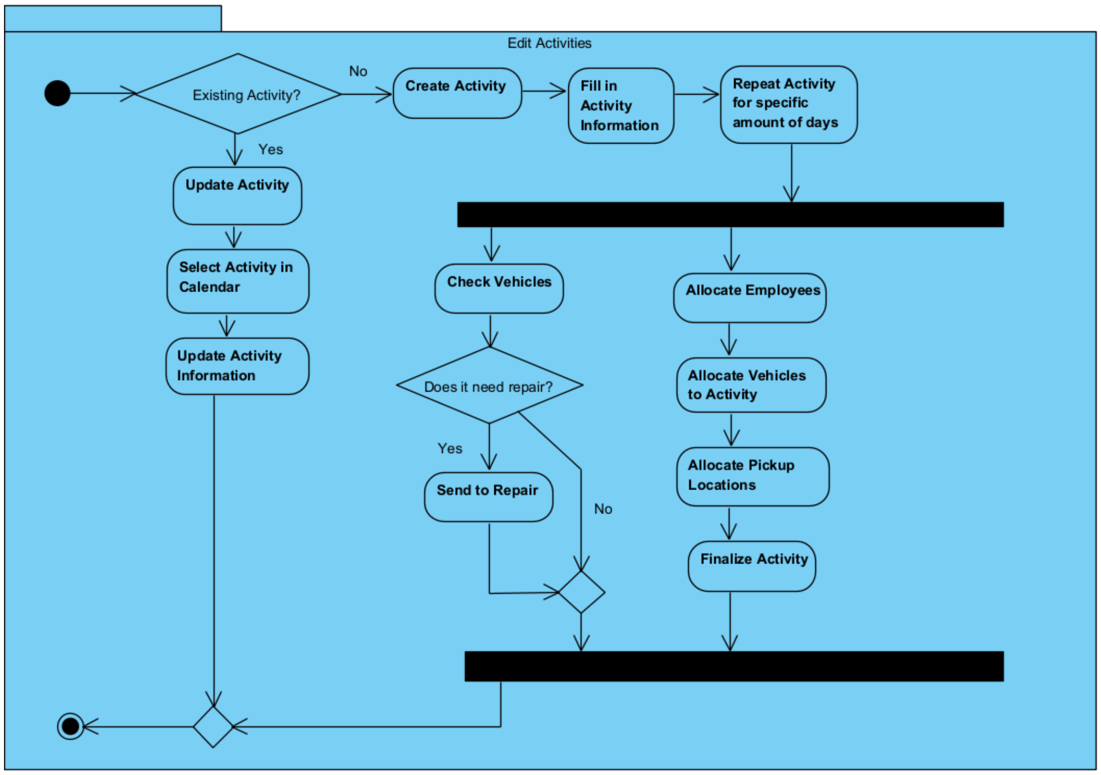

Edit Activities
Existing Activity?
No
Yes
Create Activity
Fill in Activity Information
Repeat Activity for specific amount of days
Update Activity
Select Activity in Calendar
Update Activity Information
Check Vehicles
Does it need repair?
Yes
No
Send to Repair
Allocate Employees
Allocate Vehicles to Activity
Allocate Pickup Locations
Finalize Activity

Figure A.6: Manage Activities Activity Diagram

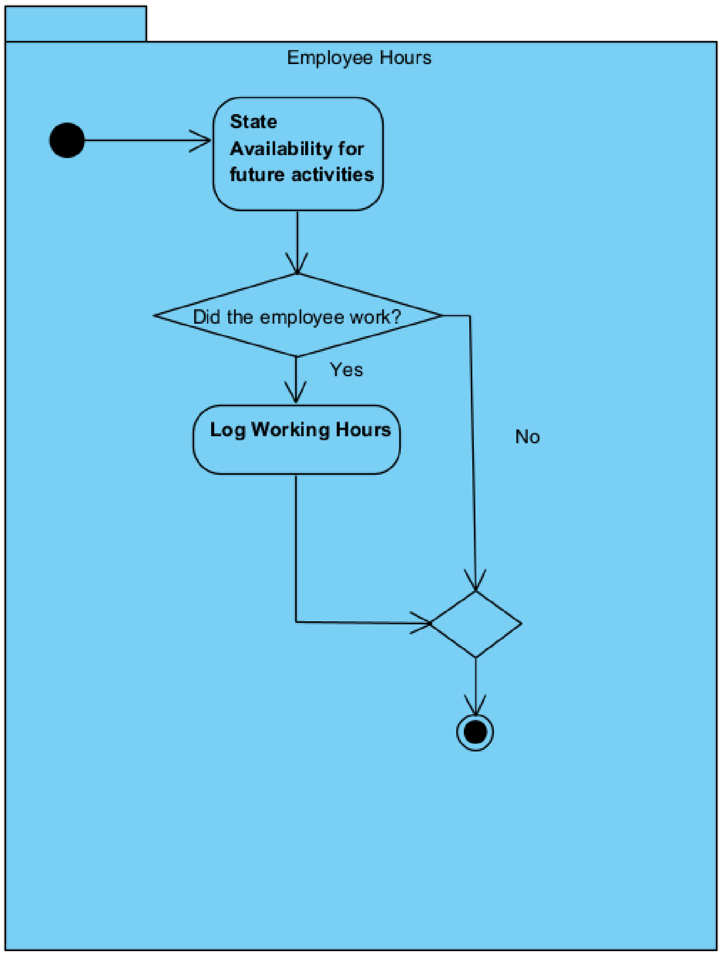


Figure A.7: Employee Hours Activity Diagram

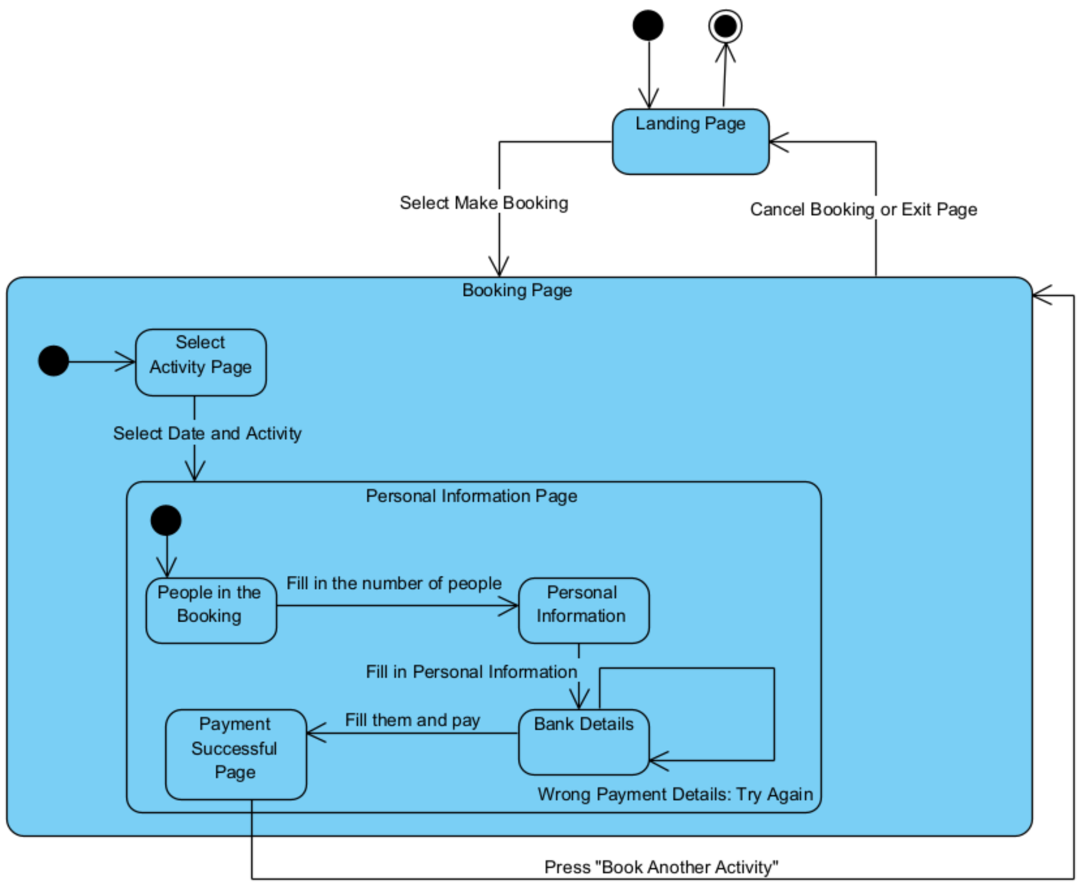


Figure A.8: Booking State Machine Diagram

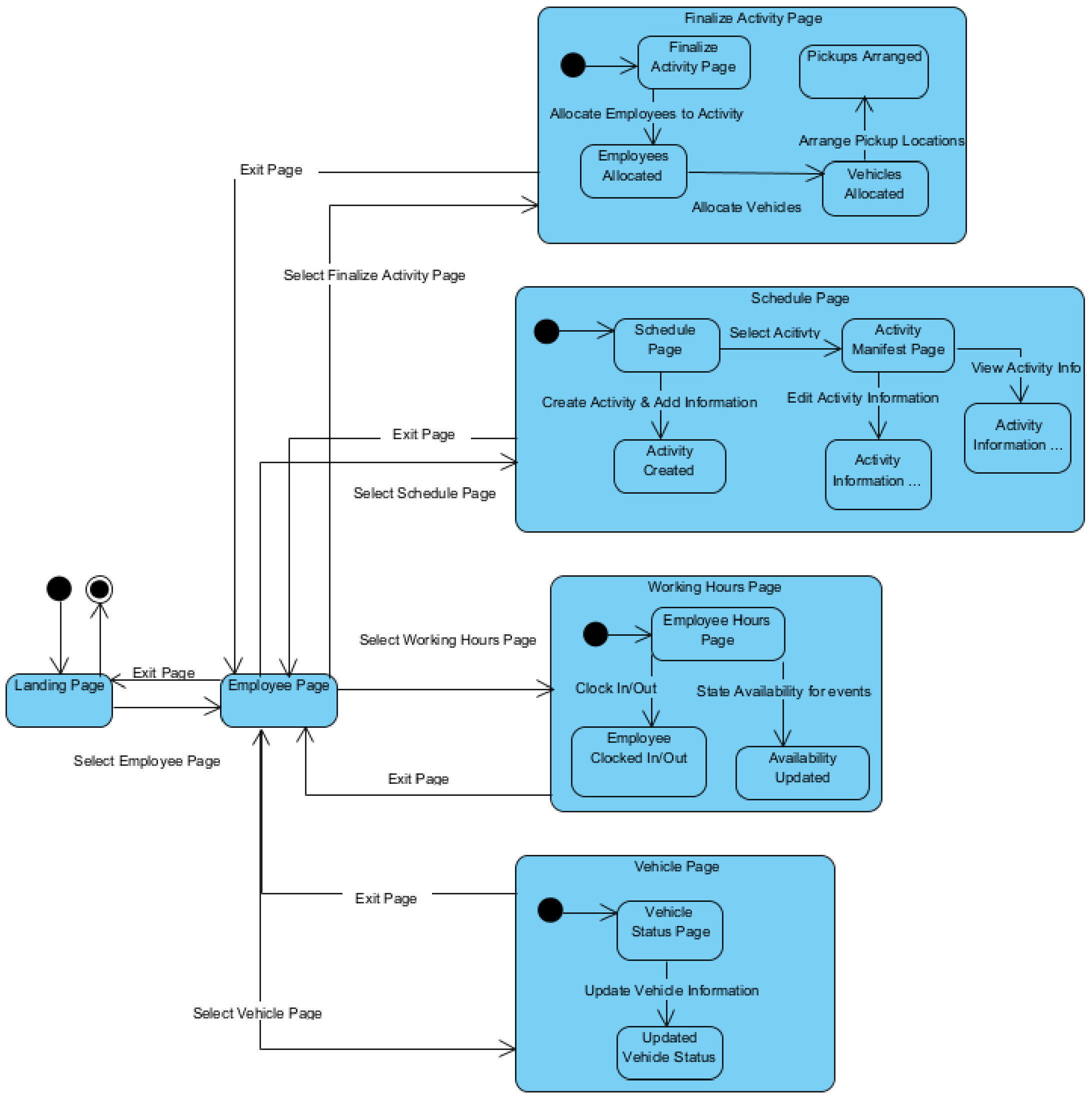


Figure A.9: Employee Dashboard State Machine Diagram

Appendix B

Class Diagram

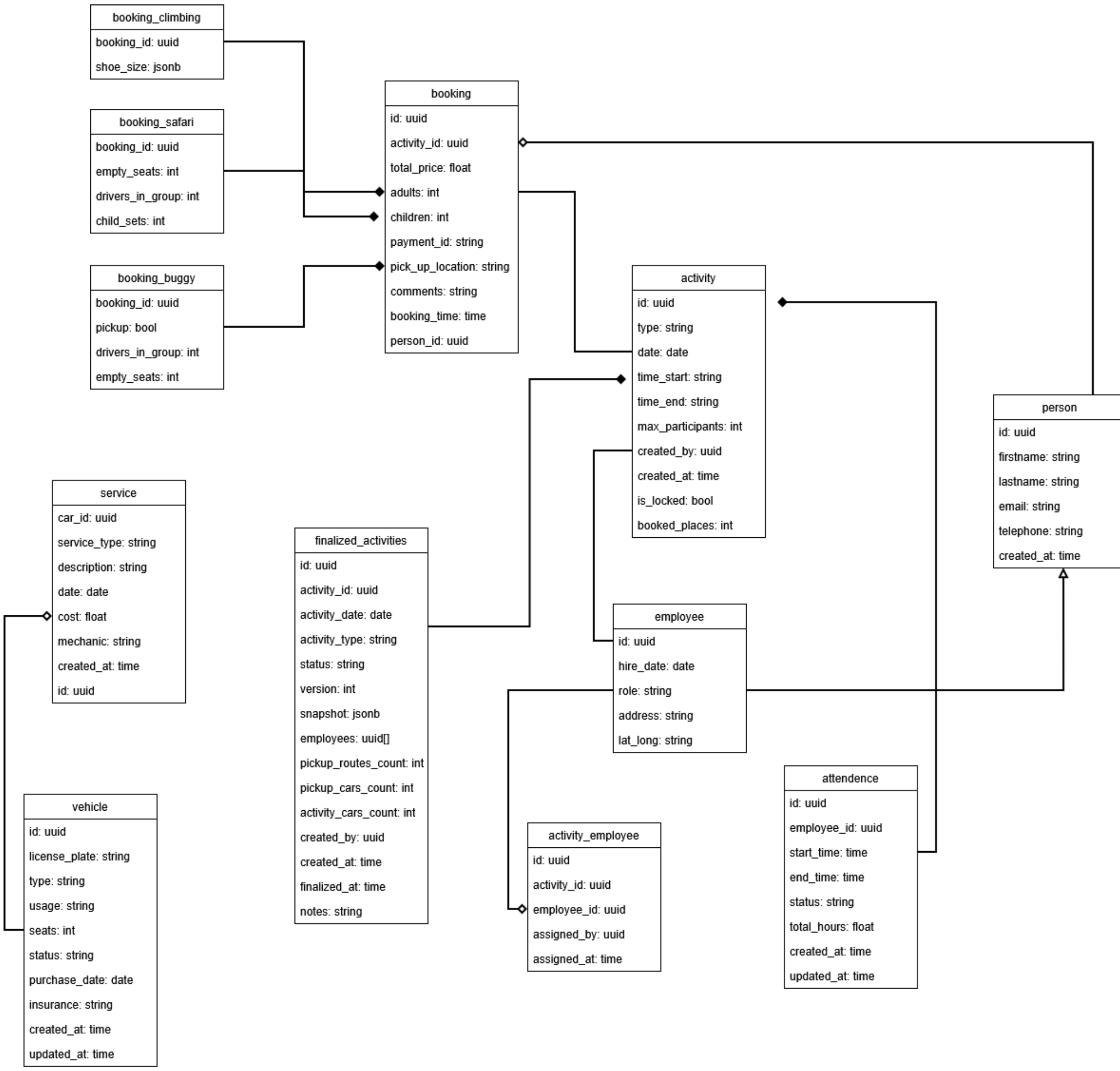


Figure B.1: Class Diagram

Appendix C

Manual

C.1 Deployment

The app can be deployed on any server infrastructure capable of running a Node.js application. Additionally, it requires a Supabase instance, either self-hosted or managed. This deployment guide demonstrates a full deployment using Coolify, a self-hosted Platform-as-a-Service (PaaS) solution, including a self-hosted Supabase instance. Coolify was chosen for this guide because it provides a straightforward deployment process with minimal configuration, making the instructions accessible and easy to follow. Note however that there are many other ways to deploy it beyond what is described here.

