CST2550

Software Engineering Management and Development Group Project

Car Hire System

2024/25

Date of Submission: Wednesday 16 April 2025.

Student Name: Ramachundrah Dhruv, Lochun Aayush, Mahomathoo-Ghaboos Syed Rumaisa, Booputh Yashvin, Goorodoyal Khorisha.

Student ID Number: M01012616, M01004726, M01012611, M01006629, M01003456

Lab Tutor: Miss Aisha Idoo

Table of Contents

Contents

Table of Figures

Project Management Plan – Car Hire System

1. Introduction

This Project Management Plan outlines the key strategies, tools, and processes used to manage the development of the HorizonDrive Ltd Car Hire System. It covers the team structure, Agile methodology, risk handling, resource requirements, deliverables, and monitoring methods. The plan ensures effective project execution and successful system delivery.

2. Project Organization

2.1 Team Structure

Name	Role	Responsibilities
Ramachundrah Dhruv	Team Leader	Oversees project workflows and ensures the team follows Agile principles. Facilitates Agile processes (such as sprint planning, retrospectives, and daily stand-ups), removes obstacles, and ensures smooth team collaboration without exerting direct authority. Acts as a guide and mentor to the team.
Lochun Aayush	Secretary	Manages administrative tasks, including documenting meeting minutes, maintaining records, scheduling, and assisting in coordinating project-related activities.
Mahomathoo-Ghaboos Syed Rumaisa	Developer	Part of the cross-functional development team. Collaborates in delivering working product increments at the end of each Sprint. Contributes to coding, problem-solving, and integrating features while actively participating in Sprint planning and review sessions. Shares responsibility for Sprint outcomes with the team.
Booputh Yashvin	Developer	Works alongside the development team to implement features and transform backlog items into functional components during each Sprint. Involved in coding, testing, and integration efforts to ensure timely and high-quality deliveries.
Goorodoyal Khorisha	Tester	Ensures software quality through systematic testing processes. Designs and executes test plans, identifies bugs, and verifies compliance with functional requirements, thus supporting the delivery of a reliable system.

2.2 Stakeholders Identification

Stakeholder	Role in the System

Company Management	Oversees operations, manages the company's own fleet, and partnerships with lessors.
Customers	Rent cars, either from the company directly or external car owners.
External Car Owners (Lessors)	List their cars for rent through the system.
Admins/System Managers	Maintain system operations, manage accounts, and address customer/lessor concerns.
Insurance Providers	Offer coverage for cars rented through the platform, ensuring customer protection.
Car Maintenance Providers	Handle maintenance for company-owned cars and optional services for external cars.
Payment Processors	Facilitate secure transactions between all parties involved.
Drivers (if applicable)	Provide chauffeur services for customers.
Regulatory Authorities	Ensure compliance with local laws and standards.
System Developers	Create and update the platform, optimizing it for company and lessor needs.

