Title: ParkItPlace

Revision History

Version #	Author(s)	Description	Date
0.0	All	Creation of document	October 11, 2024
1.0	All	Edits based on TA feedback	October 23, 2024

Personal of Project

Team Member	Email	Role
Ivan Ebos	ebosi@mcmaster.ca	Developer
Avin Lanson Tharakan	tharakaa@mcmaster.ca	Developer
Ben Lee	leeb51@mcmaster.ca	Developer
Muhammad Salim	salimm5@mcmaster.ca	Developer
John Wu	wu103@mcmaster.ca	Developer
Ritvik Deshpande	deshpr3@mcmaster.ca	Developer

Team Member Contribution

Team Member	Sections
Ivan Ebos	Scope of Work, Operational and Environmental Requirements, Open Issues
Avin Lanson Tharakan	Mandated Constraints, Product Use Case Table, Project Deliverables
Ben Lee	Naming Conventions and Terminology, Relevant Facts and Assumptions, Look and feel Requirements, Usability and Humanity Requirements
Muhammad Salim	Maintainability and Support Requirements, Compliance Requirements, New Problems
John Wu	Functional Requirements, Security Requirements
Ritvik Deshpande	Purpose of the project, Stakeholders, Performance Requirements

Table of Contents

1 Purpose of the Project

1.1 Purpose of the project

The purpose of ParkItPlace is to provide a peer-to-peer marketplace for parking spaces, allowing drivers to discover, reserve, and pay for available spots efficiently. The platform addresses the growing demand for parking in populated areas. ParkItPlace benefits both drivers, who gain access to convenient and affordable parking, and parking space owners, who can generate income from unused spaces.

1.2 Goals of the Project

The goals of ParkItPlace are as follows:

1. **Provide a User-Friendly Platform**: Ensure a seamless user experience for both drivers and parking space owners, with intuitive navigation and clear instructions.
2. **Optimize Parking Availability**: Enable drivers to find, reserve, and pay for parking spaces in real time, with minimal effort.
3. **Generate Additional Income for Space Owners**: Offer parking space owners the opportunity to list their spaces and generate income from underutilized property.
4. **Ensure Secure Transactions**: Implement robust payment systems to guarantee secure and reliable transactions for both parties.

2 Client and Stakeholders

2.1 Client

- **Commuters, workers, and tourists**: They use the platform to find, reserve, and pay for parking, benefiting from the convenience and accessibility of parking spaces.
- **Landlords and renters**: They list their unused parking spaces on the platform, generating income from these assets.
- **Event organizers and venues**: They offer parking during events
- **Businesses (e.g., restaurants, hotels, shopping malls)**: They list their parking spaces during off-peak hours, turning them into a source of revenue.
- **Delivery drivers and ride-share drivers**: They use the platform to find short-term parking for quick stops in busy areas.

2.2 in-direct Stakeholders

- **Real estate developers**: They assess parking demand using data from the platform, potentially influencing future development projects and urban planning.
- **Local government and universities**: They can manage parking congestion more effectively and leverage the platform to optimize parking in busy spaces

3 Constraints

3.1 Solution Constraints

Data Privacy and Security: All user data and payment details, must be protected in compliance with relevant data protection regulations (eg GDPR). Data integrity must be enforced using secure encryption protocol during transactions

Platform Compatibility: The solution should be compatible with multiple platforms, such as web browsers, Android, and iOS devices. The application must be compatible with different screen sizes to ensure a good user experience.

Third-Party Dependencies: Third-party services such as mapping APIs (Google maps), payments systems (Stripe, PayPal), and messaging APIs must be integrated seamlessly and must ensure service availability and uptime

3.2 Implementation Environment of the Current System

Development Stack:

- Frontend : React native for building user interfaces, ensuring a responsive design.
- Backend : Node.js for building the server-side logic. Will handle RESTful API calls.
- Database : Firebase for storing user data, listings, reviews and any other related data.
- Mapping Service : Google Maps or equivalent API for real-time map integration
- Authentication: OAuth for Google Login integration.