C.1.1 Prerequisites

Before beginning the deployment process, ensure you have the following prerequisites in place:

Technical Knowledge

This deployment guide assumes you have basic familiarity with:

- Server administration (SSH access, command-line operations, user management)
- Network administration (DNS configuration, firewall rules, port forwarding)
- Web application concepts (environment variables, API keys, database connections)

Third-Party Service Accounts

The application integrates with several third-party services. You must create accounts and obtain API keys for the following services before deployment:

- **Stripe**: For payment processing. You will need both publishable and secret API keys. Create an account at https://stripe.com and follow their documentation to obtain API keys.
- **Resend**: For sending transactional emails. You will need a Resend API key. Create an account at https://resend.com and generate an API key in your dashboard.
- **Google Cloud Platform**: For the Maps API and Distance Matrix API used in route optimization. You will need a Google API key with the appropriate APIs enabled. Create a project in the Google Cloud Console at https://console.cloud.google.com.

Creating accounts and configuring these services is beyond the scope of this manual. Please refer to each service's official documentation.

C.1.2 Deployment Overview

The deployment process consists of the following major steps:

1. Installing and configuring the Coolify platform
2. Deploying and configuring a Supabase instance
3. Initializing the database schema and creating an administrator account
4. Configuring environment variables and third-party integrations
5. Deploying the application
6. Verifying the deployment and performing post-deployment tasks

The following sections detail each of these steps.

C.1.3 Installing Coolify

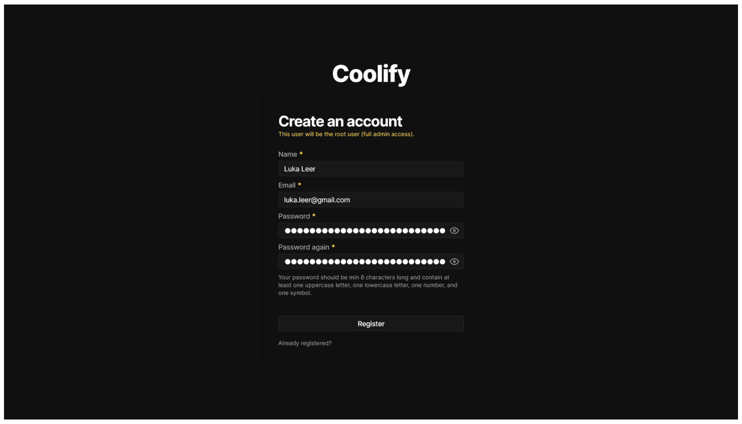
The first step in deploying the application is to install Coolify on your server. Coolify provides a streamlined approach to deploying and managing applications. To install Coolify, connect to your server (in our case running Ubuntu 24) via SSH and execute the installation script:

```
curl -fsSL https://cdn.coollabs.io/coolify/install.sh | sudo bash
```

This script automatically handles the installation of all necessary dependencies, including Docker and other required components. The installation process typically takes several minutes, depending on your server specifications and internet connection speed. Once complete, Coolify will be accessible through your server's web interface.

C.1.4 Creating a Coolify Account

After the installation completes, navigate to your server's IP address or domain name in a web browser to access the Coolify interface. You will be presented with an onboarding screen where you need to create your Coolify account, as shown in Figure C.1. This account will have full control over all applications and resources managed by Coolify, so ensure you choose a strong password and store the credentials securely.



Create an account

This user will be the root user (full admin access).

Name *

Luka Leer

Email *

luka.leer@gmail.com

Password *

●●●●●●●●●●●●●●●●●●●●●●●●●●

Password again *

●●●●●●●●●●●●●●●●●●●●●●●●●●

Your password should be min 8 characters long and contain at least one uppercase letter, one lowercase letter, one number, and one symbol.

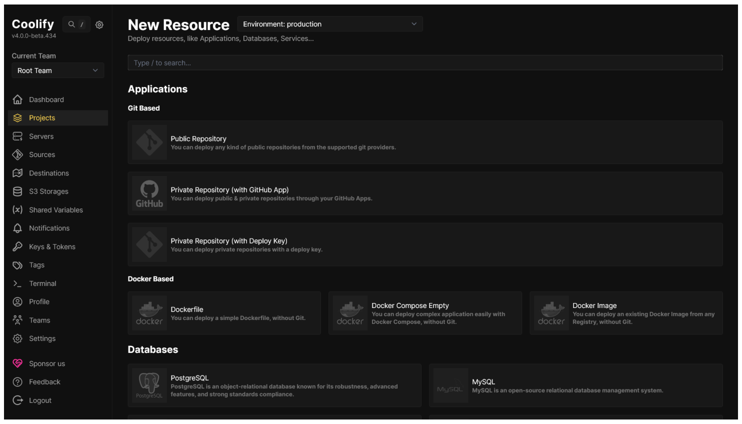
Figure C.1: Coolify Onboarding Screen

C.1.5 Setting Up Supabase

With your Coolify account created, the next step is to deploy the Supabase instance that will manage all of our data.

Creating a Project and Resource

Begin by creating a new project within Coolify to organize your resources. Projects in Coolify help you group related applications and services together. Once your project is created, select the option to add a new resource. You will be greeted with an interface similar to that in Figure C.2. Search for "Supabase" and select it.



New Resource

Deploy resources, like Applications, Databases, Services...

Applications

Git Based

Public Repository
You can deploy any kind of public repositories from the supported git providers.

Private Repository (with GitHub App)
You can deploy public & private repositories through your GitHub Apps.

Private Repository (with Deploy Key)
You can deploy private repositories with a deploy key.

Docker Based

Dockerfile
You can deploy a simple Dockerfile, without Git.

Docker Compose Empty
You can deploy complex application easily with Docker Compose, without Git.

Docker Image
You can deploy an existing Docker Image from any Registry, without Git.

Databases

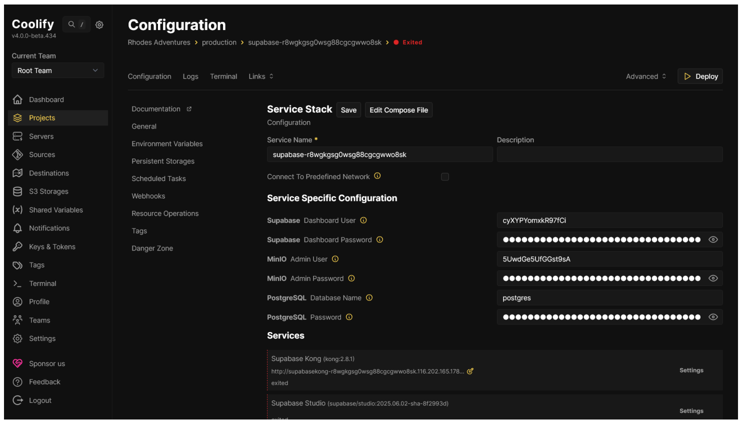
PostgreSQL
PostgreSQL is an object-relational database known for its robustness, advanced features, and strong standards compliance.

MySQL
MySQL is an open-source relational database management system.

Figure C.2: Creating a New Supabase Resource in Coolify

Deploying Supabase

After selecting Supabase, you will be presented with the configuration screen shown in Figure C.3. This screen displays critical credentials including the dashboard username and password. These credentials are essential for accessing the Supabase administration interface, so make sure to record them in a secure location before proceeding. Once you have noted these details, click the ”Deploy” button to initiate the Supabase deployment.



Configuration

Rhodes Adventures > production > supabase-r8wgkgsg0wsg88cgcgwwo8sk > ● Exited

Service Stack

Configuration

Service Name *: supabase-r8wgkgsg0wsg88cgcgwwo8sk

Description:

Connect To Predefined Network

Service Specific Configuration

Supabase Dashboard User	cyXYPYomxkR97fCi
Supabase Dashboard Password	••••••••••••••••••••••••••••••••
MinIO Admin User	5UwdGe5UfGGst9sA
MinIO Admin Password	••••••••••••••••••••••••••••••••
PostgreSQL Database Name	postgres
PostgreSQL Password	••••••••••••••••••••••••••••••••

Services

Supabase Kong (kong:2.8.1)
http://supabasekong-r8wgkgsg0wsg88cgcgwwo8sk.116.202.165.178...
exited

Supabase Studio (supabase/studio:2025.06.02-sha-8f2993d)

Figure C.3: Fresh Supabase Deployment Configuration

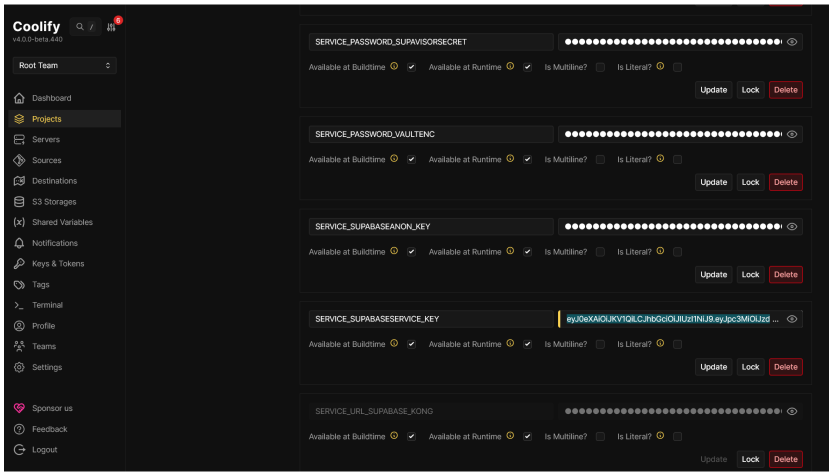
The deployment process will provision all necessary components. This may take several minutes to complete, depending on your server’s processing power and network connection.

Collecting Environment Variables

Once the Supabase deployment is complete, navigate to the "Environment Variables" tab within the Supabase resource configuration. Here you will find several critical variables that the application needs to connect to the database, as shown in Figure C.4. Specifically, you need to record the following values:

- `SERVICE_URL_SUPABASE_KONG`: The public URL for accessing the Supabase API
- `SERVICE_SUPABASEANON_KEY`: The anonymous key for client-side API requests
- `SERVICE_SUPABASESERVICE_KEY`: The service role key for server-side operations with elevated privileges

These keys will be used later when configuring the application. Store them securely as they provide access to your database.



```
SERVICE_PASSWORD_SUPAVISORSECRET   ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●
Available at Buildtime ✓   Available at Runtime ✓   Is Multiline? ☐   Is Literal? ☐

SERVICE_PASSWORD_VAULTENC          ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●
Available at Buildtime ✓   Available at Runtime ✓   Is Multiline? ☐   Is Literal? ☐

SERVICE_SUPABASEANON_KEY           ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●
Available at Buildtime ✓   Available at Runtime ✓   Is Multiline? ☐   Is Literal? ☐

SERVICE_SUPABASESERVICE_KEY        eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJzd ...
Available at Buildtime ✓   Available at Runtime ✓   Is Multiline? ☐   Is Literal? ☐

SERVICE_URL_SUPABASE_KONG          ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●
Available at Buildtime ✓   Available at Runtime ✓   Is Multiline? ☐   Is Literal? ☐
```

Figure C.4: Supabase Environment Variables

C.1.6 Configuring the Database Schema

With Supabase deployed and running, the next step is to configure the database schema that defines the structure of all our data.

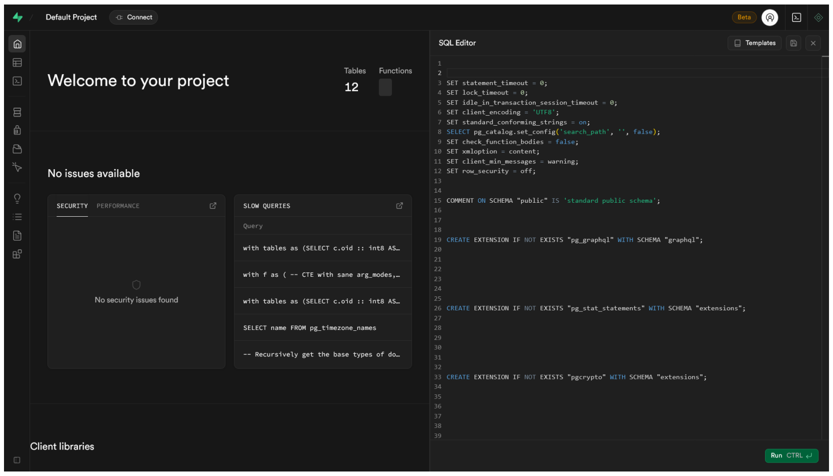
Accessing the Supabase Dashboard

Open your web browser and navigate to the URL specified in the `SERVICE_URL_SUPABASE_KONG` environment variable you recorded earlier. This will take you to the Supabase dashboard login page. Use the dashboard username and password that were displayed earlier to log in.

Running the Schema Migration

Once logged into the dashboard, navigate to the SQL Editor section. This tool allows you to execute SQL commands directly against your database. In the SQL Editor, paste the contents of the database migration file located at `supabase/migrations/20251104231632_remote_schema.sql` in the code repository.

This migration file contains all the SQL commands necessary to create the complete database structure. After pasting the migration SQL, execute it by clicking the "Run" button, as shown in Figure C.5.



```
with tables as (SELECT c.oid :: int8 AS…
with f as ( -- CTE with sane arg_modes,…
with tables as (SELECT c.oid :: int8 AS…
SELECT name FROM pg_timezone_names
-- Recursively get the base types of do…
```

```sql


SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;


COMMENT ON SCHEMA "public" IS 'standard public schema';



CREATE EXTENSION IF NOT EXISTS "pg_graphql" WITH SCHEMA "graphql";






CREATE EXTENSION IF NOT EXISTS "pg_stat_statements" WITH SCHEMA "extensions";






CREATE EXTENSION IF NOT EXISTS "pgcrypto" WITH SCHEMA "extensions";
```

Figure C.5: Supabase SQL Editor with Migration Script

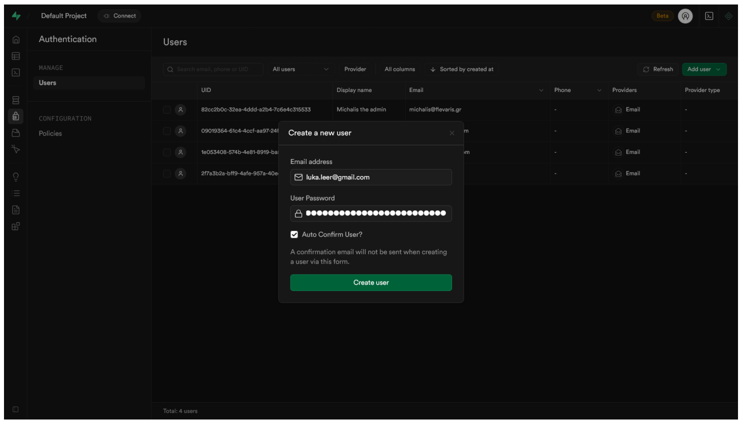
C.1.7 Creating the Administrator User

Our system uses role-based access control to manage permissions. Before the application can be used, you need to create at least one administrator user who will have full access to manage the system.

Creating the User Account

In the Supabase dashboard, navigate to the "Authentication" section and select the "Users" tab. Click the button to create a new user. You will be prompted to enter an email address and password for this administrator account, as shown in Figure C.6.

Choose a strong password and use a valid email address that can receive confirmation messages. This will be the primary administrator account for the system, so ensure these credentials are stored securely. After entering the details, create the user.



	UID	Display name	Email	Phone	Providers	Provider type
	82cc2b0c-32ea-4ddd-a2b4-7c6e4c315533	Michalis the admin	michalis@flevaris.gr	-	Email	-
	09019364-61c4-4ccf-aa97-24		m	-	Email	-
	1e053408-574b-4e81-8919-ba		m	-	Email	-
	2f7a3b2a-bff9-4afe-957a-40e			-	Email	-

Create a new user

Email address: luka.leer@gmail.com

User Password: ●●●●●●●●●●●●●●●●●●●●●●●●●

[✓] Auto Confirm User?

A confirmation email will not be sent when creating a user via this form.