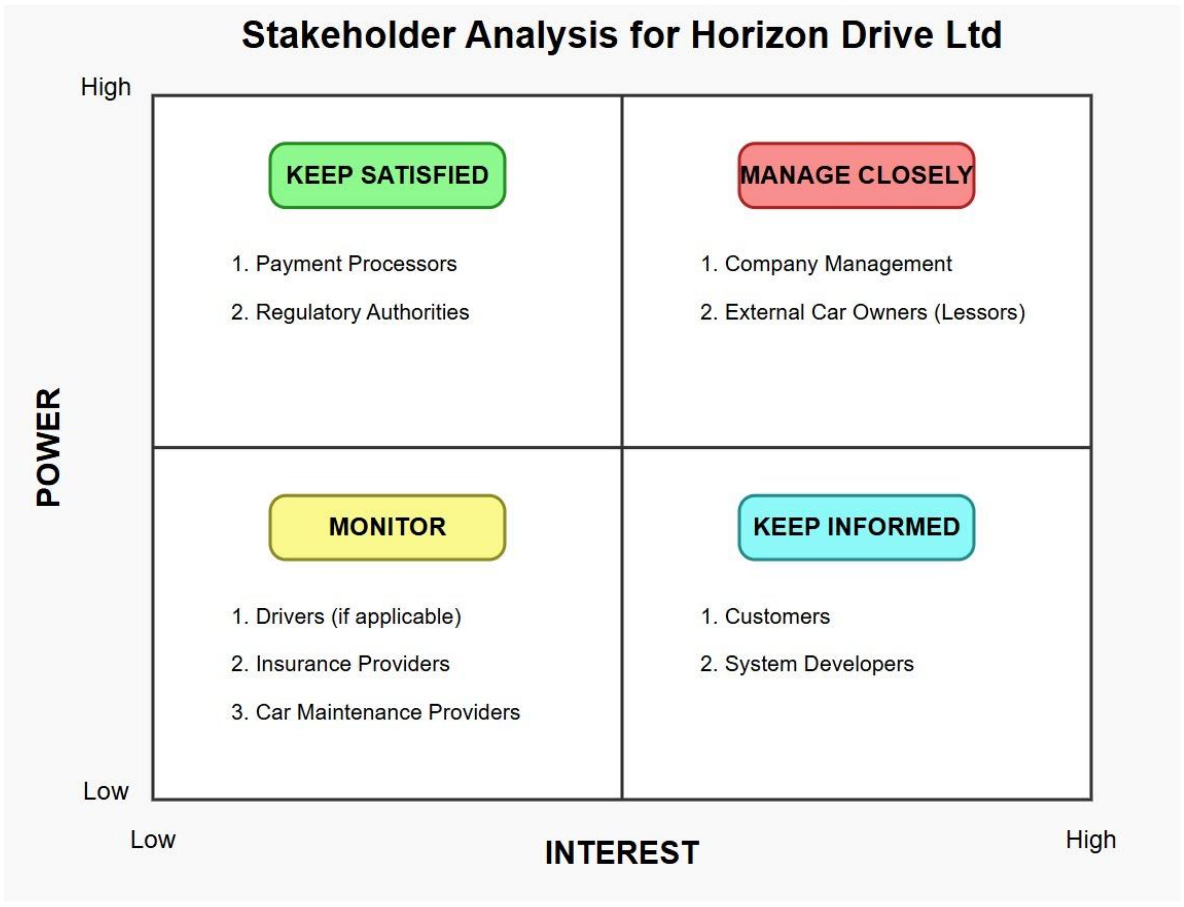


Figure 1: Power Interest Grid of Stakeholders

2.3 Task Allocation

Task	Assigned Role
UI/UX Design	Yashvin, Khorisha, Aayush
Backend (C#.NET)	Dhruv, Rumaisa
Database (SQL Server)	Dhruv, Rumaisa
Documentation	Aayush, Khorisha
Testing	Dhruv, Khorisha

3. Lifecycle Model Used

3.1. Lifecycle model selected

The project will follow the **Agile methodology**, which is well-recognized for its iterative development, continuous feedback, and adaptability (Beck et al., 2001). Agile allows for the rapid delivery of functional software through short, time-boxed iterations, known as Sprints, which typically span 2–4 weeks and conclude with a potentially shippable product increment (Schwaber and Beedle, 2002).

Key phases in the Agile lifecycle include:

- **Planning:** Involves outlining tasks and organizing the list of work items to effectively guide the project (Cockburn, 2002).
- **Iterations (Sprints):** Each Sprint targets specific tasks, with regular reviews and adjustments to accommodate changes (Highsmith, 2009).
- **Development and Testing:** Features are continuously developed and tested to ensure high quality throughout the process (Shore and Warden, 2007).
- **Delivery:** Each Sprint concludes with a working version of the product, refined based on stakeholder feedback (Cohn, 2004).

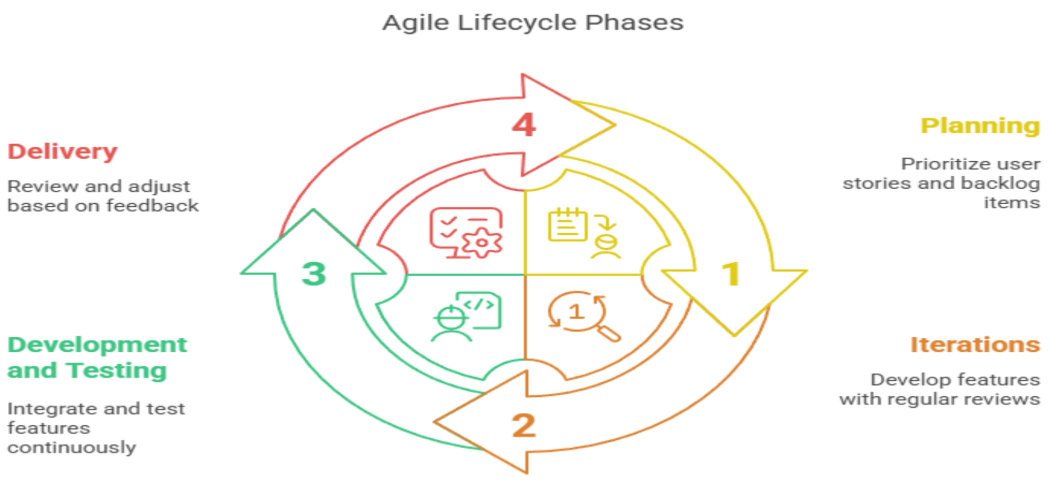

Agile Lifecycle Phases
Planning
Prioritize user stories and backlog items
1
Iterations
Develop features with regular reviews
2
Development and Testing
Integrate and test features continuously
3
Delivery
Review and adjust based on feedback
4

Figure 2: Agile Lifecycle Phases

The Agile approach has been chosen for its ability to foster **flexibility** and **continuous improvement**, ensuring **active stakeholder collaboration** at every stage of the project (Beck et al., 2001).