Deployment Environment:

- The application is to be hosted on cloud services such as firebase DBs. Continuous Integration/Deployment (CI/CD) tools must be implemented for efficient deployment, versioning, and scaling.
- The app will rely on mobile platforms (iOS and Android), making React Native for development, and ensuring code reuse across mobile devices.

3.3 Partner or Collaborative Applications

Payment Gateways : The platform will integrate a payment service provider such as Stripe or PayPal to handle transactions securely.

Cloud Hosting Services: Firebase services will provide the infrastructure for hosting databases, running the backend and storing user data securely.

Geolocation and Maps : Google Map APIs (or similar services) will provide map functionalities.

SMS and Email Services: APIs like Twilio or equivalent will handle communication services such as sending booking confirmations, reminders, or other notifications via email or SMS.

3.4 Schedule Constraints

Project Timeline: The project must be completed within the duration of the CS 4ZP6A course for 2024/25 or by restricting deadlines as mentioned by the course professor.

Milestones:

- **MVP Delivery**: The minimum viable product (MVP) must be ready by November 21st or earlier if the course professor requires it.

- **User Testing**: Testing with real users (students, commuters, etc) must be completed before the final submission to ensure time for bug fixing and feedback integration.
- **Final Submission**: The completed application as well as all documentation, must be submitted by April 4th, 2025 or earlier if the course professor requires.

3.5 Budget Constraints

Free Tier Services: The team must try their best to work within the free tier limits of various third-party services (Google Maps API, Twilio, Stripe, cloud hosting services, etc) when possible, otherwise the team must come to a decision on inevitable expenses such as App Store fees.

4 Terminology

4.1 Abbreviations, naming conventions & definitions

1. **ParkItPlace**:
   The app that connects drivers with parking spaces offered by individuals or businesses.
2. **Leaser**: Someone who lists a parking spot for rent on the platform.
3. **Commuter**: A person looking for and reserving parking spots through the app.
4. **MVP (Minimum Viable Product)**:
   The most basic version of the product that works and can be tested by users.
5. **OAuth**: A security system that lets users log in using services like Google.
6. **Stripe/PayPal**:
   External payment systems integrated to process payments securely on the app.
7. **CI/CD (Continuous Integration/Continuous Deployment)**: A method of automatically testing and updating the app to improve speed and reliability.
8. **RBAC (Role-Based Access Control)**: A system for managing what different users (leasers, commuters, admins) can access and do in the app.
9. **P0, P1, P2, etc.**: The priority levels we assign to different features. P0 is the most critical, while P4 is for optional features or future enhancements.
10. **Google Maps API**:
    A service we use to provide maps and location information in the app.
11. **Firebase**: A cloud service we use to store data, handle authentication, and run the backend of the app.
12. **API (Application Programming Interface)**:
    A way for different software components to communicate with each other.
13. **GPS (Global Positioning System)**:
    A technology that lets the app find a user’s or parking space's location using satellites.

5 Relevant Facts And Assumptions

5.2 Assumptions

1. Users will have smartphones with GPS or access to a desktop browser when using the platform.
2. Both drivers and parking space owners will be familiar with mobile and web apps, as well as online payments.