Create user

Total: 4 users

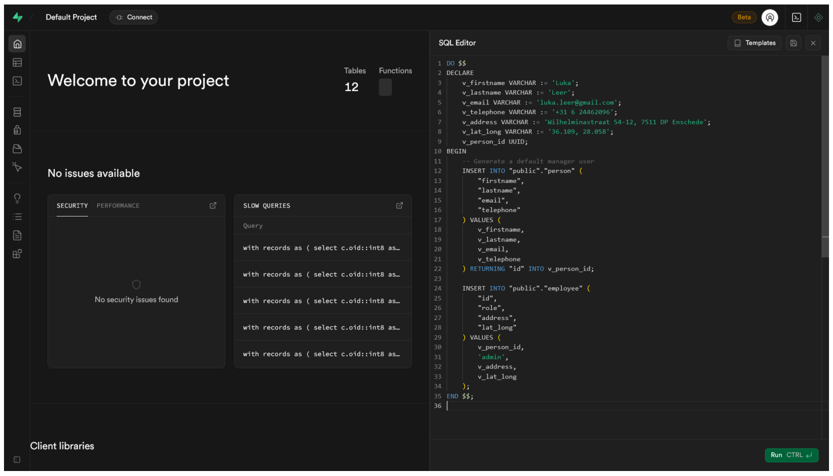
Figure C.6: Creating the Administrator User in Supabase

Granting Administrator Privileges

Creating the user account in Supabase's authentication system is not sufficient—you also need to grant this user administrator privileges. Return to the SQL Editor and insert there the contents from the seed file located at `supabase/seed.sql` in the code repository.

This seed file contains SQL commands to create the necessary Person and Employee records in the database and assign the administrator role. Before executing this file, you need to replace the variables with values relevant to you.

Once you have customised the seed data, execute it, as illustrated in Figure C.7. This will create the database records linking the Supabase authentication user to an Employee and Person record with administrator privileges.



```
with records as ( select c.oid::int8 as…
with records as ( select c.oid::int8 as…
with records as ( select c.oid::int8 as…
with records as ( select c.oid::int8 as…
with records as ( select c.oid::int8 as…
```

```sql
DO $$
DECLARE
    v_firstname VARCHAR := 'Luka';
    v_lastname VARCHAR := 'Leer';
    v_email VARCHAR := 'luka.leer@gmail.com';
    v_telephone VARCHAR := '+31 6 24462096';
    v_address VARCHAR := 'Wilhelminastraat 54-12, 7511 DP Enschede';
    v_lat_long VARCHAR := '36.109, 28.058';
    v_person_id UUID;
BEGIN
    -- Generate a default manager user
    INSERT INTO "public"."person" (
        "firstname",
        "lastname",
        "email",
        "telephone"
    ) VALUES (
        v_firstname,
        v_lastname,
        v_email,
        v_telephone
    ) RETURNING "id" INTO v_person_id;

    INSERT INTO "public"."employee" (
        "id",
        "role",
        "address",
        "lat_long"
    ) VALUES (
        v_person_id,
        'admin',
        v_address,
        v_lat_long
    );
END $$;
```

Figure C.7: Running the Database Seed Script

C.1.8 Deploying the Application

With the database fully configured, you can now deploy the application itself. The application is a Next.js web application that serves both the customer booking interface and the employee dashboard.

Creating the Application in Coolify

Return to the Coolify dashboard and create a new application within your project. Coolify offers two primary methods for deploying applications: manual code upload or Git repository integration.

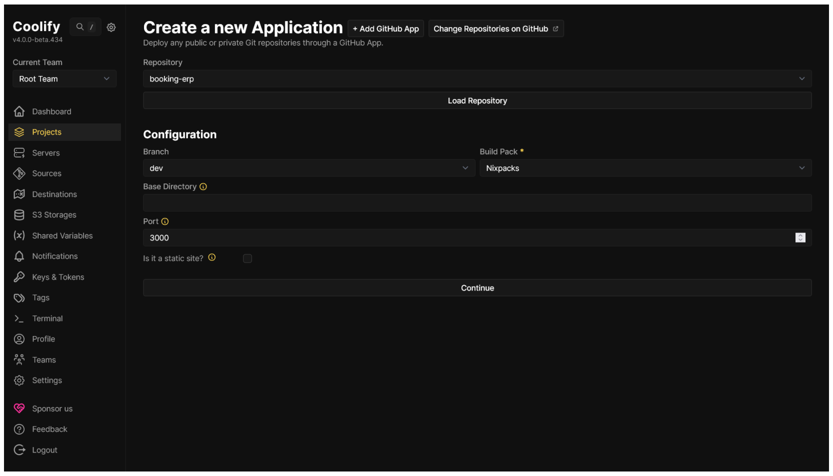
For production deployments, connecting to a Git repository is strongly recommended. This approach enables automatic deployments when code is pushed to the repository, making updates seamless. To connect a Git repository, you will need to create a GitHub App. Coolify provides straightforward instructions for this process during the application setup.

Regardless of the deployment method you choose, ensure you configure the following build settings:

- **Build method**: Select ”Nixpacks” as the build pack. Nixpacks automatically detects the Next.js application and configures the build process appropriately.

- **Port**: Set the application port to 3000, which is the default port for Next.js applications.

Figure C.8 shows the application creation screen in Coolify with these settings configured.



Create a new Application

Deploy any public or private Git repositories through a GitHub App.

Current Team: Root Team

Repository: booking-erp

Configuration

Branch: dev

Build Pack *: Nixpacks

Base Directory:

Port: 3000

Is it a static site? ☐

Figure C.8: Creating the Rhodes Adventures Application in Coolify

Configuring Environment Variables

Before deploying the application, you need to configure the environment variables that connect it to your database and third-party services. Navigate to the "Environment Variables" tab in your Coolify application settings.

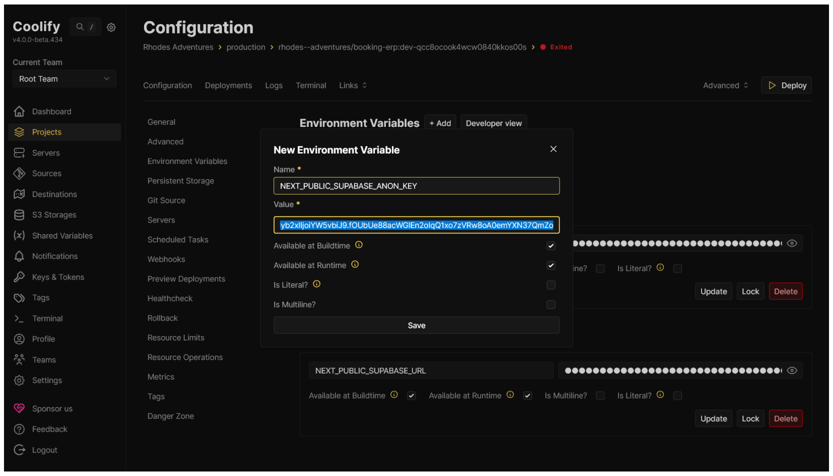
Add the following environment variables, using the values you collected during previous steps:

- `NEXT_PUBLIC_SUPABASE_URL`: Set this to the value of `SERVICE_URL_SUPABASE_KONG` from the Supabase environment variables
- `NEXT_PUBLIC_SUPABASE_ANON_KEY`: Set this to the value of `SERVICE_SUPABASEANON_KEY`
- `SUPABASE_SERVICE_ROLE_KEY`: Set this to the value of `SERVICE_SUPABASESERVICE_KEY`

Additionally, you need to provide API keys for the third-party services used by the application:

- `NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY`: Your Stripe publishable key for payment processing
- `STRIPE_SECRET_KEY`: Your Stripe secret key for server-side payment operations
- `RESEND_API_KEY`: Your Resend API key for sending transactional emails
- `GOOGLE_API_KEY`: Your Google API key for maps and route optimization

These API keys can be obtained from the respective service provider's dashboard. Figure C.9 shows the environment variables configuration screen in Coolify.



New Environment Variable

Name *: NEXT_PUBLIC_SUPABASE_ANON_KEY

Value *: yb2xIljoiYW5vbiJ9.fOUbUe88acWGlEn2olqQ1xo7zVRw8oA0emYXN37QmZo

Available at Buildtime ✓
Available at Runtime ✓
Is Literal? ☐
Is Multiline? ☐

Save

NEXT_PUBLIC_SUPABASE_URL ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●

Available at Buildtime ✓ Available at Runtime ✓ Is Multiline? ☐ Is Literal? ☐

Figure C.9: Configuring Application Environment Variables

Deploying and Verifying

With all environment variables configured, click the "Deploy" button to initiate the application deployment. Coolify will clone the repository (or upload your code), install dependencies, build the Next.js application, and start the server. The deployment process may take some time, so grab a cup of your favourite beverage and sit back.

Once deployment completes, open the URL shown by Coolify in your browser to verify that the deployment was successful. You should see the Rhodes Adventures landing page with options to make a booking or access the employee dashboard.

Test the deployment by logging in with the administrator credentials you created earlier. Verify that you can access the employee dashboard and that all features are working correctly. If you encounter any errors, check the Coolify logs for detailed error messages that can help diagnose configuration issues.

C.1.9 Post-Deployment Considerations

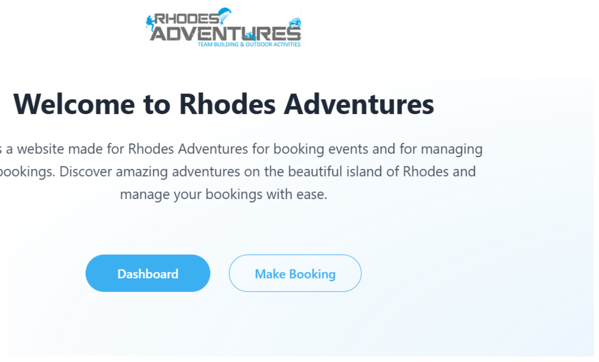
After successfully deploying the application, consider implementing the following additional steps for a production environment, as they are beyond the scope of this manual:

- **Custom domain**: Configure a custom domain name instead of using bare IP addresses or Coolify's IP-based URLs
- **SSL certificate**: Ensure HTTPS is properly configured with a valid SSL certificate (Coolify can automate this with Let's Encrypt)
- **Backups**: Configure automated database backups through Supabase or your hosting provider
- **Monitoring**: Set up application and database monitoring to track performance and errors
- **Email configuration**: Verify that transactional emails are being delivered correctly by testing the booking confirmation flow
- **Payment testing**: Thoroughly test the Stripe payment integration in test mode before switching to live mode

The application is now fully deployed and ready for use.

C.2 Using the Application

Once you open the website, the landing page gives you two options: the "Make Booking" and the "Dashboard" pages. The first one is supposed to be used by the clients, and the other one by the company employees.



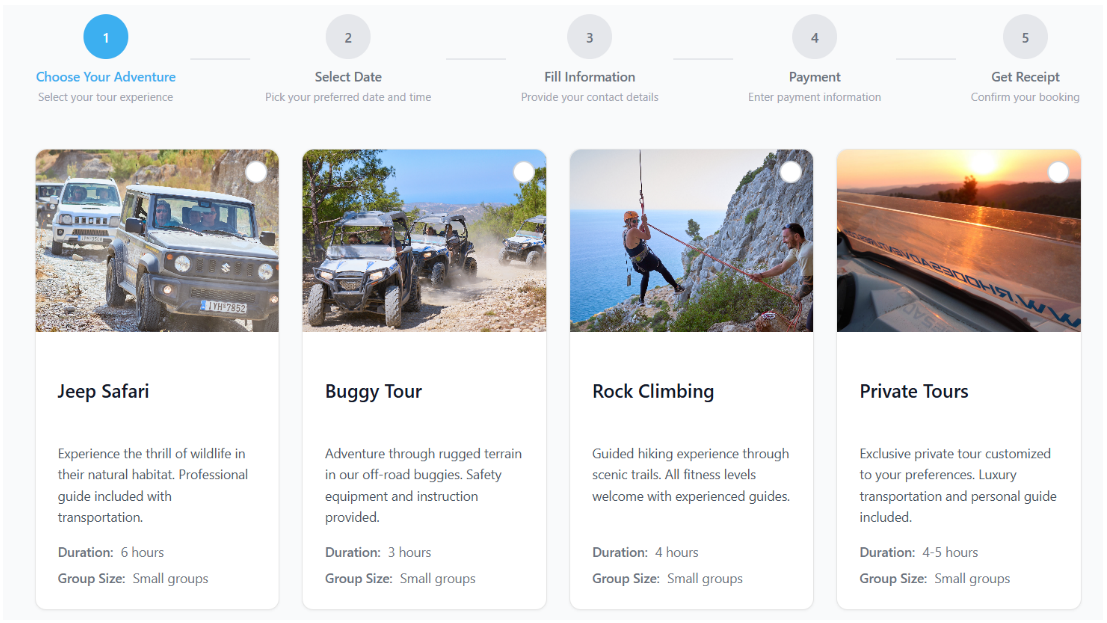
Welcome to Rhodes Adventures

This is a website made for Rhodes Adventures for booking events and for managing the bookings. Discover amazing adventures on the beautiful island of Rhodes and manage your bookings with ease.

Figure C.10: Landing Page

C.3 Make Booking Page

Once the client enters the page, they have to choose the activity that they want to do. The options are the following: Jeep Safari, Buggy Tour, Rock Climbing, and Private Tours. On that page, below each activity, the client can see a brief description of it as well as the duration of the activity. The booking process for these four activities is quite similar, with some small differences, which is why there is a separate subsection for each of them to be explained.


1
Choose Your Adventure
Select your tour experience
2
Select Date
Pick your preferred date and time
3
Fill Information
Provide your contact details
4
Payment
Enter payment information
5
Get Receipt
Confirm your booking

Jeep Safari

Experience the thrill of wildlife in their natural habitat. Professional guide included with transportation.

Duration: 6 hours

Group Size: Small groups

Buggy Tour

Adventure through rugged terrain in our off-road buggies. Safety equipment and instruction provided.

Duration: 3 hours

Group Size: Small groups

Rock Climbing

Guided hiking experience through scenic trails. All fitness levels welcome with experienced guides.

Duration: 4 hours

Group Size: Small groups

Private Tours

Exclusive private tour customized to your preferences. Luxury transportation and personal guide included.

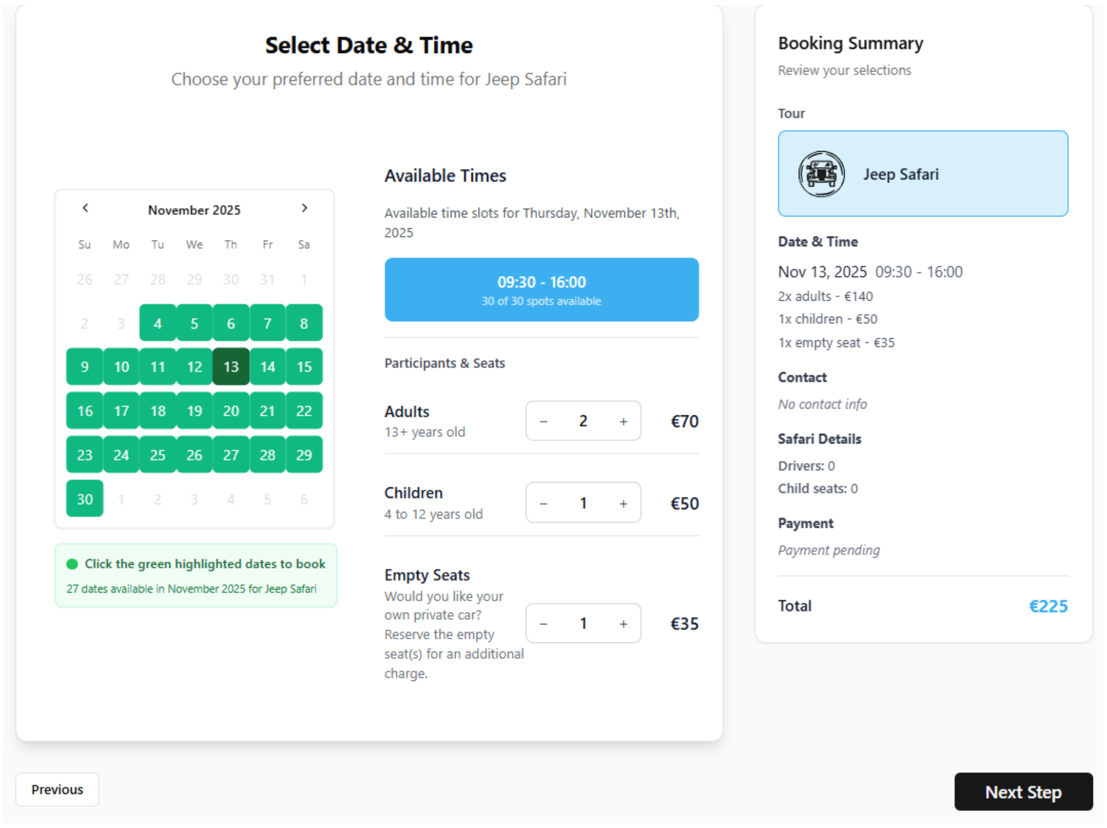
Duration: 4-5 hours

Group Size: Small groups

Figure C.11: Activities Page

C.3.1 Jeep Safari

Once the client is on the Jeep Safari activity page, they can choose the date and time of the event that they want to attend. Then, they are supposed to state how many adults and children will be part of the booking, as well as how many empty seats they want to have in the cars that they will be using, which are all paid. The children's age should range from 4 to 12 years old since these are the ages when they are required to use a child seat.