3.2. How it was applied to this project

The Agile methodology was applied to this project as follows:

- **Sprint Planning:** The team prioritized high-value features such as the **frontend development** for **booking functionality**, **car listing**, and **user login**, ensuring a user-friendly interface. Additional backend tasks, such as implementing a **hash table** and **connecting to SQL databases**, were also planned to support these features.
- **Sprint Reviews:** These reviews were conducted internally by the **System Developers**, focusing on evaluating the frontend interfaces for booking, car listing, and user login, as well as progress on backend tasks like database connectivity. Feedback within the team was used to improve functionality and design.
- **Sprint Retrospectives:** For retrospectives, **meeting minutes** served as the main form of documentation, capturing discussions about what went well, areas for improvement, and how to address challenges in future Sprints.

4. Risk Analysis

Risk Description	Likelihood	Impact	Mitigation Strategy
Team member unavailable (illness, etc.)	Medium	High	Task reallocation and adding buffer time in the schedule
Data loss or Git conflict	Medium	Medium	Use version control (Git) with regular backups
Timeline constraints due to preponed date	High	High	Prioritize critical tasks, streamline communication, and reallocate resources
Frontend feature delays (e.g., booking, car listing, user login)	Medium	High	Monitor progress closely, break tasks into smaller milestones, and conduct regular reviews
Miscommunication in task requirements	Medium	High	Encourage detailed documentation and regular team check-ins

Delays in SQL database connection setup	Medium	High	Perform early testing, use mock databases to prepare backups
Frontend design inconsistencies	Medium	High	Conduct frequent reviews and usability testing
Burnout or fatigue in team members	Medium	High	Schedule breaks, promote teamwork, and distribute workloads