3. A stable internet connection is assumed, especially when users are booking spots or processing payments.

6 The Scope of the Work

6.1 The Current Situation

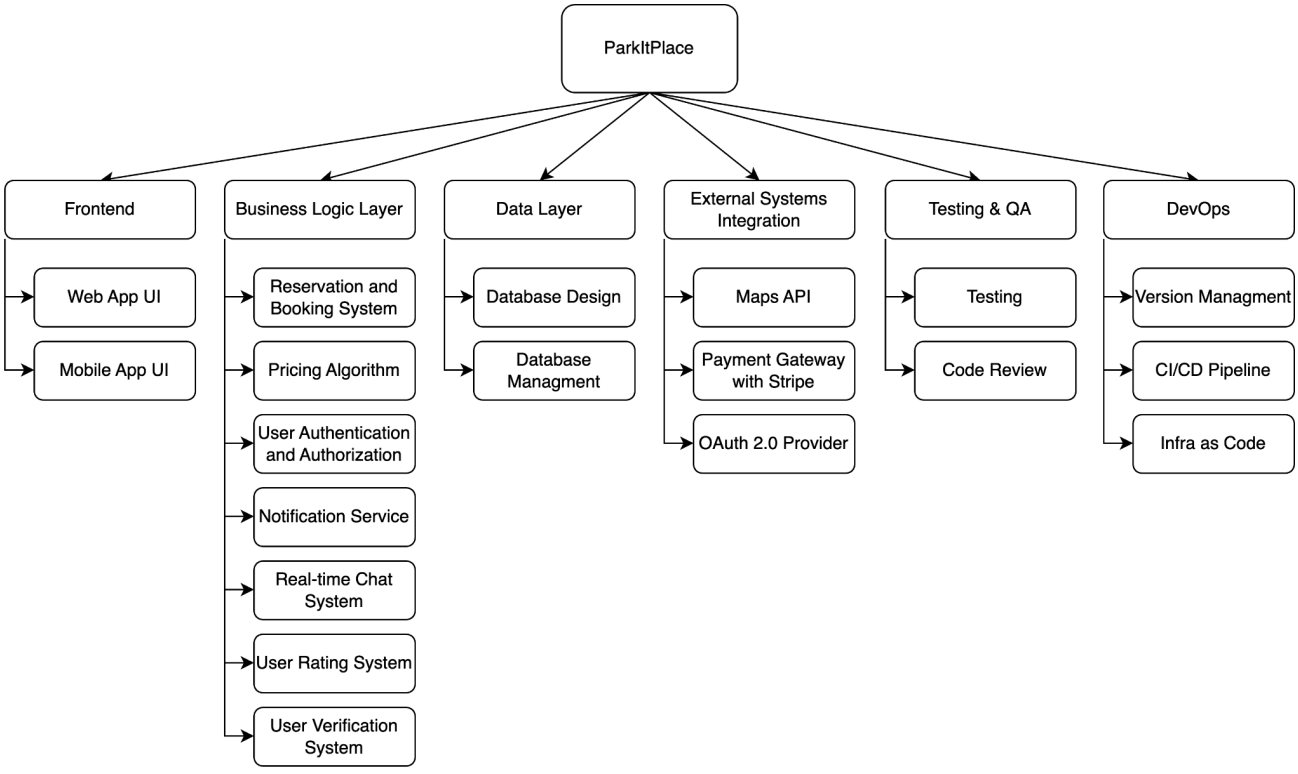
Currently, there is no peer-to-peer marketplace that effectively connects drivers with available parking spaces offered by individuals, businesses, or property owners. Existing parking solutions fall short in several key areas, such as providing **visibility** for drivers to locate the most convenient or cost-effective spot, offering **flexibility** in parking duration, and **competitive pricing** based on location and demand.

Drivers frequently struggle to find available parking in real-time, particularly in congested urban areas, while many parking spaces owned by private individuals or businesses remain **underutilized**. This creates a gap between supply and demand in parking availability, leading to frustration for drivers and missed opportunities for property owners to monetize their spaces.