Select Date & Time

Choose your preferred date and time for Jeep Safari

November 2025

Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

● Click the green highlighted dates to book

27 dates available in November 2025 for Jeep Safari

Available Times

Available time slots for Thursday, November 13th, 2025

09:30 - 16:00
30 of 30 spots available

Participants & Seats

Adults 13+ years old	2	€70
Children 4 to 12 years old	1	€50
Empty Seats Would you like your own private car? Reserve the empty seat(s) for an additional charge.	1	€35

Booking Summary

Review your selections

Tour

Jeep Safari

Date & Time

Nov 13, 2025 09:30 - 16:00

2x adults - €140

1x children - €50

1x empty seat - €35

Contact

No contact info

Safari Details

Drivers: 0

Child seats: 0

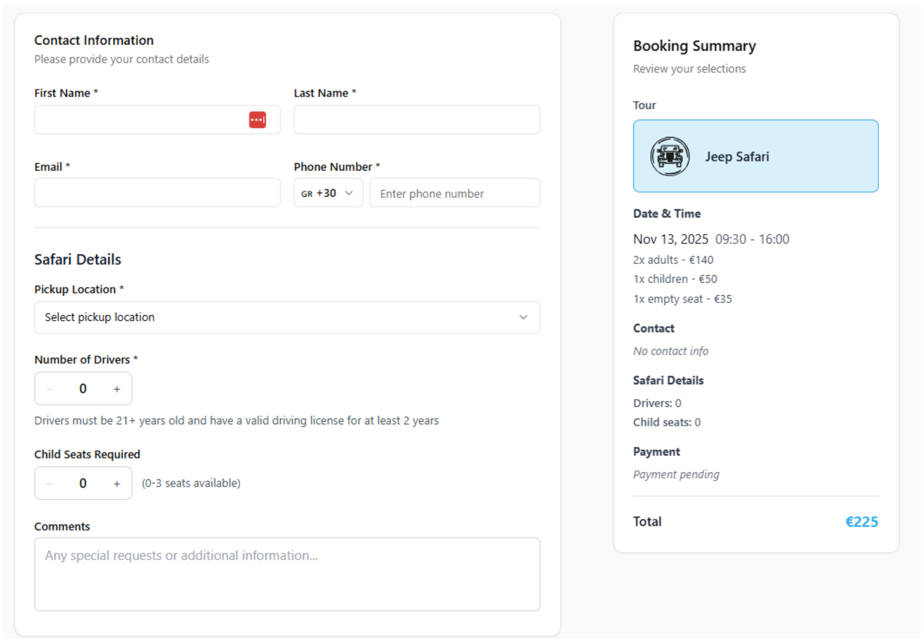
Payment

Payment pending

Total €225

Figure C.12: Select Date & Time Safari

The next step is filling in personal information like the name, email, and phone number of the person making the booking. In the safari, the clients are always picked up from their hotel, while in the buggy, the pickup is optional. So, they have to select their hotel from the dropdown menu of the available hotels on Rhodes, and after that, the number of people in the booking who can drive the car, in other words, someone 21 years old or older who has already had their driver's license for at least 2 years.


Contact Information
Please provide your contact details

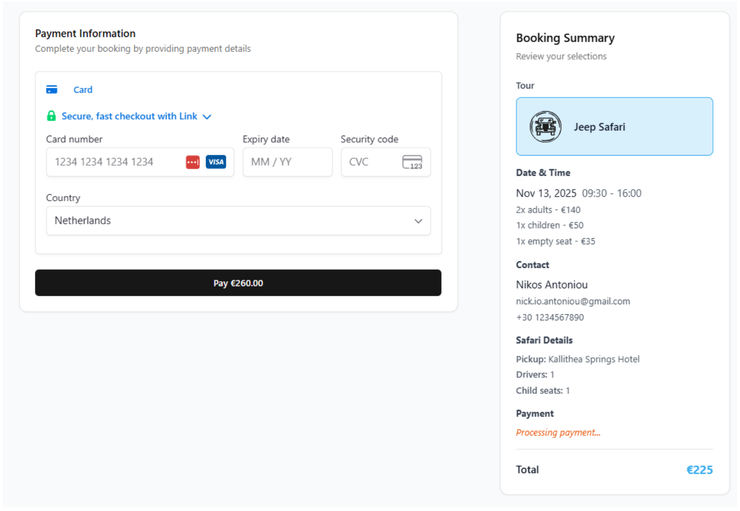
First Name *
Last Name *
Email *
Phone Number *
GR +30
Enter phone number

Safari Details
Pickup Location *
Select pickup location
Number of Drivers *
0
Drivers must be 21+ years old and have a valid driving license for at least 2 years
Child Seats Required
0
(0-3 seats available)
Comments
Any special requests or additional information...

Booking Summary
Review your selections
Tour
Jeep Safari
Date & Time
Nov 13, 2025 09:30 - 16:00
2x adults - €140
1x children - €50
1x empty seat - €35
Contact
No contact info
Safari Details
Drivers: 0
Child seats: 0
Payment
Payment pending
Total €225

Figure C.13: Contact and Pickup Information Safari

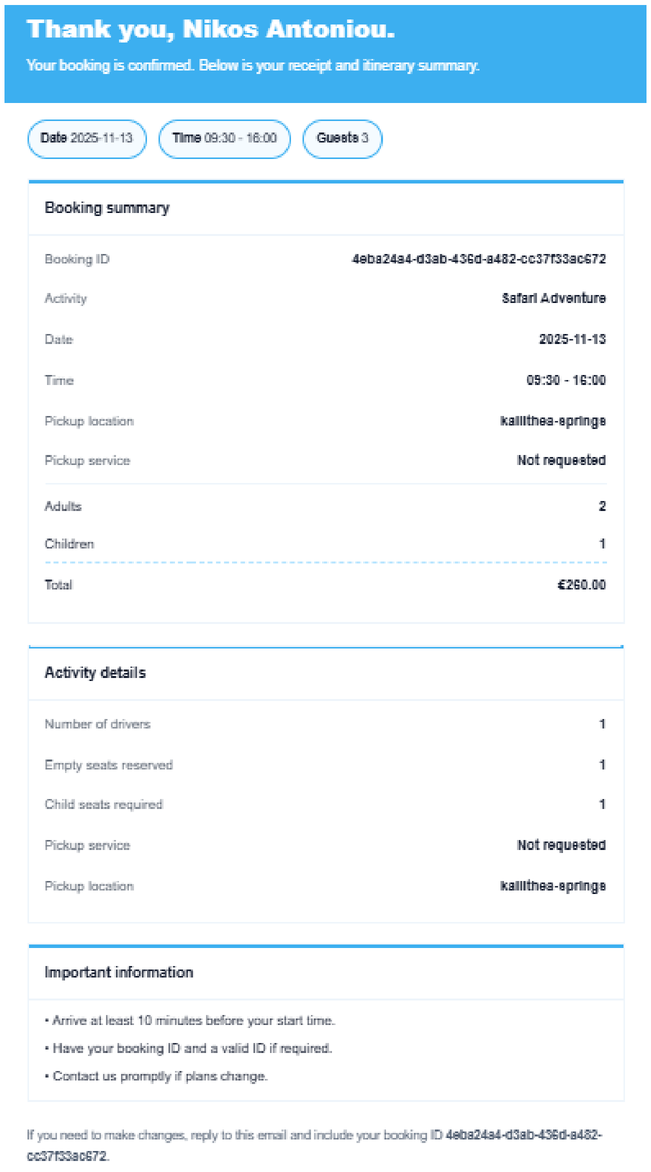
The next and final step is to fill in their debit card information and, optionally, fill in their email and phone number so that a confirmation email will be sent to them once the payment is successful.


Payment Information
Complete your booking by providing payment details
Card
Secure, fast checkout with Link
Card number
1234 1234 1234 1234
Expiry date
MM / YY
Security code
CVC
Country
Netherlands
Pay €260.00

Booking Summary
Review your selections
Tour
Jeep Safari
Date & Time
Nov 13, 2025 09:30 - 16:00
2x adults - €140
1x children - €50
1x empty seat - €35
Contact
Nikos Antoniou
nick.io.antoniou@gmail.com
+30 1234567890
Safari Details
Pickup: Kallithea Springs Hotel
Drivers: 1
Child seats: 1
Payment
Processing payment...
Total €225

Figure C.14: Safari Payment

After the booking is completed, the client is redirected to a page with an overview of the booking, with the same overview report having been sent to the client's email at the same time.



Thank you, Nikos Antoniou.

Your booking is confirmed. Below is your receipt and itinerary summary.

Date 2025-11-13 | **Time** 09:30 - 16:00 | **Guests** 3

Booking summary

Booking ID	4eba24a4-d3ab-436d-a482-cc37f33ac672
Activity	Safari Adventure
Date	2025-11-13
Time	09:30 - 16:00
Pickup location	kallithea-springs
Pickup service	Not requested
Adults	2
Children	1
Total	€260.00

Activity details

Number of drivers	1
Empty seats reserved	1
Child seats required	1
Pickup service	Not requested
Pickup location	kallithea-springs

Important information

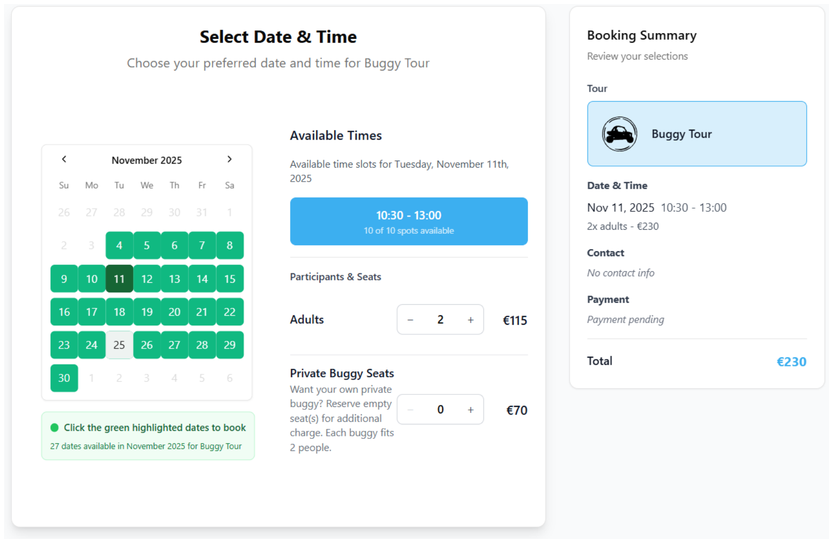
- Arrive at least 10 minutes before your start time.
- Have your booking ID and a valid ID if required.
- Contact us promptly if plans change.

If you need to make changes, reply to this email and include your booking ID 4eba24a4-d3ab-436d-a482-cc37f33ac672.

Figure C.15: Confirmation Email Safari

C.3.2 Buggy Tour

Once the client is on the Buggy Tour activity page, they can choose the date and time when they want to participate in the activity. After that, they are supposed to say how many adults will be members of this booking and whether they want to have one or more paid empty seats at the buggy or buggies that they will be using. Unlike the Jeep Safari activity, children younger than 13 years of age cannot participate in this activity.



Select Date & Time

Choose your preferred date and time for Buggy Tour

November 2025

Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

● Click the green highlighted dates to book

27 dates available in November 2025 for Buggy Tour

Available Times

Available time slots for Tuesday, November 11th, 2025

10:30 - 13:00
10 of 10 spots available

Participants & Seats

Adults − 2 + €115

Private Buggy Seats
Want your own private buggy? Reserve empty seat(s) for additional charge. Each buggy fits 2 people. − 0 + €70

Booking Summary

Review your selections

Tour

Buggy Tour

Date & Time

Nov 11, 2025 10:30 - 13:00

2x adults - €230

Contact

No contact info

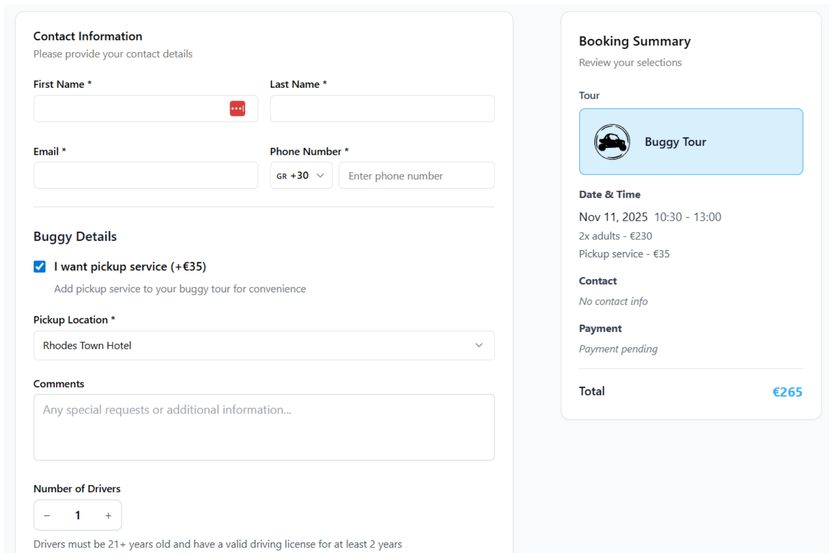
Payment

Payment pending

Total €230

Figure C.16: Select Date & Time Buggy

After that, the client is sent to the personal information page, where they are supposed to fill in personal information like their name, email, and phone number. The next step is to state whether the client wants to be picked up by the company from the hotel where they are staying at. There will be a dropdown menu for them to choose the hotel where they are staying on the island. They also have to state the number of drivers that will be in the booking. That person should be over 21 years of age, and they should already be in possession of their driver's license for 2 years.


Contact Information
Please provide your contact details

First Name *
Last Name *
Email *
Phone Number *
GR +30
Enter phone number

Buggy Details
✓ I want pickup service (+€35)
Add pickup service to your buggy tour for convenience

Pickup Location *
Rhodes Town Hotel

Comments
Any special requests or additional information...

Number of Drivers
– 1 +
Drivers must be 21+ years old and have a valid driving license for at least 2 years

Booking Summary
Review your selections

Tour
Buggy Tour

Date & Time
Nov 11, 2025 10:30 - 13:00
2x adults - €230
Pickup service - €35

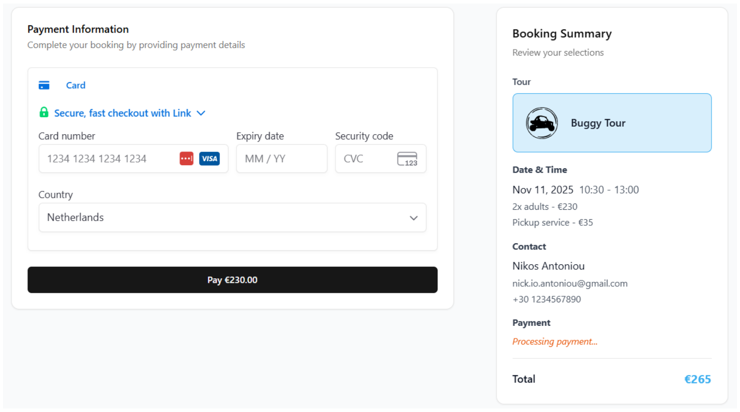
Contact
No contact info

Payment
Payment pending

Total €265

Figure C.17: Buggy Contact Information

After that, the client needs to fill in their debit card information so that they can pay. Stripe handles the payment after that information is filled. For faster checkout, the client can fill in their email or phone number again for a confirmation email to be sent to them after the payment is made.