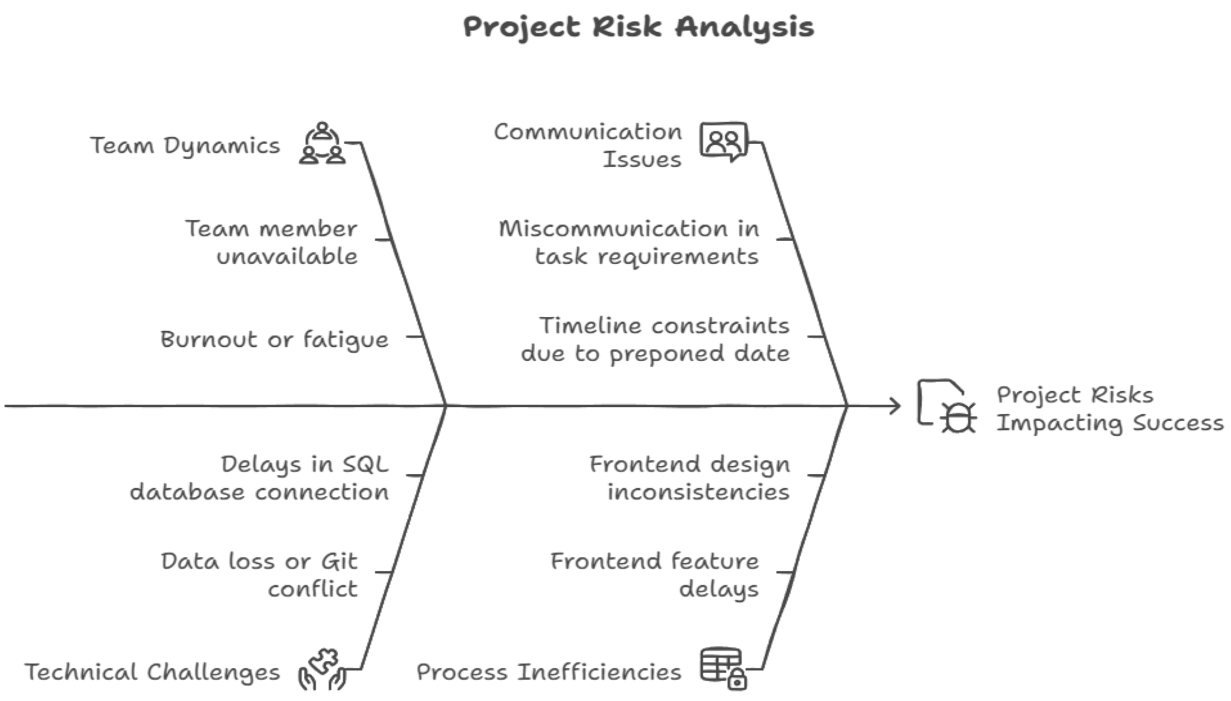

Project Risk Analysis
Team Dynamics
Team member unavailable
Burnout or fatigue
Communication Issues
Miscommunication in task requirements
Timeline constraints due to preponed date
Project Risks Impacting Success
Delays in SQL database connection
Data loss or Git conflict
Technical Challenges
Frontend design inconsistencies
Frontend feature delays
Process Inefficiencies

Figure 3: Risk Analysis Summary

5. Software Resource Requirements

Tool/Technology	Purpose
Visual Studio 2022	Frontend and backend development in C#.NET (including WinForms for frontend)
SQL Server	Database creation(connected using Entity Framework Core for implementation)
Figma	UI/UX prototyping
Lucidchart	Creating ERD diagrams
Git/GitHub	Version control for tracking changes, enabling collaboration, resolving conflicts, maintaining backups, and ensuring transparency
Google Meet	Conducting team meetings and collaborative discussions to ensure clear communication and alignment during the project lifecycle

6. Deliverables and Schedule

6.1. Planned Deliverables

The project aims to deliver the following key components:

1. **GitHub Repository**
   - Establishing a repository on GitHub to host the project's source code, including frontend, backend, and database integration.
2. **Hash Table Implementation**
   - Using hash tables as a key data structure for efficient storage and retrieval in the system.
3. **Frontend Development**
   - Designing and implementing the user interface using WinForms, enabling functionalities such as car search, booking, and lessor features (list a car, browse listings).
4. **SQL Connection**
   - Connecting to the database to ensure reliable data storage and management for bookings, cars, and users.
5. **Documentation**
   - **Developer Manual:** A detailed report covering the system's architecture, backend and frontend integration, and hash table usage.
   - **README File:** A user guide within the GitHub repository explaining how to navigate and use the system.
6. **Video Demonstration**
   - A walkthrough video showcasing the system's features, workflow, and design architecture.

6.2. Sprint Schedule

Sprint	Date	Focus Areas	Key Tasks Completed
Sprint 1	21/02/2025	Initial planning and group formation	- Formed team and assigned roles -Reviewed coursework requirements and ensured alignment - Discussed preliminary design ideas
Sprint 1	22/02/2025	Requirements gathering, research on C#.NET	- Researched car hire systems - Identified user roles: Admin, Customer, Seller - Outlined database tables: User, Car, Booking, Payment, Maintenance

Sprint 1	23/02/2025	Functional requirements & data flow	- Defined functional requirements - Designed user flow for login, registration, car browsing, listing, and booking
Sprint 1	26/02/2025	Wireframes and use case diagrams	- Created initial wireframes for main pages - Drafted use case diagrams for login, booking, car maintenance
Sprint 1	28/02/2025	Finalizing ERD and use case diagrams	- Completed ERD with 5 main tables - Finalized and reviewed use case diagrams
Sprint 1	03/03/2025	Sprint review & documentation update	- Updated and documented ERD and use cases - Planned Sprint 2 objectives
Sprint 1	04/03/2025	UI sketching and WinForms prototyping	- Began sketching splash screen, login, signup, browse pages - Explored WinForms tools
Sprint 2	06/03/2025	Database setup, UI/UX design planning	- Created database structure - Researched WinForms UI/UX - Started splash screen (Aayush), browse page (Yash), and list-a-car page design (Khorisha)
Sprint 2	07/03/2025	Frontend development, database debugging	- Implemented initial database structure - Finalized splash and list-a-car page UI - Browse page layout completed - Improved access to UI design resources
Sprint 2	08/03/2025	UI finalization, backend testing	- Finalized splash screen - Started login page - Tested DB-frontend connectivity - Researched hash table (Dhruv)
Sprint 2	12/03/2025	Backend logic, data structures research, UI alignment	- Trial and error with hashing - Started signup page - Research on human verification - Began backend linking - Redesigned list-a-car layout
Sprint 2	14/03/2025	Dynamic UI, data structure testing, backend layout	- Finalized signup layout - Continued hash table and linked list testing - Began account page layout - List-a-car layout updated with new ideas
Sprint 2	16/03/2025	Integration of hashing & data structures, functionality updates	- Integrated hash table into login - Linked list added for listings - Backend logic implemented in account page - Bug fixes and enhancements for list-a-car page
Sprint 3	21/03/2025	Backend refinement, UI enhancements	- Backend added to List-a-car page