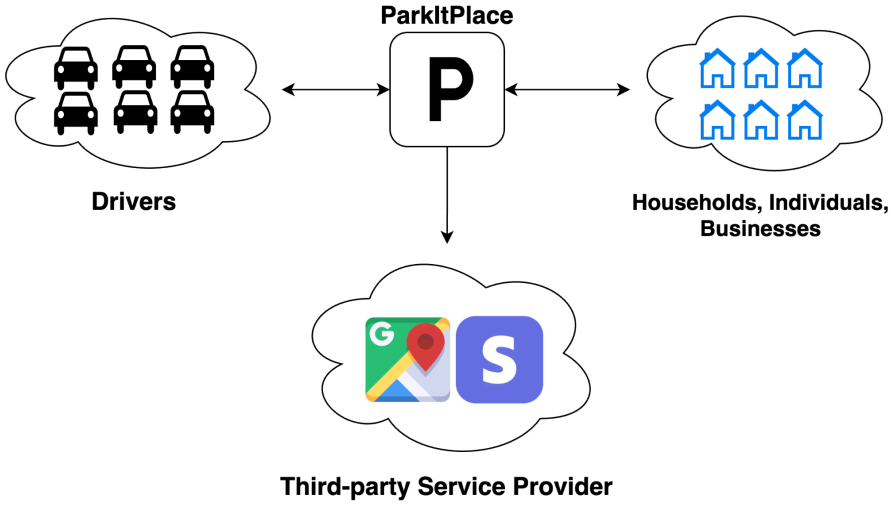
Key issues include:

- **Parking Shortages**: Commuters, tourists, and workers frequently face difficulty finding parking, especially in high-demand urban areas, leading to significant inconvenience.
- **Unused Resources**: A substantial number of privately-owned or business-operated parking spaces sit idle and are not easily accessible to drivers looking for parking.
- **Inflexible Solutions**: Current parking systems generally lack customizable options, such as hourly, daily, or weekly rentals, as well as real-time availability and variable pricing that adapts to demand and location.

6.2 Work Partitioning



6.3 Contextual View of Scope of Work



7 Product Use Case table

Use Case ID	Use Case Name	Description	Actors
U01	User Registration and Authentication	Users can register and log in using email, phone, or third-party login services (e.g. Google), ensuring secure access to their accounts.	Leasers, Commuters
U02	Create Driveway Listing	Leasers can create and manage parking spot listings by providing details such as location, price, availability, and images.	Leasers
U03	Search Parking Spots	Commuters can search for available parking spots using location, pricing, availability, and filters, displayed in both list and map views.	Commuters
U04	In-App Messaging	Leasers and commuters can communicate via in-app messaging to clarify details such as availability, pricing, or parking instructions.	Leasers, Commuters
U05	Booking and Payment	Commuters can reserve a parking spot by selecting a time slot, making a payment through integrated payment gateways, and receiving a booking confirmation.	Commuters, Leasers
U06	Rating and Review	After a parking session, leasers and commuters can rate each other and leave reviews, helping to maintain trust in the platform.	Leasers, Commuters
U07	Admin Management	Admins can manage users, handle disputes, and moderate listings and reviews.	Admin

8 Functional Requirements

P0: Critical (Highest Priority)

User Registration and Authentication

- Users must register using email, phone number, and name; mandatory fields include name, email, phone number, and password.
- Third-party login via services like Google.
- Passwords must be stored securely (e.g., hashing/salting).
- Roles are categorized into User (Commuters + Leasers) and Administrators, with mutually exclusive functionalities.

Driveway Listing Creation and Management

- Leasers can create listings with automatic or manual location input, set prices (hourly, daily, or weekly), and availability.
- Payment options: local handling or via app-moderated transaction services.
- Price suggesting algorithm based on location, market comparison, and historical data.

Search and Filtering

- Commuters can search for parking spaces by location, proximity to campus, price, availability, and ratings.
- Search results are displayed both in list view and on an interactive map with clickable pins.

In-App Communication

- Leasers and commuters can communicate via in-app messaging.
- Notifications for new messages, booking confirmations, and session reminders.

Booking and Payment Management

- Commuters book a parking spot by selecting a pin, specifying start and end time, and submitting a booking request.
- Integrated payment systems like Stripe or PayPal for transaction processing.
- Both parties receive booking confirmations via email or app notifications.

Ratings and Reviews for Credibility

- After each parking session, users rate each other on a 1-5 star scale.
- Written reviews are optional but encouraged to provide feedback on the experience.

Third-Party Technology Integration

- Integration with payment systems like PayPal or Stripe.
- Integration with mapping services (Google Maps) for location accuracy and navigation.
- Integration with services like Twilio for SMS or email notifications.