Payment Information
Complete your booking by providing payment details

Card
Secure, fast checkout with Link

Card number
1234 1234 1234 1234
Expiry date
MM / YY
Security code
CVC

Country
Netherlands

Pay €230.00

Booking Summary
Review your selections

Tour
Buggy Tour

Date & Time
Nov 11, 2025 10:30 - 13:00
2x adults - €230
Pickup service - €35

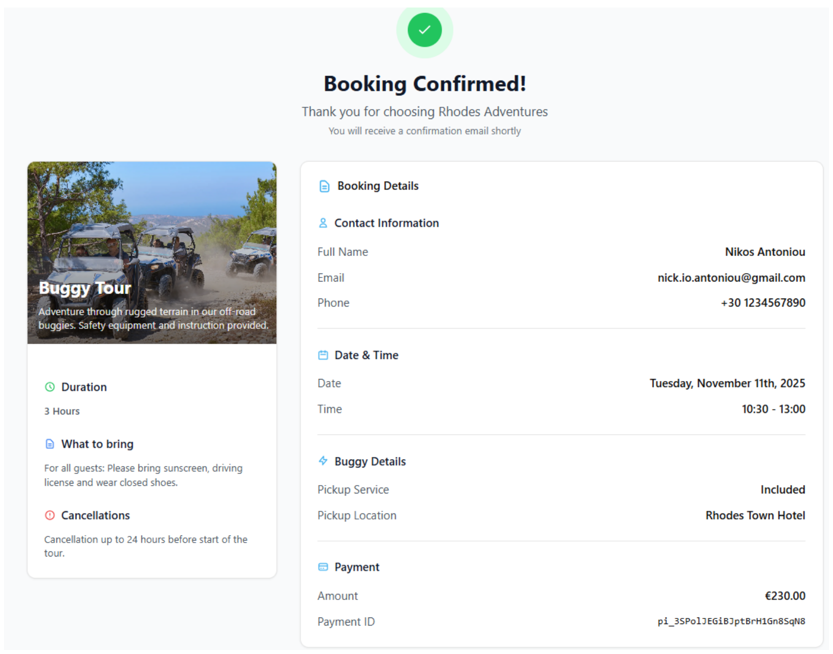
Contact
Nikos Antoniou
nick.io.antoniou@gmail.com
+30 1234567890

Payment
Processing payment...

Total €265

Figure C.18: Buggy Payment

If the payment was successful, the client is going to be redirected to a page with an overview of the booking, which can also be seen in the email received.



Booking Confirmed!

Thank you for choosing Rhodes Adventures

You will receive a confirmation email shortly

Buggy Tour

Adventure through rugged terrain in our off-road buggies. Safety equipment and instruction provided.

Duration

3 Hours

What to bring

For all guests: Please bring sunscreen, driving license and wear closed shoes.

Cancellations

Cancellation up to 24 hours before start of the tour.

Booking Details

Contact Information

Full Name	Nikos Antoniou
Email	nick.io.antoniou@gmail.com
Phone	+30 1234567890

Date & Time

Date	Tuesday, November 11th, 2025
Time	10:30 - 13:00

Buggy Details

Pickup Service	Included
Pickup Location	Rhodes Town Hotel

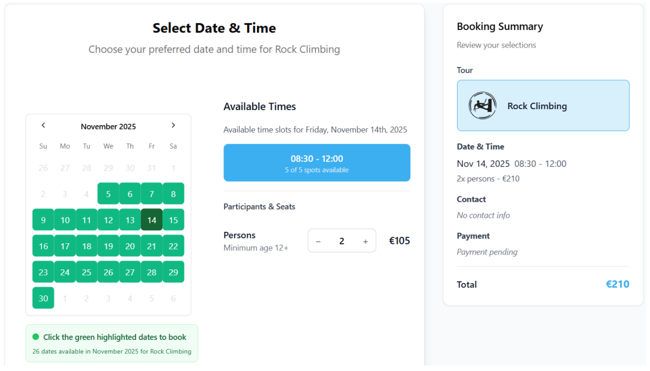
Payment

Amount	€230.00
Payment ID	pi_3SPolJEGiBJptBrH1Gn8SqN8

Figure C.19: Buggy Confirmation Page

C.3.3 Rock Climbing

Once the client selects the Rock Climbing activity, they will need to do the same process to select the date and time when they want to participate in the activity, as well as how many people are part of the booking. The available spots for this activity are a lot fewer, with a maximum of five people participating per activity.



Select Date & Time

Choose your preferred date and time for Rock Climbing

November 2025

Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

● Click the green highlighted dates to book

26 dates available in November 2025 for Rock Climbing

Available Times

Available time slots for Friday, November 14th, 2025

08:30 - 12:00
5 of 5 spots available

Participants & Seats

Persons
Minimum age 12+ − 2 + €105

Booking Summary

Review your selections

Tour

Rock Climbing

Date & Time

Nov 14, 2025 08:30 - 12:00

2x persons - €210

Contact

No contact info

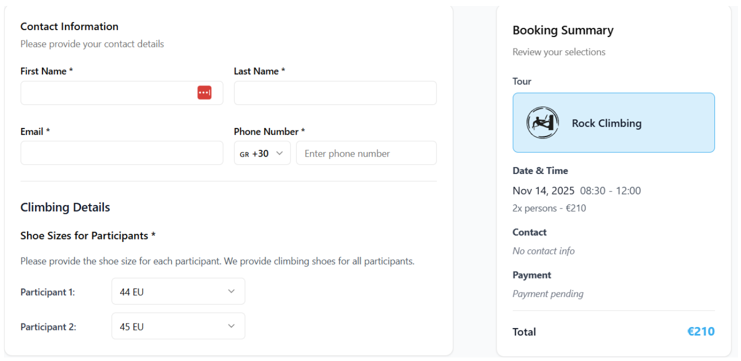
Payment

Payment pending

Total €210

Figure C.20: Select Date & Time Rock Climbing

The next step will also be to fill in the personal information of the person making the booking, which includes that person's name, email, and phone number. Unlike the previous two activities, Rock Climbing and Private Tours do not offer pickup services. There is a fixed meeting location for these activities, which is sent through email to the customers. However, the client has to fill in the shoe size of all members of the booking, since they need specific kinds of shoes to perform the rock climbing, which are provided by the company.


Contact Information
Please provide your contact details
First Name *
Last Name *
Email *
Phone Number *
GR +30
Enter phone number

Climbing Details
Shoe Sizes for Participants *
Please provide the shoe size for each participant. We provide climbing shoes for all participants.
Participant 1: 44 EU
Participant 2: 45 EU

Booking Summary
Review your selections
Tour
Rock Climbing
Date & Time
Nov 14, 2025 08:30 - 12:00
2x persons - €210
Contact
No contact info
Payment
Payment pending
Total €210

Figure C.21: Contact Information Rock CLimbing

The next step is once again the payment process, where the client needs to fill in their debit card information, their email, and phone number, and after the booking is completed, the client will be redirected to a page with an overview of the activity, and the same thing will be sent to them as a confirmation email, where the meeting location of the activity will be available. The meeting location is a link to Google Maps, with the exact coordinates of the meeting location.

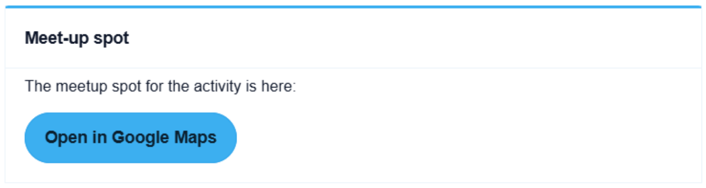
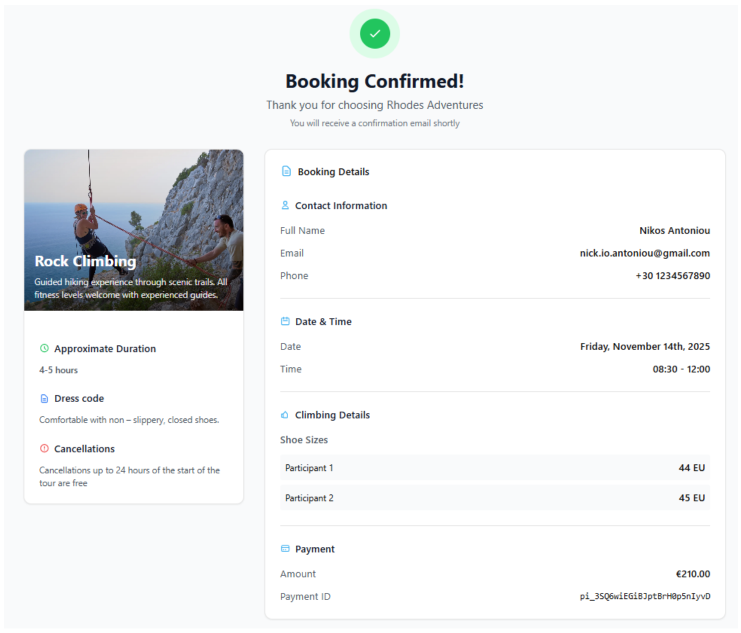

Meet-up spot
The meetup spot for the activity is here:

Figure C.22: Rock Climbing Meeting Location



Booking Confirmed!

Thank you for choosing Rhodes Adventures

You will receive a confirmation email shortly

Rock Climbing

Guided hiking experience through scenic trails. All fitness levels welcome with experienced guides.

Approximate Duration

4-5 hours

Dress code

Comfortable with non – slippery, closed shoes.

Cancellations

Cancellations up to 24 hours of the start of the tour are free

Booking Details

Contact Information

Full Name	Nikos Antoniou
Email	nick.io.antoniou@gmail.com
Phone	+30 1234567890

Date & Time

Date	Friday, November 14th, 2025
Time	08:30 - 12:00

Climbing Details

Shoe Sizes

Participant 1	44 EU
Participant 2	45 EU

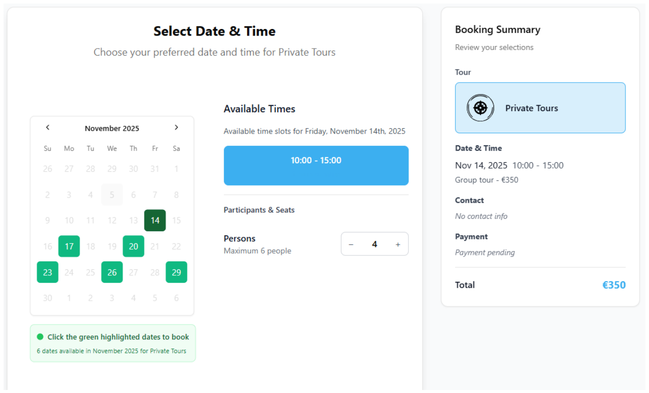
Payment

Amount	€210.00
Payment ID	pi_3SQ6wiEGiBJptBrH0p5nIyvD

Figure C.23: Rock Climbing Confirmation Page

C.3.4 Private Tours

The first step is once again to select the date and time for the event as well as the number of people in the booking.



Select Date & Time

Choose your preferred date and time for Private Tours

November 2025

Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

● Click the green highlighted dates to book

6 dates available in November 2025 for Private Tours

Available Times

Available time slots for Friday, November 14th, 2025

10:00 - 15:00

Participants & Seats

Persons

Maximum 6 people

− 4 +

Booking Summary

Review your selections

Tour

Private Tours

Date & Time

Nov 14, 2025 10:00 - 15:00

Group tour - €350

Contact

No contact info

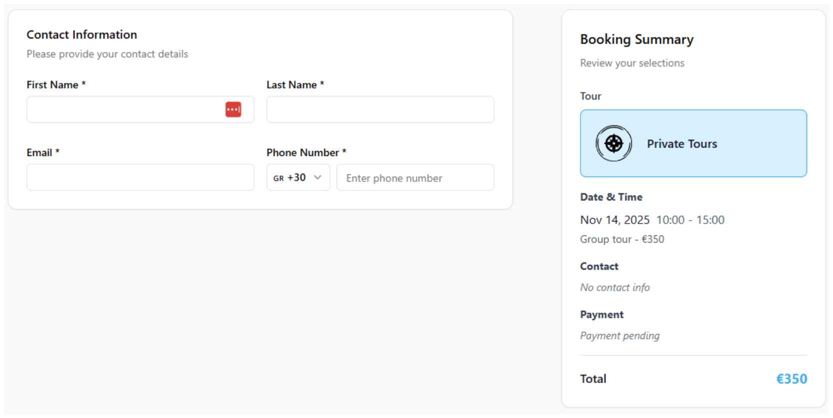
Payment

Payment pending

Total	€350

Figure C.24: Private Tours Date & Time

The next step is once again to fill in the contact information of the person that makes the booking, so the name, email, and phone number of that person.


Contact Information

Please provide your contact details

First Name *	Last Name *
Email *	Phone Number *
	GR +30 Enter phone number

Booking Summary

Review your selections

Tour

Private Tours

Date & Time

Nov 14, 2025 10:00 - 15:00

Group tour - €350

Contact

No contact info

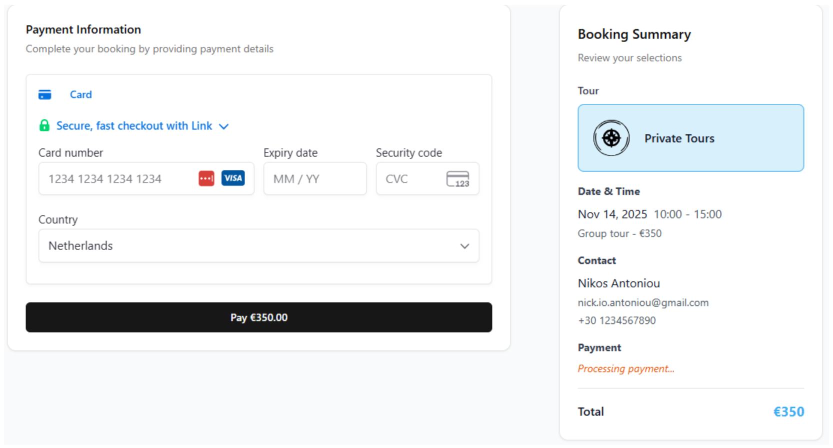
Payment

Payment pending

Total €350

Figure C.25: Private Tours Contact Information

The final step is again to pay for the activity with the same way as in the previous activities.


Payment Information

Complete your booking by providing payment details

Card

Secure, fast checkout with Link

Card number	Expiry date	Security code
1234 1234 1234 1234	MM / YY	CVC

Country

Netherlands

Booking Summary

Review your selections

Tour

Private Tours

Date & Time

Nov 14, 2025 10:00 - 15:00

Group tour - €350

Contact

Nikos Antoniou

nick.io.antoniou@gmail.com

+30 1234567890

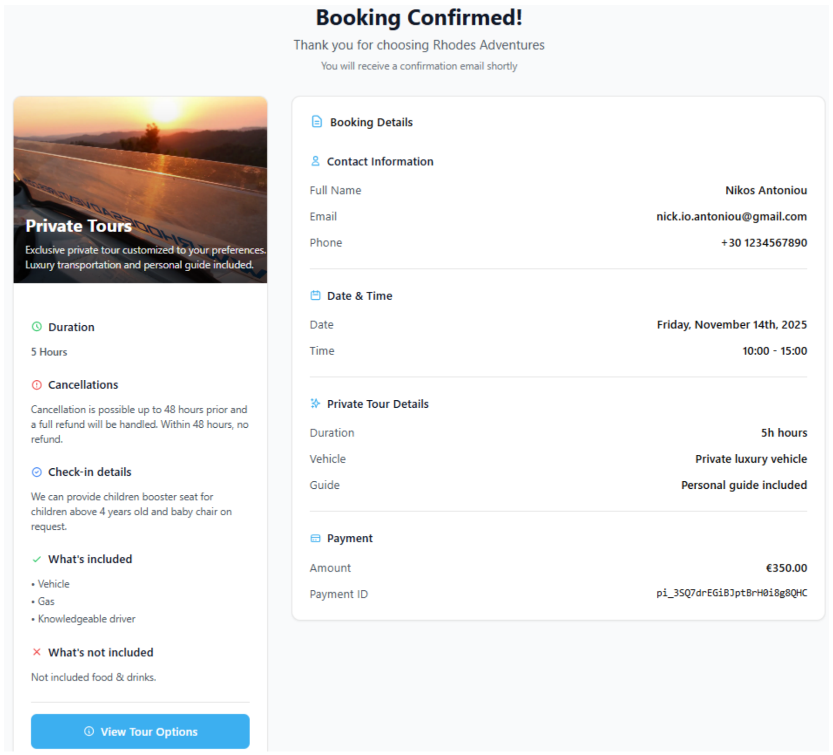
Payment

Processing payment...

Total €350

Figure C.26: Private Tours Payment

The confirmation page differs a bit from the other activities in the sense that it provides information about what is or is not included in the tour. The page also provides information about the different kinds of tours when clicking the "View Tour Options" button.



Booking Confirmed!

Thank you for choosing Rhodes Adventures

You will receive a confirmation email shortly

Private Tours

Exclusive private tour customized to your preferences. Luxury transportation and personal guide included.

Duration

5 Hours

Cancellations

Cancellation is possible up to 48 hours prior and a full refund will be handled. Within 48 hours, no refund.

Check-in details

We can provide children booster seat for children above 4 years old and baby chair on request.