			- Continued layout updates for manage listings and transactions
Sprint 3	22/03/2025	Data structure integration, UI finalization	- Hashtable tested and updated - Signup/login forms updated for backend compatibility - Linked list tested for value retrieval - Option-Account page layout finalized
Sprint 3	23/03/2025	Database alignment, new page development	- Database fields aligned with UI - Started manage listings layout - Modified Figma for design clarity
Sprint 3	24/03/2025	Directory management, admin functionalities research	- Created directories - Merged branch with master - Admin functionality research started (upload, search) - Interactive elements added to List-a-car
Sprint 4	26/03/2025	Backend merge and page integrations	- Backend implemented into List-a-car page - Linked list integrated into hashtable - Continued receipt booking layout - Began managing transactions layout
Sprint 4	28/03/2025	Admin tools and backend progress	- Research on admin search and UI - Started Search-a-user page - Backend added to login/signup for password handling - Designed receipts page and popup UI - Slider added to transactions page
Sprint 4	29/03/2025	Final backend integrations, UI interactivity	- Backend added to manage listings page - Continued enhancements to transaction management UI
Sprint 4	31/03/2025	Final review and project preparation	- Reviewed project as a whole - Final adjustments to admin functionalities - Updated documentation

6.3. Overall Timeline (Milestones)

Milestone	Estimated Completion Date
Requirements Gathering & Planning	21 Feb 2025
System Architecture Design	22 Feb–04 Mar 2025
Initial Development (Sprint 1–2)	06–16 Mar 2025

Sprint 3 (Development Continuation)	21 – 23 Mar 2025
Sprint 4 (Final Development and Testing)	26 – 31 Mar 2025
Final Implementation	01 – 14 Apr 2025
Documentation (Report Writing, Project Management Plan, README)	12 -15 Apr 2025
Video Demonstration & Submission	16 Apr 2025

6.4. Gantt Chart

This Gantt chart visually organizes the project's timeline, showcasing tasks, its durations, dependencies, and milestones for efficient planning and progress tracking.

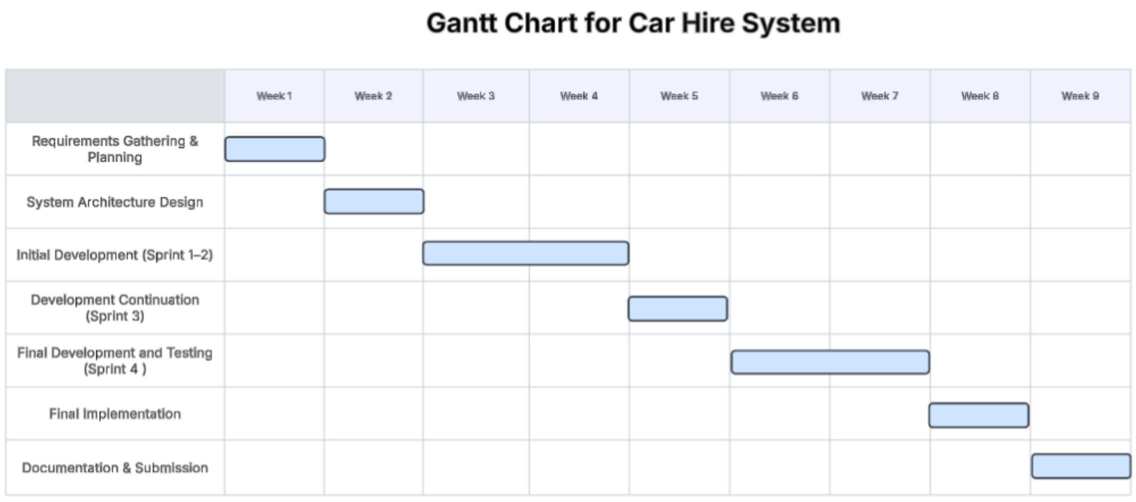


Figure 4: Gantt Chart for Car Hire

6.4. Work Breakdown Structure (WBS) for Car Hire

This Work Breakdown Structure (WBS) simplifies the complexity of this project into smaller, manageable tasks to improve planning, organization, and execution.