P1: High Priority

Driveway Listing Creation and Management

- Leasers can upload photos/videos (1-5) to showcase their parking space.
- Time-based pricing, with varying prices for peak hours or long-term bookings.
- Editing active listings (pricing, availability, images).
- Leasers can set recurring schedules for availability (e.g., every Monday/Friday).

Search and Filtering

- Users can filter by distance from campus (e.g., within 2 KM).
- Commuters can navigate directly to the parking spot using Google Maps, Apple Maps, or Waze after booking.

In-App Communication

- Users can enable or disable push notifications.

Booking and Payment Management

- System conducts a payment test before confirming the full transaction once the commuter confirms departure.
- Stay extension option, prompted 30 minutes before time ends.

Admin and Moderation Panel

- Admins should have a dashboard to monitor user activity and disputes.
- Admins should have the ability to deactivate accounts and delete listings or reviews.

User Credibility Score

- A user's credibility score is accumulated based on ratings from completed sessions and is displayed publicly.

P2: Medium Priority

User Authentication

- Complex password enforcement (minimum 8 characters, mix of upper/lowercase, numbers, special characters).
- Password recovery via email.

Profile Management

- Users can update their profile information, including name, profile picture, and payment info.
- Users can bookmark/favorite listings and view them later on a separate page.

Booking and Payment Management

- After the session, both parties receive an invoice detailing the transaction, duration, and service fees.

P3: Low Priority

User Registration and Authentication

- Optionally, users can upload proof of identity, driver's license, or proof of house ownership.

Driveway Listing Creation and Management

- Feature-based pricing, where driveway characteristics (e.g., size, security cameras) affect the suggested price.
- Manual toggle for leasers to change availability temporarily.

Search and Filtering

- Users can view their recent search history or rerun previous queries.

Messaging and Notifications

- System to manage message history size and storage duration.

Cancellation Policy

- Refund if they cancel before specified time
- Users who cancel too many bookings within a timeframe are flagged for investigation or action.

Report Misuse

- Users can report inappropriate behavior or misuse, triggering a system admin investigation.

P4: Optional (Future Enhancements)

Driveway Listing Creation and Management

- Listing status can be manually toggled by leasers for temporary changes.
- Analytics showing views, inquiries, and bookings per listing for leasers.

Time Enforcement and Penalties

- Seller can report overstaying which alerts the user

User Credibility

- Incentives for users to provide ratings and reviews.

Parking Timer and Overstay Alerts

- Parking timer begins when the reservation start time starts

9 Non-Functional Requirements

9.1 Look and feel Requirements

- **Color Scheme:** The app should utilize an overall color scheme that makes sense, resonates with the logo, and makes it easier to identify different sections in the UI.
- **Organization**: The app should organize features and functions in a way that makes sense, grouping them by similarity or relevance.
- **Uniformity**: The overall style of the app, (fonts, layout, theme) should be uniform throughout the entire app, to enforce similarity.
- **Necessity**: The information displayed to the user should be concise, minimal and necessary, avoiding redundant or irrelevant information to reduce noise.

9.2 Usability and Humanity Requirements

- **Clarity**: The app will display information in a straightforward easy to digest manner, more important and urgent information will take more precedence on the users screen to catch their attention.
- **User Feedback**: For any interaction the user makes with the app, the app will utilize visual or audio queues to notify and provide feedback to the user that the interaction was successful, i.e clicks, chimes, hovering of buttons, interaction animations, etc
- **Tutorial**: The app will provide new users with an interactive tutorial the first time they use the app to speed up the learning process of using the app
- **Concise**: The app will provide only the necessary information in a concise short form that can both easily and quickly be digested by the user
- **Accessibility:** Give users the option to mark their spots as wheelchair accessible.

9.3 Performance Requirements

Description: ParkItPlace must provide a user experience with minimal delays.

Requirement:

- The app should respond to user interactions (e.g., map navigation, parking search) quickly
- Transactions, such as booking or reserving parking spots, should be processed instantaneously ()
- API responses, such as retrieving available spots, should be under normal server load.