What's included

• Vehicle
• Gas
• Knowledgeable driver

What's not included

Not included food & drinks.

Booking Details

Contact Information

Full Name	Nikos Antoniou
Email	nick.io.antoniou@gmail.com
Phone	+30 1234567890

Date & Time

Date	Friday, November 14th, 2025
Time	10:00 - 15:00

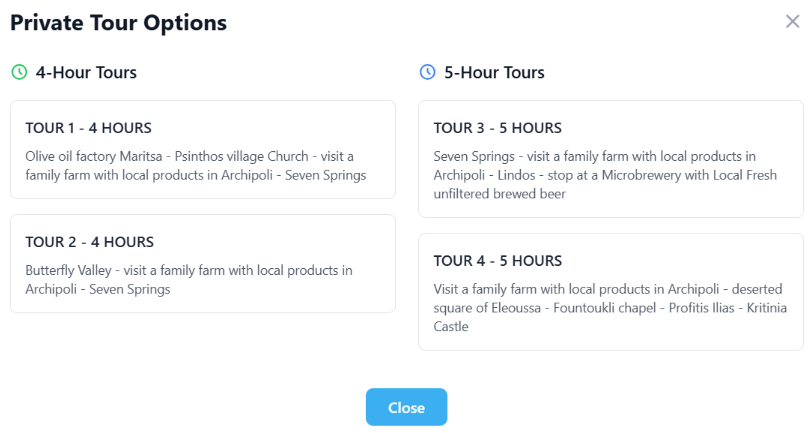
Private Tour Details

Duration	5h hours
Vehicle	Private luxury vehicle
Guide	Personal guide included

Payment

Amount	€350.00
Payment ID	pi_3SQ7drEGiBJptBrH0i8g8QHC

Figure C.27: Private Tours Confirmation Page



Private Tour Options

4-Hour Tours

TOUR 1 - 4 HOURS

Olive oil factory Maritsa - Psinthos village Church - visit a family farm with local products in Archipoli - Seven Springs

TOUR 2 - 4 HOURS

Butterfly Valley - visit a family farm with local products in Archipoli - Seven Springs

5-Hour Tours

TOUR 3 - 5 HOURS

Seven Springs - visit a family farm with local products in Archipoli - Lindos - stop at a Microbrewery with Local Fresh unfiltered brewed beer

TOUR 4 - 5 HOURS

Visit a family farm with local products in Archipoli - deserted square of Eleoussa - Fountoukli chapel - Profitis Ilias - Kritinia Castle

Figure C.28: Tour Options

C.4 Employee Dashboard

This part of the website is specifically for the employees. There is no option for a client to create an account as an employee of the company.

C.4.1 User Management

All the employees' accounts that have access to the Employee Dashboard are created by other company employees through the website. This is something that happens at the User Management Page.

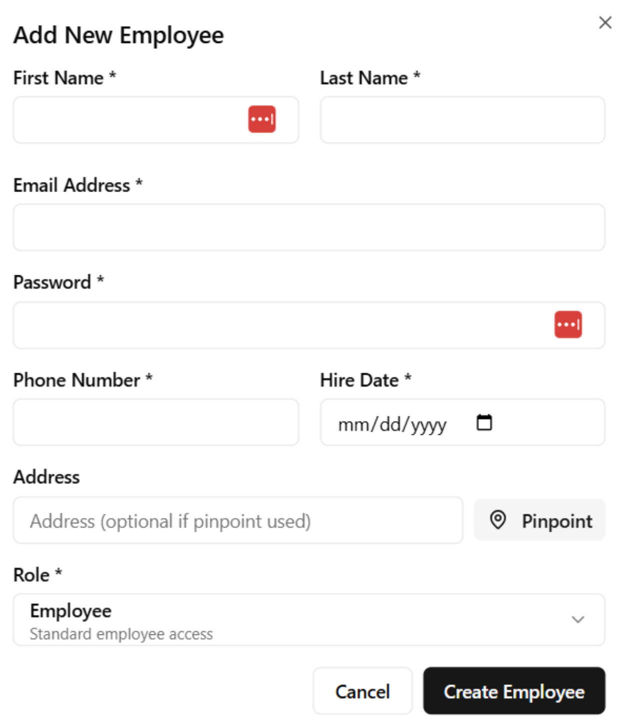
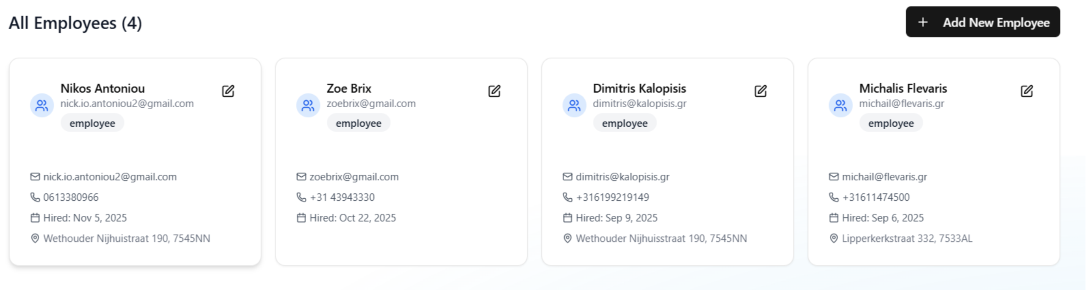


Figure C.29: Create Employee

On that page, any employee can see the information of the other company employees. Any employee can create an account for the new employees where they have to fill in their name, email, phone number, home address, and set their password to be able to log in to the website as an employee.



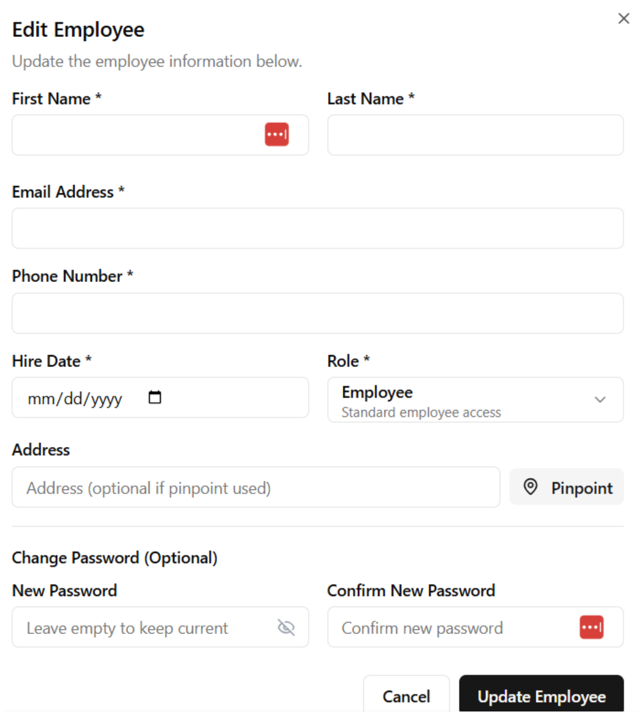
All Employees (4)

Add New Employee

Nikos Antoniou	Zoe Brix	Dimitris Kalopisis	Michalis Flevaris
nick.io.antoniou2@gmail.com	zoebrix@gmail.com	dimitris@kalopisis.gr	michail@flevaris.gr
employee	employee	employee	employee
nick.io.antoniou2@gmail.com	zoebrix@gmail.com	dimitris@kalopisis.gr	michail@flevaris.gr
0613380966	+31 43943330	+316199219149	+31611474500
Hired: Nov 5, 2025	Hired: Oct 22, 2025	Hired: Sep 9, 2025	Hired: Sep 6, 2025
Wethouder Nijhuistraat 190, 7545NN		Wethouder Nijhuisstraat 190, 7545NN	Lipperkerkstraat 332, 7533AL

Figure C.30: User Management Page

The website manager is the only one who can update an employee's information. If an employee tries to do so, they get a "Forbidden" error message on their screen.



Edit Employee

Update the employee information below.

First Name *

Last Name *

Email Address *

Phone Number *

Hire Date *: mm/dd/yyyy

Role *: Employee — Standard employee access

Address: Address (optional if pinpoint used)

Pinpoint

Change Password (Optional)

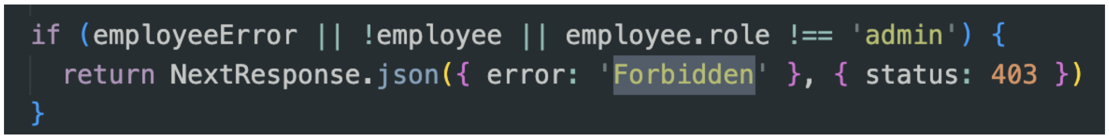
New Password: Leave empty to keep current

Confirm New Password: Confirm new password

Cancel

Update Employee

Figure C.31: Edit Employee Information

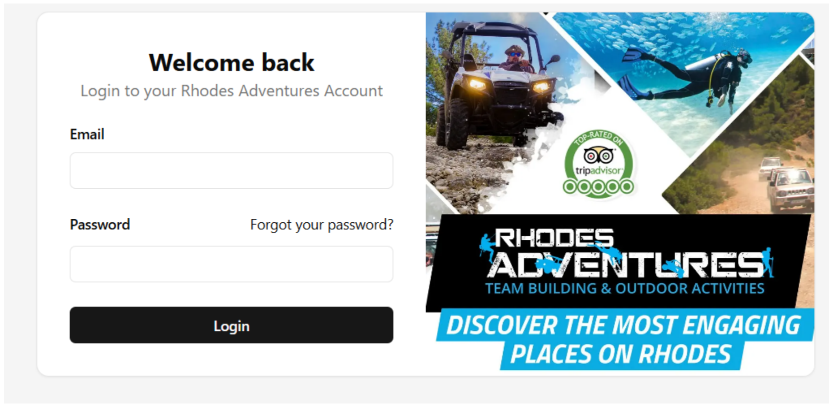


```
if (employeeError || !employee || employee.role !== 'admin') {
  return NextResponse.json({ error: 'Forbidden' }, { status: 403 })
}
```

Figure C.32: Edit Employee Error Message

C.4.2 Authentication

Since the employee has an account, they can log in to the website by filling in their email and password after selecting the dashboard page.



Welcome back

Login to your Rhodes Adventures Account

Email

Password

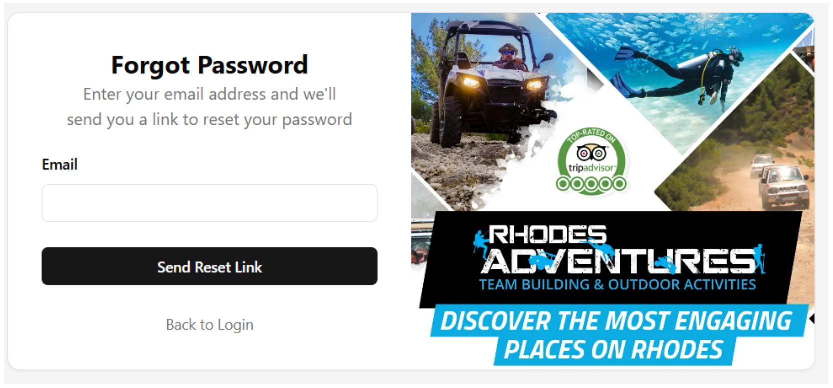
Forgot your password?

Login

RHODES
ADVENTURES
TEAM BUILDING & OUTDOOR ACTIVITIES
DISCOVER THE MOST ENGAGING PLACES ON RHODES
tripadvisor

Figure C.33: Login Page

If the employee has forgotten their password, they can reset it by clicking ”Forgot your Password?” The employee is then sent a password reset link.



Forgot Password

Enter your email address and we'll send you a link to reset your password

Email

Send Reset Link

Back to Login

RHODES
ADVENTURES
TEAM BUILDING & OUTDOOR ACTIVITIES
DISCOVER THE MOST ENGAGING PLACES ON RHODES
tripadvisor

Figure C.34: Reset Password

C.4.3 Home Page

This is where the employees are directed after logging in to the website. From that page, the employee can see some statistics about the events on that day, for example, the number of events and the number of participants in all these events in total.

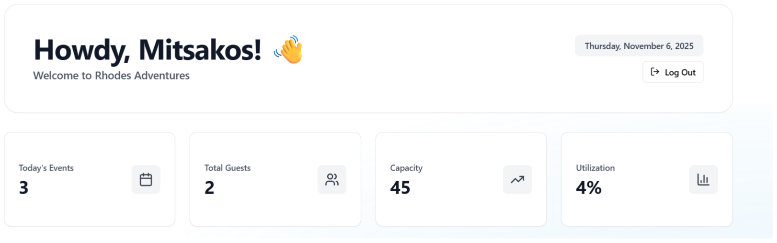


Figure C.35: Home Page Statistics

Right below, there are various options for quick actions that the employee can select. For example, they can select to book an activity for themselves as a customer, which will lead them to the booking page that was previously explained. Other options include the rest of the pages of the Employee Dashboard, which will be explained below.

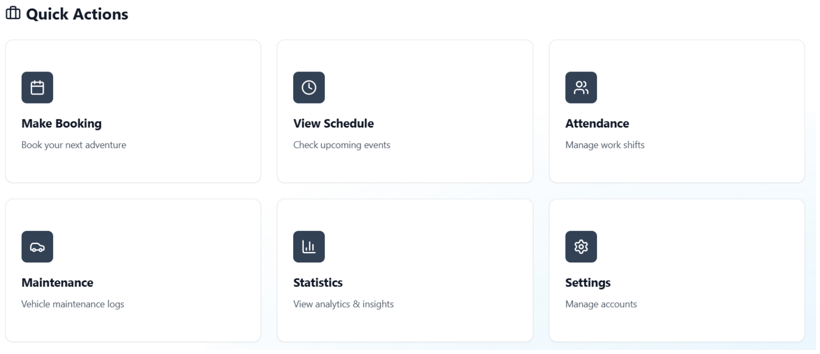
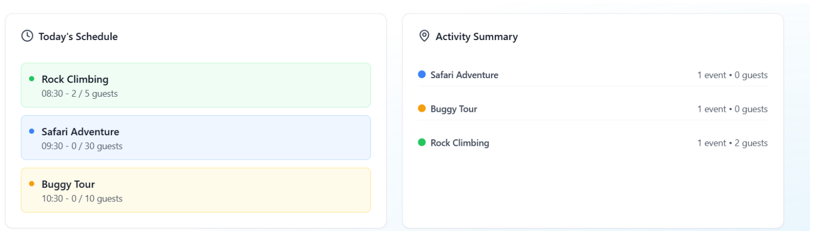


Figure C.36: Quick Actions

Below that, there is the schedule for the activities of that day. and next to it, a quick summary of those activities.



Today's Schedule

- Rock Climbing 08:30 - 2 / 5 guests
- Safari Adventure 09:30 - 0 / 30 guests
- Buggy Tour 10:30 - 0 / 10 guests

Activity Summary

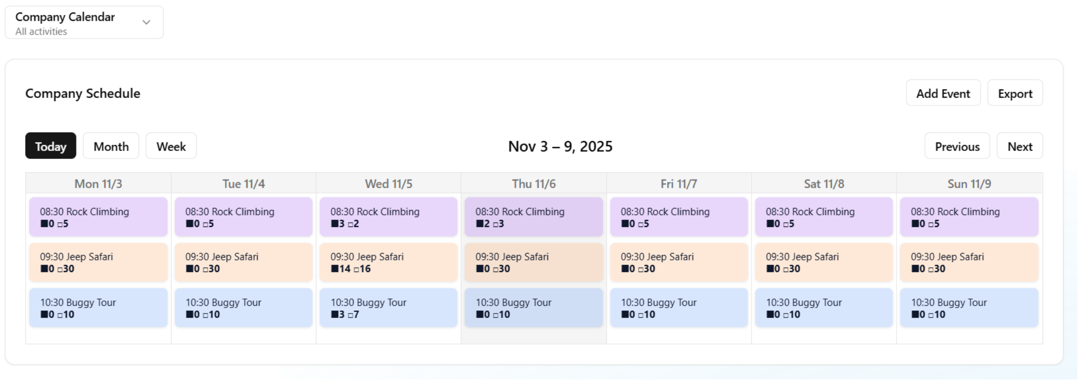
Safari Adventure	1 event • 0 guests
Buggy Tour	1 event • 0 guests
Rock Climbing	1 event • 2 guests

Figure C.37: Schedule and Activity Summaries

On the top right of the landing page, the employee can also log out of the website. If they do not do that, whenever they click the Dashboard button at the landing page of the website, they will be redirected to the home page of the dashboard without having to log in.