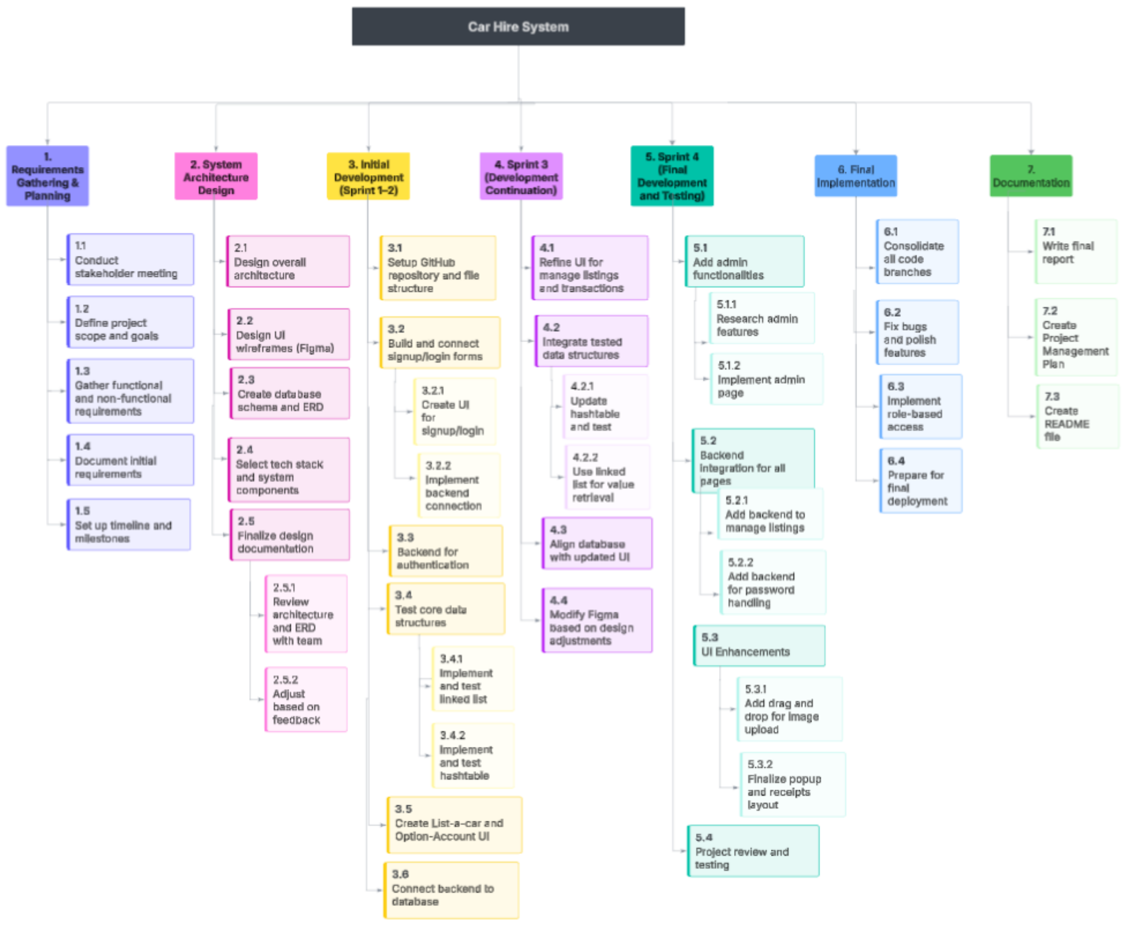

Car Hire System
1. Requirements Gathering & Planning
1.1 Conduct stakeholder meeting
1.2 Define project scope and goals
1.3 Gather functional and non-functional requirements
1.4 Document initial requirements
1.5 Set up timeline and milestones
2. System Architecture Design
2.1 Design overall architecture
2.2 Design UI wireframes (Figma)
2.3 Create database schema and ERD
2.4 Select tech stack and system components
2.5 Finalize design documentation
2.5.1 Review architecture and ERD with team
2.5.2 Adjust based on feedback
3. Initial Development (Sprint 1–2)
3.1 Setup GitHub repository and file structure
3.2 Build and connect signup/login forms
3.2.1 Create UI for signup/login
3.2.2 Implement backend connection
3.3 Backend for authentication
3.4 Test core data structures
3.4.1 Implement and test linked list
3.4.2 Implement and test hashtable
3.5 Create List-a-car and Option-Account UI
3.6 Connect backend to database
4. Sprint 3 (Development Continuation)
4.1 Refine UI for manage listings and transactions
4.2 Integrate tested data structures
4.2.1 Update hashtable and test
4.2.2 Use linked list for value retrieval
4.3 Align database with updated UI
4.4 Modify Figma based on design adjustments
5. Sprint 4 (Final Development and Testing)
5.1 Add admin functionalities
5.1.1 Research admin features
5.1.2 Implement admin page
5.2 Backend Integration for all pages
5.2.1 Add backend to manage listings
5.2.2 Add backend for password handling
5.3 UI Enhancements
5.3.1 Add drag and drop for image upload
5.3.2 Finalize popup and receipts layout
5.4 Project review and testing
6. Final Implementation
6.1 Consolidate all code branches
6.2 Fix bugs and polish features
6.3 Implement role-based access
6.4 Prepare for final deployment
7. Documentation
7.1 Write final report
7.2 Create Project Management Plan
7.3 Create README file

Figure 5: Work Breakdown Structure (WBS) for Car Hire

7. Professional Standards

7.1. Code Quality and Architecture:

Followed SOLID Principles: The project adhered to the **SOLID** principles to ensure maintainable and scalable code. These principles facilitated better modularity and reusability, which are essential for robust software design (Martin, 2003).