Description: Although ParkItPlace is not a safety-critical system like medical or automotive systems, there are still safety aspects that need to be considered, particularly around financial transactions and privacy.

Requirement:

- All financial transactions must be secured with industry-standard encryption (e.g., TLS 1.2 or higher).

- User personal data, including payment details, should be protected using encryption both at rest and in transit.

Description: The system must be able to handle errors gracefully and recover from faults without crashing or losing critical data.
Requirement:

- In the event of a network failure or a server crash, users must not lose their reservation information, and the system must ensure data consistency when the connection is restored.
- The system must retry failed API requests up to 3 times before notifying the user of an error.
- The application should maintain a 99.9% uptime for critical features like booking and payment processing.

9.4 Operational and Environmental Requirements

User Environment:

- The mobile application is designed to operate primarily on smartphones equipped with GPS capabilities.
- The web application will be accessible via modern web browsers on both desktop and mobile devices. Compatibility with major browsers

Backend Environment:

- The application will leverage Firebase as its cloud provider, utilizing various storage services to manage all ParkItPlace data, along with Firebase Hosting for secure and reliable application deployment.

Testing Requirement:

- All production releases will have completed testing processes, including unit tests, integration tests, and end-to-end tests.

9.5 Maintainability and Support Requirements

Code Modularity & Reusability

- Ensure the app's code is organized into modular components to simplify updates and fixes.

Well-Documented Code

- Maintain code documentation to help developers understand and modify the code easily.

Version Control & Continuous Integration

- Use version control (e.g., Git) to ensure code changes are automatically tested and tracked.

Bug Tracking & Issue Management

- Implement a system like Jira or GitHub Issues to track bugs and monitor their resolution.

Scalability for Future Updates

- Design the system to accommodate new features and increased usage without requiring major redesigns.

Scheduled Maintenance Windows

- Plan regular maintenance windows for updates, bug fixes, and database optimizations.

Technical Support

- Implement a help desk, tickets to assist users with issues.

User Documentation and Self-Help

- Provide user guides and FAQs to help users resolve common issues independently.

Monitoring

- Use error logs and performance metrics to monitor app performance and detect issues.

Backward Compatibility

- Maintain compatibility with older app versions to avoid disrupting users who haven't updated.

User Feedback

- Provide in-app options for users to report issues or suggest features.

9.6 Security Requirements

User Authentication

- All users (Leasers/Commuters, Admins) must authenticate before accessing any controllable or restricted areas of the platform
- Authentication will be done using logins corresponding to their Gmail accounts
- If the user chooses to sign in without the Gmail portal and user will use their own email + password combination for our platform login. Then they must make or use a password that has sufficient complexity. minimum length of 8 characters, must include upper and lower case letters, numbers, and special characters
- Optional: 2FA; introduce 2 factor authentication (maybe just for admins).

Role-Based Access Control (RBAC)

- The system must implement RBAC to ensure that users only access features and data that are relevant to their roles (Leasers/Commuters, Admins).
    - User: Have access to making postings, viewing postings and interactions that would only be available to users.
    - Admins: Have access to system-wide management features, including user management, content moderation, and dispute resolution.

Session Management

- User sessions must have a configurable timeout (e.g., 15 minutes of inactivity), after which users must re-authenticate.
- Logout Function:
  Users should be able to manually log out from the application from any device. Sessions should be securely terminated when logged out.
- Optional: IP Logging; each session's IPs will be logged to detect unauthorized or suspicious logins.

Password Storage and Management

- User passwords must be encrypted using industry standard cryptographic techniques (e.g. hashing algorithms such as Argon2 or bcrypt).
- Password Recovery:
  If a user forgets their password, they can reset it using a password recovery system. This process will require multi step verification, such as answering security questions or verifying via phone or email.

API Security

- Access to backend APIs will require valid API keys or OAuth tokens.
- Rate-limiting and throttling will be enforced to prevent abuse of APIs.