C.4.4 Schedule

On the Schedule page, the employees can see all the events that have been created, as well as how many people have booked to attend these events.



Company Calendar
All activities

Company Schedule

Nov 3 – 9, 2025

Mon 11/3	Tue 11/4	Wed 11/5	Thu 11/6	Fri 11/7	Sat 11/8	Sun 11/9
08:30 Rock Climbing ■0 □5	08:30 Rock Climbing ■0 □5	08:30 Rock Climbing ■3 □2	08:30 Rock Climbing ■2 □3	08:30 Rock Climbing ■0 □5	08:30 Rock Climbing ■0 □5	08:30 Rock Climbing ■0 □5
09:30 Jeep Safari ■0 □30	09:30 Jeep Safari ■0 □30	09:30 Jeep Safari ■14 □16	09:30 Jeep Safari ■0 □30	09:30 Jeep Safari ■0 □30	09:30 Jeep Safari ■0 □30	09:30 Jeep Safari ■0 □30
10:30 Buggy Tour ■0 □10	10:30 Buggy Tour ■0 □10	10:30 Buggy Tour ■3 □7	10:30 Buggy Tour ■0 □10	10:30 Buggy Tour ■0 □10	10:30 Buggy Tour ■0 □10	10:30 Buggy Tour ■0 □10

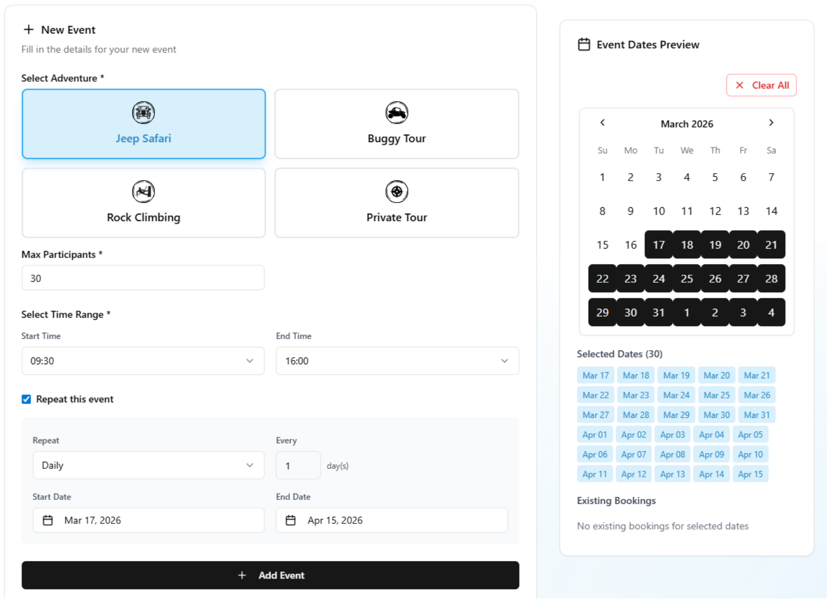
Figure C.38: Schedule Page

By clicking the "Add Event" button, they can create a new event, and by clicking on any of the activities, they can go to the manifest page of that event.

Create Activity

To create an activity, they must either press the "Add Event" button on the schedule or go to the page through the sidebar.

First, they need to select the type of activity and the date, then fill in the maximum number of participants, and select a time range for the activity, as long as there is not the same activity type on that date and time. They can also select to repeat the same event for a specific number of days that they can select on the page.


New Event
Fill in the details for your new event

Select Adventure *

Jeep Safari | Buggy Tour | Rock Climbing | Private Tour

Max Participants *
30

Select Time Range *
Start Time: 09:30
End Time: 16:00

☑ Repeat this event

Repeat: Daily
Every: 1 day(s)
Start Date: Mar 17, 2026
End Date: Apr 15, 2026

Add Event

Event Dates Preview

Clear All

March 2026

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Selected Dates (30)

Mar 17, Mar 18, Mar 19, Mar 20, Mar 21, Mar 22, Mar 23, Mar 24, Mar 25, Mar 26, Mar 27, Mar 28, Mar 29, Mar 30, Mar 31, Apr 01, Apr 02, Apr 03, Apr 04, Apr 05, Apr 06, Apr 07, Apr 08, Apr 09, Apr 10, Apr 11, Apr 12, Apr 13, Apr 14, Apr 15

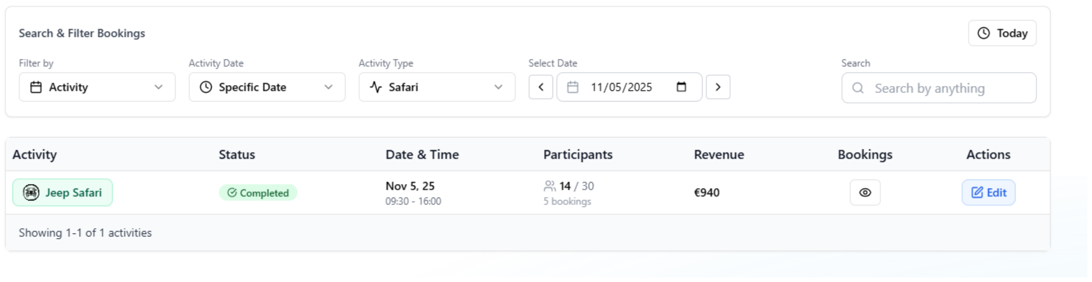
Existing Bookings

No existing bookings for selected dates

Figure C.39: Create Event

Manifest Page

As said in the schedule section, when the employee clicks at any activity, they are redirected to the manifest page of that activity. There they can view more information about the activity, like the type of activity, the number of participants, the revenue, and information about all bookings done for this activity. They can also update the information about the activity through the "Edit" button, where they can edit the time of the activity, the number of participants, or prevent this activity from being booked by people in the future by locking it.



Search & Filter Bookings

Today

Filter by: Activity

Activity Date: Specific Date

Activity Type: Safari

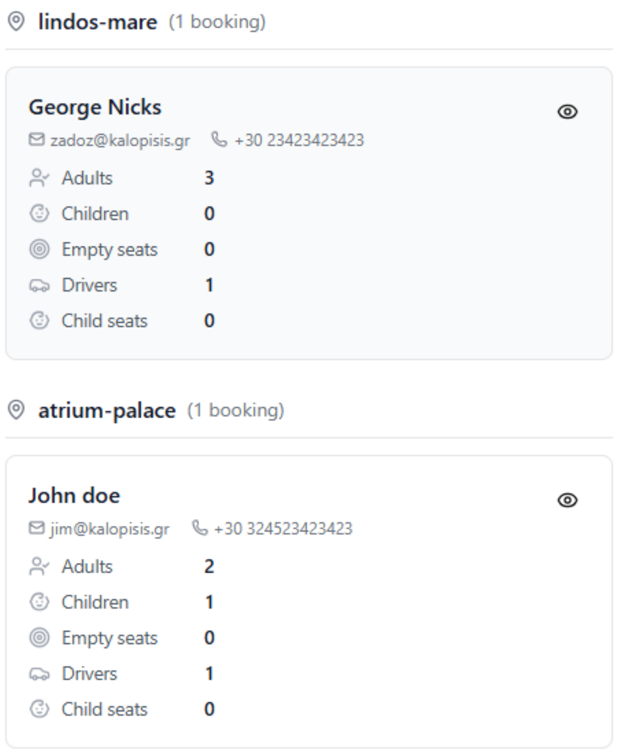
Select Date: 11/05/2025

Search: Search by anything

Activity	Status	Date & Time	Participants	Revenue	Bookings	Actions
Jeep Safari	Completed	Nov 5, 25 09:30 - 16:00	14 / 30 5 bookings	€940		Edit

Showing 1-1 of 1 activities

Figure C.40: Activity Manifest Page



lindos-mare (1 booking)

George Nicks

zadoz@kalopisis.gr +30 23423423423

Adults	3
Children	0
Empty seats	0
Drivers	1
Child seats	0

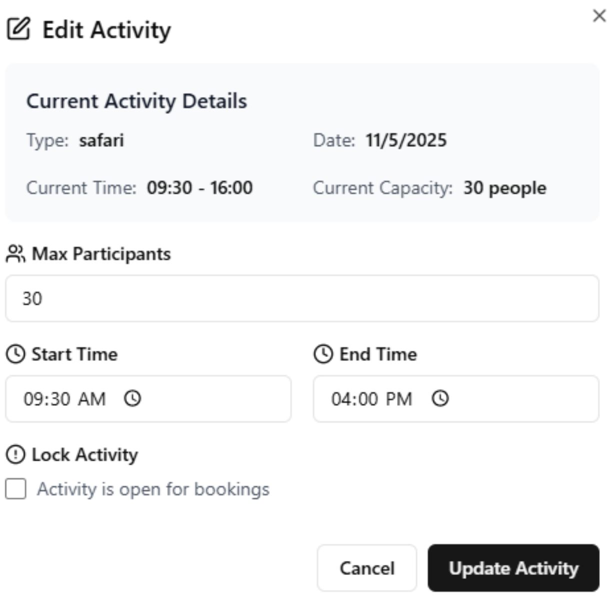
atrium-palace (1 booking)

John doe

jim@kalopisis.gr +30 324523423423

Adults	2
Children	1
Empty seats	0
Drivers	1
Child seats	0

Figure C.41: View Bookings



Edit Activity

Current Activity Details

Type: safari

Date: 11/5/2025

Current Time: 09:30 - 16:00

Current Capacity: 30 people

Max Participants: 30

Start Time: 09:30 AM

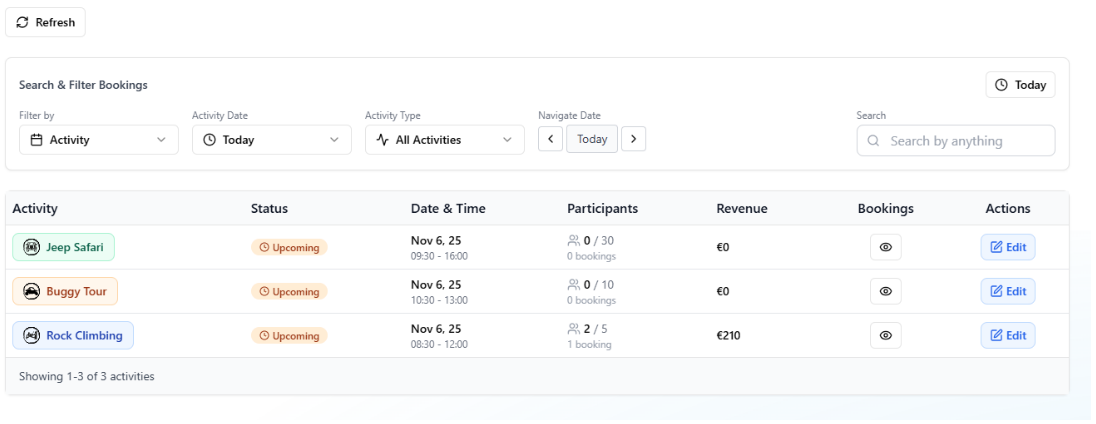
End Time: 04:00 PM

Lock Activity

Activity is open for bookings

Figure C.42: Edit Activity

In general, they can view the information about any activity. By default, when the page opens, they can see the activities scheduled for that day, but they can filter for different activities or dates to view the information about them.



Search & Filter Bookings

Filter by: Activity

Activity Date: Today

Activity Type: All Activities

Navigate Date: Today

Search: Search by anything

Activity	Status	Date & Time	Participants	Revenue	Bookings	Actions
Jeep Safari	Upcoming	Nov 6, 25 09:30 - 16:00	0 / 30 0 bookings	€0		Edit
Buggy Tour	Upcoming	Nov 6, 25 10:30 - 13:00	0 / 10 0 bookings	€0		Edit
Rock Climbing	Upcoming	Nov 6, 25 08:30 - 12:00	2 / 5 1 booking	€210		Edit

Showing 1-3 of 3 activities

Figure C.43: Manifest Page

Finalize Activities

To finalize an activity, there are some specific steps that need to be followed. First of all, the employee needs to select an activity, then they need to assign the employees for the activity, which can be done by manually assigning employees to work for that activity.

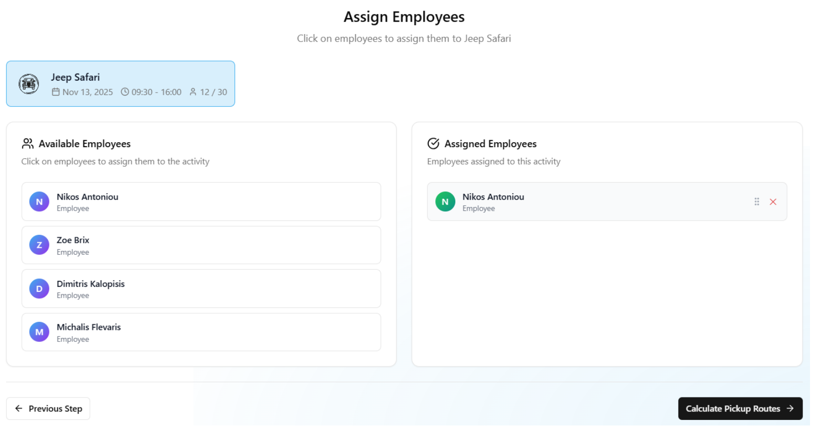


Figure C.44: Assign Employees

After the employees are assigned, the pickup algorithm discussed in Chapter 6.2 will calculate the optimal route for the assigned employees to pick up the customers from their hotels. After that is done, the route they should take can be seen through Google Maps, for which an API is used.

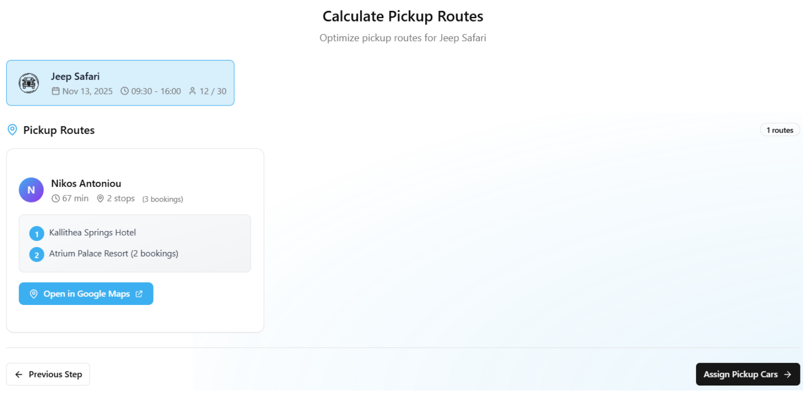
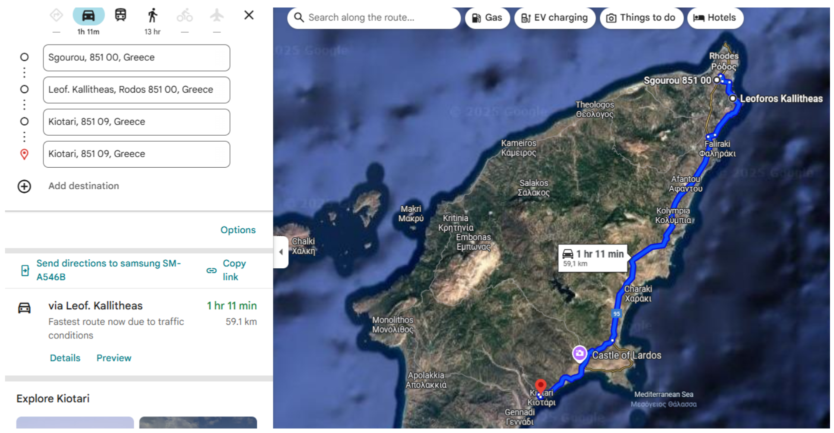

Calculate Pickup Routes
Optimize pickup routes for Jeep Safari
Jeep Safari
Nov 13, 2025 09:30 - 16:00 12 / 30
Pickup Routes
1 routes
Nikos Antoniou
67 min 2 stops (3 bookings)
1 Kallithea Springs Hotel
2 Atrium Palace Resort (2 bookings)
Open in Google Maps
Previous Step
Assign Pickup Cars

Figure C.45: Calculate Pickups


1h 11m

13 hr

Sgourou, 851 00, Greece

Leof. Kallitheas, Rodos 851 00, Greece

Kiotari, 851 09, Greece

Kiotari, 851 09, Greece

Add destination

Options

Send directions to samsung SM-A546B

Copy link

via Leof. Kallitheas — 1 hr 11 min, 59.1 km

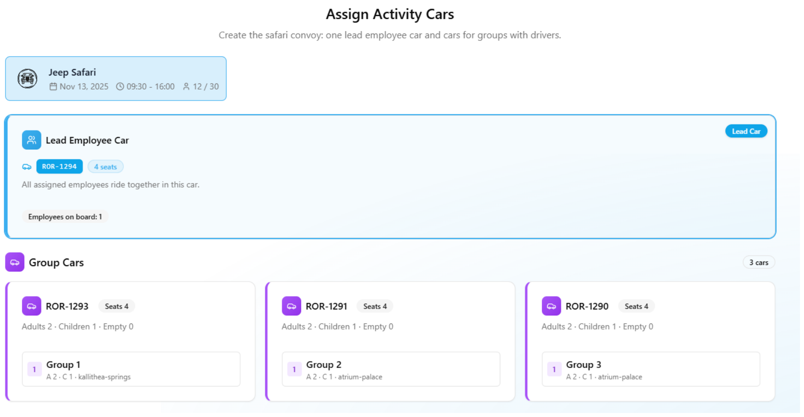
Fastest route now due to traffic conditions

Details Preview

Explore Kiotari

Figure C.46: Pickup Route

The next step is the car allocation, where the other algorithm discussed in chapter 6.1 is used. The algorithm will allocate the people of all bookings into different cars according to the mentioned constraints. And the same thing will be done to allocate the cars for the activity itself.