Naming Conventions and Modular Architecture: Consistent and meaningful naming conventions were applied for classes, methods, and variables, improving code readability and maintainability (Fowler, 1999). The architecture was designed to be modular, allowing components to function independently and enabling easier debugging and future updates.

7.2. Security Features:

- **Secure Data Storage:** Sensitive user information, such as passwords, was encrypted using secure hashing algorithms to prevent unauthorized access (Katz and Lindell, 2014).

- **Access Control:** Role-based access control mechanisms were implemented to ensure users could only access functionalities relevant to their roles, such as Admin, Customer, or Seller (Sandhu et al., 1996).

8. Monitoring, Reporting, and Controlling Mechanisms

8.1. How Tasks and Deadlines Were Tracked:

- **Meeting Minutes:** Meeting minutes were documented by the secretary during each meeting to ensure all discussions, decisions, and action points were recorded accurately. Progress updates were shared by team members in every meeting, helping to track the completion of tasks against deadlines.
- **Task Organization:** Tasks were manually tracked using shared documents, outlining deadlines and responsibilities for each team member.

8.2. Meeting Schedules and Feedback Cycles:

- **Google Meet:** Team meetings were conducted regularly via Google Meet, with the aim of holding daily sessions. Each meeting typically lasted around 30 to 45 minutes, ensuring consistent communication and effective collaboration.

- **Feedback Sessions:** Weekly feedback was sought from the tutor. During these sessions, the team checked the tutor's point of view and followed their guidelines to align the project with expectations and ensure progress was on track.

8.3. Issue and Change Management:

Summary of Obstacles Encountered: Throughout the project, the team faced the following challenges:

- Learning and Applying Git Commands.
- Establishing links between database tables and defining correct relationships.
- Reducing redundancy within database tables.
- Identifying the most suitable data structure for specific functionalities.
- Programming pages to be resizable and dynamic for better user experience.
- Resolving merge conflicts.
- Respecting referential integrity for the database

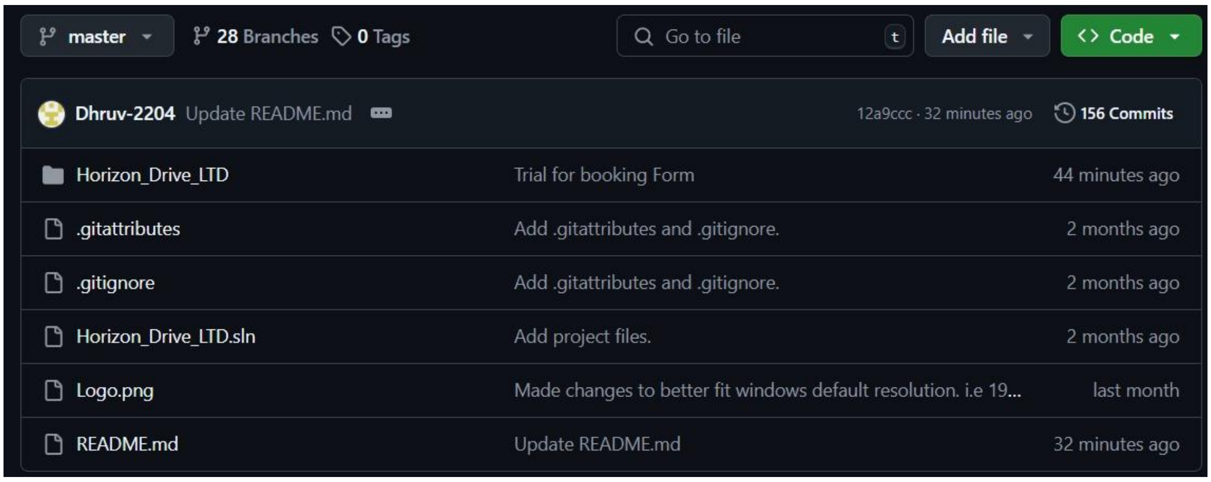
Change Management: On 2 April, the project deadline was rescheduled from 20 April to 16 April, presenting a significant challenge to the team. In response, tasks were re-prioritized, and workflows were refined to accommodate the new timeline. All changes and issues were addressed collaboratively during team meetings to ensure transparency and maintain project alignment.