Transaction Integrity:

- The system must ensure that all financial transactions (payments, refunds, penalties) are accurately recorded and processed.

- Atomic Transactions:
  Either all parts of a transaction succeed, or the system must roll back to prevent partial failures (e.g., booking without payment).
- Payment gateways (e.g., *Stripe*, *PayPal*) must use secure protocols (e.g., HTTPS, TLS 1.2+) to ensure the integrity of financial data.

Backup and Recovery:

- Daily backups of all critical data (user profiles, listings, booking history, and transaction records) must be performed
- In the event of a system failure, data integrity must be preserved through recovery procedures that restore data from backups without loss or corruption.

Data Encryption:

- All sensitive data (e.g., personally identifiable information (PII), payment information) must be encrypted during storage and transmission.

Privacy Policy and Consent:

- Users must be presented with a *Privacy Policy* at the time of registration, clearly explaining how their data will be used, stored, and shared.
- Users must provide explicit consent to collect and store personal information in compliance with *GDPR* and other relevant privacy laws.
- Users will have the ability to revoke their consent at any time and request the deletion of their data (Right to Erasure).

Audit Logging:

- The system must maintain detailed *audit logs* of critical events including:
  - User login/logout activities.
  - Changes to user profiles or listings.
  - Financial transactions (payments, refunds, cancellations).
  - Administrative actions (e.g., account bans, disputes resolved).

Log Retention:

- Audit logs will be retained for a minimum of 1 year to facilitate the investigation of disputes, security incidents, or system failures.
- Sensitive data within logs (e.g., personally identifiable information) will be redacted where necessary to protect user privacy.

Access to Logs:

- Only authorized administrators should have access to audit logs, with access restricted to a "need-to-know" basis.
- Tamper Protection: Logs must be immutable or stored in a way that prevents unauthorized modification or deletion.

Audit Reports:

- The system must support generating audit reports for financial reviews, regulatory compliance, and internal audits. These reports must include details such as transaction summaries, user activity logs, and system health metrics.

9.7 Compliance Requirements

Terms of Service & Liability

- Create clear ToS and liability clauses to define user responsibilities, such as driveway owner liability for accidents.

10 Data and Metrics

10.1 Data

User Data: All user-supplied data, such as account information, parking spot listings, reservation history, , payment details, etc. will be collected and stored securely in Firebase.
Google Maps API Data: Real-time geolocation data retrieved from the Google Maps API for accurate mapping and navigation.

10.2 Performance metrics

Monthly Active Users: Number of active users each month to track platform growth and engagement.
Revenue per User: Average revenue generated from each active user, calculated as total monthly revenue divided by the number of active users helps assess the platform's profitability

11 Risks and issues predicted.

11.1 Predicted Issues

Verification of Parking Spot Data: No process to verify location, availability, or ownership of parking spots, leading to potential inaccurate data.
User Credibility System: No framework for determining and maintaining credible ratings for buyers and sellers, leading to potential manipulation.
Marketing Strategy: Lack of defined marketing channels, target demographics, and messaging, as well as concerns about budget and scalability.
System Compatibility: Potential conflicts with existing software and platforms during deployment.
User Experience (UX) and Trust: Difficulty navigating the app for new users and concerns over privacy and security.

11.2 Risks

Increased Network Traffic: Real-time features (e.g., location updates, availability tracking) may lead to slower network performance, especially in high-traffic areas or with limited bandwidth.
Device and Network Constraints: Limited functionality on older devices or in areas with poor connectivity may impact user experience.
Post-Launch Bugs: Unexpected bugs and performance issues may arise after launch.
Scalability Issues: Rapid user growth may stress servers and degrade performance.

12 Project Deliverables

Functional Applications:
Fully developed mobile application (for IOS and Android) and web applications.
Backend Infrastructure: Secure servers, databases, and APIs.
Testing Reports: QA and testing results.
Deployment: Live applications on app stores and web hosting platforms.
Training Materials: Guides for admins and support teams. (will include guides to common user's query tickets to ensure a standard response from the team)