Assign Activity Cars

Create the safari convoy: one lead employee car and cars for groups with drivers.

Jeep Safari
Nov 13, 2025 · 09:30 - 16:00 · 12 / 30

Lead Employee Car — Lead Car
ROR-1294 · 4 seats
All assigned employees ride together in this car.
Employees on board: 1

Group Cars — 3 cars

ROR-1293 · Seats 4
Adults 2 · Children 1 · Empty 0
1 Group 1 — A 2 · C 1 · kallithea-springs

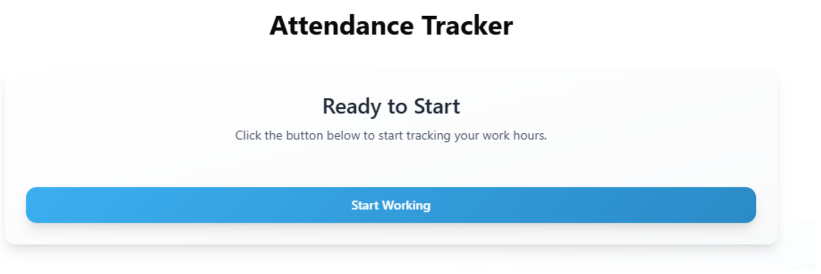
ROR-1291 · Seats 4
Adults 2 · Children 1 · Empty 0
1 Group 2 — A 2 · C 1 · atrium-palace

ROR-1290 · Seats 4
Adults 2 · Children 1 · Empty 0
1 Group 3 — A 2 · C 1 · atrium-palace

Figure C.47: Car Allocation

C.4.5 Attendance

The Attendance Page is used by all employees to track the number of working hours for each of them. Before they start working, they need to press a button that states that they have started working at that time.

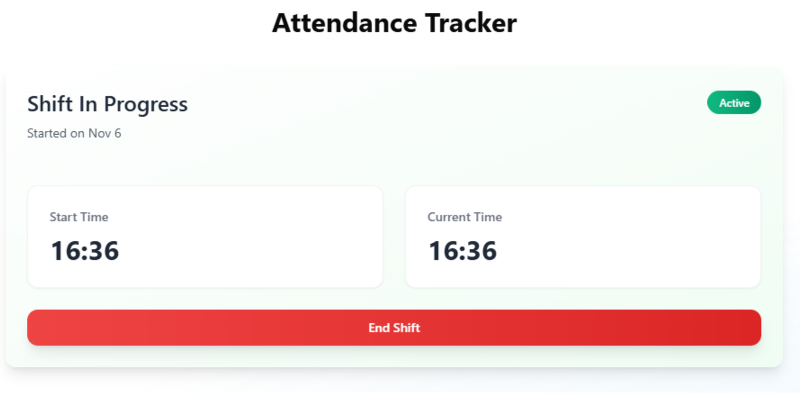

Attendance Tracker

Ready to Start

Click the button below to start tracking your work hours.

Figure C.48: Attendance Page

After the employee has finished working, they can press the button to end their shift.


Attendance Tracker

Shift In Progress

Started on Nov 6

Start Time

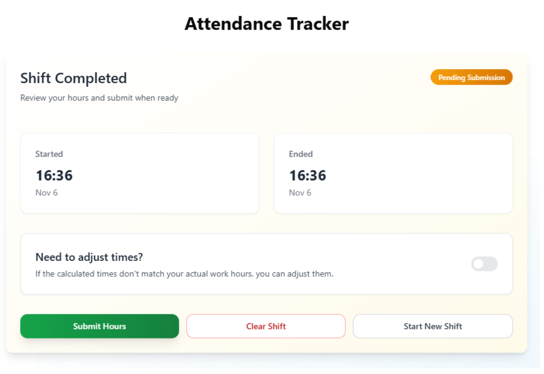
16:36

Current Time

16:36

Figure C.49: Attendance after Clocking In

After they do that, they can either submit the hours that they worked, adjust the time in case they ended up working more or less time than the time captured by the clock-in and clock-out functions, or clear the shift completely, in case, for example, they mistakenly pressed the button to start the shift. When the shift is cleared, it cannot be counted in the total hours worked for that employee, which essentially leads to their wage.


Attendance Tracker

Shift Completed

Pending Submission

Review your hours and submit when ready

Started
16:36
Nov 6

Ended
16:36
Nov 6

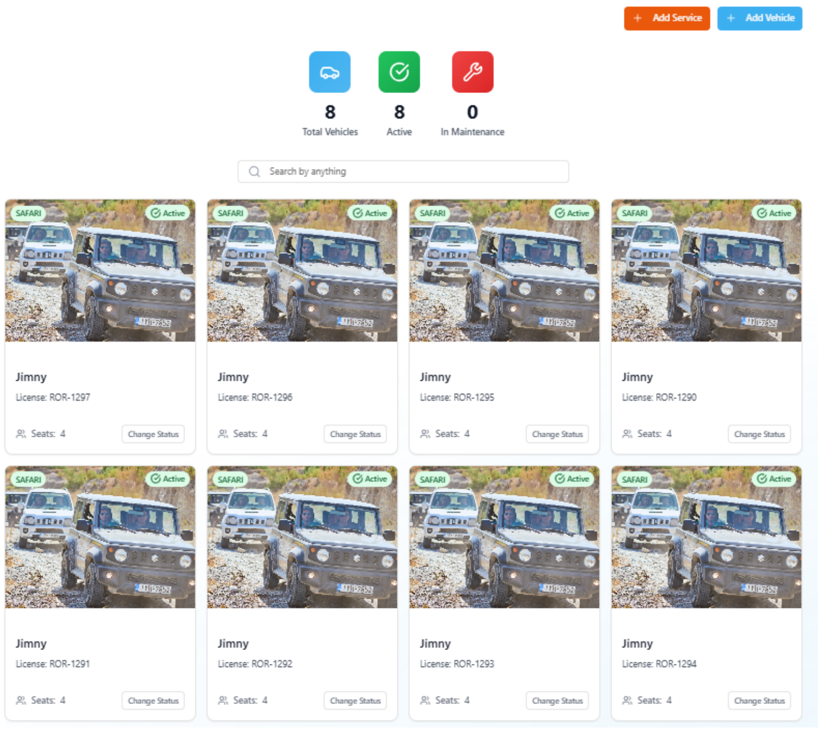
Need to adjust times?

If the calculated times don't match your actual work hours, you can adjust them.

Figure C.50: Attendance after ClockingOut

C.4.6 Vehicles

This page is used to monitor the condition of the cars. If the company wants to add a new vehicle, they need to enter the car's type, its license plate, the type of activity that it will be used for (safari or buggy), the number of seats, and the company where the car is insured.



8 Total Vehicles
8 Active
0 In Maintenance

Search by anything

SAFARI · Active · Jimny · License: ROR-1297 · Seats: 4

SAFARI · Active · Jimny · License: ROR-1296 · Seats: 4

SAFARI · Active · Jimny · License: ROR-1295 · Seats: 4

SAFARI · Active · Jimny · License: ROR-1290 · Seats: 4

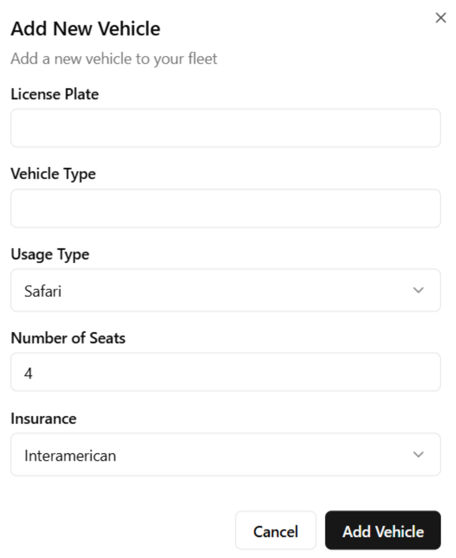
SAFARI · Active · Jimny · License: ROR-1291 · Seats: 4

SAFARI · Active · Jimny · License: ROR-1292 · Seats: 4

SAFARI · Active · Jimny · License: ROR-1293 · Seats: 4

SAFARI · Active · Jimny · License: ROR-1294 · Seats: 4

Figure C.51: Vehicle Page



Add New Vehicle

Add a new vehicle to your fleet

License Plate

Vehicle Type

Usage Type

Safari

Number of Seats

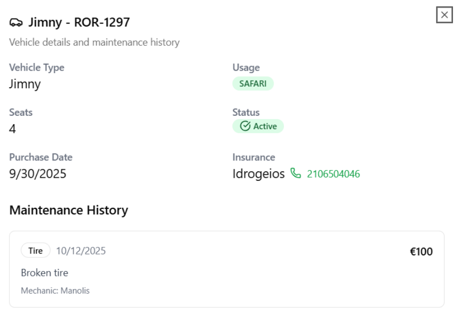
4

Insurance

Interamerican

Figure C.52: Add Vehicle

By clicking on one of the vehicles, you can see some information about the car, like its license plate, the number of seats, the date of purchase, the activity for which it is used, its status, the phone number of its insurance, and the history of previous maintenance issues. By clicking on the "Change Status," you can either retire the car if it is not in condition to be used any more or put it into maintenance if it needs to be fixed.



Jimny - ROR-1297

Vehicle details and maintenance history

Vehicle Type
Jimny

Usage
SAFARI

Seats
4

Status
Active

Purchase Date
9/30/2025

Insurance
Idrogeios 2106504046

Maintenance History

Tire 10/12/2025 €100

Broken tire

Mechanic: Manolis

Figure C.53: Vehicle Information

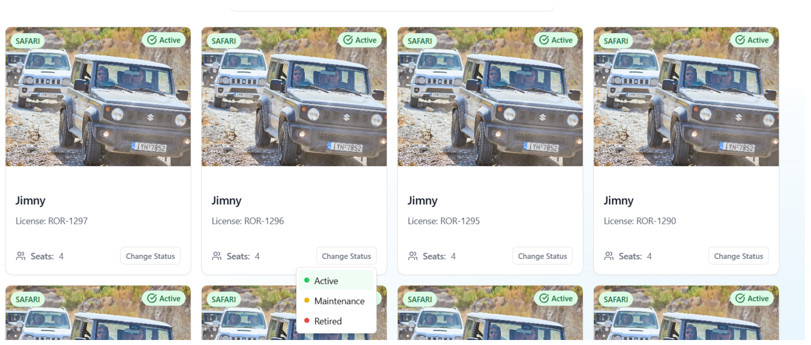
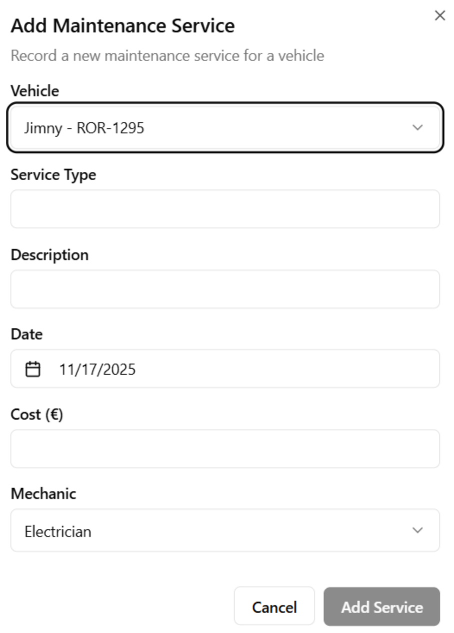


Figure C.54: Change Status

Regarding the maintenance, the employees can report issues with any of their registered vehicles through the "Add Service" button, where they can describe the car's issue, the service date, the cost, and the mechanic where the car will be sent.



Add Maintenance Service

Record a new maintenance service for a vehicle

Vehicle: Jimny - ROR-1295

Service Type:

Description:

Date: 11/17/2025

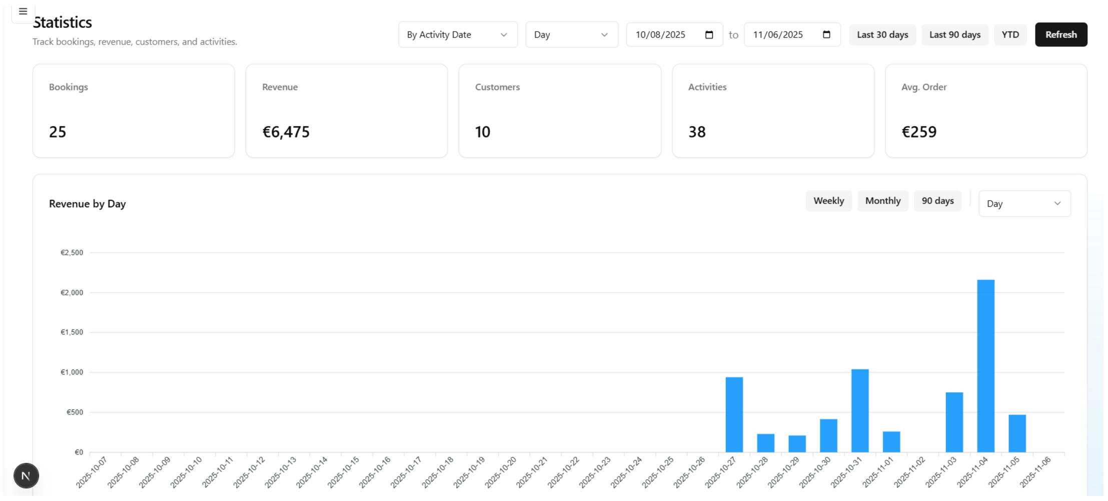
Cost (€):

Mechanic: Electrician

Figure C.55: Add Service

C.4.7 Statistics

The Statistics page provides statistics about the number of bookings per time period and the revenue made from customers for the organization of these activities. There are graphs that represent the number of activities organized and the revenue generated over time, as well as the revenue generated by each type of activity separately during that time period.



Bookings	Revenue	Customers	Activities	Avg. Order
25	€6,475	10	38	€259

Figure C.56: Statistics Page

The employees can filter these statistics by day, week, month, or year, or select specific dates and check on the statistics of that time period.

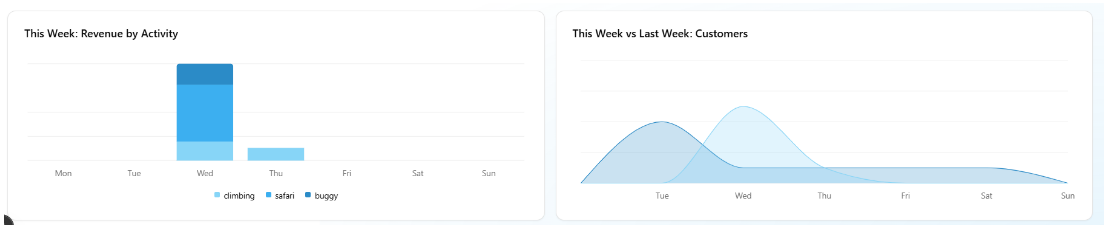


Figure C.57: More graphs

Appendix D

AI Statement

D.1 Report

While writing this report, the authors used AI platforms like Grammarly to check the grammar, spelling, and consistency of the document. Other tools that were used for the same reason were also Overleaf Smart Features. After using these tools, the authors took responsibility for the content of the work after reviewing and editing the document accordingly.

D.2 Produced Software

While preparing this product, the authors used various code generation services, like Cursor or ChatGPT, to generate and/or complete code. After using this tool/service, the authors reviewed and edited the content as needed. The authors take full responsibility for the content of the work.