9. Configuration Management

9.1. Version Control Strategy:

Branching: A feature branching strategy was adopted to separate development work, with individual branches for each feature or task. This ensured smoother integration and minimized conflicts.

Commits: As evidenced in the image below, a total of 156 commits were recorded by 15 April 2025. The commit messages demonstrated a descriptive approach, serving as effective documentation of developmental changes and facilitating progress tracking throughout the project's lifecycle.



Dhruv-2204 Update README.md 12a9ccc · 32 minutes ago 156 Commits

Horizon_Drive_LTD	Trial for booking Form	44 minutes ago
.gitattributes	Add .gitattributes and .gitignore.	2 months ago
.gitignore	Add .gitattributes and .gitignore.	2 months ago
Horizon_Drive_LTD.sln	Add project files.	2 months ago
Logo.png	Made changes to better fit windows default resolution. i.e 19...	last month
README.md	Update README.md	32 minutes ago

Figure 6: Evidence of commits on GitHub

9.2. Backup Process and Documentation Management:

Backups: The project repository was hosted on GitHub, providing automated backups and ensuring data redundancy to mitigate accidental data loss.

Documentation: Essential project documentation included wireframes, ERD diagrams, meeting minutes, and requirements. These documents were collaboratively managed in shared folders with weekly versioning to maintain traceability and accessibility.

9.3. Collaboration on GitHub:

GitHub was the primary platform for team collaboration.

Pull Requests: Changes were reviewed and discussed collaboratively through pull requests to ensure code quality before merging into the main branch.

Issue Tracking: Logged issues were discussed during team meetings to coordinate resolutions and ensure progress.

Real-time Updates: The repository facilitated real-time updates, allowing the team to collaborate efficiently and stay informed of ongoing tasks.

Reference List:

- Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J. and Thomas, D. (2001) *Manifesto for Agile Software Development*. Available at: https://agilemanifesto.org/ (Accessed: 11 April 2025).
- Schwaber, K. and Beedle, M. (2002) *Agile Software Development with Scrum*. Upper Saddle River, NJ: Prentice Hall. Available at: https://www.scrumguides.org/ (Accessed: 11 April 2025).
- Cockburn, A. (2002) *Agile Software Development*. Boston: Addison-Wesley. Available at: https://alistair.cockburn.us/Books/Agile-Software-Development (Accessed: 11 April 2025).
- Highsmith, J. (2009) *Agile Project Management: Creating Innovative Products*. 2nd edn. Boston: Addison-Wesley. Available at: https://www.informit.com/store/agile-project-management-creating-innovative-products-9780321658395 (Accessed: 11 April 2025).
- Shore, J. and Warden, S. (2007) *The Art of Agile Development*. Sebastopol, CA: O'Reilly Media. Available at: https://www.oreilly.com/library/view/the-art-of/9780596527679/ (Accessed: 11 April 2025).
- Cohn, M. (2004) *User Stories Applied: For Agile Software Development*. Boston: Addison-Wesley. Available at: https://www.mountaingoatsoftware.com/books/user-stories-applied (Accessed: 11 April 2025).

- Fowler, M. (1999). **Refactoring: Improving the Design of Existing Code**. Addison-Wesley. Available at: https://www.oreilly.com/library/view/refactoring-improving-the/9780134757681/ (Accessed: 12 April 2025).
- Katz, J. and Lindell, Y. (2014). **Introduction to Modern Cryptography**. 2nd ed. CRC Press. Available at: https://www.taylorfrancis.com/books/mono/10.1201/9781351133036/introduction-modern-cryptography-yehuda-lindell-jonathan-katz (Accessed: 12 April 2025).
- Martin, R.C. (2003). **Agile Software Development, Principles, Patterns, and Practices**. Prentice Hall. Available at: https://books.google.com/books/about/Agile_Software_Development.html?id=0HYhAQAAIAAJ (Accessed: 12 April 2025).
- Sandhu, R.S., Coyne, E.J., Feinstein, H.L. and Youman, C.E. (1996). **Role-Based Access Control Models**. IEEE Computer, 29(2), pp.38–47. Available at: https://csrc.nist.gov/CSRC/media/Projects/Role-Based-Access-Control/documents/sandhu96.pdf (Accessed: 12 April 2025).
- von Ahn, L., Blum, M. and Langford, J. (2003). **CAPTCHA: Using Hard AI Problems for Security**. Available at: https://doi.org/10.1145/772862.772865 (Accessed: 12 April 2025).